Kapitel 11: Relationale Entwurfstheorie

Ziele:

- Charakterisierung "guter" relationaler Schemata
  - jede Relation entspricht genau einer Objektmenge - eventuell unter Einbezug von N:1- oder 1:1-Relationships - oder genau einer Relationship-Menge zwischen Objekten.
  - Redundanz ist eliminiert, alle Informationen sind repräsentierbar, und es treten keinerlei "Änderungsanomalien" auf
    - Änderungen können bei Beachtung der Primärschlüssel- und Fremdschlüsselbedingung keine Inkonsistenzen hervorrufen
    - alle Informationen lassen sich unter Wahrung der Primärschlüssel- und Fremdschlüsselbedingung (ohne "Kunstgriffe") einfügen
    - Informationen können einzeln wieder gelöscht werden, ohne die Primärschlüssel- oder Fremdschlüsselbedingung zu verletzen
- Algorithmische Herleitung solcher "guter" Schemata

Negativbeispiel:

Bestellungen (Datum, KNr, PNr, Bez, Preis, Gewicht, Menge)

"Typische" Ausprägung:

Datum	**KNr**	**PNr**	**Bez**	**Preis**	**Gewicht**	**Menge**
16.7.	1	1	Papier	20.00	2.000	100
21.7.	1	1	Papier	20.00	2.000	200
26.10.	2	1	Papier	20.00	2.000	100
26.10.	2	5	Disketten	20.00	0.500	50

11.1 Funktionalabhängigkeiten (Functional Dependencies)

Definition:
Sei R eine Relation mit Attributmenge sch(R). Für X = {X1, ..., Xm} $\subseteq$ sch(R) und Y = {Y1, ..., Yn} $\subseteq$ sch(R) gilt die *Funktionalabhängigkeit* $X \rightarrow Y$, gesprochen: X bestimmt Y, wenn zu jedem Zeitpunkt für je zwei Tupel r, s $\in$ val(R) gelten muß:

(r.X1 = s.X1 $\wedge$ ... $\wedge$ r.Xm = s.Xm) $\Rightarrow$ (r.Y1 = s.Y1 $\wedge$ ... $\wedge$ r.Yn = s.Yn)

(in Kurznotation: r.X = s.X $\Rightarrow$ r.Y = s.Y).

Beispiel:
Bestellungen (Datum, KNr, PNr, Bez, Preis, Gewicht, Menge)
Es gelten beispielsweise die Funktionalabhängigkeiten:

Datum, KNr, PNR $\rightarrow$ Menge

PNr $\rightarrow$ Bez, Preis, Gewicht

Es gelten nicht die Funktionalabhängigkeiten:

KNr, PNR $\rightarrow$ Menge

PNr $\rightarrow$ Datum

Achtung: Funktionalabhängigkeiten sind Integritätsbedingungen, die für alle möglichen Ausprägungen gelten sollen. Insofern kann man aus einer Ausprägung nur schließen, welche Funktionalabhängigkeiten nicht gelten. Eine "typische" Ausprägung kann freilich Anhaltspunkte für Funktionalabhängigkeiten liefern.

Konkretisierung des Ziels der relationalen Entwurfstheorie:
Alle Funktionalabhängigkeiten sollen in Form von relationalen Primärschlüsselbedingungen verkörpert werden.
Beispiel:
Bestellungen (Datum, KNr, PNr, Bez, Preis, Gewicht, Menge)
verletzt diese Forderung, da beispielsweise PNr $\rightarrow$ Bez keine Primärschlüsselbedingung ist.

Ableitungsregeln für Funktionalabhängigkeiten

Definition:
Sei R eine Relation mit Attributmenge sch(R) und einer Menge F von spezifizierten Funktionalabhängigkeiten. Die Menge aller aus F logisch ableitbaren Funktionalabhängigkeiten wird als *transitive Hülle* F^+ von F bezeichnet.

Regeln:

Sei R eine Relation mit Attributmenge sch(R). Seien $X \subseteq sch(R)$, $Y \subseteq sch(R)$, $Z \subseteq sch(R)$ Mengen von Attributen von R.

