2.4.2 Datenverarbeitungsanforderungen

Neben der Informationsstruktur muss in der Anforderungsanalyse natürlich auch der operationale Aspekt – also die Datenverarbeitung – behandelt werden. Es empfiehlt sich, diesen Bereich in Einzelprozesse zu zergliedern, für die dann jeweils separate Anforderungsbeschreibungen erstellt werden. Genau wie für die Informationsstrukturbeschreibung, hat sich auch hierfür eine strukturierte Dokumentation in der Praxis bewährt. Wir demonstrieren dies am Beispiel der Zeugnisausstellung:

- **Prozessbeschreibung:** *Zeugnisausstellung*
  - Häufigkeit: halbjährlich
  - benötigte Daten
    - Prüfungen
    - Studienordnungen
    - Studenteninformation
    - ...
  - Priorität: hoch
  - zu verarbeitende Datenmenge
    - 500 Studenten
    - 3000 Prüfungen
    - 10 Studienordnungen

Wenn dies geeignet erscheint, kann man natürlich andere, anwendungsspezifischere „Formulare" entwerfen. Die Formulare müssen auf jeden Fall so gestaltet sein, dass man sie als Diskussionsgrundlage mit den zukünftigen Anwendern verwenden kann.

2.5 Grundlagen des Entity-Relationship-Modells

Wie der Name schon sagt, sind die grundlegendsten Modellierungsstrukturen dieses Modells die *Entities* (Gegenstände) und die *Relationships* (Beziehungen) zwischen den Entities. Zusätzlich „kennt" das Entity-Relationship-Modell (kurz ER-Modell genannt) noch *Attribute* und *Rollen.*

Gegenstände (bzw. Entities) sind wohlunterscheidbare physisch oder gedanklich existierende Konzepte der zu modellierenden Welt. Man abstrahiert ähnliche Gegenstände zu Gegenstandstypen (Entitytypen oder Entitymengen), die man grafisch als Rechtecke darstellt, wobei der Name des Entitytyps innerhalb des Rechtecks angegeben wird.

Beziehungen werden auf analoge Weise zu Beziehungstypen zwischen den Gegenstandstypen abstrahiert. Die Beziehungstypen werden als Rauten mit entsprechender Beschriftung repräsentiert. Die Rauten werden mit den beteiligten Gegenstandstypen über ungerichtete Kanten verbunden.

Im Folgenden werden wir oft die Unterscheidung zwischen Gegenständen und den Gegenstandstypen, bzw. zwischen Beziehungen und Beziehungstypen, vernachlässigen. Aus dem Kontext dürfte immer leicht ersichtlich sein, was gemeint ist.

