

Datenbanken

Diagramme und Modellierung

Thomas Studer

Institut für Informatik
Universität Bern

Datenmodellierung

Es geht darum, ein DB-Schema zu finden, so dass

- ① alle benötigten Daten im DB-Schema abgespeichert werden können,
- ② effizient auf die Daten zugegriffen werden kann und
- ③ die Datenkonsistenz gewährleistet ist.

Einfache Tabelle

Name der Tabelle
<u>Name des 1. Attributs</u>
<u>Name des 2. Attributs</u>
Name des 3. Attributs
Name des 4. Attributs

Tabelle mit vier Attributen, wobei die ersten beiden den Primärschlüssel bilden

Vereinfachung

Name der Tabelle

Konkret

Autos

Marke

Farbe

Baujahr

FahrerId

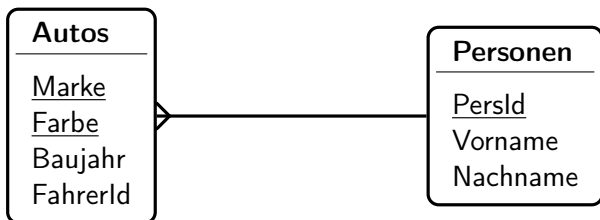
Personen

PersId

Vorname

Nachname

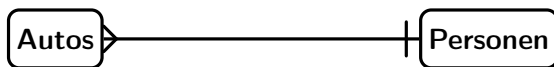
Fremdschlüssel-Beziehung



Eine solche Beziehung heisst *m:1-Beziehung*.

Existenzbedingung

Mit Hilfe eines *not null Constraints* auf dem Fremdschlüsselattribut `FahrerId` können wir verlangen, dass es zu jedem Auto *mindestens* einen Fahrer geben muss.

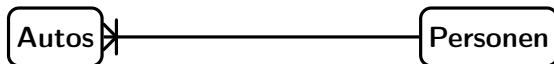


Diese Darstellung drückt zwei Sachverhalte aus:

- 1 Jedes Auto hat genau einen Fahrer, das heisst mindestens einen und auch höchstens einen Fahrer.
- 2 Jede Person kann kein, ein oder mehrere Autos fahren.

Existenzbedingung auf der anderen Seite

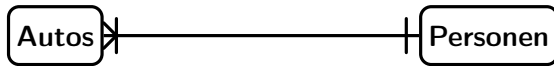
- ① Jedes Auto hat keinen oder einen Fahrer.
- ② Jede Person fährt ein oder mehrere Autos.



Diese Existenzbedingung können wir im relationalen Modell *nicht* durch einen not null Constraint auf einem Attribut ausdrücken.

Existenzbedingung auf beiden Seiten

- 1 Jedes Auto hat genau einen Fahrer.
- 2 Jede Person kann ein oder mehrere Autos fahren.



Zusammenfassung

Beschreibung	Symbol
Keine oder eine Beziehung	—
Keine, eine oder mehrere Beziehungen	—<
Genau eine Beziehung	—+
Eine oder mehrere Beziehungen	—*

m:n-Beziehung

Betrachten wir ein DB-Schema für eine Bank, welche Kunden und ihre Konten verwalten muss. Dabei soll folgendes gelten:

- 1 Ein Kunde kann mehrere Konten haben.
- 2 Ein Konto kann mehreren Kunden gemeinsam gehören.

DB-Schema:

$$\mathcal{S}_{\text{Kunden}} := (\underline{\text{KundenNr}}, \text{Name})$$

$$\mathcal{S}_{\text{Konten}} := (\underline{\text{KontoNr}}, \text{Stand})$$

$$\mathcal{S}_{\text{KuKo}} := (\underline{\text{KundenNr}}, \underline{\text{KontoNr}})$$

