Abhängigkeiten und Normalformen

Zur Erinnerung — Benannte Perspektive

In Kapitel 1 hatten wir festgelegt:

- Eine abzählbar unendliche Mengen **att** von Attribut-Namen.
  Diese Menge ist geordnet via $\leqslant_{\mathbf{att}}$.
- Eine abzählbar unendliche Menge **relname** von Relations-Namen.
  Die Mengen **att**, **dom**, **relname** seien disjunkt.
- Eine Funktion $sort : \mathbf{relname} \to \mathcal{P}^{\text{fin}}(\mathbf{att})$, die jedem Relations-Namen eine endliche Menge von Attributen zuordnet ... und zwar so, dass f.a. $U \subseteq^{\text{fin}} \mathbf{att}$ gilt: es gibt unendlich viele $R \in \mathbf{relname}$ mit $sort(R) = U$.
- Ein Relationsschema ist einfach ein Relations-Name R .
- Manchmal schreiben kurz $R[U]$ für $sort(R) = U$.
- Ein R-Tupel ist eine Funktion $t : sort(R) \to \mathbf{dom}$.
- Eine R-Relation ist eine endliche Menge von R-Tupeln.
- $inst(R)$ bezeichnet die Menge aller Relationen über R.
- $inst(U)$ bezeichnet die Menge aller Relationen über einem Relationsschema der Sorte U (für $U \subseteq^{\text{fin}} \mathbf{att}$)

Motivation (1/2)

Ziel beim Datenbank-Entwurf:

Ein DB-Schema entwickeln, so dass Informationen zum gewünschten Anwendungsbereich "sinnvoll" gespeichert werden können.
Insbesondere: Wenn möglich, Redundanzen und Inkonsistenzen vermeiden.

Beispiel: Relation *Warenlager*[Bauteil-Nr, Lager-Nr, Menge, Ort]

Warenlager:

Bauteil-Nr	Lager-Nr	Menge	Ort
2411	2	200	Riedberg
2412	3	300	Bornheim
3001	1	100	Hanau
2415	2	100	Riedberg

Unschön: Redundanz — der Ort von Lager 2 ist mehrfach gespeichert.

Dadurch können Inkonsistenzen auftreten:

Update-Anomalien = Inkonsistenzen, die durch Aktualisierung der DB auftreten können:

- Änderungs-Anomalie: den Ort in Zeile 1 durch "Westend" ersetzen
  $\rightsquigarrow$ Adresse von Lager 2 nicht mehr eindeutig
- Lösch-Anomalie: Löschen von Zeile 2
  $\rightsquigarrow$ Information über die Adresse von Lager 3 geht verloren
- Einfüge-Anomalie: Die Adresse eines neuen Lagers kann erst dann eingefügt werden, wenn mindestens ein Bauteil dort gelagert wird.

Motivation (2/2)

Zur Vermeidung dieser Update-Anomalien:

Informationen auf 2 Relationen *Adressen*[Lager-Nr,Ort] und *Lagerung*[Bauteil-Nr,Lager-Nr,Menge] aufteilen

Adressen:

Lager-Nr	Ort
2	Riedberg
3	Bornheim
1	Hanau

Abhängigkeit:
Lager-Nr $\to$ Ort

Lagerung:

Bauteil-Nr	Lager-Nr	Menge
2411	2	200
2412	3	300
3001	1	100
2415	2	100

Abhängigkeit:
Bauteil-Nr, Lager-Nr $\to$ Menge

Zur Anfrage-Optimierung:

Die "optimierte Anfrage" muss nur auf solchen Datenbanken äquivalent zur Original-Anfrage sein, die die obigen Abhängigkeiten erfüllen.

$\rightsquigarrow$ die minimale, solchermaßen äquivalente Anfrage ist evtl. noch kleiner als die, die durch die Tableau-Minimierung aus Kapitel 5 gefunden wird.

Abhängigkeiten und Normalformen

Notation

Attribut-Namen :	$A, B, C, A_1, A_2, \ldots$
Attribut-Mengen :	$X, Y, Z, X_1, X_2, \ldots, U$
Relationen :	I, J
$\{A, B, C\}$:	ABC
$X \cup Y$:	XY

Funktionale Abhängigkeiten

Definition 6.1

Sei U eine endliche Menge von Attribut-Namen.

(a) Eine funktionale Abhängigkeit (kurz: FD) über U ist ein Ausdruck der Form $X \to Y$, wobei $X, Y \subseteq U$. ("FD" steht für "functional dependency")

(b) Eine Relation $I \in inst(U)$ erfüllt die FD $X \to Y$ (kurz: $I \models X \to Y$), falls für alle Tupel t und s aus I gilt:

$$\pi_X(t) = \pi_X(s) \quad \Longrightarrow \quad \pi_Y(t) = \pi_Y(s)$$

(D.h.: Wenn t und s in sämtlichen Spalten aus X übereinstimmen, dann stimmen sie auch in jeder Spalte aus Y überein.)

(c) Ist Σ eine Menge von FDs über U, so gilt

$$I \models \Sigma \quad :\Longleftrightarrow \quad \text{für alle } f \in \Sigma \text{ gilt } I \models f$$

(d) Eine Schlüsselbedingung ist eine FD der Form $X \to U$.

Funktionale Abhängigkeiten und verlustfreie Joins

Proposition 6.2

Sei $X \to Y$ eine FD über U und sei $Z := U \setminus (X \cup Y)$.
Für jede Relation $I \in inst(U)$ gilt:
Falls $I \models X \to Y$, so ist $I = \pi_{XY}(I) \bowtie \pi_{XZ}(I)$.

Beweis: Übung.

Folgerung: Die in Relation I gespeicherte Information kann "verlustfrei" auf zwei Relationen aufgeteilt werden (eine mit den Spalten XY und eine mit den Spalten XZ), aus denen die Original-Relation rekonstruiert werden kann. (Stichwort: "lossless join")

Beispiel für einen "verlustreichen Join": Siehe Tafel.

