SoSe 2024

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Datenbanken

- Übung 3 -
Semantisches Schema (Teil 2)

Markus Schneider
Robert Hartmann

Prof. Dr. H. Knolle
Hochschule Bonn-Rhein-Sieg
Fachbereich Informatik
Grantham-Allee 20
53757 Sankt Augustin

Inhaltsverzeichnis

1 Fortgeschrittene Entity-Relationship-Modellierung

In der letzten Übung haben Sie ein Entity-Relationship-Diagramm für ein Informationssystem entworfen. Dabei haben Sie die grundlegenden ER-Konstrukte wie Entitäts- und Beziehungstypen, Attribute, Kardinalitäten und Schlüssel verwendet.

Nun geht es darum, die in der Vorlesung eingeführten semantisch fortgeschrittenen ER-Konzepte zu üben. Dazu erweitern Sie Ihren fachlichen Entwurf aus Übung 2 um die Konstrukte aus Abschnitt 2 dieses Übungsblatts.

1.1 Lernziele der Übung

Ziel ist es, folgende Kompetenzen zu entwickeln:

- ✔ Fortgeschrittene Entity-Relationship-Konstrukte einsetzen können
- ✔ Die Semantik und Auswirkungen der Konstrukte nachvollziehen und erklären können
- ✔ Beurteilen können, welche Konstrukte in einem konkreten Fall fachlich sinnvoll sind

1.2 Vorbereitung der Übung

Diese Übung baut auf Ihren Ergebnissen der Übung 2 auf und benötigt keine neue Software.

1.3 Einreichung Ihrer Ergebnisse

Die Einreichung Ihrer Übungsergebnisse erfolgt über den Praktomaten, der während des Uploads eine formale Prüfung Ihrer Dateien durchführt und Ihnen Feedback zu Umfang und syntaktischer Korrektheit gibt. Die Semantik Ihrer Einreichung wird zwar nicht geprüft, allerdings werden Hinweise gegeben, die Sie selbst prüfen und mit Ihren Übungspartnern diskutieren sollten.

Eine Einreichung umfasst folgende Dateien:

- „`EigeneDB.pdf`“: Skizze Ihrer Entitäts- und Beziehungstypen (aus Übung 1)
- „`EigeneDB.ods`“: Spezifikation Ihres Diskursbereiches (ggf. mit Anpassungen)
- „`EigeneDB.lun`“: Semantisches Schema, erweitert um Konstrukte aus Abschnitt 2

Wenn Sie Feedback und eine Diskussion zu Ihrem Entwurf im Rahmen der Übung wünschen, bestätigen Sie dies bitte wie gehabt in der „`EigeneDB.ods`“-Datei.

2 Fortgeschrittene ER-Konzepte

2.1 Rekursive Beziehungstypen

Fügen Sie Ihrem Diagramm einen rekursiven Beziehungstypen hinzu, der einen Entitätstypen semantisch sinnvoll mit demselben Entitätstypen verbindet. In aller Regel sollte sich in Ihrem Ergebnis aus Übung 2 bereits ein geeigneter Entitätstyp finden. Typische Beispiele für rekursive Beziehungstypen sind:

- „hat_chef“: Ein Mitarbeiter hat einen (anderen) Mitarbeiter als Chef.
- „hat_freund“: Eine Person ist mit einer (anderen) Person befreundet.
- „hat_unterabteilung“: Eine Abteilung besteht aus (anderen) Abteilungen.