Tabellen für m:n-Beziehungen

Kunden

KundenNr	Name
A	Ann
B	Tom
C	Eva
D	Bob

KuKo

KundenNr	KontoNr
A	1
A	2
B	2
C	3
D	3

Konten

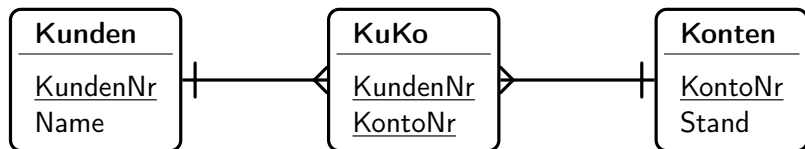
KontoNr	Stand
1	1000
2	5000
3	10

Diagramm

Zu jedem Eintrag in der KuKo-Tabelle müssen die entsprechenden Kunden und Konten existieren. Das heisst,

- 1 das Attribut KundenNr in KuKo ist ein Fremdschlüssel auf Kunden,
- 2 das Attribut KontoNr in KuKo ist ein Fremdschlüssel auf Konten.

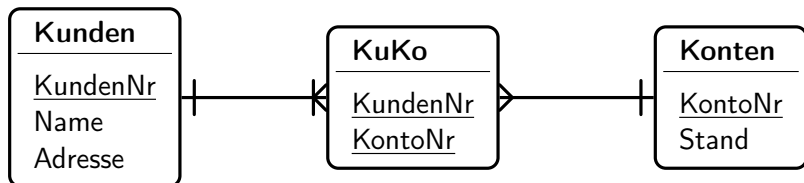
Diese Fremdschlüssel müssen einen not null Constraint erfüllen, da sie Teil des Primärschlüssels des KuKo Schemas sind.



Existenzbedingung

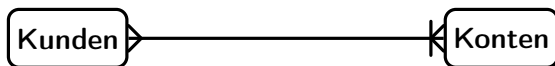
- 1 Hat jeder Kunde ein Konto?
- 2 Muss jedes Konto einem Kunden gehören?

Wir beantworten die erste Frage mit *ja*. Wer kein Konto hat, kann auch kein Kunde sein. Die zweite Frage verneinen wir. Unsere Bank ist etwas altmodisch und lässt nachrichtenlose Vermögen (d.h. Konten ohne bekannte Kundenbeziehung) zu.



Kurzform

In diesem Beispiel enthält die Tabelle KuKo keine weiteren Daten ausser den Primärschlüsseln für Kunden und Konten. Im Prinzip können wir somit die Box für dieses Schema bei der Diagrammdarstellung des DB-Schemas weglassen.



Frage

Wie könnten wir eine Prüfung modellieren, die

- 1 ein Studierender abgelegt hat,
- 2 von einer Professorin erstellt wurde,
- 3 zu einer bestimmten Vorlesung gehört,
- 4 an einem bestimmten Datum stattfand
- 5 und eine Note hat.



Ternäre Beziehungen

DB-Schema für Universität:

$\mathcal{S}_{\text{Studierende}} := (\underline{\text{MatNr}}, \text{Name})$

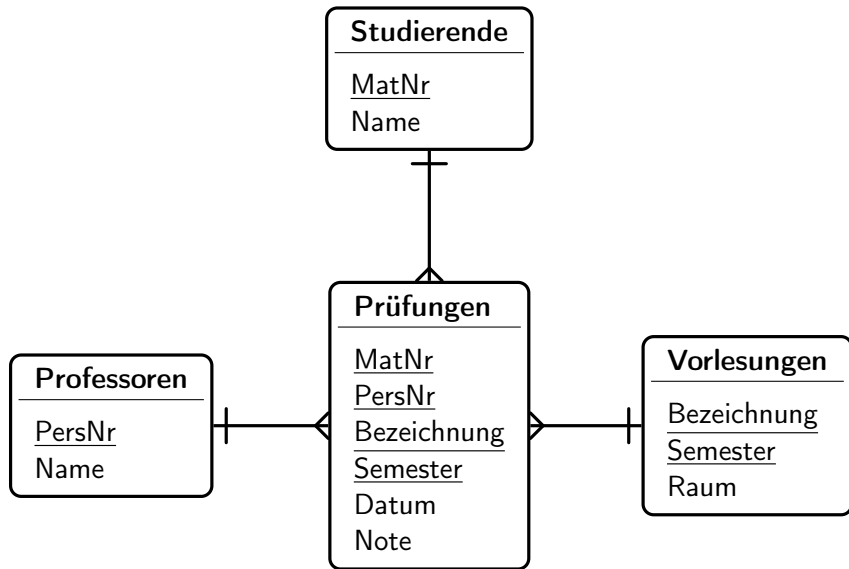
$\mathcal{S}_{\text{Professoren}} := (\underline{\text{PersNr}}, \text{Name})$