Abhängigkeiten und Normalformen

Beispiel zu "The Chase — Die Verfolgungsjagd"

Beispiel 6.3

Tableau-Anfrage $Q = (T, t)$ mit

$$T = \begin{array}{|cccc|} \hline A & B & C & D \\ \hline w & x & y & z' \\ w' & x & y' & z \\ \hline \end{array} \qquad \text{und} \qquad t = (w, x, y, z)$$

Klar: Die Anfrage $Q := (T, t)$ ist minimal im Sinne von Kapitel 5.1.

Situation jetzt:

- Gegeben sei die FD-Menge $\Sigma := \{ B \to D \}$
- Q soll nur auf solchen DBs ausgewertet werden, die Σ erfüllen
- Ziel: Vereinfache (minimiere) Q.

Details: siehe Tafel.

Äquivalenz bzw. Query Containment bzgl. Σ

Definition 6.4

Sei R ein Relationsschema, sei $\mathcal{F} \subseteq inst(R)$, und seien Q_1 und Q_2 zwei Anfragen an Relationen aus $inst(R)$.

- $Q_1 \subseteq_{\mathcal{F}} Q_2 \;:\Longleftrightarrow\; [\![Q_1]\!](I) \subseteq [\![Q_2]\!](I)$, für alle $I \in \mathcal{F}$.
- $Q_1 \equiv_{\mathcal{F}} Q_2 \;:\Longleftrightarrow\; [\![Q_1]\!](I) = [\![Q_2]\!](I)$, für alle $I \in \mathcal{F}$.

Sei Σ eine Menge von FDs über R.

- $\text{sat}(R, \Sigma) := \text{sat}(\Sigma) := \{I \in inst(R) : I \models \Sigma\}$
- $Q_1 \subseteq_\Sigma Q_2 \;:\Longleftrightarrow\; Q_1 \subseteq_{\text{sat}(\Sigma)} Q_2$
- $Q_1 \equiv_\Sigma Q_2 \;:\Longleftrightarrow\; Q_1 \equiv_{\text{sat}(\Sigma)} Q_2$

Vereinbarungen für den Rest von Kapitel 6.2

Der Einfachheit halber betrachten wir im Folgenden

- ein festes Relationsschema R
- Mengen Σ von FDs über R, in denen oBdA jede FD von der Form $X \to A$ mit $X \subseteq sort(R)$ und $A \in sort(R)$ ist
- eine feste lineare Ordnung $<$ auf der Variablenmenge **var**
- nur Tableau-Anfragen $Q = (T, t)$ über R, in denen keine Konstanten vorkommen

Bemerkung: Die Ergebnisse aus Kapitel 6.2 können leicht verallgemeinert werden auf Anfragen mit Konstanten und auf Anfragen über einem Datenbankschema.

Regel für die Verfolgungsjagd

Definition 6.5

FD-Regel:

Sei $f := (X \to A)$ eine FD über R, sei (T, t) eine Tableau-Anfrage über R.
Seien u und v Zeilen von T mit $\pi_X(u) = \pi_X(v)$ und $u(A) \neq v(A)$.
Sei $\{x, y\} := \{u(A), v(A)\} \subseteq$ **var** und sei $x < y$.

Anwenden der FD f auf u, v in (T, t) liefert die Tableau-Anfrage $(\zeta(T), \zeta(t))$, wobei ζ die Substitution mit $\zeta(y) := x$ und $\zeta(z) := z$ für alle $z \in Var((T, t)) \setminus \{y\}$.

Anwenden der FD-Regel erhält Äquivalenz bzgl. Σ

Proposition 6.6

Sei Σ eine Menge von FDs über R,
sei $f := (X \to A) \in \Sigma$,
sei $Q := (T, t)$ eine Tableau-Anfrage über R und
sei $Q' := (T', t')$ eine Tableau-Anfrage, die durch 1-maliges Anwenden der FD-Regel mit einer FD $f \in \Sigma$ aus Q entsteht.

Dann gilt: $Q \equiv_\Sigma Q'$.

Beweis: Siehe Tafel.

Verfolgungssequenzen

Definition 6.7

(a) Eine Verfolgungssequenz für (T, t) mittels Σ ist eine Folge

$$(T_0, t_0),\ (T_1, t_1),\ (T_2, t_2),\ \ldots$$

für die gilt:

- $(T_0, t_0) = (T, t)$ und
- für jedes $i \geqslant 0$ entsteht (T_{i+1}, t_{i+1}) durch 1-maliges Anwenden der FD-Regel mit einer FD aus Σ auf (T_i, t_i).

(b) Die Verfolgungssequenz ist terminiert, falls sie endlich ist und auf ihr letztes Element, (T_m, t_m), keine FD-Regel mit einer FD aus Σ mehr angewendet werden kann.
(T_m, t_m) heißt dann das Resultat der Sequenz.

Notation: $T \models \Sigma$

Definition 6.8

(a) Ein Tableau T über R erfüllt die FD $X \to A$ (kurz: $T \models X \to A$), falls für alle Zeilen u und v von T gilt:

$$\pi_X(u) = \pi_X(v) \quad \Longrightarrow \quad u(A) = v(A)$$

(D.h.: Wenn u und v in sämtlichen Spalten aus X übereinstimmen, dann stimmen sie auch in der Spalte A überein.)

(b) Ist Σ eine Menge von FDs über R, so gilt

$$T \models \Sigma \quad :\Longleftrightarrow \quad T \models f, \text{ für alle } f \in \Sigma$$

Eigenschaften des Resultats einer Verfolgungssequenz

Lemma 6.9

Sei (T', t') das Resultat einer terminierten Verfolgungssequenz für (T, t) mittels Σ. Dann gilt: $(T', t') \equiv_\Sigma (T, t)$ und $T' \models \Sigma$.

