Kapitel 2

Konzeptuelle Modellierung

2.1 Das Entity-Relationship-Modell

Die grundlegenden Modellierungsstrukturen dieses Modells sind die *Entities* (Gegenstände) und die *Relationships* (Beziehungen) zwischen den Entities. Des weiteren gibt es noch *Attribute* und *Rollen*. Die Ausprägungen eines Entity-Typs sind seine Entities, die Ausprägung eines Relationship-Typs sind seine Relationships. Nicht immer ist es erforderlich, diese Unterscheidung aufrecht zu halten.

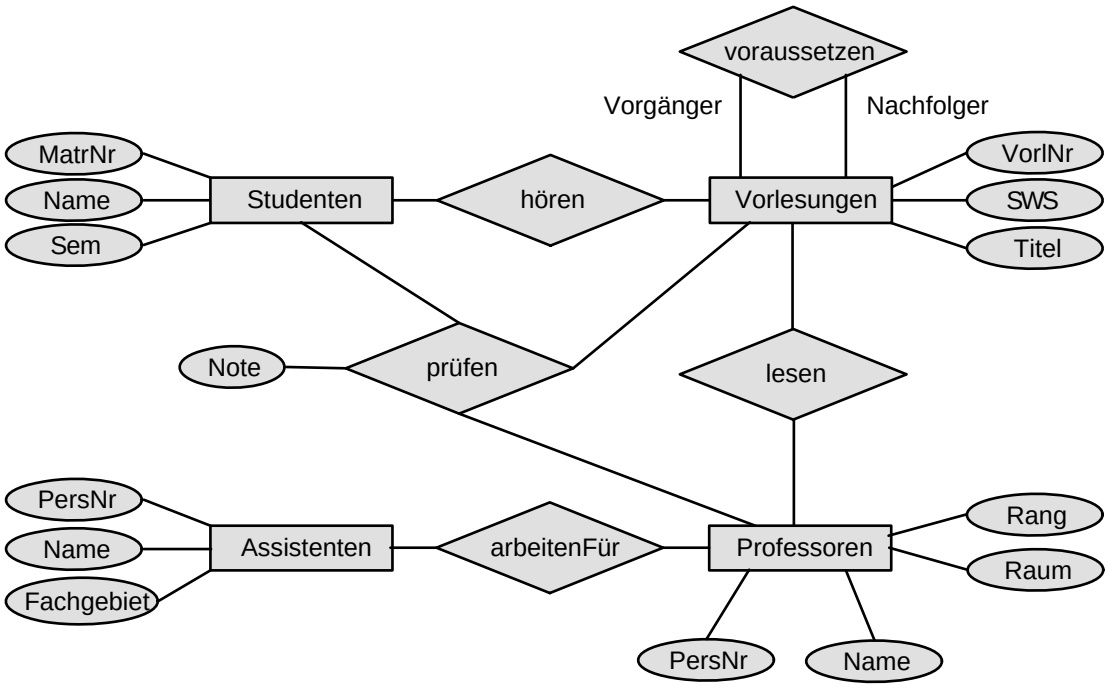


Abbildung 2.1: ER-Diagramm für Universität

Entities sind physisch oder gedanklich existierende Konzepte der zu modellierenden Welt, dargestellt

im ER-Diagramm durch Rechtecke. Attribute charakterisieren die Entities und werden durch Ovale beschrieben. Beziehungen zwischen den Entities können binär oder auch mehrwertig sein, sie werden durch Routen symbolisiert.

In Abbildung 2.1 gibt es einen dreiwertigen Beziehungstyp *prüfen*, der auch über ein Attribut *Note* verfügt. Binäre Beziehungstypen, wie z.B. *voraussetzen*, an denen nur ein Entity-Typ beteiligt ist, werden *rekursive Beziehungstypen* genannt. Durch die Angabe von *Vorgänger* und *Nachfolger* wird die Rolle eines Entity-Typen in einer rekursiven Beziehung gekennzeichnet.

2.2 Schlüssel

Eine minimale Menge von Attributen, welche das zugeordnete Entity eindeutig innerhalb aller Entities seines Typs identifiziert, nennt man *Schlüssel* oder auch *Schlüsselkandidaten*. Gibt es mehrere solcher Schlüsselkandidaten, wird einer als *Primärschlüssel* ausgewählt. Oft gibt es künstlich eingeführte Attribute, wie z.B. Personalnummer *(PersNr)*, die als Primärschlüssel dienen. Schlüsselattribute werden durch Unterstreichung gekennzeichnet. Achtung: Die Schlüsseleigenschaft bezieht sich auf Attribut--Kombinationen, nicht nur auf die momentan vorhandenen Attributwerte!