$\mathcal{S}_{\text{Vorlesungen}} := (\underline{\text{Bezeichnung}}, \underline{\text{Semester}}, \text{Raum})$

$\mathcal{S}_{\text{Prüfungen}} := (\underline{\text{MatNr}}, \underline{\text{PersNr}}, \underline{\text{Bezeichnung}}, \underline{\text{Semester}}, \text{Datum}, \text{Note})$

- 1 Beziehungen können zwischen mehr als zwei Konzepten bestehen.
- 2 Wenn ein Primärschlüssel aus mehreren Attributen besteht, so muss der ganze Primärschlüssel in der Beziehungstabelle vorkommen. Im Beispiel enthält $\mathcal{S}_{\text{Prüfungen}}$ die Attribute Bezeichnung und Semester, um eine Vorlesung zu identifizieren.
- 3 Beziehungen können zusätzliche Attribute haben.

Diagramm



Frage

Wie können wir 1:1-Beziehungen, wie zum Beispiel Ehepartner abbilden?



1:1-Beziehungen, z.B. Bundesräte und Departemente

$$\mathcal{S}_{\text{Regierung}} := (\underline{\text{BrId}}, \text{DepId})$$

$$\mathcal{S}_{\text{Departemente}} := (\underline{\text{DepId}}, \text{BrId}) .$$

wobei DepId ein Fremdschlüssel auf $\mathcal{S}_{\text{Departemente}}$ und BrId ein Fremdschlüssel auf $\mathcal{S}_{\text{Regierung}}$ ist.

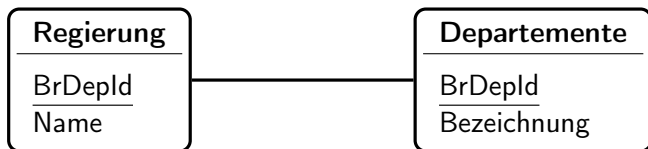
Regierung	
BrId	DepId
A	Y
B	Y
C	Z

Departemente	
DepId	BrId
Y	C
Z	A

Korrekte Modellierung

$$\mathcal{S}_{\text{Regierung}} := (\underline{\text{BrDepId}}, \text{Name})$$
$$\mathcal{S}_{\text{Departemente}} := (\underline{\text{BrDepId}}, \text{Bezeichnung}) .$$

Die graphische Darstellung dieses DB-Schemas ist nun



Instanz davon

Regierung

BrDepId	Name
1	Berset
2	Maurer

Departemente

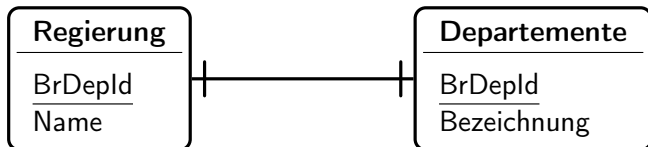
BrDepId	Bezeichnung
1	EDI
3	EJPD

Berset steht also dem EDI vor. Für Mauer gibt es jedoch keinen Eintrag in der Departemente-Tabelle, und auch wer das EJPD führt, ist in dieser Instanz nicht ersichtlich.

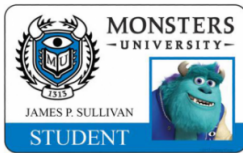
Um diese Situation zu verhindern, können wir im Diagramm noch Existenzbedingungen hinzufügen.

Mit Existenzbedingungen

- 1 Der Primärschlüssel BrDepId im Schema Regierung ist gleichzeitig ein Fremdschlüssel auf $\mathcal{S}_{\text{Departemente}}$.
- 2 Der Primärschlüssel BrDepId in Akten ist gleichzeitig ein Fremdschlüssel auf $\mathcal{S}_{\text{Angestellte}}$.



Frage



Wie könnten wir folgende Situation modellieren?

- 1 Personen haben einen Namen.
- 2 Studierende sind Personen, die zusätzlich eine Matrikelnummer haben.
- 3 Angestellt sind Personen, die zusätzlich ein Büro haben.