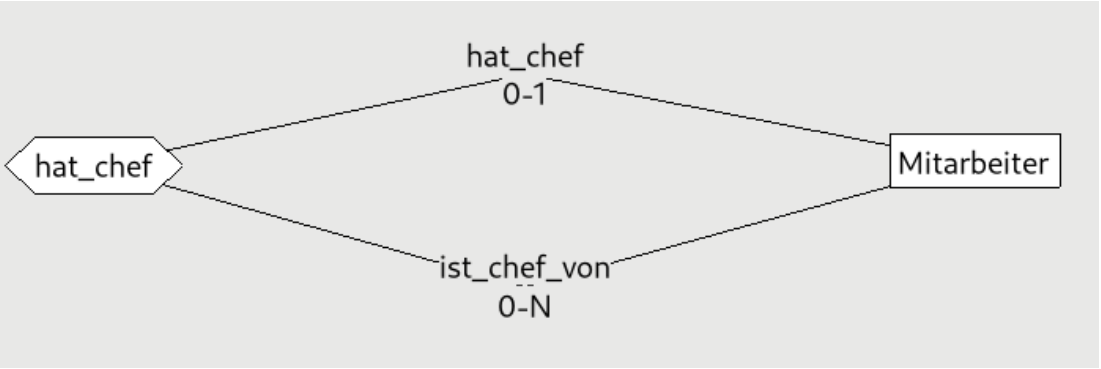

Abbildung 1: Beispiel für einen rekursiven Beziehungstypen

Um einen rekursiven Beziehungstypen in DB-Main umzusetzen, wählen Sie zunächst das Icon mit der Raute und erzeugen einen alleinstehenden Beziehungstyp. Wählen Sie im Anschluss das Fadenkreuz-Symbol und verbinden Sie den neuen Beziehungstypen doppelt mit dem Entitätstypen.

Bei rekursiven Beziehungstypen sollten die Kardinalitäten mit fachlichen Rollenbezeichnern versehen werden. In Abbildung 1 sind z.B. die Bezeichner „hat_chef“ und „ist_chef_von“ vergeben worden, um festzuhalten, in welcher Rolle die Entität wie oft an der Beziehung teilnehmen kann:

- Ein Mitarbeiter kann in der Rolle `„ist_chef_von“` bis zu N-mal teilnehmen, da er Chef von mehreren Mitarbeitern sein kann.
- Ein Mitarbeiter kann in der Rolle `„hat_chef“` maximal 1-mal teilnehmen, da er nur einen Chef haben kann. Der oberste Chef hat selbst keinen Chef.

2.2 Generalisierungs-/Spezialisierungs-Hierarchien

Das Konzept der IST-Beziehung bzw. der Generalisierungs-/Spezialisierung dient zum Modellieren von Vererbungsbeziehungen. Dadurch wird einerseits die semantische Ausdrucksstärke des Diagramms erhöht, da fachlich relevante Taxonomien („Begriffspyramiden“) abgebildet werden können. Andererseits können dadurch redundante Attributdefinitionen vermieden und fachliche Beziehungstypen zusammengefasst werden (siehe Abbildungen 2+3).

