

2 Datenmodelle

Bisher:

- Datenbankpraxis
 - Welche Objekte bietet relationale Datenbank?
 - Wie manipuliert man Datenbank-Objekte?
 - Wie greift man aus Programm auf Datenbank zu?
- Datenbankstruktur als gegeben betrachtet

Jetzt:

- Datenmodellierung
 - Wie designt man eine "vernünftige" Datenbankstruktur?
 - Welche Strukturen sind "(un)vernünftig"?

1

2.1 Überblick (2)

ANSI/SPARC Architektur

- externe Ebene
 - Benutzersicht einzelner Endanwender auf die Daten
 - z.B. in Masken des User-Interface manifestiert
 - mehrere Sichten möglich
- konzeptionelle Ebene
 - logische Gesamtsicht auf die Daten
 - das umgangssprachliche "Datenmodell" der Anwendung
- interne Ebene
 - physische Datenstruktur auf dem Rechner

Verhältnis zu den Ebenen des Datenbankentwurfs?

3

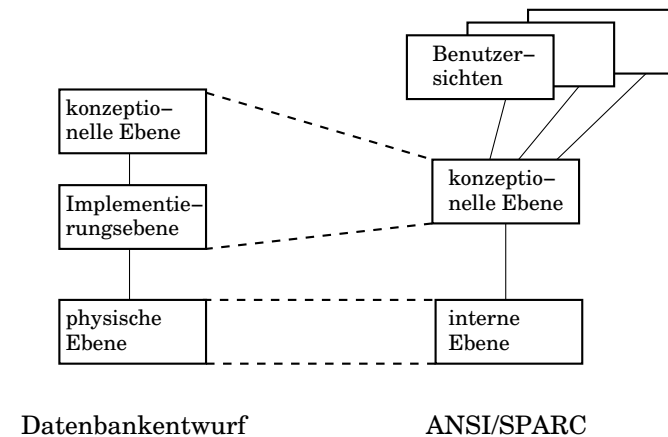
2.1 Überblick (1)

Ebenen des Datenbankentwurfs:

- konzeptionelle Ebene
 - logische Gesamtsicht des Anwenders auf die Daten
 - unabhängig vom eingesetzten DBS-Typ
- Implementierungsebene
 - konzeptionelle Datenstrukturen im Rahmen des eingesetzten DBS
 - bei relationalem DBS z.B. Tabellen
- physische Ebene
 - konkrete Implementierung der Strukturen im Rahmen des eingesetzten DBS
 - betrachtete Strukturen: Datenblöcke, Zeiger, Indexstrukturen

2

2.1 Überblick (3)



4

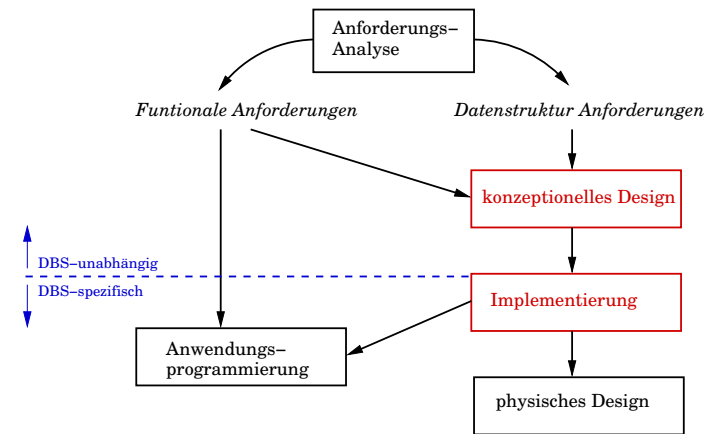
2.1 Überblick (4)

Themeneinordnung

Thema	Entwurfsebene
2.2 Relationales Modell: Definition, Normalisierung	Implementierungsebene
2.3 Relationales Modell: Redundanzvermeidung, Abhängigkeiten, Normalformen	Implementierungsebene
2.4 Semantischer Ansatz Entity/Relationship	konzeptionelle Ebene
2.5 Relationale Algebra und relationaler Kalkül	Implementierungsebene
... Speicherstrukturen	physische Ebene

5

2.1 Überblick (6)



7

2.1 Überblick (5)

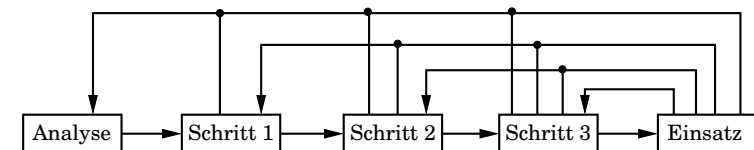
Der Entwurfsprozess

- **Anforderungsanalyse**
 - im Gespräch mit den zukünftigen Anwendern
 - Ergebnis ist *Pflichtenheft*
 - umfasst Anforderungen an Datenstruktur und Funktionen
- **konzeptionelles Design**
 - Beschreibung der Datenstrukturen auf konzeptioneller Ebene
- **Implementierung**
 - Umsetzen konzeptionelles Design in relationales Modell
 - Formulierung in Data Definition Language (DDL)
- **physisches Design**
 - erfolgt durch DBS aufgrund DDL
- **Anwendungsprogrammierung**
 - Umsetzung Funktionen in User-Interface und weitere Programme

6

2.1 Überblick (7)

Entwurfsprozess ist keine Einbahnstraße



Aber Faustregel:

- Kosten für Änderung wachsen exponentiell in
- Schrittnummer, d.h. 1 Euro Änderungskosten in
- Analysephase entsprechen 100 Euro in Realisierung

8

2.1 Überblick (8)

Vorgehen Anforderungsanalyse:

- Identifikation von Organisationseinheiten, Nutzerkreisen und zu unterstützenden Aufgaben
- Ermittlung zu befragender Personen und Sammlung Anforderungen
- Filterung der Informationen bzgl. Verständlichkeit Eindeutigkeit; iterative Rücksprache bzw. weitere Befragung
- Anforderungen klassifizieren bzgl. zu verwaltender Daten und Operationen auf den Daten
- Formalisieren der Anforderungen in einem *Pflichtenheft*, das vom fachlichen Projektbetreuer verstanden und bestätigt werden muss

9

2.2.1 Relationen (2)

Vergleich der Begriffe Relation und Tabelle

relationaler Begriff	“Umgangssprache”
Relationsschema	Tabellendefinition
Relation (-szustand, -sinstanz)	Tabelle
Attribut	Spaltenname
Domain (Wertebereich)	Datentyp
Tupel	Reihe, Zeile

Bemerkungen

- Eine *Relationen* variiert mit der Zeit, während das *Relationsschema* weitgehend konstant bleibt
- Unterschiede Tabelle und Relation später

11

2.2 Das relationale Modell

2.2.1 Relationen

Grundbegriffe einer Relation:

hnr :char(2)	name :varchar(30)	stadt :varchar(30)
H1	Henkel	Düsseldorf
H2	Pelikan	Hannover

10

2.2.1 Relationen (3)

Bestandteile eines *Relationsschemas*

$$R(A_1:\text{dom}(A_1), A_2:\text{dom}(A_2), \dots, A_n:\text{dom}(A_n))$$

- *Relationsname* R
- *Attributliste* A_1, \dots, A_n

Jedem Attribut A_i ist eine Menge möglicher Werte zugeordnet, die *Domain* $\text{dom}(A_i)$.
Domains können endlich oder unendlich sein.