Beweis: Übung.

Klar: Jede Verfolgungssequenz ist endlich und kann zu einer terminierten Sequenz vervollständigt werden.

Bemerkenswert: Man kann zeigen, dass alle terminierten Verfolgungssequenzen für (T, t) mittels Σ dasselbe Resultat liefern. Dies wird auch Church-Rosser-Eigenschaft genannt.

Church-Rosser-Eigenschaft der Verfolgungsjagd

Theorem 6.10

Sei (T, t) eine Tableau-Anfrage über R und sei Σ eine Menge von FDs über R. Dann gilt: Alle terminierten Verfolgungssequenzen für (T, t) mittels Σ liefern dasselbe Resultat.

Hier ohne Beweis.

Ein Beweis findet sich am Ende von Kapitel 8.4 des Buchs [AHV].

Definition 6.11

Ist (T, t) eine Tableau-Anfrage über R und Σ eine Menge von FDs über R, so bezeichnet $chase(T, t, \Sigma)$ das Resultat einer (bzw. sämtlicher) terminierter Verfolgungssequenzen für (T, t) mittels Σ.

Bemerkung: Von Lemma 6.9 wissen wir, dass $chase(T, t, \Sigma) \equiv_\Sigma (T, t)$ und $chase(T, t, \Sigma) \models \Sigma$.

Berechnung von $chase(T, t, \Sigma)$

Korollar 6.12

Es gibt einen Polynomialzeit-Algorithmus, der bei Eingabe einer Tableau-Anfrage (T, t) über R und einer Menge Σ von FDs über R die Tableau-Anfrage $chase(T, t, \Sigma)$ berechnet.

Beweis:

Algorithmus:

(1) Wiederhole so lange, bis keine FD-Regel bzgl. Σ mehr auf (T, t) anwendbar ist:

(1.1) (T', t') sei das Resultat der Anwendung einer FD-Regel bzgl. Σ auf (T, t).

(1.2) Setze $(T, t) := (T', t')$.

(2) Gib (T, t) aus.

Korrektheit folgt direkt aus der Church-Rosser-Eigenschaft.

Polynomielle Laufzeit, da jede FD $f \in \Sigma$ auf jedes Paar u, v von Zeilen von T höchstens 1-mal angewendet werden kann und da jede einzelne Anwendung der FD-Regel nur polynomiell viel Zeit benötigt.

Äquivalenz von Anfragen bzgl. Σ

Theorem 6.13

Seien $Q_1 := (T_1, t_1)$ und $Q_2 := (T_2, t_2)$ Tableau-Anfragen über R und sei Σ eine Menge von FDs über R. Dann gilt:

(a) $Q_1 \subseteq_\Sigma Q_2 \iff chase(T_1, t_1, \Sigma) \subseteq chase(T_2, t_2, \Sigma)$ *und*

(b) $Q_1 \equiv_\Sigma Q_2 \iff chase(T_1, t_1, \Sigma) \equiv chase(T_2, t_2, \Sigma)$ *und*

Beweis: Siehe Tafel.

Bemerkung: Aus Theorem 6.13, Korollar 6.12 und Korollar 5.6 folgt insbesondere, dass "$Q_1 \subseteq_\Sigma Q_2$" bzw. "$Q_1 \equiv_\Sigma Q_2$" entscheidbar ist und zur Komplexitätsklasse NP gehört.

Anfrage-Minimierung bzgl. Σ

Vorgehensweise:

- Eingabe: Tableau-Anfrage $Q = (T, t)$ über R und Menge Σ von FDs über R
- Schritt 1: Berechne $Q' := (T', t') := chase(T, t, \Sigma)$. Klar: $Q' \equiv_\Sigma Q$
- Schritt 2: Nutze den Algorithmus aus Korollar 5.9(a), um eine minimale Tableau-Anfrage $Q'' := (T'', t'')$ mit $Q'' \equiv Q'$ zu berechnen
  Insbesondere gilt: $Q'' \equiv_\Sigma Q' \equiv_\Sigma Q$

Notation: Ist $Q = (T, t)$ eine Tableau-Anfrage, so schreibe $min(Q) := min(T, t)$, um die gemäß Korollar 5.9(a), minimale zu Q äquivalente Tableau-Anfrage zu bezeichnen.

Lemma 6.14

Sei $Q = (T, t)$ eine Tableau-Anfrage über R und sei Σ eine Menge von FDs über R. Dann gilt: $|min(chase(T, t, \Sigma))| \leqslant |min(T, t)|$.

Hier ohne Beweis.

Eine Beweisskizze findet sich auf Seite 178 des Buchs [AHV].

Abhängigkeiten und Normalformen

Ziel der Normalisierung

Gegeben:

- eine endliche Menge U von Attribut-Namen
- eine Menge Σ von Abhängigkeiten zwischen den Attributen aus U

Ziel:

Finde ein Datenbankschema mit den Attributen aus U, das

- möglichst Redundanz-frei ist ⇝ spart Speicherplatz
  ⇝ verhindert Änderungs-Anomalien
- alle gewünschten Daten speichern kann
  "lossless", d.h. "informationsverlustfrei"
- die Abhängigkeiten aus Σ "respektiert"
  "dependency preserving", d.h. "abhängigkeitstreu"

Motivation (1/2)

Ziel beim Datenbank-Entwurf:

Ein DB-Schema entwickeln, so dass Informationen zum gewünschten Anwendungsbereich "sinnvoll" gespeichert werden können.
Insbesondere: Wenn möglich, Redundanzen und Inkonsistenzen vermeiden.

Beispiel: Relation *Warenlager* [Bauteil-Nr, Lager-Nr, Menge, Ort]

Warenlager:

Bauteil-Nr	Lager-Nr	Menge	Ort
2411	2	200	Riedberg
2412	3	300	Bornheim
3001	1	100	Hanau
2415	2	100	Riedberg

Unschön: Redundanz — der Ort von Lager 2 ist mehrfach gespeichert.

