

# Anmerkungen zu Übungsblatt 1

## ER-Modellierung und DB-Entwurf

Jürgen Kalinski  
Universität Bonn, Institut für Informatik III  
Römerstr. 164, 53117 Bonn  
*cully@cs.uni-bonn.de*

30. Oktober 1998

### Übungsblatt 1, Aufgabe 1

Entities und ihre Beziehungen untereinander sind die primären Bausteine, aus denen eine zu modellierende Mini-Welt aufgebaut ist. Kemper und Eickler [1] beschreiben Entities als *„wohlunterscheidbare physisch oder gedanklich existierende Konzepte“*. Entities wie auch Beziehungen werden durch Eigenschaften (Attribute) näher charakterisiert. Ähnliche Entities werden zu einem gemeinsamen Entity-Typ und ähnliche Beziehungen zu einem Beziehungs-Typ abstrahiert.

Betrachten wir als konkretes Beispiel die Aufgabe 1 des Übungsblattes 2. Das Schema in Abbildung 1 modelliert die Mini-Welt von Studenten, die bei Professoren ihre Prüfungen zu gewissen Vorlesungen ablegen. Jede Prüfung endet mit der Vergabe einer Note.

Das Schema gibt die Struktur der Anwendung wieder. Ihre konkrete Ausprägung zu einem Zeitpunkt „belebt“ die Entity-Typen mit einer Menge von Professoren, Studenten und Vorlesungen. In der Ausprägung des Beziehungstyps *prüfen* finden wir Verknüpfungen zwischen diesen zuvor eingeführten Entities. Attribute entsprechen Eigenschaften der Entities und Verknüpfungen.

Formal gesprochen ordnet eine Instanz  $I$  eines Entity-Relationship-Schemas jedem Entity-Typ  $E$  eine Menge  $I(E)$  von Entities und jedem Beziehungs-Typ  $R$  eine Menge  $I(R)$  von Beziehungen zwischen Entities zu. Wenn der Beziehungs-Typ  $R$  die Entity-Typen  $E_1, \dots, E_n$  miteinander verknüpft, so können wir  $I(R)$  als mathematische Relation, d.h. als Teilmenge eines kartesischen Produkts, auffassen:

$$I(R) \subseteq I(E_1) \times \dots \times I(E_n)$$

Über Attribute können den Entities und den Beziehungen zwischen Entities Eigenschaften zugeschrieben werden. Formal ausgedrückt entspricht ein Attri-

but  $A$  eines Entity-Typs  $E$  (bzw. Beziehungs-Typs  $R$ ) in einer Instanz  $I$  einer Abbildung von  $I(E)$  (bzw.  $I(R)$ ) in den Wertebereich des Attributs:

$$I(A) : I(E) \rightarrow \text{dom}(A) \quad \text{bzw.} \quad I(A) : I(R) \rightarrow \text{dom}(A)$$

Man beachte, daß diese Formalisierung berücksichtigt, daß die bei Kemper und Eickler [1] vorgestellten Konzepte der ER-Modellierung keine mengenwertigen Attribute zulassen.

Für das angegebene Beispielschema ist jede Instantiierung des Beziehungstyps *prüfen* eine Menge von Tripeln bestehend aus Studenten, Professoren und Vorlesungen:

$$\begin{aligned} I(\textit{prüfen}) &= \{ (s_1, p_1, v_1), (s_2, p_2, v_2), \dots \} \\ &\subseteq I(\textit{Student}) \times I(\textit{Professor}) \times I(\textit{Vorlesung}) \end{aligned}$$

Man beachte, daß damit aber das zunächst recht unscheinbare Entity-Relationship-Diagramm festlegt, daß es nur eine Verknüpfung zwischen  $s_1$ ,  $p_1$  und  $v_1$  geben kann (denn  $I(\textit{prüfen})$  ist eine *Menge* von Tripeln). Es ist nicht möglich, in einer Ausprägung die Information zu einer mißratenen Prüfung und ihrer Wiederholung bei demselben Professor zu repräsentieren! Dahingegen können Wiederholungsprüfungen bei anderen Professoren oder zu anderem Prüfungsstoff durchaus repräsentiert werden. Das Schema beinhaltet sogar keinerlei Einschränkungen bezüglich der erlaubten Anzahl an Wiederholungsprüfungen bei jeweils unterschiedlichen Professoren.

