7. Entwurfstheorie relationaler Datenbanken

- Wie sieht ein gutes konzeptionelles Schema der Datenbank aus?
- Wie kann die Güte eines Datenbankschemas beurteilt werden?

Beispiel:

Kunde*(<u>KName</u>, KAdr, Kto)*
Auftrag(<u>KName</u>, <u>Ware</u>, Menge)
LieferantW(<u>LName</u>, LAdr, <u>Ware</u>, Preis)

alternative Schemata:

KundenAdr(<u>KName</u>, KAdr)
KundenKto(<u>KName</u>, Kto)
Auftrag(<u>KName</u>, <u>Ware</u>, Menge)
Lieferant(<u>LName</u>, LAdr)
Angebot(<u>LName</u>, <u>Ware</u>, Preis)

Beispiel:

LieferantW (<u>LName</u>, LAdr, <u>Ware</u>, Preis)   hat Nachteile:

1. ***Redundanz:***
   für jede Ware wird die Adresse des Lieferanten gespeichert.
2. ***Änderungs-Anomalien*** *(mögliche Inkonsistenzen):*
   Man kann die Adresse eines Lieferanten in einem seiner Tupel ändern, in einem anderen jedoch unverändert lassen.
3. ***Einfüge-Anomalien:***
   Man kann keine Lieferantenadresse ohne eine Ware einfügen.
4. ***Entfernungs-Anomalien:***
   Beim Löschen der letzten Ware geht auch die Lieferantenadresse verloren.

Verbesserung (?):

Lieferant*(LName, LAdr)*

Angebot*(LName, Ware, Preis)*

Vorteile:

- ❑ keine Redundanz
- ❑ keine Anomalien

Aber:

- ❑ um zu einer Ware die Lieferantenadresse zu finden ist ein (teurer) Join nötig.

Verbesserung (?):

Lieferant*(LName, LAdr,Ware)*

Angebot*(Ware, Preis)*

vorher:

LieferantW	*LName*	*LAdr*	*Ware*	*Preis*
	Michl	*München*	*Milch*	*1,10*
	Kohl	*Frankfurt*	*Milch*	*1,30*
	Keller	*Stuttgart*	*Mehl*	*2,30*
	…	…	…	…

nachher:

Lieferant	*LName*	*LAdr*	*Ware*
	Michl	*München*	*Milch*
	Kohl	*Frankfurt*	*Milch*
	Keller	*Stuttgart*	*Mehl*

Angebot	*Ware*	*Preis*
	Milch	*1,10*
	Milch	*1,30*
	Mehl	*2,30*

Zusammenfügen durch Join ergibt:

LieferantW	*LName*	*LAdr*	*Ware*	*Preis*
	Michl	*München*	*Milch*	*1,10*
	Kohl	*Frankfurt*	*Milch*	*1,30*
	Michl	*München*	*Milch*	*1,30*
	Kohl	*Frankfurt*	*Milch*	*1,10*

Entwurfsziele

- Vermeidung von Redundanzen und Anomalien
- Vermeidung der Probleme bei der Informationsrepräsentation
- Vermeidung des Informationsverlusts
- evtl. Einbeziehung von Effizienzüberlegungen

Grundlage:

- DB-Schema + “funktionale Abhängigkeiten” (Definition folgt)

Vorgehen:

- Zerlegen des gegebenen Datenbank-Schemas in ein äquivalentes Schema ohne Redundanz und Anomalien (“Normalisierung”).

7.1 Funktionale Abhängigkeiten

- Integritätsbedingungen:
  - Bedingungen an die zugelassenen Ausprägungen des Datenbankschemas
- Funktionale Abhängigkeiten (FDs) = spezielle Integritätsbedingungen
  - FD steht für “functional dependency”
- im folgenden: Relationenschema RS = endliche Menge verschiedener Attribute

Def. (funktionale Abhängigkeit):

SeienAundBAttributmengendesRelationenschemasRS($A, B \subseteq RS$).BistvonAfunktional abhängig oder A bestimmt B funktional, geschrieben $A \rightarrow B$, gdw. zu jedem Wert in A genau ein Wert in B gehört:

$$A \rightarrow B \Leftrightarrow \forall t_1, t_2 \in r(RS) : t_1[A] = t_2[A] \Rightarrow t_1[B] = t_2[B]$$

für alle real möglichen Relationen r(RS).

Dabei bezeichnet r(RS) eine Relation r über dem Schema RS.

- Beachte: Funktionale Abhängigkeit ist abhängig von der Semantik des Schemas, nicht von der Instanz einer Relation!