Vererbung

Schema für Personen

$$\mathcal{S}_{\text{Personen}} := (\underline{\text{PersId}}, \text{Name}) .$$

Spezialisierung zu *Angestellte* und *Studierende*

$$\mathcal{S}_{\text{Studierende}} := (\underline{\text{PersId}}, \text{MatNr})$$

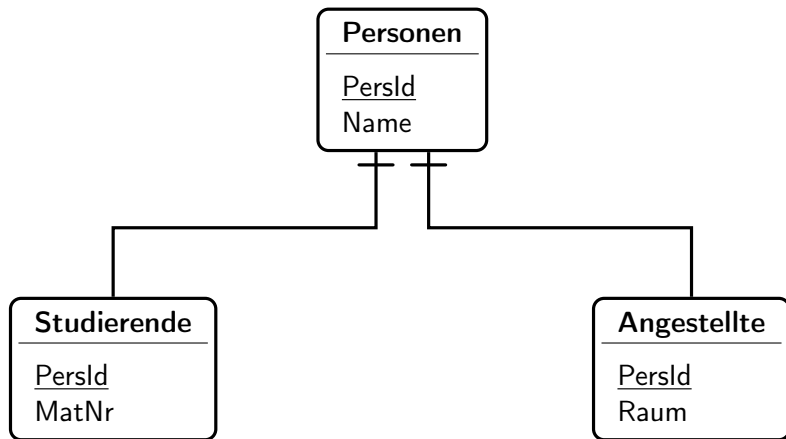
$$\mathcal{S}_{\text{Angestellte}} := (\underline{\text{PersId}}, \text{Raum}) .$$

In beiden Schemata ist der Primärschlüssel `PersId` gleichzeitig Fremdschlüssel auf $\mathcal{S}_{\text{Personen}}$.

Die Relation `Person` heisst *Basisrelation*.

Die Relationen `Studierende` und `Angestellte` heissen *abgeleitete Relationen*.

Vererbung: Diagramm



Bedingungen

Totalität Die Vererbungsrelation ist *total*, wenn es für jedes Tupel in der Basisrelation *mindestens* ein entsprechendes Tupel in einer der abgeleiteten Relationen gibt.

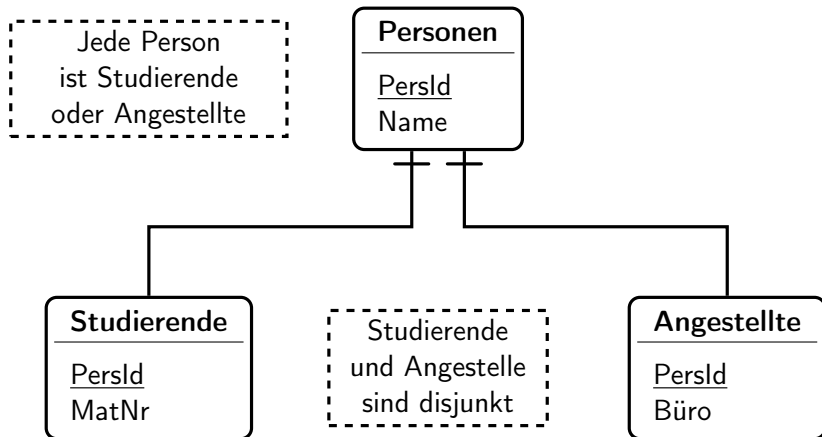
Disjunktheit Die abgeleiteten Relationen heissen *disjunkt*, wenn es für jedes Tupel in der Basisrelation *höchstens* ein entsprechendes Tupel in einer der abgeleiteten Relationen gibt.

Bedingungen: Beispiel

Totalität Es gibt keine Entitäten die nur Personen sind. Jede Person muss Studierender oder Angestellter sein.

Disjunktheit Es gibt keine Personen, die sowohl Studierende als auch Angestellte sind.

