

7 Datenmodellierung und Datenbanken

Überall dort, wo es um die Aufbewahrung oder Verarbeitung von großen Datenmengen geht, sind Datenbanksysteme im Spiel. Diese „Informations-Massenlager“ sind darauf spezialisiert, den Umgang mit großen Datenmengen zu vereinfachen. Dafür wird meist eine standardisierte Schnittstelle zur Verfügung gestellt.

Der Umgang mit einem Datenbanksystem ist relativ unproblematisch, falls die zugrundeliegende Datenstruktur gut an das Problem angepaßt ist und einige Randbedingungen erfüllt. Zur die Entwicklung solcher Datenstrukturen gibt es einige bewährte Techniken, die wir nun kennenlernen wollen.

Den gesamten Entwicklungsprozeß wollen wir anhand speziellen Beispiels aus dem Bibliotheksbereich veranschaulichen. Allerdings verzichten wir dabei auf eine vollständige Behandlung des gesamten Systeme und besprechen jeweils nur Ausschnitte daraus.

7.1 Informelle Beschreibung des Systems

Anfangs veranstalten wir eine Art „Brainstorming“, indem wir den gesamten für uns relevanten Wirklichkeitsausschnitt genau ansehen. Dazu holen wir alle nötigen Informationen über die Anforderungen ein, informieren uns über Randbedingungen, auf deren Gestaltung wir keinen Einfluß haben, bestimmen die Grenzen des Systems und seine geplante Funktionalität. Die späteren Nutzer werden ausführlich nach ihren Wünschen und Vorstellungen befragt.

In unserem Beispiel soll eine kleine Bibliothek elektronisch verwaltet werden. Dazu müssen Informationen zu Buchtiteln, Exemplaren, Autoren und Bibliothekskunden gespeichert und verarbeitet werden können.

Die Bibliothek verfügt über einen Internet-Anschluß, weshalb das Frontend des Systems über einen Webbrowser bedient werden soll.

7.2 Datenmodellierung

Nach der „weichen“ informellen Beschreibung ist es nun Zeit, das System „hart“ zu beschreiben, also formal zu beschreiben. Dafür gibt es eine Reihe von Techniken, die je nach den Eigenheiten des Systems Vor- und Nachteile haben. Im Datenbankbereich ist die Entity-Relationship Modellierung die mit Abstand häufigste und erfolgreichste Technik.

7.2.1 Entity-Relationship Modell

Nun entwerfen wir ein solches Entity-Relationship Modell für unser System. Es besteht aus einer Menge von Entitätsklassen und Beziehungen zwischen diesen Entitätsklassen. Solange wir nicht (mathematisch) exakt die Bedeutung der einzelnen Symbole von ER-Modellen festgelegt haben, haben wir es allerdings genau genommen immer noch mit einer informellen (intuitiven) Modellierungstechnik zu tun.

Entitäten

Der erste Schritt besteht in der Identifizierung der Entitäten. Hierbei handelt es sich um individuelle und identifizierbare Elemente, Objekte, Individuen, Sachen, Begriffen, Ereignisse o.ä. innerhalb des Systems. Diese Entitäten werden durch ihre Eigenschaften (Attribute) beschrieben:

Titel:	Fräulein Smillas Gespür für Schnee
Autor:	Peter Høeg
Verlag:	Carl Hanser
Ort:	München
Jahr	1994
ISBN	3-499-13599-x
Preis	19,90
Exemplare	3
Zustand	gut, sehr gut, gut
Standort	Zimmer 3, Regal 5, Fach 7,7 und 8, Platz 3, 15,12

Name	Høeg
Vorname	Peter
Land	Dänemark
andere Buchtitel	„Der Plan von der Abschaffung des Dunkels“

Name	Meier
Vorname	Peter
Geburtsdatum	24.4.1966
Straße	Tegtmüllerweg 9
PLZ	80089
Ort	München
Telefon	(089) 383 245 12
Sperre	keine

Entitätsklassen

Entitäten mit gleichen Eigenschaften werden dann unter einem Oberbegriff zu Entitätsklassen zusammengefaßt. In unserem Fall stellen wir fest, daß die Möglichkeit, mehrere Exemplare je Buch zu verwalten, bei der Klassifizierung Probleme macht. Wir führen also eine eigene Entität für die Exemplare ein. Bei den Personen stellen wir Unterschiede bei den Attributen von Autoren und Ausleihern fest. Damit erhalten wir vorerst die folgenden Entitätsklassen (mit den Attributen in Klammern):

- **Buchtitel** (Titel, Autor, Verlag, Ort, Jahr, ISBN, Preis)
- **Exemplar** (Bezeichnung, Zustand, Aufnahmedatum)
- **Standort** (Zimmer, Regal, Fach, Platz)
- **Autor** (Name, Vorname, Land, andere)
- **Kunde** (Name, Vorname, Geburtsdatum, Straße, PLZ, Ort, Telefon, Sperre)

Allerdings haben wir immer noch einige Probleme:

- Wie sollen die Zuordnungen von Buchtiteln zu Autoren, Exemplaren und Kunden geregelt werden ?
- Wie soll man mit der Liste weiterer Veröffentlichungen (andere) bei den Autoren umgehen?

Beziehungen

Die Lösung wird durch das Konzept der Beziehungen (relationships) zwischen Entitätsklassen geliefert. Buchtitel und Autor werden durch die Beziehung **verfasst_von** verbunden. Damit erübrigt sich gleichzeitig die Verwaltung von „weiteren Exemplaren“. Buchtitel und

Exemplar können durch die Beziehung **ist_vorhanden** verknüpft werden, Exemplar und Kunde durch die Beziehung **ausgeliehen_von**, Exemplar und Standort durch **steht_in**.