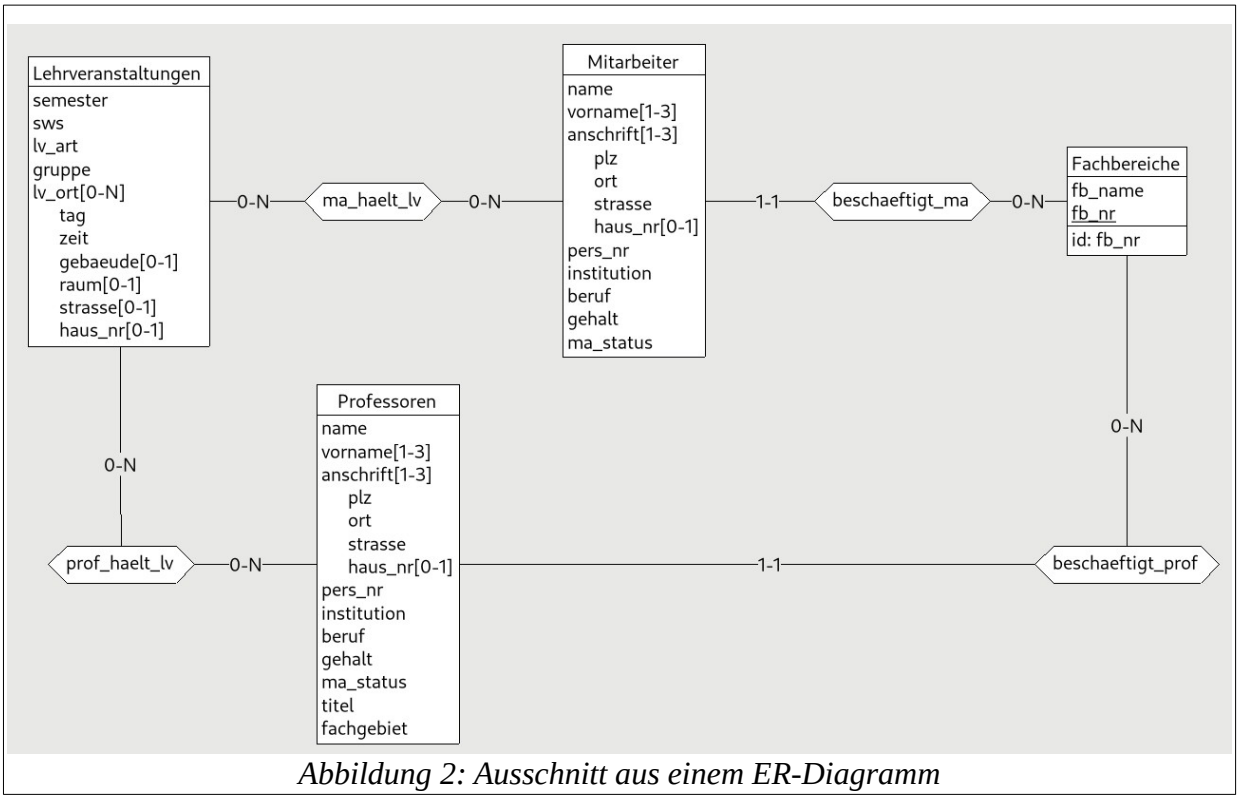

Abbildung 2: Ausschnitt aus einem ER-Diagramm

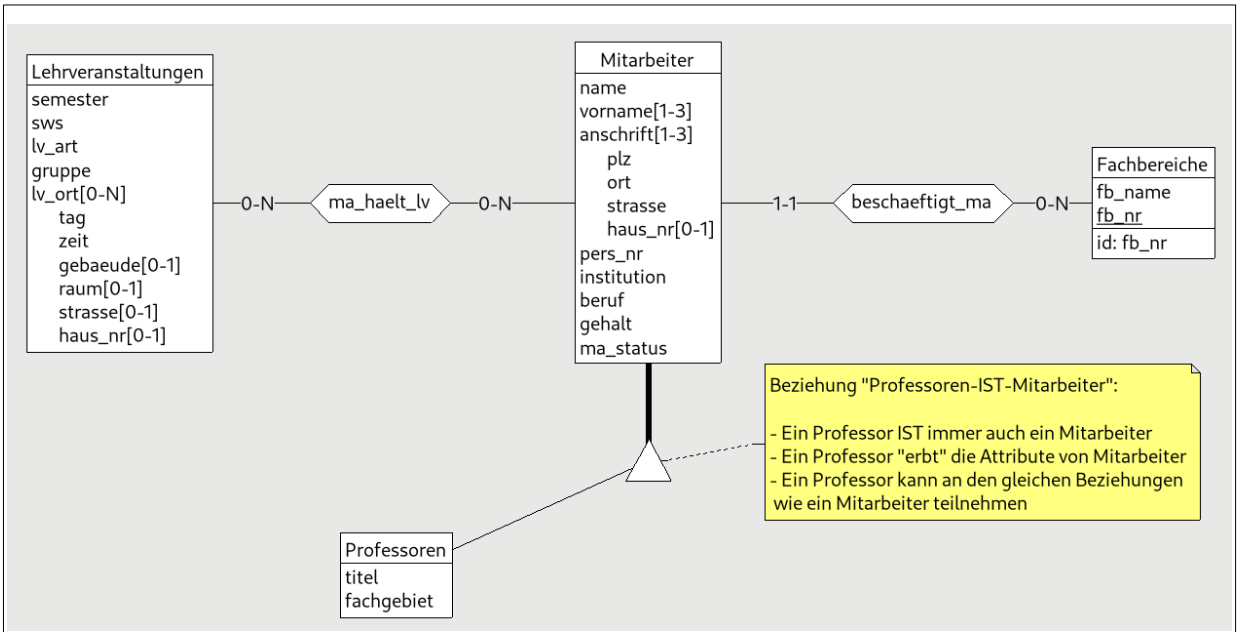

Abbildung 3: Einführung einer IST-Beziehung zwischen Professoren und Mitarbeitern

Das Einführen einer IST-Beziehung wirkt sich ggf. auf die Operationen der Anwendungsfälle aus, die Sie in Übung 1 erfasst haben: Wird beispielsweise nach der Einführung der IST-Beziehung eine Professoren-Entität neu erstellt, so muss immer auch eine zugehörige Mitarbeiter-Entität und eine Beziehung angelegt werden (Zeile 8 und 10):

	A	B	C	D	E	F	G
1	**Anwendungsfall**	**Teammitglied**	**Beziehungen**	**Entitäten**	**Lesezugriff**	**Modifikation**	**Erstellung**
8	Neuanstellung Professor			Mitarbeiter			x
9				Professoren			x
10			Professoren-IST-Mitarbeiter				x
11				Fachbereiche	x		
12			beschaeftigt_ma				x

Abbildung 4: Auswirkungen der IST-Beziehung auf die Operationen eines Anwendungsfalls

Verwenden Sie als Bezeichner für IST-Beziehungen in der Anwendungsfälle-Tabelle die Form „`Subtyp-IST-Supertyp`“, also z.B. „`Professoren-IST-Mitarbeiter`“. Dadurch versteht der Praktomat, welche IST-Beziehung in Ihrem ER-Diagramm gemeint ist.

Identifizieren Sie nun vorhandene und mögliche IST-Beziehungen in Ihrem Schema und modellieren Sie diese als Generalisierung/Spezialisierung:

- Falls Sie die IST-Beziehung bereits durch einen normalen ER-Beziehungstypen erfasst haben, löschen Sie diesen zunächst im ER-Diagramm.