Relationsschema beschreibt *Relation* mit Namen R .
Die Attributanzahl n heißt *Grad* der Relation.

12

2.2.1 Relationen (4)

Eine *Relation* $r(R)$ des Relationsschemas $R(A_1, \dots, A_n)$ ist eine endliche Menge von n -Tupeln

$$t_i = (x_1, \dots, x_n) \quad \text{mit } x_j \in \text{dom}(A_j) \cup \{NULL\}$$

NULL ist ein spezieller *Nullwert*.

Eine Relation r können wir also schreiben:

$$r = \bigcup_{i=1}^m \{t_i\}$$

Für den j -ten Wert x_j in Tupel t (dh. für den Wert des Attributs A_j im Tupel t) schreiben wir auch $t[A_j]$. 13

2.2.1 Relationen (5)

Umformulierung Relations-Definition:

Eine Relation $r(R)$ ist eine *mathematische Relation* n -ten Grades auf den Mengen $\text{dom}(A_1), \dots, \text{dom}(A_n)$.

D.h. $r(R)$ ist eine Teilmenge des kartesischen Produkts der Wertebereiche:

$$r(R) \subseteq \text{dom}(A_1) \times \text{dom}(A_2) \times \dots \times \text{dom}(A_n)$$

Konsequenzen der Definition als *Menge*:

- keine Ordnung auf den Tupeln definiert
- keine zwei identischen Tupel (Doubletten) möglich

14

2.2.1 Relationen (6)

Relationale Datenbanken und SQL basieren *nicht* auf Relationen (Mengen von Tupeln), sondern auf *ungeordneten Listen* (engl. *Bags*, *Multisets*), d.h.

- Tupel sind ungeordnet
- Doubletten sind möglich

Teilweise erlaubt SQL Wahl (*distinct* versus *all*)

Vorteile von Bags:

- Eliminierung von Doubletten ist aufwendige Operation
=> Bags sind deutlich effizienter als Mengen
- stillschweigendes Verschwinden von Tupeln ist insbes. bei Projektion auf wenige Spalten meist unerwünscht
- Aggregatfunktionen wie *count*, *avg* oft sinnlos auf Mengen

15

2.2.1 Relationen (7)

Weitere Verallgemeinerung:

- Attributreihenfolge in Relation unerheblich
- in formaler Definition Tupel nicht mehr als geordnete Liste von Werten auffassen, sondern als *Menge von Attribut / Wert Paaren*:

$$t = \bigcup_{i=1}^n \{A_i : x_i\} \quad \text{wobei } x_i \in \text{dom}(A_i) \cup \{NULL\}$$

- Voraussetzung: keine gleichnamigen Attribute in Relation
- Wir nehmen im Folgenden Attributreihenfolge als egal an, verwenden aber die (einfachere) Listennotation

Wie verhält sich SQL diesbezüglich?

- gleichnamige Attribute in Relation unzulässig
- Attribute können in beliebiger Reihenfolge angegeben werden
- Aber: *INSERT* ohne Attributnamen gemäß fester Reihenfolge

16

2.2.1 Relationen (8)

Notationskonventionen

$R(A_1, A_2, \dots, A_n)$	Relationsschema n -ten Grades Relationsnamen: Q, R, S
$r(R), r$	Relationszustand. Buchstaben: q, r, s
$t = (x_1, x_2, \dots, x_n)$	Tupel $\in r$. Buchstaben: t, u, v
$t[A_i]$	Wert des Attributs A_i im Tupel t
$t[A_{i_1}, A_{i_2}, \dots, A_{i_k}]$	Subtupel $(t[A_{i_1}], t[A_{i_2}], \dots, t[A_{i_k}])$

Weitere Konventionen:

- Name des Relationsschemas (z.B. *hersteller*) bezeichne auch aktuelle Tupelmenge (Relationszustand). Dagegen bezeichnet *hersteller(hnr, name, stadt)* nur das Relationsschema
- Attribut kann mit Relationsnamen qualifiziert werden mittels Punktnotation, z.B. *hersteller.name*

17

2.2.2 Integrität (2)

Ein *Schlüsselkandidat* ist ein *minimaler Superkey*, d.h. ein Superkey, aus dem kein Attribut entfernt werden kann ohne die Eindeutigkeits-Bedingung zu verletzen.

Primärschlüssel:

- ◻ im allg. hat Relation mehrere Schlüsselkandidaten
- ◻ ein Kandidat wird als *Primärschlüssel* ausgewählt

Bemerkungen:

- Primärschlüssel besteht im allg. aus mehreren Attributen
- gesamtes Tupel (dh. alle Attribute) ist durch Angabe der Primärschlüsselwerte identifiziert
- Notationskonvention: Primärschlüsselattribute durch angehängtes Doppelkreuz '#' kennzeichnen

19

2.2.2 Integrität (1)

Primärschlüssel

Eine Teilmenge $\{A_{k_1}, \dots, A_{k_m}\}$ von Attributen heißt *Schlüsselobermenge (Superkey)* des Relationsschemas R , wenn es keine zwei Tupel geben kann, die in allen Werten dieser Attribute gleich sind, d.h.

$$t_i[A_{k_1}, \dots, A_{k_m}] \neq t_j[A_{k_1}, \dots, A_{k_m}] \text{ für alle } i \neq j$$

Bemerkungen:

- Ob die Eindeutigkeits-Bedingung erfüllt ist, kann nur aufgrund der *Bedeutung* der Attributwerte entschieden werden
- Jede Relation hat mindestens einen Superkey (Welchen und Warum?)
- im allgemeinen gibt es mehrere Superkeys

18

2.2.2 Integrität (3)

Beispiel:

auto

amtlkz#	fahrgestellnr	hersteller	modell	jahr
KR-AD 102	A69352	VW	Beetle	2000
DU-PW 430	X83554	Ford	KA	2001

Fragen:

- ◻ Welche Superkeys gibt es?
- ◻ Wieviele Superkeys gibt es?
- ◻ Welche Schlüsselkandidaten gibt es?

