University of Applied Sciences and Arts Dortmund

Softwaretechnik 2

Persistente Datenhaltung

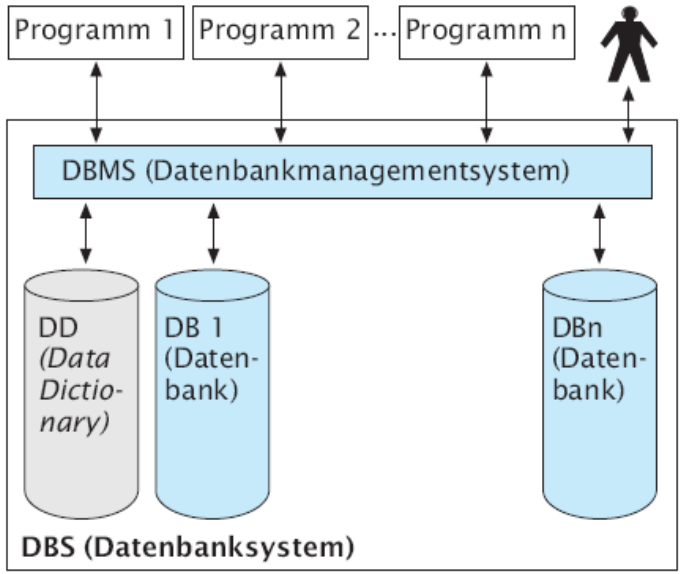
Fachhochschule
Dortmund
University of Applied Sciences and Arts

Struktur

Datenbanksysteme

Wann wird ein Datenbanksystem benötigt?

- Verschiedene Programme benötigen gemeinsame Daten
- Eigene Datenhaltung für jedes Programm bedeutet
  - Mehrfacherfassungen
  - Redundante Speicherung
  - Gefahr inkonsistenter Datenbestände
- Definition: Ein **Datenbanksystem (DBS)** besteht aus
  - einer oder mehreren Datenbanken,
    → speichern Daten
  - einem Data Dictionary (DD) und
    → enthält das Datenmodell
  - einem Datenbankmanagementsystem (DBMS).
    → verwaltet und kontrolliert die Datenhaltung



Datenbanksysteme

Eigenschaften

- Persistente Speicherung der Daten
- Zuverlässige Verwaltung der Daten
- Unabhängige Verwaltung der Daten (unabhängig von einzelnen Anwendungen)
- Komfortable Verwendung der Daten (Benutzer verwenden höhere abstrakte Schnittstelle)
- Flexibler Zugang zu den Daten (Ad hoc Zugriff durch Abfragesprachen oder andere Hilfsmittel)
- Datenschutz (Daten können vor unberechtigtem Zugriff geschützt werden)
- Verwaltung großer Datenbestände (Daten nicht vollständig im Speicher)
- Integrierte Datenbank (alle Daten werden redundanzarm gespeichert, auch wenn sie von verschiedenen Anwendungen verwendet werden)
- Mehrfachbenutzung der Datenbank (parallele Zugriffe mehrerer Benutzer und/oder Anwendungen möglich)

Datenbanksysteme

Datenmodell

Jedem DBS liegt ein Datenmodell (Konzept) zugrunde, das festlegt

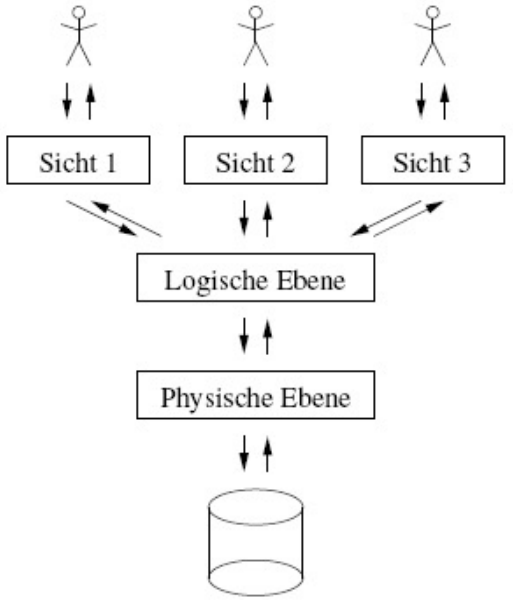
- welche Eigenschaften die Datenelemente besitzen
- welche Struktur die Datenelemente besitzen dürfen (z.B. Relationen, Tupel …)
- welche Konsistenzbedingungen einzuhalten sind (Integritätsbedingungen zur weiteren Einschränkung der Struktur)
- welche Operationen zum Speichern, Suchen, Ändern und Löschen von Datenelementen existieren → CRUD

- Bekannte Datenmodelle:
  - Hierarchisches Datenbankmodell
  - Netzwerkdatenbankmodell
  - Relationales Datenbankmodell (→ RDBS)
  - Objektorientiertes Datenbankmodell (→ ODBS)
  - NoSQL Datenbankmodell (→ Graph, Dokumentenorientiert, Zeitreihe ...)