- In DB-Main erfolgt die Erstellung einer IST-Beziehung über die „`Property box`“ des Sub-Typen. Dort erlaubt die Eigenschaft „`supertypes`“ das Einstellen eines oder mehrerer Super-Typen.

Wie in der Vorlesung diskutiert, hat eine Generalisierung/Spezialisierung bestimmte Eigenschaften, die dazu dienen, die fachliche Integrität der IST-Beziehung zu unterstützen:

- „partiell“ vs. „total“: Sind nicht spezialisierte Entitäten zulässig?

- „disjunkt“ vs. „nicht disjunkt“: Kann eine Entität mehrere Spezialisierungen annehmen?

Diese Eigenschaften werden in DB-Main über die „`Property box`“ des generalisierten Super-Typen eingestellt (Checkboxen „`total`“ und „`disjoint`“).

2.3 Assoziationen, Aggregationen und Kompositionen

Ein weiteres semantisches Konzept für Beziehungstypen ist die Klassifizierung ihrer Teil-/Ganzes-Semantik. Beispielsweise können Räume („Teile“) mit ihrem Gebäude („Ganzes“) in Beziehung gesetzt werden. Es werden folgende Kategorien unterschieden:

- **Assoziation**: Keine Teil-/Ganzes-Semantik.
- **Aggregation:** Besteht zwischen starken eigenständigen Entitätstypen, bei denen eine Seite ein „übergeordnetes Ganzes“ darstellt. Das „Ganze“ bezeichnet man dabei als Aggregat, die „Teile“-Entitäten werden als Komponenten bezeichnet.
- **Komposition:** Spezialfall der Aggregation, bei dem die Komponenten keine vollständige eigene Identität haben, also schwache Entitäten sind (siehe Abschnitt 2.4).

