Wiederholung: Normalformen

Redundanzen im DB-Schema erzeugen Anomalien (Änderungs-, Einfüge-, Entfernungs-Anomalien)

Ziel: Vermeidung von Redundanzen und Anomalien $\Rightarrow$ schrittweises Zerlegen des Schemas in ein äquivalentes Schema ohne Redundanzen und Anomalien = **Normalisierung**

(I) **Funktionale Abhängigkeiten**
Seien A, B Attributmengen des Relationenschemas R, d.h. $A, B \subseteq R$

B ist von A **funktional abhängig** (oder A bestimmt B funktional), d.h. $A \rightarrow B \Leftrightarrow$
für alle mgl. Ausprägungen von R gilt: Zu jedem Wert in A existiert genau ein Wert in B

Formal:
$A \rightarrow B \Leftrightarrow \forall r_1, r_2 \in R : r_1.A = r_2.A \Rightarrow r_1.B = r_2.B$

Bsp.: Matrikelnummer $\rightarrow$ Name

- **Triviale funktionale Abhängigkeit**: $A \rightarrow B$, falls $B \subseteq A$
- **Volle funktionale Abhängigkeit**: $A \rightarrow B$, falls keine echte Teilmenge $A' \subset A$ ex. mit $A' \rightarrow B$.
- Existiert eine solche Teilmenge, dann heißt $A \rightarrow B$ **partielle funktionale Abhängigkeit**.
- **Transitive funktionale Abhängigkeit** $A \rightarrow C$, falls gilt $A \rightarrow B$, $B \not\rightarrow A$ und $B \rightarrow C$ für $C \in R, C \notin A, B$

(II) **Schlüssel**

- Teilmenge S der Attribute eines Relationenschemas R heißt **Schlüssel**, falls gilt
  (1) Eindeutigkeit: Keine Ausprägung von R kann zwei verschiedene Tupel enthalten, die sich in allen Attributen von S gleichen
  (2) Minimalität: Keine echte Teilmenge von S erfüllt bereits Bedingung (1)
- Ein Attribut heißt **prim**, falls es Teil eines Schlüsselkandidaten ist

(III) **Normalformen**
Ziel: schrittweise Beseitigung funktionaler Abhängigkeiten (außer vom gesamten Schlüssel)

- **1. Normalform**
  - Alle Attribute enthalten **atomare** Werte (String, Integer, ...), d.h. keine Tupel, Listen, ...
  - In relationalen DB sind nicht-atomare Werte nicht möglich $\Rightarrow$ relationale DB immer in 1NF
- **2. Normalform**
  - Für jedes Attribut A gilt:
    - A ist **prim** oder
    - A ist **voll funktional abhängig von jedem Schlüsselkandidaten**
  - Beseitigt partielle funktionale Abhängigkeiten nicht-primer Attribute vom Schlüssel
  - 2NF kann nur verletzt werden, falls Schlüsselkandidat zusammengesetzt ist
  - Transformation in 2NF:
    - Erstelle eine neue Relation für jeden partiellen Schlüssel mit seinen abhängigen Attributen.
    - Attribute, die voll funktional vom Schlüssel abhängig sind, bleiben in der ursprünglichen Relation

- **3. Normalform**
  - Für alle nicht-trivialen funktionalen Abhängigkeiten $A \rightarrow B$ gilt:
    - A **enthält Schlüsselkandidaten** oder
    - B **ist prim**
  - Beseitigt funktionale Abhängigkeiten nicht-primer Attribute untereinander (= transitive Abhängigkeiten)
  - 3NF impliziert 2NF
  - Transformation in 3NF:
    - Erstelle eine neue Relation für alle Nicht-Schlüssel-Attribute und deren funktionalen Abhängigkeiten
    - Attribute, die voll funktional vom ursprünglichen Schlüssel abhängig und nicht abhängig von Nicht-Schlüssel-Attributen sind, bleiben in der ursprünglichen Relation
- **Boyce-Codd Normalform**
  - Für alle nicht-trivialen funktionalen Abhängigkeiten $A \rightarrow B$ gilt: A **enthält Schlüsselkandidaten**
  - Beseitigt funktionale Abhängigkeiten unter Attributen, die prim sind, aber nicht vollständig einen Schlüssel bilden
  - BCNF impliziert 3NF
  - Man kann nicht immer eine BCNF-Zerlegung finden, die Abhängigkeiten bewahrt

Wiederholung: Synthesealgorithmus

Zerlegung von Relationen
Zerlegung von R in $R_1, \ldots R_n$ ist

- **verlustlos**, falls gilt:
  Jede mögliche Ausprägung r von R lässt sich durch den natürlichen Join der Ausprägungen $r_1, \ldots, r_n$ konstruieren: $r = r_1 \bowtie \ldots \bowtie r_n$
- **abhängigkeitserhaltend**, falls gilt:
  Alle funktionalen Abhängigkeiten F auf R bleiben in den lokalen funktionalen Abhängigkeiten F_i bewahrt: $F = F_1 \cup \ldots \cup F_n$

Synthesealgorithmus für 3NF
Ermittelt eine verlustlose, abhängigkeitserhaltende Zerlegung von R in dritter Normalform

Gegeben: Relationenschema R mit funktionalen Abhängigkeiten F

(1) **Bestimmung der kanonischen Überdeckung F_C zu F**

- Linksreduktion (= Entfernung partieller Abhängigkeiten)
  Für jedes $\alpha \subseteq A$ ersetze $A \rightarrow B$ durch $A - \alpha \rightarrow B$, falls B schon durch $A - \alpha$ determiniert ist.
- Rechtsreduktion (= Entfernung transitiver Abhängigkeiten)
  Für jedes $\beta \subseteq B$ ersetze $A \rightarrow B$ durch $A \rightarrow B - \beta$, falls β transitiv durch A bestimmt wird.
- Entfernung von rechtsleeren Abhängigkeiten $A \rightarrow \emptyset$
- Zusammenfassen von Abhängigkeiten mit gleichen linken Seiten: $A \rightarrow B_1, \ldots, A \rightarrow B_m$ zu $A \rightarrow B_1 \cup \ldots \cup B_n$

(2) **Erzeugung eines neuen Relationenschemas aus F_C**
Für jede fkt. Abhängigkeit $A \rightarrow B \in F_C$

- Erzeuge Relationenschema $R_A = A \cup B$
- Ordne R_A die fkt. Abhängigkeiten $F_A = \{A' \rightarrow B' \mid A' \cup B' \in R_A\}$ zu

(3) **Rekonstruktion eines Schlüsselkandidaten**

- Falls ein Schema R_A aus (2) Schlüsselkandidaten von R enthält: weiter mit (4)
- Sonst: Wähle Schlüsselkandidaten $\kappa \in R$ und erzeuge das zusätzliche Schema $R_\kappa = \kappa$ mit $F_\kappa = \emptyset$

(4) **Elimination überflüssiger Relationen**
Eliminiere diejenigen Schemata R_A, die in einem anderen Schema R'_A enthalten sind, d.h. $R_A \subseteq R_{A'}$