Beispiel:

- **Lieferant**(LName, LAdr, Ware, Preis)
- Funktionale Abhängigkeiten:
  1. {LName} $\rightarrow$ {LAdr} (ein Lieferantenname bestimmt eindeutig seine Adresse)
  2. {LName, Ware} $\rightarrow$ {Preis}
     (der Schlüssel {LName, Ware} bestimmt eindeutig den Preis)
  3. {LName} $\rightarrow$ {LName} (trivial)
  4. {LName, Ware} $\rightarrow$ {Ware} (trivial)
  5. {LName, Ware} $\rightarrow$ {LAdr} (partiell)

Eine Abhängigkeit $A \rightarrow B$ ist *trivial*, wenn gilt: $B \subseteq A$

Def. (voll, partiell, transitiv):

Eine Abhängigkeit $X \rightarrow Y$ heißt *voll*, wenn es keine echte Teilmenge $Z \subset X$ gibt, so daß gilt: $Z \rightarrow Y$. Gibt es eine solche Teilmenge, dann heißt $X \rightarrow Y$ *partielle* Abhängigkeit.

Seien X und Y Attributsmengen aus RS ($X, Y \subset RS$) und gelte $X \rightarrow Y$ und $X \not\leftarrow Y$. Sei $A \in RS$ ein Attribut mit $A \notin X, Y$ und gelte $Y \rightarrow A$. Dann ist A *transitiv abhängig* von X: $X \rightarrow A$

Berechnung von FDs

- Wunsch:
  Zu einer gegebenen Menge F von FDs sollen alle gültigen FDs berechnet werden.
- F^+ ist die Menge aller FDs, die aus den funktionalen Abhängigkeiten in F ableitbar sind.
  F^+ wird auch als **Hülle** von F bezeichnet.
- Seien RS ein Relationenschema, F eine Menge von FDs und $A, B, C \subseteq RS$.

  Zur Berechnung von F^+ werden folgende Regeln genutzt (**Armstrong Axiome**):
  - Reflexivität: Sei $B \subseteq A$. Dann gilt stets $A \rightarrow B$ (Sonderfall: $A \rightarrow A$)
  - Verstärkung: Falls $A \rightarrow B$ gilt, dann gilt auch $A \cup C \rightarrow B \cup C$.
  - Transitivität: Falls $A \rightarrow B$ und $B \rightarrow C$, dann gilt auch $A \rightarrow C$
- Es kann gezeigt werden, daß die Regeln korrekt und vollständig sind.
  - abgeleitete Regeln sind für alle Relationen des Schemas gültig
  - alle gültigen FDs in F^+ sind mit Hilfe dieser Regeln auch herleitbar
- trotz dieser Eigenschaften der Amstrong-Axiome ist es komfortabler noch folgende Regeln zu benutzen:
  - Vereinigungsregel:
    Falls $A \rightarrow B$ und $A \rightarrow C$ gilt, dann gilt auch $A \rightarrow B \cup C$

- Dekompositionsregel:
  Falls $A \to B \cup C$ gilt, dann gilt auch $A \to B$ und $A \to C$
- Pseudotransivität:
  Falls $A \to B$ und $B \cup C \to D$ gilt, dann gilt auch $A \cup C \to D$

❑ Beispiel:

- Lieferanten-Relationen ***Lieferant****(LName, LAdr, Ware, Preis)*
- FDs 1-4 seien gültig (siehe oben)
- zu zeigen: FD 5 {LName, Ware} $\to$ {LAdr} ist ebenfalls erfüllt.

  Es gilt {LName} $\to$ {LAdr}.

  Auf Grund des 2-ten Amstrong-Axiom gilt auch:
  {LName, Ware} $\to$ {LAdr, Ware}

  Wegen der Dekompositionsregel ergibt sich damit FD 5.

Membership-Problem

- Fragestellung:

  Sei F eine Menge von FDs und $A \rightarrow B$ eine funktionale Abhängigkeit. Gilt $A \rightarrow B \in F^+$?
- Explizite Berechnung von F^+ ist zu aufwendig
- stattdessen: Berechnung der Hülle A^+ der Attributmenge A bzgl. der Menge F
  - A^+ besteht aus allen Attributen, die von A funktional bestimmt werden
  - Falls $B \subseteq A^+$ gilt, dann gilt auch $A \rightarrow B \in F^+$.

```
Algorithmus Hülle(F, A)
//   Eigabe:   a) eine Menge F mit funktionalen Abhängigkeiten
//             b) ein Attribut A
//   Ausgabe: A+
Erg = A;
WHILE (Änderungen an Erg) DO
     FOREACH FD B → C ∈ F  DO
          IF  B ⊆ Erg  THEN  Erg = Erg ∪ C ;
RETURN Erg;
```

Kanonische Überdeckung

Definition:

Zwei Mengen F und G von FDs eines Relationenschemas R sind äquivalent, falls $F^+ = G^+$ gilt.

