TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

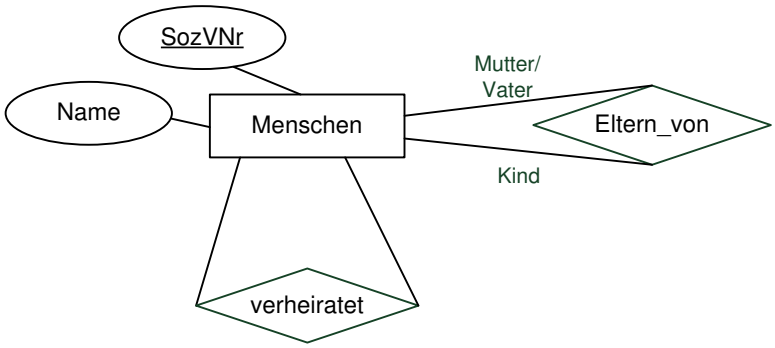
Übung zur Vorlesung *Grundlagen: Datenbanken* im WS15/16

Harald Lang, Linnea Passing (gdb@in.tum.de)
http://www-db.in.tum.de/teaching/ws1516/grundlagen/

Blatt Nr. 07

Hausaufgabe 1

Gegeben sei das folgende ER-Modell, bei dem wir die Relation *verheiratet* nach dem deutschen Gesetz (d.h. jeder Mensch kann höchstens einen Ehegatten haben) und die Relation *Eltern_von* im biologischen Sinn (d.h. jeder Mensch hat genau eine Mutter und einen Vater) modelliert haben:



Bestimmen Sie sinnvolle Min/Max-Angaben. Geben Sie dann die SQL-Statements zur Erzeugung der Tabellen an, die der Umsetzung des Diagramms in Relationen entsprechen! Verwenden Sie dabei **not null**, **primary key**, **references**, **unique** und **cascade**.

Lösung:

Die folgenden SQL-Statements erzeugen die Tabellen:

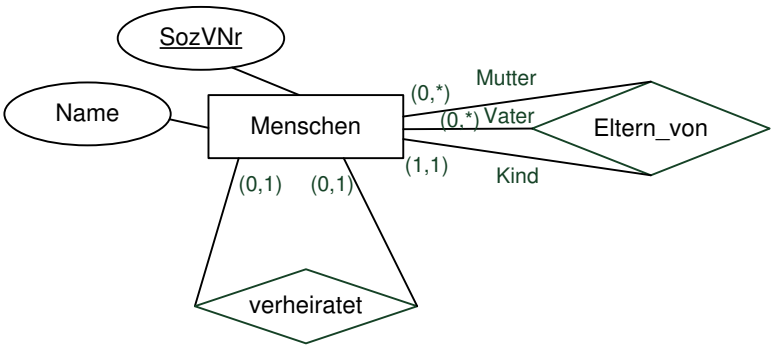
```
create table Menschen (
  SozVNr     varchar(30) not null primary key,
  Name       varchar(30)
);

create table Eltern_von (
  MutterVater  varchar(30) not null references Menschen,
  Kind         varchar(30) not null references Menschen,
  primary key (MutterVater, Kind)
);

create table verheiratet (
  Ehegatte1 varchar(30) not null references Menschen on
     delete cascade,
  Ehegatte2 varchar(30) not null references Menschen on
     delete cascade,
  primary key (Ehegatte1),
  unique (Ehegatte2)
);
```