20

2.2.2 Integrität (4)

Ein Primärschlüssel ist eine *Integritätsbedingung* für eine einzelne Relation:

- keine zwei Tupel dürfen dieselben Primärschlüsselwerte haben
- kein Primärschlüsselwert darf NULL sein

Neben diesen *Entity Integrity Constraints* gibt es auch noch *Referential Integrity Constraints*, die sich auf zwei Relationen beziehen:

- erzwingt die Existenz eines Tupels in einer Relation, wenn sich ein Tupel einer anderen Relation darauf bezieht
- realisiert durch *Fremdschlüssel*

21

2.2.2 Integrität (6)

Beispiel für Fremdschlüssel:

hersteller			produkt			
<i>hnr#</i>	<i>name</i>	<i>stadt</i>	<i>pnr#</i>	<i>name</i>	<i>preis</i>	<i>hnr</i>
H1	Henkel	Düsseldorf	P1	Pritt	2.50	H1
H2	Pelikan	Hannover	P2	Papier	5.30	NULL

- Fremdschlüssel $\{hnr\}$ von *produkt* referenziert *hersteller*
- für das Attribut *produkt.hnr* sind nur die Werte 'H1', 'H2' oder NULL zulässig
- Tupel $s = ('P1', 'Pritt', 2.50, 'H1')$ aus *produkt* referenziert $t = ('H1', 'Henkel', 'Düsseldorf')$ aus *hersteller*
- Tupel mit *hersteller.hnr* = 'H2' darf gelöscht werden, Tupel mit *hersteller.hnr* = 'H1' nicht

23

2.2.2 Integrität (5)

Eine Attributmeng $F \subseteq \{A_1, \dots, A_n\}$ der Relation R_1 ist ein *Fremdschlüssel* von R_1 , der die Relation R_2 referenziert, wenn gilt:

- die Attribute in F haben dieselben Domains wie die Primärschlüsselattribute von R_2
- die Werte $t_1[F]$ in einem Tupel $t_1 \in R_1$ kommen entweder als Primärschlüsselwerte in einem Tupel $t_2 \in R_2$ vor, oder sie sind alle *NULL*

Wir sagen: " t_1 referenziert t_2 ".

R_1 ist die *referenzierende*, R_2 die *referenzierte* Relation.

22

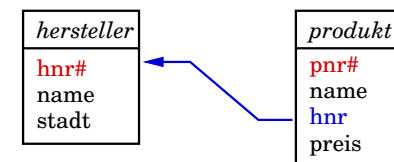
2.2.2 Integrität (7)

Komponenten eines *relationalen Datenbankschemas*:

- Menge von Relationsschemas
- Menge von Integritätsbedingungen

mögliche Darstellungen:

- textuelle Darstellung (z.B. SQL-DDL)
- grafische Darstellung mit Pfeilen für Fremdschlüssel



24

2.2.2 Integrität (8)

Relationales Schema in SQL-DDL

```
CREATE TABLE hersteller (
  hnr CHAR(2), name VARCHAR(30), stadt VARCHAR(30),
  PRIMARY KEY(hnr)
);

CREATE TABLE produkt (
  hnr CHAR(2), name VARCHAR(30), preis NUMERIC(8,2),
  PRIMARY KEY(hnr,name),
  FOREIGN KEY(hnr) REFERENCES hersteller(hnr)
);

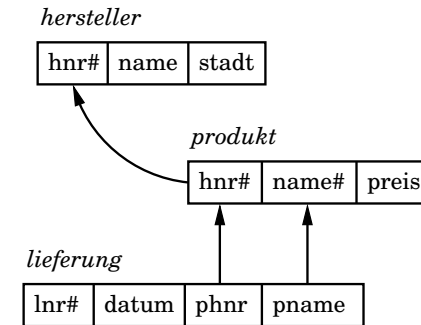
CREATE TABLE lieferung (
  lnr INT, datum DATE,
  phnr CHAR(2), pname VARCHAR(30),
  PRIMARY KEY(lnr),
  FOREIGN KEY(phnr,pname) REFERENCES produkt(hnr,name)
);
```

25

2.2.2 Integrität (10)

Grafische Darstellung

Variante 2: Relationsschema als Zeilenvektor

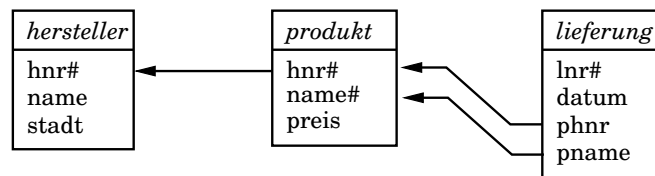


27

2.2.2 Integrität (9)

Grafische Darstellung

Variante 1: Relationsschema als Spaltenvektor



Notationen:

- Darstellung ähnelt UML-Klassendiagramm
- Primärschlüssel durch Doppelkreuz (#)
- Fremdschlüssel durch Pfeil zum referenzierten Schlüsselattribut bei zusammengesetzten Schlüsseln mehrere Pfeile (Warum?)

26

2.2.3 Normalisierung (1)

Relationales Modell nach Codd (1970):

- Domains enthalten nur *atomare* Werte
- Attributwert ist *Einzelwert* aus Domain

Ein relationales Modell mit dieser Eigenschaft ist in *erster Normalform* (1NF).

Da diese Bedingung Teil der Definition des relationalen Modells ist, ist ein relationales Schema immer in 1NF.

Das Umformulieren eines Schemas derart, dass die erste Normalform erfüllt ist, heißt *Normalisierung*.

28

2.2.3 Normalisierung (2)

Unzulässig in 1NF:

hersteller			
<i>hnr#</i>	<i>name</i>	<i>adresse</i>	<i>branche</i>
H1	Henkel	Henkelstr. 67 40191 Düsseldorf	{Klebstoff, Waschmittel, Kosmetik}

- *adresse* besteht aus drei Unterwerten
- *branche* ist Menge von Werten, d.h. selber wieder Relation

Bemerkungen:

- *Objektrelationale* DBS (z.B. PostgreSQL) erlauben benutzerdefinierte zusammengesetzte Datentypen und Arrays. Erfordert Möglichkeit, eigene Operatoren zu definieren.
- Achtung: manche Autoren (z.B. Date) lassen frei definierte Datentypen auch im rein relationalen Modell zu
- *Verschachteltes relationales Modell* erlaubt Relationen als Werte

29

2.2.3 Normalisierung (5)

Lösung 1:

<i>hnr#</i>	<i>name</i>	<i>strasse</i>	<i>plz</i>	<i>ort</i>
H1	Henkel	Henkelstr. 67	40191	Düsseldorf

<i>hnr#</i>	<i>branche#</i>
H1	Kosmetik
H1	Waschmittel
H1	Klebstoff

hersteller
hersteller_branche

- entferne Attribut *branche* aus Relation *hersteller*
- bilde neue Relation *hersteller_branche*, die *branche* und den Primärschlüssel von *hersteller* enthält
- Primärschlüssel von *hersteller_branche* ist $\{hnr, branche\}$
- für jede Branche eigenes Tupel in *hersteller_branche*

31

2.2.3 Normalisierung (3)

hersteller

<i>hnr#</i>	<i>name</i>	<i>adresse</i>	<i>branche</i>
H1	Henkel	Henkelstr. 67 40191 Düsseldorf	{Klebstoff, Waschmittel, Kosmetik}

Wie kann dieses Schema normalisiert werden?

- zusammengesetzter Wert für *adresse*:
· ersetze *adresse* durch drei Felder *strasse*, *plz*, *ort*
- Mehrfachwerte für *branche*:
· verschiedene Lösungen möglich

30

2.2.3 Normalisierung (6)

Lösung 2:

hersteller

<i>hnr#</i>	<i>name</i>	<i>strasse</i>	<i>plz</i>	<i>ort</i>	<i>branche#</i>
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Kosmetik
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Waschmittel
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Klebstoff

- Füge in *hersteller* pro *branche* ein Tupel ein, so dass für einen Hersteller mit n Branchen n Tupel vorhanden sind
- Der Primärschlüssel von *hersteller* muss dann erweitert werden zu $\{hnr, branche\}$ (Warum?)