- Wunsch: Berechne eine möglichst kleine Menge, die zu F äquivalent ist.
  - wenig Aufwand beim Testen, ob ein neues Tupel eine FD verletzt.
- F_c wird als kanonische Überdeckung von F bezeichnet, falls folgende Bedingungen erfüllt sind:
  - $F_c^+ = F^+$
  - Für alle FDs $A \to B$ in F_c gibt es keine "überflüssigen" Attribute, d. h.

    für alle Attribute C aus A gilt $(F_c - \{A \to B\} \cup \{(A - \{C\}) \to B\})^+ \neq F^+$.

    für alle Attribute D aus B gilt $(F_c - \{A \to B\} \cup \{A \to (B - \{D\})\})^+ \neq F^+$.
  - jede linke Seite der FDs in F_c kommt nur einmal vor, d. h.

    falls $A \to B$ und $A \to C$, dann wird in F_c nur die FD $A \to B \cup C$ verwendet.

Berechnung der kanonischen Überdeckung

1. Führe für jede FD $A \rightarrow B \in F$ die *Linksreduktion* durch:

   Überprüfe für alle $X \in A$, ob das Attribut X überflüssig ist, d.h. ob

   $$B \subseteq \text{Hülle}(F, A - \{X\})$$

   gilt. Ist dies der Fall, ersetze $A \rightarrow B$ durch $A - \{X\} \rightarrow B$.

2. Führe für jede verbliebene FD $A \rightarrow B \in F$ die Rechtsreduktion durch, d. h.

   Überprüfe für alle $Y \in B$, ob das Attribut Y überflüssig ist, d. h.

   $$Y \in \text{Hülle}(F - (A \rightarrow B) \cup (A \rightarrow B - \{Y\}), A)$$

   gilt. Ist dies der Fall, dann wird $A \rightarrow B$ durch $A \rightarrow B - \{Y\}$ ersetzt.

3. Entferne die FDs der Form $A \rightarrow \varnothing$ (die im 2-ten Schritt entstanden sind)
4. Ersetze alle FDs der Form $A \rightarrow B_1$, $A \rightarrow B_2$, …, $A \rightarrow B_k$ durch

   $$A \rightarrow B_1 \cup B_2 \cup \ldots \cup B_k$$

Beispiel:

- Menge F = $\{A \to B,\ B \to C,\ A \cup B \to C\ \}$
- Schritt 1:
- Schritt 2:
- Schritt 3:
- Schritt 4:

Zerlegung eines Relationenschemas

- um Anomalien zu beseitigen, wird das Relationenschema R in n Relationenschemata R_1, …, R_n zerlegt.
- Folgende Eigenschaften sollen erfüllt werden:
  - kein Informationsverlust, d.h. eine beliebige Instanz r(R) muß aus den Instanzen $r[R_1]$, …, $r[R_n]$ wieder konstruierbar sein.
  - alle FDs, die für das Schema R gelten, sollen für R_1, …, R_n effizient überprüfbar bleiben.
- Informationsverlust (n=2):

  Eine Zerlegung des Schemas R in R_1 und R_2 hat keinen Informationsverlust, wenn

  für alle Relationen r(R) folgendes gilt: $r = \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r)$.

  Es gilt:

  - Sei R ein RS und F_R die Menge der FDs. Eine Zerlegung von R in R_1 und R_2 hat keinen Informationsverlust, falls einer der folgenden Bedingungen gilt:

$$(R_1 \cap R_2 \rightarrow R_1) \in F_R^+ \quad \text{oder} \quad (R_1 \cap R_2 \rightarrow R_2) \in F_R^+$$

Beispiel:

- Die Zerlegung der Relation R(Lname, LAdr, Ware, Preis) in die Relationen
  - Lieferant(LName, LAdr, Ware)
  - Angebot(Ware, Preis)

  ist nicht verlustlos, d.h. es gilt i.a.

$$R \neq \text{Lieferant} \bowtie \text{Angebot}$$

Grund:

- Ware bestimmt nicht funktional den Preis
- Ware bestimmt nicht funktional den LName

Bewahrung funktionaler Abhängigkeiten

- Wunsch: alle FDs, die für das Schema R gelten, sollen *lokal* auf den zerlegten Schemata $R_1,\ldots,R_n$ überprüfbar sein (Effizienz !).
- formal kann dies wie folgt ausgedrückt werden:

$$F_R^+ = (F_{R_1} \cup \ldots \cup F_{R_n})^+$$

$R_1,\ldots,R_n$ wird dann auch als hüllentreue Zerlegung bezeichnet.

