Helmut Jarosch

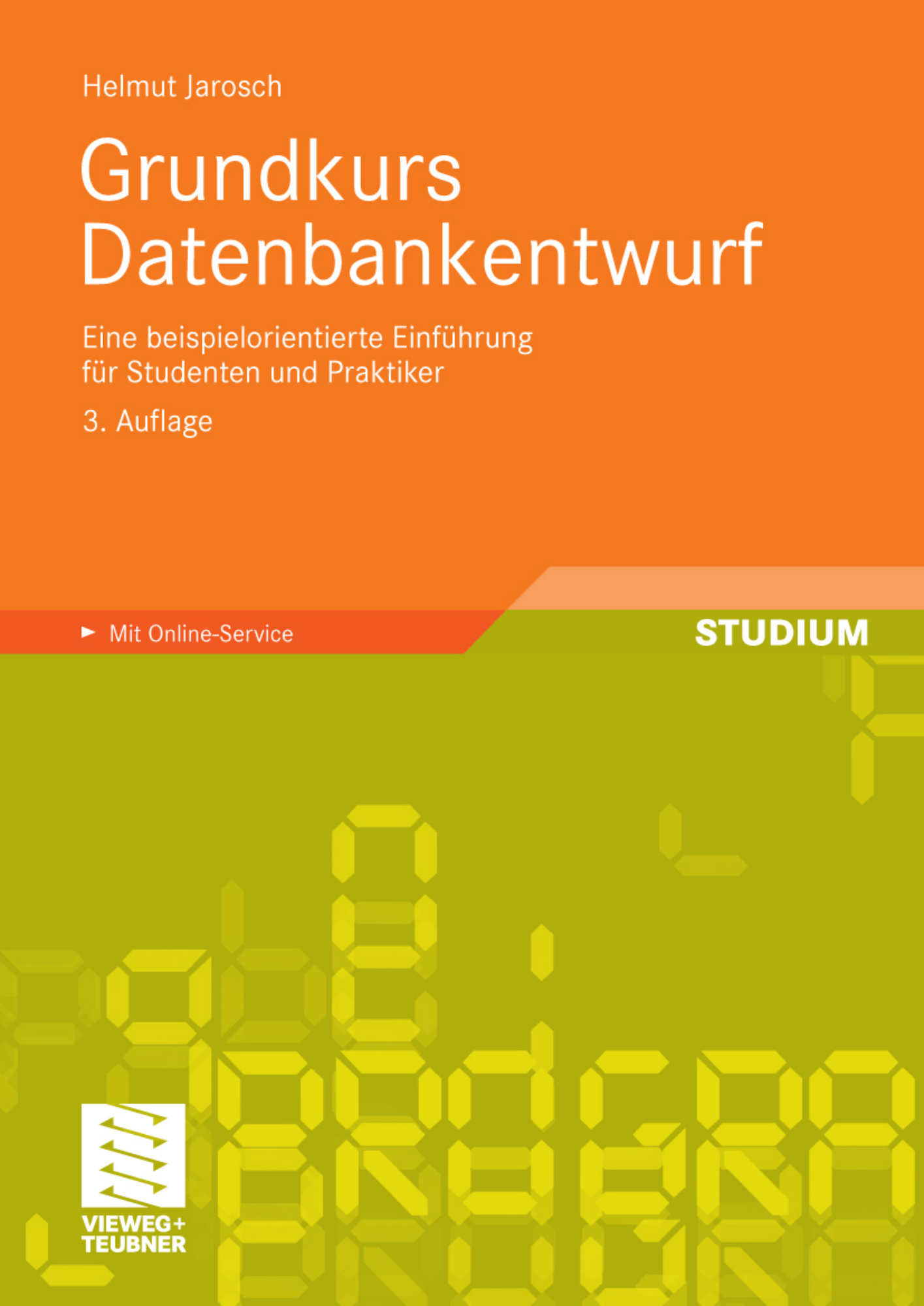
Grundkurs Datenbankentwurf

Eine beispielorientierte Einführung
für Studenten und Praktiker

3. Auflage

► Mit Online-Service

STUDIUM



VIEWEG+
TEUBNER

Leserstimmen zur 2. Auflage:

„/.../ sorgfältig ausgewählt und zusammengestellt. Eine leicht verständliche und gut strukturierte Abhandlung des Themas. Empfehlenswert."

Prof. Dr. Bernhard Bürg, FH Karlsruhe

„/.../ ein absolut lohnenswertes Buch."

Prof. Dr. Stephan Kleuker, FH Nordakademie

„/.../ zeigt einen verständlichen Weg zum Datenbankentwurf mittels KML. Sehr anschaulich sind die vielen Beispiele."

Irene Wehfritz, Rudolf-Diesel-FS für Techniker, Nürnberg

„Mit Sicherheit eines der besten Bücher zum Thema."

Prof. Dr. Robert Senger, FH Karlsruhe

„Gelungene Einführung in die Datenbanktechniken, auch für Studenten, die nicht aus der Informationstechnik kommen, geeignet. Dieses Buch empfehle ich meinen Studenten."

Prof. Dr. Stefan Hesse, FB Eletrotechnik/FH München

„/.../ sehr verständlich – sehr gelungenes Buch – ich werde es meinen Studenten empfehlen."

Prof. Dr. Britta Rösch, FH Bingen

„/.../ ausgezeichnet – sehr anschaulich – hohe Praxisrelevanz mit Fallbeispielen."

Prof. Dr.-Ing. Herbert Fischer, FH Deggendorf

„/.../ sehr gut lesbar; verständlich genau."

Prof. Dipl.-Math. Edzard de Buhr, FH Wilhelmshaven

„/.../ ausgefeilte Darstellung – übersichtlich, gute Struktur – dieses Buch werde ich meinen Studenten empfehlen, weil es eine umfassende Darstellung auf angemessenem Niveau ist und ausgezeichnete Beispiele enthält."

Prof. Dr. Wolftgang Riggert, FH Flensburg

„/.../ umfassende Behandlung des Themas – klar und verständlich – als beispielorientierte Einführung zum Datenbankentwurf empfehlenswert."

Prof. Dr. Georg Ohmayer, FH Weihenstephan

„/.../ gute Darstellung des DB-Entwurfs – angenehm zu lesen."

Prof. Dr. Achim Gottscheber, FH Heidelberg

„Endlich einmal ein Buch zur Datenmodellierung, mit dessen Hilfe Studierende begreifen, wie Datenbanken zu entwerfen sind."

Dr. Oliver Braun, Uni Saarbrücken

Helmut Jarosch

Grundkurs Datenbankentwurf

Eine beispielorientierte Einführung
für Studenten und Praktiker

3., überarbeitete und erweiterte Auflage

Mit 211 Abbildungen und 14 Tabellen

STUDIUM

VIEWEG+
TEUBNER

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der
Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über
<http://dnb.d-nb.de> abrufbar.

Höchste inhaltliche und technische Qualität unserer Produkte ist unser Ziel. Bei der Produktion und Auslieferung unserer Bücher wollen wir die Umwelt schonen: Dieses Buch ist auf säurefreiem und chlorfrei gebleichtem Papier gedruckt. Die Einschweißfolie besteht aus Polyäthylen und damit aus organischen Grundstoffen, die weder bei der Herstellung noch bei der Verbrennung Schadstoffe freisetzen.

1. Auflage 2002
2. Auflage 2003
3., überarbeitete und erweiterte Auflage 2010

Lektorat: Christel Roß | Walburga Himmel

Vieweg+Teubner ist Teil der Fachverlagsgruppe Springer Science+Business Media.
www.viewegteubner.de

Umschlaggestaltung: KünkelLopka Medienentwicklung, Heidelberg
Druck und buchbinderische Verarbeitung: MercedesDruck, Berlin
Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier.
Printed in Germany

ISBN 978-3-8348-0955-1

Meiner lieben Frau Sabine
für ihre Geduld während des Schreibens dieses Buches

Helmut Jarosch

Vorwort

Die Notwendigkeit des Aufbaus von Datenbanken hat durch die zunehmende Verbreitung datenbankgestützter interaktiver Webseiten einen starken Aufschwung erlangt. Die ständig zunehmende Akzeptanz des Entwurfs relationaler Datenbanken mit Hilfe von konzeptionellen Datenmodellen verstärkt den Bedarf nach Lehrbüchern, die den Prozess des Datenbankentwurfs mit Hilfe des Entity-Relationship-Modells ausführlich und anschaulich beschreiben.

Der Absatzerfolg der ersten beiden Auflagen dieses Lehrbuchs machte deshalb eine überarbeitete und erweiterte Neuauflage erforderlich.

Dieses Buch wendet sich an *Studierende* und an *Praktiker*, die moderne Informationssysteme nicht nur benutzen, sondern auch ihre Wirkprinzipien verstehen wollen. Es soll sie insbesondere dazu befähigen, ihr fachliches Wissen über die Daten, die in einem speziellen Umfeld benötigt werden, in exakter Weise so aufzubereiten, dass es unmittelbar für den Datenbankentwurf verwendet werden kann. Das Buch ist somit für Informatiker, Wirtschaftsinformatiker, Betriebswirte und sonstige Fachexperten geschrieben, die als Haupt- oder Nebenakteure mit dem Entwurf datenbankbasierter Anwendungssysteme zu tun haben.

In diesem Buch werden die folgenden Probleme behandelt:

- Wie wird ein *Datenbank-Anwendungssystem* entwickelt und welche Rolle spielt dabei die Datenmodellierung?
- Wie gelangt man von der Analyse der Realität zu deren Darstellung in einem *Datenmodell*?
- Welche Möglichkeiten zur Strukturierung der Daten bieten die gängigen *Datenbank-Modelle*?
- Wie erfolgt die *Transformation* eines Datenmodells in die Struktur der Datenbank?

Da sich dieses Buch als eine *Einführung in die Praxis des Datenbankentwurfs* versteht, setzt es keine speziellen Fachkenntnisse voraus. Es wird besonderer Wert auf gute Verständlichkeit

gelegt, die durch eine große Zahl anschaulicher Beispiele gefördert wird. An Hand eines *durchgehenden Beispiels* werden alle Etappen von der Beschreibung der Realität bis hin zur Festlegung der Datenbankstruktur ausführlich besprochen. Außerdem wird jedes Element zur Beschreibung von Datenstrukturen durch ein treffendes Anwendungsbeispiel veranschaulicht, das jeweils einem speziellen Unternehmensbereich oder aber dem Alltagsleben entnommen ist.

Als Titel für dieses Lehrbuch hatte ich ursprünglich vorgesehen: *„Datenbankentwurf für Fußgänger"*. Was sollte dieser eigenartige Buchtitel bedeuten? Dieses Buch ist nicht für Leser geschrieben, die es besonders eilig haben, denen ein flüchtiger Blick auf die Dinge genügt, denen es nicht darauf ankommt, richtig Bescheid zu wissen, sondern die nur „mitreden" wollen.

Im Tourismus kennt man dieses Problem zur Genüge. Touristen werden mit Großstädten wie London oder Berlin während einer Sightseeing-Tour in 2 bis 3 Stunden bekannt gemacht. Kennen sie dann die Stadt? Rundreisen durch einen ganzen Kontinent wie Australien werden in 17 Tagen absolviert. Was weiß der Reisende dann über „Land und Leute"?

Bei einem Lehrbuch steht man vor demselben Problem. Soll man den Leser im Eiltempo durch das darzustellende Wissensgebiet jagen? Er wird dann von Vielem wenig wissen. Noch schlimmer: Er wird glauben, Ahnung von Sachverhalten zu haben, von denen er doch nur „eine Ahnung" hat.

Keiner soll glauben, der Aufbau eines Datenmodells und seine Transformation in eine Datenbank-Struktur seien in einfacher Weise zu bewerkstelligen. Es handelt sich dabei um komplexe Probleme, die nicht dadurch einfacher werden, dass man sie vereinfacht darstellt. Wer sich ernsthaft mit dem Entwurf einer Datenbank beschäftigen möchte, braucht einfach das gesamte dafür nötige Rüstzeug. Bekommt er in einem Lehrbuch nur vereinfachende Bruchstücke vermittelt, muss er ständig zwischen diesen Wissensfragmenten interpolieren und läuft dabei Gefahr, Interpolationsfehler zu begehen.

Dieses Lehrbuch ist natürlich nicht das erste Buch zu diesem Thema (und sicherlich auch nicht das letzte). Was also will der Autor anders machen als die Autoren vor ihm? Er will den Leser nicht im „Reisebus" auf der Überholspur durch das Thema führen, sondern möchte mit ihm als *„Fußgänger"* die Landschaft im Schritt-Tempo erkunden. Der Leser wird sich am Ende der Wan-

derung, wenn er wieder alleingelassen wird, im durchstreiften Gebiet besser und sicherer bewegen können.

Das Schritt-Tempo der Führung ist aber nicht nur für denjenigen ein Gewinn, der sich zum ersten Mal in neuem Terrain aufhält, sondern auch für den wiederholten Besucher, ja selbst für den Reiseleiter. Er entdeckt beim „Schlendern" Details, die auch er zuvor nicht gesehen hat, und zieht Ansichten in Zweifel, die ihm bisher selbstverständlich schienen. In diesem Sinne ist das Buch nicht nur für *„Einsteiger"* geschrieben, sondern könnte auch für *„alte Hasen"* von Interesse sein.

Fünf umfangreichere *Aufgaben zum Datenbankentwurf* sollen den Leser in die Lage versetzen, die gewonnenen Erkenntnisse praktisch zu erproben. Der Vergleich mit den ausführlich kommentierten Lösungen ermöglicht eine persönliche Wissenskontrolle und macht das Buch auch für das Selbststudium geeignet.

Das Buch ist als Ergebnis meiner Lehre entstanden, die ich an der *Hochschule für Wirtschaft und Recht Berlin* auf dem Gebiet des Datenbankentwurfs durchgeführt habe. Ich danke der Hochschule für Wirtschaft und Recht Berlin für die Unterstützung während des Schreibens dieses Buchs. Mein Dank gilt auch dem *Vieweg+Teubner-Verlag* für die konstruktive Zusammenarbeit.

Begleiten Sie, verehrte Leser, mich nun als *„Fußgänger"* durch das Gebiet des Datenbankentwurfs!

Berlin, im August 2009 Helmut Jarosch

Inhaltsverzeichnis

Einführung

Information Engineering

Die Gesamtheit der Informationen in einem Unternehmen nimmt immer stärker den Charakter einer Ressource an. Für die Führung des Unternehmens muss diese Ressource ebenso verfügbar gehalten werden wie das Arbeitsvermögen, der Boden und das Kapital. Für die Verwaltung und Nutzung der Ressource Information werden Informationssysteme eingesetzt. Die Erfahrungen der Vergangenheit belegen, dass Unternehmen, die sich bei der Gestaltung solcher Informationssysteme bewusst an den strategischen Unternehmenszielen orientieren, Wettbewerbsvorteile erlangen. Die auf dieses Ziel ausgerichtete Anwendung miteinander verzahnter Methoden für die Planung, die Analyse, den Entwurf und die Implementierung von automatisierten Informationssystemen bezeichnete Martin schon 1989 treffend als *„Information Engineering"* [MART89].

Das Information Engineering ordnet sich in einen zweistufigen Prozess der Modellierung ein, der vom Entwurf fachspezifischer Modelle bis hin zur Entwicklung des betrieblichen Informationssystems reicht.

Fachspezifische Wissensmodelle

In der *ersten Etappe* des Modellierungs-Prozesses werden unter Anwendung von fachwissenschaftlichen Methoden *fachspezifische Wissensmodelle* der betrieblichen Realität aufgestellt. Jedes dieser Modelle ist die Widerspiegelung eines Ausschnitts der realen Welt aus der spezifischen Sicht einer Fachwissenschaft. Beispielsweise gewinnt man das Geschäftsmodell des Unternehmens durch die Anwendung von Methoden der Betriebswirtschaftslehre. Durch Anwendung von Methoden der Ingenieurwissenschaften erhält man dagegen ein Modell der Fertigungsprozesse des Unternehmens. Dabei kann ein und dieselbe Erscheinung der realen Welt in mehreren fachspezifischen Modellen jeweils aus unterschiedlicher Sicht dargestellt werden.

Fachkonzept

Die fachspezifischen Wissensmodelle sind Ausgangspunkt für das Information Engineering, das die *zweite Etappe* des Modellierungs-Prozesses bildet. In seinem Verlauf werden aus den fachspezifischen Wissensmodellen unter anderem *Datenmodelle* und *Funktionenmodelle* abgeleitet und in einer komplexen Modellbank (*Fachkonzept*) zusammengefasst. Diese Modellbank

wird schließlich dazu verwendet, die betrieblichen Informationssysteme zu erstellen und zu pflegen.

Schwerpunkte des Lehrbuchs

Dieses Lehrbuch befasst sich mit einem Teilaspekt des Information Engineering. Es konzentriert sich auf zwei Schwerpunkte:

- Die Struktur-Beschreibung der für die betriebliche Informationsverarbeitung benötigten Daten in Form eines *konzeptionellen Datenmodells*, das mit den Mitteln der graphischen Beschreibungssprache des *Entity-Relationship-Modells* dargestellt wird.
- Die Repräsentation des konzeptionellen Datenmodells in einer Datenbank in Form eines *logischen Datenschemas*. Die Datenbank folgt dabei den Strukturierungs-Prinzipien des *relationalen Datenbank-Modells*.

Der Aufbau dieses Lehrbuchs wird in der Abbildung E-1 veranschaulicht. Wenn der Leser den Weg durch die sechs Kapitel dieses Buchs beschreitet, wird er „als Fußgänger“ die im Vorwort angekündigte Wanderung durch das Gebiet des Datenbankentwurfs unternehmen.

Kapitel 1

Das Kapitel 1 („Die Strecke“) soll die Grundlagen für das Verständnis der Kapitel 2 bis 5 legen. Zunächst wird beschrieben, wie der Weg der *Erstellung betrieblicher Anwendungssysteme* traditionell beschritten wurde und welche Veränderungen er seitdem erfahren hat. Dann werden Konzept, Aufgaben und Struktur eines *Datenbanksystems* erläutert. Schließlich werden die *Modelle* und *Schemata* besprochen, die im Umfeld des Datenbankentwurfs von Bedeutung sind.

Kapitel 2

Das Kapitel 2 („Die erste Etappe“) befasst sich mit dem Entwurf *konzeptioneller Datenmodelle*. Als graphische Beschreibungssprache wird die Sprache des *Entity-Relationship-Modells* in einer Grundausbaustufe vorgestellt. An Hand eines durchgängigen Beispiels wird der Prozess der Datenmodellierung in vier Phasen detailliert behandelt. Die erste Phase besteht in der *Klassifizierung*, durch die Ordnung in die Vielfalt der betrieblichen Objekte gebracht wird. Die zweite Phase hat die *Abstraktion* zum Ziel: Von den inhaltlichen Details der betriebswirtschaftlichen Objekte wird abstrahiert, und sie werden auf einen Satz relevanter Eigenschaften reduziert. Die dritte Phase beschäftigt sich mit der *Identifizierung*: Für die in Klassen zusammengefassten Objekte muss geklärt werden, wie sie innerhalb der Klasse unterschieden werden können. Die vierte Phase besteht schließlich in der *Beschreibung der sachlogischen Zusammen-*

hänge zwischen den Objekten. Überlegungen zur Modellierung in *Grenzfällen des Entity-Relationship-Modells*, zur Qualitätssicherung, die im Umfeld der Datenmodellierung als *Normalisierung* bezeichnet wird, und zum *Nutzen der Datenmodellierung* runden dieses Kapitel ab.

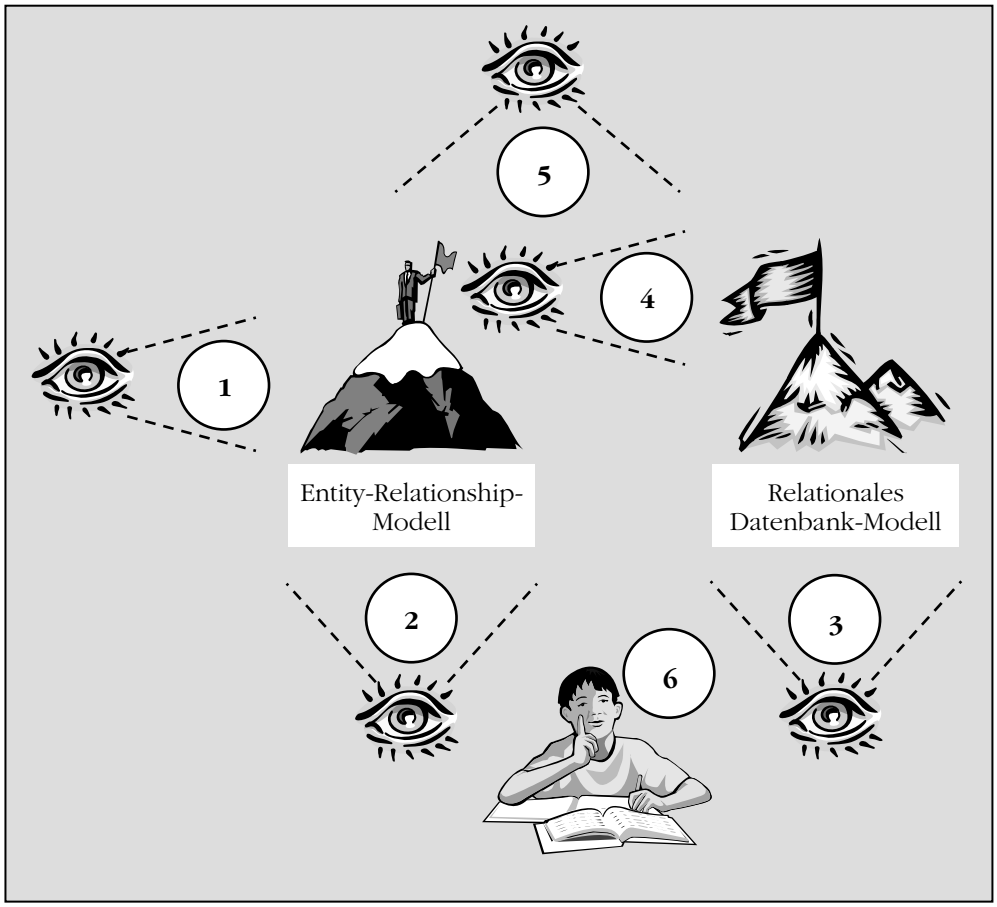


Abb. E-1: Aufbau des Lehrbuchs

Kapitel 3

Das Kapitel 3 („Die Zielstruktur“) beschreibt die Möglichkeiten zur Repräsentation von Datenstrukturen in einer Datenbank. Dabei werden drei Datenbank-Modelle nach einer gemeinsamen Systematik besprochen und einander gegenübergestellt. Zunächst werden das *hierarchische Datenbank-Modell* und das *Netzwerk-*

Datenbank-Modell kurz charakterisiert. Das *relationale Datenbank-Modell* wird in seinen Grundprinzipien ausführlich dargestellt. Besondere Schwerpunkte bilden dabei die referenzielle Integrität sowie die Repräsentation von CM:CN-Beziehungstypen und von Rekursiv-Beziehungstypen.

Kapitel 4

Das Kapitel 4 („Die zweite Etappe") befasst sich mit der *Umsetzung des konzeptionellen Datenmodells* in ein *logisches Datenschema,* das dem Gestaltungsspielraum des relationalen Datenbank-Modells entspricht. Für die einzelnen Sprachelemente des Entity-Relationship-Modells wird ausführlich und an Hand von Beispielen untersucht, ob und wie sie sich mit Hilfe der Strukturelemente des relationalen Datenbank-Modells repräsentieren lassen. Dabei werden *20 Transformationsregeln* hergeleitet, die den Algorithmus der Umsetzung darstellen. Das im Kapitel 2 als durchgängiges Beispiel entwickelte Datenmodell wird mit Hilfe dieser 20 Transformationsregeln *„von Hand"* in das relationale Datenbank-Modell überführt. Parallel dazu wird die *automatische Generierung* der Tabellen-Struktur durch das CASE-Tool *„PowerDesigner"* vorgeführt.

Kapitel 5

Das Kapitel 5 („Der Überblick") stellt sich die Aufgabe, die verstreut vorgestellten Erkenntnisse aus den Kapiteln 2, 3 und 4 zu sammeln und zu systematisieren. Der Leser, der lediglich an den *praktischen* Fragen der Datenmodellierung interessiert ist, kann dieses Kapitel überschlagen oder „diagonal" lesen. Zunächst wird untersucht, wann Fakten durch *Objekttypen* und wann durch *Beziehungstypen* darzustellen sind. Dann werden sowohl für die dualen Beziehungstypen als auch für die Rekursiv-Beziehungstypen folgende Fragen geklärt: Wie lassen sich die verschiedenen Beziehungstypen *konstruktiv* herleiten? Wann sollte man Beziehungstypen in einen *Koppel-Objekttyp* umwandeln? Welche der Beziehungstypen lassen sich im *relationalen Datenbank-Modell* repräsentieren?

Kapitel 6

Das Kapitel 6 („Die Generalprobe") enthält *fünf Aufgaben* zum Datenbankentwurf. Zur Überprüfung seines Kenntnisstands nach der Lektüre des Lehrbuchs sollte der Leser für die fünf betrieblichen Gegenstandsbereiche jeweils das Datenmodell entwickeln und es in das relationale Datenbank-Modell überführen. Die Lösungen dieser Aufgaben werden ausführlich kommentiert.

1 Die Strecke: Datenbankentwurf

Liebe Leserinnen und Leser! Wir befinden uns am Start unserer Wanderung durch das Gebiet des Datenbankentwurfs. Bevor wir unseren Weg beginnen, wollen wir uns zunächst einen Überblick über die Strecke verschaffen.

In diesem Kapitel sollen zunächst die Grundlagen für das Verständnis der folgenden Kapitel gelegt werden. Wir beschreiben deshalb, wie der Weg zur *Erstellung betrieblicher Anwendungssysteme* traditionell beschritten wurde und welche Veränderungen er seitdem erfahren hat. Dann werden Konzept, Aufgaben und Struktur eines *Datenbanksystems* erläutert. Schließlich werden die *Modelle* und *Schemata* besprochen, die im Umfeld des Datenbankentwurfs von Bedeutung sind. Die Einordnung dieses Kapitels in den Kontext des Lehrbuchs wird in der Abbildung 1-1 dargestellt.

1.1 Der Weg der Entwicklung betrieblicher Anwendungssysteme

Wie in der Einführung zu diesem Buch dargelegt wurde, bilden die fachspezifischen Wissensmodelle den Ausgangspunkt für das Information Engineering, in dessen Verlauf aus diesen Wissensmodellen unter anderem *Datenmodelle* und *Funktionenmodelle* abgeleitet werden. Diese Modelle werden aus den fachspezifischen Wissensmodellen gewonnen, indem man von der semantischen Ausprägung des Wissens abstrahiert und die Betrachtung lediglich auf die syntaktischen Aspekte des Wissens reduziert.

Semantische und syntaktische Informationsverarbeitung

Mit dem Begriffspaar der *semantischen* und der *syntaktischen Informationsverarbeitung* ist ein Grundproblem der Automatisierung im allgemeinen und der Entwicklung automatisierter Informationssysteme im speziellen angesprochen. Die Transformation semantischer Kenntnisse in syntaktische Strukturen bildet das Grundproblem jedes Automatisierungsvorhabens. Während nämlich der Mensch zur semantischen Informationsverarbeitung befähigt ist, kann der Computer lediglich eine syntaktische Informationsverarbeitung ausführen.

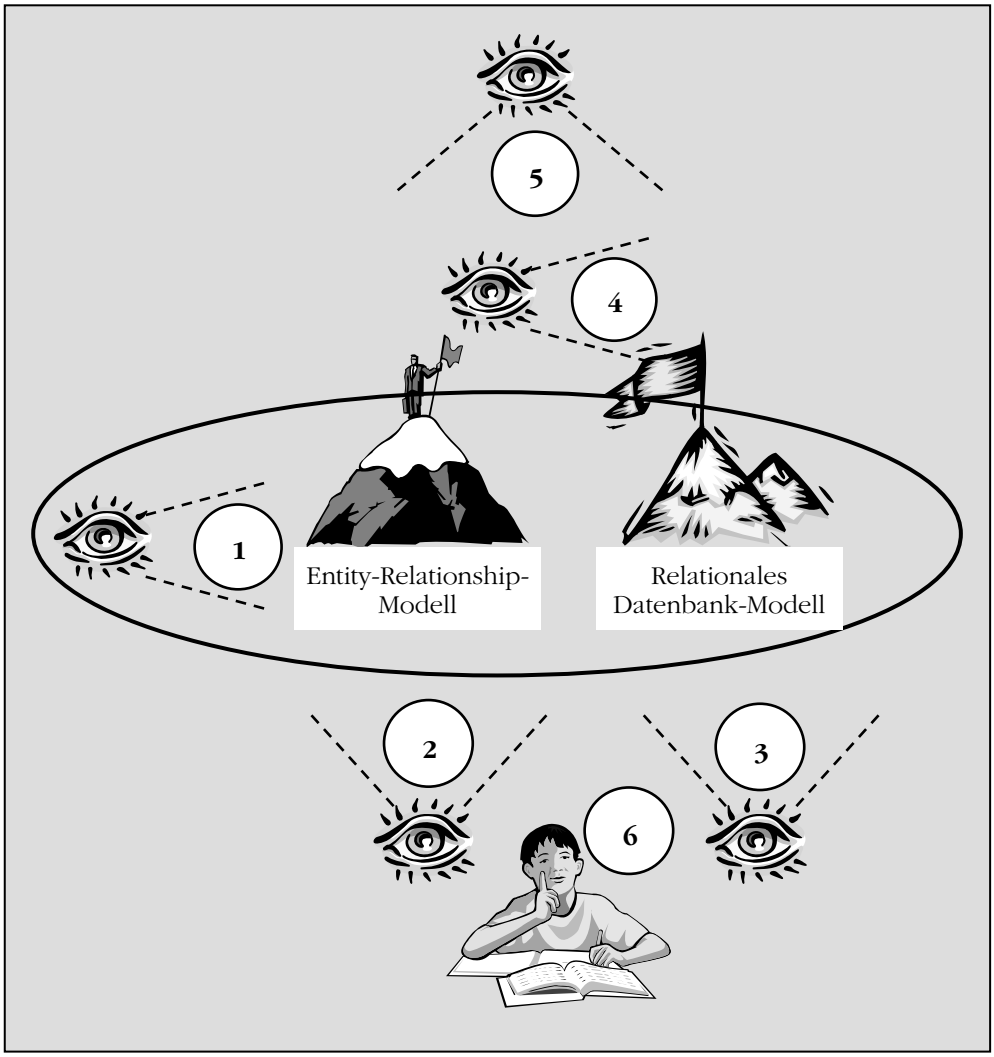


Abb. 1-1: Gegenstand des Kapitels 1

Der Mensch ist es bei den meisten Tätigkeiten gewohnt, Informationen unter Beachtung ihrer Bedeutung – also *semantisch* - zu verarbeiten und fehlende Angaben durch Hintergrundwissen zu ergänzen. So wird beispielsweise ein Personalchef bei der Beurteilung einer Bewerbung die Angabe des erlernten Berufs unter Kenntnis der Wichtigkeit dieses Berufs für sein Unterneh-

men und unter Berücksichtigung seines Hintergrundwissens über die allgemein üblichen Ausbildungswege und Ausbildungszeiten verarbeiten.

Der Computer ist dagegen nur fähig, strukturelle Informationen ohne Berücksichtigung ihrer Bedeutung zu verarbeiten, wobei alle Angaben durch *syntaktische* Konstruktionen repräsentiert sein müssen. Er kann dabei nur jene Informationen berücksichtigen, die ihm entweder explizit eingegeben wurden oder die er – nach Methoden der künstlichen Intelligenz – durch Anwendung von Ableitungsregeln aus den explizit formulierten Angaben herleiten kann.

Sollen nun Informationsverarbeitungsprozesse, die bisher vom Menschen ausgeführt wurden, automatisiert werden, so muss man zunächst erkennen, welche Informationen und Verarbeitungsregeln der Mensch verwendet. Das ist insofern problematisch, als sich der Mensch häufig nicht im klaren darüber ist, welche Informationen (aus expliziten Fakten, aus Hintergrundwissen und aus hergeleiteten Angaben) er eigentlich für seine Tätigkeit auswertet.

Traditioneller Weg

Der traditionelle Weg der Erstellung automatisierter *Anwendungssysteme* im Spannungsfeld von semantischer und syntaktischer Informationsverarbeitung ist in Abbildung 1-2 dargestellt.

Das zu erstellende Anwendungssystem befindet sich auf der Ebene der syntaktischen Informationsverarbeitung. Es enthält – grob gesprochen – zwei wesentliche Komponenten:

1. Einen Speicher für sämtliche Daten, die für die Verarbeitung erforderlich sind oder die im Ergebnis der Verarbeitung entstehen. Dieser Speicher wird als *Datenbank* bezeichnet. Die Struktur der Datenbank muss so festgelegt werden, dass alle benötigten Daten gemeinsam mit ihren sachlogischen Zusammenhängen redundanzfrei gespeichert werden können.
2. Ein *Anwendungsprogramm,* das die fachspezifischen Algorithmen repräsentiert, nach denen die Informationsverarbeitung erfolgen soll.

Die Sachkenntnis über die logische Struktur der Daten und über die konkreten Schritte der Informationsverarbeitung liegt bei den Experten der betroffenen Fachabteilung. Das primäre Problem bei der Entwicklung des Anwendungssystems besteht nun darin, das *semantische Wissen der Fachabteilung* in die *syntaktischen*

Strukturen der Datenbank und des Anwendungsprogramms zu transformieren. Zur Realisierung dieser Transformation werden traditionell *Anwendungsentwickler* herangezogen, die entweder in der DV-Abteilung des Unternehmens oder in externen Dienstleistungs-Unternehmen beschäftigt sind. Die Anwendungsentwickler haben es während ihrer Ausbildung gelernt, in syntaktischen Strukturen zu denken.

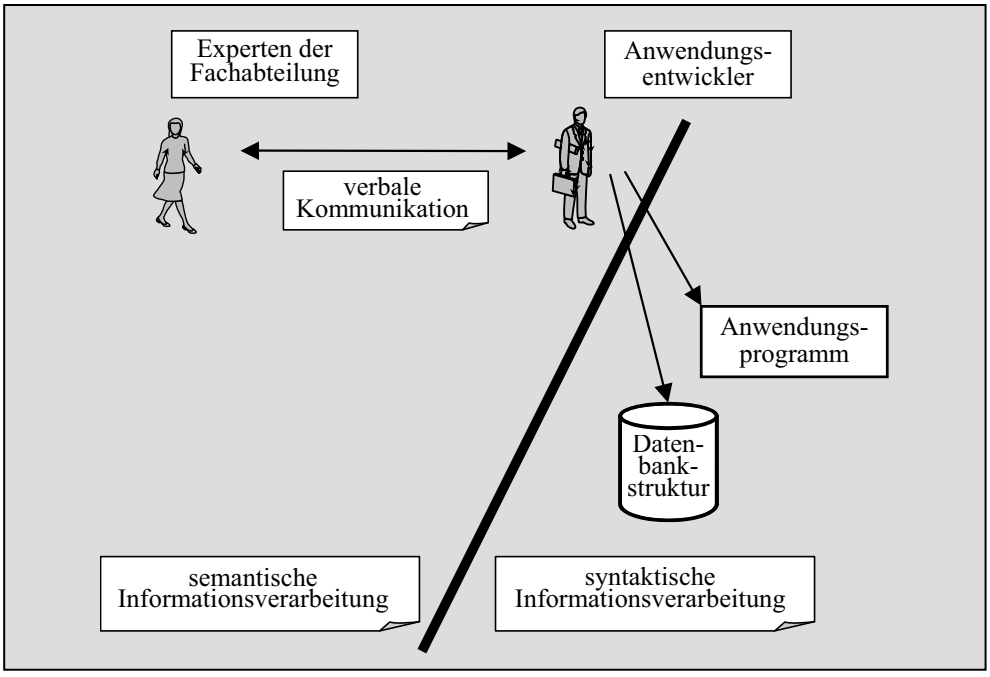


Abb. 1-2: Der traditionelle Weg der Anwendungsentwicklung

Verbale Kommunikation

Die Übertragung des Fachwissens von den Spezialisten der Fachabteilung auf die Anwendungsentwickler erfolgt mittels *verbaler Kommunikation,* also in der natürlichen Sprache (beispielsweise in Besprechungen oder in Form eines Pflichtenhefts). Diese Form der Wissensvermittlung ist sehr störanfällig. Das hat hauptsächlich die folgenden Gründe:

- Die *natürliche Sprache* ist für eine Beschreibung komplex strukturierter Fachkenntnisse nur in sehr beschränktem Maße geeignet. Sie produziert ihrer Natur nach viele Ungenauigkeiten und Mehrdeutigkeiten.

- Die Kommunikationspartner gehören *verschiedenen Fachrichtungen* an, benutzen meist eine voneinander abweichende Fachterminologie und verfügen über unterschiedliches Hintergrundwissen.
- Die Fachkenntnisse der Spezialisten, die sie in vielen Jahren der Ausbildung und der beruflichen Praxis erworben haben, lassen sich nicht in einem *„Crashkurs"* auf die Anwendungsentwickler übertragen.
- Das übermittelte Fachwissen ist ausschließlich an die *Person* des Anwendungsentwicklers gebunden; steht dieser für spätere Arbeiten oder Änderungen nicht mehr zur Verfügung, muss die Wissensvermittlung erneut erfolgen.

Der beschriebene traditionelle Weg der Anwendungsentwicklung führte zum sogenannten „Anwendungsstau", weil die Anwendungsentwickler die Vielzahl der gewünschten Projekte nicht in der geforderten Zeit bearbeiten konnten.

Veränderter Weg

Der Ausweg aus diesen Schwierigkeiten lag im Übergang zu einem veränderten Weg der Erstellung von Anwendungssystemen, bei dem der Engpass der Transformation semantischer Kenntnisse in syntaktische Strukturen vermieden wurde: Es wurden *Fachsprachen* entwickelt, die es den Experten der Fachabteilungen ermöglichen, selbst – eventuell unter methodischer Anleitung durch die Anwendungsentwickler – ihre Fachkenntnisse in syntaktischer Weise, d.h. exakt und vollständig zu repräsentieren. Bei diesen Fachsprachen handelt es sich hauptsächlich um graphische Ausdrucksmittel, deren Anwendung keine spezielle DV-Ausbildung voraussetzt.

Abbildung 1-3 zeigt diesen veränderten Weg der Erstellung von Anwendungssystemen. Die stark ausgezogenen Pfeile kennzeichnen dabei jene Entwicklungsaktivitäten, die in diesem Lehrbuch behandelt werden.

Modellierung

Die Repräsentation des Fachwissens erfolgt nun in einem *Modellierungs-Prozess*. Mit Hilfe graphischer Fachsprachen können die Experten der Fachabteilung ihre Kenntnisse über die zu automatisierende Informationsverarbeitung in einem *Fachkonzept* zusammentragen. Sie erfahren dabei methodische Anleitung durch die Anwendungsentwickler.

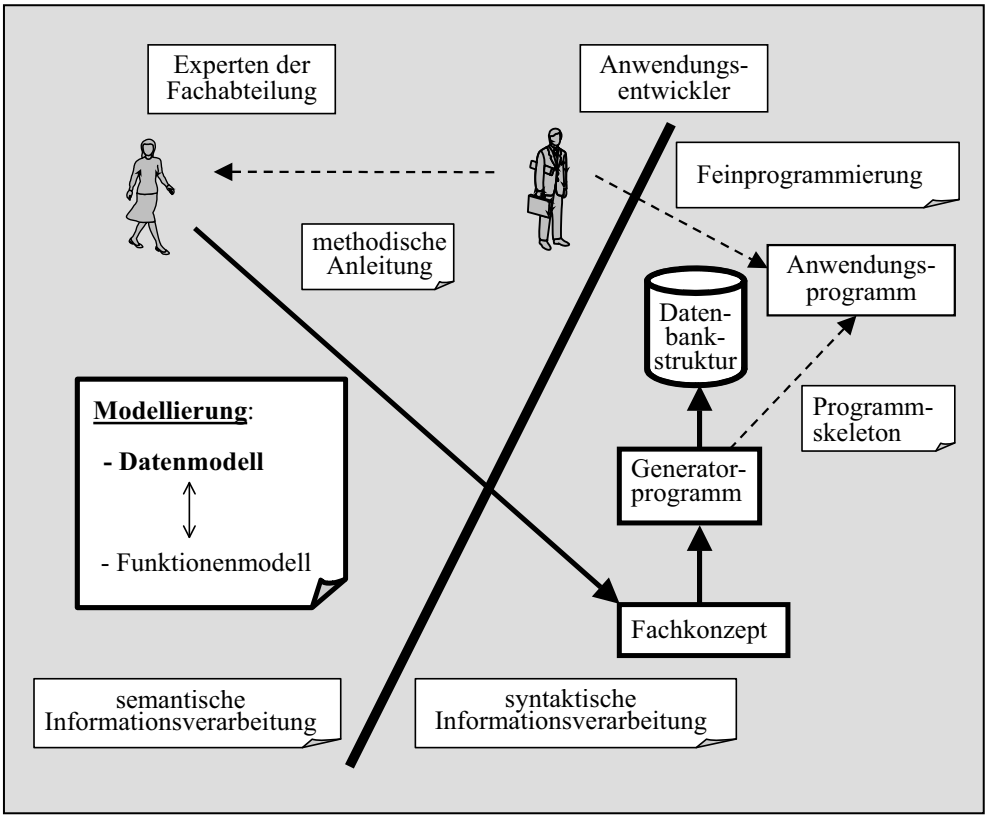


Abb. 1-3: Der veränderte Weg der Anwendungsentwicklung

Bewältigung der Komplexität

Das Fachwissen, das die Experten der Fachabteilung für die Entwicklung eines betrieblichen Informationssystems beisteuern, ist durch eine hohe Komplexität gekennzeichnet. Um diese Komplexität zu beherrschen, bedient man sich des antiken Prinzips „Divide et impera!“ („Teile und herrsche!“) und betrachtet die betriebliche Realität zunächst getrennt aus zwei verschiedenen Blickwinkeln: unter dem Aspekt der benötigten Daten und unter dem Aspekt der auszuführenden Informationsverarbeitungs-Prozesse. Die beiden unterschiedlichen Sichten auf die Realität werden später – wie durch eine stereoskopische Brille –zu einer Gesamtsicht zusammengeführt.

Fachkonzept

Das Fachkonzept enthält fachlich relevante Vorgaben für das zu erstellende Informationssystem, von denen drei Aspekte besonders hervorgehoben werden sollen:

1. Das *Datenmodell:* Es enthält alle Angaben über die Struktur der zu verarbeitenden bzw. im Ergebnis der Verarbeitung entstehenden Daten und über deren sachlogische Zusammenhänge. Als Modellierungssprache wird meist die graphische Sprache des *Entity-Relationship-Modells* benutzt. Das Datenmodell steht in diesem Lehrbuch im Zentrum des Interesses.

2. Das *Funktionenmodell:* Es enthält eine fachlich begründete Gliederung der zu unterstützenden Prozesse der Informationsverarbeitung. Als Modellierungssprache wird häufig die graphische Sprache des *Funktionenbaums* verwendet. Das Funktionenmodell werden wir im Rahmen dieses Lehrbuchs *nicht* behandeln.

3. Die *CRUD-Matrix:* Sie beschreibt das Wechselverhältnis von Datenmodell und Funktionenmodell. Als Modellierungssprache dient eine *Matrixdarstellung*. Das Akronym „CRUD" steht dabei für die vier Aktivitäten „***C***reate", „***R***ead", „***U***pdate" und „***D***elete", die eine Funktion mit den Daten ausführen kann. Die CRUD-Matrix ist *nicht* Gegenstand dieses Lehrbuchs.

Generierung

Da das Fachkonzept auf der Ebene der syntaktischen Informationsverarbeitung angesiedelt ist, werden die Probleme, die beim traditionellen Weg der Anwendungsentwicklung entstehen, weitgehend vermieden. Außerdem bietet sich nun die Möglichkeit, die Angaben des Fachkonzeptes durch automatisierte Prozesse zu verarbeiten: Mit Hilfe von *Generator-Programmen* werden aus dem Fachkonzept die Struktur der Datenbank und die Grundgerüste (Skeletons) der Anwendungsprogramme abgeleitet. Die Programmskeletons enthalten bereits wesentliche Aspekte der Informationsverarbeitung, die von den Anwendungsentwicklern im Zuge der Feinprogrammierung zu einem lauffähigen Programm ergänzt werden.

Vorzüge des veränderten Wegs

Der beschriebene Entwicklungsweg weist gegenüber dem traditionellen Weg wesentliche Vorzüge auf:

- Die Beschreibungssprachen sind *syntaktisch* orientiert: Die Angaben sind eindeutig und können zudem automatisch auf Vollständigkeit und Widerspruchsfreiheit geprüft werden.
- Die Experten der Fachabteilung und die Anwendungsentwickler arbeiten nach einer klaren *Kompetenzen-Teilung* zusammen: Die Anwendungsentwickler betreuen die Modellierung lediglich aus methodischer Sicht, während das Fachwissen von den Experten unter Verwendung der Modellierungssprachen repräsentiert wird.
- Das Fachwissen ist im Fachkonzept *personen-unabhängig* archiviert und kann bei Bedarf von den Experten der Fachabteilung selbst modifiziert werden.

CASE-Technologie

Der beschriebene Entwicklungsweg wird häufig auch als *CASE-Technologie* bezeichnet. Dabei steht das Akronym „CASE“ für „***C***omputer ***A***ided ***S***oftware ***E***ngineering“.

Der Bestandteil *„SE“* (*Software Engineering*) weist auf einen angestrebten Idealzustand der Programmentwicklung hin. In der Anfangszeit der Computer-Programmierung wurde das Programmieren wie eine Kunst betrieben[1]. Der Programmierer schrieb sein Programm auf höchst individuelle Weise, wobei eine eigenwillige Bezeichnung der Programmvariablen und eine trickreiche Algorithmen-Gestaltung üblich waren. Das führte dazu, dass die Programme von anderen Programmierern nur sehr mühsam zu verstehen und zu verändern waren. Allmählich setzte sich die Erkenntnis durch, dass auch das Programmieren aus dem Bereich der „Kunst“ in das Gebiet der Ingenieurwissenschaften gerückt werden müsse, dass also der Prozess der Software-Erstellung nach festen Methoden erfolgen muss.

Der Zusatz *„CA“* (*Computer Aided*) besagt, dass für die Entwicklung von Computer-Programmen der Computer verwendet wird. Der Begriff „CASE“ lässt sich also wie folgt beschreiben:

---

[1] Bezeichnenderweise trug das damalige Standard-Kompendium der Programmierung von Donald E. Knuth den Titel: „The ***Art*** of Computer Programming“.

CASE

Definition: Unter *CASE-Technologie* versteht man die Entwicklung von Software nach ingenierwissenschaftlichen Methoden unter Verwendung eines Computers

Der Computer wird bei dieser Verfahrensweise auf mehreren Ebenen genutzt, wie dies in Abbildung 1-4 dargestellt ist.

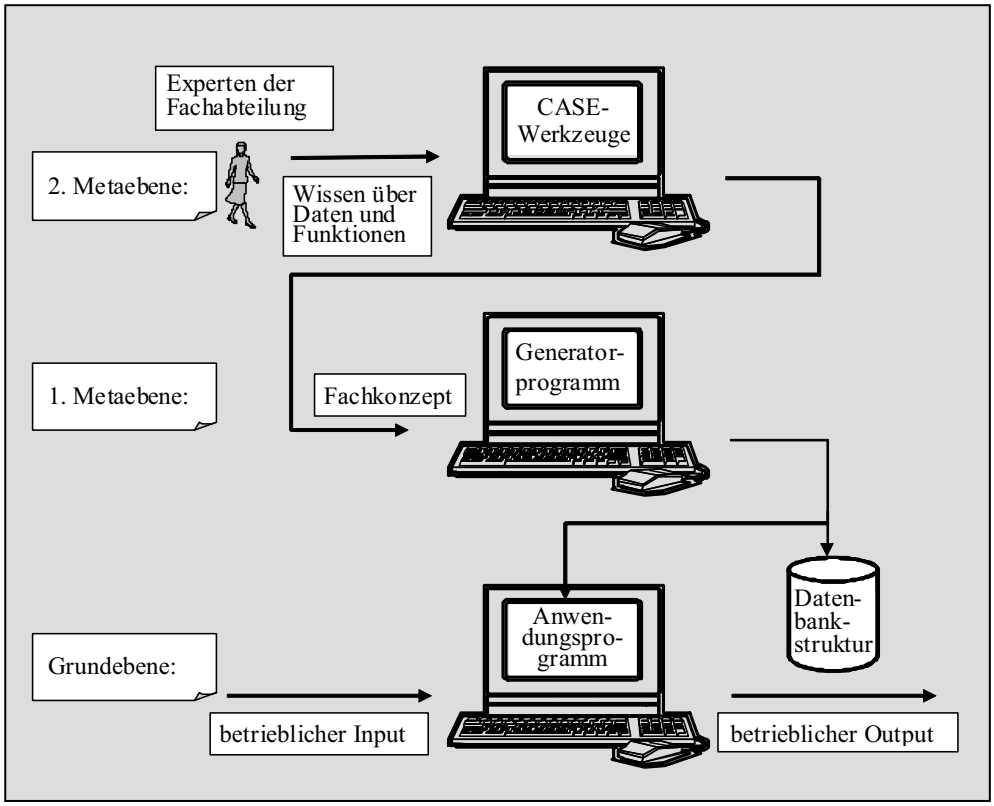


Abb. 1-4: Computer-Nutzung auf mehreren Ebenen

Computer-nutzung auf mehreren Ebenen

Auf der *Grundebene* dient der Computer dazu, die betriebliche Datenverarbeitung abzuwickeln, also betriebliche Input-Daten in betriebliche Output-Daten umzuwandeln. Das dafür benötigte Anwendungsprogramm wird auf der nächst-höheren Ebene, der *1. Metaebene*, durch ein Generator-Programm auf der Grundlage

des Fachkonzepts erzeugt. Das Fachkonzept wiederum ist das Ergebnis der Computernutzung auf der *2. Metaebene*, auf der es durch CASE-Werkzeuge aus dem Wissen zusammengestellt wird, das die Experten der Fachabteilung über Daten und Funktionen eingeben.

1.2 Das Datenbanksystem

Im vorangegangenen Abschnitt wurde die Datenbank als der Speicher sämtlicher Daten eingeführt, die für die Verarbeitung erforderlich sind oder die im Ergebnis der Verarbeitung entstehen. Doch bei weitem nicht alle betrieblichen Anwendungssysteme, die in der Praxis verwendet werden, nutzen für die Speicherung ihrer Daten eine Datenbank. Die Vorzüge, die ein Datenbank-Anwendungssystem gegenüber den herkömmlichen Datei-Anwendungssystemen hat, werden in der Gegenüberstellung dieser beiden Architektur-Prinzipien sichtbar.

Wir wollen deshalb den Weg von den Datei-Anwendungssystemen zu den Datenbank-Anwendungssystemen kurz nachzeichnen und die Vorteile deutlich machen, die man durch den Einsatz einer Datenbank gewinnt.

Datei-Anwendungssystem

Die *Datei-Anwendungssysteme* sind historisch aus dem Wunsch heraus entstanden, isolierte betriebliche Abläufe zu automatisieren. In der Anfangszeit des Computer-Einsatzes wurden in den Unternehmen für die einzelnen Geschäftsbereiche Anwendungssysteme eingeführt, die jeweils ihre eigenen Daten in Form einer Datei verwalteten. Für jedes dieser Anwendungssysteme wurde von den Entwicklern eine solche individuelle Struktur der Datei festgelegt, die – wenigstens nach Ansicht der Entwickler – für die jeweils zu verarbeitenden Daten am besten geeignet ist. So wurden beispielsweise Personaldaten häufig in einer anderen Struktur gespeichert als Produktdaten. Es entstand eine Architektur von Datei-Anwendungssystemen, wie sie in der Abbildung 1-5 dargestellt ist.

Vorteil von Datei-Anwendungssystemen

Der Vorteil dieser Architektur besteht darin, dass jedes Anwendungssystem *direkt* auf seine Daten in der jeweiligen Datei zugreifen kann. Damit wird im Routinebetrieb eine hohe Effizienz erreicht. Der „Zugriff“ auf die Daten betrifft die vier Aktivitäten, die wir bereits im Akronym „CRUD“ kennengelernt haben:

Create: Abspeichern neuer Daten,

Read: Lesen von Daten, ohne sie zu verändern,

Update: Lesen, Ändern und Zurückspeichern von Daten

Delete: Löschen von Daten.

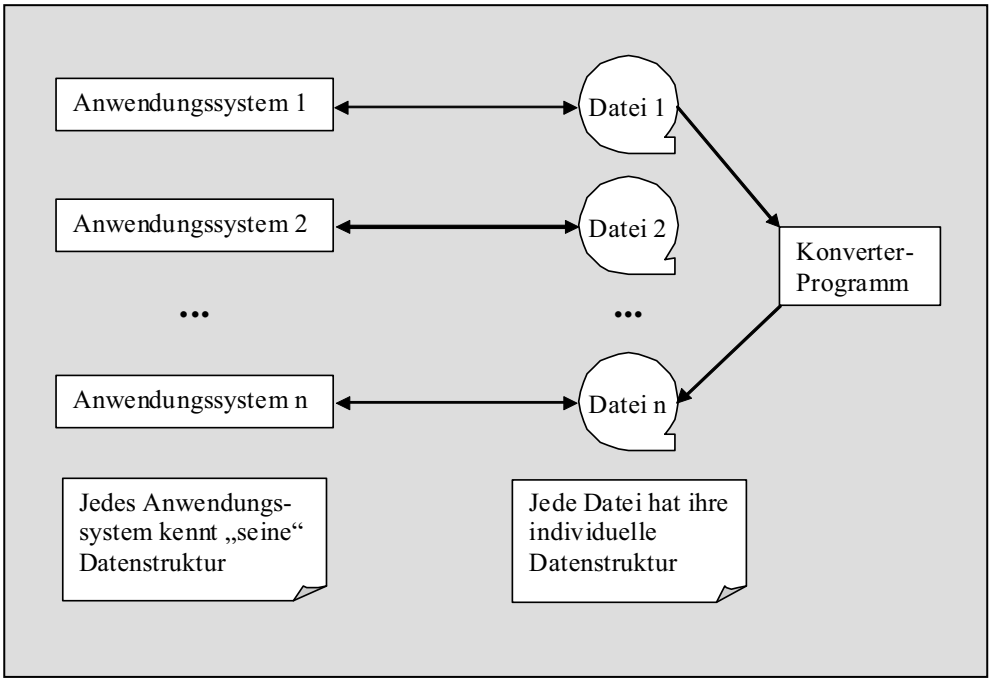


Abb. 1-5: Architektur von Datei-Anwendungssystemen

Nachteile von Datei-Anwendungssystemen

Dem Vorteil der hohen Effizienz stehen aber gewichtige Nachteile einer solchen Architektur gegenüber. Einige davon sollen hier besonders hervorgehoben werden:

1. Da jedes Anwendungssystem seine „private“ Datei verwendet, bleibt es nicht aus, dass Daten, die mehrere Anwendungssysteme benötigen, mehrfach gespeichert werden. Man spricht von *redundanter Datenspeicherung*. Auf die Problematik der redundanten Daten werden wir im Kapitel 2 (Abschnitte 2.4.2 und 2.6) näher eingehen.

2. Da jede Datei eine individuelle Datenstruktur besitzt, kann beispielsweise das „Anwendungssystem n“ nicht ohne weiteres auf die Daten der „Datei 1“ zugreifen. Durch ein *Konverter-Programm* müssen die Daten der „Datei 1“ erst in die Struktur der „Datei n“ überführt werden. Sollen nun allen n Anwendungssystemen sämtliche Daten zur Verfügung stehen, so müssen n(n-1) Konverter-Programme erstellt werden[2]. Bei nur 10 betrieblichen Anwendungssystemen sind das bereits 90 Konverter-Programme.
3. Da jedes Anwendungssystem unmittelbar auf „seine“ Datei zugreift, erfordert jede Änderung in der Datenstruktur dieser Datei eine Umprogrammierung des zugehörigen Anwendungssystems. Die Anwendungssysteme sind also *abhängig von der physischen Datenstruktur*.
4. Da jedes Anwendungssystem für die exklusive Nutzung „seiner“ Datei ausgelegt ist, können verschiedene Anwendungssysteme - selbst dann, wenn sie dieselbe Datenstruktur „erwarten“, - *keine parallelen Zugriffe* auf die Daten einer Datei ausführen. Es gibt nämlich keine zentrale „Instanz“, die diese Zugriffe koordinieren würde.
5. Jedes Anwendungssystem muss selbst für die Integrität seiner Daten sorgen. Es wäre aber zu aufwendig, in jedem Anwendungssystem alle Instrumente zur Sicherung der Daten-Integrität zu realisieren. Deshalb gibt es in den Datei-Anwendungssystemen im allgemeinen *keine Mechanismen zur Sicherung der Datenkonsistenz, des Zugriffsschutzes, des Daten-Recovery* bei Systemausfällen usw.

Vergleich mit kleiner Büchersammlung

Die Situation ist vergleichbar mit der Büchersammlung (= Datei) in der Wohnung von Herrn Abel, in der die Bücher (= Daten) nach einer Systematik (= Datenstruktur der Datei), die Herr Abel für sinnvoll hält, in den Regalen abgelegt sind. Herr Abel als Benutzer der Büchersammlung (= Anwendungssystem) geht selber an die Regale und sucht sich die benötigten Bücher heraus. Die Inhaberin der Nachbarwohnung, Frau Beck, hat sich eben-

[2] Der Ausdruck n(n-1) ist leicht einzusehen: Für ***ein*** Anwendungssystem müssen die Daten der „fremden“ (n-1) Dateien konvertiert werden. Da dies für ***jedes der n*** Anwendungssysteme gilt, sind n(n-1) Konverter-Programme erforderlich.

falls eine häusliche Büchersammlung aufgebaut. Die oben genannten Nachteile sind nun ganz offensichtlich:

1. Bücher, die sowohl Herrn Abel als auch Frau Beck interessieren, befinden sich in beiden Sammlungen.
2. Frau Beck hat Probleme, in der Büchersammlung von Herrn Abel ein gewünschtes Buch zu finden, weil sie das Ordnungsprinzip von Herrn Abel nicht kennt.
3. Wenn Frau Beck während des Urlaubs von Herrn Abel dessen Wohnung betreut und die Systematik seiner Bücheraufstellung ändert, muss dieser seinen „Suchalgorithmus" umstellen (und Frau Beck nie wieder in seine Wohnung lassen!).
4. Wenn Frau Beck bei Herrn Abel zu Besuch ist und beide parallel auf dasselbe Buch zugreifen wollen, entstehen Konflikte.
5. Es wird sich nicht lohnen, aufwendige Maßnahmen zur Sicherung der Integrität der häuslichen Büchersammlung vorzusehen. Das betrifft die Konsistenz (wenn es beispielsweise zu einem zweiten Band den ersten nicht mehr gibt), den Zugriffsschutz (wenn ein Besucher ein Buch „an sich genommen hat"), das Recovery (wenn ein Buch beschädigt oder zerstört wurde) usw.

Probleme bei großen Datensammlungen

Es ist klar, dass bei *großen Büchersammlungen* in öffentlichen Bibliotheken dieses Organisationsprinzip nicht funktioniert. Dort werden die Buchbestände „professionell" verwaltet. Dem Benutzer als „Laienkünstler in Sachen Bücherverwaltung" wird der direkte Zugriff auf die Büchern verwehrt. Er weiß nicht mehr, wo und wie sie aufgestellt sind. Deshalb ist für ihn eine Umstrukturierung der Buchbestände völlig uninteressant. Er braucht sich auch um Fragen der Integrität der Sammlung oder um Probleme des parallelen Zugriffs auf ein Buch keine Gedanken mehr zu machen. Sein Ansprechpartner ist stets der „Experte für die Büchersammlung", der Bibliothekar. Alle Wünsche hinsichtlich des Zugriffs auf die Bücher werden über den Bibliothekar abgewickelt. Die Kommunikationssprache ist die Sprache, die der Bibliothekar versteht.

Datenbank-Managementsystem

Bei der Verwaltung *großer Datenmengen* ist man denselben Weg gegangen. Statt der vielen „Laienkünstler in Sachen Datenspeicherung", als die wir die Datei-Anwendungssysteme bezeichnen können, hat man für das Management der umfangreichen Daten

einen „Experten“ hinzugezogen, der jeweils über das modernste technologische Wissen der Datenspeicherung verfügt. Dieser elektronische „Experte“ ist ein leistungsfähiges Dienstprogramm, das als *„Datenbank-Managementsystem“* bezeichnet wird. Die prinzipielle Architektur von *Datenbank-Anwendungssystemen* zeigt die Abbildung 1-6.

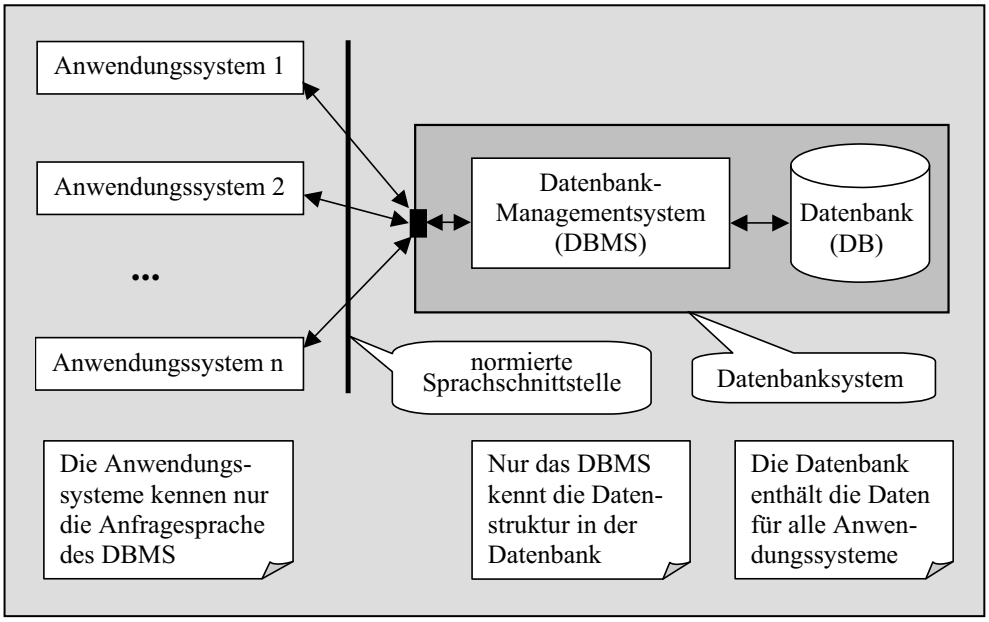


Abb. 1-6: Architektur von Datenbank-Anwendungssystemen

Datenbank-system

Die Anwendungssysteme sind nun von der Speicherstruktur der Daten „entkoppelt“, sie können nicht mehr *direkt* auf die Daten zugreifen. Ihr „Gesprächspartner“ in allen Fragen des Datenzugriffs ist das *Datenbanksystem*. Das Datenbanksystem verwaltet die Daten für alle Anwendungssysteme in einer *Datenbank*. Das „Expertenwissen“ über das Management der Daten ist in den Programmen des *Datenbank-Managementsystems* hinterlegt. Es allein „kennt“ die Struktur, in der die Daten in der Datenbank abgelegt sind. Die Kommunikation zwischen den Anwendungssystemen und dem Datenbanksystem erfolgt über eine *normierte Sprachschnittstelle*. Diese Sprachschnittstelle legt die Art und Weise fest, in der ein Anwendungssystem dem Datenbanksystem Aufträge erteilt.

Normierte Sprachschnittstelle

Die Existenz einer normierten Sprachschnittstelle für den Datenzugriff ist einer der Hauptvorteile der Architektur gemäß Abbildung 1-6. Sie bietet die Möglichkeit, das Datenbank-Managementsystem eines Anbieters gegen das eines anderen auszutauschen. Das Anwendungssystem ist somit nicht nur von der Speicherstruktur der Daten, sondern sogar vom speziellen Datenbank-Managementsystem entkoppelt.

Bei einem eventuellen Austausch des Datenbank-Managementsystems muss das Anwendungsprogramm nicht verändert werden, weil die Programm-Abschnitte, die die Kommunikation mit dem (richtiger: mit einem *beliebigen*) Datenbank-Managementsystem abwickeln, vom konkreten Datenbank-Managementsystem unabhängig sind. In unserem Bibliotheks-Beispiel kann es dem Bibliotheks-Benutzer – von Sympathien einmal abgesehen – ja auch egal sein, mit welchem Bibliothekar er kommuniziert, wenn nur alle dieselbe Sprache sprechen.

Structured Query Language

Die normierte Sprache für den Datenzugriff wird *SQL* (***S***tructured ***Q***uery ***L***anguage[3]) genannt. Neben ihren sonstigen Funktionen ist sie vor allem dafür entwickelt worden, die Anforderungen der Anwendungssysteme hinsichtlich der Speicherung, der Auswertung, der Änderung und des Löschens von Daten auszudrücken. Die Sprache SQL liegt nach ihrer ersten Standardisierung durch die ISO (***I***nternational ***S***tandardization ***O***rganization) im Jahre 1986 nun bereits als SQL:2008-Standard vor. Für weitere Einzelheiten sei der Leser auf die entsprechende Fachliteratur verwiesen, beispielsweise auf [THRO08].

Wir können nun die Begriffe der Datenbank und des Datenbank-Managementsystems definieren:

Datenbank

Definition: Eine *Datenbank* (engl. Data Base – DB) ist eine Sammlung von strukturierten Daten, zwischen denen sachlogische Zusammenhänge bestehen.

[3] Eigentlich sollte man das Akronym SQL besser als ***S****tandard* ***Q***uery ***L***anguage interpretieren, weil der Hauptnutzen der Sprache SQL in ihrer Standardisierung liegt.

Datenbank-Management-system

Definition: Ein *Datenbank-Managementsystem* (engl. Data Base Management System - DBMS) ist ein Programmsystem, das die notwendige System-Software für alle Aspekte der Datenverwaltung bereitstellt.

Vorzüge von Datenbank-Anwendungs-systemen

Aus der Gegenüberstellung mit den beschriebenen Nachteilen von Datei-Anwendungssystemen ergeben sich nun die Vorzüge eines Datenbank-Anwendungssystems gemäß der Architektur der Abbildung 1-6:

1. Da die Daten aller Anwendungssysteme in ein und derselben Datenbank abgelegt sind, müssen die Daten nur einmal gespeichert werden (*Redundanzfreiheit*). Das bedeutet aber, dass die Struktur der Daten nicht auf die Bedürfnisse eines einzelnen Anwendungssystems ausgerichtet sein kann, sondern dass sie für alle vorhandenen – und auch für künftige – Anwendungssysteme gleichermaßen geeignet sein muss. Die Anwendungssysteme mit ihren speziellen Anforderungen an die Daten, die sich nach den jeweiligen *Verarbeitungszielen* richten, sind somit entkoppelt von der Strukturierung der Daten in der Datenbank, die dem *Ziel der Redundanzfreiheit* verpflichtet ist. Man spricht von *logischer Datenunabhängigkeit*.
2. Da die Anwendungssysteme von der Speicherstruktur der Daten entkoppelt sind, kann jedes Anwendungssystem prinzipiell auf alle Daten zugreifen[4]. Dieselben Daten der Datenbank können in unterschiedlichster Weise verarbeitet werden (*Flexibilität*).
3. Da die Anwendungssysteme nur noch Bezug auf die logische und nicht mehr auf die physische Struktur der Daten nehmen, haben Änderungen in der Datenstruktur, die aus Reorganisationsgründen oder zur Effizienzsteigerung durchgeführt werden, keinen Einfluss auf die Anwendungsprogramme (*physische Datenunabhängigkeit*).

[4] Da der unkontrollierte Zugriff aber meist unerwünscht ist, kann er durch gezielt vergebene Zugriffsrechte gesteuert werden. Diese legen nämlich fest, welcher Benutzer (bzw. welche Benutzergruppe) auf welche Daten in welcher Weise (CRUD) zugreifen darf.

4. Da die Datenzugriffe aller Anwendungssysteme von einer zentralen Instanz, nämlich vom Datenbank-Managementsystem, koordiniert werden, lassen sich Konflikte vermeiden, die bei konkurrierenden Zugriffen durch mehrere Anwendungssysteme auftreten können (*Mehrbenutzerbetrieb*).
5. Die Anwendungssysteme sind nicht mehr für die Integrität der Daten zuständig. Dafür sorgt nun als zentrale Instanz das Datenbank-Managementsystem. Dieses enthält die technologisch fortschrittlichsten Mechanismen für folgende Aufgaben:
   - Sicherung der Vollständigkeit, Korrektheit und Widerspruchsfreiheit der Daten (*Datenintegrität*),
   - Schutz vor unbefugtem Zugriff (*Zugriffsschutz*),
   - Wiederherstellung eines korrekten Datenzustandes nach Störungen durch Software- und Hardwarefehler (*Recovery*).

Nachteil von Datenbank-Anwendungssystemen

Als Vorteil der Datei-Anwendungssysteme hatten wir ihre hohe Effizienz angegeben, die daher rührt, dass sie direkt auf die Daten in der Datei zugreifen. Datenbank-Anwendungssysteme nehmen dagegen für den Datenzugriff die „Vermittlungsdienste“ des Datenbank-Managementsystems in Anspruch. Diese zusätzliche „Schaltstelle“ reduziert natürlich zunächst die Ausführungsgeschwindigkeit von Datenzugriffen. Die Datenbank-Managementsysteme können diesen Nachteil aber in der Regel durch interne Vorkehrungen zur *Effizienz-Steigerung* überkompensieren. Sie beschleunigen die Zugriffe durch die Verwendung von Indizes und durch die Optimierung der Suchprozesse.

1.3 Modelle und Schemata

Der Weg des Datenbankentwurfs gemäß der CASE-Technologie, wie er in der Abbildung 1-3 veranschaulicht wurde, beginnt mit der Beschreibung der betrieblichen Realität in Form eines Fachkonzepts und endet mit der Strukturierung der Datenbank. Bei diesem Vorgehen sind auf den unterschiedlichsten Ebenen Abstraktions- und Modellbildungs-Prozesse erforderlich. Dies führte in der Fachliteratur, die den Datenbankentwurf behandelt, zu einem „inflationären“ Umgang mit dem Begriff des „Modells“. Die verwendeten Benennungen für die Ergebnisse der Modellbildungs-Prozesse sind uneinheitlich und führen zu einem „heillosen Durcheinander“, das noch dadurch gesteigert wird, dass

ein und dieselbe Benennung mitunter für völlig unterschiedliche Konzepte verwendet wird.

Wir wollen in diesem Lehrbuch Konfusionen in der Terminologie vermeiden und führen deshalb für jedes zu beschreibende Konzept eine eigene Benennung ein, die nicht unbedingt in gleicher Weise in anderen Lehrbüchern anzutreffen ist.

Wir werden zunächst zwei Beschreibungsebenen im Umfeld des Datenbankentwurfs strikt voneinander trennen:

2 Beschreibungsebenen

I. Beschreibung von *Strukturierungs-Möglichkeiten der Daten:* Diese Ebene umfasst die Beschreibungssprachen, die mit ihrer jeweiligen Mächtigkeit die prinzipiellen Möglichkeiten zur Datenstrukturierung festlegen.

II. Beschreibung von *Datenstrukturen:* Auf dieser Ebene werden die Strukturen der Daten für einen konkreten Ausschnitt der Realität beschrieben.

Beginnen wir mit der ersten Ebene - mit der Beschreibung der *Strukturierungs-Möglichkeiten der Daten.* In der Abbildung 1-7 sind vier relevante Gegenstandsbereiche dargestellt, wobei die in diesem Lehrbuch untersuchten Gegenstandsbereiche stark umrandet wurden:

Strukturierungs-Möglichkeiten der Daten

I a) *Ausschnitt der Realität*, über den Informationen gesammelt werden sollen: Dieser Ausschnitt kann die Arbeitsumgebung eines Mitarbeiters sein, er kann aber auch das gesamte Unternehmen umfassen. Der Ausschnitt der Realität wird mit Hilfe von Fachsprachen, z. B. der Fachsprache der Betriebswirtschaftslehre, beschrieben.

I b) Möglichkeiten zur Beschreibung der Datenstrukturen des Realitäts-Ausschnitts *unabhängig von der DV-technischen Realisierung:* Als Beschreibungssprache werden wir im Kapitel 2 das *„Entity-Relationship-Modell“* (ERM) kennen lernen. Andere Beschreibungssprachen, so etwa die Unified Modeling Language (UML) für die objektorientierte Modellierung von Informationssystemen, werden in diesem Buch nicht behandelt.

I c) Möglichkeiten zur Beschreibung von Datenstrukturen, die ein konkretes *Datenbank-Managementsystem* bietet: Wir bezeichnen die prinzipiellen Möglichkeiten zur Speicherung der Daten und ihrer sachlogischen Zusammenhänge als *„Datenbank-Modell“*. Im Kapitel 3 unterscheiden wir das hierarchische Datenbank-Modell, das Netzwerk-Daten-

bank-Modell und das relationale Datenbank-Modell. Für das relationale Datenbank-Modell wird eine Sprache zur Beschreibung von Relationen - die Sprache der *„Tabellen-Typbeschreibungen"* - entwickelt.

I d) Möglichkeiten zur Beschreibung der *physischen Datenstrukturen:* In diesem Gegenstandsbereich wird beschrieben, wo und wie die Daten gespeichert werden und wie der Zugriff auf diese Daten erfolgt. Als Beschreibungssprache dient eine Speicher-Beschreibungssprache DSDL (***D***ata ***S***torage ***D***escription ***L***anguage). Derartige Beschreibungssprachen sind nicht Gegenstand des Lehrbuchs.

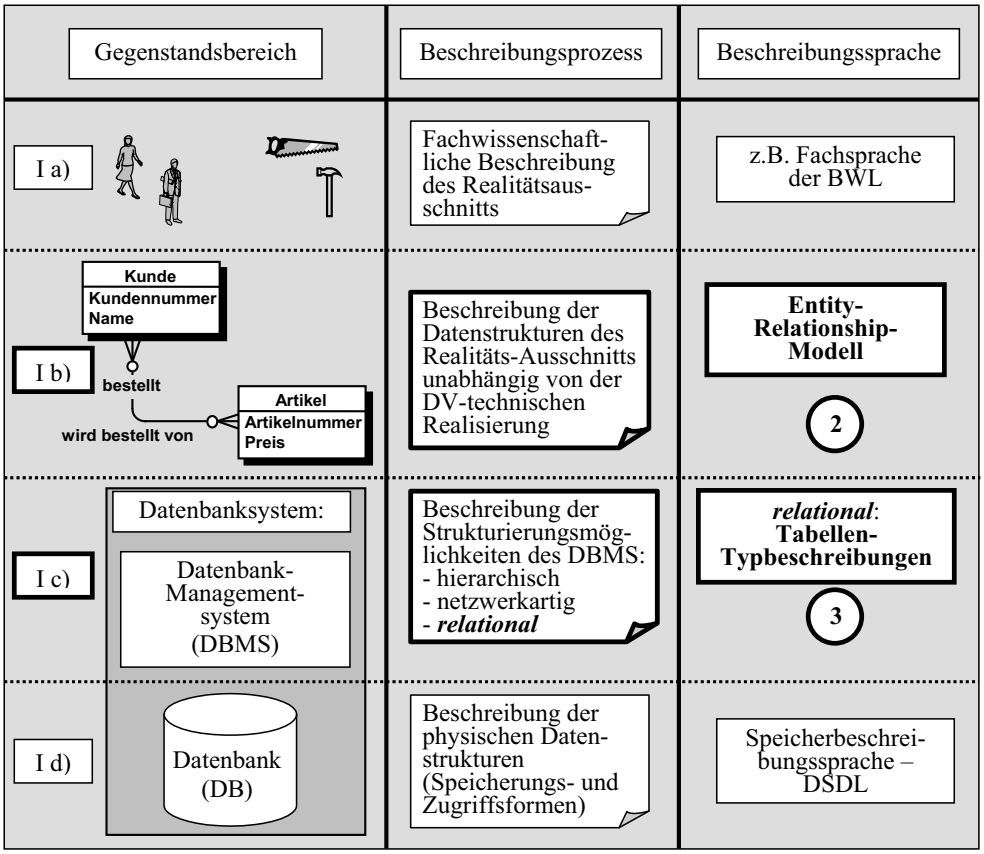


Abb. 1-7: Beschreibung von Strukturierungs-Möglichkeiten

Beschreibung von Datenstrukturen

Wir wenden uns nun der zweiten Ebene des Datenbankentwurfs zu, nämlich der Beschreibung von *Datenstrukturen*. Auf dieser Ebene sind die Ergebnisse der bisher besprochenen Beschreibungsprozesse angesiedelt. Die in den Beschreibungs-Ergebnissen festgehaltenen Angaben werden häufig auch als *Meta-Daten* bezeichnet, weil sie nicht die Nutzdaten selber repräsentieren, sondern *Daten über die Struktur der Nutzdaten* sind.

Die Beschreibung der Datenstrukturen erfolgt nach vier verschiedenen Sichten. Diese werden in der Abbildung 1-8 als *„informations-orientierte“*, *„daten-orientierte“*, *„anwendungs-orientierte“* und *„performance-orientierte“ Sicht* bezeichnet. Die Sichten, die Untersuchungsgegenstand dieses Buches sind, wurden stark umrandet. Die Aufeinanderfolge der Sichten orientiert sich wiederum an der graduell absteigenden Entkopplung der Struktur-Beschreibung der Daten von ihrer DV-technischen Repräsentation. Der dunkel dargestellte Bereich der Grafik entspricht dem *Drei-Ebenen-Modell* zur Standardisierung der Datenbank-Architektur, das von der ANSI-Normierungsgruppe für Systemarchitekturen (***A***merican ***N***ational ***S***tandards ***I***nstitute) schon 1975 vorgeschlagen wurde [ANSI75].

II a) *Informations-orientierte Sicht:* Die Daten werden hinsichtlich ihrer Struktur unabhängig davon beschrieben, mit welchem konkreten Datenbank-Managementsystem die Verwaltung der Daten erfolgen soll. Als Beschreibungssprache verwenden wir die graphische Sprache des *Entity-Relationship-Modells*. Das Beschreibungsergebnis wird als *„konzeptionelles Datenmodell“* bezeichnet. Wenn Missverständnisse ausgeschlossen sind, werden wir auch einfach vom *„Datenmodell“* sprechen. Die Entwicklung konzeptioneller Datenmodelle ist Gegenstand des Kapitels 2.

II b) *Daten-orientierte Sicht:* Die Beschreibung der Daten erfolgt unter Berücksichtigung der Möglichkeiten, die das einzusetzende Datenbank-Managementsystem für die Datenstrukturierung bietet. Diese ergeben sich hauptsächlich aus dem Datenbank-Modell, das dem Datenbank-Managementsystem zugrunde liegt (vgl. Kapitel 3). Die Beschreibung der Datenstruktur erfolgt aber unabhängig von einem konkreten Anwendungssystem und unabhängig von der physischen Repräsentation der Daten. Als Beschreibungssprache entwickeln wir im Kapitel 3 die formale Sprache der *Tabellen-Typbeschrei-*

bungen. Das Beschreibungsergebnis bezeichnen wir als *„logisches Datenschema“*. Für die Implementierung des logischen Datenschemas im jeweiligen relationalen Datenbank-Managementsystem wird in der Regel eine Datenbeschreibungssprache DDL (***D***ata ***D***escription ***L***anguage) verwendet. Mit dem Problem der Transformation eines konzeptionellen Datenmodells in ein logisches Datenschema befasst sich das Kapitel 4.

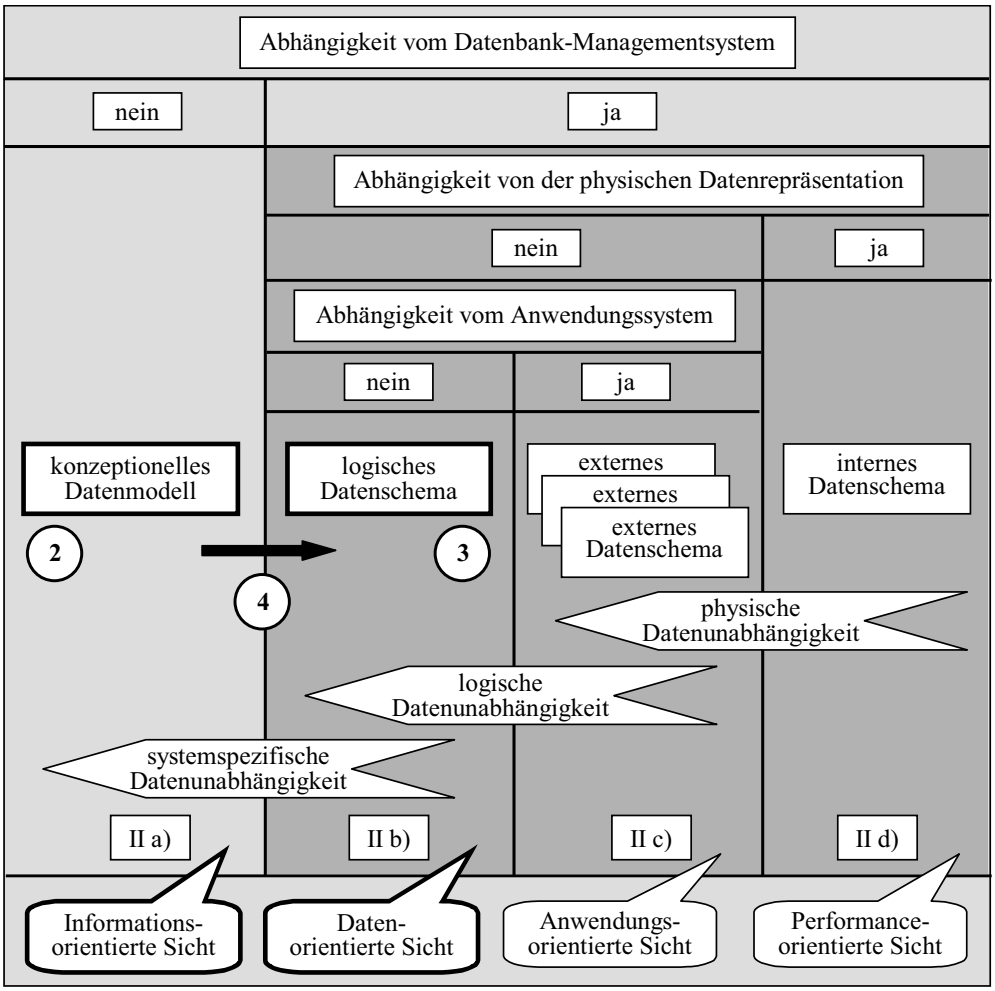


Abb. 1-8: Beschreibung von Datenstrukturen

II c) *Anwendungs-orientierte Sicht:* Die Struktur der Daten wird zwar unter Berücksichtigung eines speziellen Datenbank-Managementsystems und aus der Sicht eines konkreten Anwendungssystems beschrieben, aber unabhängig von der physischen Datenspeicherung. Die Beschreibung der Daten erfolgt also aus der Sicht (engl. view) eines speziellen Benutzers (bzw. einer Benutzergruppe) oder eines speziellen Anwendungsprogramms. Als Beschreibungssprache dient wiederum die Datenbeschreibungssprache *DDL*. Das Ergebnis einer solchen Strukturbeschreibung ist ein *„externes Datenschema"*. Da in der Regel viele Anwendungssysteme auf eine Datenbank zugreifen, die gemäß einem bestimmten logischen Datenschema aufgebaut ist, gibt es zu einem logischen Datenschema meist viele externe Datenschemata. Mit den externen Datenschemata werden wir uns in diesem Lehrbuch nicht beschäftigen.

II d) *Performance-orientierte Sicht:* Die Daten werden hinsichtlich ihrer physischen Organisation auf den jeweiligen Speichermedien beschrieben. Es wird festgelegt, wo und wie die Daten gespeichert und welche Zugriffsmechanismen verwendet werden. Dabei stehen Performance-Aspekte im Vordergrund. Als Beschreibungssprache stellen die Datenbank-Managementsysteme eine Speicherbeschreibungssprache DSDL (***D***ata ***S***torage ***D***escription ***L***anguage) bereit. Das Beschreibungsergebnis wird als *„internes Datenschema"* bezeichnet. Die performance-orientierte Sicht wird in diesem Lehrbuch nicht näher betrachtet.

Terminologie

Wir wollen dieses Kapitel mit einigen Bemerkungen zur Terminologie abschließen. Wir haben eine Daten-Strukturbeschreibung, die für ein konkretes Datenbank-Managementsystem entworfen wird, als *„Schema"* bezeichnet (logisches, externes, internes Schema). Die system-unabhängige Strukturbeschreibung der Daten nennen wir dagegen *„Datenmodell"*, genauer: *„konzeptionelles Datenmodell"*. Bei den Benennungen *„Entity-Relationship-Modell"* und *„Datenbank-Modell"* verwenden wir den Modell-Begriff, weil hier tatsächlich Modelle von Konstrukten beschrieben werden, bei denen man von der Fülle ihrer Aspekte abstrahiert und sich auf relevante Teilaspekte beschränkt.

2 Die erste Etappe: Von der Realität zum konzeptionellen Datenmodell

Im Kapitel 1 wurde die Datenmodellierung als eine Methode vorgestellt, die dazu dient, das von den Experten der jeweiligen Fachabteilung gesammelte semantische Wissen über die im betrieblichen Umfeld benötigten Informationen in einer syntaktischen Form darzustellen. Im Zuge der Datenmodellierung soll ein exaktes und vollständiges Modell für den Informationsbedarf des betrachteten Realitäts-Ausschnitts erarbeitet werden, das den Rahmen für die Entwicklung neuer oder aber für die Erweiterung bestehender Anwendungssysteme bildet.

Dieses Kapitel befasst sich mit der Beschreibung von *konzeptionellen Datenmodellen*. Als graphische Beschreibungssprache stellen wir die Sprache des *Entity-Relationship-Modells* in einer Grund-Ausbaustufe vor. An Hand eines durchgehenden Beispiels wird der Prozess der Datenmodellierung in vier Phasen detailliert behandelt. Die Einordnung dieses Kapitels in den Kontext des Lehrbuchs zeigt die Abbildung 2-1.

Entity-Relationship-Modell

Die Datenmodellierung mit der Sprache des Entity-Relationship-Modells ist eine spezielle Information-Engineering-Technik, die zur Entwicklung eines Datenmodells von hoher Qualität benutzt wird. Diese Methode wurde von Chen [CHEN76] entwickelt und seitdem mehrfach verbessert und erweitert. Im Rahmen dieser Darstellung werden nur die grundlegenden Elemente des Entity-Relationship-Modells erläutert. Für weiterführende Vertiefungen sei beispielsweise auf [HALP08] verwiesen.

Konzeptionelles Datenmodell

Das *konzeptionelle Datenmodell*, das mit Hilfe der Sprache des Entity-Relationship-Modells erstellt wird, soll nicht die konkreten *Datenwerte* wiedergeben, die in der Datenbank gespeichert werden und die die Grundlage für die betrieblichen Informationsverarbeitungs-Prozesse bilden. Es soll statt dessen lediglich die *typmäßige Struktur der Daten* in Form eines *konzeptionellen Datenmodells* beschreiben. Im Alltagsleben wäre das vergleichbar mit der Bestückung einer Küche. Zunächst überlegt man sich, welche Gefäße man für Zucker, Salz, Mehl usw. benötigt (das wäre das Modell der *Struktur der Küche*, also das Datenmodell), erst dann füllt man die Gefäße mit dem jeweiligen

Inhalt und verwendet diesen beim Kochen und Backen (das entspricht dem *Inhalt der Küche*, also der Datenbank mit ihren gespeicherten Werten).

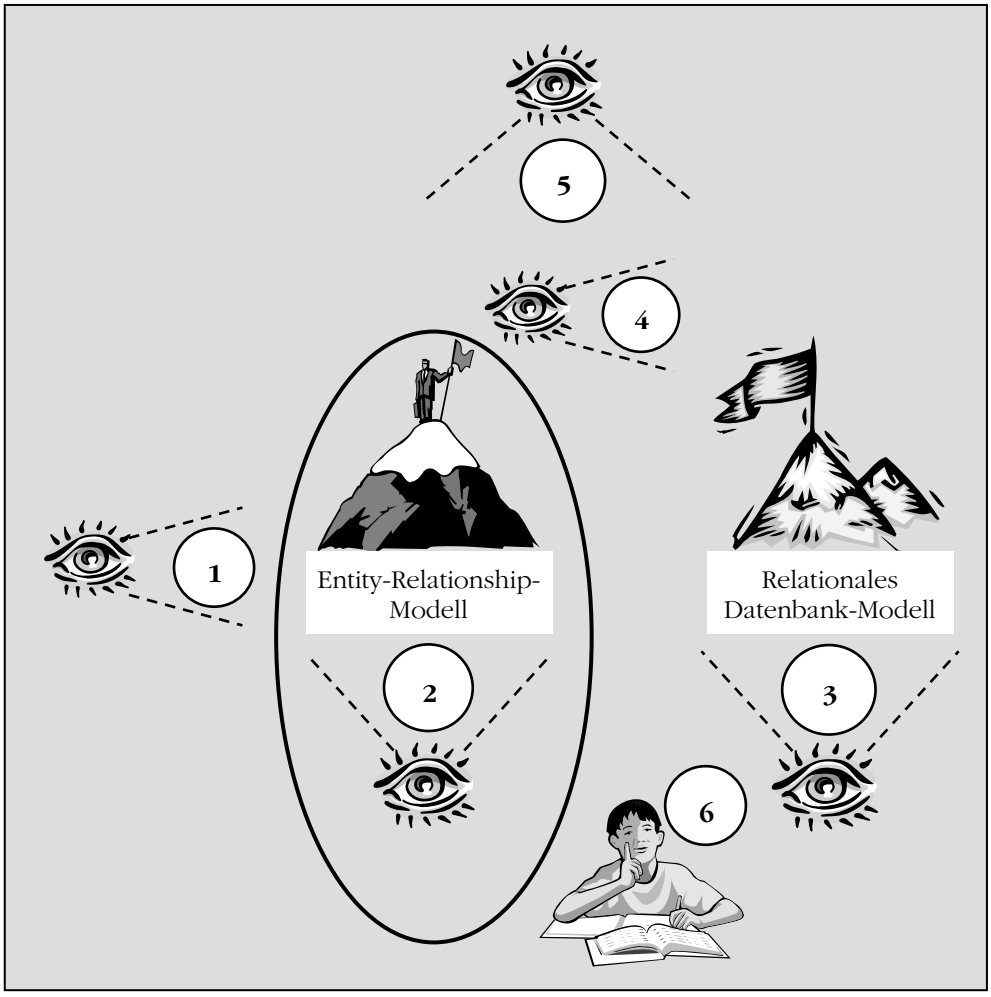


Abb. 2-1: Gegenstand des Kapitels 2

Bei der Entwicklung eines konzeptionellen Datenmodells sind zunächst vier Aufgaben zu lösen:

1. Festlegung der *Typen von Objekten*, über die Informationen gespeichert werden sollen (*Klassifizierung*).
2. *Reduktion der Vielfalt* aller bekannten Informationen über die Objekttypen auf einen definierten Satz relevanter Eigenschaften (*Abstraktion*).
3. Festlegung der Verfahrensweise, nach der die verschiedenen Objekte desselben Objekttyps voneinander unterschieden werden können (*Identifizierung*).
4. Beschreibung der *sachlogischen Zusammenhänge* zwischen den interessierenden Objekttypen.

PowerDesigner von Sybase

Die Lösung dieser Aufgaben wird in den folgenden Abschnitten ausführlich behandelt. Die Beschreibung der graphischen Sprache des Entity-Relationship-Modells erfolgt dabei unter Verwendung des CASE-Tools *„PowerDesigner“* der Firma Sybase.

Die Modellierungssprache des Entity-Relationship-Modells wird an Hand eines durchgängigen Beispiels erläutert. Für eine Landesschulbehörde wird der im Folgenden beschriebene Ausschnitt der Datenwelt analysiert:

Beispiel für die Daten-Modellierung

Beschreibung des betrachteten Ausschnitts der Datenwelt:

Die gespeicherten Daten sollen für das jeweilige Schuljahr gelten. In fünf Besprechungen werden die notwendigen Informationen über die Daten zusammengetragen:

1. *Es werden über alle größeren Orte des Landes Informationen gesammelt. Die Stadt Neustadt (Kreis Schwarzbach) hat beispielsweise 60 000 Einwohner. Die Goethe-Schule von Neustadt hat die Adresse: Wiesenweg 1, 19999 Neustadt. Fritz Fröhlich (geb. 31.12.1960, ledig, Personalnummer 54321) ist Lehrer. Englisch wird an den Schulen des Landes mindestens 2 Stunden/Woche und höchstens 6 Stunden/Woche unterrichtet. Die Klasse 11b der Goethe-Schule hat als Klassenraum den Unterrichtsraum 107. Über die Schüler muss bekannt sein: der Vorname, der Familienname, das Geburtsdatum und die Adresse.*
2. *Es gibt im Land größere Orte, die noch keine Schule besitzen. Neustadt besitzt dagegen 30 Schulen. Die Schiller-Schule von Neustadt ist gerade erst fertig gestellt: Sie hat weder Klassen noch hat sie mit Lehrern ein Beschäftigungsverhältnis abgeschlossen. Die Lehrer können mit mehreren Schulen des Landes ein Beschäftigungsverhältnis eingehen. Über arbeitslose*

Lehrer werden ebenfalls Informationen gespeichert. Jede Klasse einer Schule hat einen Lehrer als Klassenlehrer. Hat ein Lehrer seine Klasse zum Examen geführt, so ist er eine Weile kein Klassenlehrer. Fällt ein Klassenlehrer aus, übernimmt ein anderer zeitweilig seine Funktion. Ein Schüler gehört jeweils nur zu einer Klasse (kein Kurssystem). Eine Klasse besteht aus mindestens 15 und aus höchstens 30 Schülern.

3. *In jeder Schule ist einer der Lehrer Direktor und leitet die anderen Lehrer an. Über ihn werden aber dieselben Informationen gesammelt wie über die anderen Lehrer. Zwischen den Schulen können Patenschaften bestehen. Eine Schule kann zwar mehrere Paten haben, soll aber selbst nur für maximal eine andere Schule Pate sein.*

4. *Es sollen Informationen darüber gespeichert werden, welcher Lehrer welches Fach in welcher Klasse unterrichtet. Das Fach Hauswirtschaft wird im Land noch gar nicht unterrichtet. Frau Müller unterrichtet wegen Krankheit in diesem Schuljahr nicht. Herr Meier unterrichtet die 11b der Goethe-Schule im Unterrichtsraum 205 im Fach Englisch. Herr Meier hat ein Beschäftigungsverhältnis mit der Goethe-Schule seit dem 1.9.1980 und mit der Schiller-Schule seit dem 1.9.1985. Der Unterrichtsraum 205 der Neustädter Goethe-Schule hat eine Fläche von 50 m² mit 32 Sitzplätzen und ist kein Klassenraum. Ein Unterrichtsraum kann maximal 3 Klassen als Klassenraum zugeteilt werden. Der Unterrichtsraum 206 wird in diesem Schuljahr nicht für den Unterricht genutzt. Herr Lehmann gibt der Klasse 10b Unterricht in Geschichte in den Unterrichtsräumen 103 und 301.*

2.1 Klassifizierung der Objekte

Für die Abwicklung der zu automatisierenden Informationsverarbeitungs-Prozesse müssen über zahlreiche Objekte der betrieblichen Realität Daten gesammelt werden. So müssen beispielsweise über die Lehrer Fritz Fröhlich und Helga Herrlich, die Goethe-Schule und die Schiller-Schule, die Orte Neustadt und Althausen Daten gespeichert werden. Um nicht den Überblick über eine große Anzahl unterschiedlicher Objekte zu verlieren, gruppiert sie der Mensch in *Klassen*, die jeweils Objekte enthalten, über die dieselben Informationen gesammelt werden und mit denen prinzipiell die gleichen Informationsverarbeitungs-Prozesse durchgeführt werden. Er nimmt damit eine *Begriffsbildung*

Klassen

vor, die es ihm ermöglicht, bei seinen Denkprozessen von den Besonderheiten der konkreten Objekte abzusehen und nur das Typische der gebildeten Begriffsklassen zu berücksichtigen. In unserem Fall bildet er die Begriffsklassen „Lehrer“, „Schule“ und „Ort“. Abbildung 2-2 zeigt diesen Modellierungsprozess, bei dem die Objekte der betrieblichen Realität zu Objekttypen zusammengefasst werden.

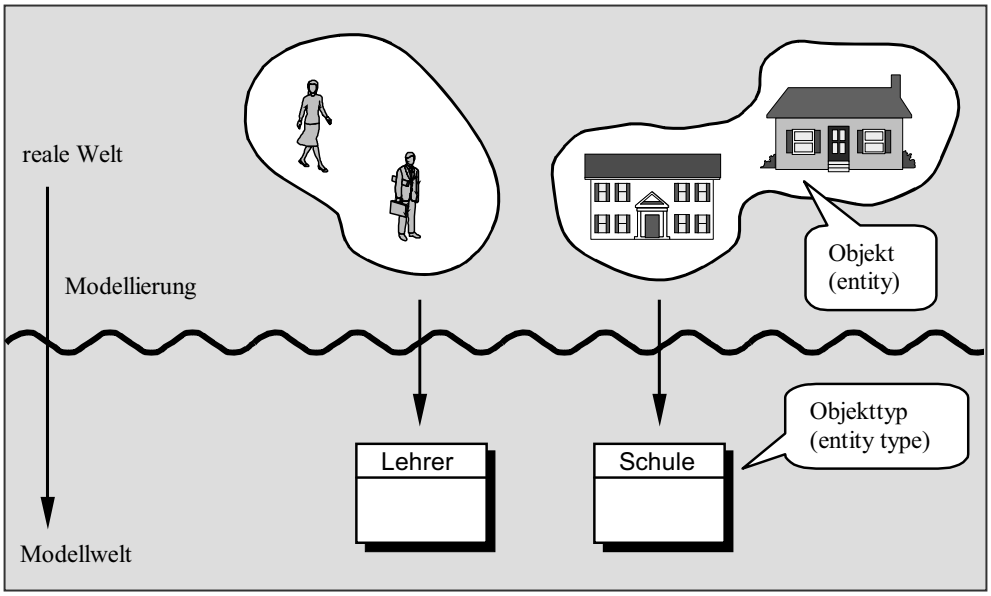


Abb. 2-2: Klassifizierung der Objekte durch Objekttypen

Objekt

Definition: Ein *Objekt* (engl. entity) ist ein Exemplar von Personen (z.B. Lehrer, Schüler), Gegenständen (z.B. Ort, Schule) oder nichtmateriellen Dingen (z.B. Beschäftigungsverhältnis, Unterrichtsverpflichtung), über das Informationen gespeichert werden.

Objekttyp

Definition: Ein *Objekttyp* (engl. entity type) ist eine durch einen Objekttyp-Namen eindeutig benannte Klasse von Objekten, über die dieselben Informationen gespeichert werden und die in prinzipiell gleicher Weise verarbeitet werden.

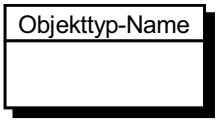
Im Zuge der Modellierung wird von den individuellen Objekten in der realen Welt abstrahiert. Statt dessen werden in der Modellwelt lediglich die relevanten Objekttypen betrachtet, die durch eindeutige Objekttyp-Namen unterschieden werden.

Für die Darstellung der Objekttypen gelten die folgenden syntaktischen Regeln:

Syntaxregeln für Objekttyp

Syntaxregeln:

1. Ein Objekttyp wird durch eine zweigeteilte Box dargestellt, in deren Kopfteil der Objekttyp-Name eingetragen wird:



2. Die Größe und die Position der Box sind bedeutungslos.
3. Der Objekttyp-Name steht im Singular und muss für das gesamte Datenmodell eindeutig sein.

Größe und Position der Box

Wenn auch die Größe der Box ohne inhaltliche Bedeutung ist, so wird doch bei den weiteren Betrachtungen ersichtlich, dass man einige Gestaltungsregeln einhalten sollte. Zunächst sollte die Box so breit gewählt werden, dass ein „sprechender" Objekttyp-Name möglichst ohne Abkürzung eingetragen werden kann. Mitunter ist auch eine bestimmte Länge der Box erforderlich, um mehrere Beziehungslinien mit der Box verbinden zu können (s. Abschnitt 2.4). Eine Box kann man innerhalb des Gesamtmodells zwar nicht richtig oder falsch, wohl aber geschickt oder ungeschickt positionieren. Eventuell sind im weiteren Verlauf der Modellierung Verschiebungen der Box sinnvoll, um die Beziehungslinien „kreuzungsfrei" zeichnen zu können. Auch das wird erst in den folgenden Abschnitten deutlich werden.

Objekttyp-Name

Der Objekttyp-Name soll nicht ein spezielles Objekt benennen (also nicht beispielsweise „Helga Herrlich" oder „Goethe-Schule"), sondern soll den Begriff bezeichnen, der durch die

Klassifizierung gebildet wird (also beispielsweise den Begriff „Lehrer“ oder den Begriff „Schule“).

Traditionelles Pendant

Die Informationsverarbeitung mit Hilfe elektronischer Datenverarbeitungsanlagen hat hinsichtlich der Datenspeicherung nur wenig prinzipiell neue Methoden entwickelt. Fast alle Konzepte, die wir im Zusammenhang mit dem Entity-Relationship-Modell besprechen werden, haben ihr Pendant in der traditionellen Informationsspeicherung. Zum besseren Verständnis der eingeführten Begriffe werden wir jeweils auf diese Zusammenhänge hinweisen. So entsprechen die Objekttypen den traditionellen Karteikästen und die Objekte eines Objekttyps den Karteikarten, die in einem Karteikasten eingeordnet sind. In traditioneller Arbeitsweise würde für „Helga Herrlich“ eine Karteikarte angelegt und im Karteikasten „Lehrer“ abgelegt werden. Durch die Festlegung der Objekttypen wird also das Ensemble der für die Informationsverarbeitung benötigten Karteikästen zusammengestellt.

Die Bildung der Begriffsklassen, die als Objekttypen im konzeptionellen Datenmodell auftreten, hängt entscheidend von den Anforderungen des jeweils zu modellierenden Gegenstandsbereichs ab. Aus der Sicht eines Großhändlers kann ein Unternehmen mit all seinen Bereichen als ein einziger Objekttyp angesehen werden, während dasselbe Unternehmen aus seiner eigenen Sicht detailliert mit seinen Bereichen, Abteilungen, Mitarbeitern, Werkshallen, Fahrzeugen usw. modelliert werden muss. Andererseits ist die Modellierung eines Unternehmens auch keine einmalige, in sich abgeschlossene Tätigkeit, weil im Lauf der Zeit Verschiebungen in den Begriffsbildungen des Unternehmens auftreten können, die eine Anpassung des konzeptionellen Datenmodells erforderlich machen.

2.2 Festlegung der relevanten Eigenschaften

Im vorangegangenen Abschnitt wurde die Klassifizierung von Objekten in Objekttypen unter dem Gesichtspunkt beschrieben, dass die Objekte eines gegebenen Objekttyps in prinzipiell gleicher Weise verarbeitet werden. Diese Verarbeitungsprozesse setzen voraus, dass über die Objekte eines Objekttyps bestimmte Angaben gespeichert werden, die sie entweder als Eingabeinformationen benötigen oder als Ausgabeinformationen liefern. Für jeden Objekttyp ist also anzugeben, welche *relevanten Eigenschaften* der in ihm zusammengefassten Objekte festgehalten werden müssen. Damit wird der durch den Objekttyp definierte

Relevante Eigenschaft

Begriff auf einen Satz relevanter Eigenschaften reduziert. Dieser Modellierungsprozess ist in Abbildung 2-3 dargestellt.

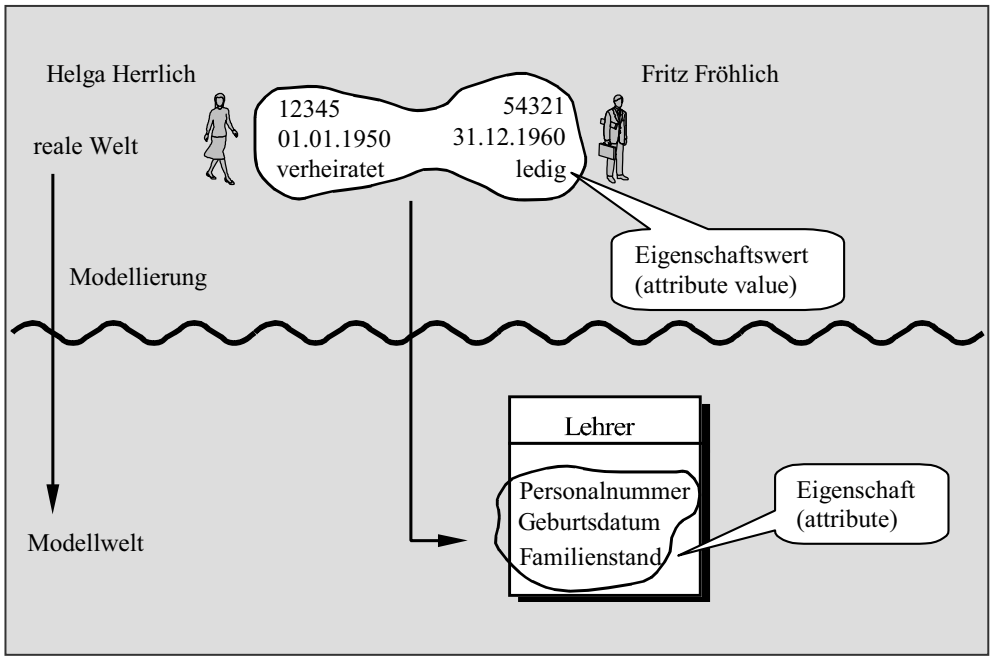


Abb. 2-3: Beschreibung der Objekttypen durch Eigenschaften

Eigenschaft

Definition: Eine *Eigenschaft* (engl. attribute) ist die Benennung für ein relevantes Merkmal aller Objekte, die in einem Objekttyp zusammengefasst werden.

Eigenschaftswert

Definition: Ein *Eigenschaftswert* (engl. attribute value) ist eine spezielle Ausprägung, die eine Eigenschaft für ein konkretes Objekt annimmt.

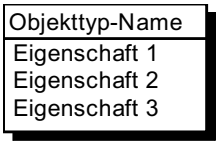
Im Zuge der Modellierung wird wiederum von den individuellen Besonderheiten der Objekte in der realen Welt abstrahiert. Statt der in der Realität zu beobachtenden konkreten Eigenschafts-

werte gibt man in der Modellwelt zunächst nur die Benennungen für die relevanten Merkmale – die Eigenschaften – an. Für die Darstellung der Eigenschaften gelten die folgenden syntaktischen Regeln:

Syntaxregeln für Eigenschaft

Syntaxregeln:

1. Die Benennung einer Eigenschaft wird in den unteren Teil der Box des Objekttyps eingetragen, für den die Eigenschaft als relevante Angabe gespeichert werden soll:



2. Die Reihenfolge der Eigenschaften ist bedeutungslos.
3. Die Benennung der Eigenschaft steht im Singular und muss für den Objekttyp eindeutig sein.

Reihenfolge der Eigenschaften

Wenn auch die Reihenfolge der Eigenschaften eines Objekttyps ohne inhaltliche Bedeutung ist, so ist es meist doch sinnvoll, die wichtigsten Eigenschaften an den Anfang der Auflistung zu setzen. Das betrifft insbesondere jene Eigenschaften, die für die Unterscheidbarkeit (Identifizierung) der Objekte des Objekttyps verantwortlich sind (s. Abschnitt 2.3). Bei großen Datenmodellen oder bei einer großen Anzahl von Eigenschaften für einen Objekttyp ist es nicht mehr praktikabel, alle Eigenschaften in der Box des Objekttyps aufzuführen. In diesen Fällen gibt man – wenn überhaupt – nur die wichtigsten Eigenschaften in der Box an und listet die vollständige Eigenschaftsliste in einer gesonderten Dokumentation auf.

1. Normalform

Die Forderung, die Benennung einer Eigenschaft stets im Singular anzugeben, soll verhindern, dass einem Objekttyp eine Eigenschaft zugewiesen wird, die *gleichzeitig mehrere Eigenschaftswerte* annehmen kann. Das würde nämlich die Bedingung der sog. 1. Normalform verletzen, die wir im Abschnitt 2.6.1 behandeln werden.

Als Eigenschaftsbenennung darf nie ein spezieller Eigenschaftswert verwendet werden (also nicht beispielsweise „01.01.1950“ oder „ledig“), sondern stets ein Name, der die Menge aller möglichen Werte dieser Eigenschaft - die sog. *Wertedomäne* - kennzeichnet (also die Benennungen „Geburtsdatum“ bzw. „Familienstand“). Da die Eigenschaftsbenennung nur im Kontext des jeweiligen Objekttyps gilt, sollte sie den Objekttyp-Namen nicht als Bestandteil aufweisen. Die Eigenschaft des Namens des Objekttyps „Schule“ sollte also nicht als „Schulname“, sondern einfach als „Name“ angegeben werden.

Komplexe Eigenschaft zerlegen?

Mitunter steht man bei der Festlegung der Eigenschaften vor der Frage, wie detailliert die relevanten Angaben zu zerlegen sind. Kann beispielsweise die Adresse einer Schule als eine geschlossene Eigenschaft betrachtet werden oder muss sie in „Straßenname“, „Hausnummer“, „Postleitzahl“ und „Ortsname“ zerlegt werden? Diese Entscheidung ist nur bei Kenntnis der Informationsverarbeitungs-Prozesse möglich, die mit diesen Angaben durchgeführt werden sollen. Wird die Adresse immer nur als Ganzes verarbeitet, kann sie als eine geschlossene Eigenschaft gespeichert werden. Benötigt man dagegen ihre Elemente einzeln, weil man beispielsweise nach allen Schulen suchen möchte, die in einem bestimmten Postleitzahlbezirk liegen, muss man die Elemente als gesonderte Eigenschaften angeben.

Eigenschaft oder Objekttyp?

Problematisch ist mitunter auch die Entscheidung, ob speicherwürdige Informationen als Eigenschaften oder als ein eigenständiger Objekttyp betrachtet werden sollen. Um einen eigenständigen Objekttyp handelt es sich immer dann, wenn er für das Unternehmen bedeutsame Objekte enthält, die relevante individuelle Eigenschaften besitzen. Auf dieses Problem werden wir in den Abschnitten 2.5.3 und 5.1 näher eingehen.

Die Festlegung der Eigenschaften, die für einen Objekttyp relevant sind, kann nur bei genauer Kenntnis des jeweils zu modellierenden Gegenstandsbereichs erfolgen. So ist es beispielsweise nicht möglich, die relevanten Eigenschaften eines Kunden unabhängig vom Informationsbedarf des jeweiligen Unternehmens anzugeben. Ist beispielsweise die Haarfarbe des Kunden eine speicherwürdige Eigenschaft? Für den Produzenten von Kaffeemaschinen sicherlich nicht, für den Betreiber eines Friseursalons dagegen auf jeden Fall. Das impliziert, dass man Modellierungsergebnisse des einen Unternehmens nicht kritiklos in einem anderen Unternehmen nachnutzen kann. Sogenannte „Referenzmodelle“ können allenfalls als „Steinbruch“ verwendet werden,

um die Modellierungselemente (Objekttypen und Eigenschaften) an die individuellen Besonderheiten anzupassen (vgl. [FETT08]).

Traditionelles Pendant

Die Festlegung der relevanten Eigenschaften hat ihre Entsprechung in der traditionellen Informationsspeicherung. Die Eigenschaften eines Objekttyps A entsprechen dabei den Feldern, die auf einer Karteikarte des Karteikastens A zur Aufnahme der relevanten Informationen angelegt werden. Durch die ausgewählten Eigenschaften wird also die einheitliche Struktur aller Karteikarten eines Karteikastens festgelegt.

Die Festlegung der relevanten Eigenschaften eines Objekttyps ist keine einmalige, abgeschlossene Tätigkeit. Durch Veränderung von Geschäftsprozessen und die Hinzunahme neuer Aufgaben (beispielsweise die Auszahlung des Kindergeldes durch das Unternehmen) müssen eventuell weitere Eigenschaften aufgenommen werden.

Schulbeispiel

Wenden wir uns nun unserem Schulbeispiel zu und führen die besprochenen ersten beiden Schritte der Datenmodellierung aus:

1. *Es werden über alle größeren Orte des Landes Informationen gesammelt. Die Stadt Neustadt (Kreis Schwarzbach) hat beispielsweise 60 000 Einwohner. Die Goethe-Schule von Neustadt hat die Adresse: Wiesenweg 1, 19999 Neustadt. Fritz Fröhlich (geb. 31.12.1960, ledig, Personalnummer 54321) ist Lehrer. Englisch wird an den Schulen des Landes mindestens 2 Stunden/Woche und höchstens 6 Stunden/Woche unterrichtet. Die Klasse 11b der Goethe-Schule hat als Klassenraum den Unterrichtsraum 107. Über die Schüler muss bekannt sein: der Vorname, der Familienname, das Geburtsdatum und die Adresse.*

Zunächst wird der Objekttyp „Ort“ mit seinen Eigenschaften „Name“, „Kreis“ und „Einwohnerzahl“ eingeführt. Für den Kreis wird hier kein eigener Objekttyp verwendet, weil in unserem Gegenstandsbereich über den Kreis keine Informationen gespeichert werden.

Der Objekttyp „Schule“ hat die Eigenschaften „Name“, „Straße“, „Hausnummer“ und „Postleitzahl“. Dabei wird die Adresse – wie bereits oben diskutiert - in ihre Bestandteile zerlegt, die jeweils als individuelle Eigenschaften fungieren. Der Ortsname muss dabei nicht als Eigenschaft der Schule gespeichert werden, weil er bereits Eigenschaft des Ortes ist. Im Abschnitt 2.4.1 werden wir sehen, wie die Schule (durch einen Beziehungstyp) mit dem Ort verbunden wird, so dass man bei der Verarbeitung der

Schuldaten jederzeit zum dazugehörigen Ort wechseln kann. Bei unserem heute gültigen Postleitzahlsystem ist die Postleitzahl eine reine Benennung des Postzustellbezirks, die mit Ortsgrenzen nicht unbedingt zusammenfallen muss. Größere Ortschaften haben mehrere Postleitzahlen, eine Postleitzahl kann andererseits mehrere kleine Ortschaften umfassen oder gar keine (im Falle von Großpostkunden). Damit kann die Postleitzahl nicht als Eigenschaft des Ortes, sondern nur als Element einer Adresse angesehen werden und ist deshalb eine Eigenschaft der Schule.

Der Objekttyp „Lehrer“ hat die Eigenschaften „Vorname“, „Familienname“, „Geburtsdatum“, „Familienstand“ und „Personalnummer“.

Für den Objekttyp „Fach“ sind die „Bezeichnung“ und die minimale sowie die maximale Stundenzahl je Woche („Min. Stunden/Woche“ und „Max. Stunden/Woche“) als Eigenschaften anzugeben.

Der Objekttyp „Klasse“ hat die Eigenschaften „Bezeichnung“ und „Klassenraum“. Der Schulname muss wiederum nicht als Eigenschaft der Klasse angegeben werden, weil er bereits Eigenschaft der Schule ist. Die Verbindung der Klasse zur zugehörigen Schule erfolgt später durch einen entsprechenden Beziehungstyp. Zum Klassenraum sind bisher keine speicherwürdigen Eigenschaften bekannt – deshalb wird er hier als Eigenschaft des Objekttyps „Klasse“ und nicht als eigener Objekttyp modelliert.

Dem Objekttyp „Schüler“ werden die Eigenschaften „Vorname“, „Familienname“, „Geburtsdatum“ und „Adresse“ zugeordnet. Dabei sei an dieser Stelle die Entscheidung getroffen, dass die Anschrift nicht weiter zerlegt werden muss.

Die Abbildung 2-4 zeigt unser Datenmodell im gegenwärtigen Bearbeitungszustand.

2.3 Festlegung der Identifizierung

Der Objekttyp wurde als eine Klasse definiert, in der sämtliche Objekte zusammengefasst sind, die dem Klassenbegriff zugeordnet werden. Im mathematischen Sinne ist die Klasse eine Menge, in der die Objekte als Elemente der Menge zusammengefasst sind. Die Elemente einer Menge müssen jedoch voneinander unterscheidbar sein. Deshalb muss festgelegt werden, auf welche Weise ein Objekt innerhalb des Objekttyps identifiziert werden kann.

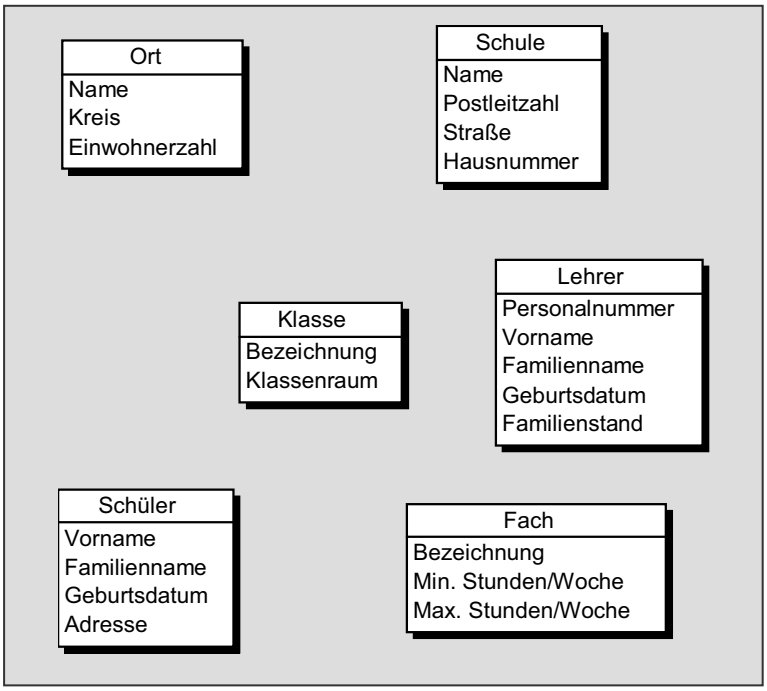


Abb. 2-4: Datenmodell für das Schulbeispiel mit Objekttypen und Eigenschaften

Möglichkeiten der Identifizierung

Die drei Möglichkeiten der Identifizierung, die wir in diesem Abschnitt besprechen werden, tragen nur vorläufigen Charakter und werden später im Abschnitt 2.4.4 ergänzt. Nach unserem bisherigen Wissensstand werden als Angaben über die Objekte eines Objekttyps lediglich die Werte ihrer relevanten Eigenschaften gespeichert. Vorläufig können wir für die Identifizierung eines Objekts innerhalb eines Objekttyps also nur die Werte verwenden, die die Eigenschaften für ein konkretes Objekt annehmen. Wir unterscheiden drei Varianten der Identifizierung durch Eigenschaften:

Eine unikale Eigenschaft

1. Identifizierung durch *eine einzige Eigenschaft*. Mitunter ist eine Eigenschaft so geartet, dass jeder Wert, den sie annehmen kann, garantiert nur bei einem Objekt des Objekttyps auftritt. Man sagt dann, dass der Eigenschaftswert „unikal“ ist, dass er also an lediglich ein Objekt gebunden ist. Durch die Angabe dieses Eigenschaftswertes ist dann das Objekt eindeutig identifiziert. Beispielsweise ist jeder Wert, den die

Eigenschaft „Bezeichnung“ des Objekttyps „Fach“ annehmen kann, jeweils nur an ein Objekt des Objekttyps gebunden. So identifiziert der Eigenschaftswert „Englisch“ das entsprechende Fach eindeutig.

Eigenschafte-Kombination

2. Identifizierung durch eine *Kombination von Eigenschaften*. Häufig ist keine der Eigenschaften eines Objekttyps geeignet, die Identifizierung der Objekte allein zu bewerkstelligen. Dann versucht man, eine solche (minimale) Kombination von Eigenschaften zu finden, für die die spezielle Kombination der Eigenschaftswerte, die bei einem Objekt angenommen werden, garantiert nur bei diesem Objekt auftritt. Durch Angabe dieser speziellen Kombination von Eigenschaftswerten kann dann das Objekt eindeutig identifiziert werden. Beispielsweise kann im Allgemeinen weder der „Name“ noch der „Kreis“ des Objekttyps „Ort“ allein für sich die Identifizierung eines Ortes bewirken. Der Name „Neustadt“ kann im betrachteten Bundesland mehrfach auftreten, und die Angabe des Kreises „Schwarzbach“ trifft im allgemeinen auf viele Orte zu. Kombiniert man dagegen beide Eigenschaften, so ist die Kombination „Neustadt+Schwarzbach“ unikal, kann also die Identifizierung des Ortes gewährleisten. Kommt man bei der Analyse des Gegenstandsbereichs zu der Überzeugung, dass man nicht garantieren kann, dass jede dieser Werte-Kombinationen nur bei einem einzigen Objekt vorkommt, müssen – soweit möglich – weitere Eigenschaften in die Kombination eingeschlossen werden.

Organisatorische Eigenschaft

3. Identifizierung mit Hilfe einer *organisatorischen Eigenschaft*. Wenn es nicht möglich ist, unter den relevanten Eigenschaften eines Objekttyps eine einzelne Eigenschaft oder eine Kombination von Eigenschaften zu finden, die die eindeutige Identifizierung ermöglichen, oder wenn die möglichen Identifizierungsvarianten zu umständlich wären, kann man eine neue künstliche Eigenschaft einführen, für die die Unikalität jedes Eigenschaftswertes durch organisatorische Maßnahmen gesichert wird. Beispielsweise kann es recht problematisch sein, eine solche Kombination von natürlichen Eigenschaften zu wählen, die einen Lehrer eindeutig identifiziert. Deshalb wird die „Personalnummer“ als eine neue „organisatorische Eigenschaft“ eingeführt, die die Identifizierung eines Lehrers ermöglicht. Dabei muss allerdings durch organisatorische Maßnahmen sichergestellt werden, dass jede Personalnummer nur *einem* Lehrer zugewiesen wird. In unserem Beispiel

ermöglicht die Angabe der Personalnummer „54321“ die eindeutige Identifizierung des Lehrers Fritz Fröhlich.

Die Identifizierung der Objekte eines Objekttyps mittels einer „organisatorischen Eigenschaft“ ist bei der automatisierten Datenverarbeitung eine beliebte Vorgehensweise. Sie wird häufig selbst dann angewendet, wenn natürliche identifizierende Eigenschaften vorhanden sind. Meist wird auf eine laufende Nummerierung der Objekte zurückgegriffen (beispielsweise auf die Personalnummer, die Kundennummer, die Artikelnummer usw.). Das hat den Vorteil, dass kurze identifizierende Eigenschaftswerte entstehen. Der Nachteil solcher „organisatorischen Eigenschaften“ besteht darin, dass die Unikalität der Eigenschaftswerte nur innerhalb eines beschränkten Bereiches gilt. So waren beispielsweise die Berliner Telefonnummern zwar in Westberlin und Ostberlin unikal, nach der Wiedervereinigung der beiden Stadthälften traten jedoch viele Telefonnummern doppelt auf, so dass Telefonteilnehmern neue Nummern zugeordnet werden mussten. Ein weiterer Nachteil der „organisatorischen Eigenschaften“ besteht darin, dass die Eigenschaftswerte nicht „sprechend“ sind, dass sie also im allgemeinen für den Bearbeiter ohne Semantik sind. Damit sind sie schlecht erinnerbar und bergen in sich die Gefahr von Verwechslungen und Fehleingaben. Die Verwendung von Nummern zur Identifizierung von Personen stößt außerdem auf moralische Bedenken (oder sollte das zumindest tun). Wenn beispielsweise Kundennummern zwar für die betriebsinterne Verarbeitung verwendet werden, versucht man vielerorts doch, sie aus Höflichkeit dem Kunden gegenüber nicht nach außen sichtbar werden zu lassen.

Mitunter gibt es für einen Objekttyp mehrere alternative Möglichkeiten der Identifizierung. So könnte ein Lehrer alternativ entweder durch seine Personalnummer oder durch die Kombination „Vorname+Familienname+Geburtsdatum“ identifiziert werden. Im Rahmen dieser Darlegung werden wir aber aus Gründen der Einfachheit für jeden Objekttyp nur *eine* Identifizierungsvariante betrachten.

Identifizierende Eigenschaft

Definition: Eine Eigenschaft ist eine *identifizierende* (bzw. *teilidentifizierende*) *Eigenschaft*, wenn sie die Identifizierung eines Objekts innerhalb eines Objekttyps herbeiführt (bzw. zu dessen Identifizierung beiträgt).

Beschreibende Eigenschaft

Definition: Eine Eigenschaft ist eine *beschreibende Eigenschaft*, wenn sie zwar speicherwürdige Aspekte des jeweiligen Objekttyps beschreibt, jedoch nicht für die Identifizierung der Objekte des Objekttyps herangezogen wird.

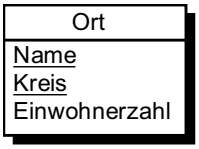
Die Festlegung der Identifizierungsform muss für jeden Objekttyp mit großer Sorgfalt erfolgen. Relationale Datenbank-Managementsysteme, für die wir unsere Modellierung durchführen, lassen es nämlich nicht zu, dass zwei Objekte desselben Objekttyps in der Kombination ihrer (teil)identifizierenden Merkmale übereinstimmen. Bleibt bei der gewählten Identifizierungsform ein „Restrisiko“ hinsichtlich der Unikalität der (teil)identifizierenden Merkmale und treten dann bei der praktischen Datenbankarbeit tatsächlich zwei Objekte mit übereinstimmenden Werten ihrer (teil)identifizierenden Merkmale auf, so lehnt das Datenbank-Managementsystem die Speicherung des zweiten Objekts ab. Eine gründlichere Diskussion der Forderungen, die für die Identifizierungsform im Umfeld relationaler Datenbank-Managementsysteme gelten, erfolgt im Abschnitt 3.1.1.

Für die Markierung der (teil)identifizierenden Eigenschaften gelten die folgenden syntaktischen Regeln:

Syntaxregeln für identifizierende Eigenschaft

Syntaxregeln:

1. Eine identifizierende Eigenschaft (bzw. jede teilidentifizierende Eigenschaft) wird durch Unterstreichung kenntlich gemacht:



2. Die Position einer identifizierenden bzw. teilidentifizierenden Eigenschaft innerhalb der Liste der Eigenschaften des Objekttyps ist bedeutungslos.

Wenn auch die Position der identifizierenden bzw. teilidentifizierenden Eigenschaften ohne inhaltliche Bedeutung ist, so ist es doch zweckmäßig, sie auf Grund ihrer besonderen Bedeutung an den Anfang der Eigenschaftsliste zu setzen. Bei großen Datenmodellen gibt man häufig nur die (teil)identifizierenden Eigenschaften in der Box an und listet die vollständige Eigenschaftsliste in einer gesonderten Dokumentation auf.

Klassisches Pendant

Bei der traditionellen Informationsspeicherung mit Hilfe von Karteikästen entspricht der Identifizierung die Festlegung eines Kennbegriffs: Üblicherweise wird im Kopf jeder Karteikarte für das betreffende Objekt ein Wert angegeben, der das Objekt innerhalb des Karteikastens identifiziert[5]. Um eine bestimmte Karteikarte manuell schneller auffinden zu können, werden die Karten des Karteikastens nach diesem Kennbegriff sortiert.

Schulbeispiel

Treffen wir nun die Entscheidung für die Identifizierungsformen der Objekttypen in unserem Schulbeispiel!

Die Objekte des Objekttyps „Ort“ werden – wie bereits diskutiert – durch die Kombination der beiden Eigenschaften „Name“ und „Kreis“ identifiziert.

Die Objekte im Objekttyp „Schule“ können nicht allein durch die Eigenschaft „Name“ identifiziert werden, weil es im betrachteten Bundesland beispielsweise mehrere Schulen mit dem Namen „Goethe-Schule“ geben kann. Nimmt man jedoch die „Postleitzahl“ als teilidentifizierende Eigenschaft hinzu, ist eine eindeutige Identifizierung gegeben. Dabei wird allerdings vorausgesetzt, dass die Landesschulbehörde organisatorisch sichert, dass in einem gegebenen Postleitzahlbezirk keine zweite Schule mit demselben Namen eingerichtet wird.

Die Objekte des Objekttyps „Lehrer“ werden durch die „Personalnummer“ identifiziert. Dabei wird vorausgesetzt, dass die Vergabe der Personalnummern auf Landesebene erfolgt und organisatorisch gesichert wird, dass eine konkrete Personalnummer nur einem Lehrer des Bundeslandes zugeordnet wird.

Die Objekte des Objekttyps „Fach“ werden – wie besprochen - allein durch die „Bezeichnung“ identifiziert.

Für die Identifizierung der Objekte des Objekttyps „Klasse“ reicht die Eigenschaft „Bezeichnung“ nicht aus, weil es im Bundesland

---

5 Dieser Kennbegriff wird mitunter auch als „Reiter“ bezeichnet.

natürlich beispielsweise viele Klassen mit der Bezeichnung „11b“ gibt. Die Kombination mit der Eigenschaft „Klassenraum“ würde da auch nicht viel helfen, und weitere natürliche Eigenschaften stehen nicht zur Verfügung. Solche Objekttypen, deren Eigenschaften nicht ausreichen, um ihre Objekte zu identifizieren, bezeichnet man als *„schwache Objekttypen“*. Ein Objekt eines solchen „schwachen Objekttyps“ kann man nur dadurch identifizieren, dass man zusätzlich noch die Beziehung berücksichtigt, die das Objekt zu einem Objekt eines anderen Objekttyps eingeht. Das wird aber erst im Abschnitt 2.4.4 erläutert werden. Vorläufig legen wir als einzige teilidentifizierende Eigenschaft des Objekttyps „Klasse“ nur die „Bezeichnung“ fest und merken uns, dass damit keine vollständige Identifizierung möglich ist.