Diagramm



Vermeidung von Null

Personen mit zusätzlichem Attribut zur Korrektur der Brille:

$$\mathcal{S}_{\text{Personen}} := (\underline{\text{PersId}}, \text{Name}, \text{GebDatum}, \text{Brille}) .$$

Eine mögliche Instanz dieses Schemas ist:

Personen			
PersId	Name	GebDatum	Brille
1	Eva	19710429	Null
2	Tom	19720404	Null
3	Eva	19680101	-3.5
4	Ann	19841214	Null
5	Bob	20140203	Null

Schema aufteilen

$\mathcal{S}_{\text{AllePersonen}} := (\underline{\text{PersId}}, \text{Name}, \text{GebDatum})$

$\mathcal{S}_{\text{Brillenträger}} := (\underline{\text{PersId}}, \text{Brille})$,

wobei PersId in $\mathcal{S}_{\text{Brillenträger}}$ ein Fremdschlüssel auf $\mathcal{S}_{\text{AllePersonen}}$ ist. Die obige Relation wird somit aufgeteilt in:

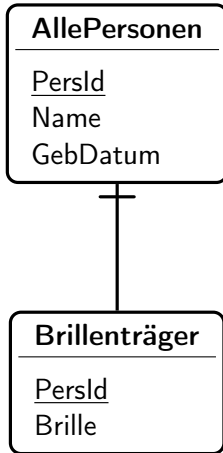
AllePersonen

<u>PersId</u>	Name	GebDatum
1	Eva	19710429
2	Tom	19720404
3	Eva	19680101
4	Ann	19841214
5	Bob	20140203

Brillenträger

<u>PersId</u>	Brille
3	-3.5

Diagramm



Zwei Bedeutungen von Null

In der Tabelle `Personen` können wir nicht unterscheiden ob,

- ① Eva keine Brille hat oder
- ② die Korrektur von Evas Brille unbekannt ist.

In beiden Fällen lautet der Eintrag in `Personen`

(1, Eva, 19710429, Null) .

Zwei Bedeutungen von Null

Wenn wir Brillenträger von AllePersonen ableiten, so können wir folgende Fälle unterscheiden:

- ① Eva hat keine Brille. Dann gibt es in der Tabelle Brillenträger keinen Eintrag mit PersId 1.
- ② Die Korrektur von Evas Brille ist unbekannt. Dann gibt es in Brillenträger einen Eintrag

(1, Null) .

Hochschuldatenbank: Anforderungen, Teil 1

In einer Hochschuldatenbank sind Daten über folgende Einzelheiten abzulegen:

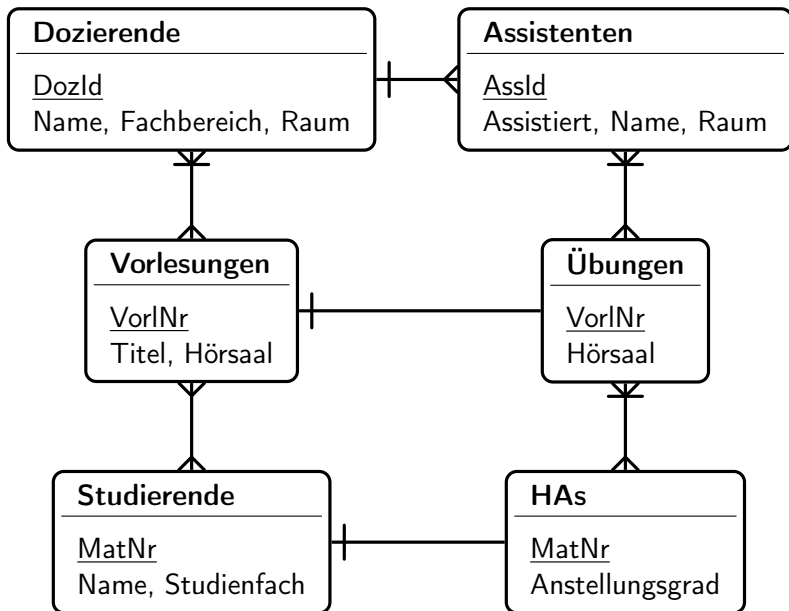
- 1 Dozierende (Name, Fachbereich, Raum),
- 2 Assistenten (Name, Raum),
- 3 Vorlesungen (Nummer, Titel, Hörsaal),
- 4 Übungen zu Vorlesungen (Nummer, Hörsaal),
- 5 Studierende (Matrikel-Nummer, Name, Studienfach),
- 6 Hilfsassistenten (Matrikel-Nummer, Name, Anstellungsgrad).