Dadurch können Inkonsistenzen auftreten:

Update-Anomalien = Inkonsistenzen, die durch Aktualisierung der DB auftreten können:

- Änderungs-Anomalie: den Ort in Zeile 1 durch "Westend" ersetzen
  ⇝ Adresse von Lager 2 nicht mehr eindeutig
- Lösch-Anomalie: Löschen von Zeile 2
  ⇝ Information über die Adresse von Lager 3 geht verloren
- Einfüge-Anomalie: Die Adresse eines neuen Lagers kann erst dann eingefügt werden, wenn mindestens ein Bauteil dort gelagert wird.

Motivation (2/2)

Zur Vermeidung dieser Update-Anomalien:

Informationen auf 2 Relationen *Adressen*[Lager-Nr,Ort] und *Lagerung*[Bauteil-Nr,Lager-Nr,Menge] aufteilen

Adressen:

Lager-Nr	Ort
2	Riedberg
3	Bornheim
1	Hanau

Abhängigkeit:
Lager-Nr → Ort

Lagerung:

Bauteil-Nr	Lager-Nr	Menge
2411	2	200
2412	3	300
3001	1	100
2415	2	100

Abhängigkeit:
Bauteil-Nr, Lager-Nr → Menge

Zur Anfrage-Optimierung:

Die "optimierte Anfrage" muss nur auf solchen Datenbanken äquivalent zur Original-Anfrage sein, die die obigen Abhängigkeiten erfüllen.

$\rightsquigarrow$ die minimale, solchermaßen äquivalente Anfrage ist evtl. noch kleiner als die, die durch die Tableau-Minimierung aus Kapitel 5 gefunden wird.

Notation

- Relationsschema:

  bisher: Relations-Name R (dem eine Attributmenge U zugeordnet ist); Schreibweise: $R[U]$

  jetzt: $(R[U], \Sigma)$, wobei R ein Relations-Name der Sorte U und Σ eine Menge von FDs über U.

- DB-Schema:

  bisher: Menge **R** von Relations-Namen

  jetzt: Menge $\{ (R_1[U_1], \Sigma_1), \ldots, (R_n[U_n], \Sigma_n) \}$ von Relationsschemata

- Implikation bzw. Äquivalenz von FD-Mengen:

  Σ und Γ seien zwei Mengen von FDs über U, f sei eine FD über U.

  - $\Sigma \models_U f \quad :\Longleftrightarrow$ für alle $I \in inst(U)$ gilt: falls $I \models \Sigma$, so $I \models f$.
  - $\Sigma \models_U \Gamma \quad :\Longleftrightarrow$ für alle $I \in inst(U)$ gilt: falls $I \models \Sigma$, so $I \models \Gamma$.
  - $\Sigma \equiv_U \Gamma \quad :\Longleftrightarrow \quad \Sigma \models_U \Gamma$ und $\Gamma \models_U \Sigma$.

Test auf Implikation

Satz 6.15

Es gibt einen Algorithmus, der bei Eingabe einer endlichen Attributmenge U, einer Menge Σ von FDs über U und einer FD $X \to Y$ über U entscheidet, ob $\Sigma \models_U X \to Y$.

Beweis: *siehe Tafel.*

Zerlegungen

Definition 6.16

(a) Eine Zerlegung (engl: decomposition) eines Relationsschemas $(R[U], \Sigma)$ ist ein DB-Schema $\{ (R_1[U_1], \Sigma_1), \ldots, (R_n[U_n], \Sigma_n) \}$, so dass

$$U_1 \cup \cdots \cup U_n = U \quad \text{und} \quad \Sigma \models_U \Sigma_1 \cup \cdots \cup \Sigma_n.$$

(b) Eine Zerlegung heißt abhängigkeitstreu (engl: dependency preserving), falls $\Sigma \equiv_U \Sigma_1 \cup \cdots \cup \Sigma_n$.

(c) Eine Zerlegung heißt informationsverlustfrei (engl: lossless), falls für alle Relationen $I \in inst(R)$ mit $I \models \Sigma$ gilt: $I = \pi_{U_1}(I) \bowtie \cdots \bowtie \pi_{U_n}(I)$.

Zur Erinnerung: $I \subseteq \pi_{U_1}(I) \bowtie \cdots \bowtie \pi_{U_n}(I)$ gilt sowieso.
Um Informationsverlustfreiheit nachzuweisen, genügt es also, nachzuprüfen, ob für alle $I \in inst(R)$ gilt: $\pi_{U_1}(I) \bowtie \cdots \bowtie \pi_{U_n}(I) \subseteq I$.

Boyce-Codd Normalform (BCNF)

Intuition: "Do not represent the same fact twice" ... Elimination von Redundanzen

Definition 6.17

(a) Ein Relationsschema $(R[U], \Sigma)$ ist in Boyce-Codd Normalform (BCNF), falls für alle FDs $X \to A$ mit $X \subseteq U$ und $A \in U \setminus X$ gilt:

$$\text{Falls } \Sigma \models X \to A, \text{ so } \Sigma \models X \to U$$ d.h. X ist ein "Superschlüssel"

(b) Ein DB-Schema $\{ (R_1[U_1], \Sigma_1), \ldots, (R_n[U_n], \Sigma_n) \}$ ist in BCNF, falls jedes seiner Relationsschemata (R_i, Σ_i) in BCNF ist.