Jedem Studenten wird in jeder Instanz eine Matrikelnummer zugewiesen, jedem Professor ein Name, jeder Vorlesung ein Vorlesungstitel — und jedem Prüfungstripel eine Note:

$$\begin{aligned} I(\textit{Note}) : \quad (s_1, p_1, v_1) &\mapsto n_1, \\ &(s_2, p_2, v_2) \mapsto n_2, \dots \end{aligned}$$

Im folgenden wollen wir die zulässigen Instanzen weiter einschränken. Abbildung 2 erfaßt die Bedingung, daß jeder Student zu einer Vorlesung höchstens einmal geprüft werden darf. Die Funktionalitätsangabe besagt nach Definition, daß in jeder zulässigen Instanz jedes Studenten-Vorlesungs-Paar den Professor eindeutig bestimmt:

$$\forall s \in I(\textit{Student}) \forall v \in I(\textit{Vorlesung}) : |\{ p \mid (s, p, v) \in I(\textit{prüfen}) \}| \leq 1$$

In anderen Worten:  $I(\textit{prüfen})$  induziert eine partielle Funktion  $I(\textit{Student}) \times I(\textit{Vorlesung}) \rightarrow I(\textit{Professor})$ .

Eine Prüfungsordnung, nach der jeder Student eine Prüfung zu einer Vorlesung bis zu zweimal ablegen darf, läßt sich mit den Darstellungsmitteln der Vorlesung nicht ausdrücken. Es sollte uns aber auch nichts davon abhalten, bei solchen Anwendungen die Modellierungssprache gegebenenfalls in geeigneter Weise zu erweitern — solange wir exakt sagen können, wovon wir reden. Eine