Hochschuldatenbank: Anforderungen, Teil 2

Ferner ist zu beachten:

- 1 Dozierende haben Assistenten,
- 2 Dozierende halten Vorlesungen,
- 3 Assistenten betreuen Übungen,
- 4 Übungen finden nur in Verbindung mit einer Vorlesung statt,
- 5 einer Übung können mehrere Hilfsassistenten zugeordnet werden (zur Korrektur von Übungsserien)
- 6 ein Hilfsassistent ist insbesondere ein Studierender,
- 7 ein Student hört Vorlesungen.

Diagramm (verkürzt)



Überlegungen

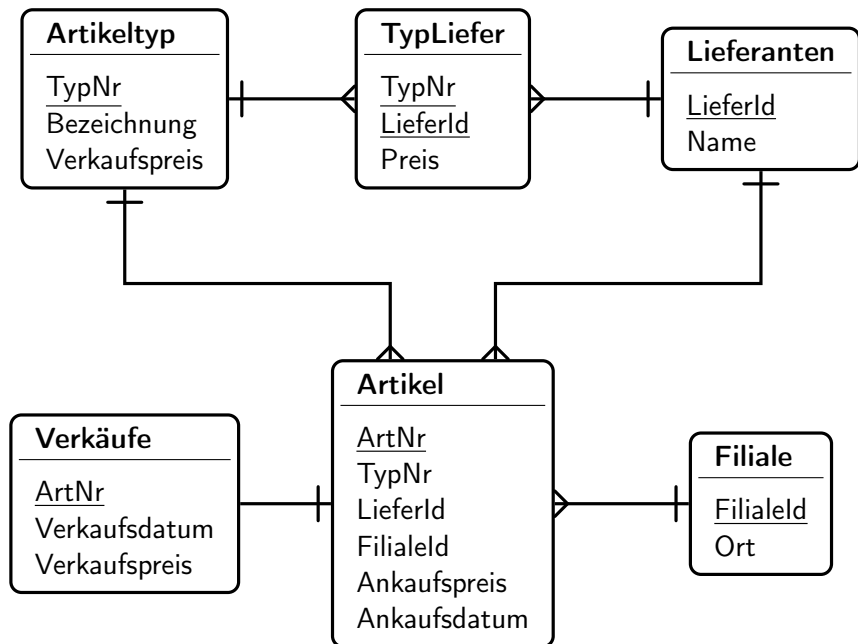
- 1 DozId (in Dozierende) und AssId (in Assistenten) sind zusätzliche Attribute welche jeweils als Primärschlüssel dienen.
- 2 Ein Assistent ist bei *genau* einem Dozierenden angestellt. Um dies zu modellieren, ist in der Tabelle Assistenten das Attribut Assistiert ein Fremdschlüssel auf Dozierende.
- 3 Zu einer Vorlesung gibt es nur eine Übung und Übungen gibt es nicht ohne entsprechende Vorlesung.
- 4 HAs ist eine Spezialisierung von Studierende. Entsprechend wird das Attribut Name von Studierende geerbt und muss nicht extra in HAs erfasst werden.
- 5 Ein Hilfsassistent kann mehrere (muss jedoch mindestens eine) Übungen betreuen.
- 6 Ein Assistent kann mehrere (ev. auch keine) Übungen betreuen. Jede Übung hat mindestens einen Assistenten.
- 7 Das Schema garantiert nicht, dass ein Assistent, welche die Übungen zu einer Vorlesung betreut, auch bei einem Dozierenden dieser Vorlesung angestellt ist.

Warenhauskette: Anforderungen

Für eine Datenbank für die Lagerverwaltung einer Warenhauskette haben wir folgende Anforderungen:

- 1 Für jede Filiale soll das Sortiment ersichtlich sein.
- 2 Für jeden Lieferanten soll ersichtlich sein, welchen Artikel er zu welchem Preis liefern kann.
- 3 Für jeden Artikel soll die Bezeichnung gespeichert werden.
- 4 Für jeden verkauften Artikel soll der Verkaufspreis und das Verkaufsdatum gespeichert werden
- 5 Für jeden gelieferten Artikel soll das Lieferdatum gespeichert werden.