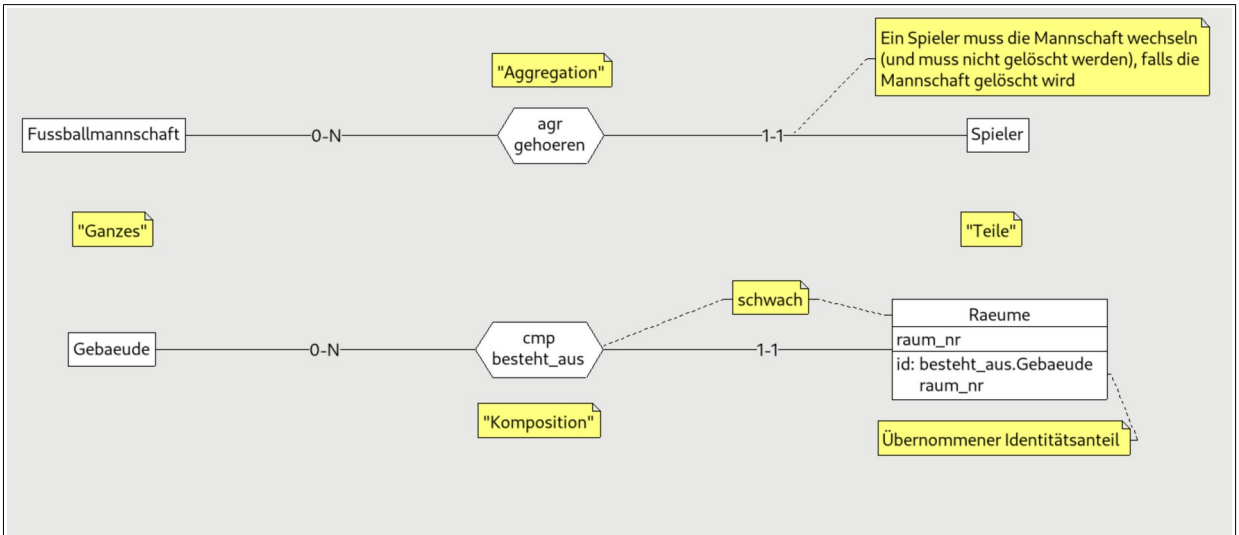

Abbildung 5: Beispiele für Aggregation und Komposition

Die Auszeichnung von Aggregationen und Kompositionen erfolgt in DB-Main über einen Klick auf die Kardinalität der übergeordneten Entität. In der „`Property box`“ kann nun unter „`Aggregation`“ zwischen „`aggregation`“, „`composition`“ und leer (Assoziation) gewählt werden.

Hinweis: Beim wiederholten Einstellen der Teil-/Ganzes-Semantik tritt in DB-Main der Fehler auf, dass die alte Kennzeichnung am Beziehungstypen stehen bleibt. In diesem Fall können Sie den Beziehungstyp anklicken, in der „`Property box`“ den Reiter „`User Prop`“ auswählen, „`Stereotype`“ und dann „…“ anklicken und dort „`Clear`“ betätigen. Danach können Sie die Kennzeichnung wie oben beschrieben neu durchführen.

2.4 Schwache Entitätstypen

Schwache Entitätstypen haben keine vollständige eigene Identität und eine Entität dieses Typs kann ohne die verbundene Entität nicht existieren. Dies hat Konsequenzen auf die folgenden Ebenen des Datenbankentwurfs, insbesondere auf die Schlüsselkonfiguration und die Kaskadierung von Löschoperationen.