Datenbanksysteme

Sicht und Ebenen

- **Sicht**
  beschreibt wie ein Benutzer die Daten sieht
- **Logische Ebene**
  beschreibt wie die Daten strukturiert sind
- **Physische Ebene**
  beschreibt wie die Daten gespeichert werden



Struktur

Relationale Datenbanksysteme

- Relationen
  - Relationale Datenbanken speichern Daten in Form von Tabellen
  - Jede Zeile der Tabelle (Tupel) repräsentiert ein Objekt der Klasse
  - Alle Tupel einer Tabelle müssen gleich lang sein
  - Die Reihenfolge der Attribute spielt keine Rolle
- Primärschlüssel
  - Identifizierung eines Tupels durch einen eindeutigen Schlüssel
- Fremdschlüssel
  - Realisierung von Assoziationen durch Schlüssel-Fremdschlüssel-Beziehungen
- Fundamentale Integritätsregeln
  - Entitäts-Integrität: Schlüsselattribute müssen immer einen Wert besitzen
  - Referentielle Integrität: Wenn in einer Tabelle ein Fremdschlüssel vorhanden ist, dann muss der Fremdschlüsselwert auch als Primärschlüsselwert in der entsprechenden Tabelle vorkommen
- ACID (Atomicity, Consistency, Isolation und Durability) Eigenschaften für Transaktionen

Relationale Datenbanksysteme

Sprachen

- Abfragesprache (Data Manipulation Language, DML)
  - dient zur Verwaltung der Daten
  - kann das logische Schema nicht verändern sondern nur lesen
- Datendefinitionssprache (Data Definition Language, DDL)
  - dient zum Anlegen, Löschen und Ändern von Datenbanken und ihren –strukturen (wie z.B. Tabellen, Views, Index)
- Gegenwärtig ist SQL (Structured Query Language) sowohl als DDL als auch als DML Standard
- JDBC (Java Database Connectivity) bietet auf Basis von SQL eine einheitliche und herstellerunabhängige Schnittstelle zwischen Java-Anwendungen und relationalen Datenbanken

Relationale Datenbanksysteme

SQL - Historie

- Sprache der 4. Generation (Deklarative Programmiersprache) d.h. keine Schleifen, keine Prozeduren, keine Rekursion, keine ausreichenden mathematischen Operationen
- In den 1970ern in den Forschungslaboratorien von IBM im Zusammenhang mit der Prototypentwicklung des relationalen Datenbanksystems SYSTEM R entworfen auf Grundlage der Arbeiten von E.F. Codd
- 1979: Oracle bringt die erste SQL-Datenbank auf den Markt
- 1989: ANSI veröffentlicht erste SQL-Spezifikation
- 1992/1992: SQL-92 bzw. SQL-2 / SQL-99 bzw. SQL-3
- 2003-2016: SQL-2003, SQL/XML-2006, SQL:2008, SQL:2011, SQL:2016
- 2019: SQL/MDA:2019

Fachhochschule Dortmund
University of Applied Sciences and Arts

Relationale Datenbanksysteme

SQL – DDL & DML Beispiele

we focus on students

- DDL:

```
create table Lieferant
      (nummer integer not null,
       firma char(30));

create table Artikel
      (nummer integer not null,
       bezeichnung char(30) not null,
       preis decimal (8,2),
       lieferantNummer integer not null);
```

Tabellenname

Muss-Attribut

Attributname

Muss-Assoziation

- DML:

```
insert into Lieferant values
      (0815, 'SchreibMitMüller');

update Artikel
      set preis = 4.95
      where nummer = 4711;
```

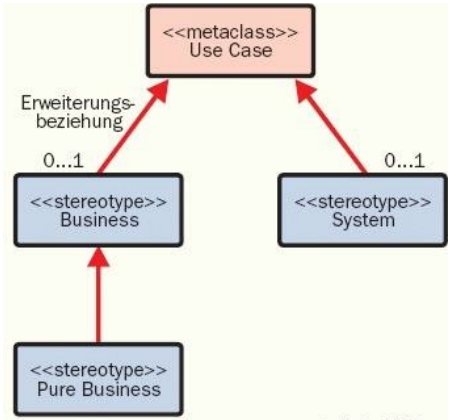
Lieferant	
Nummer	**Firma**
0815	SchreibMitMüller
1123	DruckertinteMustermann

