3 Das Datenschema

Beim Entwurf eines konzeptionellen Datenschemas geht es um die Beschreibung eines bestimmten Ausschnittes aus der realen Welt und damit um eine Modellbildung. Durch die Modellbildung wird dieser Weltausschnitt vereinfacht, diskretisiert, idealisiert, andererseits aber auch für eine systematische Darstellung zugänglich gemacht. Dazu ist eine übersichtliche Methode erforderlich, mittels welcher die Elemente dieses Ausschnitts möglichst einfach, problembezogen und dennoch präzis beschrieben werden können. Technische Randbedingungen etwa der Typ des im Moment benützten Computer -oder Datenbanksystems sollen dabei keine Rolle spielen.
Hier wird eine Entwurfsmethode benützt und auch erläutert, welche auf wohlbekannten Konzepten (ER-Modell, Relationenmodell) aufbaut, deren Vorteile übernimmt, sie miteinander kombiniert und nach Möglichkeit vereinfacht. Daraus ergibt sich eine einfache, effiziente und trotzdem vollständige Entwurfsmethode für das konzeptionelle Datenschema.

Entwurf des Datenschemas

Diese Entwurfsmethode umfasst folgende Hauptkomponenten.

► **Entitätenblockdiagramme: Eine vereinfachte Form des Entity – Relationship Modells (ER-Modell).**

► **Relationen bzw. Tabellen: Das klassische Relationenmodell.**

► **Die Kombination von Entitätenblockdiagrammen und Relationenmodell zu einem globalen Konzept.**

►**Die Umsetzung und Einbettung dieser Komponenten in konkrete Schritte einer Entwurfsmethode.**

►**Die Beobachtung von Strukturregeln, um ein „global normalisiertes“ Datenschema zu erreichen.**

Entitäten und Entitätsmengen

Dieser Unterabschnitt zeigt, wie *für* eine spätere datenmässige Darstellung eines Ausschnitts der realen Welt vorerst wichtige Elemente dieser Welt (Personen, Objekte, Strukturen) einheitlich als *Entitäten* angesprochen werden.

Definition:
Eine ***Entität (entity)*** Ist ein individuelles Exemplar von Elementen der realen oder der Vorstellungswelt. Sofern eine Beziehung zwischen Entitäten eine eigenständige Bedeutung in der realen oder in der Vorstellungswelt hat, kann auch ein individuelles Exemplar einer solchen Beziehung als Entität aufgefasst werden.

3 Das Datenschema

Beispiele:
Heidi Meier
die Ausgabe der „Neuen Zürcher Zeitung“ vom 14April 1978 ein Jahrgang der Zeitschrift „Feld und Wald“
die Ehe von Heidi und Hans

Durch die Einführung des Begriffs Entität wird die Welt diskretisiert. Dabei muss in jedem einzelnen Fall entschieden werden, wie fein diese Diskretisierung erfolgen soll. Die Beschreibung einer Entität erfolgt im allgemeinen durch Merkmale (gleichbedeutend: Attribute), wobei deren Werte angegeben werden.

Ein wesentlicher Schritt bei der Modellbildung, d.h. bei der Abstraktion von konkreten Sachverhalten, besteht in der Bildung von Entitätsmengen.

Definition:
Eine *Entitätsmenge* (entity set) ist eine *Gruppierung* von Entitäten mit gleichen oder ähnlichen Merkmalen, aber unterschiedlichen Merkmalswerten.

Beispiele:
Angestellte der Firma X
Einzelnummern der Zürcher Tageszeitungen
Jahresbände von Zeitschriften in der ETH-Bibliothek
Eheschlüsse auf dem Standesamt in Wil
Arbeitsverhältnisse zwischen der Firma X und ihren Angestellten.

Im Entitätenblockdiagramm - das ist die graphische Darstellung im hier entwickelten Datenbank-Entwurfssystem - werden Entitätsmengen als Kästchen dargestellt.

Darstellung von Entitätsmengen

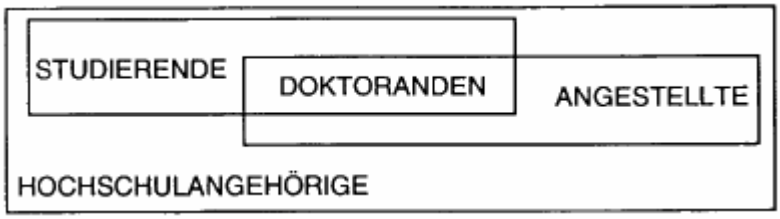
Werden mehrere Entitätsmengen betrachtet und lassen sich alle vorhandenen Entitäten genau einer dieser Entitätsmengen zuordnen, spricht man von ***disjunkten Entitätsmengen***. Das ist häufig der Fall, denn ANGESTELLTE, ZEITUNGSNUMMERN und EHEN sind selbstverständlich disjunkt.

Wichtig wird die Frage nach der gleichzeitigen Zugehörigkeit einer Entität zu mehreren Entitätsmengen dort, wo Gruppenbildungen gleichartiger Entitäten möglich sind, also etwa bei verschiedenen Personengruppen wie KINDER, SCHUELER, FUSSBALLSPIELER usw. Diese können disjunkt sein, sie können sich aber auch überlappen. Für unsere Modellierung der realen Welt ist das eine recht wichtige Unterscheidung (später mehr). Hier soll vorerst nur ein Beispiel einer Überlappung vorgestellt werden.

3 Das Datenschema

Definition:
Überlappende Entitätsmengen ergeben sich, wenn Entitäten gleichzeitig mehr als einer Entitätsmenge angehören.

Beispiel:
An einer Hochschule überlappen sich die Entitätsmengen STUDIERENDE und ANGESTELLTE: Doktoranden sind oft sowohl Studierende als auch Angestellte.



Überlappende und umfassende Entitätsmengen

Bei überlappenden Entitätsmengen kann immer eine neue Entitätsmenge definiert werden, welche die überlappenden Mengen gesamthaft umfasst (im Beispiel die Entitätsmenge HOCHSCHULANGEHORIGE).

Beziehungen zwischen Entitätsmengen

Dieser Unterabschnitt begründet die Darstellung von quantitativen Zusammenhängen zwischen Entitätsmengen. Zur Beschreibung einer Beziehung zwischen zwei Entitätsmengen EM1 und EM2 kann man von gerichteten Assoziationen (EM1, EM2) ausgehen.

Definition:
Eine ***Assoziation*** (EM 1, EM2) legt fest, wieviele Entitäten aus EM2 einer Entität aus EM1 zugeordnet sein können.

Die Anzahl ist in vielen Fällen in einem gewissen Bereich variabel und kann darin mit Unter- und Obergrenze angegeben werden.
(Beispiel: In einer Primarschule gilt: Assoziation (Lehrer, Schüler) = 15 .. 28, was als „15 bis 28“ zu lesen ist). Wir wollen im folgenden aber ganz grob nur 4 Typen von Assoziationen unterscheiden, die beim Datenbank-Entwurf von zentraler Bedeutung sind, und diese mit den Symbolen 1, c, m und mc bezeichnen.

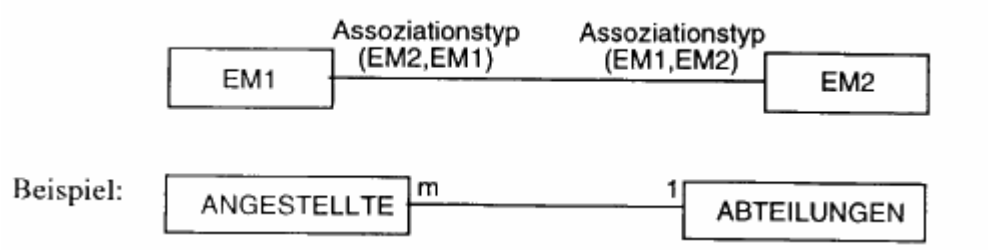
3 Das Datenschema

	Assoziationstyp (EM1, EM2):	Entitäten aus EM2, die jeder Entität aus der Menge EM1 zugeordnet sind:
1:	einfache Assoziation	genau eine
c:	konditionelle Assoziation	keine oder eine (c = 0/1)
m:	multiple Assoziation	mehrere (m >=1)
mc:	multipel-konditionelle Assoziation	keine, eine oder mehrere (mc>=0)

Vier Assoziationstypen

Definition:
Kombiniert man eine Assoziation (EM1, EM2) mit ihrer Gegen - Assoziation (EM2, EM1), so ergibt dies die ***Beziehung (relationship)*** zwischen den beiden betrachteten Entitätsmengen.