Schwacher Objekttyp

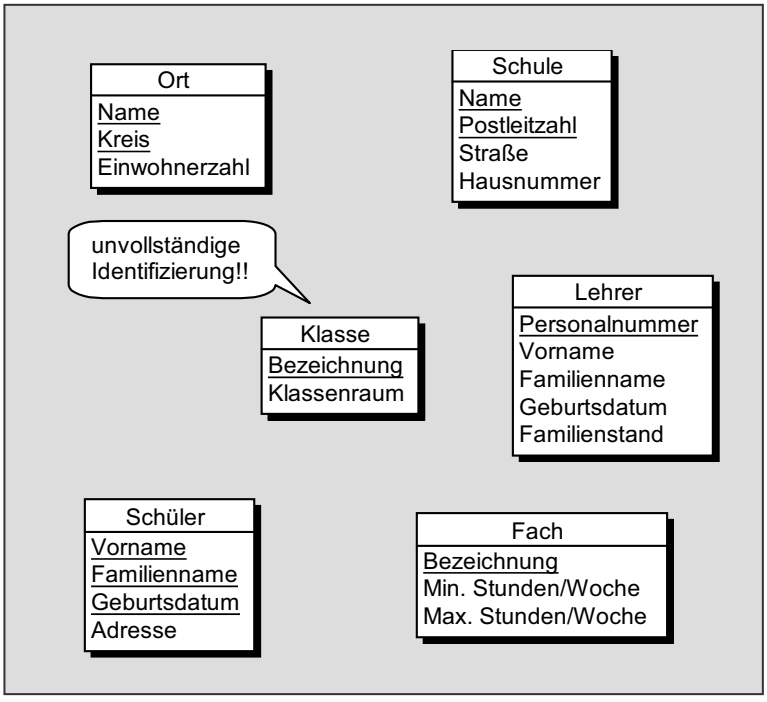


Abb. 2-5: Datenmodell für das Schulbeispiel mit Objekttypen, Eigenschaften und Angaben zur Identifizierung

Personalnummern werden in Schulen (noch) nicht verwendet. Deshalb bleibt nur die Variante der Identifizierung eines Schülers durch seine natürlichen Eigenschaften. Für eine Identifizierung der Objekte des Objekttyps „Schüler“ ist die Eigenschafte-Kombination „Vorname+Familienname+Geburtsdatum“ geeignet.

Den aktuellen Bearbeitungsstand des Datenmodells zeigt die Abbildung 2-5.

2.4 Beschreibung der sachlogischen Zusammenhänge zwischen den Objekttypen

Bisher wurden im Rahmen der Datenmodellierung die speicherwürdigen Objekte mit ihren relevanten Eigenschaften lediglich isoliert beschrieben. In der Praxis stehen die interessierenden Objekte jedoch in vielfältiger Weise miteinander im Zusammenhang: Schulen liegen in Orten, Lehrer unterrichten an Schulen, Schüler gehören zu Klassen usw. Aber auch Objekte, die demselben Objekttyp angehören, können miteinander in sachlogischem Zusammenhang stehen: Lehrer leiten Lehrer an, Schulen übernehmen eine Patenschaft für Schulen usw. Diese Zusammenhänge müssen auch im Datenmodell dargestellt werden, denn sie bringen wesentliche Aspekte des betrachteten Gegenstandsbereichs zum Ausdruck.

Die sachlogischen Zusammenhänge zwischen den Objekten werden in zwei Gruppen von Beziehungstypen unterteilt:

1. *Duale Beziehungstypen:* Dies sind Beziehungstypen, die den Zusammenhang zwischen jeweils zwei Objekten beschreiben, die verschiedenen Objekttypen angehören.
2. *Rekursiv-Beziehungstypen:* Bei diesen Beziehungstypen stehen Objekte in sachlogischem Zusammenhang, die aus demselben Objekttyp stammen.

Im folgenden Abschnitt werden zunächst die dualen Beziehungstypen erläutert. In diesem Zusammenhang werden einige Probleme behandelt, die auch für die Rekursiv-Beziehungstypen gelten. Diese werden dann im Abschnitt 2.4.5 näher untersucht.

2.4.1 Duale Beziehungstypen

Die Abbildung 2-6 zeigt, wie die in der realen Welt beobachtete Menge sachlogischer Zusammenhänge zwischen jeweils zwei Objekten als eine „verallgemeinerte“ Verbindung zwischen zwei Objekttypen modelliert wird.

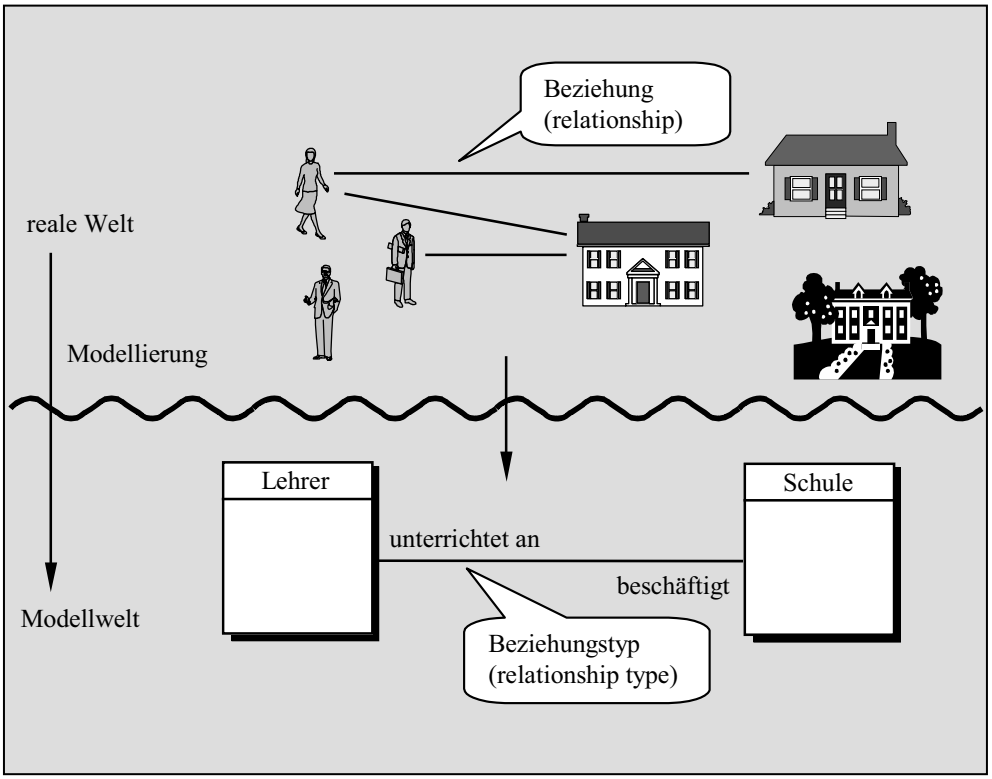


Abb. 2-6: Modellierung der Beziehungen zwischen jeweils zwei Objekten

Für die neu eingeführten Begriffe gelten die folgenden Definitionen:

Beziehung

<u>Definition</u>: Eine *Beziehung* (engl. relationship) kennzeichnet den konkreten Zusammenhang zwischen zwei realen Objekten.

Beziehungstyp

Definition: Ein (*dualer*) *Beziehungstyp* (engl. relationship type) beschreibt den typmäßigen sachlogischen Zusammenhang, der zwischen den Objekten zweier Objekttypen besteht.

In diesem Abschnitt betrachten wir ausschließlich solche Beziehungstypen, die den Zusammenhang zwischen *zwei verschiedenen* Objekttypen beschreiben, und die wir präziser als *duale Beziehungstypen* bezeichnen. Im Abschnitt 2.4.5 werden wir dann die *Rekursiv-Beziehungstypen* kennen lernen, die den Zusammenhang zwischen Objekten wiedergeben, die demselben Objekttyp angehören. Wenn Verwechslungen ausgeschlossen sind, werden wir aber vereinfachend statt von „dualen Beziehungstypen" stets von „Beziehungstypen" sprechen.

Während in der realen Welt der Zusammenhang zwischen zwei konkreten Objekten beobachtet wird, beschreibt man in der Modellwelt das verallgemeinerte Wechselspiel zwischen zwei Objekttypen. Im mathematischen Sinne ist ein Beziehungstyp zwischen den Objekttypen A und B die Menge der Beziehungen zwischen jeweils einem Objekt aus dem Objekttyp A und einem Objekt aus dem Objekttyp B. Die für den Beziehungstyp formulierten Angaben müssen somit für alle konkreten Beziehungen zwischen den betrachteten Objekttypen gültig sein.

Beziehungstyp-Richtung

Der sachlogische Zusammenhang zwischen den Objekten zweier Objekttypen, der durch einen Beziehungstyp beschrieben wird, besteht immer *in beiden Richtungen:* Lehrer stehen beispielsweise in einem Zusammenhang mit Schulen (sie unterrichten an Schulen) und Schulen stehen andererseits im Zusammenhang mit Lehrern (sie beschäftigen Lehrer). Jede der beiden *Beziehungstyp-Richtungen* wird durch drei Angaben näher bestimmt:

1. durch eine *Benennung*, die die Semantik des sachlogischen Zusammenhangs ausdrückt,
2. durch eine Angabe zur *Optionalität* und
3. durch eine Angabe zur *Kardinalität.*

Benennung der Beziehungstyp-Richtung

Die *Benennung* kennzeichnet den sachlogischen Zusammenhang, in dem die Objekte des Objekttyps A zu den Objekten des Objekttyps B stehen. Betrachtet man den Zusammenhang zwischen Lehrer und Schule, so könnte man sich für verschiedene sachlogische Zusammenhänge interessieren:

- „Ein Lehrer *war Schüler in* einer Schule“
- „Ein Lehrer *hat sich beworben an* einer Schule“
- „Ein Lehrer *wohnt in der Nähe von* einer Schule“
- „Ein Lehrer *unterrichtet an* einer Schule“
- usw.

Welcher dieser Zusammenhänge tatsächlich gespeichert werden soll, wird durch die Benennung zum Ausdruck gebracht. Im Schulbeispiel ist es natürlich der sachlogische Zusammenhang *„unterrichtet an“*.

Optionalität der Beziehungstyp-Richtung

Die Angabe zur *Optionalität* der Beziehungstyp-Richtung vom Objekttyp A zum Objekttyp B bezieht sich auf die folgende Frage: „*Muss* jedes Objekt des Objekttyps A *mit mindestens einem* Objekt des Objekttyps B in Beziehung stehen?“ Je nach Antwort auf diese Frage unterscheidet man zwei Fälle:

- „Ja“: Die Beziehungstyp-Richtung wird als *nicht-optional* (obligatorisch, „Muss-Beziehungstyp-Richtung“) bezeichnet.
- „Nein“: Die Beziehungstyp-Richtung wird als *optional* (fakultativ, „Kann-Beziehungstyp-Richtung“) bezeichnet.

Kardinalität der Beziehungstyp-Richtung

Zur Angabe der *Kardinalität* der Beziehungstyp-Richtung vom Objekttyp A zum Objekttyp B stellt man sich die folgende Frage: „*Kann* ein Objekt des Objekttyps A *mit mehreren* Objekten des Objekttyps B in Beziehung stehen?“ Je nach Antwort auf diese Frage unterscheidet man wiederum zwei Fälle:

- „Ja“: Der Beziehungstyp-Richtung wird die *Kardinalität N* (steht für eine beliebige Zahl größer als 1) zugeordnet.
- „Nein“: Der Beziehungstyp-Richtung wird die *Kardinalität 1* zugeordnet.

Kardinalitäts-Beschränkung

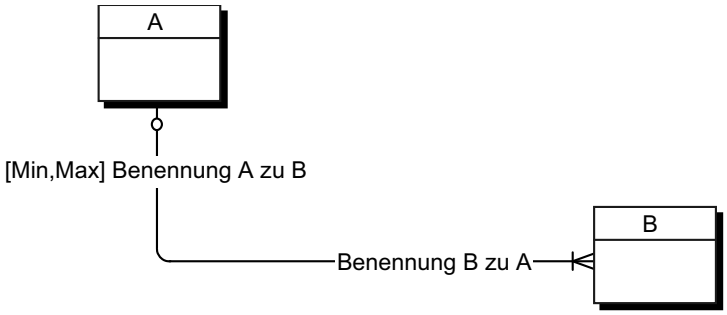
Mitunter werden im zu modellierenden Gegenstandsbereich für die Kardinalität N einer Beziehungstyp-Richtung minimale bzw. maximale Werte vorgegeben. So muss in unserem Schulbeispiel eine Klasse aus mindestens 15 und aus höchstens 30 Schülern bestehen. Sind solche Grenzen bekannt, werden sie als Präzisierung der Kardinalität N angegeben. Dabei kann die minimale Grenze den Wert 1 natürlich nicht unterschreiten[6].

[6] Die Kardinalitäts-Beschränkung betrifft *ausschließlich* die Kardinalität; die Optionalität ist davon nicht betroffen.

Syntaxregeln für den dualen Beziehungstyp

Syntaxregeln:

1. Ein Beziehungstyp zur Kennzeichnung eines sachlogischen Zusammenhangs zwischen den Objekttypen A und B wird als Verbindungslinie zwischen A und B dargestellt.
2. Die Benennung der Beziehungstyp-Richtung vom Objekttyp A zum Objekttyp B steht in der Nähe des Objekttyps A und umgekehrt.
3. Eine optionale Beziehungstyp-Richtung wird durch einen Kreis am Linienende (interpretierbar als „mindestens 0“), eine nicht-optionale Beziehungstyp-Richtung durch einen senkrechten Strich (interpretierbar als „mindestens 1“) gekennzeichnet.
4. Die Kardinalität 1 einer Beziehungstyp-Richtung wird durch einen einfachen Strich am Linienende (interpretierbar als „höchstens 1“), die Kardinalität N durch drei auseinanderstrebende Striche – durch einen „Krähenfuß“ – dargestellt (interpretierbar als „1 oder mehr als 1“).
5. Minimale und maximale Grenzen für die Kardinalität N werden in eckigen Klammern vor der Benennung der Beziehungstyp-Richtung notiert:



Die Benennung einer Beziehungstyp-Richtung „A zu B“ ist so zu wählen, dass die folgende syntaktische Konstruktion einen fehlerfreien Satz ergibt:

Ein(e)	<Objekttyp-Name A>
	<Benennung A zu B>
ein(e/en/em/er)	<Objekttyp-Name B>

Beispiel: „Ein Ort *besitzt* eine Schule“ und „Eine Schule *liegt in* einem Ort“. Außerdem sollte die Benennung möglichst präzise den sachlogischen Zusammenhang zum Ausdruck bringen, den die Objekte des Objekttyps A mit den Objekten des Objekttyps B eingehen. Wesentlich präziser als die Aussage „Eine Schule *hat* einen Lehrer“ (als was denn? – als Direktor, als Hausmeister, als Inspektor oder was?) ist die Aussage „Eine Schule *hat ein Beschäftigungsverhältnis abgeschlossen mit* einem Lehrer“ Im Interesse einer kürzeren Darstellung verzichtet man häufig auf den unbestimmten Artikel und kennzeichnet die Beziehungstyp-Richtung einfach als „Schule *hat ein Beschäftigungsverhältnis abgeschlossen mit* Lehrer“

16 Klassen von Beziehungstypen

Da die Fragen nach der Optionalität und Kardinalität jede für sich zwei verschiedene Antworten zulassen, gibt es für jede Beziehungstyp-Richtung 4 Varianten. Berücksichtigt man zudem, dass es für jede dieser Varianten der Richtung „A zu B“ jeweils wieder 4 Varianten der Richtung „B zu A“ gibt, lassen sich insgesamt 4x4=16 Klassen von Beziehungstypen unterscheiden. Diese sind in der Tabelle 2-1 zusammengestellt, wobei neben der graphischen Notation auch eine andere – häufig im Text benutzte – verbale Notation angegeben ist. Bei dieser wird die optionale Beziehungstyp-Richtung durch den Buchstaben „C“ (für „***c***onditional“) und die nicht-optionale Beziehungstyp-Richtung durch die Ziffer „1“ wiedergegeben. Die Kardinalität wird durch „1“ bzw. „N“ angegeben. Der Einfachheit halber wird „C1“ zu „C“ verkürzt, ebenso „11“ zu „1“ und „1N“ zu „N“. Liegt in beiden Beziehungstyp-Richtungen die Kardinalität N vor, wird auf der einen Seite der Buchstabe „M“ verwendet. Dadurch soll deutlich gemacht werden, dass die Kardinalitäten nicht in beiden Richtungen übereinstimmen müssen.

10 „symmetriefreie“ Klassen von Beziehungstypen

Die unterhalb der Diagonalen liegenden – grau unterlegten – Klassen stellen dabei keine interessanten Kombinationen dar, weil sie durch ein bloßes Vertauschen der Rollen der beiden beteiligten Objekttypen in ihre „Spiegelbilder“ oberhalb der Dia-

gonalen übergehen („C:1“ ⇔ „1:C“, „N:1“ ⇔ „1:N“ usw.). Sie werden deshalb im Weiteren nicht mehr betrachtet.

Bei einer p×p-Matrix mit p=4 bleiben dann aber noch

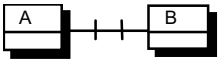
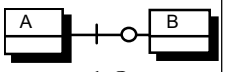
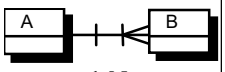
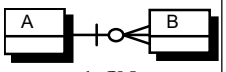
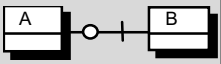
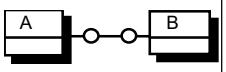
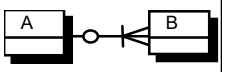
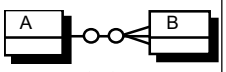
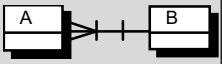
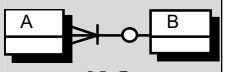
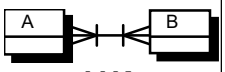
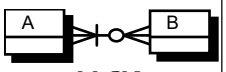
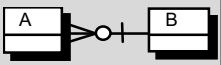
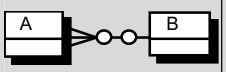
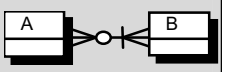
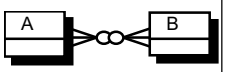
$$Z = \frac{p(p+1)}{2} = \frac{4 \cdot 5}{2} = 10$$

Klassen von Beziehungstypen[7].

Tab. 2-1: 16 Klassen von Beziehungstypen mit 10 „symmetriefreien“ Beziehungstypen

Spalte 1: Kardinalitäten 1 und N,

Spalte 2: nicht-optionale (n o) und optionale (o) Beziehungstyp-Richtungen

A→B		Kardinalität 1		Kardinalität N	
B↓A		nicht-optional	optional	nicht-optional	optional
1	n o	 1:1	 1:C	 1:N	 1:CN
	o	 C:1	 C:C	 C:N	 C:CN
N	n o	 N:1	 N:C	 M:N	 M:CN
	o	 CN:1	 CN:C	 CM:N	 CM:CN

7 Die Formel sieht man schnell ein, wenn man gedanklich die Diagonale doppelt, also zu jeder Zeile ein Element hinzufügt, und dann die entstehende Elemente-Anzahl p(p+1) halbiert.

Restriktionen gelten immer!

Die Angaben zur Optionalität und Kardinalität der beiden Beziehungstyp-Richtungen bringen wichtige Regeln des betrachteten Gegenstandsbereichs zum Ausdruck. Dabei ist zu beachten, dass diese Bedingungen *zu jedem Zeitpunkt* im Lebenszyklus der Datenbank erfüllt sein müssen.

Wird die *Optionalität* einer Beziehungstyp-Richtung „A zu B" ausgeschlossen, so muss zu jedem Zeitpunkt jedes Objekt vom Objekttyp A mit mindestens einem Objekt des Objekttyps B in Beziehung stehen. Das bedeutet insbesondere, dass ein neu zu speicherndes A-Objekt sofort mit einem B-Objekt in Beziehung gesetzt werden muss. In vielen praktischen Fällen ist das aber nicht möglich. Die Eigenschaftswerte einer neuen Schule sollen sicherlich schon gespeichert werden, wenn diese noch mit keinem Lehrer ein Beschäftigungsverhältnis abgeschlossen hat. Für diesen „Einschwingzustand" ist es häufig erforderlich, die Optionalität doch zuzulassen, weil ein neu „ins Leben tretendes" Objekt im allgemeinen mit seiner Umwelt noch nicht alle prinzipiell möglichen Beziehungen eingegangen ist.

Wird die *Kardinalität* einer Beziehungstyp-Richtung „A zu B" auf 1 festgelegt, so ist es zu keinem Zeitpunkt möglich, ein Objekt des Objekttyps A mit mehr als einem Objekt des Objekttyps B in Beziehung zu setzen. Auch im Fall der Kardinalität N bei zusätzlicher Einschränkung durch eine Minimal- und Maximalanzahl muss die Anzahl der Beziehungen, die ein A-Objekt jeweils mit B-Objekten eingeht, immer im geforderten [Min,Max]-Intervall liegen.

Zeitraum beachten!

Bei der Festlegung der Kardinalität ist außerdem zu beachten, über welchen Zeitraum hinweg die Angaben zu Beziehungen gespeichert werden sollen. Betrachtet man beispielsweise die Beziehungstyp-Richtung „Mitarbeiter *arbeitet in* Abteilung", so ist die Kardinalität auf 1 zu setzen, wenn man nur eine „Momentaufnahme" speichern möchte, weil zu jedem Zeitpunkt ein Mitarbeiter nur in einer Abteilung arbeitet. Will man aber die Zuordnungsverhältnisse über einen längeren Zeitpunkt festhalten, so ist die Kardinalität N festzulegen, weil es dann vorkommen kann, dass ein Mitarbeiter mit mehreren Abteilungen in Beziehung gebracht werden muss.

Beziehungstypen als „Brücken"

Die Modellierung der Beziehungstypen bringt eine neue Qualität in die Datenspeicherung. Ohne Beziehungen können jeweils nur die Eigenschaftswerte der einzelnen Objekte verarbeitet werden, weil es zwischen den Objekten keine Verbindungen gibt. Betrachtet man die Objekttypen als „Dateninseln", dann bilden

die Beziehungstypen die „Brücken“, die diese Inseln miteinander verbinden. Mit Hilfe der Beziehungen, die ja die konkreten Ausprägungen der Beziehungstypen darstellen, kann man nun eine „Brückenwanderung“ durchführen, indem man von den Eigenschaften eines Objekts zu den Eigenschaften des verknüpften Objekts und von diesen weiter zu den jeweils nächsten verbundenen Objekten gelangen kann. Die Beziehungen bieten also die Möglichkeit, Objekte dynamisch miteinander in Verbindung zu bringen und dadurch auch sogenannte „ad-hoc-Anfragen“ mit Hilfe der gespeicherten Daten zu beantworten, also einen Informationsbedarf zu befriedigen, der zum Zeitpunkt der Datenspeicherung noch nicht vorhersehbar war. Das ist in Zeiten flexibler Informationsanforderungen besonders wichtig, beispielsweise bei der Entscheidungsfindung im Management der Unternehmen.

Klassisches Pendant

Auch beim traditionellen Verfahren der Informationsspeicherung mit Hilfe von Karteikästen spielen die Beziehungen eine wichtige Rolle: Sie werden dort als Querverweise zwischen den Karteikästen realisiert, indem beispielsweise auf der Karteikarte einer Schule der Kennbegriff des Ortes vermerkt wird, in dem die Schule liegt.

Schulbeispiel

Die 10 interessierenden Klassen von Beziehungstypen werden im Abschnitt 4.3 im Zusammenhang mit ihrer Repräsentation im relationalen Datenbank-Modell der Reihe nach untersucht und durch Beispiele veranschaulicht. An dieser Stelle wenden wir uns wieder unserem Schulbeispiel zu und führen die Modellierung der sachlogischen Zusammenhänge zwischen den Objekttypen durch.

2. *Es gibt im Land größere Orte, die noch keine Schule besitzen. Neustadt besitzt dagegen 30 Schulen. Die Schiller-Schule von Neustadt ist gerade erst fertig gestellt: Sie hat weder Klassen noch hat sie mit Lehrern ein Beschäftigungsverhältnis abgeschlossen. Die Lehrer können mit mehreren Schulen des Landes ein Beschäftigungsverhältnis eingehen. Über arbeitslose Lehrer werden ebenfalls Informationen gespeichert. Jede Klasse einer Schule hat einen Lehrer als Klassenlehrer. Hat ein Lehrer seine Klasse zum Examen geführt, so ist er eine Weile kein Klassenlehrer. Fällt ein Klassenlehrer aus, übernimmt ein anderer zeitweilig seine Funktion. Ein Schüler gehört jeweils nur zu einer Klasse (kein Kurssystem). Eine Klasse besteht aus mindestens 15 und aus höchstens 30 Schülern.*

Ein speicherwürdiger Ort kann noch keine Schule besitzen. In der Fachsprache der Modellierung heißt das: Ein Objekt des Objekttyps „Ort“ *kann* (muss aber nicht!) mit einem Objekt des Objekttyps „Schule“ in Beziehung stehen - die Beziehungstyp-Richtung „Ein Ort *besitzt* eine Schule“ ist also optional. Ein Ort kann mehrere Schulen besitzen (beispielsweise der Ort „Neustadt“). Das heißt: Ein Objekt des Objekttyps „Ort“ kann *mit mehreren* Objekten des Objekttyps „Schule“ in Beziehung stehen – der Beziehungstyp-Richtung „Ein Ort *besitzt* eine Schule“ ist die Kardinalität N zugeordnet. Umgekehrt *muss* eine Schule in einem Ort liegen und kann nur in *einem* Ort liegen[8]. Der Beziehungstyp, der den Objekttyp „Ort“ mit dem Objekttyp „Schule“ verbindet, ist also ein 1:CN-Beziehungstyp.

Eine Schule kann (noch) mit *keinem* Lehrer ein Beschäftigungsverhältnis abgeschlossen haben (beispielsweise die gerade erst fertig gestellte Schiller-Schule von Neustadt), wird aber in der Regel mit *mehreren* Lehrern ein Beschäftigungsverhältnis abgeschlossen haben. Ein Lehrer kann mit *keiner* Schule (arbeitsloser Lehrer!), aber auch mit *mehreren* Schulen ein Beschäftigungsverhältnis eingehen. Es handelt sich in diesem Fall also um einen CM:CN-Beziehungstyp.

Eine Schule kann (noch) *keine* Klasse besitzen, besitzt aber in der Regel *mehrere* Klassen. Eine Klasse gehört stets zu *genau einer* Schule. Es liegt ein 1:CN-Beziehungstyp vor.

Bei der Wahl der Benennungen für die beiden Richtungen des Beziehungstyps zwischen Lehrer und Klasse ist zu beachten, dass der zu speichernde sachlogische Zusammenhang auch tatsächlich zum Ausdruck kommt. Aussagen wie „Ein Lehrer *hat* eine Klasse“ oder „Ein Lehrer *unterrichtet in* einer Klasse“ würden nicht den interessierenden Zusammenhang wiedergeben. Dieser wird durch die Formulierungen „Ein Lehrer *ist Klassenlehrer von* einer Klasse“ bzw. „Eine Klasse *hat als Klassenlehrer* einen Lehrer“ besser repräsentiert. Ein Lehrer kann *keine* Klasse, aber auch *mehrere* Klassen als Klassenlehrer haben. Dabei wird berücksichtigt, dass ein Lehrer, der seine Klasse zum Examen geführt hat, eine Weile pausieren darf, aber auch der Fall eingeschlossen, dass ein Lehrer beim Ausfall eines anderen Klassenlehrers eventuell zeitweilig mehr als eine Klasse betreuen muss. Eine

---

8 Wir setzen dabei voraus, dass eine Schule nicht auf der „grünen Wiese“ liegt und dass sie nicht über mehrere Orte verstreut ist.

Klasse hat zu jedem Zeitpunkt *genau einen* Lehrer als Klassenlehrer. Geht man davon aus, dass für eine Klasse immer nur der gerade aktuelle Klassenlehrer festgehalten wird, dann ist der Beziehungstyp zwischen den Objekttypen „Lehrer“ und „Klasse“ ein 1:CN-Beziehungstyp.

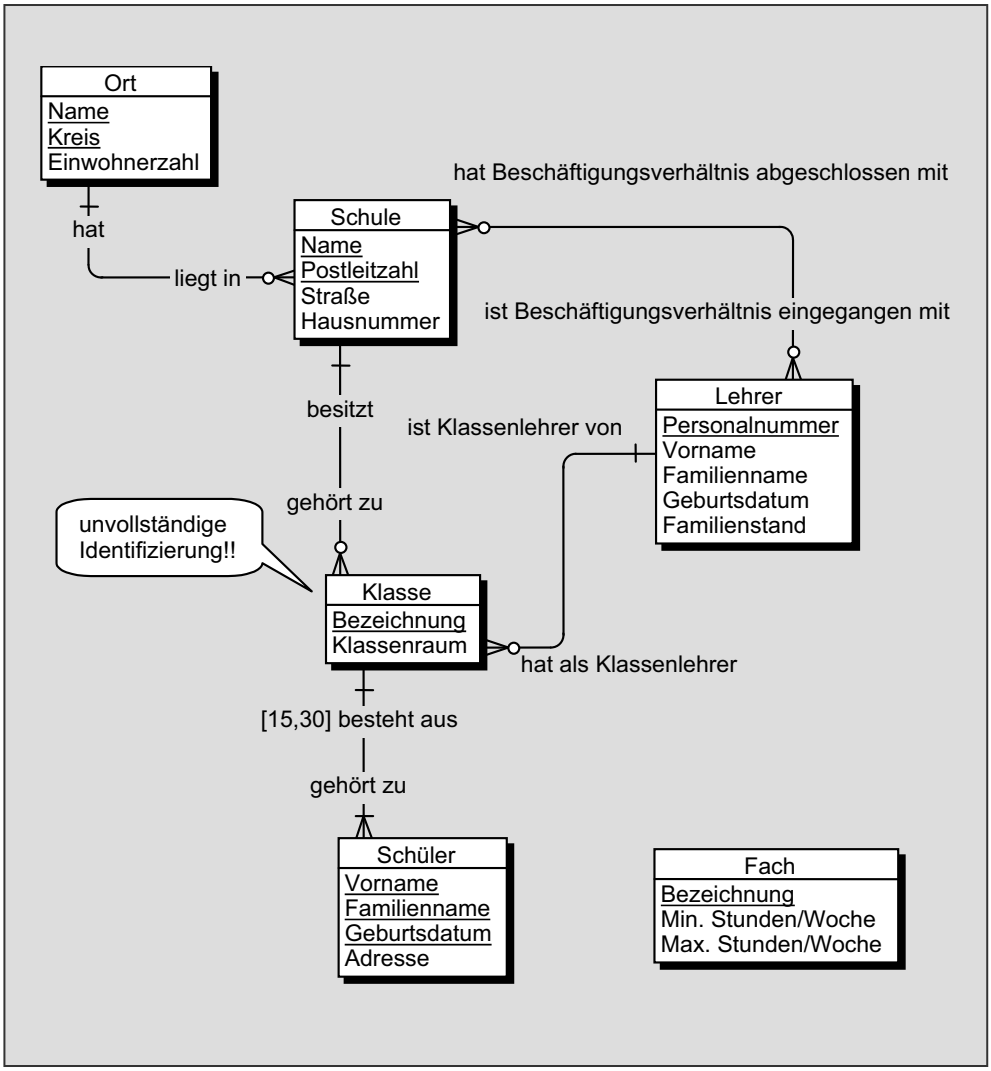


Abb. 2-7: Datenmodell für das Schulbeispiel mit Objekttypen, Eigenschaften, Identifizierungsangaben und Beziehungstypen

Jede einzelne Klasse *muss* aus Schülern bestehen (keine Optionalität!) und zwar aus *mehreren*, wobei deren Anzahl im Intervall [15,30] liegen muss. Ein Schüler gehört zu *genau einer* Klasse. Beim Beziehungstyp zwischen „Klasse“ und „Schüler“ handelt sich um einen 1:N-Beziehungstyp, der hinsichtlich der Kardinalität genauer spezifiziert wird.

Berücksichtigt man die diskutierten Beziehungstypen, so nimmt das Datenmodell den in Abbildung 2-7 dargestellten Zustand an.

Der Objekttyp „Fach“ steht mit keinem anderen Objekttyp in Beziehung. Das ist ungewöhnlich und deutet in der Praxis der Datenmodellierung meist darauf hin, dass wichtige sachlogische Zusammenhänge des Gegenstandsbereichs (noch) nicht berücksichtigt wurden. In unserem Schulbeispiel wird dieser Mangel im Abschnitt 2.5.1 behoben.

2.4.2 Redundante Beziehungstypen

Im vorigen Abschnitt wurden die Beziehungstypen mit Brücken verglichen, die eine Wanderung von einer Dateninsel A zu einer Dateninsel B ermöglichen. Nun können die Brücken aber so angelegt sein, dass es von A nach B zwei (oder mehr) verschiedene Wege gibt. Sind dann eine oder mehrere Brücken überflüssig? Bei der Datenmodellierung spricht man in solchen Fällen von *redundanten Beziehungstypen*. Das sind Beziehungstypen, die einen sachlogischen Zusammenhang zwischen zwei Datentypen beschreiben, der bereits durch die Kombination anderer Beziehungstypen in gleicher Weise beschrieben wird.

Der Begriff der *Redundanz* spielt bei der Informationsspeicherung eine große Rolle. Im praktischen Datenbankbetrieb wird Redundanz mitunter bewusst herbeigeführt, um die Suchprozesse in der Datenbank zu beschleunigen. In der Phase der Datenmodellierung sollte man Redundanz jedoch vermeiden, denn sie führt im allgemeinen zu schwerwiegenden Problemen:

Probleme bei redundanten Daten

- redundante Informationsstrukturen erfordern die *mehrfache Eingabe* derselben Daten,
- redundant gespeicherte Informationen belegen in der Datenbank *unnötig Speicherplatz*,
- bei einer Modifizierung der Informationen müssen garantiert alle Exemplare der redundant gespeicherten Informationen geändert werden, weil sonst in der Datenbank *widersprüchliche Aussagen* gespeichert sind.

Redundanz und zyklische Strukturen

Der Verdacht auf einen redundanten Beziehungstyp ergibt sich stets bei zyklischen Beziehungstyp-Strukturen. Lässt sich aber allein aus strukturellen Merkmalen des Datenmodells die Redundanz eines Beziehungstyps erkennen? Das wäre schön, denn dann könnte man redundante Beziehungstypen durch automatisierte Optimierungsprozesse eliminieren. Analysieren wir dazu das betriebswirtschaftliche Beispiel, das in Abbildung 2-8 dargestellt ist.

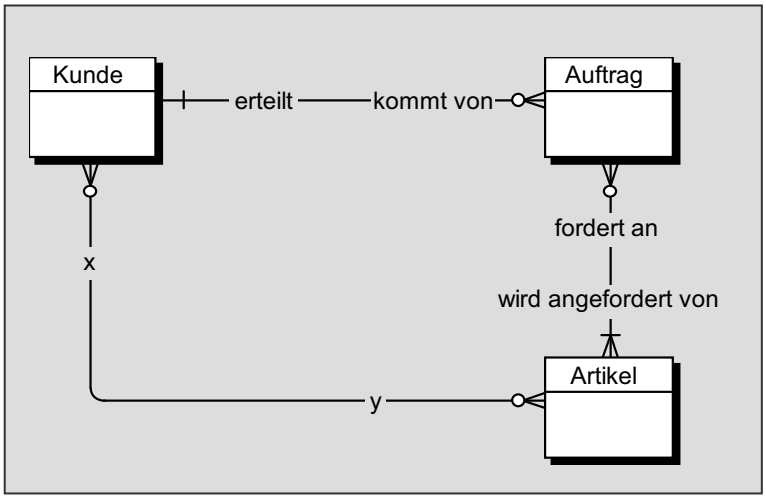


Abb. 2-8: Zyklische Struktur von Beziehungstypen

Ein Kunde erteilt unter Umständen *keinen* Auftrag (neuer Kunde, der nach der Kontaktaufnahme gerade erst gespeichert wurde), kann aber auch *mehrere* Aufträge erteilen. Ein Auftrag kommt stets von *genau einem* Kunden. Auf Auftrag fordert *einen oder mehrere* Artikel an. Ein Artikel kann von *keinem* Auftrag angefordert werden (neuer Artikel, der gerade erst in das Angebot aufgenommen wurde, oder ein „Ladenhüter", den keiner will), aber auch von *mehreren* Aufträgen.

Nun kommt der CM:CN-Beziehungstyp mit den beiden Beziehungstyp-Richtungen x und y hinzu, so dass auf Grund der entstehenden zyklischen Struktur zwei Wege vom Kunden zum Artikel führen. Ist dieser Beziehungstyp nun redundant? Betrachten wir zwei semantisch verschiedene Interpretationen dieses Beziehungstyps:

1. Der Beziehungstyp repräsentiert den sachlogischen Zusammenhang:

   x: „Ein Kunde *bestellt* einen Artikel“ und
   y: „Ein Artikel *wird bestellt von* einem Kunden”

   Dann ist der Beziehungstyp redundant, weil aus der Tatsache, dass ein Kunde einen Auftrag erteilt und der Auftrag einen Artikel anfordert, stets die Aussage folgt, dass der Kunde den Artikel bestellt.

2. Der Beziehungstyp modelliert den folgenden Sachverhalt:

   x: „Ein Kunde *reklamiert* einen Artikel” und
   y: „Ein Artikel *wird reklamiert von* einem Kunden”

   Der Beziehungstyp ist jetzt nicht redundant, weil aus der Tatsache, dass ein Kunde einen Auftrag erteilt und der Auftrag einen Artikel anfordert, nicht in jedem Falle folgt, dass der Kunde den Artikel reklamiert (wäre das der Fall, würde es das Unternehmen sicherlich nicht mehr geben).

Das Beispiel zeigt, dass sich die Frage, ob ein Beziehungstyp redundant ist, nicht auf Grund der Struktur des Datenmodells beantworten lässt, sondern dass sie nur durch eine inhaltliche Analyse der modellierten Zusammenhänge entschieden werden kann.

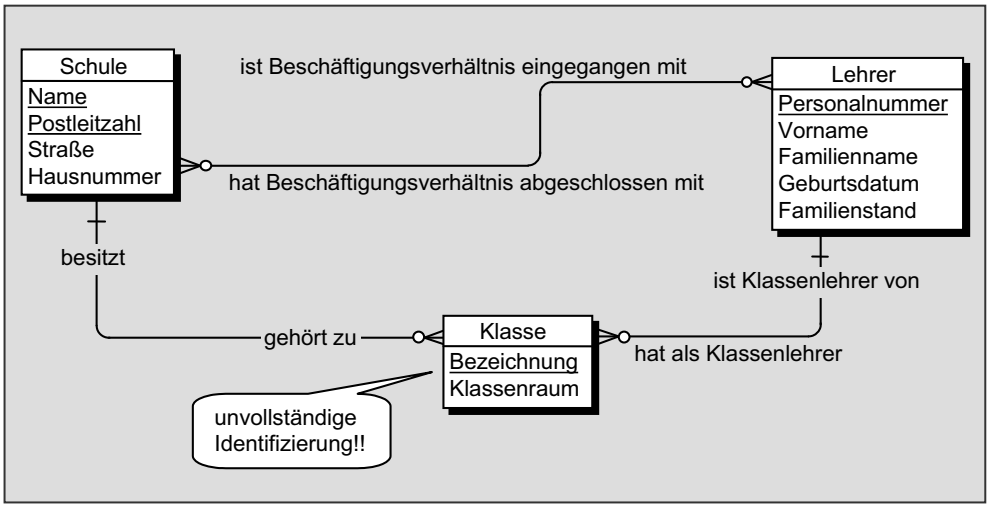


Abb. 2-9: Prüfung auf redundanten Beziehungstyp im Schulbeispiel

Schulbeispiel

In unserem Schulbeispiel muss die in Abbildung 2-9 dargestellte zyklische Struktur auf einen eventuellen redundanten Beziehungstyp hin überprüft werden.

Von „Klasse“ zu „Lehrer“ sind zwei Wege möglich. Aus der Tatsache, dass eine Klasse zu einer Schule gehört und diese mit einem Lehrer ein Beschäftigungsverhältnis abgeschlossen hat, folgt nicht zwingend, dass diese Klasse den Lehrer als Klassenlehrer hat. Es liegt also kein redundanter Beziehungstyp vor.[9]

2.4.3 Parallele Beziehungstypen

Häufig ist es bei der Speicherung von Informationen über einen Gegenstandsbereich erforderlich, *unterschiedliche* sachlogische Zusammenhänge zwischen zwei Objekttypen A und B festzuhalten, indem man mehrere Beziehungstypen zwischen A und B einführt. Diese werden dann als *parallele Beziehungstypen* bezeichnet. Sind nun Optionalität und Kardinalität der parallelen Beziehungstyp-Richtungen durch die Objekttypen vorgegeben?

Nehmen wir an, dass wir für eine bestimmte Personengruppe – beispielsweise die Mieter eines Hauses – und eine definierte Menge von Autos drei Beziehungstypen modellieren wollen, die in Abbildung 2-10 dargestellt sind.

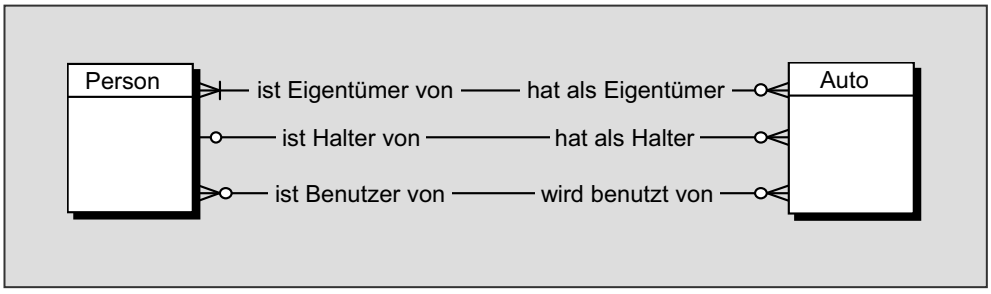


Abb. 2-10: Parallele Beziehungstypen

[9] Wenn eine Klasse einen Klassenlehrer hat, so folgt allerdings daraus, dass dieser Lehrer mit der Schule, zu der die Klasse gehört, auch ein Beschäftigungsverhältnis haben muss. Diese *partielle* Redundanz tritt jedoch erst dann ein, wenn ein bereits gespeicherter Lehrer zum Klassenlehrer ernannt wird. Dann könnte man seine – nun *redundante* – Beziehung zur Schule löschen – das ist aber kaum praktikabel.

Eine Person muss weder Eigentümer noch Halter noch Benutzer eines der betrachteten Autos sein, sie kann aber auch Eigentümer, Halter und Benutzer mehrerer Autos sein. Andererseits muss ein Auto mindestens einen, kann aber auch mehrere Eigentümer haben. Es kann keinen Halter haben, wenn es stillgelegt wurde, sonst aber höchstens einen. Es kann im betrachteten Zeitraum von keinem, aber auch von mehreren Personen benutzt werden. Man sieht, dass die Optionalität und Kardinalität nicht allein durch die beteiligten Objekttypen festgelegt sind, sondern dass sie durch die spezielle Semantik des jeweiligen sachlogischen Zusammenhangs bestimmt werden.

2.4.4 Die Beziehungstyp-Richtung als identifizierendes Element

Im Abschnitt 2.3 haben wir bei der Festlegung der Identifizierung für die Objekttypen unseres Schulbeispiels eine Situation vorgefunden, bei der die relevanten Eigenschaften eines Objekttyps nicht ausreichen, um eine eindeutige Identifizierung der Objekte dieses Objekttyps herbeizuführen. Wir wollen diese Situation an Hand des in Abbildung 2-11 dargestellten – etwas mythologisch gehaltenen – Beispiels verdeutlichen. Wir nehmen an, dass sich drei uns wohl bekannte Personen auf einer Festwiese befinden: Spartakus, Wilhelm Tell und Amor. Als natürliche Eigenschaften dieser Personen wollen wir aber lediglich zwei Eigenschaften speichern: die „Körpergröße“ mit den Werten „klein“, „mittel“ und „groß“ und die „Körperkraft“ mit den Werten „schwach“, „halbstark“ und „stark“.

Fragen wir nun „Wer ist Spartakus?“ und bekommen die Antwort „Der große Starke!“, so sind wir zu Recht unzufrieden, denn auf unserer Festwiese befinden sich zwei Personen, auf die diese Beschreibung zutrifft. Die natürlichen Eigenschaften, die wir für den Objekttyp „Person“ speichern wollen, reichen für eine Identifizierung der Objekte dieses Objekttyps nicht aus – wir haben es mit einem sogenannten *„schwachen Objekttyp“* zu tun.

Schwacher Objekttyp

Definition: Ein *schwacher Objekttyp* ist ein Objekttyp, dessen Eigenschaften es nicht erlauben, jedes Objekt dieses Objekttyps eindeutig zu identifizieren.

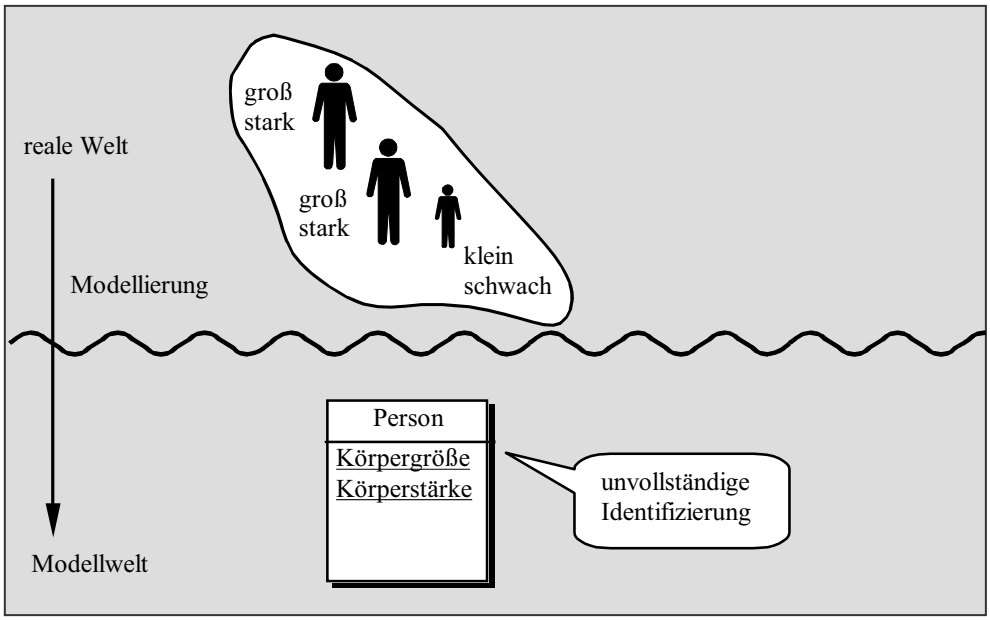


Abb. 2-11: Unvollständige Identifizierung durch Eigenschaften

Will man die Personen dennoch zweifelsfrei voneinander unterscheiden können, so muss man außer auf ihre Eigenschaften auch auf die Beziehungen zurückgreifen, die die Objekte mit anderen Objekten eingehen. In unserem Beispiel könnte man zusätzlich die charakteristischen Waffen betrachten, mit denen unsere Personen verbunden sind - Schwert, Armbrust, Pfeil & Bogen -, sowie andere Waffen. Auf die Frage „Wer ist Spartakus?“ kann man dann antworten: „Der große Starke mit dem Schwert!“. Wilhelm Tell wäre dann „Der große Starke mit der Armbrust“. Amor wäre „Der kleine Schwache mit Pfeil und Bogen“ und könnte damit von anderen kleinen und schwachen Personen unterschieden werden, die jeweils eine andere charakteristische Waffe tragen.

Die Beziehungstyp-Richtung, die den schwachen Objekttyp mit einem anderen Objekttyp verbindet, wird also zu einem identifizierenden Element für den schwachen Objekttyp.

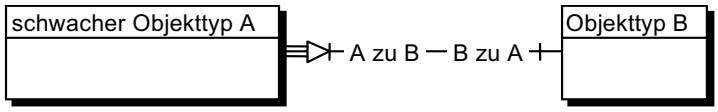
Die Beziehungstyp-Richtung „A zu B“ lässt sich allerdings nur dann als identifizierendes Element für den schwachen Objekttyp A verwenden, wenn es sich dabei um eine *nicht-optionale* Beziehungstyp-Richtung der *Kardinalität 1* handelt: Jedes konkrete

Objekt des schwachen Objekttyps A muss also stets diese Beziehung zu genau einem Objekt des Objekttyps B eingehen.

Syntaxregel für den schwachen Objekttyp

Syntaxregel:

Eine Beziehungstyp-Richtung „A zu B“, die als identifizierendes Element für einen schwachen Objekttyp A dient, wird durch ein Dreieck dargestellt, das vom schwachen Objekttyp A wegzeigt. Hat die Beziehungstyp-Richtung „B zu A“ die Kardinalität N, so wird dies nun statt durch den „Krähenfuß“ durch drei waagerechte Linien dargestellt:



Für den Fall unserer Festwiese wäre also die Modellierung in Abbildung 2-12 zu wählen.

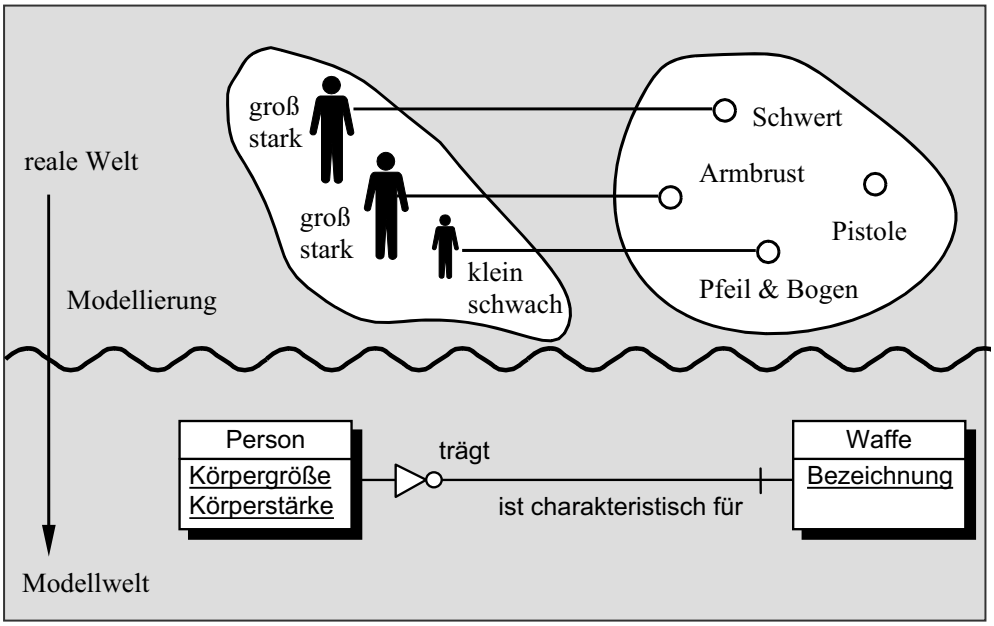


Abb. 2-12: Identifizierung mit Hilfe einer Beziehungstyp-Richtung

Schulbeispiel

In unserem Schulbeispiel gibt es einen schwachen Objekttyp: die „Klasse“. Die „Bezeichnung“ für sich allein ist – wie wir bereits argumentiert haben – für die Unterscheidung sämtlicher Klassen im Bundesland nicht ausreichend. Nimmt man jedoch die Beziehungstyp-Richtung, die die Klasse mit der Schule verbindet, als identifizierendes Element hinzu, so ist eine eindeutige Identifizierung möglich. Eine Klasse wird nun identifiziert durch zwei teilidentifizierende Elemente:

1. durch die Schule, zu der die Klasse gehört, und
2. durch die jeweilige Bezeichnung, die die Klasse innerhalb dieser Schule trägt.

Das führt zu der Form des Beziehungstyps zwischen Klasse und Schule, die Abbildung 2-13 zeigt.

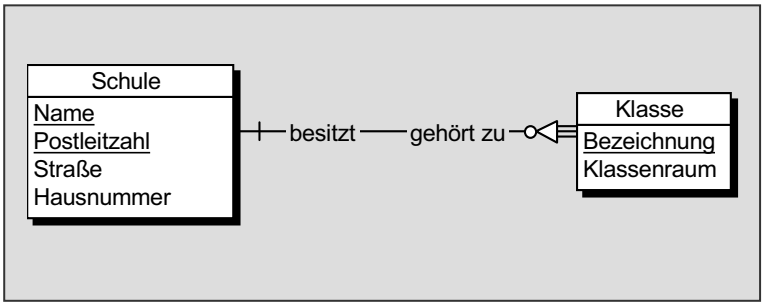


Abb. 2-13: Verwendung einer Beziehungstyp-Richtung als teilidentifizierendes Element im Schulbeispiel

Möglichkeiten zur Identifizierung

Damit lässt sich jetzt die Liste der Möglichkeiten zur Identifizierung, die im Abschnitt 2.3 nur auszugsweise behandelt wurde, komplettieren. Für die Identifizierung der Objekte eines Objekttyps gibt es die folgenden Möglichkeiten:

1. Verwendung der relevanten Eigenschaften
   1.1. Festlegung einer einzigen Eigenschaft
   1.2. Zusammenstellung einer Kombination von Eigenschaften
   1.3. Hinzufügen einer „organisatorischen Eigenschaft“
2. Verwendung von Beziehungstyp-Richtungen
   2.1. Wahl einer einzigen Beziehungstyp-Richtung
   2.2. Nutzung einer Kombination von Beziehungstyp-Richtungen
3. Verwendung einer Kombination von 1. und 2.

Wir wollen zum Abschluss dieses Abschnitts noch eine Terminologie einführen, die beim Datenbank-Design üblich ist und die uns in vielen Fällen eine kürzere Sprechweise ermöglicht:

Schlüssel

Definition: Die minimale Kombination von Eigenschaften und/oder Beziehungstyp-Richtungen, durch die die Objekte eines Objekttyps eindeutig identifiziert werden können, wird als *Schlüssel* des Objekttyps bezeichnet.

Zusammengesetzter Schlüssel

Definition: Ein Schlüssel, der sich aus mehreren teilidentifizierenden Elementen – und das können Eigenschaften und/oder Beziehungstyp-Richtungen sein – zusammensetzt, wird *zusammengesetzter Schlüssel* genannt.

Teilschlüssel

Definition: Ein *Teilschlüssel* ist eine echte Teilmenge der teilidentifizierenden Elemente, die einen zusammengesetzten Schlüssel bilden. Ein Teilschlüssel entsteht also dadurch, dass man aus einem zusammengesetzten Schlüssel wenigstens ein teilidentifizierendes Element entfernt.

Diese Terminologie werden wir insbesondere bei der Beschreibung der Normalisierung eines Datenmodells im Abschnitt 2.6 verwenden.

2.4.5 Rekursiv-Beziehungstypen

Im Abschnitt 2.4.1 haben wir sachlogische Zusammenhänge zwischen zwei Objekten modelliert, die verschiedenen Objekttypen angehören. Beispielsweise wurde der folgende Zusammenhang betrachtet: *„Die Goethe-Schule von Neustadt hat mit dem Lehrer Fritz Fröhlich ein Beschäftigungsverhältnis abgeschlossen"*. Dabei ist das erste Objekt „Goethe-Schule von Neustadt" im Objekttyp „Schule" und das zweite Objekt „Fritz Fröhlich" im Objekttyp „Lehrer" enthalten.

Häufig ist es aber wichtig, einen sachlogischen Zusammenhang zwischen zwei Objekten festzuhalten, die *demselben* Objekttyp angehören. So ist es für ein Unternehmen wichtig abzuspeichern, welcher Mitarbeiter durch welche Mitarbeiter - im Urlaubs- oder Krankheitsfall - vertreten werden kann:

Rekursiv-Beziehungstyp

Definition: Ein *Rekursiv-Beziehungstyp* (engl. recursive relationship type) beschreibt den sachlogischen Zusammenhang zwischen Objekten, die demselben Objekttyp angehören.

Für einen Rekursiv-Beziehungstyp sind dieselben Angaben erforderlich wie für einen dualen Beziehungstyp: Für jede der beiden Beziehungstyp-Richtungen werden festgelegt:

1. eine *Benennung*, die die spezielle Semantik des speicherwürdigen sachlogischen Zusammenhangs ausdrückt,
2. eine Angabe zur *Optionalität* und
3. eine Angabe zur *Kardinalität*.

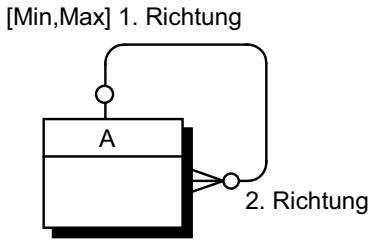
Klassisches Pendant

Bei der traditionellen Informationsspeicherung mit Hilfe von Karteikästen wird der Rekursiv-Beziehungstyp durch Querverweise innerhalb eines Karteikastens realisiert. Im betrachteten Vertretungsbeispiel würden auf der Karteikarte eines Mitarbeiters die Personalnummern seiner möglichen Vertreter notiert werden.

Syntaxregeln für den Rekursiv-Beziehungstyp

Syntaxregeln:

1. Ein Rekursiv-Beziehungstyp zur Kennzeichnung eines sachlogischen Zusammenhangs zwischen den Objekten ein- und desselben Objekttyps A wird als eine Verbindungslinie dargestellt, die den Objekttyp A mit sich selber verbindet[10].
2. Die Benennung der jeweiligen Beziehungstyp-Richtung steht in der Nähe des Austrittspunkts aus dem Objekttyp.
3. Die Optionalität und Kardinalität der jeweiligen Beziehungstyp-Richtung wird in gleicher Weise angegeben wie beim dualen Beziehungstyp:



Setzt man bei unserem Vertretungsverhältnis der Mitarbeiter voraus, dass

- ein Mitarbeiter eventuell *keine* Vertretung übernehmen muss, er aber *höchstens zwei* Mitarbeiter vertreten kann und
- ein Mitarbeiter durch *mindestens einen*, aber auch durch *mehrere* Mitarbeiter vertreten werden kann,

dann ergibt sich das Datenmodell der Abbildung 2-14. Dabei ist zu beachten, dass die [Min,Max]-Angabe lediglich die Kardinalität der Beziehungstyp-Richtung präzisiert und keine Aussage über deren Optionalität trifft.

[10] Diese Form der Verbindungslinie, die aus A austritt und wieder nach A *zurückführt*, ist der Anlass für die Bezeichnung „rekursiv“ (lat. „zurückführend“).

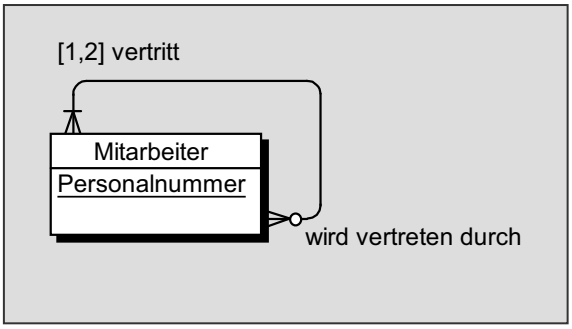


Abb. 2-14: Rekursiv-Beziehungstyp

Im Abschnitt 2.4.1 haben wir die dualen Beziehungstypen in 16 Klassen eingeteilt. Jede Klasse repräsentiert eine spezielle Kombination von Optionalität und Kardinalität in beiden Beziehungstyp-Richtungen. Auf Grund von Symmetrien - beispielsweise ist der 1:C-Beziehungstyp symmetrisch zum C:1-Beziehungstyp – blieben nur 10 interessierende Klassen übrig. Es stellt sich nun die Frage, ob alle diese 10 Klassen auch bei Rekursiv-Beziehungstypen wiederzufinden sind.

Die Besonderheit eines Rekursiv-Beziehungstyps gegenüber einem dualen Beziehungstyp, der die Objekttypen A und B miteinander verknüpft, besteht ja darin, dass die Objekttypen A und B identisch sind. Somit sind Rekursiv-Beziehungstypen nur dann möglich, wenn die Objekttypen A und B, die ja Mengen von Objekten repräsentieren, dieselbe Mächtigkeit[11] besitzen können.

Beziehungstypen und Mächtigkeit der Objekttypen

Untersuchen wir deshalb die 10 interessierenden Klassen von Beziehungstypen hinsichtlich der Forderung, die an die Mächtigkeit der beteiligten Objekttypen gestellt wird. Damit die Verhältnisse besser überschaubar sind, wollen wir uns die folgende Situation vor Augen halten: Die Schüler einer Schulklasse in Aachen (A-Schüler) schreiben Briefe an die Schüler einer Schulklasse in Bern (B-Schüler). Welche Restriktionen ergeben sich für die Klassengrößen im Falle der 10 interessierenden Beziehungstypen?

[11] Unter der Mächtigkeit einer Menge M (dargestellt durch $\overline{\overline{M}}$) versteht man die Zahl ihrer Elemente. Die Mächtigkeit eines Objekttyps ist also die Anzahl der in ihm zusammengefassten Objekte.

1:1-Beziehungstyp: Jeder A-Schüler schreibt genau einen Brief, jeder B-Schüler erhält genau einen Brief. Die Klassen müssen dieselbe Schülerzahl haben.

1:C-Beziehungstyp: Einige A-Schüler schreiben keinen Brief, aber jeder B-Schüler erhält genau einen Brief. Es muss mehr A-Schüler als B-Schüler geben.

C:C-Beziehungstyp: Einige A-Schüler schreiben keinen Brief, einige B-Schüler erhalten keinen Brief. Über das Größenverhältnis der Klassen lässt sich nichts aussagen. Sie können gleich groß sein.

1:N-Beziehungstyp: Jeder A-Schüler schreibt mindestens einen Brief, wobei wenigstens ein A-Schüler mehrere Briefe schreibt (sonst wäre es ein 1:1-Beziehungstyp). Jeder B-Schüler erhält genau einen Brief. Es muss also mehr B-Schüler als A-Schüler geben.

C:N-Beziehungstyp: Wie beim 1:N-Beziehungstyp, nur dass es mindestens einen B-Schüler gibt, der keinen Brief erhält. Es muss also erst recht mehr B-Schüler als A-Schüler geben.

1:CN-Beziehungstyp: Ein A-Schüler schreibt keinen, einen oder mehrere Briefe. Jeder B-Schüler erhält genau einen Brief. Über das Größenverhältnis der Klassen lässt sich nichts aussagen.

C:CN-Beziehungstyp: Wie beim 1:CN-Beziehungstyp, nur dass es mindestens einen B-Schüler gibt, der keinen Brief erhält. Über das Größenverhältnis der Klassen lässt sich nichts aussagen.

M:N-Beziehungstyp: Jeder A-Schüler schreibt mindestens einen Brief, wobei wenigstens ein A-Schüler mehrere Briefe schreibt (sonst wäre es ein N:1-Beziehungstyp). Analog dazu erhält jeder B-Schüler

wenigstens einen Brief, mindestens einer erhält zwei Briefe. Über das Größenverhältnis der Klassen lässt sich nichts aussagen.

M:CN-Beziehungstyp: Wie beim M:N-Beziehungstyp, nur dass es mindestens einen A-Schüler gibt, der keinen Brief schreibt. Über das Größenverhältnis der Klassen lässt sich nichts aussagen.

CM:CN-Beziehungstyp: Wie beim M:CN-Beziehungstyp, nur dass es mindestens einen A-Schüler gibt, der keinen Brief schreibt. Über das Größenverhältnis der Klassen lässt sich nichts aussagen.

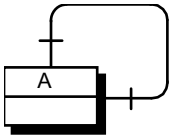
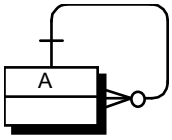
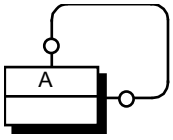
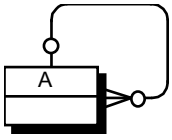
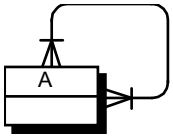
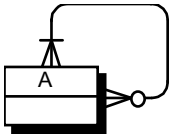
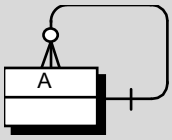
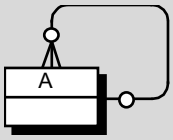
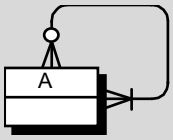
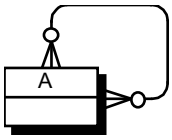
In der Tabelle 2-2 werden die Ergebnisse zusammengefasst und die Implikationen für die Rekursiv-Beziehungstypen angegeben:

Tab. 2-2: Mögliche Klassen von Rekursiv-Beziehungstypen

Beziehungstyp-Klasse	**Bedingung für das Größenverhältnis**	**Rekursiv-Beziehungstyp**
1:1	$\overline{\overline{A}} = \overline{\overline{B}}$	möglich
1:C	$\overline{\overline{A}} > \overline{\overline{B}}$	nicht möglich
C:C	keine	möglich
1:N	$\overline{\overline{A}} < \overline{\overline{B}}$	nicht möglich
C:N	$\overline{\overline{A}} < \overline{\overline{B}}$	nicht möglich
1:CN	keine	möglich
C:CN	keine	möglich
M:N	keine	möglich
M:CN	keine	möglich
CM:CN	keine	möglich

Die sieben möglichen Klassen der Rekursiv-Beziehungstypen zeigt die Tabelle 2-3 in Form einer Matrix, in der die „spiegelbildlichen“ Klassen wiederum grau unterlegt und die nicht realisierbaren Klassen dunkel dargestellt sind.

Tab. 2-3: Sieben interessierende Klassen von Rekursiv-Beziehungstypen

1. Beziehungstyp-Richtung → / 2. Beziehungstyp-Richtung ↓		Kardinalität 1		Kardinalität N	
		nicht-optional	optional	nicht-optional	optional
Kardinalität 1	nicht-optional	 1:1			 1:CN
	optional		 C:C		 C:CN
Kardinalität N	nicht-optional			 M:N	 M:CN
	optional	 CN:1	 CN:C	 CM:N	 CM:CN

Die 7 interessierenden Klassen von Rekursiv-Beziehungstypen werden im Abschnitt 4.4 im Zusammenhang mit ihrer Repräsentation im relationalen Datenbank-Modell der Reihe nach untersucht und durch Beispiele veranschaulicht.

Schulbeispiel

An dieser Stelle wenden wir uns wieder unserem Schulbeispiel zu. Dort müssen ebenfalls sachlogische Zusammenhänge zwischen Objekten desselben Objekttyps dargestellt werden:

3. *In jeder Schule ist einer der Lehrer Direktor und leitet die anderen Lehrer an. Über ihn werden aber dieselben Informationen gesammelt wie über die anderen Lehrer. Zwischen den Schulen können Patenschaften bestehen. Eine Schule kann zwar mehrere Paten haben, soll aber selbst nur für maximal eine andere Schule Pate sein.*

Da laut der Aufgabenbeschreibung für einen Direktor dieselben Eigenschaften gespeichert werden wie für jeden anderen Lehrer (der Direktor ist auch nur ein Lehrer, wenn auch mit besonderen Befugnissen), ist es nicht erforderlich, für ihn einen eigenen Objekttyp zu gründen. Statt dessen wird das Anleitungsverhältnis zwischen den Lehrern als Rekursiv-Beziehungstyp dargestellt. Dabei gibt es Lehrer, die keinen Lehrer anleiten, und eben die Direktor-Lehrer, die mehrere Lehrer anleiten. Ein Lehrer wird eventuell von keinem Lehrer angeleitet (nämlich dann, wenn er Direktor ist) oder auch von mehreren Lehrern angeleitet (weil er eventuell in mehreren Schulen beschäftigt ist).

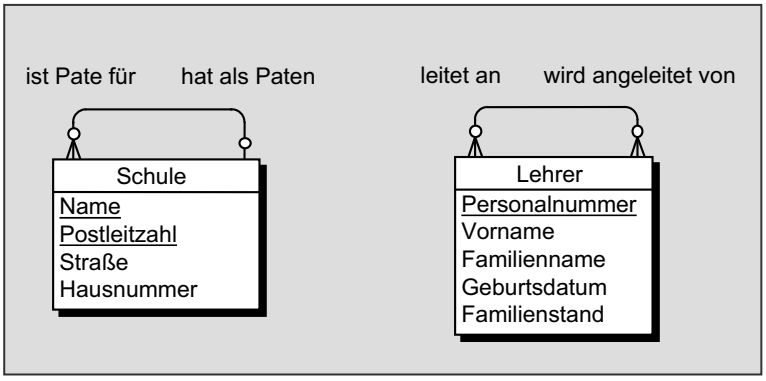


Abb. 2-15: Rekursiv-Beziehungstypen im Schulbeispiel

Das Patenschaftsverhältnis zwischen den Schulen stellt ebenfalls einen sachlogischen Zusammenhang zwischen Objekten desselben Objekttyps dar, muss also auch als Rekursiv-Beziehungstyp modelliert werden. Eine Schule braucht keine Patenschaft zu übernehmen, kann aber höchstens für eine Schule Pate sein. Andererseits muss eine Schule keinen Paten haben, kann aber auch

mehrere Schulen als Paten haben. Abbildung 2-15 zeigt die entsprechenden Ausschnitte des Datenmodells.

Hierarchische Zusammenhänge

Rekursiv-Beziehungstypen eignen sich gut zur Modellierung von *hierarchischen* sachlogischer Zusammenhänge zwischen den Objekten eines Objekttyps. Das bietet gegenüber der Verwendung eigener Objekttypen für die einzelnen Hierarchie-Ebenen den Vorteil, dass man hinsichtlich der Anzahl der Hierarchieebenen nicht festgelegt ist und ohne Änderung des Datenmodells neue Hierarchieebenen einführen kann. Bei hierarchischen Beziehungen ist die Optionalität beider Beziehungstyp-Richtungen erforderlich. Dadurch wird in beiden Richtungen der Hierarchie der Abbruch beschrieben. Im graphischen Bild einer Baumstruktur – wie beispielsweise in einem Organigramm – entspricht das der Tatsache, dass die Hierarchie in der einen Richtung in der Wurzel und in der anderen Richtung in den Blättern endet.

Beispiel: Stückliste

Ein klassisches Beispiel für eine solche hierarchische Beziehung zwischen Objekten ist die Stückliste. Jedes industrielle Produkt von hinreichender Komplexität wird aus Teilen zusammengefügt, die wiederum aus noch kleineren Teilen bestehen können. Ein Fahrrad besteht beispielsweise aus dem Rahmen, dem Lenker, dem Sattel, den Rädern usw. Ein Rad wiederum setzt sich zusammen aus Felge, Speichen, Reifen usw. Der Reifen wiederum besteht aus Schlauch und Mantel. Werden nun für jedes dieser Bauteile dieselben Eigenschaften gespeichert, so ist es nicht sinnvoll, für sie jeweils eigene Objekttypen zu bilden. Vielmehr kann man sie alle dem Objekttyp „Bauteil" zuordnen und die Teil-Ganzes-Beziehung durch einen Rekursiv-Beziehungstyp modellieren. Dabei besteht ein Bauteil eventuell nicht aus kleineren Bauteilen, nämlich dann, wenn es bereits ein elementares Bauteil – beispielsweise eine Schraube – ist. Wenn es aber aus kleineren Bauteilen besteht, dann aus mindestens 2 und höchstens aus einer nicht angebbaren Maximalzahl (N) von Bauteilen. Andererseits kann ein Bauteil Bestandteil keines größeren Bauteils sein (wenn es sich um das Gesamtprodukt handelt), aber auch Bestandteil mehrerer größerer Bauteile. Die Repräsentation dieser Verhältnisse im Datenmodell zeigt die Abbildung 2-16.

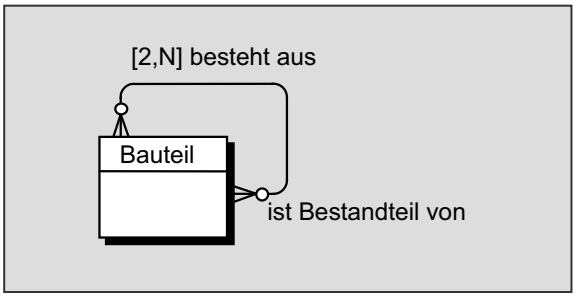


Abb. 2-16: Stückliste als Rekursiv-Beziehungstyp

2.5 Modellierung in Grenzfällen des Entity-Relationship-Modells

Die graphische Sprache des Entity-Relationship-Modells ist ganz bewusst einfach gehalten. Sie stellt – soweit hier beschrieben - nur eine sehr geringe Anzahl von Gestaltungselementen bereit, nämlich:

- den *Objekttyp* als „Container“ für jene speicherwürdigen Objekte, über die dieselben Informationen gesammelt werden und die in prinzipiell gleicher Weise verarbeitet werden,
- die *identifizierende* (bzw. *teilidentifizierende*) *Eigenschaft* zur Aufnahme eines Wertes, der ein Objekt innerhalb eines Objekttyps identifiziert (bzw. zu dessen Identifizierung beiträgt),
- die *beschreibende Eigenschaft* zur Aufnahme eines Wertes, der nicht für die Identifizierung des Objekts benötigt wird,
- den *dualen Beziehungstyp* als Menge aller sachlogischen Beziehungen gleicher Semantik zwischen zwei Objekten, die zwei verschiedenen Objekttypen angehören,
- die *Optionalitätsangabe* zur Unterscheidung von optionalen und obligatorischen Beziehungstyp-Richtungen,
- die *Kardinalitätsangabe* zur Unterscheidung der Kardinalitäten 1 bzw. N einer Beziehungstyp-Richtung,
- die *Intervallangabe für die Kardinalität* einer Beziehungstyp-Richtung zur Festschreibung von Unter- und Oberschranken der Kardinalität,

- die Festlegung einer *Beziehungstyp-Richtung als identifizierendes* (bzw. *teilidentifizierendes*) *Element*,
- den *Rekursiv-Beziehungstyp* als Menge aller sachlogischen Beziehungen übereinstimmender Semantik zwischen zwei Objekten, die zum selben Objekttyp gehören.

Problemfälle

Auf Grund der geringen Ausdrucksstärke dieser Sprache ist es nicht verwunderlich, dass bei der praktischen Modellierung eines Gegenstandsbereichs häufig Situationen auftreten, die sich mit den angegebenen graphischen Elementen nicht ohne weiteres beschreiben lassen. Für diese Fälle, in denen man an die Grenzen des Entity-Relationship-Modells stößt, müssen Verhaltensregeln entwickelt werden, die es gestatten, diese Grenzfälle dennoch mit den Sprachelementen des Entity-Relationship-Modells auszudrücken.

In den folgenden Abschnitten sollen drei der wichtigsten Grenzfälle näher besprochen werden.

2.5.1 Sachlogische Zusammenhänge zwischen mehr als 2 Objekttypen

Häufig ist es erforderlich, Sachverhalte abzuspeichern, an denen Objekte aus mehr als 2 Objekttypen beteiligt sind. Beispielsweise muss eine Fluggesellschaft Buch darüber führen, welcher Pilot (1. Objekttyp) mit welchem Flugzeug (2. Objekttyp) auf welcher Fluglinie (3. Objekttyp) geflogen ist. Mitunter müssen zu diesem Sachverhalt über die beteiligten Objekte hinaus auch noch weitere Angaben festgehalten werden, in unserem Beispiel etwa der Flugtag, die Abflugzeit und die Ankunftszeit.

Solche Konstruktionen sind aber mit den graphischen Elementen des Entity-Relationship-Modells nicht unmittelbar darstellbar. Diese ermöglichen an sich nur die Darstellung von Sachverhalten, an denen Objekte eines Objekttyps (durch den Rekursiv-Beziehungstyp) oder aus zwei Objekttypen (durch den dualen Beziehungstyp) beteiligt sind.

Im gegebenen Fall müssen aber Objekte aus *drei* Objekttypen zu einer gemeinsamen Handlung zusammengeführt werden, wie das aus Abbildung 2-17 ersichtlich ist.

Begriffserweiterung des Objekttyps

Für solche sachlogischen Zusammenhänge gibt es aber im Entity-Relationship-Modell – zumindest in der hier verwendeten Notation - keine syntaktische Konstruktion. Um derartige Situationen dennoch modellieren zu können, ist eine Erweiterung des Begriffs des Objekttyps erforderlich.

Koppel-Objekttyp

Bisher wurde der Objekttyp nur als „Container“ für elementare Objekte betrachtet, die entweder real (z.B. Lehrer, Ort) oder immateriell (z. B. Schulfach) im Gegenstandsbereich vorhanden sind. Jetzt werden als Objekte, die in einem Objekttyp zusammengefasst werden können, auch Sachverhalte betrachtet, die das *Zusammenspiel von elementaren Objekten* zum Inhalt haben. Auf Grund ihrer Funktion, die elementaren Objekte aus mehreren Objekttypen zu einer gemeinsamen Handlung „zusammenzukoppeln“, werden sie als *„Koppel-Objekttypen“* bezeichnet.

Ebenen von Objekttypen

Während die Objekttypen, in denen elementare Objekte zusammengefasst werden, die *Grundebene* der Datenspeicherung darstellen, bilden die Objekttypen, die das Zusammenspiel von elementaren Objekten beinhalten, die *erste Metaebene* der Datenspeicherung. Bei komplexeren Gegenstandsbereichen kann diese Ebenenhierarchie noch weiter nach oben ergänzt werden, indem Objekttypen der *zweiten Metaebene* eingeführt werden, die das Wechselverhältnis des Zusammenspiels von elementaren Objekten beschreiben. Prinzipiell gibt es für diese Ebenenbildung keine obere Grenze, allerdings werden dann die gedanklichen Gebilde immer komplizierter.

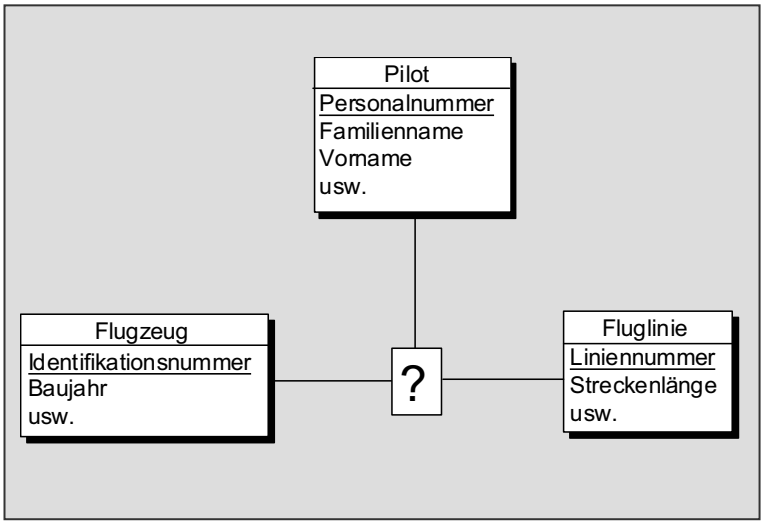


Abb. 2-17: Sachlogischer Zusammenhang mit mehr als 2 Objekttypen

Objekttyp im erweiterten Sinne

Definition: Ein *Objekttyp* (engl. entity type) – im erweiterten Sinne - ist eine durch einen Objekttyp-Namen eindeutig benannte Menge von elementaren Objekten oder von sachlogischen Verbindungen von Objekten (auf der 1., 2. usw. Metaebene), über die dieselben Informationen gespeichert werden und die in prinzipiell gleicher Weise verarbeitet werden.

Das Zusammenwirken der drei Objekttypen A, B und C mit dem Koppel-Objekttyp K wird durch Beziehungstypen beschrieben, die K jeweils mit den Objekttypen A, B und C verbinden. Wie für jeden Objekttyp muss natürlich auch für einen Koppel-Objekttyp eine Entscheidung über die Identifizierung der in ihm enthaltenen Objekte getroffen werden. Häufig bieten sich – zumindest als teilidentifizierende Elemente – die Beziehungstyp-Richtungen „K zu A“, „K zu B“ bzw. „K zu C“ an.

Im betrachteten Flugbeispiel lässt sich das Zusammenspiel eines Piloten, eines Flugzeugs und einer Fluglinie durch den Koppel-Objekttyp „Flug“ modellieren. Das Ergebnis ist in der die Abbildung 2-18 dargestellt.

Im Modell wird unterstellt, dass Piloten, Flugzeuge und Fluglinien schon dann gespeichert werden sollen, wenn sie noch an keinem Flug beteiligt waren. Natürlich können die Objekte dieser Objekttypen an mehreren Flügen beteiligt sein. Ein Flug wird identifiziert durch die Fluglinie, auf der er erfolgt, und durch den Flugtag, wobei angenommen wird, dass auf einer Fluglinie pro Tag höchstens ein Flug durchgeführt wird. Die Identifizierung des Koppel-Objekttyps „Flug“ erfolgt also durch eine Kombination aus der teilidentifizierenden Eigenschaft „Flugtag“ und der Beziehungstyp-Richtung „Flug *erfolgt auf* Fluglinie“.

Schulbeispiel

Auch in unserem Schulbeispiel soll das Zusammenspiel von Objekten festgehalten werden, die aus mehr als 2 Objekttypen stammen:

4. *Es sollen Informationen darüber gespeichert werden, welcher Lehrer welches Fach in welcher Klasse unterrichtet. Das Fach Hauswirtschaft wird im Land noch gar nicht unterrichtet. Frau Müller unterrichtet wegen Krankheit in diesem Schuljahr nicht. Herr Meier unterrichtet die 11b der Goethe-Schule im Unterrichtsraum 205 im Fach Englisch. (...)*

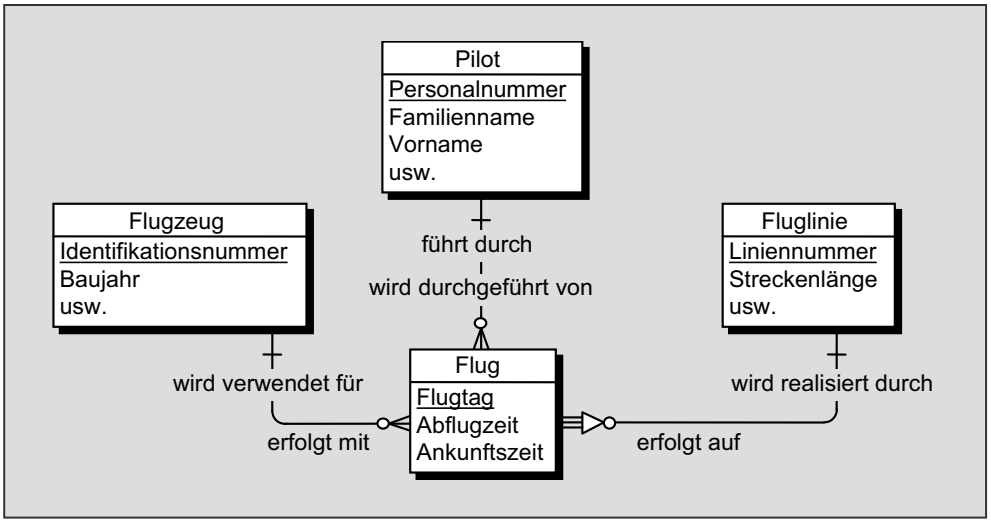


Abb. 2-18: Beispiel für einen Koppel-Objekttyp

Es soll also das spezielle Zusammenwirken von Objekten aus den Objekttypen „Lehrer“, „Fach“ und „Klasse“ modelliert werden. Wie wir wissen, ist dazu ein Koppel-Objekttyp erforderlich, dessen Objekte jeweils festhalten, dass ein Lehrer die Verpflichtung übernommen hat, im Laufe des jeweiligen Schuljahres ein Fach in einer Klasse zu unterrichten. Als Objekttyp-Name bietet sich demnach „Unterrichtsverpflichtung“ an. Den Ausschnitt aus dem Datenmodell, der das Zusammenspiel dieses Koppel-Objekttyps mit den drei Objekttypen „Lehrer“, „Fach“ und „Klasse“ beschreibt, zeigt die Abbildung 2-19.

Im Modell wird berücksichtigt, dass es Lehrer und Fächer geben kann, für die keine Unterrichtsverpflichtung besteht (langzeitkranker Lehrer, neu vorgesehenes Fach), dass für eine Klasse aber stets Unterrichtsverpflichtungen vorliegen müssen. Eine konkrete Unterrichtsverpflichtung ist gekennzeichnet durch

- den Lehrer, von dem sie übernommen wird (Beziehungstyp-Richtung „Unterrichtsverpflichtung *wird übernommen von* Lehrer“),
- das Fach, für das sie besteht (Beziehungstyp-Richtung „Unterrichtsverpflichtung *besteht für* Fach“), und
- die Klasse, für die sie besteht (Beziehungstyp-Richtung „Unterrichtsverpflichtung *besteht für* Klasse“).

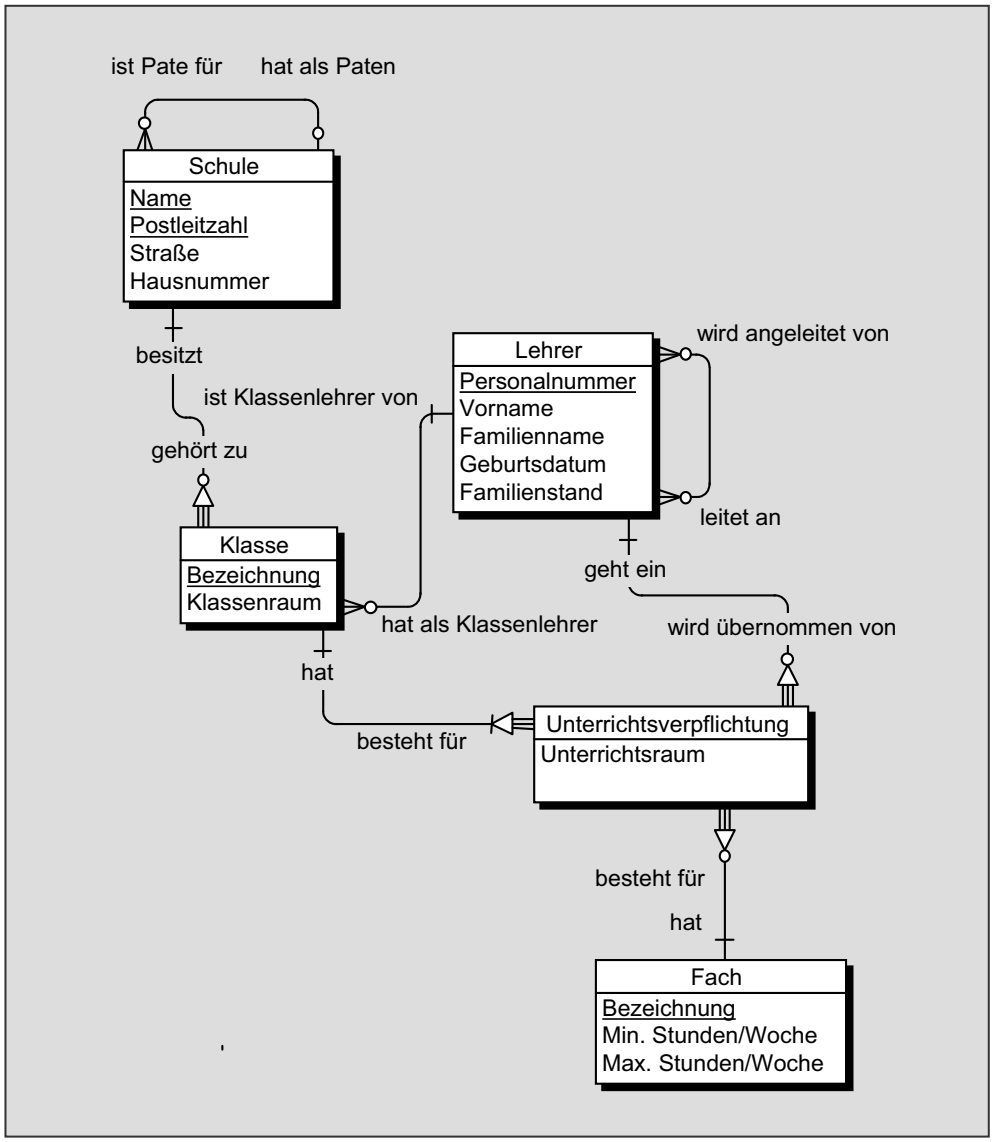


Abb. 2-19: Koppel-Objekttyp „Unterrichtsverpflichtung“ im Schulbeispiel

Die Identifizierung erfolgt also durch eine Kombination von drei Beziehungstyp-Richtungen. Die einzige beschreibende Eigenschaft des Koppel-Objekttyps „Unterrichtsverpflichtung“ ist der Unterrichtsraum.

2.5.2 Eigenschaften von Beziehungstypen

Häufig besteht die Notwendigkeit, die konkrete Beziehung, die zwei Objekte des betrachteten Gegenstandsbereichs eingehen, genauer zu spezifizieren. Betrachten wir dazu den folgenden Fall: Im Standesamt werden Informationen darüber gespeichert, welcher Mann mit welcher Frau verheiratet ist bzw. verheiratet war. Wir nehmen vereinfachend an, dass ein Mann und eine Frau jeweils durch die Kombination der Eigenschaften „Familienname+Vorname+Geburtsdatum“ identifiziert werden können. Nun soll die konkrete Verbindung zwischen einem Mann und einer Frau durch eine zusätzliche Angabe präzisiert werden – in unserem Beispiel durch das Hochzeitsdatum. Die Eigenschaft „Hochzeitsdatum“ kann aber weder dem Objekttyp „Mann“ noch dem Objekttyp „Frau“ zugeordnet werden. Sie ist eigentlich eine „Eigenschaft des Beziehungstyps“, der zwischen Mann und Frau besteht. In Abbildung 2-20 ist diese Situation dargestellt.

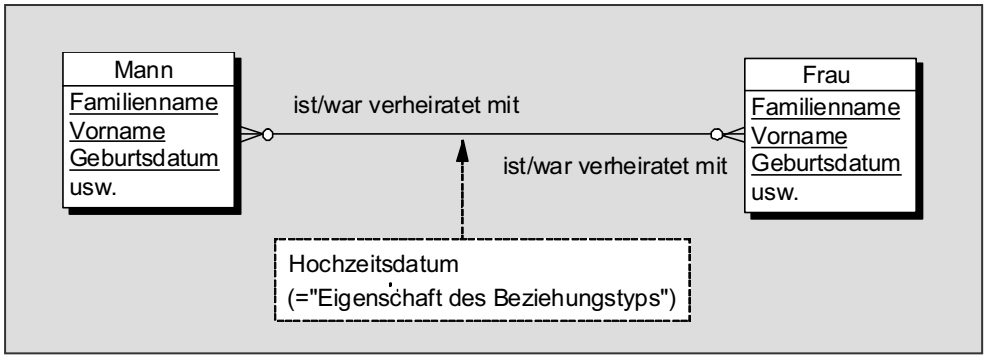


Abb. 2-20: Eigenschaft für einen Beziehungstyp

Einführung eines Koppel-Objekttyps

Diese Situation ist im Entity-Relationship-Modell – zumindest in der hier verwendeten Notation - nicht direkt darstellbar. Das liegt daran, dass das Entity-Relationship-Modell in seinen beiden Hauptkomponenten – den Objekttypen und den Beziehungstypen – unsymmetrisch ist: Ein Objekttyp kann Eigenschaften haben, ein Beziehungstyp dagegen nicht. Immer dann, wenn man über die Beziehung zwischen zwei Objekten a und b mehr aussagen möchte als den puren Sachverhalt, dass diese Beziehung existiert, wenn man also den entsprechenden Beziehungstyp „attributieren“ möchte, stößt man auf die genannte Grenze,

die uns die Sprache des Entity-Relationship-Modells auferlegt. Um diese wichtigen Praxisfälle dennoch modellieren zu können, muss man – analog zum Koppel-Objekttyp - das Zusammenspiel von a und b als Objekt eines neuen Objekttyps darstellen, dem man die zusätzliche Aussage als Eigenschaft zuordnen kann. Die Abbildung 2-21 zeigt die Lösung.

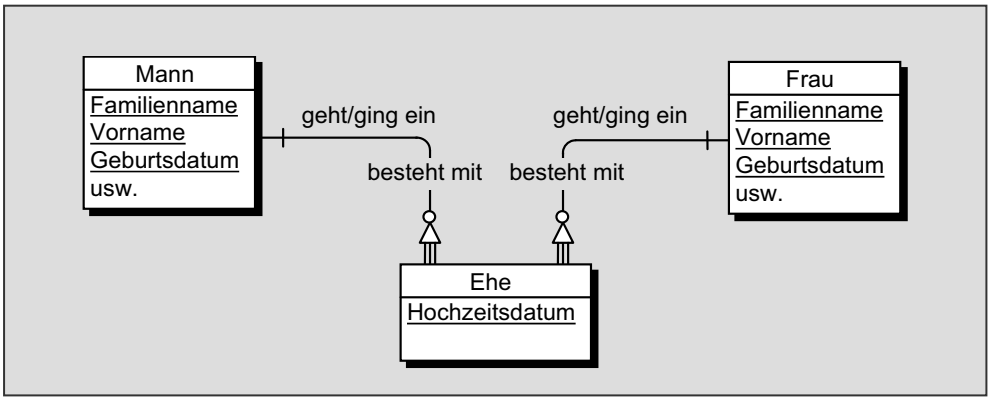


Abb. 2-21: Objekttyp zur Repräsentation eines Beziehungstyps

Für diesen neuen Objekttyp muss natürlich festgelegt werden, wie man seine Objekte identifizieren möchte. Als identifizierende Elemente bieten sich dabei wieder die Beziehungstyp-Richtungen an, die den neuen Objekttyp mit den ursprünglichen Objekttypen verbinden. In unserem Beispiel wurde neben den Beziehungstyp-Richtungen „Ehe *besteht mit* Mann“ und „Ehe *besteht mit* Frau“ als weiteres identifizierendes Element die Eigenschaft „Hochzeitsdatum“ hinzugenommen, um auch dann eine eindeutige Identifizierung der Ehen zu ermöglichen, wenn derselbe Mann und dieselbe Frau mehrfach geheiratet haben.

Rekursiv-Beziehungstyp

Der Fall, dass man die konkrete Beziehung zwischen zwei Objekten a und b näher beschreiben möchte, tritt natürlich nicht nur dann auf, wenn a und b verschiedenen Objekttypen angehören, sondern auch dann, wenn sie demselben Objekttyp entstammen. Im Abschnitt 2.4.5 hatten wir als Beispiel eines Rekursiv-Beziehungstyps das Vertretungsverhältnis zwischen Mitarbeitern modelliert, das in Abbildung 2-22 dargestellt ist.

Nun möge der Fall eintreten, dass man festhalten möchte, welcher Gehaltszuschlag einem Mitarbeiter a gezahlt wird, wenn er

den Mitarbeiter b vertritt. Dies ist eine Eigenschaft, die eigentlich den Rekursiv-Beziehungstyp attributiert. Da das aber die graphische Sprache des Entity-Relationship-Modells nicht zulässt, muss die Eigenschaft „Gehaltszuschlag“ einem neuen Objekttyp zugeordnet werden, der die konkreten Vertretungsbeziehungen zwischen den Mitarbeitern als Objekte enthält. Die Abbildung 2-23 zeigt das Ergebnis.

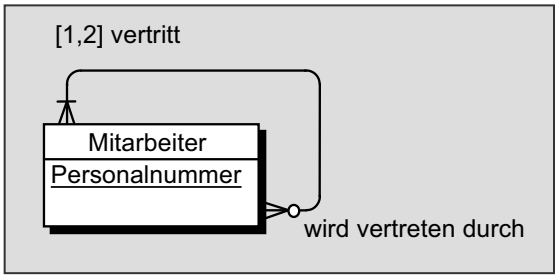


Abb. 2-22: Rekursiv-Beziehungstyp

Die Identifizierung der Objekte des Objekttyps „Vertretung“ erfolgt hier durch die beiden Beziehungstyp-Richtungen „Vertretung *besteht für* Mitarbeiter“ und „Vertretung *wird übernommen von* Mitarbeiter“.

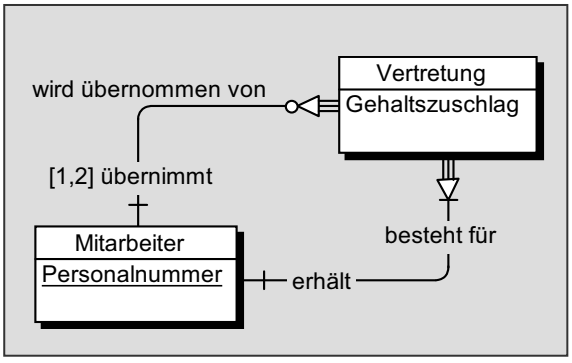


Abb. 2-23: Objekttyp zur Repräsentation eines Rekursiv-Beziehungstyps

Die Beispiele zeigen, dass sich im Entity-Relationship-Modell Objekttypen nicht eindeutig von Beziehungstypen abgrenzen lassen, sondern dass sie ineinander umgewandelt werden können.

Als Faustregel gilt:

Objekttyp oder Beziehungstyp?

Faustregel: Soll über das Zusammenwirken von jeweils zwei Objekten a und b, die zwei verschiedenen Objekttypen (bzw. demselben Objekttyp) entstammen, nicht mehr festgehalten werden als die reine Tatsache, dass a und b gemäß der angegebenen Semantik miteinander verbunden sind, dann wird dies durch einen dualen Beziehungstyp (bzw. durch einen Rekursiv-Beziehungstyp) modelliert.

Sollen jedoch über das konkrete Zusammenwirken von a und b weitere Angaben gespeichert werden, so muss der Beziehungstyp in einen neuen Objekttyp umgewandelt werden, dem die Angaben als Eigenschaften zugeordnet werden.

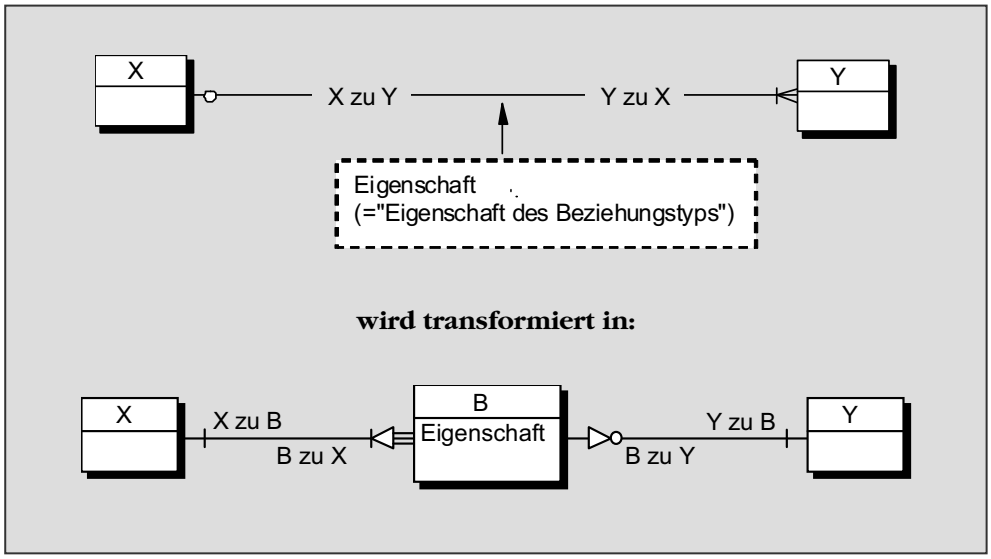


Abb. 2-24: Umwandlung eines Beziehungstyps in einen Objekttyp

Das Umwandeln eines Beziehungstyps zwischen zwei Objekttypen X und Y in einen neuen Objekttyp B folgt dem einfachen Grundmuster, das in Abbildung 2-24 veranschaulicht wird. Da diese Umwandlung nach festen Regeln erfolgt, wird sie von vielen Modellierungs-Tools unterstützt.

Da der neue Objekttyp B das Zusammenspiel genau eines Objekts aus X mit genau einem Objekt aus Y zum Inhalt hat, handelt es sich bei den Beziehungstyp-Richtungen „B zu X“ und „B zu Y“ um nicht-optionale Beziehungstypen der Kardinalität 1, die gleichzeitig auch zur Identifizierung der Objekte in B verwendet werden. Die Optionalitäten und Kardinalitäten der Beziehungstyp-Richtungen „X zu Y“ bzw. „Y zu X“ werden auf die Beziehungstyp-Richtungen „X zu B“ bzw. „Y zu B“ übertragen. Die präzisierende „Eigenschaft des Beziehungstyps“ wird dem neuen Objekttyp B zugeordnet.

Schulbeispiel

In unserem Schulbeispiel soll ebenfalls ein Beziehungstyp durch eine Eigenschaft präzisiert werden:

4. *(...) Herr Meier hat ein Beschäftigungsverhältnis mit der Goethe-Schule seit dem 1.9.1980 und mit der Schiller-Schule seit dem 1.9.1985. (...)*

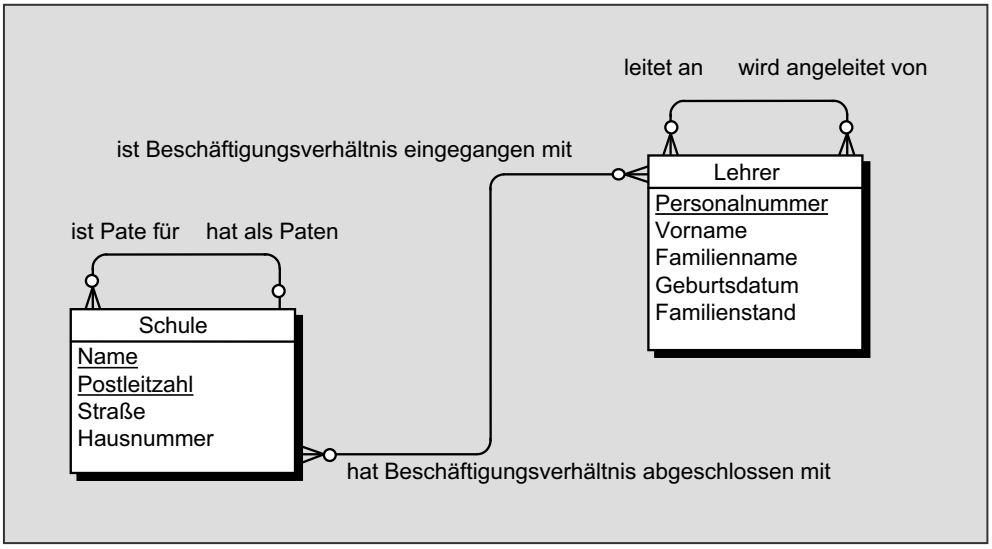


Abb. 2-25: Beziehungstyp zur Beschreibung des Zusammenwirkens von Schule und Lehrer im Schulbeispiel

Bisher wurde durch die Beziehungstyp-Richtungen „Schule *hat Beschäftigungsverhältnis abgeschlossen* mit Lehrer“ und „Lehrer *ist Beschäftigungsverhältnis eingegangen mit* Schule“ lediglich das reine Zusammenwirken einer Schule und eines Lehrers dargestellt, wie es die Abbildung 2-25 zeigt.

Wenn dieser Beziehungstyp nun durch das Vertragsdatum präzisiert werden soll, muss der Beziehungstyp gemäß dem oben beschriebenen Transformationsschema in einen Objekttyp umgewandelt werden. Das Ergebnis ist in Abbildung 2-26 dargestellt.

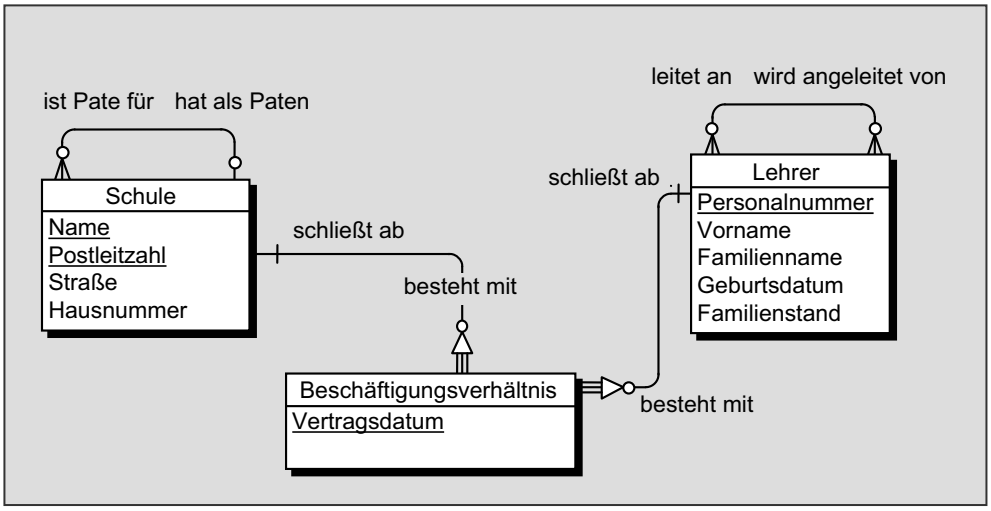


Abb. 2-26: Objekttyp zur Beschreibung des Zusammenwirkens von Schule und Lehrer im Schulbeispiel

Die Identifizierung der Objekte im Objekttyp „Beschäftigungsverhältnis“ erfolgt einerseits durch die beiden Beziehungstyp-Richtungen „Beschäftigungsverhältnis *besteht mit* Schule“ und „Beschäftigungsverhältnis *besteht mit* Lehrer“ und andererseits durch die Eigenschaft „Vertragsdatum“, um auch die Möglichkeit einzuschließen, dass ein Arbeitsvertrag zwischen einer konkreten Schule und einem konkreten Lehrer aufgehoben und zu einem späteren Zeitpunkt ein neuer Arbeitsvertrag abgeschlossen werden kann.

2.5.3 Eigenschaften von Eigenschaften

Schon mehrfach wurde betont, dass ein Datenmodell für einen Gegenstandsbereich kein statisches Gebilde ist, sondern dass es ständig an die sich ändernden Gegebenheiten angepasst werden muss. Besonders häufig tritt dabei der Fall ein, dass zusätzliche Informationen gespeichert werden müssen, durch die Angaben präzisiert werden, die bisher als Eigenschaften von Objekttypen modelliert wurden.

Beispiel für Eigenschaft einer Eigenschaft

Nehmen wir als Beispiel an, dass in einem Unternehmen Informationen über die Autos des Fuhrparks gespeichert werden, wobei natürlich, wie aus der Abbildung 2-27 zu ersehen ist, das polizeiliche Kennzeichen als (organisatorische) identifizierende Eigenschaft verwendet wird.

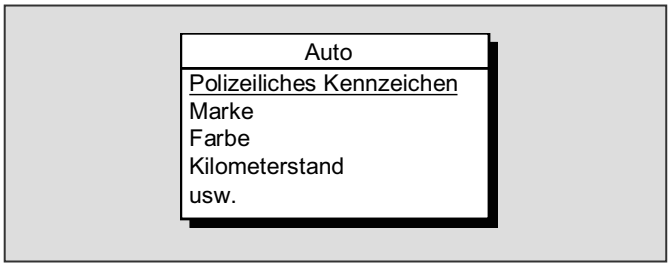


Abb. 2-27: Objekttyp „Auto“ mit der Eigenschaft „Marke“

Subeigenschaft

Nun soll als neue Forderung erhoben werden, dass für die gängigen Automarken der Mindestpreis gespeichert werden soll, zu dem ein Auto dieser Marke in der Grundausstattung gekauft werden kann. Dieser Mindestpreis ist natürlich keine Eigenschaft eines Autos, sondern ist eine Information über die Automarke. Die wurde aber bisher als Eigenschaft des Autos abgespeichert. Damit wäre der Mindestpreis eine „Eigenschaft einer Eigenschaft“, also gewissermaßen eine *„Subeigenschaft“*.

Im Entity-Relationship-Modell sind solche Subeigenschaften nicht darstellbar. Beim Entity-Relationship-Modell handelt es sich nämlich um ein 2-Ebenen-Modell, ein Modell also, das nur zwei hierarchische Beschreibungsebenen zur Verfügung stellt:

1. Ebene der Objekttypen
2. Ebene der Eigenschaften

Die Realität ist in unserer Wahrnehmung aber vielfach gestaffelt, so dass weitere Hierarchieebenen für ihre Beschreibung erforderlich wären. Für die Darstellung von Subeigenschaften gibt es nur eine Möglichkeit: die bisherige Eigenschaft muss auf die Ebene eines Objekttyps – also auf die 1. Ebene - angehoben werden, damit die 2. Ebene – also die Ebene der Eigenschaften – frei wird, um die Subeigenschaft aufzunehmen. Eigenschaft und Subeigenschaft werden also auf die jeweils nächst-höhere Ebene transferiert. Ein zusätzlicher Beziehungstyp sorgt für den „Zusammenhalt" dieser Konstruktion. Das Ergebnis der Transformation zeigt die Abbildung 2-28.

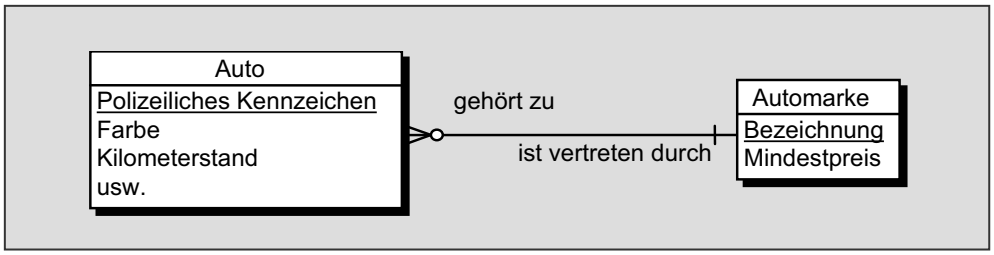


Abb. 2-28: Neuer Objekttyp „Automarke" zur Aufnahme der bisherigen Subeigenschaft „Mindestpreis" als Eigenschaft

Diese Transformation bietet zusätzlich den Vorteil, dass man nun auch Informationen über jene Automarken speichern kann, von denen im Fuhrpark noch keine Autos vorhanden sind. Für diese Fälle wurde die Beziehungstyp-Richtung „Automarke *ist vertreten durch* Auto" als optionale Beziehungstyp-Richtung angegeben.

Die nähere Präzisierung einer Eigenschaft durch eine Subeigenschaft lässt sich somit nach dem in der Abbildung 2-29 dargestellten Grundschema realisieren.

Vorgehen zur Präzisierung einer Eigenschaft

Die Beziehungstyp-Richtung „X zu E" hat stets die Kardinalität 1, da ein Objekt aus X für die ursprüngliche Eigenschaft E höchstens *einen* Wert haben kann (vgl. 1. Normalform im Abschnitt 2.6.1). Ob diese Beziehungstyp-Richtung optional ist, hängt davon ab, ob ursprünglich für jedes X-Objekt ein Wert für die Eigenschaft E angegeben werden musste oder nicht. Die Optionalität und Kardinalität der Beziehungstyp-Richtung „E zu X" sind in Abhängigkeit vom jeweiligen sachlogischen Zusammenhang festzulegen.

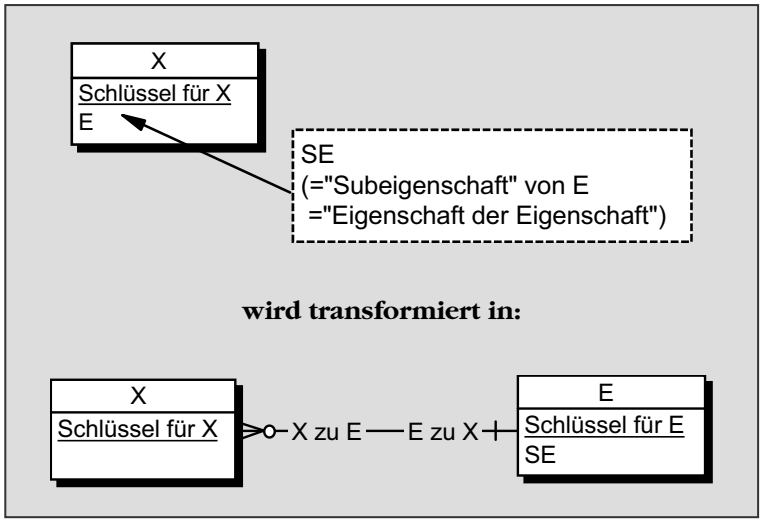


Abb. 2-29: Hinzufügen einer Subeigenschaft

In unserem Schulbeispiel tritt ebenfalls die Situation ein, dass zu einer Information, die bisher als Eigenschaft dargestellt wurde, zusätzliche Angaben gespeichert werden sollen:

Schulbeispiel

4. *(...) Der Unterrichtsraum 205 der Neustädter Goethe-Schule hat eine Fläche von 50 m² mit 32 Sitzplätzen und ist kein Klassenraum. Ein Unterrichtsraum kann maximal 3 Klassen als Klassenraum zugeteilt werden. Der Unterrichtsraum 206 wird in diesem Schuljahr nicht für den Unterricht genutzt. Herr Lehmann gibt der Klasse 10b Unterricht in Geschichte in den Unterrichtsräumen 103 und 301.*

In unserem bisherigen Datenmodell wurde der Unterrichtsraum an zwei Stellen als Eigenschaft angegeben:

- als Eigenschaft „Klassenraum“ im Objekttyp „Klasse“, also als derjenige Unterrichtsraum, der einer Klasse als Klassenraum für Besprechungen usw. zur Verfügung steht, und
- als Eigenschaft „Unterrichtsraum“ im Objekttyp „Unterrichtsverpflichtung“, also als derjenige Unterrichtsraum, der einer Unterrichtsverpflichtung zugeordnet wird.

Diese Situation ist in Abbildung 2-30 dargestellt.

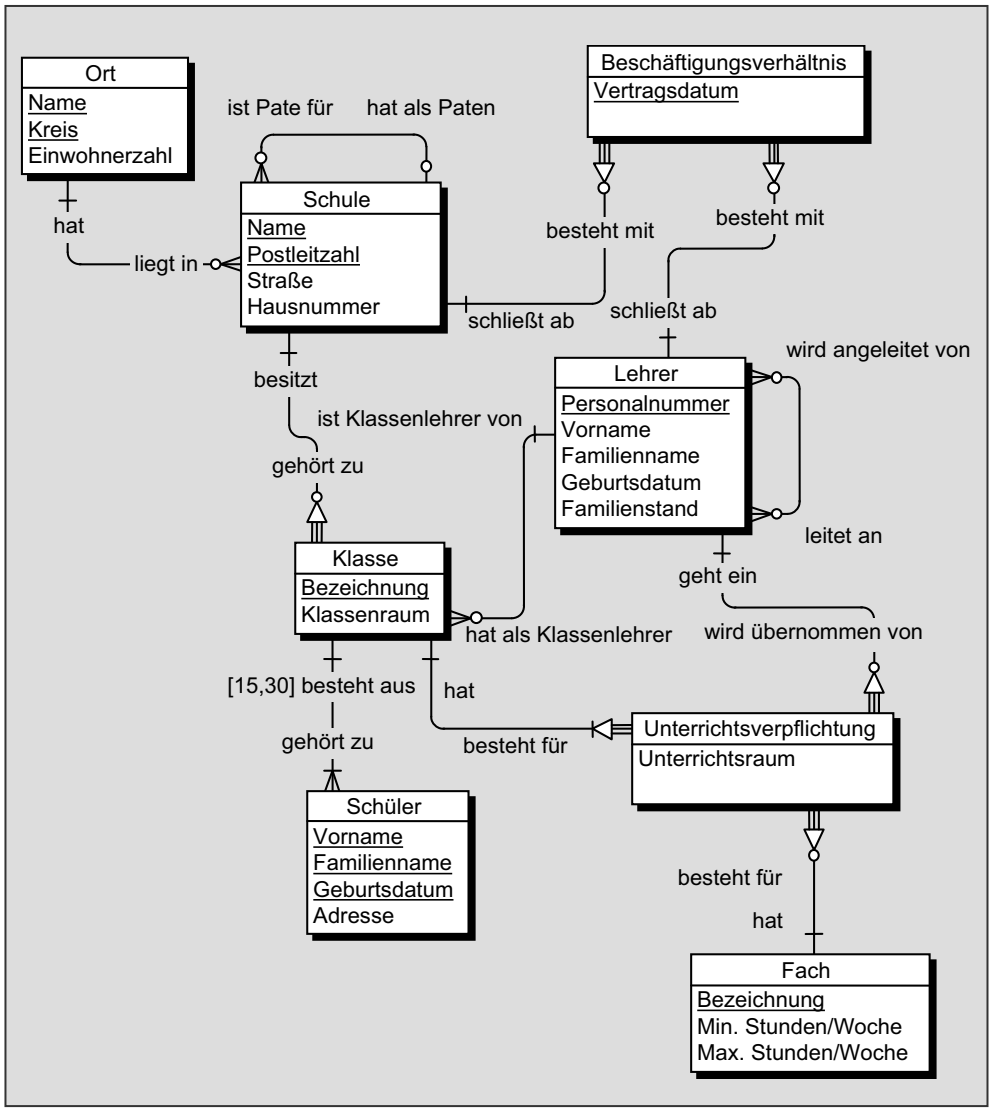


Abb. 2-30: Datenmodell für das Schulbeispiel mit „Klassenraum“ und „Unterrichtsraum“ als Eigenschaften

Nun sollen aber zum Unterrichtsraum spezielle Angaben festgehalten werden, nämlich seine Fläche und seine Sitzplatzanzahl. Dazu muss der Klassenraum bzw. der Unterrichtsraum von seinem bisherigen Status als Eigenschaft in den Status eines Objekt-

typs transformiert werden. Dies geschieht in den folgenden fünf Schritten:

1. Hinzufügen eines neuen Objekttyps „Unterrichtsraum“ mit den Eigenschaften „Nummer“, „Fläche“ und „Sitzplatzanzahl“.
2. Festlegung der Identifizierung der Objekte des Objekttyps „Unterrichtsraum“. Da die „Nummer“ nicht ausreicht, um alle Unterrichtsräume des Bundeslandes voneinander zu unterscheiden, muss die Beziehungstyp-Richtung „Unterrichtsraum *gehört zu* Schule“ hinzugenommen werden. Das erfordert aber die Modellierung des vollständigen Beziehungstyps zwischen Unterrichtsraum und Schule.
3. Streichen der Eigenschaften „Klassenraum“ (im Objekttyp „Klasse“) und „Unterrichtsraum“ (im Objekttyp „Unterrichtsverpflichtung“).
4. Darstellung des Zusammenhangs zwischen einer Klasse und dem als Klassenraum verwendeten Unterrichtsraum durch einen neuen Beziehungstyp mit den Beziehungstyp-Richtungen „Klasse *hat als Klassenraum* Unterrichtsraum“ (und zwar genau einen) und „Unterrichtsraum *ist Klassenraum für* Klasse“ (eventuell für keine, maximal aber für 3 Klassen).
5. Modellierung des Zusammenwirkens von Unterrichtsverpflichtung und Unterrichtsraum durch einen neuen Beziehungstyp mit den Beziehungstyp-Richtungen „Unterrichtsverpflichtung *findet statt in* Unterrichtsraum“ (in einem oder in mehreren Unterrichtsräumen) und „Unterrichtsraum *wird genutzt für* Unterrichtsverpflichtung“ (für keine, eine oder mehrere Unterrichtsverpflichtungen).

In Abbildung 2-31 ist das endgültige Datenmodell für das Schulbeispiel dargestellt.

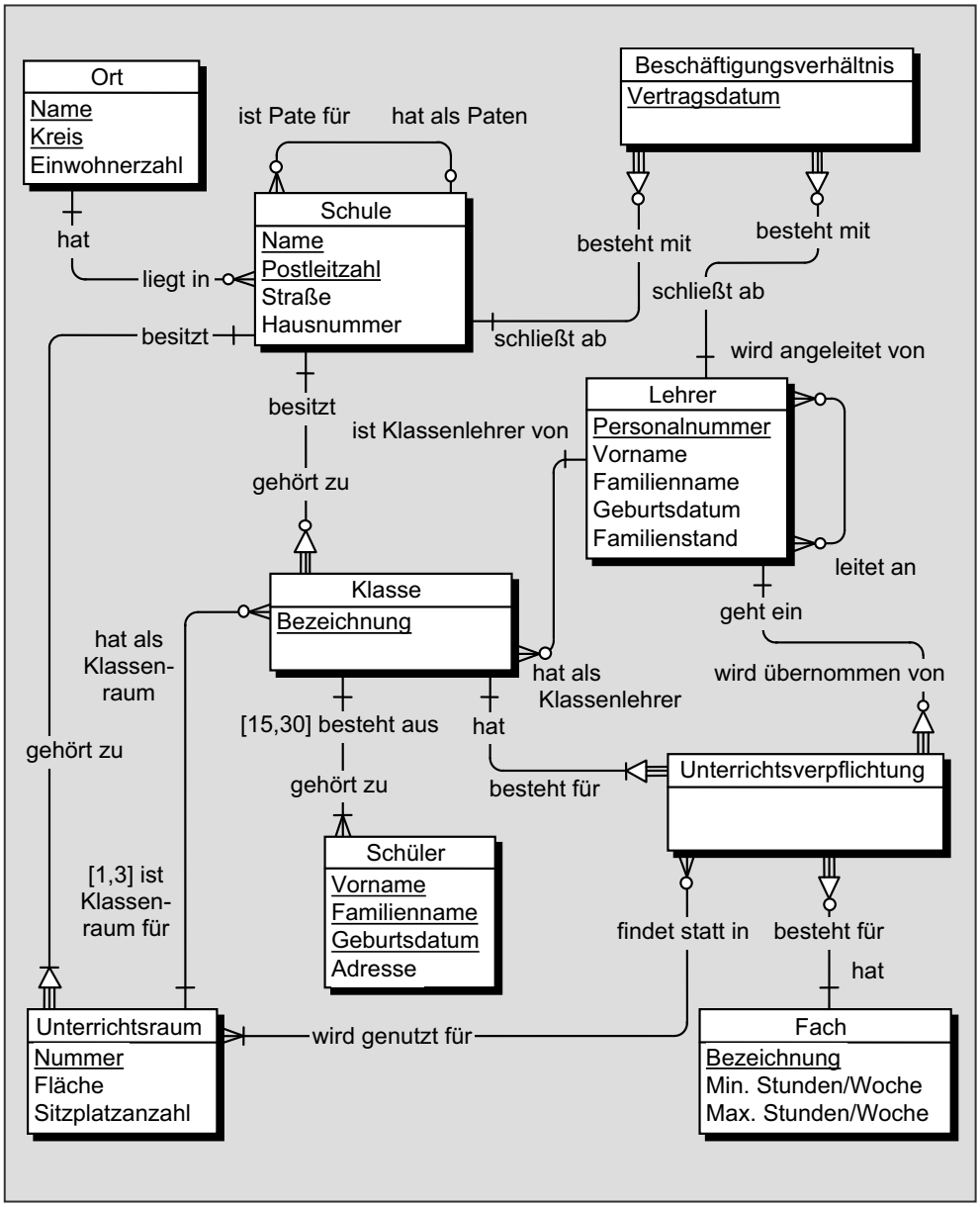


Abb. 2-31: Endgültiges Datenmodell für das Schulbeispiel

2.6 Qualitätssicherung von konzeptionellen Datenmodellen

In den vorangegangenen Abschnitten wurden die Elemente eines konzeptionellen Datenmodells mit den Mitteln der graphischen Sprache des Entity-Relationship-Modells vorgestellt. Es wurde erläutert, wie kompliziertere Situationen im Gegenstandsbereich mit Hilfe dieser eher primitiv gehaltenen Modellierungs-Sprache im Datenmodell abgebildet werden können. Dabei schienen manche Modellierungsentscheidungen eher zufällig zu sein. Insbesondere die Zuordnung von Eigenschaften zu den Objekttypen wurde eher *intuitiv* vorgenommen. Es wäre nun aber wünschenswert, ein Regelwerk zur Korrektheitsprüfung zu besitzen, mit dessen Hilfe man die Qualität eines Datenmodells, das mit Hilfe der Sprache des Entity-Relationship-Modells erstellt wurde, nach *objektiven* Kriterien überprüfen kann.

3 Normalisierungsstufen

In diesem Abschnitt sollen deshalb einige Kriterien besprochen werden, nach denen man die Qualität eines Datenmodells beurteilen kann. Wir beschränken uns dabei auf die ersten drei sog. „Normalisierungsstufen", die zwar im Kontext des relationalen Datenbank-Modells entwickelt wurden, die aber auch auf der Ebene des *konzeptionellen Datenmodells* bedeutungsvoll sind.

Unter dem Prozess der Normalisierung versteht man ein Verfahren, durch das gesichert wird, dass das zu entwickelnde konzeptionelle Datenmodell einige nützliche Standards erfüllt. In der Abbildung 2-32 ist dargestellt, wie aus dem intuitiv aufgestellten (unnormalisierten) Datenmodell schrittweise ein Datenmodell in der 3. Normalform entsteht und mit welchem Ziel der jeweilige Normalisierungsschritt erfolgt.

Filmbeispiel

Die Beschreibung der drei angegebenen Normalisierungsschritte erfolgt an Hand des folgenden durchgehenden Beispiels:

In einem Informationssystem sollen Angaben über Kinofilme gespeichert werden. Für jeden Film werden die folgenden Daten festgehalten:

- *Name des Films*
- *Name des Regisseurs*
- *Kontonummer des Regisseurs*
- *Namen der Schauspieler*
- *Adressen der Schauspieler*
- *Name des Filmstudios*
- *Adresse des Filmstudios*

Dabei hat ein Film nur einen Regisseur und wird nur in einem Filmstudio produziert, er hat aber im allgemeinen mehrere Schauspieler. Der Name eines Regisseurs, eines Schauspielers und eines Filmstudios ist jeweils unikal.

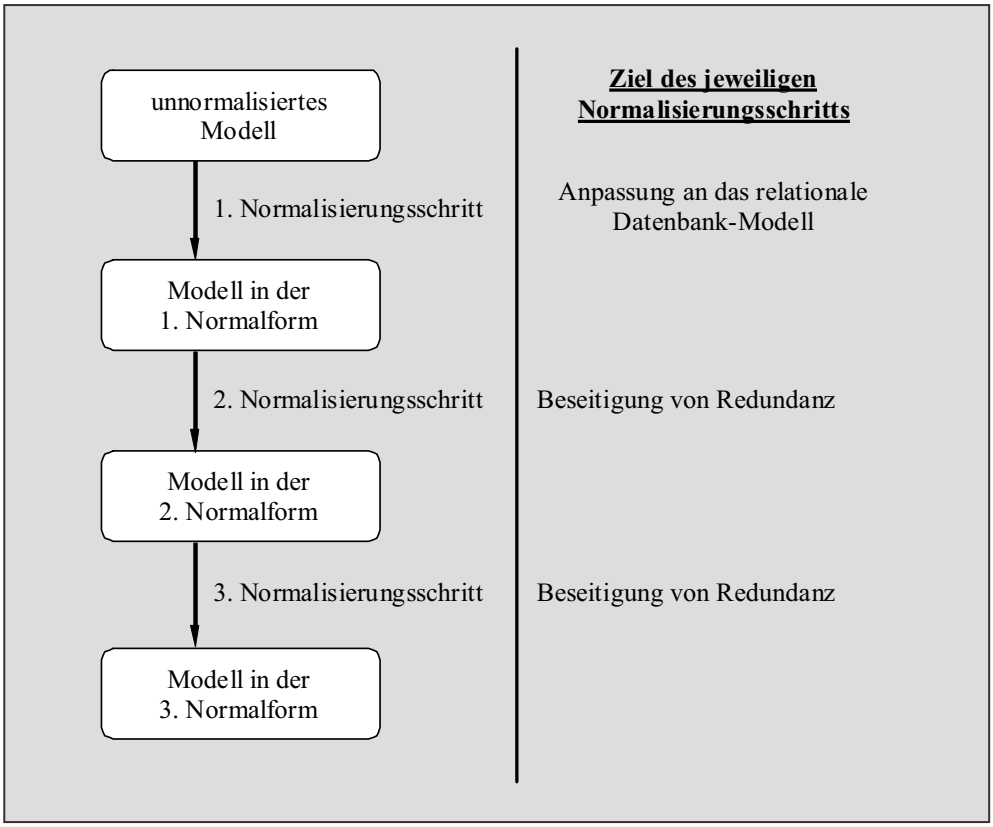


Abb. 2-32: Drei Normalisierungsschritte für das konzeptionelle Datenmodell

Zunächst werden intuitiv alle diese Angaben im Rahmen eines einzigen Objekttyps „Film“ dargestellt: Wir erhalten das in Abbildung 2-33 angegebene *unnormalisierte Datenmodell*, bei dem die Normalisierungsschritte noch nicht ausgeführt wurden. Bei der Entscheidung, in welcher Weise ein Film identifiziert werden soll, wird berücksichtigt, dass ein Filmname zwar mehrfach vor-

kommen kann, dass aber seine Kombination mit dem Namen des Regisseurs unikal ist.

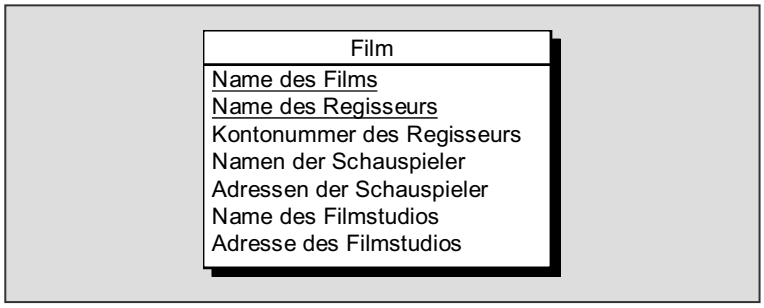


Abb. 2-33: Unnormalisiertes Datenmodell für das Film-Beispiel

Auf dieses unnormalisierte Datenmodell werden nun nacheinander die in den folgenden drei Abschnitten erläuterten Normalisierungs-Prozeduren angewendet.

2.6.1 Die erste Normalform

Der tragende Begriff, der dem 1. Normalisierungsschritt zugrunde liegt, ist der Begriff der *„multiplen Eigenschaft"*:

Multiple Eigenschaft

Definition: Eine *multiple Eigenschaft* eines Objekttyps ist eine Eigenschaft, die für ein gegebenes Objekt dieses Objekttyps zur gleichen Zeit mehrere Werte besitzen kann.

Natürlich ist es immer möglich, dass eine Eigenschaft für ein konkretes Objekt *nacheinander* verschiedene Werte annimmt. So könnte beispielsweise der Familienstand des Lehrers Fritz Fröhlich, der in unserem Beispiel den Wert „ledig" hat, demnächst den Wert „verheiratet" annehmen. Eine multiple Eigenschaft liegt erst dann vor, wenn sie für ein konkretes Objekt *zur gleichen Zeit* mehrere Werte annimmt, wenn man also die Ausprägung der Eigenschaft als Menge bzw. Liste von Werten angeben müsste. Im Film-Beispiel gibt es zwei multiple Eigenschaften: Für die Eigenschaften „Namen der Schauspieler" und „Adressen der Schauspieler" müsste man jeweils eine Liste von Werten angeben, weil an einem Film im allgemeinen mehrere Schauspie-

ler mitwirken. Eigentlich müsste man mehrere Gruppierungen „Schauspielername/Schauspieleradresse“ speichern, um die konkrete Zuordnung einer Adresse zu einem speziellen Schauspieler auszudrücken. Eine solche Gruppe von Eigenschaften, für die zu einem konkreten Objekt eine Liste von Wertegruppierungen gespeichert werden muss, bezeichnet man als *„Wiederholungsgruppe“*.

Wiederholungsgruppe

Das Ziel des 1. Normalisierungsschrittes besteht nun darin, das Datenmodell so zu modifizieren, dass es keine multiplen Eigenschaften besitzt:

1. Normalform

<u>Definition</u>: Ein Datenmodell liegt in der *1. Normalform* vor, wenn es keine multiplen Eigenschaften aufweist.