- **Beispiel:**
  Im Entity-Typ `Person` mit den Attributen `Name`, `Vorname`, `PersNr`, `Geburtsdatum`, `Wohnort` ist `PersNr` der Primärschlüssel. Die Kombination `Name`, `Vorname`, `Geburtsdatum` bildet ebenfalls einen (Sekundär-)Schlüssel, sofern garantiert wird, daß es nicht zwei Personen mit demselben Namen und demselben Geburtsdatum gibt.

2.3 Charakterisierung von Beziehungstypen

Ein Beziehungstyp *R* zwischen den Entity-Typen $E_1,\ E_2,\ ...,\ E_n$ kann als Relation im mathematischen Sinn aufgefaßt werden. Also gilt:

$$R \subset E_1 \times E_2 \times ... \times E_n$$

In diesem Fall bezeichnet man n als den Grad der Beziehung R. Ein Element $(e_1,\ e_2,\ ...,\ e_n) \in\ R$ nennt man eine Instanz des Beziehungstyps.

Man kann Beziehungstypen hinsichtlich ihrer *Funktionalität* charakterisieren (Abbildung 2.2). Ein binärer Beziehungstyp R zwischen den Entity-Typen E_1 und E_2 heißt

- *1:1-Beziehung (one-one)*, 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 höchstens ein Entity e_1 aus E_1 zugeordnet ist.
  Beispiel: *verheiratet_mit*.

- *1:N-Beziehung (one-many)*, falls jedem Entity e_1 aus E_1 beliebig viele (also keine oder mehrere) Entities aus E_2 zugeordnet sind, aber jedem Entity e_2 aus E_2 höchstens ein Entity e_1 aus E_1 zugeordnet ist.
  Beispiel: *beschäftigen*.

- *N:1-Beziehung (many-one)*, falls analoges zu obigem gilt.
  Beispiel: *beschäftigt_bei*

- *N:M-Beziehung (many-many)*, wenn keinerlei Restriktionen gelten, d.h. jedes Entity aus E_1 kann mit beliebig vielen Entities aus E_2 in Beziehung stehen und umgekehrt kann jedes Entity e_2 aus E_2 mit beliebig vielen Entities aus E_1 in Beziehung stehen.
  Beispiel: *befreundet_mit*.

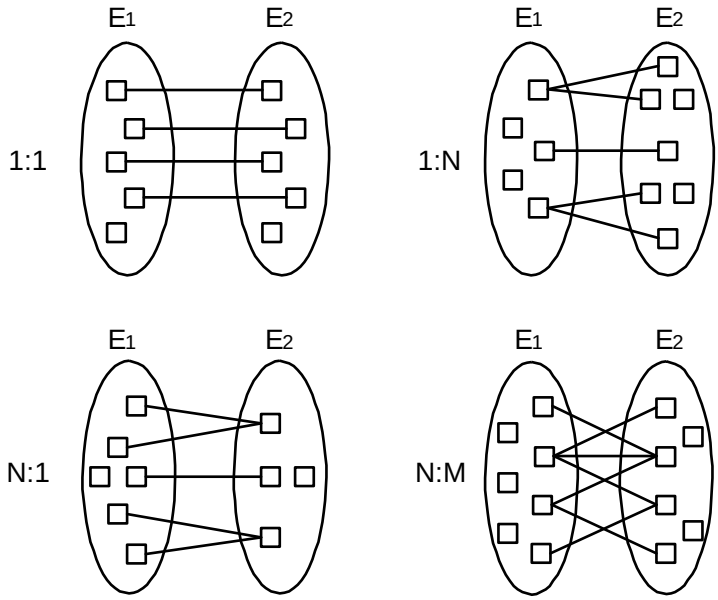


Abbildung 2.2: Mögliche Funktionalitäten von binären Beziehungen

Die binären 1:1-, 1:N- und N:1-Beziehungen kann man auch als *partielle Funktionen* ansehen, welche einige Elemente aus dem Definitionsbereich auf einige Elemente des Wertebereichs abbilden, z. B.

$$\text{beschäftigt\_bei} : \text{Personen} \rightarrow \text{Firmen}$$

2.4 Die (*min, max*)-Notation

Bei der *(min, max)*-Notation wird für jedes an einem Beziehungstyp beteiligte Entity 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 und höchstens *max*-mal. 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. Somit ist (0,*) die allgemeinste Aussage. Abbildung 2.3 zeigt die Verwendung der *(min, max)*-Notation anhand der Begrenzungsflächenmodellierung von Polyedern.