Zur Darstellung von Beziehungen zwischen zwei Entitätsmengen EM1 und EM2 werden die Kästchen durch eine Linie verbunden und die Assoziationstypen 1, c, m oder mc an deren Enden angeschrieben:



Darstellung von Beziehungen in Entitäten-Blockdiagramm

Das Beispiel ist wie folgt zu verstehen:
Zu jedem Angestellten (=Entität aus der Entitätsmenge ANGESTELLTE) existiert genau eine („1“) Abteilung, der dieser zugeordnet ist. Umgekehrt gibt es zu jeder einzelnen Abteilung „m“ zugehörige Angestellte. Diese 1—m-(„1 zu m“)-Beziehung kann auch mit ihrer Bedeutung benannt werden, sofern dies nötig und nicht selbstverständlich ist, hier etwa die „Abteilungszugehörigkeit“. In besonderen Fällen lautet der Name der Beziehung sogar unterschiedlich je nach Blickrichtung (etwa „Elternschaft“ und „Nachkommenschaft“).

In der folgenden Tabelle werden alle überhaupt möglichen Beziehungstypen unserer groben Klassierungsweise aufgezählt. Bei 4 Assoziationstypen ergeben sich zwar 16 mögliche Typen von Beziehung zwischen 2 Entitätsmengen, von denen jedoch nur 10 echt verschieden sind.

3 Das Datenschema

Entitätsmenge 1	Entitätsmenge 2	Beziehungstyp	Beziehung
linke Schuhe	rechte Schuhe	1–1	Paare
Abteilungen	Angestellte	c–1	Abteilungsleiter
Angestellte	Abteilungen	m–1	Abteilungszugehörigkeit
Kinder	Ehepaare	mc–1	Familienzugehörigkeit
Frauen	Männer	c–c	Ehe
Personen	Parteien	m–c	Parteizugehörigkeit
Projekte	Projekte	mc–c	ist Unterprojekt
Standorte	Standorte	m–m	Distanz
Vorlesungen	Studenten	mc–m	Einschreibung
Personen	Personen	mc–mc	Freundschaften

Beispiele von Beziehungen (ohne Symmetrien)

Entitätsmengen und ihre Beziehungen bilden die dominierende Struktur von Datenbeständen; sie bilden das sog. Entitätenblockdiagramm.

3 Das Datenschema

Das ER-Modell (Entity Relationship Modell)

Definition:
Eine *Entität* (entity) ist ein „Ding". welches eindeutig Identifizierbar ist.

Beispiele:
Eine bestimmte Person, ein Unternehmen, ein Ereignis.

Definition:
Eine *Beziehung* (relationship) ist eine Zuordnung zwischen Entitäten.
Beispiele:
„Vater-Sohn" und „Ehe" sind beides bestimmte Beziehungen zwischen „Person"-Entitäten

Definition:

Die *Rolle (role)* einer Entität in einer Beziehung ist die Funktion, die sie in einer Beziehung ausübt.

Beispiele:
„Ehemann" und „Ehefrau" sind Rollen von „Personen" in einer „Ehe"

Das ER-Modell hat inzwischen als logisches Datenmodell weite Verbreitung gefunden und wurde in dieser oder ähnlicher Form auch in verschiedene Datenbankentwurfsmethoden eingebaut.

Jede „Beschreibung der realen Welt" mit Hilfe eines (informatikbezogenen) Datenmodells stößt rasch an Grenzen der Komplexität. Dass gilt ganz besonders dann, wenn auch inhaltliche, sog. semantische Aspekte der Realität mitbeschrieben werden sollen. Die ER-Modelle (samt Entitätenblockdiagramm) bieten hier einen pragmatischen Ansatz, indem sie eine Grobbeschreibung ermöglichen, die noch überblickbar ist, aber bestimmte Feinheiten vernachlässigt. Daher finden sie in der Praxis gerade für den konzeptionellen Schemaentwurf Anwendung.

Attribute, Wertebereiche, Formatierung

In diesem Abschnitt werden einige grundlegende Begriffe definiert, welche zur datenmässigen Beschreibung von Entitätsmengen benötigt werden.

Definition:
Ein *Wertebereich (domain)* ist eine Menge von verschiedenen Datenwerten desselben skalaren Datentyps.

3 Das Datenschema

Beispiele:
- ‘MO‘,DI‘,‘MI‘,‘DO‘,‘FR‘,‘SA‘,‘SO‘ (= Wochentage)
- 1..23
- ganze Zahlen eines bestimmten Rechners (integer)
- Zeichen des ASCII-Zeichensatzes
- ASCII-Zeichenfolgen der Länge 10

Definition:
Ein *Attribut* ist einerseits die Beschreibung einer bestimmten Eigenschaft der Entitäten einer Entitätsmenge. Anderseits definiert das Attribut (implizit) auch die Rolle, die der zugehörige Wertebereich in der Entitätsmengen-Beschreibung spielt.

Beispiel:
Das Attribut Lohnklasse beschreibt einerseits eine Eigenschaft in der Entitätsmenge PERSONEN und definiert anderseits die Rolle, die der Wertebereich 1..23 als „Lohnklasse“ in der Beschreibung der Entitätsmenge PERSONEN spielt.

„Attribut“ und „Merkmal“ werden hier als synonyme Begriffe verwendet; den Begriff Attribut benützt man v.a. im Zusammenhang mit dem Relationenmodell.

Formatierte und unformatierte Beschreibungen, Datentypen

Datenbanksysteme unterstützen besonders effizient die *formatierte* Darstellung von Attributen. Das bedeutet, dass die Beschreibung eines bestimmten Attributs ausschließlich durch die Angabe einer oder mehrerer erlaubter Ausprägungen (Werte) aus dem zulässigen *Wertebereich* dieses Attributs erfolgt. In Programmiersprachen werden die Werte von Variablen im Rahmen von sog. *Typendeklarationen* auf bestimmte Wertebereiche beschränkt. Attribute mit Wertebereichen entsprechen somit Datentypen.

Beispiele:
Lohnklasse: 17
Wirtesonntag: “DI“
Kinder: “Felix“, “Regina“

Ein Wertebereich kann wenige oder viele Werte umfassen sowie eng oder tolerant definiert sein. Als Beispiel einer toleranten Definition betrachten wir den Wertbereich für *Familiennamen.* Dieser Wertebereich lässt sich kaum als „Liste zulässiger Namen“ definieren. Was würde denn „zulässig“ überhaupt heißen? Auch Namen wie „Giscard dEstaing“ und „Mao Tse Tung“ müssen akzeptiert werden können. Daher ist wohl eine tolerante Definition wie „25 Zeichen“ für Namen durchaus angemessen. Damit lassen sich leider Namen wie ‘XXXX‘ und QQPS2 nicht automatisch ausschließen. Namen müssen jedoch bei der Dateneingabe sowieso noch anders geprüft werden. Ein verwechselter, aber an sich korrekter Name kann nämlich auch mit dem aufwendigsten Namenprüfsystem nicht erkannt werden (MEYER‘ statt ‘MEIER‘).

3 Das Datenschema

Bei *unformatierter* Beschreibung eines Attributs wird dieses als Text (oder allgemeiner: als Bitfolge) betrachtet, dessen Bedeutung nur im Kontext verstanden werden kann.

Beispiele:

Besondere Merkmale:	‘Narbe über rechtem Auge, spricht fließend Englisch
Zusammenfassung:	‘The paper describes a new language for structured
Messwerte:	‘17.9 18.2 17.8 16.9 17.9 ……..
freie digitale Daten:	Figuren, Fotos, Videos, Tonsequenzen, Multimedia [Meyer 91]

Unformatierte Attribute können beliebige Zeichen- oder Bitfolgen enthalten. Viele moderne Datenbanksysteme erlauben die Speicherung von sog. „langen“ Attributen für Texte oder Bitfolgen (fast) unbeschränkter Länge. Solche Attribute sind aber unbrauchbar zur Darstellung von Beziehungen und relativ schlecht geeignet für Suchprozesse (obwohl die *Volltextsuche,* d.h. das Suchen nach einem bestimmten Wort in einem Text, in der Praxis zunehmend an Bedeutung gewinnt und in den Funktionsangeboten großer DBMS, etwa im „Oracle Universal Server“ bereits enthalten ist.) Unformatierte Attribute können überall dort zweckmäßig sein, wo eine formatierte (und damit relativ starre) Organisation nicht möglich oder nicht sinnvoll ist. Dies ist etwa der Fall für die Behandlung von Sonderfällen (Bsp. oben: „besondere Merkmale“), aber auch bei nicht - strukturierten oder speziellen Daten (z.B. Texte, Bilder). Die Aus-wertung unformatierter Attributwerte erfolgt dann nicht mit den üblichen Datenbank-operationen, sondern entweder manuell oder mit besonderen Programmen (z.B. mit Volltextsuchsystemen auf Texten und Bildanalyseprogrammen auf Bildern [Schäuble 97]).

Verwaltungsdaten, vor allem ihr Grundbestand, sind normalerweise in formatierter Form vorhanden (Formulare, Karteikarten schon in manuellen Systemen), vieles kann jedoch unformatiert sein. Sehr viele Dokumente (von Rechtssammlungen über Korrespondenzen bis zu Plänen) sind nämlich nicht im Sinne der Datenbanktechnik strukturiert und lassen sich daher nur unformatiert speichern.