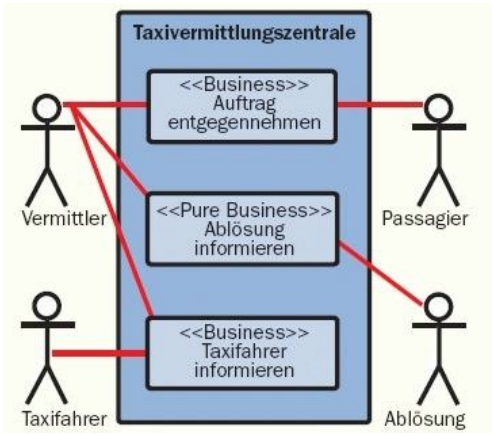
Artikel			
Nummer	**Bezeichnung**	**Preis**	**LieferantNummer**
1	Kugelschreiber	4.95	0815
2	Canon X3222 Tinte	28	1123
3	Bleistift	2.30	0815

Relationale Datenbanksysteme

UML-Notation für relationale Datenbankmodelle

- UML ist keine Sprache sondern eine Sprachfamilie, die spezifisch angepasst werden kann
- UML-Profile (profiles) ermöglichen die Erweiterung der vorhandenen UML-Notation
  Beispiel:





Relationale Datenbanksysteme

UML-Notation für relationale Datenbankmodelle

Profile zur Modellierung relationaler Datenbankmodelle

- 2005 wurde auch von der OMG ein UML-Profil für die Modellierung von relationalen Datenmodellen verwendet
  [Document -- ab/05-12-02 (Information Management Metamodel (IMM) RFP) http://www.omg.org/cgi-bin/doc?ab/05-12-02]

- Im Folgenden wird das UML-Profil von Scott Ambler für die Modellierung von relationalen Datenmodellen verwendet
  [vgl. Scott W. Ambler: Agile Database Techniques : Effective Strategies for the Agile Software Developer, John Wiley & Sons, 2003, ISBN 0-471-20283-5]

Relationale Datenbanksysteme

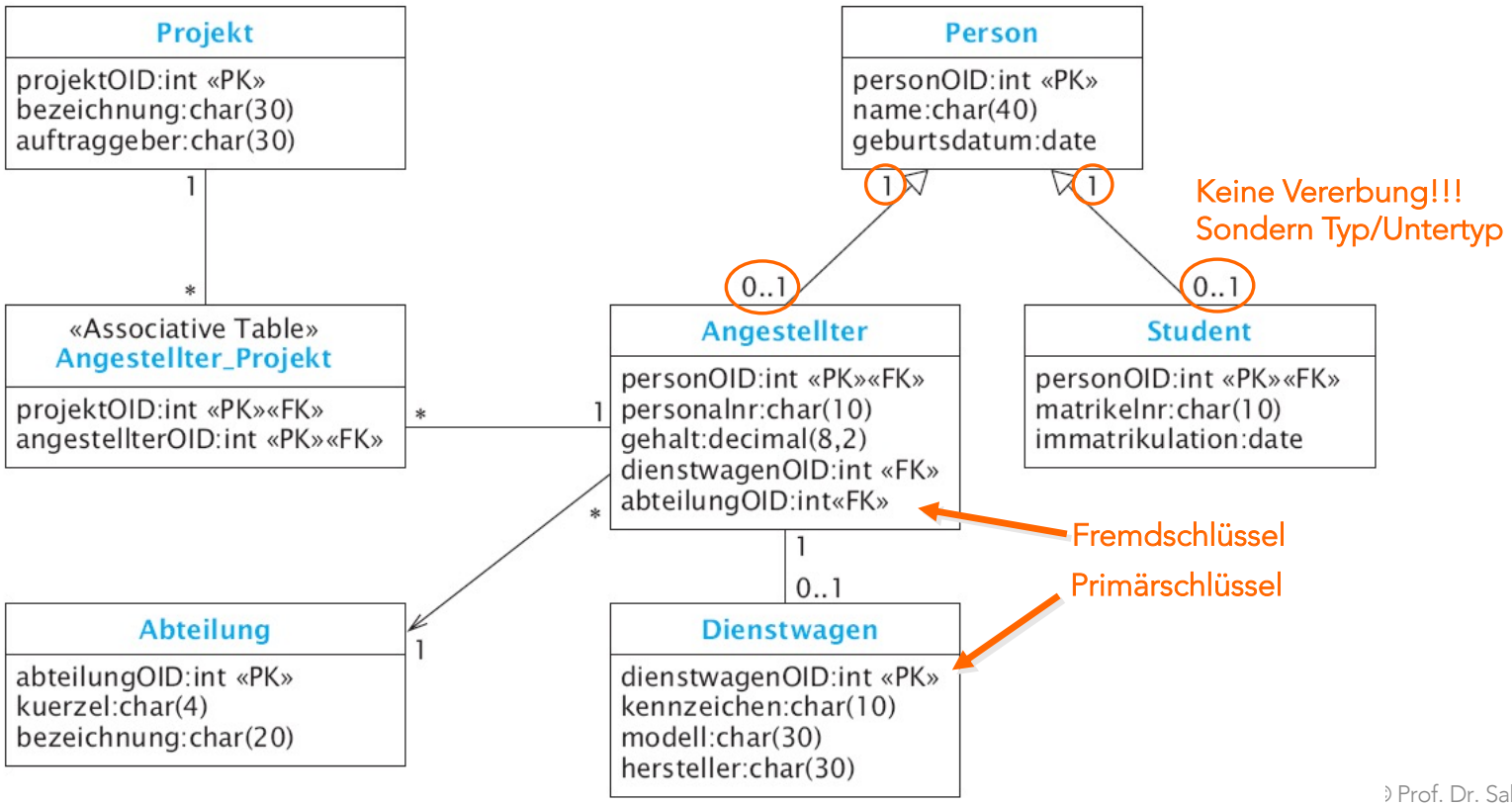
UML-Notation für relationale Datenbankmodelle