Beispiel

- $R[U]$ = *Warenlager*[Bauteil-Nr, Lager-Nr, Menge, Ort]
- Σ = { Lager-Nr $\to$ Ort, Bauteil-Nr, Lager-Nr $\to$ Menge }
- $(R[U], \Sigma)$ ist nicht in BCNF,
  denn $\Sigma \not\models$ Lager-Nr $\to$ Bauteil-Nr, Lager-Nr, Menge, Ort

- $R_1[U_1]$ = *Adressen*[Lager-Nr,Ort] und
  Σ_1 = { Lager-Nr $\to$ Ort }
- $R_2[U_2]$ = *Lagerung*[Bauteil-Nr,Lager-Nr,Menge] und
  Σ_2 = { Bauteil-Nr, Lager-Nr $\to$ Menge }
- das DB-Schema $\{ (R_1[U_1], \Sigma_1), (R_2[U_2], \Sigma_2) \}$ ist in BCNF

Notation: Für eine Menge Γ von FDs und eine Attributmenge $V \subseteq U$ setzen wir

$$\pi_V(\Gamma) := \{ X \to A : XA \subseteq V \text{ und } \Gamma \models X \to A \}.$$

Beachte: $\pi_V(\Gamma)$ kann mit Hilfe des Algorithmus aus Satz 6.15 berechnet werden.

Algorithmus zur BCNF-Dekomposition

Eingabe: Relationsschema $(R[U], \Sigma)$

Ausgabe: DB-Schema $D = \{ (R_1, \Sigma_1), \ldots, (R_n, \Sigma_n) \}$ in BCNF

(1) $D := \{ (R[U], \Sigma) \}$

(2) Solange D nicht in BCNF ist, wiederhole:

(2.1) Wähle ein Rel.schema $(S[V], \Gamma) \in D$, das nicht in BCNF ist
(2.2) Wähle Mengen $\emptyset \neq X, Y, Z \subseteq V$ mit $V = X \dot\cup Y \dot\cup Z$ so dass
$\Gamma \models X \to Y$ und, für alle $A \in Z$, $\Gamma \not\models X \to A$.
(2.3) Ersetze $(S[V], \Gamma)$ in D durch $(S_1[XY], \pi_{XY}(\Gamma))$ und $(S_2[XZ], \pi_{XZ}(\Gamma))$
(2.4) Für alle $(S'[V'], \Gamma') \in D$ und $(S''[V''], \Gamma'') \in D$ mit $V' \subseteq V''$,
entferne $(S'[V'], \Gamma')$ aus D.

Satz 6.18

Der obige Algorithmus zerlegt ein Relationsschema informationsverlustfrei in ein Datenbankschema, das in BCNF ist.

Beweis:
Der Algorithmus terminiert mit D in BCNF: klar.
D ist eine Zerlegung von (R, Σ) : klar.
Informationsverlustfreiheit der Zerlegung folgt direkt aus Proposition 6.2.

Bemerkung: Exponentielle Laufzeit wegen Berechnung von $\pi_V(\Gamma)$ und Test auf BCNF-Eigenschaft.

BCNF vs. Abhängigkeitstreue

Beispiel 6.19

- $R[U]$ = *Adresse* [Straße, Ort, PLZ]
  Σ = { PLZ $\to$ Ort, Ort, Straße $\to$ PLZ }
- Algorithmus zur BCNF-Dekomposition liefert
  D = { (S[PLZ,Ort], {PLZ $\to$ Ort}), (T[PLZ,Straße], $\emptyset$) }
- Laut Satz 6.18 ist D eine informationsverlustfreie Zerlegung in BCNF.
- D ist nicht abhängigkeitstreu (weil die FD Ort, Straße $\to$ PLZ verloren geht)
- Es gilt sogar: Es gibt keine BCNF-Zerlegung von

  $\Big($ *Adresse* [Straße, Ort, PLZ], { PLZ $\to$ Ort; Ort, Straße $\to$ PLZ } $\Big)$,

  die abhängigkeitstreu ist.

Informationsverlustfreiheit und Abhängigkeitstreue

Frage:

Gibt es an Stelle von BCNF eine andere Variante von “Normalform”, die informationsverlustfrei und abhängigkeitstreu ist?

Antwort:

Ja, die so genannte *dritte Normalform 3NF*

Im Vergleich zu BCNF ist die 3NF weniger restriktiv, eliminiert Redundanzen also weniger gründlich als BCNF.
Dafür sind in 3NF aber informationsverlustfreie und abhängigkeitstreue Zerlegungen möglich.

In der Praxis ist 3NF meist interessanter als BCNF.

Dritte Normalform (3NF)

Sei $(R[U], \Sigma)$ ein Relationsschema.

- Ein Superschlüssel ist eine Menge $X \subseteq U$ mit $\Sigma \models X \to U$.
- Ein Schlüssel ist ein (bzgl $\subseteq$) minimaler Superschlüssel.
- Ein Schlüsselattribut ist ein Attribut $A \in U$, das zu mindestens einem Schlüssel gehört.

Definition 6.20

(a) Ein Relationsschema $(R[U], \Sigma)$ ist in dritter Normalform (3NF), falls für alle FDs $X \to A$ mit $X \subseteq U$ und $A \in U \setminus X$ gilt:

$$\text{Falls } \Sigma \models X \to A, \text{ so } \quad \Sigma \models X \to U \text{ oder } A \text{ ist ein Schlüsselattribut.}$$

(b) Ein DB-Schema $\{ (R_1[U_1], \Sigma_1), \ldots, (R_n[U_n], \Sigma_n) \}$ ist in 3NF, falls jedes seiner Relationsschemata (R_i, Σ_i) in 3NF ist.

Beispiel: Das Relationsschema
(*Adresse* [Straße, Ort, PLZ], $\{$ PLZ $\to$ Ort, Ort, Straße $\to$ PLZ $\}$) ist in 3NF.

Gütekriterien für Datenbank-Entwürfe

Was ist ein **gutes** Datenbankschema?

- eins in BCNF oder 3NF (oder 2NF, 4NF, . . .)

Warum sind BCNF oder 3NF “**gut**”?