Ein genauer Blick auf die Situation zeigt uns, daß auch die Beziehungen Eigenschaften (Attribute) haben können. So ist es etwa sinnvoll, unsere Beziehung **ausgeliehen_von** mit den Attributen Datum und Bearbeiter zu versehen.

ER-Diagramme

Um die Übersicht zu bewahren, stellt man ER-Modelle meist graphisch in speziellen ER-Diagrammen dar. Seine Elemente sind Entitätsklassen (Rechtecke), Beziehungen zwischen diesen Entitätsklassen (Rauten) und Eigenschaften (Attribute) der Entitätsklassen und Beziehungen, symbolisiert durch Ellipsen. In unserem Fall erhalten wir das ER-Diagramm aus Abbildung 1.

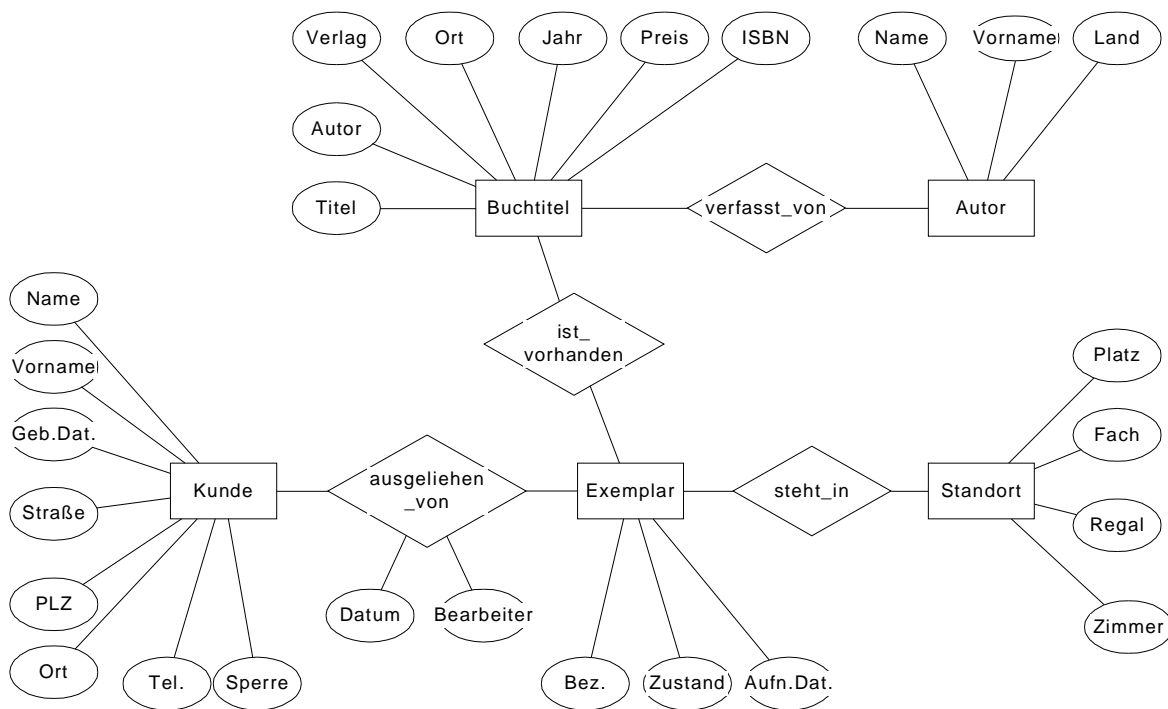


Abbildung 1: ER-Diagramm der Bibliothek

In umfangreicheren ER-Diagrammen läßt man die Attribute zugunsten der Übersichtlichkeit meist weg. Diese müssen dann allerdings getrennt in eigenen Tabellen notiert werden.

Zur Umsetzung des Entwurfs in ein reales Datenbanksystem ist es von großer Bedeutung, wieviele Entitäten der einen Seite durch eine bestimmte Beziehung mit Entitäten der anderen Seite verbunden werden können. Dies Eigenschaft heißt Kardinalität der Beziehung. In einem groben Ansatz gibt es dafür drei Möglichkeiten:

• **Kardinalität 1:1**

Einer Entität der einen Seite wird genau eine Entität der anderen Seite zugeordnet und umgekehrt.

Exemplar	steht_in	Standort
10220 ¹	↔	Zimmer 2, Regal 2, Fach 12, Platz 14

12110	↔	Zimmer 3, Regal 3, Fach 1, Platz 12
23311	↔	Zimmer 1, Regal 12, Fach 12, Platz 2

¹Über die Problematik der Identifikation von Entitäten über spezielle Schlüsselattribute werden wir uns später Gedanken machen

- **Kardinalität 1:n**

Einer Entität der einen Seite können mehrere Entitäten der anderen Seite zugeordnet werden, umgekehrt aber höchstens eine Entität

Buchtitel	ist_vorhanden	Exemplar
Fräulein Smillas ..	↔	10223
Fräulein Smillas ..	↔	10224
Fräulein Smillas ..	↔	10225

- **Kardinalität n:m**

Einer Entität der einen Seite können mehrere Entitäten der anderen Seite zugeordnet werden und umgekehrt. Dieser Fall kann immer in zwei Beziehungen mit den Kardinalitäten 1:m bzw. n:1 aufgelöst werden.