Inzwischen werden zunehmend formatierte und unformatierte Daten in einem einzigen System verwaltet; so lässt sich etwa in einem Personalregister gleich auch ein Passbild der eingetragenen Personen mitspeichern. Dabei erfolgt das Kennzeichnen und auch das Aufsuchen der Datensätze primär über die formatierten Attribute, während die unformatierten Attribute (z.B. Dokumente, Bilder) nur präsentiert werden können. Speicherplatzmäßig kann dabei der unformatierte Teil sehr wohl dominieren.

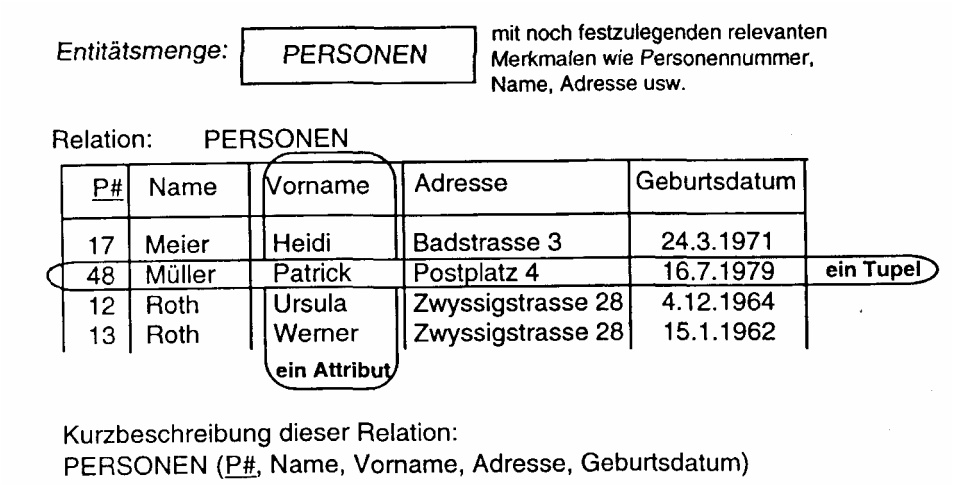
3 Das Datenschema

Das Relationenmodell

In diesem Abschnitt steigen wir von der gesamtstrukturorientierten Ebene der Entitätsmengen und Beziehungen auf die bereits „datennähere“ Ebene der Relationen herunter. Es geht in diesem Abschnitt um die Grundbegriffe des ursprünglichen Relationenmodells nach Codd [Codd 70]. Das erweiterte Relationenmodell kommt danach zur Darstellung.

Relationen

Relationen im Sinne des Relationenmodells sind zweidimensionale Tabellen. Solche Relationen eignen sich sehr gut zur datenmässigen Darstellung von Entitätsmengen. Jede Entitätsmenge wird durch eine Relation repräsentiert; jede einzelne Entität entspricht einer Zeile der Relation (ein „Tupel“) und jedes ihrer Attribute (= Merkmale) einer Spalte der Relation.



Entitätsmenge und zugehörige Relation (= Tabelle)

Das Attribut hat immer zwei verschiedene Aufgaben, es benennt eine bestimmte Eigenschaft von Entitäten und es verbindet diese mit einem bestimmten Wertebereich. Uber den Wertebereich lässt sich die Relation mengentheoretisch definieren.

Grundlage für diese Definition sind n Wertebereiche W1, die nicht notwendigerweise disjunkt sein müssen (d.h. dieselben Werte dürfen in mehreren Wertebereichen vorkommen).

Def.: Eine Relation R ist eine Teilmenge des kartesischen Produkts von n Wertebereichen:

$$R \subset W_1 \times W_2 \times \ldots \times W_n$$

3 Das Datenschema

Das "kartesische Produkt" $W_1 \times W_2 \times W_n$ bedeutet dabei die Menge aller möglichen Wertekombinationen in den Tupeln $(w_1, w_2, .., w_n)$.

Beispiel: Der Wertebereich W_1 gehört zum Attribut „Geschlecht" und ist eine zweiwertige Menge {M, F} (für Mann/Frau)

Der Wertebereich W_2 gehört zum Attribut „Zivilstand" und ist eine vierwertige Menge {le, vh, vw, gs} (für ledig/verheiratet/verwitwet/geschieden)

Das kartesische Produkt $W_1 \times W_2$ ist wiederum eine Menge *(die sog. Potenzmenge) und enthält 8 Tupel, nämlich alle 8 maximal möglichen Kombinationen von W_1 und W_2:

$$W_1 \times W_2 = \{ \begin{matrix} M, le, & M, vh, & M, vw, & M, gs, \\ F, le, & F, vh, & F, vw, & F, gs \} \end{matrix}$$

Jede beliebige Untermenge aus dieser Menge ist eine zulässige Relation auf W_1 und W_2.

Zwei wichtige Konsequenzen folgen gleich aus der Definition der Relation; die eine betrifft die Reihenfolge der Tupel, die andere deren Einzigartigkeit.

Weil jede Relation als Menge definiert ist, sind ihre Tupel grundsätzlich ungeordnet. Eine (mathematische) Menge umfasst Elemente, zwischen denen vorerst keine Ordnung oder Reihenfolge besteht.
(Verlangt ein Benutzer die Tupel in einer bestimmten Reihenfolge, so muss diese zuerst erzeugt werden.)

Weil in einem kartesischen Produkt jede mögliche Wertekombination nur einmal vorkommt und jede Relation eine Teilmenge eines kartesischen Produktes ist, sind alle Tupel einer Relation voneinander verschieden (keine Doubletten möglich!). Daher ist die folgende Definition zur obenstehenden gleichwertig:

Def.: Eine *Relation* ist eine Menge von unterschiedlichen Tupeln der Form

$(w_1, w_2, ..., w_n)$ mit $w_i \in W_i$ (w_i = Element von W_i)

Der Begriff "Tupel" ist übrigens eine Abkürzung von "n-tupel", eine Verallgemeinerung von "Paar, Tripel, Quadrupel, ..". Jede Zeile einer Relation bildet ein solches Tupel und repräsentiert eine Entität.
Die gesamte Relation repräsentiert eine Entitätsmenge. Und aus (beliebig vielen) Relationen setzt sich nun die gesamte Datenbasis zu sammen:

3 Das Datenschema

Def.: Eine *Datenbasis* besteht aus Relationen.

Der Zusammenhang zwischen mehreren Relationen einer Datenbasis (d.h. die allgemeine Beschreibung ihrer Beziehungen untereinander) wird weiter unten detailliert behandelt.

Identifikationsschlüssel

Def.: Ein *Identifikationsschlüssel* ist ein Attribut oder eine *minimale* Attributskombination,
- die jedes Tupel einer Relation identifiziert und
- deren Wert sich während der Existenz des Tupels nicht ändert.

Ein zusammengesetzter Schlüssel ist dann minimal, wenn ohne Verlust der eindeutigen identifizierbarkeit kein Attribut der Attributskombination weggelassen werden kann.

Beispiel: In der Relation PERSONEN kann die Attributskombination "P#, Name, Vorname" die Tupel eindeutig identifizieren, sie ist aber nicht minimal. "P#" allein genügt zur Identifikation und entspricht der neuen, restriktiveren Definition.

Für den Datenbankentwurf ist die Festlegung eines Identifikationsschlüssels für jede Relation (und damit auch für die zugehörige Entitätsmenge) von zentraler Bedeutung. Gelegentlich bieten sich mehrere Attribute oder Attributskombinationen als Kandidaten für Identifikationsschlüssel an; das bringt für den Datenbankentwurf keine Probleme. Unsere erste Strukturregel für Datenbanken besagt nur, dass für Relationen a priori ein Identifikationsschlüssel festgelegt werden muss:

(SR 1) Bei der Darstellung von Entitätsmengen durch Relationen muss für jede Rotation ein Identifikationsschlüssel existieren.

Sprechende Schlüssel sind durchaus erlaubt, doch muss darauf geachtet werden, dass die zweite Forderung der obigen Definition nicht verletzt wird (Unveränderlichkeit!). Unter Umständen ist es deshalb notwendig, zusätzliche Schlüsselattribute einzuführen: **künstliche Schlüssel**. Bei der formalen Beschreibung wird im folgenden jeweils der gewählte Identifikationsschlüssel in der Liste der Attribute zur Verdeutlichung unterstrichen.

3 Das Datenschema

Abhängigkeiten

Abhängigkeiten bezeichnen Zusammenhänge zwischen verschiedenen Attributen einer Relation. Verschiedene Arten von Abhängigkeiten werden hier vorerst ohne weitere Begründung eingeführt. Ihre Bedeutung wird anschließend im nächsten Unterabschnitt "Der Normalisierungsprozess" sichtbar und dort diskutiert.