Regel 1 (Reflexivität): Wenn $Y \subseteq X$, dann gilt: $X \rightarrow Y$.
Regel 2 (Erweiterung): Wenn $X \rightarrow Y$, dann gilt: $XZ \rightarrow YZ$
(ausführliche Notation: $(X \cup Z) \rightarrow (Y \cup Z)$).
Regel 3 (Transitivität): Wenn $X \rightarrow Y$ und $Y \rightarrow Z$, dann gilt: $X \rightarrow Z$.

Diese drei Regeln werden auch als Armstrong-Regeln oder Armstrong-Axiome (eines Inferenzsystems für Funktionalabhängigkeiten) bezeichnet. Aus ihnen folgen weitere Ableitungsregeln, beispielsweise:

Regel 4 (Vereinigung): Wenn $X \rightarrow Y$ und $X \rightarrow Z$, dann gilt: $X \rightarrow YZ$.
Regel 5 (Pseudotransitivität): Wenn $X \rightarrow Y$ und $WY \rightarrow Z$ mit $W \subseteq sch(R)$, dann gilt: $XW \rightarrow Z$.
Regel 6 (Zerlegung): Wenn $X \rightarrow Y$ und $Z \subseteq Y$, dann gilt: $X \rightarrow Z$.

Beispiel:
Bestellungen (Datum, KNr, PNr, Bez, Preis, Gewicht, Menge)
Sei F = { Datum, KNr, PNR $\rightarrow$ Menge,
PNr $\rightarrow$ Bez, Preis, Gewicht }.

Es folgen:	Datum, KNr, PNR $\rightarrow$ PNr	nach der Reflexivitätsregel,
	Bez, Preis, Gewicht $\rightarrow$ Bez	nach der Reflexivitätsregel,
	PNr $\rightarrow$ Bez	nach der Transitivitätsregel und schließlich
	Datum, KNr, PNr $\rightarrow$ Bez	nach der Transitivitätsregel.

Satz:
Sei R eine Relation mit einer Menge F von Funktionalabhängigkeiten. Durch endlichmalige Anwendung der drei Armstrong-Regeln lassen sich genau alle Funktionalabhängigkeiten in F^+ herleiten.
Die Armstrong-Regeln bilden also einen korrekten und vollständigen Ableitungskalkül für die Funktionalabhängigkeiten in F^+.

Beweis:
siehe Vorlesung

Ableitbarkeitstest für Funktionalabhängigkeiten

Definition:

Sei R eine Relation mit Attributmenge sch(R) und einer Menge F von Funktionalabhängigkeiten, und sei $X \subseteq sch(R)$. Die *Hülle* X^+ der Attributmenge X ist die Menge aller Attribute $A \in sch(R)$, für die gilt $X \rightarrow A \in F^+$.

Algorithmus zur Bestimmung von X^+:

Eingabe: Attributmenge sch(R)
Funktionalabhängigkeiten F
Attributmenge $X \subseteq sch(R)$

Ausgabe: X^+

```
procedure xplus (X: set of attribute): set of attribute;
var   H: set of attribute;
      newattr: Boolean;
begin
   H := X; newattr := true;
   while newattr do (* Invariante: X → H *)
      newattr := false;
      for each Y → Z ∈ F do
         if Y ⊆ H then
            if not (Z ⊆ H) then H := H ∪ Z; newattr := true; fi
         fi
      od
   od
   return H
end xplus;
```

Korrektheitsskizze:

Es gilt die Schleifeninvariante, was durch Induktion über die Anzahl der Schleifendurchläufe gezeigt werden kann.

Verankerung:

Vor Eintritt in die Schleife ist H = X, und es gilt $X \rightarrow X$ nach Armstrong-Regel 1.

Induktionsschluß:

Es gelte $X \rightarrow H(i-1)$ vor dem i-ten Schleifendurchlauf; dabei sei H(i-1) der Wert von H nach dem (i-1)-ten Durchlauf. Nach dem i-ten Schleifendurchlauf gilt dann:
$X \rightarrow H(i-1)$ und $H(i-1) \rightarrow Y \rightarrow Z$ sowie nach Regel 3 auch $X \rightarrow Z$ und nach Regel 4 auch $X \rightarrow H(i-1)\ Z$.
Mit der Zuweisung H(i) := H(i-1) Z am Ende des Schleifenrumpfes folgt also $X \rightarrow H(i)$.