In DB2 müssen alle Attribute, die Teil des Primärschlüssels sind, als **not null** definiert werden. Grundsätzlich ist es auch sinnvoll, **null**-Werte bei Schlüsselattributen auszuschließen. Obwohl der SQL-Standard von 1992 vorschreibt, dass Primärschlüsselattribute implizit als **not null** definiert sind, wird das nicht von allen Datenbanksystemen implementiert (z.B. DB2). In der Tabelle *Menschen* können wir für Namen **null**-Werte zulassen wenn wir davon ausgehen, dass Eltern einige Wochen bis Monate Zeit haben, um einen Namen auszusuchen, das Kind aber zu dieser Zeit schon registriert ist. In *Eltern_von* sollen nur bekannte Eltern-Kind-Beziehungen eingetragen werden, deshalb sind alle Attribute als **not null** deklariert. Sowohl *MutterVater* als auch *Kind* sind *Menschen*, referenzieren also die *Menschen*-Tabelle. Da ein Kind zwei Elternteile hat, setzt sich der Primärschlüssel aus beiden Attributen *MutterVater* und *Kind* zusammen. Als Primärschlüssel von *verheiratet* kann entweder *Ehegatte1* oder *Ehegatte2* gewählt werden. Der jeweils andere Ehegatte muss als **unique** gekennzeichnet werden. Da mit dem Tod eines Menschen dessen Ehe auch beendet ist, wurden die Fremdschlüssel *Ehegatte1* und *Ehegatte2* mit dem Zusatz **on delete cascade** angelegt. Da davon auszugehen ist, dass sich die Sozialversicherungsnummer niemals ändert, haben wir kein **on update cascade** verwendet. Könnte sie sich ändern, wäre bei allen Attributen, die *Menschen* referenzieren **on update cascade** hinzugefügt werden. (Hinweis: DB2 unterstützt kein **on update cascade**, sondern lediglich **on update restrict**.)

Man hätte den Ehepartner auch in die Relation *Menschen* mit aufnehmen können, da es sich um eine 1:1-Beziehung handelt. Dies würde aber zu vielen **null**-Werten führen (weil sehr viele Menschen keinen Ehepartner haben) und ist daher nicht empfehlenswert. Die Relation *Eltern_von* könnte alternativ auch mit den drei Attributen *Mutter*, *Vater* und *Kind* umgesetzt werden. Dies würde dem folgenden ER-Modell entsprechen:



Die Umsetzung wäre dann:

```
create table Eltern_von (
  Mutter  varchar(30) not null references Menschen,
  Vater   varchar(30) not null references Menschen,
  Kind    varchar(30) not null references Menschen,
  primary key (Kind)
);
```

Hausaufgabe 2

Gegeben sei eine Relation

$$R : \{[A : \text{integer}, B : \text{integer}, C : \text{integer}, D : \text{integer}, E : \text{integer}]\},$$

die schon sehr viele Daten enthält (Millionen Tupel). Sie „vermuten“, dass folgendes gilt:

(a) AB ist ein Schlüssel der Relation

(b) $DE \rightarrow B$

Formulieren Sie SQL-Anfragen, die Ihre Vermutungen bestätigen oder widerlegen.

Lösung:

(a) Da jedes Tupel in einer Relation einen eindeutigen Schlüssel besitzt, kann nach der Gruppierung nach A und B anhand der Anzahl der Tupel ermittelt werden, ob hier eine Verletzung der Schlüsseleigenschaft vorliegt. Werden also mindestens zwei Tupel mit den gleichen Werten für A und B als Ergebnis ausgegeben, so bildet AB keinen Schlüssel der Relation, ist das Ergebnis der Anfrage leer, so ist AB ein Schlüssel.

```
select A, B
from R
group by A, B
  having count(*) > 1;
```

(b) In diesem Fall muss nur gelten, dass für alle Tupel, die gleiche Werte in D und E besitzen, auch die Werte für das Attribut B gleich sind. D.h. wenn nach D und E gruppiert wird, muss die Anzahl der verschiedenen Werte für B kleiner oder gleich 1 sein. Es gilt wieder, dass das Ergebnis der Anfrage alle Tupel enthält, die die Vermutung verletzen. Ist das Ergebnis leer, so gilt $DE \rightarrow B$.

```
select D, E
from R
group by D, E
  having count(distinct B) > 1;
```

Hausaufgabe 3

Betrachten Sie das Relationenschema

PunkteListe: {Name, Aufgabe, Max, Erzielt, KlausurSumme, KNote, Bonus, GNote}

mit der folgenden beispielhaften Ausprägung:

PunkteListe							
Name	Aufgabe	Max	Erzielt	KlausurSumme	KNote	Bonus	GNote
Bond	1	10	4	18	2	ja	1.7
Bond	2	10	10	18	2	ja	1.7
Bond	3	11	4	18	2	ja	1.7
Maier	1	10	4	9	4	nein	4
Maier	2	10	2	9	4	nein	4
Maier	3	11	3	9	4	nein	4

1. Bestimmen Sie die geltenden FDs.
2. Bestimmen Sie die Kandidatenschlüssel.

Lösung:

1. Im Relationenschema gelten die folgenden funktionalen Abhängigkeiten:
   - {KNote, Bonus} → {GNote}
   - {Aufgabe} → {Max}
   - {KlausurSumme} → {KNote}
   - {Name, Aufgabe} → {Erzielt}
   - {Name} → {KlausurSumme, Bonus}

   Natürlich gelten auch alle anderen funktionalen Abhängigkeiten, die mit Hilfe der Armstrong-Axiome daraus hergeleitet werden können.
2. Der Kandidatenschlüssel ist {Name, Aufgabe}. Aus {Name} können die Attribute {KlausurSumme, Bonus}, aus {KlausurSumme} wiederum {KNote}, und aus {KNote, Bonus} dann {GNote} abgeleitet werden. Aus {Aufgabe} kann {Max} abgeleitet werden, und aus {Name, Aufgabe} noch das verbleibende Attribut {Erzielt}.

Hausaufgabe 4

Betrachten Sie ein abstraktes Relationenschema $\mathcal{R} = \{A, B, C, D, E, F\}$ mit den FDs

1. $A \to BC$
2. $C \to DA$
3. $E \to ABC$
4. $F \to CD$
5. $CD \to BEF$

(a) Berechnen Sie die Attributhülle von A.

(b) Bestimmen Sie alle Kandidatenschlüssel.

(c) Bestimmen Sie zu den gegebenen FDs die kanonische Überdeckung.

(d) Überführen Sie die Relation in die dritte Normalform, indem Sie den Synthesealgorithmus anwenden.

Lösung:

Attributhülle von A

Berechnung der Attributhülle von A mit Hilfe des bekannten *AttrHülle*-Algorithmus.

Aufruf: $\mathit{AttrHülle}(FD, \{A\})$.

Schritt	betrachtete FD	Ergebnis
init		$\{A\}$
1.	$A \to BC$	$\{A, B, C\}$
2.	$C \to DA$	$\{A, B, C, D\}$
3.	$CD \to BEF$	$\{A, B, C, D, E, F\}$

Damit enthält die Attributhülle von A alle Attribute.

Kandidatenschlüssel

$\{A\}$ ist nach der vorherigen Berechnung (Attributhülle von A) ein Superschlüssel. Da $\{A\}$ außerdem minimal ist, ist $\{A\}$ ein Kandidatenschlüssel. Da man aus $\{C\}$ und $\{E\}$ direkt A folgern kann, handelt es sich hier ebenfalls um Superschlüssel und da sie einelementig sind (also minimal sind) auch um Kandidatenschlüssel. Aus $\{F\}$ wiederum kann C und somit A gefolgert werden. Damit ist $\{F\}$ analog zu oben auch ein Kandidatenschlüssel.

$\{B\}$ und $\{D\}$ sind dagegen keine Kandidatenschlüssel. $\{B\}$ ist nicht einmal Superschlüssel. $\{CD\}$ wäre zwar ein Superschlüssel, allerdings kein Kandidatenschlüssel, da nicht minimal.

Kandidatenschlüssel sind: $\{A\}, \{C\}, \{E\}, \{F\}$.

Kanonische Überdeckung

Gegeben ist die Ausgangsmenge $F = \{A \to BC, C \to DA, E \to ABC, F \to CD, CD \to BEF\}$.

1. *Führe für jede FD $\alpha \to \beta \in F$ die Linksreduktion durch.*

   Einzige in Betracht kommende FD ist $CD \to BEF$.

   - Ist C überflüssig?
     $\mathit{AttrHülle}(F, \{D\}) = \{D\} \not\supseteq \{B, E, F\}$
   - Ist D überflüssig?
     $\mathit{AttrHülle}(F, \{C\}) =$

Schritt	betrachtete FD	Ergebnis
init		$\{C\}$
1.	$C \to DA$	$\{A, C, D\}$
2.	$CD \to BEF$	$\{A, B, C, D, E, F\}$

     $\{C\}^+ = \{A, B, C, D, E, F\} \supseteq \{B, E, F\}$

   Damit kann $CD \to BEF$ zu $C \to BEF$ reduziert werden.