32

2.2.3 Normalisierung (7)

Lösung 3:

hersteller

hnr#	name	strasse	plz	ort
H1	Henkel	Henkelstr. 67	40191	Düsseldorf

branche1	branche2	...	brancheN
Klebstoff	Waschmittel	...	NULL

- möglich, wenn maximale Anzahl Branchen N bekannt
- ersetze in *hersteller* das Attribut *branche* durch die N Attribute *branche1*, *branche2*, ..., *brancheN*
- bei weniger Branchen Werte mit NULL auffüllen

33

2.2.3 Normalisierung (9)

Redundanzen und Anomalien

Die in Lösung 2 eingeführten Redundanzen sind besonders unerwünscht, weil sie zu Inkonsistenzen bei Änderungen führen (*Update-Anomalien*)

Das Problem ist, dass bei Änderungen *mehrere* Tupel verändert werden müssen, wenn die Information nicht inkonsistent werden soll.

Problem betrifft alle Änderungen:

- Einfüge-Anomalien
- Lösch-Anomalien
- Modifikations-Anomalien

35

2.2.3 Normalisierung (8)

Bewertung der drei Lösungen

- Nachteile von Lösung 3:
 - begrenzt maximale Anzahl Branchen
 - führt Nullwerte ein für Hersteller mit weniger als N Branchen
- Nachteile von Lösung 2:
 - führt Redundanzen ein: überflüssige Mehrfachspeicherung der Attribute *hnr*, *name*, *strasse*, *plz*, *ort*
=> Speicherplatzverschwendung
=> Updates kompliziert, da Gefahr der Inkonsistenz
- Lösung 1 ist unbedingt vorzuziehen

34

2.2.3 Normalisierung (10)

hersteller

hnr#	name	strasse	plz	ort	branche#
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Kosmetik
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Waschmittel
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Klebstoff

Einfüge-Anomalien (1)

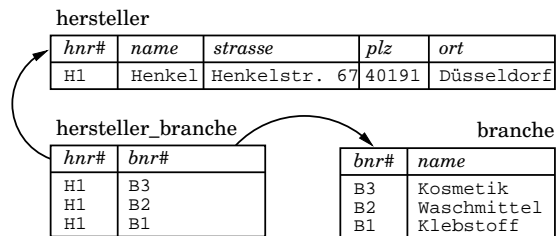
- Wenn zum Hersteller 'H1' eine weitere Branche ergänzt werden soll, müssen für *name*, *strasse*, *plz*, *ort* die passenden Werte eingefügt werden (sonst Inkonsistenz!)
- Korrektheit dieser Werte lässt sich (außer über Trigger) nicht einfach durch Integrity Constraints erzwingen
=> keine Garantie für konsistenten Inhalt

36

2.2.3 Normalisierung (11)

Lösch-Anomalien

- Wenn ein Hersteller gelöscht wird, der zufällig als einziger einer bestimmten Branche angehört, werden die Informationen über diese Branche alle mitgelöscht
- Problem betrifft auch Lösung 1 und kann durch Einführung einer Referenztabelle *branche* gelöst werden



37

2.2.3 Normalisierung (13)

hersteller

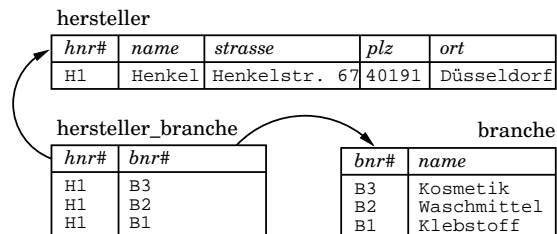
hnr#	name	strasse	plz	ort	branche#
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Kosmetik
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Waschmittel
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Klebstoff

Modifikations-Anomalien

- Wenn sich die Adresse des Herstellers 'H1' ändert, muss sie in allen Tupeln gleich geändert werden
- eine nicht gleichartige oder nur teilweise Änderung lässt sich nicht automatisch abfangen und abweisen

39

2.2.3 Normalisierung (12)



Einfüge-Anomalien (2)

- Lösung mit Referenztabelle *branche* ermöglicht auch das Einfügen von Branchen *unabhängig* von Herstellern
- insbesondere wird es dadurch möglich, Branchen einzufügen, denen noch kein Hersteller zugeteilt ist

38

2.2.4 Designrichtlinien

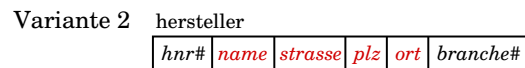
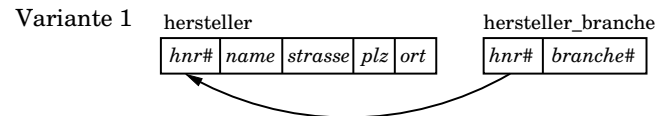
Informelle Qualitätsmaße für den Entwurf eines Relationsschemas:

- Inhaltliche Bedeutung (Semantik) der Relationen und Attribute sollte leicht verständlich sein
- Reduzierung redundanter Werte in Tupeln
- Reduzierung der Nullwerte in Tupeln
- Verhinderung der Erzeugung "unechter" Tupel

40

2.3 Normalformen (1)

Bei Normalisierung entstanden Modelle mit selbem Informationsgehalt, aber unterschiedlicher "Güte":



- Variante 2 führt zu unerwünschten *Redundanzen*, die *Update-Anomalien* verursachen
- Variante 1 hat diesen Nachteil nicht

Ziele:

- formale Definition von Redundanzen
- Kriterien für Redundanzfreiheit

41

2.3.1 Abhängigkeit (1)

2.3.1 Funktionale Abhängigkeit

Eine *funktionale Abhängigkeit* " $X \rightarrow Y$ " zwischen zwei Attributmengen X, Y eines Relationsschemas R besteht, wenn für beliebige Tupel $t_1, t_2 \in r(R)$ gilt:

$$t_1[X] = t_2[X] \implies t_1[Y] = t_2[Y]$$

Die Werte der X -Komponente eines Tupels bestimmen also eindeutig alle Werte der Y -Komponente.

Die Y -Werte sind also eine Funktion der X -Werte:

$$t[Y] = f(t[X])$$

43

2.3 Normalformen (2)

Redundanz eines Attributs:

- Ein Attribut ist *redundant*, wenn einzelne Attributwerte ohne Informationsverlust weggelassen werden können.

hersteller

redundante Attribute

hnr#	name	strasse	plz	ort	branche#
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Kosmetik
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Waschmittel
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Klebstoff

überflüssige Werte

Wie können wir Redundanz formal definieren?