Beispiel:

sch(R) = {A, B, C, D, E, G}
F = {AB → C, ACD → B, BC → D, BE → C, C → A, CG → BD, CE → AG, D → EG}
X = {B, D}

nach Schleifendurchlauf	H
0	{B, D}
1	{B, D, E, G}
2	{B, D, E, G, C, A}
3	{B, D, E, G, C, A}

Definition:

Sei R eine Relation mit Attributmenge sch(R) und einer Menge F von Funktionalabhängigkeiten, und sei $X \subseteq sch(R)$. Eine Attributmenge $X \subseteq sch(R)$ ist ein *Schlüsselkandidat*, wenn $X \rightarrow sch(R) \in F^+$ gilt und es keine echte Teilmenge von X gibt, die diese Eigenschaft hat.
Ein Attribut $A \in sch(R)$ heißt Schlüsselattribut, wenn es in mindestens einem Schlüsselkandidaten von R vorkommt.

Bemerkung:
Diese Definition eines Schlüsselkandidaten ist konsistent zur Definition von Kapitel 2.

Algorithmus zur Bestimmung aller Schlüsselkandidaten einer Relation:

Eingabe: Attributmenge sch(R)
Funktionalabhängigkeiten F
Ausgabe: alle Schlüsselkandidaten von R

```
procedure findkeys: set of set of attribute;
var   K: set of set of attribute;
      iskey: Boolean;
begin
   K := ∅;
   for each S ∈ 2^sch(R) do
      iskey := true;
      if xplus (S) ≠ sch(R) then iskey := false fi;
      for each A ∈ S while iskey do
         if xplus (S - {A}) = sch(R) then iskey := false fi
      od
      if iskey then K := K ∪ {S} fi
   od
   return K
end findkeys;
```

11.2 Relationale Normalformen

Definition Boyce-Codd-Normalform (BCNF):

Sei R eine Relation mit Attributmenge sch(R) und einer Menge F von Funktionalabhängigkeiten. R ist in *Boyce-Codd-Normalform (BCNF)*, wenn für jede Funktionalabhängigkeit $X \rightarrow A \in F^+$ mit $X \subseteq sch(R)$, $A \in sch(R)$ und $A \notin X$ gilt, daß X einen Schlüsselkandidaten enthalten muß.

Definition 3. Normalform (3NF):

Sei R eine Relation mit Attributmenge sch(R) und einer Menge F von Funktionalabhängigkeiten. R ist in *3. Normalform (3NF)*, wenn für jede Funktionalabhängigkeit $X \rightarrow A \in F^+$ mit $X \subseteq sch(R)$, $A \in sch(R)$ und $A \notin X$ gilt, daß X einen Schlüsselkandidaten enthalten muß oder A ein Schlüsselattribut ist.

Satz:

Jede Relation in BCNF ist auch in 3NF.

Algorithmus zum Testen, ob eine Relation in BCNF ist

Eingabe: Attributmenge sch(R)
Funktionalabhängigkeiten F
Ausgabe: true g.d.w. die Relation in BCNF ist

```
procedure BCNFtest: Boolean;
var   isbcnf: Boolean;
begin
   isbcnf := true;
   for each X → A ∈ F while isbcnf do
      if A ∉ X then if XPlus(X) ≠ sch(R) then isbcnf := false fi fi
   od
   return isbcnf;
end BCNFtest;
```

Beispiele:

1) Best (Datum, KNr, PNr, Menge) mit F = {Datum, KNr, PNr $\rightarrow$ Menge}
und
Prod (PNr, Bez, Preis, Gewicht) mit F = {PNr $\rightarrow$ Bez, Preis, Gewicht}
sind in BCNF.

2) Vorlesungen (Titel, Zeit, Raum)
mit F = { Titel $\rightarrow$ Raum,
Zeit, Raum $\rightarrow$ Titel}
ist in 3NF, nicht aber in BCNF.
Schlüsselkandidaten sind {Titel, Zeit} sowie {Zeit, Raum}.
Die Funktionalabhängigkeit Titel $\rightarrow$ Raum verletzt die BCNF-Bedingung.
Die Anomalie, daß eine Vorlesung zu verschiedenen Zeiten in verschiedenen Räumen stattfindet, wäre also nur durch die Primärschlüsselbedingung nicht auszuschließen.