Def.: *Funktionale Abhängigkeit:* Das Attribut bzw. die Attributskombination B ist funktional abhängig vom Attribut bzw. von der Attributskombination A derselben Relation R, wenn zu einem bestimmten Wert von A höchstens ein Wert von B möglich ist.

Die funktionale Abhängigkeit gibt an, welche Attributswerte fest mit anderen Attributswerten gekoppelt sind.

Beispiel: Wird eine Person mit einer Personennummer identifiziert, so sind Name, Vorname usw. dieser Person mit deren Personennummer gekoppelt. Damit ist in der Relation PERSONEN das Attribut PNm funktional abhängig von P#.

Die Rolle des Identifikationsschlüssels kann formal mittels der funktionalen Abhängigkeit folgendermassen beschrieben werden:

Def.: *Identifikationsschlüssel:* Der Identifikationsschlüssel ist ein Attribut oder eine Attributskombination, so dass gilt: Jedes Attribut einer Relation ist vom Identifikationsschlüssel funktional abhängig, und kein Attribut aus dem Identifikationsschlüssel ist von den übrigen Attributen des Identifikationsschlüssels funktional abhängig.

Für die Untersuchung der Bedeutung einzelner Attribute einer Relation benötigt man weitere Abhängigkeitsbegriffe. Da in unserem Entwurfsverfahren lediglich die Abhängigkeiten vom Identifikationsschlüssel einer Relation benötigt werden, schränken wir bereits die Definitionen auf diese Situation ein.

Def.: Seien A ein zusammengesetzter Identifikationsschlüssel und B *ein Attribut oder eine Attributskombination der gleichen* Relation R.

B ist genau dann *voll abhängig* von A, wenn B von der Attributskombination A funktional abhängig ist (R.A → R.B), aber nicht bereits von Teilen der Attributskombination A.

Beispiel: Das Attribut Pj-P-Zeit ist voll abhängig von der Attributskombination (P#,Pj#).

"Voll abhängig" kann auch so umschrieben werden: Das voll abhängige Attribut bezeichnet eine spezifische Eigenschaft genau dieser Entitätsmenge.

Def.: A sei der Identifikationsschlüssel einer Relation R. B und C seien zwei weitere Attribute oder Attributskombinationen von R derart, dass die drei Attribute (Attributskombinationen) untereinander je distinkt sind. C ist *transitiv abhängig* von A, falls jederzeit gilt:

$R.A \rightarrow R.B \quad R.B \not\rightarrow R.A$ ($\not\rightarrow$ bedeutet „*nicht* funktional abhängig")

$R.B \rightarrow R.C$

Beispiel: In der Relation PERSONEN (P#,PNm,Abt#,AbtNm) ist AbtNm transitiv abhängig von P#, da Abt# funktional von P# und AbtNm funktional von Abt# abhängt, während P# nicht funktional von Abt# abhängt.

"Transitiv abhängig" ist also gleichbedeutend mit "abhängig auch über schlüsselfremde Umwege".

Diese Definitionen von Abhängigkeiten werden nun anschliessend für den sog. Normalisierungsprozess verwendet.

3 Das Datenschema

Der Normalisierungsprozess

Dieser Abschnitt zeigt anhand des Normalisierungsprozesses gemäß klassischem Relationenmodell, wie schon aus einer genauen Analyse der inneren Datenzusammenhänge irmerhalb einer Relation wesentliche Zusammenhänge erkannt, vor allem aber Redundanzen eliminiert und damit Mutationsanomalien vermieden werden können.

Redundanz ist in einem Datenbestand genau dann vorhanden, wenn ein Teil des Bestandes ohne Informationsverlust weggelassen werden kann. Dieser weglassbare Teil heißt entsprechend "redundante Daten". Redundanz beansprucht Speicherplatz und Verwaltungsaufwand im Computersystem; das ist aber nicht das Ausschlaggebende für die Vermeidung von Redundanz. Redundante Datenbestände müssen nämlich bei allen Mutationen korrekt mitmutiert werden.
Wird dies aus irgendeinem Grund nicht oder unvollständig gemacht, entstehen inhaltliche Differenzen und damit Widersprüche innerhalb der Datenbasis; es kommt zu sog. Mutationsanomalien, einer großen Gefahr für die Konsistenz einer Datenbasis.

Beim konzeptionellen Datenentwurf sollen daher Redundanzen im Datenbestand frühzeitig erkannt und soweit möglich eliminiert werden. Im konzeptionellen Schema ist Redundanz von Daten grundsätzlich unerwünscht. (Auf der physischen Ebene ist das anders; hier sind redundante Datenbestände als Sicherheitskopien und für Zugriffsbeschleunigungen durchaus notwendig. Diese Redundanz wird aber systematisch zugefügt und bildet daher keine Gefahr für Mutationsanomalien. Solche absichtlichen Redundanzerhöhungen werden noch genauer behandelt.

Zur Erkennung von Redundanzen innerhalb einer Relation müssen primär die Abhängigkeiten zwischen den Attributen untersucht werden. Diese Abhängigkeiten zeigen, welche Attributswerte fest mit anderen Attributswerten gekoppelt sind. Daraus lässt sich die Normalisierung einzelner Relationen ableiten. Ziel der Normalisierung einer Relation ist es, die Attribute allenfalls derart auf mehrere Relationen aufzuteilen, so Redundanzen verschwinden.

Der klassische Normalisierungsprozess nach Codd läuft über mehrere sog. Normalformen, die im folgenden schrittweise am Beispiel "Fertigungsbetrieb" vorgestellt werden.

Def.: Eine *Relation* befindet sich in *1. Normalform*, wenn ihre Attribute nur einfache Attributswerte aufweisen.

Eine Relation ist immer eine Tabelle. Diese Definition der 1. Normalform liefert daher geradezu eine Kurzdefinition des Begriffs Relation, sofern noch das Verbot von Duplikattupeln zugefügt wird. Diese Definition ist wesentlich leichter lesbar als der mengentheoretische Ansatz.

3 Das Datenschema

Durch die 1. Normalform werden die Relationen formatiert: Es werden nur einfache (sog. skalare) Attribute zugelassen; Mengen werden als Attribut ausgeschlossen.

Beispiel: Man kann unter dem Attribut "Kinder" nicht mehrere Namen aufführen.

Die 1. Normalform impliziert aber auch, dass die Attribute keine innere Struktur haben dürfen bzw. dass eine auffällige innere Struktur nicht ausgenützt werden kann. Wenn beispielsweise als ein Attribut "Geburtsdatum" eingeführt wird, so muss der Wert "9.9.1999" als Ganzes betrachtet werden. Natürlich lässt sich daraus der Geburtsmonat ausrechnen, aber "Geburtsmonat" ist damit noch kein eigenes Attribut.

Nun soll der Normalisierungsprozess anhand einer mit Datenwerten gefüllten Tabelle begründet und erläutert werden. Die Daten passen zum Beispiel "Fertigungsbetrieb".

PERSONEN

P#	PNm	Abt#	AbtNm	Pj#	PjNm	P-Pj-Zeit
101	Hans	1	Physik	11,12	A,B	60,40
102	Rolf	2	Chemie	13	C	100
103	Urs	2	Chemie	11,12,13	A,B,C	20,50,30
104	Paul	1	Physik	11,13	A,C	80,20

Tabelle, nicht in 1. Normalform

Die obige Fig. beschreibt vier Personen aus der Menge PERSONEN und ist für den menschlichen Leser sehr wohl lesbar. Dennoch ist diese Tabelle keine Relation in erster Normalform, da Pj#, PjNm und P-Pj-Zeit keine einfachen Attribute sind (man beachte die Mehrfacheinträge).

Diese Tabelle muss deshalb umgeschrieben werden, indem der Inhalt der Tupel mit Mehrfacheinträgen auf mehrere Tupel aufgeteilt wird. Dabei kann sich allerdings die Bedeutung der einzelnen Tupel und damit auch der ganzen Relation ändern.

Der Informationsgehalt der gesamten Relation bleibt bei dieser Transformation völlig unverändert.

3 Das Datenschema

P-PJ-TÄTIGKEITEN

P#	PNm	Abt#	AbtNm	Pj#	PjNm	P-Pj-Zeit
101	Hans	1	Physik	11	A	60
101	Hans	1	Physik	12	B	40
102	Rolf	2	Chemie	13	C	100
103	Urs	2	Chemie	11	A	20
103	Urs	2	Chemie	12	B	50
103	Urs	2	Chemie	13	C	30
104	Paul	1	Physik	11	A	80
104	Paul	1	Physik	13	C	20

Relation in 1. Normalform

Die Fig. zeigt jetzt eine definitionsgerechte Relation (in 1. Normalform). Die Attributskombination (P#, Pj#) ist Identifikationsschlüssel dieser Relation.