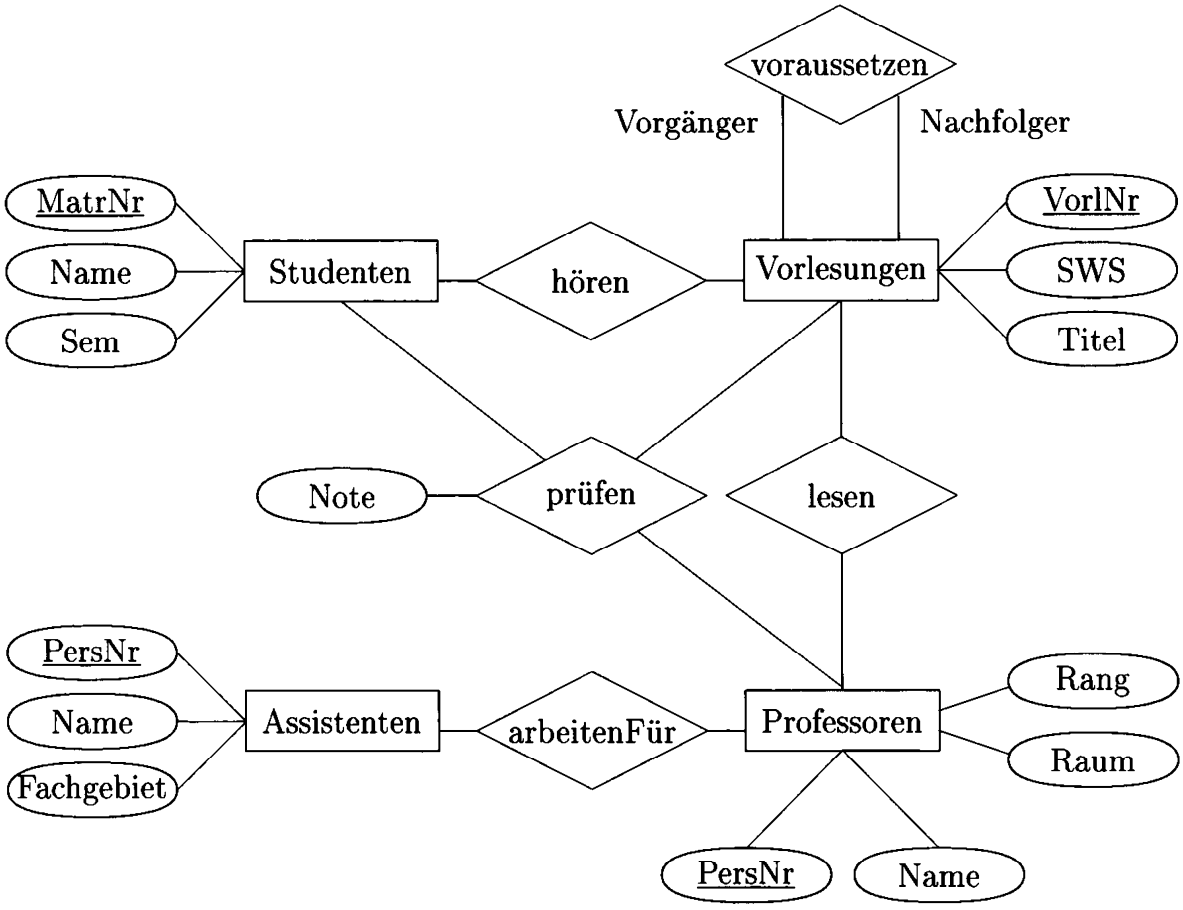

Abb. 2.3: Ein konzeptuelles Universitätsschema

Beispiele für Gegenstandstypen in der Universitätswelt sind *Studenten*, *Vorlesungen*, *Professoren* und *Assistenten*. Beziehungen zwischen diesen Entitytypen sind z.B. *hören* (zwischen *Studenten* und *Vorlesungen*) und *prüfen* (zwischen *Professoren*, *Vorlesungen* und *Studenten*).

Attribute dienen dazu, Gegenstände bzw. Beziehungen zu charakterisieren. Folglich werden Attribute, die durch Kreise oder Ovale grafisch beschrieben werden, den Rechtecken (für Gegenstandstypen) bzw. den Rauten (für Beziehungstypen) durch verbindende Kanten zugeordnet.

In dem in Abbildung 2.3 gezeigten ER-Schema gibt es vier Gegenstandstypen (*Studenten*, *Vorlesungen*, *Professoren*, *Assistenten*) und fünf Beziehungstypen (*hören*, *prüfen*, *voraussetzen*, *arbeitenFür*, *lesen*). Den Gegenstandstypen sind jeweils ein identifizierendes Attribut als Schlüssel und noch weitere beschreibende Attribute zugeordnet. Zum Beispiel werden *Studenten* durch die *MatrNr* (Matrikelnummer) eindeutig identifiziert; wohingegen der *Name* bzw. das *Semester* als weitere Attribute angegeben sind, die aber i.A. einen Studenten nicht eindeutig identifizieren, sondern nur zur (detaillierteren) Beschreibung dienen.

Die Beziehungen *hören*, *lesen* und *arbeitenFür* sind *binäre* Beziehungen zwischen zwei unterschiedlichen Entitytypen. Auch die Beziehung *voraussetzen* ist binär, aber es ist nur ein Entitytyp beteiligt. In diesem Fall spricht man von einer *rekursiven* Beziehung. Weiterhin wurden in der Beschreibung der Beziehung *voraussetzen* Rollen zugeordnet, nämlich *Vorgänger* und *Nachfolger*. Dadurch wird die Rolle eines

Gegenstandes in dieser Beziehung dokumentiert, d.h. in diesem Fall legen die Rollen fest, ob die betreffende Vorlesung als Nachfolger auf der anderen Vorlesung aufbaut oder umgekehrt. Rollen werden als Text an die jeweiligen „Ausgänge" (Kanten) der Beziehungsraute geschrieben.

2.6 Schlüssel

Eine minimale Menge von Attributen, deren Werte das zugeordnete Entity eindeutig innerhalb aller Entities seines Typs identifiziert, nennt man *Schlüssel.* Sehr oft gibt es einzelne Attribute, die als Schlüssel „künstlich" eingebaut werden, wie z.B. Personalnummer (*PersNr*), Vorlesungsnummer (*VorlNr*), etc. Schlüsselattribute werden durch Unterstreichung (manchmal auch durch doppelt gezeichnete Kreise bzw. Ovale) gekennzeichnet.

Manchmal gibt es auch zwei unterschiedliche Schlüsselkandidaten: Dann wählt man einen dieser Kandidaten-Schlüssel als Primärschlüssel aus.

2.7 Charakterisierung von Beziehungstypen

Ein Beziehungstyp R zwischen den Entitytypen $E_1, E_2, \ldots, E_n$ kann als Relation im mathematischen Sinn angesehen werden. Demnach stellt die Ausprägung der Beziehung R eine Teilmenge des kartesischen Produkts der an der Beziehung beteiligten Entitytypen dar. Also gilt:

$$R \subseteq E_1 \times E_2 \times \cdots \times E_n$$

In diesem Fall bezeichnet man n als den Grad der Beziehung R – die in der Praxis mit Abstand am häufigsten vorkommenden Beziehungstypen sind *binär*.

Ein Element $(e_1, e_2, \ldots, e_n) \in R$ nennt man eine Instanz des Beziehungstyps, wobei $e_i \in E_i$ für alle $1 \leq i \leq n$ gelten muss. Eine solche Instanz ist also ein Tupel aus dem kartesischen Produkt $E_1 \times E_2 \times \cdots \times E_n$.

Man kann jetzt auch den Begriff der Rolle etwas formaler fassen. Dazu veranschaulichen wir uns nochmals die Beziehung *voraussetzen* aus unserem Beispielschema (siehe Abbildung 2.3). Gemäß dem oben skizzierten Formalismus gilt:

$$\text{voraussetzen} \subseteq \text{Vorlesungen} \times \text{Vorlesungen}$$

Um einzelne Instanzen $(v_1, v_2) \in$ *voraussetzen* genauer zu charakterisieren, wird die jeweilige Rolle, nämlich (*Vorgänger* : v_1, *Nachfolger* : v_2), benötigt. Dadurch wird also unmissverständlich festgelegt, dass die Vorlesung v_1 die Voraussetzung für die Vorlesung v_2 darstellt.