Beispielausprägung zu Beispiel 2:

Titel	**Zeit**	**Raum**
Datenbanken	Di 9-11	HS 2
Datenbanken	Do 9-11	HS 2
Compiler	Di 9-11	HS 3
Compiler	Mi 13-15	HS 3
Betriebssysteme	Mi 13-15	HS 2

11.3 Relationendekomposition

Definition Abhängigkeitsbewahrung:

Sei R eine Relation mit Attributmenge sch(R) und einer Menge F von Funktionalabhängigkeiten. Eine Zerlegung von R in R1 (X1), ..., Rk (Xk) mit Xi $\subseteq$ sch(R) und
X1 $\cup$ .. $\cup$ Xk = sch(R) heißt *abhängigkeitsbewahrend*, wenn gilt:

$$\left((F^+ \mid_{X1})^+ \cup ... \cup (F^+ \mid_{Xk})^+ \right)^+ = F^+,$$

wobei $F^+ \mid_{Xi}$ diejenigen Funktionalabhängigkeiten aus F^+ bezeichnen, deren linke und rechte Seite in Xi enthalten sind.

Beispiele:

1) Bestellungen (Datum, KNr, PNr, Bez, Preis, Gewicht, Menge}
   mit F = { Datum, KNr, PNR $\rightarrow$ Menge,
   PNr $\rightarrow$ Bez, Preis, Gewicht}
   Zerlegung in
   Best (Datum, KNr, PNr, Menge) mit F1 = {Datum, KNr, PNR $\rightarrow$ Menge} und
   Prod (PNr, Bez, Preis, Gewicht) mit F2 = {PNr $\rightarrow$ Bez, Preis, Gewicht}
   ist abhängigkeitsbewahrend.

2) Vorlesungen (Titel, Zeit, Raum)
   mit F = { Titel $\rightarrow$ Raum,
   Zeit, Raum $\rightarrow$ Titel}
   Zerlegung in
   Vorlesungsräume (Titel, Raum) mit F1 = {Titel $\rightarrow$ Raum} und
   Vorlesungszeiten (Titel, Zeit) mit F2 = $\varnothing$
   ist nicht abhängigkeitsbewahrend.

Definition Verlustfreiheit:

Sei R eine Relation mit Attributmenge sch(R) und einer Menge F von Funktionalabhängigkeiten. Eine Zerlegung von R in R1 (X1), ..., Rk (Xk) mit Xi $\subseteq$ sch(R) und X1 $\cup$ ... $\cup$ Xk = sch(R) heißt *(projektion-join-) verlustfrei*, wenn für alle möglichen Ausprägungen, die F erfüllen, gilt:

π[X1](R) |×| ... |×| π[Xk](R) = R.

Beispiel:

Bestellungen (Datum, KNr, PNr, Bez, Preis, Gewicht, Menge)
mit F = { Datum, PNr, KNr $\rightarrow$ Menge,
PNr $\rightarrow$ Bez, Preis, Gewicht} und
der Beispielausprägung:

Datum	**KNr**	**PNr**	**Bez**	**Preis**	**Gewicht**	**Menge**
16.7.	1	1	Papier	20.00	2.000	100
16.7.	1	5	Disketten	20.00	0.500	50

Zerlegung in
Best1 Best2

Datum	**KNr**	**PNr**	**Menge**
16.7.	1	1	100
16.7.	1	5	50

und

Datum	**Bez**	**Preis**	**Gewicht**
16.7.	Papier	20.00	2.000
16.7.	Disketten	20.00	0.500

ist nicht verlustfrei, denn Best1 |×| Best2 hat die folgende Ausprägung:

Datum	**KNr**	**PNr**	**Bez**	**Preis**	**Gewicht**	**Menge**
16.7.	1	1	Papier	20.00	2.000	100
16.7.	1	5	Disketten	20.00	0.500	50
16.7.	1	1	Disketten	20.00	0.500	100
16.7.	1	5	Papier	20.00	2.000	50

Satz:
Zu jeder Relation R mit Funktionalabhängigkeiten F gibt es eine Zerlegung von R in 3NF-Relationen, die abhängigkeitsbewahrend und verlustfrei ist.