Buchtitel	verfaßt_von	Autor
Physik, Jahrgangsstufe 8	↔	Herbert Knauth
Physik, Jahrgangsstufe 8	↔	Siegfried Kühnel
Physik, Jahrgangsstufe 8	↔	Hubert Schafbauer
Fräulein Smillas Gespür für Schnee	↔	Peter Høeg
Der Plan von der Abschaffung des Dunkels	↔	Peter Høeg

Diese n:m-Beziehung könnte man auflösen in

verfaßt_u.a._von (1:n) und **hat_verfaßt** (m:1)

Im ER-Diagramm notieren wir die Kardinalitäten mit Hilfe der entsprechenden Zahlen am linken und rechten Anknüpfungspunkt der Beziehungen.

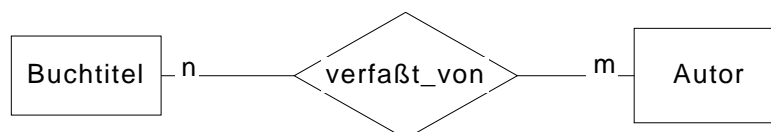


Abbildung 2: Notation der Kardinalität im ER-Diagramm

7.2.2 Relationale Modellierung

Parallel zum ER-Modell könnte man auch gleich von Beginn an ein relationales Modell aufbauen, mit dem man bereits auf der Ebene der Implementierung in einer relationalen Datenbank (siehe unten) angekommen wäre. Die Einhaltung von Normalformen sorgt bei diesem Vorgehen dafür, daß am Ende eine brauchbare Datenstruktur entsteht. In der Regel ist das Ergebnis dieser Strategie identisch mit dem Produkt, das man durch Umformung eines guten ER-Modells in ein relationales Modell erhält. Diese Umformung wird im nächsten Kapitel beschrieben.

Relationen

Der Name des Modells stammt vom mathematischen Begriff der Relation ab. Eine mathematische Relation R ist eine Menge von Tupeln $r = (r_1, \dots, r_n)$, deren Komponenten r_i jeweils aus einer bestimmten Grundmenge R_i stammen. Dann ist die Menge aller Tupel genau das Kartesische Mengenprodukt $R_1 \times R_2 \times \dots \times R_n$. Eine Relation R ist eine Teilmenge dieses Produktes:

$$R = \{r \mid r = (r_1, \dots, r_n) \wedge r_1 \in R_1, \dots, r_n \in R_n\} \subset R_1 \times \dots \times R_n$$

Ein entscheidendes Merkmal einer Relation ist ihre (feste) Stelligkeit, d.h. die Anzahl der Elemente ihrer Tupel. In der Informatik verwendet man als Grundmengen anstelle der in der Mathematik üblichen Zahlenmengen oft selbstdefinierte Mengen von Bezeichnern (Namen), sogenannte Sorten oder Typen, wie etwa **Ort** = {Bad Aibling, Kolbermoor, München, Rosenheim, Großkarolinenfeld, ...} oder **Männer** = {Ernst Huber, Franx Muxeneder, Hanno Buckmann, ...}

Beispiele für Relationen:

- Wir betrachten die Mengen $A = \{2, 3, 4\}$ und $B = \{5, 6, 8\}$ sowie die Relation $R_1 = \{(a, b) \in A \times B \mid a \text{ teilt } b\}$. Dann besteht das kartesische Produkt $A \times B$ aus den allen Paaren (2-Tupeln), die man durch Kombination eines Elementes aus A und eines Elementes aus B bilden kann, also aus $3 \cdot 3 = 9$ Paaren :

$$A \times B = \{(2, 5), (2, 6), (2, 8), (3, 5), (3, 6), (3, 8), (4, 5), (4, 6), (4, 8)\}$$

R_1 enthält dagegen nur die Paare $(a, b) \in A \times B$, für welche die Relationsbedingung „a teilt b“ erfüllt ist:

$$R_1 = \{(2, 6), (2, 8), (3, 6), (4, 8)\}.$$

- Die Relation $R_2 = \text{ist_verheiratet_mit}$ ist eine Teilmenge des 2-stelligen Mengenproduktes **Männer** \times **Frauen**:
(Anna Müller, Ernst Huber) $\in R_2 \Leftrightarrow$ Anna Müller **ist_verheiratet_mit** Ernst Huber
- Die 3-stellige Relation $R_3 = \text{Adresse}$ ist ein Teil des Mengenproduktes **Straße** \times **PLZ** \times **Ort**. Dazu könnten die folgenden 3-Tupeln gehören:

(Sudetenstr. 16, 83059, Kolbermoor), (Westendstr. 4a, 83043 Bad Aibling)

Je höher die Stelligkeit einer Relation ist, desto übersichtlicher wird es, sie in Tabellenform zu beschreiben. Unser letztes Beispiel könnte dann etwa so aussehen:

Sudetenstr. 16	83059	Kolbermoor
Westendstr. 4a	83043	Bad Aibling
Römerstr. 11	80333	München

Relationales Modell

Ein relationales Modell besteht aus einer Menge von Tabellen (T_1, \dots, T_n), die jeweils aus einer (grundsätzliche unbeschränkten) Menge von Datensätzen (Tupeln) mit gleicher Struktur

bestehen: $T_i = \{d_1, \dots, d_k\}$. Ein Datensatz d_i enthält eine Reihe von Daten, die als Werte der Attribute aufgefaßt werden können. $d_k = (x_{k1}, \dots, x_{km})$. Die Struktur einer Tabelle wird durch eine Liste von Attributnamen (a_1, \dots, a_m) festgelegt.