2.7.1 Funktionalitäten der Beziehungen

Man kann Beziehungstypen hinsichtlich ihrer *Funktionalität* charakterisieren. Ein binärer Beziehungstyp R zwischen den Entitytypen E_1 und E_2 heißt

- 1:1-*Beziehung*, falls jedem Entity e_1 aus E_1 höchstens ein Entity e_2 aus E_2 zugeordnet ist und umgekehrt jedem Entity e_2 aus E_2 maximal ein Entity e_1 aus E_1 zugeordnet ist. Man beachte, dass es auch Entities aus E_1 (bzw. E_2) geben kann, denen kein „Partner" aus E_2 (bzw. E_1) zugeordnet ist.

  Ein Beispiel einer „realen" 1:1-Beziehung ist *verheiratet* zwischen den Entitytypen *Männer* und *Frauen* – zumindest nach europäischem Recht.

- 1:N-*Beziehung*, falls jedem Entity e_1 aus E_1 beliebig viele (also mehrere oder auch gar keine) Entities aus E_2 zugeordnet sein können, aber jedes Entity e_2 aus der Menge E_2 mit maximal einem Entity aus E_1 in Beziehung stehen kann.

  Ein anschauliches Beispiel für eine 1:N-Beziehung ist *beschäftigen* zwischen *Firmen* und *Personen*, wenn wir davon ausgehen, dass eine Firma i.A. mehrere Personen beschäftigt, aber eine Person nur bei einer (oder gar keiner) Firma angestellt ist.

- N:1-*Beziehung*, falls analoges zu obigem gilt.

- N:M-*Beziehung*, wenn keinerlei Restriktionen gelten müssen, d.h. jedes Entity aus E_1 mit beliebig vielen Entities aus E_2 in Beziehung stehen kann und umgekehrt jedes Entity aus E_2 mit beliebig vielen Entities aus E_1 assoziiert werden darf.

Man beachte, dass die Funktionalitäten Integritätsbedingungen darstellen, die in der zu modellierenden Welt immer gelten müssen. D.h. diese Bedingungen sollen nicht nur im derzeit existierenden Zustand der Miniwelt (rein zufällig) gelten, sondern sie sollen Gesetzmäßigkeiten darstellen, deren Einhaltung erzwungen wird.

Die binären 1:1-, 1:N- und N:1-Beziehungen kann man auch als *partielle Funktionen* ansehen. Bei einer 1:1-Beziehung R zwischen E_1 und E_2 kann die Funktion sowohl als $R : E_1 \rightarrow E_2$ wie auch als $R^{-1} : E_2 \rightarrow E_1$ gesehen werden.

Bezogen auf unser Beispiel einer 1 : 1-Beziehung haben wir also:

$$\begin{aligned} \text{Ehemann} &: \text{Frauen} \rightarrow \text{Männer} \\ \text{Ehefrau} &: \text{Männer} \rightarrow \text{Frauen} \end{aligned}$$

Bei einer 1:N-Beziehung ist die „Richtung" der Funktion zwingend. Die Beziehung *beschäftigen* ist z.B. eine partielle Funktion von *Personen* nach *Firmen*, also:

$$\text{beschäftigen} : \text{Personen} \rightarrow \text{Firmen}$$

Die Funktion geht also von dem „N"-Entitytyp zum „1"-Entitytyp. Wir werden später – im Zusammenhang mit der Umsetzung von ER-Schemata in relationale Schemata – nochmals auf diesen wichtigen Punkt zurückkommen. Analoges gilt natürlich wieder für N:1-Beziehungen, wobei wiederum die „Richtung" der Funktion zu beachten ist.

Die den 1:1- bzw. 1:N-Beziehungen zugeordneten Funktionen sind partiell, weil es Entities aus dem Definitionsbereich geben kann, die gar keine Beziehung eingehen. Für diese Entities ist die Funktion somit nicht definiert.

In Abbildung 2.4 sind die oben verbal beschriebenen Funktionalitäten grafisch veranschaulicht. Die Ovale repräsentieren die Entitytypen: das linke den Entitytyp