Satz:
Zu jeder Relation R mit Funktionalabhängigkeiten F gibt es eine Zerlegung von R in BCNF-Relationen, die verlustfrei ist.

Satz:
Sei R eine Relation mit Attributmenge sch(R) und einer Menge F von Funktionalabhängigkeiten. Sei $X \rightarrow Y \in F^+$ mit $X \cap Y = \emptyset$.
Die Zerlegung von R in R' (X, Y) und R" (X, sch(R) - Y) ist verlustfrei.

Algorithmus für verlustfreie BCNF-Dekomposition

Eingabe: Relationenschema sch(R)
Funktionalabhängigkeiten F
Ausgabe: verlustfreie Zerlegung in R1, ..., Rk, so daß R1, ..., Rk alle in BCNF sind

Algorithmus:
Teste, ob R in BCNF ist
Falls R nicht in BCNF ist
(* es gibt eine Funktionalabhängigkeit $X \rightarrow Y \in F^+$ mit $X \cap Y = \emptyset$ und $X \rightarrow sch(R) \notin F^+$ *)
dann
zerlege R in R' (X, Y) und R" (X, sch(R) - Y)
wiederhole den Zerlegungsalgorithmus für R' und R"

Beispiel:

R (FlugNr, Datum, Abflugzeit, FSNr, SitzNr, TicketNr, Name, Adresse, GNr, Gewicht)
F = { FlugNr, Datum → Abflugzeit, FSNr,
FlugNr, Datum, TicketNr → SitzNr,
TicketNr → Name, Adresse,
GNr → Gewicht }

Schlüsselkandidat: FlugNr, Datum, TicketNr, GNr

Zerlegungsschritte:

1. Zerlegung von R entlang der Funktionalabhängigkeit: FlugNr, Datum → Abflugzeit, FSNr:
   R1 (FlugNr, Datum, Abflugzeit, FSNr) in BCNF und
   R2 (FlugNr, Datum, SitzNr, TicketNr, Name, Adresse, GNr, Gewicht)

2. Zerlegung von R2 entlang der Funktionalabhängigkeit: FlugNr, Datum, TicketNr → SitzNr:
   R21 (FlugNr, Datum, TicketNr, SitzNr) in BCNF und
   R22 (FlugNr, Datum, TicketNr, Name, Adresse, GNr, Gewicht)

3. Zerlegung von R22 entlang der Funktionalabhängigkeit: TicketNr → Name, Adresse:
   R221 (TicketNr, Name, Adresse) in BCNF und
   R222 (TicketNr, FlugNr, Datum, GNr, Gewicht)

4. Zerlegung von R222 entlang der Funktionalabhängigkeit: GNr → Gewicht:
   R2221 (GNr, Gewicht) in BCNF und
   R2222 (GNr, FlugNr, Datum, TicketNr) in BCNF

Umbenennungen:

R1	in Flüge
R21	in Sitzreservationen
R221	in PassagiereMitTicket
R2221	in Gepäckstücke
R2222	in Check-in

Achtung:
Das Resultat der Relationendekomposition ist abhängig von der Reihenfolge der Zerlegungsschritte und der Wahl der Funktionalabhängigkeiten für die Zerlegungen.
Ein intuitiv guter Entwurf wird u.U. nur durch geschickte Wahl der Zerlegungsschritte erreicht.

11.4 Relationensynthese

Definition Überdeckung:
Sei eine Relation mit Attributmenge sch(R) und einer Menge F von Funktionalabhängigkeiten. Eine Menge F' von Funktionalabhängigkeiten über sch(R) heißt Überdeckung (engl. cover) von F, wenn gilt: $(F')^+ = F^+$.