$T_1 = \text{Autor}$

a_1	a_2	a_3	
Name	Vorname	Land	
Høeg	Peter	Dänemark	d_1
Knauth	Herbert	Deutschland	d_2
Kühnel	Siegfried	Deutschland	d_3

$T_2 = \text{Buchtitel}$

a_1	a_2	a_3	a_4
Titel	Verlag	Ort	Jahr
Physik, Jahrgangsstufe 8	Oldenbourg	München	1994
Fräulein Smillas Gespür für Schnee	Hanser	München	1991
Der Plan von der Abschaffung des Dunkels	Hanser	München	1996

Wertemengen

Jedes Attribut a_i kann Werte aus einer bestimmten Wertemenge (*Sorte* oder *Typ*) S_i annehmen. Zum Beispiel in der Tabelle T_2 :

$S_1 = \text{Text}$,	(Standardsorte)
$S_2 = \text{Verlagsname}$	= {Springer, Hanser, Oldenbourg, O'Reilly, ... }
$S_3 = \text{Ortsbezeichnung}$,	= {München, Wien, Zürich, New York, .. }
$S_4 = \text{Jahresangabe}$	= {1500, 1501, 1502, ... 9999 }

Die Festlegung dieser Mengen (im Datenbankjargon auch *Domänen* genannt) erlaubt zumindest prinzipiell eine gewisse Kontrolle der Gültigkeit (Validität) von Benutzereingaben. Oft wird dann jedoch bei der Implementierung aus Effizienzgründen darauf verzichtet und eine Standardsorte verwendet.

Schema

Die Struktur der Tabelle wird in kompakter Weise durch das sogenannte *Schema* festgelegt:

$$T_k (a_1:S_1, \dots, a_m:S_m),$$

etwa **Buchtitel**(Titel: **Text**, Verlag: **Verlagsname**, Ort: **Verlagsort**, Jahr: **Jahresangabe**).

Mathematisch gesehen handelt es sich bei einer Tabelle mit dem Schema $S = (a_1:S_1, \dots, a_m:S_m)$ um eine Relation:

$$R \subset S_1 \times \dots \times S_m$$

• Punktnotation

Falls man es mit mehreren Tabellen zu tun hat, wird zur Vermeidung von Unklarheiten die Tabelle, zu der das Attribut gehört, (durch einen Punkt getrennt) dem Attributnamen vorangestellt:

$$T_k.a_i$$

bezeichnet somit den Namen des i -ten Attributs der Tabelle T_k . Beispiele dafür sind

Autor.Name, Buchtitel.Verlag.

Schlüssel

Wenn wir annehmen, daß es in unserer Tabelle **Autoren** zwei (verschiedene) deutsche Autoren mit dem Namen Hans Meier gibt, bekommen wir Probleme, da diese Datensätze nicht unterscheidbar sind. Will man beispielsweise die Anzahl der Buchtitel eines der beiden Autoren feststellen, so werden automatisch auch die Bücher des anderen Hans Meier mitgezählt.

Jede Tabelle (Relation) muß also ein Attribut oder eine Kombination von Attributen enthalten, die sicherstellen, daß auf jeden einzelnen Datensatz zugegriffen werden kann. Ein solches Attribut bzw. eine solche Kombination von Attributen heißt *Schlüssel* der Tabelle T bzw. der Relation R.

Mathematisch ausgedrückt gilt (im Fall eines einzelnen Attributs):

a_i heißt *Schlüssel* einer Relation R \Leftrightarrow

für zwei Tupel $r, s \in R$ mit $r = (r_1, \dots, r_m)$ und $s = (s_1, \dots, s_m)$ gilt: $r_i = s_i \Rightarrow r = s$.

Zumindest die Kombination aller Attribute sollte auf jeden Fall ein Schlüssel der Tabelle sein. Andernfalls enthält die Tabelle identische Datensätze. In unserer Autorentabelle müssen wir also mindestens ein neues Attribut einführen, um einen Schlüssel zu erhalten. Mit großer Wahrscheinlichkeit wäre dann die Kombination aller Attribute ein brauchbarer Schlüssel. Da jedoch auch diese Eindeutigkeit nicht absolut sicher ist, greift man in der Datenbanktechnik meist auf *künstliche Schlüsselattribute* (Identifikationsnummern) zurück, die per definitionem eindeutig sind. Wir ändern also das Schema unserer Autoren:

Autor (Autor_nr: **Integer**, Name: **Text**, Vorname: **Text**, Land: **Ländername**).

Damit haben wir den künstlichen Schlüssel **Autor_nr** eingeführt. In der tabellarischen Darstellung von Relationen unterstreichen wir ab jetzt die Schlüsselattribute.

7.2.3 Optimierung des relationalen Modells

Bei Verzicht auf ER-Modellierung sind die ersten Entwürfe des relationalen Modells meist in mehrfacher Hinsicht nicht optimal. Um diese Schwachstellen zu beseitigen, gibt es eine Reihe von Normalformen, die sicherstellen, daß der Entwurf datentechnisch zumindest. brauchbar ist.

Erste Normalform (1NF)

Eine Tabelle ist in erster Normalform, falls alle **Attribute nur atomare Werte annehmen können**. Es dürfen also Mengen, Aufzählungstypen oder Wiederholungsgruppen als Werte der Attribute auftreten.