Man sieht nun sofort, dass die Relation Redundanzen enthält: So ist der Personenname mit der Personalnummer gekoppelt (=abhängig) und müsste eigentlich nicht für jedes Projekt wiederholt werden. Das kostet Platz und Arbeitsaufwand und es könnte eine sog. Mutationsanomalie auftreten: Ändert der Name der Person mit P# = 101 (z.B. von Hans zu John), wird die Relation widersprüchlich, wenn die Namensänderung nur bei einem und nicht bei allen Tupeln mit P# = 101 ausgeführt wird. Das Problem liegt offensichtlich darin, dass die Relation gleichzeitig verschiedenartige Sachverhalte (Personalien, Abteilungszugehörigkeit und Projektaspekte) beschreibt, die voneinander unabhängig sind und zu unterschiedlichen Zeitpunkten ändern können. Diese verschiedenen Sachverhalte sollten auch in verschiedenen Relationen dargestellt werden (Aufspaltung).

Def.: Eine Relation ist in *2. Normalform,* wenn sie in 1. Normalform ist und jedes nicht zum Identifikationsschlüssel gehörige Attribut *voll* von diesem abhängig ist.

Die 2. Normalform erzwingt eine Gruppierung der Attribute einer Relation nach Sachgebieten und eliminiert dadurch Redundanzen.

Beispiel: Die Relation P-PJ-TÄTIGKEITEN (P#, PNm, Abt#, AbtNm, Pj#, PjNm, P-Pj-Zeit) mit dem Schlüssel (P#, Pj#) ist nicht in 2. Normalform, weil die Attribute PNm, Abt#, AbtNm, PjNm von diesem Schlüssel nicht voll abhängig sind. (Der Leser prüfe dies anhand der Definition der vollen Abhängigkeit nach!

Für den 2. Normalisierungsschritt ist die Relation deshalb aufzuspalten in die folgenden drei Relationen:

PERSONEN

P#	PNm	Abt#	AbtNm
101	Hans	1	Physik
102	Rolf	2	Chemie
103	Urs	2	Chemie
104	Paul	1	Physik

PROJEKTE

Pj#	PjNm
11	A
12	B
13	C

P-PJ-TÄTIGKEITEN

P#	Pj#	P-Pj-Zeit
101	11	60
101	12	40
102	13	100
103	11	20
103	12	50
103	13	30
104	11	80
104	13	20

Datenbasis in 2. Normalform

Die Gesamtheit der drei Relationen in obiger Fig. hat den gleichen Informationsgehalt wie die eine Relation in der Fig. davor.
Die Verknüpfung der Relationen erfolgt implizit durch korrespondierende Attribute, sog. globale Attribute.
Die Relationen PERSONEN und P-PJ-TATIGKEITEN sind z.B. über das in beiden Relationen vorkommende Attribut P# verbunden. Daraus ergibt sich, dass zwischen Attributen, die über eine Relation hinaus von Bedeutung sind, und solchen, die nur innerhalb einer Relation eine Rolle spielen, zu unterscheiden ist. Diese Unterscheidung geschieht weiter unten (globale, lokale Attribute), wobei dann gewisse Mischfälle auszuschliessen sind.

Ist damit alle Redundanz eliminert? In obiger Fig. enthalten nun die Relationen PROJEKTE und P-PJ-TATIGKEITEN keine Redundanz mehr. Aber in der Relation PERSONEN steckt noch Redundanz drin:
Hier ist für jede Person der Abteilungsname gespeichert, obwohl dieser Name mit der Abteilungsnummer gekoppelt ist. Ändert der Abteilungsname, könnten noch immer Mutationsanomalien auftreten. Um dies zu verhindern, ist ein 3. Normalisierungsschritt nötig.

Def.: Eine Relation befindet sich in *3. Normalform,* wenn sie in 2. Normalform ist und kein Attribut, das nicht zum Identifikationsschlüssel gehört, transitiv von diesem abhängt.

Die 3. Normalform bringt somit ein weiteres Kriterium zum Aufspalten von Relationen

und eliminiert damit beim 2. Normalisierungsschritt noch verbliebene Redundanzen von Attributen.

Beispiel: Die Relation PERSONEN (P#, PNm, Abt#, AbtNm) ist zwar in 2., nicht aber in 3. Normalform, weil das Attribut AbtNm über Abt# transitiv vom Schlüssel P# abhängt (und umgekehrt auch Abt# über AbtNm).

Eine Aufspaltung der Relation PERSONEN liefert zwei Relationen PERSONEN und ABTEILUNGEN in 3. Normalform.
Relationen in 3. Normalform heißen oft auch kurz „normalisiert".

PERSONEN

P#	PNm	Abt#
101	Hans	1
102	Rolf	2
103	Urs	2
104	Paul	1

ABTEILUNGEN

Abt#	AbtNm
1	Physik
2	Chemie

PROJEKTE

Pj#	PjNm
11	A
12	B
13	C

P-PJ-TÄTIGKEITEN

P#	Pj#	P-Pj-Zeit
101	11	60
101	12	40
102	13	100
103	11	20
103	12	50
103	13	30
104	11	80
104	13	20

Datenbasis in 3. Normalform

Def.: *Schlüsselkandidat:* Attribut oder minimale Attributskombination, welche jedes Tupel einer Relation identifiziert. („Minimal" heisst, dass im Fall einer Attributskombination kein Attribut daraus weggelassen werden kann, ohne dass die verbleibenden Attribute zur Identifikation nicht mehr ausreichen.)

3 Das Datenschema

Beziehungen zwischen Relationen – globale Normalisierung

In diesem Abschnitt wird unser Datenbank-Entwurfsverfahren so erweitert, dass Beziehungen zwischen Entitätsmengen direkt mit der Normalisierung des Relationenmodells in Zusammenhang gebracht werden (,,globale Normalisierung").

Global- und Lokalattribute

Eine wesentliche Leistung des klassischen Relationenmodells nach Codd ist die Elimination von Redundanz in einer Relation bei allen Attributen, welche nicht Teil eines Identifikationsschlüssels sind. Die Untersuchung der funktionalen Abhängigkeit bewirkt nämlich über die Normalisierung, dass solche Attribute redundanzfrei gespeichert werden.

Die klassische Normalisierung vermag aber Redundanzen dann nicht zu erkennen und zu eliminieren, wenn diese zwischen zwei und mehr Relationen bestehen. Der Datenbankentwurf ist aber sehr häufig mit dieser Situation konfrontiert:
Der Datenbankingenieur kennt meist bereits zu Beginn seiner Arbeit eine Vielzahl von Entitätsmengen bzw. von entsprechenden Relationen. Und zwischen diesen Entitätsmengen bzw. Relationen bestehen möglicherweise vielfache Beziehungen. Diese werden aber nur indirekt durch "gleiche" Attribute in verschiedenen Relationen ausgedrückt, d.h. durch Attribute mit gleichem Wertebereich und gleicher Bedeutung, oft (aber nicht immer) auch mit gleichem Namen, etwa eine Personennummer P#.

Somit geht es nun darum, Redundanzen auch zwischen beliebigen Relationen zu erkennen und zu eliminieren. Das ist nicht mehr ein relationsinternes, sondern ein globales Problem in der Datenbasis.

Dazu führen wir vorerst die Begriffe der globalen bzw. lokalen Attribute ein.

Def.: Ein Attribut heisst *global,* wenn es mindestens in einer Relation im Identifikationsschlüssel vorkommt.

Def.: Ein Attribut heisst *lokal,* wenn es nur in einer einzigen Relation und dort *nicht* im Identifikationsschlüssel vorkommt.

Nun lassen sich in einer Datenbasis auch Attribute denken, welche weder global noch lokal sind. Das ist etwa dann der Fall, wenn die gleiche Eigenschaft einer Entität in zwei verschiedenen Relationen beschrieben wird. Diese Situation tritt bei überlappenden Entitätsmengen auf. Wir betrachten dazu nochmals das vorgestellte Beispiel aus dem Hochschulbereich :
"Doktoranden" können gleichzeitig "Studierende" und "Angestellte" sein. Wenn nun Name und Adresse dieser Doktoranden sowohl in der Relation "Studierende" als auch in der Relation "Angestellte" mitgeführt werden, führt das zu Redundanz und Mutationsanomalien, obwohl alle Relationen einzeln in 3. Normalform sind.

Dieses Problem lässt sich durch die Einführung einer übergeordneten, umfassenden Entitätsmenge lösen, wobei die gemeinsamen Merkmale (hier also Namen und Adresse) nicht den überlappenden Entitätsmengen, sondern einzig der übergeordneten Relation zugeordnet werden. Diese Maßnahme ist auf der Ebene der Relationen gleichwertig mit der "Elimination von Attributen, die weder global noch lokal sind".