42

2.3.1 Abhängigkeit (2)

Beispiel

hersteller

hnr#	name	strasse	plz	ort	branche#
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Kosmetik
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Waschmittel
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Klebstoff

Es gilt

- $\{hnr, branche\} \rightarrow \{name, strasse, ort, plz\}$
- $\{hnr\} \rightarrow \{strasse, ort\}$
- $\{hnr, name\} \rightarrow \{plz\}$

Es gilt nicht

- $\{hnr\} \rightarrow \{branche\}$
- $\{branche\} \rightarrow \{name, plz\}$

44

2.3.1 Abhängigkeit (3)

hersteller					
hnr#	name	strasse	plz	ort	branche#
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Kosmetik
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Waschmittel
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Klebstoff

Bemerkungen (1)

- funktionale Abhängigkeit ist Eigenschaft des *Relationsschemas R* und nicht eines *Relationszustands r(R)*
 - ▷ In obiger Beispielrelation gilt $\{strasse\} \rightarrow \{ort\}$
 - ▷ Trotzdem keine funktionale Abhängigkeit
- funktionale Abhängigkeit ist Eigenschaft der *Bedeutung (Semantik) der Attributwerte*
 - ▷ keine beweisbaren Tatsachen
 - ▷ erfordert Intuition des Datenbankdesigners

45

2.3.1 Abhängigkeit (5)

Armstrongs Regeln

IR1 Reflexivität:

$$Y \subseteq X \Rightarrow X \rightarrow Y$$

IR2 Augmentation:

$$X \rightarrow Y \Rightarrow X \cup Z \rightarrow Y \cup Z$$

IR3 Transitivität:

$$X \rightarrow Y \text{ und } Y \rightarrow Z \Rightarrow X \rightarrow Z$$

(Beweis der Regeln an der Tafel)

47

2.3.1 Abhängigkeit (4)

hersteller					
hnr#	name	strasse	plz	ort	branche#
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Kosmetik
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Waschmittel
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Klebstoff

Bemerkungen (2)

- Wenn $P = \{hnr, branche\}$ die Menge der Primärschlüsselattribute ist, dann gilt für jede beliebige Attributmenge X : $P \rightarrow X$
 - Aus gegebenen funktionalen Abhängigkeiten lassen sich weitere Abhängigkeiten ohne Kenntnis der Attributbedeutung ableiten
 - ▷ z.B. folgt aus $\{hnr\} \rightarrow \{strasse, ort\}$ automatisch $\{hnr\} \rightarrow \{strasse\}$
- Für die Ableitung weiterer Abhängigkeiten gibt es Regeln, die sog. *Inferenzregeln (Inference Rules)*

46

2.3.1 Abhängigkeit (6)

Anmerkungen zu Armstrongs Regeln:

- Regeln IR1 und IR2 aus Buch von Elmasri, Navathe sind falsch!
- Die durch IR1 gegebenen Abhängigkeiten heißen *trivial*
- Armstrong hat 1974 gezeigt, dass die Regeln IR1-3 *vollständig* sind:
 - Wenn man diese Regeln solange auf eine Menge F von funktionalen Abhängigkeiten anwendet, bis keine neuen Abhängigkeiten mehr erzeugt werden, so erhält man *alle Abhängigkeiten, die aus F herleitbar sind*
- Die Menge aller aus F herleitbaren Abhängigkeiten heißt *Hülle (Closure) von F* (symbolisch F^+)

48

2.3.1 Abhängigkeit (7)

Beispiel:

hersteller					
hnr#	name	strasse	plz	ort	branche#
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Kosmetik
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Waschmittel
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Klebstoff

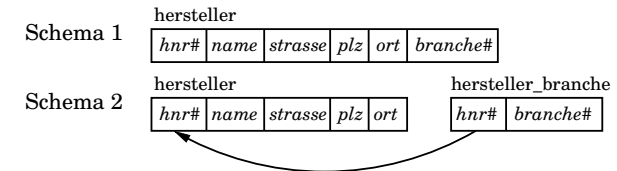
Aus $\{hnr\} \rightarrow \{name, strasse, plz, ort\}$ lässt sich nur mit Armstrongs Regeln ableiten (Wie?):

- $\{hnr\} \rightarrow \{name\}$
- $\{hnr, branche\} \rightarrow \{hnr, name, strasse, plz, ort, branche\}$

49

2.3.1 Abhängigkeit (9)

Erste Anwendung Abhängigkeiten:



- Schema 2 kann als *Zerlegung* von Schema 1 aufgefasst werden
- Relation von Schema 1 lässt sich aus den Relationen von Schema 2 komplett rekonstruieren:

```
SELECT h.hnr, name, strasse, plz, ort, branche
FROM hersteller h, hersteller_branche b
[ LEFT OUTER ] JOIN ON h.hnr = b.hnr;
```

Frage: Wann ist Zerlegung verlustfrei (additiver Join)?

51

2.3.1 Abhängigkeit (8)

Weitere, aus IR1-3 ableitbare Inferenzregeln:

IR4 Zerlegung:

$$X \rightarrow Y \cup Z \Rightarrow X \rightarrow Y \text{ und } X \rightarrow Z$$

IR5 Vereinigung:

$$X \rightarrow Y \text{ und } X \rightarrow Z \Rightarrow X \rightarrow Y \cup Z$$

IR6 Pseudotransitivität:

$$X \rightarrow Y \text{ und } W \cup Y \rightarrow Z \Rightarrow W \cup X \rightarrow Z$$

IR7 Komposition:

$$X \rightarrow Y \text{ und } V \rightarrow W \Rightarrow X \cup V \rightarrow Y \cup W$$

(Beweis: IR5+7 an Tafel, IR4+6 Übung)

50

2.3.1 Abhängigkeit (10)

Theorem (Heath 1971):

- Sei $R(A,B,C)$ ein Relationsschema, wobei A , B und C Attributmengen sind.
- Wenn die funktionale Abhängigkeit $A \rightarrow B$ gilt, dann ist jeder Relationszustand $r(R)$ gleich dem Join seiner Projektionen über $\{A,B\}$ und $\{A,C\}$

Bemerkungen:

- die Bedingung ist hinreichend, aber nicht notwendig
- äquivalentes Kriterium über mehrwertige Abhängigkeiten (später)
- Dieses Theorem ist die Grundlage des Normalisierungs-Prozesses

52

2.3.1 Abhängigkeit (11)

<i>hnr</i>	<i>name</i>	<i>strasse</i>	<i>plz</i>	<i>ort</i>	<i>branche</i>
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Kosmetik
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Waschmittel
H2	Unilever	Dammtorwall 15	20355	Hamburg	Waschmittel

Zerlegung 1

hst1					hst2	
<i>hnr</i>	<i>branche</i>	<i>strasse</i>	<i>plz</i>	<i>ort</i>	<i>hnr</i>	<i>name</i>

Zerlegung 2

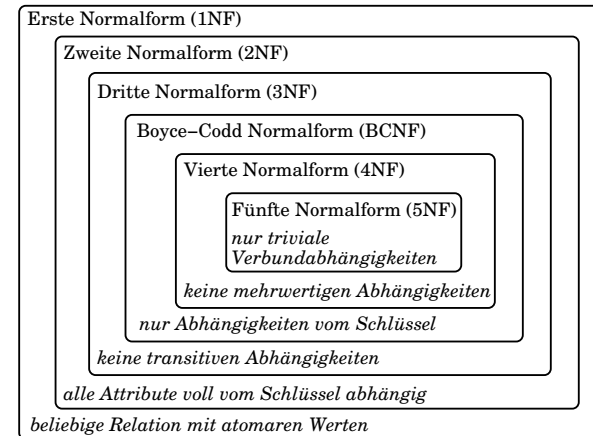
hst3					hst4	
<i>branche</i>	<i>name</i>	<i>strasse</i>	<i>plz</i>	<i>ort</i>	<i>hnr</i>	<i>branche</i>