Beispiel:

- Sei das Schema PV(Straße, Ort, BLand, PLZ) gegeben Es sollen folgende Bedingungen gelten:
  - Orte werden durch “Ort” und “BLand” eindeutig charakterisiert.
  - innerhalb einer Straße ändert sich “PLZ” nicht.
  - PLZ-Gebiete gehen nicht über Ortsgrenzen, Orte nicht über Bundeslandgrenzen.
- FDs: {PLZ} --> {Ort, BLand} und {Straße, Ort, BLand} --> {PLZ}
- Welche Eigenschaften besitzt die Zerlegung {PLZ, Straße} und {PLZ, Ort, BLand}?

7.2 Die ersten drei Normalformen

- Durch Normalformen wird definiert, was unter einem guten Datenbankdesign zu verstehen ist.

Def. (1. Normalform):

Ein Relationenschema ist genau dann in der *1. Normalform (NF)*, wenn die Wertebereiche aller Attribute nur atomare Werte besitzen, die nicht weiter zerlegbar sind.

Diese Eigenschaft ist integraler Bestandteil des relationalen Modells und wird deshalb bei den folgenden Betrachtungen stets vorausgesetzt.

NF^2-Relationen (NF^2 = **N**on**F**irst**N**ormal**F**orm)

- Die 1NF ist bei der Modellierung von Daten zu inflexibel, so daß derzeit sogar die relationalen DBMS, NF^2-Relationen unterstützen (siehe auch Objektorientierung).
- *Beispiel:*NF^2 für Literatur (BÜCHER, AUTOR, STICHWORT)

BÜCHER	*AUTOR*	*STICHWORT*
B_1	*Boyce*	*Normalisierung*
		Abhängigkeit
B_1	*Codd*	*Normalisierung*
		Abhängigkeit

Def. (atomar): Eine nicht-triviale funktionale Abhängigkeit der Form $X \to \{A\}$ heißt *atomar*. Wir verwenden im folgenden die Schreibweise $X \to A$.

Def. (2. Normalform):

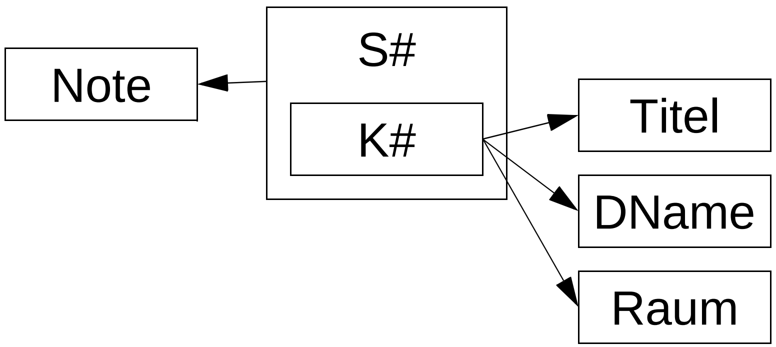
Ein Relationenschema ist genau dann in *2. Normalform* (2NF), wenn für alle atomaren funktionalen Abhängigkeiten $X \to A$ gilt: Wenn A nicht Teil eines Schlüssels und X ein Schlüssel ist, dann gibt es keine atomare funktionale Abhängigkeit $Y \to A$ mit $Y \subset X$.

Beispiel:

- Leistungsnachweis (<u>S#, K#</u>, Titel, DName, Raum#, Note)
- Tupel *(s, k, t, d, r, n)* bedeutet: Student *s* hat die Note *n* erzielt im Kurs mit Nummer *k*, der den Titel *t* trug und im Raum mit Nummer *r* vom Dozenten *d* abgehalten wurde.