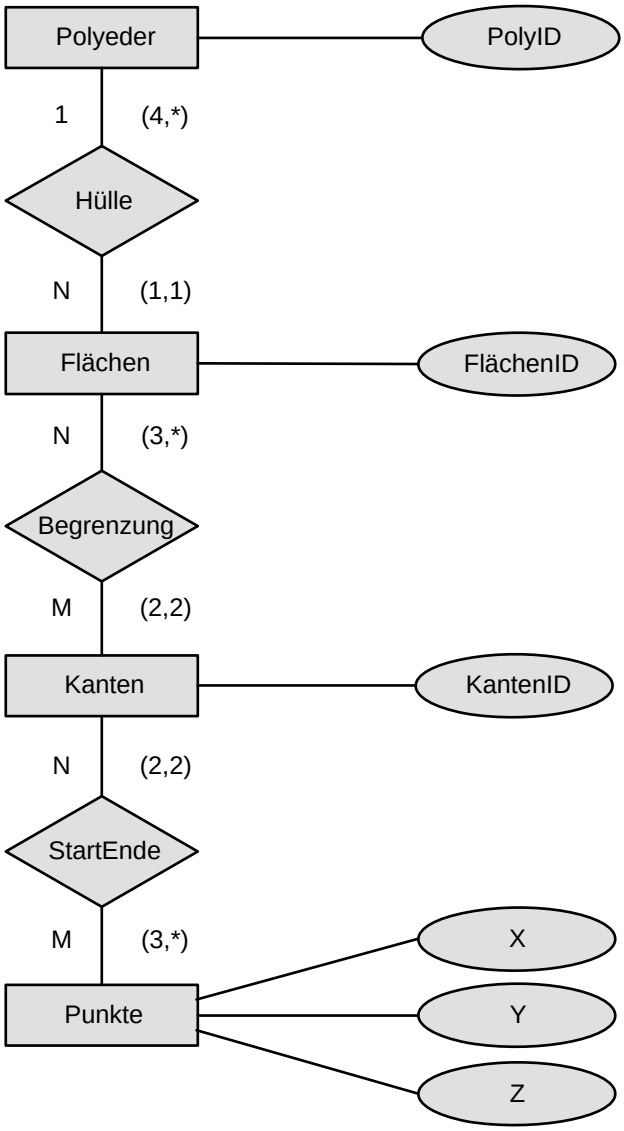


Abbildung 2.3: ER-Diagramm für Begrenzungsflächendarstellung von Polyedern

2.5 Existenzabhängige Entity-Typen

Sogenannte *schwache* Entities können nicht autonom existieren, sondern

- sind in ihrer Existenz von einem anderen, übergeordneten Entity abhängig
- und sind nur in Kombination mit dem Schlüssel des übergeordneten Entity eindeutig identifizierbar.

Abbildung 2.4 verdeutlicht dieses Konzept anhand von Gebäuden und Räumen. Räume können ohne Gebäude nicht existieren. Die Raumnummern sind nur innerhalb eines Gebäudes eindeutig. Da-

her wird das entsprechende Attribut gestrichelt markiert. Schwache Entities werden durch doppelt gerahmte Rechtecke repräsentiert und ihre Beziehung zum übergeordneten Entity-Typ durch eine Verdoppelung der Raute und der von dieser Raute zum schwachen Entity-Typ ausgehenden Kante markiert.

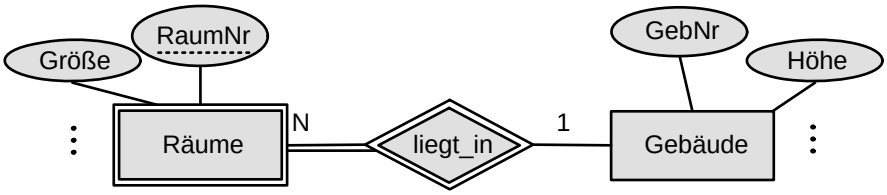


Abbildung 2.4: Ein existenzabhängiger (schwacher) Entity-Typ

2.6 Generalisierung

Zur weiteren Strukturierung der Entity-Typen wird die *Generalisierung* eingesetzt. Hierbei werden Eigenschaften von ähnlichen Entity-Typen einem gemeinsamen *Obertyp* zugeordnet. Bei dem jeweiligen *Untertyp* verbleiben nur die nicht faktorisierbaren Attribute. Somit stellt der Untertyp eine *Spezialisierung* des Obertyps dar. Diese Tatsache wird durch eine Beziehung mit dem Namen **is-a** (ist ein) ausgedrückt, welche durch ein Sechseck, verbunden mit gerichteten Pfeilen symbolisiert wird.

In Abbildung 2.5 sind *Assistenten* und *Professoren* jeweils Spezialisierungen von *Angestellte* und stehen daher zu diesem Entity-Typ in einer *is-a* Beziehung.