Der Grund für die Herbeiführung der 1. Normalform liegt in der Anpassung des konzeptionellen Datenmodells an das relationale Datenbank-Modell, das als Eigenschaftswert eines Objekts lediglich einen atomaren Wert - also keine Menge bzw. Liste von Werten - zulässt. Um diese Beschränkung zu überwinden, werden im Kontext der objektorientierten Datenbank-Managementsysteme sogenannte NF^2-Datenbank-Modelle entwickelt. Dabei steht die Bezeichnung

NF^2–Modell

„NF^2–Modell“ für:
„NFNF-Modell“ und dies wiederum für:
„Non-First-Normal-Form-Modell“,

also für ein solches Datenbank-Modell, das die 1. Normalform nicht erfordert. Diese Datenbank-Modelle lassen als Wert einer Eigenschaft komplexe Werte-Aggregationen zu, deren Werte rekursiv wieder aus komplexen Werte-Aggregationen bestehen können. Im Rahmen dieses Lehrbuchs wird die Datenmodellierung jedoch für das relationale Datenbank-Modell beschrieben, so dass ein gegebenes konzeptionelles Datenmodell in jedem Fall in die 1. Normalform zu überführen ist.

Herbeiführen der 1. Normalform

Das Eliminieren einer multiplen Eigenschaft ME aus einem Objekttyp X lässt sich dadurch erreichen, dass man ME aus X herauslöst und zu einem eigenen Objekttyp ME werden lässt. Der sachlogische Zusammenhang wird durch einen Beziehungstyp zwischen X und ME dargestellt, wobei die Beziehungstyp-Richtung „X zu ME“ die Kardinalität N aufweist.

Der 1. Normalisierungsschritt wird also durch das algorithmische Vorgehen beschrieben, das Abbildung 2-34 wiedergibt.

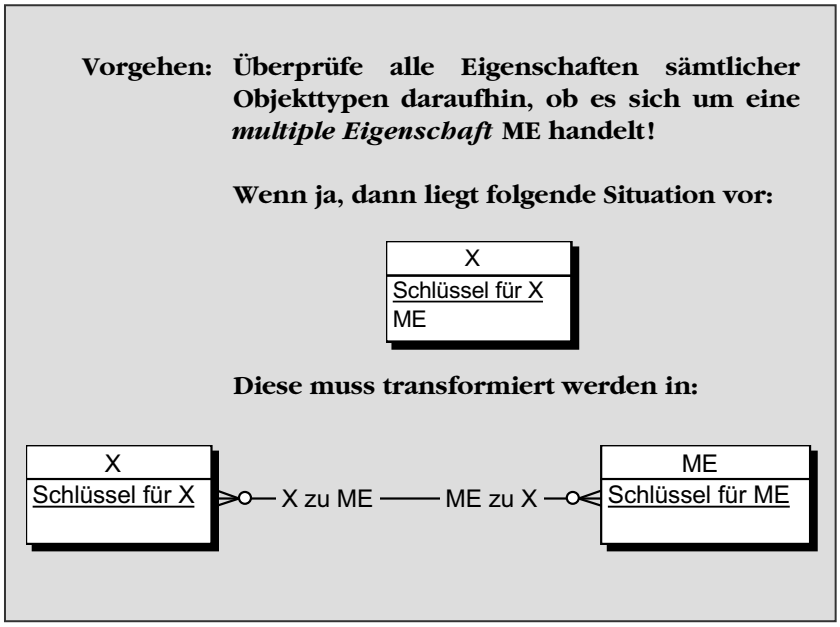


Abb. 2-34: Algorithmus des 1. Normalisierungsschritts

Eliminieren einer Wiederholungsgruppe

Liegt eine Wiederholungsgruppe vor, so werden alle ihre Eigenschaften nach ME transferiert, wobei i.d.R. eine davon die Schlüsselfunktion übernimmt. Die Beziehungstyp-Richtung „X zu ME“ muss natürlich die Kardinalität N haben, damit ein Objekt aus X mit mehreren Objekten aus ME in Beziehung gebracht werden kann. Durch die beschriebene Transformation wird die ursprüngliche Multiplizität der Eigenschaft ME in die Kardinalität N der Beziehungstyp-Richtung „X zu ME“ überführt. Sie wird damit in der Sprache des Entity-Relationship-Modells darstellbar. Die sonstigen Optionalitäten und Kardinalitäten beider Bezie-

hungstyp-Richtungen müssen entsprechend den sachlogischen Gegebenheiten gewählt werden.

Filmbeispiel

Im Filmbeispiel müssen die beiden multiplen Eigenschaften „Namen der Schauspieler“ und „Adressen der Schauspieler“ gemäß dem beschriebenen Algorithmus in den neuen Objekttyp „Schauspieler“ transferiert werden. Das führt zum Datenmodell in der 1. Normalform, das die Abbildung 2-35 zeigt.

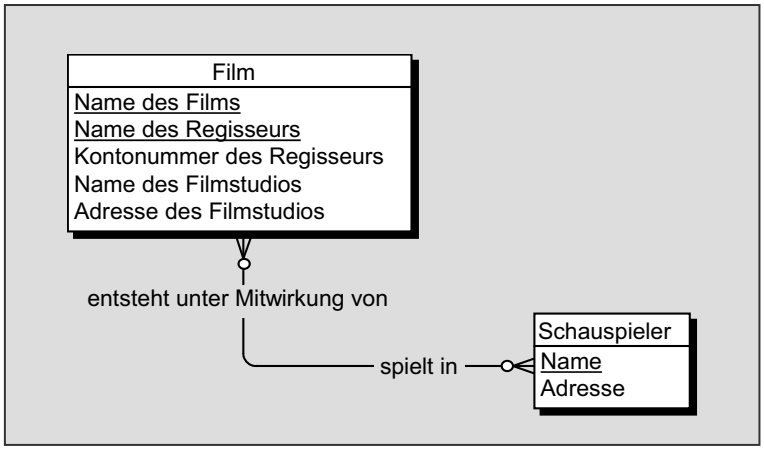


Abb. 2-35: Datenmodell für das Film-Beispiel in der 1. Normalform

Im Kontext des beschriebenen 1. Normalisierungsschrittes wird nun die im Abschnitt 2.2 erhobene Forderung verständlich, dass die Benennung einer Eigenschaft steht im Singular gewählt werden soll. Damit soll nämlich schon während der „intuitiven“ Modellierung verhindert werden, dass multiple Eigenschaften ins Datenmodell gelangen. Eigenschaftsbenennungen wie „Namen der Schauspieler“ und „Adressen der Schauspieler“ sind eigentlich von vornherein unzulässig, so dass sich schon an dieser Stelle die Notwendigkeit ergeben hätte, einen separaten Objekttyp „Schauspieler“ einzuführen.

2.6.2 Die zweite Normalform

Der Begriff, der sowohl für die zweite als auch für die dritte Normalform eine fundamentale Bedeutung besitzt, ist der Begriff der „funktionalen Abhängigkeit". Wir wollen diesen Begriff zunächst im *engeren Sinne* betrachten:

Funktionale Abhängigkeit im engeren Sinne

Definition: Innerhalb eines Objekttyps ist eine Eigenschaft B dann von einer Eigenschaft A *funktional abhängig*, wenn sich für jedes konkrete Objekt dieses Objekttyps aus dem Wert der Eigenschaft A direkt auf den Wert der Eigenschaft B schließen lässt.

Funktionale Abhängigkeit liegt also dann vor, wenn man – zumindest gedanklich – eine Konkordanzliste aufstellen kann, durch die jedem Wert der Eigenschaft A eindeutig ein Wert der Eigenschaft B zugeordnet wird:

Eigenschaft A	Eigenschaft B
Wert a1	Wert b1
Wert a2	Wert b2
usw.	usw.

Beispielsweise ist im Objekttyp „Lehrer" der Abbildung 2-36 die Eigenschaft „Familienname" von der Eigenschaft „Personalnummer" funktional abhängig, weil nämlich von jedem Wert der Eigenschaft „Personalnummer" unmittelbar auf den zugehörigen Wert der Eigenschaft „Familienname" geschlossen werden kann.

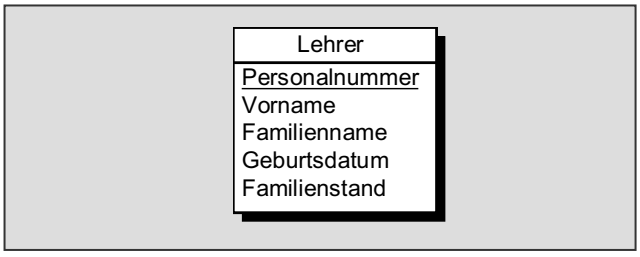


Abb. 2-36: Funktionale Abhängigkeit einer Eigenschaft von einer anderen Eigenschaft

Für die Beschreibung des 2. Normalisierungsschrittes muss der Begriff der funktionalen Abhängigkeit jedoch allgemeiner gefasst werden: Eine Eigenschaft B kann auch von einer Kombination von Elementen E_1, E_2, ..., E_n funktional abhängig sein, wobei ein Element E_i entweder eine Eigenschaft oder eine Beziehungstyp-Richtung sein kann. Wir definieren den Begriff der funktionalen Abhängigkeit im *weiteren Sinne* wie folgt:

Funktionale Abhängigkeit im weiteren Sinne

Definition: Innerhalb eines Objekttyps ist eine Eigenschaft B dann von einer Kombination von Elementen (Eigenschaften und/oder Beziehungstyp-Richtungen) E_1, E_2, ..., E_n *funktional abhängig*, wenn sich für jedes konkrete Objekt dieses Objekttyps aus der Kombination der Werte dieser Elemente direkt auf den Wert der Eigenschaft B schließen lässt.

Beispielsweise ist im Objekttyp „Unterrichtsraum“ der Abbildung 2-37 die Eigenschaft „Fläche“ funktional abhängig von der Kombination zweier Elemente, nämlich von der Beziehungstyp-Richtung E_1 = „Unterrichtsraum *gehört zu* Schule“ und von der Eigenschaft E_2 = „Nummer“. Wenn nämlich bekannt ist, zu welcher Schule ein Unterrichtsraum gehört und welche (schulinterne) Nummer er trägt, dann kann unmittelbar auf den zugehörigen Wert der Eigenschaft „Fläche“ geschlossen werden.

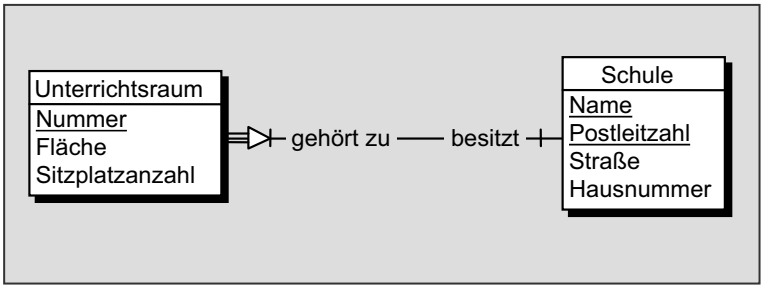


Abb. 2-37: Funktionale Abhängigkeit einer Eigenschaft von einer Kombination aus einer Beziehungstyp-Richtung und einer Eigenschaft

Für die 2. Normalform gilt nun die folgende Definition:

2. Normalform

Definition: Ein Datenmodell liegt in der *2. Normalform* vor, wenn es sich in der 1. Normalform befindet und jede beschreibende Eigenschaft eines Objekttyps zwar vom Gesamtschlüssel, aber nicht bereits von einem Teilschlüssel dieses Objekttyps funktional abhängig ist.

Eine Verletzung der Forderung der 2. Normalform kann logischerweise nur bei Objekttypen auftreten, die einen *zusammengesetzten Schlüssel* aufweisen, also einen Schlüssel, der sich aus mehreren teilidentifizierenden Elementen – das können Eigenschaften und/oder Beziehungstyp-Richtungen sein – zusammensetzt. Nur bei diesen kann nämlich der Fall eintreten, dass eine beschreibende Eigenschaft bereits schon von einem Teilschlüssel und nicht nur vom Gesamtschlüssel funktional abhängig ist.

Filmbeispiel

Im Filmbeispiel liegt eine Verletzung der 2. Normalform vor. Der Gesamtschlüssel des Objekttyps „Film“ besteht aus zwei Elementen: aus den teilidentifizierenden Eigenschaften „Name des Films“ und „Name des Regisseurs“. Die beschreibende Eigenschaft „Kontonummer des Regisseurs“ ist jedoch bereits vom Teilschlüssel „Name des Regisseurs“ funktional abhängig. Um die Kontonummer des Regisseurs angeben zu können, muss man nämlich den Namen des Films gar nicht kennen, sondern lediglich den Namen des Regisseurs.

Das Ziel der Herbeiführung der 2. Normalform liegt in der Beseitigung von Redundanzen bei der Datenspeicherung. Das wird deutlich, wenn man die Speicherung mehrerer Filme nach dem Datenmodell der 1. Normalform betrachtet, wobei die in diesem Zusammenhang unwesentlichen Schauspieler unberücksichtigt bleiben. Abbildung 2-38 zeigt die Situation.

Man erkennt, dass bei allen Filmen, die der Regisseur „Konrad Kurbel“ gedreht hat, dessen Kontonummer „1234567890“ abgespeichert wird. Einerseits führt das zu unnötig belegtem Speicherplatz und zu einem zusätzlichen Aufwand bei der Informationseingabe. Andererseits birgt das große Gefahren in sich bei der Datenänderung: Erhält nämlich der Regisseur Konrad Kurbel eine neue Kontonummer, muss diese Änderung bei allen seinen Filmen vorgenommen werden. Vergisst man dabei einen seiner

Filme, entstehen auf Grund der unterschiedlichen Kontoangaben Widersprüche in den Daten.

Herbeiführen der 2. Normalform

Die 2. Normalform lässt sich wie folgt herbeiführen: Ist in einem Objekttyp X eine Eigenschaft E vom Teilschlüssel TS funktional abhängig, dann werden TS und E aus X entfernt und bilden einen neuen Objekttyp Y mit dem Schlüssel TS. Der logische Zusammenhang wird durch einen Beziehungstyp zwischen X und Y repräsentiert. Der nun in X fehlende Teil des Schlüssels wird durch die Beziehungstyp-Richtung „X zu Y“ ersetzt.

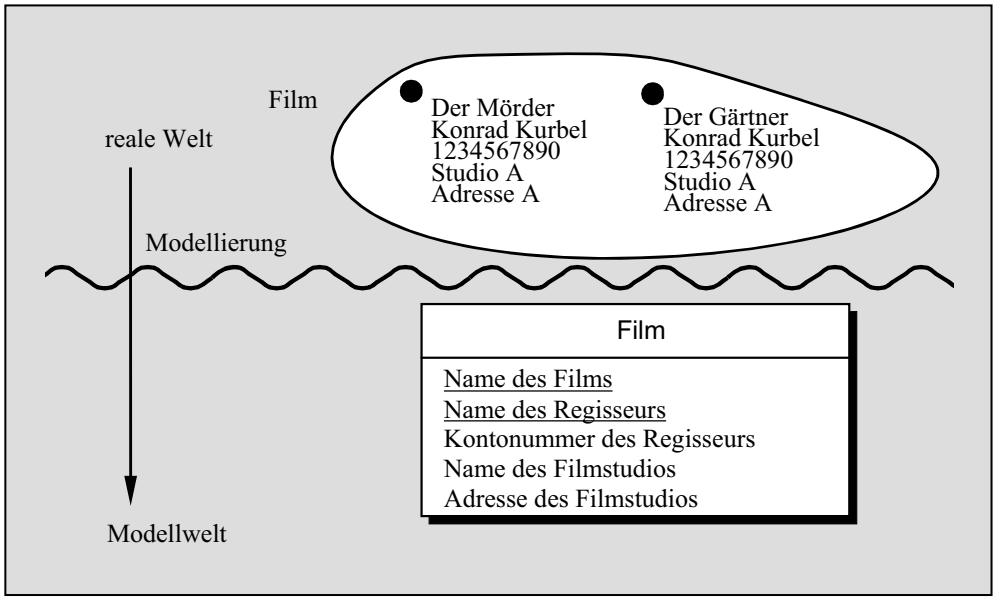


Abb. 2-38: Redundanz bei Verletzung der 2. Normalform

Der 2. Normalisierungsschritt wird – bezogen auf einen leicht präzisierten Fall - durch das algorithmische Vorgehen beschrieben, das in Abbildung 2-39 dargestellt wird.

Die Beziehungstyp-Richtung „X zu Y“ ist nicht-optional (weil der ursprüngliche Teilschlüssel S2 für jedes X-Objekt einen Wert haben musste) und hat stets die Kardinalität 1 (sonst wäre ja die ursprüngliche teilidentifizierende Eigenschaft S2 eine multiple Eigenschaft). Die Beziehungstyp-Richtung „X zu Y“ ermöglicht so die Teilidentifizierung von X. Die Kardinalität der Beziehungstyp-

Richtung „Y zu X“ ist stets N (sonst wären ja S1 und S2 voneinander funktional abhängig und könnten nicht gemeinsam den Schlüssel von X bilden). Ihre Optionalität ist in Abhängigkeit vom sachlogischen Zusammenhang zu wählen.

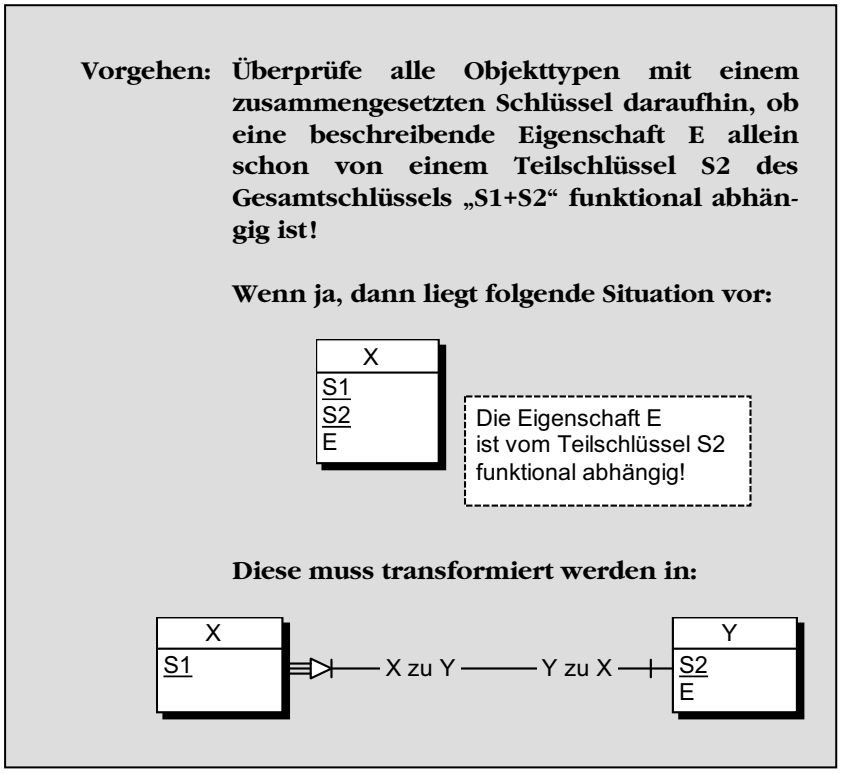


Abb. 2-39: Algorithmus des 2. Normalisierungsschritts

Filmbeispiel

Im Filmbeispiel müssen gemäß dem oben beschriebenen Vorgehen die teilidentifizierende Eigenschaft „Name des Regisseurs“ und die von ihr funktional abhängige beschreibende Eigenschaft „Kontonummer des Regisseurs“ aus dem Objekttyp „Film“ herausgelöst werden. Sie bilden den neuen Objekttyp „Regisseur“, der mit dem Objekttyp „Film“ durch einen 1:N-Beziehungstyp verbunden wird. Das entstehende Datenmodell in der 2. Normalform ist in Abbildung 2-40 dargestellt.

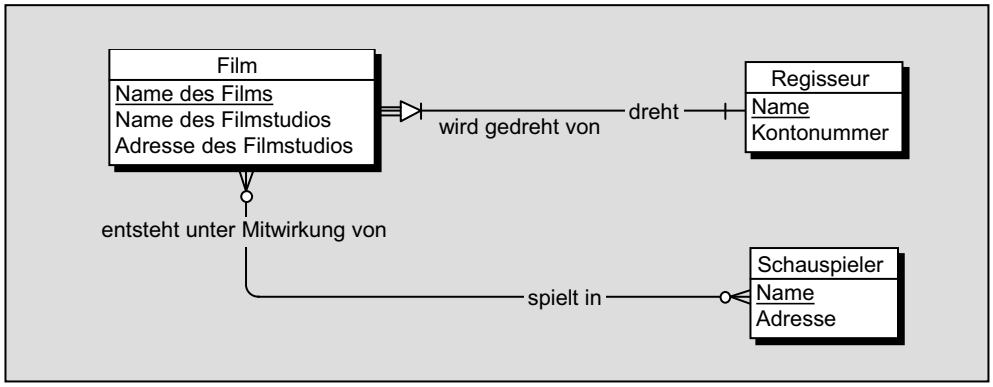


Abb. 2-40: Datenmodell für das Film-Beispiel in der 2. Normalform

Die Beziehungstyp-Richtung „Regisseur *dreht* Film“ wurde als nicht-optional festgelegt, weil angenommen wird, dass ein Regisseur erst dann im Informationssystem gespeichert wird, wenn er den ersten Film dreht.

Weiter oben wurde als Grund für den 2. Normalisierungsschritt angegeben, dass dadurch die Redundanz vermieden werden soll, die in Abbildung 2-38 veranschaulicht wurde. Dass dies tatsächlich der Fall ist, zeigt die Abbildung 2-41, bei der dieselben Informationen nach dem Modell der 2. Normalform gespeichert sind (wieder unter Vernachlässigung der Schauspieler).

Man erkennt, dass hier dasselbe Verfahren zur Redundanzvermeidung angewendet wird wie bei der traditionellen Informationsspeicherung: Die zuvor an mehreren Stellen redundant gespeicherten Angaben werden nun an anderer Stelle einmalig abgespeichert. Damit man an den Stellen, an denen sie zuvor standen, trotzdem auf diese Daten zurückgreifen kann, wird durch eine Beziehung auf die nun redundanzfrei gespeicherten Informationen verwiesen.

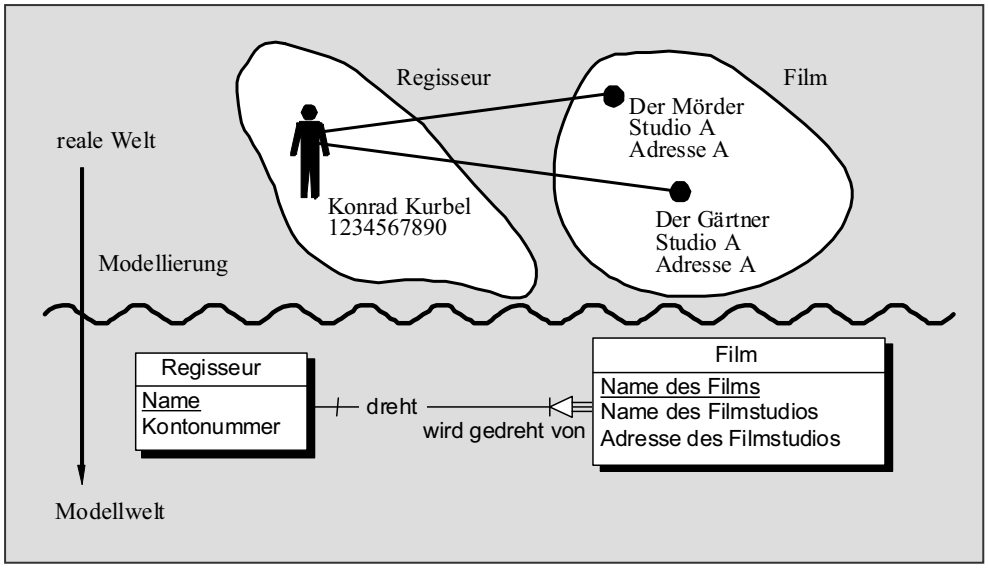


Abb. 2-41: Redundanzvermeidung durch 2. Normalform

Das bekannte Prinzip *„Object Linking and Embedding"* (OLE) von Microsoft basiert ebenfalls auf diesem Vorgehen. Beim „Object Embedding" (Einbetten von Objekten) wird ein fremdes Objekt (beispielsweise eine Grafik) in ein Container-Objekt (häufig ein Text) redundant „hineinkopiert". Beim „Object Linking" bleibt das fremde Objekt dagegen redundanzfrei einmalig gespeichert, und im Container-Objekt wird lediglich auf das Objekt verwiesen.

2.6.3 Die dritte Normalform

Für die Darlegung des 3. Normalisierungsschrittes ist wiederum eine Erweiterung des Begriffs der *funktionalen Abhängigkeit* erforderlich:

Transitiv funktionale Abhängigkeit

Definition: Innerhalb eines Objekttyps ist eine Eigenschaft B von einer Kombination von Elementen $E = E_1, E_2, ..., E_n$ (bestehend aus Eigenschaften und/oder Beziehungstyp-Richtungen) *transitiv funktional abhängig*, wenn B nicht auf direktem Wege, sondern vermittelt über eine weitere

Eigenschaft C von E funktional abhängig ist, d.h.: B ist von C und C ist von E funktional abhängig.

Filmbeispiel

In unserem Filmbeispiel bestehen für den Objekttyp „Film“ die folgenden Abhängigkeiten:

- Die Eigenschaft „Adresse des Filmstudios“ ist von der Eigenschaft „Name des Filmstudios“ *funktional abhängig.*
- Die Eigenschaft „Name des Filmstudios“ ist von einer Kombination aus der Eigenschaft „Name des Films“ und der Beziehungstyp-Richtung „Film *wird gedreht von* Regisseur“ *funktional abhängig.*
- Also ist die Eigenschaft „Adresse des Filmstudios“ von einer Kombination aus der Eigenschaft „Name des Films“ und der Beziehungstyp-Richtung „Film *wird gedreht von* Regisseur“ *transitiv funktional abhängig.*

Für die 3. Normalform gilt nun die folgende Definition:

3. Normaform (Definition 1)

Definition: Ein Datenmodell liegt in der *3. Normalform* vor, wenn es sich in der 2. Normalform befindet und keine beschreibende Eigenschaft eines Objekttyps vom Schlüssel dieses Objekttyps transitiv funktional abhängig ist.

Eine in der Literatur häufig angegebene einfachere – und zu der angegebenen Definition in den meisten Fällen äquivalente – Definition lautet:

3. Normalform (Definition 2)

Definition: Ein Datenmodell liegt in der *3. Normalform* vor, wenn es sich in der 2. Normalform befindet und keine beschreibende Eigenschaft eines Objekttyps von einer anderen beschreibenden Eigenschaft dieses Objekttyps funktional abhängig ist.

Eine Verletzung dieser Forderung kann natürlich nur bei Objekttypen auftreten, die wenigstens zwei beschreibende Eigenschaften besitzen.

Filmbeispiel

Im Filmbeispiel liegt beim Objekttyp „Film“ eine Verletzung der 3. Normalform vor. Wie bereits erläutert wurde, ist die Eigenschaft „Adresse des Filmstudios“ vom Schlüssel des Objekttyps „Film“ *transitiv funktional abhängig*.

Der Grund für die Herbeiführung der 3. Normalform liegt wiederum in der Vermeidung von Redundanzen bei der Datenspeicherung. Das ist aus Abbildung 2-42 ersichtlich, wenn man sich die Speicherung mehrerer Filme nach dem Modell der 2. Normalform näher anschaut und dabei die Schauspieler und Regisseure unberücksichtigt lässt.

Man erkennt, dass bei sämtlichen Filmen, die im „Studio A“ produziert werden (und das können sehr viele sein), redundant die „Adresse A“ gespeichert wird. Das bedeutet wieder: Vergeudung von Speicherplatz, erhöhter Eingabeaufwand und die Gefahr von Inkonsistenzen, wenn eine Adressenänderung des Filmstudios nicht bei allen Filmen vermerkt wird, die von diesem Filmstudio produziert werden.

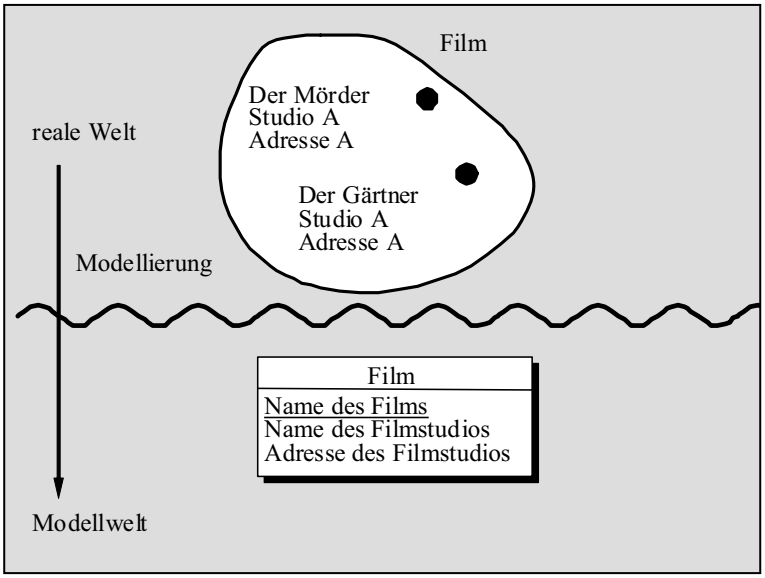


Abb. 2-42: Redundanz bei Verletzung der 3. Normalform

Die 3. Normalform eines Datenmodells lässt sich durch das folgende Verfahren gewinnen. Wir nehmen an, dass ein Objekttyp X außer seinem Schlüssel S wenigstens zwei beschreibende

Eigenschaft E1 und E2 enthält. Dabei ist es unerheblich, ob es sich bei S um einen elementaren oder einen zusammengesetzten Schlüssel handelt. Wir setzen weiterhin voraus, dass E2 vom Schlüssel S transitiv funktional abhängig ist, dass nämlich einerseits E2 von E1 und andererseits E1 von S funktional abhängig sind. Dann werden E1 und E2 aus X entfernt und bilden einen neuen Objekttyp Y mit E1 als Schlüssel und E2 als beschreibender Eigenschaft. Der logische Zusammenhang wird auch hier durch einen Beziehungstyp zwischen X und Y dargestellt.

Herbeiführen der 3. Normalform

Der 3. Normalisierungsschritt kann also durch das algorithmische Vorgehen beschrieben werden, das in Abbildung 2-43 dargestellt ist.

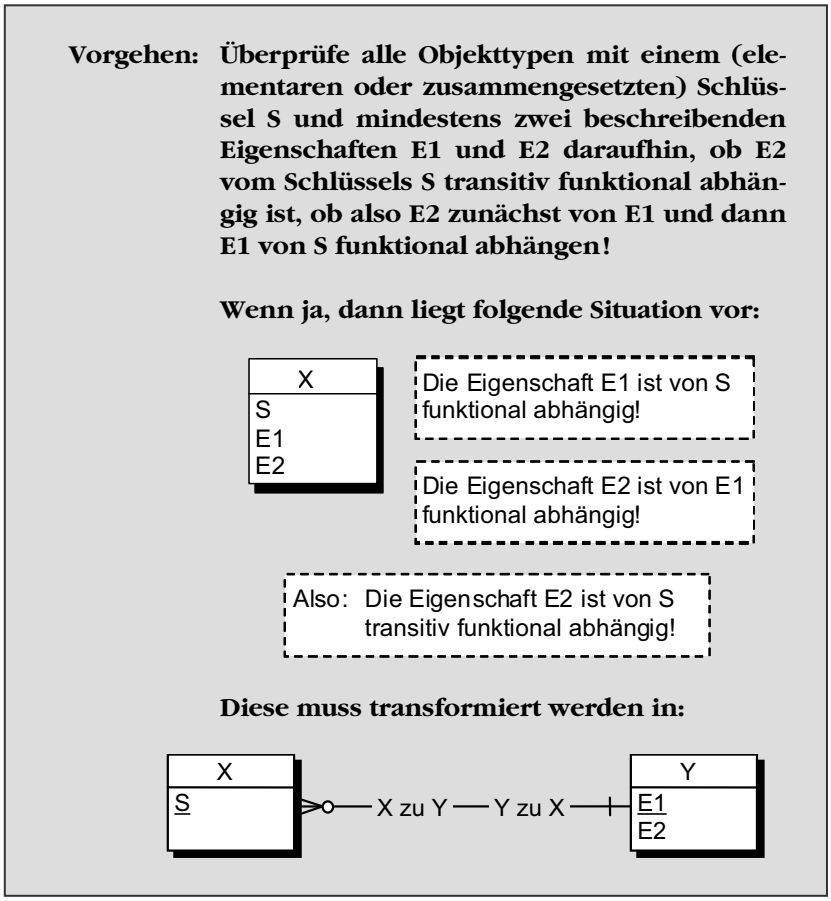


Abb. 2-43: Algorithmus des 3. Normalisierungsschritts

Die Beziehungstyp-Richtung „X zu Y“ hat stets die Kardinalität 1 (sonst wäre E1 im Objekttyp X eine multiple Eigenschaft gewesen). Die weiteren Optionalitäten und Kardinalitäten der beiden Beziehungstyp-Richtungen ergeben sich aus dem sachlogischen Zusammenhang.

Filmbeispiel

Im Filmbeispiel müssen die beiden beschreibenden Eigenschaften „Name des Filmstudios“ und „Adresse des Filmstudios“ aus dem Objekttyp „Film“ herausgelöst werden. Sie bilden – wie in Abbildung 2-44 zu sehen ist - den neuen Objekttyp „Filmstudio“.

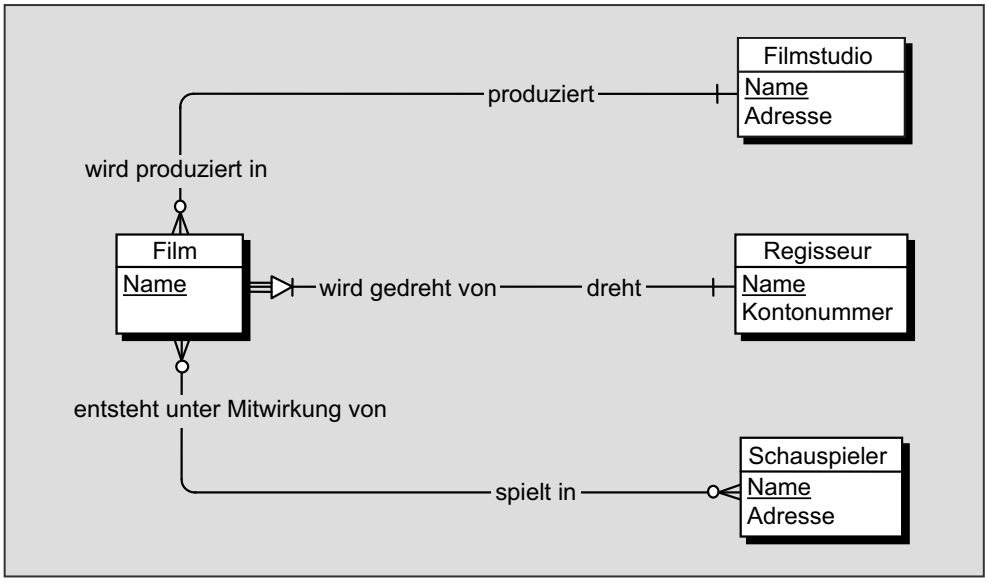


Abb. 2-44: Datenmodell für das Film-Beispiel in der 3. Normalform

Dabei wurde angenommen, dass einerseits jeder Film in einem Filmstudio produziert wird und dass andererseits ein Filmstudio mehrere Filme produzieren kann, dass es aber auch als „künftige Option“ gespeichert werden kann, wenn es noch keinen Film produziert hat. Dieser Umstand offenbart – über die Redundanzvermeidung hinaus - einen weiteren Vorteil des Normalisierungsprozesses: Es wird nun nämlich möglich, Informationen über ein Filmstudio festzuhalten, in dem noch keiner der gespeicherten Filme produziert wurde. Im unnormalisierten Modell war das nicht möglich, weil sämtliche Informationen am Film „festgemacht“ wurden.

Dass durch die 3. Normalform die oben besprochene Redundanz vermieden wird, zeigt die Abbildung 2-45.

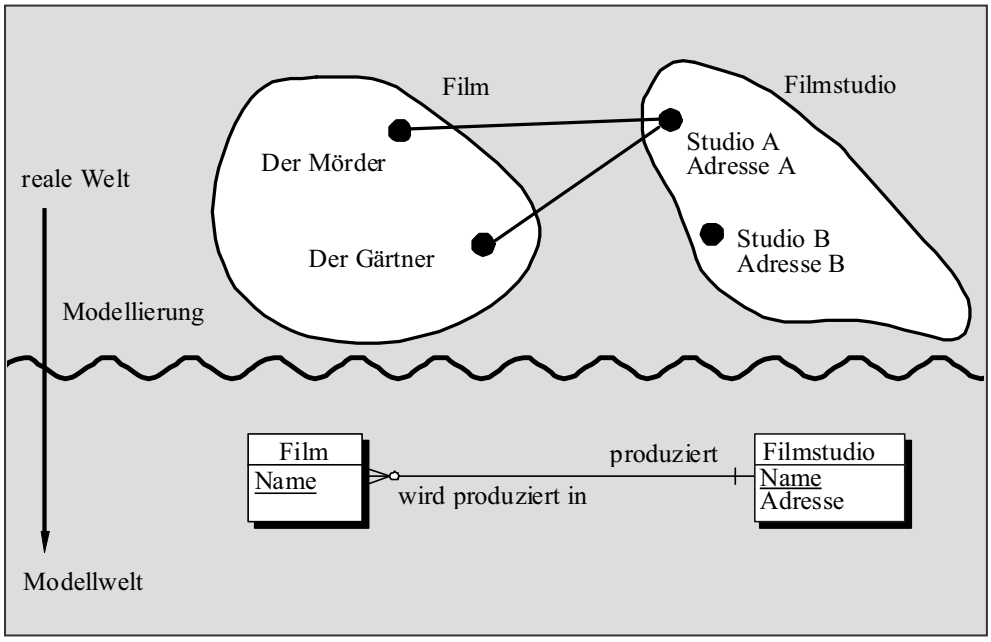


Abb. 2-45: Redundanzvermeidung durch 3. Normalform

Zur Redundanzvermeidung werden nun die Angaben zu einem Filmstudio nur noch an einer Stelle abgespeichert, nämlich als ein Objekt im Objekttyp „Filmstudio“. Von den entsprechenden Objekten des Objekttyps „Film“ wird durch Beziehungen auf dieses Filmstudio verwiesen.

2.6.4 Denormalisierung

In den vorangegangenen Abschnitten wurde der Prozess der Normalisierung in seinen ersten drei Stufen beschrieben. Neben der Anpassung des konzeptionellen Datenmodells an das relationale Datenbank-Modell (1. Normalform) dienten sie hauptsächlich der Beseitigung von Datenredundanzen (2. und 3. Normalform). Durch die Einführung neuer Objekttypen wurde verhindert, dass dieselben Aussagen mehrfach in der Datenbank abgelegt werden. Redundante Daten sind in der Datenbank

unerwünscht, weil einerseits unnötig Speicherplatz und menschliche Arbeitskraft für die Dateneingabe vergeudet wird und weil andererseits bei Datenänderungen inkonsistente Datenzustände entstehen können. Die Ausführung der besprochenen Normalisierungsschritte ist deshalb für die Entwicklung qualitätsgerechter Datenmodelle von fundamentaler Bedeutung.

Denormalisierung

In manchen Fällen - insbesondere im Interesse der Performance-Verbesserung des zu entwickelnden Anwendungssystems - kann es jedoch wünschenswert sein, einen Normalisierungsschritt wieder rückgängig zu machen. Man spricht dann von *Denormalisierung*. Auf eine solche Denormalisierung sollte dann aber im Modell ausdrücklich hingewiesen werden, weil hier - im Interesse einer besseren Performance - bewusst eine Quelle eventueller Speicheranomalien in Kauf genommen wird.

Filmbeispiel

Zur Erläuterung der Denormalisierung soll unser Filmbeispiel etwas präzisiert werden, indem die Adresse des Schauspielers in ihre Bestandteile zerlegt wird. Den so modellierten Objekttyp „Schauspieler“ zeigt die Abbildung 2-46.

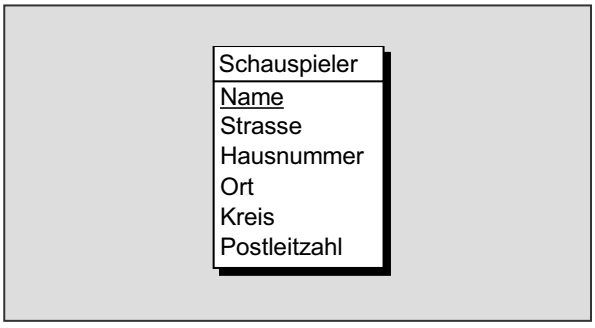


Abb. 2-46: Objekttyp „Schauspieler“ mit Verletzung der 3. Normalform

Der Objekttyp „Schauspieler“ liegt nicht in der 3. Normalform vor: Die beschreibende Eigenschaft „Postleitzahl“ ist nämlich funktional abhängig von der Kombination der Eigenschaften „Straße+Hausnummer+Ort+Kreis“, die ihrerseits wiederum funktional abhängig ist vom Schlüssel „Name“. Damit ist also die beschreibende Eigenschaft „Postleitzahl“ *transitiv funktional abhängig vom Schlüssel*, was eine Verletzung der 3. Normalform darstellt. Die „tabellarische“ Zuordnung der „Postleitzahl“ zu der Kombination „Strasse+Hausnummer+Ort+Kreis“ erfolgt im Post-

leitzahlenbuch der Deutschen Bundespost. Man könnte nun die 3. Normalform durch die Transformation herstellen, die in der Abbildung 2-47 vollzogen wurde.

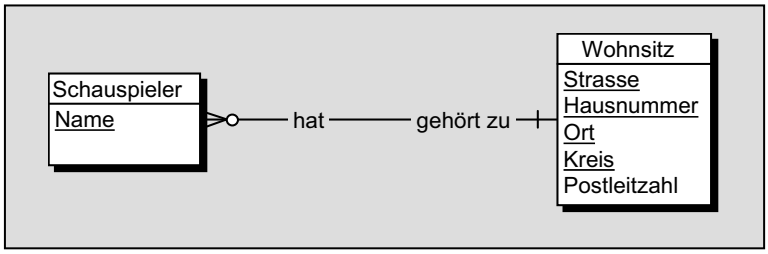


Abb. 2-47: Objekttypen „Schauspieler“ und „Wohnsitz“ in der 3. Normalform

Das Modell in der 3. Normalform setzt voraus, dass es Wohnsitze gibt, die keinem Schauspieler zugeordnet sind, dass aber ein Wohnsitz auch zu mehreren Schauspielern gehören kann (wenn sie zusammen wohnen). Der Objekttyp „Wohnsitz“ entspricht dabei dem elektronisch gespeicherten Postleitzahlenbuch. Jeder konkrete Wohnsitz wird durch die Eigenschafte-Kombination „Straße+Hausnummer+Ort+Kreis“ identifiziert. Die beschreibende Eigenschaft „Postleitzahl“ ist eigentlich nur für die internen Bearbeitungsprozesse beim Postdienst von Bedeutung. Der Vorteil dieser Speicherform wäre, dass im Falle der Änderung des Postleitzahlensystems die Zuordnung der Postleitzahl zu einem Wohnsitz nur an einer Stelle erfolgen müsste und somit Inkonsistenzen in den Daten ausgeschlossen wären. Da dies aber wohl nicht so bald geschehen wird (hoffen wir es, lieber Leser!), wiegt dieses Argument nicht allzu schwer. Außerdem kommt es sicherlich nicht oft vor, dass mehrere Schauspieler denselben Wohnsitz haben, so dass die Modellierung gemäß Abbildung 2-46 ohnehin nicht zu Datenredundanzen führen würde. Im Interesse einer höheren Performance des Informationssystems wird man an dieser Stelle die 3. Normalform zurücknehmen, also eine *Denormalisierung* durchführen.

2.7 Nutzen des konzeptionellen Datenmodells

Die Datenmodellierung mit Hilfe der Sprache des Entity-Relationship-Modells wurde als eine Technik zur präzisen und redundanzfreien Beschreibung des Informationsbedarfs für die automatisierte Informationsverarbeitung in einem abgegrenzten Gegenstandsbereich vorgestellt. Wie im Kapitel 1 ausführlich begründet wurde, kann sie insbesondere die Kenntnisübertragung zwischen den Experten der betroffenen Fachabteilung und den Anwendungsentwicklern in einer exakten, syntaktisch orientierten Weise unterstützen. Das im Fachkonzept strukturiert zusammengefasste Wissen über die Daten und über ihre sachlogischen Beziehungen kann in mannigfacher Weise Nutzen bringen. Nachfolgend werden die wichtigsten Nutzensaspekte des konzeptionellen Datenmodells kurz aufgeführt:

1. Das konzeptionelle Datenmodell entsteht im Ergebnis einer terminologischen Normierung. Im Zuge der Entwicklung des Modells werden Vorzugsbenennungen für

   - die zu *Begriffsklassen zusammengefassten Objekte* im Gegenstandsbereich (Objekttyp-Namen),
   - die *relevanten Eigenschaften* der Objekte (Benennungen der Eigenschaften) und
   - die *sachlogischen Zusammenhänge* zwischen den Objekten (Benennungen der Beziehungstyp-Richtungen)

   festgelegt. Diese terminologische Normierung stellt - auch ohne Automatisierungsvorhaben - einen eigenständigen Wert dar: sie erleichtert die Kommunikation zwischen den einzelnen Bereichen eines Unternehmens.

2. Das konzeptionelle Datenmodell beschreibt die *informations-orientierte Sicht* auf die Daten. Durch die Anwendung spezieller Generator-Programme kann diese informationsorientierte Sicht in die *daten-orientierte Sicht* – das *logische Datenschema* – der Datenbank transformiert werden. Bei dieser Transformation erfolgt die Abbildung auf die Besonderheiten des zu verwendenden Datenbank-Managementsystems. Dieser Vorgang kann heute in großen Teilen automatisiert ablaufen. Mit diesem Transformationsschritt befasst sich das Kapitel 4.

3. Die im konzeptionellen Datenmodell verwendeten Benennungen können als *Variablen-Bezeichner* in den Anwendungsprogrammen verwendet werden. Wenn an Stelle von DV-technischen Variablen-Bezeichnern die im Fachbereich üblichen Benennungen benutzt werden, entstehen Programme, die leichter zu verstehen und damit einfacher an neue Bedingungen anzupassen sind.

4. Die Benennungen des konzeptionellen Datenmodells sollten die Grundlage für die Gestaltung der *Benutzeroberfläche* des Anwendungssystems bilden. Wenn die im Unternehmen üblichen Benennungen für die Objekt- und Beziehungstypen in der Kommunikation des Anwendungssystems mit dem Benutzer (beispielsweise in Masken, Formularen und Menüs) Verwendung finden, verkleinert sich die subjektive Hemmschwelle der Mitarbeiter in der betroffenen Fachabteilung hinsichtlich der Nutzung des automatisierten Systems.

3 Die Zielstruktur: Datenbank-Modelle

Das wichtigste Charakteristikum eines konzeptionellen Datenmodells, das mit Hilfe der Sprache des Entity-Relationship-Modells erstellt wird, besteht darin, dass seine „Formulierung“ auf der *syntaktischen Ebene* erfolgt. Das bedeutet zweierlei:

1. Alle Aussagen des konzeptionellen Datenmodells liegen in Form von *strukturellen Informationen* vor, die keine Interpretation durch den Menschen mehr erfordern.
2. Die für den Aufbau eines konzeptionellen Datenmodells verwendeten Programme (CASE-Tools) bieten die Möglichkeit, das Datenmodell auf Vollständigkeit und Widerspruchsfreiheit zu prüfen. Nach Abschluss der Modellierungsaktivitäten kann man also davon ausgehen, dass alle erforderlichen Angaben *explizit*, *vollständig* und *widerspruchsfrei* vorliegen.

Somit sind alle Voraussetzungen dafür gegeben, dass das konzeptionelle Datenmodell automatisiert weiterverarbeitet werden kann. Aus dem *konzeptionellen Datenmodell*, das eine informations-orientierte – und damit vom Datenbank-Managementsystem unabhängige – Sicht auf die Datenstrukturen darstellt, kann mit Hilfe spezieller Generator-Programme die daten-orientierte Sicht für ein spezielles Datenbank-Managementsystem, nämlich das *logische Datenschema* – abgeleitet werden. Dabei sind natürlich die Möglichkeiten zu berücksichtigen, die das jeweilige Datenbank-Managementsystem für die Strukturierung der Datenbank vorsieht. Diese hängen aber in erster Linie vom *Datenbank-Modell* ab, das dem Datenbank-Managementsystem zugrunde liegt.

Dieses Kapitel beschreibt deshalb die Möglichkeiten, die die verschiedenen Datenbank-Modelle für die Repräsentation von Datenstrukturen bieten. Dabei steht das *relationale Datenbank-Modell* im Zentrum des Interesses. Die Einordnung dieses Kapitels in den Kontext des Lehrbuchs zeigt die Abbildung 3-1.

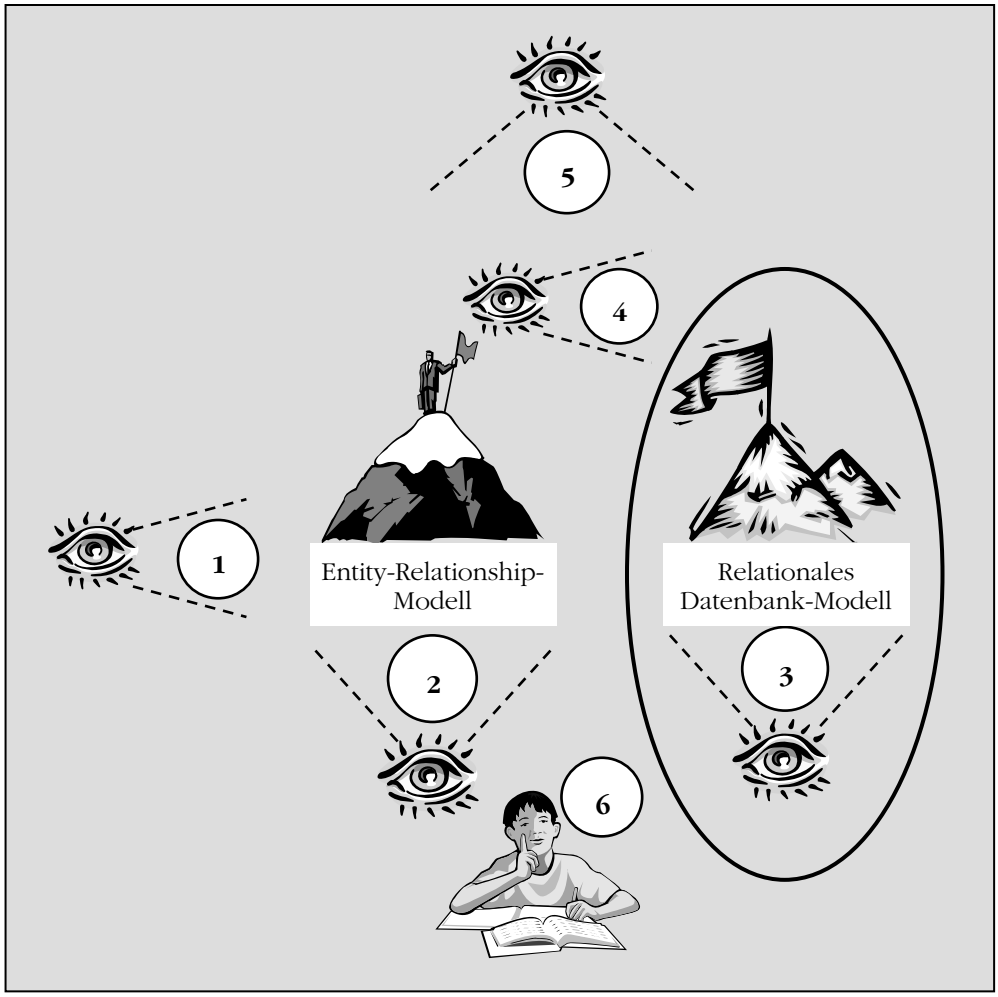


Abb. 3-1: Gegenstand des Kapitels 3

Ehe wir den eigentlichen Prozess der Transformation des konzeptionellen Datenmodells in das logische Datenschema näher beschreiben können, müssen wir zunächst die gängigen Datenbank-Modelle – und dabei in größerer Tiefe das relationale Datenbank-Modell – vorstellen. Im Kapitel 4 wird dann detailliert auf die Transformationsregeln eingegangen.

3.1 Der Begriff des Datenbank-Modells

Die Möglichkeiten, Angaben über Objekte und über ihre sachlogischen Zusammenhänge in einer Datenbank abzulegen, hängen wesentlich vom *Datenbank-Modell* ab, dem die Datenbank zuzuordnen ist:

Datenbank-Modell

Definition: Ein *Datenbank-Modell* (engl. data base model) ist ein logisches Beziehungsgebilde, durch das festgelegt wird, in welcher Weise Datensätze, durch die Objekte beschrieben werden, miteinander in Verbindung gebracht werden können.

Datensatz

Datensätze sind beispielsweise:

- Der Satz der Eigenschaftswerte („Peter“, „Silie“, „01.01.1990“, „Gartenweg 1, 12345 Grüntal“), die für einen *konkreten Schüler* von den Eigenschaften „Vorname“, „Familienname“, „Geburtsdatum“ und „Adresse“ angenommen werden.
- Der Satz der Eigenschaftswerte („Goethe-Schule“, „19999“, „Wiesenweg“, „1“), die für eine *konkrete Schule* von den Eigenschaften „Name“, „Postleitzahl“, „Straße“ und „Hausnummer“ angenommen werden.

Das Datenbank-Modell legt nun fest, welche speziellen Möglichkeiten bestehen, das logische Datenschema der Datenbank so zu gestalten, dass man beispielsweise vom Datensatz des Schülers Peter Silie zum Datensatz der Goethe-Schule, in der er unterrichtet wird, gelangen kann.

Objekttypen als „Dateninseln“

Wir kommen hier auf das anschauliche Bild zurück, das bei der Beschreibung der sachlogischen Zusammenhänge im Abschnitt 2.4 verwendet wurde. Dort wurden die Objekttypen als „Dateninseln“ bezeichnet, die durch die Beziehungstypen im Sinne von „Brücken“ miteinander verbunden werden. Diese Brücken bieten bei der Informationssuche die Möglichkeit, von einem Datensatz zu einem anderen zu „navigieren“. Das Datenbank-Modell legt nun im Wesentlichen dreierlei fest:

1. Von welchen „Dateninseln“ kann eine Wanderung über die „Brücken“ begonnen werden? Dies ist die Frage nach den möglichen *Einstiegspunkten* für die Informationssuche.
2. Gibt es *strukturelle Beschränkungen* für die „Brücken“, die die „Dateninseln“ miteinander verbinden?
3. Werden die „Brücken“ schon beim Speichern der Daten oder erst bei der „Brückenwanderung“ – also während der Informationssuche – angelegt? Dies ist die Frage nach dem *Zeitpunkt des Brückenschlags*.

In historischer Folge wurden drei Datenbank-Modelle entwickelt, die sich im Wesentlichen darin unterscheiden, wie die genannten Fragen beantwortet werden:

3 Datenbank-Modelle

- *Hierarchisches Datenbank-Modell:* Eines der bekanntesten hierarchischen Datenbank-Managementsysteme ist *IMS/DB* (***I***nformation ***M***anagement ***S***ystem / ***D***ata ***B***ase) der Firma IBM [GARV98].
- *Netzwerk-Datenbank-Modell*[12]*:* Ein bekannter Vertreter der Netzwerk-Datenbank-Managementsysteme ist *IDMS* (***I***ntegrated ***D***atabase ***M***anagement ***S***ystem) der Firma Cullinet.
- *Relationales Datenbank-Modell:* In den folgenden Beispielen wird als populärer Vertreter relationaler Datenbank-Managementsysteme das DBMS *„Access“* der Firma Microsoft verwendet [ACCE03].

Nachdem sich das relationale Datenbank-Modell zum Marktstandard entwickelt hat, kommt den beiden erstgenannten Datenbank-Modellen heute vielerorts eine eher historische Bedeutung zu (mitunter werden sie geradezu als „Altlasten“ betrachtet). Aus diesem Grunde wird nur das relationale Datenbank-Modell in größerer Tiefe dargestellt - und auch nur so weit, wie es für das Verständnis des betrachteten Transformationsprozesses erforderlich ist. Weitergehende Informationen zu den drei Datenbank-Modellen findet der interessierte Leser in der reichlich vorhandenen Datenbank-Literatur (beispielsweise in [KEMP09]).

[12] Dieses Datenbank-Modell wird auch als „netzwerk*artiges*“ Datenbank-Modell bezeichnet.

3.2 Das hierarchische Datenbank-Modell

Das hierarchische Datenbank-Modell ordnet sich wie folgt in den Fragenkatalog des Abschnitts 3.1 ein:

1. *Einstiegspunkte:* In der Datenbank kann nur ein einziger Objekttyp als Einstiegspunkt verwendet werden. Jede Informationssuche muss von diesem einen Einstiegspunkt ausgehen.

2. *Strukturelle Beschränkungen:* Die Objekttypen, die die Dateninseln bilden, können ausschließlich in monohierarchischer Form miteinander verbunden werden. Ein übergeordneter Objekttyp (owner, master) kann mit mehreren untergeordneten Objekttypen (member, detail) durch Brücken verbunden werden. Ein Member-Objekttyp darf nur mit einem Owner-Objekttyp verbunden sein. Als Brückenstruktur ist somit nur eine Baumstruktur zulässig, bei der die Wurzel den Einstiegspunkt für die Informationssuche bildet. Die Brücken sind als „Einbahnstraßen" ausgelegt: sie können nur von der Wurzel aus „abwärts" überquert werden. Eine Brücke bildet somit einen dualen 1:CN-Beziehungstyp ab, bei dem die 1-Seite in Richtung der Wurzel weist.

3. *Zeitpunkt des Brückenschlags:* Die Brücken zwischen den Dateninseln werden bereits zum Zeitpunkt der Speicherung angelegt. Sie sind dann - wie man sagt - „fest verdrahtet".

Beispiel für das hierarchische Datenbank-Modell

Die Abbildung 3-2 zeigt einen Ausschnitt aus dem konzeptionellen Datenmodell für das Schulbeispiel, der sich als Baumstruktur in ein logisches Datenschema des hierarchischen Datenbank-Modells abbilden lässt:

Die Dreiecke symbolisieren mit ihren auseinanderstrebenden Schenkeln den 1:CN-Beziehungstyp, die eingeschlossenen Pfeile geben die Richtung der „Einbahnstraße" an. Als einziger Einstiegspunkt für die Informationssuche dient die Wurzel des Baums, also der Objekttyp „Schule". Vom Datensatz einer bestimmten Schule – beispielsweise der Goethe-Schule – kann man über die entsprechenden Brücken zu den Datensätzen aller ihrer Unterrichtsräume bzw. aller ihrer Klassen gelangen. Vom Datensatz einer dieser Klassen ist die Fortsetzung der Brückenwanderung zu den Datensätzen aller Schüler dieser Klasse möglich. Die folgenden Aufgaben der Informationssuche sind beispielsweise jedoch nicht lösbar:

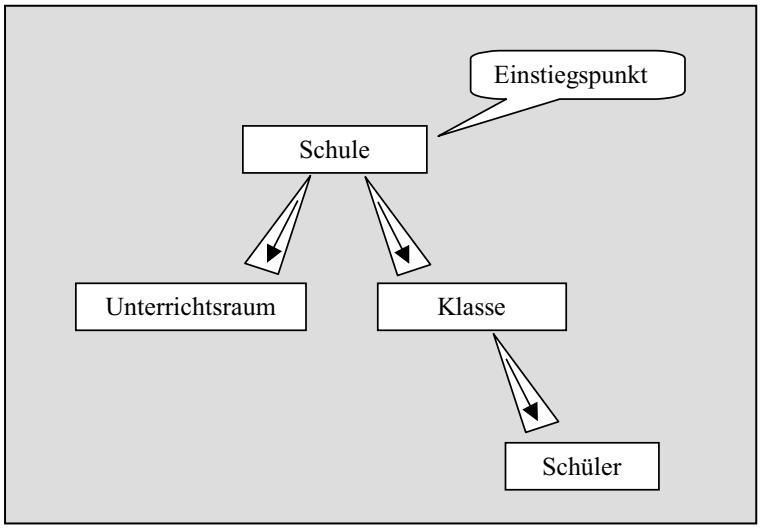


Abb. 3-2: Hierarchisches Datenbank-Modell

Grenzen des hierarchischen Datenbank-Modells

- Die Frage nach den Daten des Schülers Peter Silie. Der Objekttyp „Schüler“ ist nicht die Wurzel des Baums, er ist somit kein möglicher Einstiegspunkt für die Informationssuche. Er kann nur durch Navigation vom Objekttyp „Schule“ aus über den Objekttyp „Klasse“ erreicht werden.
- Die Frage, zu welcher Klasse der Schüler Peter Silie gehört. Da die Brücke zwischen den Objekttypen „Klasse“ und „Schüler“ eine Einbahnstraße in Richtung „Schüler“ ist, kann man vom Objekttyp „Schüler“ nicht zum Objekttyp „Klasse“ gelangen.
- Die Frage, von welchen Klassen der Unterrichtsraum 205 der Goethe-Schule als Klassenraum genutzt wird. Der Objekttyp „Klasse“ ist kein Member-Objekttyp zum Objekttyp „Unterrichtsraum“. Somit gibt es keine Navigationsmöglichkeit vom Unterrichtsraum zu den Klassen.

Die aufgezeigten Mängel lassen sich im hierarchischen Datenbank-Modell nur dadurch beheben, dass man unabhängig von der gezeigten Baumstruktur weitere Baumstrukturen[13] anlegt, in denen man Kopien der jeweils benötigten Datensätze speichert.

[13] Eine solche Menge von Bäumen bezeichnet man als „Wald“.

Die dadurch entstehenden Redundanzen führen zu den ernsthaften Problemen, die bereits im Abschnitt 2.4.2 diskutiert wurden. Eine andere Möglichkeit zur Lösung des Problems besteht in der Verwendung von Zeigerstrukturen, auf die hier aber nicht näher eingegangen wird.

Monohierarchien

Mit hierarchischen Datenbank-Modellen lassen sich also lediglich monohierarchische Beziehungstypen effizient repräsentieren. In einer Vielzahl von Anwendungsfällen reicht das aus, weshalb sich Datenbank-Anwendungen, die dem hierarchischen Modell verpflichtet sind, auch heute noch im Einsatz befinden. Beispiele dafür sind:

- *Stücklistenauflösungen* in produzierenden Unternehmen,
- *Zugriffe auf Verträge* durch Navigation, ausgehend von den Kundendaten.

Will man jedoch kompliziertere Zusammenhänge zwischen den Objekttypen (etwa CM:CN-Beziehungstypen oder Rekursiv-Beziehungstypen) repräsentieren, stößt man auf die Grenzen des hierarchischen Datenbank-Modells.

3.3 Das Netzwerk-Datenbank-Modell

Durch das Netzwerk-Datenbank-Modell werden einige Schwächen des hierarchischen Datenbank-Modells gemildert. Beantwortet man den Fragenkatalog des Abschnitts 3.1 für das Netzwerk-Datenbank-Modell, so ergibt sich das folgende Bild:

1. *Einstiegspunkte:* Prinzipiell kann jeder Objekttyp als Einstiegspunkt verwendet werden. Allerdings muss diese Festlegung beim Entwurf des logischen Datenschemas erfolgen. Objekttypen, die beim Strukturentwurf nicht als Einstiegspunkte ausgewiesen wurden, können später bei der Informationssuche nicht als Ausgangspunkt für die Navigation verwendet werden. Die in solchen Objekttypen enthaltenen Datensätze können dann lediglich durch Navigation, ausgehend von einem Einstiegspunkt, erreicht werden.
2. *Strukturelle Beschränkungen:* Die einzelnen Objekttypen können durch ein beliebig strukturiertes Netzwerk miteinander verbunden werden. Die Brücken dieses Netzwerkes bilden jeweils einen 1:CN-Beziehungstyp ab, können aber in beiden Richtungen überquert werden.

3. *Zeitpunkt des Brückenschlags:* Die Brücken zwischen den Dateninseln werden bereits zum Zeitpunkt der Speicherung angelegt.

Beispiel für das Netzwerk-Datenbank-Modell

Im logischen Datenschema des Netzwerk-Datenbank-Modells lässt sich das Datenmodell des Schulbeispiels leichter abbilden, weil es nicht mehr erforderlich ist, das Beziehungsgeflecht der Objekttypen in eine Baumstruktur zu „verbiegen". Das zeigt der Ausschnitt des Datenmodells, der in Abbildung 3-3 dargestellt ist.

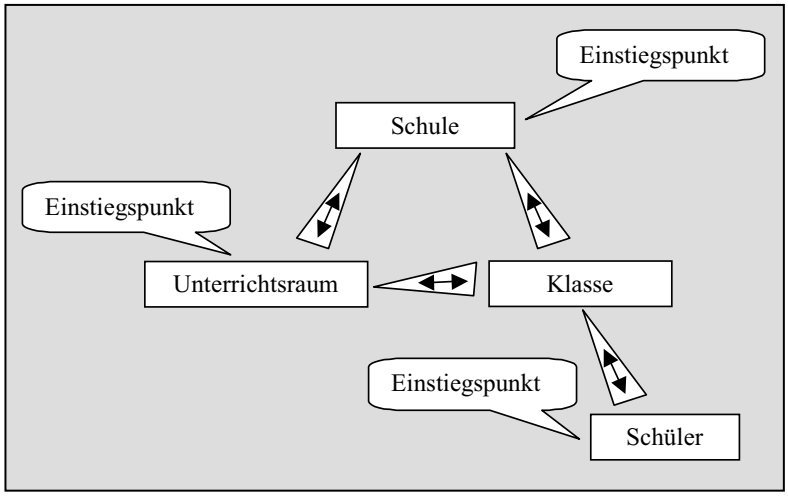


Abb. 3-3: Netzwerk-Datenbank-Modell

Die Struktur weicht von der Baumstruktur der Abbildung 3-2 ab. Beispielhaft wurden drei Einstiegspunkte definiert. Die „Brücken" zwischen den Objekttypen können nun in beiden Richtungen überquert werden. Ebenso wie beim hierarchischen Datenbank-Modell kann man vom Datensatz einer Schule über die entsprechenden Brücken zu den Datensätzen aller ihrer Unterrichtsräume bzw. aller ihrer Klassen gelangen. Zu einer Klasse kann man nun aber auch durch Navigation über einen Unterrichtsraum gelangen. Vom Datensatz einer Klasse ist die Fortsetzung der Brückenwanderung zu den Datensätzen aller Schüler dieser Klasse möglich. Die im hierarchischen Datenbank-Modell nicht lösbaren Beispiel-Aufgaben der Informationssuche sind nun lösbar:

- Die Frage nach den Daten des Schülers Peter Silie. Der Objekttyp „Schüler“ ist jetzt ein möglicher Einstiegspunkt für die Informationssuche. Somit kann direkt auf seine Datensätze zugegriffen werden.
- Die Frage, zu welcher Klasse der Schüler Peter Silie gehört. Da die Brücke zwischen den Objekttypen „Klasse“ und „Schüler“ in beiden Richtungen überquert werden kann, kann man vom Schüler zu seiner Klasse gelangen.
- Die Frage, von welchen Klassen der Unterrichtsraum 205 der Goethe-Schule als Klassenraum genutzt wird. Da nun der Objekttyp „Unterrichtsraum“ ein möglicher Einstiegspunkt ist, kann die Informationssuche beim Unterrichtsraum 205 der Goethe-Schule begonnen werden. Über die Brücke zwischen den Objekttypen „Unterrichtsraum“ und „Klasse“ gelangt man zu allen Klassen, die diesen Unterrichtsraum als Klassenraum nutzen.

Vorteile des Netzwerk-Datenbank-Modells

Das Netzwerk-Datenbank-Modell beruht auf einem Vorschlag der *CODASYL* Data Base Task Group [CODA71] bzw. auf dessen Weiterentwicklung [CODA73]. Gegenüber dem hierarchischen Datenbank-Modell besteht der Vorteil des Netzwerk-Datenbank-Modells darin, dass im Prinzip jeder beliebige Objekttyp als Einstiegspunkt der Informationssuche genutzt werden kann, von dem aus durch Navigieren ein weiterer Zugriff auf mit diesem Objekttyp sachlogisch verbundene Objekttypen möglich ist. Ein weiterer Vorteil besteht in der Möglichkeit, multiple Eigenschaften und Wiederholungsgruppen von Eigenschaften repräsentieren zu können. Die im Abschnitt 2.7.1 beschriebene 1. Normalform ist für das Netzwerk-Datenbank-Modell nicht erforderlich[14]. Geblieben ist jedoch der Nachteil, dass nur 1:CN-Beziehungstypen in natürlicher Weise abgebildet werden können. Will man CM:CN-Beziehungstypen oder Rekursiv-Beziehungstypen darstellen, so ist man auf die Einführung eines technisch bedingten Objekttyps (Koppel-Objekttyp) angewiesen. Das dazu erforderliche Vorgehen wird im Rahmen der Darstellung des relationalen Datenbank-Modells im Abschnitt 3.4.3 beschrieben.

[14] Dieser Fortschritt wird beim relationalen Datenbank-Modell wieder aufgegeben.

3.4 Das relationale Datenbank-Modell

Das relationale Datenbank-Modell hat hinsichtlich der Möglichkeiten zur Navigation zwischen den Objekttypen eine etwas größere Mächtigkeit als das Netzwerk-Datenbank-Modell. Es unterscheidet sich von diesem aber prinzipiell durch den Zeitpunkt, zu dem die Brücken zwischen den Objekttypen geschlagen werden. Der Fragenkatalog des Abschnitts 3.1 wird für das relationale Datenbank-Modell in folgender Weise beantwortet:

1. *Einstiegspunkte:* Jeder Objekttyp ist – ohne besondere Deklarationsmaßnahmen - ein möglicher Einstiegspunkt für die Informationssuche.
2. *Strukturelle Beschränkungen:* Die einzelnen Objekttypen können durch ein beliebig strukturiertes Netzwerk miteinander verbunden werden. Die Brücken dieses Netzwerks bilden unterschiedliche Beziehungstypen[15] ab. Sie können in beiden Richtungen überquert werden.
3. *Zeitpunkt des Brückenschlags:* Die Brücken zwischen den Dateninseln werden nicht zum Zeitpunkt der Speicherung, sondern erst zum Zeitpunkt der Informationssuche angelegt.

Dynamisch erstellte „Brücken“

Die Besonderheit des relationalen Datenbank-Modells gegenüber dem hierarchischen und dem Netzwerk-Datenbank-Modell besteht also darin, dass es in der Speicherstruktur keinerlei Verbindungen zwischen den Datensätzen gibt. Wurden die Datensätze auf den Dateninseln gespeichert, dann existieren zwischen den Dateninseln keine Brücken. Die Brücken werden erst dann dynamisch erstellt, wenn es erforderlich ist, von einer Dateninsel zu einer anderen zu gelangen, also erst während der Informationssuche.

Stärke und Schwäche des relationalen Datenbank-Modells

In diesem Grundprinzip des relationalen Datenbank-Modells liegt zugleich seine Stärke und seine Schwäche. Diese machen sich insbesondere dann bemerkbar, wenn kompliziert strukturierte Objekte bzw. Sachverhalte gespeichert werden sollen. Eine Analogie soll dies verdeutlichen. Als kompliziert strukturiertes Objekt betrachten wir ein Auto, das gespeichert – das heißt im realen Leben: in der Garage abgestellt – werden soll. Nach den Prinzipien des relationalen Datenbank-Modells wäre es erforderlich, das Auto dazu in seine sämtlichen Bestandteile zu zerlegen. An

[15] Welche Beziehungstypen im relationalen Datenbank-Modell repräsentiert werden können, wird im Kapitel 5 ausführlich untersucht.

den Wänden der Garage befinden sich Regale, die die Blechteile, Schrauben, Laschen, Muttern usw. aufnehmen. Der Aufgabe der Informationssuche entspricht das erneute Zusammensetzen der Teile zu einem funktionsfähigen Auto. Damit das überhaupt gelingen kann, muss man sich merken, welche der nun verstreut in den Regalen liegenden Teile mit welchen anderen Teilen verbunden waren. Die durch die Demontage „gekappten" Zusammenhänge zwischen den Teilen müssen durch zusätzlich an den Teilen angebrachte Informationen aufbewahrt werden. Beispielsweise müsste man an die beiden Enden einer Lasche, die ehemals die Teile A und B miteinander verbunden hat, kleine Zettel anbringen, auf denen jeweils eine Identifikation des Teils A bzw. des Teils B steht. Bei der erneuten Montage lässt sich dann durch die Auswertung dieser Zettel der ursprüngliche Zusammenhang der Teile rekonstruieren. Die Abbildung 3-4 soll diese Situation verdeutlichen.

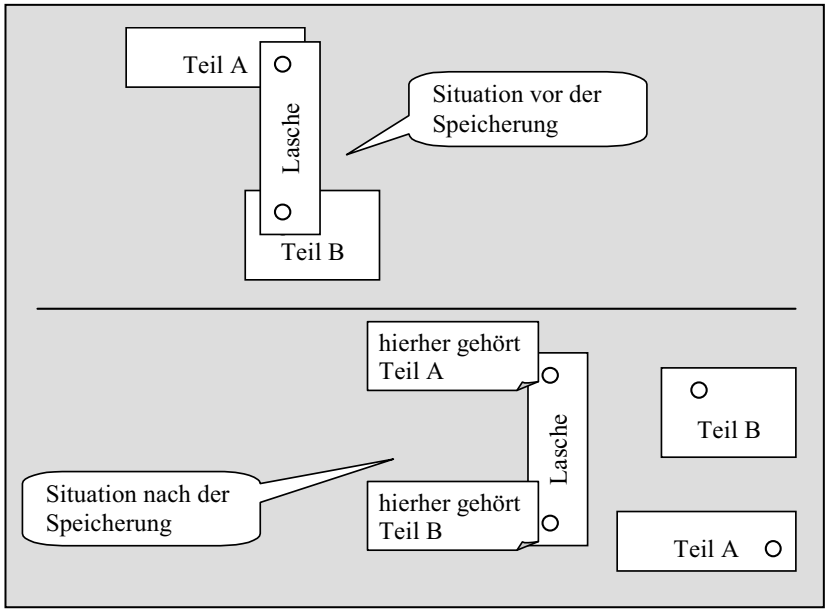


Abb. 3-4: Repräsentation von sachlogischen Zusammenhängen im relationalen Datenbank-Modell

Die Stärke sowie die Schwäche dieser Form der Speicherung liegen nun auf der Hand:

- *Stärke:* Da die einzelnen Teile nicht miteinander „fest verdrahtet“ sind, ist es möglich, sie bei Bedarf in anderer Weise als vor der Speicherung zusammenzusetzen. Wenn es die Gesamtmenge der Autoteile zulässt, könnte man daraus heute ein Kabriolett und morgen eine Limousine zusammenstellen. Eventuell könnte man auch das Lenkrad einmal auf der linken und ein anderes Mal auf der rechten Seite montieren. Auf die Informationssuche übertragen, heißt das: Man kann die Daten aus den Datensätzen in einer Weise miteinander verbinden, dass Zusammenhänge dargestellt werden, von denen bei der Speicherung niemand gedacht hätte, dass sie einmal von Bedeutung sein könnten. Derartige „Ad-hoc-Anfragen“, die einen Informationsbedarf befriedigen, der über die operationale Routinearbeit hinausgeht, gewinnen in unserer schnell-lebigen Zeit, in der die Unternehmen zunehmend dem Zwang unterliegen, sich rasch an neue Marktbedingungen anzupassen, immer stärker an Bedeutung. Solche Ad-hoc-Anfragen lassen sich mit dem hierarchischen Datenbank-Modell gar nicht und mit dem Netzwerk-Datenbank-Modell nur in eingeschränkter Weise beantworten.
- *Schwäche:* Das erneute Zusammenfügen der auseinandergerissenen Einzelteile ist mit zusätzlichem Aufwand verbunden. Man muss im Falle der oben angeführten Lasche erst die Angaben auf den beiden Zetteln auswerten und dann in den Regalen der Garage nach den Teilen A bzw. B suchen, ehe man sie wieder zusammensetzen kann. Bei der Informationssuche müssen die verstreut gespeicherten Datensätze, die im jeweils gefragten sachlogischen Zusammenhang stehen, erst durch Suchprozesse wieder zusammengeführt werden. Bei „fest verdrahteten“ Verbindungen, wie sie das hierarchische und das Netzwerk-Datenbank-Modell verwenden, ist das Zusammenfügen der benötigten Datensätze immer dann wesentlich einfacher, wenn eine Verbindung zwischen ihnen beim Entwurf des internen Schemas vorgesehen wurde.

Trotz der Schwächen des relationalen Datenbank-Modells hinsichtlich seines Laufzeitverhaltens bei Anfragen, die die Verknüpfung einer großen Anzahl von Datensätzen erfordern, hat sich dieses Datenbank-Modell allgemein durchgesetzt. Das liegt daran, dass sich für dieses Datenbank-Modell – wie wir im Folgenden sehen werden - eine anschauliche und aus dem Alltagsleben gewohnte Darstellungsform, nämlich die *Tabelle*, anbietet.

3.4.1 Grundprinzipien des relationalen Datenbank-Modells

Das relationale Datenbank-Modell wurde bereits 1970 von Codd vorgeschlagen [CODD70] und von ihm danach ständig weiterentwickelt. Die Stärke dieses Modells liegt hauptsächlich in seiner exakten theoretischen Fundierung. Legt man die von Codd formulierten strengen Anforderungen an ein relationales Datenbank-Managementsystem zugrunde, dann sind alle praktischen Implementationen nur mehr oder weniger „relational". Gab es 1986 lediglich die berühmten 12 Codd'schen Regeln [CODD86] für ein relationales Datenbank-Managementsystem, so wurden diese bis 1990 auf 333 Eigenschaften erweitert [CODD90].

Es folgt eine kurze Beschreibung des relationalen Datenbank-Modells, bei der wir uns auf die Möglichkeiten zur *Strukturierung der Daten* konzentrieren, während auf die Informationssuche nur am Rande eingegangen wird.

Typbeschreibung

Den Ausgangspunkt des relationalen Datenbank-Modells bildet der Begriff des *Attributs* A, durch das eine wesentliche Eigenschaft eines *Objekttyps* T zum Ausdruck gebracht wird. Für die Dokumentation der Datenstruktur werden beiden Gestaltungselementen *Bezeichnungen* zugeordnet: *Bez*[A] bzw. *Bez*[T]. Die Dokumentation der Semantik eines Objekttyps T erfolgt in einer *Typbeschreibung* dadurch, dass man hinter der Bezeichnung des Objekttyps in einem Klammerausdruck die Liste der Bezeichnungen seiner relevanten Attribute A_1, A_2, ..., A_n angibt:

Typbeschreibung: *Bez*[T] (*Bez*[A_1], *Bez*[A_2], ..., *Bez*[A_n])

Der Objekttyp T mit *Bez*[T] = „Lehrer" in unserem Schulbeispiel wird durch fünf Attribute A_1, A_2, A_3, A_4 und A_5 beschrieben mit:

Bez[A_1] = „Personalnummer"
Bez[A_2] = „Vorname"
Bez[A_3] = „Familienname"
Bez[A_4] = „Geburtsdatum"
Bez[A_5] = „Familienstand"

Die Typbeschreibung für den Objekttyp „Lehrer" hat die Form:

Lehrer(Personalnummer,Vorname,Familienname,
Geburtsdatum,Familienstand)

Wertebereich (Domäne)

Für jedes Attribut A_i lässt sich sein *Wertebereich* - die sogenannte *Domäne* – in Form einer Menge angeben:

$$\textit{Wertebereich}: \quad \textit{Dom}[A_i] = \{ a_{i1}, a_{i2}, ..., a_{in} \}$$

Der Wertebereich $\textit{Dom}[A_i]$ enthält diejenigen Attributwerte, die das Attribut A_i im Falle eines konkreten Objekts als Ausprägung annehmen kann. Eine wesentliche Beschränkung des relationalen Datenbank-Modells besteht darin, dass als Attributwerte eines Wertebereichs nur *atomare Werte* zugelassen sind, die in sich nicht strukturiert sind. Insbesondere ist als Attributwert keine Menge von Werten zugelassen. Die Daten müssen sich also in der 1. Normalform befinden, bei der es keine multiplen Eigenschaften gibt (vgl. Abschnitt 2.7.1).

Element des Wertebereichs

Um auszudrücken, dass ein Wert a_{ik} Element des Wertebereichs des Attributs A_i ist, schreibt man:

$$\textit{Element des Wertebereichs}: \quad a_{ik} \in \textit{Dom}[A_i]$$

NULL-Marke

Ist für ein Attribut A nicht bei jedem konkreten Objekt ein Wert angebbar, weil er für das Objekt irrelevant ist (beispielsweise der Wert des Attributs „Anzahl der Geburten“ für Männer) oder weil er bei der Dateneingabe einfach noch nicht bekannt ist (beispielsweise der Wert des Attributs „Abgabedatum“ zum Zeitpunkt der Speicherung einer gerade erst vergebenen Diplomarbeit), so muss das Attribut mit einer speziellen Markierung - der sogenannte „NULL-Marke“ [16] - belegt werden.

Diese *NULL-Marke* ist strikt von der Zahl „0“ und von der Buchstabenfolge „NULL“ zu unterscheiden. Die Bezeichnung NULL-*„Marke“* – anstelle von NULL-*„Wert“* - soll ausdrücklich darauf hinweisen, dass es sich dabei nicht um einen *Wert*, sondern um die Angabe handelt, dass das Attribut A für ein Objekt eben *keinen Wert* annimmt.

In einigen Fällen ist die Domäne eines Attributs *extensional*, also durch Aufzählung ihrer Elemente, angebbar. Das gilt beispielsweise im Falle des Familienstands:

[16] NULL (gesprochen: „nall“) bedeutet im Englischen „wertlos“ bzw. „leer“.

Dom[Familienstand] = { „ledig“, „verheiratet“,
„geschieden“, „verwitwet“}

In anderen Fällen ist sie in praktikabler Weise nur *intensional* zu beschreiben, also durch die Formulierung einer Bedingung, der ihre Elemente genügen müssen. Dies trifft beispielsweise auf die Personalnummer zu. Lässt man eine fünfstellige Personalnummer zu, dann gilt:

$$Dom[\text{Personalnummer}] = \{ n \mid 0 < n < 100000 \}$$

n-Tupel

Für ein konkretes Objekt, das zu einem Objekttyp T mit der Typbeschreibung *Bez*[T] (*Bez*[A_1], *Bez*[A_2], ..., *Bez*[A_n]) gehört, kann jedes der Attribute A_i mit i = 1, 2, ..., n einen Wert a_i aus der jeweiligen Domäne *Dom*[A_i] annehmen. Die Folge dieser Werte wird in runde Klammern eingeschlossen und als *n-Tupel* bezeichnet:

$$n\text{-Tupel: } (a_1, a_2, ..., a_n) \text{ mit } a_i \in Dom[A_i] \text{ für } i = 1, 2, ..., n$$

Für ein Objekt kann im Allgemeinen jedes Attribut unabhängig von den anderen Attributen einen Wert aus seiner jeweiligen Domäne annehmen. Dies wird mathematisch durch die *Produktmenge* ausgedrückt. Betrachtet man zunächst nur zwei Mengen A und B, dann gilt:

Produktmenge

Definition: Unter der *Produktmenge* zweier Mengen A und B versteht man die Menge aller geordneten Paare (2-Tupel) aus je einem Element der Menge A und der Menge B. Man schreibt für die Produktmenge aus A und B:

$$A \times B = \{ (a,b) \mid a \in A \text{ und } b \in B \}$$

Betrachtet man nur die beiden Attribute „Personalnummer“ und „Familienstand“, so ist die Produktmenge der Wertebereiche *Dom*[Familienstand] und *Dom*[Personalnummer] durch die Menge der 2-Tupel gegeben, die sämtliche möglichen Kombinationen von Attributwerten aus den beiden Wertebereichen enthält:

Dom[Familienstand] × *Dom*[Personalnummer] =

{ (ledig,1), (verheiratet,1), (geschieden,1), (verwitwet,1),
(ledig,2), (verheiratet,2), (geschieden,2), (verwitwet,2),
(ledig,3), (verheiratet,3), (geschieden,3), (verwitwet,3),
.....
(ledig,99999), (verheiratet,99999), (geschieden,99999),
(verwitwet,99999)
}

Wenn man die Situation auf n Mengen A_1, A_2, ..., A_n verallgemeinert, dann ist ihre Produktmenge definiert als:

$$A_1 \times A_2 \times ... \times A_n = \{ (a_1, a_2, ..., a_n) \mid a_i \in A_i \text{ für } i = 1, 2, ..., n \}$$

Die Menge der n-Tupel der theoretisch möglichen Objekte des Objekttyps „Lehrer“ wird durch die Produktmenge der Wertebereiche der Attribute dieses Objekttyps gebildet:

Menge der möglichen Lehrer-Objekte:

Dom[Personalnummer] × *Dom*[Vorname] ×
Dom[Familienname] × *Dom*[Geburtsdatum] ×
Dom[Familienstand]

Die Menge der tatsächlich zu speichernden n-Tupel für die Objekte vom Objekttyp „Lehrer“ ist natürlich wesentlich kleiner als die theoretisch mögliche Menge. An dieser Stelle kommt der mathematische Begriff der *Teilmenge* zum Tragen:

Teilmenge

Definition: Eine Menge A heißt *Teilmenge* der Menge B, wenn jedes Element von A auch Element von B ist. Man schreibt diesen Sachverhalt wie folgt:

$A \subseteq B$

Daraus folgt, dass jede Menge zu sich selbst Teilmenge ist.

Wir können nun den Begriff der *Relation* einführen, der dem relationalen Datenbank-Modell seinen Namen gibt:

Relation

Definition: Eine Teilmenge

$$\mathbf{R \subseteq Dom(A_1) \times Dom(A_2) \times ... \times Dom(A_n)}$$

heißt (n-stellige) *Relation* über den Wertebereichen der Attribute A_1, A_2, ..., A_n. Dabei ist n der *Grad* (engl. degree) der Relation.

Eine Relation ist also eine Teilmenge der möglichen n-Tupel, die unter Beachtung der Wertebereiche der Attribute eines Objekttyps gebildet werden können. Das bedeutet aber, dass eine Relation die Menge der Objekte beschreibt, die für einen gegebenen Objekttyp gespeichert werden sollen.

Die besondere Attraktivität des relationalen Datenbank-Modells besteht nun darin, dass die Kombination

Typbeschreibung	*Bez*[T]	(***Bez*[A_1],**	***Bez*[A_2],**	**...,**	***Bez*[A_n])**
+					
Relation	R ⊆	*Dom*(A_1) ×	*Dom*(A_2) ×	...×	*Dom*(A_n)

in zwangloser Weise als Tabelle repräsentiert werden kann. Die Abbildung 3-5 zeigt das für drei Objekte des Objekttyps „Lehrer“.

Stellt man Typbeschreibung und Relation eines Objekttyps in Form einer Tabelle dar, so werden aus den Bezeichnungen der Attribute die Spaltenbenennungen und aus den Attributwerten die Spaltenwerte. Jedes Element – also jedes n-Tupel - der Relation entspricht einer Zeile der Tabelle. Da es sich bei der Relation um eine Menge handelt und die Tabelle lediglich deren graphische Darstellung ist, gelten die folgenden Regeln:

- Die Reihenfolge der Elemente einer Menge ist nicht definiert, also ist die *Reihenfolge der Tabellen-Zeilen beliebig.*
- In einer Menge darf kein Element doppelt auftreten, also gibt es in der Tabelle *keine übereinstimmenden Zeilen.*

- In einer Relation $R \subseteq Dom(A_1) \times Dom(A_2) \times ... \times Dom(A_n)$ eines Objekttyps ist die Reihenfolge der Wertebereiche der Attribute prinzipiell nicht von Bedeutung, also ist die *Reihenfolge der Tabellen-Spalten beliebig*. Sie muss lediglich mit der Reihenfolge der Attributbezeichnungen in der Typbeschreibung übereinstimmen. Der Zugriff auf die einzelnen Tabellen-Werte in einer Spalte erfolgt nicht über die Spaltennummer, sondern ausschließlich über die Bezeichnung des der Spalte zugeordneten Attributs. Das impliziert die Forderung, dass die einzelnen Attribut-Bezeichnungen für eine Tabelle paarweise verschieden sein müssen.
- Die Relation R als eine Menge von n-Tupeln kann *mit der Zeit variieren*. Es können neue Elemente zu R hinzugefügt werden. Bereits in R enthaltene Elemente können in ihren Komponenten verändert werden. Elemente können aus der Relation R entfernt werden.

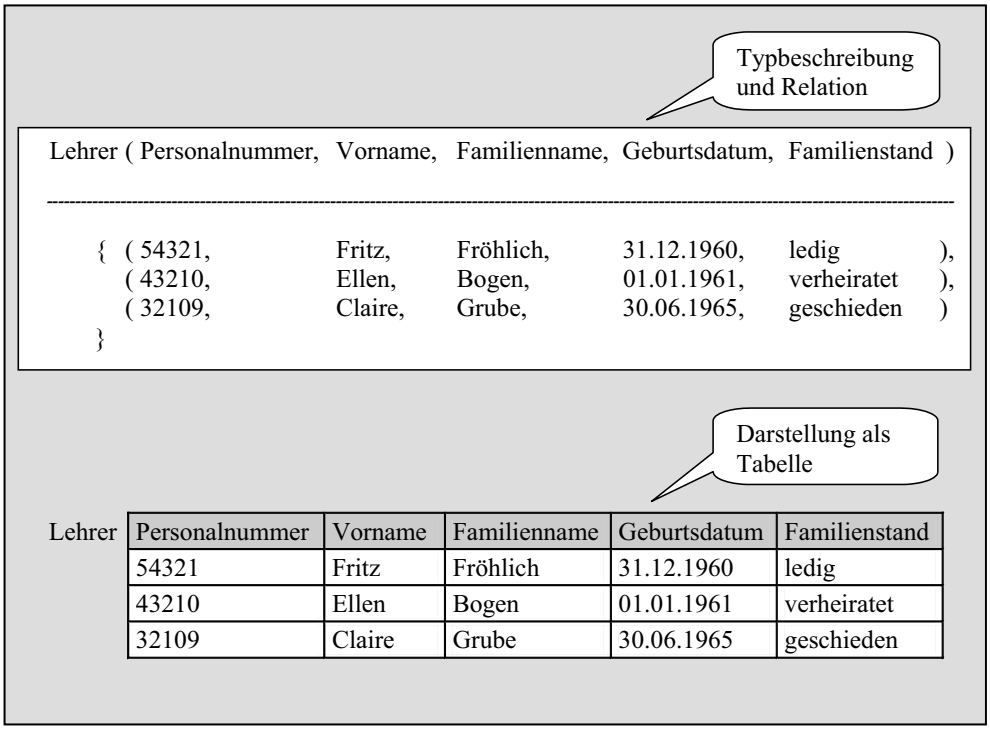


Abb. 3-5: Repräsentation von Typbeschreibung und Relation als Tabelle

Im Folgenden werden wir nicht mehr von einer Relation im mathematischen Sinne sprechen, sondern für die Darlegungen ausschließlich die anschauliche Tabellen-Interpretation verwenden.

Zugriff auf einen Datensatz

Im relationalen Datenbank-Modell erfolgt der Zugriff auf einen Datensatz - also auf eine Zeile einer Tabelle - nicht durch eine hardwarenahe Adressierung auf der Ebene der physischen Speicherung der Daten - also etwa durch Adressverweise (Pointer) - wie beim hierarchischen und beim Netzwerk-Datenmodell. Der Zugriff auf eine Tabellenzeile muss ausschließlich auf der logischen (Tabellen-)Ebene erfolgen. In einer Tabelle ist eine Zeile aber nur über die speziellen Werte ansprechbar, die die Attribute in dieser Zeile annehmen. Für die Identifizierung einer Zeile ist folglich ein *Schlüssel* festzulegen, so wie das beim Entity-Relationship-Modell bereits besprochen wurde (vgl. Abschnitte 2.3 und 2.4.4). Der Begriff des Schlüssels leitet sich aus dem bildlichen Vergleich mit dem Begriffspaar „Schlüssel und Schloss“ ab: Will man auf die Daten einer Tabellenzeile zugreifen, muss man zunächst das „Schloss“ dieser Zeile öffnen. Dazu benötigt man genau denjenigen „Schlüssel“, der zu diesem „Schloss“ passt.

Für jede Tabelle muss also ein Attribut (bzw. eine Kombination mehrerer Attribute) so gefunden werden, dass seine Attributwerte (bzw. die Kombination seiner Attributwerte) für kein Paar von Tabellenzeilen übereinstimmt:

Schlüssel

Definition: Ein *einfacher Schlüssel* (engl. key) einer Tabelle ist ein Attribut und ein *zusammengesetzter Schlüssel* ist eine Attribute-Kombination mit der Eigenschaft, dass jede Zeile der Tabelle eindeutig durch die Angabe des Wertes dieses Attributs (bzw. der Werte-Kombination dieser Attribute) zu identifizieren ist.

Unikalität des Schlüssels

Ein gegebener Attributwert eines einfachen Schlüssels (bzw. eine gegebene Kombination der Attributwerte eines zusammengesetzten Schlüssels) darf also in einer Tabelle nur ein einziges Mal auftreten, muss also *unikal* sein. In der Typbeschreibung der Tabelle wird dies typographisch dadurch kenntlich gemacht, dass das entsprechende Attribut (bzw. die Attribute-Kombination) im **Fettdruck** angegeben wird.

Unikales Attribut

unikales Attribut: ***Bez*[A]**

unikale Attribute-Kombination: $\boldsymbol{Bez}[A_1]+\boldsymbol{Bez}[A_2]+ \ldots +\boldsymbol{Bez}[A_n]$

Durch die Verwendung von *Pluszeichen* anstelle von Kommas wird typographisch deutlich gemacht, dass nicht jedes Attribut *für sich* unikal ist, sondern lediglich die *Attribute-Kombination.*

Mehrere Schlüssel für eine Tabelle

Für eine Tabelle kann es mehrere Schlüssel geben. Es gibt für eine Tabelle aber immer mindestens einen Schlüssel. Im Extremfall muss er aus der Kombination sämtlicher Attribute der Tabelle bestehen. Diese Kombination ermöglicht in jedem Fall eine eindeutige Identifizierung, weil es definitionsgemäß keine identischen Tabellenzeilen geben kann.

Im Falle des Objekttyps „Lehrer" gilt für die Tabelle als einfacher Schlüssel das Attribut „Personalnummer". Das Attribut „Familienname" ist allein im Allgemeinen kein möglicher Schlüssel, denn die Landesschulbehörde wird sicherlich mehrere Lehrer mit demselben Familiennamen zu verwalten haben. Im Normalfall lässt sich jedoch die Kombination „Vorname+Familienname+Geburtsdatum" als zusammengesetzter Schlüssel verwenden. Die beiden Schlüssel können also wie folgt dargestellt werden:

1) Lehrer(**Personalnummer**,Vorname,Familienname, Geburtsdatum,Familienstand)

2) Lehrer(Personalnummer, **Vorname+Familienname+Geburtsdatum**, Familienstand)

Forderungen an einen Schlüssel

Aus praktischen Gründen stellt man an einen Schlüssel zusätzliche Forderungen. Da man für den Zugriff auf eine Tabellenzeile den Wert bzw. die Werte-Kombination eines Schlüssels angeben - also zumeist eintippen - muss, sollte der Schlüssel die folgenden Eigenschaften haben:

- *Minimalität:* Ein zusammengesetzter Schlüssel sollte stets so gewählt werden, dass kein Attribut entfernt werden kann, ohne dass dadurch das Identifikationsvermögen des Schlüssels verloren geht. Im Falle des Objekttyps „Lehrer" würde die weitere Hinzunahme des Attributs „Familienstand" in den zusammengesetzten Schlüssel gegen dieses Minimalitätsgebot

verstoßen, weil es für die Zwecke der Identifikation der Tabellenzeilen entbehrlich ist.

- *Bekanntheit:* Da für den Zugriff auf eine Tabellenzeile die Angabe seines Schlüsselwertes erforderlich ist, muss der Benutzer den Wert des Schlüssels kennen. Im Falle des Objekttyps

  Ort(Name,Kreis,Einwohnerzahl)

  würde das Attribut „Einwohnerzahl“ sicherlich identifizierend wirken, wäre also ein einfacher Schlüssel, denn zwei Orte, die in ihrer Einwohnerzahl exakt übereinstimmen, sind nicht zu erwarten. Da diese Angabe aber eher Ziel der Informationssuche ist, als dass sie dem Benutzer als Eingabe bekannt wäre, ist sie für einen Schlüssel ungeeignet.

- *Konstanz:* Die Attribute, die den Schlüssel bilden, sollten in ihren Werten konstant sein. Sich ändernde Schlüsselwerte machen nicht nur für den Benutzer „Umlernprozesse“ erforderlich, sondern bilden auch eine Gefahr für die Datenintegrität, wie an späterer Stelle dieses Kapitels sichtbar wird. Deshalb wird man für die Lehrer-Tabelle das Attribut „Familienname“ nicht gern im Schlüssel verwenden, weil sich sein Wert – beispielsweise durch Heirat – ändern kann. Für die Orts-Tabelle ist das Attribut „Einwohnerzahl“ als Schlüssel bzw. Schlüsselbestandteil schon deshalb völlig ungeeignet, weil sich sein Wert laufend verändert.

- *Wertepflicht:* Zur Sicherung des Zugriffs muss es für jede Zeile einer Tabelle einen eindeutigen Schlüsselwert geben. Ein Attribut, das Bestandteil des Schlüssels ist, darf somit für keine Zeile undefiniert bleiben. Man spricht von der *objektbezogenen Integritätsbedingung* (engl. entity integrity constraint).

Aus der Menge der Schlüssel einer Tabelle wird einer als *Primärschlüssel* ausgezeichnet:

Primärschlüssel

Definition: Ein *Primärschlüssel* (engl. primary key) einer Tabelle ist ein ausgewählter Schlüssel dieser Tabelle, der in erster Linie für den Zugriff auf die Tabellenzeilen verwendet wird. Alle anderen Schlüssel der Tabelle heißen *Sekundärschlüssel* (engl. secondary key) oder *Alternativschlüssel* (engl. alternate key).

Welcher der Schlüssel zum Primärschlüssel gewählt wird, hängt nur von praktischen und nicht von theoretischen Erwägungen ab. Prinzipiell ist jeder Schlüssel als Primärschlüssel geeignet.

Darstellung des Primärschlüssels

In der Typbeschreibung des Objekttyps wird der Primärschlüssel dadurch kenntlich gemacht, dass das entsprechende Attribut (bzw. die Attribute-Kombination) unterstrichen und - auf Grund der Unikalität - im **Fettdruck** dargestellt wird.

Unikalität des zusammengesetzten Schlüssels

Es sei ausdrücklich darauf hingewiesen, dass im Falle eines zusammengesetzten Primärschlüssels nur die – durch Pluszeichen verbundene - *Kombination* seiner Attribute *unikal* ist. Umgekehrt gilt auch, dass jedes Attribut eines zusammengesetzten Primärschlüssels für sich allein *nicht-unikal* ist. Das ergibt sich aus der Forderung nach Minimalität des Primärschlüssels: Wäre nämlich eines der Attribute eines zusammengesetzten Primärschlüssels für sich allein schon unikal, dann könnten die weiteren Attribute aus dem zusammengesetzten Primärschlüssel entfernt werden.

Wir haben uns bei der Tabellen-Typbeschreibung dafür entschieden, sämtliche - durch Pluszeichen verbundenen und gemeinsam unterstrichenen - Attribute eines zusammengesetzten Primärschlüssels auch **gemeinsam+im+Fettdruck** darzustellen, um die *Unikalität* des zusammengesetzten Primärschlüssels zu betonen. Der Leser möge stets daran denken, dass jedes einzelne Attribut *nicht-unikal* ist – also für sich allein eigentlich nicht im Fettdruck dargestellt werden müsste.

Auf die Kennzeichnung der Sekundärschlüssel, die im Kontext unserer Betrachtungen nicht von Belang sind, wird im Weiteren verzichtet.

Die Typbeschreibungen der Objekttypen „Lehrer“ und „Ort“ nehmen somit die folgende Form an:

Lehrer(**Personalnummer**,Vorname,
Familienname,Geburtsdatum,Familienstand)

Ort(**Name+Kreis**,Einwohnerzahl)

Brücken zwischen den Dateninseln

Wir können uns nun der Frage zuwenden, wie im relationalen Datenbank-Modell die „Brücken" zwischen den „Dateninseln" gebildet werden können. Am Anfang dieses Abschnitts wurde als besonderes Merkmals des relationalen Datenbank-Modells herausgestellt, dass ein solcher „Brückenschlag" nicht „fest verdrahtet" durch Adressverweise (Pointer), sondern ausschließlich auf der logischen (Tabellen-)Ebene erfolgt. Außerdem werden die „Brücken" nicht zum Zeitpunkt der Datenspeicherung, sondern dynamisch erst bei der Informationssuche hergestellt.

Wenn eine Tabellenzeile a der Tabelle A auf eine Zeile b der Tabelle B verweisen soll (a→b), dann geschieht das dadurch, dass in a als zusätzlicher Wert dieser Tabellenzeile der Identifikator von b abgelegt wird. Der Identifikator von b ist aber der Wert, den der Primärschlüssel der Tabelle B in der Zeile b annimmt. Da im konzeptionellen Datenmodell nicht Beziehungen auf der Ebene der Objekte, sondern Beziehungstypen auf der Ebene der Objekttypen beschrieben werden, wird diese Form der Querverweise auf die Ebene der Tabellen übertragen:

Fremdschlüssel

Definition: Sollen die Zeilen einer Tabelle A jeweils auf eine Zeile der Tabelle B verweisen, so wird der Primärschlüssel von B als zusätzliches Attribut (bzw. als Attribute-Kombination) in die Tabelle A aufgenommen. In A wird der Primärschlüssel von B als *Fremdschlüssel* (engl. foreign key) bezeichnet.

Durch die Bezeichnung „Fremdschlüssel" soll ausgedrückt werden, dass es sich dabei um den (Primär-)„Schlüssel" einer „fremden" Tabelle handelt. Diese Bezeichnung ist unglücklich gewählt, denn die oben angegebene Definition schließt nicht aus, dass die Tabelle A mit der Tabelle B identisch ist. Dann verweist eine Zeile von A auf eine andere Zeile derselben Tabelle. Damit lassen sich rekursive sachlogische Zusammenhänge beschreiben, wie wir im Abschnitt 3.4.4 sehen werden. In diesem Fall ist die Bezeichnung „Fremdschlüssel" unsinnig, weil

der Verweis nicht durch den Primärschlüssel einer *„fremden"* Tabelle, sondern durch den Primärschlüssel der *eigenen* Tabelle herbeigeführt wird. Die „besitzneutrale" Bezeichnung *„Verweisschlüssel"* wäre deshalb eher angebracht, ist aber nicht üblich. Wir werden deshalb im weiteren stets vom „Fremdschlüssel" sprechen.

Mehrere Fremdschlüssel in einer Tabelle

Soll eine Zeile a der Tabelle A außer auf eine Zeile b der Tabelle B auch noch auf eine Zeile c der Tabelle C verweisen (a→b und a→c), dann müssen in die Tabelle A sowohl der Primärschlüssel der Tabelle B als auch der Primärschlüssel der Tabelle C als Fremdschlüssel aufgenommen werden. Eine Tabelle kann somit zwar nur einen Primärschlüssel besitzen, kann aber beliebig viele Fremdschlüssel aufnehmen.

Darstellung eines Fremdschlüssels

In der Typbeschreibung einer Tabelle wird ein Fremdschlüssel dadurch kenntlich gemacht, dass die entsprechende Attributbezeichnung (bzw. die Bezeichnungen der kombinierten Attribute) in nach oben weisende Pfeile eingeschlossen wird. Damit soll typographisch zum Ausdruck gebracht werden, dass der Fremdschlüssel die Funktion eines „Verweisschlüssels" erfüllt:

Fremdschlüssel: ⇑Bez[A]⇑
⇑Bez[A_1]+Bez[A_2]+ ... + Bez[A_n]⇑

Durch die Verwendung von Pluszeichen anstelle von Kommas soll ausgedrückt werden, dass erst eine Kombination aller Attributwerte den Verweis auf ein Objekt bildet.

Beispiel für Fremdschlüssel

Als Beispiel für die Repräsentation eines Beziehungstyps im relationalen Datenbank-Modell soll der sachlogische Zusammenhang zwischen den Objekttypen „Ort" und „Schule" betrachtet werden, nämlich der Zusammenhang „Schule *liegt in* Ort". Ohne die Repräsentation des Beziehungstyps haben die Tabellen die folgenden Typbeschreibungen:

Ort(**Name+Kreis**,Einwohnerzahl)

Schule(**Name+Postleitzahl**,Straße,Hausnummer)

Jedes Objekt des Objekttyps „Schule" soll nun auf ein Objekt des Objekttyps „Ort" verweisen, nämlich auf denjenigen Ort, in dem die betreffende Schule liegt. Dazu ist es erforderlich, in jede Zeile der Tabelle „Schule" denjenigen Wert des Primärschlüssels

der Tabelle „Ort“ aufzunehmen, durch den der jeweiligen Ort identifiziert wird. Die beiden Tabellen haben nun die folgenden Typbeschreibungen:

Ort(**Name+Kreis**,Einwohnerzahl)

Schule(**Name+Postleitzahl**,Straße,Hausnummer,
⇑⇑Ortsname+Kreis⇑⇑)

Bei der „Dopplung“ des Primärschlüssels der Tabelle „Ort“ als Fremdschlüssel der Tabelle „Schule“ wurde die Attributbezeichnung „Name“ in „Ortsname“ umgewandelt, weil die Attributsbezeichnungen einer Tabelle voneinander paarweise verschieden sein müssen.

Beispiel-Tabellen für die beiden Objekttypen sind in Abbildung 3-6 dargestellt.

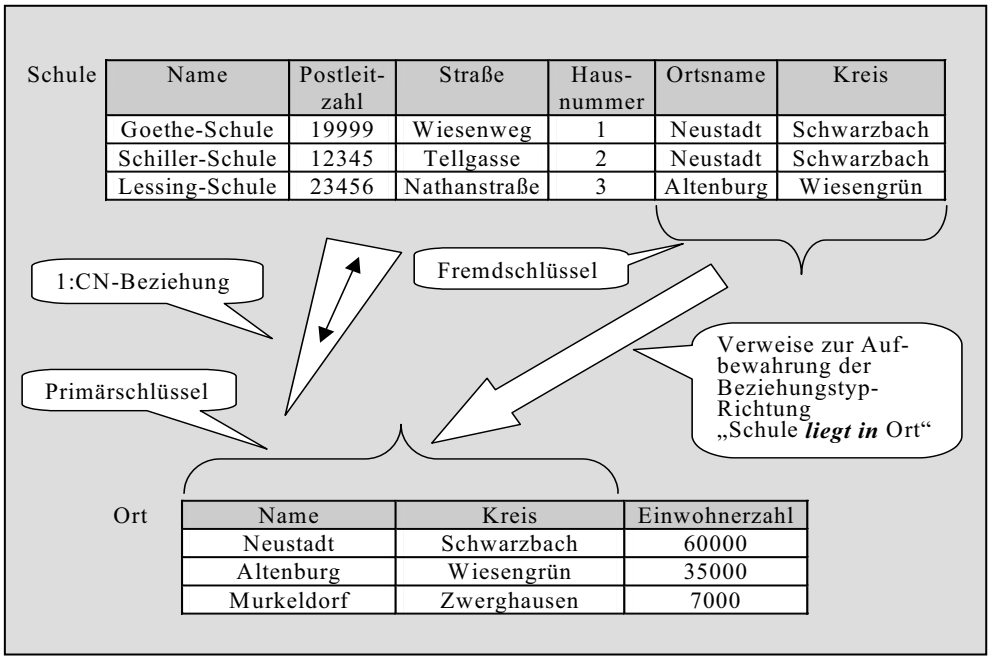


Abb. 3-6: Repräsentation eines Beziehungstyps zwischen zwei Tabellen

Aus diesem kleinen Beispiel kann man bereits die am Anfang dieses Abschnitts besprochenen Besonderheiten des relationalen Datenbank-Modells hinsichtlich der Möglichkeiten zur Navigation zwischen den Objekttypen erkennen:

Einstiegspunkte

1. *Einstiegspunkte: Jeder Objekttyp ist ein möglicher Einstiegspunkt für die Informationssuche.*

   Die Informationssuche kann sowohl vom Objekttyp „Schule" als auch vom Objekttyp „Ort" aus beginnen. Ein anzugebender Wert für den Primärschlüssel der jeweiligen Tabelle bietet die Möglichkeit, auf die Daten eines speziellen Objekts *o* zuzugreifen und eine „Brückenwanderung" zu denjenigen Objekten zu unternehmen, die mit *o* sachlogisch in Beziehung stehen. Diese Brückenwanderung wird im Punkt 3 beschrieben. Hier soll zunächst per „Direkteinstieg" nach Daten eines Objekts gesucht werden, und zwar:

   a) nach „Straße und Hausnummer" der „Lessing-Schule+23456" und

   b) nach der „Einwohnerzahl" von „Murkeldorf+Zwerghausen".

Structured Query Language

Die beiden Informationssuchen können mit Hilfe der Datenbank-Abfragesprache *„Structured Query Language"* (SQL) durch folgende Selektionsanweisungen durchgeführt werden:

a)
```
SELECT  Schule.Straße, Schule.Hausnummer
FROM    Schule
WHERE   Schule.Name = "Lessing-Schule" AND
        Schule.Postleitzahl = 23456;
```

b)
```
SELECT  Ort.Einwohnerzahl
FROM    Ort
WHERE   Ort.Name = "Murkeldorf" AND
        Ort.Kreis = "Zwerghausen";
```

In den Abfragen wird zunächst jeweils angewiesen, welche Daten in der Ergebnistabelle erscheinen sollen, dann wird die Tabelle genannt, der diese Daten zu entnehmen sind und schließlich wird die Bedingung formuliert, der die auszugebenden Tabellenzeilen genügen müssen. Die Ergebnisse der Abfragen werden in Tabellenform dargestellt:

a)

Straße	Hausnummer
Nathanstraße	3

b)

Einwohnerzahl
7000

Strukturelle Beschränkungen

2. *Strukturelle Beschränkungen: Die Objekttypen können durch ein beliebig strukturiertes Netzwerk miteinander verbunden werden. Die Brücken dieses Netzwerkes bilden unterschiedliche Beziehungstypen ab, können aber in beiden Richtungen überquert werden.*

Bei dem Beziehungstyp zwischen den Objekttypen „Ort“ und „Schule“ handelt es sich um einen 1:CN-Beziehungstyp. Man erkennt aus der Abbildung 3-6, dass die Verweisrichtung „Fremdschlüssel ⇒ Primärschlüssel“ von der CN-Seite zur 1-Seite des Beziehungstyps führt.

Jede Zeile der Tabelle „Schule“ verweist nur auf *einen* Ort. Im Gegensatz dazu wird auf eine gegebene Zeile der Tabelle „Ort“ (nämlich auf „Neustadt+Schwarzbach“) von *mehreren* Schulen (von der „Goethe-Schule+19999“ und von der „Schiller-Schule+12345“) her verwiesen. Auf eine andere Zeile der Tabelle „Ort“ („Murkeldorf+Zwerghausen“) wird gar nicht verwiesen. Wir werden im Kapitel 5 ausführlich untersuchen, welche Beziehungstypen sich im relationalen Datenbank-Modell repräsentieren lassen. Hier sei vorerst nur soviel gesagt, dass der *1:CN-Beziehungstyp* den Grundtyp der repräsentierbaren Beziehungstypen bildet, der im Sonderfall zum 1:C-Beziehungstyp „entarten“ kann. Dagegen können *CM:CN-Beziehungstypen* nicht in natürlicher Weise repräsentiert werden.

1:CN-Beziehungstyp

Dass der 1:CN-Beziehungstyp eine Navigation in beiden Richtungen ermöglicht, wird im folgenden Punkt 3 deutlich werden.

Zeitpunkt des Brückenschlags

3. *Zeitpunkt des Brückenschlags: Die Brücken zwischen den Dateninseln werden nicht zum Zeitpunkt der Speicherung, sondern erst zum Zeitpunkt der Informationssuche angelegt.*

Zwischen den Tabellen „Schule“ und „Ort“ bestehen *keine physischen* - d.h. auf der Ebene der Speicherstruktur realisierten - Zusammenhänge. Jede Tabelle existiert für sich und

unabhängig von der anderen: Es gibt keine „Brücke“ zwischen den „Dateninseln“. Der ursprüngliche Zusammenhang zwischen den sachlogisch verbunden Zeilen wurde durch die Form der Tabellen-Speicherung „zerrissen“ und lediglich auf der *logischen* Ebene in Form von Verweisen (durch die Fremdschlüsselwerte) aufgehoben. Diese entsprechen den Zetteln in der Abbildung 3-4, die die „Montage-Anleitung“ zum erneuten Zusammenfügen der Einzelteile bilden. Der sachlogische Zusammenhang lässt sich nur durch die sog. *Join-Operation* wiederherstellen. Durch diese Operation werden zwei Tabellen so zusammengefasst, dass Zeilen eine Einheit bilden, bei denen die Werte von Fremdschlüssel und Primärschlüssel übereinstimmen.

Join-Operation

Will man beispielsweise in einer Tabelle alle Attribute der Schulen zusammenstellen und zu jeder Schule zusätzlich die Attribute des zugehörigen Ortes angeben, so ist der sachlogische Zusammenhang „Schule *liegt in* Ort“ wiederherzustellen. Dies erfolgt beispielsweise in *„Access“* mit Hilfe der folgenden Abfrage nach der Methode *„Query by Example“* (QBE). Abbildung 3-7 zeigt einen Ausschnitt des „Access“-Bildschirms mit der Entwurfsansicht der Datenbank-Abfrage.

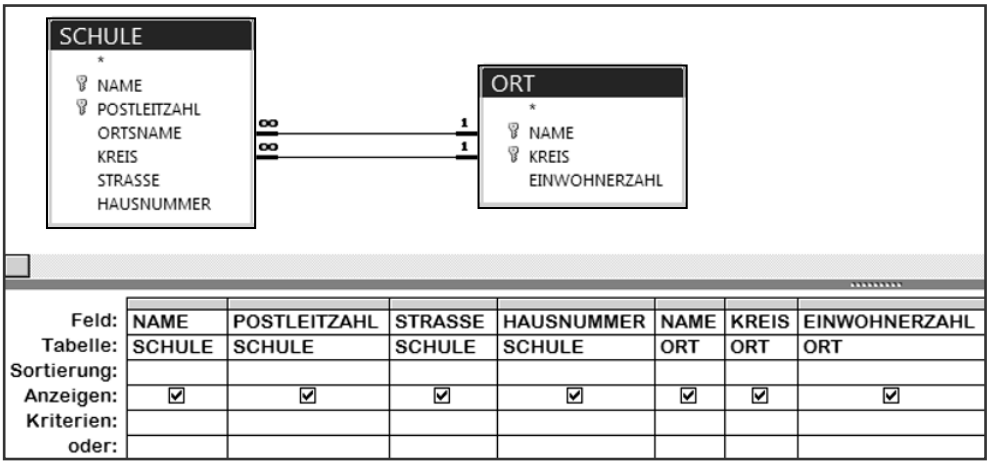
Query by Example

Im oberen Teil sind die Tabellen angegeben, die die benötigten Daten enthalten. Die Primärschlüssel der Tabellen sind durch das Schlüssel-Symbol kenntlich gemacht[17]. Der Verweis des Fremdschlüssels „⇑Ortsname+Kreis⇑“ der Tabelle „Schule“ auf den Primärschlüssel „**Name+Kreis**“ der Tabelle „Ort“ ist als *N:1-Beziehungstyp*[18] graphisch repräsentiert (∞ bzw. 1). Im unteren Teil sind die Felder der Tabellen angegeben, die in die Ergebnistabelle aufgenommen werden sollen.

---

[17] In früheren „Access“-Versionen wurden die Primärschlüssel im Fettdruck dargestellt.

[18] Eigentlich liegt hier ein CN:1-Beziehungstyp vor, aber in „Access“ wird fälschlich immer von einem 1:N- bzw. N:1-Beziehungstyp gesprochen, wenn es sich um einen 1:CN- bzw. CN:1-Beziehungstyp handelt. Im Kapitel 5 werden wir sehen, dass sich ein 1:N- bzw. N:1-Beziehungstyp gar nicht im relationalen Datenbank-Modell repräsentieren lässt.



Feld:	NAME	POSTLEITZAHL	STRASSE	HAUSNUMMER	NAME	KREIS	EINWOHNERZAHL
Tabelle:	SCHULE	SCHULE	SCHULE	SCHULE	ORT	ORT	ORT
Sortierung:							
Anzeigen:	☑	☑	☑	☑	☑	☑	☑
Kriterien:							
oder:							

Abb. 3-7: QBE-Abfrage in „Access“ zur Kopplung von Tabellen

SQL

Die in Tabellenform formulierte Datenbank-Abfrage lässt sich in äquivalenter Weise auch mit Hilfe der Datenbank-Abfragesprache *„Structured Query Language“* (SQL) formulieren:

```
SELECT   Schule.Name, Schule.Postleitzahl,
         Schule.Straße, Schule.Hausnummer,
         Ort.Name,Ort.Kreis,Ort.Einwohnerzahl
FROM     Ort  INNER JOIN  Schule
         ON   Ort.Kreis = Schule.Kreis  AND
              Ort.Name = Schule.Ortsname;
```

Diese Selektionsanweisung führt ebenfalls erst die auszugebenden Attribute auf und weist dann an, dass der bei der Speicherung „gekappte“ sachlogische Zusammenhang durch eine Join-Operation zwischen den beiden Tabellen „Ort“ und „Schule“ zurückgewonnen werden soll, indem auf Gleichheit zwischen dem Fremdschlüssel in der Tabelle „Schule“ und dem Primärschlüssel der Tabelle „Ort“ getestet wird. Die Operation „INNER JOIN“ entspricht dem dynamischen Herstellen der „Brücke“ zwischen den „Dateninseln“ der Tabellen „Ort“ und „Schule“ im Zuge der Informationssuche.

Beide Formulierungen führen durch die Ausführung der Join-Operation zur Ergebnistabelle, die in Abbildung 3-8 dargestellt ist. In der Ergebnistabelle wurden die beiden nun an

sich gleichlautenden Spaltenbezeichnungen „Name“ mittels einer Qualifizierung durch die jeweilige Tabellenbezeichnung eindeutig gemacht.

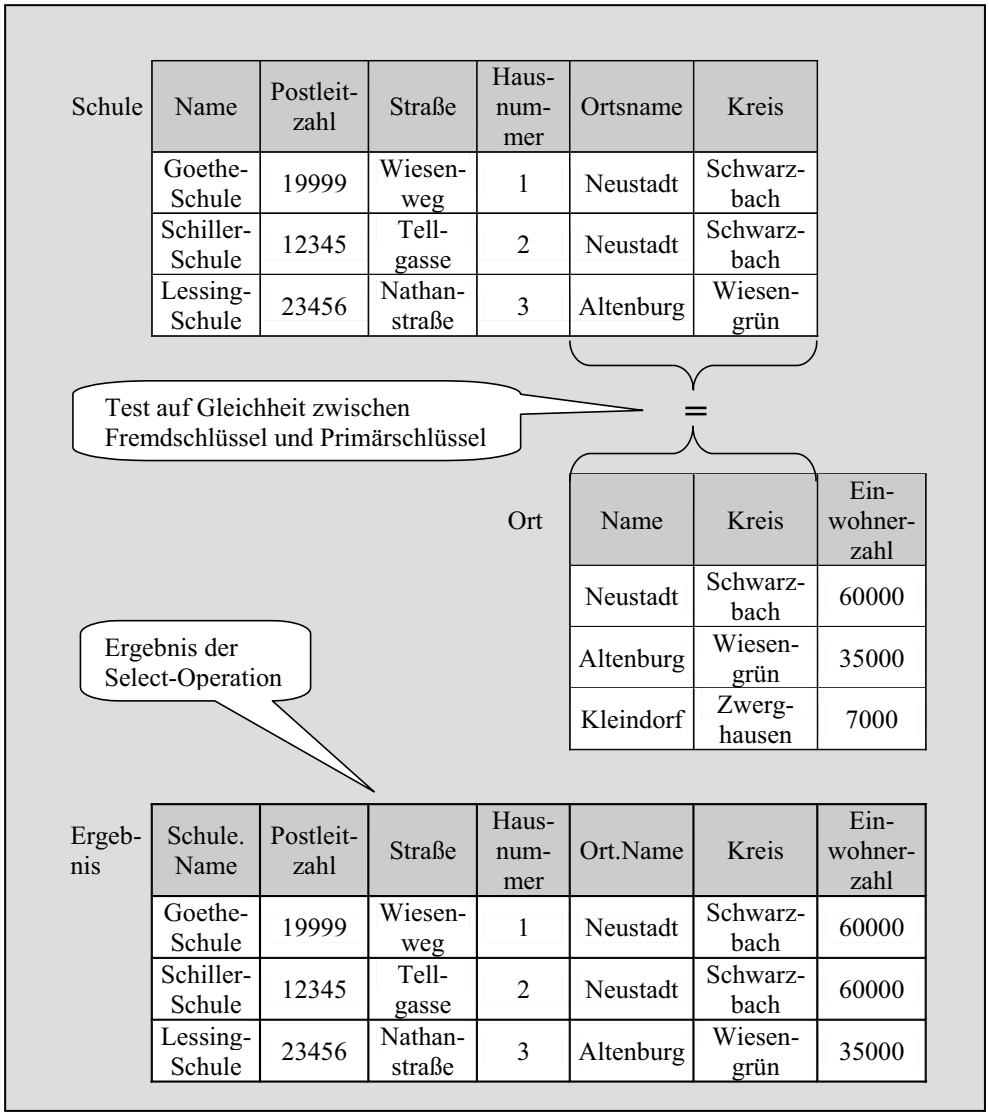


Abb. 3-8: Join-Operation zwischen zwei Tabellen

Die „Brückenwanderung“ der gerade besprochenen Informationssuche führt von der „Schule“ zum „Ort“, also von der N-Seite der „Brücke“ zur 1-Seite. Sie kann aber auch in der Gegenrichtung passiert werden. Wir fragen deshalb nach den Attributen aller Schulen, die im Ort „Neustadt+Schwarzbach“ liegen:

```
SELECT   Schule.Name, Schule.Postleitzahl,
         Schule.Straße, Schule.Hausnummer
FROM     Ort  INNER JOIN  Schule
         ON   Ort.Kreis = Schule.Kreis  AND
              Ort.Name = Schule.Ortsname
WHERE    Ort.Name = "Neustadt"  AND
         Ort.Kreis = "Schwarzbach";
```

Auch diese Abfrage weist an, dass der sachlogische Zusammenhang zwischen den Tabellen „Ort“ und „Schule“ durch eine Join-Operation wiederhergestellt werden soll, indem auf Gleichheit zwischen dem Fremdschlüssel in der Tabelle „Schule“ und dem Primärschlüssel der Tabelle „Ort“ getestet wird. Das Ergebnis ist die folgende Tabelle:

Name	Postleitzahl	Straße	Haus-nummer
Goethe-Schule	19999	Wiesenweg	1
Schiller-Schule	12345	Tellgasse	2

Außer den vorstehend diskutierten drei Besonderheiten des relationalen Datenbank-Modells hinsichtlich der Navigationsmöglichkeiten zwischen den Objekttypen sind für unsere Betrachtungen noch drei weitere Charakteristika dieses Datenbank-Modells von Bedeutung, nämlich

- die *referenzielle Integrität*,
- die Repräsentation von dualen *CM:CN-Beziehungstypen*,
- die Repräsentation von *Rekursiv-Beziehungstypen*.

Diese Besonderheiten des relationalen Datenbank-Modells sollen in den folgenden Abschnitten kurz erläutert werden.

3.4.2 Die referenzielle Integrität

Im relationalen Datenbank-Modell wird der sachlogische Zusammenhang zwischen einer Zeile a in der Tabelle A und einer Zeile b in der Tabelle B dadurch repräsentiert, dass der Fremdschlüsselwert in a mit dem Primärschlüsselwert in b übereinstimmt. Der Fremdschlüsselwert in a repräsentiert somit einen Verweis auf die Tabellenzeile b. In der Gedankenwelt des World Wide Web ist das vergleichbar mit einem Hyperlink, der auf eine andere Web-Seite verweist.

Verweisprinzip

Ein solches Verweisprinzip, das auf der Wertgleichheit von Attributen in Tabellen basiert, die voneinander physisch unabhängig sind, birgt naturgemäß Gefahren für die Integrität der gespeicherten Daten in sich. Es muss durch spezielle Maßnahmen gesichert werden, dass kein Verweis „ins Leere geht", wenn er nämlich auf eine nicht existierende Tabellenzeile zielt. Im World Wide Web würde das der Situation entsprechen, dass ein Hyperlink auf eine Web-Seite verweist, die es gar nicht (mehr) gibt. Man nennt dies einen „toten" Verweis (engl. dead link).

Ein wesentlicher Grundsatz des relationalen Datenbank-Modells besteht nun gerade im Verbot solcher „toten" Verweise, was – positiv formuliert – auch als Gebot der „referenziellen Integrität" bezeichnet wird:

Referenzielle Integrität

Definition: Wenn der sachlogische Zusammenhang zwischen zwei Tabellen A und B dadurch repräsentiert wird, dass die Werte des Fremdschlüssels in A mit Werten des Primärschlüssels in B übereinstimmen, dann besagt das *Gebot der referenziellen Integrität:*

a) *entweder* muss jeder in A auftretende Wert des Fremdschlüssels mit einem Wert des Primärschlüssels in B übereinstimmen

b) *oder* der Fremdschlüssel in A muss mit der NULL-Marke belegt sein.

Das Gebot der referenziellen Integrität fordert also, dass ein Verweis entweder auf *ein existierendes Objekt* zielen muss oder dass explizit zum Ausdruck gebracht werden muss, dass *eben kein Verweis* vorliegt.

NULL-Marke im Fremdschlüssel

Hier wird ein wesentlicher Unterschied zwischen Primär- und Fremdschlüssel sichtbar: *Keines* der Attribute eines Primärschlüssels darf mit der NULL-Marke belegt werden, dagegen können die Attribute des Fremdschlüssels *gemeinschaftlich* mit der NULL-Marke belegt sein.

Bei der Deklaration der Tabellen im relationalen Datenbank-Modell kann nun festgelegt werden, ob für die Werte eines Fremdschlüssels beide Möglichkeiten a) und b) erlaubt sind oder ob die Möglichkeit b) ausgeschlossen wird. Im Datenbank-Managementsystem *„Access“* kann für einen Fremdschlüssel – wie für jedes Tabellen-Attribut – festgelegt werden, dass das Attribut für jede Tabellenzeile einen anzugebenden Wert annehmen muss (Eingabe erforderlich → „ja“, Gültigkeitsregel → „Nicht ist Null“). Dadurch wird die NULL-Marke für den Fremdschlüssel verboten und die Möglichkeit b) explizit ausgeschlossen.

Nicht-eingabepflichtiger Fremdschlüssel

In der Typbeschreibung einer Tabelle wird ein *nicht-eingabepflichtiger Fremdschlüssel*, also ein Fremdschlüssel, dessen Elemente gemeinschaftlich mit der NULL-Marke belegt werden können, dadurch kenntlich gemacht, dass die Bezeichnungen seiner Attribute – einschließlich der „Verweispfeile“ - in Kursivschrift dargestellt werden:

nicht-eingabepflichtiger Fremdschlüssel:	$\uparrow\uparrow Bez[A]\uparrow\uparrow$ $\uparrow\uparrow Bez[A_1]+Bez[A_2]+ \ldots + Bez[A_n]\uparrow\uparrow$

Das Prinzip der referenziellen Integrität besagt, dass eine konkrete Angabe für den Fremdschlüssel entweder auf ein *existierendes Objekt* oder aber auf *gar kein Objekt* verweisen muss. Es verbietet nicht, dass der Fremdschlüssel in mehreren Zeilen denselben Wert annehmen kann, dass also mehrere Zeilen einer Tabelle auf dasselbe Objekt verweisen. Mitunter will man aber gerade das verhindern, um die Kardinalität „1“ einer Beziehungstyp-Richtung zu garantieren. In *„Access“* besteht deshalb die Möglichkeit, für den Fremdschlüssel einen Index anzulegen und Duplikate im Index zu untersagen (Indiziert → „Ja (Ohne Duplikate)“).

Unikaler Fremdschlüssel

Wenn ein Fremdschlüssel nur unikale Werte besitzen darf, dann wird dies in der Typbeschreibung der Tabelle dadurch kenntlich gemacht, dass das entsprechende Attribut (bzw. die Attribute-Kombination) im Fettdruck angegeben wird. Dieses typographische Gestaltungsmerkmal wurde schon im Abschnitt 3.4.1 im Zusammenhang mit dem stets unikalen Primärschlüssel eingeführt:

unikaler	**⇑Bez[A]⇑**
Fremdschlüssel:	**⇑Bez[A1]+Bez[A2]+...+Bez[An]⇑**

Natürlich kann ein Fremdschlüssel sowohl nicht-eingabepflichtig als auch unikal sein. Er wird dann ***kursiv und im Fettdruck*** dargestellt.

Betrachten wir noch einmal das Beispiel in Abbildung 3-6. In der Tabelle „Schule“ treten nur solche Werte des Fremdschlüssels ⇑Ortsname+Kreis⇑ auf, die auch als Werte des Primärschlüssels „**Name+Kreis**“ der Tabelle „Ort“ vorhanden sind. Damit ist die Möglichkeit a) der referenziellen Integrität verwirklicht. Andererseits enthält jede Zeile der Tabelle „Schule“ eine konkrete Werte-Kombination für den Fremdschlüssel. Die Fremdschlüssel-Elemente werden also nie gemeinschaftlich mit der NULL-Marke belegt. Die Möglichkeit b) der referenziellen Integrität wurde somit nicht realisiert. Der Fremdschlüssel ⇑Ortsname+Kreis⇑ ist also eingabepflichtig und nicht-unikal. Dem entspricht die Typbeschreibung:

Ort(**Name+Kreis**,Einwohnerzahl)
Schule(**Name+Postleitzahl**,Straße,Hausnummer,
⇑Ortsname+Kreis⇑)

Betrachtet man dagegen Orte und Gaststätten, so muss eine Gaststätte nicht unbedingt in einem Ort liegen, sondern kann sich mitten im Wald befinden. Dem entspricht die Typbeschreibung:

Ort(**Name+Kreis**,Einwohnerzahl)
Gaststätte(**Bezeichnung**,Adresse,Sitzplatzanzahl,
⇑Ortsname+Kreis⇑)

Der Fremdschlüssel *⇑Ortsname+Kreis⇑* ist nicht-eingabepflichtig und nicht-unikal. In einer Zeile der Tabelle „Gaststätte“, die eine Gaststätte im Wald beschreibt, können somit die Fremdschlüssel-Attribute gemeinschaftlich mit der NULL-Marke belegt werden.