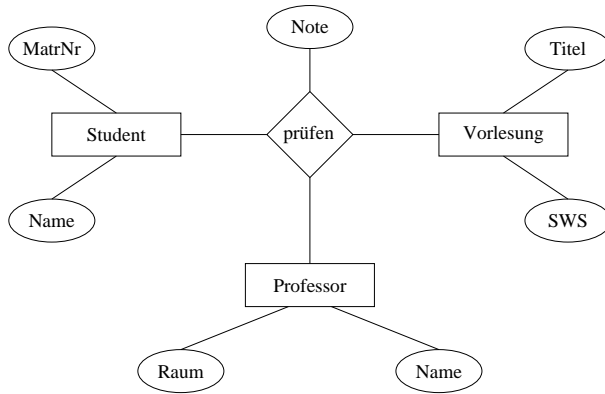


Abbildung 1: Prüfungs-Schema

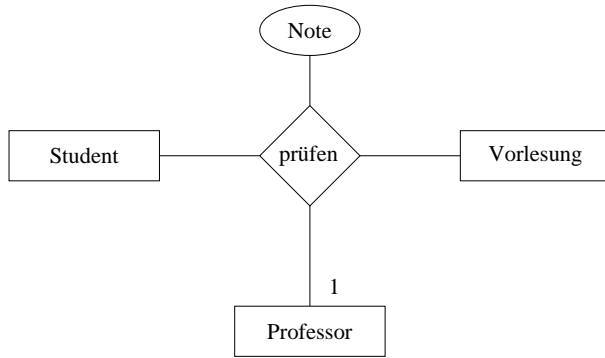


Abbildung 2: Prüfungs-Schema mit Funktionalitätsangabe

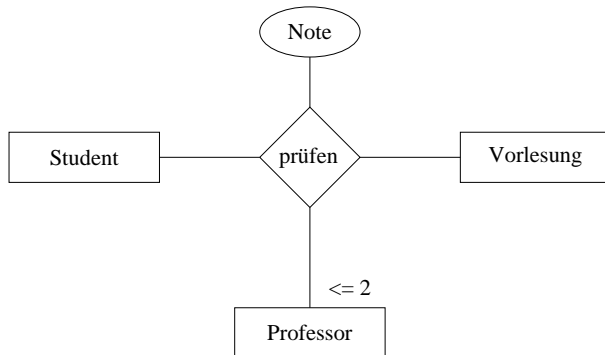


Abbildung 3: Prüfungs-Schema mit Funktionalitätsangabe

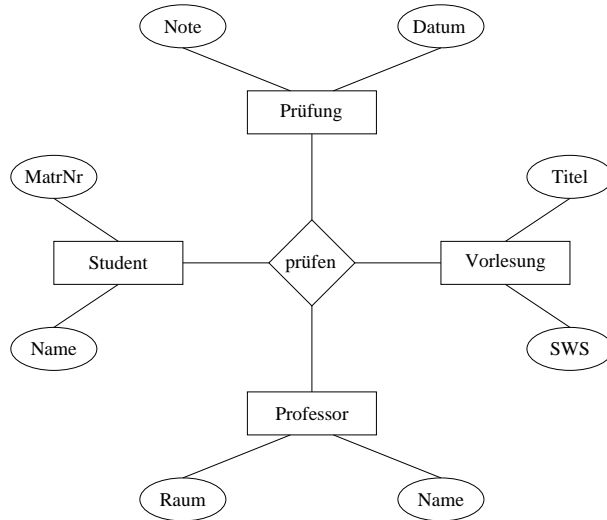


Abbildung 4: Alternatives Prüfungs-Schema

Notation wie die in Abbildung 3 gewählte besagt, daß in jeder zulässigen Instanz  $I$  des Beziehungs-Typs *prüfen* für alle  $s \in I(\text{Student})$  und  $v \in I(\text{Vorlesung})$  gilt:

$$|\{p \mid (s, p, v) \in I(\text{prüfen})\}| \leq 2$$

Man verdeutliche sich in analoger Weise, welche Auswirkungen eine Änderung des ER-Schemas mit sich führt, bei der die Note zu einem eigenen Entity-Typ erhoben wird und/oder die Beziehung um ein Attribut/Entity *Datum* erweitert wird. Wir wollen abschließend noch kurz eine Modellierung betrachten, bei der Prüfungen als Entities aufgefaßt werden (siehe Abbildung 4): Prüfungen werden dabei als selbständige Konzepte gesehen, denen attributiv eine Note und ein Datum zugeschrieben werden und die über universitäre Verwaltungsvorgänge mit weiteren Entities verknüpft werden. Diese Modellierung gilt nun auch für Prüfungsordnungen, bei denen ein Student bei einem Professor mehr als einmal zu derselben Vorlesung geprüft werden kann.

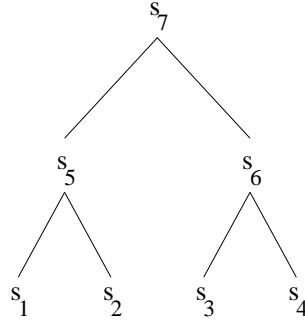


Abbildung 5: Spielbaum eines Tennis-Turniers

## Übungsblatt 1, Aufgabe 2

Wir entwickeln im folgenden ein ER-Schema, dessen Instanzen Daten zu jeweils *einem* bereits abgeschlossenen Tennisturnier verwalten. Das Schema besteht aus nur zwei Entity-Typen *Spiel* und *Spieler*. Eine Beziehung *teilnehmen* verknüpft ein Spiel mit den beiden teilnehmenden Spielern. Zur Unterscheidung der zwei Kanten, die im ER-Diagramm von der Raute des Beziehungs-Typs zum Rechteck des Entity-Typs *Spieler* laufen, versehen wir sie mit den Rollenangaben *Sieger* und *Verlierer*.

Jedem Spieler können sein Name als Schlüsselattribut und seine Punkte in der Weltrangliste als weiteres Attribut zugeschrieben werden, jedem Spiel eine identifizierende Nummer und das Austragungsdatum.

Abgesehen von den Spielen der ersten Runde ergeben sich die Teilnehmer eines Spiels aus jeweils genau zwei vorangegangenen Spielen. Die Vater-Sohn-Beziehung zwischen den Knoten des Spielbaums (siehe Abbildung 5) wollen wir ebenfalls modellieren; eine rekursive Beziehung *vor* verknüpft ein Spiel mit einem vorangegangenen Spiel. Während die Beziehung *teilnehmen* ein Spiel mit zwei Spielern verknüpft, kann den zwei direkten Vorgänger eines Spiels kein unterschiedlicher Status zugeschrieben werden. Statt einer dreistelligen Beziehung ziehen wir hier deshalb eine binäre Beziehung vor. Um zwischen den Kanten der rekursiven Beziehung unterscheiden zu können, markieren wir sie wiederum mit Rollenangaben (*Vorgänger* und *Nachfolger*).

Wie schon gesagt, hat abgesehen von den Spielen der ersten Runde jedes Spiel genau einen Nachfolger. Desweiteren hat jedes Spiel außer dem Finalspiel genau zwei Vorgänger. Diese Einschränkungen können mit den in Abbildung 6 vorgenommenen (min,max)-Markierungen ausgedrückt werden. Wenn  $(s_5, s_7) \in I(vor)$  bedeutet, daß der Gewinner des Spiels  $s_5$  zu einem der Teilnehmer des Spiels  $s_7$  wird, so besagt die (0, 2)-Markierung beim Nachfolger nach Definition: Zu jedem  $s \in I(Spiel)$  gibt es mindestens 0 und höchstens 2 Paare in  $I(vor)$ , bei dem  $s$  an der zweiten (d.h. der Nachfolger-) Position steht:

$$I(vor) = \{(s_1, s_5), (s_2, s_5), (s_3, s_6), (s_4, s_6), (s_5, s_7), (s_6, s_7)\}$$