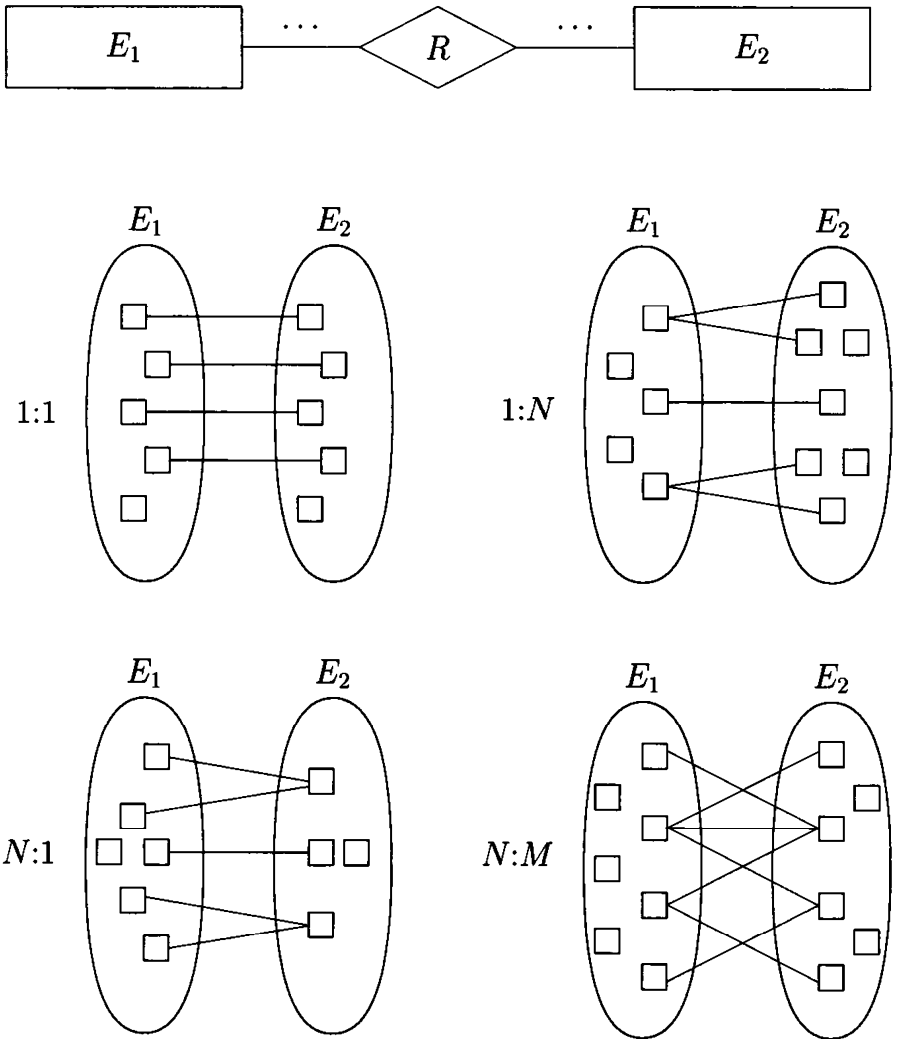

Abb. 2.4: Grafische Veranschaulichung der Funktionalitäten einer binären Beziehung R zwischen E_1 und E_2.

E_1 und das rechte den Entitytyp E_2. Die kleinen Quadrate innerhalb der Ovale stellen die Entities des jeweiligen Typs dar und die die Entities verbindenden Linien repräsentieren jeweils eine Instanz der Beziehung R.

In Abbildung 2.7 (auf Seite 44) sind die Funktionalitäten des konzeptuellen Universitätsschemas eingezeichnet.

2.7.2 Funktionalitätsangaben bei n-stelligen Beziehungen

Die Angabe von Funktionalitäten wurde bisher nur für binäre Beziehungstypen definiert, sie kann aber auf n-stellige Beziehungen erweitert werden. Sei also R eine Beziehung zwischen den Entitymengen $E_1, \ldots, E_n$, wobei die Funktionalität bei der Entitymenge E_k $(1 \leq k \leq n)$ mit einer „1" spezifiziert sein soll, bei den anderen Mengen ebenfalls mit „1" oder mit einem in dem Beziehungstyp eindeutigen Buchstaben, der wie vorher „viele" repräsentiert. Dann muss gelten, dass durch R die

folgende partielle Funktion vorgegeben wird:

$$R : E_1 \times \ldots \times E_{k-1} \times E_{k+1} \times \ldots \times E_n \rightarrow E_k$$

Solche funktionalen Beziehungen müssen dann natürlich für alle Entitymengen von R gelten, die bei der Funktionalitätsangabe ebenfalls mit einer „1" gekennzeichnet sind. Die Leser mögen sich hier vergegenwärtigen, dass die in Abschnitt 2.7.1 vorgestellten Funktionalitäten für binäre Beziehungen Spezialfälle der obigen Definition sind.

Als anschauliches Beispiel ist in Abbildung 2.5 die dreistellige Beziehung *betreuen* zwischen den Entitytypen *Studenten*, *Professoren* und *Seminarthemen* mit Kardinalitätsangabe N:1:1 grafisch dargestellt. Gemäß obiger Definition kann man die Beziehung *betreuen* demnach als partielle Funktionen wie folgt auffassen:

$$\begin{aligned} \text{betreuen} : \text{Professoren} \times \text{Studenten} &\rightarrow \text{Seminarthemen} \\ \text{betreuen} : \text{Seminarthemen} \times \text{Studenten} &\rightarrow \text{Professoren} \end{aligned}$$

Die Kardinalitätsangaben dieses Beispielschemas legen folgende Konsistenzbedingungen fest, die im Wesentlichen die Studienordnung unserer Universität darstellen:

1. Studenten dürfen bei demselben Professor bzw. derselben Professorin nur ein Seminarthema „ableisten" (damit ein breites Spektrum abgedeckt wird).

2. Studenten dürfen dasselbe Seminarthema nur einmal bearbeiten – sie dürfen also nicht bei anderen Professoren ein schon einmal erteiltes Seminarthema nochmals bearbeiten.

Es sind aber folgende Datenbankzustände nach wie vor möglich:

- Professoren können dasselbe Seminarthema „wiederverwenden" – also dasselbe Thema auch mehreren Studenten erteilen.

- Ein Thema kann von mehreren Professoren vergeben werden – aber an unterschiedliche Studenten.

In Abbildung 2.6 sind vier legale Ausprägungen der Beziehung *betreuen* – mit b_1, b_2, b_3, b_4 markiert – und zwei illegale Ausprägungen b_5 und b_6 dargestellt. Die Ausprägung b_5 ist illegal, weil Student/in s_1 bei Professor/in p_1 zwei Seminare belegen will. Die Beziehungsausprägung b_6 ist illegal, weil Student/in s_3 – gemäß dem Prinzip des geringsten Aufwands – versucht, dasselbe Thema t_1 zweimal bei unterschiedlichen Professoren p_3 und p_4 zu bearbeiten.