Diagramm



Überlegungen zum Schema

- ① Wir haben allen Tabellen Ids hinzugefügt als Primärschlüssel.
- ② Ein Artikel ist ein bestimmter Gegenstand, der an Lager sein kann oder bereits verkauft wurde. Falls z.B. die Warenhauskette Computer verkauft, so sind die einzelnen Computer Artikel. Ein bestimmtes ComputermodeLL ist dann ein Artikeltyp.
- ③ Ein Artikel der ins Lager kommt, wird in der Artikel-Tabelle eingetragen. Wird er dann verkauft, so wird er auch noch in die Tabelle Verkäufe eingetragen (bleibt aber in Artikel).
- ④ Preis ist ein Attribut von TypLiefer, das angibt, welcher Lieferant diesen Artikeltyp gegenwärtig zu welchem Preis liefert.

Überlegungen zum Schema 2

- 1 Das Attribut Verkaufspreis in Artikeltyp gibt an, was der aktuelle Verkaufspreis für Artikel dieses Typs ist. So ist gewährleistet, dass alle gleichen Artikel auch gleich viel kosten. Bei Preisanpassungen (Aktionen) ist nur ein Update nötig und es muss nicht der Preis bei allen Artikeln einzeln angepasst werden.
- 2 Das Attribut Ankaufspreis in Artikel gibt an, zu welchem Preis dieser Artikel tatsächlich eingekauft wurde. Der aktuelle Preis, der in der Tabelle TypLiefer abgespeichert ist, kann davon abweichen.
- 3 Das Attribut Verkaufspreis in Verkäufe gibt an, zu welchem Preis dieser Artikel tatsächlich verkauft wurde. Der aktuelle Wert von Verkaufspreis in Artikeltyp kann davon abweichen.

Überlegungen zum Schema 3

Zur Vermeidung von Null-Werten haben wir das Schema in die Tabellen Artikel und Verkäufe aufgeteilt. So gibt es keine Null-Werte in den Attributen Verkaufsdatum und Verkaufspreis.

In unserem DB-Schema ist es sehr aufwändig folgende Abfrage zu bearbeiten:

Welches ist der aktuelle Lagerbestand eines gegebenen Artikeltyps? (1)

Dazu müssen wir nämlich diejenigen Artikel des gegebenen Typs finden, welche *nicht* in der Verkäufe-Tabelle eingetragen sind.

Variante

Ohne die Aufteilung zur Vermeidung von Null-Werten wird die Tabelle Verkäufe nicht mehr benötigt und sowohl Verkaufspreis als auch Verkaufsdatum sind Attribute von Artikel.

Diese Attribute haben den Wert Null, falls der Artikel noch nicht verkauft wurde. Die Abfrage (1) wird nun bearbeiten indem alle Artikel des Typs gesucht werden, deren Verkaufsdatum den Wert Null hat.

Es ergibt sich nun ein neues Problem. Um die Abfrage

Finde alle verkauften Artikel eines gegebenen Typs (2)

zu beantworten, müssen alle Artikel des gegebenen Typs gefunden werden, deren Verkaufsdatum nicht Null ist.

Es kann vorkommen, dass Abfragen, die auf nicht Null testen, nicht effizient durchführbar sind.

Variante 2

Um eine gute Performance zu erreichen, fügen wir der Artikel-Tabelle ein neues Attribut Status hinzu, welches zwei Werte annehmen kann:

- 1 *an Lager* und
- 2 *verkauft*.

Die Abfrage (2) kann nun ganz einfach ausgeführt werden, indem auf Status *verkauft* getestet wird.

In der endgültigen Version braucht es also die Verkäufe-Tabelle nicht mehr und die Tabelle Artikel hat die Attribute

$$\mathcal{S}_{\text{Artikel}} := (\underline{\text{ArtNr}}, \text{TypNr}, \text{LieferId}, \text{FilialeId}, \text{Ankaufspreis}, \\ \text{Ankaufsdatum}, \text{Verkaufspreis}, \text{Verkaufsdatum}, \\ \text{Status}) .$$