Beispiel: Die beiden Relationen
STUDIERENDE (S#, Nm, Adr, Fakultät)
ANGESTELLTE (M. Mm, Adr, Lohnklasse)

werden in folgende Relationen übergeführt:

HOCHSCHULANGEHÖRIGE (P#, Nm, Adr)
STUDIERENDE (P#, S#, Fakultät)
ANGESTELLTE (P#, A#, Lohnklasse)

Die Vorgehensweise lässt sich in einer zweiten Strukturregel zusammenfassen:

(SR 2) Die Datenbasis muss aus Relationen in dritter Normalform bestehen, welche nur Global- und Lokalattribute enthalten.

Durch das Konzept der Global- und Lokalattribute kann auch auf globaler Ebene Redundanzlosigkeit erzwungen werden.
Sämtliche Beziehungen zwischen verschiedenen Relationen und somit auch zwischen verschiedenen Entitätsmengen werden ausschliesslich über Globalattribute hergestellt. Lokalattribute beschreiben zusätzliche Eigenschaften und zwar ausschliesslich in einer einzigen Relation.

Fremdschlüssel, dynamische Wertebereiche

Wenn die Zahl der Entitätsmengen / Relationen in einer Datenbasis größer wird, geht leicht der Überblick darüber verloren, ob die zwischen den Entitätsmengen bestehenden Beziehungen durch die Globalattribute der Relationen richtig oder eventuell mehrfach oder widersprüchlich wiedergegeben werden. Der Grund für diesen unbefriedigenden Sachverhalt liegt darin, dass im klassischen Relationenmodell beim Normalisierungsprozess (beginnend in 1. Normalform) ausschließlich einzelne Relationen betrachtet und redundanzfrei gemacht werden. Das Konzept der globalen Attribute führt etwas weiter, schließt Konsistenz-verletzungen allerdings noch keineswegs aus.

Beispiel: Offensichtlich hat in der Relation P-PJ-TÄTIGKEITEN ein Tupel mit einer bestimmten Wertekombination (P#,Pj#) nur dann einen Sinn, wenn in den Relationen PERSONEN bzw. PROJEKTE je ein entsprechendes Tupel mit Identifikationsschlüssel P# bzw. Pj# existiert. So kann in Fig. 3.11 das Tupel (105, 12, 30) nicht in die Relation P-PJ- TÄTIGKEITEN eingefügt werden, weil in PERSONEN keine Person mit P#= 105 existiert, obwohl im übrigen alle Attributswerte den entsprechenden Wertebereichen entsprechen würden.

Diese Überlegungen sind nahe verwandt mit jenen, die wir bei der Einführung der Assoziationen und Beziehungen zwischen Entitätsmengen in Abschnitt 3.3 gemacht haben. Auf der Ebene der Entitätsmengen kennen wir die "1-..-Beziehungen". Wir präzisieren:

Def.: Eine *H-Beziehung (hierarchische Beziehung)* enthält eine Assoziation vom Assoziationstyp 1.
(1-...-Beziehung Assoziation (EM2,EM1) = 1)

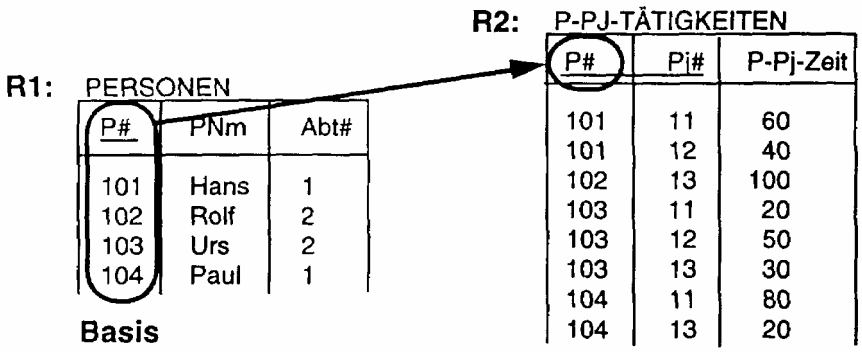
Das heißt in der Sprache der Entitäten: ‚jede Entität in der zweiten Entätsmenge hat zu genau einer Entität in der ersten Entitätsmenge eine Beziehung."

Nun wechseln wir zur Sprache der Relationen. Hier geht es analog um Tupel und ihre Identifikationsschlüssel sowie um zulässige Wertebereiche für Attribute.
Wir benötigen jetzt zusätzlich eine Formulierungsmöglichkeit für Wertebereiche, welche vom aktuellen Inhalt einer anderen Relation abhängig sind (Bsp. "Wertebereich, bestehend aus allen Werten von Personalnummern P#, die als Identifikationsschlüssel der aktuell in PERSONEN vorhandenen Tupel vorkommen."). Dazu dient der Begriff des Fremdschlüssels.

Def.: Ein *Fremdschlüssel (foreign key)* in einer Relation R2 ist ein Attribut (oder Attributskombination), welches dem Identifikationsschlüssel in einer andern Relation R1 entspricht und dessen zugehöriger Wertebereich die Menge genau jener Werte umfasst, welche die aktuell in R1 vorhandenen Tupel identifizieren. R1 heisst *Basisrelation* des Fremdschlüssels.

3 Das Datenschema

Beispiel: In der Relation P-PJ-TÄTIGKEITEN ist P# (aus PERSONEN) Fremdschlüssel (Fig.3.13), aber auch Pj# (aus PROJEKTE).



Der Fremdschlüssel P# in R2 basiert auf dem Identifikationsschlüssel in R1

Mit diesem Sprachelement des Fremdschlüssels lassen sich jetzt Beziehungen zwischen Entitätsmengen auf der Ebene der Relationen präzis ausdrücken. Für Attribute stehen jetzt zwei Arten von Wertebereichen zur Verfügung:

Jeder dynamische Wertebereich umfasst zu jedem Zeitpunkt genau so viele Werte (oder Wertkombinationen bei zusammengesetztenSchlüsseln) wie Tupel in der Basisrelation des Fremdschlüssels existieren.

Beispiele für statische Wertebereiche:

1..24, ('MO', 'DI', 'MI', 'DO', 'FR', 'SA', 'SO')

Beispiele für dynamische Wertebereiche:

Alle Werte 101, 102, 103 und 104 des Identifikationsschlüssels P# der Relation PERSONEN in Fig. 3.13.

Alle Wertkombinationen (101,11), (101,12) usw. des Identifikationsschlüssels (Pj#,P#) in der Relation P-PJ-TÄTIGKEITEN in Fig. 3.13 (etwa zur Nutzung als Fremdschlüssel in einer weiteren Relation).

Mit einem Fremdschlüssel lässt sich nun die H-Beziehung *zwischen Relationen* definieren:

(SR 3)
- Lokal-Attribute müssen statische Wertebereiche verwenden.
- Jedes Global-Attribut darf nur in einer einzigen Relation auf einem statischen Wertebereich basieren und muss in dieser Relation Identifikationsschlüssel sein. In allen anderen Relationen muss es auf einem dynamischen Wertebereich basieren, d.h. als Fremdschlüssel aus einer anderen Relation eingebracht werden.

3 Das Datenschema

Normalisierung im Entitätenblockdiagramm

Wir wollen nun mit Hilfe der Erkenntnisse über Globalattribute und Fremdschlüssel konkrete Datenstrukturen aufbauen und darstellen.
Dabei soll soweit wie möglich die übersichtliche (weil die Attribute ausblendende) Darstellungsform des Entitätenblockdiagramms verwendet werden. Erfreulicherweise lässt sich der ganze Normalisierungsprozess vollständig auf der Stufe des Entitätenblockdiagramms abwickeln; die Attribute müssen dazu überhaupt nicht herbeigezogen werden.

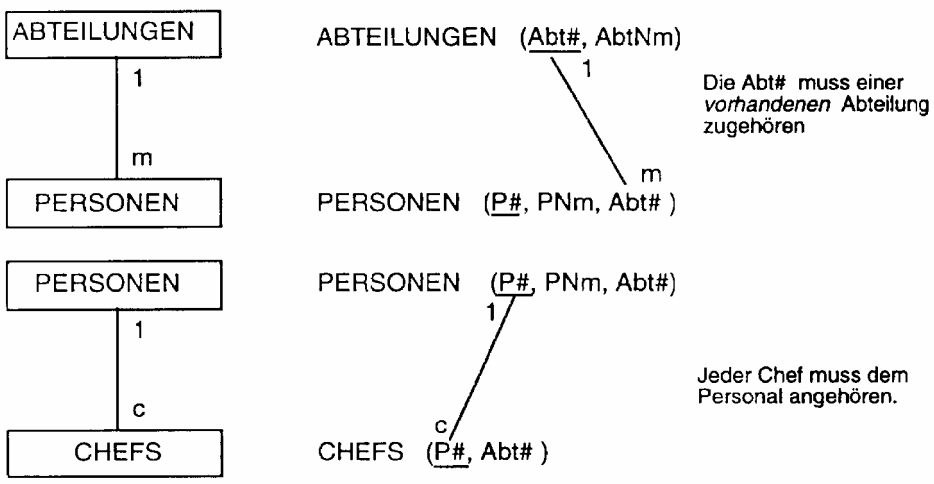
Wir beginnen dazu mit der Untersuchung der verschiedenen Beziehungstypen. Es wurde darauf hingewiesen, dass ausgehend von den vier Assoziationstypen (1,c,m,mc) zwischen zwei Relationen (ohne Berücksichtigung der Symmetrie) 16 verschiedene Beziehungstypen möglich sind; Tab. 3.14 zeigt sie alle.