Das Gebot der referenziellen Integrität lässt sich anschaulich darstellen, wenn man die Menge der Primärschlüssel-Werte und die Menge der Fremdschlüssel-Werte (ohne NULL-Marke) als Flächen zeichnet, wie dies in der Abbildung 3-9 geschehen ist.

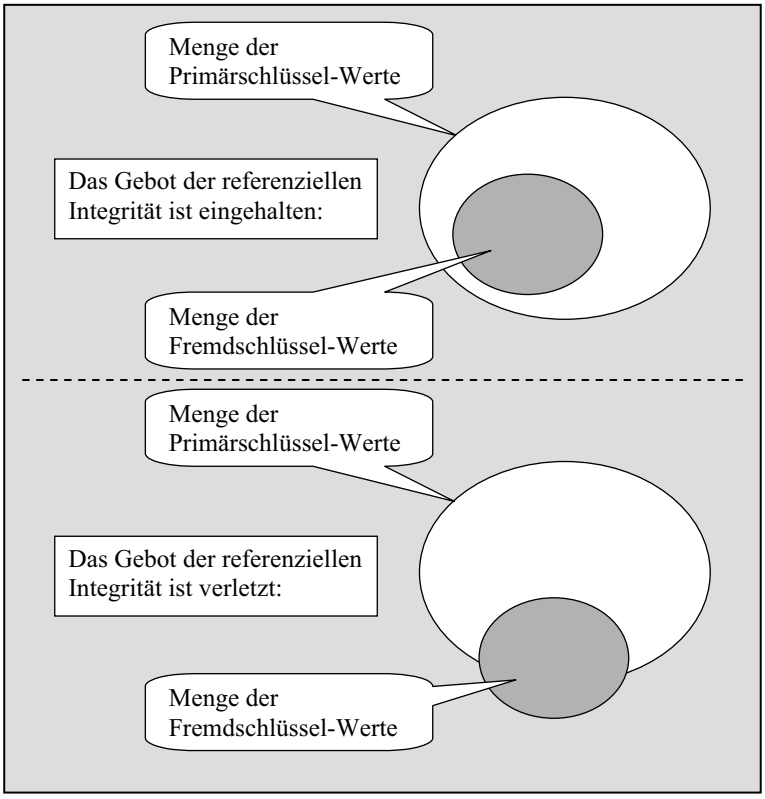


Abb. 3-9: Verhältnis der Mengen von Primärschlüssel-Werten und Fremdschlüssel-Werten

Das Gebot der referenziellen Integrität ist dann verletzt, wenn die Fläche der Fremdschlüssel-Werte nicht vollständig von der Fläche der Primärschlüssel-Werte eingeschlossen wird, wenn also

die Menge der Fremdschlüssel-Werte keine Teilmenge der Menge der Primärschlüssel-Werte ist.

Verletzung der referenziellen Integrität

Das Gebot der referenziellen Integrität kann im Lebenszyklus der Datenbank durch drei Aktivitäten verletzt werden:

1. durch die Eingabe eines Datensatzes, der einen Fremdschlüssel-Wert enthält, zu dem es keinen Primärschlüssel-Wert gibt, oder durch die Änderung eines Fremdschlüssel-Wertes auf einen Wert, der nicht als Primärschlüssel-Wert vorkommt;
2. durch die Änderung des Primärschlüssels eines Datensatzes, auf den andere Datensätze verweisen, und
3. durch das Löschen eines Datensatzes, auf den andere Datensätze verweisen.

In jedem dieser drei Fälle müssen vom Datenbank-Managementsystem geeignete Maßnahmen getroffen werden, die eine Verletzung des Gebots der referenziellen Integrität verhindern.

Wir wollen alle drei Situationen an einem einfachen Beispiel genauer untersuchen. Wir betrachten Aufträge, die von Kunden erteilt werden. Ein Auftrags-Datensatz verweist auf genau einen Kunden-Datensatz, während auf einen Kunden-Datensatz von mehreren Auftrags-Datensätzen aus verwiesen werden kann. Diese Situation entspricht den folgenden Tabellen-Typbeschreibungen:

Kunde(**Kundennummer**,Name)

Auftrag(**Auftragsnummer**,Datum,⇑Kundennummer⇑)

Der Fremdschlüssel ⇑Kundennummer⇑ ist eingabepflichtig, weil jeder Auftrag einem Kunden zugeordnet sein muss. Er ist nichtunikal, denn mehrere Aufträge können zum selben Kunden gehören. Wir gehen bei unseren Überlegungen von der Situation aus, die in der Abbildung 3-10 dargestellt ist.

Fremdschlüssel ohne zugehörigen Primärschlüssel

1. Wir nehmen an, dass der Auftrag „A004“ eines neuen Kunden „K004“ *gespeichert* werden soll, ehe dieser Kunde in die Tabelle „Kunde“ aufgenommen wird. Die Tabelle „Auftrag“ würde dann einen Datensatz enthalten, dessen Fremdschlüssel-Wert keine Entsprechung durch einen Primärschlüssel-Wert der Tabelle „Kunde“ hat. Dieselbe Verletzung des Gebots der referenziellen Integrität würde eintreten, wenn man im Auftrag „A003“ den Fremdschlüssel-Wert „K002“ auf

„K004“ *ändern* würde. Das Datenbank-Managementsystem lässt beide Operationen nicht zu.

Kunde

Kunden-nummer	**Name**
K001	Schulz
K002	Müller
K003	Meier

Auftrag

Auftrags-nummer	**Datum**	**Kunden-nummer**
A001	1.1.02	K001
A002	2.2.02	K001
A003	3.2.02	K002

Abb. 3-10: Ausgangssituation der Tabellen „Kunde“ und „Auftrag“

2. Nun soll der Primärschlüssel-Wert des Kunden „Schulz“ von „K001“ auf „K101“ *geändert* werden. Das würde dazu führen, dass die Fremdschlüssel-Werte „K001“ in den beiden Aufträgen „A001“ und „A002“ zu „toten“ Verweisen werden. Die relationalen Datenbank-Managementsysteme bieten für diesen Fall meistens drei Reaktionen an, von denen man bei der Deklaration der Datenstruktur eine auswählen kann:

On Update Restrict

- Die Änderungs-Operation wird abgelehnt (*On Update Restrict*). Dies ist in der Theorie relationaler Datenbanken die Standard-Reaktion.

On Update Cascade

- Die Änderungs-Operation führt dazu, dass in allen Auftrags-Datensätzen, die auf den veränderten Kunden-Datensatz verweisen, der Fremdschlüssel-Wert ebenfalls auf „K101“ verändert wird (*On Update Cascade*). Das betrifft die Aufträge „A001“ und „A002“. Bei komplexeren Datenstrukturen kann der geänderte Fremdschlüssel-Wert in der Tabelle B Bestandteil des Primärschlüssel-Wertes von B sein. Verweist nun der Fremdschlüssel-Wert einer dritten Tabelle C auf den Datensatz in B, so muss auch in C der Fremdschlüssel-Wert angepasst werden. Das bedeutet ein „kaskadierendes“ Ändern und motiviert die Bezeichnung „Cascade“.

On Update Set Null

- Die Änderungs-Operation führt dazu, dass alle entstehenden „toten“ Verweise mit der NULL-Marke belegt werden (*On Update Set Null*). Das ist natürlich nur dann möglich, wenn der Fremdschlüssel nicht eingabepflichtig

ist. Diese Reaktion wird von vielen Datenbank-Managementsystemen – so auch von *„Access“* - nicht angeboten. Sie ist in der Praxis auch wenig sinnvoll. Änderungen des Primärschlüssel-Wertes sollten bei den korrespondierenden Fremdschlüssel-Werten kaskadierend „nachgeführt“ werden.

Eine Änderung des Primärschlüssel-Wertes „K003“ des Kunden „Meier“ würde dagegen keine Komplikationen hervorrufen, denn er kommt in der Tabelle „Auftrag“ nicht als Fremdschlüssel-Wert vor.

3. Nun soll der Kunde „Schulz“ mit dem Primärschlüssel-Wert „K001“ *gelöscht* werden. Das würde ebenfalls dazu führen, dass die Auftrags-Datensätze „A001“ und „A002“ auf einen nicht mehr vorhandenen Kunden-Datensatz verweisen. Hier sind dieselben drei Reaktionen möglich:

On Delete Restrict

- Die Lösch-Operation wird abgelehnt (*On Delete Restrict*). Dies ist wiederum die Standard-Reaktion.

On Delete Cascade

- Die Lösch-Operation führt dazu, dass alle Auftrags-Datensätzen, die auf den gelöschten Kunden-Datensatz verweisen, ebenfalls“ gelöscht werden (*On Delete Cascade*). In unserem Beispiel würden die Aufträge „A001“ und „A002“ „mitgelöscht“ werden. Gibt es bei komplexeren Datenstrukturen Datensätze, die ihrerseits auf die „mitgelöschten“ Datensätze verweisen, so sind diese ebenfalls zu löschen. Es findet also ein „kaskadierendes“ Löschen statt.

On Delete Set Null

- Die Lösch-Operation bewirkt, dass alle entstehenden „toten“ Verweise mit der NULL-Marke belegt werden (*On Delete Set Null*). Das ist natürlich nur dann möglich, wenn der Fremdschlüssel nicht eingabepflichtig ist. Diese Reaktion kann wiederum bei vielen Datenbank-Managementsystemen – so auch bei „Access“ – nicht gewählt werden.

Der Kunde „Meier“ könnte jedoch ohne weiteres gelöscht werden, weil kein Auftrags-Datensatz auf ihn verweist.

3.4.3 Die Repräsentation von dualen CM:CN-Beziehungstypen

Es wurde schon mehrfach darauf hingewiesen, dass sich die dualen CM:CN-Beziehungstypen im relationalen Datenbank-Modell nicht in natürlicher Weise repräsentieren lassen. Dies ist eine Folge der Grundbedingung dieses Modells, dass ein Attribut keine zusammengesetzten Werte – und insbesondere keine Wertemenge – annehmen darf (das war auch der Grund für die Herbeiführung der 1. Normalform für das konzeptionelle Datenmodell, vgl. Abschnitt 2.7.1). Es ist also nicht möglich, dass der Fremdschlüsselwert einer Tabellenzeile auf mehrere andere Tabellenzeilen verweist. Die Kardinalität N einer Beziehungstyp-Richtung ist sozusagen nur durch das „passive Wahlrecht", also dadurch zu realisieren, dass *von mehreren Zeilen auf dieselbe* Tabellenzeile verwiesen wird. Das „aktive Wahlrecht" dagegen ermöglicht lediglich den Verweis auf *eine* Tabellenzeile.

Aktives und passives „Wahlrecht"

Das folgende Beispiel soll diesen Umstand demonstrieren. Wir wollen Daten über Mitarbeiter und Projekte speichern. Die vereinfachten Typbeschreibungen der Tabellen „Mitarbeiter" und „Projekt", in denen noch kein sachlogischer Zusammenhang berücksichtigt wird, haben die folgende Form:

Mitarbeiter(**Personalnummer**,Name,Gehalt)

Projekt(**Projektnummer**,Bezeichnung,Beginn)

Der nun zusätzlich zu repräsentierende sachlogische Zusammenhang zwischen den Mitarbeitern und den Projekten besteht darin, dass ein Mitarbeiter an keinem, einem oder mehreren Projekten beteiligt ist und dass ein Projekt von keinem, einem oder mehreren Mitarbeitern bearbeitet wird. Es handelt sich also um einen CM:CN-Beziehungstyp. Somit müsste die Möglichkeit bestehen, dass der Fremdschlüsselwert in einer Zeile der Mitarbeiter-Tabelle auf *mehrere* Zeilen der Projekt-Tabelle oder dass der Fremdschlüsselwert in einer Zeile der Projekt-Tabelle auf *mehrere* Zeilen der Mitarbeiter-Tabelle verweist. Da aber ein Fremdschlüsselwert – wie jeder Wert in einer Tabelle – nur atomar sein kann, muss man eine andere Lösung finden, um den CM:CN-Beziehungstyp darzustellen.

CM:CN-Beziehungstyp

Zur Lösung des Problems führt man eine neue Tabelle ein, deren Zeilen jeweils *einen* Mitarbeiter mit *einem* Projekt koppeln. Eine solche – ausschließlich DV-technisch bedingte – Tabelle bezeich-

Koppel-Tabelle

nen wir als *„Koppel-Tabelle“*. Damit die in unserem Beispiel erforderliche Koppel-Tabelle „Mitarbeiter/Projekt“ ihre Funktion erfüllen kann, muss sie den Primärschlüssel **Personalnummer** der Mitarbeiter-Tabelle und zusätzlich den Primärschlüssel **Projektnummer** der Projekt-Tabelle als eingabepflichtige nicht-unikale Fremdschlüssel enthalten. Das führt zunächst zu den folgenden vorläufigen Tabellen-Typbeschreibungen:

Mitarbeiter(**Personalnummer**,Name,Gehalt)
Mitarbeiter/Projekt(⇑Personalnummer⇑,⇑Projektnummer⇑)
Projekt(**Projektnummer**,Bezeichnung,Beginn)

Primärschlüssel der Koppel-Tabelle

Die Koppel-Tabelle „Mitarbeiter/Projekt“ muss – so wie jede Tabelle - einen Primärschlüssel besitzen. Sie hat aber nur die beiden Fremdschlüssel ⇑Personalnummer⇑ und ⇑Projektnummer⇑ als Attribute. Keiner dieser nicht-unikalen Fremdschlüssel kann für sich allein als Primärschlüssel dienen, weil es einerseits mehrere Zeilen geben kann, die auf denselben Mitarbeiter verweisen, und weil es andererseits mehrere Zeilen geben kann, die auf dasselbe Projekt verweisen. Die Kopplung der beiden Fremdschlüssel ist jedoch ein geeigneter Primärschlüssel:

Mitarbeiter(**Personalnummer**,Name,Gehalt)
Mitarbeiter/Projekt(**⇑Personalnummer⇑+**
⇑Projektnummer⇑)
Projekt(**Projektnummer**,Bezeichnung,Beginn)

Man erkennt, dass in einer Tabelle A ein Fremdschlüssel, der Primärschlüssel in einer anderen Tabelle B ist, zugleich Bestandteil des Primärschlüssels von A sein kann. Die Zusammenhänge zwischen den drei Tabellen werden in Abbildung 3-11 veranschaulicht.

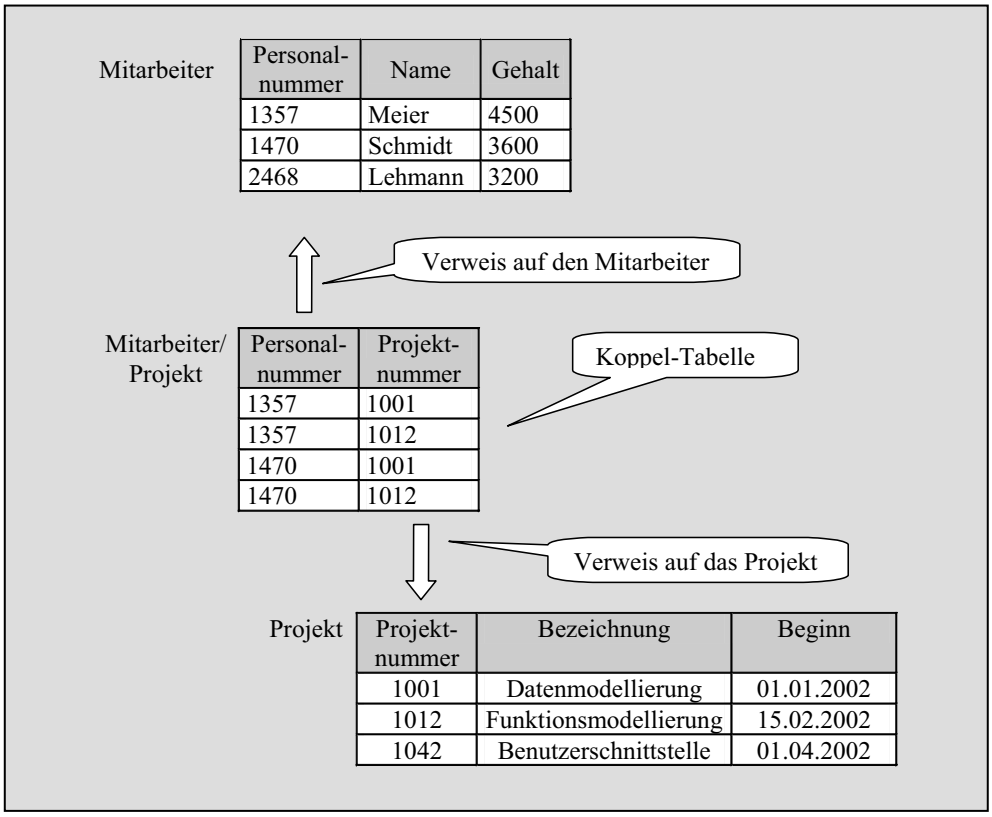


Abb. 3-11: Repräsentation des CM:CN-Beziehungstyps durch eine Koppel-Tabelle

Man sieht, dass durch die Koppel-Tabelle ein Mitarbeiter (z.B. Meier mit der Personalnummer „1357“) zwei Projekten (Projektnummern „1001“ und „1012“) zugeordnet ist, andererseits ist ein Projekt (z.B. das Projekt mit der Projektnummer „1001“) mit zwei Mitarbeitern (Personalnummern „1357“ und „1470“) verbunden. Lehmann (Personalnummer „2468“) arbeitet an keinem Projekt mit. Für das Projekt „Benutzerschnittstelle“ mit der Projektnummer „1042“ wurden (noch) keine Mitarbeiter benannt.

Das Problem, einen CM:CN-Beziehungstyp zwischen den Objekttypen A und B im relationalen Datenbank-Modell zu repräsentieren, wurde also dadurch gelöst, dass eine zusätzliche *Koppel-Tabelle* A/B eingeführt wurde und dass ein *1:CN-* und ein *CN:1-Beziehungstyp* deklariert werden, nämlich zwischen A und A/B bzw. zwischen A/B und B.

3.4.4 Die Repräsentation von Rekursiv-Beziehungstypen

Bei der Diskussion der Bezeichnung „Fremdschlüssel“ wurde schon darauf hingewiesen, dass der Wert, den ein Fremdschlüssel der Tabelle A in der Zeile a annimmt, nicht zwangsläufig auf eine Zeile b in einer „fremden“ Tabelle B verweisen muss, sondern dass das Verweisziel durchaus in der „eigenen“ Tabelle A liegen kann. Dies entspricht der Repräsentation eines Rekursiv-Beziehungstyps, bei dem ein sachlogischer Zusammenhang zwischen Objekten dargestellt wird, die zum selben Objekttyp gehören. In diesem Fall stimmt der „Fremdschlüssel“ mit dem Primärschlüssel der Tabelle überein. Da jedoch die Spaltenbezeichnungen einer Tabelle paarweise voneinander verschieden sein müssen, muss die Bezeichnung des „gedoppelten“ Primärschlüssels natürlich verändert werden.

Beispiel für Rekursiv-Beziehungstyp

Betrachten wir als Beispiel das typische Unterstellungsverhältnis der Mitarbeiter im Unternehmen. Über die Chefs und ihre Mitarbeiter werden dieselben Informationen gesammelt, so dass es keinen Grund dafür gibt, sie in zwei verschiedene Objekttypen einzuordnen. Wir gehen wieder von folgender Tabellen-Typbeschreibung aus:

Mitarbeiter(**Personalnummer**,Name,Gehalt)

Soll nun gespeichert werden, wer wen als Chef hat, so ist dies ein sachlogischer Zusammenhang zwischen Objekten, die im selben Objekttyp „Mitarbeiter“ vereint sind. Ein Mitarbeiter kann keinen anderen Mitarbeiter oder – in seiner Rolle als Chef – mehrere andere Mitarbeiter anleiten. Andererseits hat ein Mitarbeiter keinen (wenn er sich in der obersten Leitungsebene befindet) oder einen Chef. Es handelt sich dabei also um einen C:CN-Rekursiv-Beziehungstyp.

Will man nun den C:CN-Rekursiv-Beziehungstyp im relationalen Datenbank-Modell repräsentieren, so muss in der Tabelle „Mitarbeiter“ der Primärschlüssel **Personalnummer** mit einer veränderten Spalten-Bezeichnung „gedoppelt“ und als Fremdschlüssel vereinbart werden. Da der Fremdschlüssel auf den jeweiligen Chef eines Mitarbeiters verweist, wählen wir die Bezeichnung „Chef-Personalnummer“:

Mitarbeiter(**Personalnummer**,Name,Gehalt,
⇑Chef-Personalnummer⇑)

Der Fremdschlüssel ist nicht-eingabepflichtig (kursiv gesetzt) und ist nicht-unikal (nicht im Fettdruck). Ein Beispiel für die entsprechende Tabelle zeigt die Abbildung 3-12.

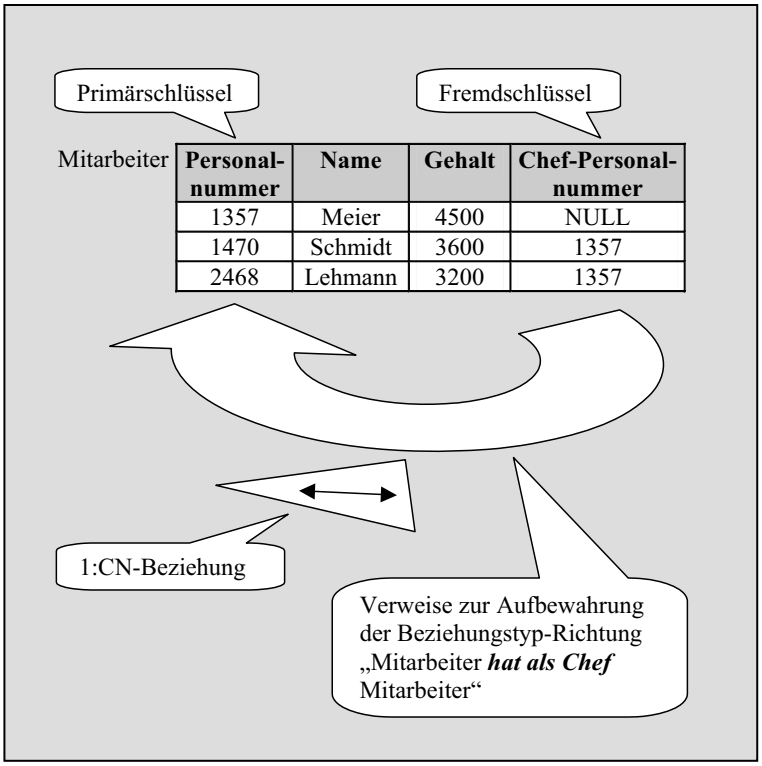


Abb. 3-12: Repräsentation eines Rekursiv-Beziehungstyps in einer Tabelle

Man erkennt, dass es sich hierbei um einen C:CN-Beziehungstyp handelt:

- *C-Seite:* In der Zeile mit dem Primärschlüssel-Wert „1357“ bringt die Belegung des Fremdschlüssels mit der NULL-Marke zum Ausdruck, dass der betrachtete Mitarbeiter keinen Chef hat. Andererseits kann in einer Zeile (z.B. mit dem Pri-

märschlüssel-Wert „1470“) nur auf einen einzigen Chef (auf den Mitarbeiter „1357“) verwiesen werden.

- *CN-Seite:* Ein Wert des Primärschlüssels (z.B. „1470“) kommt nicht als Fremdschlüsselwert vor (der Mitarbeiter „1470“ ist kein Chef), ein anderer Primärschlüssel-Wert („1357“) taucht mehrfach als Fremdschlüssel-Wert auf (der Mitarbeiter „1357“ ist Chef von 2 anderen Mitarbeitern).

4 Die zweite Etappe: Vom Datenmodell zur Datenbank

Im Kapitel 2 wurde das *Entity-Relationship-Modell* als Sprache für die graphische Repräsentation des konzeptionellen Datenmodells vorgestellt. Das Kapitel 3 befasste sich dann mit den Möglichkeiten, die die *Datenbank-Modelle* für die Repräsentation der Objekte und der zwischen ihnen bestehenden sachlogischen Zusammenhänge bereitstellen. Dabei standen die Möglichkeiten zur Daten-Strukturierung, die das relationale Datenbank-Modell anbietet, im Mittelpunkt des Interesses.

Nun können wir uns der Frage zuwenden, wie sich die Aussagen des *konzeptionellen Datenmodells* in ein *logisches Datenschema* überführen lassen, das dem Gestaltungsspielraum des relationalen Datenbank-Modells entspricht. Für die Sprachelemente des Entity-Relationship-Modells wird ausführlich und an Hand von Beispielen untersucht, ob und wie sie sich mit Hilfe der Strukturelemente des relationalen Datenbank-Modells repräsentieren lassen. Dabei werden *20 Transformationsregeln* hergeleitet, die den Algorithmus der Umsetzung darstellen. Das im Kapitel 2 entwickelte konzeptionelle Datenmodell für das Schulbeispiel wird mit Hilfe dieser Transformationsregeln *„von Hand“* in das relationale Datenbank-Modell überführt. Parallel dazu wird die *automatische Generierung* der Tabellen-Struktur durch das CASE-Tool *„PowerDesigner“* vorgeführt.

In diesem Kapitel werden die 20 Transformationsregeln im Sinne eines „Kochbuchs“ einzeln nacheinander entwickelt. Dem Leser bietet sich damit ein „bunter Blumenstrauß“ von Verhaltensmustern, von denen er sich beim Datenbankentwurf leiten lassen kann. Eine systematische – eher theoretisch orientierte – Analyse der Möglichkeiten des Datenbankentwurfs wird später im Kapitel 5 vorgenommen.

Die Einordnung dieses Kapitels in den Kontext des Lehrbuchs zeigt die Abbildung 4-1.

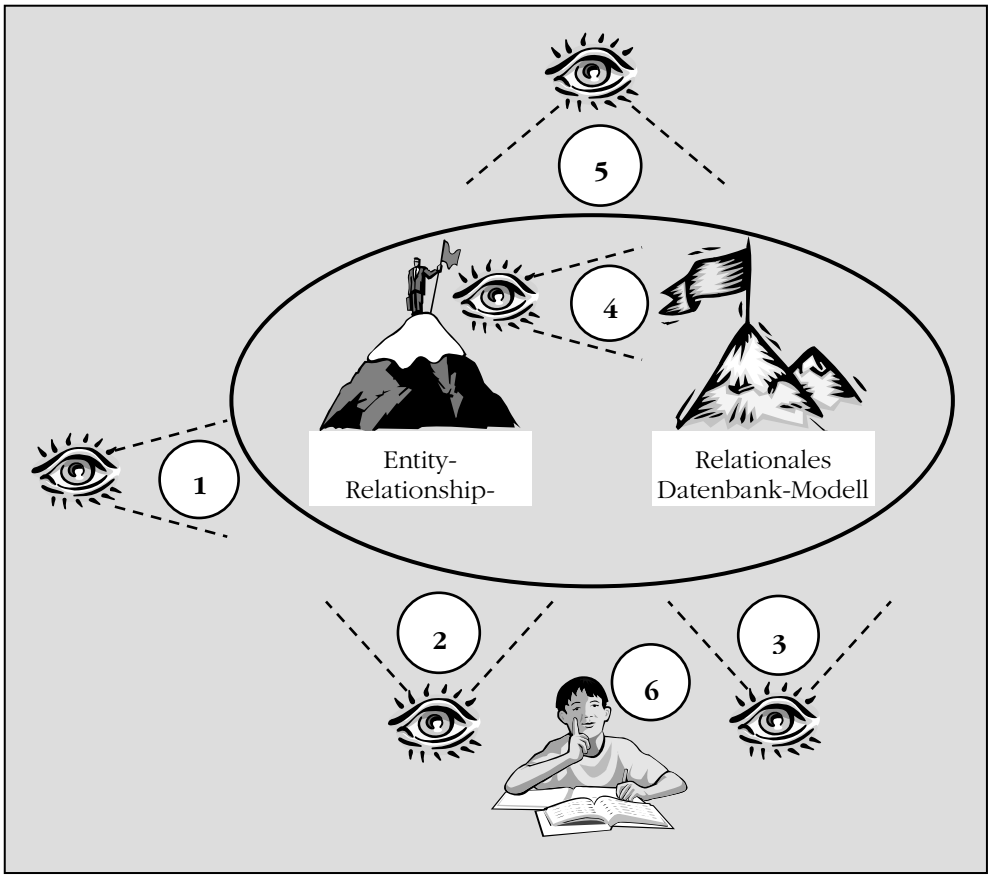


Abb. 4-1: Gegenstand des Kapitels 4

Ruft man sich den Entwicklungsweg betrieblicher Datenbank-Anwendungssysteme nach der CASE-Technologie in Erinnerung, wie er im Kapitel 1 beschrieben wurde, so geht es um die folgende Aufgabe: Im Prozess der Modellierung wurde als eines der Bestandteile des Fachkonzepts das *konzeptionelle Datenmodell* entwickelt. Dieses liegt auf der syntaktischen Ebene der Informationsverarbeitung vor und kann vom Computer weiterverarbeitet werden. Generator-Programme bieten nun die Möglichkeit, aus diesem Datenmodell die Struktur der Datenbank - also das *logische Datenschema* - automatisiert abzuleiten. In der Abbildung 4-2 wird dieser Transformationsschritt durch eine dunkle Ellipse hervorgehoben.

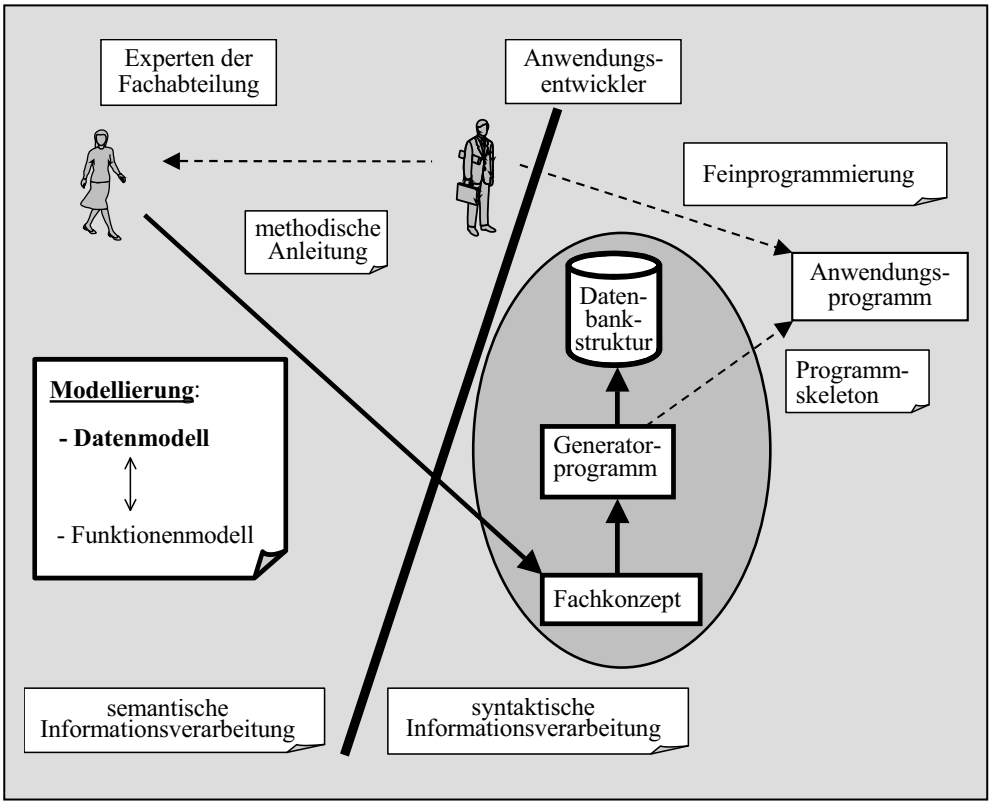


Abb. 4-2: Transformation des Datenmodells in die Datenbank-Struktur

In diesem Transformationsprozess wird die *informations-orientierte Sicht* auf die Daten, wie sie im *konzeptionellen Datenmodell* zum Ausdruck kommt, auf die *daten-orientierte Sicht*, also auf das *logische Datenschema* der Datenbank, abgebildet (vgl. Abschnitt 1.3). Bei der Beschreibung der informationsorientierten Sicht mit Hilfe der Sprache des Entity-Relationship-Modells bestand die Zielrichtung darin, Objekttypen und Beziehungstypen unabhängig vom später einzusetzenden Datenbank-Managementsystem und somit auch unabhängig von einem speziellen Datenbank-Modell zu beschreiben. Nun muss der Versuch unternommen werden, das konzeptionelle Datenmodell - möglichst ohne Einbuße an semantischem Gehalt - mit den verfügbaren Strukturierungsmitteln eines kommerziellen Datenbank-

Managementsystems wiederzugeben. Dabei sind zwei Fragen zu beantworten:

1. Welches *Datenbank-Modell* soll der zu erstellenden Datenbank zugrunde liegen?
2. Welche Möglichkeiten bietet das zu verwendende *Datenbank-Managementsystem* für die Gestaltung der Datenstrukturen?

Hinsichtlich der ersten Fragestellung wollen wir uns im weiteren auf das *relationale Datenbank-Modell* beschränken. Als kommerzielles Datenbank-Managementsystem wählen wir wegen seiner großen Verbreitung in der Welt der Personal-Computer *„Access“* von Microsoft.

In den folgenden Abschnitten diesen Kapitels werden zunächst 20 Transformationsregeln hergeleitet, nach denen das konzeptionelle Datenmodell in ein logisches Datenschema, also in unserem Falle in eine Tabellenstruktur für das relationale Datenbank-Managementsystem *„Access“*, überführt werden kann. Nachdem eine derartige Umsetzung „von Hand“ beschrieben wurde, erläutern wir die automatisierte Transformation mit Hilfe des von uns eingesetzten Modellierungs-Tools *„PowerDesigner“*.

4.1 Transformation von Objekttypen

Die Objekttypen des Datenmodells stellen das Haupt-Ordnungsprinzip der Datenmodellierung dar. Sie beschreiben die Klassen-Struktur, in die die speicherrelevanten Objekte des Gegenstandsbereichs eingeordnet werden. Im relationalen Datenbank-Modell wird diese Klassen-Struktur durch einen Satz von Tabellen wiedergegeben.

Transformationsregel T01

Für jeden Objekttyp des Datenmodells wird eine Tabelle vereinbart. Dabei gilt die Transformationsregel T01, die in Abbildung 4-3 wiedergegeben ist.

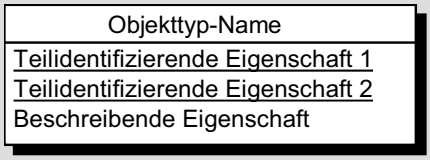
Schulbeispiel

Unser Datenmodell für das Schulbeispiel hatte in seiner endgültigen Fassung die Form, die in der Abbildung 4-4 noch einmal dargestellt wird.

Die Anwendung der Transformationsregel T01 führt zu dem in Abbildung 4-5 dargestellten Satz vorläufiger Tabellen-Typbeschreibungen, die wir in den weiteren Transformationsschritten komplettieren werden. Die Tabelle „Unterrichtsverpflichtung“ hat noch keine Attribute, weil der zugehörige Objekttyp keine Eigenschaften besitzt.

Transformationsregel T01 (Objekttyp):

Entity-Relationship-Modell		Relationales Datenbank-Modell
Objekttyp-Name	⇒	Tabellen-Bezeichnung
(Teil)identifizierende Eigenschaft	⇒	Primärschlüssel-Attribut
Beschreibende Eigenschaft	⇒	Spalten-Bezeichnung



Objekttyp-Name

Teilidentifizierende Eigenschaft 1	**Teilidentifizierende + Eigenschaft 2**	Beschreibende Eigenschaft

Abb. 4-3: Transformation eines Objekttyps[19]

4.2 Transformation von Beziehungstyp-Richtungen als identifizierende Elemente

Schwacher Objekttyp

Im Abschnitt 2.4.4 hatten wir den *schwachen Objekttyp* kennen gelernt. Dabei handelt es sich um einen Objekttyp, dessen Eigenschaften es nicht erlauben, jedes seiner Objekte eindeutig zu identifizieren. Um die Objekte dennoch zweifelsfrei voneinander

[19] Es sei noch einmal daran erinnert, dass die Elemente eines zusammengesetzten Primärschlüssels durch Pluszeichen verbunden, unterstrichen und **gemeinsam+im+Fettdruck** dargestellt werden. Jedes Element des zusammengesetzten Primärschlüssels ist dabei für sich ***nicht***-unikal.

unterscheiden zu können, wird die Beziehungstyp-Richtung, die den schwachen Objekttyp mit einem anderen Objekttyp verbindet, als identifizierendes Element für den *schwachen Objekttyp* verwendet.

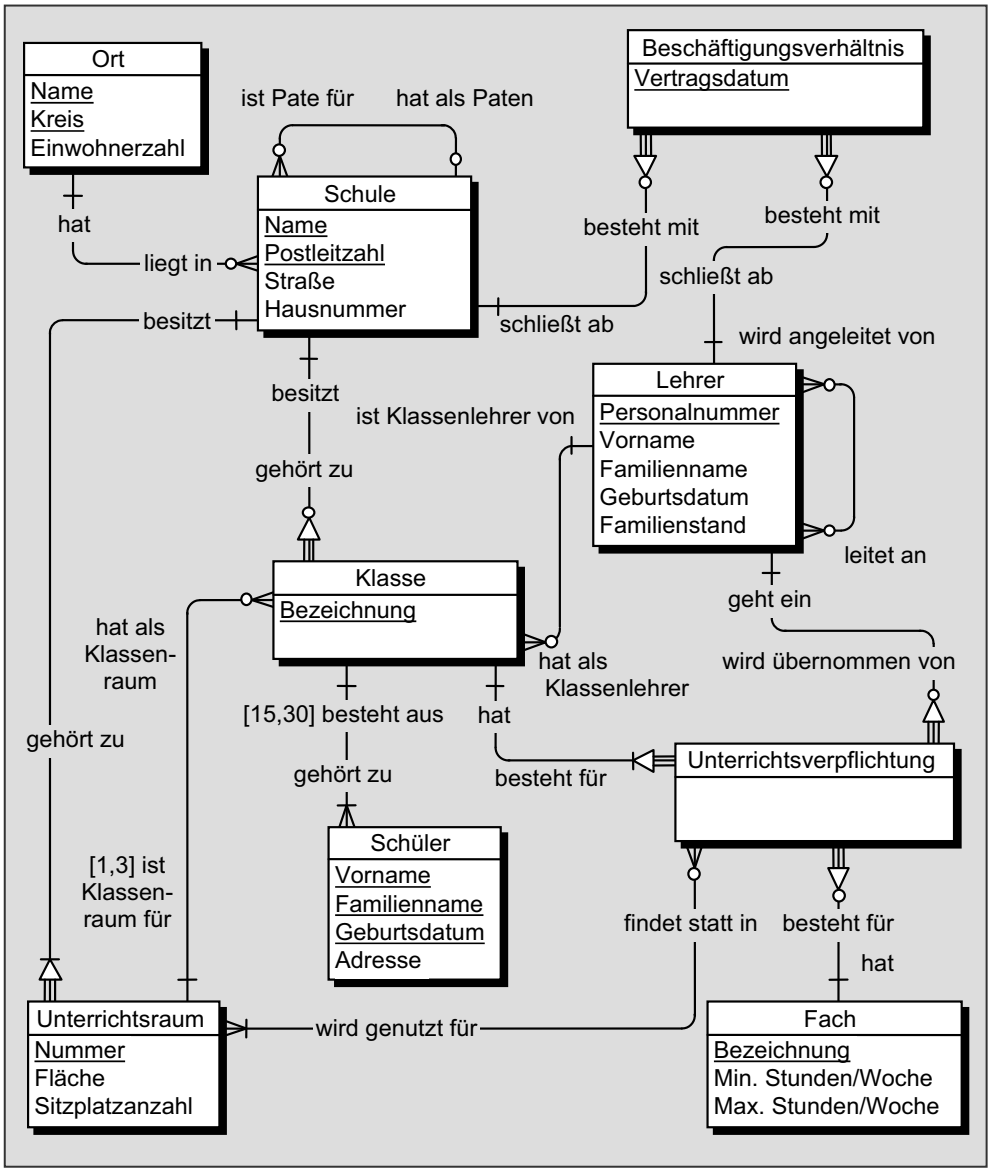


Abb. 4-4: Datenmodell für das Schulbeispiel

Ort(**Name+Kreis**,Einwohnerzahl)
Schule(**Name+Postleitzahl**,Straße,Hausnummer)
Beschäftigungsverhältnis(**Vertragsdatum**)
Lehrer(**Personalnummer**,Vorname,Familienname,
Geburtsdatum,Familienstand)
Klasse(**Bezeichnung**)
Schüler(**Vorname+Familienname+Geburtsdatum**,
Adresse)
Unterrichtsverpflichtung()
Fach(**Bezeichnung**,Min. Stunden/Woche,
Max. Stunden/Woche)
Unterrichtsraum(**Nummer**,Fläche,Sitzplatzanzahl)

Abb. 4-5: Tabellen-Typbeschreibungen für die Objekttypen

Transformationsregel T02

Will man also ein Objekt eines schwachen Objekttyps A identifizieren, so muss seine Beziehung zu einem Objekt des sachlogisch verbunden Objekttyps B berücksichtigt werden. In der Sprache des relationalen Datenbank-Modells heißt das aber: Der Primärschlüssel **SB** des Objekttyps B muss Bestandteil des Primärschlüssels des schwachen Objekttyps A werden. Dem entspricht die Transformationsregel T02, die in Abbildung 4-6 dargestellt ist.

Wir beschreiben im Folgenden die Veränderungen, die auf Grund der Transformationsregel T02 an den Tabellen-Typbeschreibungen unseres Schulbeispiels vorzunehmen sind. Im Interesse der Übersichtlichkeit nehmen wir dabei eine typographische Vereinfachung vor. Ein Fremdschlüssel wird – wie bisher stets – in die Verweispfeile eingeschlossen (⇑xyz⇑). Sollte der Fremdschlüssel seinerseits wieder Fremdschlüssel enthalten, dann werden für die „inneren“ Fremdschlüssel die Verweispfeile weggelassen:

Innerer Fremdschlüssel

statt: ⇑abc+⇑def⇑+ghi⇑

wird vereinfacht geschrieben: ⇑abc+def+ghi⇑

Transformationsregel T02 (Beziehungstyp-Richtung als identifizierendes Element):

Entity-Relationship-Modell		**Relationales Datenbank-Modell**
Objekttyp A mit der Beziehungstyp-Richtung „A zu B“ als identifizierendes Element	⇒	Der Primärschlüssel **SB** der Tabelle B wird als eingabepflichtiger nicht-unikaler Fremdschlüssel in A aufgenommen und als zusätzliches Element des Primärschlüssels von A vereinbart. **SA** und **⇑SB⇑** sind *gemeinsam* unikal.[20]

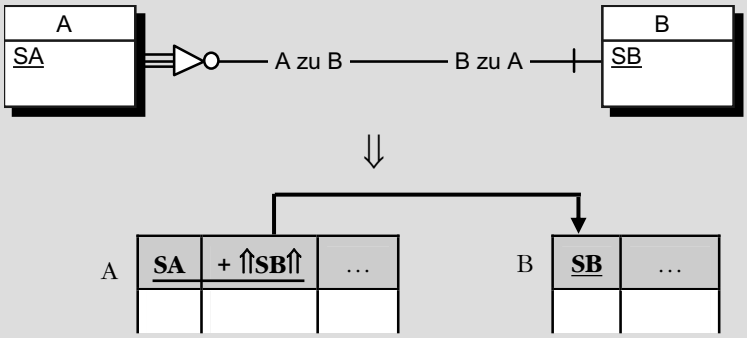


Abb. 4-6: Transformation einer Beziehungstyp-Richtung als identifizierendes Element

[20] Im eher selten auftretenden Fall, dass der Objekttyp A keine *eigenen* teilidentifizierenden Eigenschaften hat, dass der Objekttyp A also *ausschließlich* durch die Beziehungstyp-Richtung „A zu B“ identifiziert wird, bildet der Fremdschlüssel **⇑SB⇑** den *vollständigen* Primärschlüssel der Tabelle A und ist dann natürlich für sich allein *unikal*. Das kann jedoch nur bei 1:1-Beziehungstypen und C:1-Beziehungstypen vorkommen.

Beziehungstyp-Richtung: Beschäftigungsverhältnis zu Schule
Neue Typbeschreibung: Beschäftigungsverhältnis
(⇑**Schul-Name+Postleitzahl**⇑+
Vertragsdatum)

Beziehungstyp-Richtung: Beschäftigungsverhältnis zu Lehrer
Neue Typbeschreibung: Beschäftigungsverhältnis
(⇑**Personalnummer**⇑+
⇑**Schul-Name+Postleitzahl**⇑+
Vertragsdatum)

Beziehungstyp-Richtung: Klasse zu Schule
Neue Typbeschreibung: Klasse
(⇑**Schul-Name+Postleitzahl**⇑+**Bezeichnung**)

Beziehungstyp-Richtung: Unterrichtsverpflichtung zu Lehrer
Neue Typbeschreibung: Unterrichtsverpflichtung
(⇑**Personalnummer**⇑)

Beziehungstyp-Richtung: Unterrichtsverpflichtung zu Fach
Neue Typbeschreibung: Unterrichtsverpflichtung
(⇑**Fach-Bezeichnung**⇑+⇑**Personalnummer**⇑)

Beziehungstyp-Richtung: Unterrichtsverpflichtung zu Klasse
Neue Typbeschreibung: Unterrichtsverpflichtung
(⇑**Schul-Name+Postleitzahl+**
Klassen-Bezeichnung⇑+
⇑**Fach-Bezeichnung**⇑+⇑**Personalnummer**⇑)

Beziehungstyp-Richtung: Unterrichtsraum zu Schule
Neue Typbeschreibung: Unterrichtsraum
(⇑**Schul-Name+Postleitzahl**⇑+**Nummer**,
Fläche,Sitzplatzanzahl)

4.3 Transformation dualer Beziehungstypen

Im Abschnitt 3.4.1 wurde dargestellt, dass sich Beziehungstypen im relationalen Datenbank-Modell nur dadurch repräsentieren lassen, dass der Primärschlüssel einer Tabelle „gedoppelt“ und als Fremdschlüssel an „anderer Stelle“ aufgenommen wird. Wenn es sich bei dieser „anderen Stelle“ um eine andere Tabelle handelt, wird ein dualer Beziehungstyp dargestellt, der den sachlogischen Zusammenhang zwischen zwei Objekten aus verschiedenen Objekttypen beschreibt (vgl. Abschnitt 2.4.1). Liegt diese „andere Stelle“ dagegen in derselben Tabelle, aus der der Primärschlüssel stammt, dann wird ein Rekursiv-Beziehungstyp repräsentiert, der den sachlogischen Zusammenhang zwischen zwei Objekten widerspiegelt, die demselben Objekttyp angehören (vgl. Abschnitt 2.4.5). Wir wenden uns zunächst der Transformation dualer Beziehungstypen zu und behandeln die Rekursiv-Beziehungstypen im Abschnitt 4.4.

16 Klassen von Beziehungstypen

Im Abschnitt 2.4.1 wurden die Beziehungstypen in Klassen eingeteilt. Die Klassifizierung erfolgte in *zwei* Dimensionen mit je *zwei* Werten, die jeweils für *zwei* Beziehungstyp-Richtungen festgelegt wurden. Das ergab 2×2×2=16 Klassen. Als Dimensionen wurden verwendet:

1. die *Kardinalität* mit den Werten „1“ bzw. „N“ und
2. die *Optionalität* mit den Werten „optional“ (abgekürzt mit „o“) bzw. „nicht optional“ (abgekürzt mit „n o“).

Das führte zu der Matrix, die in Tabelle 4-1 noch einmal wiedergegeben ist.

Die unter der Diagonale liegenden – grau unterlegten - Klassen von Beziehungstypen müssen nicht untersucht werden, da sie durch ein Vertauschen der Rollen der beteiligten Objekttypen aus ihren an der Diagonale gespiegelten Partnern hervorgehen.

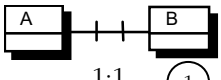
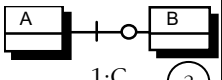
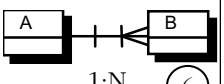
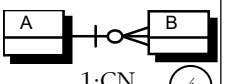
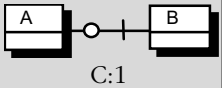
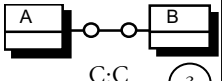
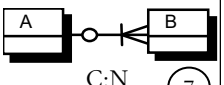
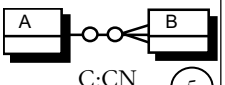
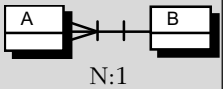
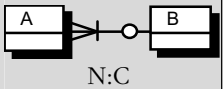
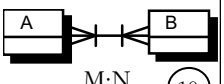
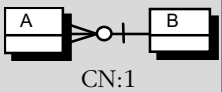
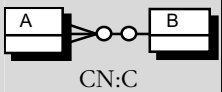
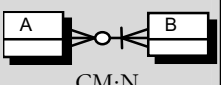
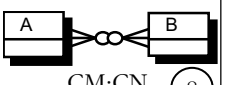
In den folgenden Abschnitten wird untersucht, wie die dualen Beziehungstypen in das relationale Datenbank-Modell transformiert werden können. Wir gehen dabei zunächst induktiv vor und betrachten jeden der 10 Beziehungstypen für sich. Die Reihenfolge der Betrachtung wird in der Tabelle 4-1 durch die *laufende Nummer in den Kreisen* angegeben. Es wird sich dabei zeigen, dass einige Elemente des Entity-Relationship-Modells nicht im relationalen Datenbank-Modell dargestellt werden können. Im Kapitel 5 wird deshalb deduktiv abgeleitet, welche zusätzlichen Eigenschaften das relationale Datenbank-Modell haben

müsste, um eine äquivalente Repräsentation der Beziehungstypen zu ermöglichen.

Tab. 4-1: 16 Klassen von Beziehungstypen mit 10 „symmetriefreien" Beziehungstypen

Spalte 1: Kardinalitäten 1 und N,

Spalte 2: nicht-optionale (n o) und optionale (o) Beziehungstyp-Richtungen

A→B / B→A		Kardinalität 1		Kardinalität N	
		nicht-optional	optional	nicht-optional	optional
1	n o	 1:1 (1)	 1:C (2)	 1:N (6)	 1:CN (4)
	o	 C:1	 C:C (3)	 C:N (7)	 C:CN (5)
N	n o	 N:1	 N:C	 M:N (10)	 M:CN (9)
	o	 CN:1	 CN:C	 CM:N	 CM:CN (8)

Für die Transformation eines Beziehungstyps in eine Tabellen-Struktur gibt es mitunter mehrere Möglichkeiten. Wir werden im Folgenden nur diejenigen angeben, die von besonderer praktischer Bedeutung sind. Mitunter ist aber eine Auswahl unter mehreren sinnvollen Varianten zu treffen. Um nun eine Entscheidungshilfe dafür zu geben, welche der in Frage kommenden Umsetzungsmethoden man im gegebenen Fall anwenden sollte, wird zusätzlich zur Transformationsregel der Speicherbedarf angegeben, der für die Darstellung des Beziehungstyps benötigt wird.

4.3.1 Der 1:1-Beziehungstyp

Beim 1:1-Beziehungstyp ist jedes Objekt des Objekttyps A mit genau einem Objekt des Objekttyps B verbunden und umgekehrt. Je ein A-Objekte und ein B-Objekt gehen eine feste Paarung ein. Die Abbildung 4-7 zeigt dies schematisch.

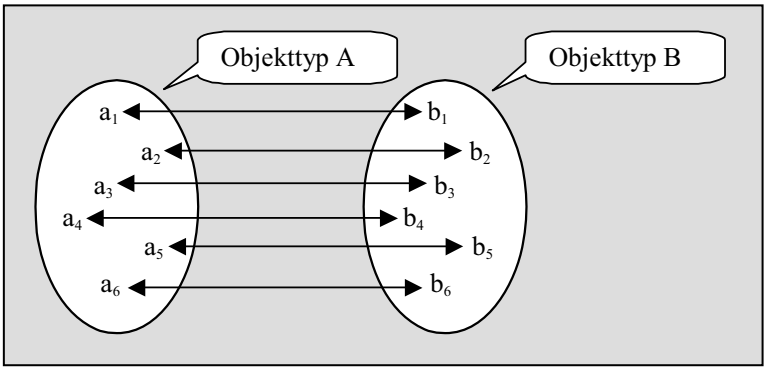


Abb. 4-7: 1:1-Beziehungstyp

Überflüssiger 1:1-Beziehungstyp

1:1-Beziehungstypen sind bereits im konzeptionellen Datenmodell kritisch zu betrachten, weil sie meist überflüssig sind. Ist nämlich jedes Objekt des Objekttyps A mit genau einem Objekt des Objekttyps B - und umgekehrt - verbunden, dann bilden die beiden Objekte eine derart feste Kopplung, dass sie in der Regel als ein einziges – komplexeres - Objekt betrachtet werden können.

Transformationsregel T03

Ist ein überflüssiger 1:1-Beziehungstyp zwischen den Objekttypen A und B noch nicht im konzeptionellen Datenmodell beseitigt worden, dann wird er bei der Umsetzung in das relationale Datenbank-Modell eliminiert. Dazu werden alle Attribute von B in die Tabelle A eingefügt. Die 1:1-Kopplung wird dadurch erzwungen, dass der Schlüssel von B in der Tabelle A als eingabepflichtig und unikal deklariert wird. Ein B-Objekt kann nun nicht losgelöst von „seinem“ A-Objekt gespeichert werden, und zu jedem A-Objekt muss es genau ein B-Objekt geben, für das zumindest seine identifizierenden Attributwerte anzugeben sind. Diese Transformationsregel T03 ist in Abbildung 4-8 wiedergegeben.

Transformationsregel T03 (überflüssiger 1:1-Beziehungstyp):

Entity-Relationship-Modell		**Relationales Datenbank-Modell**
Objekttyp A mit Schlüssel SA	⇒	Tabelle A mit Primärschlüssel **SA**
Objekttyp B mit Schlüssel SB	⇒	Alle Eigenschaften werden in A aufgenommen. **SB** wird als eingabepflichtig und unikal vereinbart
1:1-Beziehungstyp	⇒	wird nicht gesondert dargestellt
Speicherbedarf für Beziehungstyp		0

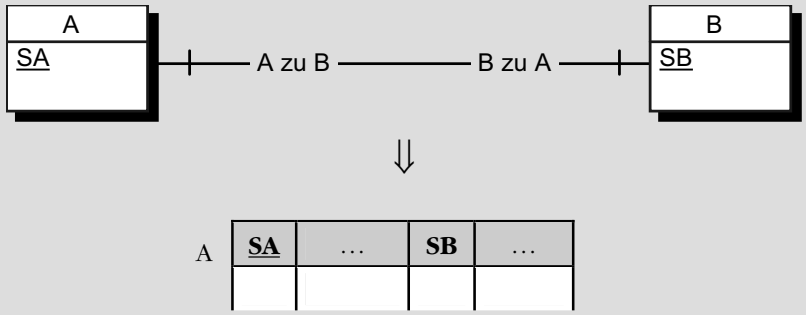


Abb. 4-8: Transformation eines überflüssigen 1:1-Beziehungstyps

Beispiel für Transformationsregel T03

Als ein Beispiel für einen überflüssigen 1:1-Beziehungstyp betrachten wir die Mitarbeiter eines Unternehmens, von denen jeder genau einen Dienstausweis besitzt. Ein gegebener Dienstausweis ist natürlich für genau einen Mitarbeiter ausgestellt. Die erforderliche Tabellenrepräsentation gemäß der Transformationsregel T03 zeigt die Abbildung 4-9.

Da das Attribut „Ausweisnummer“ eingabepflichtig ist, muss jeder gespeicherte Mitarbeiter einen Ausweis haben. Andererseits ist das Attribut unikal, so dass eine Ausweisnummer nur einem Mitarbeiter zugeordnet sein kann.

Sinnvoller 1:1-Beziehungstyp

Mitunter kann ein 1:1-Beziehungstyp aber auch sinnvoll sein. Will man beispielsweise die Informationen über Objekte in öffentliche und vertrauliche Daten unterteilen, kann man zwei Objekttypen A und B mit demselben Schlüssel S ins Datenmodell

Transformationsregel T04

aufnehmen. Der Schlüssel S wird sowohl in A als auch in B als unikaler eingabepflichtiger Fremdschlüssel vereinbart[21]. Abbildung 4-10 zeigt die entsprechende Transformationsregel T04.

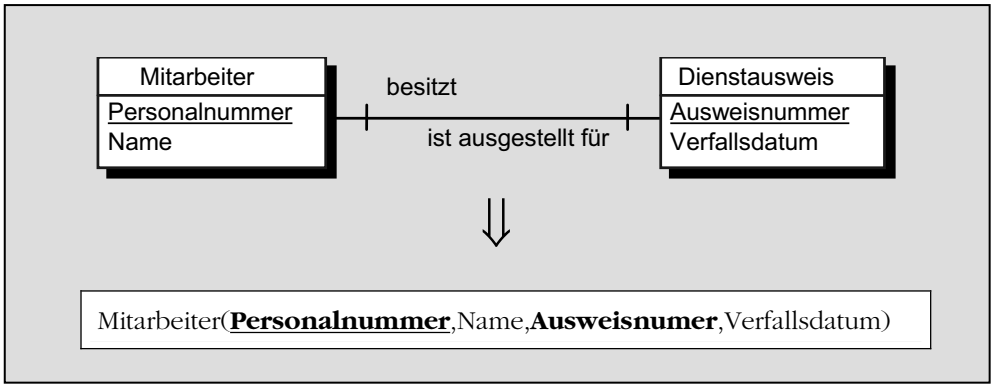


Abb. 4-9: Beispiel für die Transformationsregel T03

In der Formel für den Speicherbedarf ist $\overline{\overline{B}}$ die Mächtigkeit des Objekttyps B, also die Anzahl der Objekte, die in B zusammengefasst sind. Das entspricht der Anzahl der Zeilen in der Tabelle B. Der Speicherbedarf entsteht dadurch, dass für jedes Objekt im Objekttyp B der Schlüssel S mit seiner Länge *Len*(S) gespeichert werden muss. Würde man beide Objekttypen in einer einzigen Tabelle speichern, könnten diese Angaben entfallen.

Beim 1:1-Beziehungstyp kann ein neues Objekt weder allein in A noch allein in B gespeichert werden. Das würde ja auch der Forderung nach Nicht-optionalität beider Beziehungstyp-Richtungen widersprechen. Es muss vom Anwendungsprogramm im Rahmen einer *Transaktion*[22] gesichert werden, dass zu einem

[21] Unikal und eingabepflichtig ist S ohnehin, weil S für beide Tabellen der Primärschlüssel ist.

[22] Eine Transaktion ist eine Folge von Operationen in einer Datenbank, bei der gesichert wird, dass entweder alle Operationen fehlerfrei beendet werden oder dass keine der Operationen ausgeführt wird. Transaktionen werden bei der Datenbankarbeit so zugeschnitten, dass jeweils vor ihrem Beginn und nach ihrem Ende ein konsistenter Zustand der Datenbank vorliegt.

neuen Schlüsselwert gemeinsam je eine Zeile in die Tabellen A und B eingetragen wird.

Transformationsregel T04 (sinnvoller 1:1-Beziehungstyp):

Entity-Relationship-Modell		Relationales Datenbank-Modell
Objekttyp A mit Schlüssel S	⇒	Tabelle A mit Primärschlüssel **S**
Objekttyp B mit Schlüssel S	⇒	Tabelle B mit Primärschlüssel **S**
1:1-Beziehungstyp	⇒	**S** wird sowohl in A als auch in B als unikaler eingabepflichtiger Fremdschlüssel vereinbart
Speicherbedarf für Beziehungstyp		$\overline{\overline{B}} \cdot Len(S)$

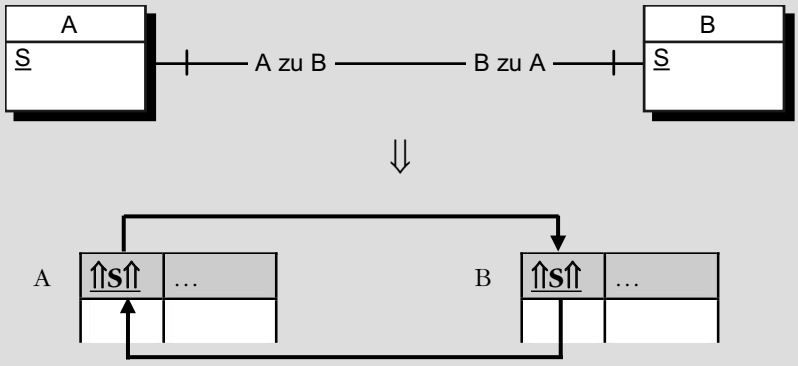


Abb. 4-10: Transformation eines sinnvollen 1:1-Beziehungstyps

Beispiel für Transformationsregel T04

Beispielsweise kann man das Gehalt eines Mitarbeiters getrennt von seinen sonstigen Eigenschaften in einer anderen Tabelle speichern wollen, wie dies in der Abbildung 4-11 dargestellt ist.

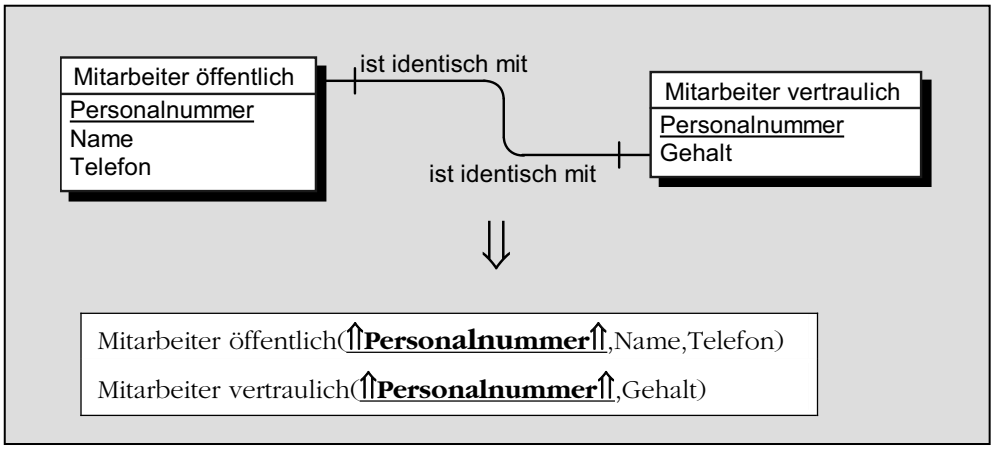


Abb. 4-11: Beispiel für die Transformationsregel T04

Man hat nun die Möglichkeit, für die beiden Tabellen unterschiedliche Zugriffsrechte zu erteilen. Da die unikalen Fremdschlüssel **⇑Personalnummer⇑** jeweils auf die andere Tabelle verweisen, muss es nach den Regeln der referenziellen Integrität zu jedem Wert des Fremdschlüssels genau einen Datensatz in der anderen Tabelle geben. Das realisiert den 1:1-Beziehungstyp.

4.3.2 Der 1:C-Beziehungstyp

Beim 1:C-Beziehungstyp kann jedes Objekt des Objekttyps A mit höchstens einem Objekt des Objekttyps B verbunden sein. Ein Objekt aus B muss aber stets mit einem Objekt aus A gekoppelt sein.

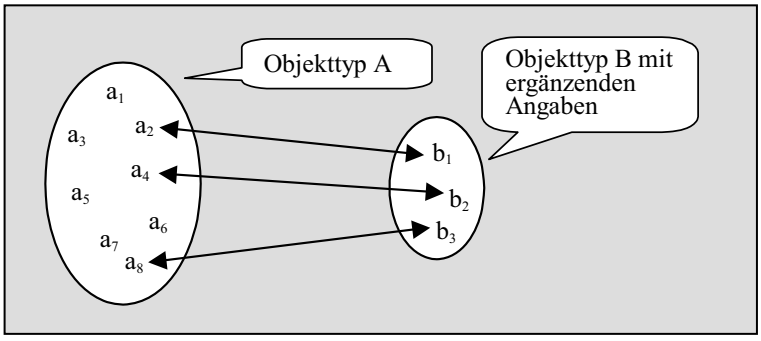


Abb. 4-12: 1:C-Beziehungstyp

Die Informationen, die im B-Objekt gespeichert sind, können somit als *fakultative ergänzende Angaben* zum A-Objekt interpretiert werden. Die Abbildung 4-12 soll dies verdeutlichen.

Transformationsregel T05 (1:C-Beziehungstyp mit selten realisierter Optionalität):

Entity-Relationship-Modell		**Relationales Datenbank-Modell**
Objekttyp A mit Schlüssel SA	⇒	Tabelle A mit Primärschlüssel **SA**
Objekttyp B mit Schlüssel SB	⇒	Alle Eigenschaften von B werden als nicht-eingabepflichtige Attribute in A aufgenommen. SB wird außerdem als unikal vereinbart.
1:C-Beziehungstyp	⇒	wird nicht gesondert dargestellt
Speicherbedarf für Beziehungstyp		$\left(\overline{\overline{A}} - \overline{\overline{B}}\right) \cdot Len(B)$

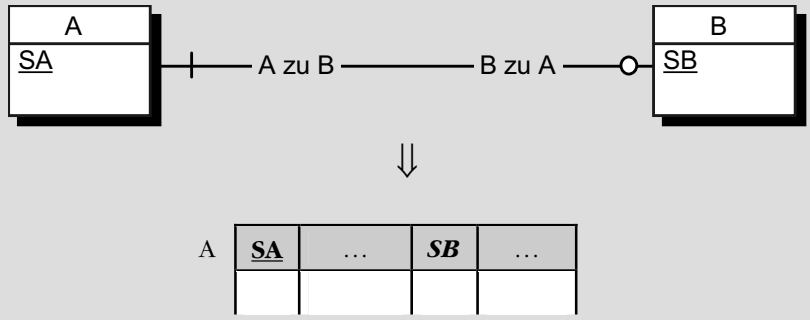


Abb. 4-13: Transformation eines 1:C-Beziehungstyps mit selten realisierter Optionalität

Die Art der Transformation richtet sich nun danach, wie hoch der Anteil jener A-Objekte ist, für die ergänzende Angaben gemacht werden, die also in Beziehung zu einem B-Objekt stehen. Das ist äquivalent zur Frage, ob der Objekttyp A wesentlich mehr Objekte enthält als der Objekttyp B.

Transformationsregel T05

Gibt es bei einem 1:C-Beziehungstyp nur unwesentlich mehr A-Objekte als B-Objekte, können die Daten beider Objekttypen in einer Tabelle zusammengefasst werden. Die Eigenschaften von B werden dabei als nicht-eingabepflichtig deklariert. Der Schlüssel SB von B wird in der Tabelle A als unikal deklariert. Ein B-Objekt, das damit nur zu einem einzigen A-Objekt gehören kann, kann nicht losgelöst von „seinem" A-Objekt gespeichert werden. Bei den wenigen A-Objekten, die nicht mit einem B-Objekt verbunden sind, werden sämtliche B-Attribute mit der NULL-Marke belegt. Die Transformationsregel T05 ist in Abbildung 4-13 wiedergegeben.

Der Speicherbedarf entsteht dadurch, dass für sämtliche A-Objekte, die kein B-Objekt als Partner haben, alle Felder des Objekttyps B mit ihrer Gesamtlänge *Len*(B) leer bleiben.

Beispiel für Transformationsregel T05

Werden beispielsweise fast alle Mitarbeiter eines Unternehmens mit genau einem Handy ausgestattet, so sind die ergänzenden Angaben für das Handy bei nahezu allen Mitarbeitern erforderlich. Die entsprechende Transformation zeigt die Abbildung 4-14.

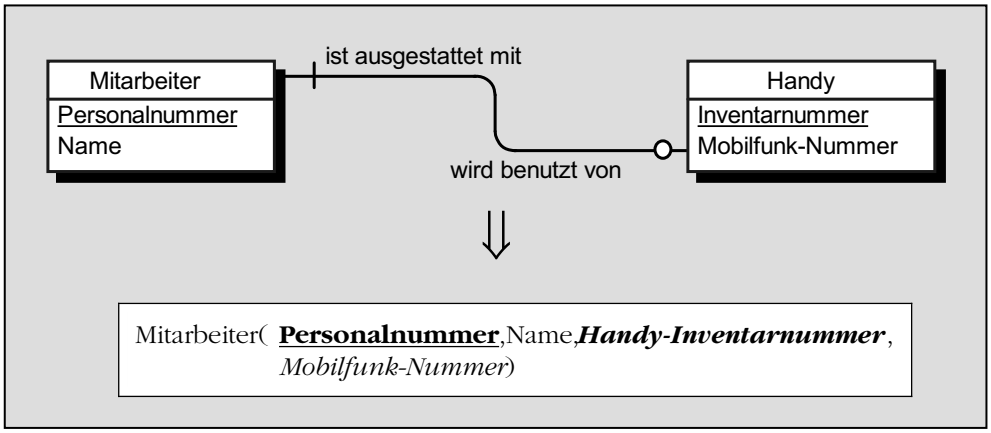


Abb. 4-14: Beispiel für die Transformationsregel T05

Die ursprüngliche Eigenschaftsbezeichnung „Inventarnummer" wurde durch den Zusatz "Handy" in ihrem neuen Kontext „sprechender" gewählt. Die „Handy-Spalten" der Tabelle „Mitarbeiter" sind nicht-eingabepflichtig, d.h. sie werden für die wenigen Mitarbeiter, die nicht mit einem Handy ausgestattet sind, mit der NULL-Marke belegt. Da das Attribut ***„Handy-Inventarnummer"***

als unikal deklariert ist, kann ein und dasselbe Handy nicht mehreren Mitarbeitern zugeordnet werden.

Transformationsregel T06

Betrachten wir nun den Fall, dass es wesentlich mehr A-Objekte als B-Objekte gibt. Würde man jetzt die beiden Objekttypen in einer Tabelle vereinen, wären die Attribute der B-Objekte in vielen Zeilen mit der NULL-Marke belegt. Um das zu vermeiden, werden zwei Tabellen angelegt: eine A-Tabelle für alle A-Objekte und eine B-Tabelle für die seltenen B-Objekte, die durch einen eingabepflichtigen Fremdschlüssel jeweils auf „ihr“ A-Objekt verweisen. Die Kardinalität „1“ in der Beziehungstyp-Richtung „B zu A“ wird dadurch erzwungen, dass der Fremdschlüssel in B als unikal deklariert wird. Diese Umsetzung entspricht der Transformationsregel T06, die in Abbildung 4-15 dargestellt ist.

Transformationsregel T06 (1:C-Beziehungstyp mit oft realisierter Optionalität):

Entity-Relationship-Modell		**Relationales Datenbank-Modell**
Objekttyp A mit Schlüssel SA	⇒	Tabelle A mit Primärschlüssel **SA**
Objekttyp B mit Schlüssel SB	⇒	Tabelle B mit Primärschlüssel **SB**
1:C-Beziehungstyp	⇒	**SA** wird in B als eingabepflichtiger unikaler Fremdschlüssel aufgenommen
Speicherbedarf für Beziehungstyp		$\overline{=}_{B} \cdot Len(\mathbf{SA})$

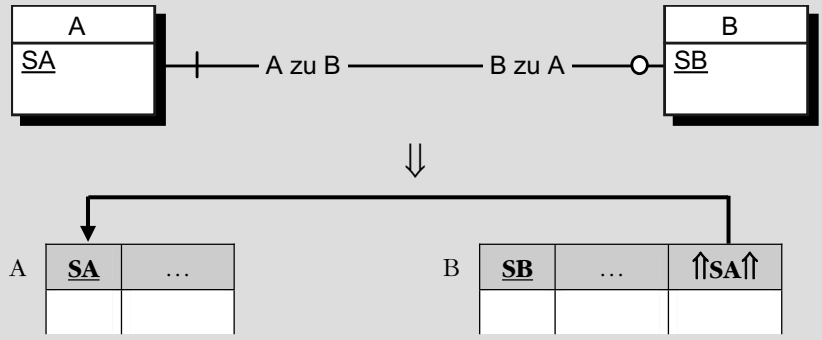


Abb. 4-15: Transformation eines 1:C-Beziehungstyps mit oft realisierter Optionalität

Der Speicherbedarf für den 1:C-Beziehungstyp entsteht dadurch, dass bei jedem Objekt des Objekttyps B ein Verweis mit der Länge *Len*(SA) auf das zugehörige A-Objekt zu hinterlegen ist.

Beispiel für Transformationsregel T06

Soll beispielsweise festgehalten werden, welcher Mitarbeiter ein - und höchstens ein – Projekt leitet, wobei für jedes Projekt ein Mitarbeiter verantwortlich ist, so wird es viele Mitarbeiter ohne Projektleitung geben. Deshalb ist in diesem Fall die Transformationsregel T06 anzuwenden, wie dies in Abbildung 4-16 geschehen ist.

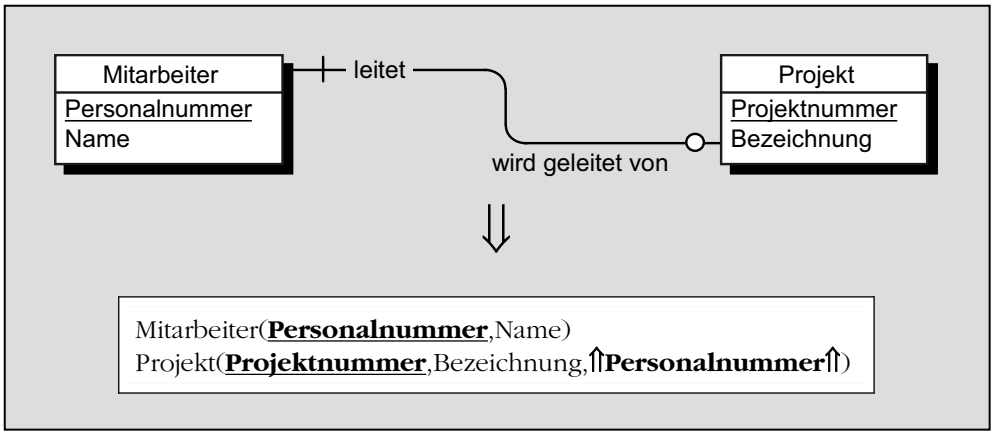


Abb. 4-16: Beispiel für die Transformationsregel T06

Da der Fremdschlüssel ⇑**Personalnummer**⇑ eingabepflichtig ist, muss jedes Projekt genau einen Projektleiter haben. Andererseits ist der Fremdschlüssel unikal, so dass ein Mitarbeiter nur für ein Projekt als Leiter ausgewiesen sein kann. Da man nicht sichern kann, dass jede Personalnummer eines Mitarbeiters auch tatsächlich als Wert des Fremdschlüssels auftritt, kann es Mitarbeiter geben, die mit keinem Projekt als Leiter verbunden sind.

4.3.3 Der C:C-Beziehungstyp

Beim C:C-Beziehungstyp ist jedes Objekt des Objekttyps A mit keinem oder einem Objekt des Objekttyps B verbunden und umgekehrt. Es gibt also A-Objekte ohne B-Partner und umgekehrt. Jedes der Objekte kann aber höchstens einen Partner haben. Abbildung 4-17 zeigt dies schematisch.

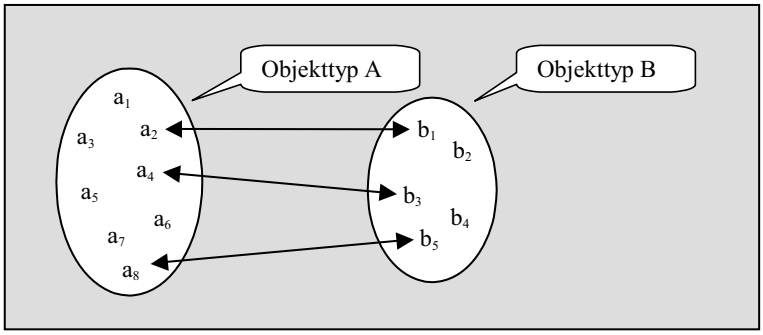


Abb. 4-17: C:C-Beziehungstyp

Wir nehmen für unsere Betrachtungen – ohne Beschränkung der Allgemeinheit - an, dass es höchstens so viele B-Objekte gibt wie A-Objekte (andernfalls müssten die Objekttypen einfach die Seiten tauschen). Da sowohl die A-Objekte als auch die B-Objekte unabhängig voneinander existieren können, müssen sie jeweils in einer eigenen Tabelle gespeichert werden. Die Art der Darstellung des Beziehungstyps richtet sich aber danach, wie hoch der Anteil jener B-Objekte ist, die nicht in Beziehung zu einem A-Objekt stehen. Das ist gleichbedeutend mit der Frage, ob es wenige oder viele Partnerbeziehungen zwischen A und B gibt.

Transformationsregel T07

Kommt es nur selten vor, dass ein B-Objekt keinen Partner im Objekttyp A hat, ist es sinnvoll, bei den B-Objekten einen Verweis auf „ihren“ A-Partner zu hinterlegen. Bei den „partnerlosen“ B-Objekten wird dieser Verweis dann mit der NULL-Marke belegt. Deshalb wird der Primärschlüssel von A in der Tabelle B als nicht-eingabepflichtiger Fremdschlüssel aufgenommen. Fordert man für den Fremdschlüssel außerdem die Unikalität, kann auf ein A-Objekt nur von höchstens einem B-Objekt aus verwiesen werden. Die Transformationsregel T07 ist in Abbildung 4-18 angegeben.

Der Speicherbedarf entsteht dadurch, dass bei jedem B-Objekt ein Feld für den Fremdschlüssel ⇑***SA***⇑ vorgesehen werden muss.

Beispiel für Transformationsregel T07

Als Beispiel nehmen wir an, dass viele Mitarbeiter im Unternehmen ihren eigenen Schreibtisch haben. Werkstatt-Mitarbeiter besitzen aber keinen Schreibtisch. Die meisten im Unternehmen registrierten Schreibtische wurden für einen Mitarbeiter bereitgestellt, einige stehen jedoch zur Reserve im Keller. Die Tabellen-

repräsentation entsprechend der Transformationsregel T07 zeigt die Abbildung 4-19.

Transformationsregel T07 (C:C-Beziehungstyp mit vielen Partnerbeziehungen):

Entity-Relationship-Modell		**Relationales Datenbank-Modell**
Objekttyp A mit Schlüssel SA	⇒	Tabelle A mit Primärschlüssel **SA**
Objekttyp B mit Schlüssel SB	⇒	Tabelle B mit Primärschlüssel **SB**
C:C-Beziehungstyp	⇒	**SA** wird in B als nicht-eingabepflichtiger unikaler Fremdschlüssel aufgenommen
Speicherbedarf für Beziehungstyp		$\overline{\overline{B}} \cdot Len(\mathbf{SA})$

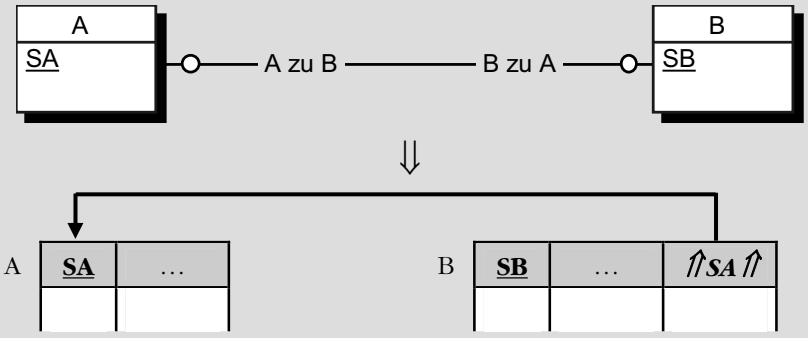


Abb. 4-18: Transformation eines C:C-Beziehungstyps mit vielen Partnerbeziehungen

Bei den „herrenlosen“ Schreibtischen wird der nicht-eingabepflichtige Fremdschlüssel ***⇑Personalnummer⇑*** mit der NULL-Marke belegt. Wegen der Unikalität dieses Fremdschlüssels kann einem Mitarbeiter höchstens ein Schreibtisch zugeordnet sein.

Diese Form der Repräsentation eines C:C-Beziehungstyps wird allerdings dann ungünstig, wenn jeweils nur wenige B-Objekte mit einem A-Objekt gepaart sind, weil dann bei vielen B-Objekten der Fremdschlüssel mit der NULL-Marke belegt ist. In diesem Fall ist es besser, die wenigen Paarungen in einer dritten Koppel-Tabelle A/B darzustellen.

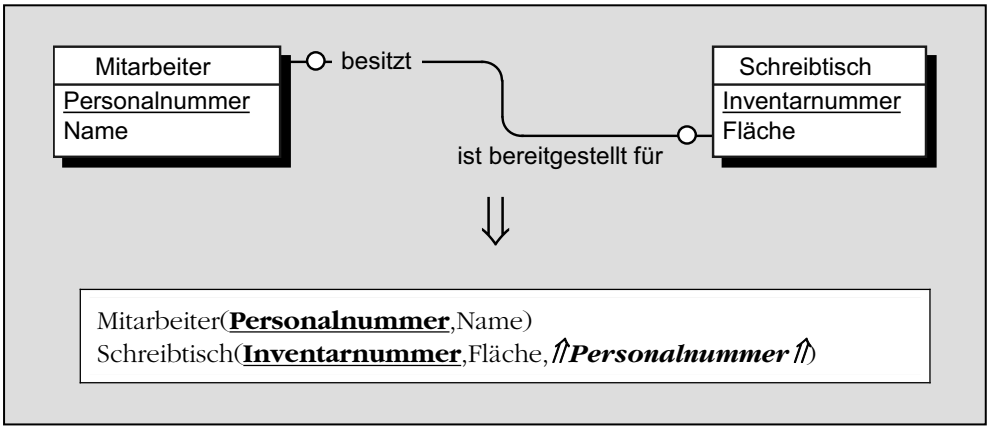


Abb. 4-19: Beispiel für die Transformationsregel T07

Koppel-Objekttyp

Die Repräsentation des C:C-Beziehungstyps in einer eigenen Tabelle entspricht einer Transformation im Datenmodell, die bereits im Abschnitt 2.6.2 beschrieben wurde. Dort wurde ein *Koppel-Objekttyp* eingeführt, um einem Beziehungstyp Eigenschaften zuweisen zu können. Hier erfolgt sie ausschließlich aus technischen Gründen, um nämlich die Belegung des Fremdschlüssels mit der NULL-Marke zu vermeiden. Die Transformation des C:C-Beziehungstyps in einen 1:C- und einen C:1-Beziehungstyp erfolgt nach dem Muster, das in Abbildung 4-20 dargestellt ist.

Man erkennt, dass die Optionalität/Kardinalität der Beziehungstyp-Richtungen „A zu B“ bzw. „B zu A“ auf die Beziehungstyp-Richtungen „A zu A/B“ bzw. „B zu A/B“ übertragen wurden. Die Identifizierung des Koppel-Objekttyps A/B kann entweder durch die Beziehungstyp-Richtung „A/B zu A“ oder durch die Beziehungstyp-Richtung „A/B zu B“ erfolgen. In der Abbildung 4-20 wurde die Beziehungstyp-Richtung „A/B zu A“ gewählt.

Transformationsregel T08

Stellt man den neugebildeten Objekttyp A/B im relationalen Datenbank-Modell als eine zusätzliche Koppel-Tabelle dar, so verweist eine Zeile dieser Tabelle sowohl auf ein A-Objekt als auch auf ein B-Objekt, enthält also die Primärschlüssel beider Tabellen jeweils als eingabepflichtigen unikalen Fremdschlüssel. Als Primärschlüssel von A/B kann einer der beiden Fremdschlüssel dienen. Die Transformationsregel T08 ist in Abbildung 4-21 dargestellt.

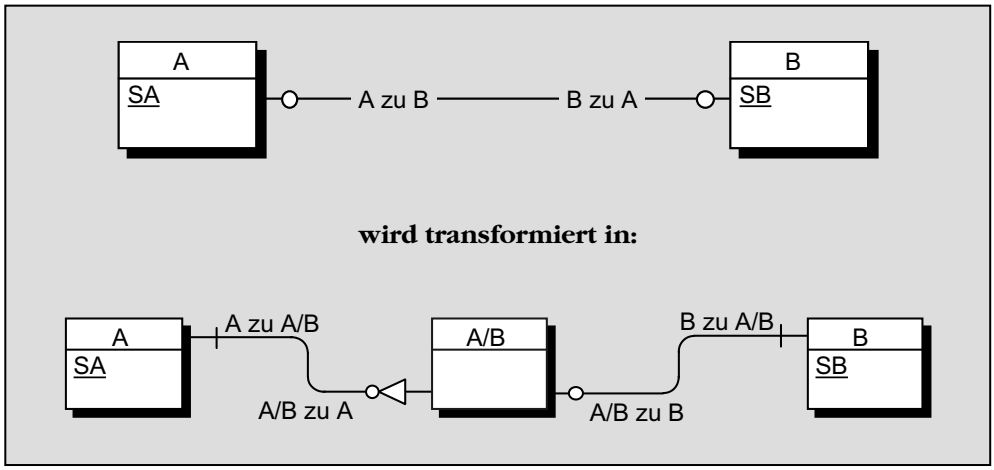


Abb. 4-20: Umwandlung eines C:C-Beziehungstyps in einen 1:C- und einen C:1-Beziehungstyp

Die Koppel-Tabelle A/B repräsentiert den Beziehungstyp als Menge der Paarungen eines A-Objekts mit einem B-Objekt. $\overline{\overline{A/B}}$ ist die Mächtigkeit des Beziehungstyps, also die Anzahl der Paarungen. Das Produkt aus der Anzahl dieser Paarungen und der Gesamtlänge beider Fremdschlüssel drückt den Speicherbedarf für den C:C-Beziehungstyp aus.

Beispiel für Transformationsregel T08

Beispielsweise möge es einige wenige Mitarbeiter geben, die jeweils über einen personengebundenen Dienstwagen verfügen. Andererseits gibt es aber sehr viele Dienstwagen, die an keinen Mitarbeiter persönlich gebunden sind. Aus Abbildung 4-22 ist die entsprechende Transformation zu ersehen.

Beide Fremdschlüssel ⇑**Personalnummer**⇑ und ⇑**Pol. Kennzeichen**⇑ sind jeweils für sich unikal[23]. Somit kann ein Mitarbeiter ebenso wie ein Dienstwagen nur an einer einzigen Paarung beteiligt sein. Jeder der beiden Fremdschlüssel ist damit als Primärschlüssel der Tabelle „Mitarbeiter/Dienstwagen“ geeignet. Wir haben uns im Beispiel willkürlich für den Fremdschlüssel ⇑**Personalnummer**⇑ entschieden.

[23] Wären sie nur gemeinsam unikal, würden sie durch das Pluszeichen miteinander verbunden sein.

Transformationsregel T08 (C:C-Beziehungstyp mit wenigen Partnerbeziehungen):

Entity-Relationship-Modell		**Relationales Datenbank-Modell**
Objekttyp A mit Schlüssel SA	⇒	Tabelle A mit Primärschlüssel **SA**
Objekttyp B mit Schlüssel SB	⇒	Tabelle B mit Primärschlüssel **SB**
C:C-Beziehungstyp	⇒	Koppel-Tabelle A/B mit **SA** und **SB** als jeweils eingabepflichtiger unikaler Fremdschlüssel. Entweder ⇑**SA**⇑ oder ⇑**SB**⇑ bildet den Primärschlüssel von A/B.
Speicherbedarf für Beziehungstyp		$\overline{\overline{A/B}} \cdot \left(Len(SA) + Len(SB) \right)$

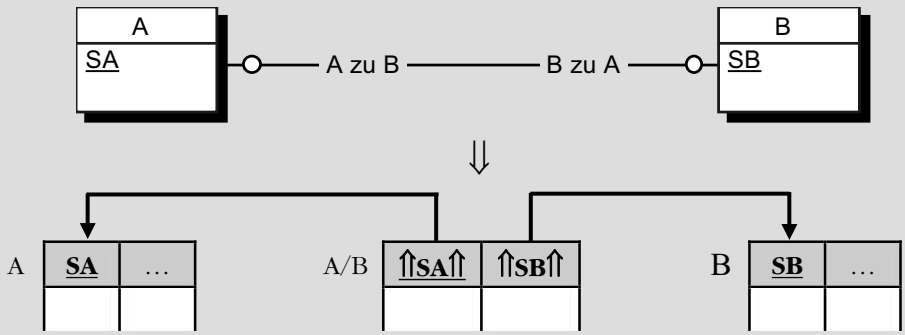


Abb. 4-21: Transformation eines C:C-Beziehungstyps mit wenigen Partnerbeziehungen

Nicht jeder Wert des Primärschlüssels **Personalnummer** in der Tabelle „Mitarbeiter“ muss zwangsläufig auch als Wert des Fremdschlüssels ⇑**Personalnummer**⇑ in der Koppel-Tabelle „Mitarbeiter/Dienstwagen“ auftreten. Das trifft auf jene Mitarbeiter zu, die nicht mit einem Dienstwagen gekoppelt sind. Analoges gilt natürlich auch für die Dienstwagen.

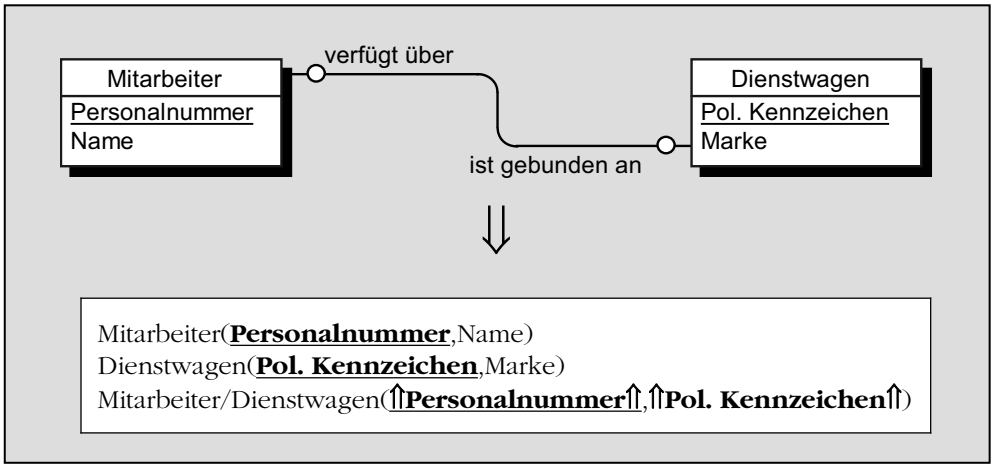


Abb. 4-22: Beispiel für die Transformationsregel T08

4.3.4 Der 1:CN-Beziehungstyp

Beim 1:CN-Beziehungstyp ist jedes Objekt des Objekttyps A mit keinem, einem oder mehreren Objekten des Objekttyps B verbunden. Ein B-Objekt ist dagegen immer mit genau einem A-Objekt gekoppelt. Abbildung 4-23 zeigt dies schematisch.

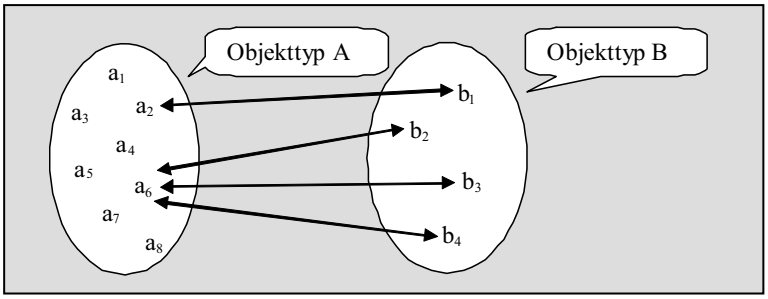


Abb. 4-23: 1:CN-Beziehungstyp

Transformationsregel T09

Der Beziehungstyp wird im relationalen Datenbank-Modell durch je eine Tabelle für die A-Objekte und die B-Objekte repräsentiert, wobei der Primärschlüssel SA von A als eingabepflichtiger Fremdschlüssel in B aufgenommen wird. Jedes B-Objekt muss

dann auf genau ein A-Objekt verweisen. Da der Fremdschlüssel aber nicht als unikal vereinbart ist, können mehrere B-Objekte mit demselben A-Objekt gekoppelt sein. Die entsprechende Transformationsregel T09 zeigt die Abbildung 4-24.

Transformationsregel T09 (1:CN-Beziehungstyp):

Entity-Relationship-Modell		**Relationales Datenbank-Modell**
Objekttyp A mit Schlüssel SA	⇒	Tabelle A mit Primärschlüssel **SA**
Objekttyp B mit Schlüssel SB	⇒	Tabelle B mit Primärschlüssel **SB**
1:CN-Beziehungstyp	⇒	**SA** wird in B als eingabepflichtiger nicht-unikaler Fremdschlüssel aufgenommen
Speicherbedarf für Beziehungstyp		$\overline{\overline{B}} \cdot Len(SA)$

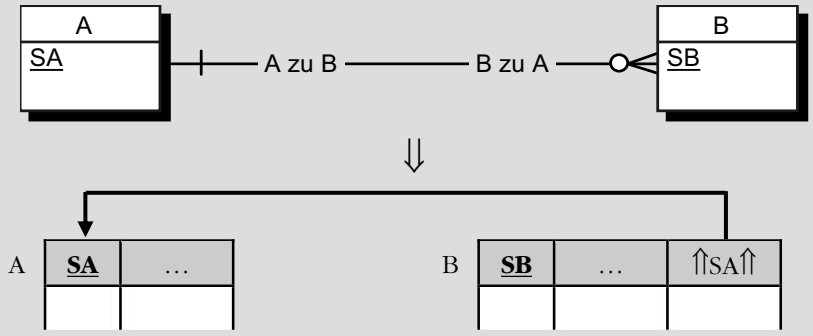


Abb. 4-24: Transformation eines 1:CN-Beziehungstyps

Bei der Angabe des Speicherbedarfs wurde berücksichtigt, dass bei jedem B-Objekt der Fremdschlüssel ⇑SA⇑ abgelegt wird.

Beispiel für Transformationsregel T09

Beispielsweise kann eine spezielle Gehaltsgruppe noch keinem, erst einem Mitarbeiter oder bereits mehreren Mitarbeitern zugeordnet sein. Andererseits wird jeder Mitarbeiter in genau eine Gehaltsgruppe eingeordnet. Die Abbildung 4-25 zeigt die erforderliche Transformation.

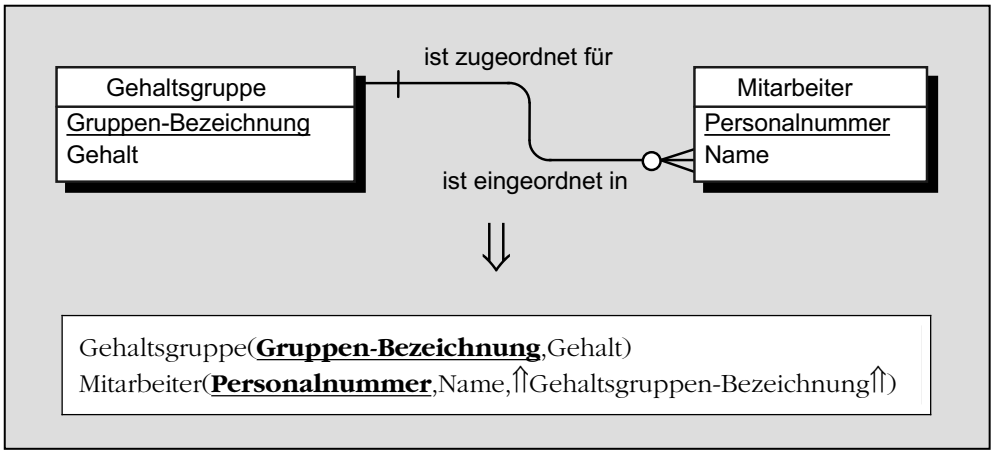


Abb. 4-25: Beispiel für die Transformationsregel T09

Da der Fremdschlüssel ⇑Gehaltsgruppen-Bezeichnung⇑ eingabepflichtig ist, muss jedem Mitarbeiter genau eine Gehaltsgruppe zugeordnet werden. Andererseits ist der Fremdschlüssel nicht-unikal, so dass mehrere Mitarbeiter auf dieselbe Gehaltsgruppe verweisen können. Da nicht zu fordern ist, dass jeder Wert des Primärschlüssels **Gruppen-Bezeichnung** auch tatsächlich als ein Wert des Fremdschlüssels ⇑Gehaltsgruppen-Bezeichnung⇑ auftreten muss, kann es Gehaltsgruppe geben, auf die (noch) kein Mitarbeiter verweist.

4.3.5 Der C:CN-Beziehungstyp

Beim C:CN-Beziehungstyp kann ein A-Objekt mit keinem, einem oder mehreren B-Objekten gekoppelt sein. Ein B-Objekt kann jedoch nur zu höchstens einem A-Objekt in Beziehung stehen. Die Situation zeigt Abbildung 4-26.

Die Transformation eines C:CN-Beziehungstyps in eine Tabellenstruktur hängt davon ab, wie viele B-Objekte an ein A-Objekt gebunden sind.

Transformationsregel T10

Betrachten wir zunächst den Fall, dass die meisten B-Objekte mit einem A-Objekt gepaart sind. Dann wird der Primärschlüssel von A als nicht-eingabepflichtiger und nicht-unikaler Fremdschlüssel in B aufgenommen. Abbildung 4-27 zeigt die entsprechende Transformationsregel T10.

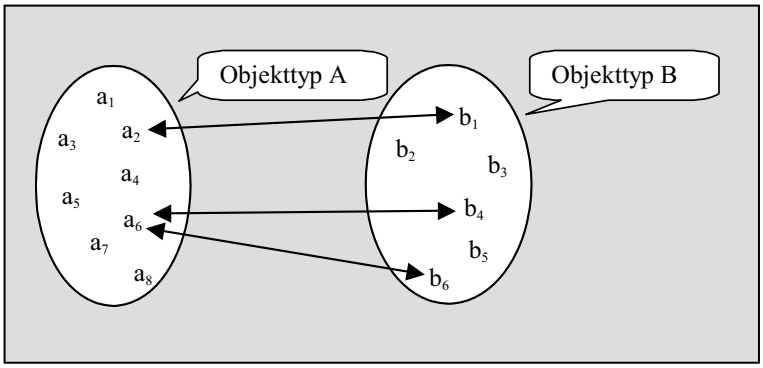


Abb. 4-26: C:CN-Beziehungstyp

Beispiel für Transformationsregel T10

Als Beispiel betrachten wir eine Abteilung, die noch keinen, schon einen oder bereits mehrere Mitarbeiter hat. Die meisten Mitarbeiter sind in eine Abteilung eingeordnet, einige wenige - mit zentralen Aufgaben - gehören jedoch zu keiner Abteilung. Die Transformation in das relationale Datenbank-Modell ist in Abbildung 4-28 wiedergegeben.

Bei den mit zentralen Aufgaben betrauten Mitarbeitern wird der Fremdschlüssel *⇑Abteilungsname⇑* mit der NULL-Marke belegt. Da der Fremdschlüssel nicht-unikal ist, können mehrere Mitarbeiter mit derselben Abteilung in Verbindung gebracht werden. Da man ohnehin nicht fordern kann, dass jeder Wert des Primärschlüssels **Abteilungsname** auch als Wert des Fremdschlüssels *⇑Abteilungsname⇑* auftritt, kann es Abteilungen ohne Mitarbeiter geben.

Koppel-Tabelle

Sind bei einem C:CN-Beziehungstyp nur wenige Objekte des Objekttyps B mit einem A-Objekt verknüpft, dann wird der Fremdschlüssel in der Tabelle B häufig mit der NULL-Marke belegt. Um dies zu verhindern, stellt man die Paarungen zwischen den A-Objekten und den B-Objekten – so wie schon beim C:C-Beziehungstyp geschehen - in einer Koppel-Tabelle A/B dar.

Die Repräsentation eines C:CN-Beziehungstyps durch eine Koppel-Tabelle entspricht im Datenmodell der Transformation des C:CN-Beziehungstyps in einen 1:CN- und einen C:1-Beziehungstyp. Das entsprechende Umwandlungsverfahren ist in der Abbildung 4-29 dargestellt.

Transformationsregel T10 (C:CN-Beziehungstyp mit vielen gekoppelten B-Objekten):

Entity-Relationship-Modell		**Relationales Datenbank-Modell**
Objekttyp A mit Schlüssel SA	⇒	Tabelle A mit Primärschlüssel **SA**
Objekttyp B mit Schlüssel SB	⇒	Tabelle B mit Primärschlüssel **SB**
C:CN-Beziehungstyp	⇒	**SA** wird in B als nicht-eingabepflichtiger nicht-unikaler Fremdschlüssel aufgenommen
Speicherbedarf für Beziehungstyp		$\overline{\overline{B}} \cdot Len(SA)$

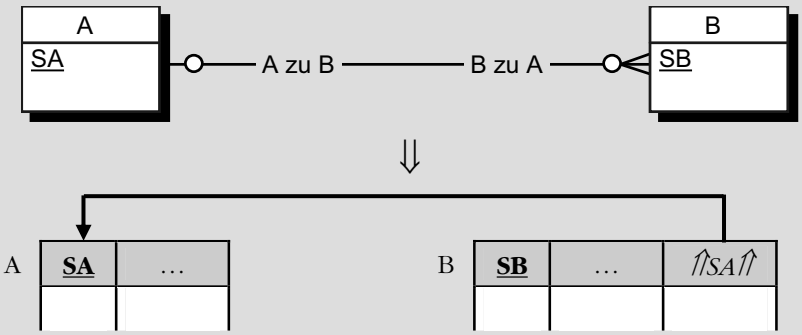


Abb. 4-27: Transformation eines C:CN-Beziehungstyps mit vielen gekoppelten B-Objekten

Man sieht, dass die Optionalität/Kardinalität der Beziehungstyp-Richtungen „A zu B“ bzw. „B zu A“ auf die Beziehungstyp-Richtungen „A zu A/B“ bzw. „B zu A/B“ übertragen wurden. Die Beziehungstyp-Richtung „A/B zu B“ übernimmt die Identifizierung des neu gebildeten Objekttyps A/B.

Stellt man den Koppel-Objekttyp A/B im relationalen Datenbank-Modell als Koppel-Tabelle dar, so verweist eine Zeile dieser Tabelle sowohl auf ein A-Objekt als auch auf ein B-Objekt, enthält also die Primärschlüssel beider Tabellen jeweils als eingabepflichtigen unikalen Fremdschlüssel. Als Primärschlüssel von A/B wird der Fremdschlüssel verwendet, der auf das B-Objekt verweist. In Abbildung 4-30 ist die Transformationsregel T11 dargestellt.

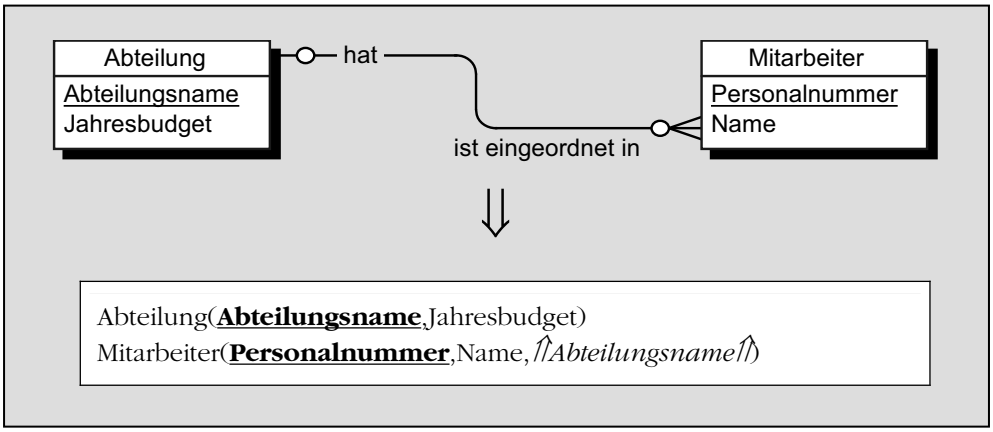


Abb. 4-28: Beispiel für die Transformationsregel T10

Der Fremdschlüssel ⇑SA⇑ ist nich-tunikal. Damit können mehrere Zeilen der Tabelle A/B auf dasselbe A-Objekt verweisen, so dass ein A-Objekt an mehreren Paarungen beteiligt sein kann. Da der Fremdschlüssel ⇑**SB**⇑ aber unikal sein muss, kann ein B-Objekt nur höchstens eine Paarung eingehen. Darum eignet sich ⇑**SB**⇑ als Primärschlüssel der Tabelle A/B.

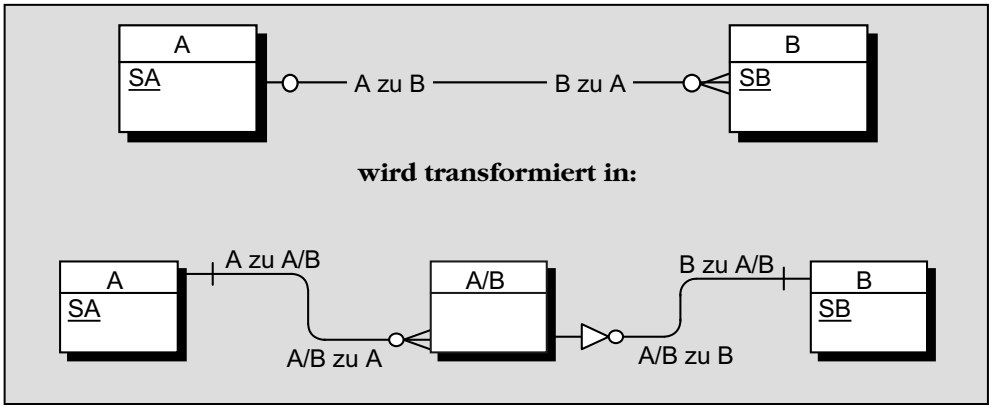


Abb. 4-29: Umwandlung eines C:CN-Beziehungstyps in einen 1:CN- und einen C:1-Beziehungstyp

Transformationsregel T11 (C:CN-Beziehungstyp mit wenigen gekoppelten B-Objekten):

Entity-Relationship-Modell		Relationales Datenbank-Modell
Objekttyp A mit Schlüssel SA	⇒	Tabelle A mit Primärschlüssel **SA**
Objekttyp B mit Schlüssel SA	⇒	Tabelle B mit Primärschlüssel **SB**
C:CN-Beziehungstyp	⇒	Koppel-Tabelle A/B mit **SA** und **SB** als jeweils eingabepflichtiger Fremdschlüssel. **SA** wird als nicht-unikal vereinbart. **SB** wird als unikal vereinbart und bildet den Primärschlüssel von A/B.
Speicherbedarf für Beziehungstyp		$\overline{\overline{A/B}} \cdot (Len(SA) + Len(SB))$

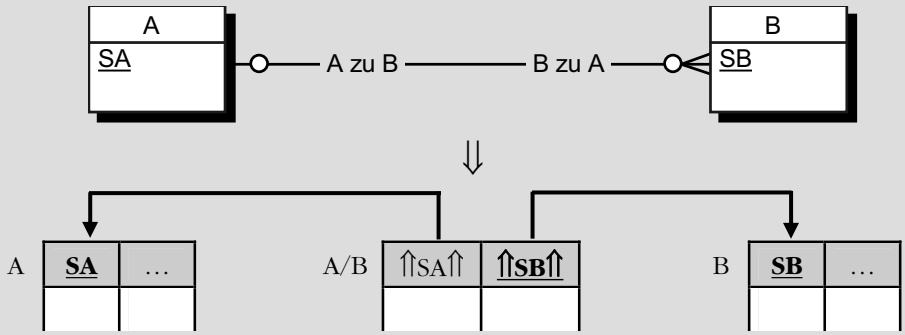


Abb. 4-30: Transformation eines C:CN-Beziehungstyps mit wenigen gekoppelten B-Objekten

Beispiel für Transformationsregel T11

Betrachten wir als Beispiel die Büroräume eines Umzugs-Unternehmens. Die Räume können mit keinem (leerstehender Raum), mit einem oder mit mehreren Mitarbeitern belegt sein. Die meisten Mitarbeiter fahren die Umzugswagen und arbeiten in keinem Raum, einige arbeiten jedoch in einem Büroraum. Die Abbildung 4-31 zeigt die notwendige Transformation.

Da der Fremdschlüssel ⇑Raumnummer⇑ nicht-unikal ist, kann ein Raum an mehreren Paarungen „Büroraum↔Mitarbeiter“ beteiligt sein. Auf Grund der Unikalitäts-Forderung für den Fremdschlüssel **⇑Personalnummer⇑** darf ein Mitarbeiter nur einmal

mit einem Büroraum gekoppelt sein. Deshalb bildet dieser Fremdschlüssel auch den Primärschlüssel der Koppel-Tabelle „Büroraum/Mitarbeiter“.

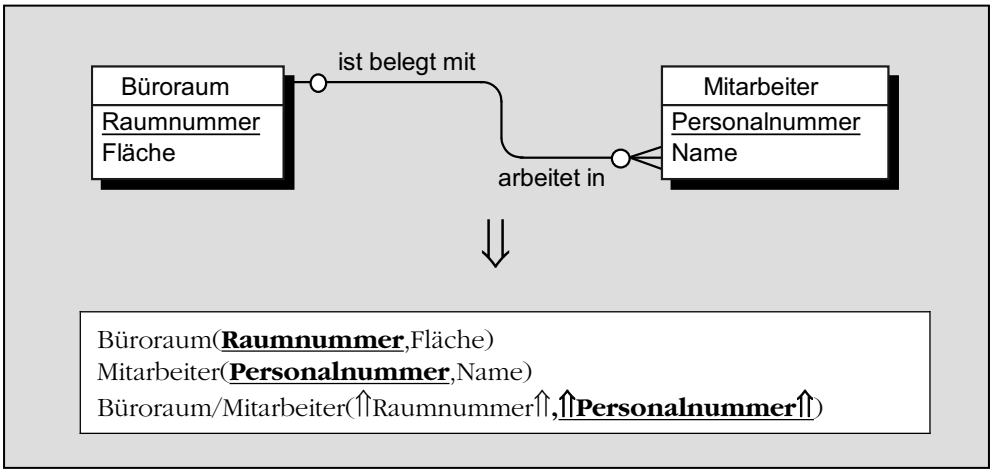


Abb. 4-31: Beispiel für die Transformationsregel T11

4.3.6 Der 1:N-Beziehungstyp

Der 1:N-Beziehungstyp unterscheidet sich vom 1:CN-Beziehungstyp nur dadurch, dass jedes Objekt des Objekttyps A mit mindestens einem Objekt des Objekttyps B verbunden sein muss. Die Situation ist in Abbildung 4-32 dargestellt.

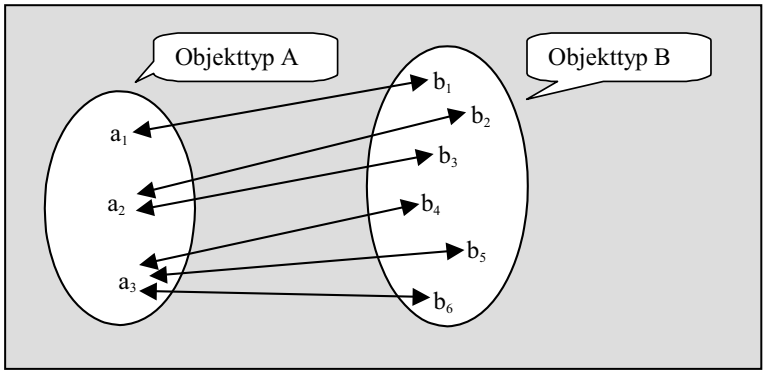


Abb. 4-32: 1:N-Beziehungstyp

Der 1:N-Beziehungstyp kann nicht repräsentiert werden

In der Literatur zum relationalen Datenbank-Modell wird der 1:N-Beziehungstyp gewöhnlich als die „meistverbreitete Art" von Beziehungstypen in relationalen Datenbanken bezeichnet. Das ist aber falsch, denn dieser Beziehungstyp lässt sich im relationalen Datenbank-Modell gar nicht repräsentieren. Das liegt daran, dass es auf der Ebene der Tabellen-Typbeschreibungen keine Möglichkeit gibt, die *Nicht-optionalität* der Beziehungstyp-Richtung „A zu B" zu repräsentieren.

Betrachten wir die Situation im Einzelnen! Die Standardlösung zur Transformation eines Beziehungstyps besteht ja darin, den Primärschlüssel der A-Tabelle in der B-Tabelle als Fremdschlüssel aufzunehmen. Soll nun jedes A-Objekt mit mindestens einem B-Objekt gekoppelt sein, so ist das gleichbedeutend mit der Forderung, dass jeder Wert, den der Primärschlüssel in A annimmt, wenigstens einmal als Wert des Fremdschlüssels in B auftauchen muss. Diese Forderung hat aber nichts mit der bereits besprochenen referenziellen Integrität zu tun. Diese fordert nur, dass jeder Wert des Fremdschlüssels in B entweder mit der NULL-Marke belegt sein muss oder dass er als Wert des Primärschlüssels in A vorhanden sein muss. Die Umkehrbedingung, dass jeder Wert des Primärschlüssels auch als Wert des Fremdschlüssels auftauchen muss, lässt sich im relationalen Datenbank-Modell nicht formulieren.

Transformationsregel T09

Die Transformation des 1:N-Beziehungstyps in das relationale Datenbank-Modell kann nur genau so erfolgen wie für den 1:CN-Beziehungstyp, also gemäß der Transformationsregel T09. Dabei muss man allerdings in Kauf nehmen, dass wichtige semantische Informationen, die im Datenmodell repräsentiert wurden, verloren gehen. Diese Informationen müssen im Rahmen der Anwendungs-Programmierung berücksichtigt werden.

Beispiel für 1:N-Beziehungstyp

Als Beispiel betrachten wir ein Unternehmen, in dem eine Geschäftsregel besagt, dass ein Kunde erst dann gespeichert wird, wenn er die erste Bestellung vornimmt. Im Laufe der Zeit können einem Kunde natürlich viele Bestellungen zugeordnet werden. Jede Bestellung kommt von genau einem Kunden. Ein Kunde, für den keine Bestellung mehr besteht (weil er alle seine Bestellungen storniert hat), wird wieder gelöscht. Das Datenmodell muss nach der Transformationsregel T09 entsprechend der Abbildung 4-33 in das relationale Datenbank-Modell umgesetzt werden.

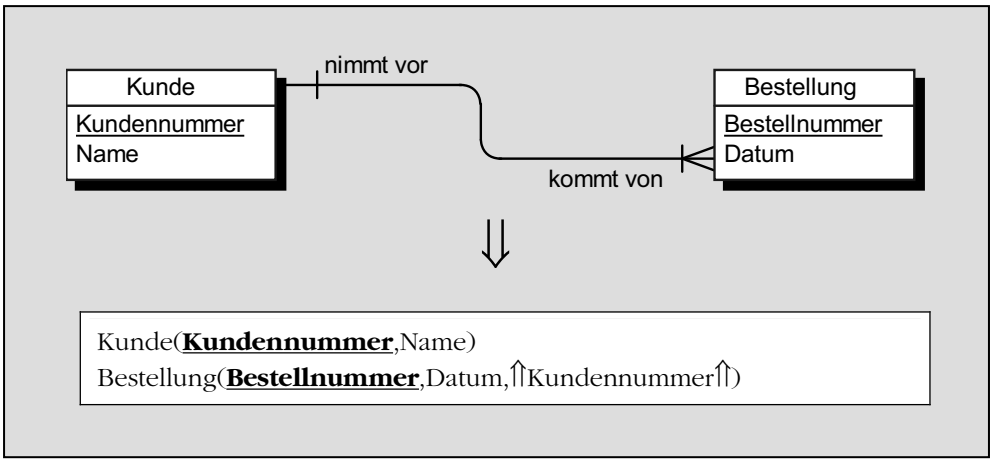


Abb. 4-33: Beispiel für die Transformation eines 1:N-Beziehungstyps

Der Fremdschlüssel ⇑Kundennummer⇑ ist eingabepflichtig: Jede Bestellung wird somit genau einem Kunden zugeordnet. Der Fremdschlüssel ist nicht-unikal: Mehrere Bestellungen können also auf denselben Kunden verweisen. Es wird aber nicht gesichert, dass jeder Kunden mit mindestens einer Bestellung verknüpft ist. Diese Geschäftsregel kann nicht beim Datenbankentwurf „festgeschrieben" werden, sondern muss durch die Anwendungs-Software erzwungen werden.

4.3.7 Der C:N-Beziehungstyp

Der C:N-Beziehungstyp unterscheidet sich vom 1:N-Beziehungstyp dadurch, dass nicht jedes Objekt des Objekttyps B mit einem Objekt des Objekttyps A gekoppelt sein muss. Abbildung 4-34 stellt diese Situation dar.

Wie schon beim 1:N-Beziehungstyp, so ist auch hier die Nichtoptionalität der Beziehungstyp-Richtung „A zu B" im relationalen Datenbank-Modell nicht zu erzwingen. Die Transformation des C:N-Beziehungstyps kann nur genau so erfolgen wie für den C:CN-Beziehungstyp, also entweder gemäß der Transformationsregel T10 (viele B-Objekte gehen eine Paarung ein) oder gemäß der Transformationsregel T11 (wenige B-Objekte haben einen A-Partner). In jedem Fall muss jedoch ein Verlust an semantischen Informationen in Kauf genommen werden.

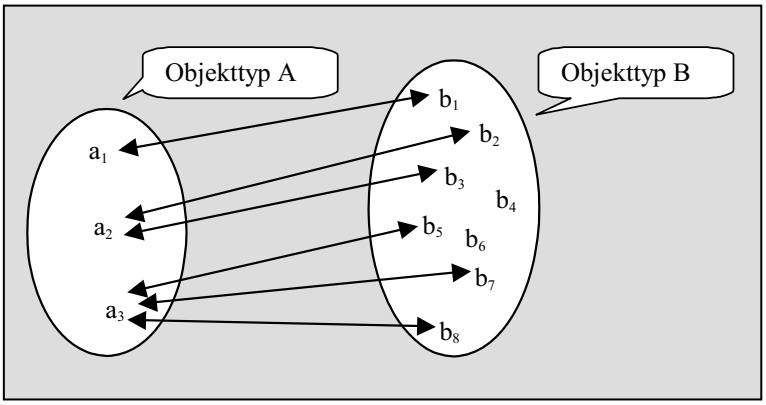


Abb. 4-34: C:N-Beziehungstyp

Beispiel für C:N-Beziehungstyp

Betrachten wir als Beispiel einen Eisenbahnzug, der aus einem oder mehreren Wagen bestehen kann. Einen Zug, der aus keinem Wagen besteht, gibt es nicht. Zu einem gegebenen Zeitpunkt gehören die meisten Wagen des Eisenbahn-Unternehmens zu einem Eisenbahnzug. Nur wenige Wagen stehen in der Reserve und gehören zu keinem Zug. Die Transformation muss dann entsprechend der Transformationsregel T10 vorgenommen werden, wie dies aus Abbildung 4-35 ersichtlich ist.

Der Fremdschlüssel *⇑Zugnummer⇑* ist nicht-eingabepflichtig: Ein Wagen muss also nicht unbedingt zu einem Eisenbahnzug gehören. Der Fremdschlüssel ist nicht-unikal: Mehrere Wagen können zum selben Eisenbahnzug gehören. Es lässt sich aber nicht erzwingen, dass jeder Eisenbahnzug aus mindestens einem Wagen besteht. Die Datenbank würde es also durchaus zulassen, dass ein neuer Eisenbahnzug gespeichert wird, ohne dass ihm ein Wagen zugeordnet wird. Will man diesen in der Praxis nicht tolerierbaren Fehler vermeiden, kann dies nur durch eine entsprechende Gestaltung der Anwendungs-Software erreicht werden.

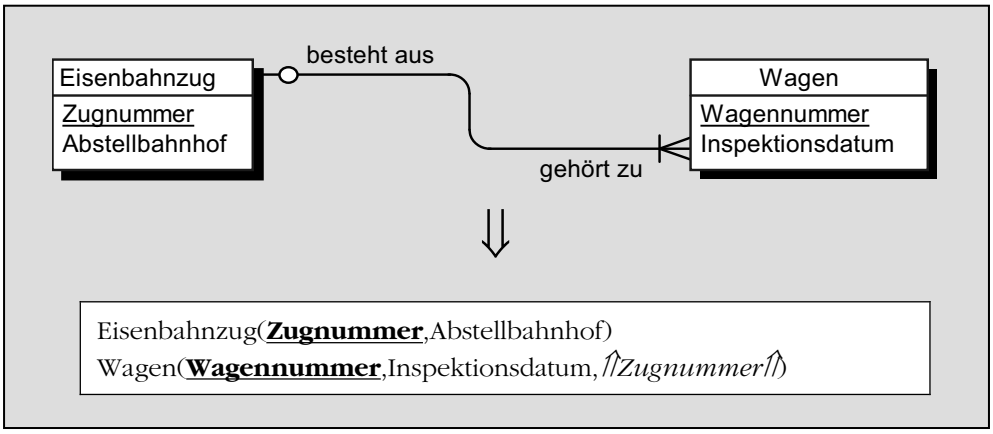


Abb. 4-35: Beispiel für die Transformation eines C:N-Beziehungstyps

4.3.8 Der CM:CN-Beziehungstyp

Im Abschnitt 3.4.3 wurde bereits ausgeführt, dass sich im relationalen Datenbank-Modell Beziehungstypen, die in beiden Beziehungstyp-Richtungen die Kardinalität N aufweisen, nicht direkt darstellen lassen.

Der Grund dafür liegt im Hauptprinzip des relationalen Datenbank-Modells, demzufolge Attributwerte nur atomar sein können. Der Wert eines Fremdschlüssels kann somit nur auf *eine* Zeile und nicht auf *mehrere* Zeilen verweisen.

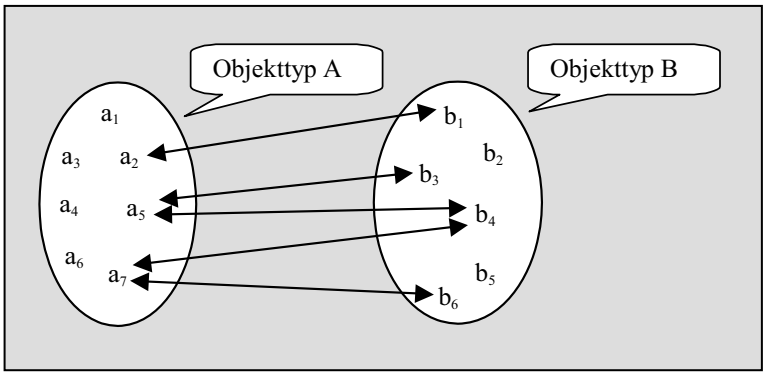


Abb. 4-36: CM:CN-Beziehungstyp

Beim CM:CN-Beziehungstyp kann ein Objekt des Objekttyps A jedoch nicht nur mit keinem oder einem, sondern auch mit mehreren Objekten des Objekttyps B verbunden sein, ebenso wie ein B-Objekt mit keinem, einem oder mehreren A-Objekten gekoppelt sein kann. Abbildung 4-36 zeigt dafür ein Beispiel.

Koppel-Objekttyp

Die Repräsentation des CM:CN-Beziehungstyps ist im relationalen Datenbank-Modell nur in der Form möglich, dass man einen neuen – rein technisch bedingten – Koppel-Objekttyp A/B einführt. A/B wird mit den Objekttypen A bzw. B jeweils durch einen CN:1-Beziehungstyp verbunden. Das Ergebnis dieser Umwandlung zeigt die Abbildung 4-37.

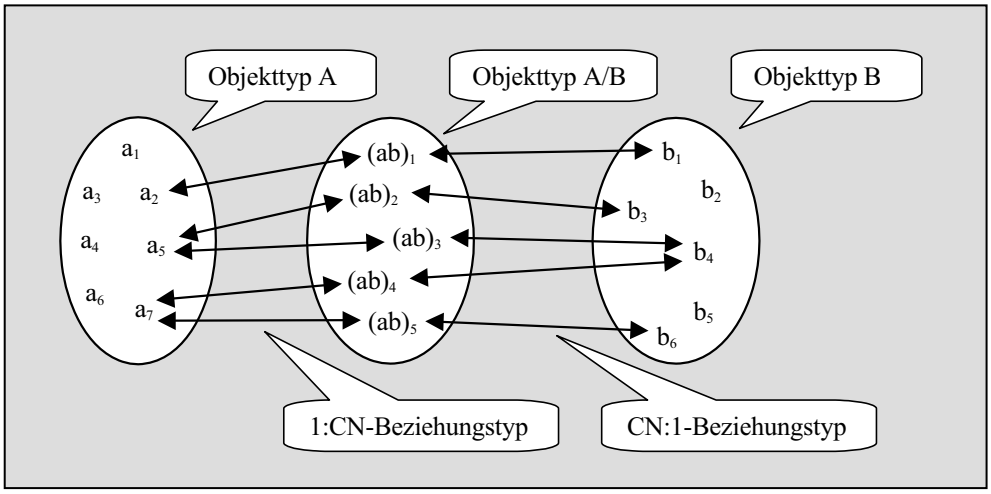


Abb. 4-37: Aufgelöster CM:CN-Beziehungstyp

Die Umwandlung entspricht der Transformation, die bereits im Abschnitt 2.6.2 beschrieben wurde. Dort war sie erforderlich, weil wir einem Beziehungstyp Eigenschaften zuweisen wollten. Die Umwandlung eines CM:CN-Beziehungstyps in einen 1:CN- und einen CN:1-Beziehungstyp läuft nach dem Muster ab, das in Abbildung 4-38 dargestellt ist.

Man erkennt, dass die Optionalität/Kardinalität der Beziehungstyp-Richtungen „A zu B“ bzw. „B zu A“ auf die Beziehungstyp-Richtungen „A zu A/B“ bzw. „B zu A/B“ übertragen wurden. Die Objekte des Koppel-Objekttyps „A/B“ werden durch die beiden Beziehungstyp-Richtungen „A/B zu A“ und „A/B zu B“ identifiziert.

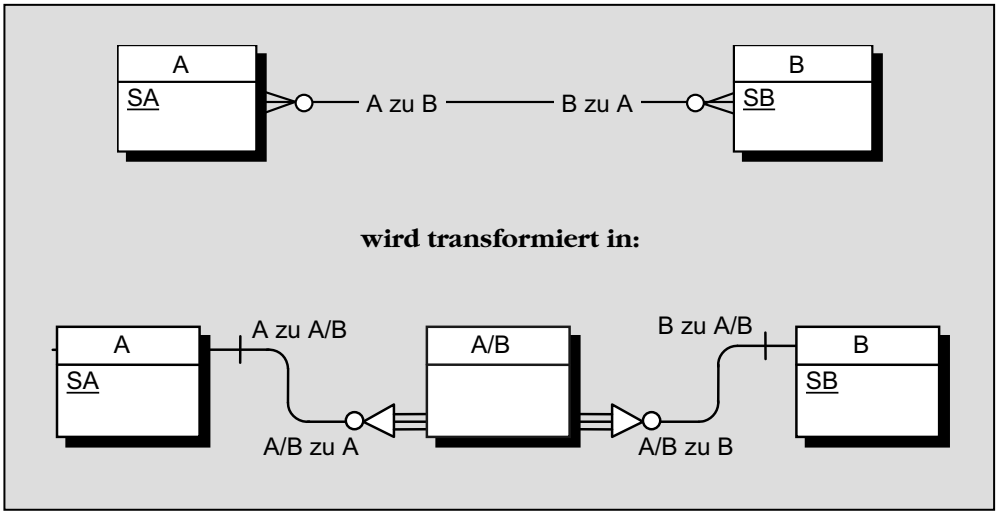


Abb. 4-38: Umwandlung eines CM:CN-Beziehungstyps in einen 1:CN- und einen CN:1-Beziehungstyp

Transformationsregel T12

Die Repräsentation der (teil-)identifizierenden Beziehungstyp-Richtungen in Abbildung 4-38 erfolgt nach der Transformationsregel T02, die bereits im Abschnitt 4.2 erläutert wurde. Die beiden neu gebildeten 1:CN-Beziehungstypen werden nach der Transformationsregel T09 in das relationale Datenbank-Modell überführt. Insgesamt entspricht das der Transformationsregel T12, die in Abbildung 4-39 wiedergegeben ist.

Bei der Berechnung des Speicherbedarfs für die Darstellung des CM:CN-Beziehungstyps wurde berücksichtigt, dass für jede Kombination eines A-Objekts und eines B-Objekts deren Schlüsselwerte als Fremdschlüsselwerte abgelegt werden müssen. Die Anzahl dieser Kombinationen ist gerade die Mächtigkeit der Menge der A↔B-Verbindungen, also der Koppel-Tabelle.

Beispiel für Transformationsregel T12

Als Beispiel für einen CM:CN-Beziehungstyp betrachten wir noch einmal die Mitarbeiter, die an keinem, einem oder mehreren Projekten beteiligt sein können, wobei ein Projekt (noch) von keinem, von einem Mitarbeiter oder auch von mehreren Mitarbeitern bearbeitet wird (vgl. Abschnitt 3.4.3). Die Tabellenrepräsentation gemäß der Transformationsregel T12 ist in der Abbildung 4-40 wiedergegeben.

Transformationsregel T12 (CM:CN-Beziehungstyp):

Entity-Relationship-Modell		**Relationales Datenbank-Modell**
Objekttyp A mit Schlüssel SA	⇒	Tabelle A mit Primärschlüssel **SA**
Objekttyp B mit Schlüssel SB	⇒	Tabelle B mit Primärschlüssel **SB**
CM:CN-Beziehungstyp	⇒	Koppel-Tabelle A/B. **SA** und **SB** werden als eingabepflichtige nicht-unikale Fremdschlüssel in A/B aufgenommen. Die Kombination der Fremdschlüssel ⇑SA⇑ und ⇑SB⇑ wird als unikal vereinbart. Sie bildet den Primärschlüssel von A/B.
Speicherbedarf für Beziehungstyp		$\overline{\overline{A/B}} \cdot (Len(SA) + Len(SB))$

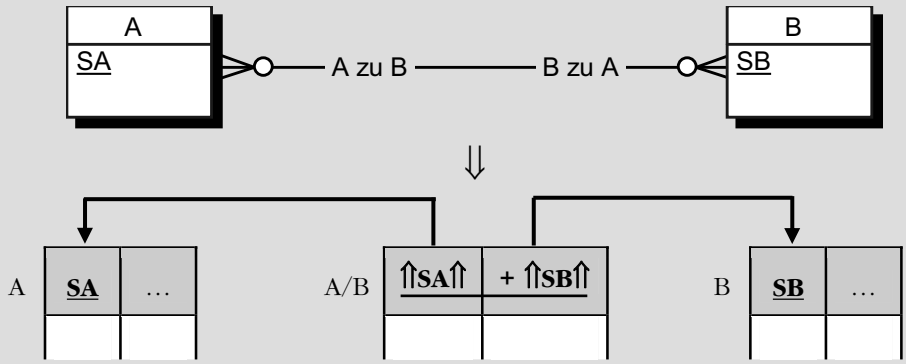


Abb. 4-39: Transformation eines CM:CN-Beziehungstyps

Eine Zeile der Koppel-Tabelle „Mitarbeiter/Projekt“ verweist auf genau einen Mitarbeiter und auf genau ein Projekt. Da der Fremdschlüssel ⇑Personalnummer⇑ für sich alleine nicht-unikal ist, kann es mehrere Zeilen der Koppel-Tabelle geben, die auf denselben Mitarbeiter verweisen. Die Nicht-unikalität des Fremdschlüssels ⇑Projektnummer⇑ ermöglicht es, dass mehrere Zeilen der Koppel-Tabelle mit demselben Projekt verknüpft sind. Erst die Kombination **⇑Personalnummer⇑+⇑Projektnummer⇑** ist als unikal vereinbart und bildet den Primärschlüssel der Koppel-Tabelle.