Folgende Abhängigkeiten bestehen:

1. {S#, K#} $\to$ {Note}
2. {K#} $\to$ {Titel}
3. {K#} $\to$ {DName}
4. {DName} $\to$ {Raum#}
5. {K#} $\to$ {Raum#}



- Die Relation ‘Leistungsnachweis’ ist nicht in 2NF. Folgende Anomalien können auftreten:
  - Informationen über einen neuen Kurs sind nur dann verfügbar, wenn bereits ein Student für diesen Kurs eingetragen ist.
  - Dozent ist nur dann in der Datenbank, wenn er/sie einen Kurs hält
  - Namensänderung eines Kurses ist sehr aufwendig (1 Update pro Student)
  - Falls alle Studenten den Kurs 27 verlassen und die dazugehörigen Tupel gelöscht werden, verschwinden alle Informationen über den Kurs.

- Transformation in 2NF behebt diese Anomalien:
  - Aufspalten der Relation ‘Leistungsnachweis’ in folgende zwei Relationen:
  - Schema in 2NF:

    Leistungsnachweis (<u>S#,</u> <u>K#</u>, Note)

    Kurs (<u>K#</u>, Titel, DName, Raum#)

Bemerkung:

- Verletzung der 2NF nur bei zusammenengesetzten Schlüsseln
- Beseitigung von partiellen funktionalen Abhängigkeiten zwischen einem Schlüssel und einem Nicht-Schlüssel-Attribut
- geringe Bedeutung der 2NF (siehe weitere Normalformen)

Def. (3. Normalform):

Ein Relationenschema ist genau in *3. Normalform* (3NF), wenn für alle atomaren funktionalen Abhängigkeiten $X \rightarrow A$ gilt: Wenn A nicht Bestanteil eines Schlüssels ist, muß X einen Schlüssel enthalten.

- ❑ Die Relation ‘Kurs’ ist nicht in 3NF, da die Abhängigkeit DName → Raum# besteht und DName weder Schlüssel noch Raum# Bestandteil eines Schlüssel ist.

  Folgende Anomalien können auftreten:

  - Informationen über Dozenten und Raum sind ohne Zuordnung eines Kurses nicht verfügbar
  - Ändern der Raumnr. eines Dozenten bedingt die Änderung für jeden Kurs
  - Falls ein Dozent keinen Kurs gibt, werden alle Informationen über den Dozent und seinen Raum aus der Datenbank gelöscht.

  Schema in 3NF:

  Leistungsnachweis (<u>S#,</u> <u>K#</u>, Note)

  Kurs (<u>K#</u>, Titel, DName)

  Dozent (<u>DName</u>, Raum#)

- ❑ Hinweis:Die 3. Normalform beseitigt Abhängigkeiten von Nicht-Schlüssel-Attributen.

7.3 Synthesealgorithmus

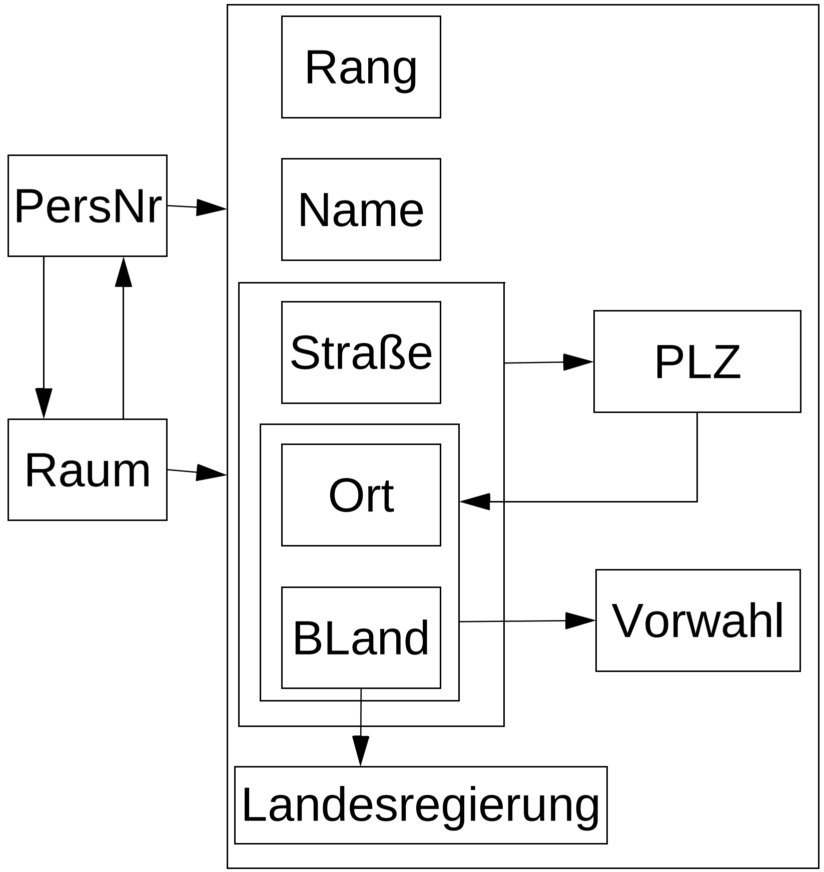
- Ziel ist die Zerlegung eines Relationenschema R mit funktionalen Abhängigkeiten F in Relationenschemata $R_1,\ldots,R_n$, so daß folgende Bedingungen erfüllt sind:
  - kein Informationsverlust,
  - Bewahrung der funktionalen Abhängigkeiten,
  - Relationenschemata $R_1,\ldots,R_n$ erfüllen die dritte Normalform.
- folgender Algorithmus generiert eine solche Zerlegung:
  1. Bestimme die kanonische Überdeckung F_c der Menge F.
  2. Führe für jede FD $A \to B \in F_c$ folgende Anweisungen aus:
     - Erzeuge ein Relationenschema R_A und ordne R_A die FDs $F_A =$ $\{C \to D \in F_c \mid C \cup D \subseteq R_A\}$
  3. Falls alle der in Schritt 2 erzeugten Schemata keinen Kandidatenschlüssel des ursprünglichen Schemas R enthalten, so erzeuge zusätzlich eine Relation mit dem Schema $R_K = K$ und $F_K = \emptyset$, wobei K ein Kandidatenschlüssel von R ist.
  4. Eliminiere die Schemata R, die in einem anderen Schema enthalten sind.