DB-Main bietet kein vollständiges Konzept zur Kennzeichnung schwacher Entitätstypen an, wir empfehlen das folgende Vorgehen:

- Markieren Sie den schwachen Entitätstypen mit Hilfe einer gelben Notiz („schwach“).
- Im Fall von binären Beziehungstypen mit schwachen Entitätstypen (wie in Abbildung 5) definieren Sie einen Schlüssel im schwachen Entitätstypen, der auf der Identität des übergeordneten Entitätstypen basiert: Wählen Sie ein (Schlüssel-) Attribut des schwachen Entitätstypen aus und klicken den „`ID`“-Button zum Definieren eines Schlüssels. In der „`Property box`“ wählen Sie anschließend unter „`components`“ den Beziehungstypen zum übergeordneten Entitätstypen als (zusätzliche) Schlüsselkomponente aus.
- Im Fall von Beziehungstypen mit höherem Grad als zwei kann der Schlüssel mit den übernommenen Anteilen nicht in DB-Main konfiguriert werden. Dokumentieren Sie den zusammengesetzten Schlüssel daher in einer Notiz (siehe Abbildung 9).

2.5 Versionierte „N:M“-Beziehungstypen

Ein „N:M“-Beziehungstyp erlaubt, dass eine Entität mit beliebig vielen Entitäten der anderen Seite in Beziehung stehen kann. Im Beispiel in Abbildung 6 kann ein Student beliebig viele Bücher ausleihen. Umgekehrt kann ein Buch von beliebig vielen Studenten ausgeliehen werden.

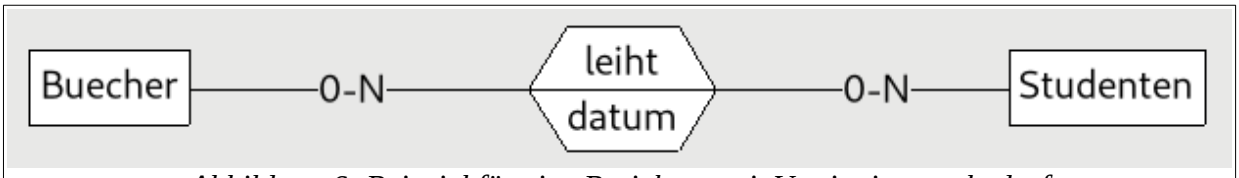

Abbildung 6: Beispiel für eine Beziehung mit Versionierungsbedarf

Was in dieser Modellierung noch nicht erfasst werden kann: Das Szenario, dass ein Student wiederholt dasselbe Buch ausleiht. Wie in der Vorlesung diskutiert, ergibt sich die Identität einer Beziehung aus den beteiligten Entitäten. Daraus folgt, dass eine bestimmte Kombination von Entitäten im Rahmen eines Beziehungstyps nur einmal erfasst werden kann. Der Fall in Abbildung 7 ist also mit dem obigen Beziehungstypen nicht darstellbar.