Wir wollen die Angabe von Funktionalitäten jetzt noch für die ternäre Beziehung *prüfen* unseres konzeptuellen Universitätsschemas aus Abbildung 2.7 diskutieren. Studenten sollten daran gehindert werden, dieselben Vorlesungen von unterschiedlichen Professoren prüfen zu lassen. Mit anderen Worten, zu einem Paar aus *Studenten* und *Vorlesungen* soll es maximal einen Professor bzw. eine Professorin als Prüfer geben. Die Beziehung *prüfen* sollte also den Eigenschaften einer partiellen Funktion der folgenden Form genügen:

$$\text{prüfen} : \text{Studenten} \times \text{Vorlesungen} \rightarrow \text{Professoren}$$

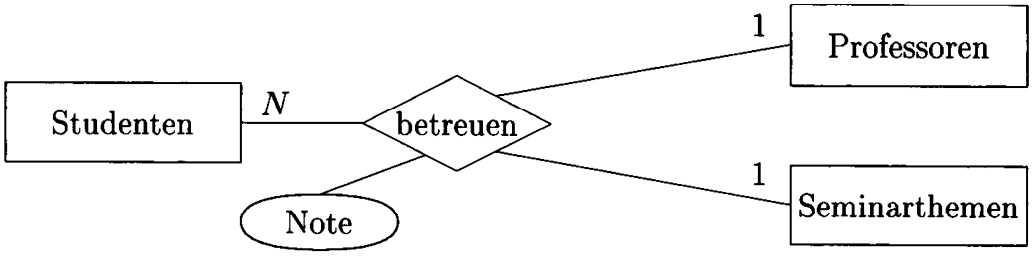

Abb. 2.5: Die Betreuung von Seminarthemen als Entity-Relationship-Diagramm

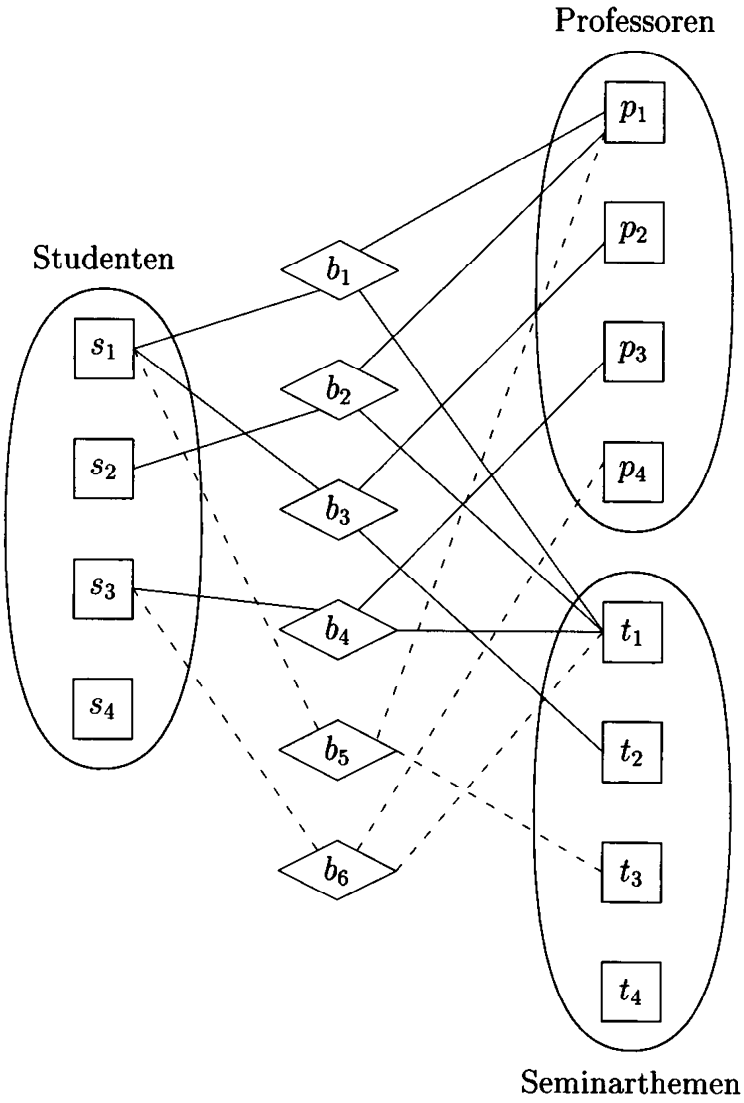

Abb. 2.6: Ausprägungen der Beziehung *betreuen*: gestrichelte Linien markieren illegale Ausprägungen