Beispiel:

- Relationenschema ProfessorenAdr:
  PersNr, Raum, Rang, Name, Straße, Ort, BLand, Landesreg., PLZ, Vorwahl

Annahmen:

- Ort ist der Erstwohnsitz des Profs
- Landesregierung ist die Partei der Ministerpräsidentin
- Ortsnamen sind eindeutig innerhalb der Bundesländer
- PLZ ändert sich nicht innerhalb einer Straße
- Städte und Straßen liegen vollständig in Bundesländern
- ein Prof hat genau ein Büro (und er teilt es nicht)



1-ter Schritt: Berechnung einer kanonischen Überdeckung

- FD_1 {PersNr} → {Raum, Name, Rang, Straße, Ort, BLand}
- FD_2 {Raum} → {PersNr}
- FD_3 {Sraße, Ort, Bland} → {PLZ}
- FD_4 {Ort, BLand} → {Vorwahl}
- FD_5 {BLand} → {Landesregierung}
- FD_6 {PLZ} → {Ort, BLand}

2-ter Schritt:

- aus FD_1 ergibt sich:
  - {PersNr, Name, Rang, Raum, Straße, Ort, Bland}
  - FD_1 und FD_2 werden zugeordnet
- aus FD_3 ergibt sich:
  - {Straße, Ort, Bland, PLZ}
  - FD_3 und FD_6 werden zugeordnet

- aus FD_4 ergib sich:
  - {Ort, BLand, Vorwahl}
  - FD_4 wird zugeordnet
- aus FD_5 ergibt sich:
  - {BLand, Landesregierung}
  - FD_5 wird zugeordnet

Schritt 3:

- {PersNr} ist Kandidatenschlüssel des ursprünglichen Schemas und befindet sich in einem Relationenschema.

Schritt 4:

- ist schon im Schritt 2 passiert

7.4 Boyce-Codd Normalform

Def. (Boyce/Codd-NF):

Ein Relationenschema ist genau dann in *Boyce/Codd-Normalform* (BCNF), wenn für alle atomaren funktionalen Abhängigkeiten $X \rightarrow A$ gilt: X muß einen Schlüssel enthalten.

- Die Boyce/Codd-Normalform beseitigt Abhängigkeiten unter Attributen, die *Bestandteil* eines Schlüssels sind.

Beispiel: Autoverzeichnis (<u>Hersteller</u>, HerstellerNr, <u>ModellNr</u>)

- Folgende Abhängigkeiten bestehen:
  - Hersteller $\rightarrow$ HerstellerNr(1:1-Beziehung zwischen Hersteller und HerstellerNr)
  - HerstellerNr $\rightarrow$ Hersteller(s.o.)
- Beispiel ist in 3NF (alle Attribute sind Schlüsselkandidaten), aber nicht in BCNF
- Folgende Anomalien können auftreten:
  - Einfügen des selben Herstellers mit verschiedenen HerstellerNr. ist möglich
  - 1:1-Beziehung von Hersteller und HerstellerNr. ist an die ModellNr gekoppelt