Nicht in erster Normalform wäre die folgende Tabelle:

Kunde

Name	Vorname	Adresse
Müller	Anna	Sudetenstr. 18, 83222, Rosenheim
Huber	Karl	Knallerweg 19, 86321, Ganselham

Die erkennt man bereits am Schema

Kunde(Name: **Text**, Vorname: **Text**, Adresse: **Text x PLZ x Ortsname**)

Eine Überführung in die erste Normalisierung müßte das Attribut Adresse in drei Attribute mit atomaren Wertebereichen aufteilen:

Kunde

Name	Vorname	Straße	PLZ	Ort
Müller	Anna	Sudetenstr. 18	83222	Rosenheim
Huber	Karl	Knallerweg 19	86321	Ganselham

Zweite Normalform (2NF)

Beim Entwurf eines Bibliothekssystems (ohne ER-Modell!) könnte ein Entwickler auf die folgende Tabelle kommen:

Bücher

<u>Titel Nr</u>	Titel	Autor	<u>Exemplar nr</u>	Zimmer	Regal	Fach	Platz
1222	Fräulein Smilla...	Peter Høeg	01	2	3	12	3
1222	Fräulein Smilla	Peter Høeg	02	2	3	12	4
1222	Fräulein Smilla	Peter Høeg	03	2	3	12	5
1333	Das Parfüm	Patrick Süßkind	01	2	3	13	1
1333	Das Parfüm	Patrick Süßkind	02	2	3	13	2
1333	Das Parfüm	Patrick Süßkind	03	2	3	13	3

Bei der Arbeit mit dieser Tabelle treten u.a. die folgenden Probleme auf:

- ☞ Datenredundanz: Für jedes Exemplar werden Titel und Autor erneut eingetragen, obwohl die nötige Information bereits durch das erste Exemplar vorhanden ist.
- ☞ Ungewollter Datenverlust: Sind vorübergehend, etwa wegen Beschädigung, keine Exemplare eines Buches mehr vorhanden, so gehen die Informationen über Titel und Autor verloren. Bei der Neuanschaffung müssen diese Informationen wieder besorgt werden.
- ☞ Mögliche Inkonsistenz: Falls bei der Eintragung eines neuen Exemplars von „Das Parfüm“ die Datentypistin beispielsweise aus Versehen den Autor Edgar Wallace einträgt, gibt es den Titel „Das Parfüm“ jetzt von zwei verschiedenen Autoren.

Das Problem geht offensichtlich darauf zurück, daß für die gesamte Tabelle mindestens die Attributkombination **Titel_nr** und **Exemplar_nr** als Schlüssel gebraucht wird, während für einige Attribute wie Titel und Autor bereits **Titel_nr** als Schlüssel ausreicht.

- **Funktionale Abhängigkeit**

Um dieses Problem kompakter und exakter beschreiben zu können, führen wir den Begriff der Funktionalen Abhängigkeit ein:

Ein Attribut a_i einer Relation R heißt *funktional abhängig* von einem Attribut a_k , falls für alle Tupel $r, s \in R$ gilt:

$$r_k = s_k \Rightarrow r_i = s_i .$$

Wir schreiben dann kurz: $a_k \rightarrow a_i$.

Falls ein Attribut a_i ist von einem Attribut a_k funktional abhängig ist, können zwei Tupel also nur dann verschiedene Werte von a_i aufweisen, wenn sich auch die Werte von a_k unterscheiden.

Für die Funktionale Abhängigkeit von einer Kombination zweier Attribute gilt dann entsprechend:

$$a_k a_m \rightarrow a_i \quad \Leftrightarrow \quad \forall r, s \in R : r_k = s_k \wedge r_m = s_m \Rightarrow r_i = s_i .$$

In unserem Beispiel sind die Attribute **Autor**, **Titel** von **Titel_nr** funktional abhängig. Dagegen hängen **Zimmer**, **Regal**, **Fach** und **Platz** funktional von **Titel_nr** und **Exemplar_nr**. ab:

Titel_nr \rightarrow Titel
 Titel_nr \rightarrow Autor
 Titel_nr, Exemplar_nr \rightarrow Zimmer
 Titel_nr, Exemplar_nr \rightarrow Regal
 Titel_nr, Exemplar_nr \rightarrow Fach
 Titel_nr, Exemplar_nr \rightarrow Platz.

Zwei Datensätze mit verschiedenen Autoren müssen sich also auch in der Titel_nr unterscheiden. Falls die Regale oder die Plätze unterschiedlich sind, dürfen die Kombinationen der Titel_nr und der Exemplar-nr nicht identisch sein.

Als Schlüssel ist in unserer Tabelle ist daher nur die Kombination aus **Titel_nr** und **Exemplar_nr** brauchbar. Das Problem liegt darin, daß einige Attribute bereits von einem Teil der Schlüsselkombination funktional abhängig sind. Genau diese Abhängigkeit wird durch die 2. Normalform ausgeschlossen.

- **2. Normalform**

Eine Tabelle ist in zweiter Normalform, wenn sie in erster Normalform ist und wenn jedes Attribut, das zu keiner Schlüsselkombination gehört, nur **von der gesamten Schlüsselkombination**, nicht jedoch bereits von einem Teil davon funktional abhängig ist.

Die Umformung in die zweite Normalform bringt daher eine Abspaltung neuer Tabellen mit den kritischen Attributen mit sich, wobei jeweils der Teilschlüssel, von dem die problematische Abhängigkeit vorlag, als neuer Schlüssel verwendet wird. Die anderen Teile

des ursprünglichen Schlüssels verbleiben dennoch in der Ausgangstabelle, um die ursprünglichen Zuordnungen zu erhalten.

In unserem Beispiel wird eine Tabelle **Buchtitel** mit den Attributen **Titel_nr**, **Titel** und **Autor** abgespalten. Die Tabelle **Bücher** wird sinnvollerweise in **Exemplare** umbenannt. Die Schlüssel von Exemplar bleibt weiterhin die Kombination aus **Exemplar_nr** und **Buchtitel_nr**.

Buchtitel

<u>Titel_nr</u>	Titel	Autor
1222	Fräulein Smillas..	Peter Høeg
1333	Das Parfüm	Patrick Süßkind

Exemplare

<u>Exemplar_nr</u>	<u>Titel_nr</u>	Zimmer	Regal	Fach	Platz
01	1222	2	3	12	3
02	1222	2	3	12	4
03	1222	2	3	12	5
01	1333	2	3	13	1
02	1333	2	3	13	2
03	1333	2	3	13	3

Dritte Normalform (3NF)

Trotz Vorliegens der 2. Normalform kann es noch zu Datenredundanzen kommen. Wir betrachten unsere obige Kundentabelle, die wir noch um einen künstlichen Schlüssel **Kunde_nr** ergänzt haben.