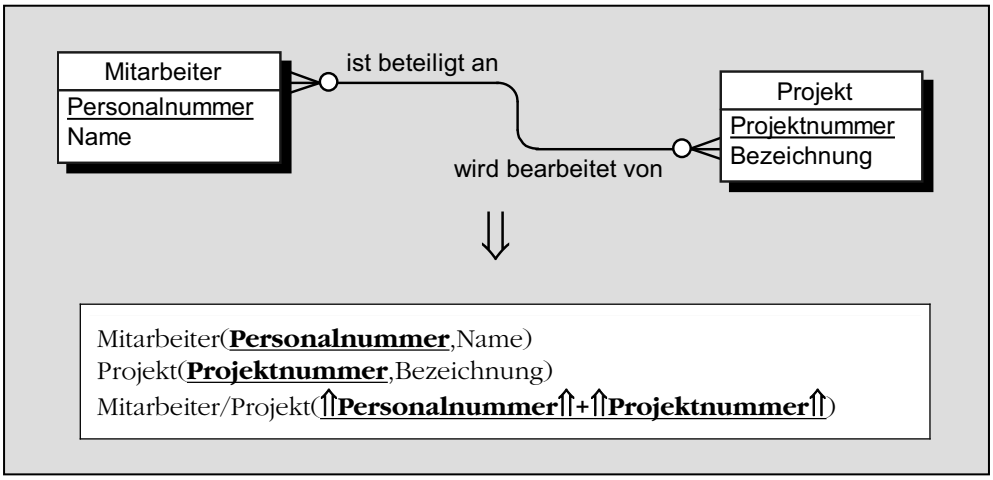


Abb. 4-40: Beispiel für die Transformationsregel T12

4.3.9 Der M:CN-Beziehungstyp

Beim M:CN-Beziehungstyp kann ein Objekt des Objekttyps A mit keinem, einem oder mehreren Objekten des Objekttyps B verbunden sein, ein B-Objekt jedoch nur mit einem oder mehreren A-Objekten. Abbildung 4-41 zeigt diese Situation schematisch.

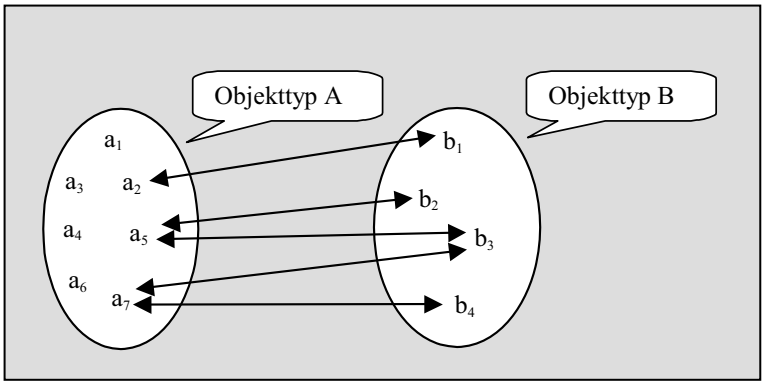


Abb. 4-41: M:CN-Beziehungstyp

Analog zum CM:CN-Beziehungstyp muss der M:CN-Beziehungstyp vor seiner Transformation in das relationale Datenbank-Modell durch die Einführung eines neuen Objekttyps A/B in einen 1:CN-Beziehungstyp und einen N:1-Beziehungstyp umgeformt werden. Diese Umwandlung zeigt die Abbildung 4-42.

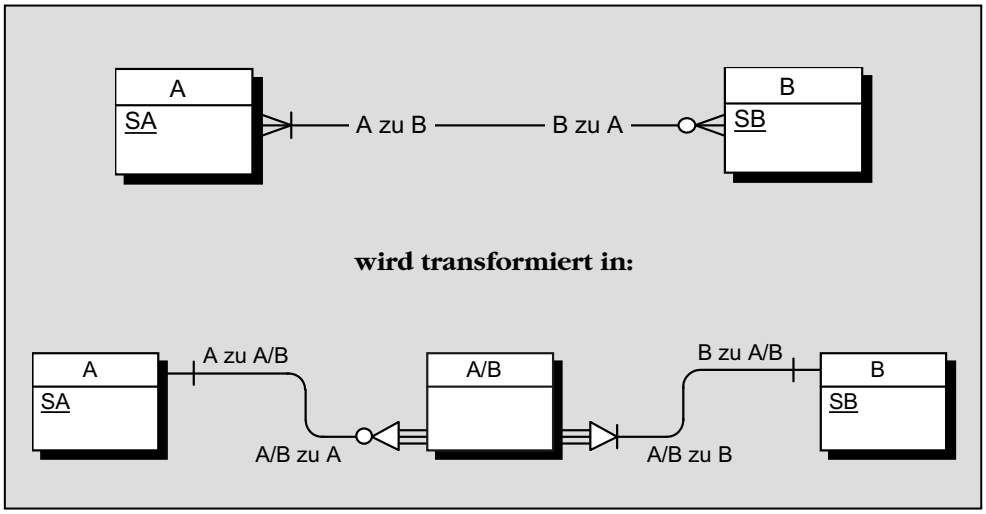


Abb. 4-42: Umwandlung eines M:CN-Beziehungstyps in einen 1:CN- und einen N:1-Beziehungstyp

Transformationsregel T12

Im Abschnitt 4.3.6 wurde gezeigt, dass der N:1-Beziehungstyp im relationalen Datenbank-Modell nur - unter Verlust von semantischen Informationen – als CN:1-Beziehungstyp repräsentiert werden kann. Die Transformation des M:CN-Beziehungstyps erfolgt damit wie beim CM:CN-Beziehungstyp nach der Transformationsregel T12.

Beispiel für M:CN-Beziehungstyp

Als Beispiel betrachten wir Ärzte, die (noch) keine, bereits eine Operation oder schon mehrere Operationen ausgeführt haben. Eine Operation ohne Ärzte gibt es – Gott sei Dank! – (noch) nicht. Mitunter wird eine Operation aber von mehreren Ärzten ausgeführt. Die Transformation zeigt die Abbildung 4-43.

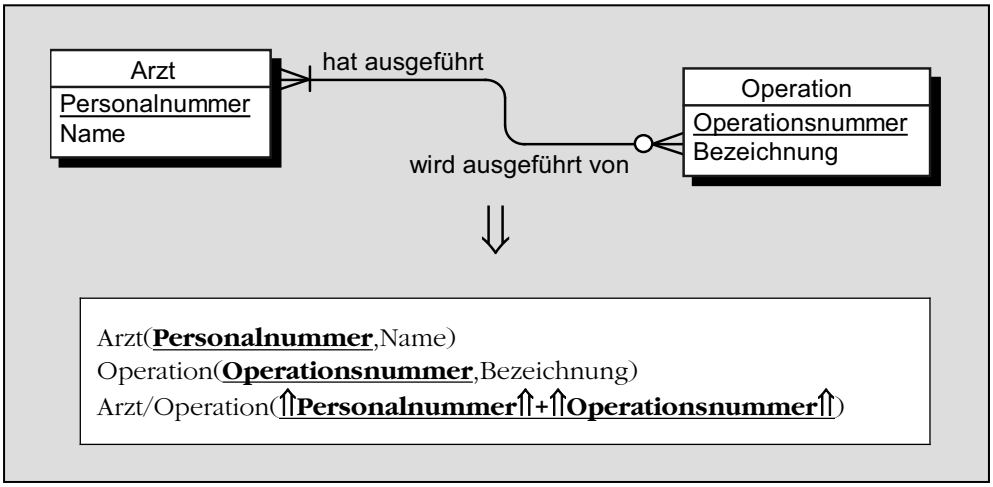


Abb. 4-43: Beispiel für die Transformation eines M:CN-Beziehungstyps

Durch jede Zeile der Koppel-Tabelle „Arzt/Operation“ wird ein Arzt mit einer Operation in Verbindung gebracht. Keiner der beiden Fremdschlüssel ⇑Personalnummer⇑ bzw. ⇑Operationsnummer⇑ muss für sich genommen unikal sein. Damit können mehrere Zeilen der Koppel-Tabelle auf denselben Arzt verweisen. Ebenso können mehrere Zeilen dieselbe Operation betreffen. Es kann aber durch die Tabellen-Typbeschreibungen nicht erzwungen werden, dass jede Operationsnummer auch tatsächlich als Fremdschlüssel in der Koppel-Tabelle auftaucht. In der Datenbank kann also die falsche Aussage gespeichert werden, dass einer Operation kein Arzt zugeordnet ist. Dieser – in der Praxis wahrscheinlich tödliche - Fehler kann nur durch softwaretechnische Maßnahmen vermieden werden.

4.3.10 Der M:N-Beziehungstyp

Beim M:N-Beziehungstyp ist jedes Objekt des Objekttyps A mit einem oder mehreren Objekten des Objekttyps B verbunden, ebenso wie jedes B-Objekt mit einem oder mehreren A-Objekten gekoppelt ist. Abbildung 4-44 gibt ein Beispiel für diese Situation.

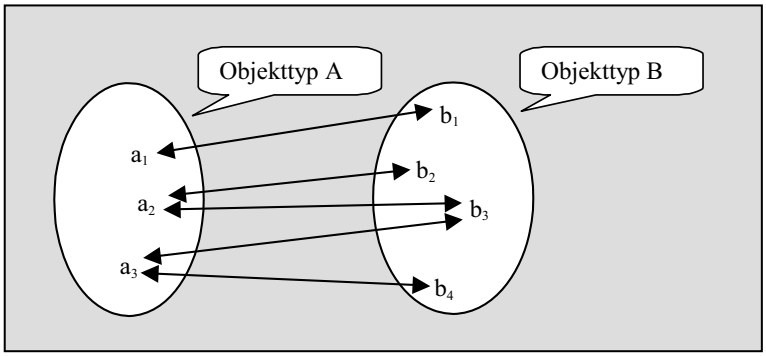


Abb. 4-44: M:N-Beziehungstyp

Der M:N-Beziehungstyp muss vor seiner Transformation in das relationale Datenbank-Modell in einen 1:N-Beziehungstyp und einen N:1-Beziehungstyp umgeformt werden, wie dies aus der Abbildung 4-45 ersichtlich ist.

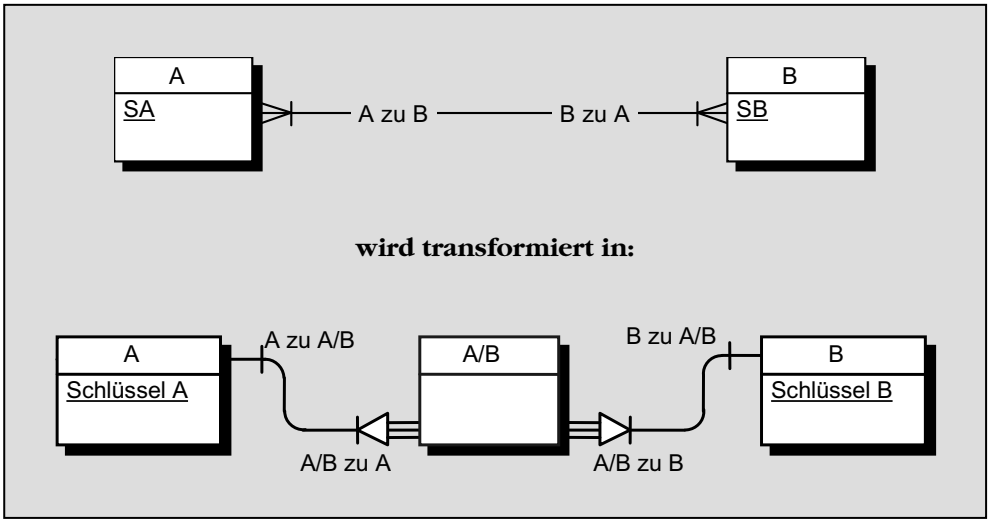


Abb. 4-45: Umwandlung eines M:N-Beziehungstyps in einen 1:N- und einen N:1-Beziehungstyp

Beide Beziehungstypen müssen - wiederum nur unter Semantikverlust - als 1:CN- bzw. CN:1-Beziehungstypen in das relationale Datenbank-Modell überführt werden. Die Transformation des M:N-Beziehungstyps erfolgt damit - so wie für den CM:CN-Beziehungstyp - nach der Transformationsregel T12.

Beispiel für M:N-Beziehungstyp

Als Beispiel betrachten wir eine Bibliothek, in der die Bücher den einzelnen Sachgruppen zugeordnet werden. Ein Buch wird meist nur einer Sachgruppe, mitunter aber auch mehreren Sachgruppen zugeordnet. Eine Sachgruppe wird erst dann eingeführt, wenn sie wenigstens ein Buch enthält. Abbildung 4-46 zeigt die erforderliche Transformation.

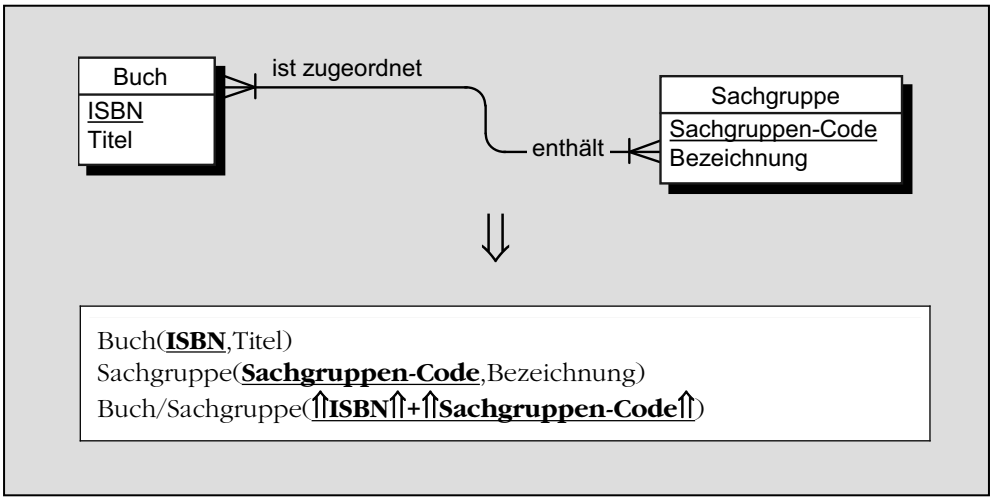


Abb. 4-46: Beispiel für die Transformation eines M:N-Beziehungstyps

Jede Zeile in der Koppel-Tabelle „Buch/Sachgruppe“ ordnet ein Buch jeweils einer Sachgruppe zu. Da weder der Fremdschlüssel ⇑ISBN⇑ noch der Fremdschlüssel ⇑Sachgruppen-Code⇑ für sich unikal sind, können mehrere Zeilen der Koppel-Tabelle auf dasselbe Buch oder auch auf dieselbe Sachgruppe verweisen. Die Tabellen-Typbeschreibungen sichern aber weder, dass einem Buch wenigstens eine Sachgruppe zugeordnet wird, noch können sie garantieren, dass es keine „leere“ Sachgruppe gibt. Diese Geschäftsregeln müssen von der Anwendungs-Software durchgesetzt werden.

4.3.11 Transformation der dualen Beziehungstypen für das Schulbeispiel

Das Datenmodell unseres Schulbeispiels wurde in den beiden Abschnitten 4.1 und 4.2 gemäß den Transformationsregeln T01 und T02 in die vorläufigen Tabellen-Typbeschreibungen überführt, die in der Abbildung 4-47 noch einmal angegeben sind.

Ort(**Name+Kreis**,Einwohnerzahl)
Schule(**Name+Postleitzahl**,Straße,Hausnummer)
Beschäftigungsverhältnis(⇑**Personalnummer**⇑+
⇑**Schul-Name+Postleitzahl**⇑+
Vertragsdatum)
Lehrer(**Personalnummer**,Vorname,Familienname,Geburtsdatum,Familienstand)
Klasse(⇑**Schul-Name+Postleitzahl**⇑+**Bezeichnung**)
Schüler(**Vorname+Familienname+Geburtsdatum**,Adresse)
Unterrichtsverpflichtung
(⇑**Schul-Name+Postleitzahl+Klassen-Bezeichnung**⇑+
⇑**Fach-Bezeichnung**⇑+
⇑**Personalnummer**⇑)
Fach(**Bezeichnung**,Min. Stunden/Woche,Max. Stunden/Woche)
Unterrichtsraum(⇑**Schul-Name+Postleitzahl**⇑+**Nummer**,Fläche,Sitzplatzanzahl)

Abb. 4-47: Vorläufige Tabellen-Typbeschreibungen

Wir wollen jetzt unter Beachtung der Transformationsregeln T03 bis T10 die dualen Beziehungstypen in die bisherigen Tabellen-Typbeschreibungen „einarbeiten". Mitunter ist die angegebene Transformation nicht erforderlich, weil die zu realisierenden Effekte bereits durch die Transformationsregel T02 (Beziehungstyp-Richtung als identifizierendes Element) erreicht wurden.

Beziehung: Ort zu Schule
Beziehungstyp: 1:CN
Transformationsregel: T09
Modifikation: Primärschlüssel von „Ort“ wird Fremdschlüssel in „Schule“
Neue Typbeschreibung: Schule(**Name+Postleitzahl**,⇑Orts-Name+Kreis⇑, Straße,Hausnummer)

Beziehung: Schule zu Beschäftigungsverhältnis
Beziehungstyp: 1:CN
Transformationsregel: T09
Modifikation: Primärschlüssel von „Schule“ wird Fremdschlüssel in „Beschäftigungsverhältnis“
Neue Typbeschreibung: Keine Veränderung, da die Modifikation bereits durch T02 erfolgt ist.

Beziehung: Lehrer zu Beschäftigungsverhältnis
Beziehungstyp: 1:CN
Transformationsregel: T09
Modifikation: Primärschlüssel von „Lehrer“ wird Fremdschlüssel in „Beschäftigungsverhältnis“
Neue Typbeschreibung: Keine Veränderung, da die Modifikation bereits durch T02 erfolgt ist.

Beziehung: Lehrer zu Klasse
Beziehungstyp: 1:CN
Transformationsregel: T09
Modifikation: Primärschlüssel von „Lehrer“ wird Fremdschlüssel in „Klasse“
Neue Typbeschreibung: Klasse(⇑**Schul-Name+Postleitzahl**⇑+ **Bezeichnung**, ⇑Personalnummer⇑)

Beziehung: Schule zu Klasse
Beziehungstyp: 1:CN
Transformationsregel: T09
Modifikation: Primärschlüssel von „Schule“ wird Fremdschlüssel in „Klasse“
Neue Typbeschreibung: Keine Veränderung, da die Modifikation bereits durch T02 erfolgt ist.

Beziehung:	Klasse zu Unterrichtsverpflichtung
Beziehungstyp:	1:N (muss als 1:CN-Beziehungstyp realisiert werden)
Transformationsregel:	T09
Modifikation:	Primärschlüssel von „Klasse“ wird Fremdschlüssel in „Unterrichtsverpflichtung“
Neue Typbeschreibung:	Keine Veränderung, da die Modifikation bereits durch T02 erfolgt ist.

Beziehung:	Lehrer zu Unterrichtsverpflichtung
Beziehungstyp:	1:CN
Transformationsregel:	T09
Modifikation:	Primärschlüssel von „Lehrer“ wird Fremdschlüssel in „Unterrichtsverpflichtung“
Neue Typbeschreibung:	Keine Veränderung, da die Modifikation bereits durch T02 erfolgt ist.

Beziehung:	Fach zu Unterrichtsverpflichtung
Beziehungstyp:	1:CN
Transformationsregel:	T09
Modifikation:	Primärschlüssel von „Fach“ wird Fremdschlüssel in „Unterrichtsverpflichtung“
Neue Typbeschreibung:	Keine Veränderung, da die Modifikation bereits durch T02 erfolgt ist.

Beziehung:	Klasse zu Schüler
Beziehungstyp:	1:N (muss als 1:CN-Beziehungstyp realisiert werden)
Transformationsregel:	T09
Modifikation:	Primärschlüssel von „Klasse“ wird Fremdschlüssel in „Schüler“
Neue Typbeschreibung:	Schüler(**Vorname+Familienname+ Geburtsdatum**, ⇑⇑Schul-Name+Postleitzahl+ Klassen-Bezeichnung⇑⇑, Adresse)

Beziehung: Schule zu Unterrichtsraum
Beziehungstyp: 1:CN
Transformationsregel: T09
Modifikation: Primärschlüssel von „Schule“ wird Fremdschlüssel in „Unterrichtsraum“
Neue Typbeschreibung: Keine Veränderung, da die Modifikation bereits durch T02 erfolgt ist.

Beziehung: Unterrichtsraum zu Klasse
Beziehungstyp: 1:CN
Transformationsregel: T09
Modifikation: Primärschlüssel von „Unterrichtsraum“ wird Fremdschlüssel in „Klasse“
Neue Typbeschreibung: Klasse(⇑**Schul-Name-1+Postleitzahl-1**⇑+
Bezeichnung,
⇑Personalnummer⇑,
⇑Schul-Name-2+Postleitzahl-2+
Raum-Nummer⇑)
Bemerkung: Die Attribute-Kombination „Schul-Name+Postleitzahl“ bildet *zweimal* einen Fremdschlüssel, der auf eine Schule verweist. Der *erste* Fremdschlüssel verweist auf die Schule, zu der die Klasse gehört (Schul-Name-1+Postleitzahl-1), der *zweite* auf die Schule, zu der der Unterrichtsraum gehört, den die Klasse als Klassenraum nutzt (Schul-Name-2+Postleitzahl-2). Durch die Nummerierung werden die Attribute-Bezeichnungen innerhalb der Tabelle „Klasse“ eindeutig gemacht.

Beziehung: Unterrichtsraum zu Unterrichtsverpflichtung
Beziehungstyp: M:CN
(muss als CM:CN-Beziehungstyp realisiert werden)
Transformationsregel: T12
Neue Typbeschreibung: Unterrichtsraum/Unterrichtsverpflichtung
(⇑**Schul-Name-1+Postleitzahl-1+**
Raum-Nummer⇑+
⇑**Schul-Name-2+Postleitzahl-2+**
Klassen-Bezeichnung+
Fach-Bezeichnung+
Personalnummer⇑)

Bemerkung:

Der Primärschlüssel der Tabelle besteht aus *zwei* Bestandteilen, die jeweils ein Fremdschlüssel sind. Der *erste* Fremdschlüssel ist der Primärschlüssel der Tabelle „Unterrichtsraum". Dieser besteht aus dem Fremdschlüssel, der auf diejenige Schule verweist, in der der Unterrichtsraum liegt, und aus der „Raum-Nummer".
Der *zweite* Fremdschlüssel ist der Primärschlüssel der Tabelle „Unterrichtsverpflichtung". Dieser besteht seinerseits aus drei Fremdschlüsseln, die auf die Klasse, auf das Fach und auf den Lehrer verweisen.
Auf eine Schule wird also zweimal verwiesen:
1. auf die Schule, in der der Unterrichtsraum liegt,
2. auf die Schule, zu der die Klasse gehört.
Durch die Nummerierung müssen die doppelt auftretenden Attributbezeichnungen „Schul-Name" und „Postleitzahl" in der Tabelle „Unterrichtsraum/Unterrichtsverpflichtung" eindeutig gemacht werden.

4.4 Transformation von Rekursiv-Beziehungstypen

Im Abschnitt 2.5 haben wir Rekursiv-Beziehungstypen besprochen, mit deren Hilfe man einen sachlogischen Zusammenhang zwischen Objekten beschreiben kann, die demselben Objekttyp angehören. Wir wenden uns nun der Frage zu, wie die Rekursiv-Beziehungstypen im relationalen Datenbank-Modell repräsentiert werden können.

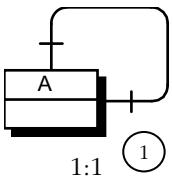
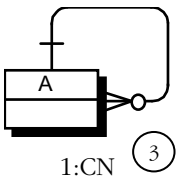
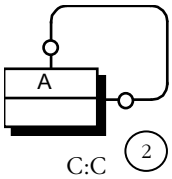
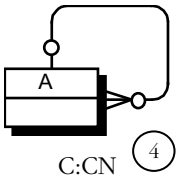
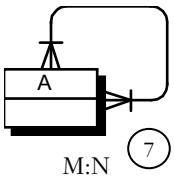
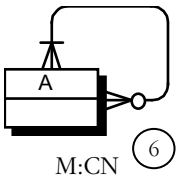
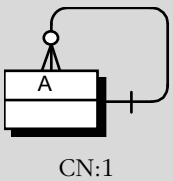
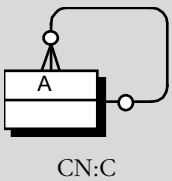
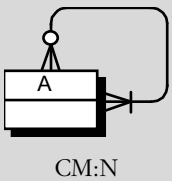
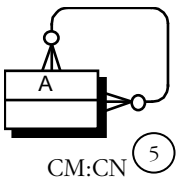
7 Klassen von Rekursiv-Beziehungstypen

In Tabelle 4-2 sind die interessierenden 7 Klassen von Rekursiv-Beziehungstypen einschließlich ihrer an der Diagonale gespiegelten „Gegenstücke" zusammengestellt. Die Reihenfolge der Betrachtung in den folgenden Abschnitten wird durch die laufende Nummer in den Kreisen angegeben.

Da bei Rekursiv-Beziehungstypen die miteinander gekoppelten Objekte beide im selben Objekttyp liegen, können sie nicht – wie bei den dualen Beziehungstypen – durch ihre Zugehörigkeit zum jeweiligen Objekttyp unterschieden werden. Bei Rekursiv-Beziehungstypen müssen die Objekte jedoch hinsichtlich ihrer *Rolle* unterschieden werden, die sie in der sachlogischen Beziehung spielen.

Wir greifen deshalb die Sprachregelung wieder auf, die wir zuvor in den Untersuchungen des Abschnitts 2.5 eingeführt hatten, und sprechen bei einem Rekursiv-Beziehungstyp allgemein von einem „Sender“ (linke Seite), der eine Nachricht an einen „Empfänger“ (rechte Seite) sendet. Als Beispiel dafür ist in Abbildung 4-48 der C:CN-Rekursiv-Beziehungstyp dargestellt.

Tab. 4-2: Sieben Klassen von Rekursiv-Beziehungstypen

1. Beziehungstyp-Richtung →		Kardinalität 1		Kardinalität N	
2. Beziehungstyp-Richtung ↓		nicht-optional	optional	nicht-optional	optional
Kardinalität 1	nicht-optional				
	optional				
Kardinalität N	nicht-optional				
	optional				

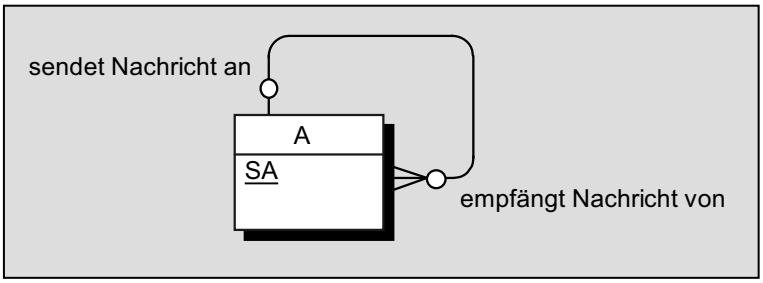


Abb. 4-48: Allgemeine Darstellung von Rekursiv-Beziehungstypen

Unmögliche Rekursiv-Beziehungstypen 1:C, 1:N und C:N

Mit dieser Sprachregelung können wir uns noch einmal der Erklärung der Tabelle 4-2 zuwenden. Im Abschnitt 2.4.5 wurde bereits begründet, warum es die in der Tabelle 4-2 fehlenden Rekursiv-Beziehungstypen 1:C, 1:N und C:N (einschließlich ihrer an der Diagonalen gespiegelten „Gegenstücke“) nicht geben kann. Die Begründung erfolgte dort nach dem Ausschluss-Verfahren. Es wurde nämlich untersucht, bei welchen dualen Beziehungstypen die beteiligten Objekttypen A und B eine unterschiedliche Mächtigkeit haben müssen. In all diesen Fällen sind Rekursiv-Beziehungstypen unmöglich, weil die in Beziehung stehenden Objekte *demselben* Objekttyp angehören.

Man kann aber auch durch eine konstruktive Überlegung zum selben Ergebnis gelangen.

Betrachten wir zunächst die erste Zeile der Tabelle 4-2! Sie verzeichnet diejenigen Fälle, in denen jedes Objekt genau eine Nachricht empfängt (nicht-multipler, nicht-optionaler Empfang). Soll jedes Objekt eines Objekttyps genau eine Nachricht empfangen, dann gibt es dafür nur zwei Möglichkeiten:

1. Jedes Objekt sendet genau eine Nachricht (1:1-Rekursiv-Beziehungstyp).
2. Mindestens ein Objekt a_1 sendet keine Nachricht (optionale Nachrichtensendung). Dann muss es die Aufgabe seiner Nachrichtensendung aber auf ein anderes Objekt a_2 übertragen, denn sonst würde ein Objekt beim Nachrichten-Empfang „leer ausgehen“. Das Objekt a_2 sendet also *zusätzlich* zu „seiner“ Nachricht auch noch eine Nachricht anstelle des Objekts a_1, sendet also *mehrere* Nachrichten (multiple

Nachrichtensendung). Das entspricht dem 1:CN-Rekursiv-Beziehungstyp.

Wie steht es nun mit der zweiten Zeile der Tabelle 4-2? Sie enthält jene Fälle, in denen die Objekte höchstens eine Nachricht empfangen, mindestens ein Objekt aber keine Nachricht empfängt (nicht-multipler, optionaler Empfang). Diese Situation ist wiederum nur durch zwei Möglichkeiten zu realisieren:

1. Mindestens ein Objekt sendet keine Nachricht, die anderen senden höchstens eine Nachricht (C:C-Rekursiv-Beziehungstyp).
2. Mindestens ein Objekt sendet keine Nachricht (optionale Nachrichtensendung), andere senden dafür mehrere Nachrichten (multiple Nachrichtensendung). Dies erfolgt aber so, dass die Summe der gesendeten Nachrichten die Anzahl der Objekte unterschreitet. Das entspricht dem C:CN-Rekursiv-Beziehungstyp.

Bei der graphischen Darstellung der dualen Beziehungstypen auf der Objektebene hatten wir die A- bzw. B-Objekte in Ellipsen eingeordnet, die durch ihre linke bzw. rechte Position eindeutig als A-Objekttyp bzw. B-Objekttyp gekennzeichnet waren. Die Beziehung zwischen einem A- und einem B-Objekt konnte durch einen Doppelpfeil gezeichnet werden, weil durch die Richtungen links→rechts bzw. links←rechts die jeweilige Semantik der Assoziation klar ersichtlich war. Bei den Rekursiv-Beziehungstypen fallen nun A- und B-Objekttyp zusammen, so dass bei der graphischen Darstellung der Beziehung zwischen zwei Objekten eine Richtungsangabe erforderlich wird. Wir symbolisieren durch einen Pfeil das Senden einer Nachricht, so dass eine Beziehung $a_1 \rightarrow a_2$ bedeutet, dass das Objekt a_1 in seiner Rolle als „Sender“ eine Nachricht an das Objekt a_2 schickt und dass das Objekt a_2 in seiner Rolle als „Empfänger“ eine Nachricht von a_1 erhält.

In den folgenden Abschnitten untersuchen wir die 7 interessierenden Rekursiv-Beziehungstypen in der Reihenfolge, die durch die laufende Zahl in den Kreisen der Tabelle 4-2 angegeben ist. Wir werden dabei wiederum sehen, dass sich einige Rekursiv-Beziehungstypen nicht im relationalen Datenbank-Modell repräsentieren lassen.

4.4.1 Der 1:1-Rekursiv-Beziehungstyp

Beim 1:1-Rekursiv-Beziehungstyp muss jedes Objekt des Objekttyps genau eine Nachricht an ein Objekt desselben Objekttyps senden. Umgekehrt muss jedes Objekt genau eine Nachricht von einem anderen Objekt empfangen. Abbildung 4-49 zeigt die vorliegenden Verhältnisse.

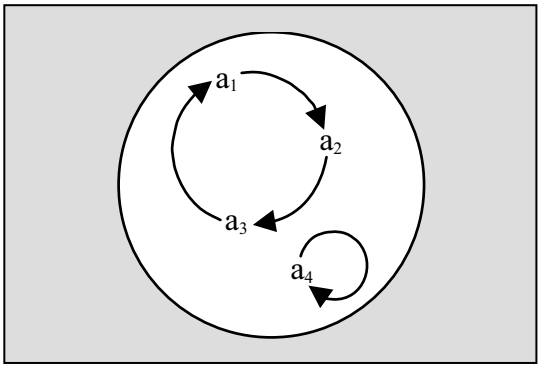


Abb. 4-49: 1:1-Rekursiv-Beziehungstyp

Wie man in der Abbildung sieht, können ausschließlich *Objekte-Zyklen* ($a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow a_1$) gebildet werden, die im Minimalfall nur ein Objekt enthalten und dann *Ein-Objekt-Zyklen* darstellen ($a_4 \rightarrow a_4$).

Bei der Repräsentation von Rekursiv-Beziehungstypen im relationalen Datenbank-Modell muss zunächst jedes Objekt des Objekttyps A - unabhängig von den anderen Objekten dieses Objekttyps – in einer Tabellenzeile gespeichert werden. Die Beziehung zwischen zwei Objekten a_1 und a_2 kann nun dadurch ausgedrückt werden, dass in der Tabellenzeile von a_2 (beim Empfänger) auf das Objekt a_1 (auf den Sender) verwiesen wird. Der Verweis wird – wie schon bei den dualen Beziehungstypen – durch Abspeichern des Identifikators von a_1 realisiert. Dieser Identifikator ist aber der Wert, den der Primärschlüssel von A im Falle des Objekts a_1 annimmt. Die Tabelle für den Objekttyp A muss somit den Schlüssel des Objekttyps A in doppelter Ausführung aufnehmen:

Schlüssel in doppelter Ausführung

1. als *Primärschlüssel*, um jede Tabellenzeile eindeutig identifizieren zu können;
2. als *Fremdschlüssel*, um auf eine andere (oder auch auf dieselbe) Zeile der Tabelle, nämlich auf den Sender verweisen zu können.

Da die Spaltenbezeichnungen einer Tabelle paarweise voneinander verschieden sein müssen, ist es erforderlich, für das Attribut (bzw. die Attribute) des Primärschlüssels im Falle ihrer „Wiedergeburt“ als Fremdschlüssel andere Bezeichnungen zu vergeben.

Transformationsregel T13 (1:1-Rekursiv-Beziehungstyp):

Entity-Relationship-Modell		**Relationales Datenbank-Modell**
Objekttyp A mit Schlüssel SA	⇒	Tabelle A mit Primärschlüssel **SA**
1:1-Rekursiv-Beziehungstyp	⇒	**SA** wird in A mit anderen Attributbezeichnungen „gedoppelt“ und als eingabepflichtiger unikaler Fremdschlüssel ⇑**SA'**⇑ vereinbart, der auf den Sender verweist.
Speicherbedarf für Beziehungstyp		$\overline{=}_{A} \cdot Len(SA)$

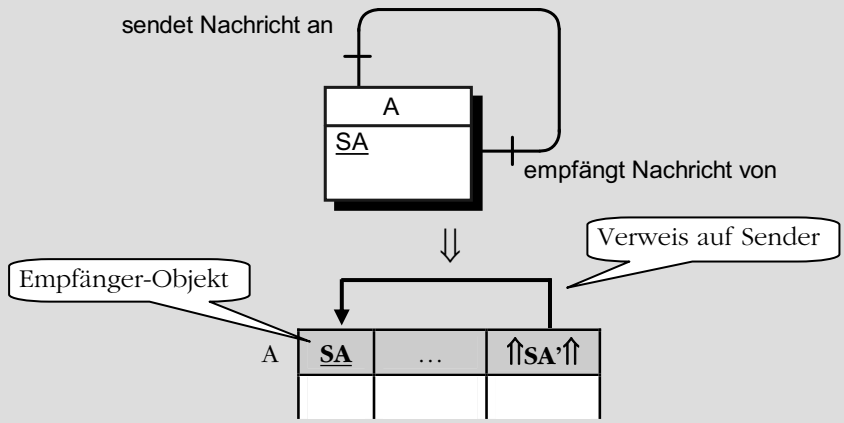


Abb. 4-50: Transformation eines 1:1-Rekursiv-Beziehungstyps

Im relationalen Datenbank-Modell lässt sich die Forderung, dass jedes Objekt genau eine Nachricht empfangen muss, einfach dadurch sichern, dass der Fremdschlüssel, der auf den Sender verweist, als *eingabepflichtig* deklariert wird, dass also jedes Objekt auf „sein" Sender-Objekt verweisen muss. Dass ein und dasselbe Objekt nicht von mehreren Empfängern als „ihr" Sender angegeben werden kann, wird durch die *Unikalität* des Fremdschlüssels verhindert.

Transformationsregel T13

Wie wir aber bei den dualen Beziehungstypen im Abschnitt 4.3 gesehen haben, lässt es sich im relationalen Datenbank-Modell im Allgemeinen nicht durchsetzen, dass auf jedes Objekt auch tatsächlich ein Verweis zeigt, dass also jedes Objekt ein Sender sein muss. Damit könnte man eigentlich einen 1:1-Rekursiv-Beziehungstyp gar nicht ohne Semantikverlust repräsentieren. Nun kommt aber bei Rekursiv-Beziehungstypen die zusätzliche Bedingung hinzu, dass Sender und Empfänger im selben Objekttyp A liegen. Diese Bedingung erzwingt, dass jede von einem A-Objekt empfangene Nachricht auch von einem A-Objekt gesendet werden muss, dass also auf jedes A-Objekt tatsächlich genau ein Verweis zeigt. Die Transformationsregel T13 ist in Abbildung 4-50 dargestellt.

Beispiel für Transformationsregel T13

Als Beispiel betrachten wird eine Turnergruppe, die bei einem Sportfest einen Kreis bilden soll. Für jeden Turner wird festgelegt, wer sein rechter Nachbar ist. In Abbildung 4-51 ist die erforderliche Transformation wiedergegeben.

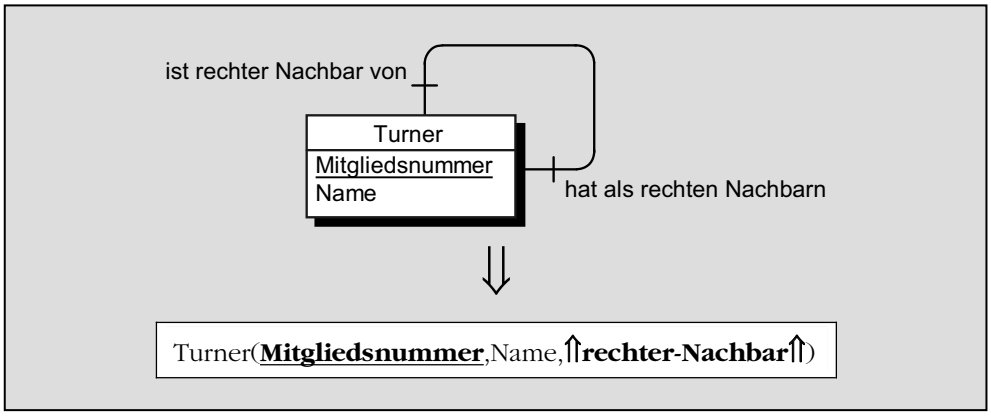


Abb. 4-51: Beispiel für die Transformationsregel T13

Von jedem Turner aus wird obligatorisch auf seinen (einzigen) rechten Nachbarn verwiesen. Durch die Unikalität des Fremdschlüssels wird außerdem gesichert, dass ein Turner nur für *einen* anderen Turner rechter Nachbar sein kann. Da jedem Turner ein rechter Nachbar zugeordnet wird, muss auch jeder Turner rechter Nachbar genau eines anderen Turners sein. Damit ist die Kreisstruktur der Aufstellung der Turner erzwungen. Allerdings lässt sich durch die Tabellen-Typbeschreibungen nicht ausschließen, dass ein Turner als sein eigener rechter Nachbar eingesetzt wird (Ein-Objekt-Zyklus).

Integritätsverletzung

Die Vereinbarung eines 1:1-Rekursiv-Beziehungstyps durch eine entsprechende Tabellen-Typbeschreibung ist somit in einfacher Weise möglich. Schwieriger wird es, die Einspeicherung von Datensätzen in die Tabelle derart vorzunehmen, dass keine Integritätsverletzungen auftreten. Dabei muss man zwei Fälle unterscheiden:

1. *Ein-Objekt-Zyklen sind erlaubt:* Der erste Datensatz lässt sich speichern, wenn er auf sich selbst verweist, also einen Ein-Objekt-Zyklus bildet. Jeder weitere Datensatz muss entweder wieder auf sich selbst verweisen oder er muss im Rahmen einer Transaktion in einen der bereits bestehenden Objekte-Zyklen eingefügt werden.
2. *Ein-Objekt-Zyklen sind nicht erlaubt:* Bei der ersten Speicherung müssen in einer Transaktion mindestens 2 Datensätze gespeichert werden, die aufeinander verweisen und dadurch einen Zwei-Objekte-Zyklus bilden. Soll jedoch ein neuer Zwei-Objekte-Zyklus angelegt werden, müssen in gleicher Weise 2 Datensätze gespeichert werden. Soll ein Datensatz allein gespeichert werden, so muss er im Rahmen einer Transaktion in einen bereits bestehenden Objekte-Zyklus eingefügt werden.

In unserem Turner-Beispiel sind Ein-Objekt-Zyklen verboten. Schließlich kann ein Turner nicht selbst sein rechter Nachbar sein! Soll in einen bisher aus drei Turnern bestehenden Kreis („12“→„15“→„17“→„12“) als rechter Nachbar des Turners „17“ der Turner mit der Mitgliedsnummer „22“ eingefügt werden, so sind in einer Transaktion zwei Aktivitäten auszuführen:

1. Speichern des Datensatzes für Turner „22“ mit dem Verweis auf den Turner „12“ als rechten Nachbarn.
2. Ändern des Verweises bei Turner „17“, der nun als rechten Nachbarn den neuen Turner „22“ erhält.

Die Abbildung 4-52 zeigt die Tabelle vor, während und nach der Transaktion.

vor Beginn der Transaktion:

Turner

Mitgliedsnummer	Name	rechter Nachbar
12	Jahn	15
15	Meier	17
17	Müller	12

nach der 1. Aktivität der Transaktion:

Turner

Mitgliedsnummer	Name	rechter Nachbar
12	Jahn	15
15	Meier	17
17	Müller	12
22	Lehmann	12

nach Beendigung der Transaktion:

Turner

Mitgliedsnummer	Name	rechter Nachbar
12	Jahn	15
15	Meier	17
17	Müller	22
22	Lehmann	12

Abb. 4-52: Transaktion zur Speicherung bei einem 1:1-Rekursiv-Beziehungstyp

Nach Abschluss der ersten Aktivität ist die Datenintegrität verletzt, weil zwei Turner als rechten Nachbarn den Turner „12“ ausweisen. Erst nach der zweiten Aktivität ist diese Verletzung wieder beseitigt. Man erkennt, dass beide Aktivitäten unbedingt in einer Transaktion ablaufen müssen, die sichert, dass entweder beide Aktivitäten gemeinsam ausgeführt werden oder dass keine der Aktivitäten ausgeführt wird.

4.4.2 Der C:C-Rekursiv-Beziehungstyp

Beim C:C-Rekursiv-Beziehungstyp kann ein Objekt keine oder eine Nachricht an ein anderes Objekt (oder an sich selbst) senden. Andererseits kann ein Objekt keine oder eine Nachricht empfangen. In der Abbildung 4-53 ist diese Situation schematisch dargestellt.

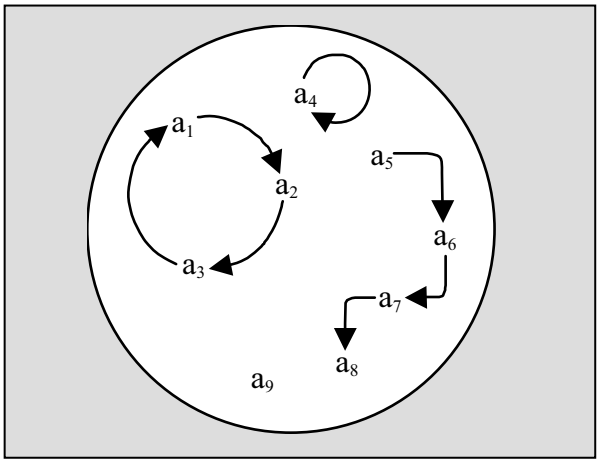


Abb. 4-53: C:C-Rekursiv-Beziehungstyp

Beim C:C-Rekursiv-Beziehungstyp sind also – wie schon beim 1:1-Rekursiv-Beziehungstyp – *Objekte-Zyklen* ($a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow a_1$) möglich, die im Minimalfall *Ein-Objekt-Zyklen* ($a_4 \rightarrow a_4$) sind. Darüber hinaus kann es *Objekte-Ketten* ($a_5 \rightarrow a_6 \rightarrow a_7 \rightarrow a_8$) geben, die gegebenenfalls nur ein einziges Objekt enthalten (a_9). Ein solches *singuläres Objekt* ist dann weder Sender noch Empfänger.

Die Art der Realisierung eines C:C-Rekursiv-Beziehungstyps im relationalen Datenbank-Modell hängt davon ab, ob es viele oder wenige Objekte gibt, die Empfänger einer Nachricht sind.

Transformationsregel T14

Betrachten wir zunächst den Fall, dass nahezu alle Objekte eine Nachricht empfangen, also einen Verweis auf „ihren“ Sender benötigen. Dieser Sender kann der Empfänger selbst oder ein anderes A-Objekt sein. Da nicht jedes Objekt ein Empfänger ist, muss der Fremdschlüssel, der auf den Sender verweist, als nicht-eingabepflichtig deklariert werden. Andererseits muss er

unikal sein, damit ein Objekt a_1 nur von höchstens einem Objekt a_2 als „sein“ Sender ausgewiesen werden kann. Abbildung 4-54 zeigt die entsprechende Transformationsregel T14.

Transformationsregel T14 (C:C-Rekursiv-Beziehungstyp mit vielen Empfängern):

Entity-Relationship-Modell		**Relationales Datenbank-Modell**
Objekttyp A mit Schlüssel SA	⇒	Tabelle A mit Primärschlüssel **SA**
C:C-Rekursiv-Beziehungstyp	⇒	**SA** wird in A mit anderen Attributbezeichnungen „gedoppelt“ und als nicht-eingabepflichtiger unikaler Fremdschlüssel ⇑***SA'***⇑ vereinbart, der auf den Sender verweist.
Speicherbedarf für Beziehungstyp		$\overline{\overline{A}} \cdot Len(SA)$

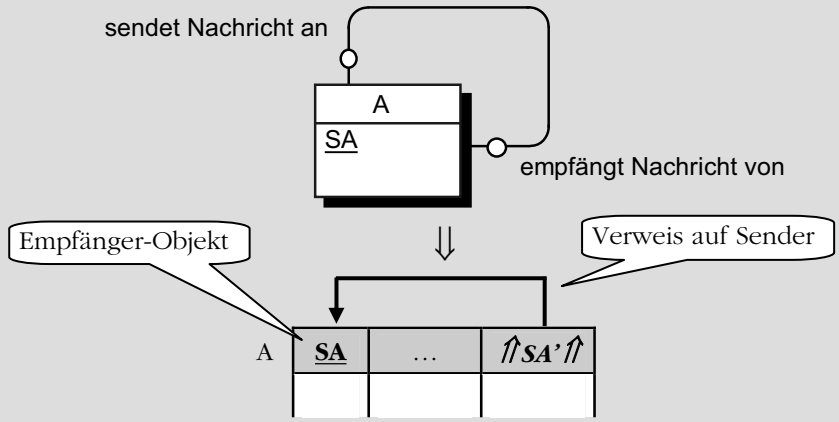


Abb. 4-54: Transformation eines C:C-Rekursiv-Beziehungstyps mit vielen Empfängern

Beispiel für Transformationsregel T14

Nehmen wir als Beispiel an, dass in einem Busreise-Unternehmen festgelegt wird, welcher Bus als Ersatz verwendet werden soll, wenn ein Bus defekt ist. Dabei soll ein Bus höchstens für einen anderen als Ersatz dienen. Für die wenigen neuen Busse

mit geringer Ausfallwahrscheinlichkeit wird kein Ersatz vorgesehen, und sie können auch nicht als Ersatz dienen, da sie ständig im Einsatz sind (singuläre Objekte). Für andere Busse wird festgelegt, dass sie bei Schäden schnell repariert werden; sie sind dann Ersatz für sich selbst (Ein-Objekt-Zyklen). Typischerweise wird für jeden Bus ein anderer als Ersatz festgelegt und jeder Bus kann als Ersatz für einen Bus dienen (Objekte in einem Mehr-Objekte-Zyklus oder „innere Objekte“ in einer Objekte-Kette). Es ist in seltenen Fällen aber auch nicht auszuschließen, dass ein Bus zwar nicht als Ersatz dienen kann, aber im Notfall einen Ersatz braucht (Anfangsglied einer Objekte-Kette). Ebenso kann ein Bus lediglich in der Reserve gehalten werden, also nur als Ersatz dienen, ohne selbst einen Ersatz zu benötigen (Endglied einer Objekte-Kette).

Es gibt nur wenige Busse, die nicht auf „ihren“ Ersatzbus verweisen. Die Repräsentation dieses C:C-Rekursiv-Beziehungstyps erfolgt also nach der Transformationsregel T14, wie dies in Abbildung 4-55 gezeigt wird.

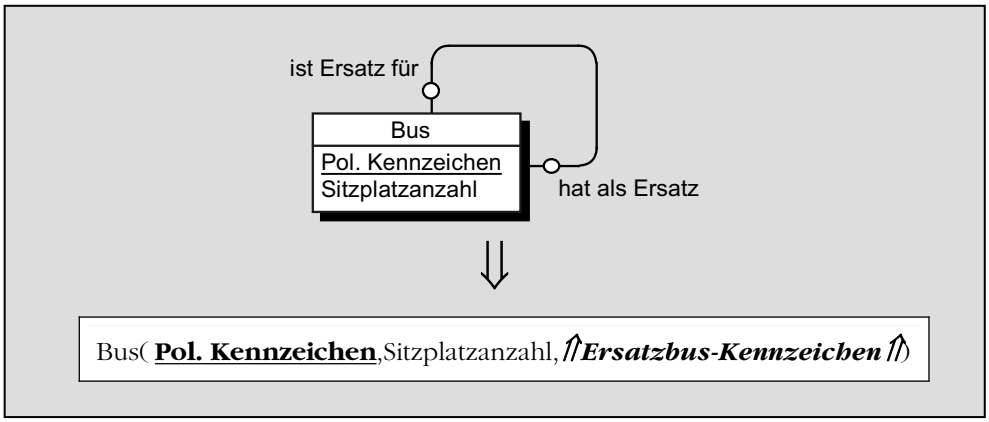


Abb. 4-55: Beispiel für die Transformationsregel T14

Da der Fremdschlüssel ***⇑Ersatzbus-Kennzeichen⇑*** nicht-eingabepflichtig ist, kann ein Bus auf keinen oder auf einen Bus (auch auf sich selbst) als Ersatzbus verweisen. Da der Fremdschlüssel unikal ist, kann auf einen Ersatzbus nur von *einem* anderen Bus verwiesen werden. Andererseits ist nicht gefordert – und kann auch gar nicht gefordert werden -, dass jeder Wert des Primärschlüssels **Pol. Kennzeichen** auch tatsächlich als Fremdschlüs-

selwert auftritt. Somit kann es auch Busse geben, die nicht durch einen Fremdschlüsselwert als Ersatzbusse ausgewiesen sind.

C:C-Rekursiv-Beziehungstyp mit wenigen Empfängern

Gibt es in einem Objekttyp A, dessen Objekte durch einen C:C-Rekursiv-Beziehungstyp miteinander verbunden sind, nur wenige Empfänger-Objekte, dann wird der Fremdschlüssel, der auf den Sender verweist, häufig mit der NULL-Marke belegt. Die Transformationsregel T14 führt dann zu einer ungünstigen Tabellen-Struktur. In diesem Fall ist es besser, wenn man eine Koppel-Tabelle A/A einführt, in der die Sender-Empfänger-Kopplungen gespeichert werden.

Die Repräsentation des C:C-Rekursiv-Beziehungstyps durch eine Koppel-Tabelle entspricht im Datenmodell der Umwandlung des Rekursiv-Beziehungstyps in einen neuen Objekttyp, mit dem der Objekttyp A durch zwei 1:C-Beziehungstypen verbunden ist. Wir hatten die bisherigen Beziehungstyp-Richtungen mit der Semantik „A *sendet Nachricht an* A“ bzw. „A *empfängt Nachricht von* A“ belegt. Der neue Objekttyp entspricht dann einer „Sendung“, die von einem A-Objekt ausgelöst bzw. von einem A-Objekt empfangen wird. Dieses Umwandlungsverfahren ist in der Abbildung 4-56 dargestellt.

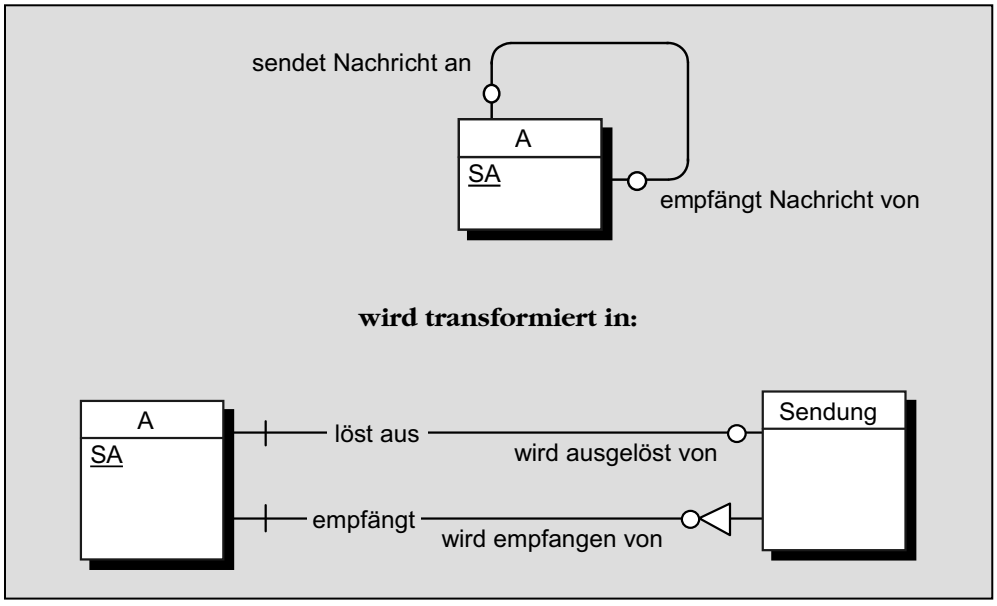


Abb. 4-56: Umwandlung eines C:C-Rekursiv-Beziehungstyps in zwei 1:C-Beziehungstypen

Jede der Beziehungstyp-Richtungen, die von der „Sendung“ zu A führen, kann die Identifizierung der Objekte im Objekttyp „Sendung“ übernehmen. In Abbildung 4-56 wurde die Beziehungstyp-Richtung „Sendung *wird empfangen von* A“ gewählt.

Stellt man den neugebildeten Objekttyp „Sendung“ im relationalen Datenbank-Modell als Koppel-Tabelle A/A dar, so wird der Primärschlüssel von A in diese Tabelle in doppelter Ausführung aufgenommen:

1. als Fremdschlüssel, der auf den *Empfänger* einer Nachricht verweist, und
2. als Fremdschlüssel, der auf den *Sender* der Nachricht verweist.

Transformationsregel T15

Jeder dieser Fremdschlüssel ist für sich unikal. Damit kann ein Objekt nur einmal als Empfänger bzw. als Sender auftreten. Beide Fremdschlüssel werden als eingabepflichtig vereinbart. So muss es zu jedem Empfänger auch einen Sender geben und umgekehrt. Da jeder Fremdschlüssel unikal ist, kann einer der beiden als Primärschlüssel der Koppel-Tabelle dienen. Wir wählen bei unserer Darstellung den Fremdschlüssel, der auf den Empfänger verweist. Die Transformationsregel T15 ist in der Abbildung 4-57 wiedergegeben.

Die Koppel-Tabelle A/A symbolisiert den Beziehungstyp als Menge der Paarungen (Sender↔Empfänger). $\overline{\overline{A/A}}$ ist dann die Mächtigkeit des Beziehungstyps, also die Anzahl der Paarungen. Das Produkt aus der Anzahl dieser Paarungen und der doppelten Länge des Fremdschlüssels drückt den Speicherbedarf für den Beziehungstyp aus.

Beispiel für Transformationsregel T15

Die Transformationsregel T15 soll am folgenden Beispiel veranschaulicht werden. In großen Städten gibt es einige wenige lange Straßenzüge, die - aus historischen Gründen oder im Interesse einer besseren Orientierungsmöglichkeit - abschnittsweise unterschiedliche Namen tragen. Wir nehmen nun an, dass ein Straßenabschnitt nur einen anderen Straßenabschnitt als Nachfolger haben kann und selbst höchstens Nachfolger eines Straßenabschnitts sein kann. Dabei muss man natürlich eine Nachfolge-Richtung vereinbaren. Diese könnte vom Stadtzentrum zur Stadtgrenze zeigen oder – bei Ringstraßen – dem Uhrzeigersinn folgen. Da man die Nachfolge-Beziehung in der Menge der Straßen eher als Ausnahme werten muss, ist die Transformationsregel T15 anzuwenden, wie dies Abbildung 4-58 zeigt.

Transformationsregel T15 (C:C-Rekursiv-Beziehungstyp mit wenigen Empfängern):

Entity-Relationship-Modell		Relationales Datenbank-Modell
Objekttyp A mit Schlüssel SA	⇒	Tabelle A mit Primärschlüssel **SA**
C:C-Rekursiv-Beziehungstyp	⇒	Koppel-Tabelle A/A, die **SA** in zwei Exemplaren als eingabepflichtige unikale Fremdschlüssel enthält: als Empfänger-Fremdschlüssel ⇑**SA**⇑ und als Sender-Fremdschlüssel ⇑**SA'**⇑. Der Empfänger-Fremdschlüssel bildet den Primärschlüssel von A/A.
Speicherbedarf für Beziehungstyp		$\overline{\overline{A/A}} \cdot 2 \cdot Len(SA)$

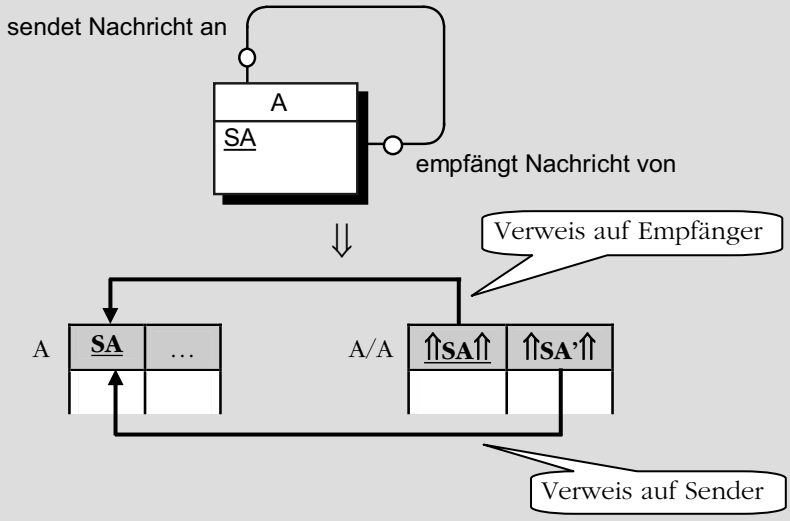


Abb. 4-57: Transformation eines C:C-Rekursiv-Beziehungstyps mit wenigen Empfängern

Als Primärschlüssel der Koppel-Tabelle „Straßenabschnitt/Straßenabschnitt“ wurde der Name des Vorgänger-Straßenabschnitts gewählt. Da jedes Exemplar des Fremdschlüssels *für sich* unikal ist, kann jeder Straßenabschnitt höchstens Vorgänger oder Nach-

folger *eines* Straßenabschnitts sein. In einzelne Abschnitte gegliederte Ringstraßen sind auch möglich. Allerdings lassen sich durch die Typbeschreibungen Ein-Objekt-Zyklen nicht ausschließen (beide Fremdschlüssel haben dann denselben Wert). Dies muss durch das Anwendungsprogramm verhindert werden.

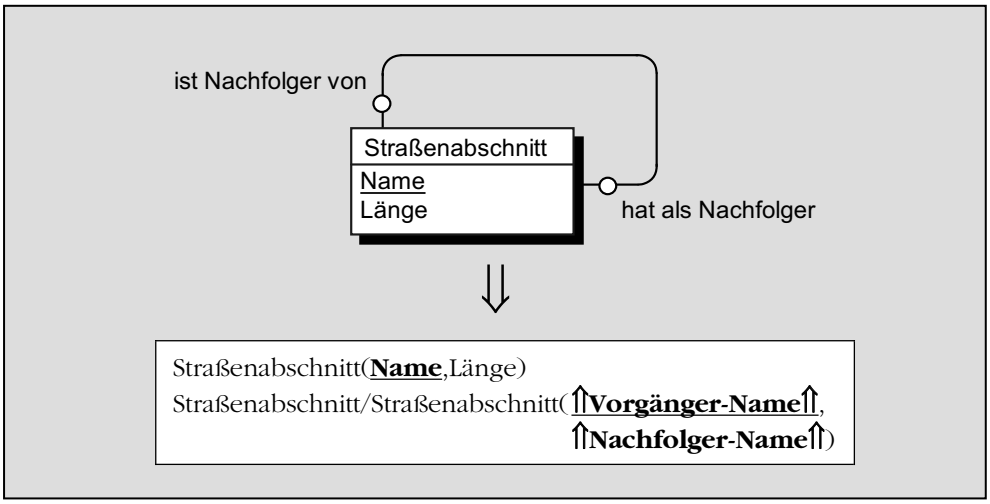


Abb. 4-58: Beispiel für die Transformationsregel T15

4.4.3 Der 1:CN-Rekursiv-Beziehungstyp

Beim 1:CN-Rekursiv-Beziehungstyp kann ein Objekt keine, eine oder mehrere Nachrichten senden, es muss aber stets genau eine Nachricht empfangen. Das entspricht den Bedingungen eines 1:1-Rekursiv-Beziehungstyps zuzüglich der Möglichkeit eines Objekts, entweder gar nicht als Sender oder aber als Sender mehrerer Nachrichten in Erscheinung zu treten. Da jedes Objekt des Objekttyps A genau eine Nachricht empfangen muss, müssen auch $\overline{\overline{A}}$ Nachrichten gesendet werden. Für jedes Nicht-Sender-Objekt muss somit ein anderes – gewissermaßen sein Vertreter – eine zusätzliche Nachricht senden. Diese Situation veranschaulicht die Abbildung 4-59.

Strukturen des Objekte-Graphen

Wie schon beim 1:1-Rekursiv-Beziehungstyp, so sind auch hier die Grundstrukturen des möglichen Objekte-Graphen *Objekte-Zyklen* ($a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow a_1$) mit dem Spezialfall der *Ein-Objekt-*

Zyklen ($a_6 \rightarrow a_6$). Jedes Objekt eines Zyklus' kann die Wurzel einer Monohierarchie[24], also eines Baumes, sein. In der Abbildung 4-59 sind dies die Objekte a_3 und a_6.

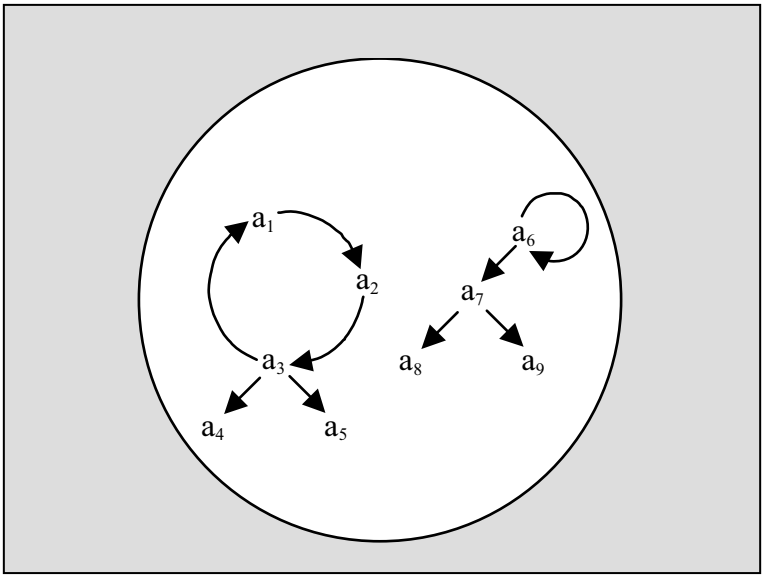


Abb. 4-59: 1:CN-Rekursiv-Beziehungstyp

Transformationsregel T16

Bei der Transformation des 1:CN-Rekursiv-Beziehungstyps muss der Identifikator des Senders eingabepflichtig beim Empfänger hinterlegt werden. Wird dieser Fremdschlüssel als nicht-unikal deklariert, kann ein Objekt von mehreren Empfänger-Objekten als „ihr" Sender ausgewiesen werden. Abbildung 4-60 zeigt die Transformationsregel T16.

Beispiel für Transformationsregel T16

Nehmen wir als Beispiel an, dass für die Festschrift eines Vereins von allen Mitgliedern ein Foto benötigt wird. Im Interesse der Kostendämpfung wird kein Berufs-Fotograf hinzugezogen. Jedes Mitglied muss genau einmal fotografiert werden. Manche Mitglieder fotografieren überhaupt nicht, andere müssen dafür umso mehr Fotos machen. Damit auch wirklich jeder Fotograf selbst

[24] In einer Monohierarchie hat jedes Objekt kein, ein oder mehre untergeordnete Objekte, jedes Objekt – mit Ausnahme der Wurzel - hat aber genau ein übergeordnetes Objekt.

auf einem Foto zu sehen ist, ist es letztlich erforderlich, dass sich ein Fotograf entweder selbst fotografiert (Ein-Objekt-Zyklus) oder dass ihn jemand fotografiert, der selbst bereits fotografiert wurde (Mehr-Objekte-Zyklus). Die erforderliche Transformation zeigt die Abbildung 4-61.

Transformationsregel T16 (1:CN-Rekursiv-Beziehungstyp):

Entity-Relationship-Modell		**Relationales Datenbank-Modell**
Objekttyp A mit Schlüssel SA	⇒	Tabelle A mit Primärschlüssel **SA**
1:CN-Rekursiv-Beziehungstyp	⇒	**SA** wird in A mit anderen Attributbezeichnungen „gedoppelt“ und als eingabepflichtiger nicht-unikaler Fremdschlüssel ⇑SA'⇑ vereinbart, der auf den Sender verweist.
Speicherbedarf für Beziehungstyp		$\overline{\overline{A}} \cdot Len(SA)$

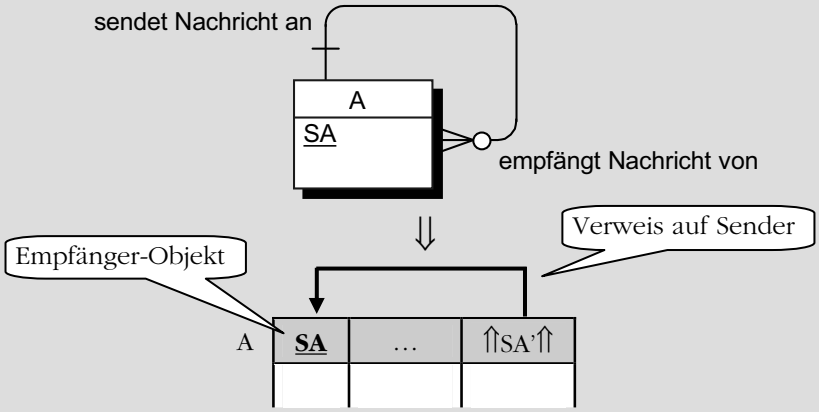


Abb. 4-60: Transformation eines 1:CN-Rekursiv-Beziehungstyps

Da der Fremdschlüssel ⇑Fotografen-Mitgliedsnummer⇑ eingabepflichtig ist, muss jedes Mitglied auf ein anderes Mitglied oder auf sich selbst als Fotograf verweisen. Da der Fremdschlüssel

nicht-unikal ist, können sich mehrere Mitglieder auf denselben Fotografen beziehen.

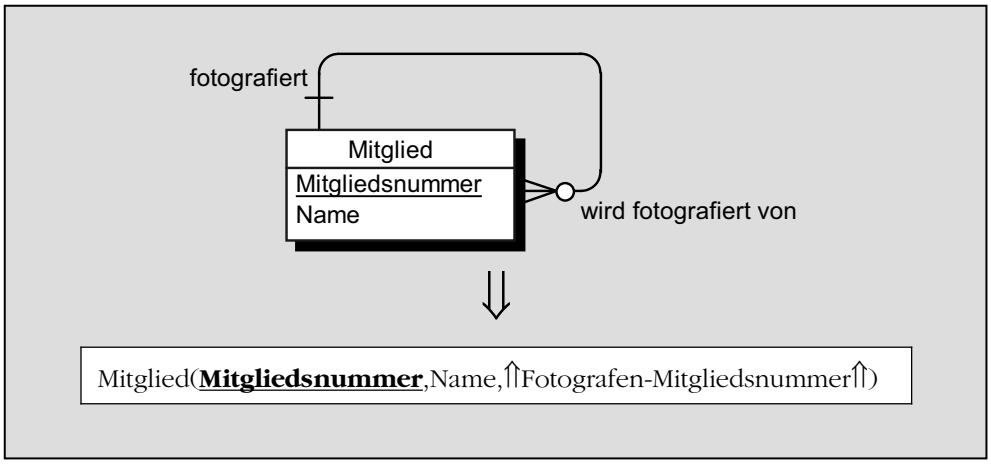


Abb. 4-61: Beispiel für die Transformationsregel T16

4.4.4 Der C:CN-Rekursiv-Beziehungstyp

Der C:CN-Rekursiv-Beziehungstyp unterscheidet sich von dem 1:CN-Rekursiv-Beziehungstyp nur durch die Aufhebung der Forderung, dass jedes Objekt eine Nachricht empfangen muss. Die Abbildung 4-62 zeigt dafür ein repräsentatives Beispiel.

Die Palette der möglichen Strukturen des Objekte-Graphen wird um solche Monohierarchien erweitert, deren Wurzeln nicht mehr Elemente eines Objekte-Zyklus' sind. Der Baum mit der Objekte-Menge $\{a_{10}, a_{11}, a_{12}, a_{13}, a_{14}, a_{15}\}$ ist ein Beispiel dafür. Der C:CN-Rekursiv-Beziehungstyp ist somit prädestiniert für die Repräsentation von Monohierarchien, wie sie in der Praxis beispielsweise in Form von Organigrammen oder Stücklisten auftreten. Als Spezialfall eines Baumes ist auch eine Objekte-Liste ($a_{16} \rightarrow a_{17}$) möglich. Diese kann auch nur aus einem Element (a_{18}) bestehen.

Transformationsregel T17

Bei der Transformation des C:CN-Rekursiv-Beziehungstyps in das relationale Datenbank-Modell ist zu berücksichtigen, ob es viele oder wenige Objekte gibt, die keine Nachricht empfangen. Sind die meisten Objekte Empfänger einer Nachricht, dann verweisen diese Objekte auf „ihren" jeweiligen Sender. Deshalb wird der

Fremdschlüssel als nicht-eingabepflichtig und nicht-unikal deklariert. Die Transformationsregel T17 ist in Abbildung 4-63 dargestellt.

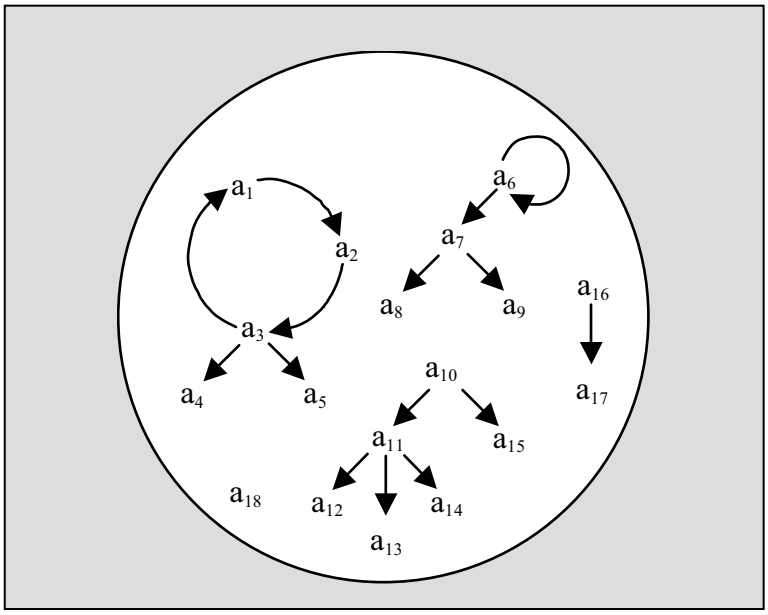


Abb. 4-62: C:CN-Rekursiv-Beziehungstyp

Beispiel für Transformationsregel T17

Betrachten wir als Beispiel die Leitungshierarchie im Unternehmen. Ein Mitarbeiter kann ohne Leitungsfunktion sein, er kann aber auch andere Mitarbeiter anleiten. Dies kann über mehrere Hierarchiestufen erfolgen. Andererseits haben die meisten Mitarbeiter genau einen Chef. Die Mitarbeiter der obersten Leitungsebene haben keinen Chef. Die Transformation ist in der Abbildung 4-64 wiedergegeben.

Der Fremdschlüssel *⇑Chef-Personalnummer⇑* ist nicht-eingabepflichtig – somit kann es Mitarbeiter ohne Chef geben. Da der Fremdschlüssel nicht-unikal ist, können mehrere Mitarbeiter auf denselben Chef verweisen. An diesem Beispiel sieht man deutlich, dass der C:CN-Rekursiv-Beziehungstyp wesentlich mehr Objekte-Strukturen umfasst, als für die Repräsentation monohierarchischer Zusammenhänge benötigt werden:

Transformationsregel T17 (C:CN-Rekursiv-Beziehungstyp mit vielen Empfängern):

Entity-Relationship-Modell		Relationales Datenbank-Modell
Objekttyp A mit Schlüssel SA	⇒	Tabelle A mit Primärschlüssel **SA**
C:CN-Rekursiv-Beziehungstyp	⇒	**SA** wird in A mit anderen Attributbezeichnungen „gedoppelt“ und als nicht-eingabepflichtiger nicht-unikaler Fremdschlüssel ⇑SA'⇑ vereinbart, der auf den Sender verweist.
Speicherbedarf für Beziehungstyp		$\overline{\overline{A}} \cdot Len(SA)$

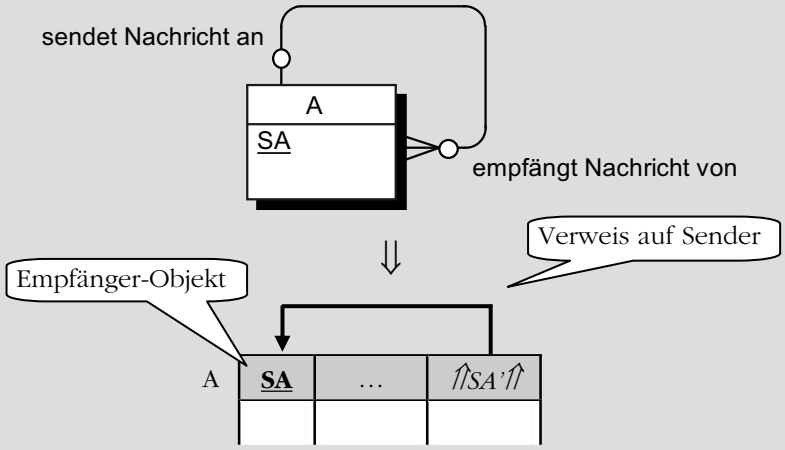


Abb. 4-63: Transformation eines C:CN-Rekursiv-Beziehungstyps mit vielen Empfängern

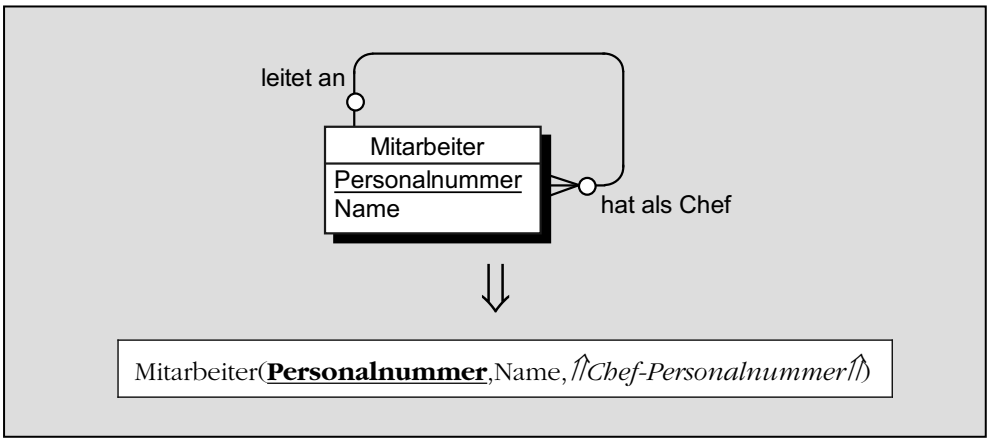


Abb. 4-64: Beispiel für die Transformationsregel T17

- *Mehr-Objekte-Zyklen:* Der Mitarbeiter mit der Personalnummer „0815“ kann Chef des Mitarbeiters „0816“ sein. Dieser kann den Mitarbeiter „0817“ anleiten, der wiederum Chef des Mitarbeiters „0815“ ist.
- *Ein-Objekt-Zyklen:* Ein Mitarbeiter kann als sein eigener Chef ausgewiesen sein, wenn nämlich in einem Datensatz die Werte von Primärschlüssel und Fremdschlüssel übereinstimmen.
- *Mehr-Objekte-Ketten:* Ein Mitarbeiter kann einen einzigen Mitarbeiter anleiten, der wiederum Chef eines einzigen Mitarbeiters ist.
- *Ein-Objekt-Ketten:* Ein Mitarbeiter, der keinen Chef hat, leitet selbst auch niemanden an (singuläres Objekt).

In der Praxis sind solche Situationen gewöhnlich verboten, sie können aber durch die Tabellen-Typbeschreibungen nicht verhindert werden. Die Anwendungs-Software muss sichern, dass die beschriebenen Möglichkeiten des C:CN-Rekursiv-Beziehungstyps nicht realisiert werden.

C:CN-Rekursiv-Beziehungstyp mit wenigen Empfängern

Betrachten wir nun den Fall, dass es viele Objekte gibt, die keine Nachricht empfangen. Man hat es dann mit Objekte-Strukturen zu tun, bei denen viele singuläre Objekte und/oder viele kurze Bäume bzw. Ketten existieren. In einer solchen Situation wäre der Fremdschlüssel, der ja auf den Sender verweist, häufig mit der NULL-Marke belegt. Dann ist es günstiger, die wenigen Emp-

fänger-Sender-Kopplungen in einer gesonderten Koppel-Tabelle A/A zu repräsentieren.

Koppel-Tabelle

Die Darstellung des C:CN-Rekursiv-Beziehungstyps durch eine Koppel-Tabelle entspricht einer Umwandlung des Datenmodells, bei der der Rekursiv-Beziehungstyp zu einem neuen Objekttyp „Sendung“ wird. Der Objekttyp A ist mit dem Koppel-Objekttyp „Sendung“ durch einen 1:CN-Beziehungstyp und durch einen 1:C-Beziehungstyp verbunden. Abbildung 4-65 zeigt das Umwandlungsverfahren.

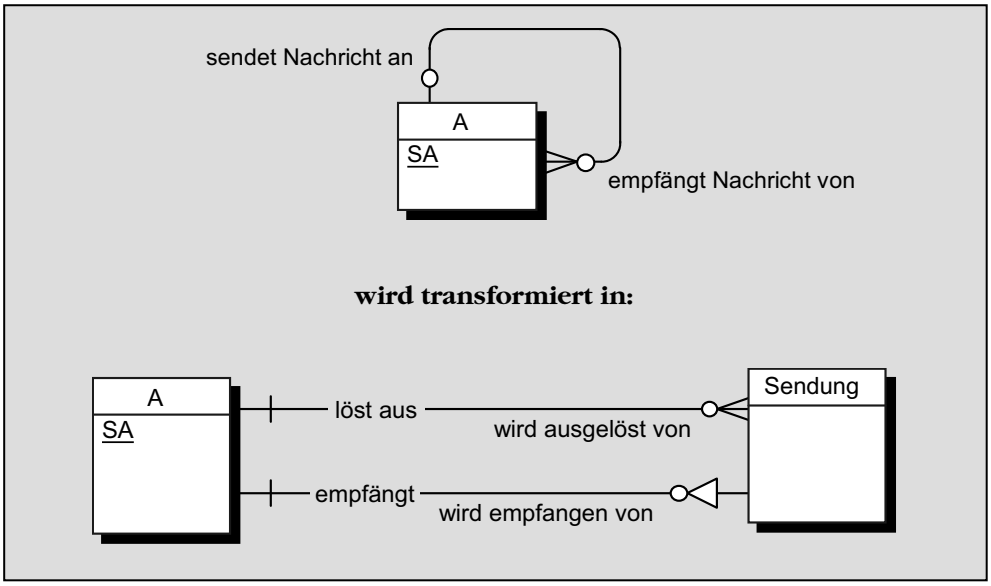


Abb. 4-65: Umwandlung eines C:CN-Rekursiv-Beziehungstyps in einen 1:CN- und einen 1:C-Beziehungstyp

Die Beziehungstyp-Richtung „Sendung *wird empfangen von* A“ übernimmt die Identifizierung der Objekte im Objekttyp „Sendung“.

Stellt man den Objekttyp „Sendung“ im relationalen Datenbank-Modell als Koppel-Tabelle A/A dar, so taucht der Primärschlüssel von A in dieser Tabelle zweimal als eingabepflichtiger Fremdschlüssel auf. Er verweist einmal auf den Empfänger und einmal auf den Sender der Nachricht. Der Empfänger-Fremdschlüssel ist unikal: Ein Objekt kann somit nur einmal als Empfänger auftreten. Dieser Fremdschlüssel dient auch als Primärschlüssel der

Koppel-Tabelle. Der Sender-Fremdschlüssel wird dagegen als nicht-unikal vereinbart: mehrere Objekte können dann auf denselben Sender verweisen. Die Transformationsregel T18 in der Abbildung 4-66 beschreibt dieses Vorgehen.

Transformationsregel T18 (C:CN-Rekursiv-Beziehungstyp mit wenigen Empfängern):

Entity-Relationship-Modell		**Relationales Datenbank-Modell**
Objekttyp A mit Schlüssel SA	⇒	Tabelle A mit Primärschlüssel **SA**
C:CN-Rekursiv-Beziehungstyp	⇒	Koppel-Tabelle A/A, die **SA** in zwei Exemplaren als eingabepflichtige Fremdschlüssel enthält. Der Empfänger-Fremdschlüssel ⇑**SA**⇑ wird als unikal vereinbart und bildet den Primärschlüssel von A/A, der Sender-Fremdschlüssel ⇑SA'⇑ wird als nicht-unikal vereinbart.
Speicherbedarf für Beziehungstyp		$\overline{\overline{A/A}} \cdot 2 \cdot Len(SA)$

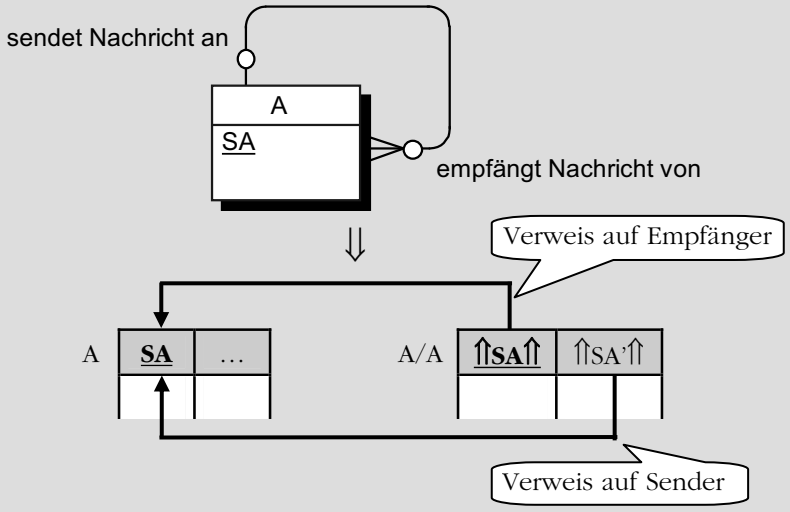


Abb. 4-66: Transformation eines C:CN-Rekursiv-Beziehungstyps mit wenigen Empfängern

Beispiel für Transformationsregel T18

Betrachten wir als Beispiel einen Chor, in dem es für die vier Stimmgruppen (Sopran, Alt, Tenor und Bass) jeweils einen Stimmgruppenführer gibt. Nach einer Chorprobe informiert der Stimmgruppenführer diejenigen Chormitglieder seiner Stimmgruppe, die bei einer Probe gefehlt haben, über die Festlegungen, die auf der Probe getroffen wurden. Es ist nun nach einer Chorprobe zu speichern, welches Chormitglied von welchem Stimmgruppenführer informiert wird. Da in einem guten Chor bei den Proben nur wenige Chormitglieder fehlen, ist die Transformationsregel T18 anzuwenden, wie dies Abbildung 4-67 zeigt.

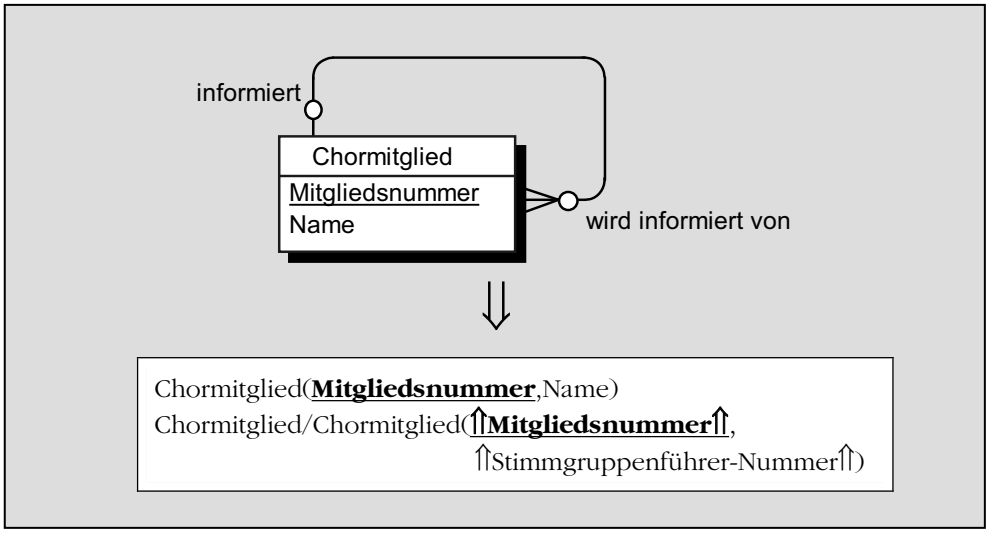


Abb. 4-67: Beispiel für die Transformationsregel T18

Der Fremdschlüssel ⇑**Mitgliedsnummer**⇑ ist unikal, ein Chormitglied kann also nur einmal informiert werden. Der Fremdschlüssel ⇑Stimmgruppenführer-Nummer⇑ ist dagegen nicht-unikal, so dass ein Stimmgruppenführer mehrere Chormitglieder informieren kann. Da nicht gefordert wird – und auch gar nicht gefordert werden kann -, dass jeder Wert des Primärschlüssels **Mitgliedsnummer** der Tabelle „Chormitglied“ als Wert einer der beiden Fremdschlüssel in der Tabelle „Chormitglied/Chormitglied“ auftauchen soll, kann es Chormitglieder geben, die nicht informieren, andere, die nicht informiert werden, und wieder andere, die weder informieren noch informiert werden.

4.4.5 Der CM:CN-Rekursiv-Beziehungstyp

Der CM:CN-Rekursiv-Beziehungstyp stellt keinerlei einschränkende Bedingungen an die Struktur, nach der die Objekte eines Objekttyps A miteinander in Beziehung stehen. Jedes Objekt kann keine, eine oder mehrere Nachrichten senden, ebenso kann jedes Objekt keine, eine oder auch mehrere Nachrichten empfangen. Die Abbildung 4-68 vermittelt davon einen Eindruck, ohne dass sie alle Möglichkeiten berücksichtigen kann.

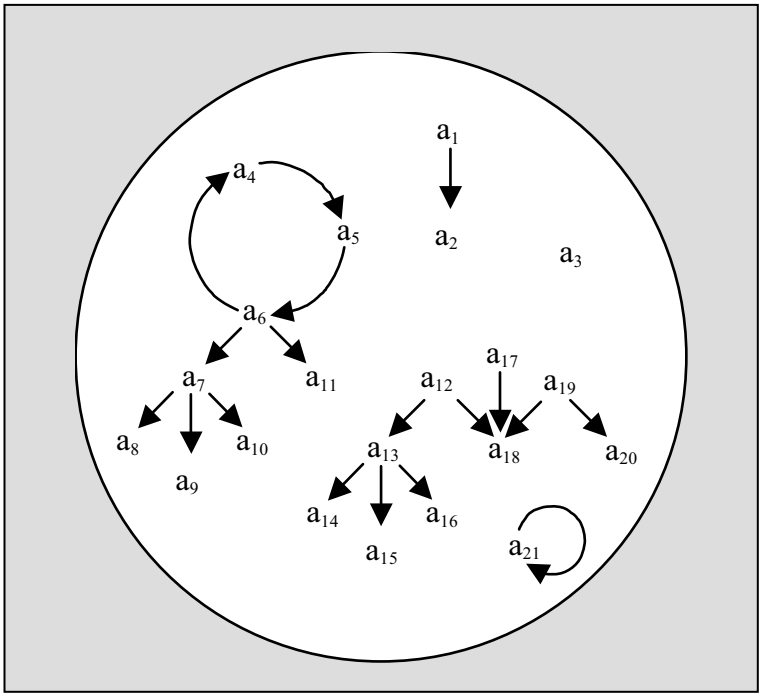


Abb. 4-68: CM:CN-Rekursiv-Beziehungstyp

Beliebig strukturierte Netzwerke

Es lassen sich beliebig strukturierte Netzwerke darstellen. Insbesondere können dies *Polyhierarchien*[25] wie in der Objektmenge $\{a_{12}, a_{13}, ..., a_{20}\}$ sein. In dieser Objektmenge hat das Objekt a_{18}

[25] In einer Polyhierarchie hat jedes Objekt kein, ein oder mehre untergeordnete Objekte, jedes Objekt kann kein, ein oder mehrere übergeordnete Objekte besitzen.

drei übergeordnete Objekte, nämlich die Elemente a_{12}, a_{17} und a_{19}. Als Sonderfälle sind natürlich auch Monohierarchien wie in der Objektmenge $\{a_6, a_7, a_8, a_9, a_{10}, a_{11}\}$ möglich. Diese entarten mitunter zur Objekte-Kette ($a_1 \rightarrow a_2$), die eventuell auch nur aus einem Element (a_3) bestehen kann. Die Objekte-Kette kann in sich geschlossen sein: sie wird dann zu einem Objekte-Zyklus ($a_4 \rightarrow a_5 \rightarrow a_6 \rightarrow a_4$), der im Minimalfall auch zu einem Ein-Objekt-Zyklus ($a_{21} \rightarrow a_{21}$) werden kann.

Koppel-Objekttyp

Der CM:CN-Rekursiv-Beziehungstyp lässt sich in das relationale Datenbank-Modell nur nach vorheriger Umwandlung in einen neuen Objekttyp überführen. Dieser Koppel-Objekttyp, den wir wieder als „Sendung“ bezeichnen wollen, repräsentiert die Sender-Empfänger-Beziehungen zwischen den einzelnen Objekten des Objekttyps A. Der Objekttyp A ist mit dem Objekttyp „Sendung“ durch zwei 1:CN-Beziehungstypen zu verbinden, wie dies aus der Abbildung 4-69 ersichtlich ist.

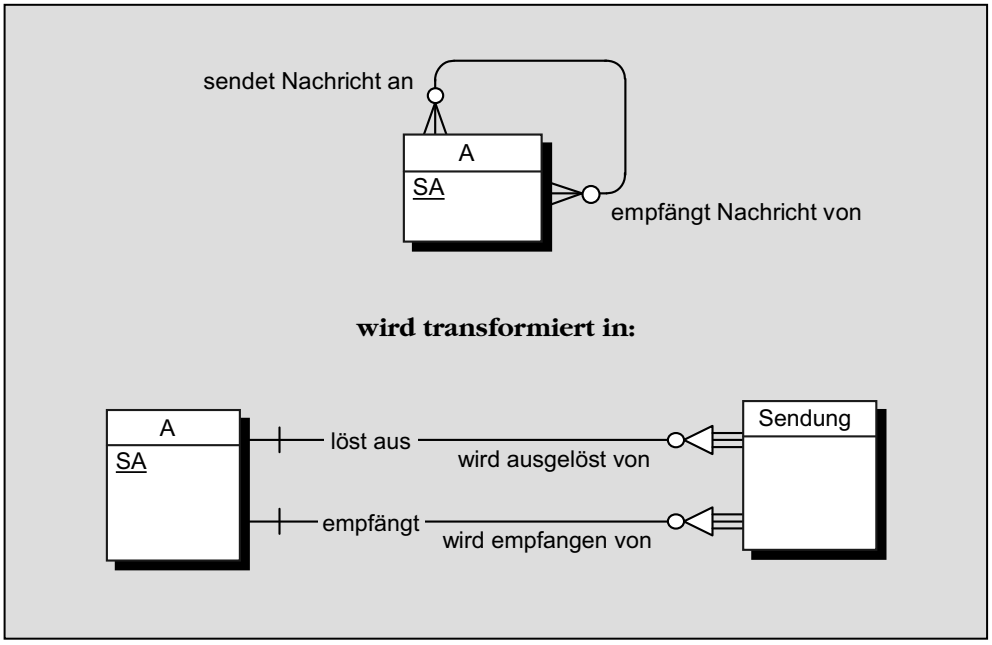


Abb. 4-69: Umwandlung eines CM:CN-Rekursiv-Beziehungstyps in zwei 1:CN-Beziehungstypen

Die Beziehungstyp-Richtungen „Sendung *wird ausgelöst von* A“ und „Sendung *wird empfangen von* A“ übernehmen gemeinsam die Identifizierung der Objekte im Objekttyp „Sendung“.

Transformationsregel T19 (CM:CN-Rekursiv-Beziehungstyp):

Entity-Relationship-Modell		**Relationales Datenbank-Modell**
Objekttyp A mit Schlüssel SA	⇒	Tabelle A mit Primärschlüssel **SA**
CM:CN-Rekursiv-Beziehungstyp	⇒	Koppel-Tabelle A/A, die **SA** in zwei Exemplaren als eingabepflichtige nicht-unikale Fremdschlüssel enthält: als Empfänger-Fremdschlüssel ⇑SA⇑ und als Sender-Fremdschlüssel ⇑SA'⇑. Die Kombination beider Fremdschlüssel bildet den Primärschlüssel von A/A.
Speicherbedarf für Beziehungstyp		$\overline{\overline{A/A}} \cdot 2 \cdot Len(SA)$

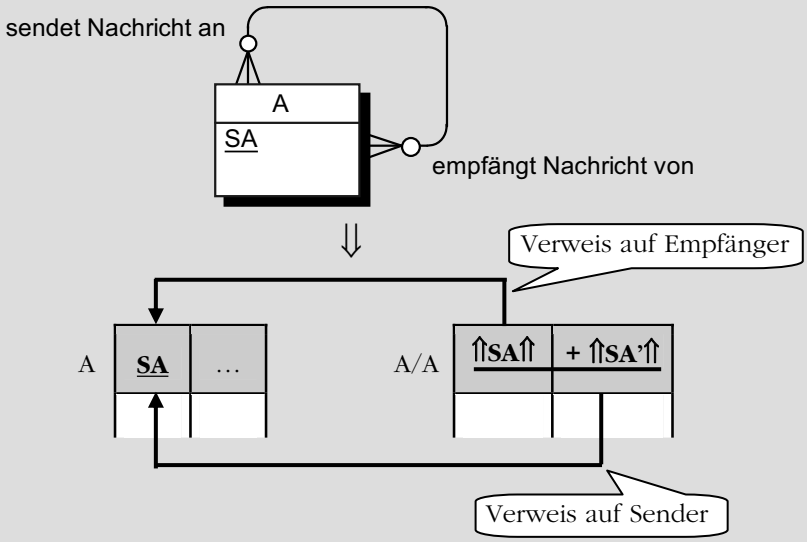


Abb. 4-70: Transformation eines CM:CN-Rekursiv-Beziehungstyps

Transformationsregel T19

Stellt man den Objekttyp „Sendung“ im relationalen Datenbank-Modell als Koppel-Tabelle A/A dar, so enthält A/A den Primärschlüssel des Objekttyps A zweimal als eingabepflichtige Fremdschlüssel, die auf den Empfänger bzw. auf den Sender einer Nachricht verweisen. Beide Fremdschlüssel sind für sich genommen nicht-unikal, so dass ein Objekt in mehreren Tabellenzeilen als Empfänger bzw. als Sender auftreten kann. Die Kombination der beiden Fremdschlüssel ist der Primärschlüssel der Koppel-Tabelle. Dieses Vorgehen wird in der Transformationsregel T19 beschrieben, die in Abbildung 4-70 dargestellt ist.

Beispiel für Transformationsregel T19

Zur Illustration der Transformationsregel T19 betrachten wir eine Stückliste, die den Aufbau eines komplexen Bauteils – beispielsweise eines Autos - aus kleineren Bauteilen beschreibt. Ein Bauteil kann elementar sein, kann sich also nicht mehr in kleinere Bauteile zerlegen lassen. Beispielsweise lässt sich eine Schraube nicht mehr in Einzelteile zerlegen. Ein Bauteil kann sich aber auch in mehrere kleinere Bauteile zerlegen lassen. So besteht ein Motor aus vielen Einzelteilen. Andererseits ist es möglich, dass ein gegebenes Bauteil nicht Bestandteil eines größeren Bauteils ist, wenn es nämlich beispielsweise bereits das komplette Auto darstellt. Es kann aber auch Bestandteil mehrerer größerer Bauteile sein, wenn ein bestimmter Motortyp in mehrere Automodelle eingebaut werden kann. Die erforderliche Transformation in das relationale Datenbank-Modell zeigt die Abbildung 4-71.

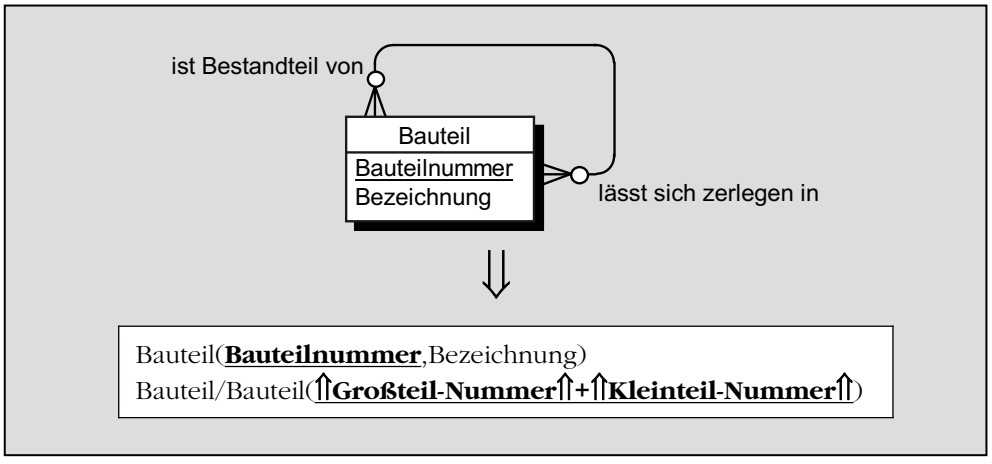


Abb. 4-71: Beispiel für die Transformationsregel T19

Jeder der Fremdschlüssel ist für sich nicht-unikal, so dass mehrere Bauteile jeweils als Großteil bzw. als Kleinteil in der Stückliste auftreten können. Es kann auch sein, dass eine bestimmte Bauteilnummer nicht als Wert des Fremdschlüssels ⇑Großteil-Nummer⇑ auftritt. Dann ist dieses Bauteil nicht weiter zerlegbar. Ist die Bauteilnummer nie Wert des Fremdschlüssels ⇑Kleinteil-Nummer⇑, dann gehört es zu keinem größeren Bauteil.

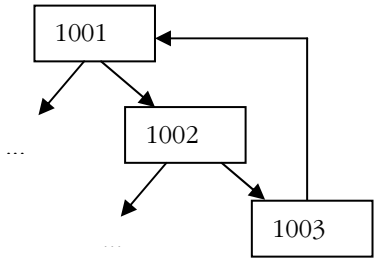
Die Tabellen-Typbeschreibungen lassen allerdings wieder einige Situationen zu, die man im angegebenen Praxisfall eigentlich ausschließen möchte. Dazu gehören beispielsweise:

- *Mehr-Objekte-Zyklen:* Das Bauteil „1001“ hat als eines seiner Bestandteile das Bauteil „1002“, für das wiederum als eines seiner Bestandteile das Bauteil „1003“ angegeben ist. Für das Bauteil „1003“ ist aber angegeben, dass es das Bauteil „1001“ als Bestandteil enthält. Die Koppel-Tabelle „Bauteil/Bauteil“ enthält dann die folgenden Zeilen:

Bauteil/Bauteil

Großteil-Nummer	Kleinteil-Nummer
...	...
1001	1002
1002	1003
1003	1001
...	...

Das entspricht dem folgenden Stücklistenfragment, in dem die Pfeile jeweils vom Großteil auf das Kleinteil zeigen:



- *Ein-Objekt-Zyklen:* Ein Bauteil kann als sein eigener Bestandteil ausgewiesen sein, wenn seine Bauteilnummer in einer Zeile der Koppel-Tabelle in beiden Fremdschlüsseln als Wert angegeben wird.
- *Mehr-Objekte-Ketten:* Ein Bauteil kann als Bestandteil nur ein einziges anderes Bauteil haben. Seine Nummer kommt dann in der Koppel-Tabelle „Bauteil/Bauteil“ nur einmal als Wert des Fremdschlüssels ⇑⇑Großteil-Nummer⇑⇑ vor. Das ist eine unsinnige Konstruktion, denn ein Bauteil ist entweder gar nicht zerlegbar oder es lässt sich in *mindestens zwei* Bestandteile zerlegen.
- *Ein-Objekt-Ketten:* Die Typbeschreibungen lassen die Speicherung eines Bauteils zu, das weder zerlegbar, noch Bestandteil eines größeren Bauteils ist. Das wäre beispielsweise eine Schraube, die in keinem größeren Bauteil Verwendung findet. Ihre Bauteilnummer taucht dann in der Koppel-Tabelle „Bauteil/Bauteil“ an keiner Stelle auf. Die Speicherung solcher Bauteile ist aber im betrachteten Zusammenhang sinnlos.

Die angegebenen Situationen kann das Datenbank-Managementsystem nicht verhindern. Sie müssen durch eine entsprechende Anwendungs-Programmierung unterbunden werden.

4.4.6 Der M:CN-Rekursiv-Beziehungstyp

Beim M:CN-Rekursiv-Beziehungstyp kann ein Objekt keine, eine oder mehrere Nachrichten senden, es muss aber stets mindestens eine Nachricht empfangen. Die Abbildung 4-72 vermittelt einen Eindruck von den möglichen Beziehungsstrukturen, ohne alle möglichen Varianten auszuschöpfen.

Es lassen sich zunächst Ein-Objekt-Zyklen ($a_1 \rightarrow a_1$) und Mehr-Objekte-Zyklen ($a_2 \rightarrow a_3 \rightarrow a_4 \rightarrow a_2$) bilden. Von jedem Objekt eines Zyklus' kann eine Objekte-Kette ausgehen. Diese kann in den Zyklus zurückführen ($a_4 \rightarrow a_5 \rightarrow a_7 \rightarrow a_3$), kann aber auch „offen“ sein ($a_4 \rightarrow a_5 \rightarrow a_6$). Sie kann auch zu einem anderen Zyklus führen ($a_4 \rightarrow a_5 \rightarrow a_7 \rightarrow a_8$). Von jedem Objekt einer Kette kann wiederum eine Kette ausgehen ($a_7 \rightarrow a_9 \rightarrow a_{12} \rightarrow a_{13}$). Eine vollständige, konstruktive Beschreibung der möglichen Sender-Empfänger-Strukturen erfolgt im Abschnitt 5.3.1.2.

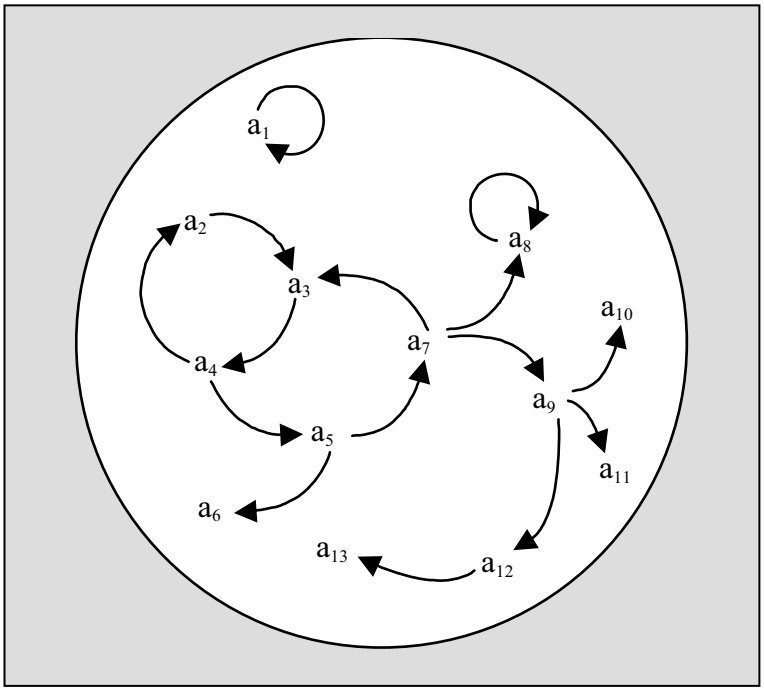


Abb. 4-72: M:CN-Rekursiv-Beziehungstyp

Koppel-Objekttyp

Der M:CN-Rekursiv-Beziehungstyp lässt sich in das relationale Datenbank-Modell nur nach vorheriger Umwandlung in einen Koppel-Objekttyp „Sendung“ überführen. Der Objekttyp A ist mit dem Objekttyp „Sendung“ einerseits durch einen 1:CN- und andererseits durch einen 1:N-Beziehungstyp verbunden. Die Abbildung 4-73 zeigt das Schema der Umwandlung.

Die Kombination der Beziehungstyp-Richtungen „Sendung *wird ausgelöst von* A“ und „Sendung *wird empfangen von* A“ übernimmt die Identifizierung der Objekte im Objekttyp „Sendung“.

Transformationsregel T19

Wie man aus Abbildung 4-73 sieht, muss der M:CN-Rekursiv-Beziehungstyp durch einen 1:CN- und einen 1:N-Beziehungstyp dargestellt werden. Nun haben wir aber im Abschnitt 4.3.6 gesehen, dass sich ein 1:N-Beziehungstyp im relationalen Datenbank-Modell nur als 1:CN-Beziehungstyp repräsentieren lässt. Damit kann der M:CN-Rekursiv-Beziehungstyp nur gemäß der Transformationsregel T19 wie ein CM:CN-Rekursiv-Beziehungstyp dargestellt werden.

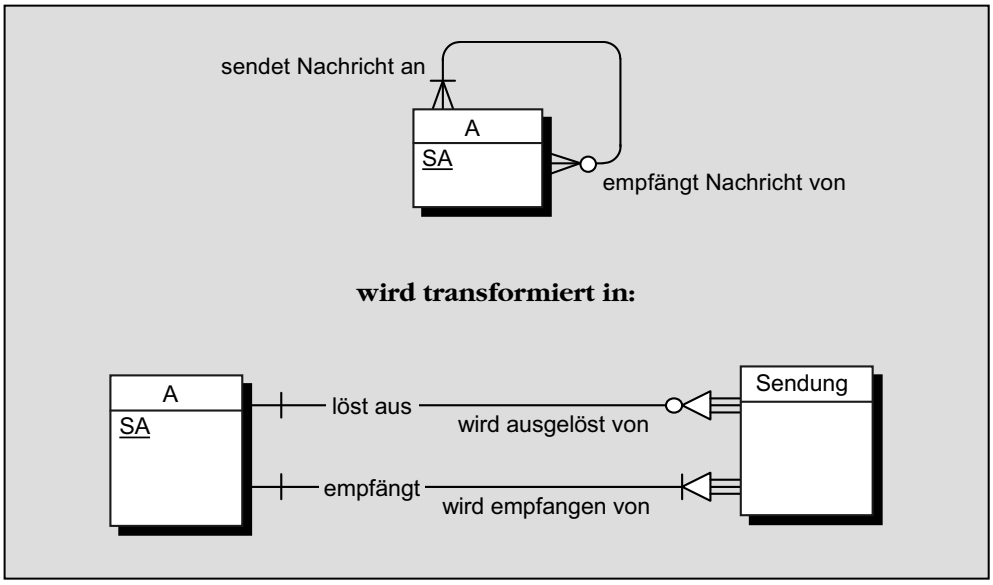


Abb. 4-73: Umwandlung eines M:CN-Rekursiv-Beziehungstyps in einen 1:CN- und einen 1:N-Beziehungstyp

Beispiel für M:CN-Rekursiv-Beziehungstyp

Betrachten wir als Beispiel die Literaturverweise in Lehrbüchern. Jedes Lehrbuch verweist auf mindestens ein Vorgänger-Lehrbuch. Auf ein gerade erst erschienenes Lehrbuch wird noch nicht verwiesen. Handelt es sich um ein interessantes Lehrbuch, so wird es im Laufe der Zeit von vielen Nachfolger-Lehrbüchern zitiert. Die entsprechende Tabellen-Repräsentation gemäß der Transformationsregel T19 zeigt die Abbildung 4-74.

Jeder der beiden Fremdschlüssel ist für sich nicht-unikal, so dass ein gegebenes Lehrbuch mehrmals als Vorgänger bzw. als Nachfolger in Erscheinung treten kann. Ist eine Lehrbuch-ISBN kein einziges Mal Wert des Fremdschlüssels ⇑Vorgänger-ISBN⇑, dann wurde dieses Lehrbuch noch nicht zitiert.

Die Tabellen-Typbeschreibungen lassen wiederum einige Situationen zu, die nicht der Semantik des Beispiels entsprechen:

- *Vernachlässigung der Nicht-optionalität:* Die nicht-optionale Beziehungstyp-Richtung „Lehrbuch *verweist auf* Lehrbuch“ wird als optionale Beziehungstyp-Richtung repräsentiert. Es ist somit möglich, dass eine Lehrbuch-ISBN nie als Wert des Fremdschlüssels ⇑Nachfolger-ISBN⇑ auftaucht. Dann ist die-

ses Lehrbuch nicht als Nachfolger eines anderen Lehrbuchs ausgewiesen, d.h. es verweist – entgegen der Praxisregel – auf kein Vorgänger-Lehrbuch.

- *Zyklen:* Ein Lehrbuch kann auf sich selbst als Vorgänger verweisen (Ein-Objekt-Zyklus) oder eine Objekte-Kette kann sich schließen (Mehr-Objekte-Zyklus). Da Vorgänger-Verweise immer „in die Vergangenheit" zeigen, sind Zyklen eigentlich verboten.

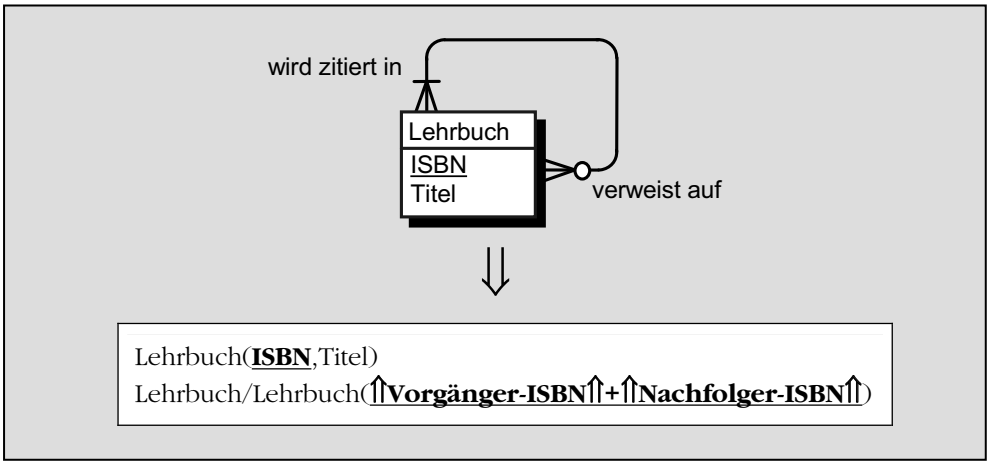


Abb. 4-74: Beispiel für die Transformation eines M:CN-Rekursiv-Beziehungstyps

Die beschriebenen „pathologischen" Situationen lassen sich nicht durch die Tabellen-Typbeschreibungen ausschließen. Ihre Entstehung kann nur durch ein entsprechendes Anwendungsprogramm verhindert werden.

4.4.7 Der M:N-Rekursiv-Beziehungstyp

Der M:N-Rekursiv-Beziehungstyp unterscheidet sich vom zuvor behandelten M:CN-Rekursiv-Beziehungstyp dadurch, dass ein Objekt mindestens eine Nachrichten senden muss. Diese Forderung hat jedoch entscheidende Konsequenzen für die möglichen Beziehungsstrukturen, von denen die Abbildung 4-75 einen Eindruck geben soll.

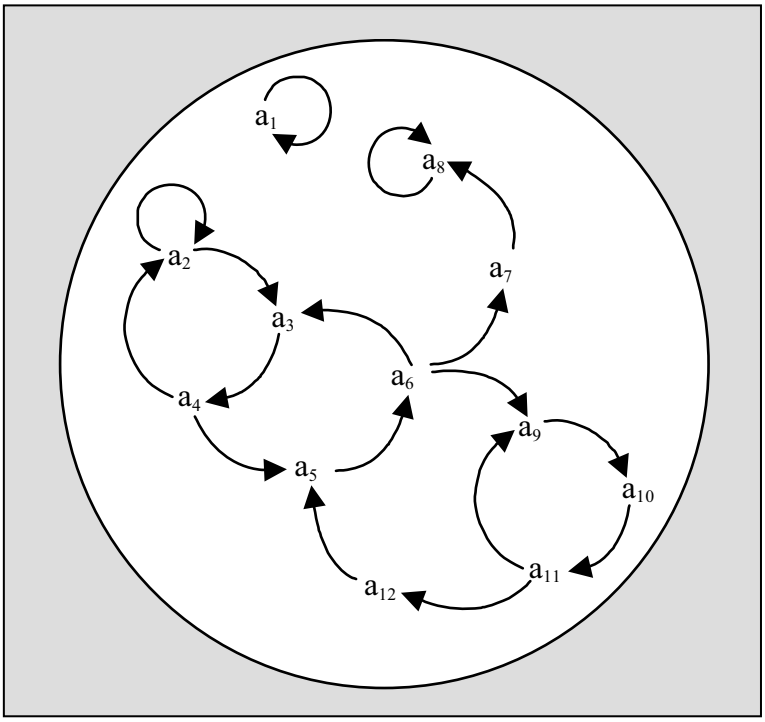


Abb. 4-75: M:N-Rekursiv-Beziehungstyp

Sender-Empfänger-Strukturen

Die Möglichkeiten der Sender-Empfänger-Strukturen sind gegenüber dem M:CN-Rekursiv-Beziehungstyp eingeschränkt. Aus einem Objekte-Zyklus können zwar nach wie vor Objekte-Ketten austreten, sie dürfen aber nicht mehr „offen" sein. Sie müssen also beispielsweise in den Zyklus zurückführen ($a_4 \rightarrow a_5 \rightarrow a_6 \rightarrow a_3$) oder in einen anderen Zyklus einmünden ($a_4 \rightarrow a_5 \rightarrow a_6 \rightarrow a_7 \rightarrow a_8$). Alle möglichen Strukturen werden im Abschnitt 5.3.1.2 konstruktiv beschrieben.

Auch der M:N-Rekursiv-Beziehungstyp lässt sich in das relationale Datenbank-Modell nur nach vorheriger Umwandlung in einen neuen Objekttyp überführen. Der Objekttyp A ist mit dem Koppel-Objekttyp „Sendung" durch zwei 1:N-Beziehungstypen verbunden. Abbildung 4-76 zeigt das Schema der Umwandlung.

Die Kombination der Beziehungstyp-Richtungen „Sendung *wird ausgelöst von* A" und „Sendung *wird empfangen von* A" übernimmt die Identifizierung der Objekte im Objekttyp „Sendung".

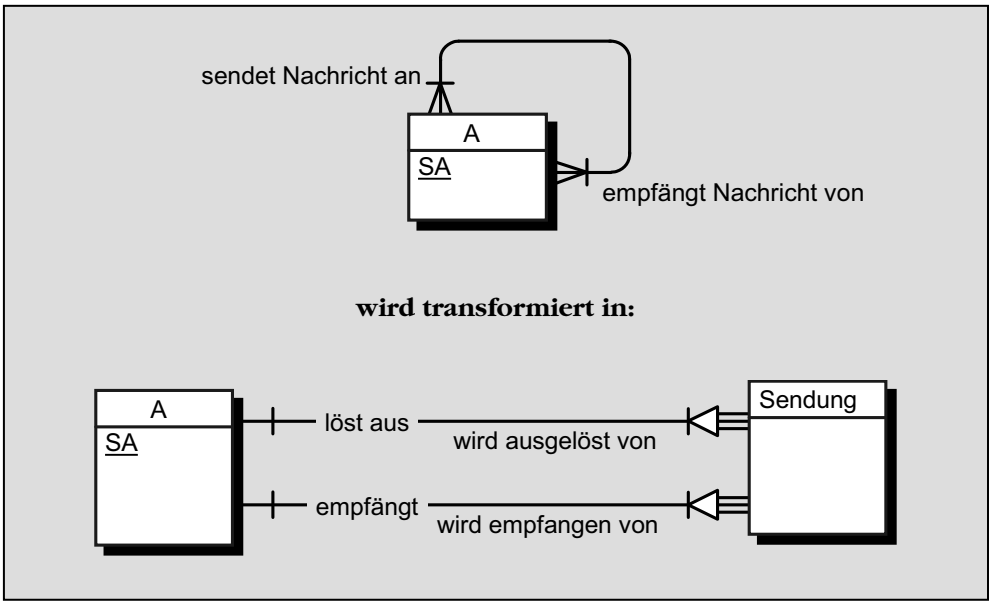


Abb. 4-76: Umwandlung eines M:N-Rekursiv-Beziehungstyps in zwei 1:N-Beziehungstypen

Transformationsregel T19

Da sich ein 1:N-Beziehungstyp im relationalen Datenbank-Modell nur als 1:CN-Beziehungstyp repräsentieren lässt, bleibt nur die Möglichkeit, den M:N-Rekursiv-Beziehungstyp – unter Einbuße an semantischen Informationen - gemäß der Transformationsregel T19 wie einen CM:CN-Rekursiv-Beziehungstyp darzustellen.

Beispiel für M:N-Rekursiv-Beziehungstyp

Als Beispiel betrachten wir eine Kunsthochschule, die Informationen über Pianisten sammelt, die in der Ausbildung tätig sind. Wir nehmen der Einfachheit halber an, dass die Pianisten eindeutig durch ihren Namen unterschieden werden können (jeder Pianist wird schon selber darauf achten, dass er nicht mit anderen Pianisten verwechselt werden kann). Jeder dieser Pianisten hat wenigstens einen anderen Pianisten als Schüler. Andererseits ist jeder Pianist Schüler eines oder mehrerer anderer Pianisten (Autodidakten werden nicht berücksichtigt). Die Abbildung 4-77 zeigt die erforderlichen Tabellen-Typbeschreibungen gemäß der Transformationsregel T19.

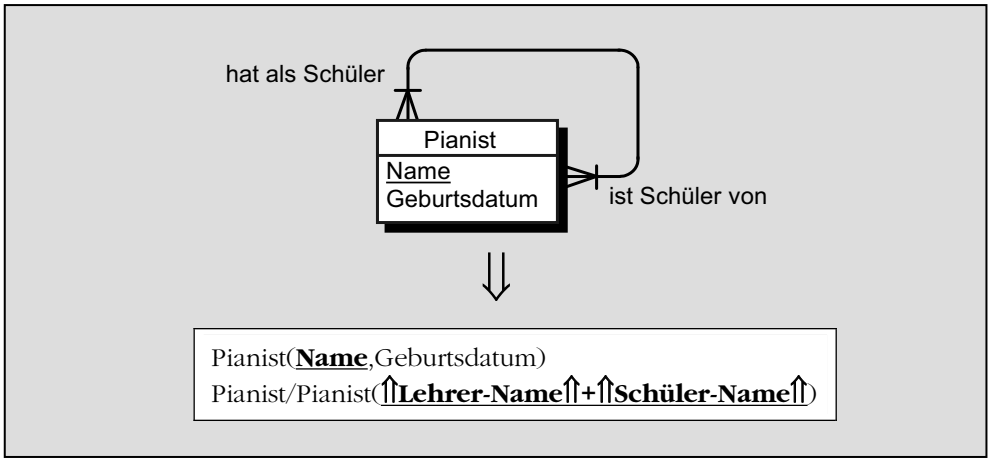


Abb. 4-77: Beispiel für die Transformation eines M:N-Rekursiv-Beziehungstyps

Jeder der beiden Fremdschlüssel ist für sich nicht-unikal, so dass ein Pianist mehrmals als Lehrer bzw. als Schüler in Erscheinung treten kann.

Die Tabellen-Typbeschreibungen ermöglichen Beziehungsstrukturen, die in der Praxis unzulässig sind, so beispielsweise:

- *Vernachlässigung der Nicht-optionalitäten:* Die beiden nicht-optionalen Beziehungstyp-Richtungen „Pianist *hat als Schüler* Pianist“ und „Pianist *ist Schüler von* Pianist“ werden als optionale Beziehungstyp-Richtungen repräsentiert. Es ist somit möglich, dass eine Pianist keine Schüler hat und dass ein Pianist nicht Schüler eines Pianisten ist.
- *Zyklen:* Ein Pianist kann sein eigener Schüler sein (Ein-Objekt-Zyklus) oder eine Schüler-Kette kann zu ihrem Ausgangspunkt zurückkehren (Mehr-Objekte-Zyklus).

Diese unzulässigen Objekte-Strukturen müssen durch ein entsprechendes Anwendungs-Programm verhindert werden.

4.4.8 Transformation der Rekursiv-Beziehungstypen für das Schulbeispiel

In unserem Datenmodell für das Schulbeispiel gibt es zwei rekursive Beziehungstypen, die in Abbildung 4-78 noch einmal wiedergegeben sind.

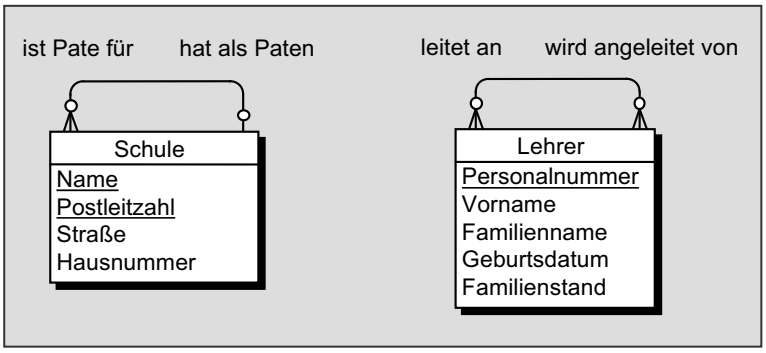


Abb. 4-78: Rekursiv-Beziehungstypen im Schulbeispiel

Betrachten wir nun die Transformation dieser beiden Rekursiv-Beziehungstypen in das relationale Datenbank-Modell.

Bisher hatte die Tabellen-Typbeschreibung für den Objekttyp „Schule“ die Form:

Schule(**Name+Postleitzahl**,⇑Orts-Name+Kreis⇑,
Straße,Hausnummer)

„Schule“ ist mit „Schule“ durch einen C:CN-Rekursiv-Beziehungstyp verknüpft. Da im Deutschen die Bezeichnung „Pate“ nicht eindeutig ist und sich sowohl auf den Betreuer als auch auf den Betreuten beziehen kann, werden für die Sender- bzw. Empfängerrolle der Deutlichkeit halber die Benennungen „Betreute-Schule“ und „Betreuende-Schule“ verwendet. Die Beziehungstyp-Richtungen werden entsprechend in „Schule *wird betreut von* Schule“ und „Schule *betreut* Schule“ umbenannt. In der Abbildung 4-79 wird der C:CN-Rekursiv-Beziehungstyp in der Sender-Empfänger-Form, die wir in diesem Kapitel durchgängig verwendet haben, neu gezeichnet. Die Rolle des „Senders“ (linke Seite des C:CN-Rekursiv-Beziehungstyps) übernimmt in diesem Fall die betreute Schule. Die Rolle des „Empfängers“ (rechte Seite des C:CN-Rekursiv-Beziehungstyps) spielt die betreuende Schule.

Transformationsregel T17 oder T18?

Für die Umwandlung in das relationale Datenbank-Modell stehen die Transformationsregeln T17 (mit vielen Empfängern) und T18 (mit wenigen Empfängern) zur Verfügung. Wir nehmen an, dass es nur wenige „Empfänger“ – in unserem Fall also nur wenige

betreuende Schulen - gibt, dass also die Übernahme einer Patenschaft für eine andere Schule nur in seltenen Fällen erfolgt. Somit ist die Transformationsregel T18 anzuwenden.

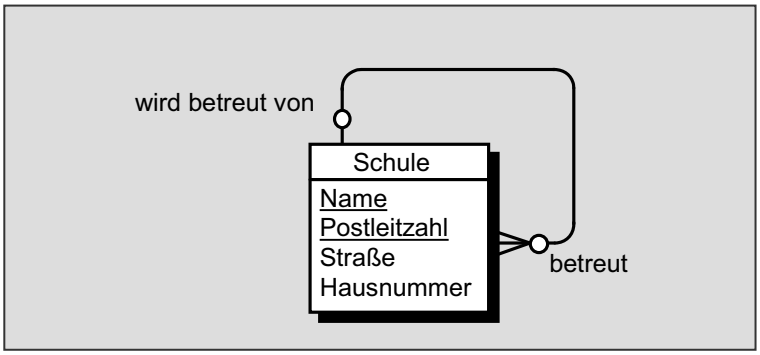


Abb. 4-79: C:CN-Rekursiv-Beziehungstyp für den Objekttyp „Schule“

Es wird also eine Koppel-Tabelle „Schule/Schule“ eingeführt, die den Rekursiv-Beziehungstyp repräsentiert. Dabei bleibt die Typbeschreibung für die Tabelle „Schule“ unverändert:

Schule(**Name+Postleitzahl**,⇑Orts-Name+Kreis⇑,
Straße,Hausnummer)

Schule/Schule
(⇑**Betreuende-Schule-Name+Betreuende-Schule-Postleitzahl**⇑,
⇑Betreute-Schule-Name+Betreute-Schule-Postleitzahl⇑)

Der Fremdschlüssel ⇑**Betreuende-Schule-Name+Betreuende-Schule-Postleitzahl**⇑, der den Primärschlüssel der Koppel-Tabelle bildet, ist unikal. Eine Schule kann also nur einmal als betreuende Schule auftreten. Da der Fremdschlüssel ⇑Betreute-Schule-Name+Betreute-Schule-Postleitzahl⇑ nicht-unikal ist, kann eine betreute Schule mehrfach in Kopplung mit verschiedenen betreuenden Schule stehen.

Wenden wir uns nun dem zweiten Rekursiv-Beziehungstyp in unserem Schulbeispiel zu. Für den Objekttyp „Lehrer“ gilt bisher die folgende Tabellen-Typbeschreibung:

Lehrer(**Personalnummer**,Vorname,Familienname,
Geburtsdatum,Familienstand)

Der sachlogische Zusammenhang, der den „Lehrer“ mit dem „Lehrer“ verbindet, ist durch einen CM:CN-Rekursiv-Beziehungstyp gegeben. Der Deutlichkeit halber zeichnen wir den Rekursiv-Beziehungstyp zunächst in die gewohnte Sender-Empfänger-Form um, wie dies in Abbildung 4-80 geschehen ist. Die Senderrolle übernimmt dabei der „Direktor“, die Empfängerrolle wird weiterhin als „Lehrer“ bezeichnet.

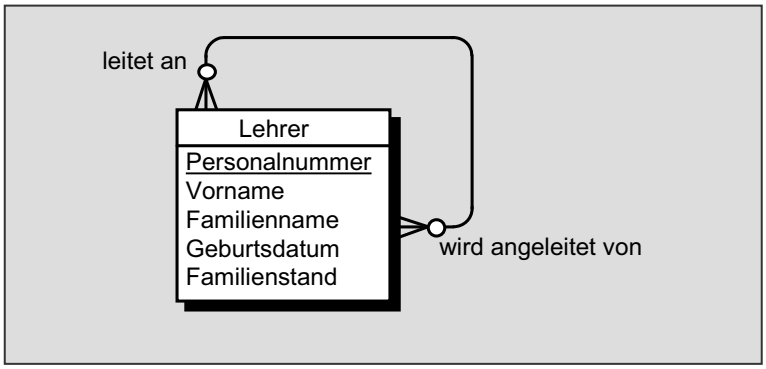


Abb. 4-80: CM:CN-Rekursiv-Beziehungstyp
für den Objekttyp „Lehrer“

Die Transformation des CM:CN-Rekursiv-Beziehungstyps in das relationale Datenbank-Modell muss gemäß der Transformationsregel T19 durch die Einführung einer neuen Koppel-Tabelle „Lehrer/Lehrer“ erfolgen. Die Typbeschreibung für die Tabelle „Lehrer“ bleibt wiederum unverändert:

Lehrer(**Personalnummer**,Vorname,Familienname,
Geburtsdatum,Familienstand)

Lehrer/Lehrer(**⇑Lehrer-Personalnummer⇑+**
⇑Direktor-Personalnummer⇑)

Beide Fremdschlüssel sind für sich allein nicht-unikal, so dass in der Koppel-Tabelle sowohl auf einen „Lehrer“ als auch auf einen „Direktor“ mehrfach verwiesen werden kann.