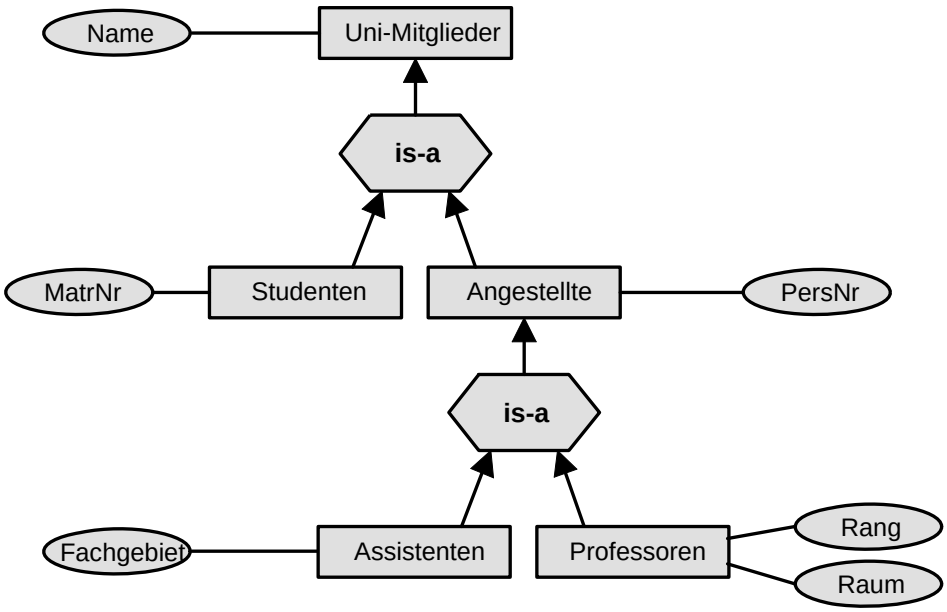


Abbildung 2.5: Spezialisierung der Universitätsmitglieder

Bezüglich der Teilmengensicht ist von Interesse:

- die *disjunkte* Spezialisierung: die Entitymengen der Untertypen sind paarweise disjunkt

- die *vollständige* Spezialisierung: die Obermenge enthält keine direkten Elemente, sondern setzt sich komplett aus der Vereinigung der Entitymengen der Untertypen zusammen.

In Abbildung 2.5 ist die Spezialisierung von *Uni-Mitglieder* vollständig und disjunkt, die Spezialisierung von *Angestellte* ist disjunkt, aber nicht vollständig, da es noch andere, nichtwissenschaftliche Angestellte (z.B. Sekretärinnen) gibt.

2.7 Aggregation

Durch die *Aggregation* werden einem übergeordneten Entity-Typ mehrere untergeordnete Entity-Typen zugeordnet. Diese Beziehung wird als *part-of* (Teil von) bezeichnet, um zu betonen, daß die untergeordneten Entities Bestandteile der übergeordneten Entities sind. Um eine Verwechselung mit dem Konzept der Generalisierung zu vermeiden, verwendet man nicht die Begriffe *Obertyp* und *Untertyp*.

zeigt die Aggregationshierarchie eines Fahrrads. Zum Beispiel sind *Rohre* und *Lenker* Bestandteile des *Rahmen*; *Felgen* und *Speichen* sind Bestandteile der *Räder*.