- Vermeidung von Update-Anomalien
  (bei BCNF “gründlicher” als bei 3NF)
- Existenz von Algorithmen, die ein gegebenes DB-Schema in ein “gutes” DB-Schema überführen:
  - informationsverlustfrei
  - bei 3NF auch: abhängigkeitstreu

Jetzt: Weitere Rechtfertigung der Güte von BCNF mittels Informationstheorie

Vorteil:

- Dieser Ansatz benutzt ausschließlich Charakteristika der **Daten** (bzw. des Schemas (R, Σ)) und hängt nicht von Begriffen wie “Update-Anomalien” und “Informationsverlustfreiheit” ab.
- Der Ansatz funktioniert auch für andere Datenmodelle als das relationale Modell; insbesondere auch für XML.

  Dort: noch einigermaßen unklar, was genau “Update-Anomalien” bzw. Update-Sprachen und was “Informationsverlustfreiheit” sein soll.

Literatur:

Marcelo Arenas und Leonid Libkin.
An Information-Theoretic Approach to Normal Forms for Relational and XML Data.
Journal of the ACM, Volume 52, No. 2, 2005, pages 246–283.
(Diese Arbeit hat bei der Konferenz PODS 2003 den “Best Paper Award” gewonnen.)

“Crashkurs” in Informationstheorie

Informationstheorie — Motivation (1/2)

- 1948 von Claude Elwood Shannon begründet
- Fragestellung:
  - Was ist Information?
  - Wie kann man den Informationsgehalt verschiedener Informationsquellen miteinander vergleichen?
- Szenario: (Beispiel)
  - eine Informationsquelle erzeugt Texte in deutscher Sprache
  - ein “Sender” liest die Texte und will sie an einen “Empfänger” schicken
  - dazu werden die Texte als Bitstrings repräsentiert
  - Ziel: Bitstring soll möglichst kurz sein
  - Annahme: Originaltext mit Buchstaben aus $\{a, \ldots, z\}$: 26 Stück

Informationstheorie — Motivation (2/2)

1. Möglichkeit: Jeden Buchstaben als Bitstring der Länge $\lceil \lg(26) \rceil = 5$ kodieren.
   Originaltext der Länge n $\rightsquigarrow$ Bitstring der Länge $5n$

Beobachtung: In Texten deutscher Sprache kommen manche Buchstaben häufiger vor als andere. Zum Beispiel:

- häufigster Buchstabe: e mit Wahrscheinlichkeit $P(e) = 0,175$
  (d.h.: jeder 6-te Buchstabe ist ein e)
- zweithäufigster Buchstabe: n mit Wahrscheinlichkeit $P(n) = 0,1$
  (d.h.: jeder 10-te Buchstabe ist ein n)
- seltenster Buchstabe: q mit Wahrscheinlichkeit $P(q) = 0,01$
  (d.h.: nur jeder 100-te Buchstabe ist ein q)

Naheliegende Idee:

Kodiere Buchstaben, die oft vorkommen durch *kurze* Bitstrings und Buchstaben, die selten vorkommen, durch längere Bitstrings.
Bei optimaler Kodierung dann:

Originaltext der Länge n $\rightsquigarrow$ Bitstring der Länge $4n$

D.h.: im Schnitt nur 4 Bits pro Original-Buchstabe
(zum Vergleich: 5 Bits bei herkömmlicher Kodierung)

Informationsgehalt und Entropie — Anschaulich

Begriffe:

- Entropie von Texten deutscher Sprache $\approx$ 4
  $\approx$ “durchschnittliche Länge des Bitstrings pro Buchstaben des Originaltextes”
  (bei optimaler Kodierung)
- Informationsgehalt eines Buchstabens $\alpha \in \{a, \ldots, z\}$
  $\approx$ “Länge des Bitstrings, der den Buchstaben α kodiert”
  (bei optimaler Kodierung)

Intuition: Lesen des Buchstabens “y” gibt mir mehr Information als Lesen des Buchstabens “e”, da es weniger Worte gibt, in denen “y” vorkommt als Worte, in denen “e” vorkommt.

Informationsgehalt und Entropie — Präzise

Definition 6.21

(a) Ein Zufallsraum $\mathcal{E} = (\{e_1, \ldots, e_m\}, P_\mathcal{E})$ besteht aus einer Menge $\{e_1, \ldots, e_m\}$ von Ereignissen, denen Wahrscheinlichkeiten $p_1 = P_\mathcal{E}(e_1), \ \ldots, \ p_m = P_\mathcal{E}(e_m)$ zugeordnet sind, so dass $0 \leqslant p_i \leqslant 1$, für alle $i \in \{1, \ldots, m\}$, und $\sum_{i=1}^{m} p_i = 1$.

(b) Der Informationsgehalt von Ereignis e_i ist (lg ist Logarithmus zur Basis 2)

$$\lg\left(\frac{1}{p_i}\right) \quad \Big(\ = \ -\lg(p_i) \ \Big)$$

(c) Die Entropie von $\mathcal{E}$, kurz: $H(\mathcal{E})$, ist

$$H(\mathcal{E}) \ := \ H(p_1, \ldots, p_m) \ := \ \sum_{i=1}^{m} p_i \cdot \lg\left(\frac{1}{p_i}\right) \quad \Big(\ = \ -\sum_{i=1}^{m} p_i \cdot \lg(p_i) \ \Big)$$

Konvention für $p_i = 0$: $\quad 0 \cdot \lg\left(\frac{1}{0}\right) \ := \ -0 \cdot \lg(0) \ := \ 0$

Bemerkungen zu Entropie und Informationsgehalt

- Man kann beweisen, dass

$$H(\mathcal{E}) \ \leqslant \ \begin{pmatrix}\text{durchschnittliche Länge des Bitstrings, der ein Ereignis aus } \{e_1, \ldots, e_m\} \text{ kodiert (bei optimaler Kodierung)}\end{pmatrix} \ \leqslant \ H(\mathcal{E}) + 1$$