- **Tabellen** und **Views**
  - werden mithilfe der **Klassennotation** modelliert
  - **Stereotypen** spezifizieren die verschiedenen Arten

- **Stereotypen**, die verwendet werden
  - «Associative Table»: kennzeichnet Assoziationstabellen
  - «Index»: modelliert einen Index, der immer durch eine Abhängigkeits-beziehung zu der Tabelle ergänzt werden muss
  - «Stored Procedures»: kennzeichnet Tabellen, die nur Operationssignaturen für *Stored Procedures* der Datenbank enthalten
  - «View»: kennzeichnet einen View

Relationale Datenbanksysteme

UML-Notation für relationale Datenbankmodelle



Struktur

Objektrelationale Abbildung

- Heute werden oft objektorientierte Techniken und Programmiersprachen eingesetzt, während Tabellen in einer relationalen Datenbank gespeichert werden
  ⇒ Brückenbildung notwendig

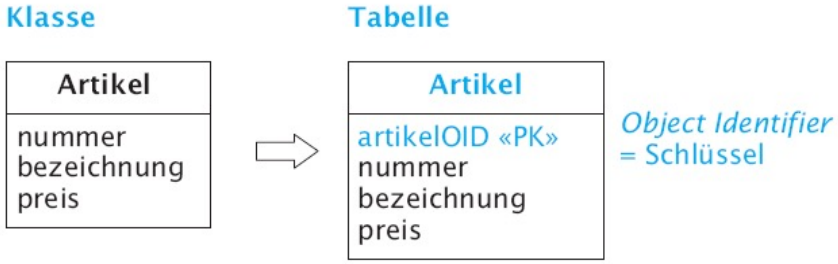
- Einfachster Fall:
  Abbildung eines Attributs einer Klasse auf ein Attribut einer Tabelle

- Feste Abbildungsvorschriften für alle komplexeren Fälle:
  - Abbilden einer Klasse
  - OID-Attribut
  - Abbilden einer Generalisierungshierachie (3 Alternativen)
  - Abbilden einer Assoziation (1:1, 1:m, m:m)
    (Assoziationsklasse, Komposition/Aggregation, geordnete Assoziation, reflexive Assoziation)
  - Abbilden von Klassenattributen (3 Alternativen)
  - Abbilden von Datentypen (neuere SQL-Versionen)