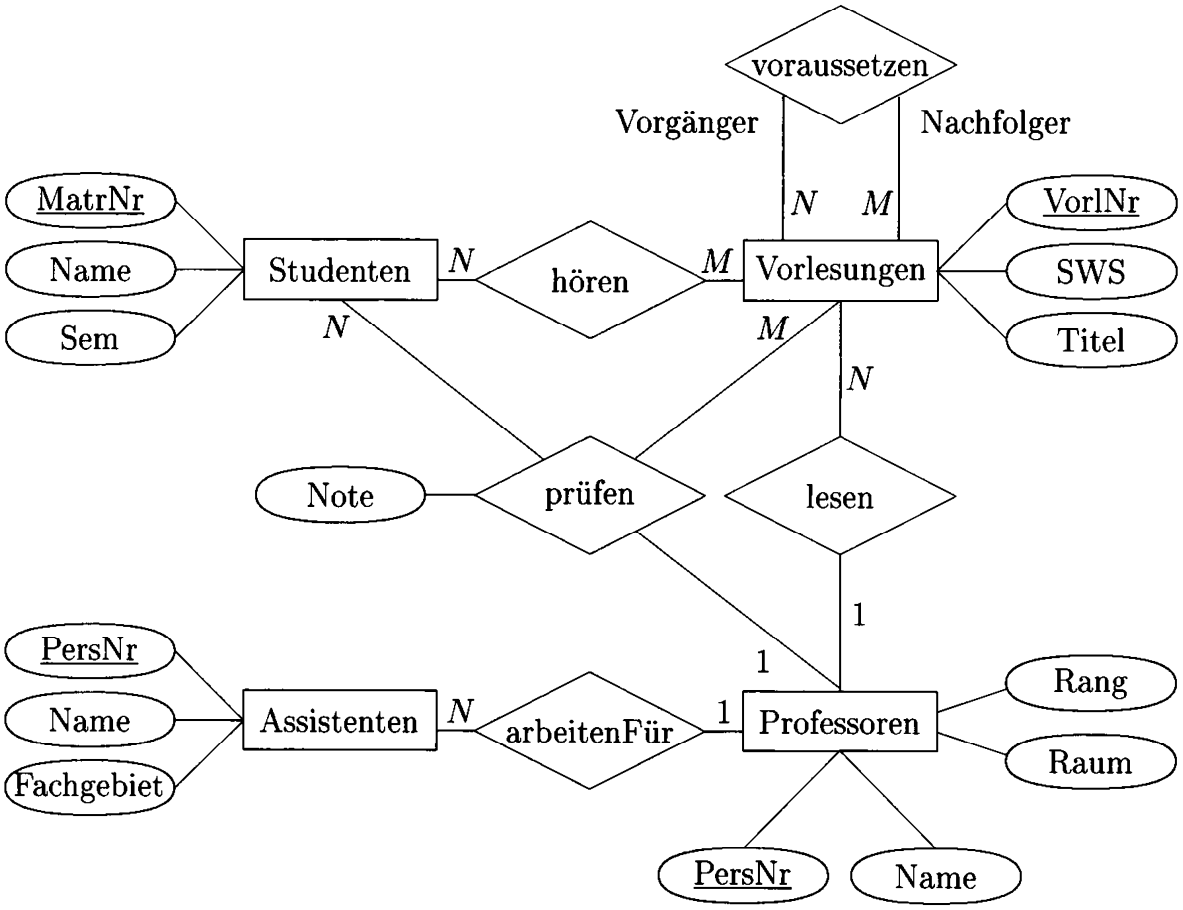

Abb. 2.7: Markierung der Funktionalitäten im Universitätsschema

Dies wird durch die N:M:1-Funktionalitätsangabe im ER-Schema erzwungen. Weitere Einschränkungen dieser Art gibt es nicht, da wir annehmen, dass Studenten mehrere Vorlesungen von demselben Professor bzw. derselben Professorin abprüfen lassen können und Professoren selbstverständlich mehrere Studenten über dieselbe Vorlesung prüfen.

2.7.3 Die (min, max)-Notation

Im vorangegangenen Abschnitt haben wir die Funktionalitäten von Beziehungstypen beschrieben. Hier wollen wir einen Formalismus einführen, mit dem Beziehungen oft präziser charakterisiert werden können. Bei der Angabe der Funktionalität ist für ein Entity nur die maximale Anzahl von Beziehungsinstanzen relevant. Wann immer diese Zahl größer als eins ist, wird sie, ohne genauere Aussagen zu machen, als N oder M (also „viele“) angegeben. Bei der (min, max)-Notation werden nicht nur die Extremwerte – nämlich *eins* oder *viele* – angegeben sondern, wann immer möglich, präzise Unter- und Obergrenzen festgelegt. Es wird also auch die minimale Anzahl angegeben, weil dies für viele Beziehungstypen eine sinnvolle und manchmal auch essentielle Integritätsbedingung darstellt.

Bei der (min, max)-Notation wird für jeden an einem Beziehungstyp beteiligten Entitytyp ein Paar von Zahlen, nämlich min und max, angegeben. Dieses Zahlenpaar sagt aus, dass jedes Entity dieses Typs mindestens min-mal in der Beziehung steht