2. *Führe für jede (verbliebene) FD $\alpha \to \beta$ die Rechtsreduktion durch.*

   Bisheriges Zwischenergebnis:

   $$A \to BC \quad (1)$$
   $$C \to DA \quad (2)$$
   $$E \to ABC \quad (3)$$
   $$F \to CD \quad (4)$$
   $$C \to BEF \quad (5)$$

   Betrachte FD (1)**:**

   - Ist B überflüssig?
     $B \in \mathit{AttrHülle}(F - \text{FD } (1) \cup (A \to C), A)$, da $A \to C \to BEF$.
   - Ist C überflüssig?
     $C \notin \mathit{AttrHülle}(F - \text{FD } (1) \cup (A \to \emptyset), A)$.

   Damit erhält man für FD (1): $A \to C$.

   Betrachte FD (2)**:**

- Ist D überflüssig?
  $D \in \mathit{AttrHülle}(F - \text{FD } (2) \cup (C \to A), C)$, da $C \to BEF, F \to CD$.
- Ist A überflüssig?
  $A \in \mathit{AttrHülle}(F - \text{FD } (2) \cup (C \to \emptyset), C)$, da $C \to BEF, E \to ABC$.

Damit erhält man für FD (2): $C \to \emptyset$.

Betrachte FD (3)**:**

- Ist A überflüssig?
  $A \notin \mathit{AttrHülle}(F - \text{FD } (3) \cup (E \to BC), E)$.
- Ist B überflüssig?
  $B \in \mathit{AttrHülle}(F - \text{FD } (3) \cup (E \to AC), E)$, da $E \to AC, C \to BEF$.
- Ist C überflüssig?
  $C \in \mathit{AttrHülle}(F - \text{FD } (3) \cup (E \to A), E)$, da $E \to A, A \to C$.

Damit erhält man für FD (3): $E \to A$.

Betrachte FD (4)**:**

- Ist C überflüssig?
  $C \notin \mathit{AttrHülle}(F - \text{FD } (4) \cup (F \to D), F)$.
- Ist D überflüssig?
  $D \notin \mathit{AttrHülle}(F - \text{FD } (4) \cup (F \to C), F)$.

Damit bleibt FD (4) unverändert.

Betrachte FD (5)**:**

- Ist B überflüssig?
  $B \notin \mathit{AttrHülle}(F - \text{FD } (5) \cup (C \to EF), C)$.
- Ist E überflüssig?
  $E \notin \mathit{AttrHülle}(F - \text{FD } (5) \cup (C \to BF), C)$.
- Ist F überflüssig?
  $F \notin \mathit{AttrHülle}(F - \text{FD } (5) \cup (C \to BE), C)$.

Damit bleibt FD (5) unverändert.

3. *Entferne die FDs der Form $\alpha \to \emptyset$.*

   Bisheriges Zwischenergebnis:

$$\begin{array}{lcl} A & \to & C \\ C & \to & \emptyset \\ E & \to & A \\ F & \to & CD \\ C & \to & BEF \end{array} \tag{6}$$

   FD (6) wird eliminiert.

4. *Fasse mittels der Vereinigungsregel FDs der Form $\alpha \to \beta_1, \ldots, \alpha \to \beta_n$ zusammen.*

Bisheriges Zwischenergebnis:

$$F_c = \begin{cases} A \to C \\ E \to A \\ F \to CD \\ C \to BEF \end{cases}$$

Es werden keine FDs vereinigt, da es keine zwei FDs mit gleicher linker Seite gibt.

F_c ist eine kanonische Überdeckung zur Ausgangsmenge F.

Dritte Normalform

Bestimmen der kanonischen Überdeckung siehe oben. Für jede funktionale Abhängigkeit aus der kanonischen Überdeckung wird ein Relationenschema erstellt:

$$\begin{aligned} R_1 &= \{\underline{A}, C\} \\ R_2 &= \{\underline{E}, A\} \\ R_3 &= \{\underline{F}, C, D\} \\ R_4 &= \{\underline{C}, B, E, F\} \end{aligned}$$

R_1 enthält einen der Kandidatenschlüssel (sogar zwei: nämlich $\{A\}$ und $\{C\}$), so dass kein zusätzliches Schema erstellt werden muss. Keines der Relationenschemata ist in einem anderen Schema enthalten, so dass nichts eliminiert werden kann.