	1	c	m	mc	
1	1–1	c–1	m–1	mc–1	hierarchische Beziehung
c	1–c	c–c	m–c	mc–c	konditionelle Beziehung
m	1–m	c–m	m–m	mc–m	netzwerkartige Beziehung
mc	1–mc	c–mc	m–mc	mc–mc	

Tabelle 3.14: Beziehungstypen zwischen Relationen

Einige dieser Beziehungstypen, und zwar genau die hierarchischen, lassen sich nun direkt mit Hilfe von Fremdschlüsseln und dynamischen Wertebereichen darstellen; in diesem Fall sind keine zusätzlichen Maßnahmen und keine zusätzlichen Entitätsmengen nötig. Die Beziehungen vom Typ 1-1, 1-c, 1-m und 1-mc sind unmittelbar H- Beziehungen; die Beziehungen vom Typ c-1, in-1, mc-1 müssen bloß von hinten gelesen werden. Fig. 3.15 zeigt links die Entitätsmengenbeziehung, rechts die entsprechende H-Beziehung zwischen Relationen, realisiert durch einen Fremdschlüssel für ein bestimmtes Globalattribut. (Man beachte, dass das am Fremdschlüssel beteiligte Attribut in der hierarchischen Vaterrelation immer Identifikationsschlüssel ist, in der Sohnrelation aber Schlüssel- oder Nichtschlüsselattribut sein kann.)

3 Das Datenschema

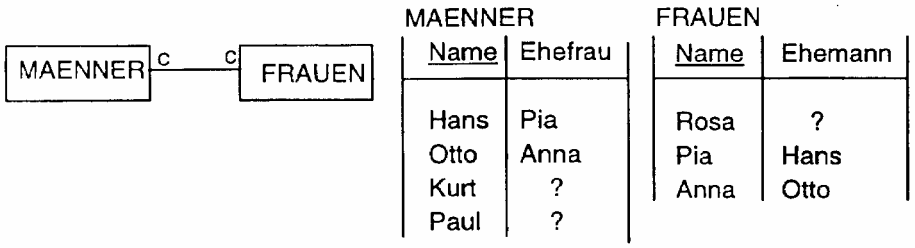


Figur 3.15: Zwei von vornherein hierarchische Beziehungen

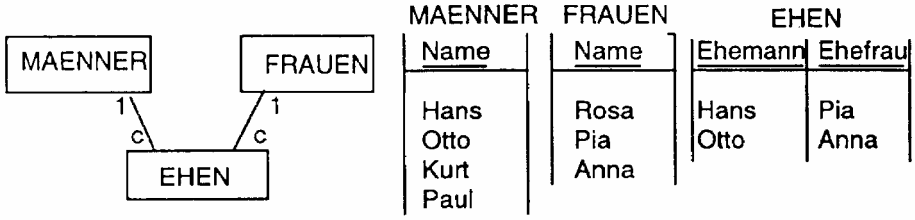
Bei den konditionellen Beziehungstypen ist die Einführung einer zusätzlichen Relation und parallel dazu einer zusätzlichen Entitätsmenge zur Darstellung der Beziehung notwendig, weil sonst den verbindenden Attributen in gewissen Fällen (c = 0) kein Attributswert zugeordnet werden könnte.

Konditionelle Beziehung vor und nach einer Transformation

Fig. 3.16 und Fig.3.l7 zeigen ein Beispiel. Für die bei der Transformation entstehende neue Relation EHEN (Fig. 3.17) werden zwei Fremdschlüssel verwendet: "EHEN.Ehemann" ist Fremdschlüssel bezüglich "MAENNER.Name", "EHEN.Ehefrau" bezüglich "FRAUEN.Name".



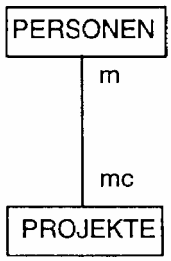
Figur 3.16: Konditionelle Beziehung *vor* Transformation



Figur 3.17: Konditionelle Beziehung *nach* Transformation (neu: 2 H-Beziehungen)

Zur Darstellung netzwerkartiger Beziehungen ist ebenfalls eine zusätzliche Relation notwendig. Auch hier lässt sich zwar die Information über die Beziehungen in den zwei Ausgangsrelationen unterbringen (Fig. 3.18), aber ebenfalls in unsauberer Form, indem eine der beiden Relationen (hier PERSONEN) mehr Tupel umfassen müsste, als "Personen" existieren. Diese Relation ist aber auch nicht in 2. Normalform und enthält somit Redundanz! Durch die Transformation mit Einführung einer zusätzlichen Relation und zweier H-Beziehungen lassen sich all diese Probleme sofort und korrekt lösen
(Fig.3.19).

Beispiel:



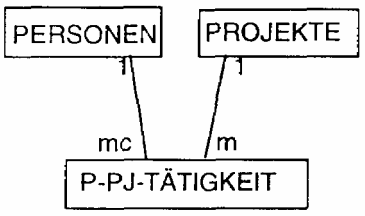
PERSONEN

P#	Pj#	P-Pj-Zeit	PNm
101	11	60	Hans
101	12	40	Hans
102	13	100	Rolf
103	11	20	Urs
103	12	50	Urs
103	13	30	Urs
104	11	80	Paul
104	13	20	Paul

PROJEKTE

Pj#	PjNm
11	A
12	B
13	C
14	D

Figur 3.18: Netzwerkartige Beziehung *vor* Transformation



PERSONEN

P #	PNm
101	Hans
102	Rolf
103	Urs
104	Paul

PROJEKTE

Pj #	PjNm
11	A
12	B
13	C
14	D

P-PJ-TÄTIGKEIT

P#	Pj #	P-Pj-Zt
101	11	60
101	12	40
102	13	100
103	11	20
103	12	50
103	13	30
104	11	80
104	13	20

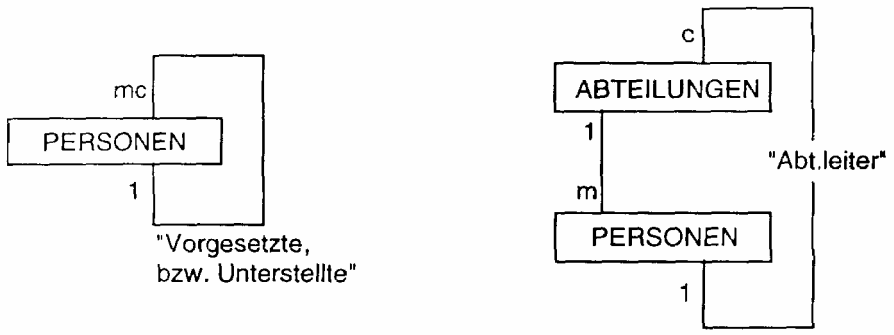
Figur 3.19: Netzwerkartige Beziehung *nach* Transformation
(neu: 2 H-Beziehungen)

Eine m/mc - m/mc Beziehung wird durch die Einführung einer zusätzlichen Beziehungsentitätsmcnge und der entsprechenden Beziehungsrelation in zwei H-Beziehungen 1 - m/mc transformiert. Wer diese Transformation in den Fig.3.18 und 3.19 mit dem 2.Normalisierungsschritt in 3.4.4 (Fig.3.9 und 3.10) vergleicht, stellt Übereinstimmung fest. Die Forderung nach voller Abhängigkeit der Nichtschlüsselattribute ist erst in Fig.3.19 erfüllt. Da über Beziehungen zwischen Globalattributen keine transitiven Abhängigkeiten entstehen können, führt die Transformation nach Fig.3.19 auf Relationen in 3. Normalform; sie sind normalisiert.

Auf analoge Weise lassen sich alle 16 Beziehungstypen (Tab. 3.14) so transformieren, dass das resultierende Beziehungsnetz ausschliesslich aus hierarchischen Beziehungstypen 1-1, 1-c, 1-m und 1-mc besteht; durch die Strukturregeln wird damit auch eine Halbordnung unter den Relationen erreicht. Diese können nun konsequent so präsentiert werden, dass immer die Relation auf der "1"-Seite oben steht (wie in den Fig. 3.15 - 3.19); dadurch wird das Lesen eines Datenschemas wesentlich erleichtert.