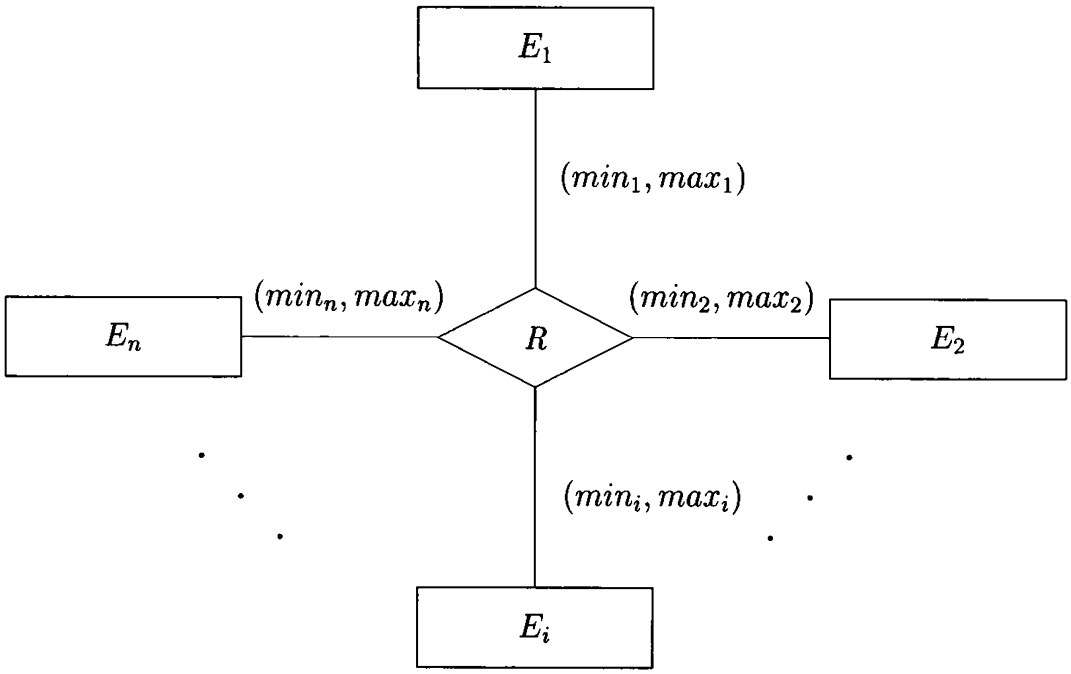

Abb. 2.8: Abstrakte n-stellige Beziehung R mit (min, max)-Angabe

und höchstens max-mal.

Um das etwas formaler auszudrücken, schauen wir uns die Grafik in Abbildung 2.8 an. In dieser Abbildung ist eine abstrakte n-stellige Beziehung R gezeigt. Somit stellt R eine Relation zwischen den Entitymengen $E_1, E_2, \ldots, E_n$ dar, wobei also gilt:

$$R \subseteq E_1 \times \cdots \times E_i \times \cdots \times E_n$$

Die Markierung (min_i, max_i) gibt an, dass es für alle $e_i \in E_i$ mindestens min_i Tupel der Art $(\ldots, e_i, \ldots)$ und höchstens max_i viele solcher Tupel in R gibt. Wiederum ist gefordert, dass diese Angaben Gesetzmäßigkeiten der modellierten realen Welt darstellen und nicht einfach nur den derzeitigen zufälligen Zustand der Datenbasis beschreiben.

Sonderfälle werden wie folgt gehandhabt:

- Wenn es Entities geben darf, die gar nicht an der Beziehung „teilnehmen“, so wird min mit 0 angegeben.
- Wenn ein Entity beliebig oft an der Beziehung „teilnehmen darf“, so wird die max-Angabe durch $*$ ersetzt.

Die Angabe $(0, *)$ ist somit die allgemeinste, da sie aussagt, dass betreffende Entities gar nicht oder beliebig oft in der Beziehung vorkommen können.

Wir wollen uns nun noch ein illustratives Beispiel für die (min, max)-Notation anschauen. Dazu verlassen wir (kurzzeitig) unsere Universitätswelt und schauen uns die Begrenzungsflächenmodellierung von Polyedern an. Ein Polyeder wird dabei durch die Hülle seiner begrenzenden Flächen beschrieben, diese wiederum werden durch ihre Begrenzung, bestehend aus Kanten, modelliert. Eine Kante hat einen Start und ein Ende in der Form eines Punktes im dreidimensionalen Raum. Diese Modellierung

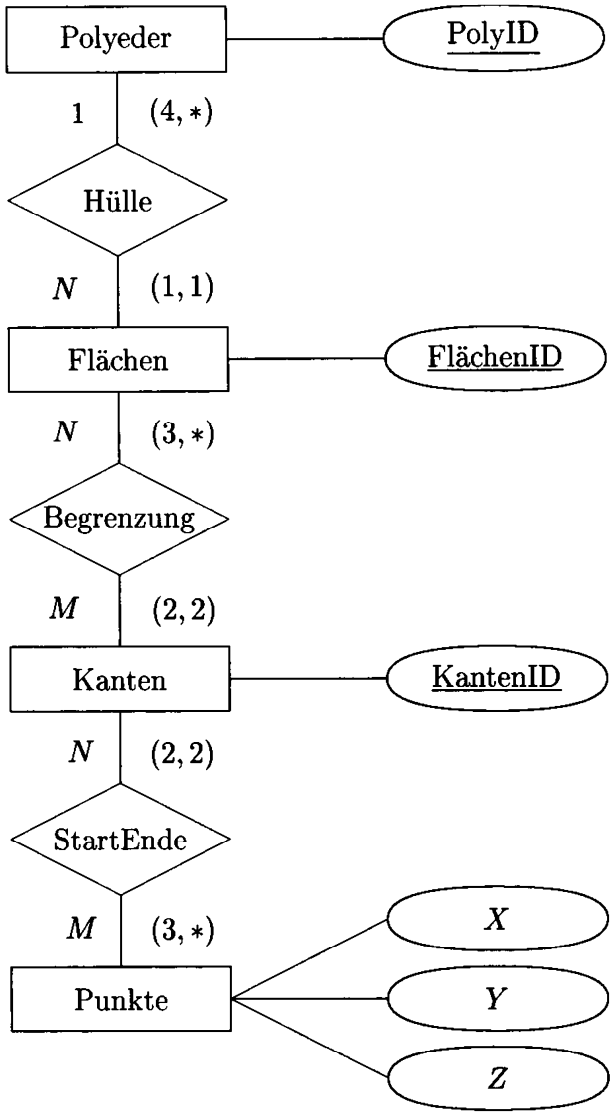

Abb. 2.9: Konzeptuelles Schema der Begrenzungsflächendarstellung von Polyedern

von Polyedern ist in Abbildung 2.9 als ER-Schema mit der vergleichenden Angabe von Funktionalitäten und der (*min*, *max*)-Notation gezeigt.