- Join von Zerlegung 1 erzeugt wieder Ursprungstabelle
Muss laut Heath's Theorem so sein wegen $\{hnr\} \rightarrow \{name\}$
- Join von Zerlegung 2 erzeugt Zusatztuple (Welche?)
Bezeichnung in Literatur: "nicht additiver Join"
Da Heath's Theorem nur hinreichende Bedingung angibt,
kann das nicht aus fehlender Abhängigkeit gefolgert werden

53

2.3.2 2NF und 3NF (2)

NF's auf Basis projektiver Zerlegung



55

2.3.2 2NF und 3NF (1)

2.3.2 Zweite und Dritte Normalform

Normalisierungsprozess

- Formalismus zum Erkennen von Redundanzen (Normalformtests)
- Zerlegungsregeln um redundante Schemas durch weniger redundante Schemas zu ersetzen (Randbedingung: additiver Join)
- Die verschiedenen Normalformen unterscheiden sich durch die Art der Redundanzen, die noch zugelassen werden.
Höhere Normalform = strengere Kriterien

Denormalisierung

- oft werden Redundanzen geduldet. Mögliche Gründe:
 - Performance: Join über viele Tabellen oft zeitraubend
 - Praktikabilität: Anwender hat kein Recht Referenzwerte zu pflegen
- Zulassung von Redundanzen heißt *Denormalisierung*

54

2.3.2 2NF und 3NF (3)

Bemerkungen zu den Normalformen:

- jede Stufe definiert echt strengere Kriterien, d.h. ein Relationsschema in x -NF ist auch in y -NF für alle $y < x$
- 2NF, 3NF und BCNF basieren auf funktionalen Abhängigkeiten
- 4NF und 5NF basieren auf mehrwertigen Abhängigkeiten
- in der Praxis beschränkt man sich meist auf 3NF bzw. BCNF, da die Abhängigkeiten für 4NF relativ selten sind und die Abhängigkeiten für 5NF schwer zu erkennen sind
- es gibt weitere Normalformen, die allerdings nicht (ausschließlich) auf Projektion und Join beruhen, z.B. Domain-Key NF

56

2.3.2 2NF und 3NF (3)

hersteller

redundante Attribute

<i>hnr#</i>	<i>name</i>	<i>strasse</i>	<i>plz</i>	<i>ort</i>	<i>branche#</i>
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Kosmetik
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Waschmittel
H1	Henkel	Henkelstr. 67	40191	Düsseldorf	Klebstoff

überflüssige Werte

Fragen:

- Wodurch kommt Redundanz der Attribute?
- Formale Definition über funktionale Abhängigkeit möglich?

Antwort:

- redundante Attribute hängen funktional von einem Teil (*hnr*) des Primärschlüssels (*hnr, branche*) ab

57

2.3.2 2NF und 3NF (5)

Beispiel:

hst1

<i>hnr#</i>	<i>name</i>	<i>strasse</i>	<i>plz</i>	<i>ort</i>	<i>branche#</i>

hst2

<i>hnr#</i>	<i>name</i>	<i>strasse</i>	<i>plz</i>	<i>ort</i>

hst3

<i>hnr#</i>	<i>branche#</i>

- Relation *hst1* ist nicht in zweiter Normalform, denn Attribut *name* hängt von *hnr* ab, Primärschlüssel ist aber {*hnr, branche*}
- Relation *hst2* ist in zweiter Normalform, denn der Primärschlüssel enthält nur ein Attribut
- Relation *hst3* ist in zweiter Normalform, denn es gibt keine nicht primären Attribute

59

2.3.2 2NF und 3NF (4)

Definition zweite Normalform (Annahme: nur ein Schlüsselkandidat)

- Ein Relationsschema ist in *zweiter Normalform (2NF)*, wenn jedes nicht primäre Attribut *voll funktional* vom Primärschlüssel abhängt.

Erläuterungen:

- *primäre* Attribute heißen die Attribute des Primärschlüssels
- Eine Attributmeng *Y* heißt von einer Attributmeng *X* *voll funktional abhängig*, wenn sie von keiner echten Teilmenge von *X* abhängt. Tut sie es doch, heißt sie *partiell abhängig* von *X*.

58

2.3.2 2NF und 3NF (6)

Ein Relationsschema, das nicht in 2NF ist, kann in mehrere Schemas in 2NF zerlegt werden

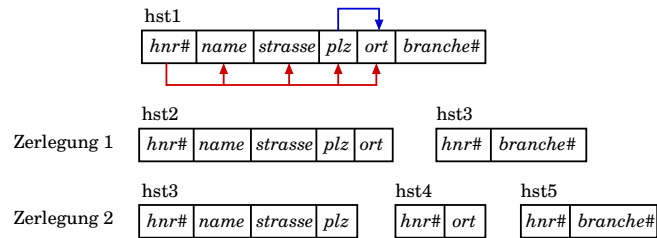
- a) Fasse alle nicht primären Attribute, die nur von einem Teilschlüssel abhängen, mit diesem Teilschlüssel als Primärschlüssel in einer eigenen Tabelle zusammen.
- b) Alle Attribute, die vom selben Teilschlüssel abhängen, müssen in derselben Tabelle zusammengefasst werden.
- c) Entferne die ausgelagerten nichtprimären Attribute aus der Ursprungstabelle.

Bemerkungen:

- Bedingung a) und Heath's Theorem gewährleisten additiven Join
- Bedingung b) nötig, um Verlust von funktionalen Abhängigkeiten zu vermeiden

60

2.3.2 2NF und 3NF (7)



- Zerlegung 1 gemäß Regeln a), b) und c)
- Zerlegung 2 verletzt Regel b), denn *ort* und *plz* hängen vom selben Teilschlüssel *hnr* ab, sind aber nicht in derselben Relation zusammengefasst
=> funktionale Abhängigkeit $\{plz\} \rightarrow \{ort\}$ geht verloren
- beide Zerlegungen erfüllen "additiven Join"

61

2.3.2 2NF und 3NF (9)

Definition Dritte Normalform
(Annahme: nur ein Schlüsselkandidat)

- Ein Relationenschema ist in *dritter Normalform (3NF)*, wenn es in 2NF ist und kein nicht primäres Attribut transitiv vom Primärschlüssel abhängt.

Bemerkungen:

- *B* heißt transitiv abhängig von *A*, wenn es eine Attributmenge *C* gibt mit $A \rightarrow C$ und $C \rightarrow B$
- die beiden Abhängigkeiten bei der Transitivität dürfen nicht trivial ($C \subseteq A$ oder $B \subseteq C$) sein

63

2.3.2 2NF und 3NF (8)

Mit der 2NF werden bestimmte Redundanzen ausgeschlossen. Andere aber nicht:

<i>hnr#</i>	<i>name</i>	<i>strasse</i>	<i>plz</i>	<i>ort</i>
H1	de Beukelaer	Arnoldstr. 62	47906	Kempen
H2	Polyant	Speefeld 7	47906	Kempen

redundanter Wert

- Relation ist in 2NF (Warum?)
- Attribut *ort* ist redundant
- Ursache:
funktionale Abhängigkeit von nichtprimärem Attribut *plz*
bzw. *transitive* Abhängigkeit $\{hnr\} \rightarrow \{plz\} \rightarrow \{ort\}$

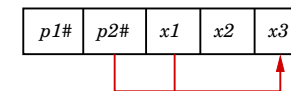
62

2.3.2 2NF und 3NF (10)

Gelegentlich andere "Definition" in Literatur:

Ein Relationenschema ist in 3NF, wenn es in 2NF ist und kein nicht primäres Attribut von einer Menge anderer nichtprimärer Attribute abhängt.