- Wegen $H(\mathcal{E}) = \sum_{i=1}^{m} p_i \cdot \lg\left(\frac{1}{p_i}\right)$ gilt somit:

$$\begin{pmatrix}\text{Informationsgehalt von Ereignis } e_i\end{pmatrix} \ \stackrel{\text{Def}}{=} \ \lg\left(\frac{1}{p_i}\right) \ \approx \ \begin{pmatrix}\text{Länge des Bitstrings, der } e_i \text{ kodiert (bei optimaler Kodierung)}\end{pmatrix}$$

Eigenschaften von Entropie und Informationsgehalt

Eigenschaften des Informationsgehalts:

- Ereignis, das mit Wahrscheinlichkeit 1 eintrifft: Informationsgehalt 0
  (denn: $\lg\left(\frac{1}{1}\right) = \lg(1) = 0$)
- Informationsgehalte zweier voneinander unabhängiger Ereignisse addieren sich:
  $\lg\left(\frac{1}{p_i \cdot p_j}\right) \ = \ \lg\left(\frac{1}{p_i}\right) + \lg\left(\frac{1}{p_j}\right)$

Eigenschaften der Entropie:

- $H(p_1, \ldots, p_m)$ ist maximal, falls $p_1 = \cdots = p_m = \frac{1}{m}$.
  Es gilt: $H(\frac{1}{m}, \ldots, \frac{1}{m}) = \sum_{i=1}^{m} \frac{1}{m} \lg(m) \ = \ \lg(m)$
- $H(p_1, \ldots, p_m)$ ist minimal, falls es ein $i \in \{1, \ldots, m\}$ gibt mit $p_i = 1$ und $p_j = 0$ f.a. $j \neq i$.
  Es gilt: $H(1, 0, \ldots, 0) = 1 \cdot \lg\left(\frac{1}{1}\right) + 0 + \cdots + 0 \ = \ 0$
- Somit gilt stets: $0 \ \leqslant \ H(p_1, \ldots, p_m) \ \leqslant \ \lg(m)$

Lemma 6.22
Für jeden Zufallsraum $\mathcal{E} = (\{e_1, \ldots, e_m\}, P_\mathcal{E})$ *mit* $(p_1, \ldots, p_m) := (P_\mathcal{E}(e_1), \ldots, P_\mathcal{E}(e_m))$
gilt: $H(p_1, \ldots, p_m) = 0 \iff$ *Es gibt ein* $i \in \{1, \ldots, m\}$ *mit* $p_i = 1$ *und* $p_j = 0$ *f.a.* $j \neq i$.

Beweis: Siehe Tafel.

Informationstheorie und Normalformen

Notation

Der Einfachheit halber sei jetzt **dom** $:= \mathbb{N}_{\geqslant 1} = \{1, 2, 3, \ldots\}$

Definition 6.23

Sei R ein Relations-Name der Sorte $sort(R)$.

(a) Für eine Relation $I \in inst(R)$ ist

$$Pos(I) := \{ (R, t, A) : t \in I,\ A \in sort(R) \}$$

die Menge der Positionen von I.

(Klar: $|Pos(I)| = ||I|| \stackrel{\text{Def}}{=} |I| \cdot |sort(R)|$)

(b) Für ein Relationsschema (R, Σ) setzen wir

$$inst(R, \Sigma) := \{ I \in inst(R) : I \models \Sigma \}$$

und für ein Element $k \in$ **dom** $= \mathbb{N}_{\geqslant 1}$ setzen wir

$$inst_k(R, \Sigma) := \{ I \in inst(R, \Sigma) : adom(I) \subseteq \{1, \ldots, k\} \}$$

Entropie zum Messen von Redundanz — Bsp (1/2)

Beispiel: Relationsschema $R[ABC]$ mit $\Sigma = \{A \to B\}$

(Klar: Ist nicht in BCNF, da A kein Superschlüssel)

Instanz $I \in inst(R, \Sigma)$:

$I =$

A	B	C
1	2	3
1	2	4

Wähle eine Position von I $\rightsquigarrow$

A	B	C
1	2	3
1	2	4

Löschen des Eintrags $\rightsquigarrow$

A	B	C
1	2	
1	2	4

- Wähle eine endliche Teilmenge von **dom**, die echt größer ist als $adom(I)$, etwa: $\{1, \ldots, 6\}$
- Wahrscheinlichkeitsverteilung für's Einsetzen einzelner Elemente an der gelöschten Position:
  $P(4) = 0$ (denn: I hat 2 Zeilen!) und f.a. $i \in \{1, \ldots, 6\} \setminus \{4\}$ gilt: $P(i) = \frac{1}{5}$.
- Entropie = $\lg(5) \approx 2,322$ = durchschnittlicher Informationsgehalt eines Eintrags an der ausgewählten Position

Entropie zum Messen von Redundanz — Bsp (2/2)

Beispiel: Relationsschema $R[ABC]$ mit $\Sigma = \{A \to B\}$

(Klar: Ist nicht in BCNF, da A kein Superschlüssel)

Instanz $I \in inst(R, \Sigma)$:

$I =$

A	B	C
1	2	3
1	2	4

Wähle eine Position von I $\rightsquigarrow$

A	B	C
1	2	3
1	2	4

Löschen des Eintrags $\rightsquigarrow$

A	B	C
1		3
1	2	4

- Wie eben: Wähle eine endliche Teilmenge von **dom**, die echt größer ist als $adom(I)$, etwa: $\{1, \ldots, 6\}$
- Jetzt aber: Wähle eine andere Position von I
- Wahrscheinlichkeitsverteilung für's Einsetzen einzelner Elemente an der gelöschten Position:
  $P(2) = 1$ (denn: $I \models \Sigma$ und $(A \to B) \in \Sigma$) und
  f.a. $i \in \{1, \ldots, 6\} \setminus \{2\}$ gilt: $P(i) = 0$.
- Entropie = $\lg(1) =$ **0** = durchschnittlicher Informationsgehalt eines Eintrags an der ausgewählten Position

Entropie zum Messen von Redundanz

Grundidee:

Messen, wieviel Information man gewinnt, wenn der Eintrag an einer bestimmten Position verloren geht und dann an dieser Position wieder ein Eintrag eingesetzt wird (entweder der Originaleintrag oder ein anderer, der die FDs aus Σ nicht verletzt und die Zeilenanzahl von I nicht verändert).