4.5 Transformation von Kardinalitäts-Beschränkungen

Bei der Einführung der Beziehungstypen im Abschnitt 2.4.1 hatten wir empfohlen, im konzeptionellen Datenmodell einschränkende Bedingungen für die Kardinalität festzuhalten, wenn diese durch die „Geschäftsregeln“ der betrieblichen Realität vorgegeben sind. Nehmen wir zum Beispiel an, dass ein Unternehmen Informationen über die Dienstwagen und über die (Sommer- und Winter-) Räder speichern will, die dem Unternehmen gehören. Jedes Auto ist - inklusive Ersatzrad - mit genau 5 Rädern ausgerüstet. Einige Räder werden als Reserve im Lager aufbewahrt. Dies entspricht einem C:N-Beziehungstyp, für den N nicht beliebige Werte annehmen kann, sondern nur den Wert 5. In der Abbildung 4-81 ist das konzeptionelle Datenmodell angegeben, wobei die Kardinalität N durch eine [Min,Max]-Angabe eingeschränkt wird.

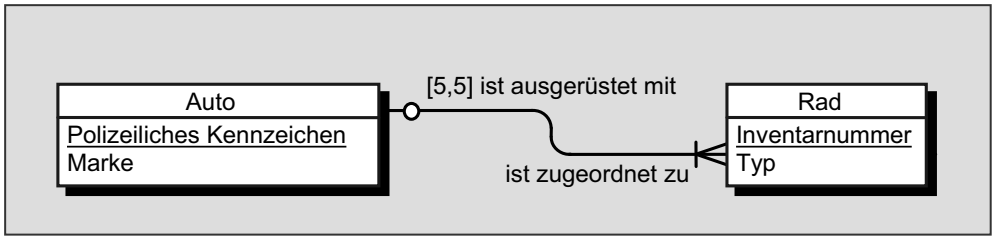


Abb. 4-81: [Min,Max]-Beschränkungen für die Kardinalität N

Wie wird diese Geschäftsregel in den Tabellen-Typbeschreibungen berücksichtigt? Leider gar nicht! Das relationale Datenbank-Modell bietet auf der Ebene des logischen Datenschemas keine Möglichkeit, die Kardinalität einzugrenzen. Das gilt sowohl für den dualen Beziehungstyp als auch für den Rekursiv-Beziehungstyp. Es ist ja nicht einmal möglich, die Kardinalität N, die ja eine ganze Zahl *größer als Null* fordert, überhaupt zu realisieren. Sie kann nur - mit Semantikverlust - unter Einschluss der Optionalität als CN-Beziehungstyp-Richtung repräsentiert werden. Das haben wir in diesem Kapitel schon mehrfach erlebt. Im Kapitel 5 wird diese Beschränkung des relationalen Datenbank-Modells systematisch untersucht.

Für die Repräsentation von Kardinalitäts-Beschränkungen gilt somit die Negativ-Transformationsregel T20, die in der Abbildung 4-82 dargestellt ist.

Transformationsregel T20 (Kardinalitäts-Beschränkung):

Entity-Relationship-Modell		Relationales Datenbank-Modell
Kardinalitäts-Beschränkung [Min,Max] für die N-Seite eines Beziehungstyps	⇒	Eine [Min,Max]-Angabe lässt sich nicht in der Typbeschreibung repräsentieren.

Abb. 4-82: Transformation einer Kardinalitäts-Beschränkung

Auch wenn sich Kardinalitäts-Beschränkungen nicht in das relationale Datenbank-Modell transformieren lassen, sollte dennoch ihre Notierung im konzeptionellen Datenmodell erfolgen, weil sie wichtige Geschäftsregeln darstellen. Diese müssen - wenn schon nicht während der Deklaration der Tabellenstruktur - so doch wenigstens bei der Programmierung der Anwendungs-Software berücksichtigt werden.

Schulbeispiel

In unserem Schulbeispiel hatten wir für zwei Kardinalitäten einschränkende Bedingungen formuliert:

- Eine Klasse muss aus mindestens 15 Schüler und aus höchstens 30 Schülern bestehen.
- Ein Unterrichtsraum ist höchstens 3 Klassen als Klassenraum zuzuordnen.

Den entsprechenden Ausschnitt aus dem konzeptionellen Datenmodell für das Schulbeispiel gibt die Abbildung 4-83 wieder.

Gemäß der Transformationsregel T20 führen die Kardinalitäts-Beschränkungen zu keinen Veränderungen in den Tabellen-Typbeschreibungen.

Warum lässt nun eigentlich das relationale Datenbank-Modell keine Kardinalitäts-Beschränkungen zu? Ganz einfach: weil es kompliziert ist, sie im Datenbank-Betrieb aufrecht zu erhalten. Bei der Deklaration von Tabellen-Typbeschreibungen lassen sich nur solche Bedingungen angeben, die ohne Transaktionen – also nur durch elementare Datenbank-Aktionen – durchgesetzt werden können. Das ist im Falle von Kardinalitäts-Beschränkungen

aber nicht möglich. Untersuchen wir beispielsweise den Beziehungstyp zwischen Klasse und Schüler:

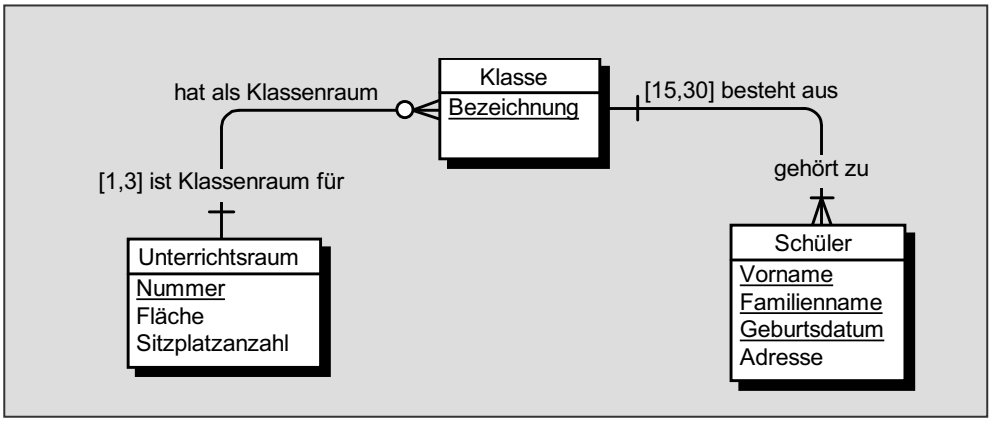


Abb. 4-83: Kardinalitäts-Beschränkungen im Datenmodell des Schulbeispiels

- *Speichern einer neuen Klasse:* Wird eine neue Klasse gespeichert, müssen ihr sofort mindestens 15 Schüler zugeordnet werden. Diese Schüler können aber nicht schon vorher gespeichert worden sein, weil ja jeder Schüler unbedingt zu einer Klasse gehören muss. Es ist also eine Transaktion erforderlich, die folgende Aktivitäten enthält:

  a) Speichern des neuen Datensatzes für die Klasse mit dem Primärschlüsselwert k.

  b) Speichern von mindestens 15 neuen Datensätzen für Schüler, wobei in jedem dieser Datensätze als Wert des Fremdschlüssels, der auf die Klasse verweist, der Wert k einzutragen ist.

- *Speichern eines weiteren Schülers:* Wenn ein neuer Schüler der Klasse k zugeordnet werden soll, so ist das problemlos möglich, solange die Anzahl der Schüler nicht die Maximal-Kardinalität 30 überschreitet. In diesem Fall wird der Schüler-Datensatz gespeichert, wobei als Wert des Klassen-Fremdschlüssels k eingetragen wird. Würde der Schüler allerdings zum 31. Schüler dieser Klasse, dann muss in einer Transaktion eine neue Klasse eingerichtet werden:

a) Speichern eines Datensatzes für eine neue Klasse mit dem Primärschlüsselwert k'.

b) Ändern des Klassen-Fremdschlüsselwertes bei 14 Schüler-Datensätzen auf den Wert k'.

c) Speichern des Datensatzes für den neuen Schüler mit k' als Wert des Klassen-Fremdschlüssels.

- *Löschen eines Schülers:* Wenn der Datensatz eines Schülers gelöscht werden soll (weil er beispielsweise in eine andere Schule wechselt), so ist das problemlos möglich, solange die Anzahl der Schüler in seiner Klasse nicht die Minimal-Kardinalität 15 unterschreitet. Andernfalls müssen die Schüler seiner Klasse auf andere Klassen aufgeteilt werden. Dabei ist es fraglich, ob das überhaupt gelingt, ohne die geforderten Kardinalitäts-Beschränkungen zu verletzen.

4.6 Konzeptionelles Datenmodell *versus* logisches Datenschema

In den vorangegangenen Abschnitten dieses Kapitels haben wir mit den 20 Transformationsregeln das Rüstzeug zusammengetragen, um aus dem *konzeptionellen Datenmodell* die Struktur der Datenbank, also das *logische Datenschema*, ableiten zu können. Durch Anwendung dieser Transformationsregeln wird das endgültige Datenmodell für das Schulbeispiel in die Tabellen-Typbeschreibungen transformiert, die die Datenbank-Struktur für das relationale Datenbank-Managementsystem festlegen.

Damit die Transformation klar nachvollziehbar wird, geben wir in der Abbildung 4-84 noch einmal das konzeptionelle Datenmodell für das Schulbeispiel an und stellen diesem Datenmodell in der Abbildung 4-85 die Tabellen-Typbeschreibungen, also das logische Datenschema, gegenüber.

Nun stellt sich natürlich die Frage, warum man die Struktur der aufzubauenden Datenbank, also ihr *logisches Datenschema*, nicht direkt in Form der Tabellen-Typbeschreibungen der Abbildung 4-85 entwirft. Warum sind wir den Umweg über das *konzeptionelle Datenmodell* gegangen, aus dem wir erst die Tabellen-Struktur gemäß den 20 Transformationsregeln abgeleitet haben? Sind beide nicht Repräsentationen derselben Informationen über den Ausschnitt der betrieblichen Realität? Betrachten wir die Unterschiede beider Darstellungsformen etwas genauer!

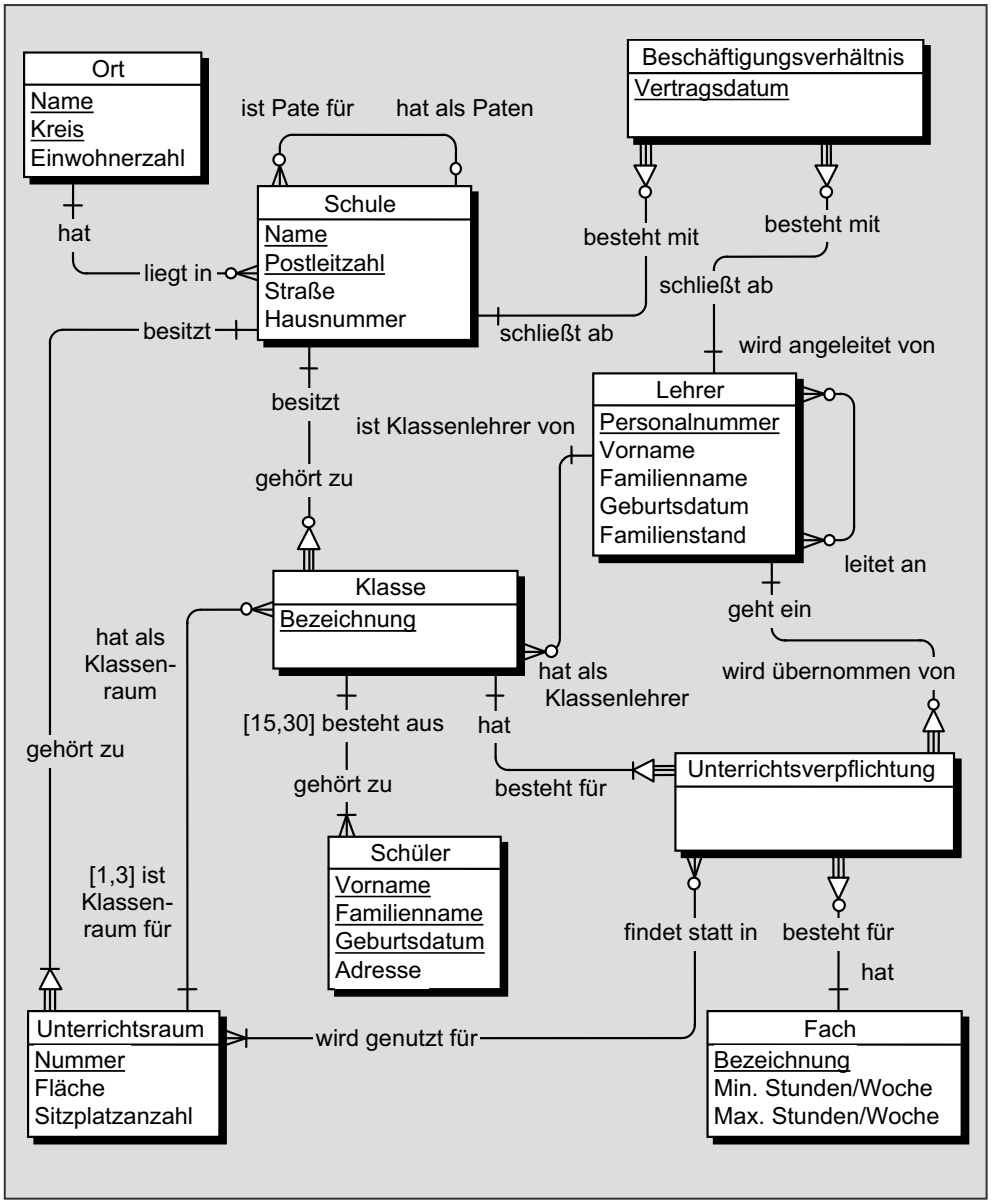


Abb. 4-84: Konzeptionelles Datenmodell für das Schulbeispiel

Ort(**Name+Kreis**,Einwohnerzahl)

Schule(**Name+Postleitzahl**,⇑Orts-Name+Kreis⇑,Straße,Hausnummer)

Schule/Schule(⇑**Betreuende-Schule-Name+Betreuende-Schule-Postleitzahl**⇑,
⇑Betreute-Schule-Name+Betreute-Schule-Postleitzahl⇑)

Beschäftigungsverhältnis(⇑**Personalnummer**⇑+⇑**Schul-Name+Postleitzahl**⇑+
Vertragsdatum)

Lehrer(**Personalnummer**,Vorname,Familienname,Geburtsdatum,Familienstand)

Lehrer/Lehrer(⇑**Lehrer-Personalnummer**⇑+⇑**Direktor-Personalnummer**⇑)

Klasse(⇑**Schul-Name-1+Postleitzahl-1**⇑+**Bezeichnung**,
⇑Personalnummer⇑,
⇑Schul-Name-2+Postleitzahl-2+Raum-Nummer⇑)

Schüler(**Vorname+Familienname+Geburtsdatum**,
⇑Schul-Name+Postleitzahl+Klassen-Bezeichnung⇑,
Adresse)

Fach(**Bezeichnung**,Min. Stunden/Woche,Max. Stunden/Woche)

Unterrichtsraum(⇑**Schul-Name+Postleitzahl**⇑+**Nummer**,
Fläche,Sitzplatzanzahl)

Unterrichtsraum/Unterrichtsverpflichtung
(⇑**Schul-Name-1+Postleitzahl-1+Raum-Nummer**⇑+
⇑**Schul-Name-2+Postleitzahl-2+Klassen-Bezeichnung+**
Fach-Bezeichnung+
Personalnummer⇑)

Unterrichtsverpflichtung
(⇑**Schul-Name+Postleitzahl+Klassen-Bezeichnung**⇑+
⇑**Fach-Bezeichnung**⇑+
⇑**Personalnummer**⇑)

Abb. 4-85: Tabellen-Typbeschreibungen für das Schulbeispiel

Vergleich des Informationsgehalts

Das konzeptionelle Datenmodell in der Abbildung 4-84 hat einen *größeren Informationsgehalt* als die Tabellen-Typbeschreibungen der Abbildung 4-85. Das Datenmodell enthält zusätzlich:

- Geschäftsregeln, die in Form von *1:N-Beziehungstypen* formuliert sind:

  a) Es ist nicht erlaubt, dass eine Schule *keinen* Unterrichtsraum besitzt.

  b) Es ist unzulässig, dass eine Klasse *keine* Unterrichtsverpflichtung hat.

  c) Es darf nicht vorkommen, dass eine Klasse *keinen* Schüler hat.

  Diese Forderungen können auf Grund der Grenzen des relationalen Datenbank-Modells nicht in den Tabellen-Typbeschreibungen ausgedrückt werden, sondern sind nur durch die Anwendungs-Software zu realisieren.

- Geschäftsregeln, die als *Kardinalitäts-Beschränkungen* formuliert sind:

  a) Eine Klasse besteht aus *mindestens 15* Schülern und darf *höchstens 30* Schüler haben.

  b) Ein Unterrichtsraum darf *höchstens 3* Klassen als Klassenraum zugeteilt werden.

  Derartige Kardinalitäts-Beschränkungen lassen sich in den Tabellen-Typbeschreibungen nicht „festschreiben". Auch hier hilft wieder nur die Anwendungs-Software.

Vergleicht man die Abbildungen 4-84 und 4-85, so erkennt man außerdem, dass das Datenmodell – trotz seines größeren Informationsgehalts – wesentlich *übersichtlicher* ist als das Ensemble der Tabellen-Typbeschreibungen. Der Grund dafür liegt insbesondere darin, dass die Typbeschreibungen ein hohes Maß an redundanten Informationen enthalten, die im Datenmodell vermieden werden. Wenn wir wiederum die Analogie zum „Object Linking and Embedding"-Prinzip heranziehen, dann können wir folgende Unterscheidung treffen:

Linking and Embedding

- *Linking:* Im konzeptionellen Datenmodell wird das Verweis-Prinzip an zwei Stellen realisiert:

  a) „Schwache" Objekttypen (vgl. Abschnitt 2.4.4) verweisen auf den Objekttyp, dessen Schlüssel sie für ihre Identifizierung „nachnutzen". Wir haben diesen Verweis als

„identifizierende Beziehungstyp-Richtung“ bezeichnet. Der Schlüssel wird also nicht kopiert, sondern auf ihn wird durch den „Abhängigkeitspfeil“ (⇛ bzw. ⇾) verwiesen.

b) Beziehungstypen werden durch Verweislinien repräsentiert. Dadurch werden - optisch gut sichtbar – *beide* Beziehungstyp-Richtungen zum Ausdruck gebracht.

- *Embedding:* In den Tabellen-Typbeschreibungen wird dagegen das Kopier-Prinzip eingesetzt:

  a) In die Tabellen-Typbeschreibung eines „schwachen“ Objekttyps wird der Primärschlüssel der referenzierten Tabelle „hineinkopiert“. Insbesondere bei kaskadierenden „schwachen“ Objekttypen[26] entstehen dadurch recht unübersichtliche Primärschlüssel. So nimmt beispielsweise der Primärschlüssel der Tabelle „Unterrichtsverpflichtung“ unter anderem auch den Primärschlüssel der Tabelle „Klasse“ in sich auf, der wiederum den Primärschlüssel der Tabelle „Schule“ enthält.

  b) Beziehungstypen werden durch „gedoppelte“ Primärschlüssel repräsentiert, die entweder in die Tabelle eines der assoziierten Objekttypen oder aber in eine neu geschaffene Koppel-Tabelle „kopiert“ werden. Dadurch wird jeweils nur eine Beziehungstyp-Richtung, nämlich die Richtung „Tabelle des Fremdschlüssels → Tabelle des Primärschlüssels“ direkt repräsentiert. Die umgekehrte Beziehungstyp-Richtung ist zwar implizit in den Tabellen-Typbeschreibungen enthalten, ist optisch aber nur „versteckt“ dargestellt.

Probleme bei Modifikationen der Datenstruktur

Die redundanzbehaftete Informationsdarstellung in den Tabellen-Typbeschreibungen führt neben einem Verlust an Übersichtlichkeit vor allem zu gravierenden Problemen bei *Modifikationen der Datenstruktur*, die im Laufe der Zeit zwangsläufig erforderlich werden. So führt beispielsweise eine Änderung des Primär-

[26] Kaskadierende „schwache“ Objekttypen liegen dann vor, wenn die Identifizierung eines „schwachen“ Objekttyps A durch eine Beziehungstyp-Richtung zu einem Objekttyp B erfolgt, wobei der Objekttyp B ebenfalls ein „schwacher“ Objekttyp ist, der über eine Beziehungstyp-Richtung zu einem dritten Objekttyp C identifiziert wird.

schlüssels einer Tabelle oft zu schwer überschaubaren Veränderungen in den Typbeschreibungen der anderen Tabellen.

Schulbeispiel

Betrachten wir ein ganz elementares Beispiel! Nehmen wir an, dass in der Landesschulbehörde entschieden wird, die Schulen des Landes künftig durch eine unikale Schulnummer voneinander zu unterscheiden. Das führt zu den folgenden Konsequenzen für das konzeptionelle Datenmodell und für die Tabellen-Typbeschreibungen:

- *Konzeptionelles Datenmodell:* Der Objekttyp „Schule“ wird um die Eigenschaft „Schulnummer“ erweitert. Wie in der Abbildung 4-86 dargestellt ist, übernimmt nun diese Eigenschaft „Schulnummer“ als organisatorische Eigenschaft die Identifizierung anstelle der bisherigen Eigenschafte-Kombination „Name+Postleitzahl“. Weitere Änderungen des Datenmodells sind nicht erforderlich.

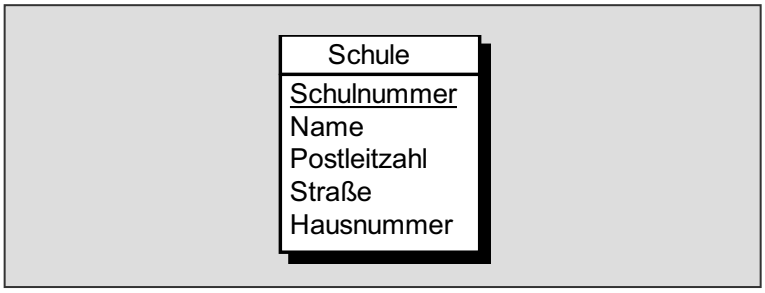


Abb. 4-86: Modifizierter Objekttyp „Schule“

- *Tabellen-Typbeschreibungen:* Von der Veränderung sind fast alle Tabellen betroffen:

  a) Die Tabelle „Schule“ erhält die neue Eigenschaft „Schulnummer“ als Primärschlüssel.

  b) Wegen des veränderten Primärschlüssels der Tabelle „Schule“ ändern sich auch die beiden Fremdschlüssel in der Koppel-Tabelle „Schule/Schule“.

  c) In der Tabelle „Beschäftigungsverhältnis“ muss der Fremdschlüssel, der auf die „Schule“ verweist, angepasst werden.

  d) In der Tabelle „Unterrichtsraum“, die einen „schwachen“ Objekttyp darstellt, muss der Primärschlüssel-Anteil, der auf die „Schule“ verweist, verändert werden.

e) In der Tabelle „Klasse“ muss der Primärschlüssel-Anteil, der die „Schule“ identifiziert, angepasst werden. Außerdem ist der Fremdschlüssel, der auf den „Unterrichtsraum“ verweist, zu modifizieren, weil der „Unterrichtsraum“ über die „Schule“ identifiziert wird.

f) Da in der Tabelle „Schüler“ der Fremdschlüssel auf die „Klasse“ verweist, die über die „Schule“ identifiziert wird, müssen auch hier Änderungen vorgenommen werden.

g) Auch in der Tabelle „Unterrichtsverpflichtung“ sind Änderungen erforderlich, weil sie für einen „schwachen“ Objekttyp angelegt wurde, der über die „Klasse“ – und somit über die „Schule“ - identifiziert wird.

h) In der Koppel-Tabelle „Unterrichtsraum/Unterrichtsverpflichtung“ müssen die Fremdschlüssel, die auf den Unterrichtsraum bzw. auf die Unterrichtsverpflichtung verweisen, modifiziert werden.

Es müssen also fast alle Tabellen-Typbeschreibungen wegen dieser eigentlich „harmlosen“ Änderung modifiziert werden. In der Abbildung 4-87 sind die modifizierten Typbeschreibungen durch ein Sternchen gekennzeichnet.

Es sprechen also mehrere Gründe dafür, die Datenbank-Struktur - also das *logische Datenschema* - nicht auf direktem Wege, sondern auf dem „Umweg“ über das *konzeptionelle Datenmodell* zu entwerfen:

1. Das Datenmodell bietet eine *graphische Repräsentation* der Beziehungstypen und ist damit viel übersichtlicher als die rein textlichen Tabellen-Typbeschreibungen. Die bekannte Aussage, dass „ein Bild mehr sagt als 1000 Wörter“, beweist auch beim Datenbankentwurf ihre Gültigkeit.

2. In der Phase der Anforderungsanalyse ist eine rege *Kommunikation* zwischen Fachexperten und Systementwicklern erforderlich. Das Datenmodell eignet sich durch seine leicht erlernbaren und übersichtlichen Strukturen viel besser für den partnerschaftlichen Gedankenaustausch als die unanschauliche Tabellenstruktur mit ihren verwirrenden Verweisen in Gestalt von Fremdschlüssel-Primärschlüssel-Kongruenzen.

Ort(**Name+Kreis**,Einwohnerzahl)

* Schule(**Schulnummer**,Name,Postleitzahl,⇑Orts-Name+Kreis⇑,
Straße,Hausnummer)

* Schule/Schule(⇑**Betreuende-Schule-Schulnummer**⇑,
⇑Betreute-Schule-Schulnummer⇑)

* Beschäftigungsverhältnis(⇑**Personalnummer**⇑+⇑**Schulnummer**⇑+
Vertragsdatum)

Lehrer(**Personalnummer**,Vorname,Familienname,Geburtsdatum,Familienstand)

Lehrer/Lehrer(⇑**Lehrer-Personalnummer**⇑+⇑**Direktor-Personalnummer**⇑)

* Klasse(⇑**Schulnummer-1**⇑+**Bezeichnung**,
⇑Personalnummer⇑,
⇑Schulnummer-2+Raum-Nummer⇑)

* Schüler(**Vorname+Familienname+Geburtsdatum**,
⇑Schulnummer+Klassen-Bezeichnung⇑,
Adresse)

Fach(**Bezeichnung**,Min. Stunden/Woche,Max. Stunden/Woche)

* Unterrichtsraum(⇑**Schulnummer**⇑+**Raum-Nummer**,
Fläche,Sitzplatzanzahl)

* Unterrichtsraum/Unterrichtsverpflichtung
(⇑**Schulnummer-1+Raum-Nummer**⇑+
⇑**Schulnummer-2+Klassen-Bezeichnung+**
Fach-Bezeichnung+
Personalnummer⇑)

* Unterrichtsverpflichtung
(⇑**Schulnummer+Klassen-Bezeichnung**⇑+
⇑**Fach-Bezeichnung**⇑+
⇑**Personalnummer**⇑)

Abb. 4-87: Tabellen-Typbeschreibungen mit verändertem Primärschlüssel der Tabelle „Schule“

3. Wie wir oben gesehen haben, können im Datenmodell *zusätzliche Angaben* über den betrieblichen Gegenstandsbereich hinterlegt werden, die sich nicht in Tabellen-Typbeschreibungen ausdrücken lassen.
4. *Änderungen* an der Datenstruktur sind auf Grund der konsequenten Anwendung des Linking-Prinzips viel einfacher auszuführen als in den Typbeschreibungen, die entsprechend dem Embedding-Prinzip viele Redundanzen enthalten.

Die Umsetzung des konzeptionellen Datenmodells in die Tabellen-Typbeschreibungen ist nach einem leicht zu formulierenden Algorithmus möglich, den wir in Form von 20 Transformationsregeln angegeben haben. Sie lässt sich somit einfach automatisieren. Von der automatisierten Umwandlung des Datenmodells in die relationale Tabellenstruktur handelt der folgende Abschnitt.

4.7 Automatisierte Generierung des logischen Datenschemas

Das von uns für die Erstellung des konzeptionellen Datenmodells verwendete CASE-Tool *„PowerDesigner“* führt den Prozess der Umwandlung des Datenmodells in eine relationale Tabellenstruktur - also in das logische Datenschema – für das Datenbank-Managementsystem *„Access“* in 4 Generierungs-Schritten durch:

4 Generierungs-Schritte

1. Das entwickelte *konzeptionelle Datenmodell* („Conceptual Data Model“ – CDM) wird unter Beachtung der Möglichkeiten, die *„Access“* für die Tabellen-Deklarationen bietet, in ein *physisches Datenmodell* („Physical Data Model“ - PDM) umgewandelt.
2. Das physische Datenmodell kann *modifiziert* werden. Dadurch lassen sich die automatisiert getroffenen Entscheidungen - insbesondere die nach formalen Regeln gebildeten Benennungen für Koppel-Tabellen und Fremdschlüssel-Attribute - „von Hand“ verändern.
3. Aus dem physischen Datenmodell wird ein *Script* mit der Bezeichnung „CREBAS.DAT“ abgeleitet, das eine textuelle Beschreibung der zu erzeugenden Datenstruktur enthält.
4. Das Meta-Datenbank-Anwendungssystem „ACCESS2K.MDB“ wird gestartet. Dieses Datenbank-Anwendungssystem dient lediglich dazu, ein Visual-Basic-Programm bereitzustellen, das aus der Script-Datei „CREBAS.DAT“ die Ziel-Datenbank *generiert*. Diese Datenbank hat nun eine Tabellen-Struktur,

also ein logisches Datenschema, das dem konzeptionellen Datenmodell entspricht.

Schulbeispiel

Dieser Prozess soll im Folgenden für unser Schulbeispiel nachvollzogen werden. Wir wollen dabei zwei Konstrukte des konzeptionellen Datenmodells besonders im Auge behalten:

a) Wir betrachten den CN:C-Rekursiv-Beziehungstyp „Schule *ist Pate für* Schule - Schule *hat als Paten* Schule“

b) Wir verfolgen, was mit dem M:CN-Beziehungstyp zwischen dem „Unterrichtsraum“ und der „Unterrichtsverpflichtung“ geschieht.

Das Datenmodell für unser Schulbeispiel wurde zur besseren Übersicht in der Abbildung 4-88 auf die beiden genannten Konstrukte - mit ihrem notwendigen Kontext - reduziert.

1. Generierungs-Schritt

Im 1. Generierungs-Schritt wird aus dem konzeptionellen Datenmodell das *physische Datenmodell* abgeleitet. Dies erfolgt nach den Transformationsregeln des „PowerDesigner“. Diese Regeln sind weniger differenziert als die Transformationsregeln T01 bis T20, wie wir im Folgenden sehen werden.

Die Abbildung 4-89 zeigt das generierte physische Datenmodell für das gesamte Schulbeispiel der Abbildung 2-31. Durch Pfeile werden die sachlogischen Zusammenhänge zwischen den Tabellen dargestellt, wobei die Pfeile vom Fremdschlüssel zum Primärschlüssel weisen. An den Pfeil-Linien ist jeweils angegeben, aus welchen Attributen – bzw. Attribute-Kombinationen - die Primärschlüssel bzw. die korrespondierenden Fremdschlüssel gebildet sind.

Im Generierungsprozess, bei dem die Primärschlüssel „gedoppelt“ und zu Fremdschlüsseln „umgewandelt“ wurden, sind die Fremdschlüssel-Attribute immer dann umbenannt worden, wenn es sonst zu Namens-Dopplungen in einer Tabelle gekommen wäre. Die automatisch generierten Benennungen für die Fremdschlüssel-Attribute sind natürlich nicht „sprechend“. Deshalb wurden sie im 2. Generierungs-Schritt „von Hand“ in die Benennungen umgewandelt, die wir in unseren Tabellen-Typbeschreibungen verwendet haben. Damit ist ein besserer Vergleich der „intellektuell“ erzeugten mit den „automatisch“ generierten Tabellen-Strukturen möglich.

2. Generierungs-Schritt

Wie werden unsere beiden zu verfolgenden Konstrukte nun im physischen Datenmodell dargestellt?

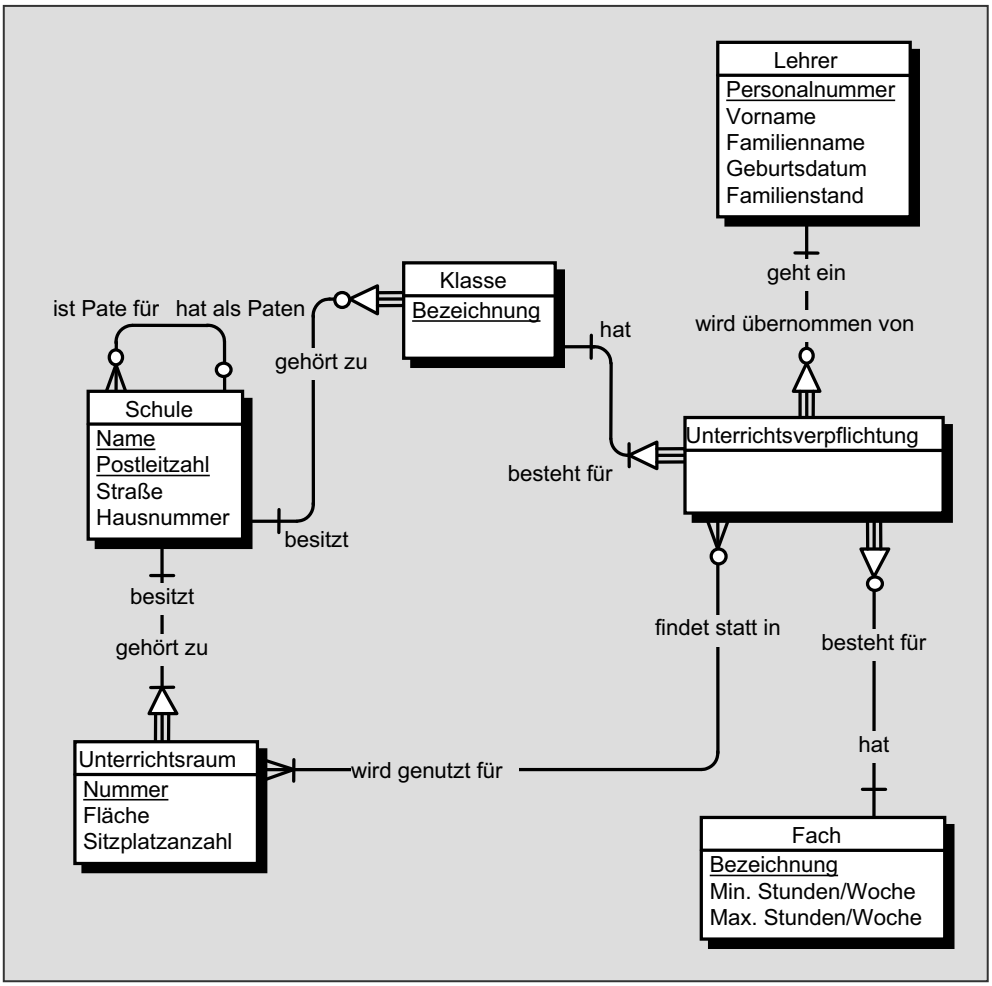


Abb. 4-88: Ausschnitt aus dem Datenmodell für das Schulbeispiel

a) Wir hatten den CN:C-Rekursiv-Beziehungstyp „Schule *ist Pate für* Schule - Schule *hat als Paten* Schule“ gemäß Transformationsregel T18 durch eine Koppel-Tabelle repräsentiert. Dies erfolgte unter der Annahme, dass Patenschaftsverhältnisse unter den Schulen selten sind. In diesem Fall führt die Koppel-Tabelle zu einer Speicherplatz-Einsparung gegenüber der Verwendung eines Fremdschlüssels in der Objekttyp-Tabelle „Schule“.

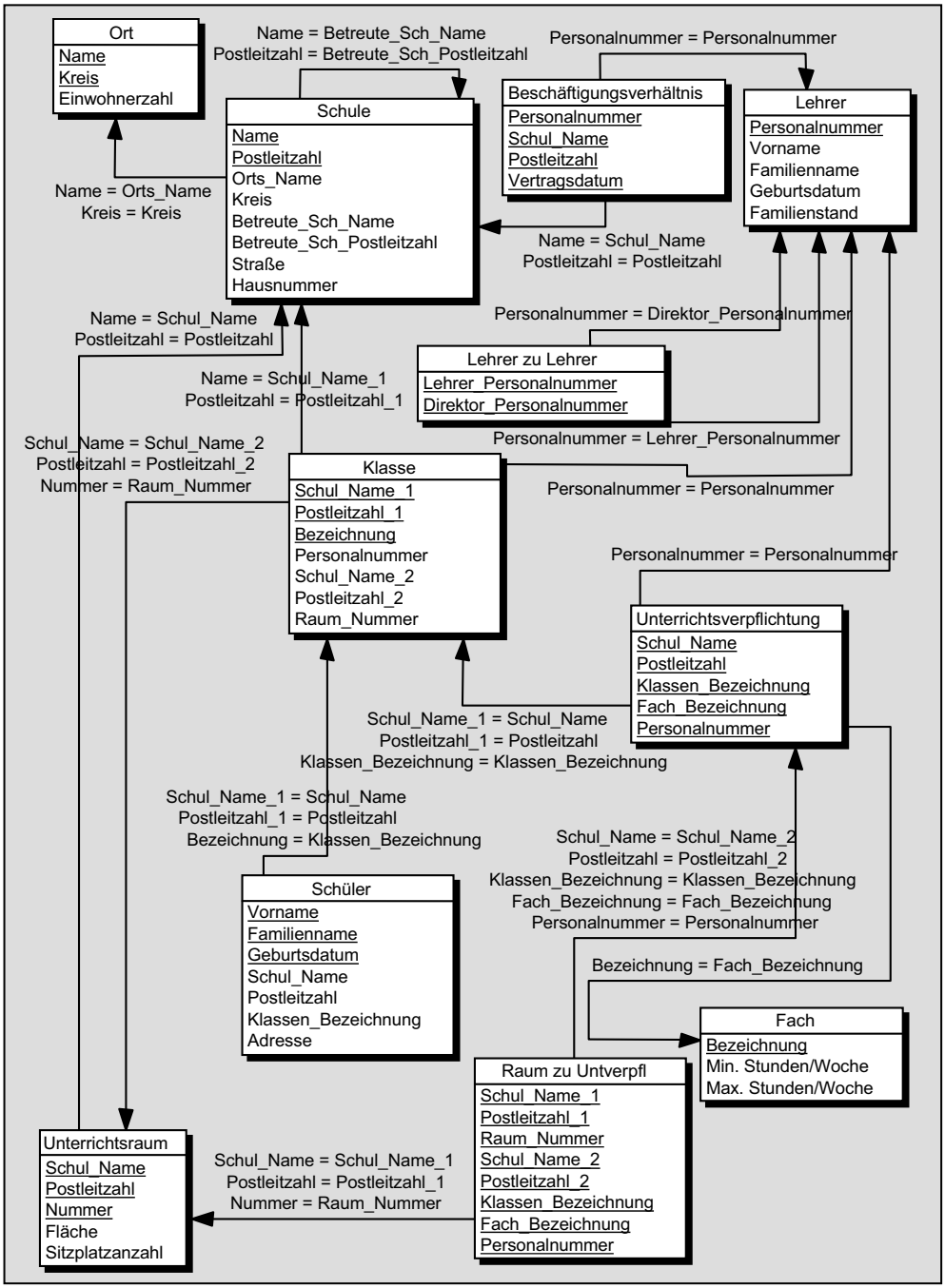


Abb. 4-89: Physisches Datenmodell für das Schulbeispiel

Dem „PowerDesigner“ stehen diese zusätzlichen Informationen nicht zur Verfügung. Er kennt folglich keine differenzierten Regeln für die Repräsentation eines CN:C-Beziehungstyps und stellt ihn durch einen *Fremdschlüssel* in der Objekttyp-Tabelle „Schule“ dar, der die folgende Form hat:

⇑Betreute_Sch_Name+Betreute_Sch_Postleitzahl⇑

b) Der M:CN-Beziehungstyp, der zwischen dem „Unterrichtsraum“ und der „Unterrichtsverpflichtung“ besteht, müsste nach unseren Überlegungen wie ein CM:CN-Beziehungstyp gemäß der Transformationsregel T19 durch eine Koppel-Tabelle repräsentiert werden. Der „PowerDesigner“ generiert tatsächlich eine *Koppel-Tabelle* mit der Bezeichnung „Raum zu Untverpfl“ (die Benennung wurde aus Platzgründen „verstümmelt“). Die beiden eingabepflichtigen nicht-unikalen Fremdschlüssel, die aus der Koppel-Tabelle auf die Tabellen „Unterrichtsraum“ und „Unterrichtsverpflichtung“ verweisen und die gemeinsam den Primärschlüssel der Koppel-Tabelle bilden, lauten:

⇑Schul_Name_1+Postleitzahl_1+Raum_Nummer⇑

⇑Schul_Name_2+Postleitzahl_2+Klassen_Bezeichnung+
Fach_Bezeichnung+Personalnummer⇑

3. Generierungs-Schritt

Im 3. Generierungs-Schritt wird aus dem physischen Datenmodell das *Script* „CREBAS.DAT“ abgeleitet. Durch dieses Script wird die letztlich in der Datenbank zu erzeugende Datenstruktur in Textform beschrieben. Unsere betrachteten Konstrukte führen zu den folgenden Script-Abschnitten:

a) Für den CN:C-Rekursiv-Beziehungstyp „Schule *ist Pate für* Schule - Schule *hat als Paten* Schule“ enthält das Script zunächst die Typbeschreibung der Tabelle „SCHULE“:

```
CreateTble C=SCHULE N="Schule"
(
   C=NAME T=Text(50) P=Yes M=Yes N="Name",
   C=POSTLEITZAHL T=Integer P=Yes M=Yes
                  N="Postleitzahl",
   C=ORTS_NAME T=Text(50) P=No M=Yes N="Orts_Name",
   C=KREIS T=Text(50) P=No M=Yes N="Kreis",
   C=BETREUTE_SCH_NAME  T=Text(50) P=No M=No
                       N="Betreute_Sch_Name",
   C=BETREUTE_SCH_POSTLEITZAHL T=Integer P=No M=No
                  N="Betreute_Sch_Postleitzahl",
   C=STRASSE T=Text(50) P=No M=No N="Straße",
   C=HAUSNUMMER T=Integer P=No M=No N="Hausnummer"
);
```

Die Tabellen-Typbeschreibung enthält Angaben über die Namen (N) und die in der Datenbank zu verwendenden Codes (C) der Tabelle und ihrer Attribute. Für die Attribute werden weiterhin angegeben: ihr Datentyp (T), ihre Zugehörigkeit zum Primärschlüssel (P) und ihre Eingabepflicht (M – wie „***M***andatory“). Man erkennt, dass die Attribute des Primärschlüssels „NAME“ und „POSTLEITZAHL“ natürlich eingabepflichtig sind. Man sieht weiterhin, dass die beiden Attribute

„BETREUTE_SCH_NAME“ und
„BETREUTE_SCH_POSTLEITZAHL“,

die den Fremdschlüssel zur Repräsentation des Rekursiv-Beziehungstyps bilden, als nicht-eingabepflichtig angegeben sind: Nicht jede Schule muss auf eine betreute Schule verweisen.

Das Script enthält weiterhin eine Join-Beschreibung, durch die erst die Attribute-Kombination

„BETREUTE_SCH_NAME+BETREUTE_SCH_POSTLEITZAHL“

als Fremdschlüssel vereinbart wird:

```
CreateJoin C=FK_SCHULE_SCHULE_ZU_SCHULE  T=SCHULE
           P=SCHULE  D=restrict U=restrict
(
    P=NAME F=BETREUTE_SCH_NAME,
    P=POSTLEITZAHL F=BETREUTE_SCH_POSTLEITZAHL
);
```

Der Join ist durch seinen Code (C) benannt. Der Fremdschlüssel befindet sich in der Tabelle T und verweist auf die Tabelle P. Im Falle des Rekursiv-Beziehungstyps fallen diese Tabellen natürlich zusammen. Die nächsten beiden Parameter legen fest, wie das Datenbank-Managementsystem auf eine mögliche Verletzung der referenziellen Integrität durch Löschen (D – wie „***D***elete“) bzw. durch Verändern (U – wie „***U***pdate“) eines Primärschlüssel-Wertes reagieren soll. Durch „restrict“ wird angewiesen, dass beide Aktivitäten verhindert werden sollen (vgl. Abschnitt 3.4.2).

Für jedes Attribut des Fremdschlüssels wird sein Code (F) sowie der Code des korrespondierenden Attributs des Primärschlüssels (P) angegeben.

b) Als Repräsentation des M:CN-Beziehungstyp zwischen den beiden Objekttypen „Unterrichtsraum“ und „Unterrichtsverpflichtung“ enthält das Script zunächst die Typbeschreibungen für die drei benötigten Tabellen „RAUM“ (Code für „Unterrichtsraum“), „RAUM_ZU_UNTVERPFL“ (Koppel-Tabelle) und „UNTVERPFL“ (Code für „Unterrichtsverpflichtung“):

```
CreateTble C=RAUM N="Unterrichtsraum"
(
   C=SCHUL_NAME  T=Text(50) P=Yes M=Yes
                 N="Schul_Name",
   C=POSTLEITZAHL T=Integer P=Yes M=Yes
                  N="Postleitzahl",
   C=NUMMER T=Integer P=Yes M=Yes N="Nummer",
   C=FLAECHE T=Integer P=No M=No N="Fläche",
   C=SITZPLATZANZAHL T=Integer P=No M=No
                     N="Sitzplatzanzahl"
);
```

```
CreateTble C=RAUM_ZU_UNTVERPFL N="Raum zu Untverpfl"
(
   C=SCHUL_NAME_1 T=Text(50) P=Yes M=Yes
                  N="Schul_Name_1",
   C=POSTLEITZAHL_1  T=Integer P=Yes M=Yes
                     N="Postleitzahl_1",
   C=RAUM_NUMMER  T=Integer P=Yes M=Yes
                  N="Raum_Nummer",
   C=SCHUL_NAME_2  T=Text(50) P=Yes M=Yes
                   N="Schul_Name_2",
   C=POSTLEITZAHL_2  T=Integer P=Yes M=Yes
                     N="Postleitzahl_2",
   C=KLASSEN_BEZEICHNUNG T=Text(3) P=Yes M=Yes
                         N="Klassen_Bezeichnung",
   C=FACH_BEZEICHNUNG T=Text(15) P=Yes M=Yes
                      N="Fach_Bezeichnung",
   C=PERSONALNUMMER  T=Integer P=Yes M=Yes
                     N="Personalnummer"
);

CreateTble C=UNTVERPFL N="Unterrichtsverpflichtung"
(
   C=SCHUL_NAME  T=Text(50) P=Yes M=Yes
                 N="Schul_Name",
   C=POSTLEITZAHL T=Integer P=Yes M=Yes
                  N="Postleitzahl",
   C=KLASSEN_BEZEICHNUNG T=Text(3) P=Yes M=Yes
                         N="Klassen_Bezeichnung",
   C=FACH_BEZEICHNUNG T=Text(15) P=Yes M=Yes
                      N="Fach_Bezeichnung",
   C=PERSONALNUMMER  T=Integer P=Yes M=Yes
                     N="Personalnummer"
);
```

Es folgen zwei Join-Beschreibungen, die den sachlogischen Zusammenhang zwischen den Tabellen spezifizieren:

```
CreateJoin  C=FK_RAUM_ZU__WIRD_GENU_RAUM
            T=RAUM_ZU_UNTVERPFL  P=RAUM
            D=restrict U=restrict
(
    P=SCHUL_NAME F=SCHUL_NAME_1,
    P=POSTLEITZAHL F=POSTLEITZAHL_1,
    P=NUMMER F=RAUM_NUMMER
);


CreateJoin  C=FK_RAUM_ZU__FINDET_ST_UNTVERPF
            T=RAUM_ZU_UNTVERPFL  P=UNTVERPFL
            D=restrict U=restrict
(
    P=SCHUL_NAME F=SCHUL_NAME_2,
    P=POSTLEITZAHL F=POSTLEITZAHL_2,
    P=KLASSEN_BEZEICHNUNG F=KLASSEN_BEZEICHNUNG,
    P=FACH_BEZEICHNUNG F=FACH_BEZEICHNUNG,
    P=PERSONALNUMMER F=PERSONALNUMMER
);
```

Der Join „FK_RAUM_ZU__WIRD_GENU_RAUM“ beschreibt den Fremdschlüssel in der Tabelle „RAUM_ZU_UNTVERPFL“ (T), der auf die Tabelle „RAUM“ (P) verweist.

Durch den Join „FK_RAUM_ZU__FINDET_ST_UNTVERPF“ wird die Korrespondenz des Fremdschlüssels in der Tabelle „RAUM_ZU_UNTVERPFL“ (T) mit dem entsprechenden Primärschlüssel der Tabelle „UNTVERPFL“ (P) festgelegt.

4. Generierungs-Schritt

Im 4. Generierungs-Schritt wird im Kontext des Datenbank-Anwendungssystems „ACCESS2K.MDB“ die gewünschte Ziel-Datenbank durch Auswertung des Scripts „CREBAS.DAT“ generiert. Die entstandene Tabellen- und Beziehungsstruktur ist in der Abbildung 4-90 wiedergegeben.

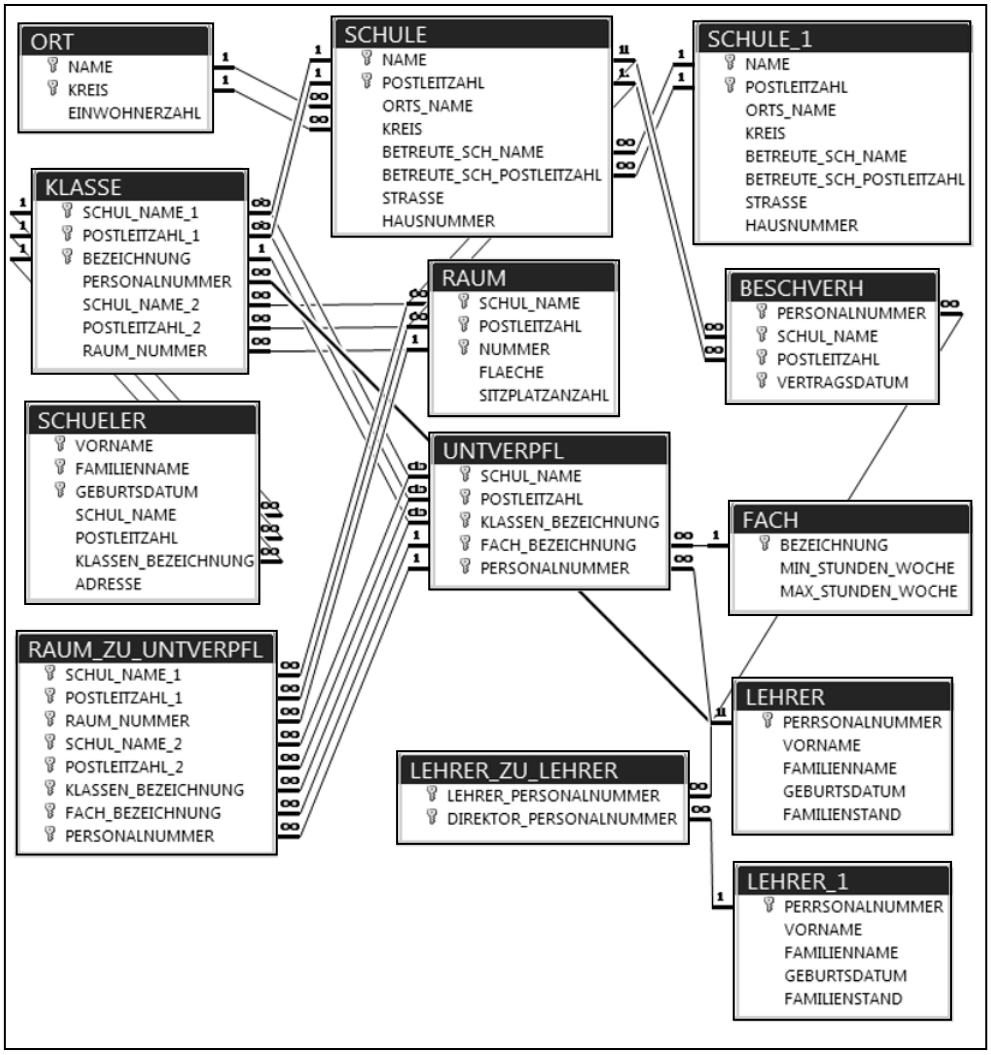


Abb. 4-90: Struktur der Datenbank für das Schulbeispiel

Wir konzentrieren uns wiederum auf die beiden herausgegriffenen Konstrukte:

a) *„Access“* stellt den CN:C-Rekursiv-Beziehungstyp „Schule *ist Pate für* Schule - Schule *hat als Paten* Schule“ graphisch dar, indem die Tabelle „SCHULE“ in einem zweiten Exemplar „SCHULE_1“ erscheint. Die beiden Fremdschlüssel-Attribute „BETREUTE_SCH_NAME“ bzw. „BETREUTE_SCH_POSTLEIT-

ZAHL“ in der Tabelle „SCHULE“ sind mit den Primärschlüssel-Attributen „NAME“ bzw. „POSTLEITZAHL“ in der Tabelle „SCHULE_1“ verknüpft. Die Kardinalitäts-Angaben (∞ bzw. 1) an den Verbindungslinien besagen, dass einer Schule *höchstens eine* betreute Schule zugeordnet werden kann, während auf eine betreute Schule *mehrere* Schulen verweisen können.

b) Der M:CN-Beziehungstyp zwischen den beiden Objekttypen „Unterrichtsraum“ und „Unterrichtsverpflichtung“ wurde durch die Koppel-Tabelle „RAUM_ZU_UNTVERPFL“ repräsentiert. Die ersten drei Attribute der Koppel-Tabelle bilden den Fremdschlüssel, der auf den Raum verweist. Die restlichen fünf Attribute der Koppel-Tabelle stellen den Fremdschlüssel dar, der auf die Unterrichtsverpflichtung (Tabelle „UNTVERPFL“) verweist. Die Kardinalitäten besagen im Einzelnen:

   - Eine Zuordnung in der Koppel-Tabelle betrifft stets *genau einen* Raum (1) und *genau eine* Unterrichtsverpflichtung (1).[27]
   - Ein Raum bzw. eine Unterrichtsverpflichtung kann *mehrfach* in der Koppel-Tabelle (∞) auftreten.

[27] Die Nicht-Optionalität wird durch die Angabe „M=Yes“ (***M***andatory) für die Fremdschlüssel-Attribute in der Tabellen-Typbeschreibung für die Koppel-Tabelle „RAUM_ZU_UNTVERPFL“ ausgedrückt.

5 Der Überblick: Möglichkeiten und Grenzen des Entity-Relationship-Modells und des relationalen Datenbank-Modells

In den Kapiteln 2 und 3 wurden das *Entity-Relationship-Modell* und das *relationale Datenbank-Modell* jeweils einzeln für sich behandelt. Im Kapitel 4 wurde dann untersucht, wie aus einem *konzeptionellen Datenmodell*, das mit den Mitteln der Sprache des Entity-Relationship-Modells aufgebaut wurde, die Tabellenstruktur für ein relationales Datenbank-Managementsystem – also das *logische Datenschema* - abgeleitet werden kann.

In allen drei Kapiteln erfolgte die Darlegung des Stoffs aus einer praxisorientierten Sicht und entsprach eher einer *phänomenologischen* Betrachtung. Der „rote Faden" folgte der inneren Logik, nach denen die Sprachkonstrukte des Entity-Relationship-Modells für die Abbildung der betrieblichen Realität herangezogen wurden. Viele Aussagen mögen dabei unsystematisch erscheinen. Manche Wiederholung war im Interesse des „Schritt-Tempos" unserer „Fußgängertour" nicht zu vermeiden.

In diesem Kapitel soll nun der Versuch unternommen werden, den „bunten Blumenstrauß" der bisher beschriebenen Konstrukte etwas zu sortieren. Insbesondere soll Ordnung in den „Dschungel" der dualen Beziehungstypen und der Rekursiv-Beziehungstypen gebracht werden.

Die Einordnung dieses Kapitels in den Kontext des Lehrbuchs zeigt die Abbildung 5-1.

In diesem Kapitel sollen die folgenden Fragen geklärt werden:

- Wo liegen die *Grenzen* der Modelle bei der Repräsentation von Objekttypen?
- Ist die CN-Notation der Beziehungstypen, die das Entity-Relationship-Modell verwendet, ausreichend und lassen sich die 10 dualen Beziehungstypen *konstruktiv herleiten*?
- Wann sollte ein dualer Beziehungstyp in einen *Koppel-Objekttyp* umgewandelt werden?

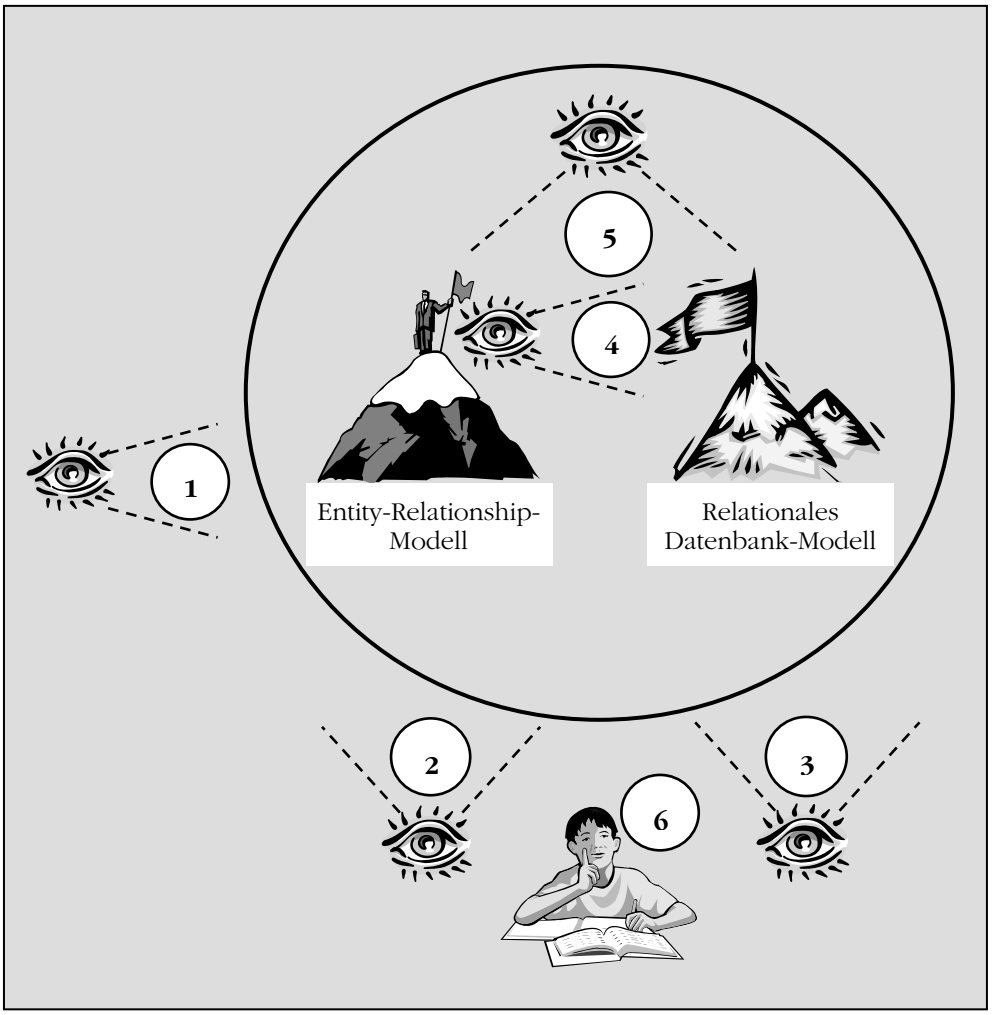


Abb. 5-1: Gegenstand des Kapitels 5

- Welche der 10 dualen Beziehungstypen des Entity-Relationship-Modells sind überhaupt *erforderlich* und lassen sich diese im relationalen Datenbank-Modell *repräsentieren*?
- Wie lassen sich die 7 Rekursiv-Beziehungstypen des Entity-Relationship-Modells *konstruktiv herleiten*?
- Wann sollte ein Rekursiv-Beziehungstyp in einen *Koppel-Objekttyp* umgewandelt werden?

- Welche der 7 Rekursiv-Beziehungstypen werden *benötigt* und welche lassen sich im relationalen Datenbank-Modell *repräsentieren*?

Diese Probleme sollen in den folgenden Abschnitten systematisch untersucht werden. Die Darlegungen enthalten hauptsächlich theoretische Überlegungen, die jedoch durch Beispiele veranschaulicht werden.

5.1 Der Objekttyp

Im Abschnitt 2.1 hatten wir als ersten Schritt des Modellierungsprozesses die Aufgabe der Klassifizierung besprochen. Im Zuge der Klassifizierung wird die Vielfalt und Komplexität der betrieblichen Realität dadurch reduziert, dass die relevanten Objekte zu Klassen, nämlich zu den *Objekttypen*, zusammengefasst werden. Wir hatten den Objekttyp als eine - durch einen Objekttyp-Namen eindeutig benannte - Klasse von Objekten definiert, über die dieselben Informationen gespeichert werden und die in prinzipiell gleicher Weise verarbeitet werden.

Bei der Verwendung von Objekttypen im Entity-Relationship-Modell bzw. bei ihrer Repräsentation im relationalen Datenbank-Modell stößt man auf Probleme und Grenzen, von denen wir im Folgenden einige näher untersuchen wollen.

5.1.1 Darstellung dualer sachlogischer Zusammenhänge durch Objekttypen

Neben dem Aspekt der prinzipiell gleichen Verarbeitung besteht ein wesentliches Merkmal der Objekte eines *Objekttyps* darin, dass über sie Informationen (Fakten) gespeichert werden sollen. Beziehungen als Instanzen eines *Beziehungstyps* dienen aber ebenfalls dazu, Fakten festzuhalten. In der Praxis der Datenmodellierung steht man somit häufig vor der Entscheidung, einen zu speichernden Fakt entweder durch einen Objekttyp oder durch einen Beziehungstyp darzustellen.

Um diese Entscheidung fundierter treffen zu können, wollen wir Fakten in zwei Gruppen einteilen – in *Objekttyp-Fakten* und in *Beziehungstyp-Fakten*. Betrachten wir dazu drei Fakten, die für die Marketing-Abteilung eines Reiseunternehmens gespeichert werden sollen! Wir formulieren sie in einer Form, die eine Verallgemeinerung ihrer Struktur in einfacher Weise ermöglicht:

1. *Für den* ***Kunden*** *mit der Kundennummer* ***12345*** *gilt, dass seine* ***Hausnummer*** *den Wert* ***42*** *hat.*
2. *Für das* ***Reiseziel*** *mit der Bezeichnung* ***Mauritius*** *gilt, dass seine* ***Einwohnerzahl*** *den Wer* ***1.100.000*** *hat.*
3. *Für den sachlogischen Zusammenhang* ***„Kunde hat besucht Reiseziel / Reiseziel wurde besucht von Kunde“*** *gilt, dass der* ***Kunde*** *mit der Kundennummer* ***12345*** *das* ***Reiseziel*** *mit der Bezeichnung* ***Mauritius*** *besucht hat.*

Die Fakten 1 und 2 haben die Struktur eines Objekttyp-Fakts:

Objekttyp-Fakt

Definition: Ein *Objekttyp-Fakt* ist eine Aussage mit der folgenden Struktur:

Im Rahmen eines Objekttyps T gilt, dass für eines seiner *Objekte* O die *Eigenschaft* E den *Wert* W besitzt.

Ein Objekttyp-Fakt ist also ein Quadrupel der Form:

(T,O,E,W).

Er wird auf der Ebene des konzeptionellen Datenmodells durch einen Objekttyp T mit der Eigenschaft E dargestellt, zu deren Domäne der Wert W gehört.

Die Information 3 hat dagegen eine ganz andere Struktur. Informationen dieser Art wollen wir als *duale Beziehungstyp-Fakten* bezeichnen. Eine Erweiterung auf höhergradige Beziehungstyp-Fakten betrachten wir im Abschnitt 5.1.2.

Dualer Beziehungstyp-Fakt

Definition: Ein *dualer Beziehungstyp-Fakt* ist eine Aussage mit der folgenden Struktur:

Im Rahmen eines *Beziehungstyps* T, der einen sachlogischen Zusammenhang zwischen den beiden Objekttypen A und B beschreibt, besagt eine konkrete Bezie-

hung, dass ein *Objekt* a *des Objekttyps* A den betrachteten sachlogischen Zusammenhang mit einem *Objekt* b *des Objekttyps* B eingeht. Dabei können die Objekttypen A und B identisch sein (Rekursiv-Beziehungstyp-Fakt).

Ein Beziehungstyp-Fakt ist also ein Tripel der Form:

$$(T,a,b) \quad \text{mit} \quad a \in A,\ b \in B.$$

Er wird auf der Ebene des konzeptionellen Datenmodells durch einen Beziehungstyp T dargestellt, der den sachlogischen Zusammenhang zwischen den Objekttypen A und B beschreibt.

Richten wir nun unser Augenmerk darauf, wie sich ein sachlogischer Zusammenhang im konzeptionellen Datenmodell darstellen lässt!

Im Zuge der Datenmodellierung würde aus unseren drei Beispiel-Fakten das Datenmodell abgeleitet werden, das in der Abbildung 5-2 wiedergegeben ist. Dabei wurde angenommen, dass Kunden auch dann gespeichert werden, wenn sie noch kein Reiseziel besucht haben, und dass es neue Reiseziele geben kann, die noch von keinem Kunden besucht wurden.

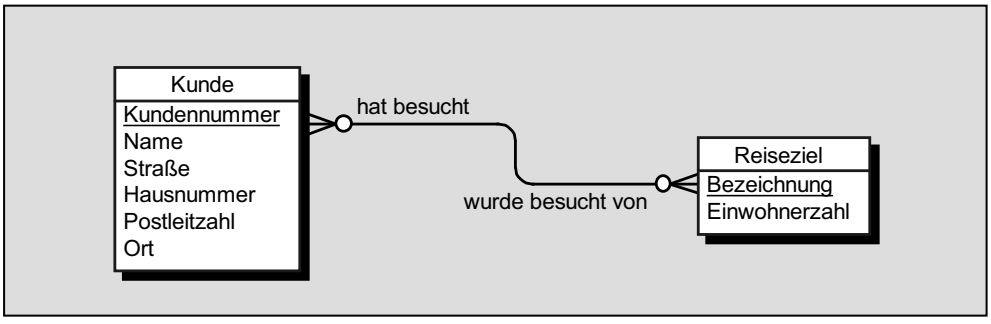


Abb. 5-2: Darstellung des sachlogischen Zusammenhangs durch einen dualen Beziehungstyp

Beachtung der Granularität

Die Struktur des Fakts 3 hängt nun aber stark von der Granularität ab, mit der die reale Welt beschrieben werden soll. In der Buchungsabteilung des Reiseunternehmens würde man sich wohl nicht damit zufrieden geben, den sachlogischen Zusammenhang zwischen dem Kunden und dem Reiseziel lediglich durch den „dürren" Fakt zu beschreiben, dass der Kunde das Reiseziel besucht hat. Dort interessiert man sich für viele Details dieses Zusammenhangs, so beispielsweise für Beginn und Ende der Reise, für den Reisepreis und für vieles mehr. Der Fakt 3 müsste für die Belange der Buchungsabteilung wie folgt präzisiert werden:

3a) *Für den sachlogischen Zusammenhang „Kunde hat besucht Reiseziel / Reiseziel wurde besucht von Kunden", also für die* ***Reise*** *des* ***Kunden 12345 zum Reiseziel Mauritius*** *gilt, dass ihr* ***Beginn*** *den Wert* ***01.06.2009*** *hat.*

3b) *Für die* ***Reise*** *des* ***Kunden 12345 zum Reiseziel Mauritius*** *gilt, dass ihr* ***Ende*** *den Wert* ***15.06.2009*** *hat.*

3c) *Für die* ***Reise*** *des* ***Kunden 12345 zum Reiseziel Mauritius*** *gilt, dass ihr* ***Preis*** *den Wert* ***1.400 EUR*** *hat.*

Nun hat der Beziehungstyp-Fakt 3 die Struktur von drei Objekttyp-Fakten - also die Struktur (T,O,E,W) - angenommen:

- Die Rolle des Objekttyps T spielt der bisherige Beziehungstyp „Kunde hat besucht Reiseziel/Reiseziel wurde besucht von Kunde", der nun den Namen „Reise" annimmt.
- Das Objekt O wird durch die Kombination der Eigenschaften „Kunde 12345 + Reiseziel Mauritius" identifiziert.[28]
- Die Eigenschaft E heißt im Fall 3a „Beginn".
- Der Wert W lautet im Fall 3a „01.06.2009".

Will man die Realität in der angegebenen Granularität beschreiben, muss der sachlogische Zusammenhang bei der Datenmodellierung durch den Objekttyp „Reise" dargestellt werden, dessen Objekte durch ihren Zusammenhang mit einem Kunden und einem Reiseziel identifiziert werden und der durch die Eigenschaften „Beginn", „Ende" und „Preis" beschrieben wird. Abbildung 5-3 zeigt das Modellierungs-Ergebnis.

[28] Dabei wird unterstellt, dass ein Kunde ein Reiseziel nur einmal besucht.

Man erkennt, das ein sachlogischer Zusammenhang im Entity-Relationship-Modell als Beziehungstyp, aber auch als eigenständiger Objekttyp dargestellt werden kann. Im Abschnitt 2.5.2 hatten wir diesen Sachverhalt unter der Überschrift „Eigenschaften von Beziehungstypen“ behandelt.

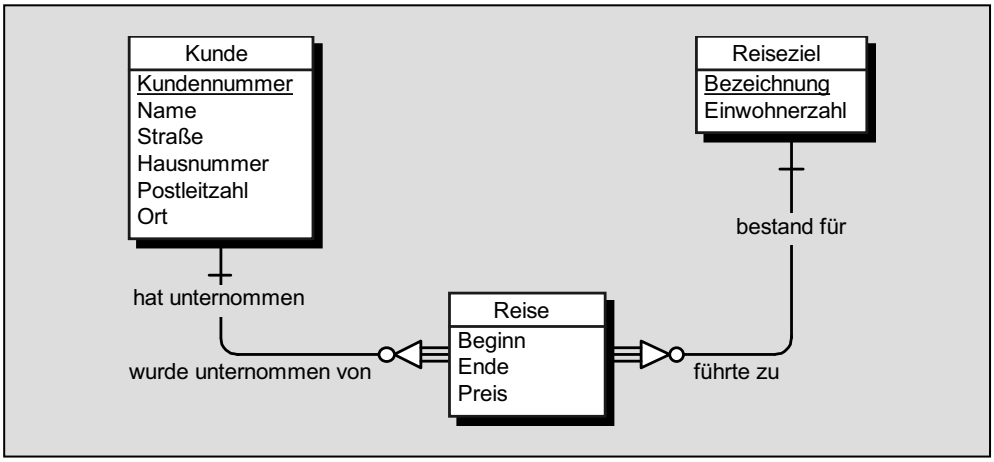


Abb. 5-3: Darstellung des sachlogischen Zusammenhangs durch einen Objekttyp

Beziehungstypen lassen sich nicht „attributieren“

In dieser Tatsache offenbart sich ein wesentlicher Mangel des Entity-Relationship-Modells: Beziehungstypen können nämlich - im Gegensatz zu den Objekttypen - nicht durch Eigenschaften näher beschreiben werden. Man sagt: Beziehungstypen lassen sich nicht *„attributieren“*. Will man über einen sachlogischen Zusammenhang also detailliertere Fakten speichern, so müssen sie im Datenmodell mit Hilfe eines eigenständigen Objekttyps dargestellt werden.

Die Entscheidung für oder gegen die Verwendung eines Objekttyps zur Darstellung eines sachlogischen Zusammenhangs kann also nicht – wie es wünschenswert wäre - auf semantischer Grundlage erfolgen, sondern geschieht wegen der Ausdrucksschwäche des Entity-Relationship-Modells aus rein strukturellen Erwägungen. Im Abschnitt 2.5.2 hatten wir dafür bereits eine einfache Faustregel angegeben, die hier noch einmal wiederholt werden soll:

Objekttyp oder Beziehungstyp?

Faustregel: Soll über das Zusammenwirken von jeweils zwei Objekten a und b, die zwei verschiedenen Objekttypen (bzw. demselben Objekttyp) entstammen, nicht mehr festgehalten werden als die reine Tatsache, dass a und b gemäß der angegebenen Semantik miteinander verbunden sind, dann wird dies durch einen dualen Beziehungstyp (bzw. durch einen Rekursiv-Beziehungstyp) modelliert.

Sollen jedoch über das konkrete Zusammenwirken von a und b weitere Angaben gespeichert werden, so muss der Beziehungstyp in einen neuen Objekttyp umgewandelt werden, dem die Angaben als Eigenschaften zugeordnet werden.

Koppel-Objekttyp

Im relationalen Datenbank-Modell treten zusätzliche Einschränkungen auf. Duale Beziehungstypen mit beidseitiger Kardinalität N – also M:N-, M:CN- und CM:CN-Beziehungstypen – lassen sich zwar im Entity-Relationship-Modell in natürlicher Weise darstellen, vor ihrer Überführung in das relationale Datenbank-Modell müssen sie jedoch erst in einen *Koppel-Objekttyp* umgewandelt werden. Die Elemente dieses Koppel-Objekttyps sind von ihrer Semantik her eigentlich keine *Objekte*, sondern *nicht-attributierte Beziehungen zwischen Objekten*. Die Repräsentation durch eine Koppel-Tabelle ist somit eine reine „Verlegenheitslösung", die auf der methodischen Unzulänglichkeit des relationalen Datenbank-Modells beruht, nur 1:CN-Beziehungstypen adäquat darstellen zu können.

Umwandlung in einen Koppel-Objekttyp?

Mit der Frage, ob man gegebenenfalls auch andere duale Beziehungstypen vor ihrer Überführung in das relationale Datenbank-Modell besser in Koppel-Objekttypen umwandeln sollte, werden wir uns im Abschnitt 5.2.3 beschäftigen.

5.1.2 Darstellung höhergradiger sachlogischer Zusammenhänge durch Objekttypen

Wir wollen nun die im vorangegangenen Abschnitt formulierte Faustregel relativieren. Ein sachlogischer Zusammenhang muss mitunter auch dann durch einen Objekttyp dargestellt werden, wenn der Zusammenhang gar nicht durch Eigenschaften näher

präzisiert werden soll. Das ist nämlich immer dann erforderlich, wenn sich der sachlogische Zusammenhang nicht in Form eines *dualen* Beziehungstyp-Fakts formulieren lässt, sondern nur in Form eines *höhergradigen Beziehungstyp-Fakts:*

Höhergradiger Beziehungstyp-Fakt

Definition: Ein *höhergradiger Beziehungstyp-Fakt* ist eine Aussage mit folgender Struktur:

Im Rahmen eines *sachlogischen Zusammenhangs Z*, an dem die Objekttypen A_1, A_2, ..., A_n beteilt sind, besagt ein konkreter Zusammenhang, dass die Objekte a_1, a_2, ..., a_n zu einer gemeinsamen „Aktivität" zusammengeführt sind.

Er ist also ein (n+1)-Tupel der Form:

$$(Z, a_1, a_2, ..., a_n) \text{ mit}$$
$$a_1 \in A_1, a_2 \in A_2, ..., a_n \in A_n.$$

Er wird auf der Ebene des Datenmodells durch einen Objekttyp Z dargestellt, der mit den Objekttypen A_1, A_2, ..., A_n durch duale Beziehungstypen verbunden ist.

Als Beispiel betrachten wir wieder unser Reiseunternehmen, das nun höhergradige Beziehungstyp-Fakten der folgenden Art speichern möchte:

> *Für den sachlogischen Zusammenhang **„Kunde besucht Reiseziel in Saison mit Fluggesellschaft"** gilt, dass der **Kunde** (Kundennummer **12345**) das **Reiseziel** (Name **Mauritius**) in der **Saison** (Bezeichnung **A**) mit der **Fluggesellschaft** (Code **LH**) besucht hat.*

Da das Entity-Relationship-Modell keine Konstrukte für die unmittelbare Darstellung höhergradiger Beziehungstyp-Fakten bereitstellt, muss – wieder zweckentfremdet – das Konstruktionselement „Objekttyp" verwendet werden. Wir hatten darauf bereits im Abschnitt 2.5.1 unter der Überschrift „Sachlogische Zusammenhänge zwischen mehr als 2 Objekttypen" hingewiesen. Der Beispiel-Fakt führt zu dem Datenmodell, das in der Abbildung 5-4 dargestellt ist.

Für den Objekttyp, dessen Elemente jeweils vier Objekte aus den Objekttypen „Kunde“, „Reiseziel“, „Saison“ und „Fluggesellschaft“ zu einer gemeinsamen Aktivität zusammenführen, wurde ganz bewusst die semantisch leere Bezeichnung „Vorgang“ gewählt. Damit soll ausgedrückt werden, dass er eigentlich keine Klasse von *Objekten* darstellt, sondern eine Klasse (höhergradiger) *sachlogischer Zusammenhänge*.

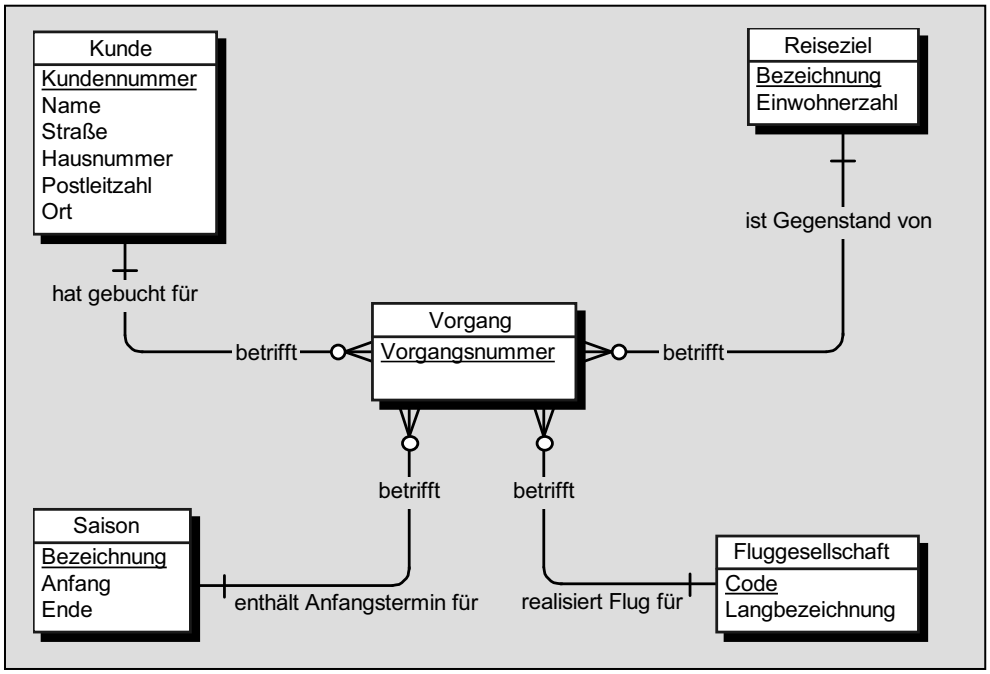


Abb. 5-4: Darstellung eines höhergradigen Beziehungstyp-Fakts durch einen Objekttyp

5.1.3 Hierarchisch geordnete Objekttypen

Ein Nachteil des Entity-Relationship-Modells besteht – zumindest in der hier beschriebenen Ausbaustufe – darin, dass sämtliche Objekttypen auf derselben Ebene liegen, dass es zwischen ihnen keine *hierarchische Ordnung* gibt. In späteren Ausbaustufen wurde dieser Mangel überwunden, indem die Prinzipien von *Generalisierung* und *Spezialisierung* in das methodische Arsenal des Entity-Relationship-Modells aufgenommen wurden. Wegen

der angestrebten Einfachheit dieses Lehrbuchs haben wir auf die Darstellung dieser Prinzipien verzichtet. Der interessierte Leser sei auf andere Lehrbücher - beispielsweise auf [MOOS04] - verwiesen.

Das relationale Datenbank-Modell kennt keine Hierarchien

Selbst dann, wenn man bei der Modellierung mit der Sprache des Entity-Relationship-Modells eine hierarchische Ordnung der Objekttypen beschreibt, muss man sie vor der Umsetzung des Datenmodells in das relationale Datenbank-Modell wieder aufgeben. Ein Grundprinzip des relationalen Datenbank-Modells besteht nämlich gerade darin, dass sämtliche Tabellen voneinander unabhängig sind. Das bedeutet insbesondere, dass es zwischen ihnen auch keine hierarchische Ordnung geben kann.

5.1.4 Komplex strukturierte Objekte

Im Abschnitt 5.1.1 wurden die Objekttyp-Fakten als Quadrupel (T,O,E,W) eingeführt. Sie drücken den Sachverhalt aus, dass ein zum Objekttyp T gehöriges Objekt O für die Eigenschaft E den Wert W annimmt. Die 1. Normalform (vgl. Abschnitt 2.6.1) fordert nun aber, dass der Wert W atomar sein muss. Das heißt: Er darf nicht mehrgliedrig sein, er darf insbesondere keine Liste oder Menge von Werten enthalten.

Mit der Forderung, dass sich das konzeptionelle Datenmodell in der 1. Normalform befinden soll, wird schon bei der Datenmodellierung der Beschränkung relationaler Datenbank-Managementsysteme Rechnung getragen, die für die Attribute der Tabellen nur solche Domänen zulassen, die als Mengen atomarer Werte definiert sind.

Beschränkung auf atomare Attributwerte

Die Beschränkung auf atomare Attributwerte ist die einschneidendste Bedingung sowohl des Entity-Relationship-Modells als auch des relationalen Datenbank-Modells. Sie führt dazu, dass sich relationale Datenbanken für sogenannte *Nicht-Standard-Anwendungen*, in denen man Informationen über komplex strukturierte Objekte speichern muss, nur mit großen Problemen oder gar nicht einsetzen lassen. Zu diesen Bereichen zählen beispielsweise Systeme zur rechnergestützten Konstruktion (CAD – ***C***omputer ***A***ided ***D***esign) und Systeme zur Prozess-Steuerung.

„Flache“ Tabellen

Im relationalen Datenbank-Modell müssen Daten über komplex strukturierte Objekte auf viele „flache“ Tabellen verteilt werden. Der sachlogische Zusammenhang zwischen diesen Daten muss bei ihrer Verarbeitung jedes Mal neu durch aufwendige Join-Operationen hergestellt werden. Im Abschnitt 3.2 wurde dafür

der Vergleich mit einem Auto herangezogen, das man bei seiner Speicherung – also beim Parken in der Garage – erst in seine Einzelteile zerlegen muss, wobei der ursprüngliche Zusammenhang zwischen den Teile durch ein Verweissystem - durch die Fremdschlüssel - darzustellen ist.

Komplex strukturiertes Objekt: Polyeder

Betrachten wir als ein einfaches Beispiel die Speicherung der räumlichen Position eines Polyeders, dessen Oberfläche aus wenigstens 4 verschiedenfarbigen Polygonflächen, den Seitenflächen, gebildet wird. Bei der Speicherung muss das Polyeder zunächst in seine Seitenflächen zerlegt werden. Jede Polygonfläche muss wiederum durch die Raumpositionen von mindestens 3 Eckpunkten beschrieben werden. Die Abbildung 5-5 zeigt das entsprechende Datenmodell.

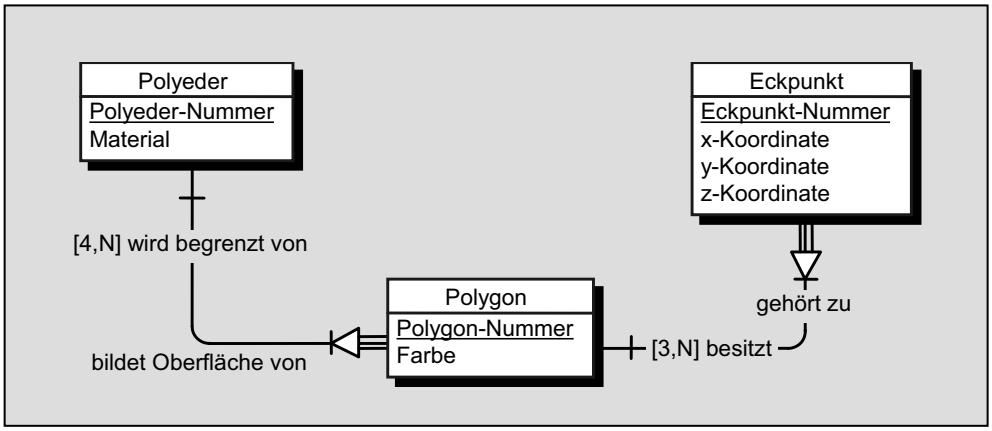


Abb. 5-5: Darstellung eines Polyeders durch „flache“ Objekttypen

Den Ausweg aus den Beschränkungen, die durch die 1. Normalform der Datenspeicherung auferlegt werden, bieten die sogenannten NF^2-Systeme (=NFNF-Systeme = Non-First-Normal-Form-Systeme), auf die bereits im Abschnitt 2.6.1 hingewiesen wurde. Zu diesen Systemen gehören insbesondere die objektorientierten Datenbank-Managementsysteme (OODBMS), die im Rahmen dieses Lehrbuchs nicht behandelt werden. Der interessierte Leser sei auf die spezielle Fachliteratur - beispielsweise auf [MCLA07] und [HEIN08] - verwiesen.

5.2 Die dualen Beziehungstypen

Ein dualer Beziehungstyp D entspricht einer Menge sachlogischer Zusammenhänge, die zwischen jeweils zwei Objekten aus unterschiedlichen Objekttypen bestehen. Er repräsentiert somit die Menge der Objektpaare (a,b), wobei a ein Element des Objekttyps A ist und b ein Element des Objekttyps B ist:

$$D = \{(a,b) \mid a \in A, b \in B\}$$

Im Datenmodell werden nicht die konkreten Objektpaarungen beschrieben, sondern es werden für die Menge D aller Paarungen lediglich 4 charakterisierende Angaben festgehalten:

1. Die Semantik der Beziehungstyp-Richtung A→B. Wir gehen bei den weiteren Überlegungen von der Semantik „Element des Objekttyps A *sendet eine Nachricht an* Element des Objekttyps B“ aus.
2. Die Semantik der Beziehungstyp-Richtung A←B, die wir wie folgt annehmen: „Element des Objekttyps B *empfängt eine Nachricht von* Element des Objekttyps A“.
3. Die *Optionalität* und die *Kardinalität* der Beziehungstyp-Richtung A→B.
4. Die *Optionalität* und die *Kardinalität* der Beziehungstyp-Richtung A←B.

Im folgenden Abschnitt 5.2.1 untersuchen wir zunächst die Notationen von Optionalität und Kardinalität näher und leiten dann im Abschnitt 5.2.2 daraus eine Systematik der dualen Beziehungstypen her.

5.2.1 Optionalität und Kardinalität einer Beziehungstyp-Richtung

Wir beschränken unsere Überlegungen zunächst auf die eine Beziehungstyp-Richtung A→B.

Optionalität

Die Festlegung der *Optionalität* der Beziehungstyp-Richtung beantwortet *eine einzige* Frage:

1. Kann es sein, dass ein Objekt a ∈ A *keine* Nachricht an ein Objekt b ∈ B sendet?

Kardinalität

Die Festlegung der *Kardinalität* der Beziehungstyp-Richtung beantwortet *zwei* Fragen:

1. Kann es sein, dass ein Objekt a ∈ A *eine* Nachricht an ein Objekt b ∈ B sendet?
2. Kann es sein, dass ein Objekt a ∈ A *mehr als eine* Nachricht an ein Objekt b ∈ B sendet?

Insgesamt sind es also drei Entscheidungsfragen, die jede für sich mit „ja“ oder „nein“ beantwortet werden kann. Stellt man die p=3 Bedingungen in einer Entscheidungstabelle dar, so ergeben sich kombinatorisch

$$n = 2^p = 2^3 = 8$$

01N-Notation

unterschiedliche Fälle[29], die durch eine Notation voneinander unterschieden werden müssen. Wir wählen für die positive bzw. negative Beantwortung der ersten Frage die Symbole 0^+ bzw. 0^-.

Analog dazu führt die zweite Frage zu den Symbolen 1^+ bzw. 1^-. Drückt man – wie bei der Beschreibung von Datenstrukturen allgemein üblich - das Prädikat „mehr als eine Nachricht“ mit dem Buchstaben N aus, so ergeben sich die Symbole N^+ bzw. N^-.

Die sich mit dieser *01N-Notation* ergebende Fallunterscheidung von Optionalität/Kardinalität ist in der Abbildung 5-6 dargestellt.

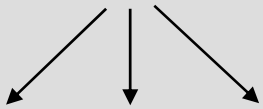
Die kombinatorische Vielfalt der Entscheidungstabelle liefert zwei Fälle, denen keine Notation zugeordnet wurde, weil sie für unsere Belange nicht relevant sind:

Fall 1: Ein Objekt a ∈ A kann *weder keine, noch eine, noch mehr als eine* Nachricht an ein Objekt b ∈ B senden. Das bedeutet, dass nur eine *negative Anzahl* von Nachrichten gesendet werden kann. Dieser Fall „imaginärer“ Nachrichtensendungen hat in der Praxis keine Entsprechung.

[29] Die angegebene Formel für die kombinatorische Anzahl der Fälle bei p voneinander unabhängigen Entscheidungsfragen ist leicht einzusehen. Die erste Frage kann auf 2 Arten („ja“ bzw. „nein“) beantwortet werden. Für jeden dieser Fälle kann die zweite Frage wieder auf 2 Arten beantwortet werden. Das ergibt bereits 4 Fälle. Für jeden dieser Fälle kann die dritte Frage wieder auf 2 Arten beantwortet werden. Man erhält also 8 Fälle. Die Verallgemeinerung auf p Entscheidungsfragen führt dazu, dass die Zahl 2 p-mal als Faktor auftritt.

Fall 5: Ein Objekt a ∈ A kann *zwar keine, aber weder eine, noch mehr als eine* Nachricht an ein Objekt b ∈ B senden. Das bedeutet, dass *kein* A-Objekt eine Nachricht an ein B-Objekt sendet, dass es also keinen sachlogischen Zusammenhang zwischen A-Objekten und B-Objekten gibt. Folglich liegt auch kein dualer Beziehungstyp vor.

Kann es sein, dass ein Objekt a ∈ A



Fall	*keine*	*eine*	*mehr als eine*	Notation
1	nein	nein	nein	
2	nein	nein	ja	$0^{-}1^{-}N^{+}$
3	nein	ja	nein	$0^{-}1^{+}N^{-}$
4	nein	ja	ja	$0^{-}1^{+}N^{+}$
5	ja	nein	nein	
6	ja	nein	ja	$0^{+}1^{-}N^{+}$
7	ja	ja	nein	$0^{+}1^{+}N^{-}$
8	ja	ja	ja	$0^{+}1^{+}N^{+}$

Nachricht an ein Objekt b ∈ B sendet?

Abb. 5-6: Kombinatorische Fallunterscheidung von Optionalität/Kardinalität

[Min,Max]-Notation

Die kombinatorisch sinnvollen 6 Fälle werden auf 5 Fälle reduziert, wenn man die kombinatorische Vielfalt auf eine Intervall-Betrachtung einengt. Als Notation wird nun die Intervall-Angabe [Min,Max] verwendet. Diese Betrachtung führt zur intervallbasierten Fallunterscheidung der Abbildung 5-7.

Die Intervall-Betrachtung liefert nur einen Fall ohne zugeordnete Notation:

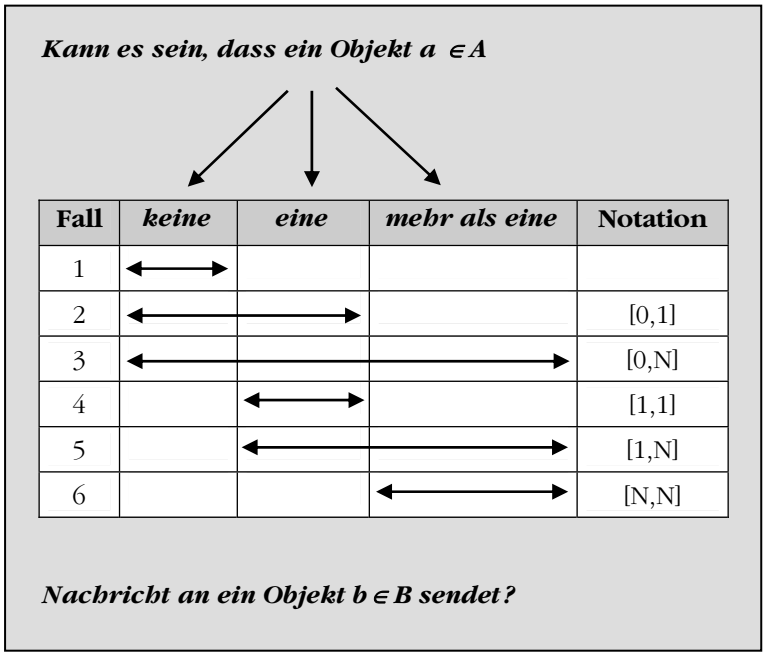


Abb. 5-7: Intervallbasierte Fallunterscheidung von Optionalität/Kardinalität

Fall 1: Ein Objekt a ∈ A kann lediglich keine Nachricht an ein Objekt b ∈ B senden, so dass gar kein sachlogischer Zusammenhang vorliegt.

C1N-Notation und „Krähenfuß"-Notation

In der üblichen Praxis der Datenstruktur-Beschreibung engt man die Fallunterscheidung noch weiter ein, indem man nur diejenigen Intervalle zulässt, die die Aussage „ein Objekt a ∈ A kann *eine* Nachricht an ein Objekt b ∈ B senden" einschließen. Dann ergeben sich nur noch 4 sinnvolle Fälle, für die die beiden Notationen gebräuchlich sind, die wir bisher in diesem Buch verwendet haben, nämlich als graphische Notation die *„Krähenfuß"-Notation* und als alphanumerische Notation die *C1N-Notation*. Aus der Abbildung 5-8 wird die schrittweise „Verwässerung" der Fallunterscheidung ersichtlich.

Man erkennt aus der Abbildung 5-8, dass zwei wichtige Fälle der Optionalität/Kardinalität durch die Krähenfuß- bzw. C1N-Notation nicht dargestellt werden können:

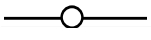
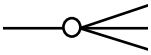
Fall	01N-Notation	[Min,Max]-Notation	Krähenfuß-Notation	C1N-Notation
1	$0^{-}1^{-}N^{+}$	[N,N]		
2	$0^{-}1^{+}N^{-}$	[1,1]		1
3	$0^{-}1^{+}N^{+}$	[1,N]		N
4	$0^{+}1^{-}N^{+}$			
5	$0^{+}1^{+}N^{-}$	[0,1]		C
6	$0^{+}1^{+}N^{+}$	[0,N]		CN

Abb. 5-8: Notationen für Optionalität/Kardinalität

Fall 1: Ein Objekt a ∈ A kann *nur mehr als eine* Nachricht an ein Objekt b ∈ B senden. Dieser Fall ist in der Praxis häufig anzutreffen. Nehmen wir an, an einer Hochschule wird ein vorgesehenes Seminar erst dann gespeichert, wenn sich mindestens 5 Studenten dafür eingeschrieben haben. Die Abbildung 5-9 zeigt die Darstellung dieses Sachverhalts im Datenmodell.

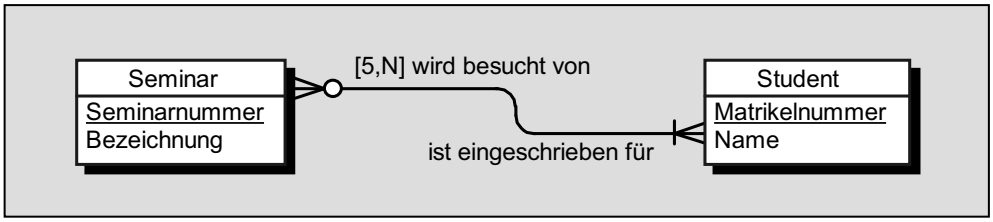


Abb. 5-9: Beispiel für die Optionalität/Kardinalität $0^{-}1^{-}N^{+}$

Ein Objekt des Objekttyps „Seminar“ bezieht seine Daseinsberechtigung dann nur aus dem Umstand, dass es mit *mehr als einem* (nämlich mit mindestens fünf) „Studenten“ in einem sachlogischen Zusammenhang steht. Im Entity-Relationship-Modell lässt sich aber dieser

Sachverhalt *graphisch* – und damit in syntaktisch auswertbarer Form - nur „vergröbert“ mit Hilfe der Notation für den Fall 3 (*ein oder mehrere* Studenten) beschreiben. Durch eine Minimalangabe für die Kardinalität kann man die Forderung zwar *verbal* dokumentieren, sie lässt sich entsprechend unseren Ausführungen im Abschnitt 4.5 jedoch nicht in den Tabellen-Typbeschreibungen verankern.

Fall 4: Ein Objekt a ∈ A kann entweder keine Nachricht oder mehr als eine Nachricht an ein Objekt b ∈ B senden. Auch dieser Fall ist in der Praxis mitunter anzutreffen. Nehmen wir an, dass ein Unternehmen, das Autos vermietet, eine neue Vermietstation einrichtet und in die Datenbank aufnimmt. Zunächst hat eine im Aufbau befindliche Vermietstation kein Auto. Wenn sie das erste Mal mit Autos bestückt wird, dann werden prinzipiell mit einem Großtransporter 8 Autos auf einmal angeliefert. Jede Vermietstation hat also entweder kein Auto oder mindestens 8 Autos. In der Abbildung 5-10 ist ein Versuch gezeigt, diesen Umstand im Datenmodell auszudrücken.

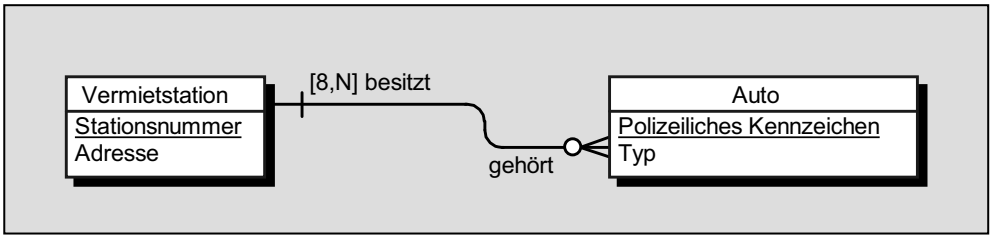


Abb. 5-10: Beispiel für die Optionalität/Kardinalität $0^{+}\,1^{-}\,N^{+}$

Die Minimalzahl 8 schränkt dabei ausschließlich die *Kardinalität* ein und tangiert nicht die *Optionalität* der Beziehungstyp-Richtung.

Die beschriebenen Probleme bei der Repräsentation von Optionalität und Kardinalität im Datenmodell resultieren also aus dem Umstand, dass sich im Entity-Relationship-Modell nur solche [Min,Max]-Intervalle darstellen lassen, die jeweils die Kardinalität 1 einschließen. Diese Beschränkung führt zu einer Systematik der Beziehungstypen, die im folgenden Abschnitt vorgestellt wird.

5.2.2 Die Systematik der dualen Beziehungstypen

Im Abschnitt 5.2.1 wurde untersucht, welche Fallunterscheidungen für die Kombination Optionalität/Kardinalität einer Beziehungstyp-Richtung zu treffen sind. In zwei Schritten wurden die prinzipiell möglichen 6 Fälle auf 4 Fälle reduziert.

Ein Beziehungstyp setzt sich aber aus zwei Beziehungstyp-Richtungen zusammen. Ein Objekt des Objekttyps A sendet nicht nur Nachrichten an Objekte des Objekttyps B, sondern ein B-Objekt empfängt auch Nachrichten von A-Objekten. Dabei ergeben sich 5 Konstellationen, die die Grundelemente für die Beziehungstypen bilden. Diese sind in der Abbildung 5-11 dargestellt.

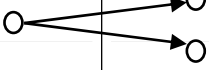
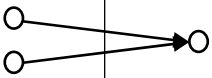
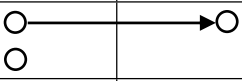
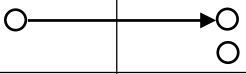
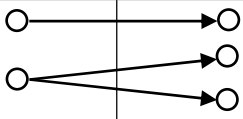
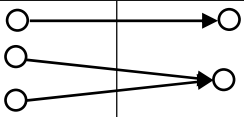
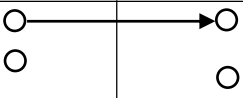
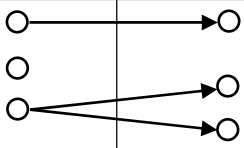
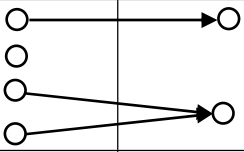
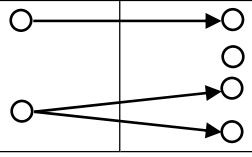
Grund-element	Objekt-typ A	Objekt-typ B	Erläuterungen
1			A-Objekt sendet einmal, B-Objekt empfängt einmal
2			A-Objekt sendet nicht
3			B-Objekt empfängt nicht
4			A-Objekt sendet mehrmals, B-Objekt empfängt einmal
5			A-Objekt sendet einmal, B-Objekt empfängt mehrmals

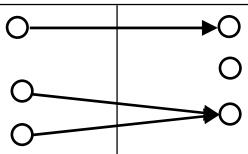
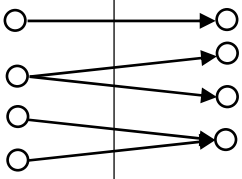
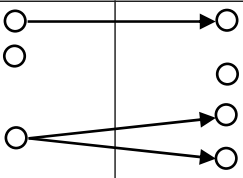
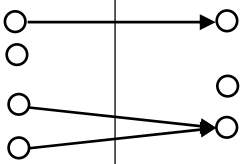
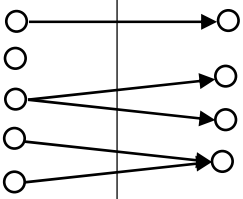
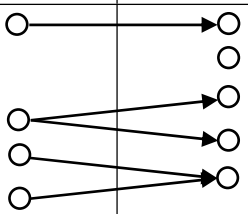
Abb. 5-11: Grundelemente der Beziehungstypen

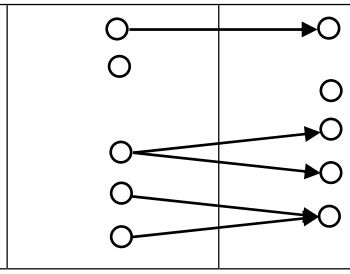
Aus diesen 5 Grundelementen lassen sich nun alle Beziehungstypen systematisch ableiten, die im Entity-Relationship-Modell dargestellt werden können. Wir hatten im Abschnitt 5.2.1 bereits darauf hingewiesen, dass im Entity-Relationship-Modell nur solche [Min,Max]-Intervalle der Optionalität/Kardinalität darstellbar sind, die die Kardinalität 1 einschließen. Da dies für beide Beziehungstyp-Richtungen gilt, ist das Grundelement 1 notwendiger Bestandteil aller konstruierbaren Beziehungstypen.

Geht man nun also davon aus, dass die Grundelemente zu einer beliebigen Kombination zusammengefasst werden können, solange diese Kombination das Grundelement 1 enthält, so gelangt man konstruktiv zu den 16 Klassen von Beziehungstypen, die in der Tabelle 5-1 aufgeführt sind.

Tab. 5-1: Konstruktive Herleitung der 16 Klassen von Beziehungstypen

Kombination der Grundelemente	Objekttyp A	Objekttyp B	Klasse der Beziehungstypen
1			1:1
1 + 2			1:C
1 + 3			C:1
1 + 4			1:N
1 + 5			N:1
1 + 2 + 3			C:C
1 + 2 + 4			1:CN
1 + 2 + 5			N:C
1 + 3 + 4			C:N

1 + 3 + 5		CN:1
1 + 4 + 5		M:N
1 + 2 + 3 + 4		C:CN
1 + 2 + 3 + 5		CN:C
1 + 2 + 4 + 5		M:CN
1 + 3 + 4 + 5		CM:N

1 + 2 + 3 + 4 + 5		CM:CN

Die hergeleiteten 16 Klassen von Beziehungstypen sind bereits in der Tabelle 2-1 im Abschnitt 2.4.1 in Form einer Matrix dargestellt worden.

5.2.3 Die Umwandlung in einen Koppel-Objekttyp

In den Transformationsregeln T3 – T12, die die Repräsentation der dualen Beziehungstypen im relationalen Datenbank-Modell beschreiben, wurden in einigen Fällen die Beziehungstypen zunächst in einen neu gebildeten Koppel-Objekttyp umgeformt. Im Kapitel 4 wurde für jeden dualen Beziehungstyp untersucht, wann das geschehen soll und wann nicht. Eine Begründung der jeweiligen Entscheidung wurde dem Leser allerdings bisher vorenthalten. Das soll in diesem Abschnitt nachgeholt werden. Wir ordnen zu diesem Zweck zunächst die 10 relevanten Beziehungstypen in 4 Klassen ein, wobei wieder die an der Diagonale gespiegelten Beziehungstypen unberücksichtigt bleiben. Das Ergebnis der Klassifizierung ist in der Tabelle 5-2 wiedergegeben.

4 Klassen von Beziehungstypen

Betrachten wir die 4 Klassen von Beziehungstypen im einzelnen:

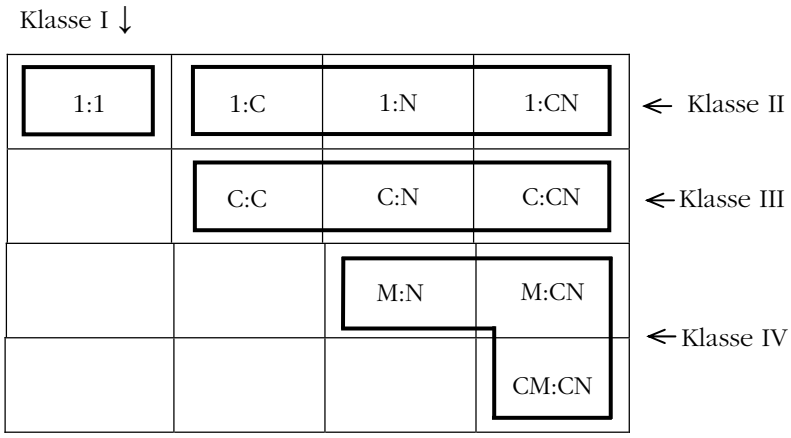
Klasse I: *Beziehungstyp mit Sender-Empfänger-Paaren.* Die durch den Beziehungstyp miteinander verbundenen Objekttypen sind wechselseitig voneinander existenzabhängig. Weder Sender noch Empfänger können ohne ihren Partner existieren.

Klasse II: *Beziehungstypen mit Empfangspflicht.* Jedes der B-Objekte muss genau eine Nachricht empfangen. Es ist also von seinem Sender existenzabhängig.

Klasse III: *Beziehungstypen ohne Empfangspflicht.* Jedes der B-Objekte kann eine – und höchstens eine - Nachricht empfangen. Es kann also auch ohne „seinen“ Sender existieren.

Klasse IV: *Beziehungstypen mit multiplen Empfängern.* Jedes A-Objekt kann mehrere Nachrichten senden, ebenso wie jedes B-Objekt mehrere Nachrichten empfangen kann.

Tab. 5-2: Klassifizierung der Beziehungstypen



Abweichende Terminologie

In einigen Lehrbüchern über Datenbanken – so beispielsweise in [ZEHN05] - werden die Klassen I und II gemeinsam als „hierarchische“ Beziehungstypen bezeichnet. Diese Benennung ist aber irreführend, weil der 1:1- und der 1:C-Beziehungstyp keine Hierarchien repräsentieren. Die Klasse III wird mitunter als die Klasse der „konditionellen“ Beziehungstypen bezeichnet. Dann versteht man aber nicht, warum beispielsweise der 1:C-Beziehungstyp nicht „konditionell“ sein soll. Auch die Bezeichnung „netzwerkförmige“ Beziehungstypen für die Klasse IV ist abzulehnen, weil der Begriff des Netzwerks nur für Beziehungen innerhalb *einer* Klasse definiert ist, also bestenfalls auf Rekursiv-Beziehungstypen anwendbar wäre.

Wir wollen nun untersuchen, ob und wann ein Beziehungstyp in einen Koppel-Objekttyp umgewandelt werden sollte. Die Umwandlung in einen Koppel-Objekttyp folgt dabei dem Schema, das in der Abbildung 5-12 dargestellt ist.

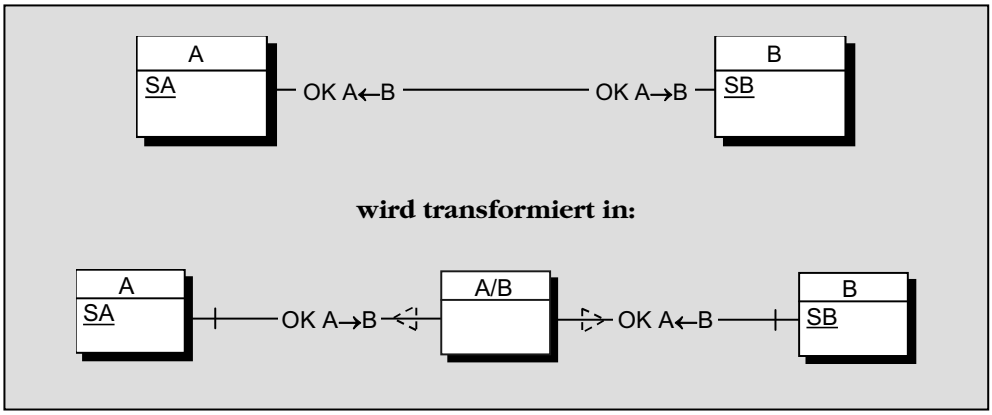


Abb. 5-12: Umwandlung eines Beziehungstyps in einen Koppel-Objekttyp

In der Abbildung 5-12 steht die Buchstaben-Folge „OK“ für „***O***ptionalität/***K***ardinalität“ in der jeweils durch den Pfeil angegebenen Beziehungstyp-Richtung. Die Nutzung der beiden Beziehungstyp-Richtungen für die Identifizierung des Koppel-Objekttyps hängt vom speziellen Beziehungstyp ab. Darum wurden sie gestrichelt dargestellt.

Untersuchen wir nun für die vier Klassen von Beziehungstypen, was ihre Umwandlung in einen Koppel-Objekttyp bewirkt.

5.2.3.1 Klasse I (Beziehungstyp mit Sender-Empfänger-Paaren)

Beim 1:1-Beziehungstyp entstehen durch die Umwandlung in einen Koppel-Objekttyp zwei 1:1-Beziehungstypen, wie dies in der Abbildung 5-13 zu sehen ist.

Gemäß der Transformationsregel T03 sind beide 1:1-Beziehungstypen nicht zu repräsentieren, sondern durch eine Zusammenfassung der zueinander gehörigen Tabellen-Zeilen in einer einzigen Tabelle darzustellen. Die Datensätze von A und von B werden in die Koppel-Tabelle A/B „hineingezogen“, wobei natürlich die als Fremdschlüssel „gedoppelten“ Primärschlüssel gestrichen werden:

Tabellenrepräsentation:

1:1
A/B(**SA**,**SB**)

Beide Schlüssel sind eingabepflichtig und unikal, einer der beiden wird als Primärschlüssel der Tabelle A/B gewählt.

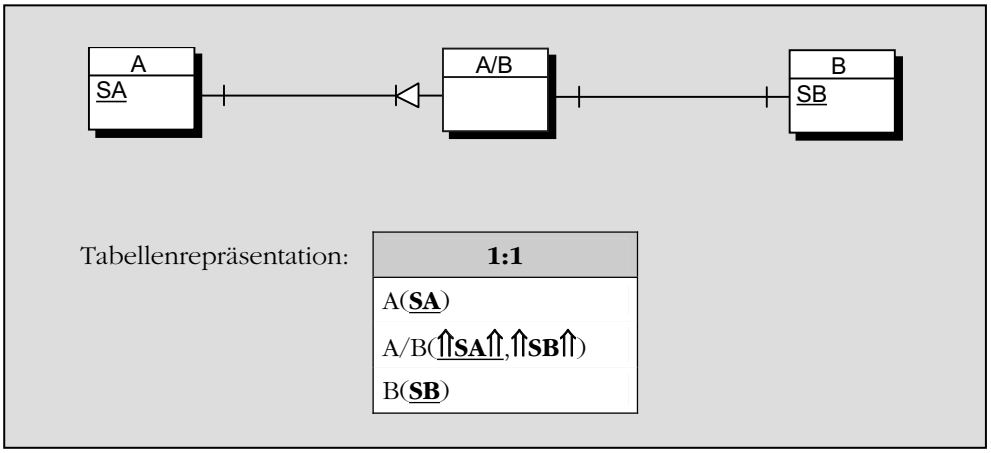


Abb. 5-13: Umwandlung eines 1:1-Beziehungstyps in einen Koppel-Objekttyp

Wechselseitige Existenzabhängigkeit

Beim 1:1-Beziehungstyp handelt es sich in der Regel um einen derart engen sachlogischen Zusammenhang mit wechselseitiger Existenzabhängigkeit, dass eine Darstellung der Kopplung zweier Objekte durch „räumliche Nähe" – also durch eine Zusammenlegung in *einen* Datensatz - auf der Hand liegt. Die mit der Transformationsregel T04 beschriebene Darstellung in zwei Tabellen sollte eher die Ausnahme bleiben.

Beispiel für Beziehungstyp der Klasse I

Als ein Beispiel nehmen wir an, dass ein Unternehmen für seine Autos Navigationssysteme anschafft, wobei jedes Auto mit genau einem solchen System ausgerüstet wird und jedes System in ein Auto eingebaut wird. Die Abbildung 5-14 zeigt die Transformation des 1:1-Beziehungstyps in einen Koppel-Objekttyp sowie die entsprechende Tabellenrepräsentation.

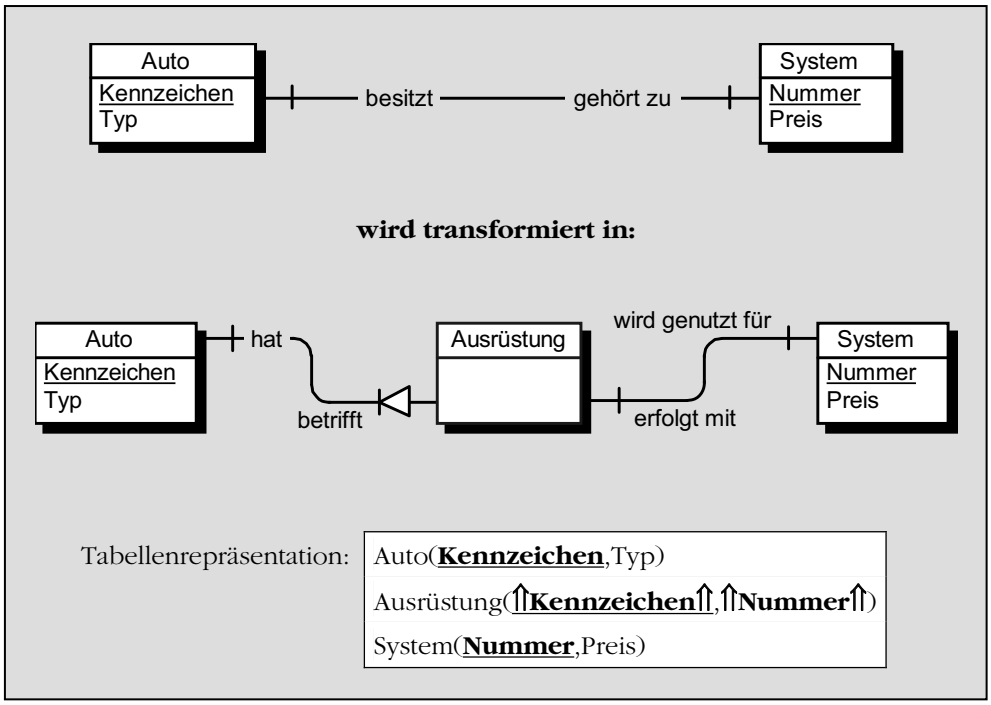


Abb. 5-14: Beispiel für die Umwandlung eines Beziehungstyps der Klasse I in einen Koppel-Objekttyp

Betrachten wir auf der Ebene der Objekte die drei Tabellen „Auto“, „Ausrüstung“ und „System“ mit Beispieldaten, wie sie in der Abbildung 5-15 angegeben sind.

Auto

Kennzeichen	Typ
B-AU 12	Audi
D-OM 23	BMW
M-UH 34	VW

Ausrüstung

Kennzeichen	Nummer
B-AU 12	S1001
D-OM 23	S1002
M-UH 34	S1003

System

Nummer	Preis
S1001	1200
S1002	1300
S1003	1400

Abb. 5-15: Beispieltabellen für einen Beziehungstyp der Klasse I

Man erkennt in der Abbildung 5-15, dass auf Grund der beiden 1:1-Beziehungstypen die Spalten „Kennzeichen“ in den Tabellen „Auto“ und „Ausrüstung“ bzw. die Spalten „Nummer“ in den Tabellen „Ausrüstung“ und „System“ identisch sind. Das bedeutet eine hohe Redundanz in der Datenspeicherung. Außerdem kann der „Brückenschlag“ vom Auto zum Navigationssystem – und umgekehrt - nur durch Vermittlung der Tabelle „Ausrüstung“ erfolgen.

Performance-Verbesserung

Zur Vermeidung von Redundanz und zur Verbesserung der Performance sollten die Tabellen „Auto“ und „System“ in die Koppel-Tabelle „Ausrüstung“ hineingezogen werden. Weil das Navigationssystem zum Bestandteil des Autos wird, sollte die Koppel-Tabelle die Bezeichnung „Auto“ erhalten. Das entspricht der vereinfachten Tabellenrepräsentation:

Tabellenrepräsentation:

Auto(**Kennzeichen**,Typ,
Systemnummer,Preis)

Dabei ist die Spaltenbezeichnung „Nummer“ für das Navigationssystem in die „sprechendere“ Bezeichnung „Systemnummer“ umgewandelt worden. Außerdem ist diese Spalte als unikal gekennzeichnet worden. Für die Beispieldaten entsteht die Tabelle der Abbildung 5-16.

Auto

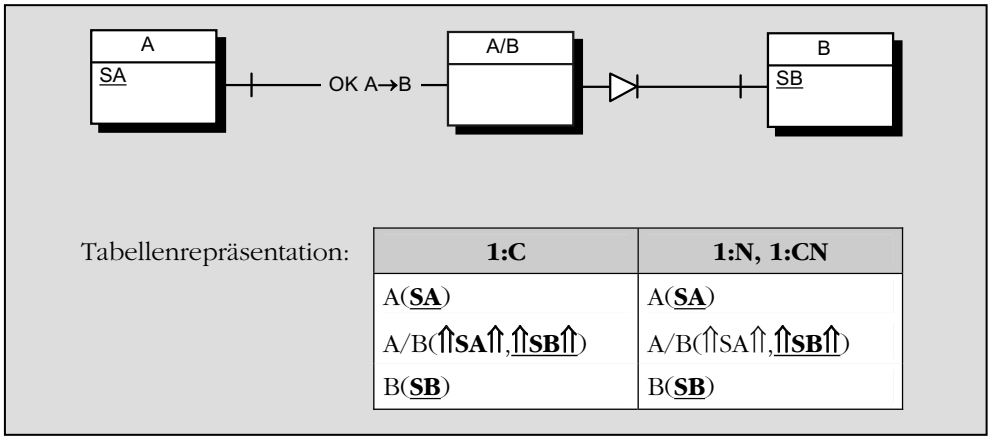
Kennzeichen	Typ	Systemnummer	Preis
B-AU 12	Audi	S1001	1200
D-OM 23	BMW	S1002	1300
M-UH 34	VW	S1003	1400

Abb. 5-16: Vereinfachte Beispieltabellen für einen Beziehungstyp der Klasse I

Fazit: **Der *Beziehungstyp der Klasse I* (Beziehungstyp mit Sender-Empfänger-Paaren - also 1:1) wird in der Regel repräsentiert, indem man beide Objekttypen zu einem gemeinsamen Objekttyp vereinigt, der in *einer Koppel-Tabelle* dargestellt wird.**

5.2.3.2 Klasse II (Beziehungstypen mit Empfangspflicht)

Bei den Beziehungstypen mit Empfangspflicht, also bei den 1:C-, 1:N- und 1:CN-Beziehungstypen, führt die Umwandlung in einen Koppel-Objekttyp zu einem 1 : (OK A→B)-Beziehungstyp und zu einem 1:1-Beziehungstyp. Dies ist in der Abbildung 5-17 dargestellt.



Tabellenrepräsentation:

1:C	1:N, 1:CN
A(**SA**)	A(**SA**)
A/B(⇑**SA**⇑,⇑**SB**⇑)	A/B(⇑SA⇑,⇑**SB**⇑)
B(**SB**)	B(**SB**)

Abb. 5-17: Umwandlung eines Beziehungstyps der Klasse II in einen Koppel-Objekttyp

Transformationsregel T03

Der 1:1-Beziehungstyp ist entsprechend der Transformationsregel T03 nicht zu repräsentieren. Stattdessen werden die Datensätze der Tabelle A/B an die Datensätze der Tabelle B angehängt, wodurch der Fremdschlüssel ⇑**SB**⇑ überflüssig wird:

Tabellenrepräsentation:

1:C	1:N, 1:CN
A(**SA**)	A(**SA**)
B(**SB**,⇑**SA**⇑)	B(**SB**,⇑SA⇑)

Integration der Koppel-Tabelle in die Objekttyp-Tabelle

Diese Umformung entspricht einer Integration der Koppel-Tabelle in die Objekttyp-Tabelle. Der Grund für diese Manipulation ist ausschließlich technischer Natur: Die Anzahl der Daten-Zugriffe wird reduziert, wie aus dem nachfolgenden Beispiel ersichtlich wird. Den Preis für diese Performance-Steigerung zahlt man allerdings mit einer „Vernebelung“ der klaren Struktur

des Datenmodells: Das Strukturelement „Beziehungstyp“ wird nun nicht mehr gesondert repräsentiert, sondern „verschwindet“ im Objekttyp.

Beispiel für Beziehungstyp der Klasse II

Betrachten wir als Beispiel für die Darstellung von Beziehungstypen der Klasse II Kunden, die eventuell noch keinen Auftrag, bereits einen oder schon mehrere Aufträge erteilt haben, wobei ein Auftrag stets von genau einem Kunden erteilt wird. In der Abbildung 5-18 sind die Transformation des Beziehungstyps in einen Koppel-Objekttyp sowie die entsprechende Tabellenrepräsentation zusammengefasst.

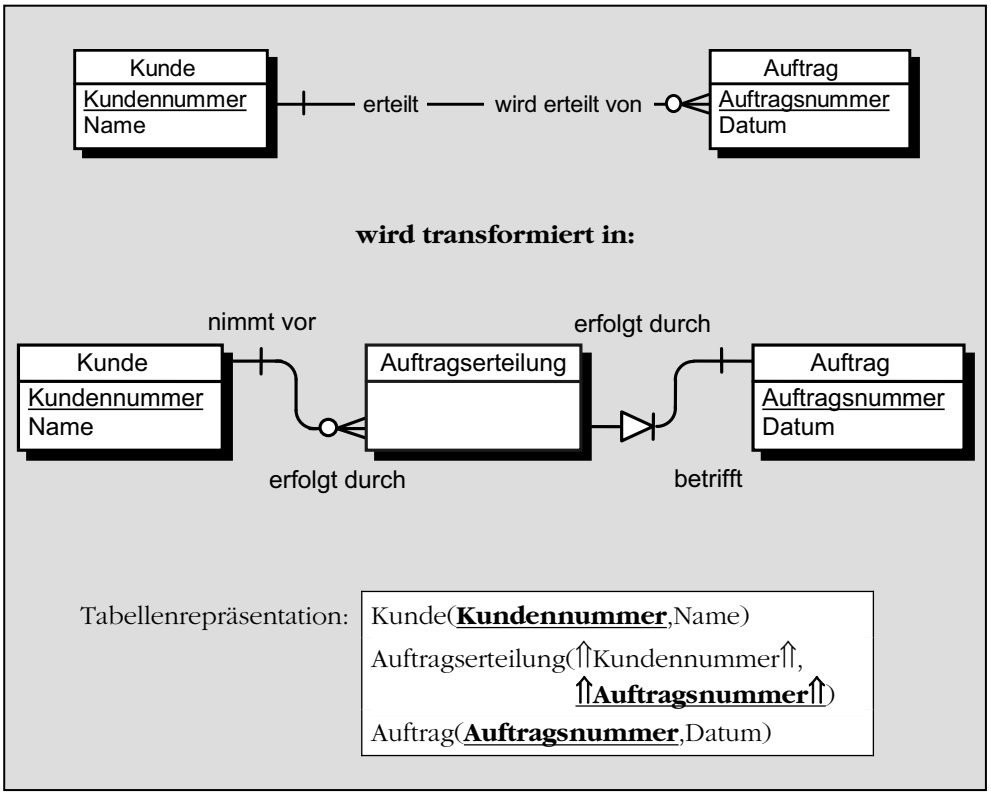


Abb. 5-18: Beispiel für die Umwandlung eines Beziehungstyps der Klasse II in einen Koppel-Objekttyp

Auf der Ebene der Objekte könnten die drei Tabellen „Kunde“, „Auftragserteilung“ und „Auftrag“ die Gestalt annehmen, die in Abbildung 5-19 wiedergegeben ist. Dabei wurde aus Platzgrün-

den das Suffix „nummer“ in den Spaltenbezeichnungen durch das Nummernzeichen „#“ dargestellt.

Kunde

Kunden#	Name
K001	Schulz
K002	Müller
K003	Meier

Auftragserteilung

Kunden#	Auftrags#
K001	A001
K001	A002
K002	A003

Auftrag

Auftrags#	Datum
A001	1.1.09
A002	2.2.09
A003	3.3.09

Abb. 5-19: Beispieltabellen für einen Beziehungstyp der Klasse II

Soll bei dieser Tabellenrepräsentation ein „Brückenschlag“ zwischen den Kunden und den Aufträgen erfolgen, sollen also beispielsweise für den Kunden „K001“ dessen Name sowie sämtliche Angaben zu seinen Aufträgen ermittelt werden, so sind drei Tabellen-Zugriffe erforderlich:

1. Lesen des Datensatzes zur Kundennummer „K001“ aus der Tabelle „Kunde“. Aus der Spalte „Name“ lässt sich nun der gesuchte Kundenname entnehmen.
2. Lesen sämtlicher Datensätze aus der Tabelle „Auftragserteilung“, in denen der Fremdschlüssel „Kunden#“ den Wert „K001“ annimmt. In diesen Datensätzen befinden sich die Verweise auf die jeweiligen Aufträge in Form der Werte des Fremdschlüssels „Auftrags#“. Als Menge der Aufträge des Kunden „K001“ wird {„A001“, „A002“} ermittelt. Wollte man nur die Identifikatoren der Aufträge ermitteln, wäre die Aufgabe hiermit erfüllt. Da aber sämtliche Angaben zu den Aufträgen – in unserem Beispiel also auch das Auftragsdatum - ermittelt werden sollen, müssen diese der Tabelle „Auftrag“ entnommen werden.
3. Lesen der beiden Datensätze aus der Tabelle „Auftrag“, für die der Primärschlüssel „Auftrags#“ den Wert „A001“ bzw. „A002“ annimmt.

Performance-Verbesserung

Im Interesse einer besseren Performance lässt sich die Zahl der Tabellen-Zugriffe verringern. Man nutzt dazu aus, dass es auf Grund des 1:1-Beziehungstyps zu jeder Zeile der Tabelle „Auftragserteilung“ genau eine Zeile der Tabelle „Auftrag“ gibt. Die Zeilen der Tabelle „Auftragserteilung“ werden an die entspre-

chenden Zeilen der Tabelle „Auftrag“ angehängt, wobei die doppelt auftretende Spalte „Auftrags#“ nur einmal aufgeführt wird. Das entspricht der reduzierten Tabellenrepräsentation:

Tabellenrepräsentation:

Kunde(**Kundennummer**,Name)
Auftrag(**Auftragsnummer**,Datum,
⇑Kundennummer⇑)

Unsere Beispieltabellen nehmen dann die reduzierte Form an, die in der Abbildung 5-20 dargestellt ist. Die Anzahl der Tabellen-Zugriffe in unserer Beispielaufgabe verringert sich nun auf 2:

1. Lesen des Datensatzes zur Kundennummer „K001“ aus der Tabelle „Kunde“, der den Kundennamen enthält.
2. Lesen sämtlicher Datensätze aus der Tabelle „Auftrag“, in denen der Fremdschlüssel „Kunden#“ den Wert „K001“ hat. In diesen Datensätzen befinden sich die gesuchten Werte für das jeweilige Auftragsdatum.

Kunde

Kunden#	Name
K001	Schulz
K002	Müller
K003	Meier

Auftrag

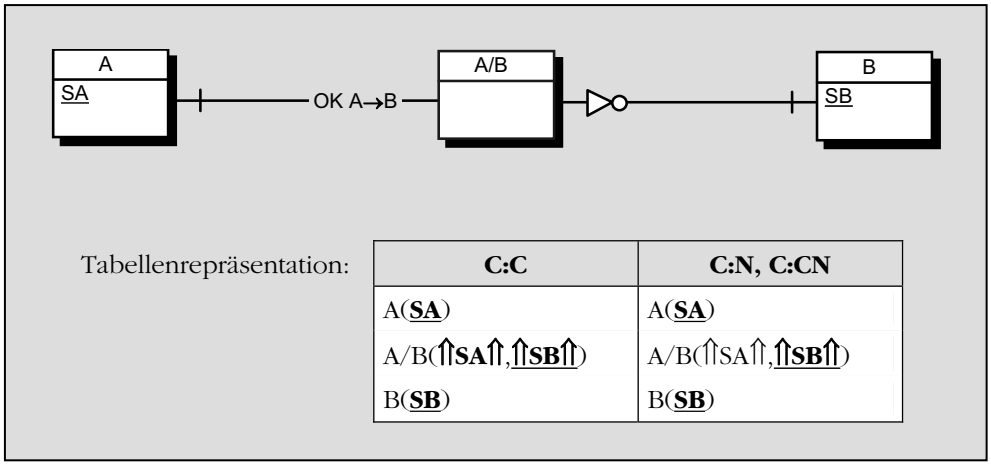
Auftrags#	Datum	Kunden#
A001	1.1.09	K001
A002	2.2.09	K001
A003	3.3.09	K002

Abb. 5-20: Reduzierte Beispieltabellen für einen Beziehungstyp der Klasse II

Fazit: **Die *Beziehungstypen der Klasse II* (Beziehungstypen mit Empfangspflicht, also 1:C, 1:N und 1:CN) werden aus Performance-Gründen *nicht als Koppel-Tabelle* repräsentiert. Stattdessen wird in der Empfänger-Tabelle auf den Sender verwiesen.**

5.2.3.3 Klasse III (Beziehungstypen ohne Empfangspflicht)

Auch die Beziehungstypen ohne Empfangspflicht - also die C:C-, C:N- und C:CN-Beziehungstypen - lassen sich gemäß der Abbildung 5-21 in einen Koppel-Objekttyp umwandeln. Dies führt dann einerseits zu einem 1:(OK A→B)-Beziehungstyp und andererseits zu einem C:1-Beziehungstyp.