Kunde

<u>Kunde_nr</u>	Name	Vorname	Straße	PLZ	Ort
00012	Müller	Anna	Sudetenstr. 18	83222	Rosenheim
00013	Huber	Karl	Knallerweg 19	86321	Ganselham
00014	Meier	Amelie	Körperweg 18	83222	Rosenheim
00015	Hanser	Kurt	Kästnerstr. 10 A	83222	Rosenheim

Wegen des atomaren Schlüssels ist die Tabelle in 2. Normalform. Das Attribut **Ort** enthält jedoch redundante Daten, da bereits das Attribut PLZ die Werte von Ort eindeutig festlegt. Aus der Sicht funktionaler Abhängigkeit gilt hier:

Kunde_nr → PLZ → Ort

Eine solche Kette funktionaler Abhängigkeiten heißt *transitive funktionale Abhängigkeit*. Genau diese Eigenschaft der Tabelle verursacht die Datenredundanzen. Wir fordern also für die dritte Normalform:

Eine Tabelle ist in dritter Normalform, wenn sie sich in zweiter Normalform befindet und wenn kein Nichtschlüsselattribut **transitiv abhängig** von irgendeinem Schlüsselattribut ist.

Die Umformung in die dritte Normalform zwingt zu einer erneuten Aufspaltung, wobei aus dem „mittleren“ Schlüssel in der Kette der Abhängigkeiten zusammen mit den davon abhängigen Attributen eine neue Tabelle gebildet wird. Das neue Schlüsselattribut verbleibt aber auch in der ursprünglichen Tabelle, um Informationsverluste auszugleichen. Die davon abhängigen Attribute verschwinden jedoch. In unserem Beispiel:

T₁: Kunde

Kunde_nr	Name	Vorname	PLZ	Straße
00012	Müller	Anna	83222	Sudetenstr. 18
00013	Huber	Karl	86321	Knallerweg 19
00014	Meier	Amelie	83222	Körperweg 18
00015	Hanser	Kurt	83222	Kästnerstr. 10 A

T₂: PLZ

PLZ	Ort
83222	Rosenheim
86321	Ganselham

7.2.4 Umsetzung von ER-Modellen in Relationale Modelle

Von einem gut durchdachten ER-Modell ausgehend kann man in der Regel schnell und einfach ein sauberes relationales Modell erstellen. Die Umwandlung erfolgt mit Hilfe relativ einfacher Regeln, für die nur relativ selten Ausnahmen zu beachten sind.

Entitäten

Entitäten werden zu Tabellen, Attribute zu Spalten. In jeder Tabelle wird ein Schlüsselattribut (oder eine Kombination von Attributen als Schlüssel) definiert.

Wir erhalten aus unserem ER-Modell der Bibliothek für die Entitäten die folgenden Tabellenschemata:

Buchtitel (Buchtitel_nr, Titel, Autor, Verlag, Ort, Jahr, ISBN, Preis)

Exemplar (Exemplar_nr, Bezeichnung, Zustand, Aufnahmedatum)

Standort (Standort_nr, Zimmer, Regal, Fach, Platz)

Autor (Autor_nr, Name, Vorname, Land, andere)

Kunde (Kunde_nr, Name, Vorname, Geburtsdatum, Straße, PLZ, O

Beziehungen

- **Kardinalität 1:1**

Beziehungen der Kardinalität 1:1 können in eine der beiden Tabellen der beteiligten Entitäten eingebaut werden, indem man das Schlüsselattribut der anderen Tabelle aufnimmt. Dieses Attribut heißt dort *Fremdschlüssel*.

Die 1:1-Beziehung **steht_in** wird durch den Fremdschlüssel Standort_nr in die Tabelle Exemplar aufgenommen. Ebenso gut könnte man die Tabellen Exemplar und Standort zu einer einzigen vereinigen, falls man keine anderweitigen Beziehungen der beiden Entitäten findet.

Exemplar (Exemplar_nr, Bezeichnung, Zustand, Aufnahmedatum, **Standort_nr**)

- **Kardinalität 1:n**

Beziehungen mit der Kardinalität 1:n werden umgewandelt, indem man das Schlüsselattribut der 1-Seite als zusätzliches Attribut in der Tabelle der n-Seite aufnimmt.

Unsere 1:n-Relation **ist_vorhanden** wird durch den Fremdschlüssel **Buchtitel_nr** in der Tabelle **Exemplare** realisiert:

Exemplar (Exemplar_nr, Bezeichnung, Zustand, Aufnahmedatum, **Buchtitel_nr**)

- **Kardinalität m:n**

Beziehungen mit der Kardinalität m:n werden als eigene Tabelle, die mindestens die Schlüsselfelder der beiden beteiligten Entitätentabellen enthält, umgesetzt.

So wird aus unserer Beziehung **verfaßt_von** nun die Tabelle

Verfaßt_von (Buchtitel_nr, Autor_nr)

- **Ausnahme**

Falls eine Relation eigene Attribute hat, die auch anderweitig von Belang sind, ist es entgegen dieser Regeln meist günstiger, eine eigene Tabelle für die Relation zu definieren.