Objektrelationale Abbildung

Abbilden einer Klasse

- Einfachster Fall: Klasse kann auf eine einzige Tabelle abgebildet werden



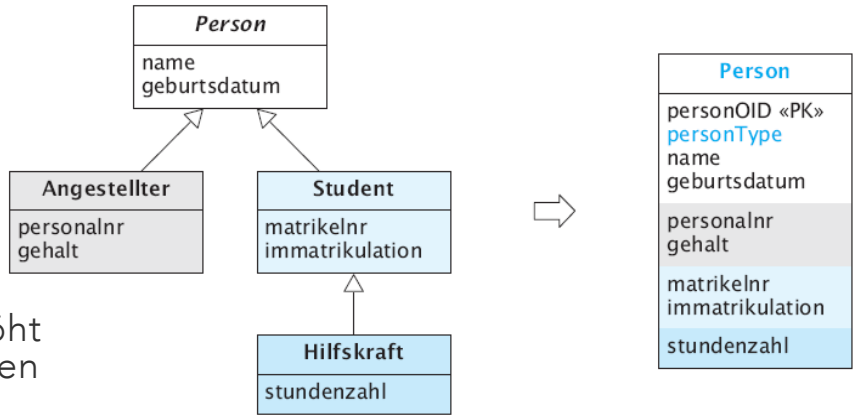
- OID-Attribut
  - Erweiterung jeder Tabelle um ein eindeutiges OID-Attribut, das die Rolle des Schlüsselattributs spielt, der Wert kann sich nicht ändern
  - OID-Attribute dürfen keinesfalls eine semantische Bedeutung besitzen
  - Realisierungen:
    - Zähler, der für jedes neue Tupel erhöht wird
    - Durch UUIDs *(universally unique identifiers)* generiert (128-Bit-Wert)
    - High/Low-Strategie (Kombination: Wert aus OID-Tabelle / Zähler in Anwendung)

Objektrelationale Abbildung

Generalisierungshierarchie 1/3

Abbilden der gesamten Hierarchie auf eine einzige Tabelle

- Vorteile
  - Einfach
  - Schneller Datenzugriff
  - Ad-hoc-Anfragen einfach zu realisieren
  - Neue Klassen können problemlos hinzugefügt werden
- Nachteile
  - Kopplung innerhalb der Generalisierungshierarchie wird erhöht
    Erweiterung um 1 Attribut in einer Klasse → alle Tupel anfassen
  - Speicherplatzverschwendung
    ⇒ viele null Werte



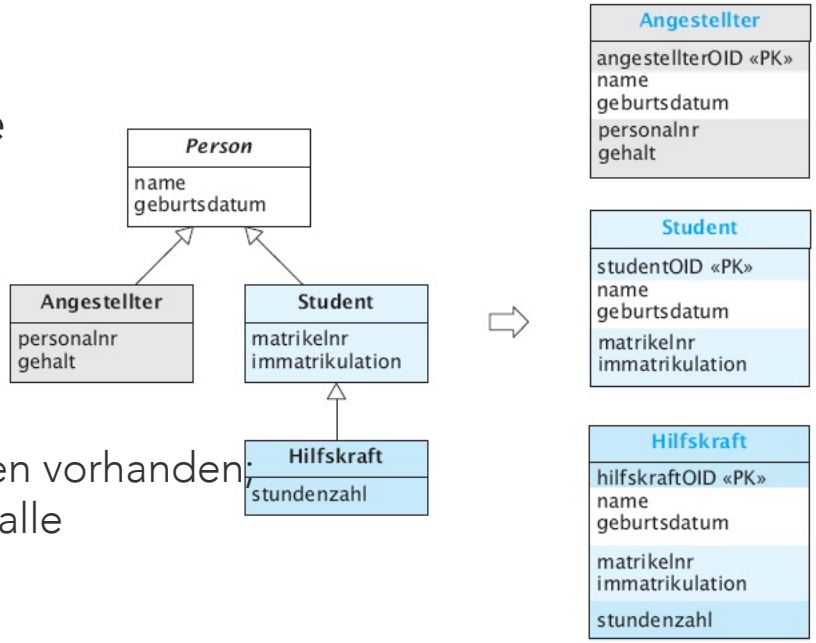
⇒ Am besten für kleine Generalisierungshierarchien geringer Tiefe geeignet