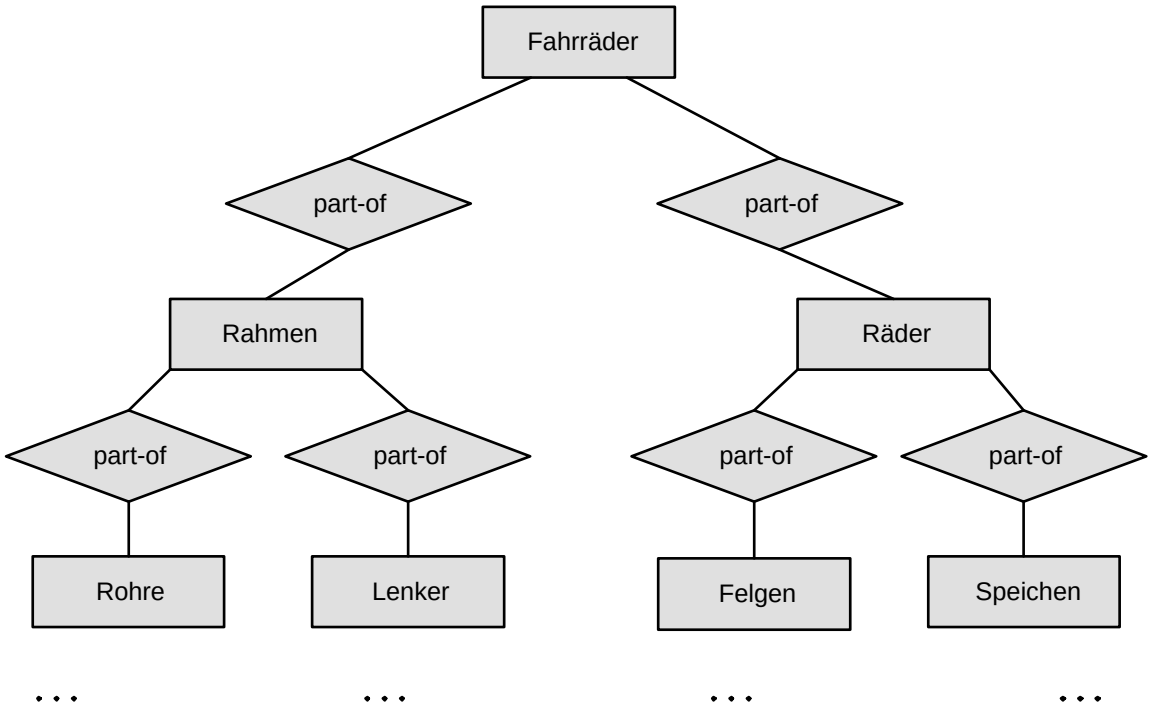


Abbildung 2.6: Aggregationshierarchie eines Fahrrads

2.8 Konsolidierung

Bei der Modellierung eines komplexeren Sachverhaltes bietet es sich an, den konzeptuellen Entwurf zunächst in verschiedene Anwendersichten aufzuteilen. Nachdem die einzelnen Sichten modelliert sind, müssen sie zu einem globalen Schema i zusammengefaßt werden (Abbildung 2.7).

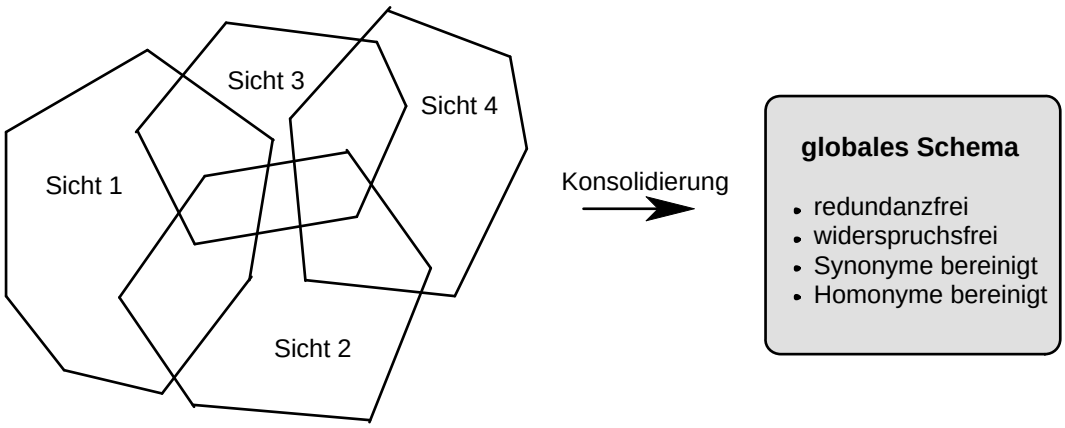


Abbildung 2.7: Konsolidierung überlappender Sichten

Probleme entstehen dadurch, daß sich die Datenbestände der verschiedenen Anwender teilweise überlappen. Daher reicht es nicht, die einzelnen konzeptuellen Schemata zu vereinen, sondern sie müssen *konsolidiert* werden.

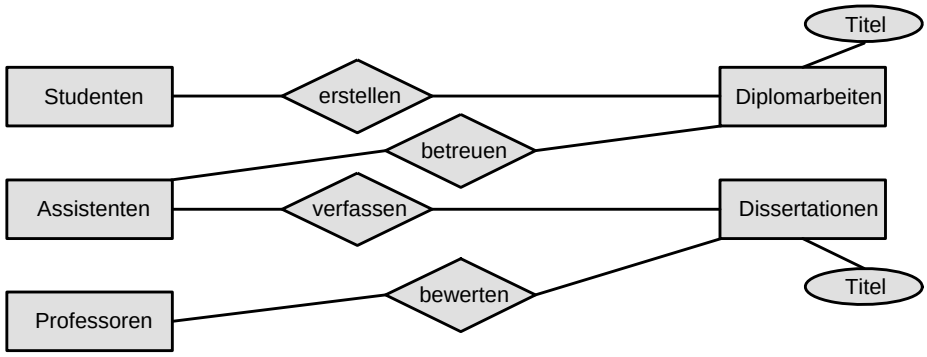
Darunter versteht man das Entfernen von Redundanzen und Widersprüchen. Widersprüche entstehen durch *Synonyme* (gleiche Sachverhalte wurden unterschiedlich benannt) und durch *Homonyme* (unterschiedliche Sachverhalte wurden gleich benannt) sowie durch unterschiedliches Modellieren desselben Sachverhalts zum einen über Beziehungen, zum anderen über Attribute. Bei der Zusammenfassung von ähnlichen Entity-Typen zu einem Obertyp bietet sich die Generalisierung an.