- Dies ist keine äquivalente 3NF Definition, sondern nur eine *notwendige* (aber nicht hinreichende!) Bedingung für 3NF
- Gegenbeispiel:



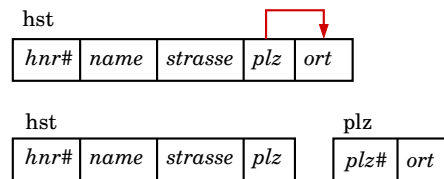
erfüllt obige "Definition"
enthält aber transitive Abhängigkeit $\{p1\#, p2\#\} \rightarrow \{p2\#, x1\} \rightarrow \{x3\}$

64

2.3.2 2NF und 3NF (11)

Ein Relationenschema, das nicht in 3NF ist, kann in mehrere Schemas in 3NF zerlegt werden

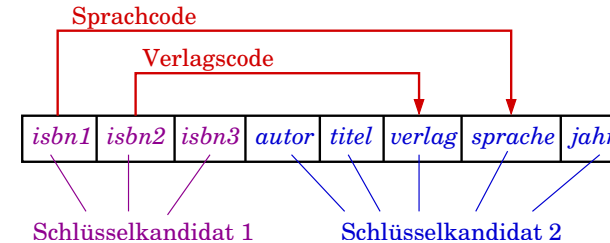
- Fasse die transitiv abhängigen nicht primären Attribute mit den Attributen, von denen sie direkt abhängen, in einer eigenen Tabelle zusammen.
- Entferne die ausgelagerten abhängigen Attribute aus der Ursprungstabelle.



65

2.3.2 2NF und 3NF (13)

Beispiel:



- Relation *buch* hat mehrere Schlüsselkandidaten
- Relation ist in Codd-2NF, da alle Attribute zu Schlüsselkandidaten gehören
- Relation ist nicht in Kent-2NF, da *sprache* partiell abhängig ist vom Teil *isbn1* der ISBN-Nr

67

2.3.2 2NF und 3NF (12)

Verallgemeinerung Definition 2NF auf beliebige Anzahl Schlüsselkandidaten

- Definition 2NF Codd (1971)
Jedes Attribut, das zu keinem Schlüsselkandidaten gehört, ist von jedem Schlüsselkandidaten voll funktional abhängig.
- Definition 2NF Kent (1973)
Jedes Attribut im Komplement eines Schlüsselkandidaten ist von diesem Schlüsselkandidaten voll funktional abhängig.

Bemerkungen:

- Definitionen stimmen für nur einen Schlüsselkandidaten überein
- im allgemeinen Fall ist die Codd-2NF *schwächer* (d.h. lässt mehr Redundanzen zu) als die Kent-2NF (vgl. Übung 5.1)

66

2.3.2 2NF und 3NF (14)

Verallgemeinerung Definition 3NF auf beliebige Anzahl Schlüsselkandidaten

- Definition 3NF Codd
2NF und kein Attribut, das zu keinem Schlüsselkandidaten gehört, ist von einem Schlüsselkandidaten transitiv abhängig.
- Definition 3NF Kent
2NF und kein Attribut im Komplement eines Schlüsselkandidaten ist von diesem Schlüsselkandidaten transitiv abhängig.

Bemerkungen:

- Definitionen stimmen für nur einen Schlüsselkandidaten überein
- wieder ist die Codd-3NF *schwächer*

68

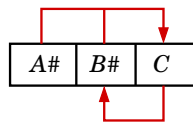
2.3.2 2NF und 3NF (15)

Boyce/Codd Normalform (BCNF)

- Für jede nicht triviale Abhängigkeit $X \rightarrow A$
- ist X ein Superschlüssel der Relation.

Bemerkungen:

- erfordert keine Überprüfung von 2NF
- ursprünglich als einfachere Definition der 3NF vorgeschlagen, dann aber nachgewiesen, dass BCNF strenger als 3NF ist: $BCNF \Rightarrow 3NF$, aber nicht umgekehrt:



Relation in 3NF,
aber nicht in BCNF

69

2.3.3 4NF (1)

Mehrwertige Abhängigkeiten

Betrachte Relation mit mehreren unabhängigen mengenwertigen Attributen (d.h. nicht in 1NF!)

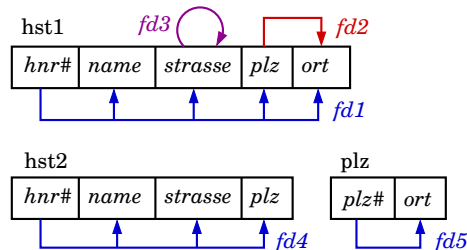
<i>ketten#</i>	<i>branche</i>	<i>standort</i>
Karstadt	{Lebensmittel, Bekleidung}	{Essen, Dortmund}
Leffers	{Bekleidung}	{Dortmund, Hannover}

Ersetzen durch skalare Attribute führt zu Redundanzen, die keiner funktionalen Abhängigkeit entsprechen

71

2.3.2 2NF und 3NF (16)

Relation R ist in BCNF, wenn für jede nicht triviale Abhängigkeit $X \rightarrow A$ die Menge X ein Superschlüssel von R ist.



- Abhängigkeiten $fd1$ in Relation $hst1$ sind mit BCNF verträglich
- Abhängigkeit $fd3$ auch (Warum?), aber $fd2$ nicht
- Relationen $hst2$ und plz sind in BCNF

70

2.3.3 4NF (2)

Schlechte Umformung in 1NF:

<i>ketten#</i>	<i>branche#</i>	<i>standort#</i>
Karstadt	Lebensmittel	Essen
Karstadt	Lebensmittel	Dortmund
Karstadt	Bekleidung	Essen
Karstadt	Bekleidung	Dortmund
Leffers	Bekleidung	Dortmund
Leffers	Bekleidung	Hannover

- Tabelle ist redundant:
 - pro Kette jede Branche mit jedem Standort kombiniert (nötig, da Branche und Standort unabhängige Eigenschaften)
- Trotzdem ist die Relation in BCNF (Warum?)