Objektrelationale Abbildung

Generalisierungshierarchie 2/3

Abbilden jeder konkreten Klasse auf eine Tabelle

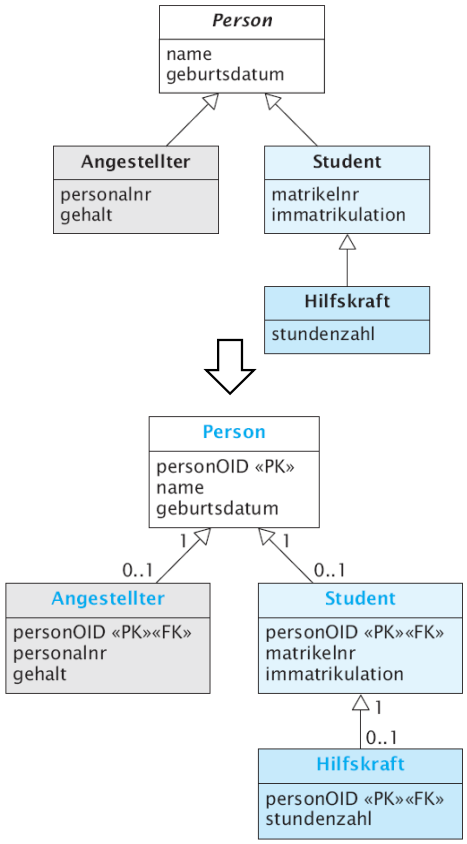
- Vorteile
  - Ad-hoc-Anfragen einfach zu formulieren
  - Effizienter Datenzugriff
- Nachteile
  - Attribute der abstrakten Oberklasse in mehreren Tabellen vorhanden; wenn entsprechende Attribute modifiziert werden, sind alle betroffenen Tabellen zu aktualisieren



⇒ Am besten für Generalisierungshierarchien geeignet, bei denen die Klassen nur wenige gemeinsame Attribute besitzen

Objektrelationale Abbildung

Generalisierungshierarchie 3/3



Abbilden jeder Klasse auf eine Tabelle

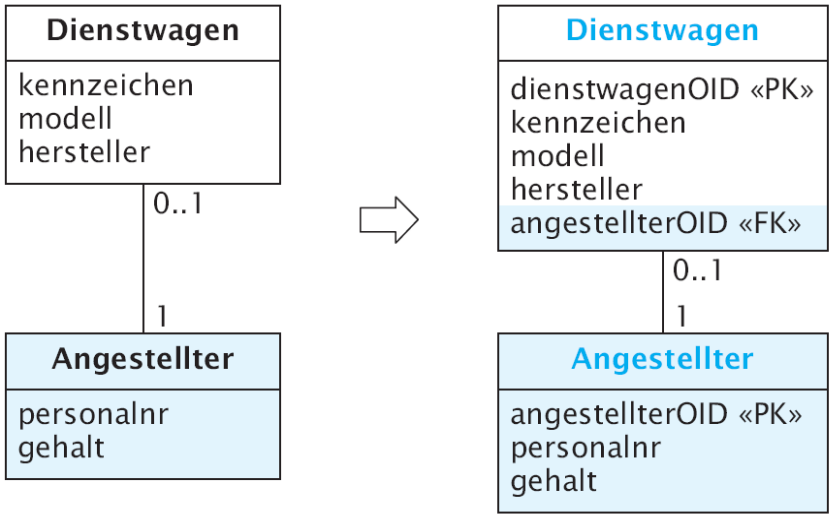
- Vorteile
  - Entspricht dem objektorientierten Konzept am besten
  - Änderungen in der Oberklasse mit minimalem Aufwand durchführbar
  - Neue Attribute können einfach ergänzt werden
- Nachteile
  - Viele Tabellen in der Datenbank
  - Zugriffe dauern länger
  - Ad-hoc-Anfragen schwieriger zu formulieren, sofern keine Views aufgebaut werden
  - Geringere Performance durch zahlreiche Joins

⇒ Am besten für Generalisierungshierarchien geeignet, bei denen die Klassen viele gemeinsame Attribute besitzen und Änderungen mit hoher Wahrscheinlichkeit zu erwar

Objektrelationale Abbildung

Abbilden einer Assoziation (1:1)