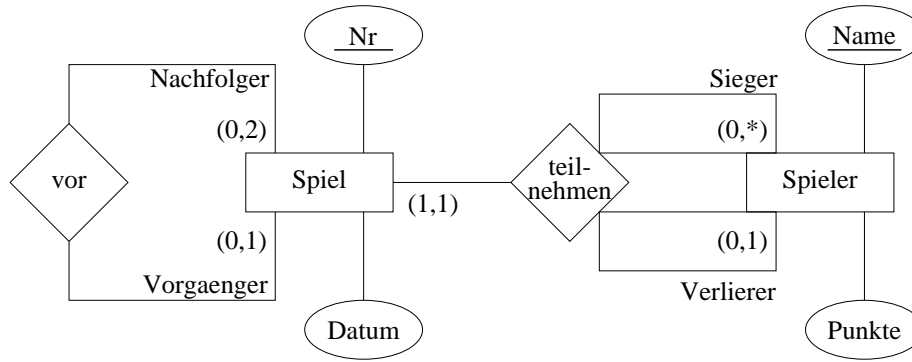


Abbildung 6: ER-Schema für Tennis-Turnier

Es gilt sogar (was sich allerdings mit (min,max)-Markierungen nicht ausdrücken läßt): Entweder kommt  $s$  kein einziges Mal in der Nachfolgerposition vor (nämlich dann, wenn  $s$  ein Erstrundenspiel ist) oder genau zweimal.

Untersuchen wir nun, welche Einschränkungen sich direkt aus dem Austragungsmodus (k.o.-System) ergeben: Jeder Spieler nimmt an mindestens einem Spiel teil. Er kann beliebig häufig als Gewinner, jedoch nur einmal als Verlierer aus einem Spiel herausgehen. Die beiden letzten Bedingungen werden erneut durch (min,max)-Markierungen ausgedrückt. Die (0, 1)-Markierung besagt, daß jeder Spieler  $p \in \text{Spieler}$  in mindestens 0 und höchstens einem Tupel von  $I(\text{teilnehmen})$  als Verlierer auftritt. Die (1, 1)-Markierung beinhaltet, daß jedes Spiel  $s \in \text{Spiel}$  in genau einem  $I(\text{teilnehmen})$ -Tupel vorkommt.

Bei der Umsetzung eines ER-Schemas in ein relationales Datenbankschema können wir in einem ersten Schritt für jeden Entity-Typ und eine Relation definieren. Die Relation umfaßt alle Attribute des Entity-Typs und besitzt den gleichen Schlüssel. Aus den Entity-Typen des in Abbildung 6 dargestellten Schemas können somit zwei Relationen hergeleitet werden:

$$\begin{aligned} \text{Spiel} &= \{ [\underline{\text{Nr}}, \text{Datum}] \} \\ \text{Spieler} &= \{ [\underline{\text{Name}}, \text{Punkte}] \} \end{aligned}$$

Wir werden für diese Relationen im folgenden den Begriff Entity-Relationen benutzen. Eine Instanz wie

$$\begin{aligned} I(\text{Spiel}) &= \{ s, \dots \} \\ I(\text{Nr}) &= \{ s \mapsto 1427, \dots \} \\ I(\text{Datum}) &= \{ s \mapsto 13. \text{ April}, \dots \} \\ I(\text{Spieler}) &= \{ p_1, p_2, \dots \} \\ I(\text{Name}) &= \{ p_1 \mapsto \text{John.Smith}, p_2 \mapsto \text{William.Miller}, \dots \} \\ I(\text{Punkte}) &= \{ p_1 \mapsto 35, p_2 \mapsto 39, \dots \} \end{aligned}$$

<i>Spiel</i>	Nr	Datum	<i>Spieler</i>	Name	Punkte
	1427	13. April		John Smith	35
	⋮	⋮		William Miller	39
				⋮	⋮

Abbildung 7: Entity-Relationen des Tennis-Turniers

<i>teilnehmen</i>	Nr	Sieger	Verlierer
	1427	John Smith	William Miller
	⋮	⋮	⋮

Abbildung 8: Darstellung der Beziehung (Version 1)

impliziert den in Abbildung 7 aufgezeigten Datenbankzustand.