Auflösung von Rekursionen

Die globale Normalisierung muss nicht bloß nichthierarchische Beziehurigen durch zwei hierarchische ersetzen. Ein analoges Problem ergibt sich auch für rekursive hierarchische Beziehungen. Fig. 3.20 zeigt zwei solche Beispiele.



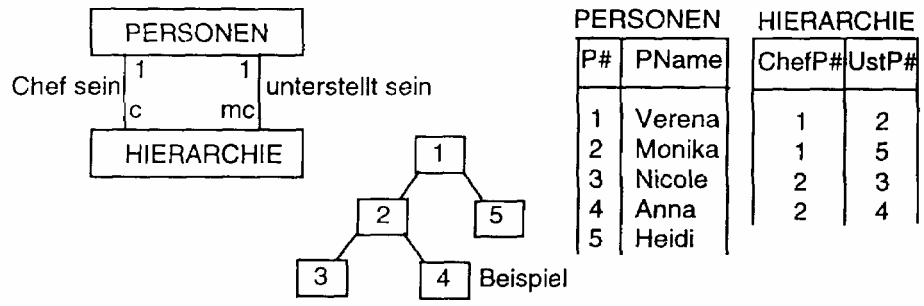
Figur 3.20: Verbotene rekursive Beziehungen

Beispiele: Jeder Chef einer Person ist wiederum eine Person.
(Direkte Rekursion, Fig.3.20 links)

Jede Person gehört zu einer Abteilung, jede Abteilung hat eine Person als Chef.
(Indirekte Rekursion, Fig.3.20 rechts)

Solche rekursiven " Beziehungsschleifen" sind im Entitätenblockdiagramm verboten. Sie können aber wiederum durch Einführung einer Beziehungsentitätsmenge/ Beziehungsrelation aufgelöst werden, welche die gewünschte Beziehung explizit beschreibt und damit gleichzeitig die verbotene Rekursion aufsprengt.
Fig.3.21 zeigt links die Strukturdarstellung mit HIERARCHIE als zusätzlicher Beziehungsrelation, in der Mitte ein Beispiel einer realen Hierarchie und rechts deren datenmässige Darstellung.

3 Das Datenschema



PERSONEN

P#	PName
1	Verena
2	Monika
3	Nicole
4	Anna
5	Heidi

HIERARCHIE

ChefP#	UstP#
1	2
1	5
2	3
2	4

Figur 3.21: Aufgelöste rekursive Beziehung

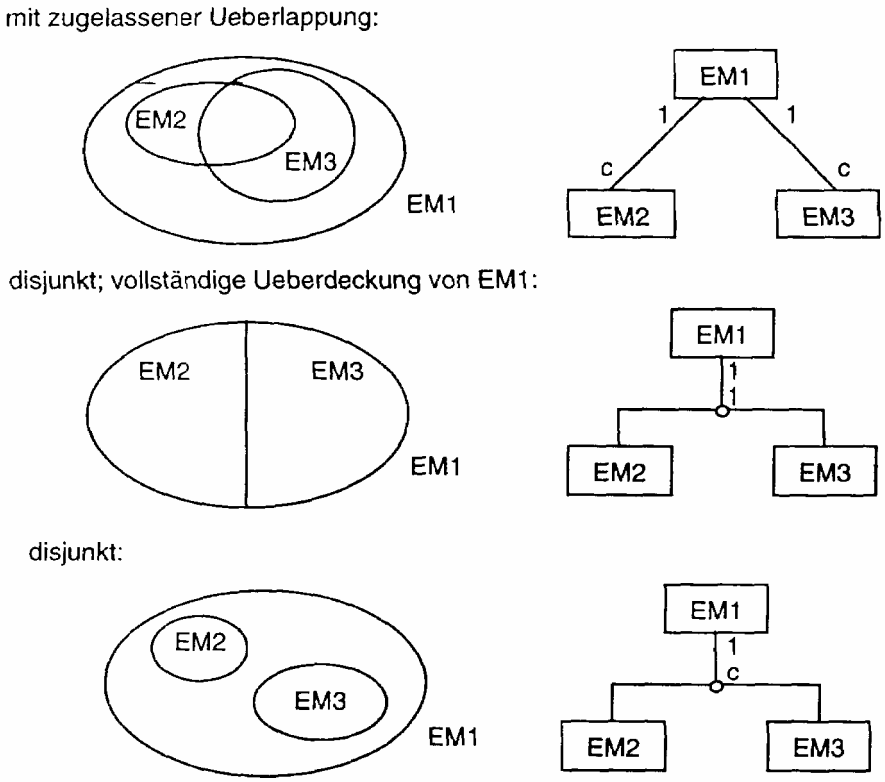
Es ist typisch für aufgelöste Hierarchien, dass aus der Beziehungsrelation gerade zwei Fremdschlüssel (hier ChefP# und UstP#=UnterstelltenP#) auf die gleiche Basisrelation (hier PERSONEN) zurückgreifen, allerdings in unterschiedlicher Bedeutung (Rolle), weshalb es sich empfiehlt, diese Bedeutung im Entitätenblockdiagramm direkt an der Beziehungslinie anzuschreiben. Wir verzichten hier darauf, auch die Auflösung der indirekten Rekursion (Fig.3.20 rechts) im einzelnen vorzustellen; das Ergebnis der Auflösung findet sich nämlich in Fig.3.28 oben links (die Beziehungsentitätsmenge heisst dort ABTEILUNGSLEITER). Auch in Fig.3.27 wird die Auflösung rekursiver Strukturen nochmals gezeigt, dort am Musterbeispiel "Stückliste".

(SR 4) Rekursive Beziehungen zwischen Relationen sind untersagt; ein Globalattribut in einer Relation darf nur mit einem solchen Fremdschlüssel gebildet werden, dessen Ausgangsrelation unabhängig von dieser Relation definiert werden kann.

Generalisierungen

Besonders interessante Beziehungen zwischen Relationen ergeben sich dort, wo die in den verschiedenen Entitätsmengen enthaltenen Entitäten gleichartig und zum Teil gar identisch sind. So sind etwa FESTANGESTELLTE und AUSHILFEN und KAUFLEUTE und MECHANIKER alle auch PERSONEN. Die PERSONEN bilden den Oberbegriff oder die Generalisierung, umgekehrt sind AUSHILFEN eine Spezialisierung von PERSONEN. Zwischen Ober- und Untermengen besteht grundsätzlich eine 1-c-Beziehung. Nun betrachten wir aber auch nebeneinander stehende Entitätsmengen. So sind FESTANGESTELLTE und AUSHILFEN disjunkte Entitätsmengen, während sich KAUFLEUTE und FESTANGESTELLTE überlappen.

Für die Modellierung der realen Welt ist es beim Datenbankentwurf oft wichtig, auch solche Querbeziehungen klar bezeichnen zu können. Dazu werden die Sprachelemente des Entitätenblockdiagramms noch geringfügig um eine Darstellungsmöglichkeit für disjunkte Entitätsmengen erweitert (Fig. 3.22).



Figur 3.22: Generalisierung / Spezialisierung bei unterschiedlicher Überlappung von EM2 und EM3 (EM1 ist die Obermenge)

Der Fall der Überlappung (Fig.3.22 oben) kann bereits mit den bisher bekannten Sprachelementen dargestellt werden; er entspricht Fig.3.2 und unterliegt der Strukturregel SR2; der Uberlappungsbereich kann allerdings leer bleiben. Wenn die Untermengen aber disjunkf sein niüssen (keine Uberlappung zulässig), lässt sich dies mit einem speziellen Sprachelement, der Gabellinie, angeben. Falls die Untermengen zusammen alle Entitäten der Obermenge umfassen müssen, wird der Gabelungspunkt mit 1 angeschrieben (Fig.3.22 Mitte), sonst mit c (Fig.3.22 unten). Mit diesem Sprachelement lassen sich manche Entitätenblockdiagramme wesentlich präziser strukturieren.

(SR 5) Allfällige Unter- und Obermengenbeziehungen zwischen Relationen sind so präzis wie möglich festzuhalten. Die Zuordnung einer Entität zu disjunkten spezialisierten Untermengen wird durch ein diskriminierendes Attribut in der generalisierten Relation ausgedrückt.

3 Das Datenschema

Zum Abschluss dieses Abschnittes über Beziehungen zwischen Relationen sei noch eine 6. Strukturregel formuliert, worin es um eine globale Aussage über mehrstufige (indirekte) Beziehungen zwischen Entitätsmengen geht. Sie kann daher erst anhand eines grösseren Beispiels in Abschnitt 3.7 unter "Bereinigung der Konsistenzbedin-Gungen“ erläutert werden:

(SR 6) Die Globalattribute einer Relation, die nicht auf statischen Wertebereichen basieren, sind als Fremdschlüssel auf jenen Relationen zu basieren, welche die grösstmögliche Einschränkung des zulässigen Wertebereichs erlauben.

Eine Datenbasis, deren Relationen allen sechs Strukturregeln entsprechen, heißt global normalisiert.