Brauchen weiteres Kriterium für Redundanz:

mehrwertige Abhängigkeiten

72

2.3.3 4NF (3)

Definition:

Die Attributmengengruppe $Y \in R$ ist *mehrwertig abhängig* von der Attributmengengruppe $X \in R$, symbolisch $X \twoheadrightarrow Y$, wenn für alle Tupel t_1, t_2 mit $t_1[X] = t_2[X]$

Tupel t_3, t_4 existieren mit

- $t_3[X] = t_4[X] = t_1[X] = t_2[X]$
- $t_3[Y] = t_1[Y]$ und $t_4[Y] = t_2[Y]$
- $t_3[Z] = t_2[Z]$ und $t_4[Z] = t_1[Z]$ wobei $Z = R \setminus (X \cup Y)$

Bemerkungen:

- die vier Tupel sind nicht notwendig verschieden
- wegen der Symmetrie in Definition folgt aus $X \twoheadrightarrow Y$ auch $X \twoheadrightarrow Z$. Man schreibt deshalb auch $X \twoheadrightarrow Y \mid Z$

73

2.3.3 4NF (5)

kette#	branche#	standort#
Karstadt	Lebensmittel	Essen
Karstadt	Lebensmittel	Dortmund
Karstadt	Bekleidung	Essen
Karstadt	Bekleidung	Dortmund
Leffers	Bekleidung	Dortmund
Leffers	Bekleidung	Hannover

Anschauliche Bedeutung von $X \twoheadrightarrow Y \mid Z$:

- Das Ergebnis des Statements

```
SELECT DISTINCT Y FROM R
WHERE X = '...' AND Z = '...';
```

ist für alle (vorhandenen) Werte von Z gleich
- Existieren zwei Tupel mit gleichem X aber unterschiedlichem Y , so müssen diese Y -Werte in getrennten Tupeln für jeden unterschiedlichen Wert von Z wiederholt werden

75

2.3.3 4NF (4)

Zu t_1, t_2 mit $t_1[X] = t_2[X]$ existieren t_3, t_4 mit

- $t_3[X] = t_4[X] = t_1[X] = t_2[X]$
- $t_3[Y] = t_1[Y]$ und $t_4[Y] = t_2[Y]$
- $t_3[Z] = t_2[Z]$ und $t_4[Z] = t_1[Z]$ wobei $Z = R \setminus (X \cup Y)$

Beispiel:

kette#	branche#	standort#
Karstadt	Lebensmittel	Essen
Karstadt	Lebensmittel	Dortmund
Karstadt	Bekleidung	Essen
Karstadt	Bekleidung	Dortmund
Leffers	Bekleidung	Dortmund
Leffers	Bekleidung	Hannover

← t_1
← t_4
← t_3
← t_2

- es gilt $\{kette\} \twoheadrightarrow \{standort\}$ und $\{kette\} \twoheadrightarrow \{branche\}$
- es gilt nicht $\{standort\} \twoheadrightarrow \{kette\}$

74

2.3.3 4NF (6)

Triviale mehrwertige Abhängigkeiten

- wenn Y Teilmenge von X ist oder $X \cup Y = R$, folgt automatisch $X \twoheadrightarrow Y$
- sagt also nichts Wesentliches aus: "triviale" Abhängigkeit

Zusammenhang zu funktionaler Abhängigkeit

- aus $X \rightarrow Y$ folgt $X \twoheadrightarrow Y$
- Umkehrung gilt nicht

76

2.3.3 4NF (7)

Theorem (Fagin 1977):

- Sei $R(A,B,C)$ ein Relationenschema, wobei A , B und C Attributmengen sind.
- Jeder Relatszustand $r(R)$ ist gleich dem Join seiner Projektionen über $\{A,B\}$ und $\{A,C\}$, genau dann wenn die Abhängigkeit $A \twoheadrightarrow B \mid C$ gilt.

Bemerkungen:

- Im Unterschied zu Heath's Theorem äquivalente Bedingung für "additiven Join" (Heath lieferte nur hinreichende Bedingung)
- Heath's Theorem folgt direkt aus diesem Theorem (Warum?)

77

2.3.3 4NF (9)

Beispiel:

<i>kette#</i>	<i>branche#</i>	<i>standort#</i>
Karstadt	Lebensmittel	Essen
Karstadt	Lebensmittel	Dortmund
Karstadt	Bekleidung	Essen
Karstadt	Bekleidung	Dortmund
Leffers	Bekleidung	Dortmund
Leffers	Bekleidung	Hannover

- Relation ist nicht in 4NF, denn es gilt $\{kette\} \twoheadrightarrow \{standort\}$, aber weder $standort$ noch $branche$ sind funktional abhängig von $kette$
- Beispiel ist typisch: die meisten Relationen mit nichttrivialen mehrwertigen Abhängigkeiten enthalten nur Schlüsselattribute

79

2.3.3 4NF (8)

Vierte Normalform (4NF)

- Wenn eine nichttriviale mehrwertige Abhängigkeit $A \twoheadrightarrow B$ gilt, dann sind alle Attribute der Relation *funktional abhängig* von A .

Anders ausgedrückt:

- alle nichttrivialen Abhängigkeiten (mehrwertig oder funktional) sind funktionale Abhängigkeiten von einem Superkey. Somit ist eine Relation in 4NF auch immer in BCNF.
- 4NF = BCNF und alle mehrwertigen Abhängigkeiten sind funktionale Abhängigkeiten von Schlüsseln.

78

2.3.3 4NF (10)

Umformung in 4NF:

<i>kette#</i>	<i>branche#</i>	<i>standort#</i>
Karstadt	Lebensmittel	Essen
Karstadt	Lebensmittel	Dortmund
Karstadt	Bekleidung	Essen
Karstadt	Bekleidung	Dortmund
Leffers	Bekleidung	Dortmund
Leffers	Bekleidung	Hannover

<i>kette#</i>	<i>branche#</i>	<i>kette#</i>	<i>standort#</i>
Karstadt	Lebensmittel	Karstadt	Essen
Karstadt	Bekleidung	Karstadt	Dortmund
Leffers	Bekleidung	Leffers	Dortmund
		Leffers	Hannover

- Zerlege Relation $R=(A,B,C)$ mit Abhängigkeit $A \twoheadrightarrow C$ per Projektion in zwei Relationen $R1=(A,B)$ und $R2=(A,C)$
- $R1$ und $R2$ sind in 4NF (Warum?)
- Aufgrund Fagin's Theorem ist diese Zerlegung additiv

80

2.3.3 4NF (11)

<i>kette#</i>	<i>branche</i>	<i>standort</i>
Karstadt	{Lebensmittel, Bekleidung}	{Essen, Dortmund}
Leffers	{Bekleidung}	{Dortmund, Hannover}

Mehrwertige Abhängigkeiten hätten früh im Design vermieden werden können bei Umformung in 1NF

- Faustregel: unabhängige mengenwertige Attribute
· in separate Relationen trennen!
- Theorie der 4NF gibt strenge Begründung
· für diese Faustregel

81

2.3.3 4NF (12)

Weitere Normalformen

- Project-Join Normalform oder 5NF
 - betrachtet "Join Dependencies", das sind Bedingungen bzgl. der projektiven Zerlegung über Attributmengen
 - Verletzungen treten in Praxis selten auf (pathologische Fälle?) und sind schwer zu entdecken
=> geringe praktische Bedeutung
- Domain-Key Normalform (DKNF)
 - erfasst beliebige Constraints
 - alle Constraints sollen aus Domain-Constraints und Key-Constraints folgen
 - impliziert 5NF
 - Bedingungen ungeklärt, wann DKNF möglich

82