Eigenschaften eines Schema in BCNF

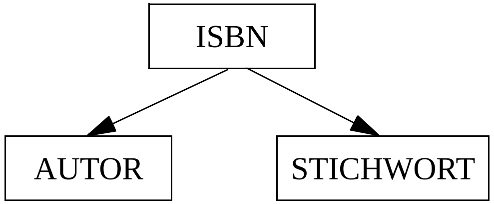
- Sei RS ein Relationenschema und FD eine Menge funktionaler Abhängigkeiten. Dann gilt:
  Es gibt eine Zerlegung von RS in $RS_1,\ldots,RS_n$, so daß
  - die Zerlegung verlustlos ist und,
  - RS_i die Boyce-Codd Normalform erfüllen.
- schlechte Nachrichten:
  - es kann nicht immer eine abhängigkeitsbewahrende Zerlegung gefunden werden.

Praktische Vorgehensweise

- Zerlegung eines Schema von 3 NF in BCNF.
- Falls diese Zerlegung abhängigkeitsbewahrend ist, wird dieses Schema verwendet. Ansonsten benutzt man das ursprüngliche Schema in 3NF.

7.5 Mehrwertige Abhängigkeiten

- sind eine Verallgemeinerung funktionaler Abhängigkeiten
- Beispiel: Relationenschema Buch mit {ISBN, Autor, Stichwort}



  - ein Buch kann mehrere Autoren besitzen
  - mehrere Stichwörter verweisen auf das Buch
  - AUTOR bzw. STICHWORT sind mehrwertig abhängig von BÜCHER

Def. (mehrwertig abhängig):

Sei RS ein Relationenschema und $A, B, C \subseteq RS$ mit $RS = A \cup B \cup C$. Dann ist C *mehrwertig abhängig* von A, $A \twoheadrightarrow C$, wenn in jeder Relation des Schemas RS gilt: für jedes Paar von Tupel t1 und t2 mit t1[A] = t2[A] existieren zwei Tupel t3 und t4 mit t3[A] = t4[A] = t1[A] und mit folgenden Eigenschaften:

t3[B] = t1[B]	t3[C] = t2[C]
t4[B] = t2[B]	t4[C] = t1[C]

Beispiel:

- Buch (ISBN, AUTOR, STICHWORT)

ISBN	*AUTOR*	*STICHWORT*
I-1	*Boyce*	*Normalisierung*
I-1	*Boyce*	*Abhängigkeit*
I-1	*Codd*	*Normalisierung*
I-1	*Codd*	*Abhängigkeit*

- Buch besitzt nicht nur zwei Autoren, sondern es existieren auch mindestens zwei Stichpunkte
- Zerlegen des Relationenschema Buch in zwei Relationenschemata RS_1 und RS_2
  - RS_1 = {ISBN, Autor}
  - RS_2 = {ISBN, Stichwort}
  - Es gilt sogar: Die Zerlegung ist verlustfrei!

Sei R ein RS und F_R die Menge der MVDs. Eine Zerlegung von R in R_1 und R_2 hat keinen Informationsverlust, genau dann falls einer der folgenden Bedingungen gilt:

$$(R_1 \cap R_2 \twoheadrightarrow R_1) \in F_R^+ \text{ oder } (R_1 \cap R_2 \twoheadrightarrow R_2) \in F_R^+$$

Vierte Normalform

- ist eine Verstärkung der Boyce-Codd Normalform
- Vermeidung der durch mehrwertige Abhängigkeiten verursachten Redundanz

Sei RS ein Relationenschema und $A, C \subseteq RS$. Eine mehrwertige Abhängigkeit $A \twoheadrightarrow C$ ist trivial, falls eine der folgenden Bedingungen gilt:

1. $C \subseteq A$
2. $C = RS - A$

Definition (4NF):

Sei RS ein Relationenschema und M eine Menge mehrwertiger Abhängigkeiten. RS ist genau dann in *vierter Normalform* (4NF), wenn für jede nicht-triviale mehrwertige Abhängigkeit $A \twoheadrightarrow C \in M$ folgende Bedingung gilt:

A enthält einen Schlüssel von R

7.6 Grenzen der Normalisierung

- durch Normalformen bedingt ist Information über mehrere Relationen mit “wenigen” Attributen verteilt
  - Vorteil: Anomalien sind beseitigt, geringe Redundanz
  - Nachteil: “ineffiziente” Anfragebearbeitung

Beispiel:

- Gesucht sind Namen und Adressen aller Lieferanten, die ‘Mehl’ liefern.
  - Schema: ***Liefert****(<u>LName</u>, LAdr, <u>Ware</u>, Preis)*