Die 1:n-Relation **ausgeliehen_von** zwischen Kunde und Exemplar könnte man mitsamt den beiden Attributen Ausleihdatum und Bearbeiter in die Tabelle Exemplar einbauen. Spätestens, wenn die Verwaltung einer Entität Bearbeiter für die Personalangehörigen in Betracht kommt, wird jedoch auch dafür eine eigene Tabelle sinnvoll:

Ausgeliehen_von (Kunde_nr, Exemplar_nr, Ausleihdatum, Bearbeiter_nr).

7.3 Abfragen und Berichte

7.3.1 Relationenalgebra

Für das Verständnis von Abfragesprachen wie SQL ist es sehr günstig einige mathematische Grundlagen mit Hilfe des relationalen Modells zu legen. Wir führen dazu die folgenden Operationen auf Relationen (Tabellen) ein.

- **Projektion** $P(a_1, \dots, a_k)(R) = \{(x_1, \dots, x_k) \mid r \in R \wedge r.a_i = x_i\}$

Mit Hilfe der Projektion $P(a_1, \dots, a_k)$ kann man die Attribute (Spalten) a_1, \dots, a_k aus der Tabelle extrahieren. $r.a_i = x_i$ steht dabei für die Aussage, daß das Attribut a_i im Tupel r den Wert x_i annimmt.

$P(\text{Titel, Autor})(\text{Buchtitel}) =$
 $\{(\text{Fräulein Smillas..}, \text{Peter Høeg}), (\text{Das Parfüm}, \text{Patrick Süßkind}) \dots\}$

- **Auswahl** $W_P(R) = \{r \mid r \in R \wedge P(r) = \text{wahr}\}$

Mit dieser sehr mächtigen Operation kann eine Teilmenge der Tupel ausgesucht werden, die eine bestimmte Aussageform erfüllen.

Mit $P = [\text{Ort} = \text{Rosenheim}]$ wird dann

$W(P)(\text{Kunde}) = \{(00012, \text{Müller}, \text{Anna}, \dots)\}$

(00014, Meier, Amelie, ..),
 (00015, Hanser, Kurt, ..)}

- **Kreuzprodukt** $R \times S = \{(r_1, \dots, r_n, s_1, \dots, s_m) \mid (r_1, \dots, r_n) \in R \wedge (s_1, \dots, s_m) \in S\}$
 Das Kreuzprodukt kombiniert jedes Tupel aus R mit allen Tupeln aus S, wobei die Schemata vereinigt werden. Es ist vor allem als Grundmenge für Join-Operationen (s.unten) sehr wichtig.

Wir betrachten das ganze am Beispiel zweier Tabellen **Konto** und **Kunde** aus dem Bankbereich.

Kunde

Name	Vorname	Konto_nr
Meier	Hans	364 234
Müller	Anna	23 244
Huber	Karl	123 444

Konto

Konto_nr	Saldo	Kreditrahmen
364 234	+1298	12000
23 244	+13455	12000
123 444	-1099	6000

Kunde x **Konto** ergibt dann die folgende auf den ersten Blick ziemlich sinnlose Kombination:

Kunde. Name	Kunde. Vorname	Kunde. Konto_nr	Konto. Konto_nr	Konto. Saldo	Konto. Kreditrahmen
Meier	Hans	364 234	364 234	+1298	12000
Meier	Hans	364 234	23 244	+13455	12000
Meier	Hans	364 234	123 444	-1099	6000
Müller	Anna	23 244	364 234	+1298	12000
Müller	Anna	23 244	23 244	+13455	12000
Müller	Anna	23 244	123 444	-1099	6000
Huber	Karl	123 444	364 234	+1298	12000
Huber	Karl	123 444	23 244	+13455	12000
Huber	Karl	123 444	123 444	-1099	6000

Hier haben wir die Schreibweise T.a für ein Attribut a aus der Tabelle T verwendet, um die Zugehörigkeit der Attribute deutlich zu machen.

- **Equi-Join** $R \times_{R.a_i=S.b_k} S = \{(r_1, \dots, a_n, b_1, \dots, b_m) \mid a_i = S.b_k\}$

Der Equi-Join liefert die Menge aller Tupel aus $R \times S$, bei denen die Werte der angegebenen Attribute identisch sind. Es kann eine beliebige Anzahl von Gleichheitsforderungen angegeben werden.

Kunde $\times_{\text{Kunde.Konto_nr}=\text{Konto.Konto_nr}}$ **Konto** ergibt die folgende Tabelle:

Kunde. Name	Kunde. Vorname	Kunde. Konto_nr	Konto. Konto_nr	Konto. Saldo	Konto. Kreditrahmen
Meier	Hans	364 234	364 234	+1298	12000
Müller	Anna	23 244	23 244	+13455	12000
Huber	Karl	123 444	123 444	-1099	6000

Eine Verallgemeinerung des Equi-Join ist der Theta-Join, bei dem man anstatt einer Gleichheitsforderung eine beliebige Vergleichsoperation wie $\geq, <$ etc. zulässt.

- **Natural Join** $R \otimes S$

Die Ergebnismenge des Natural Join der beiden Relationen R und S besteht aus allen Tupeln aus $R \times S$, bei denen diejenigen Attribute, die in **R und in S** vorkommen, den gleichen Wert haben. Der Natural Join ist ein Equi-Join, der alle gemeinsamen Attribute zweier Tabellen erfasst, ohne dass man diese explizit angeben muss. Er liefert die Kombination zweier Tabellen, die man am häufigsten benötigt. Die gemeinsamen Attribute werden sinnvollerweise nur noch einmal in der Tabelle aufgeführt.

In unserem Bankbeispiel ergibt **Kunde** \otimes **Konto** die folgende Tabelle, wobei das redundante Attribut **Konto.Konto_nr** weggelassen wird.