Schauen wir uns zuerst die Funktionalitäten an: Ein Polyeder wird von mehreren Flächen umhüllt, wobei jede Fläche nur zu einem Polyeder gehören soll. Also ist *Hülle* eine 1:N-Beziehung. Eine Fläche wird von mehreren Kanten begrenzt und jede Kante gehört zu mehreren Flächen. Folglich ist *Begrenzung* eine allgemeine N:M-Beziehung. Auch *StartEnde* ist eine N:M-Beziehung, da jeder Kante zwei (d.h. mehrere) Punkte zugeordnet sind und ein Punkt mehreren (sogar beliebig vielen) Kanten eines Polyeders zugeordnet sein kann.

Diese sehr grobe Charakterisierung der Beziehungen kann durch Verwendung der (*min*, *max*)-Notation viel präziser erfolgen. Für die Angabe der in Abbildung 2.9 gezeigten *min*-Werte sollten wir uns den Polyeder mit der minimalen Flächen-, Kanten- und Punkteanzahl vergegenwärtigen: Eine grafische Skizze des Tetraeders ist in Abbildung 2.10 gezeigt.

In dieser Abbildung ist erkennbar, dass ein Polyeder minimal vier umhüllende Flächen besitzt – die maximale Anzahl ist beliebig und wird somit durch * ange-

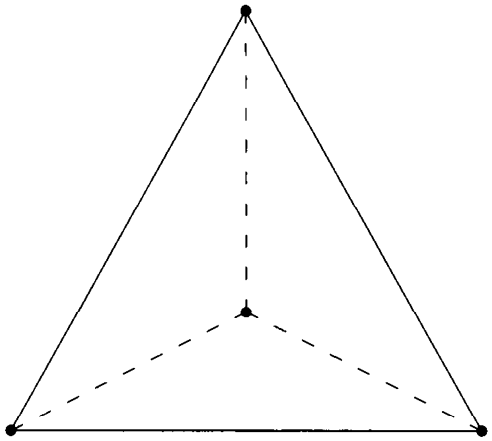

Abb. 2.10: Grafische Darstellung eines Tetraeders

geben. Eine Fläche wird – im Falle eines Dreiecks – durch ein Minimum von drei Kanten begrenzt; wiederum ist die maximale Anzahl von begrenzenden Kanten beliebig. Jede Kante begrenzt bei einem Polyeder genau zwei Flächen. Eine Kante wird durch genau zwei Punkte beschrieben, die über die Beziehung *StartEnde* der Kante zugeordnet werden. Bei einem Polyeder gehört jeder Begrenzungspunkt – und nur solche sind in der Entitymenge *Punkte* enthalten – zu mindestens drei Kanten (siehe Tetraeder) und maximal zu beliebig vielen.

Wir sollten noch darauf hinweisen, dass der Vergleich der 1:N-Angabe mit der (min, max)-Notation in gewisser Weise „kontra-intuitiv" ist. Dazu betrachte man die 1:N-Beziehung *Hülle* aus Abbildung 2.9. Hierbei ist der Entitytyp *Flächen* mit N markiert; andererseits „wandert" bei der (min, max)-Notation die Angabe „viele", also das Sternchen $*$, zu dem Entitytyp *Polyeder*. Dies liegt in der Definition der (min, max)-Notation begründet – siehe dazu auch die Übungsaufgabe 2.1.

In Abbildung 2.14 (auf Seite 51) sind die (min, max)-Angaben für unser Universitätsschema angegeben. Alle Vorlesungen werden von mindestens drei Studenten gehört – andernfalls finden sie nicht statt. Assistenten sind genau einem Professor bzw. einer Professorin als Mitarbeiter/in zugeordnet. Weiterhin werden Vorlesungen von genau einem Professor bzw. einer Professorin gelesen. Professoren können zeitweilig – aufgrund von Forschungssemestern oder anderer Verpflichtungen in der Universitätsleitung – vom Vorlesungsbetrieb freigestellt werden; daraus resultiert die Markierung $(0, *)$.

Man beachte, dass die Ausdruckskraft der Funktionalitätsangaben und der (min, max)-Angaben bei n-stelligen Beziehungen mit $n > 2$ unvergleichbar ist: Es gibt Konsistenzbedingungen, die mit Funktionalitätsangaben, aber nicht mit (min,max)-Angaben ausdrückbar sind und wiederum andere Konsistenzbedingungen, die mit der (min,max)-Angabe formulierbar sind, aber nicht durch Funktionalitätseinschränkungen. Siehe dazu auch Übungsaufgabe 2.2.

Grundsätzlich gilt natürlich, dass es viele anwendungsspezifische Konsistenzbedingungen gibt, die mit den Mitteln des Entity-Relationship-Modells nicht formulierbar sind. Diese müssen im Zuge der Anforderungsanalyse und des konzeptuellen Entwurfs anderweitig (z.B. in natürlicher Sprache) festgehalten werden, damit sie beim Implementationsentwurf und bei der Realisierung der Anwendungsprogramme