    $\pi_{\text{LName, LAdr}}(\sigma_{\text{Ware = Mehl}}(\text{Liefert}))$

  - alternatives Schema:

    Lieferant(<u>LName</u>, LAdr)(Schema in 2NF)

    Angebot (<u>L Name</u>, <u>Ware</u>, Preis)

    $\pi_{\text{LName, LAdr}}(\sigma_{\text{Ware = Mehl}}(\text{Lieferant} \bowtie \text{Angebot}))$

7.6.1 Nicht-Standard Anwendungen

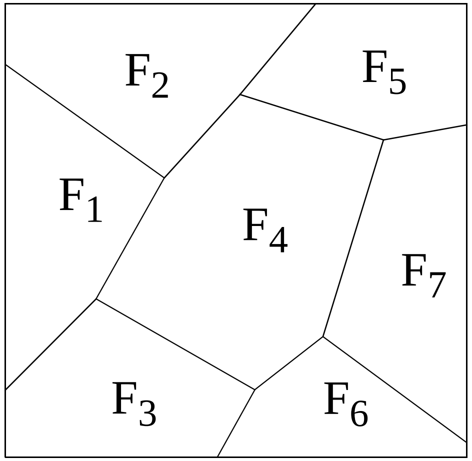
Bisher:

- Beispiele aus dem betriebswirtschaftlich/administrativen Bereich Handel, Banken, Versicherungen, etc., d.h. Standard-Datenbanken und -systeme
- kleine Datenobjekte exakt festgelegter Struktur
- meist einfache Integritätsbedingungen
- viele, kurze Transaktionen auf den Datenbanken (z.B. Buchungen)

Jetzt:

- Betrachtung sog. “Nicht-Standard-Datenbanksysteme” (NDBS)
  - CAD / CAM / CIM
  - Geographie und Kartographie
  - Medizin und Biologie
- komplexe, unterschiedlich strukturierte geometrische Objekte
- oft komplexe Integritätsbedingungen (z.B. aus der Geographie)
- häufig sehr lange Operationen (Transaktionen) auf wenigen Objekten (z.B. CAD)

Beispiel aus der Geographie



Parzellen

FNr	KNr
F_1	K_1
F_1	K_2
F_1	K_3
F_1	K_4
F_4	K_2
F_4	K_5
F_4	K_6
F_4	K_7
F_4	K_8
F_4	K_9
F_7	K_7
F_7	K_{10}
F_7	K_{11}
F_7	K_{12}

Kanten

KNr	PNr_1	PNr_2
K_1	P_1	P_2
K_2	P_2	P_3
K_3	P_3	P_4
K_4	P_4	P_1
K_5	P_2	P_5
K_6	P_5	P_6
K_7	P_6	P_7
K_8	P_7	P_8
K_9	P_8	P_3
K_{10}	P_6	P_9
K_{11}	P_9	P_{10}
K_{12}	P_{10}	P_7

Punkte

PNr	X-Koord.	Y-Koord
P_1	X_{P1}	Y_{P1}
P_2	X_{P2}	Y_{P2}
P_3	X_{P3}	Y_{P3}
P_4	X_{P4}	Y_{P4}
P_5	X_{P5}	Y_{P5}
P_6	X_{P6}	Y_{P6}
P_7	X_{P7}	Y_{P7}
P_8	X_{P8}	Y_{P8}
P_9	X_{P9}	Y_{P9}
P_{10}	X_{P10}	Y_{P10}

- redundanzfreie Repräsentation der Parzellen erfordert die Verteilung der Informationen auf drei Relationen: ‘Parzellen’, ‘Kanten’ und ‘Punkte’.
- Anfragen auf den Parzellen müssen die erforderlichen Informationen zunächst zusammengesetzten:
  - *Beispiel:* Gesucht sind alle Eckpunkte der Parzelle mit Flurnr. 2.

<u>select</u>	Punkte.PNr, X-Koord, Y-Koord		
<u>from</u>	Parzellen, Kanten, Punkte		
<u>where</u>	FNr = “2”		<u>and</u>
	Parzellen.KNr = Kanten.KN	r	<u>and</u>
	(Kanten.PNr$_1$ = Punkte.PNr	<u>or</u>	Kanten.PNr$_2$ = Punkte.PNr)

- einfache Anfrage erfordert mehrere Joins
- Ursache: Datenmodellierung und Normalisierung der Relationen
- bessere Datenmodellierung für komplex (speziell geometrische) Objekte