Tabellenrepräsentation:

C:C	C:N, C:CN
A(**SA**)	A(**SA**)
A/B(⇑**SA**⇑,⇑**SB**⇑)	A/B(⇑SA⇑,⇑**SB**⇑)
B(**SB**)	B(**SB**)

Abb. 5-21: Umwandlung eines Beziehungstyps der Klasse III in einen Koppel-Objekttyp

Transformationsregel T05

Der C:1-Beziehungstyp *kann* entsprechend der Transformationsregel T05 eliminiert werden, indem der Koppel-Objekttyp A/B in den Objekttyp B integriert wird. Das ist insbesondere dann sinnvoll, wenn die Optionalität der Beziehungstyp-Richtung „B zu A/B“ selten realisiert wird, wenn also der C:1-Beziehungstyp fast zum 1:1-Beziehungstyp „entartet“:

Tabellenrepräsentation:

C:C	C:N, C:CN
A(**SA**)	A(**SA**)
B(**SB**, ⇑***SA***⇑)	B(**SB**, ⇑*SA*⇑)

Der aus der Tabelle A/B in die Tabelle B überführte Fremdschlüssel ⇑*SA*⇑ ist in B als nicht-eingabepflichtig zu deklarieren, weil es B-Objekte geben kann, die auf kein A-Objekt verweisen.

Performance-Verbesserung

Die beschriebene Manipulation führt zu einer Verbesserung der Performance, wenn häufig zwischen den Objekttypen A und B navigiert werden soll. Allerdings kann sich der Speicherplatz-Bedarf erhöhen, wenn der Fremdschlüssel $\Uparrow SA \Uparrow$ häufig mit der NULL-Marke belegt ist.

Wir haben es hier also mit dem typischen Zielkonflikt zu tun, den man in Anlehnung an die „goldene Regel der Mechanik“ (Was man an Kraft einspart, muss man an Weg zusetzen) als die „goldene Regel der Datenspeicherung“ bezeichnen könnte:

Goldene Regel der Datenspeicherung

Goldene Regel der Datenspeicherung:

Was man an Rechenzeit einspart, muss man an Speicherplatz zusetzen!

Betrachtet man den Speicherplatz-Bedarf zur Darstellung eines Beziehungstyps der Klasse III, dann ergeben sich für die beiden Repräsentationsformen die folgenden Ausdrücke:

a) Repräsentation ***mit*** Koppel-Tabelle:	b) Repräsentation ***ohne*** Koppel-Tabelle:
$\overline{\overline{\mathrm{A/B}}} \cdot \left(Len(\underline{\mathrm{SA}}) + Len(\underline{\mathrm{SB}}) \right)$	$\overline{\mathrm{B}} \cdot Len(\underline{\mathrm{SA}})$

Der Speicherplatz-Bedarf im Fall a) ergibt sich daraus, dass in jeder Zeile der Koppel-Tabelle A/B je ein Wert der Fremdschlüssel $\Uparrow$SA$\Uparrow$ und $\Uparrow$SB$\Uparrow$ gespeichert werden muss. Im Fall b) wird in jeder Zeile der Tabelle B der Speicherplatz für einen Wert des Fremdschlüssels $\Uparrow$SA$\Uparrow$ bereitgestellt, auch dann, wenn dieser Fremdschlüssel mit der NULL-Marke belegt ist. Setzt man beide Ausdrücke gleich, so benötigt man dann denselben Speicherplatz, wenn gilt:

$$\frac{\overline{\overline{\mathrm{A/B}}}}{\overline{\mathrm{B}}} = \frac{Len(\underline{\mathrm{SA}})}{Len(\underline{\mathrm{SA}}) + Len(\underline{\mathrm{SB}})}$$

Lässt man den Performance-Aspekt außer Acht und betrachtet die beiden Repräsentationsformen ausschließlich im Hinblick auf die Minimierung des Speicherplatz-Bedarfs, dann kommt man für einige ausgewählte Situationen zu den Entscheidungen, die in der Tabelle 5-3 dargestellt sind.

Tab. 5-3: Entscheidung für oder gegen eine Koppel-Tabelle bei der Repräsentation eines Beziehungstyps der Klasse III

Situation	Bedeutung	Gleicher Speicherplatz-Bedarf bei	Repräsentation mit Koppel-Tabelle
$\overline{\overline{\mathrm{A/B}}} = \overline{\mathrm{B}}$	Jedes B-Objekt hat einen A-Partner	$Len(\underline{\mathrm{SB}}) = 0$	nie
$\overline{\overline{\mathrm{A/B}}} = \frac{1}{2}\overline{\mathrm{B}}$	Jedes zweite B-Objekt hat einen A-Partner	$Len(\underline{\mathrm{SA}}) = Len(\underline{\mathrm{SB}})$	wenn $Len(\underline{\mathrm{SA}}) > Len(\underline{\mathrm{SB}})$
$\overline{\overline{\mathrm{A/B}}} = \frac{1}{4}\overline{\mathrm{B}}$	Jedes vierte B-Objekt hat einen A-Partner	$Len(\underline{\mathrm{SA}}) = \frac{1}{3} Len(\underline{\mathrm{SB}})$	wenn $Len(\underline{\mathrm{SA}}) > \frac{1}{3} Len(\underline{\mathrm{SB}})$

Bei zusätzlicher Beachtung des Performance-Verlustes im Falle der Repräsentation durch eine Koppel-Tabelle kann die Entscheidung natürlich anders ausfallen.

Beispiel für Beziehungstyp der Klasse III

Als ein Beispiel für einen Beziehungstyp der Klasse III betrachten wir Zimmer in einem Krankenhaus, in denen Patienten untergebracht sind. Ein Patientenzimmer kann keinen Patienten (weil es renoviert wird), einen oder auch mehrere Patienten beherbergen. Ein Patient ist in keinem Patientenzimmer untergebracht, wenn er ambulant behandelt wird, ansonsten liegt er stationär in einem Patientenzimmer. Die Abbildung 5-22 zeigt die Transformation des Beziehungstyps in einen Koppel-Objekttyp sowie die entsprechende Tabellenrepräsentation.

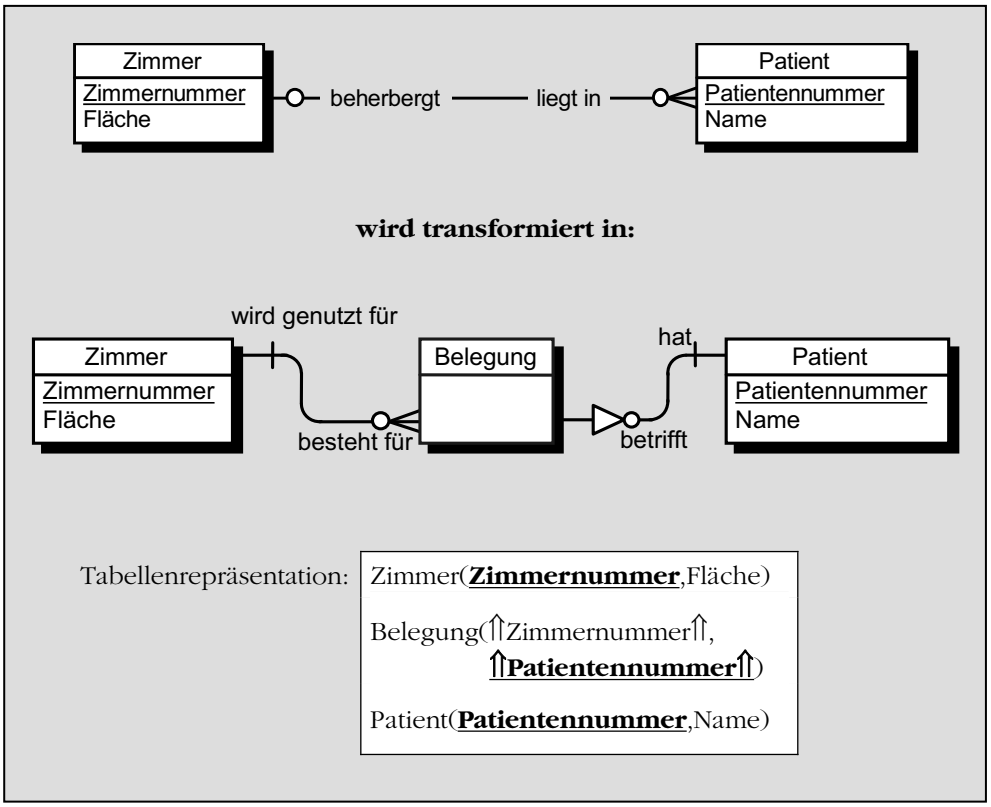


Abb. 5-22: Beispiel für die Umwandlung eines Beziehungstyps der Klasse III in einen Koppel-Objekttyp

Beispieldaten für die drei Tabellen „Zimmer“, „Belegung“ und „Patient“ zeigt die Abbildung 5-23, in der wiederum das Suffix „nummer“ in den Spaltenbezeichnungen zum Nummernzeichen „#“ verkürzt wurde.

Man entnimmt der Abbildung 5-23, dass es ein nichtbelegtes Zimmer („503“) gibt und dass drei Patienten („012345“, „123456“ und „456789“) – also nur die Hälfte aller Patienten – einem Patientenzimmer zugeordnet sind. Es gilt somit:

$$Len(\underline{\text{Zimmer\#}}) = 3$$

$$Len(\underline{\text{Patienten\#}}) = 6$$

$$\overline{\overline{\text{Belegung}}} = \frac{1}{2}\overline{\overline{\text{Patient}}}$$

Zimmer

Zimmer#	Fläche
501	20
502	17
503	25

Belegung

Zimmer#	Patienten#
501	012345
501	123456
502	456789

Patient

Patienten#	Name
012345	Meier
123456	Schulz
234567	Lehmann
345678	Müller
456789	Fischer
567890	Becker

Abb. 5-23: Beispieltabellen für einen Beziehungstyp der Klasse III mit Koppel-Tabelle

Da $Len(\underline{Zimmer\#}) < Len(\underline{Patienten\#})$, benötigt gemäß Tabelle 5-3 die Repräsentation des Beziehungstyps ohne Koppel-Tabelle weniger Speicherplatz als die Repräsentation mit Koppel-Tabelle. Die Beispiel-Tabellen nehmen somit die Gestalt an, die in der Abbildung 5-24 wiedergegeben ist.

Zimmer

Zimmer#	Fläche
501	20
502	17
503	25

Patient

Patienten#	Name	Zimmer#
012345	Meier	501
123456	Schulz	501
234567	Lehmann	NULL
345678	Müller	NULL
456789	Fischer	502
567890	Becker	NULL

Abb. 5-24: Beispieltabellen für einen Beziehungstyp der Klasse III ohne Koppel-Tabelle

Fazit: **Die *Beziehungstypen der Klasse III* (die Beziehungstypen ohne Empfangspflicht, also C:C, C:N und C:CN) können *sowohl mit als auch ohne Koppel-Tabelle* repräsentiert werden. Die Entscheidung muss unter dem Zielkonflikt „entweder weniger Speicherplatz-Bedarf oder geringere Rechenzeit“ gefällt werden.**

5.2.3.4 Klasse IV (Beziehungstypen mit multiplen Empfängern)

Bei der Umwandlung der Beziehungstypen mit multiplen Empfängern, also der M:N-, M:CN- und CM:CN-Beziehungstypen, in einen Koppel-Objekttyp entstehen ein 1:(C)N-Beziehungstyp und ein (C)N:1-Beziehungstyp. Die Abbildung 5-25 zeigt das Ergebnis, wobei die eventuelle Optionalität der Beziehungstyp-Richtungen „A zu A/B“ bzw. „B zu A/B“ jeweils durch den punktierten Kreis symbolisiert wird.

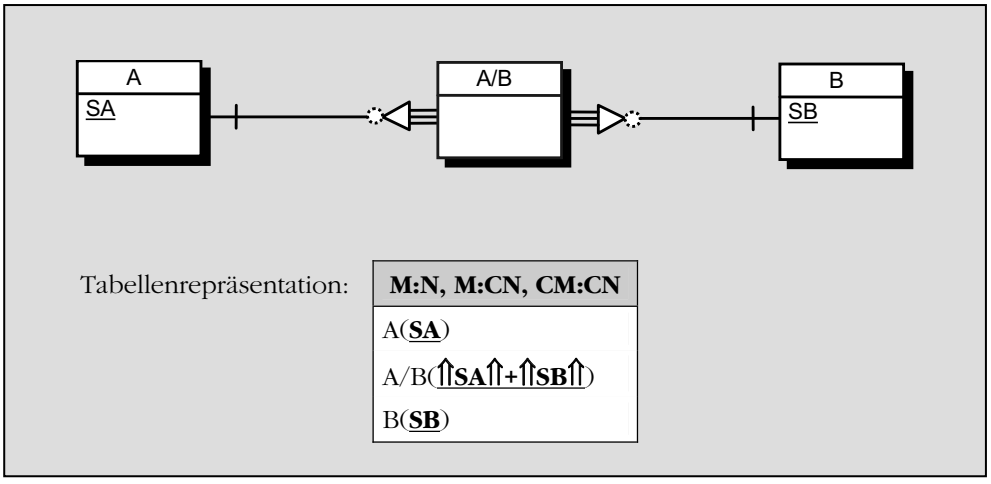


Abb. 5-25: Umwandlung eines Beziehungstyps der Klasse IV in einen Koppel-Objekttyp

Die Fremdschlüssel ⇑SA⇑ und ⇑SB⇑ sind beide eingabepflichtig, jeder für sich ist nicht-unikal. Erst in der Kopplung **⇑SA⇑+⇑SB⇑** bilden sie den unikalen Primärschlüssel der Koppel-Tabelle A/B.

Keine Vereinfachung möglich

Eine Vereinfachung der Tabellenstruktur ist nicht möglich. Bei einer Vereinigung der Tabellen A und A/B müsste ein A-Objekt gegebenenfalls auf *mehrere* B-Objekte verweisen. Analog dazu müsste bei einer Zusammenführung der Tabellen B und A/B ein B-Objekt Verweise auf *mehrere* A-Objekte enthalten können. Beides würde aber der 1. Normalform widersprechen.

Beispiel für einen Beziehungstyp der Klasse IV

Betrachten wir als Beispiel für einen Beziehungstyp der Klasse IV Busfahrer, die Reisegruppen befördern. Ein (neueingestellter) Busfahrer kann noch keine Reisegruppe befördert haben, andere Busfahrer haben schon eine oder mehrere Reisegruppen befördert. Eine Reisegruppe wird von mindestens einem Busfahrer befördert. Bei langen Reisen sind es mehrere Busfahrer, die sich abwechseln. Die Abbildung 5-26 zeigt die Transformation des M:CN-Beziehungstyps in einen Koppel-Objekttyp sowie die entsprechende Tabellenrepräsentation.

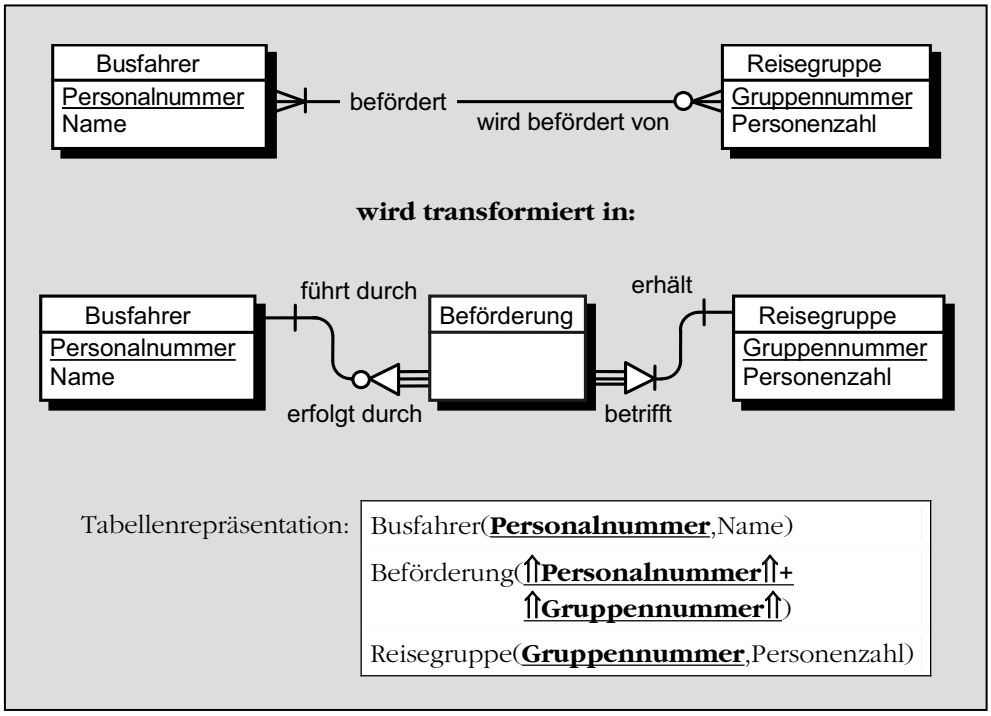


Abb. 5-26: Beispiel für die Umwandlung eines Beziehungstyps der Klasse IV in einen Koppel-Objekttyp

Die Abbildung 5-27 enthält Beispieldaten für die drei Tabellen „Busfahrer“, „Beförderung“ und „Reisegruppe“, wobei wiederum das Suffix „nummer“ in den Spaltenbezeichnungen zum Nummernzeichen „#“ verkürzt wurde.

Busfahrer

Personal#	Name
P1001	Anna Konda
P2002	Tom Bola
P3003	Wilma Keiner
P4004	Lore Ley

Beförderung

Personal#	Gruppen#
P1001	G001
P2002	G002
P2002	G003
P3003	G003

Reisegruppe

Gruppen#	Personenzahl
G001	24
G002	85
G003	40

Abb. 5-27: Beispieltabellen für einen Beziehungstyp der Klasse IV

Die Beispieldaten in der Abbildung 5-27 zeigen, dass der Busfahrer „P4004“ noch keine Beförderung vorgenommen hat. Die Busfahrer „P1001“ und „P3003“ haben jeweils erst eine, der Busfahrer „P2002“ hat dagegen schon zwei Reisegruppen befördert. Die Reisegruppen „G001“ und „G002“ hatten nur einen, die Reisegruppe „G003“ hatte dagegen zwei Busfahrer.

Fazit: **Die *Beziehungstypen der Klasse IV* (die Beziehungstypen mit multiplen Empfängern, also M:N, M:CN und CM:CN) können *nur mit einer Koppel-Tabelle* repräsentiert werden.**

5.2.3.5 Obligatorische, fakultative und reduzible Beziehungstypen

In den vorangegangenen Abschnitten wurden für die vier Klassen von dualen Beziehungstypen die Möglichkeiten ihrer Repräsentation mit bzw. ohne Koppel-Tabelle untersucht. Bei der Repräsentation eines Beziehungstyps mit Hilfe einer Koppel-Tabelle

wurde der jeweilige Beziehungstyp auf einfachere Beziehungstypen zurückgeführt.

Auf Grund dieser Betrachtungen können wir nun die Beziehungstypen in drei Gruppen einteilen:

1. *Obligatorische Beziehungstypen:* Dies sind diejenigen Beziehungstypen, die unbedingt erforderlich sind, weil sie sich nicht auf andere Beziehungstypen zurückführen lassen.
2. *Fakultative Beziehungstypen:* Dies sind Beziehungstypen, die nicht unbedingt erforderlich sind, da sie sich auf obligatorische Beziehungstypen zurückführen lassen. Sie sind jedoch nützlich, da sie gegebenenfalls eine Verringerung des Speicherplatz-Bedarfs oder eine Verbesserung der Performance herbeiführen können.
3. *Reduzible Beziehungstypen:* Dies sind Beziehungstypen, die sich nicht ohne Verletzung der 1. Normalform in das relationale Datenbank-Modell transformieren lassen. Sie müssen durch Umwandlung in einen Koppel-Objekttyp auf obligatorische Beziehungstypen zurückgeführt werden.

Die Abbildung 5-28 zeigt diese Gruppierung der Beziehungstypen, wobei durch einen gestrichelten Pfeil die *fallweise* und durch einen ausgezogenen Pfeil die *unbedingte* Zurückführung eines Beziehungstyps auf die obligatorischen Beziehungstypen dargestellt wird.

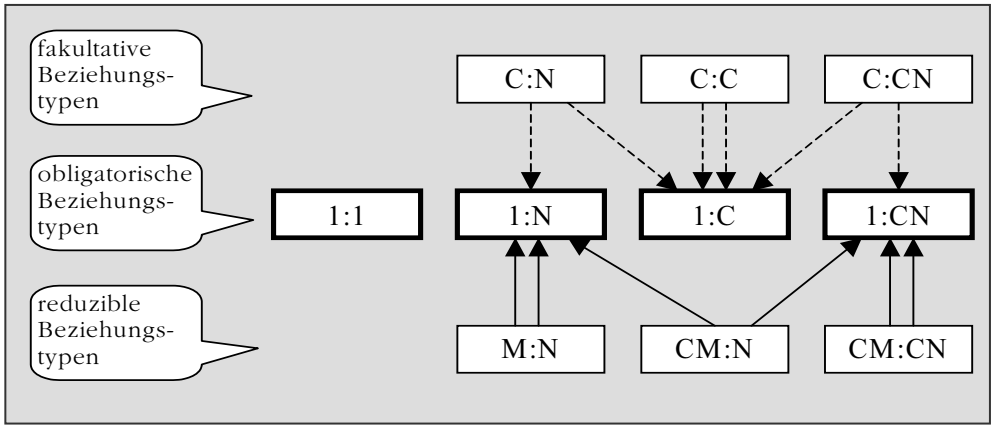


Abb. 5-28: Fakultative, obligatorische und reduzible Beziehungstypen

In der Abbildung 5-28 wurden die vier obligatorischen Beziehungstypen (1:1, 1:N, 1:C und 1:CN) stark umrandet. Mit ihrer Darstellbarkeit im jeweiligen Datenbank-Modell steht und fällt die Vollständigkeit der Repräsentation der sachlogischen Zusammenhänge im betrachteten Gegenstandsbereich der Realität. Im folgenden Abschnitt wird darum untersucht, wie es um die Darstellbarkeit dieser Beziehungstypen im relationalen Datenbank-Modell bestellt ist.

5.2.4 Die Repräsentationsmöglichkeit im relationalen Datenbank-Modell

Das Grundprinzip zur Darstellung von Beziehungstypen im relationalen Datenbank-Modell besteht - wie im Kapitel 3 ausführlich beschrieben wurde - darin, dass der *Primärschlüssel* der Sender-Tabelle in die Empfänger-Tabelle als *Fremdschlüssel* aufgenommen wird. Der Sender wird vom Empfänger *referiert*, indem der Primärschlüsselwert der Sender-Zeile als Wert des Fremdschlüssels in der Empfänger-Zeile gespeichert wird. Dieses Referenzprinzip ist in der Abbildung 5-29 noch einmal dargestellt.

Referenzprinzip

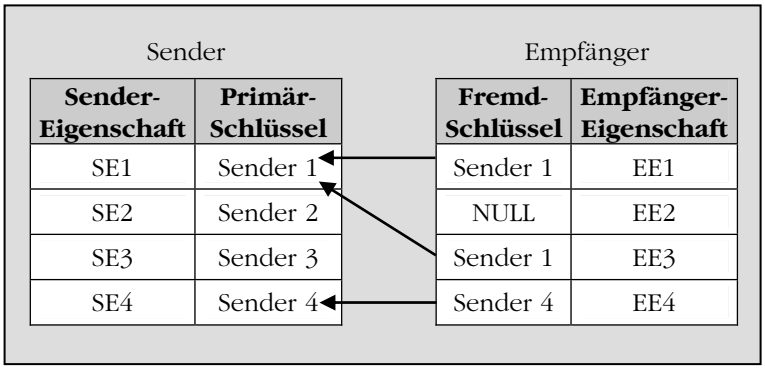


Abb. 5-29: Referenzprinzip zur Darstellung von Beziehungstypen

Unikalität und referenzielle Integrität

Für den Primärschlüssel und für den Fremdschlüssel gilt je eine Bedingung, die *stets* erfüllt sein muss: Für den Primärschlüssel ist es die Pflicht zur *Unikalität*, für den Fremdschlüssel ist es die Pflicht zur *referenziellen Integrität*.

Eingabepflicht und Unikalität

Für den Datenbankentwurf wichtiger sind dagegen die Bedingungen, die *wahlweise* gefordert werden können. Zwei derartige Bedingungen für den Fremdschlüssel haben wir bereits im Abschnitt 3.4 kennengelernt: die *Eingabepflicht* und die *Pflicht zur Unikalität*.

Wir wollen an dieser Stelle eine wählbare bzw. abwählbare Bedingung für den Fremdschlüssel hinzufügen, die wir als *Referenzpflicht* bezeichnen:

Referenzpflicht

Definition: Besteht für einen Fremdschlüssel *Referenzpflicht*, so bedeutet dies, dass jeder Wert des Primärschlüssels mindestens einmal als Wert des korrespondierenden Fremdschlüssels auftreten muss.

Gegen die Referenzpflicht wird verstoßen, wenn es in der Sender-Tabelle einen Wert des Primärschlüssels gibt, der nicht als Wert des Fremdschlüssels in der Empfänger-Tabelle auftritt.

Die Referenzpflicht lässt sich im relationalen Datenbank-Modell nicht auf der Ebene der Tabellen-Typbeschreibungen einfordern: Dafür stellen relationale Datenbank-Managementsysteme keine Mechanismen bereit. Sie kann lediglich durch die Anwendungs-Software durchgesetzt werden. Dies hat einen simplen Grund: die Referenzpflicht ist ohne das Transaktionskonzept nicht durchsetzbar.

Beispiel für Referenzpflicht

Betrachten wir dazu das folgende Beispiel! Ein Krankenhaus hat mindestens einen Operationssaal, der natürlich in genau einem Krankenhaus liegt. Die Repräsentation dieses Sachverhalts im Datenmodell und durch die Tabellen-Typbeschreibungen zeigt die Abbildung 5-30.

Der Fremdschlüssel ⇑Name⇑ ist in der Typbeschreibung der Tabelle „OP-Saal“ als *eingabepflichtig* (jeder OP-Saal muss zu einem Krankenhaus gehören) und als *nicht-unikal* (mehrere OP-Säle können zum selben Krankenhaus gehören) vereinbart. Eigentlich müsste er auch als *referenzpflichtig* deklariert werden, denn zu jedem Krankenhaus-Datensatz muss es in der Tabelle „OP-Saal“ wenigstens einen Datensatz geben, dessen Fremdschlüsselwert einen Verweis auf diesen Krankenhaus-Datensatz darstellt.

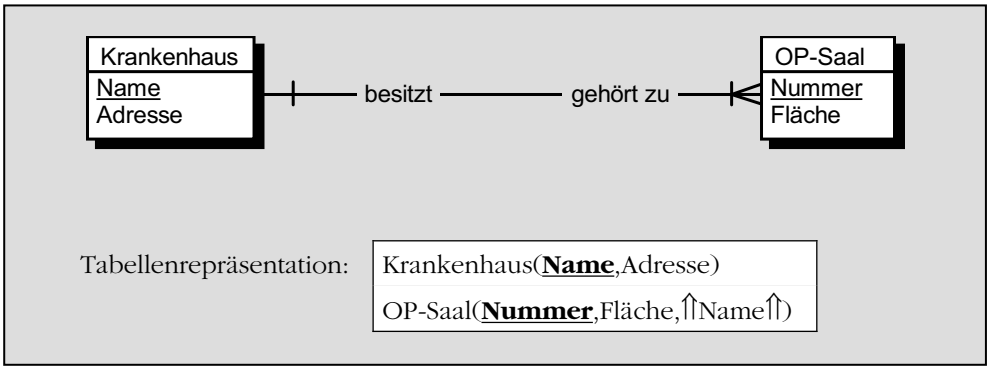


Abb. 5-30: Beispiel für die Referenzpflicht eines Fremdschlüssels

Probleme mit der Referenzpflicht

Warum ist aber die Referenzpflicht nur mit Hilfe von Transaktionen durchsetzbar? Nehmen wir an, dass die beiden Tabellen „Krankenhaus“ und „OP-Saal“ noch leer sind. Wir wollen nun das erste Krankenhaus mit dem Namen „Südklinik“ speichern. Wenn wir den Datensatz für das Krankenhaus „Südklinik“ in die Tabelle „Krankenhaus“ aufnehmen, haben wir schon gegen die Referenzpflicht verstoßen, denn es gibt ja noch keinen Datensatz in der Tabelle „OP-Saal“, der auf die „Südklinik“ verweisen könnte. Das Speichern eines neuen Krankenhaus-Datensatzes mit einem Primärschlüsselwert „K“ ist also nur mit Hilfe einer Transaktion realisierbar, die zugleich auch mindestens einen OP-Saal-Datensatz mit dem Fremdschlüsselwert „K“ speichert.

Probleme ergeben sich auch beim Löschen eines OP-Saal-Datensatzes. Wenn dies nämlich der einzige Datensatz war, der auf ein Krankenhaus „K“ verwiesen hat, wäre nach dem Löschen die Referenzpflicht verletzt. Mit dem OP-Saal-Datensatz müsste also in einer Transaktion auch das zugehörige Krankenhaus gelöscht werden.

Untersuchen wir nun die Klassen von Beziehungstypen, die sich mit Hilfe des Referenzprinzips der Abbildung 5-29 repräsentieren lassen! Zunächst muss sich die Empfänger-Tabelle, die den Fremdschlüssel enthält, stets in der 1. Normalform befinden. Somit kann der Fremdschlüssel nur einen atomaren Wert annehmen, d.h. er kann *höchstens auf einen* Sender verweisen. Die Optionalität/Kardinalität der Beziehungstyp-Richtung „Sender←Empfänger“ kann also nur „1“ oder „C“ sein.

Repräsentierbarkeit der Beziehungstypen

Welche Auswirkungen hat aber das Fordern bzw. Nichtfordern von Referenzpflicht, Eingabepflicht und Unikalität des Fremdschlüssels auf den Beziehungstyp? Die folgende Aufstellung beantwortet diese Frage:

Referenzpflicht: Wenn sie gefordert wird, muss die Beziehungstyp-Richtung „Sender→Empfänger“ nicht-optional sein, ansonsten optional.

Eingabepflicht: Wenn sie gefordert wird, muss die Beziehungstyp-Richtung „Sender←Empfänger“ nicht-optional (also „1“) sein, ansonsten optional (also „C“).

Unikalität: Wenn sie gefordert wird, muss die Beziehungstyp-Richtung „Sender→Empfänger“ die Kardinalität „1“ haben, ansonsten „N“.

Diese Überlegungen führen zu der kombinatorischen Betrachtung der Tabelle 5-4. Alle Kombinationen, in denen die Referenzpflicht gefordert ist, sind im relationalen Datenbank-Modell nicht repräsentierbar.

Tab. 5-4: Kombinationen von Referenz-, Eingabepflicht und Unikalität für duale Beziehungstypen

Kombinatorische Betrachtung

Fall	Referenzpflicht	Eingabepflicht	Unikalität	Beziehungstyp	repräsentierbar
1	nein	nein	nein	C:CN	ja
2	nein	nein	ja	C:C	
3	nein	ja	nein	1:CN	
4	nein	ja	ja	1:C	
5	ja	nein	nein	C:N	nein
6	ja	nein	ja	C:1	
7	ja	ja	nein	1:N	
8	ja	ja	ja	1:1	

In der Tabelle 5-4 sind die drei *Beziehungstypen mit multiplen Empfängern* (M:N, M:CN und CM:CN) nicht enthalten, weil sie mit dem einfachen Referenzprinzip der Abbildung 5-29 nicht repräsentierbar sind.

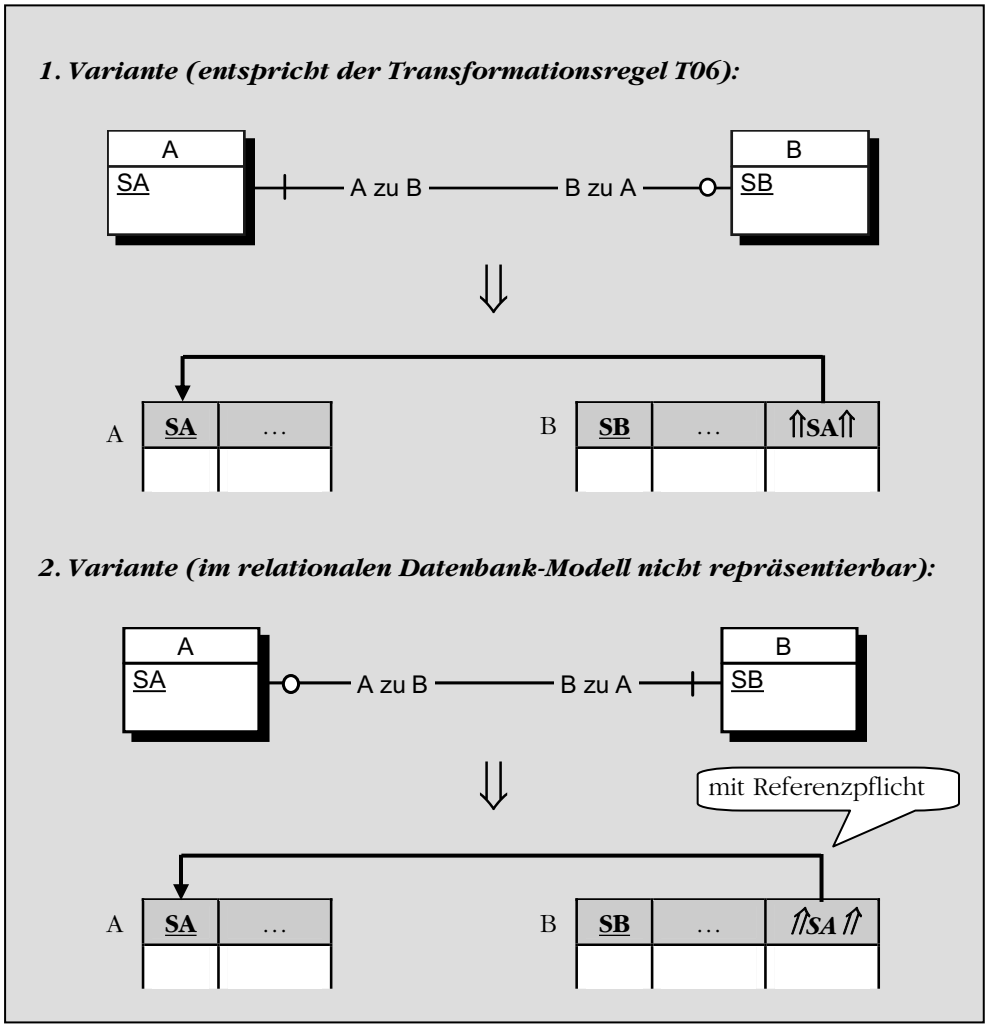


Abb. 5-31: Zwei Darstellungsvarianten für den 1:C-Beziehungstyp

Der *1:C-Beziehungstyp* kommt gemeinsam mit seinem „Spiegelbild“ - dem *C:1-Beziehungstyp* - in der Tabelle vor. Das weist auf

zwei Darstellungsvarianten für diesen Beziehungstyp hin: Die erste (Fall 4) entspricht der Transformationsregel T06, die zweite (Fall 6) ist im relationalen Datenbank-Modell nicht repräsentierbar. Die Abbildung 5-31 zeigt die beiden Darstellungsvarianten.

Überschneidung der Klassifizierungen

Im Abschnitt 5.2.3.5 hatten wir die Beziehungstypen in *obligatorische, fakultative* und *reduzible* Beziehungstypen unterschieden. Nun haben wir eine Unterscheidung in *repräsentierbare* und *nicht-repräsentierbare* Beziehungstypen getroffen. Diese beiden Klassifizierungen überschneiden sich, wie dies in der Abbildung 5-32 dargestellt ist. Die Menge der obligatorischen Beziehungstypen wird von einer Ellipse umrandet, und das Gebiet der repräsentierbaren Beziehungstypen ist als dunkler Kreis dargestellt.

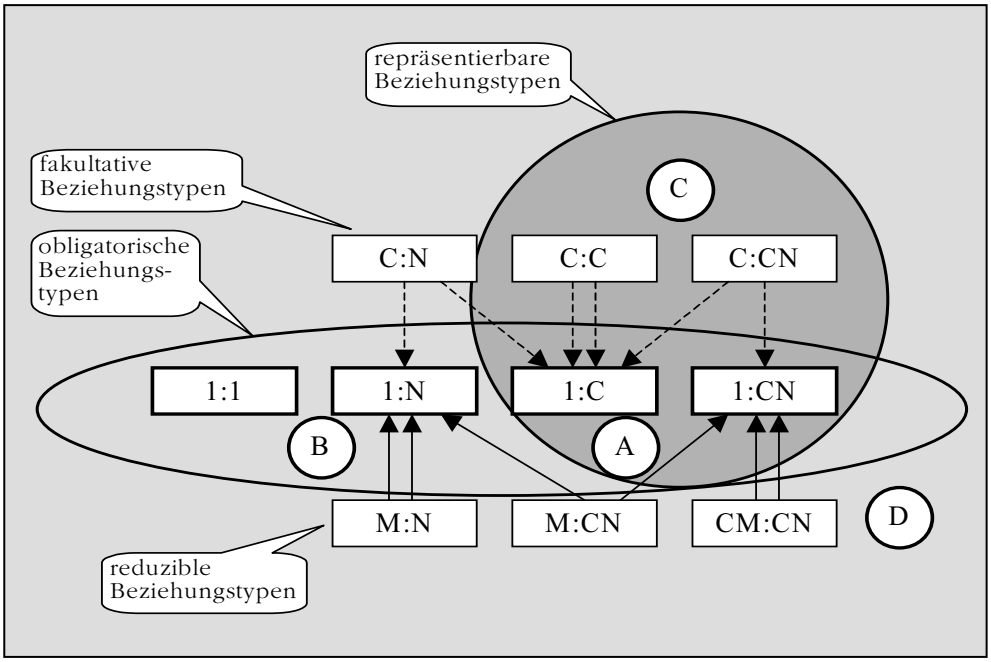


Abb. 5-32: Bereichsbildung für die dualen Beziehungstypen

Die Mengen der obligatorischen und der repräsentierbaren Beziehungstypen überlappen sich und bilden vier Bereiche A, B, C und D, die wir nacheinander betrachten wollen.

Bereich A: Obligatorische und repräsentierbare Beziehungstypen (1:C und 1:CN). Diese Beziehungstypen werden unbedingt benötigt und lassen sich auch im relationalen Datenbank-Modell darstellen.

Bereich B: Obligatorische, aber nicht repräsentierbare Beziehungstypen (1:1 und 1:N). Dass der 1:1-Beziehungstyp durch das Referenzprinzip nicht repräsentierbar ist, bereitet in der Praxis keine Probleme, weil Objekttypen, die durch einen 1:1-Beziehungstyp miteinander verbunden sind, gemäß der Transformationsregel T03 in einer gemeinsamen Tabelle gespeichert werden können: Das Referenzprinzip wird durch das „Prinzip der räumlichen Nähe“ ersetzt.

Die Nicht-Repräsentierbarkeit des 1:N-Beziehungstyps ist dagegen eine ernst zu nehmende Schwachstelle des relationalen Datenbank-Modells. Diese Form des sachlogischen Zusammenhangs zwischen Objekttypen ist in der Praxis ein häufig auftretender Fall. Beispielsweise gilt:

- Einer Bustour ist *mindestens ein* Fahrer zugeordnet.
- Ein Zug hat *mindestens einen* Wagen.
- Eine Bestellung besitzt *mindestens eine* Bestellposition.

Jegliche Form der Zerlegung eines „Ganzen“ in seine „Teile“ muss zu mindestens einem Teil führen. Diese Forderung lässt sich aber im relationalen Datenbank-Modell nicht auf der Ebene der Tabellen-Typbeschreibungen formulieren, sondern kann nur durch die Anwendungs-Software durchgesetzt werden.

Bereich C: Fakultative, aber repräsentierbare Beziehungstypen (C:C und C:CN). Diese Beziehungstypen werden nicht unbedingt benötigt, weil sie sich auf obligatorische Beziehungstypen reduzieren lassen. Es ist aber von Vorteil, dass sie sich dennoch im relationalen Datenbank-Modell repräsentieren lassen, weil sie gemäß unseren Ausführungen im Abschnitt 5.2.3 gegebenenfalls zu einer Reduzierung des Speicher-

platzbedarfs und/oder zu einer Verbesserung der Performance führen.

Bereich D: Nicht-obligatorische und nicht-repräsentierbare Beziehungstypen (C:N, M:N, M:CN und CM:CN). Diese Beziehungstypen werden eigentlich nicht benötigt, weil sie auf die obligatorischen Beziehungstypen zurückführbar sind. Da jedoch der obligatorische 1:N-Beziehungstyp im Bereich B liegt und damit nicht repräsentierbar ist, entsteht auch hier ein gewichtiger Schwachpunkt des relationalen Datenbank-Modells. Nur der CM:CN-Beziehungstyp lässt sich „sinn-erhaltend" repräsentieren, bei den anderen drei Beziehungstypen ist ein Semantikverlust in Kauf zu nehmen. Die Tabelle 5-5 zeigt das im Einzelnen.

Tab. 5-5: Repräsentierbarkeit der Beziehungstypen des Bereichs D

Beziehungstyp	Im relationalen Datenbank-Modell repräsentierbar?	Repräsentation mit Semantikverlust
C:N	1:N + C:1 nicht möglich, da 1:N fehlt	C:CN
M:N	1:N + N:1 nicht möglich, da 1:N fehlt	CM:CN
M:CN	1:CN + N:1 nicht möglich, da 1:N fehlt	CM:CN
CM:CN	1:CN + CN:1 möglich	/

Fazit:

Gesamtzahl der interessierenden Beziehungstypen (ohne „Spiegelbilder" an der Diagonale): 10

Davon:

Direkt im relationalen Datenbank-Modell repräsentierbar (einschließlich Darstellung in nur einer Tabelle): 5

Durch Umwandlung in einen Koppel-Objekttyp repräsentierbar: 1

Nur mit Semantikverlust repräsentierbar: 4

In der Abbildung 5-33 sind die 10 Beziehungstypen noch einmal graphisch dargestellt.

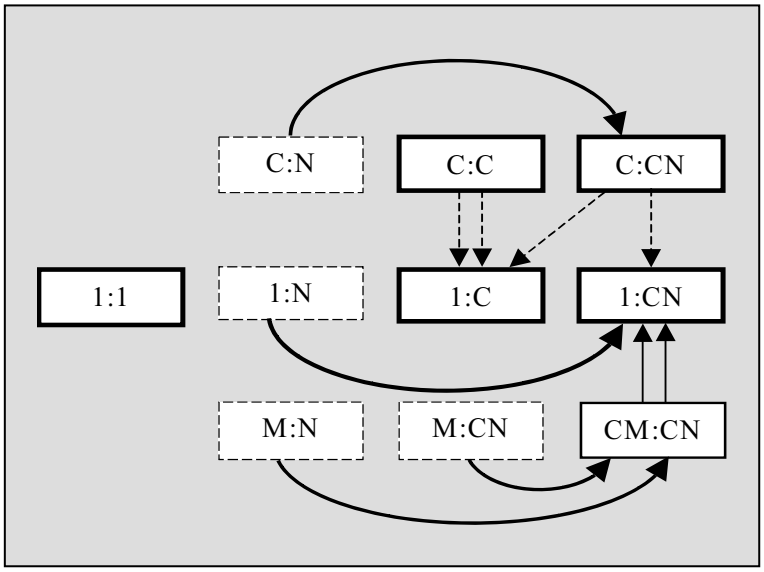


Abb. 5-33: Repräsentation der Beziehungstypen im relationalen Datenbank-Modell

Die 5 direkt repräsentierbaren Beziehungstypen (*C:C*, *C:CN*, *1:1*, *1:C* und *1:CN*) sind stark umrandet. Die wahlweise Reduktion der fakultativen Beziehungstypen C:C und C:CN auf die 1:C- bzw. 1:CN-Beziehungstypen ist durch gestrichelte Pfeile angedeutet. Der ohne Semantikverlust durch einen Koppel-Objekttyp darstellbare *CM:CN*-Beziehungstyp hat einen dünnen Rand, wobei durch ausgezogene Pfeile seine Zurückführung auf den direkt repräsentierbaren 1:CN-Beziehungstyp dargestellt ist. Die restlichen 4 Beziehungstypen (*C:N*, *1:N*, *M:N*, *M:CN*) haben einen gestrichelten Rand. Sie können nur mit Semantikverlust durch die Beziehungstypen repräsentiert werden, auf die jeweils der gebogene Pfeil zeigt.

5.3 Die Rekursiv-Beziehungstypen

Der Rekursiv-Beziehungstyp R dient im konzeptionellen Datenmodell dazu, die Menge der sachlogischen Zusammenhänge zu beschreiben, die zwischen Objekten ein und desselben Objekttyps bestehen:

$$R = \{ (a_1,a_2) \mid a_1 \in A, a_2 \in A \}$$

Angaben für den Rekursiv-Beziehungstyp

Er steht also für die Menge der geordneten Objektpaare (a_1,a_2), wobei sowohl a_1 als auch a_2 Elemente derselben Objektmenge A sind. Im Datenmodell werden jedoch nicht die einzelnen geordneten Objektpaare beschrieben. Stattdessen werden für die Menge sämtlicher Objektpaare – wie auch bei den dualen Beziehungstypen – nur 4 charakterisierende Angaben festgehalten:

1. Die Semantik der einen Beziehungstyp-Richtung, die wir im Weiteren wie folgt annehmen wollen:

   „Element des Objekttyps A *sendet eine Nachricht an* Element des Objekttyps A“.

2. Die Semantik der entgegengesetzten Beziehungstyp-Richtung, für die wir entsprechend annehmen:

   „Element des Objekttyps A *empfängt eine Nachricht von* Element des Objekttyps A“.

3. Die *Optionalität* und die *Kardinalität* der Beziehungstyp-Richtung „Sender→Empfänger“.

4. Die *Optionalität* und die *Kardinalität* der Beziehungstyp-Richtung „Sender←Empfänger“.

Im folgenden Abschnitt 5.3.1 erarbeiten wir zunächst eine Systematik der Rekursiv-Beziehungstypen, untersuchen dann im Abschnitt 5.3.2, wann ein Rekursiv-Beziehungstyp durch einen Koppel-Objekttyp dargestellt werden sollte, und analysieren schließlich im Abschnitt 5.3.3, welche Rekursiv-Beziehungstypen sich im relationalen Datenbank-Modell repräsentieren lassen.

5.3.1 Die Systematik der Rekursiv-Beziehungstypen

In diesem Abschnitt wird der Versuch unternommen, die interessierenden 7 Rekursiv-Beziehungstypen, die im Abschnitt 2.4.5 nach dem Ausschluss-Verfahren ermittelt wurden, nunmehr konstruktiv herzuleiten. Im Interesse der besseren Übersicht werden die 7 Rekursiv-Beziehungstypen in der Tabelle 5-6 noch einmal zusammengefasst. Um sie deutlich von den dualen Beziehungstypen unterscheiden zu können, werden sie in Klammern eingeschlossen und durch ein tiefgestelltes „R“ markiert.

Tab. 5-6: Rekursiv-Beziehungstypen (ohne „Spiegelbilder“)

7 Rekursiv-Beziehungstypen

$(1:1)_R$			$(1:CN)_R$
	$(C:C)_R$		$(C:CN)_R$
		$(M:N)_R$	$(M:CN)_R$
			$(CM:CN)_R$

Konstruktive Herleitung der Rekursiv-Beziehungstypen

Wie lassen sich nun diese 7 Rekursiv-Beziehungstypen konstruktiv herleiten? Die Verhältnisse sind hier komplizierter als bei den dualen Beziehungstypen. Bei einem dualen Beziehungstyp, der den sachlogischen Zusammenhang zwischen einem Sender-Objekttyp und einem Empfänger-Objekttyp beschreibt, verteilen sich die Restriktionen, die der jeweilige Beziehungstyp stellt, auf die Sender- und die Empfänger-Objekte. Bei den Rekursiv-Beziehungstypen liegen aber Sender- und Empfänger-Objekte im selben Objekttyp. Die Restriktionen des Rekursiv-Beziehungstyps gelten somit in ihrer Gesamtheit für alle Objekte des Objekttyps.

Wir gehen bei unseren Betrachtungen wie folgt vor:

1. Wir betrachten zunächst nur die 4 Rekursiv-Beziehungstypen, die in den ersten beiden Zeilen der Tabelle 5-6 stehen. Es handelt sich dabei um Rekursiv-Beziehungstypen mit dem *Verbot des multiplen Empfangs*. Das bedeutet, dass jeder Empfänger *höchstens eine* Nachricht empfangen kann. Wir beginnen mit dem Rekursiv-Beziehungstyp, der die rigorosesten Einschränkungen fordert. Wir lockern dann schrittweise die Einschränkungen und gelangen so zu den restlichen 3 Rekursiv-Beziehungstypen.

2. Bei den 3 Rekursiv-Beziehungstypen der letzten beiden Zeilen, für die das *Verbot des multiplen Empfangs nicht besteht*, gehen wir genau umgekehrt vor. Wir betrachten zunächst den Rekursiv-Beziehungstyp, der keinerlei Einschränkungen fordert, und leiten dann die restlichen 2 Rekursiv-Beziehungstypen durch schrittweise Hinzunahme von Bedingungen her.

5.3.1.1 Rekursiv-Beziehungstypen mit Verbot des multiplen Empfangs

Wir beginnen die Betrachtung mit dem *1:1-Rekursiv-Beziehungstyp*, der die Sende-Empfangs-Möglichkeiten am rigorosesten einschränkt und der damit – in Analogie zu den dualen Beziehungstypen - die Grundlage für alle anderen Rekursiv-Beziehungstypen darstellt. Es handelt sich – neben dem Verbot des multiplen Empfangs – um die folgenden Einschränkungen:

1:1-Rekursiv-Beziehungstyp

- *Empfangs-Pflicht:* Jedes Objekt muss ein Empfänger sein, und zwar – wegen des Verbots des multiplen Empfangs – der Empfänger genau einer Nachricht.
- *Verbot der multiplen Sendung:* Kein Objekt darf mehrere Nachrichten senden. Im Kontext der Empfangs-Pflicht für jedes Objekt bedeutet das aber, dass jedes Objekt genau eine Nachricht senden muss.

Wie wir bereits im Abschnitt 4.4.1 begründet haben, lässt der 1:1-Rekursiv-Beziehungstyp nur *Objekte-Zyklen* zu, die zu einem *Ein-Objekt-Zyklus* entarten können, bei dem ein Objekt sich selbst eine Nachricht sendet. Diese Zyklen bilden die Grundstruktur sachlogischer Zusammenhänge innerhalb eines Objekttyps. Sie sind bei allen anderen Rekursiv-Beziehungstypen wiederzufinden.

Wir heben nun gegenüber dem 1:1-Rekursiv-Beziehungstyp die Empfangspflicht auf und haben damit die folgende Situation zu untersuchen:

C:C-Rekursiv-Beziehungstyp

- *Keine Empfangs-Pflicht:* Es gibt mindestens ein Objekt, das keine Nachricht empfängt, die anderen empfangen – wegen des Verbots des multiplen Empfangs – genau eine Nachricht.
- *Verbot der multiplen Sendung:* Kein Objekt darf mehrere Nachrichten senden. Da aber mindestens ein Objekt keine Nachricht empfängt, muss auch mindestens ein Objekt keine Nachricht senden.

Diese Situation wird durch den *C:C-Rekursiv-Beziehungstyp* beschrieben. Im Abschnitt 4.4.2 hatten wir nachgewiesen, dass der C:C-Rekursiv-Beziehungstyp neben Objekte-Zyklen auch *Objekte-Ketten* zulässt, die zu einer Kette mit nur einem Objekt, also zu einem singulären Objekt, entarten können.

Wir heben nun gegenüber dem 1:1-Rekursiv-Beziehungstyp das Verbot der multiplen Sendung auf und gelangen zu den folgenden Festlegungen:

1:CN-Rekursiv-Beziehungstyp

- *Empfangs-Pflicht:* Jedes Objekt empfängt genau eine Nachricht.
- *Kein Verbot der multiplen Sendung:* Mindestens ein Objekt sendet mehrere Nachrichten. Da aber jedes Objekt genau eine Nachricht empfängt, bedeutet das: Für jede „überzählige" Nachricht, die das Objekt sendet, muss ein anderes Objekt auf seine Nachrichtensendung „verzichten".

Die angegebene Kombination wird durch den *1:CN-Rekursiv-Beziehungstyp* widergespiegelt. Dieser Rekursiv-Beziehungstyp führt – gemäß unseren Überlegungen im Abschnitt 4.4.3 – zu Objekte-Zyklen mit *„angebundenen" Monohierarchien.* Dies sind Objekte-Zyklen, bei denen jedes Objekt die Wurzel einer Monohierarchie - also eines Baums - sein kann.

Wir heben nun im letzten Schritt sowohl die Empfangs-Pflicht als auch das Verbot der multiplen Sendung auf. Es besteht also nur noch das Verbot des multiplen Empfangs:

C:CN-Rekursiv-Beziehungstyp

- *Keine Empfangs-Pflicht:* Mindestens ein Objekt empfängt keine Nachricht, die anderen genau eine.
- *Kein Verbot der multiplen Sendung:* Mindestens ein Objekt sendet mehrere Nachrichten. Sendet ein Objekt zwei Nachrichten, müssen wenigstens drei andere Objekte auf ihre Nachrichtensendung „verzichten". Damit nicht doch Empfangs-Pflicht besteht, muss nämlich die Summe der gesendeten Nachrichten kleiner sein als die Anzahl der Objekte.

Diese Situation wird durch den *C:CN-Rekursiv-Beziehungstyp* ausgedrückt. Im Abschnitt 4.4.4 hatten wir begründet, dass durch diesen Rekursiv-Beziehungstyp Objekte-Zyklen mit „angebundenen" und isolierten Monohierarchien dargestellt werden. Bei den isolierten Monohierarchien ist das Wurzel-Objekt nicht Bestandteil eines Objekte-Zyklus'.

Die beschriebenen Schritte der Aufhebung von Einschränkungen sind in der Abbildung 5-34 dargestellt, wobei für jeden Rekursiv-Beziehungstyp eine charakteristische Sender-Empfänger-Struktur angegeben wird.

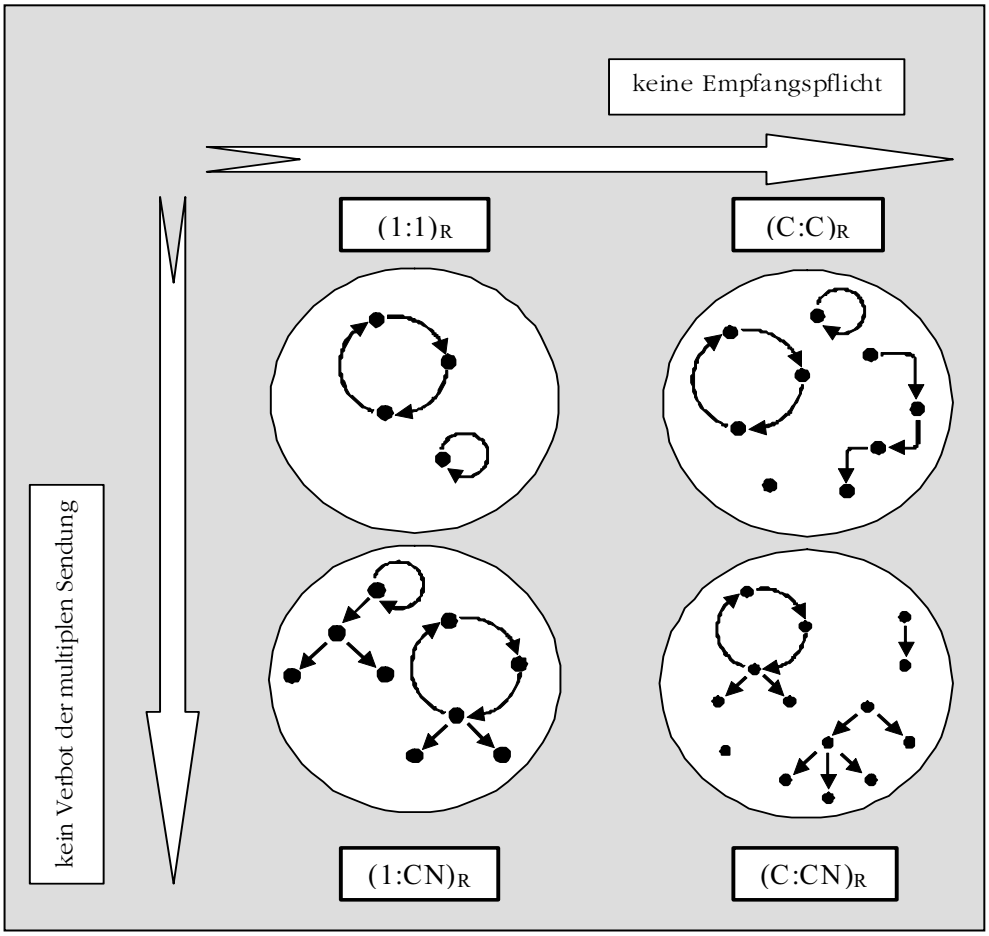


Abb. 5-34: Herleitung der Rekursiv-Beziehungstypen mit Verbot des multiplen Empfangs

5.3.1.2 Rekursiv-Beziehungstypen ohne Verbot des multiplen Empfangs

Bei der Herleitung der Rekursiv-Beziehungstypen ohne Verbot des multiplen Empfangs gehen wir von demjenigen Fall aus, der keine Einschränkungen für die Sender-Empfänger-Struktur fordert. Es handelt sich dabei um den *CM:CN-Rekursiv-Beziehungstyp*, bei dem jedes Objekt beliebig viele - natürlich auch keine - Nachrichten senden und empfangen kann. Die Objekte und ihre Beziehungslinien können somit ein *beliebiges Netzwerk* bilden. Die Abbildung 5-35 zeigt dafür ein Beispiel, das natürlich nicht erschöpfend sein kann.

CM:CN-Rekursiv-Beziehungstyp

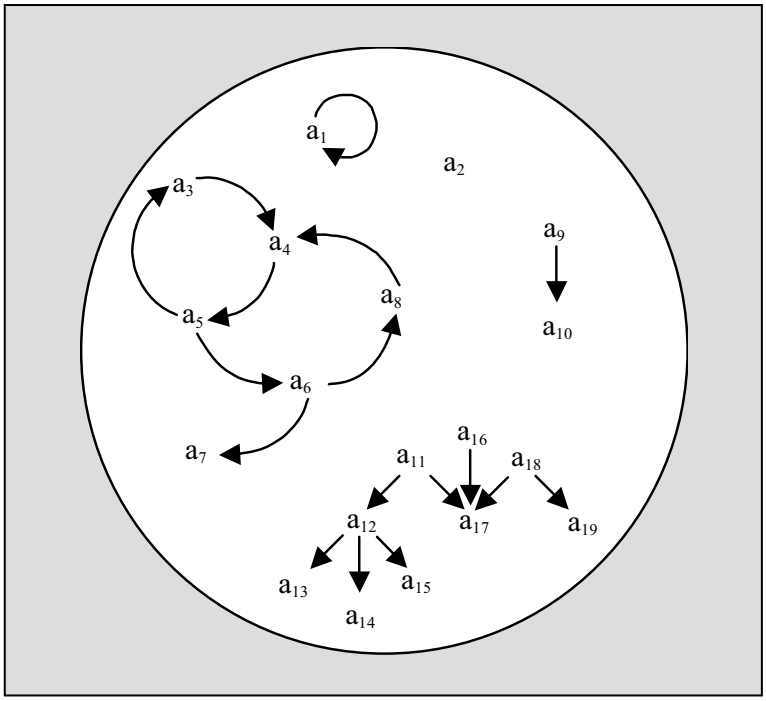


Abb. 5-35: Beispiel-Struktur für den CM:CN-Rekursiv-Beziehungstyp

Wir reduzieren nun durch Rand-Bedingungen schrittweise die Möglichkeiten zur Netzwerk-Bildung. Zunächst führen wir die Empfangspflicht ein:

- *Empfangs-Pflicht:* Jedes Objekt muss mindestens eine Nachricht empfangen.

Wir gelangen damit zum *M:CN-Rekursiv-Beziehungstyp*, für den die Abbildung 5-36 noch einmal eine Beispiel-Struktur zeigt.

M:CN-Rekursiv-Beziehungstyp

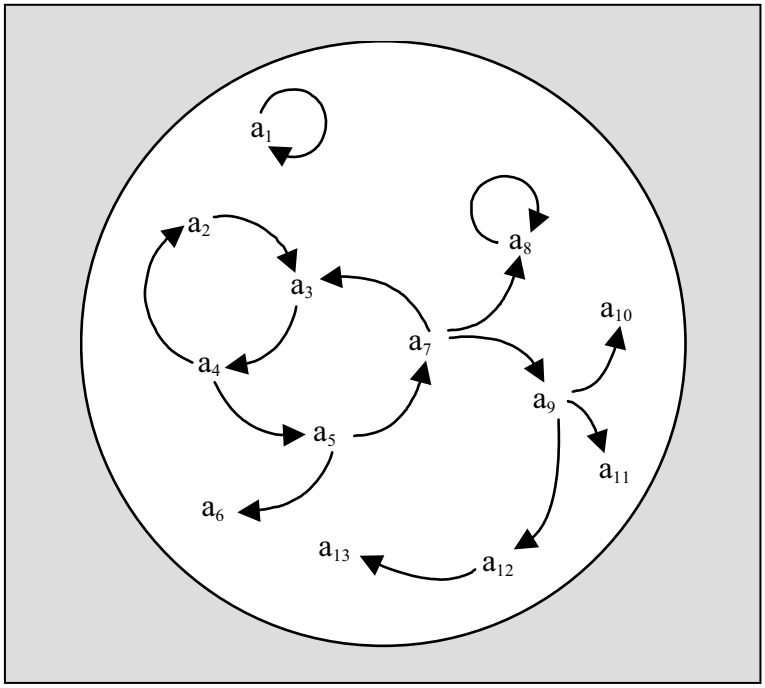


Abb. 5-36: Beispiel-Struktur für den M:CN-Rekursiv-Beziehungstyp

Konstruktive Beschreibung

Die möglichen Sender-Empfänger-Strukturen, die durch diesen Rekursiv-Beziehungstyp repräsentiert werden, lassen sich am einfachsten konstruktiv beschreiben:

1. Zunächst liegt ein leerer Objekttyp vor, der noch kein einziges Objekt enthält.
2. Das erste aufzunehmende Objekt kann nur als Ein-Objekt-Zyklus hinzugefügt werden ($a_1 \rightarrow a_1$). Es muss nämlich wenigstens eine Nachricht empfangen. Da es aber bisher „allein auf der Welt“ ist, kann es nur selbst Sender dieser Nachricht sein.
3. Die folgenden Aktionen können beliebig oft und in beliebiger Aufeinanderfolge ausgeführt werden:

a) Ein neues Objekt kann als Ein-Objekt-Zyklus aufgenommen werden. Es ist – wie unter Punkt 2 erläutert – dann Empfänger der von ihm selbst gesendeten Nachricht. Beispielsweise war ursprünglich ($a_2 \rightarrow a_2$) ein solcher Ein-Objekt-Zyklus.

b) Ein neues Objekt kann in einen bereits bestehenden Objekte-Zyklus „eingebaut" werden. Es ist dann Sender und Empfänger genau einer Nachricht. So wurde der Ein-Objekt-Zyklus ($a_2 \rightarrow a_2$) durch die neuen Objekte a_3 und a_4 zum Drei-Objekte-Zyklus ($a_2 \rightarrow a_3 \rightarrow a_4 \rightarrow a_2$) erweitert.

c) Ein neues Objekt O kann hinzugefügt werden, ohne Sender einer Nachricht zu sein, solange es Empfänger einer Nachricht ist, die ein bereits existierendes Objekt an O sendet. So wurde beispielsweise das Objekt a_6 hinzugefügt, das selbst keine Nachricht sendet, aber von a_5 eine Nachricht erhält.

d) Von einem bereits existierenden Objekt kann eine Nachricht an sich selbst ($a_8 \rightarrow a_8$) oder an ein anderes existierendes Objekt ($a_7 \rightarrow a_3$) gesendet werden.

Im nächsten Schritt zur Einengung der Sender-Empfänger-Strukturen fordern wir zusätzlich zur Empfangspflicht auch noch die Sendepflicht:

- *Empfangs-Pflicht:* Jedes Objekt muss mindestens eine Nachricht empfangen.
- *Sende-Pflicht:* Jedes Objekt muss mindestens eine Nachricht senden. Für jede gesendete Nachricht, die über die *eine* Pflicht-Sendung hinausgeht, muss es ein Objekt geben, das diese „Überschuss-Nachricht" - über seinen Pflicht-Empfang von *einer* Nachricht hinaus - entgegennimmt.

Diese Konstellation wird durch den *M:N-Rekursiv-Beziehungstyp* zum Ausdruck gebracht, für den die Abbildung 5-37 ein Beispiel zeigt.

M:N-Rekursiv-Beziehungstyp

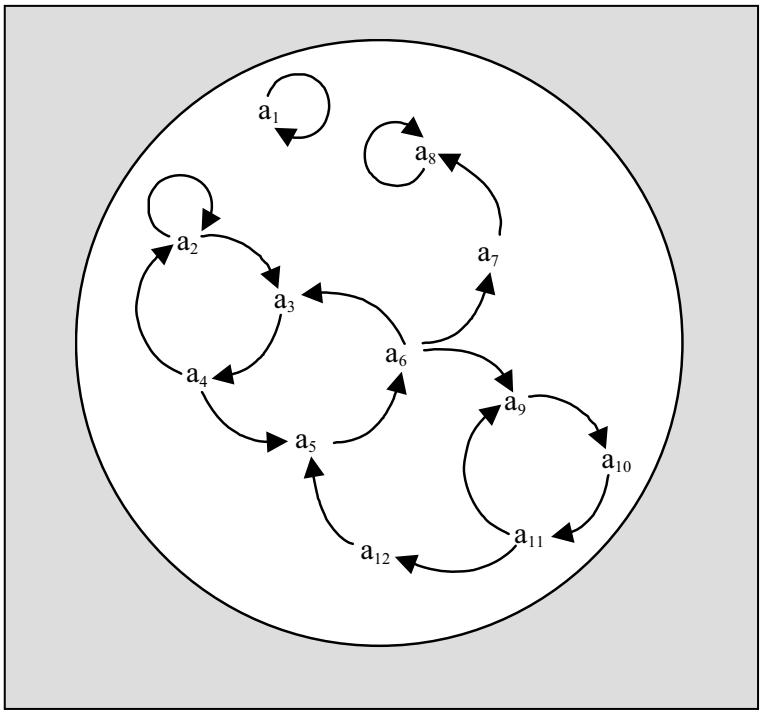


Abb. 5-37: Beispiel-Struktur für den M:N-Rekursiv-Beziehungstyp

Konstruktive Beschreibung

Die möglichen Sender-Empfänger-Strukturen sollen nun ebenfalls konstruktiv beschrieben werden:

1. Zunächst liegt ein leerer Objekttyp vor, der noch kein einziges Objekt enthält.
2. Das erste aufzunehmende Objekt kann nur in Form eines Ein-Objekt-Zyklus hinzugefügt werden ($a_1 \rightarrow a_1$). Es muss nämlich wenigstens eine Nachricht senden und eine empfangen.
3. Die folgenden Aktionen können beliebig oft und in beliebiger Aufeinanderfolge ausgeführt werden:
   a) Ein neues Objekt kann als Ein-Objekt-Zyklus aufgenommen werden ($a_2 \rightarrow a_2$). Es ist – wie unter Punkt 2 erläutert – dann Empfänger der von ihm selbst gesendeten Nachricht.

b) Ein neues Objekt kann in einen bereits bestehenden Objekte-Zyklus „eingebaut“ werden. Es ist dann Sender und Empfänger genau einer Nachricht. So kann der Ein-Objekt-Zyklus ($a_2 \rightarrow a_2$) zum Zyklus ($a_2 \rightarrow a_3 \rightarrow a_4 \rightarrow a_2$) erweitert werden.

c) An einem bereits existierenden Objekt kann eine neue Objekte-Kette - mit keinem oder mehreren neuen Objekten als „Zwischenstationen“ - beginnen, die aber wieder in einem existierenden Objekt enden muss. Enthält die Kette keine „Zwischenstationen“, handelt es sich einfach darum, dass ein bestehendes Objekt eine Nachricht an ein anderes bestehendes Objekt sendet ($a_6 \rightarrow a_9$). Ein Beispiel für eine Kette mit einer „Zwischenstation“ ist ($a_{11} \rightarrow a_{12} \rightarrow a_5$).

Die schrittweisen Einschränkungen der durch den CM:CN-Rekursiv-Beziehungstyp repräsentierten allgemeinen Netzwerk-Struktur ist zusammenfassend in der Abbildung 5-38 dargestellt, wobei für jeden Rekursiv-Beziehungstyp eine charakteristische Sender-Empfänger-Struktur angegeben wird.

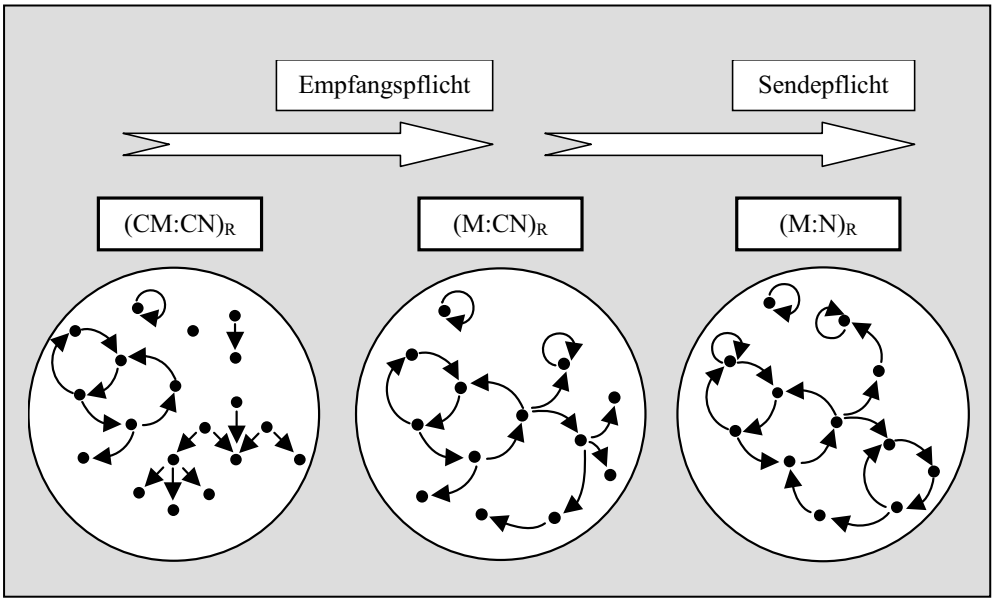
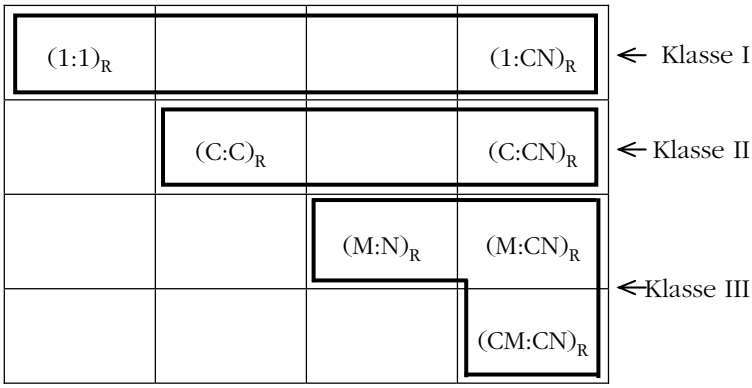


Abb. 5-38: Herleitung der Rekursiv-Beziehungstypen ohne Verbot des multiplen Empfangs

5.3.2 Die Umwandlung in einen Koppel-Objekttyp

Mit den Transformationsregeln T13 – T19 wurde im Abschnitt 4.4 ein Vorgehen beschrieben, nach dem die Rekursiv-Beziehungstypen im relationalen Datenbank-Modell repräsentiert werden können. Für einige der Rekursiv-Beziehungstypen wurde dabei gefordert, dass sie gegebenenfalls in einen Koppel-Objekttyp umzuformen sind. In diesem Abschnitt soll nun untersucht werden, in welchen Fällen das sinnvoll ist und wann nicht. Die Ausführungen erfolgen analog zu den Überlegungen für die dualen Beziehungstypen im Abschnitt 5.2.3.

Tab. 5-7: Klassifizierung der Rekursiv-Beziehungstypen (ohne „Spiegelbilder“)



Für die weiteren Betrachtungen fassen wir die 7 relevanten Rekursiv-Beziehungstypen in drei Klassen zusammen, wie dies aus der Tabelle 5-7 zu ersehen ist. Die Klassen haben die folgende Bedeutung:

Klasse I: *Rekursiv-Beziehungstypen mit Empfangspflicht.* Jedes Objekt muss genau eine Nachricht empfangen.

Klasse II: *Rekursiv-Beziehungstypen ohne Empfangspflicht.* Mindestens ein Objekt empfängt keine, die anderen Objekte genau eine Nachricht.

Klasse III: *Rekursiv-Beziehungstypen mit multiplen Empfängern.* Mindestens ein Objekt empfängt mehrere und mindestens ein Objekt sendet mehrere Nachrichten.

Für die Umwandlung eines Rekursiv-Beziehungstyps in einen Koppel-Objekttyp gilt das in der Abbildung 5-39 wiedergegebene Schema.

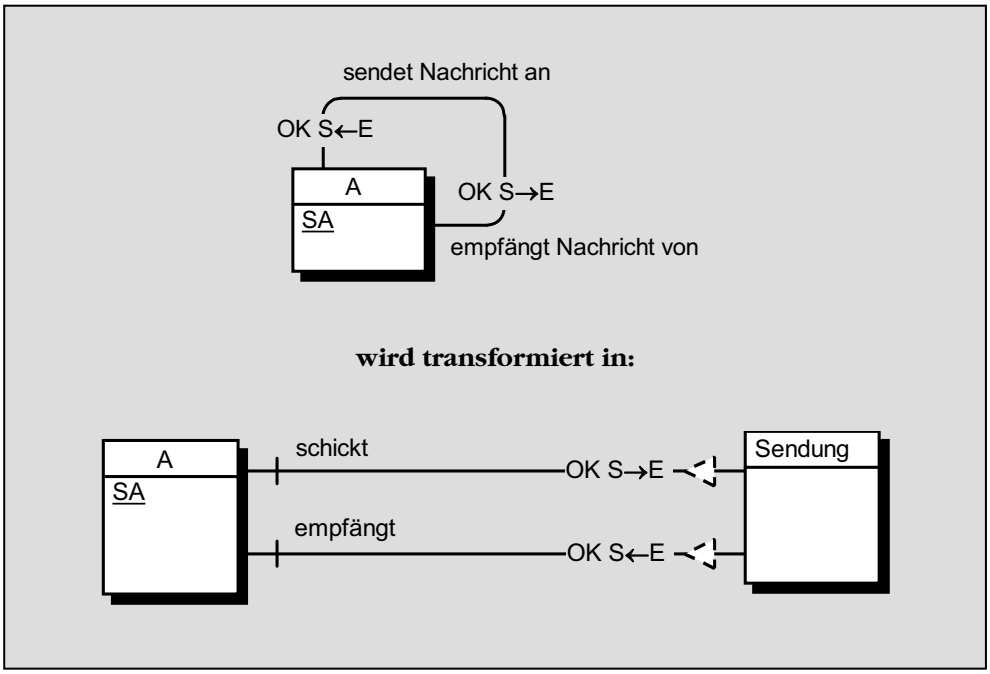


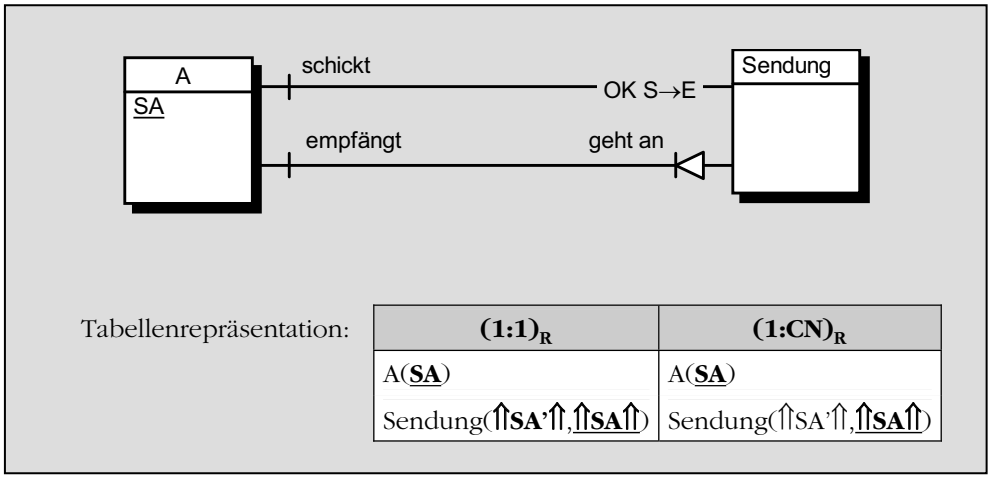
Abb. 5-39: Umwandlung eines Rekursiv-Beziehungstyps in einen Koppel-Objekttyp

In der Abbildung 5-39 steht „OK S→E“ für die Optionalität/Kardinalität der Beziehungstyp-Richtung „Objekt *sendet Nachricht an* Objekt“ und „OK S←E“ für die Optionalität/Kardinalität der Beziehungstyp-Richtung „Objekt *empfängt Nachricht von* Objekt“ Die Nutzung der Beziehungstyp-Richtungen für die Identifizierung des Koppel-Objekttyps hängt vom speziellen Rekursiv-Beziehungstyp ab. Darum wurde sie gestrichelt dargestellt.

In den folgenden Abschnitten wird für die drei Klassen von Rekursiv-Beziehungstypen untersucht, wann ihre Umwandlung in einen Koppel-Objekttyp sinnvoll ist.

5.3.2.1 Klasse I (Rekursiv-Beziehungstypen mit Empfangspflicht)

Bei den Rekursiv-Beziehungstypen mit Empfangspflicht – also $(1:1)_R$ und $(1:CN)_R$ - führt ihre Umwandlung in einen Koppel-Objekttyp zu einem 1:(OK S→E)-Beziehungstyp und zu einem 1:1-Beziehungstyp. Dies ist in der Abbildung 5-40 dargestellt.



Tabellenrepräsentation:

$(1:1)_R$	$(1:CN)_R$
A(**SA**)	A(**SA**)
Sendung(⇑**SA'**⇑,⇑**SA**⇑)	Sendung(⇑SA'⇑,⇑**SA**⇑)

Abb. 5-40: Umwandlung eines Rekursiv-Beziehungstyps der Klasse I in einen Koppel-Objekttyp

Transformationsregel T03

In der Tabelle „Sendung" verweist der Fremdschlüssel ⇑SA'⇑ auf den Sender und der Fremdschlüssel ⇑**SA**⇑, der zugleich Primärschlüssel der Tabelle ist, auf den Empfänger. Der 1:1-Beziehungstyp ist entsprechend der Transformationsregel T03 nicht zu repräsentieren. Stattdessen werden die Datensätze der Koppel-Tabelle „Sendung" an die Empfänger-Datensätze der Tabelle „A" angehängt, wobei natürlich der Primärschlüssel ⇑**SA**⇑ nicht gedoppelt wird:

Tabellenrepräsentation:

$(1:1)_R$	$(1:CN)_R$
A(**SA**,⇑**SA'**⇑)	A(**SA**,⇑SA'⇑)

Jeder Empfänger einer Nachricht verweist somit durch den Fremdschlüssel ⇑SA'⇑ auf „seinen" Sender. Diese Umformung

entspricht einer Integration der Koppel-Tabelle in die Objekttyp-Tabelle. Dadurch wird einerseits Speicherplatz eingespart und andererseits wächst die Performance beim Auswertung der sachlogischen Zusammenhänge zwischen den Objekten.

Beispiel für Rekursiv-Beziehungstyp der Klasse I

Als ein Beispiel für die Darstellung von Rekursiv-Beziehungstypen der Klasse I betrachten wir eine Schulklasse, die zur Abiturfeier eine Zeitung herausgeben möchte. In dieser Zeitung soll jeder Schüler durch einen Artikel vorgestellt werden. Manche Schüler schreiben mehrere Artikel, andere dafür keinen. Die Abbildung 5-41 zeigt die Transformation des 1:CN-Rekursiv-Beziehungstyps in einen Koppel-Objekttyp sowie die entsprechende Tabellenrepräsentation.

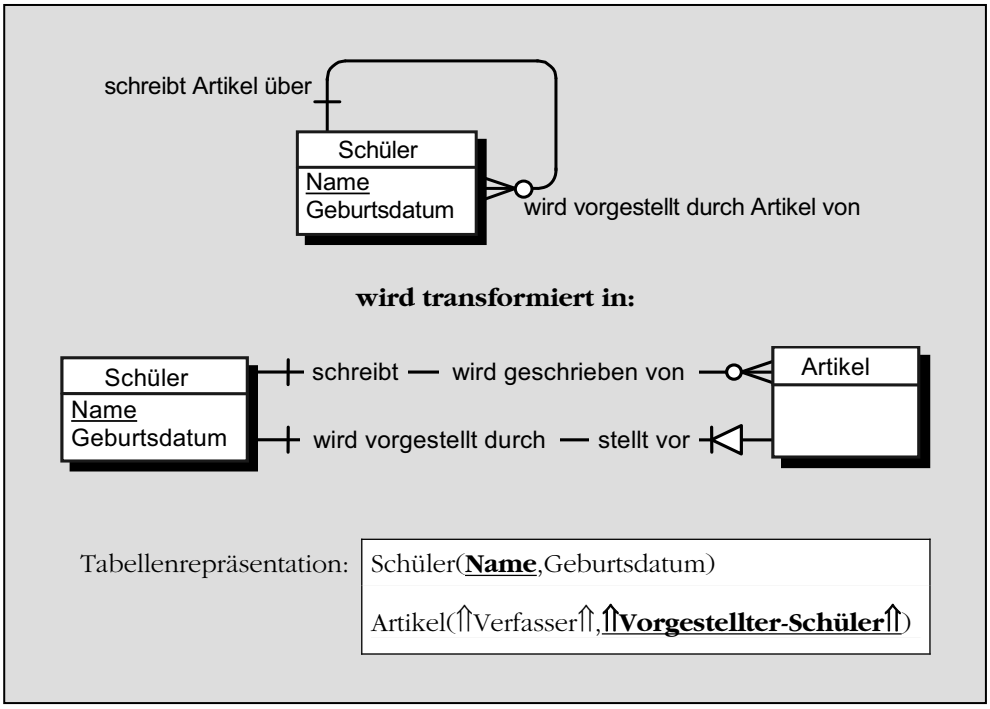


Abb. 5-41: Beispiel für die Umwandlung eines Rekursiv-Beziehungstyps der Klasse I in einen Koppel-Objekttyp

Die Tabellen „Schüler“ und „Artikel“ könnten Beispieldaten enthalten, wie sie in der Abbildung 5-42 angegeben sind.

Schüler

Name	Geburtsdatum
Reiner Unsinn	1.1.91
Theo Retisch	2.2.91
Marga Rine	3.3.91

Artikel

Verfasser	Vorgestellter-Schüler
Marga Rine	Reiner Unsinn
Reiner Unsinn	Theo Retisch
Reiner Unsinn	Marga Rine

Abb. 5-42: Beispieltabellen für einen Rekursiv-Beziehungstyp der Klasse I

Soll bei dieser Tabellenrepräsentation der Verfasser (einschließlich seines Geburtsdatums) ermittelt werden, durch dessen Artikel „Theo Retisch“ in der Abiturzeitung vorgestellt wird, so muss zunächst die Artikel-Tabelle ausgewertet werden. In ihr findet man, dass der Verfasser des Artikels über „Theo Retisch“ der Schüler „Reiner Unsinn“ ist. Sein Geburtsdatum muss nun noch der Tabelle „Schüler“ entnommen werden.

Im Interesse einer besseren Performance sollte man den Zugriff auf die Tabelle „Artikel“ jedoch vermeiden. Da es zu jeder Zeile der Tabelle „Schüler“ genau eine Zeile der Tabelle „Artikel“ gibt, können die Zeilen der Tabelle „Artikel“ an die entsprechenden Zeilen der Tabelle „Schüler“ angehängt werden, wobei natürlich die identischen Spalten „Name“ und „Vorgestellter-Schüler“ nur einmal aufgeführt werden. Dieses Vorgehen entspricht der oben beschriebenen Integration der Koppel-Tabelle („Artikel“) in die Objekttyp-Tabelle („Schüler“). Die reduzierte Tabellen-Typbeschreibung hat dann die Form:

Tabellenrepräsentation:

Schüler(**Name**,Geburtsdatum,
⇑Verfasser⇑)

Die Beispieltabellen nehmen die reduzierte Form an, die in der Abbildung 5-43 dargestellt ist. Zur Lösung der Beispielaufgabe muss nun lediglich die eine Tabelle „Schüler“ ausgewertet werden.

Schüler

Name	Geburtsdatum	Verfasser
Reiner Unsinn	1.1.83	Marga Rine
Theo Retisch	2.2.83	Reiner Unsinn
Marga Rine	3.3.83	Reiner Unsinn

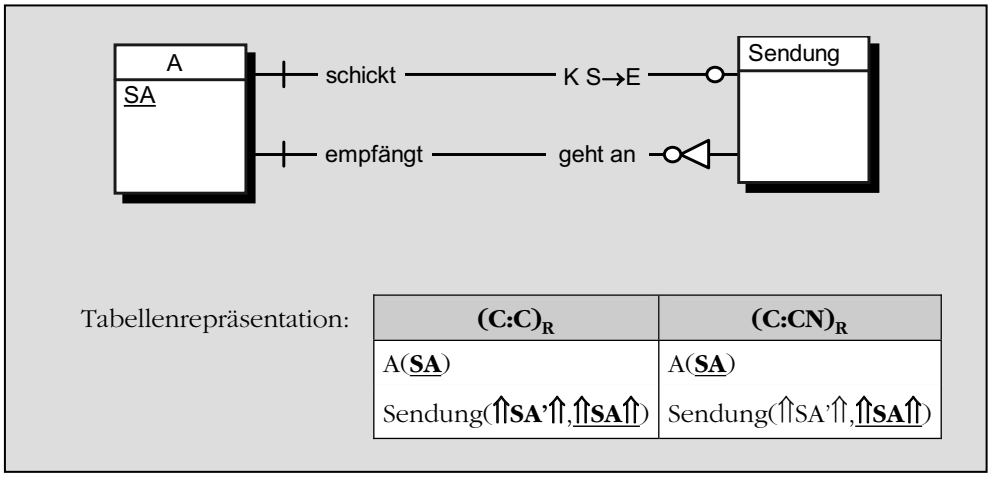
Abb. 5-43: Reduzierte Beispieltabelle für einen Rekursiv-Beziehungstyp der Klasse I

Fazit: **Die *Rekursiv-Beziehungstypen der Klasse I* (die Rekursiv-Beziehungstypen mit Empfangspflicht, also $(1:1)_R$ und $(1:CN)_R$) werden aus Performance-Gründen *nicht als Koppel-Tabelle* repräsentiert. Stattdessen wird im Empfänger-Datensatz auf den Sender verwiesen.**

5.3.2.2 Klasse II (Rekursiv-Beziehungstypen ohne Empfangspflicht)

Die Rekursiv-Beziehungstypen ohne Empfangspflicht - also $(C:C)_R$ und $(C:CN)_R$ - lassen sich gemäß Abbildung 5-44 in einen Koppel-Objekttyp umwandeln. Der Objekttyp „A“ ist in seiner Rolle als Sender mit dem Koppel-Objekttyp „Sendung“ durch einen 1:C(K S→E)-Beziehungstyp verbunden, wobei die Kardinalität (K S→E) entweder 1 oder N ist. Zwischen dem Objekttyp „A“ in seiner Rolle als Empfänger und dem Koppel-Objekttyp „Sendung“ besteht ein 1:C-Beziehungstyp.

In der Tabelle „Sendung“ verweist der Fremdschlüssel ⇑SA'⇑ auf den Sender und der Fremdschlüssel **⇑SA⇑**, der zugleich Primärschlüssel der Tabelle ist, auf den Empfänger.



Tabellenrepräsentation:

$(C:C)_R$	$(C:CN)_R$
A(**SA**)	A(**SA**)
Sendung(⇑**SA'**⇑,⇑**SA**⇑)	Sendung(⇑SA'⇑,⇑**SA**⇑)

Abb. 5-44: Umwandlung eines Rekursiv-Beziehungstyps der Klasse II in einen Koppel-Objekttyp

Transformationsregel T05

Der 1:C-Beziehungstyp *kann* entsprechend der Transformationsregel T05 eliminiert werden, indem die Datensätze der Koppel-Tabelle „Sendung" an die zugeordneten Empfänger-Datensätze der Tabelle „A" angehängt werden. Dabei wird der Primärschlüssel ⇑**SA**⇑ natürlich nur einmal aufgeführt und der Verweis ⇑*SA'*⇑ auf den Sender wird als nicht-eingabepflichtig vereinbart. Das ist insbesondere dann sinnvoll, wenn die Optionalität der Beziehungstyp-Richtung „A *empfängt* Sendung" selten realisiert wird:

Tabellenrepräsentation:

$(C:C)_R$	$(C:CN)_R$
A(**SA**,⇑***SA'***⇑)	A(**SA**,⇑*SA'*⇑)

Diejenigen Objekte, die Empfänger einer Nachricht sind, verweisen durch den Fremdschlüssel ⇑*SA'*⇑ auf „ihren" Sender. Bei denjenigen Objekten, die keine Nachricht empfangen, wird der Fremdschlüssel mit der NULL-Marke belegt.

Performance-Verbesserung

Die beschriebene Eliminierung der Koppel-Tabelle führt zu einer Performance-Verbesserung, wenn zwischen Sendern und Empfängern navigiert werden soll, weil so nur noch eine Tabelle auszuwerten ist. Allerdings kann sich der Speicherplatz-Bedarf erhö-

hen, wenn der Fremdschlüssel ⇑*SA'*⇑ häufig mit der NULL-Marke belegt ist.

Betrachtet man den Speicherplatz-Bedarf zur Darstellung eines Beziehungstyps der Klasse II, dann ergeben sich für die beiden Repräsentationsformen die folgenden Ausdrücke:

a) Repräsentation ***mit*** Koppel-Tabelle:	b) Repräsentation ***ohne*** Koppel-Tabelle:
$\overline{\overline{\text{Sendung}}} \cdot 2 \cdot Len(\underline{\text{SA}})$	$\overline{\text{A}} \cdot Len(\underline{\text{SA}})$

Der Speicherplatz-Bedarf im Fall a) berücksichtigt, dass in jeder Zeile der Koppel-Tabelle „Sendung“ der Primärschlüssel „SA“ zweimal gespeichert werden muss. Im Fall b) wird in jeder Zeile der Tabelle „A“ Speicherplatz für einen Wert des Fremdschlüssels ⇑SA'⇑ bereitgestellt, auch dann, wenn der Fremdschlüssel mit der NULL-Marke belegt ist. Setzt man beide Ausdrücke gleich, so benötigt man denselben Speicherplatz, wenn gilt:

$$\overline{\overline{\text{Sendung}}} = \frac{1}{2}\overline{\text{A}},$$

wenn also nur jedes zweite Objekt eine Sendung empfängt. Die Verwendung einer Koppel-Tabelle führt immer dann zu einem Speicherplatz-Gewinn, wenn *weniger als die Hälfte* der Objekte eine Sendung empfangen. Ist eine höhere Performance aber wichtiger als ein geringerer Speicherplatz-Bedarf, wird man sich trotzdem gegen die Koppel-Tabelle entscheiden.

Beispiel für Rekursiv-Beziehungstyp der Klasse II

Als ein Beispiel für die Darstellung von Rekursiv-Beziehungstypen der Klasse II betrachten wir die Bewohner eines Hauses. Die Hausverwaltung speichert Informationen darüber, welcher Bewohner welche anderen Bewohner zur Untermiete aufgenommen hat. Die meisten Bewohner haben keine Untermieter. Einige haben einen und ganz wenige sogar mehrere Untermieter. Die wenigen Untermieter wohnen jeweils nur bei *einem* Bewohner zur Untermiete. In der Abbildung 5-45 sind die Transformation des C:CN-Rekursiv-Beziehungstyps in einen Koppel-Objekttyp sowie die entsprechende Tabellenrepräsentation wiedergegeben.

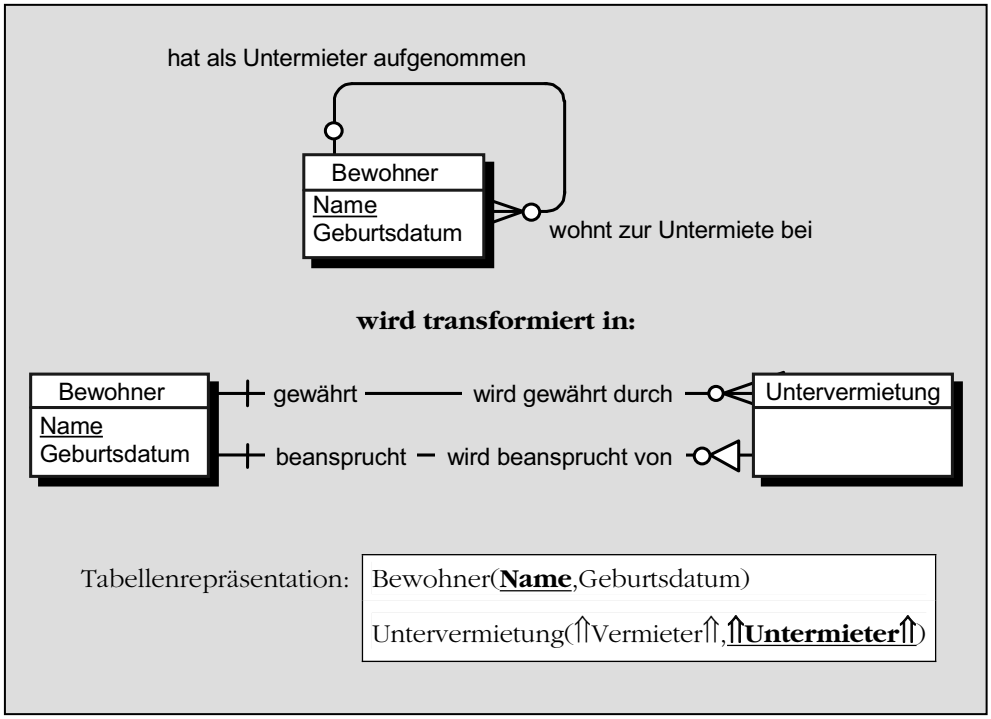


Abb. 5-45: Beispiel für die Umwandlung eines Rekursiv-Beziehungstyps der Klasse II in einen Koppel-Objekttyp

Die Tabellen „Bewohner“ und „Untervermietung“ könnten Beispieldaten enthalten, wie sie in der Abbildung 5-46 angegeben sind.

Unter den 9 Bewohnern gibt es nur 3 Untermieter. Somit ist die Repräsentation des C:CN-Rekursiv-Beziehungstyps durch eine Koppel-Tabelle die Alternative mit dem geringsten Speicherplatz-Bedarf. Da außerdem anzunehmen ist, dass die Untermiet-Verhältnisse selten abgefragt werden und damit die Performance-Aspekte nicht im Vordergrund stehen, kann man es bei dieser Darstellungsform belassen.

Bewohner

Name	Geburtsdatum
Linda Hauch	31.07.1947
Hans Arostock	19.01.1944
Klara Fall	13.12.1969
Wim Pernstift	16.03.1966
Marie Nade	16.05.1980
Otto Mane	01.01.1960
Ron Dell	03.07.1934
Marta Pfahl	06.06.1978
Pitt Bull	05.05.1982

Untervermietung

Vermieter	Untermieter
Ron Dell	Marie Nade
Ron Dell	Pitt Bull
Hans Arostock	Marta Pfahl

Abb. 5-46: Beispieltabellen für einen Rekursiv-Beziehungstyp der Klasse II

Fazit: **Die *Rekursiv-Beziehungstypen der Klasse II* (die Rekursiv-Beziehungstypen ohne Empfangspflicht, also $(C:C)_R$ und $(C:CN)_R$) können *sowohl mit als auch ohne Koppel-Tabelle* repräsentiert werden. Die Entscheidung muss unter dem Zielkonflikt „entweder weniger Speicherplatz-Bedarf oder geringere Rechenzeit“ gefällt werden.**

5.3.2.3 Klasse III (Rekursiv-Beziehungstypen mit multiplen Empfängern)

Bei der Umwandlung der Rekursiv-Beziehungstypen mit multiplen Empfängern, also der M:N-, M:CN- und CM:CN-Rekursiv-Beziehungstypen, in einen Koppel-Objekttyp entstehen ein dualer 1:(C)N-Beziehungstyp und ein dualer (C)N:1-Beziehungstyp. Die Abbildung 5-47 zeigt das Ergebnis, wobei die eventuelle Optionalität der Beziehungstyp-Richtungen „A *schickt* Sendung“ bzw. „B *empfängt* Sendung“ durch den punktierten Kreis symbolisiert wird.

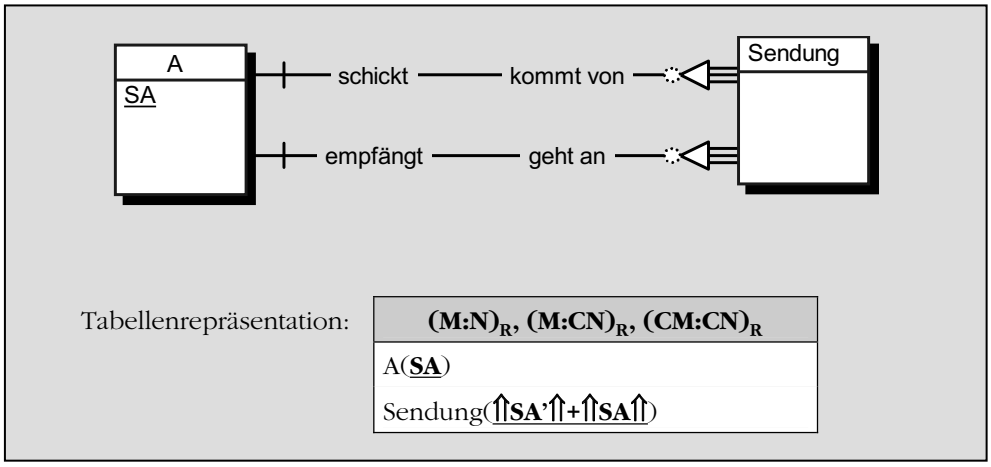


Abb. 5-47: Umwandlung eines Rekursiv-Beziehungstyps der Klasse III in einen Koppel-Objekttyp

In der Tabelle „Sendung" verweist der Fremdschlüssel ⇑SA'⇑ auf den Sender und der Fremdschlüssel ⇑SA⇑ auf den Empfänger. Die beiden Fremdschlüssel sind eingabepflichtig und jeder für sich nicht-unikal. Erst in der Kopplung ⇑SA'⇑+⇑SA⇑ bilden sie den unikalen Primärschlüssel der Koppel-Tabelle „Sendung".

Eine Vereinfachung der Tabellenstruktur ist nicht möglich. Wollte man nämlich die Zeilen der Tabelle „Sendung" an die entsprechenden Sender-Zeilen bzw. an die entsprechenden Empfänger-Zeilen der Tabelle „A" anhängen, so würde das zu einer Verletzung der 1. Normalform führen, weil es zu einer Sender-Zeile bzw. zu einer Empfänger-Zeile mehrere Zeilen der Tabelle „Sendung" geben kann.

Beispiel für Rekursiv-Beziehungstyp der Klasse III

Als ein Beispiel für einen Rekursiv-Beziehungstyp der Klasse III betrachten wir eine Weiterbildungsstätte, in der Lehrgänge angeboten werden. Einige Lehrgänge bilden die Grundlage für einen oder für mehrere weiterführende Lehrgänge. Es gibt auch Lehrgänge, für die es keine Weiterführung gibt. Ein Lehrgang kann einen oder mehrere Lehrgänge voraussetzen, kann aber auch grundständig sein und keine Voraussetzungen erfordern. Die Abbildung 5-48 zeigt die Transformation des CM:CN-Rekursiv-Beziehungstyps in einen Koppel-Objekttyp sowie die entsprechende Tabellenrepräsentation.

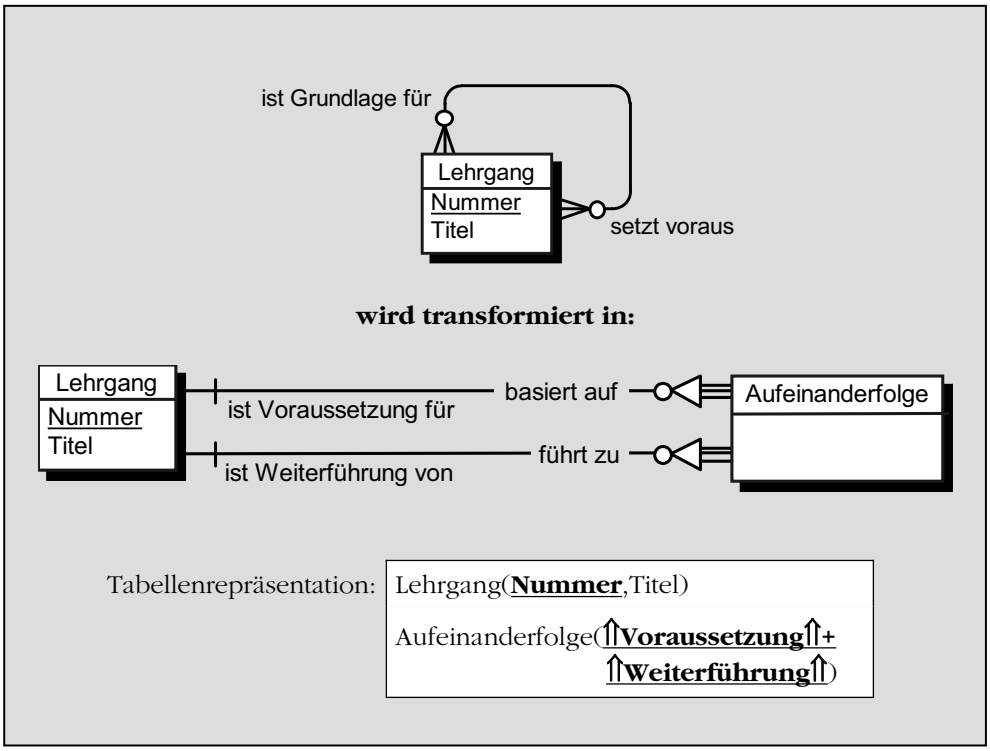


Abb. 5-48: Beispiel für die Umwandlung eines Rekursiv-Beziehungstyps der Klasse III in einen Koppel-Objekttyp

Beispieldaten für die beiden Tabellen „Lehrgang“ und „Aufeinanderfolge“ sind in der Abbildung 5-49 angegeben.

Man erkennt an den Daten der Abbildung 5-49, dass es sich tatsächlich um einen CM:CN-Rekursiv-Beziehungstyp handelt. Die Lehrgänge „102“ und „105“ haben keine Weiterführung, der Lehrgang „101“ hat zwei Weiterführungen. Die Lehrgänge „101“ und „104“ haben keine Voraussetzung, der Lehrgang „105“ setzt dagegen zwei Lehrgänge voraus.

Lehrgang

Nummer	Titel
101	Strukturierte Modellierung
102	Objektorientierte Modellierung
103	Datenbanken
104	Wissensverarbeitung
105	Data-Warehouse-Systeme

Aufeinanderfolge

Voraus-setzung	Weiter-führung
101	102
101	103
103	105
104	105

Abb. 5-49: Beispieltabellen für einen Rekursiv-Beziehungstyp der Klasse III

Fazit: **Die *Rekursiv-Beziehungstypen der Klasse III* (die Rekursiv-Beziehungstypen mit multiplen Empfängern, also $(M:N)_R$, $(M:CN)_R$ und $(CM:CN)_R$) können *nur durch eine Koppel-Tabelle* repräsentiert werden.**

5.3.2.4 Obligatorische, fakultative und reduzible Rekursiv-Beziehungstypen

In den vorangegangenen Abschnitten wurde untersucht, ob die drei Klassen von Rekursiv-Beziehungstypen mit oder ohne Koppel-Tabelle repräsentiert werden sollten. Das ist gleichbedeutend mit der Frage, ob sie auf duale Beziehungstypen zurückzuführen sind.

Die erzielten Ergebnisse bieten uns – analog zu den Überlegungen im Abschnitt 5.2.3.5 - nun die Möglichkeit, die 7 Rekursiv-Beziehungstypen in drei Gruppen einzuteilen:

1. *Obligatorische Rekursiv-Beziehungstypen:* Dies sind Rekursiv-Beziehungstypen, die unbedingt erforderlich sind, weil sie sich nicht in sinnvoller Weise auf duale Beziehungstypen zurückführen lassen.
2. *Fakultative Rekursiv-Beziehungstypen:* Dies sind jene Rekursiv-Beziehungstypen, die nicht unbedingt erforderlich sind, da sie sich auf duale Beziehungstypen zurückführen lassen.

Sie sind jedoch nützlich, da sie gegebenenfalls eine Verringerung des Speicherplatz-Bedarfs oder eine Verbesserung der Performance herbeiführen können.

3. *Reduzible Rekursiv-Beziehungstypen:* Dies sind jene Rekursiv-Beziehungstypen, die stets - durch Umwandlung in einen Koppel-Objekttyp - auf duale Beziehungstypen zurückgeführt werden müssen.

Die Abbildung 5-50 zeigt diese Gruppierung der Rekursiv-Beziehungstypen, wobei durch einen gestrichelten Pfeil die *fallweise* und durch einen ausgezogenen Pfeil die *unbedingte* Zurückführung eines Rekursiv-Beziehungstyps auf duale Beziehungstypen dargestellt wird.

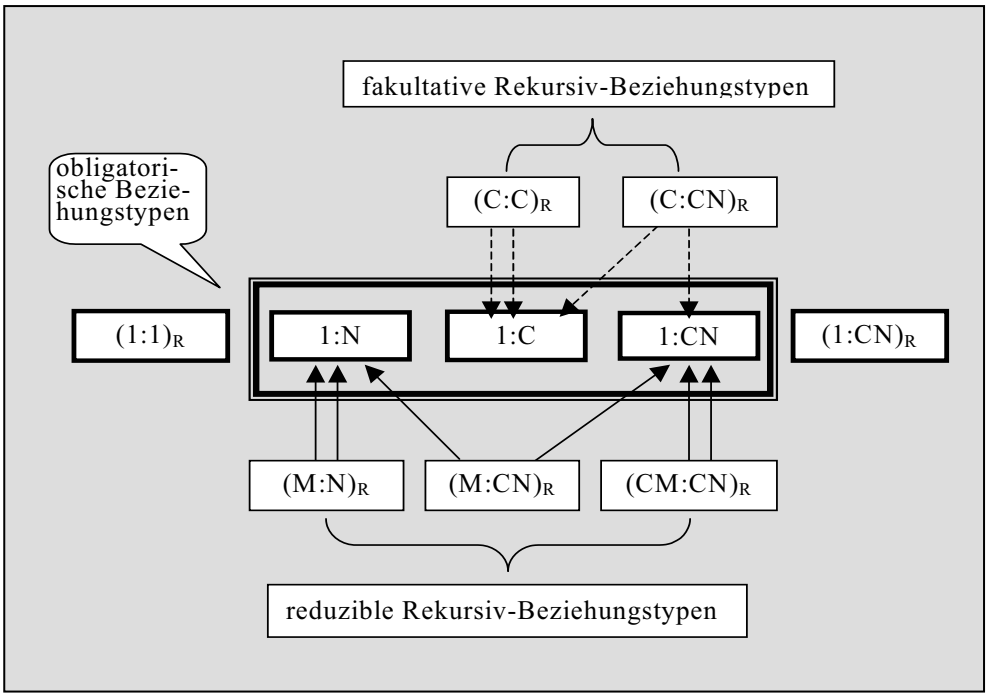


Abb. 5-50: Fakultative, obligatorische und reduzible Rekursiv-Beziehungstypen

In der Abbildung 5-50 wurden die 2 obligatorischen *Rekursiv-Beziehungstypen* $(1:1)_R$ und $(1:CN)_R$ sowie die 3 obligatorischen *dualen Beziehungstypen* (1:N, 1:C und 1:CN) stark umrandet. Von ihrer Darstellbarkeit hängt die Vollständigkeit der Repräsen-

tation der Rekursiv-Beziehungstypen ab. Im folgenden Abschnitt wird darum untersucht, wie es um die Darstellbarkeit dieser Beziehungstypen im relationalen Datenbank-Modell bestellt ist.

5.3.3 Die Repräsentationsmöglichkeit im relationalen Datenbank-Modell

Referenzprinzip für Rekursiv-Beziehungstypen

Im Abschnitt 5.3.2 wurde ausführlich dargelegt, in welcher Weise die obligatorischen Rekursiv-Beziehungstypen - also $(1:1)_R$ und $(1:CN)_R$ - sowie die fakultativen Rekursiv-Beziehungstypen - also $(C:C)_R$ und $(C:CN)_R$ - im relationalen Datenbank-Modell zu repräsentieren sind. Das Grundprinzip bestand darin, dass in die Objekttyp-Tabelle eine Kopie des *Primärschlüssels* - mit einer anderen Bezeichnung - als *Fremdschlüssel* aufgenommen wird. Im Empfänger-Datensatz wird der Sender *referiert*. Die Referenz wird dadurch realisiert, dass in der Empfänger-Zeile als Wert des Fremdschlüssels der Primärschlüsselwert der Sender-Zeile abgelegt wird. Dieses Prinzip der Referenz innerhalb ein und derselben Tabelle ist in der Abbildung 5-51 dargestellt.

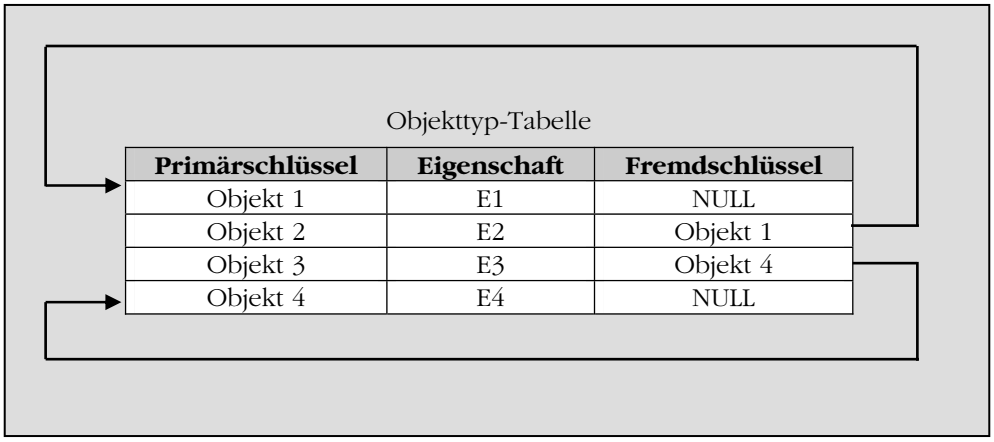


Abb. 5-51: Referenzprinzip zur Darstellung von Rekursiv-Beziehungstypen

Für den Primärschlüssel und für den Fremdschlüssel gelten wiederum Bedingungen, die *stets* erfüllt sein müssen: Für den Primärschlüssel ist es die Pflicht zur *Unikalität*, für den Fremdschlüssel ist es die Pflicht zur *referenziellen Integrität*.

Referenzpflicht, Eingabepflicht und Unikalität

In Analogie zum Abschnitt 5.2.4 betrachten wir drei weitere Bedingungen, die *wahlweise* an den Fremdschlüssel gestellt werden können: die *Referenzpflicht*, die *Eingabepflicht* und die Pflicht zur *Unikalität*.

Tab. 5-8: Kombinationen von Referenz-, Eingabepflicht und Unikalität für Rekursiv-Beziehungstypen

Fall	Referenz-pflicht	Eingabe-pflicht	Unikalität	Beziehungs-typ	repräsen-tierbar
1	nein	nein	nein	$(C:CN)_R$	ja
2	nein	nein	ja	$(C:C)_R$	ja
3	nein	ja	nein	$(1:CN)_R$	ja
4	nein	ja	ja	/	/
5	ja	nein	nein	/	/
6	ja	nein	ja	/	/
7	ja	ja	nein	/	/
8	ja	ja	ja	$(1:1)_R$	ja

Repräsentierbarkeit der Rekursiv-Beziehungstypen

Wir untersuchen nun, welche Rekursiv-Beziehungstypen sich mit Hilfe des Referenz-Prinzips der Abbildung 5-51 repräsentieren lassen.

- Da sich die Objekttyp-Tabelle in der 1. Normalform befinden muss, kann der Fremdschlüssel nur einen atomaren Wert annehmen, d.h. er kann höchstens auf *einen* Sender verweisen. Die Optionalität/Kardinalität der Beziehungstyp-Richtung „Sender←Empfänger“ kann also nur „1“ oder „C“ sein. Sie ist dann „1“, wenn für den Fremdschlüssel *Eingabepflicht* besteht, ansonsten ist sie „C“.
- Besteht für den Fremdschlüssel *Referenzpflicht*, so muss die Beziehungstyp-Richtung „Sender→Empfänger“ nicht-optional sein, ansonsten optional.
- Besteht für den Fremdschlüssel die Pflicht zur *Unikalität*, muss die Beziehungstyp-Richtung „Sender→Empfänger“ die Kardinalität „1“ haben, ansonsten „N“.

Diese Überlegungen sind in der Tabelle 5-8 zusammengefasst. Man sieht aus der Tabelle, dass alle obligatorischen und fakultativen Rekursiv-Beziehungstypen im relationalen Datenbank-Modell repräsentierbar sind.

Die drei *Rekursiv-Beziehungstypen mit multiplen Empfängern* - also $(M:N)_R$, $(M:CN)_R$ und $(CM:CN)_R$ - sind in der Tabelle 5-8 nicht enthalten, denn sie lassen sich mit dem einfachen Referenzprinzip der Abbildung 5-51 nicht repräsentieren. Zu einigen Kombinationen sind zusätzliche Bemerkungen erforderlich:

Fall 4: Wenn in jeder Zeile der Tabelle ein *eingabepflichtiger* Verweis *unikal* sein soll, muss zwangsläufig die *Referenzpflicht* erfüllt sein. Das ist leicht einzusehen: Wenn $\overline{\overline{A}}$ Objekte auf $\overline{\overline{A}}$ unterschiedliche Objekte verweisen, dann muss auf jedes Objekt ein Verweis zeigen – die Referenzpflicht ist also erfüllt. Die Kombination „Referenzpflicht = nein, Eingabepflicht = ja, Unikalität = ja“ ist für Rekursiv-Beziehungstypen nicht realistisch.

Fall 5: Da *keine Eingabepflicht* besteht, also wenigstens ein Objekt a keinen Verweis enthält, lässt sich die *Referenzpflicht* nicht erfüllen. Es müsste nämlich „in Vertretung für a“ ein anderes Objekt zwei Verweise enthalten, was eine Verletzung der 1. Normalform bedeuten würde. Die Kombination „Referenzpflicht = ja, Eingabepflicht = nein“ ist für Rekursiv-Beziehungstypen unmöglich.

Fall 6: Es gelten dieselben Überlegungen wie im Fall 5.

Fall 7: Wenn zwar *Eingabepflicht* besteht, aber wegen der *nicht geforderten Unikalität* wenigstens zwei Objekte einen übereinstimmenden Verweis enthalten, bleibt zumindest ein Objekt *„unreferenziert“*. Die Kombination „Referenzpflicht = ja, Eingabepflicht = ja, Unikalität = nein“ ist somit für Rekursiv-Beziehungstypen nicht real.

Fall 8: Aus der *Eingabepflicht* und der Pflicht zur *Unikalität* folgt zwangsläufig die *Referenzpflicht* (vgl. Fall 4). Da jedoch im relationalen Datenbank-Modell die Eingabepflicht und die Unikalität gefordert werden können, ist der 1:1-Rekursiv-Beziehungstyp - im Gegensatz zum dualen 1:1-Beziehungstyp - repräsentierbar.

Mit den Ergebnissen der Abbildung 5-50 und den Aussagen der Tabelle 5-8 haben wir zwei unterschiedliche Klassifizierungen der Rekursiv-Beziehungstypen vorgenommen:

- einerseits die Klassifizierung in *fakultative, obligatorische* und *reduzible* Beziehungstypen,
- andererseits die Klassifizierung in *repräsentierbare* und in *nicht-repräsentierbare* Beziehungstypen.

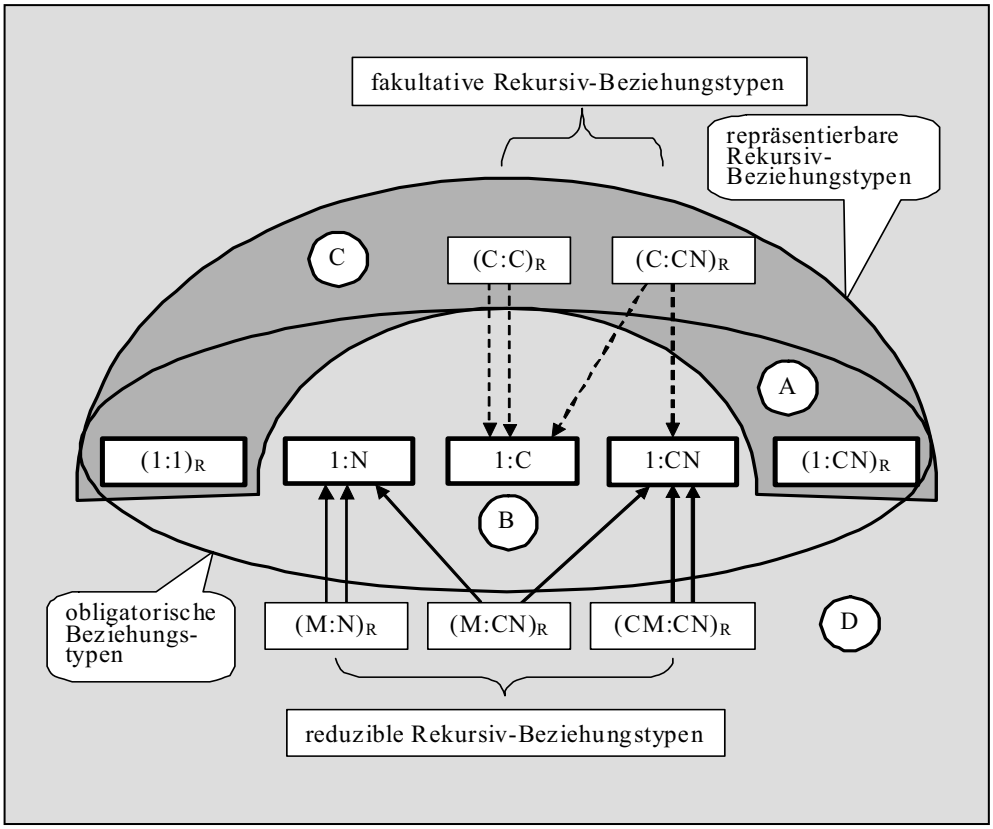


Abb. 5-52: Bereichsbildung für die Rekursiv-Beziehungstypen

Überschneidung der Klassifizierungen

In der Abbildung 5-52 sind die beiden Klassifizierungen dargestellt. Die Menge der obligatorischen Beziehungstypen wird von einer Ellipse umrandet. Die Menge der repräsentierbaren Rekursiv-Beziehungstypen befindet sich im dunklen hufeisen-förmigen

Gebiet. Die beiden Mengen überschneiden sich und bilden vier Bereiche A, B, C und D, die wir nacheinander betrachten wollen.

Bereich A: *Obligatorische und repräsentierbare Rekursiv-Beziehungstypen:* $(1:1)_R$ und $(1:CN)_R$. Diese beiden Rekursiv-Beziehungstypen werden unbedingt benötigt und lassen sich auch im relationalen Datenbank-Modell darstellen.

Bereich B: *Obligatorische duale Beziehungstypen:* 1:N, 1:C und 1:CN. Die beiden dualen Beziehungstypen 1:N und 1:CN werden unbedingt benötigt, um die Rekursiv-Beziehungstypen $(M:N)_R$, $(M:CN)_R$ und $(CM:CN)_R$ repräsentieren zu können. Wie wir im Abschnitt 5.2.4 gesehen haben, kann aber der 1:N-Beziehungstyp nicht im relationalen Datenbank-Modell dargestellt werden. Der duale 1:C-Beziehungstyp wäre für die Rekursiv-Beziehungstypen nicht unbedingt erforderlich, weil die Rekursiv-Beziehungstypen $(C:C)_R$ und $(C:CN)_R$ auch ohne duale Beziehungstypen repräsentierbar sind. Seine Darstellbarkeit im relationalen Datenbank-Modell gibt uns aber höhere Flexibilität, um eventuell den Speicherplatz-Bedarf für die Daten reduzieren zu können.

Bereich C: *Fakultative, aber repräsentierbare Rekursiv-Beziehungstypen:* $(C:C)_R$ und $(C:CN)_R$. Diese Rekursiv-Beziehungstypen werden nicht unbedingt benötigt, weil sie sich auf die dualen Beziehungstypen 1:C bzw. 1:CN reduzieren lassen. Es ist aber von Vorteil, dass sie sich dennoch im relationalen Datenbank-Modell repräsentieren lassen, weil sie unter Umständen zu einer Reduzierung des Speicherplatzbedarfs und/oder zu einer Verbesserung der Performance führen.

Bereich D: *Reduzible Rekursiv-Beziehungstypen:* $(M:N)_R$, $(M:CN)_R$ und $(CM:CN)_R$. Diese drei Rekursiv-Beziehungstypen müssen auf die dualen Beziehungstypen 1:N und 1:CN zurückgeführt werden. Da sich aber der 1:N-Beziehungstyp im relationalen Datenbank-Modell nicht darstellen lässt, liegt hier

ein gewichtiger Schwachpunkt dieses Datenbank-Modells vor. Nur der CM:CN-Rekursiv-Beziehungstyp lässt sich „sinn-erhaltend“ repräsentieren, bei den anderen beiden ist ein Semantikverlust in Kauf zu nehmen.

Die Tabelle 5-9 zeigt das im Einzelnen.

Tab. 5-9: Repräsentierbarkeit der Rekursiv-Beziehungstypen des Bereichs D

Rekursiv-Beziehungstyp	Durch duale Beziehungstypen repräsentierbar?	Repräsentation mit Semantikverlust
$(M:N)_R$	1:N + N:1 nicht möglich, da 1:N fehlt	$(CM:CN)_R$
$(M:CN)_R$	1:CN + N:1 nicht möglich, da 1:N fehlt	$(CM:CN)_R$
$(CM:CN)_R$	1:CN + CN:1 möglich	/

Fazit:

Gesamtzahl der interessierenden Rekursiv-Beziehungstypen (ohne „Spiegelbilder“ an der Diagonale): ***7***

Davon:

Direkt im relationalen Datenbank-Modell repräsentierbar: ***4***

Durch Umwandlung in einen Koppel-Objekttyp repräsentierbar: ***1***

Nur mit Semantikverlust repräsentierbar: ***2***

In der Abbildung 5-53 sind die Verhältnisse noch einmal in graphischer Form dargestellt.

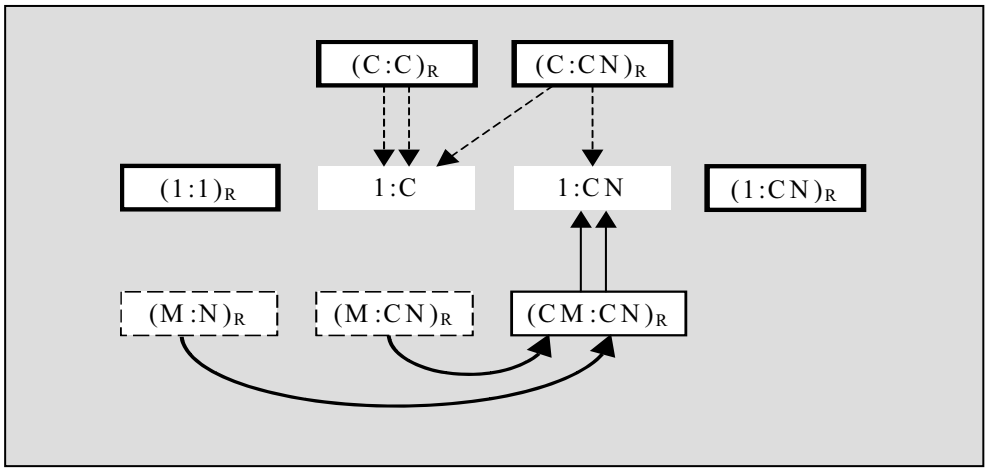


Abb. 5-53: Repräsentation der Rekursiv-Beziehungstypen im relationalen Datenbank-Modell

Die 4 direkt repräsentierbaren Rekursiv-Beziehungstypen *(1:1)*$_R$, *(C:C)*$_R$, *(C:CN)*$_R$ und *(1:CN)*$_R$ sind stark umrandet. Die wahlweise Reduktion der fakultativen Rekursiv-Beziehungstypen (C:C)$_R$ und (C:CN)$_R$ auf die dualen 1:C- bzw. 1:CN-Beziehungstypen ist durch gestrichelte Pfeile angedeutet. Der ohne Semantikverlust darstellbare reduzible Rekursiv-Beziehungstyp *(CM:CN)*$_R$ besitzt einen dünnen Rand, wobei durch ausgezogene Pfeile seine Zurückführung auf den dualen 1:CN-Beziehungstyp dargestellt ist. Die restlichen 2 reduziblen Rekursiv-Beziehungstypen *(M:N)*$_R$ und *(M:CN)*$_R$ haben einen gestrichelten Rand. Sie können lediglich mit Semantikverlust wie ein CM:CN-Rekursiv-Beziehungstyp repräsentiert werden, wie dies durch die gebogenen Pfeile gezeigt wird.

6 Die Generalprobe: Aufgaben zum Datenbankentwurf

Liebe Leser! Wir haben das Ziel unserer Wanderung durch das Gebiet des Datenbankentwurfs erreicht. Nun ist es an der Zeit zu prüfen, ob wir den Weg auch mit Gewinn zurückgelegt haben. In diesem Kapitel werden deshalb fünf praktische Aufgaben zum Datenbankentwurf gestellt. Die Einordnung dieses Kapitels in den Kontext des Lehrbuchs zeigt die Abbildung 6-1.

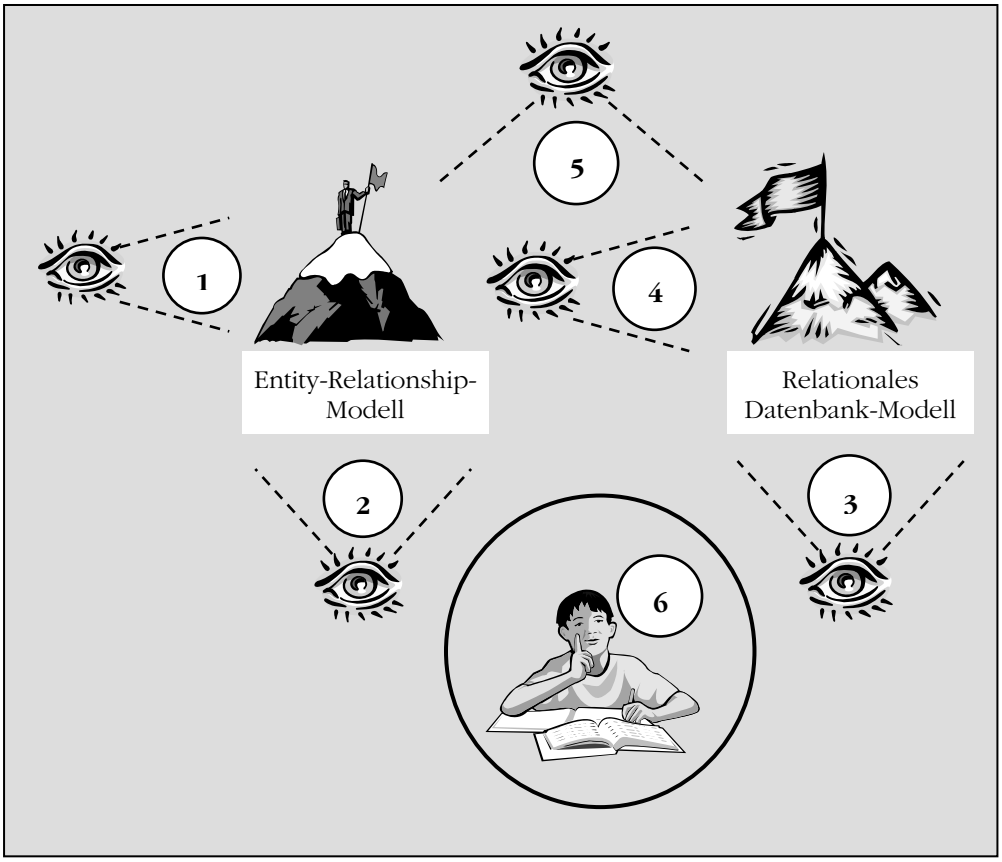


Abb. 6-1: Gegenstand des Kapitels 6

Für jeden der fünf Gegenstandsbereiche sind zwei Aufgaben zu erfüllen:

1. Entwicklung eines *konzeptionellen Datenmodells* mit Hilfe der graphischen Sprache des Entity-Relationship-Modells.
2. Überführung des konzeptionellen Datenmodells in das *logische Datenschema* für das relationale Datenbank-Managementsystem *„Access“*.

Die Lösungen der Aufgaben sind angegeben und werden ausführlich kommentiert.

Somit können Sie, verehrte Leser, die Korrektheit Ihrer Ergebnisse überprüfen, ehe sie darangehen, Datenbanken für Ihren eigenen Gegenstandsbereich zu entwerfen.