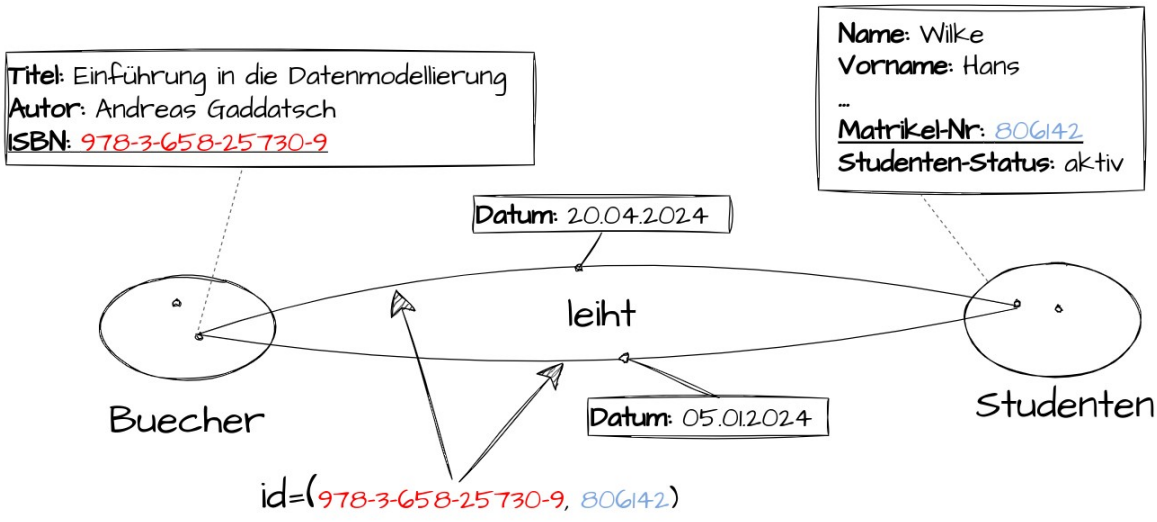

Abbildung 7: Nicht möglich: Zwei Verbindungen zwischen denselben Entitäten

Die Lösung besteht darin, die Beziehungen um Versionierung-Entitäten zu erweitern, die das Datum speichern (siehe Abbildung 8). Danach hat jede „leiht“-Beziehung insgesamt drei beteiligte Entitäten und die beiden Beziehungen haben unterschiedliche Identitäten.

Die zugehörige Modellierung ist in Abbildung 9 dargestellt. Eine Versionierungsentität ist selbst schwach und braucht keine eigene vollständige Identität: Die Identität eines Ausleihvorgangs besteht aus Buch, Student und Datum. Dies ist leider in DB-Main nicht direkt modellierbar und sollte durch eine Notiz dokumentiert werden.

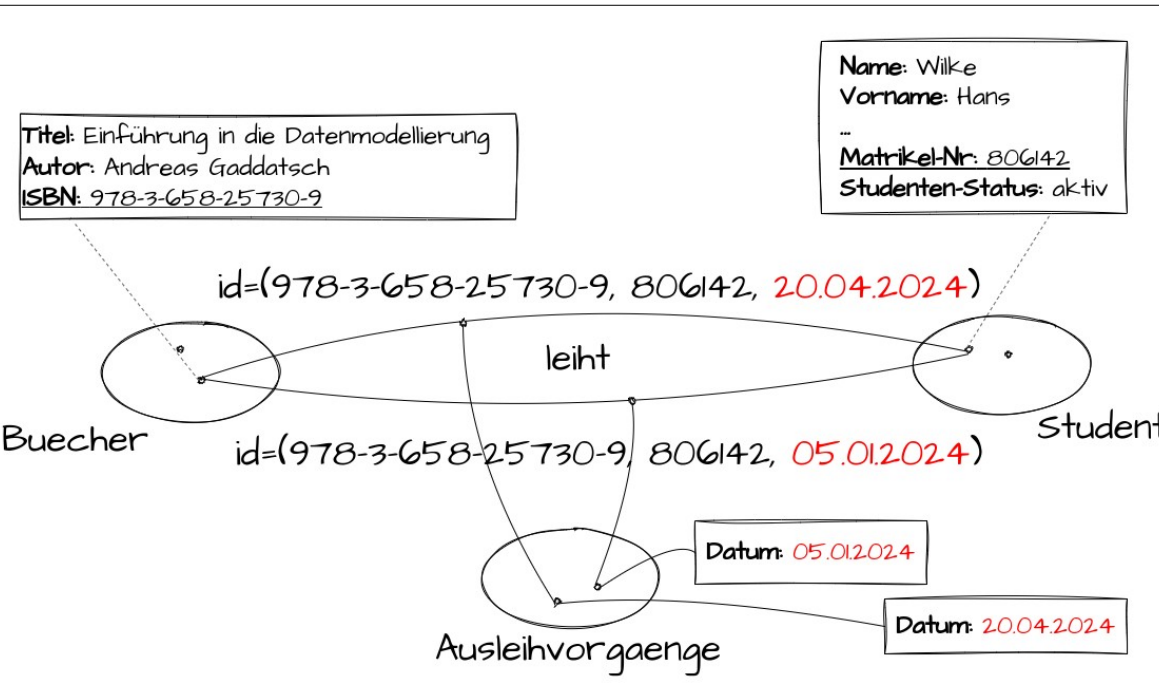

Abbildung 8: Die Versionierungsentität liefert einen weiteren Identitätsanteil für die Beziehung