Abbildung 2.8 zeigt drei Sichten einer Universitätsdatenbank. Für eine Konsolidierung sind folgende Beobachtungen wichtig:

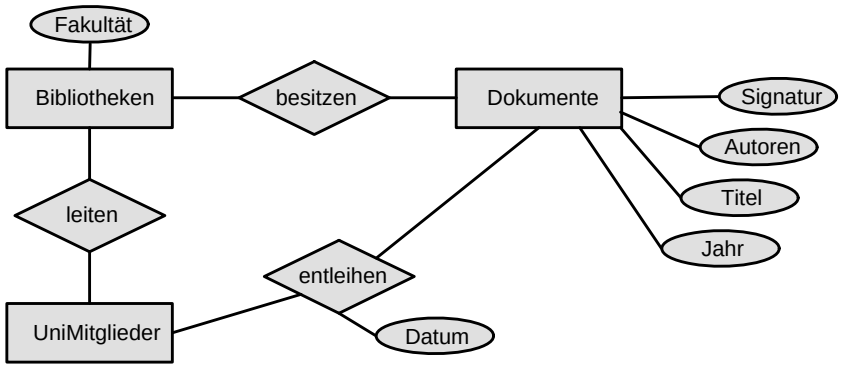
- *Professoren* und *Dozenten* werden synonym verwendet.
- *UniMitglieder* ist eine Generalisierung von *Studenten*, *Professoren* und *Assistenten*.
- *Bibliotheken* werden nicht von beliebigen *UniMitglieder* geleitet, sondern nur von *Angestellte*.
- *Dissertationen*, *Diplomarbeiten* und *Bücher* sind Spezialisierungen von *Dokumente*.
- Die Beziehungen *erstellen* und *verfassen* modellieren denselben Sachverhalt wie das Attribut *Autor*.

Abbildung 2.9 zeigt das Ergebnis der Konsolidierung. Generalisierungen sind zur Vereinfachung als fettgedruckte Pfeile dargestellt. Das ehemalige Attribut *Autor* ist nun als Beziehung zwischen Dokumenten und Personen modelliert. Zu diesem Zweck ist ein neuer Entity-Typ *Personen* erforderlich, der *UniMitglieder* generalisiert. Damit werden die ehemaligen Beziehungen *erstellen* und *verfassen*

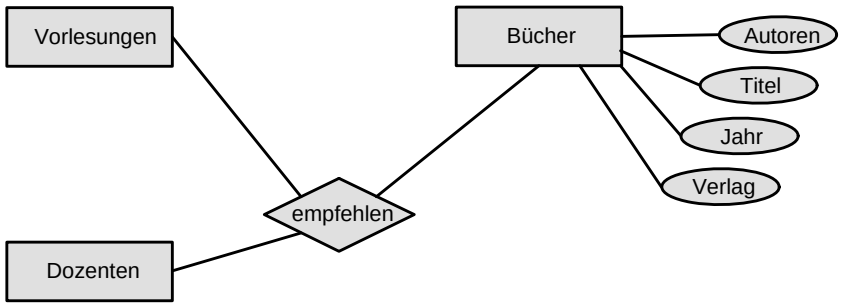
redundant. Allerdings geht im konsolidierten Schema verloren, daß Diplomarbeiten von *Studenten* und *Dissertationen* von *Assistenten* geschrieben werden.