6.1 Eine Autovermietung

6.1.1 Beschreibung des Gegenstandsbereichs

Ein Autovermietungs-Unternehmen hat in Murkelstadt (unikaler Stadtcode „MUR“, 12 000 Einwohner) noch keine Vermietstation, betreibt aber beispielsweise in Gigantow (Stadtcode „GIG“, 675 000 Einwohner) bereits fünf Vermietstationen. Die Vermietstationen, die in ein und derselben Stadt liegen, werden durch eine laufende Nummer voneinander unterschieden. Die Vermietstation in Gigantow mit der laufenden Nummer 3 befindet sich in der Riesenstraße 100 (PLZ 98765) und hat 13 Mitarbeiter.

Das Unternehmen vermietet Autos grundsätzlich nur nach vorheriger Reservierung. Bei einer Reservierung, die stets von genau einer Vermietstation vorgenommen wird, kann der Kunde, über den der Name und die Anschrift gespeichert werden und der durch seine Führerschein-Nummer eindeutig gekennzeichnet ist, eine Autokategorie auswählen. Die Autokategorie „A“ hat beispielsweise einen Grundtarif von 45 Euro/Tag und einen Kilometerpreis von 0,25 Euro/km. Sie umfasst unter anderen die Autotypen „Seat Arosa“ (Benzinverbrauch 5,0 l/100 km) und „Fiat Cinquecento“ (Benzinverbrauch 4,5 l/100 km). Zu jedem Autotyp werden die Extraausstattungen gespeichert, die für diesen Autotyp möglich sind. Für einen „Seat Arosa“ ist die Extraausstattung „Schiebedach“ beispielsweise für einen Aufpreis von 9 Euro/Tag erhältlich. Für den „Fiat Cinquecento“ beträgt der Aufpreis für die Extraausstattung „Schiebedach“ 10 Euro/Tag. Die Vermietstation vergibt für die von ihr vorgenommenen Reservierungen jeweils

eine laufende Nummer. Für jede Reservierung wird das gewünschte Anfangs- und Enddatum festgehalten.

Eine Reservierung kann zu einem Mietvertrag führen, der bei Übergabe eines Autos abgeschlossen wird. Ein Mietvertrag ist durch die zugehörige Reservierung eindeutig gekennzeichnet. Zum Mietvertrag wird der Kilometerstand bei der Übergabe und später der Kilometerstand bei der Rückgabe festgehalten. Außerdem wird festgelegt, welches konkrete Auto Gegenstand des Mietvertrags wird. Da es häufig vorkommt, dass Reservierungen nicht zu einem Mietvertrag führen, sollen lediglich für die tatsächlich zustande gekommenen Mietverträge die Mietvertragsdaten gespeichert werden.

Die Autos des Unternehmens werden durch ihr polizeiliches Kennzeichen unterschieden. Zu jedem Auto muss ersichtlich sein, welchen Kilometerstand es hat (diese Angabe wird jeweils bei einer Rückgabe aktualisiert), welche Farbe es hat, zu welchem Autotyp es gehört und von welcher Vermietstation es gerade verwaltet wird. Außerdem wird festgehalten, über welche Extraausstattungen es verfügt.

Weiterhin ist zu beachten:

- Gerade erst eingerichtete Vermietstationen verwalten noch keine Autos und haben noch keine Reservierungen vorgenommen.
- Ein Kunde wird erst dann gespeichert, wenn er die erste Reservierung vorgenommen hat.
- Es ist möglich, dass eine Autokategorie, die stets mindestens drei Autotypen umfasst, noch bei keiner Reservierung gewünscht wurde.
- Ein Autotyp wird in genau eine Autokategorie eingeordnet. Zu einem bereits gespeicherten Autotyp kann das Unternehmen auch noch kein Auto besitzen. Einen Autotyp ohne Extraausstattungen gibt es zwar nicht, aber ein konkretes Auto kann keine der möglichen Extraausstattungen aufweisen. Es kann vorkommen, dass eine mögliche Extraausstattung bei keinem der Autos des Unternehmens vorkommt.
- Ein gerade erst gekauftes Autos wurden noch nicht vermietet. Es wird dann im Laufe der Zeit vielen Mietverträgen zugeordnet. Ein Auto wird zu jedem Zeitpunkt von genau einer Vermietstation verwaltet.

6.1.2 Konzeptionelles Datenmodell

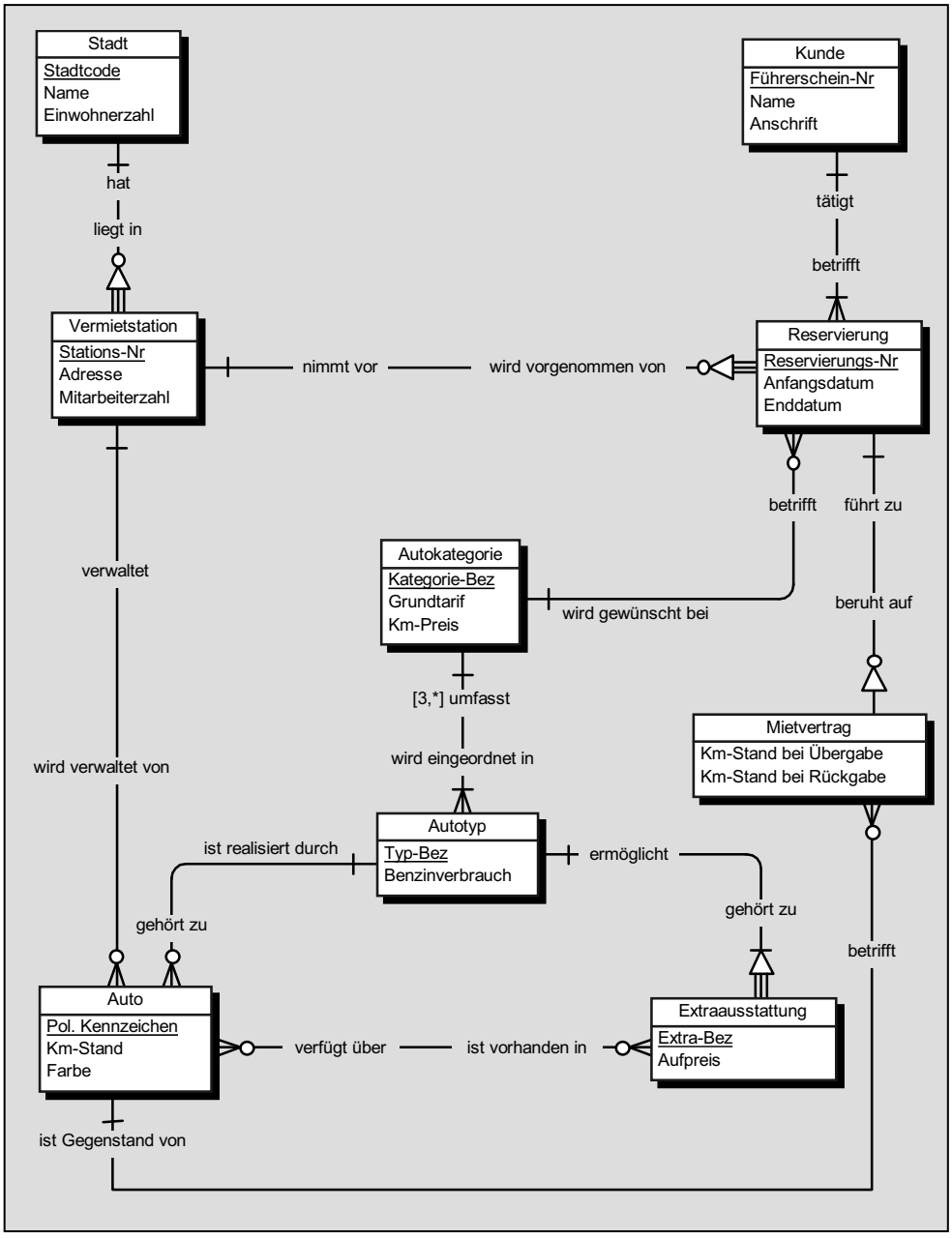


Abb. 6-2: Konzeptionelles Datenmodell für die Autovermietung

Bemerkungen:

- Da die Vermietstationen, die in ein und derselben Stadt liegen, durch eine laufende Nummer voneinander unterschieden werden, erfolgt die Identifizierung des Objekttyps „Vermietstation“ durch eine Kombination aus der Beziehungstyp-Richtung „Vermietstation *liegt in* Stadt“ und der Eigenschaft „Stations-Nr“.
- Die Vermietstation vergibt für die von ihr vorgenommenen Reservierungen eine laufende Nummer. Deshalb erfolgt die Identifizierung des Objekttyps „Reservierung“ durch eine Kombination aus der Beziehungstyp-Richtung „Reservierung *wird vorgenommen von* Vermietstation“ und der Eigenschaft „Reservierungs-Nr“.
- Die Beziehungstyp-Richtung „Kunde *tätigt* Reservierung“ ist nicht-optional, da ein Kunde erst dann gespeichert wird, wenn er die erste Reservierung vorgenommen hat.
- Im vorliegenden Fall ist der 1:C-Beziehungstyp zwischen „Reservierung“ und „Mietvertrag“ gerechtfertigt. Die Mietvertragsdaten stellen zusätzliche Angaben für jene (wenigen) Reservierungen dar, bei denen es zu einem Mietvertrag gekommen ist. Würde man beide Objekttypen zu einem einzigen Objekttyp vereinigen, wären die Eigenschaften „Km-Stand bei Übergabe“ und „Km-Stand bei Rückgabe“ häufig mit der NULL-Marke belegt.
- Da ein Mietvertrag durch die zugehörige Reservierung eindeutig gekennzeichnet ist, wird der Objekttyp „Mietvertrag“ allein durch die Beziehungstyp-Richtung „Mietvertrag *beruht auf* Reservierung“ identifiziert.
- Die Eigenschaft „Extra-Bez“ des Objekttyps „Extraausstattung“ kann für verschiedene Autotypen denselben Wert haben (beispielsweise „Schiebedach“). Deshalb wird der Objekttyp „Extraausstattung“ durch eine Kombination aus der Beziehungstyp-Richtung „Extraausstattung *gehört zu* Autotyp“ und der Eigenschaft „Extra-Bez“ identifiziert.

6.1.3 Transformation in das logische Datenschema

Zunächst werden die Objekttypen in die entsprechenden Tabellen transformiert (Transformationsregel T01). Danach erfolgt die Transformation der Beziehungstyp-Richtungen, die als identifizierende Elemente für die „schwachen" Objekttypen verwendet wurden (Transformationsregel T02). Im Ergebnis dieser beiden Transformationen ergeben sich die folgenden vorläufigen Tabellen-Typbeschreibungen:

Stadt(**Stadtcode**,Name,Einwohnerzahl)

Vermietstation(⇑**Stadtcode**⇑**+Stations-Nr**,Adresse,Mitarbeiterzahl)

Reservierung(⇑**Stadtcode+Stations-Nr**⇑**+Reservierungs-Nr**,
Anfangsdatum,Enddatum)

Kunde(**Führerschein-Nr**,Name,Anschrift)

Autokategorie(**Kategorie-Bez**,Grundtarif,Km-Preis)

Autotyp(**Typ-Bez**,Benzinverbrauch)

Extraausstattung(⇑**Typ-Bez**⇑**+Extra-Bez**,Aufpreis)

Auto(**Pol. Kennzeichen**,Km-Stand,Farbe)

Mietvertrag(⇑**Stadtcode+Stations-Nr+Reservierungs-Nr**⇑,
Km-Stand bei Übergabe,Km-Stand bei Rückgabe)

Dabei wurden folgende Primärschlüssel als Fremdschlüssel in die Primärschlüssel der „schwachen" Objekttypen aufgenommen:

Primärschlüssel	des Objekttyps	wird aufgenommen in	Primärschlüssel von
Stadtcode	Stadt	⇒	Vermietstation
Stadtcode+Stations-Nr	Vermietstation	⇒	Reservierung
Typ-Bez	Autotyp	⇒	Extraausstattung
Stadtcode+Stations-Nr+ Reservierungs-Nr	Reservierung	⇒	Mietvertrag

Im nächsten Schritt werden die dualen Beziehungstypen in das relationale Datenbank-Modell transformiert (Transformationsregeln T03 bis T12). Die veränderten Tabellen-Typbeschreibungen sind jeweils durch einen Stern gekennzeichnet:

Stadt(**Stadtcode**,Name,Einwohnerzahl)

Vermietstation(⇑**Stadtcode**⇑**+Stations-Nr**,Adresse,Mitarbeiterzahl)

* Reservierung(⇑**Stadtcode+Stations-Nr**⇑**+Reservierungs-Nr**,
⇑Führerschein-Nr⇑,⇑Kategorie-Bez⇑,Anfangsdatum,Enddatum)

Kunde(**Führerschein-Nr**,Name,Anschrift)

Autokategorie(**Kategorie-Bez**,Grundtarif,Km-Preis)

* Autotyp(**Typ-Bez**,⇑Kategorie-Bez⇑,Benzinverbrauch)

Extraausstattung(⇑**Typ-Bez**⇑**+Extra-Bez**,Aufpreis)

* Auto(**Pol. Kennzeichen**,⇑Typ-Bez⇑,⇑Stadtcode+Stations-Nr⇑,
Km-Stand,Farbe)

* Ausstattung(⇑**Pol. Kennzeichen**⇑**+**⇑**Typ-Bez+Extra-Bez**⇑)

* Mietvertrag(⇑**Stadtcode+Stations-Nr+Reservierungs-Nr**⇑,
⇑Pol. Kennzeichen⇑,
Km-Stand bei Übergabe,Km-Stand bei Rückgabe)

Bemerkungen:

- Der 1:N-Beziehungstyp zwischen „Kunde“ und „Reservierung“ kann nur als 1:CN-Beziehungstyp repräsentiert werden (Transformationsregel T09). Es kann also nicht gesichert werden, dass ein gespeicherter Kunde wenigstens eine Reservierung getätigt hat.
- Analog dazu kann der 1:N-Beziehungstyp zwischen „Autokategorie“ und „Autotyp“ nur als 1:CN-Beziehungstyp repräsentiert werden (Transformationsregel T09). Die zusätzlich geforderte Kardinalitäts-Beschränkung kann erst recht nicht dargestellt werden. Durch diese Tabellen-Typbeschreibung lässt sich nicht erzwingen, dass eine Autokategorie wenigstens drei Autotypen umfasst.

- In gleicher Weise lässt sich der 1:N-Beziehungstyp zwischen „Autotyp“ und „Extraausstattung“ nur als 1:CN-Beziehungstyp repräsentieren (Transformationsregel T09). Die Tabellen-Typbeschreibung lässt es somit – allerdings im Widerspruch zur Realität - zu, dass es zu einem Autotyp keine Extraausstattung gibt.
- Der 1:C-Beziehungstyp zwischen „Reservierung“ und „Mietvertrag“ muss gemäß der Transformationsregel T06 (1:C-Beziehungstyp mit oft realisierter Optionalität) transformiert werden, da es häufig vorkommt, dass Reservierungen nicht zu einem Mietvertrag führen. Damit muss der Primärschlüssel von „Reservierung“ in der Tabelle „Mietvertrag“ als eingabepflichtig und unikal vereinbart werden. Das ist er aber schon, weil der „schwache“ Objekttyp „Mietvertrag“ durch die „Reservierung“ identifiziert wird.
- Der CM:CN-Beziehungstyp zwischen „Auto“ und „Extraausstattung“ wird gemäß Transformationsregel T12 durch eine Koppeltabelle repräsentiert, die die Bezeichnung “Ausstattung“ erhalten hat.

6.1.4 „Physisches Datenmodell“ des PowerDesigner

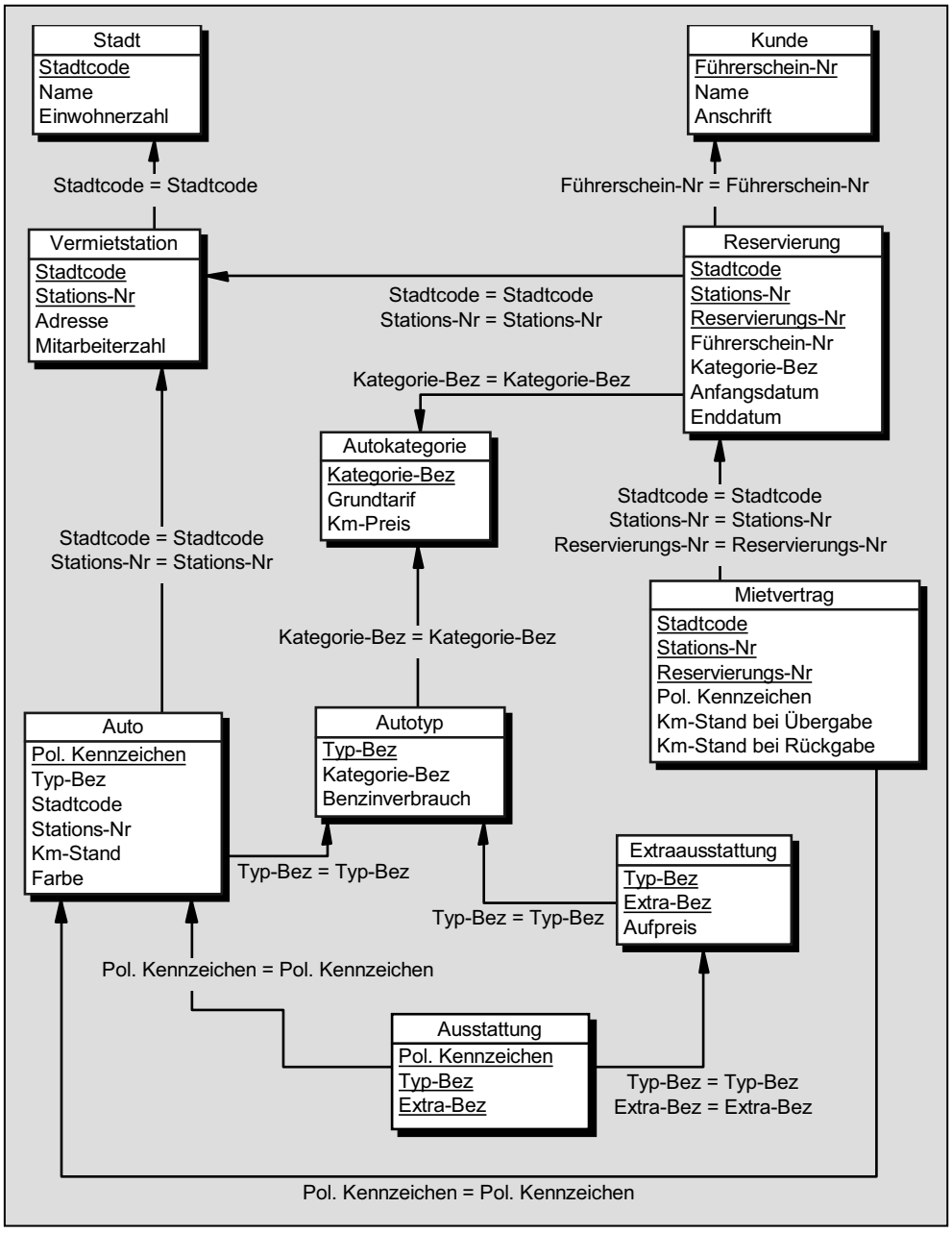


Abb. 6-3: „Physisches Datenmodell“ für die Autovermietung

6.1.5 Datenbank-Struktur für Access

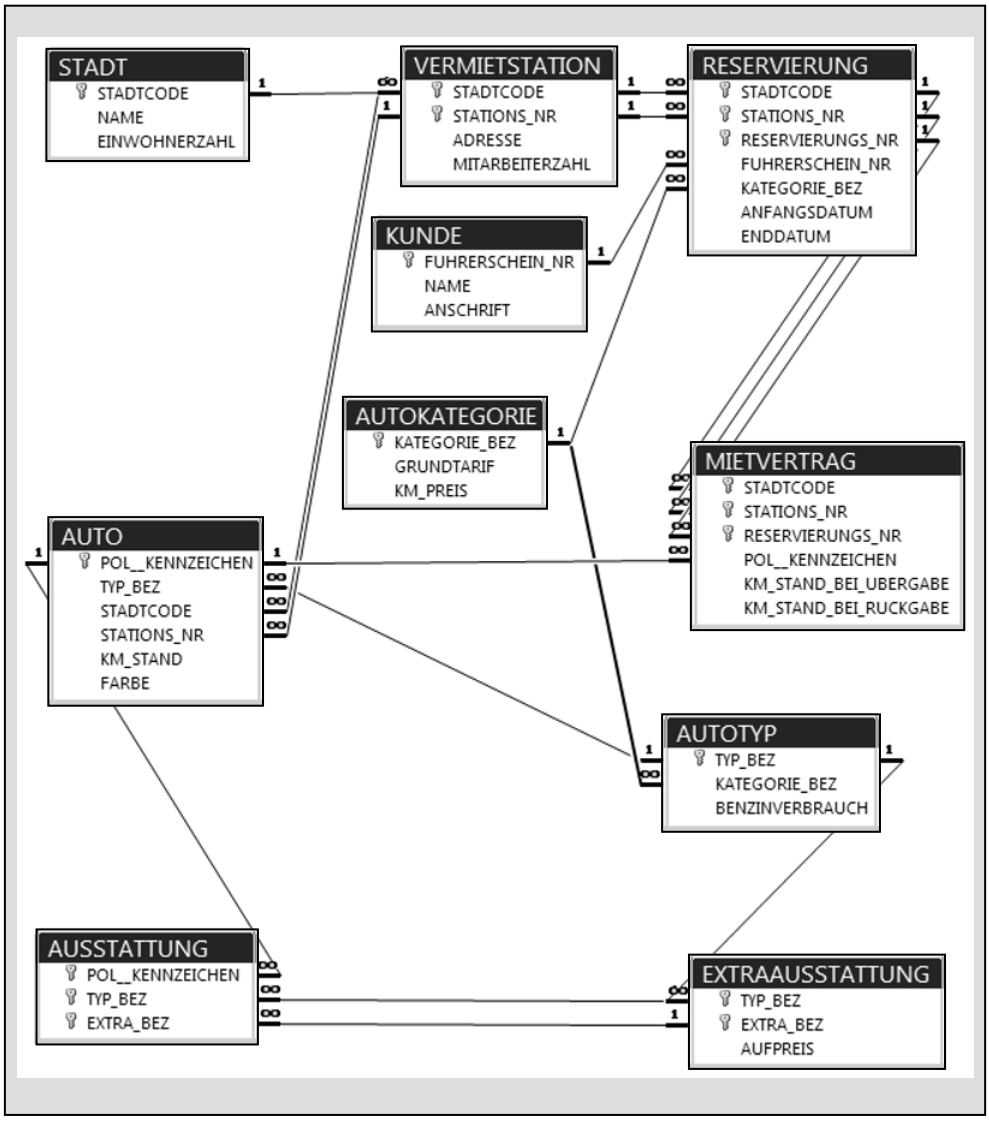


Abb. 6-4: Datenbank-Struktur für die Autovermietung

6.2 Eine Fluggesellschaft

6.2.1 Beschreibung des Gegenstandsbereichs

Mehrere Fluggesellschaften möchten unter Verwendung einer gemeinsamen Datenbank-Anwendung die Ausnutzung ihrer „täglichen Linienflugangebote“[30] kontrollieren.

Jede der beteiligten Fluggesellschaften stellt ein oder mehrere tägliche Linienflugangebote bereit. Die täglichen Linienflugangebote ein und derselben Fluggesellschaft werden durch eine laufende Nummer voneinander unterschieden. Jedes tägliche Linienflugangebot hat eine geplante Abflugzeit, genau einen Flughafen als Ausgangspunkt und genau einen Flughafen als Zielpunkt. Die Fluggesellschaft mit der Kurzbezeichnung FA (Fantasia Airlines, Hauptsitz in Utopia City) stellt unter anderen das tägliche Linienflugangebot mit der laufenden Nummer 333 bereit, das vom Flughafen mit dem Code TXL (Berlin-Tegel, 2 Start/Landepisten) beginnt und auf dem Flughafen mit dem Code UTC (Utopia City, 1 Start/Landepiste) endet. Die geplante Abflugzeit ist 7:03 Uhr.

Im Informationssystem sollen Informationen über die tatsächlich durchgeführten Flüge gespeichert werden. Jeder dieser Flüge entspricht genau einem Linienflugangebot, wobei sich die Flüge, die demselben Linienflugangebot entsprechen, durch den Flugtag unterscheiden. Zu jedem Flug muss außerdem bekannt sein, zu welcher Zeit er tatsächlich begonnen hat und mit welchem Flugzeug er durchgeführt wurde. Der Flug, der am 1.8.2009 entsprechend dem oben beschriebenen Linienflugangebot durchgeführt wurde, hatte beispielsweise als Abflugzeit 7:44 Uhr und erfolgte mit einem Flugzeug vom Typ „Air wing“, das die Identitätsnummer 4711 trägt.
Für jeden Flug werden ein oder mehrere Flugscheine ausgegeben. Ein Flugschein kann den Status „open“ haben, kann also noch keinem Flug zugeordnet sein. Will niemand mit diesem Flug mitfliegen, findet der Flug erst gar nicht statt. Jeder Flugschein gehört zu einem Flugticket. In einem Flugticket können mehrere Flugscheine zusammengefasst sein (falls eine Reise aus mehreren Flügen besteht). Die einzelnen Flugscheine innerhalb

[30] Ein „tägliches Linienflugangebot“ ist ein regelmäßiger Linienflug, den eine Fluggesellschaft für jeden Tag anbietet. Er führt von einem Flughafen A zu einem Flughafen B und beginnt stets zur selben Zeit (zumindest ist es so geplant).

eines Flugtickets werden durch eine laufende Nummer unterschieden und durch die Flugklasse näher bestimmt.

Die Flugtickets müssen von einem autorisierten Reisebüro ausgestellt werden. Beispielsweise enthält das Flugticket, das vom Reisebüro „Kurze Ferien“ (15230 Frankfurt/Oder, Bahnhofsplatz 2, 120 m^2 Verkaufsfläche, 2 Mitarbeiter) unter der vom Reisebüro vergebenen laufenden Nummer 112233 am 4.7.2009 zum Preis von 123,45 Euro verkauft wurde, nicht nur einen Flugschein für den angegebenen Flug in der Flugklasse BC, sondern auch noch weitere Flugscheine. Dieses Ticket wurde an den Passagier Vera Cruz (15234 Frankfurt/Oder, Akazienweg 1, weiblich) verkauft, die schon zuvor oftmals geflogen ist.

Weiterhin ist zu beachten:

- Fluggesellschaften, die keine täglichen Linienflugangebote anbieten und die keine Flugzeuge besitzen, werden nicht berücksichtigt.
- In Vorbereitung auf eventuelle Erweiterungen der Linienflugangebote werden bereits Flughäfen erfasst, die bisher noch für kein Linienflugangebot der Fluggesellschaften Start- oder Lande-Flaghafen sind.
- Zu einem neuen Linienflugangebot wurden noch keine Flüge durchgeführt.
- Die Identitätsnummer eines Flugzeuges gilt nur innerhalb der Fluggesellschaft, der es gehört. Neue Flugzeuge haben noch keine Flüge absolviert.
- Innerhalb eines Postleitzahl-Bezirks kann es nicht mehrere namensgleiche Reisebüros geben. Ein Reisebüro wird erfasst, ehe es das erste Flugticket verkauft hat.
- Unabhängig vom konkreten Reisebüro werden bei einer Verkaufsfläche von 120 m^2 stets 2 Mitarbeiter beschäftigt. Die bereits vorgesehene Verkaufsfläche von 250 m^2 (mit 5 Mitarbeitern) gibt es noch in keinem Reisebüro.

6.2.2 Konzeptionelles Datenmodell

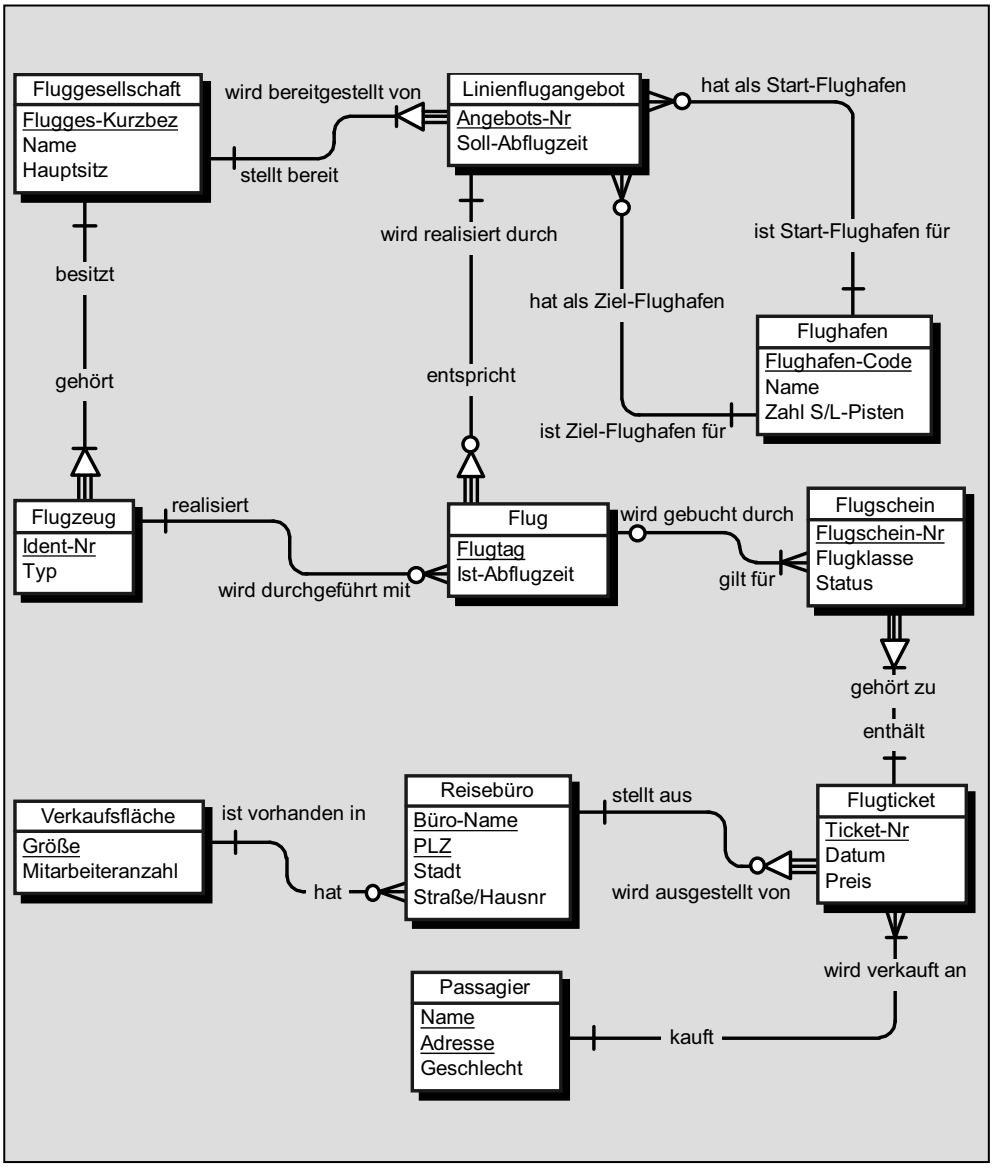


Abb. 6-5: Konzeptionelles Datenmodell für die Fluggesellschaft

Bemerkungen:

- Da sich die Linienflugangebote, die von ein und derselben Fluggesellschaft bereitstellt werden, durch eine laufende Nummer voneinander unterscheiden, erfolgt die Identifizierung des Objekttyps „Linienflugangebot“ durch eine Kombination aus der Beziehungstyp-Richtung „Linienflugangebot *wird bereitgestellt von* Fluggesellschaft“ und der Eigenschaft „Angebots-Nr“.
- Ein Flughafen ist nicht an sich „Start-Flughafen“ oder „Ziel-Flughafen“. Dies ist eine Frage der sachlogischen Zusammenhänge zwischen dem Linienflugangebot und den Flughäfen. Es müssen also zwei parallele Beziehungstypen zwischen „Linienflugangebot“ und „Flughafen“ modelliert werden.[31]
- Die durchgeführten Flüge, die demselben Linienflugangebot entsprechen, unterscheiden sich voneinander durch den Flugtag. Deshalb wird der Objekttyp „Flug“ durch eine Kombination aus der Beziehungstyp-Richtung „Flug *entspricht* Linienflugangebot“ und der Eigenschaft „Flugtag“ identifiziert.
- Die Speicherung der tatsächlichen Abflugzeit („Ist-Abflugzeit“) zusätzlich zur geplanten Abflugzeit („Soll-Abflugzeit“) bringt keine Redundanz mit sich. Jeder, der schon einmal geflogen ist, weiß, dass diese beiden Zeiten miteinander nur in losem Zusammenhang stehen.
- Da die Ident-Nummer eines Flugzeugs von der Fluggesellschaft vergeben wird, der es gehört, erfolgt die Identifizierung des Objekttyps „Flugzeug“ durch eine Kombination aus der Beziehungstyp-Richtung „Flugzeug *gehört* Fluggesellschaft“ und der Eigenschaft „Ident-Nr“.
- Es werden nur die tatsächlich durchgeführten Flüge gespeichert. Deshalb muss es zu jedem Flug wenigstens einen Flugschein geben (keine Optionalität!).
- Der Flugschein wird über seine Beziehung zum Flugticket und durch seine Eigenschaft „Flugschein-Nr“ identifiziert.

---

[31] Das Datenmodell schließt nicht aus, dass ein Linienflugangebot denselben Flughafen als Start und Ziel hat (Rundflug).

- Das Flugticket wird über seine Beziehung zum Reisebüro und durch seine Eigenschaft „Ticket-Nr“ identifiziert.
- Es werden nur Passagiere gespeichert, die wenigstens ein Flugticket gekauft haben. Somit ist die Beziehungstyp-Richtung „Passagier *kauft* Flugticket“ nicht-optional.
- Da die Mitarbeiteranzahl von der Verkaufsfläche funktional abhängt, muss diese Abhängigkeit im Rahmen eines gesonderten Objekttyps „Verkaufsfläche“ dargestellt werden, nicht innerhalb des Objekttyps „Reisebüro“, denn das wäre eine Verletzung der 3. Normalform.

6.2.3 Transformation in das logische Datenschema

Zunächst werden die Objekttypen in die entsprechenden Tabellen transformiert (Transformationsregel T01). Danach erfolgt die Transformation der Beziehungstyp-Richtungen, die als identifizierende Elemente für die „schwachen“ Objekttypen verwendet wurden (Transformationsregel T02). Im Ergebnis dieser beiden Transformationen ergeben sich die folgenden vorläufigen Tabellen-Typbeschreibungen:

Fluggesellschaft(**Flugges-Kurzbez**,Name,Hauptsitz)

Linienflugangebot(⇑**Flugges-Kurzbez**⇑**+Angebots-Nr**,Soll-Abflugzeit)

Flughafen(**Flughafen-Code**,Name,Zahl S/L-Pisten)

Flug(⇑**Flugges-Kurzbez+Angebots-Nr**⇑**+Flugtag**,Ist-Abflugzeit)

Flugzeug(⇑**Flugges-Kurzbez**⇑**+Ident-Nr**,Typ)

Flugschein(⇑**Büro-Name+PLZ+Ticket-Nr**⇑**+Flugschein-Nr**,
Flugklasse,Status)

Flugticket(⇑**Büro-Name+PLZ**⇑**+Ticket-Nr**,Datum,Preis)

Reisebüro(**Büro-Name+PLZ**,Stadt,Straße/Hausnr)

Passagier(**Name+Adresse**,Geschlecht)

Verkaufsfläche(**Größe**,Mitarbeiteranzahl)

Dabei wurden folgende Primärschlüssel als Fremdschlüssel in die Primärschlüssel der „schwachen“ Objekttypen aufgenommen:

Primärschlüssel	**des Objekttyps**	**wird aufgenommen in**	**Primärschlüssel von**
Flugges-Kurzbez	Fluggesellschaft	⇒	Linienflugangebot
Flugges-Kurzbez+Angebots-Nr	Linienflugangebot	⇒	Flug
Flugges-Kurzbez	Fluggesellschaft	⇒	Flugzeug
Büro-Name+PLZ	Reisebüro	⇒	Flugticket
Büro-Name+PLZ+Ticket-Nr	Flugticket	⇒	Flugschein

Im nächsten Schritt werden die dualen Beziehungstypen in das relationale Datenbank-Modell transformiert (Transformationsregeln T03 bis T12). Die veränderten Tabellen-Typbeschreibungen sind jeweils durch einen Stern gekennzeichnet:

Fluggesellschaft(**Flugges-Kurzbez**,Name,Hauptsitz)

* Linienflugangebot(⇑**Flugges-Kurzbez**⇑**+Angebots-Nr**,
⇑Start-Flughafen⇑,⇑Ziel-Flughafen⇑,Soll-Abflugzeit)

Flughafen(**Flughafen-Code**,Name,Zahl S/L-Pisten)

* Flug(⇑**Flugges-Kurzbez+Angebots-Nr**⇑**+Flugtag**,
⇑ [Flugges-Kurzbez+] Ident-Nr⇑,Ist-Abflugzeit)

Flugzeug(⇑**Flugges-Kurzbez**⇑**+Ident-Nr**,Typ)

* Flugschein(⇑**Büro-Name+PLZ+Ticket-Nr**⇑**+Flugschein-Nr**,
⇑*Flugges-Kurzbez+Angebots-Nr+Flugtag*⇑,Flugklasse,Status)

* Flugticket(⇑**Büro-Name+PLZ**⇑**+Ticket-Nr**,
⇑Passagier-Name+Adresse⇑,Datum,Preis)

* Reisebüro(**Büro-Name+PLZ**,⇑Größe⇑,Stadt,Straße/Hausnr)

Passagier(**Name+Adresse**,Geschlecht)

Verkaufsfläche(**Größe**,Mitarbeiteranzahl)

Bemerkungen:

- Die folgenden 1:N-Beziehungstypen können lediglich unter Semantikverlust als 1:CN-Beziehungstypen, also gemäß der Transformationsregel T09, repräsentiert werden:

Fluggesellschaft	←→	Linienflugangebot
Fluggesellschaft	←→	Flugzeug
Flugticket	←→	Flugschein
Passagier	←→	Flugticket

  Es kann also durch die Tabellen-Typbeschreibungen nicht gesichert werden, dass

  - eine Fluggesellschaft mindestens ein Linienflugangebot bereitstellt,
  - eine Fluggesellschaft wenigstens ein Flugzeug besitzt,
  - ein Flugticket mindestens einen Flugschein enthält,
  - ein Passagier zumindest ein Flugticket kauft.

- Der C:N-Beziehungstyp zwischen „Flug“ und „Flugschein“ lässt sich nur als C:CN-Beziehungstyp repräsentieren. Die Tabellen-Typbeschreibung kann also nicht garantieren, dass es zu einem durchgeführten Flug mindestens einen Flugschein gibt (ein Flugzeug kann somit auch ohne Passagiere fliegen!).

  Für den C:CN-Beziehungstyp stehen zwei Transformationsregeln zur Verfügung:

  a) Transformationsregel T10 (die meisten Flugscheine sind für einen konkreten Flug ausgestellt),
  b) Transformationsregel T11 (die meisten Flugscheine haben den Status „open“).

  Es wurde unterstellt, dass der Fall a) vorliegt. Darum wurde keine Koppel-Tabelle eingeführt. Stattdessen wurde in die Tabelle „Flugschein“ ein *nicht-eingabepflichtiger* Verweis auf den „Flug“ aufgenommen (deshalb der Kursivdruck!).

- Da der Primärschlüssel der Tabelle „Flughafen“ zweimal als Fremdschlüssel in die Tabelle „Linienflugangebot“ aufgenommen werden muss, wurden die sich sonst doppelnde Benennung „Flughafen-Code“ durch „Start-Flughafen“ bzw. durch „Ziel-Flughafen“ unterschieden.
- In der Tabelle „Flug“ müsste der Primärschlüssel der Fluggesellschaft (**Flugges-Kurzbez**) eigentlich zweimal auftauchen:

  a) als Bestandteil des Primärschlüssels:
  ⇑Flugges-Kurzbez+Angebots-Nr⇑+Flugtag

  b) als Bestandteil des Verweises auf das Flugzeug:
  ⇑Flugges-Kurzbez+Ident-Nr⇑

  Geht man davon aus, dass stets die Fluggesellschaft, die das Linienflugangebot bereitstellt, identisch ist mit derjenigen, die das Flugzeug besitzt, muss ihr Primärschlüssel natürlich nicht zweimal gespeichert werden. Er wurde deshalb beim Verweis auf das Flugzeug in eckige Klammern gesetzt:

  ⇑ [Flugges-Kurzbez+] Ident-Nr⇑

  Als zusätzliches Attribut wird lediglich die „Ident-Nr“ aufgenommen. Dieses bildet dann gemeinsam mit dem Attribut „Flugges-Kurzbez“ *aus dem Primärschlüssel* den Verweis auf das Flugzeug.
- In der Tabelle „Flugticket“ wurde der ursprüngliche Primärschlüssel **Name+Adresse** der Tabelle „Passagier“ wegen der besseren Verständlichkeit in ⇑Passagier-Name+Adresse⇑ umgewandelt.

6.2.4 „Physisches Datenmodell“ des PowerDesigner

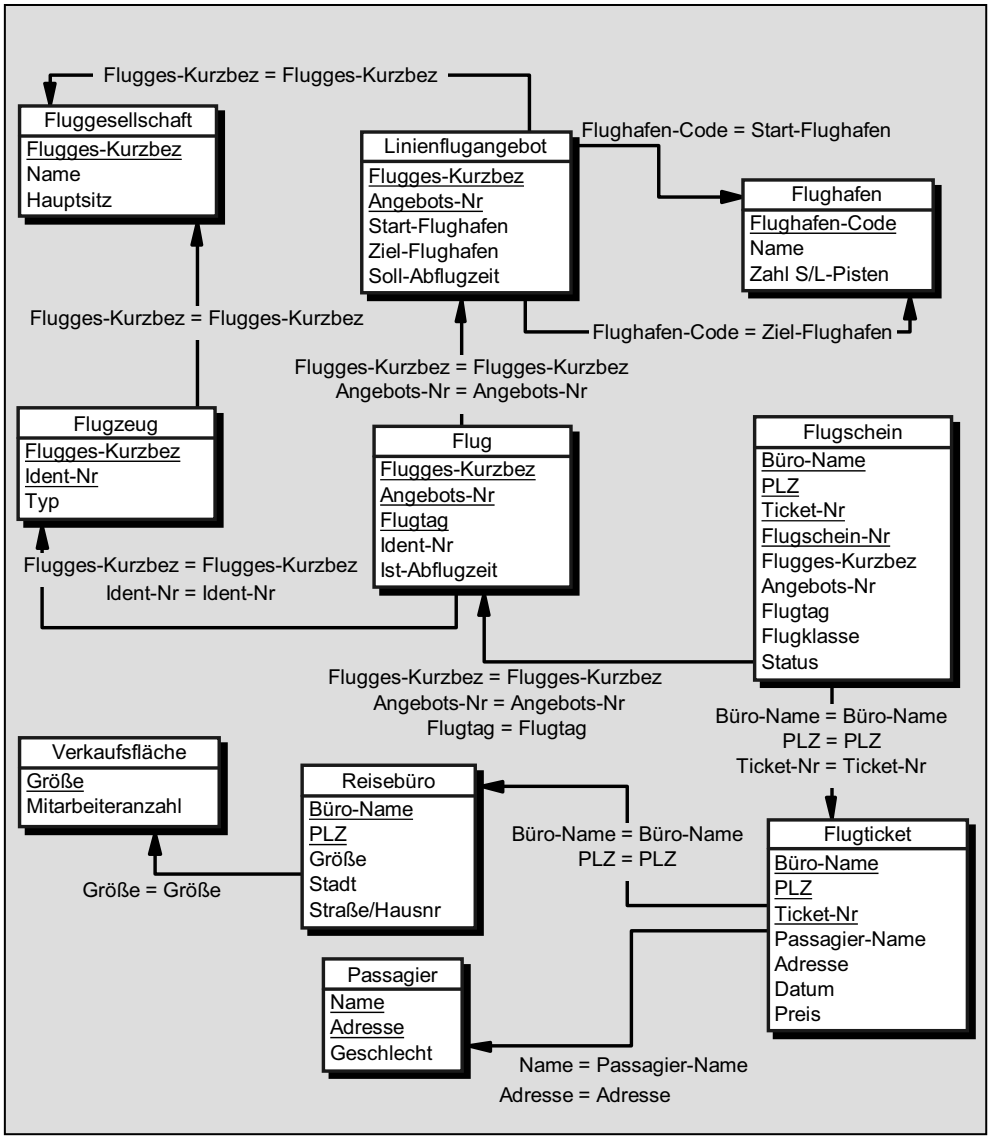


Abb. 6-6: „Physisches Datenmodell“ für die Fluggesellschaft

6.2.5 Datenbank-Struktur für Access

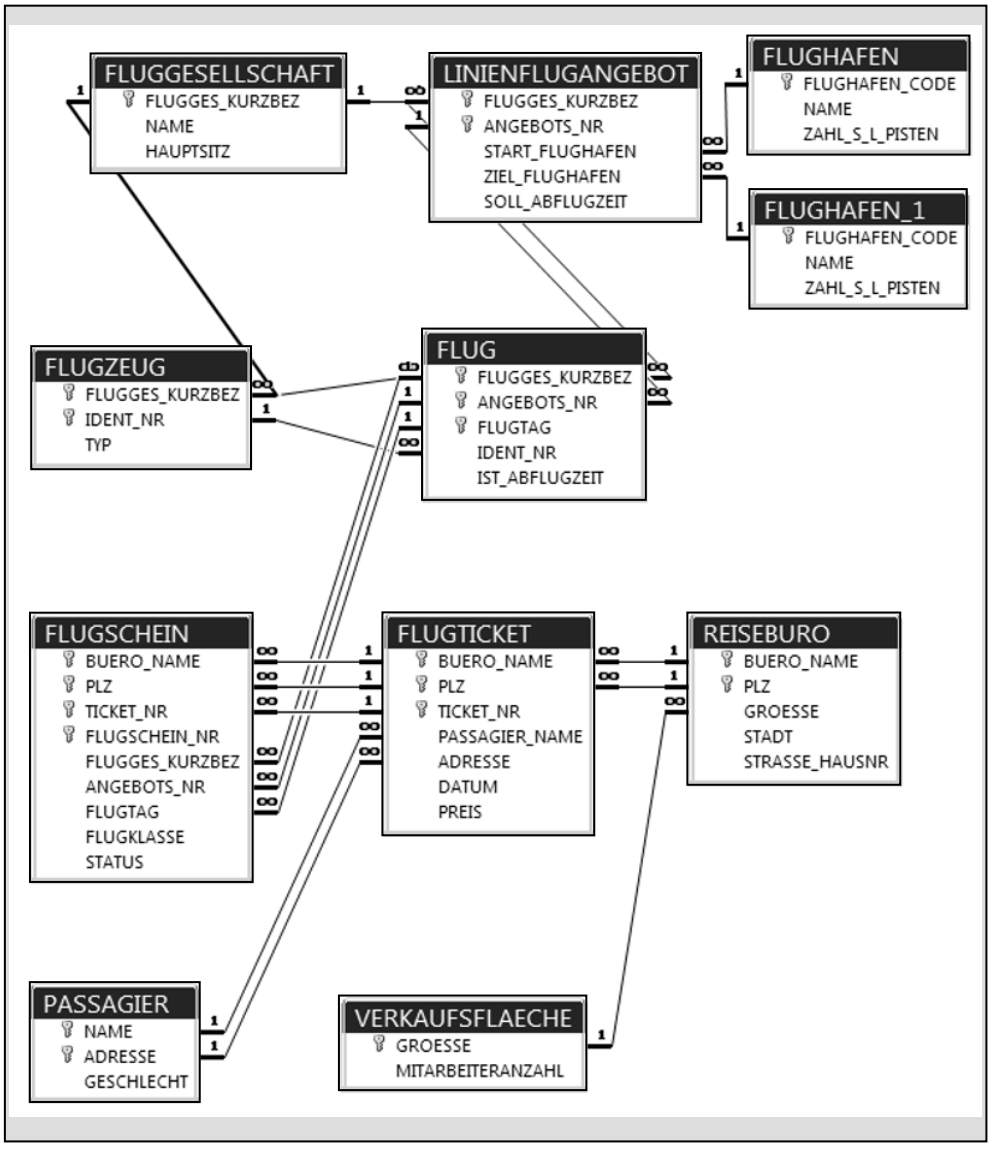


Abb. 6-7: Datenbank-Struktur für die Fluggesellschaft

6.3 Ein Schnellbahn-Unternehmen

6.3.1 Beschreibung des Gegenstandsbereichs

Das Schnellbahn-Unternehmen der Stadt Ökopolis verfügt über moderne Wagen zur Personenbeförderung, die durch eine unikale Wagennummer voneinander unterschieden werden. Der Wagen mit der Nummer W1234 (Baujahr 2006) hat beispielsweise 50 Sitzplätze und ist bisher 22.200 km gefahren. Mehrere Wagen (mindestens zwei) gehören gemeinsam zu einem Zug, der durch seine Zugnummer identifiziert wird. Dabei ist jeweils nur die aktuelle Zugzusammenstellung von Interesse. Eine Anzahl von Wagen wird zudem stets in der Reserve gehalten. Jedem Zug ist eine Funk-Frequenz zugeordnet, über die die Funkabfertigung auf den Bahnsteigen erfolgt. Außerdem ist für jeden Zug festgelegt, auf welchem Bahnhof er nach Betriebsschluss abgestellt wird. Auf dem Bahnhof „Waldruhe“ (unikaler Bahnhofscode WR, 3 Abstellgleise) wird beispielsweise neben anderen Zügen stets der Zug Z111 abgestellt. Es gibt auch Bahnhöfe, die kein einziges Abstellgleis haben.

Das Unternehmen betreibt ein Gitternetz von Linien, die jeweils in Nord/Süd- bzw. West/Ost-Ausrichtung verlaufen. Die Linie D (Länge 17 km) hat beispielsweise als Grenzpunkte die Bahnhöfe „Nordend“ und „Südwiesen“. Der Bahnhof „Südwiesen“ ist auch Grenzpunkt für die Linie G. Der Bahnhof „Rathaus“ ist für keine Linie Grenzpunkt.

Auf einer Linie werden am Tag mehrere Fahrten durchgeführt, die jeweils vom einen Grenzpunkt der Linie zum entgegengesetzten Grenzpunkt führen. Eine einzelne Fahrt ist gekennzeichnet durch die Linie, den Tag, die Anfangszeit der Fahrt und durch die Fahrtrichtung (NS, SN, WO, bzw. OW). Für jede Fahrt soll außerdem ersichtlich sein, mit welchem Zug sie erfolgte und wer der Fahrer war. Beispielsweise wurde die Fahrt, die auf der Linie D am 24.08.2009 um 5.30 Uhr begann und in Nord-Süd-Richtung (NS) erfolgte, mit dem Zug Z111 durchgeführt, den Claus Thaler (Personalnummer 4711, geb. 24.08.63) gefahren hat - und das ausgerechnet an seinem Geburtstag!
Auf der Grundlage umfangreicher Sicherheitsvorschriften erfolgt eine regelmäßige technische Überprüfung der Wagen des Unternehmens. So wird ein Wagen erst dann in den Bestand aufgenommen und gespeichert, wenn die erste technische Überprüfung erfolgt ist. An einer solchen Überprüfung nehmen zwei zu-

gelassene Prüfer und ein Fahrer teil. Die verschiedenen technischen Überprüfungen ein und desselben Wagens werden durch den jeweiligen Prüftag voneinander unterschieden. Beispielsweise wurde der Wagen W1234 zum ersten Mal am 06.06.2006 überprüft. An dieser Prüfung waren der Fahrer Claus Thaler sowie die PrüferInnen Rudi Ment (Personalnummer 9999) und Lotte Rie (Personalnummer 8888) beteiligt. Dabei wurde die Prüfnote 2 vergeben.

Weiterhin ist zu beachten:

- Ein gerade erst zusammengestellter Zug hat noch keine Fahrten absolviert.
- Die Linie N wird erst morgen eingeweiht. Auf ihr wurden noch keine Fahrten durchgeführt. Fahrten bleiben auch dann noch gespeichert, wenn der Zug, mit dem sie durchgeführt wurden, schon wieder aufgelöst und in der Datenbank gelöscht wurde. Wird dagegen eine Linie gelöscht, werden auch die auf ihr durchgeführten Fahrten mitgelöscht.
- Neu eingestellte Fahrer werden erst geschult, ehe sie Fahrten durchführen. Natürlich nehmen sie in dieser Zeit auch keine technischen Überprüfungen vor. Verlässt ein Fahrer oder ein Prüfer das Unternehmen, bleiben deren Daten weiterhin gespeichert.
- Um Personalengpässe zu vermeiden, werden stets einige Prüfer als „Reserve“ gespeichert, die noch keine technische Überprüfung durchgeführt haben.

6.3.2 Konzeptionelles Datenmodell

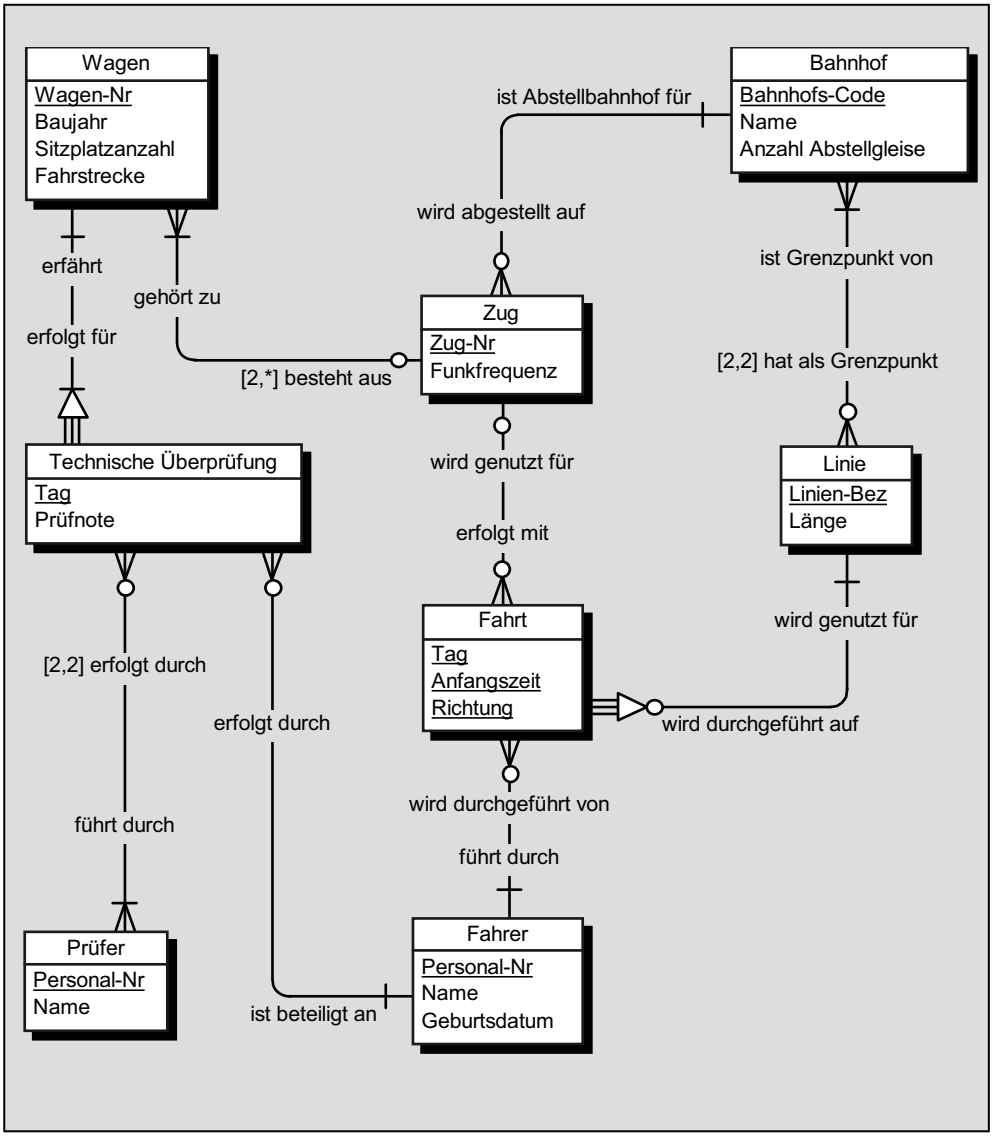


Abb. 6-8: Konzeptionelles Datenmodell für das Schnellbahn-Unternehmen

Bemerkungen:

- Da nur die aktuelle Zugzusammenstellung von Interesse ist, kann ein Wagen zu höchstens einem Zug gehören (Kardinalität 1). Andererseits werden Wagen in der Reserve gehalten: sie gehören dann zu keinem Zug (Optionalität).
- Eine einzelne Fahrt ist gekennzeichnet durch die Linie, den Tag, die Anfangszeit der Fahrt und durch die Fahrtrichtung (NS, SN, WO, bzw. OW). Die Identifizierung des Objekttyps „Fahrt" erfolgt somit durch eine Kombination aus der Beziehungstyp-Richtung „Fahrt *wird durchgeführt auf* Linie" und aus den Eigenschaften „Tag", „Anfangszeit" und „Richtung".
- Fahrten bleiben auch dann noch gespeichert, wenn der Zug, mit dem sie durchgeführt wurden, schon wieder aufgelöst und in der Datenbank gelöscht wurde. Somit muss es erlaubt sein, dass eine gespeicherte Fahrt mit keinem Zug in Beziehung steht. Die Beziehungstyp-Richtung „Fahrt *erfolgt mit* Zug" muss also optional sein.
- Da das Auflösen einer Linie das Löschen der auf ihr durchgeführten Fahrten nach sich zieht, muss eine gespeicherte Fahrt stets mit einer Linie in Beziehung stehen. Die Beziehungstyp-Richtung „Fahrt *wird durchgeführt auf* Linie" ist deshalb nicht-optional.
- Die Daten eines Mitarbeiters bleiben auch dann gespeichert, wenn er das Unternehmen verlässt. Deshalb können die Beziehungstyp-Richtungen „Fahrt *wird durchgeführt von* Fahrer" und „Technische Überprüfung *erfolgt durch* Fahrer" als nicht-optional angegeben werden.
- Da ein Wagen erst dann in den Bestand aufgenommen und gespeichert wird, wenn die erste technische Überprüfung erfolgt ist, muss die Beziehungstyp-Richtung „Wagen *erfährt* Technische Überprüfung" nicht-optional sein.
- Die Technische Überprüfung wird über ihre Beziehung zum Wagen und durch den Prüftag identifiziert.

6.3.3 Transformation in das logische Datenschema

Zunächst werden die Objekttypen in die entsprechenden Tabellen transformiert (Transformationsregel T01). Danach erfolgt die Transformation der Beziehungstyp-Richtungen, die als identifizierende Elemente für die „schwachen" Objekttypen verwendet wurden (Transformationsregel T02). Im Ergebnis dieser beiden Transformationen ergeben sich die folgenden vorläufigen Tabellen-Typbeschreibungen:

Wagen(**Wagen-Nr**,Baujahr,Sitzplatzanzahl,Fahrstrecke)

Zug(**Zug-Nr**,Funkfrequenz)

Bahnhof(**Bahnhofs-Cod**e,Name,Anzahl Abstellgleise)

Linie(**Linien-Bez**,Länge)

Fahrt(⇑**Linien-Bez**⇑**+Tag+Anfangszeit+Richtung**)

Fahrer(**Personal-Nr**,Name,Geburtsdatum)

Technische Überprüfung(⇑**Wagen-Nr**⇑**+Tag**,Prüfnote)

Prüfer(**Personal-Nr**,Name)

Dabei wurden folgende Primärschlüssel als Fremdschlüssel in die Primärschlüssel der „schwachen" Objekttypen aufgenommen:

Primärschlüssel	des Objekttyps	wird aufgenommen in	Primärschlüssel von
Linien-Bez	Linie	⇒	Fahrt
Wagen-Nr	Wagen	⇒	Technische Überprüfung

Im nächsten Schritt werden die dualen Beziehungstypen in das relationale Datenbank-Modell transformiert (Transformationsregeln T03 bis T12). Die veränderten Tabellen-Typbeschreibungen sind jeweils durch einen Stern gekennzeichnet:

* Wagen(**Wagen-Nr**, *⇑Zug-Nr⇑*,Baujahr,Sitzplatzanzahl,Fahrstrecke)

* Zug(**Zug-Nr**,⇑Bahnhofs-Code⇑,Funkfrequenz)

Bahnhof(**Bahnhofs-Code**,Name,Anzahl Abstellgleise)

* Linienbegrenzung(**⇑Bahnhofs-Code⇑+⇑Linien-Bez⇑**)

Linie(**Linien-Bez**,Länge)

* Fahrt(**⇑Linien-Bez⇑+Tag+Anfangszeit+Richtung**,⇑Personal-Nr⇑, *⇑Zug-Nr⇑*)

Fahrer(**Personal-Nr**,Name,Geburtsdatum)

* Technische Überprüfung(**⇑Wagen-Nr⇑+Tag**,⇑Personal-Nr⇑,Prüfnote)

* Prüferaktivität(**⇑Wagen-Nr+Tag⇑+⇑Personal-Nr⇑**)

Prüfer(**Personal-Nr**,Name)

Bemerkungen:

- Der 1:N-Beziehungstyp zwischen „Wagen“ und „Technische Überprüfung“ kann nur als 1:CN-Beziehungstyp repräsentiert werden (Transformationsregel T09). Es kann also – entgegen den Sicherheitsbestimmungen - nicht garantiert werden, dass ein gespeicherter Wagen wenigstens eine Technische Überprüfung erfahren hat.
- Der C:N-Beziehungstyp zwischen „Zug“ und „Wagen“ lässt sich nur als C:CN-Beziehungstyp repräsentieren. Die Kardinalitäts-Beschränkung ist erst recht nicht darstellbar. Mit der Tabellen-Typbeschreibung kann also nicht verhindert werden, dass ein Zug gespeichert wird, der keinen einzigen oder nur *einen* Wagen hat.

  Für den C:CN-Beziehungstyp stehen zwei Transformationsregeln zur Verfügung:

  a) Transformationsregel T10 (die meisten Wagen gehören zu einem Zug),
  b) Transformationsregel T11 (die meisten Wagen werden in der Reserve gehalten).

  Sicherlich kann man den Fall a) annehmen. Deshalb wurde keine Koppel-Tabelle eingeführt. Stattdessen wurde in die Tabelle „Wagen“ ein *nicht-eingabepflichtiger* Verweis auf den „Zug“ aufgenommen (deshalb der Kursivdruck!).

- Für den C:CN-Beziehungstyp zwischen „Zug“ und „Fahrt“ stehen wiederum zwei Transformationsregeln zur Verfügung:

  a) Transformationsregel T10 (den meisten Fahrten ist ein gespeicherter Zug zugeordnet),

  b) Transformationsregel T11 (die meisten Fahrten wurden mit einem Zug durchgeführt, der im Speicher schon gelöscht wurde)

  Es wurde unterstellt, dass die Zug-Zusammenstellungen relativ stabil sind und dass gegebenenfalls Züge, die zwar physisch nicht mehr existieren, in der Datenbank trotzdem noch eine Weile gespeichert bleiben. Deshalb wurde keine Koppel-Tabelle eingeführt. Stattdessen wurde in die Tabelle „Fahrt“ ein *nicht-eingabepflichtiger* Verweis auf den „Zug“ aufgenommen (deshalb der Kursivdruck!).

- Die folgenden M:CN-Beziehungstypen lassen sich nur unter Semantikverlust als CM:CN-Beziehungstypen - also gemäß der Transformationsregel T12 - repräsentieren:

  Bahnhof ←→ Linie

  Prüfer ←→ Technische Überprüfung

  Die Kardinalitäts-Beschränkungen lassen sich erst recht nicht durchsetzen. Es kann also durch die Typbeschreibungen der Koppel-Tabellen „Linienbegrenzung“ bzw. „Prüferaktivität“ nicht gesichert werden, dass

  - eine Linie genau 2 Bahnhöfe als Grenzpunkte hat (es kann beispielsweise geschehen, dass für eine Linie gar kein, nur ein Bahnhof oder sogar drei Bahnhöfe als Grenzpunkte angegeben werden),
  - eine Technische Überprüfung durch genau 2 Prüfer erfolgt (es kann vorkommen, dass ihr kein Prüfer zugeordnet wird oder dass ihr beispielsweise drei Prüfer zugeordnet werden).

6.3.4 „Physisches Datenmodell“ des PowerDesigner

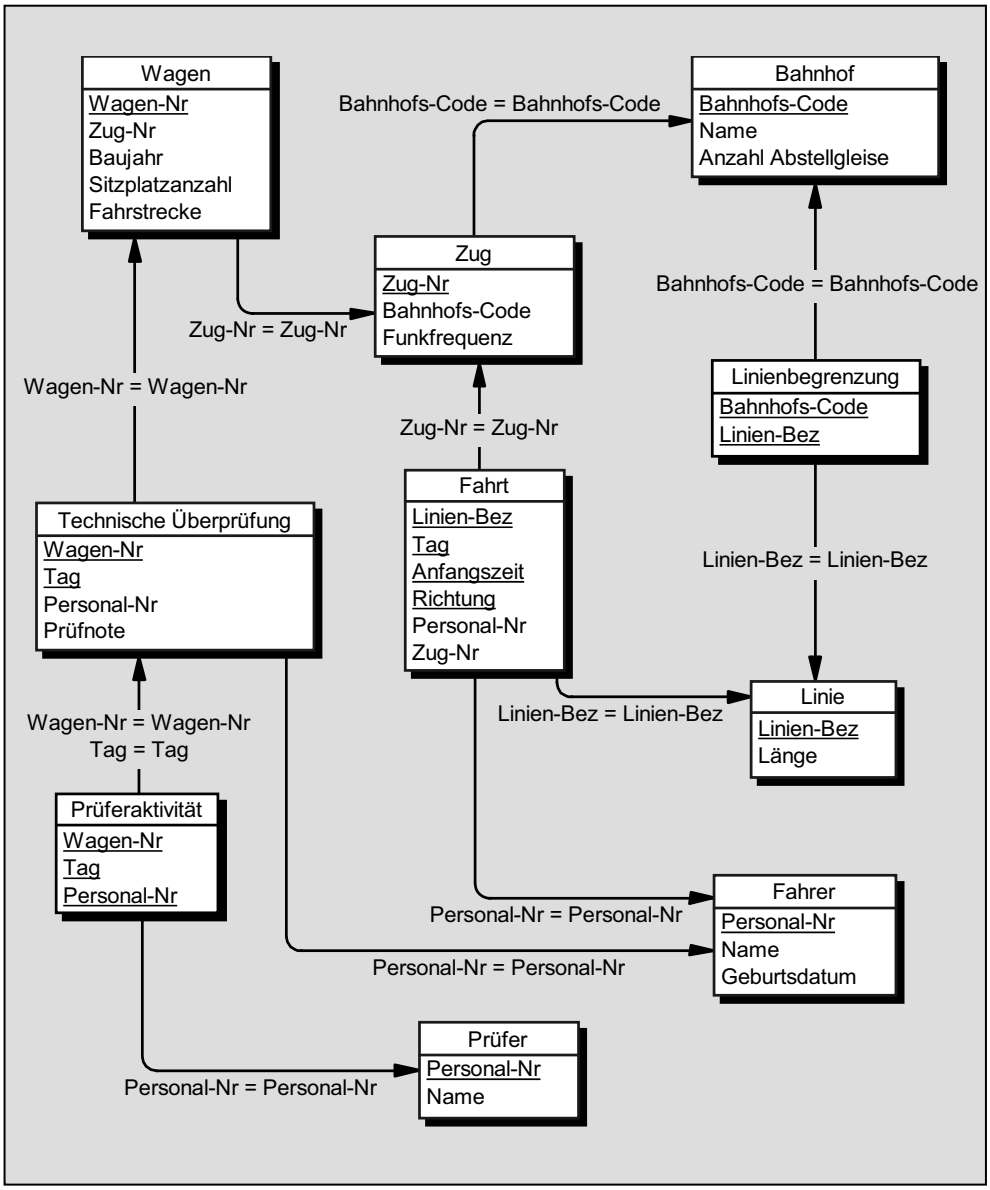


Abb. 6-9: „Physisches Datenmodell“ für das Schnellbahn-Unternehmen

6.3.5 Datenbank-Struktur für Access

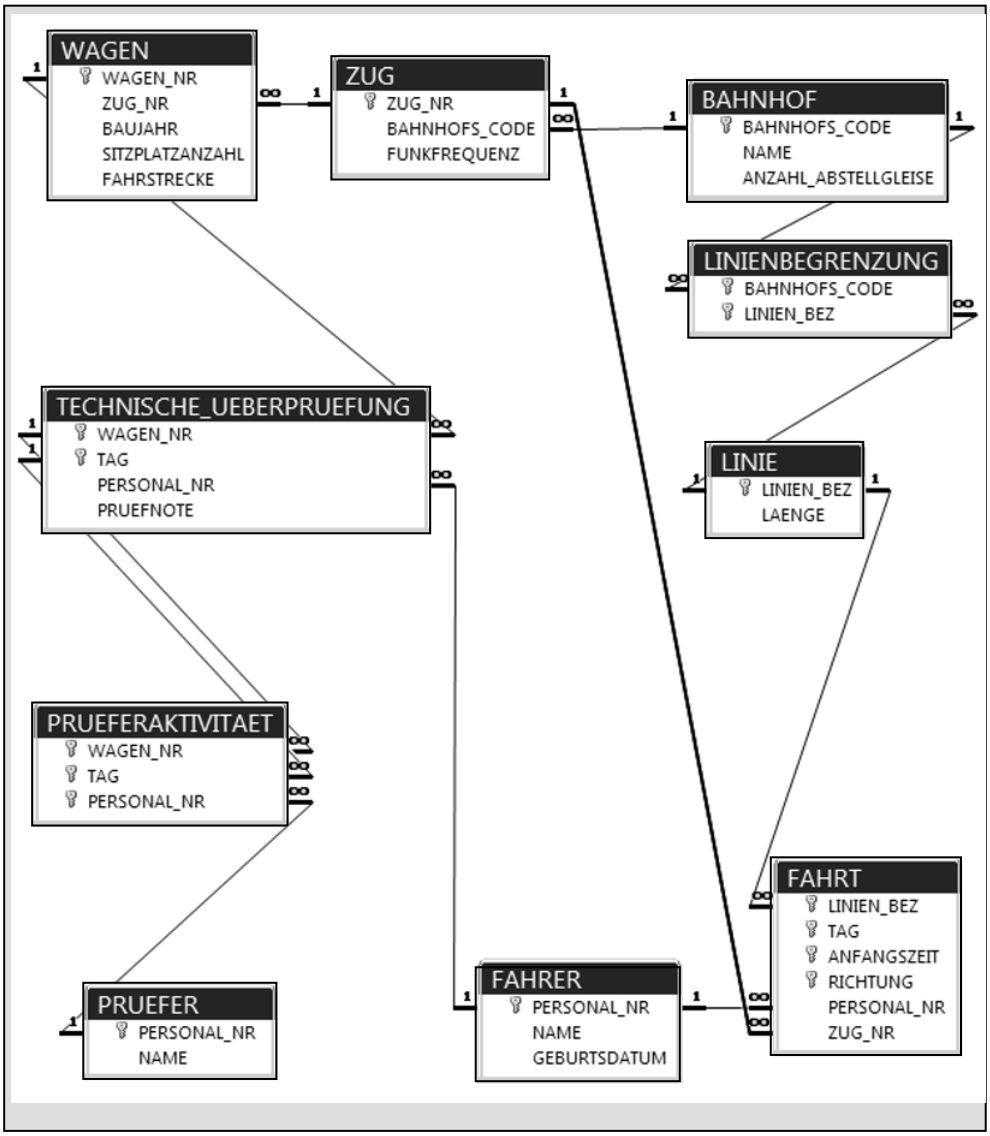


Abb. 6-10: Datenbank-Struktur für das Schnellbahn-Unternehmen

6.4 Eine Tankstellenkette

6.4.1 Beschreibung des Gegenstandsbereichs

Eine Tankstellenkette möchte in allen größeren Orten der Region Tankstellen einrichten und die relevanten Daten in einer Datenbank speichern.

In Kleinkennstenich (Kreis Hintermberg, 2 000 Einwohner) gibt es noch keine Tankstelle, in Benzhausen (Kreis Mobilland, 75 000 Einwohner) bereits drei Tankstellen. Eine dieser Tankstellen befindet sich in der Rennstraße 77 (PLZ 98765) und hat eine Fläche von 3 700 m^2. Sie hat 13 Mitarbeiter, über die Personalnummer, Name und Adresse gespeichert werden. Ein Mitarbeiter kann mehreren - höchstens drei - Tankstellen zugeordnet sein. Mitarbeiter können andere Mitarbeiter anleiten, wobei ein Mitarbeiter mehrere Chefs haben kann und ein Chef mindestens 3 Mitarbeiter anleitet.

Jede Tankstelle der Kette kann selbst wählen, von welchem Großhändler sie den Kraftstoff bezieht. Der Großhändler „Peter Petrolius AG“ ist für die Kette von großem Interesse, auch wenn er noch mit keiner Tankstelle zusammenarbeitet. Zu jedem Großhändler wird über den unikalen Firmennamen hinaus noch die Anschrift des Hauptsitzes gespeichert.

Die oben genannte Tankstelle hat 8 Kraftstofftanks, wobei der Tank mit der Nummer 3 ein Fassungsvermögen von 70 000 l und einen Füllstand von 35% hat. Er enthält den Kraftstoff mit der Bezeichnung „Superbenzin bleifrei“. Der Kraftstoff „Superbenzin bleifrei“ hat eine Oktanzahl von 95 und kostet heute an allen Tankstellen der Kette 1,36 Euro. Die Tankstelle verfügt über 12 Zapfsäulen. Eine Zapfsäule ist jeweils mit genau 4 Kraftstofftanks verbunden, ein Tank kann mehrere Zapfsäulen speisen. Der Tank 8 ist vorübergehend stillgelegt: Er enthält keinen Kraftstoff und versorgt keine Zapfsäule.

Es werden Angaben über die Tankvorgänge gespeichert. Zu jedem Tankvorgang muss ersichtlich sein, an welcher Zapfsäule welcher Kraftstoff in welcher Menge getankt wurde und wie hoch der Tankpreis war. Notfalls kann dem Tankvorgang auch das betankte Fahrzeug zugeordnet werden - jedoch nur dann, wenn das Bezahlen „vergessen“ wurde. So wurde beispielsweise in der betrachteten Tankstelle am 08.08.2008 um 08:08 Uhr an der Zapfsäule 11 ein blauer VW Golf mit dem polizeilichen Kennzeichen „GAUN ER 007“ betankt, ohne dass bezahlt wurde.

Das ist besonders ärgerlich, weil dieses Fahrzeug schon zum dritten Mal einem Tankvorgang zugeordnet wurde.

Weiterhin ist zu beachten:

- Die Tankstellen der Kette werden nur in Orten und nicht auf freier Strecke eingerichtet.
- Eine neu eingerichtete Tankstelle hat noch keine Mitarbeiter eingestellt und hat noch zu keinem Großhändler Kontakt. Später kann sie dann aber nur von einem Großhändler den Kraftstoff beziehen.
- Die Kraftstofftanks und die Zapfsäulen werden innerhalb einer Tankstelle jeweils durchnummeriert. Es gibt somit in der Tankstellenkette sehr viele Tanks bzw. Zapfsäulen mit der Nummer 1.
- Eine Tankstelle hat mindestens 4 Kraftstofftanks und mindestens 2 Zapfsäulen. Kraftstoff einer Sorte kann sich in mehreren Tanks befinden. Ein Kraftstoff ist in wenigstens einem Tank vorhanden.
- An einer Zapfsäule kann zu einem Zeitpunkt nur ein Tankvorgang stattfinden. Einer neu installierten Zapfsäule wurde noch kein Tankvorgang zugeordnet.
- Es werden nur solche Fahrzeuge gespeichert, die wenigstens einem „finanzierungsfreien“ Tankvorgang zugeordnet wurden.

6.4.2 Konzeptionelles Datenmodell

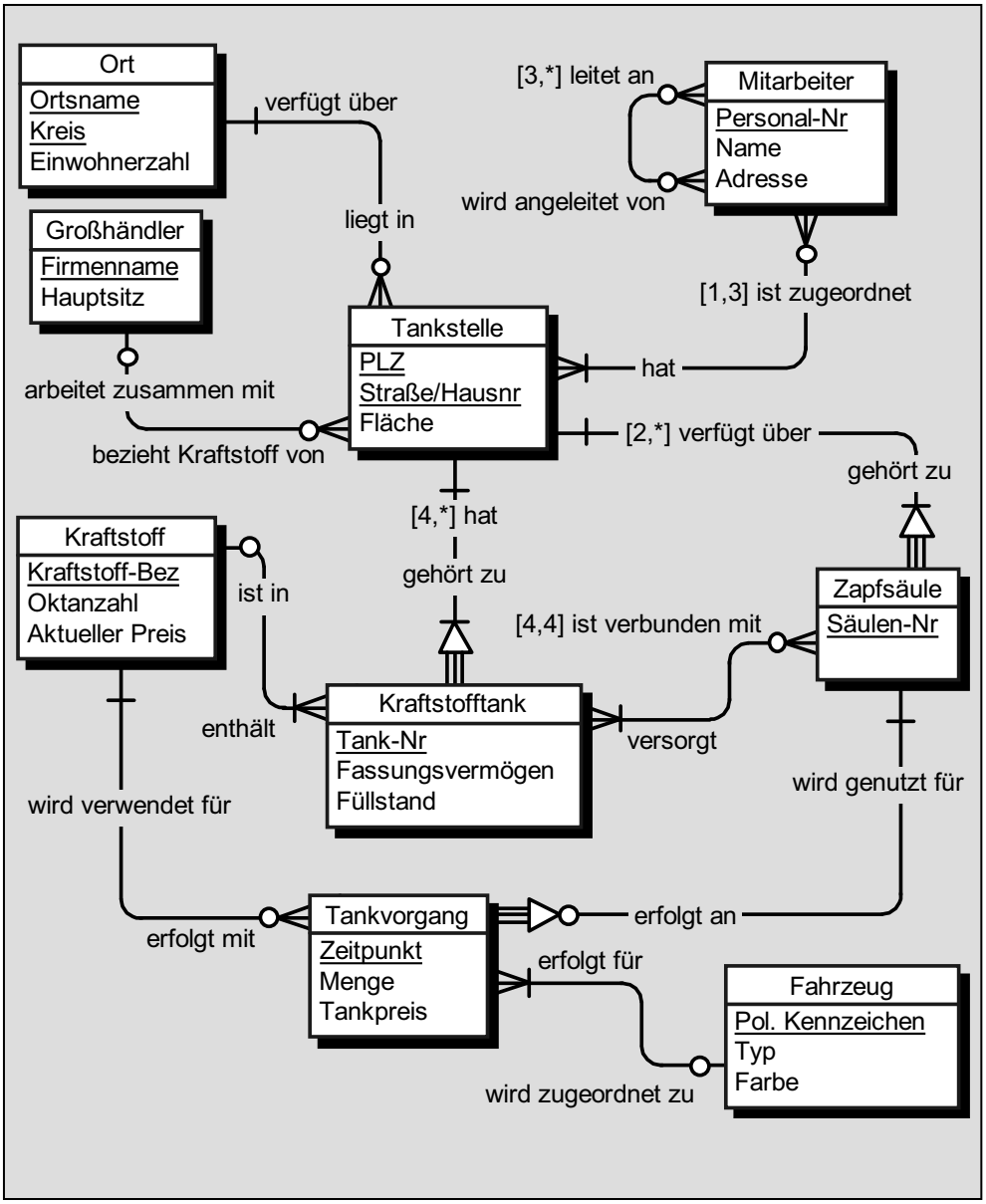


Abb. 6-11: Konzeptionelles Datenmodell für die Tankstellenkette

Bemerkungen:

- Da nicht gesichert ist, dass es in einem Postleitzahl-Bezirk nur eine einzige Tankstelle gibt, wird der Objekttyp „Tankstelle“ durch eine Kombination aus den Eigenschaften „PLZ“ und „Straße/Hausnr“ identifiziert.
- Die Kraftstofftanks und die Zapfsäulen werden jeweils innerhalb einer Tankstelle durchnummeriert. Deshalb erfolgt die Identifizierung des Objekttyps „Kraftstofftank“ (bzw. „Zapfsäule“) durch eine Kombination aus der Beziehungstyp-Richtung „Kraftstofftank *gehört zu* Tankstelle“ (bzw. „Zapfsäule *gehört zu* Tankstelle“) und der Eigenschaft „Tank-Nr“ (bzw. „Säulen-Nr“).
- An einer Zapfsäule kann zu einem Zeitpunkt nur ein Tankvorgang stattfinden. Deshalb ist der Objekttyp „Tankvorgang“ durch eine Kombination aus der Beziehungstyp-Richtung „Tankvorgang *erfolgt an* Zapfsäule“ und der Eigenschaft „Zeitpunkt“ identifizierbar.
- Die Speicherung des Tankpreises im Objekttyp „Tankvorgang“ bringt keine Redundanz mit sich. Der aktuelle Preis des Kraftstoffs ändert sich nämlich häufig, so dass sich aus der Kenntnis des Kraftstoffs und der getankten Menge später nicht mehr der Tankpreis rekonstruieren lässt.
- Ein und dasselbe Fahrzeug kann mehreren „finanzierungsfreien“ Tankvorgängen zugeordnet werden. Deshalb muss die Beziehungstyp-Richtung „Fahrzeug *wird zugeordnet zu* Tankvorgang“ die Kardinalität N aufweisen.
- Es wird angenommen, dass eventuell ein neuer Kraftstoff gespeichert werden muss, der noch für keinen Tankvorgang verwendet wurde. Deshalb ist die Beziehungstyp-Richtung „Kraftstoff *wird verwendet für* Tankvorgang“ optional.

6.4.3 Transformation in das logische Datenschema

Zunächst werden die Objekttypen in die entsprechenden Tabellen transformiert (Transformationsregel T01). Danach erfolgt die Transformation der Beziehungstyp-Richtungen, die als identifizierende Elemente für die „schwachen“ Objekttypen verwendet wurden (Transformationsregel T02). Im Ergebnis dieser beiden Transformationen ergeben sich die folgenden vorläufigen Tabellen-Typbeschreibungen:

Ort(**Ortsname+Kreis**,Einwohnerzahl)

Tankstelle(**PLZ+Straße/Hausnr**,Fläche)

Mitarbeiter(**Personal-Nr**,Name,Adresse)

Großhändler(**Firmenname**,Hauptsitz)

Kraftstofftank(⇑**PLZ+Straße/Hausnr**⇑**+Tank-N**r,Fassungsvermögen,Füllstand)

Kraftstoff(**Kraftstoff-Bez**,Oktanzahl,Aktueller Preis)

Zapfsäule(⇑**PLZ+Straße/Hausnr**⇑**+Säulen-Nr**)

Tankvorgang(⇑**PLZ+Straße/Hausnr+Säulen-Nr**⇑**+Zeitpunkt**,Menge,Tankpreis)

Fahrzeug(**Pol. Kennzeichen**,Typ,Farbe)

Dabei wurden folgende Primärschlüssel als Fremdschlüssel in die Primärschlüssel der „schwachen" Objekttypen aufgenommen:

Primärschlüssel	des Objekttyps	wird aufgenommen in	Primärschlüssel von
PLZ+Straße/Hausnr	Tankstelle	⇒	Kraftstofftank
PLZ+Straße/Hausnr	Tankstelle	⇒	Zapfsäule
PLZ+Straße/Hausnr+Säulen-Nr	Zapfsäule	⇒	Tankvorgang

Im nächsten Schritt werden die dualen Beziehungstypen in das relationale Datenbank-Modell transformiert (Transformationsregeln T03 bis T12). Dann erfolgt die Transformation des CM:CN-Rekursiv-Beziehungstyps von „Mitarbeiter" gemäß der Transformationsregel T19. Die veränderten Typbeschreibungen sind jeweils durch einen Stern gekennzeichnet:

Ort(**Ortsname+Kreis**,Einwohnerzahl)

* Tankstelle(**PLZ+Straße/Hausn**r,⇂*Firmenname*⇂,⇑Ortsname+Kreis⇑,Fläche)

* Personal-Zuordnung(⇑**PLZ+Straße/Hausnr**⇑+⇑**Personal-Nr**⇑)

Mitarbeiter(**Personal-Nr**,Name,Adresse)

* Anleitung(⇑**Personal-Nr**⇑+⇑**Chef-Personal-Nr**⇑)

Großhändler(**Firmenname**,Hauptsitz)

* Kraftstofftank(⇑**PLZ+Straße/Hausnr**⇑**+Tank-Nr**,⇂*Kraftstoff-Bez*⇂,
Fassungsvermögen,Füllstand)

* Rohrverbindung(⇑**PLZ+Straße/Hausnr+Säulen-Nr**⇑+
⇑ **[PLZ+Straße/Hausnr+] Tank-Nr**⇑)

Kraftstoff(**Kraftstoff-Bez**,Oktanzahl,Aktueller Preis)

Zapfsäule(⇑**PLZ+Straße/Hausnr**⇑**+Säulen-Nr**)

* Tankvorgang(⇑**PLZ+Straße/Hausnr+Säulen-Nr**⇑**+Zeitpunkt**,
⇑Kraftstoff-Bez⇑,Menge,Tankpreis)

* Unbezahlter Tankvorgang(⇑**PLZ+Straße/Hausnr+Säulen-Nr+Zeitpunkt**⇑,
⇑Pol. Kennzeichen⇑)

Fahrzeug(**Pol. Kennzeichen**,Typ,Farbe)

Bemerkungen:

- Die folgenden 1:N-Beziehungstypen können lediglich unter Semantikverlust als 1:CN-Beziehungstypen, also gemäß der Transformationsregel T09, repräsentiert werden:

  Tankstelle ←→ Kraftstofftank

  Tankstelle ←→ Zapfsäule

  Die Kardinalitäts-Beschränkungen lassen sich erst recht nicht durchsetzen. Es kann also durch die Tabellen-Typbeschreibungen nicht gesichert werden, dass

  - eine Tankstelle mindestens 4 Kraftstofftanks hat,
  - eine Tankstelle über wenigstens 2 Zapfsäulen verfügt.

- Der C:N-Beziehungstyp zwischen „Kraftstoff“ und „Kraftstofftank“ lässt sich nur als C:CN-Beziehungstyp repräsentieren. Die Tabellen-Typbeschreibung kann also nicht garantieren, dass sich jeder Kraftstoff in mindestens einem Kraftstofftank befindet.

  Für den C:CN-Beziehungstyp stehen nun zwei Transformationsregeln zur Verfügung:

  a) Transformationsregel T10 (die meisten Kraftstofftanks enthalten einen Kraftstoff),

  b) Transformationsregel T11 (die meisten Kraftstofftanks sind leer).

  Natürlich kann sich eine Tankstelle nur den Fall a) leisten. Deshalb wurde keine Koppel-Tabelle eingeführt. Stattdessen wurde in die Tabelle „Kraftstofftank“ ein *nicht-eingabepflichtiger* Verweis auf den „Kraftstoff“ aufgenommen (deshalb der Kursivdruck!).

- Der C:N-Beziehungstyp zwischen „Fahrzeug“ und „Tankvorgang“ lässt sich wiederum nur als C:CN-Beziehungstyp repräsentieren. Die Tabellen-Typbeschreibung lässt dann unsinnigerweise die Speicherung eines Fahrzeugs zu, das keinem Tankvorgang zugeordnet wird.

  Für den C:CN-Beziehungstyp stehen wiederum zwei Transformationsregeln zur Verfügung:

  a) Transformationsregel T10 (den meisten Tankvorgängen wird ein Fahrzeug zugeordnet, d.h. fast alle Kunden der Tankstelle „vergessen“ das Bezahlen),

  b) Transformationsregel T11 (den meisten Tankvorgängen wird kein Fahrzeug zugeordnet, weil die Kunden ordnungsgemäß bezahlt haben).

  Im Interesse der Tankstelle wollen wir annehmen, dass der Fall b) vorliegt. Deshalb wurde eine Koppel-Tabelle „Unbezahlter Tankvorgang“ eingeführt, durch die die wenigen „finanzierungsfreien“ Tankvorgänge mit dem jeweiligen Fahrzeug in Verbindung gebracht werden (im Unterschied zum PowerDesigner, s. Abschnitt 4.4).

- Für den C:CN-Beziehungstyp zwischen „Großhändler“ und „Tankstelle“ stehen wiederum zwei Transformationsregeln zur Verfügung:

  a) Transformationsregel T10 (die meisten Tankstellen beziehen Kraftstoff von einem Großhändler),

  b) Transformationsregel T11 (die meisten Tankstellen stehen noch mit keinem Großhändler in Kontakt).

  Sicherlich wird der Fall a) vorliegen. Deshalb wurde keine Koppel-Tabelle eingeführt. Stattdessen wurde in die Tabelle „Tankstelle“ ein *nicht-eingabepflichtiger* Verweis auf den „Großhändler“ aufgenommen (deshalb der Kursivdruck!).

- Die folgenden M:CN-Beziehungstypen können nur unter Semantikverlust als CM:CN-Beziehungstypen, also gemäß der Transformationsregel T12, repräsentiert werden:

Tankstelle	←→	Mitarbeiter
Kraftstofftank	←→	Zapfsäule

  Die Kardinalitäts-Beschränkungen lassen sich erst recht nicht durchsetzen. Es kann also durch die Tabellen-Typbeschreibungen der beiden Koppel-Tabellen „Personal-Zuordnung“ bzw. „Rohrverbindung“ nicht gesichert werden, dass

  - ein Mitarbeiter mindestens einer und höchstens 3 Tankstellen zugeordnet ist,
  - eine Zapfsäule mit genau 4 Kraftstofftanks verbunden ist.

- In der Tabelle „Rohrverbindung“ müsste der Primärschlüssel der Tankstelle (**PLZ+Straße/Hausnr**) eigentlich zweimal auftauchen:

  a) im Verweis auf die Zapfsäule, die über die Tankstelle identifiziert wird:
  ⇑PLZ+Straße/Hausnr+Säulen-Nr⇑

  b) im Verweis auf den Kraftstofftank, der ebenfalls über die Tankstelle identifiziert wird:
  ⇑PLZ+Straße/Hausnr+Tank-Nr⇑

Da jedoch Zapfsäule und Kraftstofftank sicherlich zur selben Tankstelle gehören, muss deren Primärschlüssel nicht doppelt aufgeführt werden. Er wurde deshalb beim Verweis auf den Kraftstofftank in eckige Klammern gesetzt:

⇑ [PLZ+Straße/Hausnr+] Tank-Nr⇑

Als zusätzliches Attribut wird lediglich die **Tank-Nr** aufgenommen. Diese bildet gemeinsam mit den beiden Attributen **PLZ+Straße/Hausnr** *aus dem Verweis auf die Zapfsäule* den Verweis auf den Kraftstofftank.

- Der CM:CN-Rekursiv-Beziehungstyp des Objekttyps „Mitarbeiter“ wird gemäß der Transformationsregel T19 durch die Koppel-Tabelle „Anleitung“ repräsentiert. Allerdings lässt sich die Kardinalitäts-Beschränkung nicht darstellen. Es kann also durch die Typbeschreibung der Koppel-Tabelle nicht durchgesetzt werden, dass ein Chef mindestens 3 Mitarbeiter anleitet.

6.4.4 „Physisches Datenmodell“ des PowerDesigner

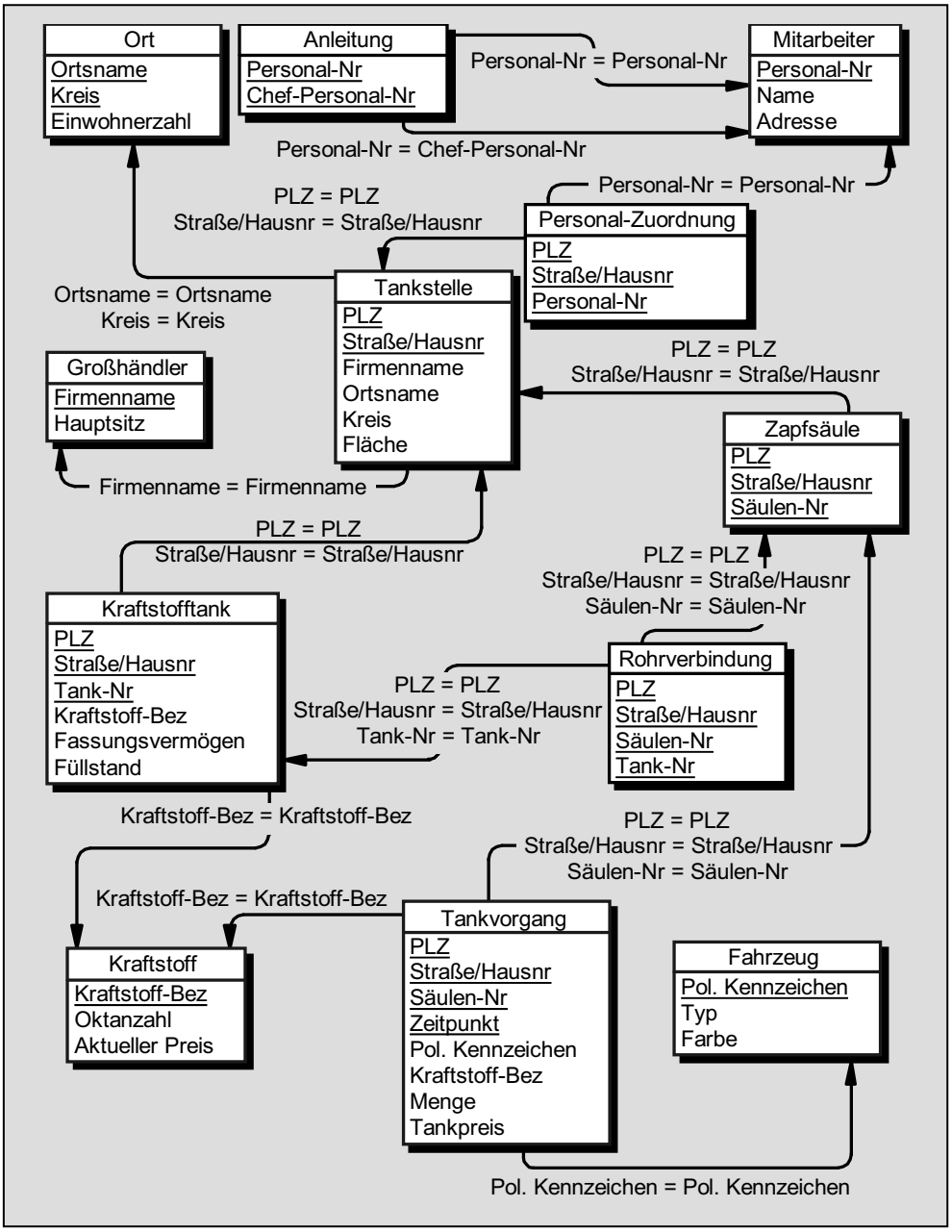


Abb. 6-12: „Physisches Datenmodell“ für die Tankstellenkette

Bemerkung:

Dem PowerDesigner stehen keine Angaben über die Häufigkeit zur Verfügung, mit der die Optionalität einer Beziehungstyp-Richtung realisiert wird. Er kann deshalb bei der Transformation des C:N-Beziehungstyps zwischen „Fahrzeug“ und „Tankvorgang“, der ja als C:CN-Beziehungstyp repräsentiert werden muss, keine Auswahl unter den Transformationsregeln T10 (selten realisierte Optionalität) und T11 (häufig realisierte Optionalität) treffen. Er repräsentiert einen C:CN-Beziehungstyp immer nach der Transformationsregel T10. Deshalb wurde – entgegen der von uns vorgenommenen Transformation im Rahmen der Tabellen-Typbeschreibungen (Abschnitt 4.3) - im „physischen“ Datenmodell in die Tabelle „Tankvorgang“ ein *nicht-eingabepflichtiger* Verweis auf das „Fahrzeug“ aufgenommen (deshalb der Kursivdruck!):

* Tankvorgang(⇑**PLZ+Straße/Hausnr+Säulen-Nr**⇑**+Zeitpunkt**,
⇑*Pol. Kennzeichen*⇑,⇑Kraftstoff-Bez⇑,Menge,Tankpreis)

6.4.5 Datenbank-Struktur für Access

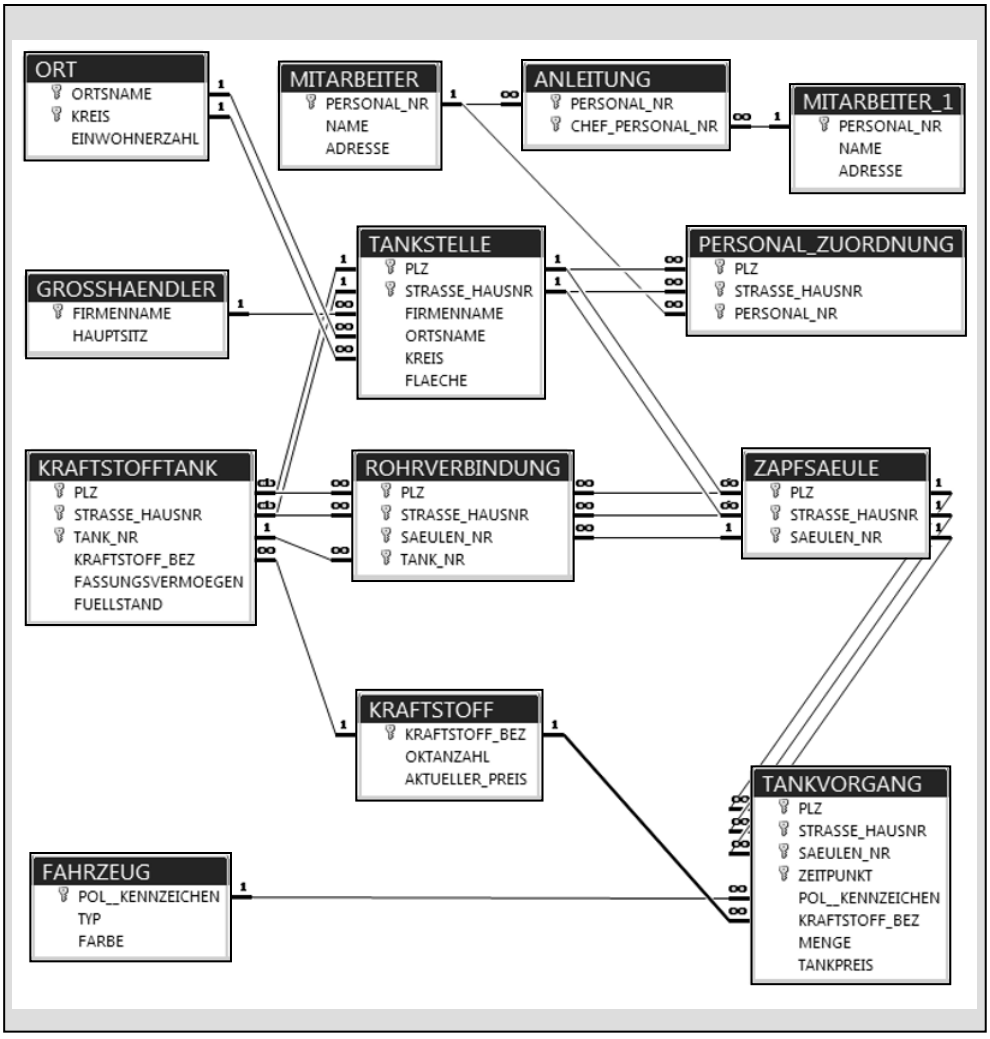


Abb. 6-13: Datenbank-Struktur für die Tankstellenkette

6.5 Ein Videoverleih

6.5.1 Beschreibung des Gegenstandsbereichs

Ein traditionsbewusstes Unternehmen befasst sich ausschließlich mit dem Verleih von Filmklassikern auf Videokassetten und will das Bestell- und Verleihwesen auf ein Datenbank-Anwendungssystem umstellen.

Das Unternehmen möchte in allen Städten mit mehr als 10 000 Einwohnern präsent sein. Über alle diese Städte werden deshalb Daten erfasst. In Flimmerhausen (Sachsen-Anhalt, 130 000 Einwohner) gibt es bereits 3 Filialen, in Dunkelfingen (Baden-Württemberg, 11 000 Einwohner) noch keine. Eine der Filialen in Flimmerhausen (Skladanowskystr. 1, PLZ 34567) hat eine Verkaufsfläche von 110 m^2 und beschäftigt 3 Mitarbeiter. Sie hält den Kontakt zu mehreren Filmverlegern, so z. B. zu der „Kientopp GmbH“ (Schnittstr. 2, 12345 Filmbach). Mit diesem Filmverleger sind auch andere Filialen in Kontakt. Der Filmverleger „Großkotz und Co.“ (Flegelstr. 7, 76543 Absahnitz) bietet so schlechte Bedingungen, dass alle Filialen den Kontakt zu ihm abgebrochen haben. Er bleibt allerdings weiterhin gespeichert.

Die Filmverleger bieten jeweils mehrere Filmtitel an, wobei ein interessierender Filmtitel bei mehreren Filmverlegern (oder auch bei keinem) im Angebot sein kann. Die Filialen schicken nun an die einzelnen Filmverleger Bestellungen. Zu jeder Bestellung muss ersichtlich sein, von welcher Filiale sie kommt, an welchen Filmverleger sie gerichtet ist und an welchem Tag sie erstellt wurde. Eine Bestellung enthält im allgemeinen mehrere Bestellpositionen, die durch eine laufende Nummer voneinander unterschieden werden. Eine Bestellposition bezieht sich auf genau einen Filmtitel und verzeichnet die gewünschte Anzahl der Videokassetten von diesem Filmtitel.

Über alle Videokassetten des Unternehmens wird Buch geführt, wobei die Videokassetten jeweils einer Filiale durch eine fortlaufende Identifikationsnummer unterschieden werden. Beispielsweise enthält die Videokassette der Flimmerhausener Filiale in der Skladanowskystraße 1 (PLZ 34567) mit der Identifikationsnummer 445566 den Filmtitel „Panzerkreuzer Potemkin“ (1925, Regisseur: Sergej Eisenstein).

Zu einem Ausleihvorgang wird festgehalten, welcher Kunde welche Videokassette ausgeliehen hat und zu welchem Datum er sie zurückgeben muss. Wurde die Videokassette zurückgegeben,

wird der Ausleihvorgang gelöscht: für eine Videokassette wird also höchstens ein Ausleihvorgang gespeichert. Zu jedem Kunden wird der Name, die Adresse und das Alter festgehalten. Kinder können nur dann Kunden sein, wenn wenigstens ein Elternteil Kunde ist. Es wird deshalb bei den Kunden vermerkt, welches Kind zu welchen Eltern gehört.

Weiterhin ist zu beachten:

- Eine neu eingerichtete Filiale hat noch zu keinem Filmverleger Kontakt, somit auch noch keine Bestellungen aufgegeben und besitzt also auch noch keine Videokassetten.
- Eine Filiale sendet pro Tag an einen Filmverleger höchstens eine Bestellung.
- Beliebte Filmtitel werden von den Filialen häufig bestellt, andere dagegen überhaupt nicht.
- Die Identifikationsnummer für eine Videokassette wird innerhalb einer Filiale vergeben. Im Unternehmen kann also dieselbe Identifikationsnummer mehrfach auftreten.
- Die Filmklassiker lassen sich durch ihren unikalen Namen unterscheiden.
- Auf einer Videokassette befindet sich immer nur ein Filmtitel.
- Ein Kunde kann höchstens 10 Videokassetten ausleihen. Viele Videokassetten sind gerade nicht ausgeliehen und stehen in den Regalen.

6.5.2 Konzeptionelles Datenmodell

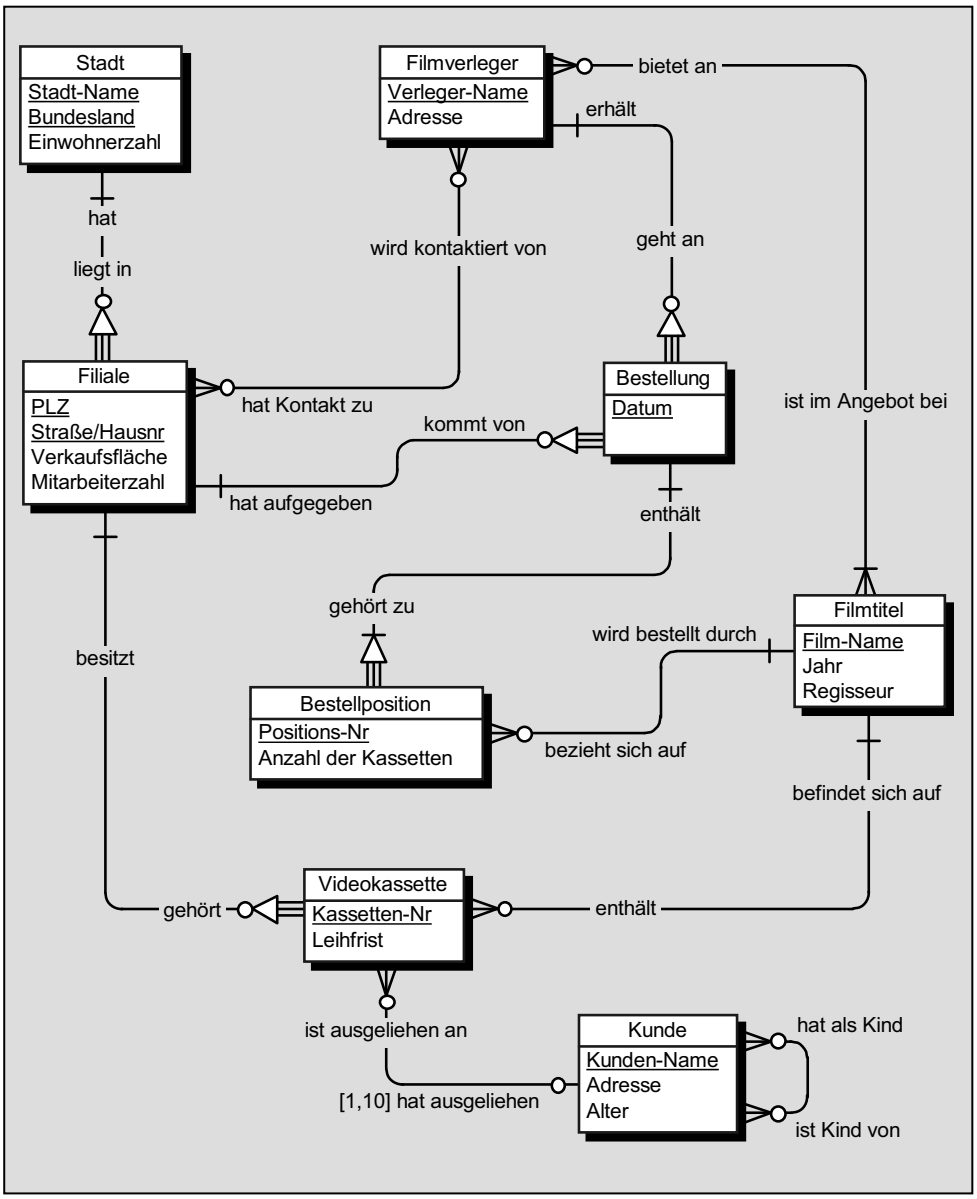


Abb. 6-14: Konzeptionelles Datenmodell für den Videoverleih

Bemerkungen:

- Es ist damit zu rechnen, dass mehrere namensgleiche Städte gespeichert werden müssen. Deshalb wird der Objekttyp „Stadt“ durch eine Kombination der beiden Eigenschaften „Stadt-Name“ und „Bundesland“ identifiziert.
- Eine Filiale kann nur bei Angabe ihrer vollständigen Adresse zweifelsfrei bestimmt werden. Deshalb wird der Objekttyp „Filiale“ durch eine Kombination aus der Beziehungstyp-Richtung „Filiale *liegt in* Stadt“ und aus den Eigenschaften „PLZ“ und „Straße/Hausnr“ identifiziert.
- Da Filmverleger in das Handelsregister eingetragen werden, muss ihr Name unikal sein. Deshalb ist für die Identifizierung des Objekttyps „Filmverleger“ die Eigenschaft „Verleger-Name“ ausreichend.
- Filmverleger bleiben auch dann noch gespeichert, wenn alle Filialen den Kontakt zu ihnen abgebrochen haben. Deshalb muss die Beziehungstyp-Richtung „Filmverleger *wird kontaktiert von* Filiale“ optional sein.
- Filmverleger, die keine Filmtitel im Angebot haben, gibt es nicht. Deshalb ist die Beziehungstyp-Richtung „Filmverleger *bietet an* Filmtitel“ nicht-optional.
- Eine Filiale sendet pro Tag an einen Filmverleger höchstens eine Bestellung. Aus diesem Grund kann die Identifizierung des Objekttyps „Bestellung“ durch eine Kombination aus den beiden Beziehungstyp-Richtungen „Bestellung *kommt von* Filiale“ und „Bestellung *geht an* Filmverleger“ sowie aus der Eigenschaft „Datum“ erfolgen.
- Die Bestellpositionen ein und derselben Bestellung unterscheiden sich in ihrer Positions-Nummer. Deshalb wird der Objekttyp „Bestellposition“ durch eine Kombination aus der Beziehungstyp-Richtung „Bestellposition *gehört zu* Bestellung“ und der Eigenschaft „Positions-Nr“ identifiziert.
- Manche gespeicherte Filmtitel werden überhaupt nicht bestellt. Deshalb muss die Beziehungstyp-Richtung „Filmtitel *wird bestellt durch* Bestellposition“ optional sein.
- Da die Identifikationsnummer einer Videokassette nur innerhalb einer Filiale eindeutig ist, wird der Objekttyp „Videokassette“ über seine Beziehung zur Filiale und durch seine Eigenschaft „Kassetten-Nr“ identifiziert.

- Für eine Videokassette wird nur der aktuelle Ausleihvorgang gespeichert, abgeschlossene Ausleihvorgänge werden sofort gelöscht. Deshalb ist kein eigener Objekttyp für den Ausleihvorgang erforderlich. Die notwendigen Angaben können unmittelbar an der Videokassette „festgemacht“ werden.
- Es soll gespeichert werden, welcher „Kind-Kunde“ zu welchem „Eltern-Kunden“ gehört. Deshalb muss ein CM:CN-Rekursiv-Beziehungstyp für den Objekttyp „Kunde“ modelliert werden.

6.5.3 Transformation in das logische Datenschema

Zunächst werden die Objekttypen in die entsprechenden Tabellen transformiert (Transformationsregel T01). Danach erfolgt die Transformation der Beziehungstyp-Richtungen, die als identifizierende Elemente für die „schwachen“ Objekttypen verwendet wurden (Transformationsregel T02). Im Ergebnis dieser beiden Transformationen ergeben sich die folgenden vorläufigen Tabellen-Typbeschreibungen:

Stadt(**Stadt-Name+Bundesland**,Einwohnerzahl)

Filiale(⇑**Stadt-Name+Bundesland**⇑**+PLZ+Straße/Hausnr**,
Verkaufsfläche,Mitarbeiterzahl)

Filmverleger(**Verleger-Name**,Adresse)

Filmtitel(**Film-Name**,Jahr,Regisseur)

Bestellung(⇑**Verleger-Name**⇑**+**
⇑**Stadt-Name+Bundesland+PLZ+Straße/Hausnr**⇑**+**
Datum)

Bestellposition(⇑**Verleger-Name+**
Stadt-Name+Bundesland+PLZ+Straße/Hausnr+Datum⇑**+**
Positions-Nr,Anzahl der Kassetten)

Videokassette(⇑**Stadt-Name+Bundesland+PLZ+Straße/Hausnr**⇑**+**
Kassetten-Nr,Leihfrist)

Kunde(**Kunden-Name+Adresse**,Alter)

Dabei wurden folgende Primärschlüssel als Fremdschlüssel in die Primärschlüssel der „schwachen“ Objekttypen aufgenommen:

Primärschlüssel	des Objekttyps	wird aufgenommen in	Primärschlüssel von
Stadt-Name+Bundesland	Stadt	⇒	Filiale
Verleger-Name	Filmverleger	⇒	Bestellung
Stadt-Name+Bundesland+ PLZ+Straße/Hausnr	Filiale	⇒	Bestellung
Verleger-Name+ Stadt-Name+Bundesland+ PLZ+Straße/Hausnr+Datum	Bestellung	⇒	Bestellposition
Stadt-Name+Bundesland+ PLZ+Straße/Hausnr	Filiale	⇒	Videokassette

Im nächsten Schritt werden die dualen Beziehungstypen in das relationale Datenbank-Modell transformiert (Transformationsregeln T03 bis T12). Dann erfolgt die Transformation des CM:CN-Rekursiv-Beziehungstyps von „Kunde“ gemäß der Transformationsregel T19. Die veränderten Typbeschreibungen sind jeweils durch einen Stern gekennzeichnet:

Stadt(**Stadt-Name+Bundesland**,Einwohnerzahl)

Filiale(⇑**Stadt-Name+Bundesland**⇑**+PLZ+Straße/Hausnr**,
Verkaufsfläche,Mitarbeiterzahl)

* Kontakt(⇑**Verleger-Name**⇑**+**
⇑**Stadt-Name+Bundesland+PLZ+Straße/Hausnr**⇑)

Filmverleger(**Verleger-Name**,Adresse)

* Angebot(⇑**Verleger-Name**⇑**+**⇑**Film-Name**⇑)

Filmtitel(**Film-Name**,Jahr,Regisseur)

Bestellung(⇑**Verleger-Name**⇑**+**
⇑**Stadt-Name+Bundesland+PLZ+Straße/Hausnr**⇑**+**
Datum)

* Bestellposition(⇑**Verleger-Name+**
Stadt-Name+Bundesland+PLZ+Straße/Hausnr+Datum⇑**+**
Positions-Nr,⇑Film-Name⇑,Anzahl der Kassetten)

* Videokassette(⇑**Stadt-Name+Bundesland+PLZ+Straße/Hausnr**⇑**+**
Kassetten-Nr,⇑Film-Name⇑,⇑Kunden-Name+Adresse⇑,Leihfrist)

Kunde(**Kunden-Name+Adresse**,Alter)

* Verwandtschaft(⇑**Kind-Name+Kind-Adresse**⇑**+**
⇑**Eltern-Name+Eltern-Adresse**⇑)

Bemerkungen:

- Der 1:N-Beziehungstyp zwischen „Bestellung“ und „Bestellposition“ lässt sich lediglich unter Semantikverlust als ein 1:CN-Beziehungstyp, also gemäß der Transformationsregel T09, repräsentieren. Die Tabellen-Typbeschreibung lässt es dann unsinnigerweise zu, dass eine Bestellung gespeichert wird, die keine einzige Bestellposition enthält.

- Für den C:CN-Beziehungstyp zwischen „Kunde“ und „Videokassette“ stehen zwei Transformationsregeln zur Verfügung:

  a) Transformationsregel T10 (die meisten Videokassetten sind ausgeliehen),

  b) Transformationsregel T11 (die meisten Videokassetten stehen in den Regalen).

  Es lässt sich nicht mit Bestimmtheit sagen, welcher der beiden Fälle vorliegt. Wir haben eine rege Ausleihtätigkeit vorausgesetzt, also den Fall a) angenommen. Deshalb wurde keine Koppel-Tabelle eingeführt. Stattdessen wurde in die Tabelle „Videokassette“ ein *nicht-eingabepflichtiger* Verweis auf den „Kunden“ aufgenommen (deshalb der Kursivdruck!). Die Kardinalitäts-Beschränkung lässt sich durch die Tabellen-Typbeschreibung nicht repräsentieren. Man kann – zumindest ohne besondere Software-Maßnahmen – also nicht verhindern, dass ein Kunde mehr als 10 Videokassetten ausleiht.
- Der M:CN-Beziehungstyp zwischen „Filmtitel“ und „Filmverleger“ lässt sich lediglich unter Semantikverlust als CM:CN-Beziehungstyp, also gemäß der Transformationsregel T12, repräsentieren. Durch die Typbeschreibung der Koppel-Tabelle „Angebot“ kann nicht gesichert werden, dass jeder Filmverleger wenigstens einen Filmtitel anbietet.
- Der CM:CN-Rekursiv-Beziehungstyp des Objekttyps „Kunde“ wird gemäß der Transformationsregel T19 durch die Koppel-Tabelle „Verwandtschaft“ repräsentiert. Der doppelt aufzunehmende Fremdschlüssel des Kunden wurde unterschiedlich benannt:

  a) im Falle des Verweises auf ein Kind:
  ⇑Kind-Name+Kind-Adresse⇑,

  b) im Falle des Verweises auf ein Elternteil:
  ⇑Eltern-Name+Eltern-Adresse⇑.

6.5.4 „Physisches Datenmodell“ des PowerDesigner

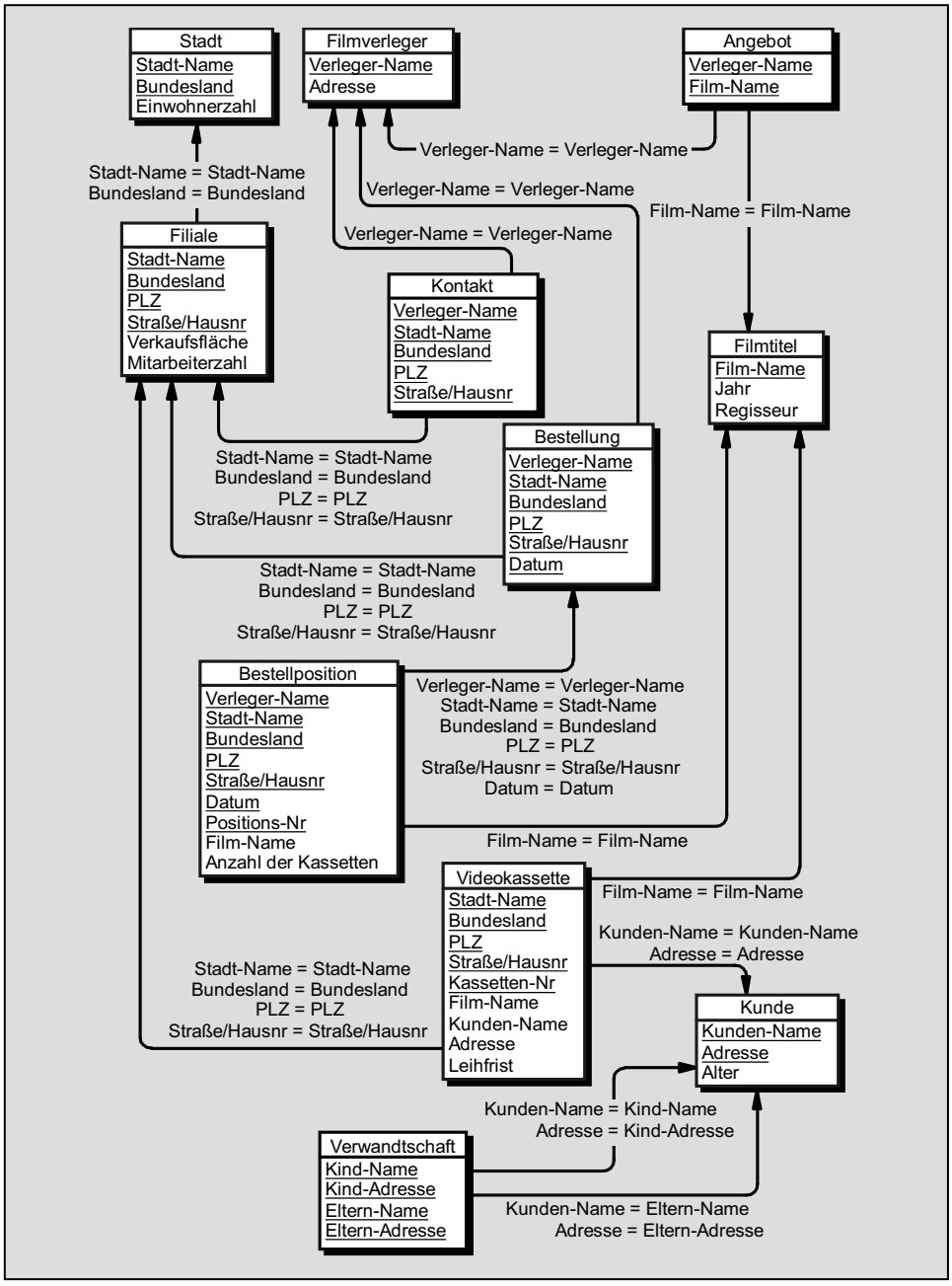


Abb. 6-15: „Physisches Datenmodell“ für den Videoverleih

6.5.5 Datenbank-Struktur für Access

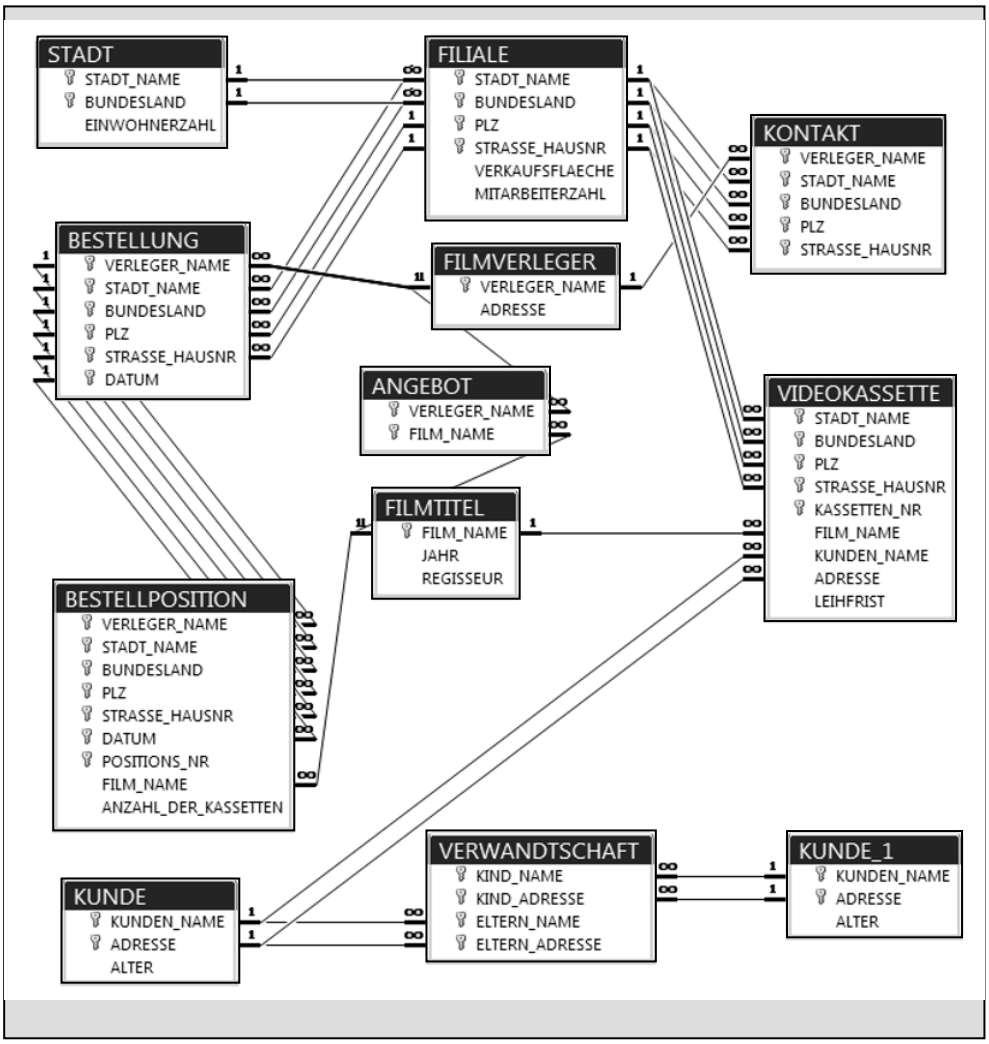


Abb. 6-16: Datenbank-Struktur für den Videoverleih

Literaturverzeichnis

ACCE03 *Access. Grundlagen für Datenbank-Entwickler.*
Regionales Rechenzentrum für Niedersachsen / Universität Hannover: 2003.

ANSI75 *ANSI/X3/SPARC Study Group on Data Base Management Systems. Interim Report 75-02-08.*
FDT Bull. of ACM SIGMOD, **7**(1975).

CHEN76 Chen, P. P.: *The Entity-Relationship Model – Towards a Unified View of Data.*
ACM Transactions on Database Systems **1**(1976)1, S. 9-36.

CODA71 *CODASYL Data Base Task Group.*
April 71 Report.

CODA73 *CODASYL DDL Journal of Development.*
June 73 Report.

CODD86 Codd, E. F.: *The Twelve Rules for Relational DBMS.*
The Relational Institute Technical Report EFC-6.
San Jose, 1986.

CODD70 Codd, E. F.: *A Relational Model for Large Shared Data Banks.*
Communications of the ACM **13**(1970)6.

CODD90 Codd, E. F.: *The Relational Model for Database Management – Version 2.*
Reading (Mass.): Addison-Wesley, 1990.

CONN04 Connolly, T.; Begg, C.: *Database Solutions. A step-by-step guide to building databases.*
Pearson Education, 2004.

CORD02 Cordts, S.: *Datenbankkonzepte in der Praxis.*
München, Boston, …: Addison-Wesley, 2002.

DATE09 Date, C. J.: *SQL and Relational Theorie.*
Köln: O'Reilly, 2009.

ELMA09 Elmasri, R.; Navathe, S. B.: *Grundlagen von Datenbanksystemen.*
3. Auflage.
München: Pearson Studium, 2009.

ERBS03 Erbs, H.-E.; Karczewski, S.; Schestag, I.: *Datenbanken.*
Berlin: VDE Verlag, 2003.

FETT08 Fettke, P.: *Empirisches Business Engineering – Grundlegung und ausgewählte Ergebnisse.*
Saarbrücken: Universität des Saarlandes, 2008.

FRIT02 Fritze, J.; Marsch, J.: *Erfolgreiche Datenbankanwendung mit SQL3.*
Braunschweig, Wiesbaden: Vieweg, 2002.

GARV98 Garvin, C.; Eckols, S.: *DB2 for the COBOL Programmer.*
Murach, 1998.

GEIS09 Geisler, F.: *Datenbanken. Grundlagen und Design.*
Bonn: mitp-Verlag, 2009.

HALP08 Halpin, T.; Morgan, T.: *Information Modelling and Relational Databases. From Conceptual Analysis to Logical Design.*
2. Auflage.
San Francisco: Morgan Kaufmann, 2001.

HEIN08 Heinrich, G.; Mairon, K.: *Objektorientierte Systemanalyse.*
München: Oldenbourg Verlag, 2008.

HEUE08 Heuer, A.; Saake, G.; Sattler, K.-U.: *Datenbanken. Konzepte und Sprachen.*
3. Auflage.
Bonn: mitp-Verlag, 2008.

JARO07 Jarosch, H: *Information Retrieval und Künstliche Intelligenz.*
Wiesbaden: Deutscher Universitätsverlag, 2007.

KEMP09 Kemper, A.; Eickler, A.: *Datenbanksysteme. Eine Einführung.*
7. Auflage.
München, Wien: Oldenbourg, 2009.

KIFE05 Kifer, M.; Bernstein, A.; Lewis, P. M.: *Database Systems. An Application-Oriented Appreach.*
2. Auflage.
Boston: Pearson Education, 2005.

KIYO06 Kiyoki, Y.; Henno, J.; Jaakkola, H.: *Information Modelling and Knowledge Bases.*
IOS Press, 2006.

KLEU06 Kleuker, S.: *Grundkurs Datenbankentwicklung. Von der Anforderungsanalyse zur komplexen Datenbankanfrage.*
Wiesbaden: Vieweg, 2006.

KLUG08 Klug, U.: *Datenbank-Anwendungen entwerfen & programmieren: Von der objektorientierten Analyse bis zur SQL-Implementierung.*
Witten: W3L GmbH, 2008.

LEIT03 Leitenbauer, G.: *Datenbank-Modellierung. Unternehmensdatenmodelle entwickeln und verstehen.*
Poing: Franzis Verlag, 2003.

MART89 Martin, J.: *Information Engineering. Book I: Introduction.*
Englewood Cliffs, New Jersey: Prentice Hall, 1989.

MCLA07 McLaughlin, B. D.; Pollice, G.; West, D.: *Objektorientierte Analyse und Design von Kopf bis Fuß.*
Köln: O'Reilly, 2007.

MEIE07 Meier, A.: *Relationale Datenbanken. Leitfaden für die Praxis.*
5. Auflage.
Berlin, Heidelberg, New York: Springer-Verlag, 2007.

MOOS04 Moos, A.: *Datenbank-Engineering.*
3. Auflage.
Braunschweig, Wiesbaden: Vieweg, 2004.

PREI07 Preiß, N.: *Entwurf und Verarbeitung relationaler Datenbanken.*
München: Oldenbourg, 2007.

RAUH97 Rauh, O.; Stickel, E.: *Konzeptuelle Datenmodellierung.*
Wiesbaden: Teubner-Verlag, 1997.

RICC03 Riccardi, G.: *Database management with Web site development applications*.
Boston: Pearson Education, 2003.

ROB07 Rob, P.; Coronel, C.: *Database Systems. Design, Implementation, and Management*.
Thomson, 2007.

ROLL03 Rolland, F. D.: *Datenbanksysteme*.
München: Pearson Studium, 2003.

SCHN01 Schnauder, V.; Jarosch, H.; Thieme, I: *Praxis der Software-Entwicklung*.
Renningen-Malmsheim: expert verlag, 2001.

SCHN04 Schnauder, V.; Jarosch, H.; Mages, M: *Datenbankgestützte Vertriebs- und Informationssysteme*.
Berlin: Logos Verlag, 2004.

SCHU07 Schubert, M.: *Datenbanken. Theorie, Entwurf und Programmierung relationaler Datenbanken*.
2. Auflage.
Stuttgart, Leipzig, Wiesbaden: Teubner, 2007.

STAI08 Stair, R. M.; Reynold, G. W.: *Principles of Information Systems. A Managerial Approach*.
8. Auflage.
Thomson, 2008.

STEI06 Steiner, R.: *Grundkurs relationale Datenbanken*.
6. Auflage.
Braunschweig, Wiesbaden: Vieweg, 2006.

THRO08 Throll, M.; Bartosch, O.: *Einstieg in SQL 2008*.
2. Auflage.
Bonn: Galileo Press, 2008.

TIEM97 Tiemeyer, E.; Konopasek, K.: *Professionelles Datenbank-Design mit ACCESS*.
2. Auflage.
Braunschweig, Wiesbaden: Vieweg, 1997.

UMAN07 Umanath, N. S.; Scamell, R. W.: *Data Modeling and Database Design*.
Thomson, 2007.

VOSS08 Vossen, G.: *Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme.* München: Oldenburg Wissenschaftsverlag, 2008.

WARD06 Ward, P.; Dafoulas, G.: *Database Management Systems.* Thomson, 2006.

ZEHN05 Zehnder, C. A.: *Informationssysteme und Datenbanken.* Zürich: vdf Hochschulverlag, 2005.

Schlagwortverzeichnis

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z