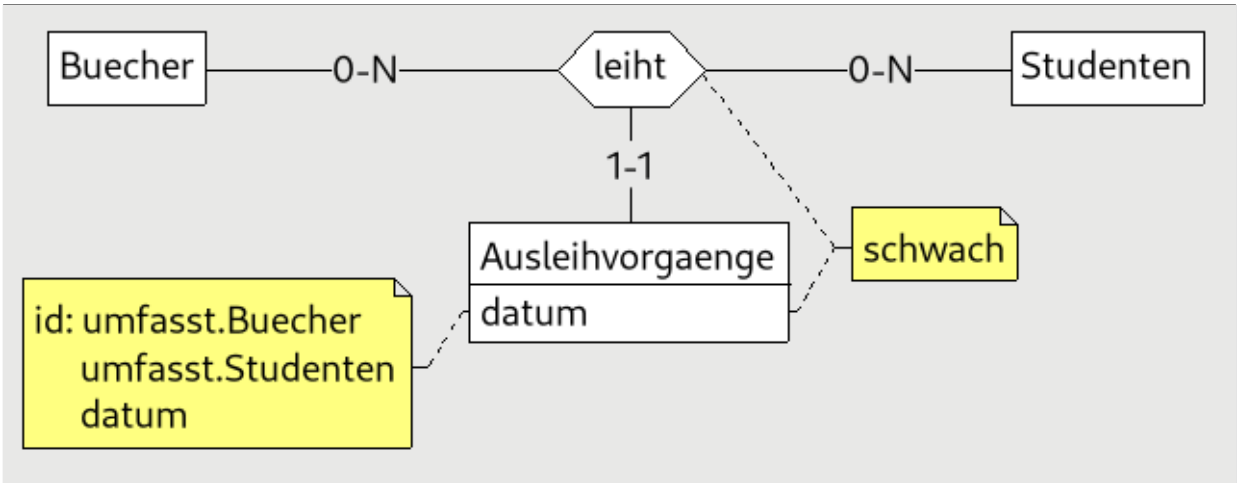

Abbildung 9: Einführung einer Versionierungsentität

Man bezeichnet Beziehungstypen, die solche Mehrfachbeziehungen zwischen denselben Entitäten zulassen als Versionierung bzw. Historisierung. Das erklärt sich daraus, dass sie meist dazu dienen, sich verändernde Beziehungen im Verlauf der Zeit erfassen zu können.

Es gibt übrigens nur versionierte „N:M“-Beziehungen:

- Würde man eine „1:N“-Beziehung versionieren, so entsteht letztendlich ein „N:M“-Beziehungstyp. Beispiel: Eine Person hat zu einem Zeitpunkt eine Adresse, eine Adresse kann gleichzeitig zu mehreren Personen gehören. Diese „1:N“-Beziehung erweitert sich zwangsläufig zu einer „N:M“-Beziehung, wenn man die historischen Adressen einer Person erfassen möchte, da Personen nun auch mit ihren vergangenen (also insgesamt mehreren) Adressen in Beziehung stehen.
- Möchte man in einer „1:1“-Beziehung historische Daten berücksichtigen, so entsteht zunächst zwangsläufig ein „1:N“-Beziehungstyp. Beispiel: Ein Student hat zu einem Zeitpunkt genau einen Studienplatz, ein Studienplatz kann zu einem Zeitpunkt höchstens einem Studenten zugeordnet werden. Wenn man an der Historie aller Studienplätze einer Person interessiert ist, so geht dies mit einer „1:N“-Beziehung, denn eine Person hat über die Zeit betrachtet eventuell mehrere Studienplätze. Damit steht aber umgekehrt ein Studienplatz potentiell mit mehreren Studenten in Beziehung. Das geht wiederum nur mit einer „N:M“-Beziehung. Wenn nun ein Student auch denselben Studienplatz wiederbekommen kann, liegt wieder eine versionierte „N:M“-Beziehung vor.