Kunde. Name	Kunde. Vorname	Kunde. Konto_nr	Konto. Saldo	Konto. Kreditrahmen
Meier	Hans	364 234	+1298	12000
Müller	Anna	23 244	+13455	12000
Huber	Karl	123 444	-1099	6000

Eine gewisse Problematik liegt in der Bestimmung der gemeinsamen Attribute, falls diese nicht völlig identische Namen haben. Man denke etwa an das folgende Tabellensystem:

Buchtitel (Buchtitel_nr, Titel, Autor, Verlag, Ort, Jahr, ISBN, Preis)

Autor (Autor_nr, Name, Vorname, Land, andere)

In **Buchtitel** \otimes **Autor** ist nicht unbedingt klar, daß die Attribute **Buchtitel.Autor** und **Autor.Name** als identisch zu betrachten sind. Im Zweifelsfall sollte man daher auf einen entsprechenden Equi-Join zurückgreifen.

Die folgenden Operationen Vereinigung, Durchschnitt und Differenz liefern nur dann wieder eine Relation, wenn man sich beim Ergebnis auf den Durchschnitt der Schemata von R und S, also auf die **gemeinsamen Attribute**, beschränkt. Der Einfachheit halber vereinbaren wir, diese drei Operationen nur auf Relationen mit identischen Schemata anzuwenden. In unserem Beispiel reduzieren wir dazu die Tabellen **Kunde** und **Autor** auf die gemeinsamen Attribute Name und Vorname.

- **Vereinigung** $R \cup S = \{ r \mid r \in R \vee r \in S \}$

Die Vereinigungsmenge besteht aus allen Tupeln, die entweder der einen Relation oder der anderen Relation angehören.

Kunde \cup **Autor** = { (Meier, Hans), (Müller, Emil), ...
(Høeg, Peter.), (Süßkind, Patrick), ... }

- **Durchschnitt** $R \cap S = \{ r \mid r \in R \wedge r \in S \}$

Der Durchschnitt besteht aus allen Tupeln, die sowohl der einen Relation als auch der anderen Relation angehören.

In unserem Beispiel liefert **Kunde** \cap **Autor** Name und Vorname aller Personen, die sowohl als Autor wie auch als Kunde registriert sind.

- **Differenz** $R - S = \{ r \in R \wedge r \notin S \}$

Die Differenz von R und S ist die Menge aller Tupel, die R, aber nicht S angehören.

Kunde - Autor ergibt die Namen und Vornamen aller Personen, die Kunde, aber nicht Autor sind.

7.3.2 Die Abfragesprache SQL

Zur Durchführung von Abfragen in realen Datenbanksystemen verwendet man nicht den oben beschriebenen mathematischen Relationenkalkül, sondern spezielle Abfragesprachen. Am meisten Verbreitung hat dabei die Abfragesprache SQL (Structured Query Language) gewonnen, die sich aus der von IBM für das Datenbanksystem R entwickelten Vorläufersprache SEQUEL entwickelt hat.

1) Abfragen aus einer Tabelle

- Die SELECT-Anweisung

Die Hauptkomponente einer SQL-Abfrage ist die SELECT-Anweisung. Ihre Syntax lautet

```
SELECT    <Attributliste>
FROM      <Tabellenliste>
WHERE     <Bedingungsliste>
```

Eine Abfrage nach den Titeln und dem Erscheinungsjahr aller Bücher, die von Peter Høeg nach 1990 erschienen sind, sieht in SQL so aus:

```
SELECT    Titel, Jahr
FROM      Buchtitel
WHERE     Autor = 'Peter Høeg' AND Jahr > 1990
```

- SELECT und Projektion

Der Vergleich mit dem Relationenkalkül liefert die exakte Bedeutung der SELECT-Anweisung:

```
SELECT    a, b
FROM      T
```

ist offensichtlich gleichbedeutend mit der Projektion $P(a,b)(T)$: Die Anweisung liefert die Spalten (Attribute) a, b aus der Tabelle T. Ein * gibt an, daß man alle Attribute der Tabelle haben will.

- WHERE und Auswahl

Die Anweisung

```
SELECT    *
FROM      T
WHERE     P
```

bewirkt eine Auswahl $W_{(P)}(T)$. Wir erhalten alle Datensätze, für welche die Aussage P wahr ist.

2) Verknüpfung mehrerer Tabellen

- kartesisches Produkt

Die Angabe mehrerer Tabellen nach FROM ohne einschränkende WHERE-Klausel (s.unten) liefert das kartesische Produkt der angegebenen Tabellen

```
SELECT *
```

```
FROM S, T
```

ist äquivalent mit $S \times T$. Zur Projektion von Spalten aus dem Produkt kann die Punktnotation verwendet werden:

```
SELECT Buchtitel.Titel, Buchtitel.Jahr, Autor.*
FROM Buchtitel, Autor
```

- Equi-Join

Der Equi-Join wird einfacherweise durch die Angabe der Vergleichsbedingung hinter WHERE realisiert:

```
SELECT *
FROM R, S
WHERE R.a = S.b
```

ist gleichbedeutend mit $R \underset{R.a_i=S.b_k}{\times} S$.

7.4 Literatur

Korth Henry F., Silberschatz Abraham: Database System Concepts. Mac Graw - Hill, New York 1991.

Horn C., Kerner I.O.(Hrsg.): Lehr- und Übungsbuch Informatik. Band 3: Praktische Informatik. Fachbuchverlag Leipzig, 1997.

Schwinn H.: Relationale Datenbanksysteme. Carl Hanser Verlag, München, Wien, 1992.

Zentralstelle für Computer im Unterricht [Hrsg.]: Datenbank. Arbeitskreis „Datenbanken im Unterricht“. Augsburg 1997.