Definition minimale Überdeckung:
Eine Überdeckung F' von F heißt minimal, wenn gilt:

1) In jeder Funktionalabhängigkeit von F' hat die rechte Seite ein Attribut.
2) Keine echte Teilmenge von F' ist eine Überdeckung von F.
3) Für jede Funktionalabhängigkeit $X \rightarrow A \in F'$ und jede echte Teilmenge X' von X ist $F'' := F' - \{ X \rightarrow A\} \cup \{X' \rightarrow A\}$ keine Überdeckung mehr von F.

Beispiel:
sch(R) = {Ang(estellten)Nr, Name, Gehalt, Leistung(sbeurteilung), Tarif(klasse),
Abt(eilungs)Nr, ProjektNr, Projektleiter, Abt(eilungs)leiter}
F = { AngNr → Name,
AngNr → Gehalt,
AngNr → Tarif,
AngNr → AbtNr,
AngNr → Abtleiter,
ProjektNr → Projektleiter,
AbtNr → Abtleiter,
AngNr, Abtleiter → Leistung,
Tarif, Leistung → Gehalt}

Schritt 1: Entferne redundante Attribute auf den linken Seiten
→ { AngNr → Name,
AngNr → Gehalt,
AngNr → Tarif,
AngNr → AbtNr,
AngNr → Abtleiter,
ProjektNr → Projektleiter,
AbtNr → Abtleiter,
AngNr → Leistung,
Tarif, Leistung → Gehalt}

Schritt 2: Entferne redundante Funktionalabhängigkeiten
→ { AngNr → Name,
AngNr → Tarif,
AngNr → AbtNr,
ProjektNr → Projektleiter,
AbtNr → Abtleiter,
AngNr → Leistung,
Tarif, Leistung → Gehalt} = F'

Algorithmus zur Berechnung einer minimalen Überdeckung:

Eingabe: Attributmenge sch(R)
Funktionalabhängigkeiten F
(Annahme: alle rechten Seiten von F bestehen aus einem Attribut)
Ausgabe: minimale Überdeckung von F

```
procedure mincover: set of FDs;

  var G: set of FDs;
  procedure xplus (arg1: set of FDs, arg2: set of attribute): set of attribute;
  procedure reduce (arg: set of FDs): set of FDs;
    var H: set of FDs;
    begin
    H := arg;
    for each X→A∈H do
      for each C∈X do
        Hred := H - { X→A} ∪ {(X-{C})→A};
        if A ∈ xplus (H, X-{C})) then H := Hred fi;
      od;
    od;
    return H
    end (* reduce *);

begin
G := reduce(F);
for each X→A∈G do
  G := G - { X→A};
  if not (A ∈ xplus (G, X)) then G := G ∪ { X→A} fi;
od;
return G
end mincover;
```

Achtung:
Es kann mehrere, verschiedene, minimale Überdeckungen von F geben. Das Ergebnis des Algorithmus ist also abhängig von der Reihenfolge, in der die Funktionalabhängigkeiten inspiziert werden.

Algorithmus zur Relationensynthese:

Eingabe: Attributmenge sch(R)
Funktionalabhängigkeiten F
Ausgabe: Menge von Relationen R1, ..., Rn mit Schemata sch(R1), ..., sch(Rn) in 3NF
die eine abhängigkeitsbewahrende Zerlegung von R bilden.

Schritt 1: Bereche eine minimale Überdeckung F' von F.
Schritt 2: Partitioniere die Funktionalabhängigkeiten in F' nach gleichen linken Seiten, d.h. für jede Attributmenge Xi, die als linke Seite einer oder mehrerer Funktionalabhängigkeiten in F' vorkommt, alle Funktionalabhängigkeiten $Xi \rightarrow A1$, ..., $Xi \rightarrow Ak_i$ zusammen.
Schritt 3: Bilde für jede Partition von Schritt 2 eine Relation Ri mit $sch(Ri) = Xi \cup \{Ai, ..., Ak_i\}$.
Schritt 4: Falls $sch(R) - (\cup_i sch(Ri))$ nichtleer ist (es also Attribute in sch(R) gibt, die in keiner Funktionalabhängigkeit von F' vorkommen), bilde eine weitere Relation R' mit $sch(R') = sch(R) - (\cup_i sch(Ri))$.
Schritt 5: Falls keine der erzeugten Relationen Ri, R' einen Schlüssel von R enthält, erzeuge eine weitere Relation R", deren Schema aus einem Schlüssel von R besteht.