Entropie zum Messen von Redundanz — Präzise

Definition 6.24

Sei (R, Σ) ein Relationsschema, $k \in \mathbf{dom} = \mathbb{N}_{\geqslant 1}$, $I \in inst_k(R, \Sigma)$ und sei $pos \in Pos(I)$ eine Position von I.

(a) Für jedes $a \in \mathbf{dom}$ sei $I_{pos \leftarrow a}$ die Relation, die aus I entsteht, indem an Position pos der Wert a eingesetzt wird.
D.h.: Ist $pos = (R, t, A)$ mit $t \in I$ und $A \in sort(R)$, so ist $I_{pos \leftarrow a} := (I - \{t\}) \cup \{t_{pos \leftarrow a}\}$, wobei für alle $B \in sort(R)$ gilt:

$$t_{pos \leftarrow a}(B) := \begin{cases} a & \text{falls } B = A \\ t(B) & \text{sonst} \end{cases}$$

(b) Der Zufallsraum $\mathcal{E}^k_\Sigma(I, pos) = (\{1, \ldots, k+1\}, P)$ besteht aus den Ereignissen $\{1, \ldots, k+1\}$, denen die folgenden Wahrscheinlichkeiten $p_a := P(a)$, für alle $a \in \{1, \ldots, k+1\}$ zugeordnet werden:

$$p_a := P(a) := \begin{cases} 0\,, & \text{falls } \big(I_{pos \leftarrow a} \not\models \Sigma \text{ oder } |I_{pos \leftarrow a}| \neq |I| \big) \\ \dfrac{1}{|\{b \in \{1, \ldots, k+1\} : I_{pos \leftarrow b} \models \Sigma \text{ und } |I_{pos \leftarrow b}| = |I|\}|}\,, & \text{sonst} \end{cases}$$

Insbesondere: Die Entropie $H(\mathcal{E}^k_\Sigma(I, pos)) = \sum_{a \in \{1, \ldots, k+1\}} p_a \cdot \lg\left(\frac{1}{p_a}\right)$ gibt in etwa den mittleren Informationsgehalt von Position pos von I an.

Informationstheoretische Charakterisierung von BCNF

Theorem 6.25 (Arenas, Libkin, 2003)

Sei (R, Σ) ein Relationsschema (insbes. sei Σ eine Menge von FDs über $sort(R)$). Dann gilt:

(R, Σ) *ist in BCNF*

$\iff$ *Für alle* $k \in \mathbf{dom} = \mathbb{N}_{\geqslant 1}$, $I \in inst_k(R, \Sigma)$ *und* $pos \in Pos(I)$ *gilt:* $H\big(\mathcal{E}^k_\Sigma(I, pos)\big) > 0$

Beweis: Siehe Tafel.

Bemerkung: Anschaulich besagt dieses Theorem, dass (R, Σ) genau dann in BCNF ist, wenn es keine Redundanzen gibt, d.h. keine Position den mittleren Informationsgehalt 0 hat.

Bemerkungen

- Theorem 6.25 liefert eine elegante Charakterisierung der BCNF.
- **ABER:** Als "gutes" Maß für die Redundanz an Position pos ist $H\big(\mathcal{E}^k_\Sigma(I, pos)\big)$ zu grobkörnig, da es z.B. keinen Unterschied zwischen den beiden folgenden Relationen macht:

$R[ABC]$ mit $\Sigma := \{A \to B\}$

$I =$

A	B	C
1	2	3
1	■	4

Entropie $H(\mathcal{E}^k_\Sigma(I, pos)) = 0$

$J =$

A	B	C
1	2	3
1	2	4
1	■	5

Entropie $H(\mathcal{E}^k_\Sigma(J, pos)) = 0$

Somit: In beiden Fällen ist die Entropie gleich.
Aber intuitiv ist die "Redundanz" bei J größer als bei I.

Ausblick

Marcelo Arenas und Leonid Libkin entwickelten ein komplizierteres Maß $\mathrm{RIC}_I(pos \,|\, \Sigma)$, den so genannten relativen Informationsgehalt von I an Position pos unter den Abhängigkeiten Σ, so dass gilt:

- $0 \leqslant \mathrm{RIC}_I(pos \,|\, \Sigma) \leqslant 1$
- je näher bei 1 desto weniger Redundanz, d.h. desto größer der Informationsgehalt
- für die Beispiele I und J von der vorherigen Folie gilt: $\mathrm{RIC}_I(pos \,|\, \Sigma) = 0,875 > 0.781 = \mathrm{RIC}_J(pos \,|\, \Sigma)$.
- Ein Relationsschema (R, Σ) heißt gut-entworfen (engl.: well-designed), falls für alle $I \in inst(R, \Sigma)$ und alle Positionen $pos \in Pos(I)$ gilt: $\mathrm{RIC}_I(pos \,|\, \Sigma) = 1$
  (d.h.: jede Instanz hat in jeder Position den maximalen Informationsgehalt)
- (R, Σ) ist gut-entworfen $\iff$ (R, Σ) ist in BCNF
- Beim Durchführen des Algorithmus zur BCNF-Dekomposition sinkt der Informationsgehalt an keiner Position.
- Ähnliche Resultate auch für andere Normalformen (3NF, 4NF, ...) sowie für XML.