Sicht 1: Erstellung von Dokumenten als Prüfungsleistung



Sicht 2: Bibliotheksverwaltung



Sicht 3: Buchempfehlungen für Vorlesungen

Abbildung 2.8: Drei Sichten einer Universitätsdatenbank

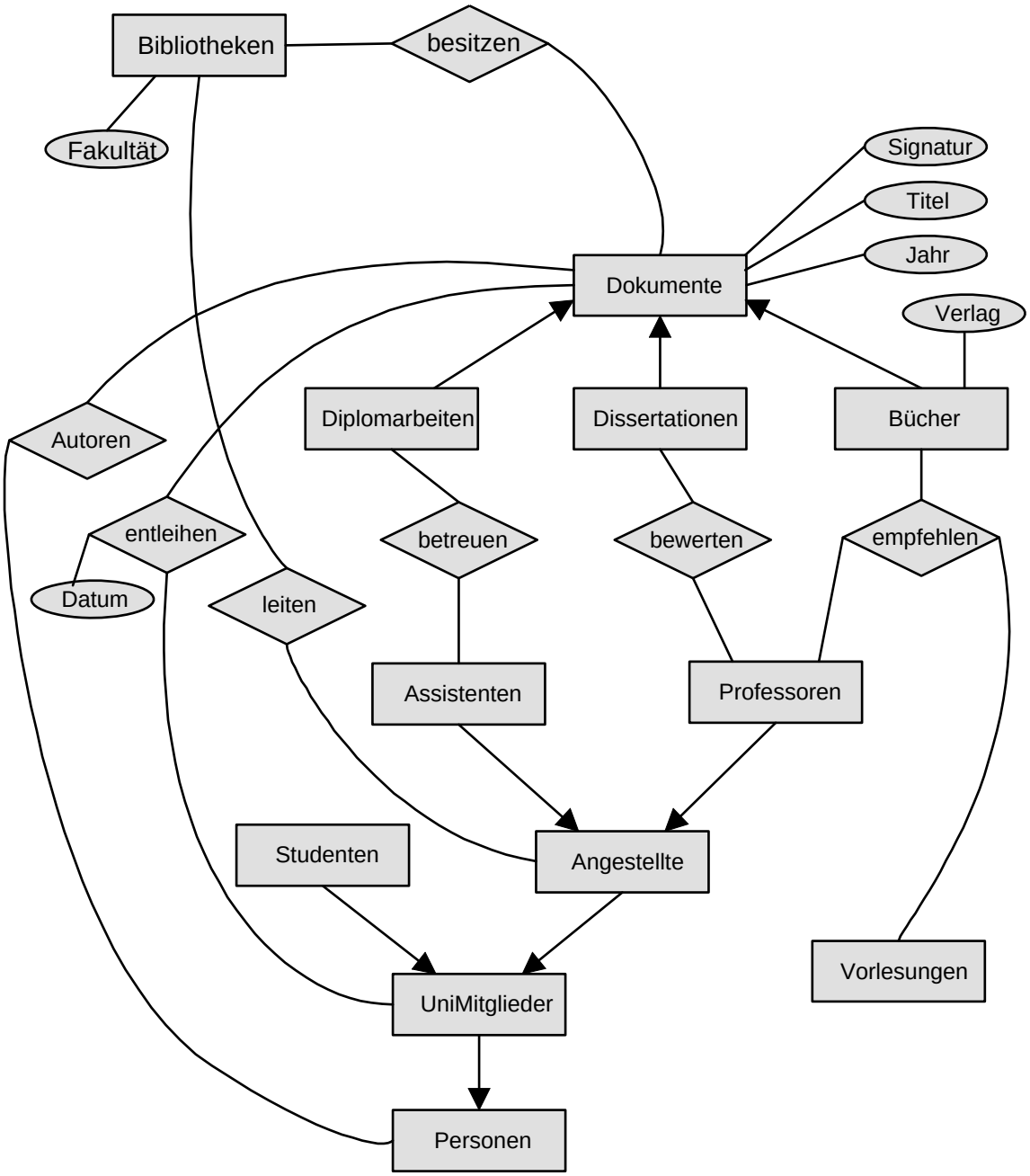


Abbildung 2.9: Konsolidiertes Schema der Universitätsdatenbank

2.9 UML

Im Bereich Software Engineering hat sich die objektorientierte Modellierung mit Hilfe von *UML (Unified Modelling Language)* durchgesetzt. Dieser Formalismus lässt sich auch für den Datenbankentwurf benutzen.

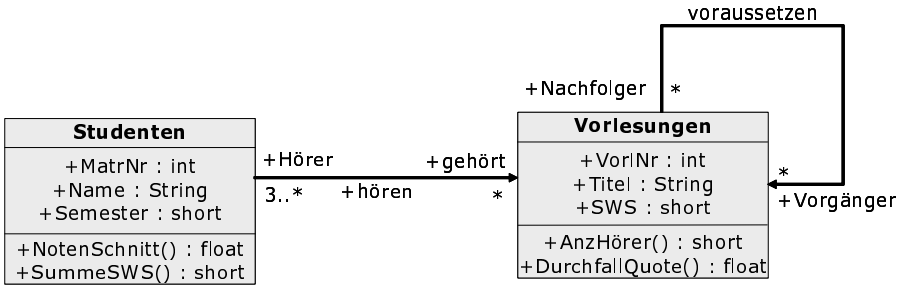


Abbildung 2.10: UML: Klassen und Assoziationen

Abbildung 2.10 zeigt die beiden Klassen `Studenten` und `Vorlesungen`, die neben den Datenfeldern auch Methoden aufweisen, welche das Verhalten beschreiben. Öffentlich sichtbare Komponenten werden mit + gekennzeichnet; Methoden sind an den beiden Klammern () zu erkennen und Datentypen werden nach dem Doppelpunkt : genannt. Beziehungen zwischen den Klassen werden durch Assoziationen ausgedrückt, welche Aufgrund der Implementierung durch Referenzen über eine Richtung verfügen: So lassen sich effizient zu einem Studenten die Vorlesungen ermitteln, umgekehrt ist das nicht (so leicht) möglich. Am Ende der Pfeile sind jeweils die Rollen vermerkt: `Hörer`, `gehört`, `Nachfolger` und `Vorgänger`. Die sogenannte *Multiplizität* drückt die von der (min,max)-Notation bekannte Komplexität aus, allerdings jeweils äm anderen Endenotiert: mindestens drei Hörer sitzen in einer Vorlesung; ein Student kann beliebig viele Vorlesungen besuchen.