Desweiteren wird auch aus jedem Beziehungs-Typ eine Relation hergeleitet. Instantiierungen eines Beziehungs-Typs verknüpfen Entities miteinander. Im relationalen Modell gibt es streng genommen keine Entities, aber jedem Entity entspricht genau ein Tupel in der korrespondierenden Relation. Dieses Tupel wiederum wird durch die Belegung der Schlüsselattribute eindeutig bestimmt. Eine Verknüpfung mehrerer Entities entspricht somit einer Kombination von deren Schlüsselwerten. Das Schema einer Beziehungs-Relation ergibt sich daher im ersten Schritt aus der Vereinigung der Schlüsselattribute der beteiligten Entity-Relationen zusammen mit den Beziehungsattributen.

Wir sehen, daß dieses Verfahren in unserem Beispiel nicht zu wörtlich genommen werden darf. Jedes Element der Beziehung *teilnehmen* verknüpft ein Spiel mit zwei Spielern. Jedes Tupel der Beziehungs-Relation besteht also aus der identifizierenden Nummer eines Spiels und den Namen der zwei Spieler. Eine einfache Übernahme der Bezeichnungen der Schlüsselattribute aus den Entity-Relationen führt bei den Spielern zu einem Namenskonflikt. Wir benennen die Namensschlüssel in der Beziehungs-Relation deshalb in Anlehnung an die Rollenangaben um (und aus ästhetischen Gründen auch die Spielnummer):

$$teilnehmen = \{ [ \text{Spiel}, \text{Sieger}, \text{Verlierer} ] \}$$

Die Instanz  $I(teilnehmen) = \{ (s, p_1, p_2), \dots \}$  impliziert den Datenbankzustand aus Abbildung 8.

Falls die Beziehung keinen bekannten Einschränkungen unterliegt, bildet die Vereinigung der Schlüssel der beteiligten Entity-Relationen (in unserem Fall  $\{ \text{Spiel}, \text{Sieger}, \text{Verlierer} \}$ ) den Schlüssel der Beziehungs-Relation. Im vorliegenden Beispiel müssen wir allerdings etwas mehr Sorgfalt walten lassen.

Keine Spielnummer kommt in zwei Tupeln der Relation *teilnehmen* vor. Die Spielnummer allein bildet also schon einen Schlüssel. Ferner kann kein Spieler zweimal als Verlierer in der Relation *teilnehmen* verzeichnet sein. Auch das

<i>Spiel</i>	Nr	Datum	Sieger	Verlierer
	1427	13. April	John Smith	William Miller
	⋮	⋮	⋮	⋮

<i>Spieler</i>	Name	Punkte
	John Smith	35
	William Miller	39
	⋮	⋮

Abbildung 9: Relationen des Tennis-Turniers (Version 2)

<i>Spiel</i>	Nr	Datum
	1427	13. April
	⋮	⋮

<i>Spieler</i>	Name	Punkte	Spiel	Gewinner
	John Smith	35	-	-
	William Miller	39	1427	John Smith
	⋮	⋮	⋮	⋮

Abbildung 10: Relationen des Tennis-Turniers (Version 3)

Attribut Verlierer ist deshalb ein Schlüssel. Wir betrachten im folgenden beide Varianten:

1. Nehmen wir zunächst an, daß wir die Spielnummer als Primärschlüssel der Beziehungs-Relation auswählen. Dann gibt es eine weitere Vereinfachungsmöglichkeit. Wenn ein Spiel seinen Gewinner und Verlierer eindeutig determiniert, können diese Angaben auch gleich in die Relation *Spiel* mit eingezogen werden. Wir erhalten das Schema und die Beispielausprägung aus Abbildung 9.
2. Alternativ können wir auch den Verlierer als Primärschlüssel auswählen. Dann gilt, daß die *teilnehmen*-Information in die Relation *Spieler* mit aufgenommen werden kann. Zu jedem Spieler wird verzeichnet, in welchem Spiel er von welchem anderen Spieler geschlagen wurde (siehe Abbildung 10). Diese Variante krankt allerdings (abgesehen von ihrer geringeren Natürlichkeit, was zugegebenermaßen ein recht subjektives Kriterium sein mag) daran, daß beim Turniersieger unter den Attributen Spiel und Gewinner Nullwerte aufgenommen werden müssen.

Übungsblatt 3 wird die Problematik von Nullwerten kurz ansprechen. Es sollte aber klar sein, daß für die betrachtete Anwendung die Version 2 das bessere Datenbankschema darstellt.

## Literatur

- [1] KEMPER, ALFONS und ANDRÉ EICKLER: *Datenbanksysteme: Eine Einführung*. R. Oldenbourg Verlag, München, Wien, 1996.