Satz:
Der Algorithmus zur Relationensynthese erzeugt eine abhängigkeitsbewahrende und verlustfreie 3NF-Zerlegung von R.

Beispiel:
sch(R) = {Ang(estellten)Nr, Name, Gehalt, Leistung(sbeurteilung), Tarif(klasse),
Abt(eilungs)Nr, ProjektNr, Projektleiter, Abt(eilungs)leiter}
mit minimaler Überdeckung
F' = { AngNr $\rightarrow$ Name,
AngNr $\rightarrow$ Tarif,
AngNr $\rightarrow$ AbtNr,
ProjektNr $\rightarrow$ Projektleiter,
AbtNr $\rightarrow$ Abtleiter,
AngNr $\rightarrow$ Leistung,
Tarif, Leistung $\rightarrow$ Gehalt}

Ergebnis der Relationensynthese:
Angestellte (<u>AngNr</u>, Name, Tarif, Leistung, AbtNr)
Projekte (<u>ProjektNr</u>, Projektleiter)
Abteilungen (<u>AbtNr</u>, Abtleiter)
Lohn (<u>Tarif, Leistung</u>, Gehalt)
Projektmitarbeit (<u>AngNr, ProjektNr</u>)

11.5 Ergänzungen zum algorithmischen Entwurf

Das (algorithmische) Resultat der Relationendekomposition oder der Relationensynthese sollte stets einer kritischen Prüfung unterzogen werden.

Beispiele für sinnvolle "Nachbesserungen von Hand":

- weitere Zerlegung von BCNF-Relationen:
  Angestellte (ANr, Informatikkenntnisse, Kinder) mit F = ∅
  Beispielausprägung:

ANr	Informatikkenntnisse	Kinder
1	Leda	Sabrina
1	Oracle	Sabrina
2	Leda	Pierre
2	Leda	Sabrina

  sollte weiter zerlegt werden in
  AngInfKenntnisse (ANr, Informatikkenntnisse) und
  AngKinder (ANR, Kinder)
  → Theorie der mehrwertigen Abhängigkeiten (Multivalued Dependencies)
  → 4. Normalform (4NF)

- unnötig feine Zerlegungen:
  Produkte (PNr, Bez, Preis)
  mit F = { PNr → Bez,
  PNr → Preis}
  sollte nicht zerlegt werden in
  Produktbez (PNr, Bez) und
  Produktpreise (PNr, Preis)

- Zerlegung aufgrund irrelevanter Funktionalabhängigkeiten:
  Bestellungen (Datum, KNr, PNr, Bez, Preis, Gewicht, Menge, Summe)
  mit F = { Datum, KNr, PNr → Menge,
  PNr → Bez, Preis, Gewicht,
  Preis, Menge → Summe}
  sollte nicht zur folgenden Relation führen
  Preise (Preis, Menge, Summe)

- Nicht-BCNF-Relationen, die bezüglich Redundanz und potentieller Inkonsistenz unproblematisch sind:
  Kunden (KNr, Name, Postleitzahl, Stadt, Strasse)
  mit F = { KNr → Name, Postleitzahl, Stadt, Strasse,
  Postleitzahl → Stadt }
  muß nicht notwendigerweise zerlegt werden in
  Kundenadressen (KNr, Name, Postleitzahl, Strasse) und
  Postleitzahlenverzeichnis (Postleitzahl, Stadt)

- M:N-Relationships ohne Attribute:
  Sitze (FlugNr, Datum, SitzNr)
  muß bei der Relationendekomposition u.U. manuell hinzugefügt werden (falls diese Information relevant ist)

Ergänzende Literatur zu Kapitel 11:

D. Maier, The Theory of Relational Databases, Computer Science Press, 1983

H. Mannila, K.-J. Raiha, The Design of Relational Databases, Addison-Wesley, 1992

C. Fleming, B. von Halle, Handbook of Relational Database Design, Addison-Wesley, 1988

P. Atzeni, C. de Antonellis, Relational Database Theory, Benjamin Cummings, 1992