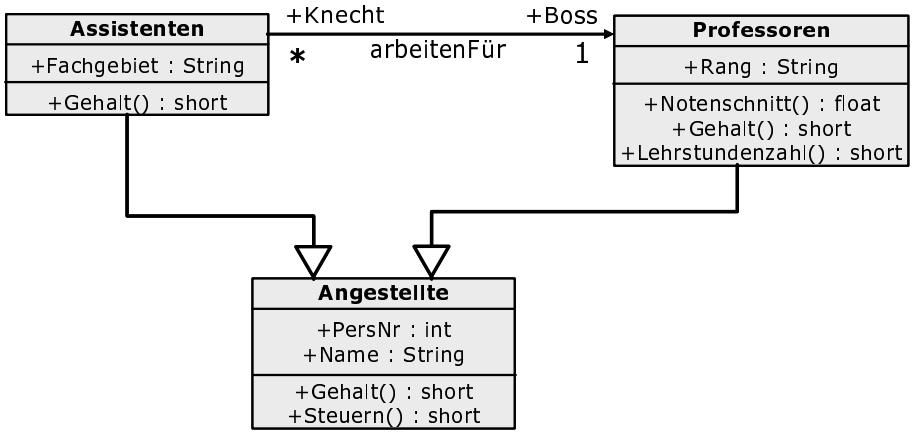


Abbildung 2.11: UML: Generalisierung

In der Abbildung 2.11 werden `Assistenten` und `Professoren` zu `Angestellten` verallgemeinert. Die gemeinsamen Datenfelder lauten `PersNr` und `Name`. Da sich die Berechnung des Gehaltes bei Assistenten und Professoren unterscheidet, erhalten sie beide ihre eigenen Methoden dafür;

sie verfeinern dadurch die in der Oberklasse vorhandene Methode `Gehalt()`. Die Steuern hingegen werden bei beiden nach demselben Algorithmus berechnet, daher reicht eine Methode `Steuern()` in der Oberklasse.

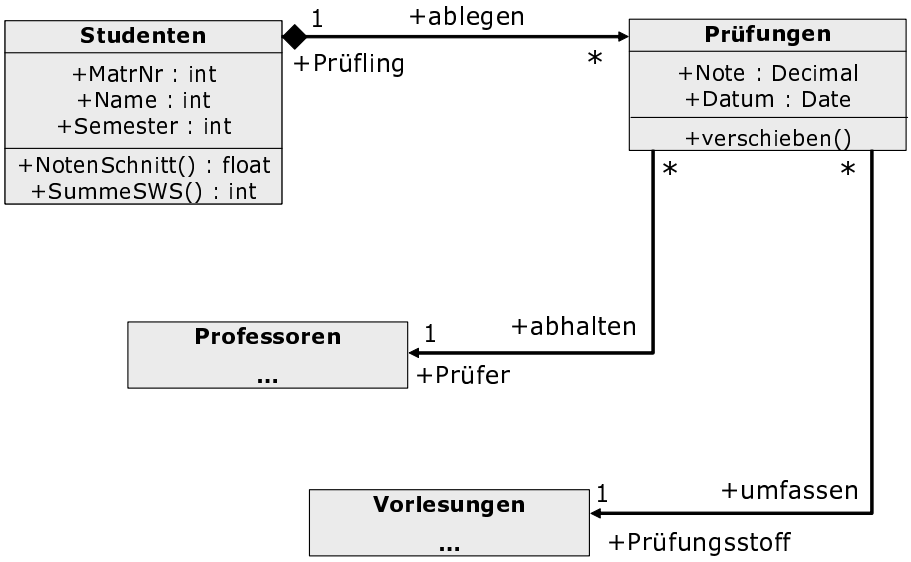


Abbildung 2.12: UML: Aggregation

Durch eine schwarze Raute wird in Abbildung 2.12 eine Aggregation ausgedrückt, welche die Modellierung eines schwachen Entity-Typen übernimmt. Die Beziehung zwischen `Studenten` und `Prüfungen` wird durch eine Assoziation mit dem Namen `ablegen` modelliert; der `Student` übernimmt dabei die Rolle eines `Prüfling`. Die Beziehung zwischen `Professoren` und `Prüfungen` wird durch eine Assoziation mit dem Namen `abhalten` modelliert; der Professor übernimmt dabei die Rolle des `Prüfer`. Die Beziehung zwischen `Vorlesungen` und `Prüfungen` wird durch eine Assoziation mit dem Namen `umfassen` modelliert; die Vorlesung übernimmt dabei die Rolle des `Prüfungsstoff`.