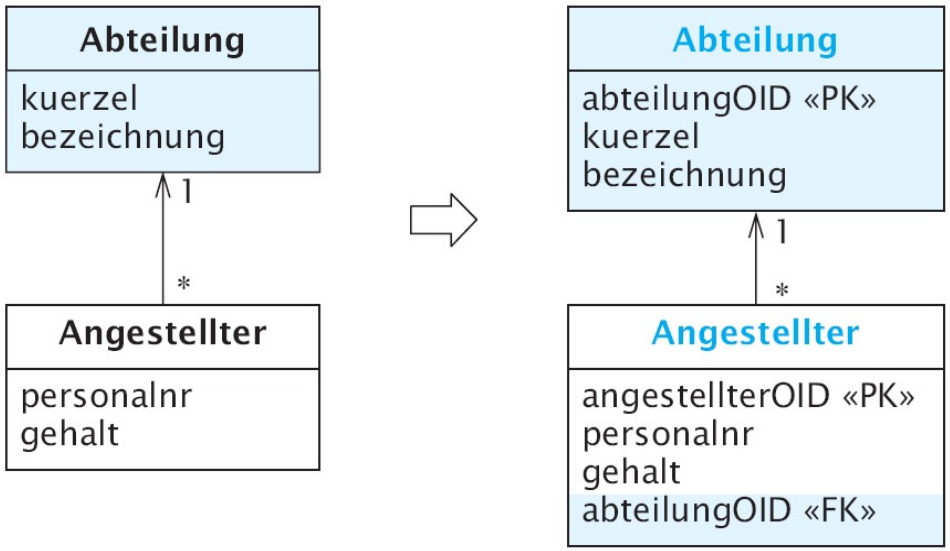
- Fremdschlüssel kann in eine der beiden Tabellen eingetragen werden



Objektrelationale Abbildung

Abbilden einer Assoziation (1:m)

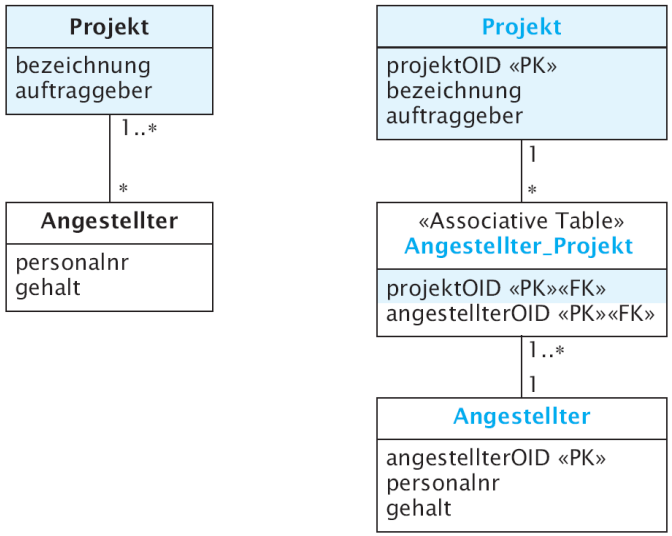
- Fremdschlüssel in Tabelle mit der many-Multiplizität einfügen
- Alternative: Realisierung durch eine Tabelle wie bei der m:m-Assoziation



Objektrelationale Abbildung

Abbilden einer Assoziation (m:m)

- Immer auf separate Tabelle abbilden, Assoziationstabelle enthält die beiden Fremdschlüssel
- Eigene OID der Assoziationstabelle vorteilhaft, weil dann alle Tabellen gleichbehandelt werden

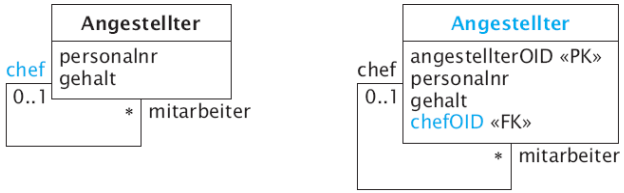


Objektrelationale Abbildung

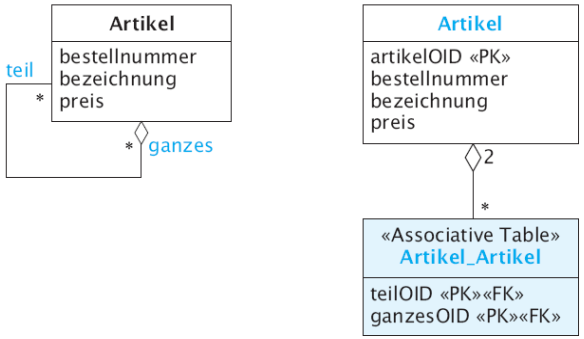
Abbilden einer Assoziation

- Komposition und Aggregation werden wie eine einfache Assoziation behandelt
- Reflexive Assoziation:

1:m Assoziation

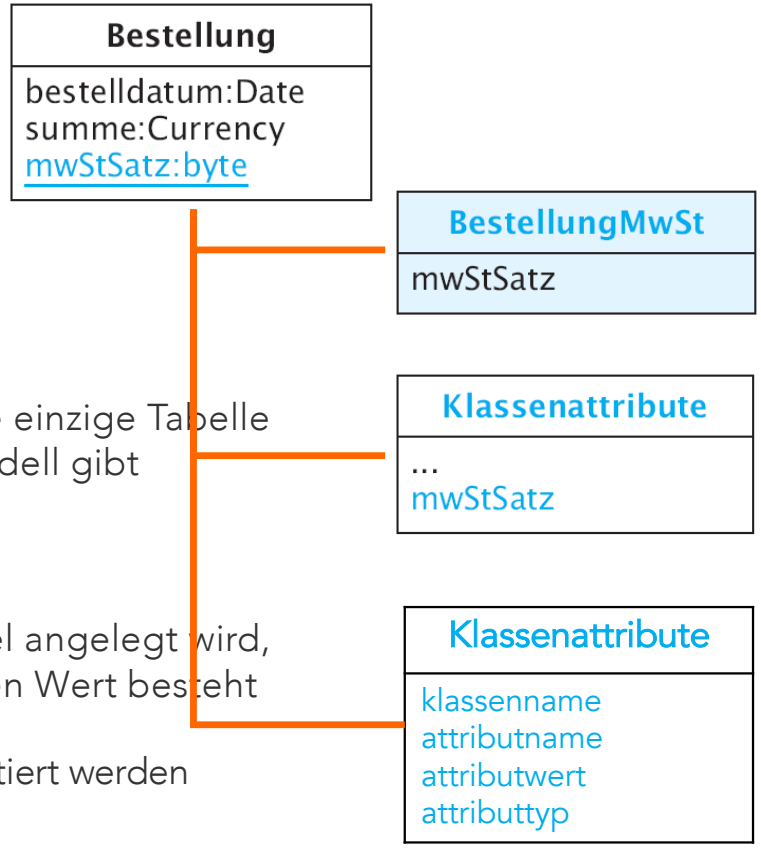


m:m Assoziation
⇒ separate Assoziationstabelle



Objektrelationale Abbildung

Klassenattribute



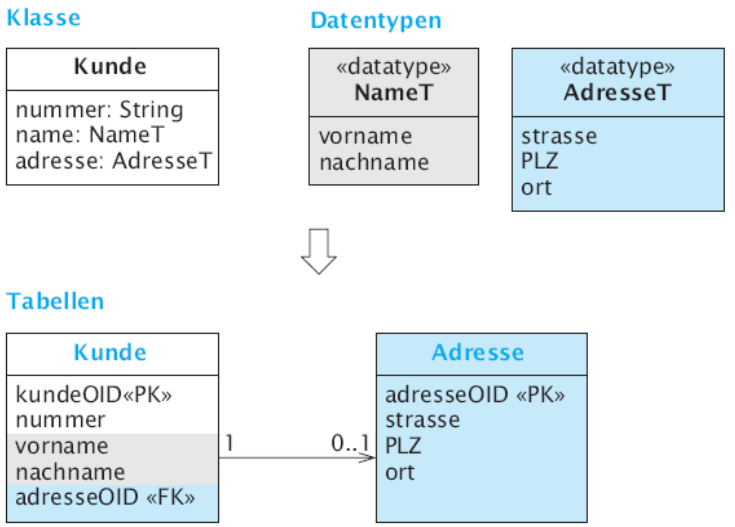
- Erste Alternative: Jedes Klassenattribut auf eine Tabelle mit einer Spalte und einem Tupel abbilden
  - Vorteil: einfacher, schneller Zugriff
  - Nachteil: viele Tabellen
- Zweite Alternative: alle Klassenattribute der Anwendung werden in eine einzige Tabelle eingetragen, die so viele Spalten besitzt, wie es Klassenattribute im Modell gibt
  - Vorteil: nur eine einzige weitere Tabelle
  - Nachteil: potentielle Gefahr für Nebenläufigkeitsprobleme
- Dritte Alternative: eine Tabelle, wobei für jedes Klassenattribut ein Tupel angelegt wird, das aus dem Klassenamen, dem Namen des Klassenattributs und dessen Wert besteht
  - Vorteil: keine Nebenläufigkeitsprobleme
  - Nachteil: Werte werden als String gespeichert und müssen ggf. konvertiert werden

Objektrelationale Abbildung

Komplexer Datentyp

- Optimaler Fall:
  Datentypen der UML können auf benutzerdefinierte Datentypen oder Collection-Typen der Datenbank abgebildet werde.

- Alternative
  - Integration des Datentyps NameT in Tabelle Kunde
  - Darstellung des Datentyps AdressetT in eigener Tabelle



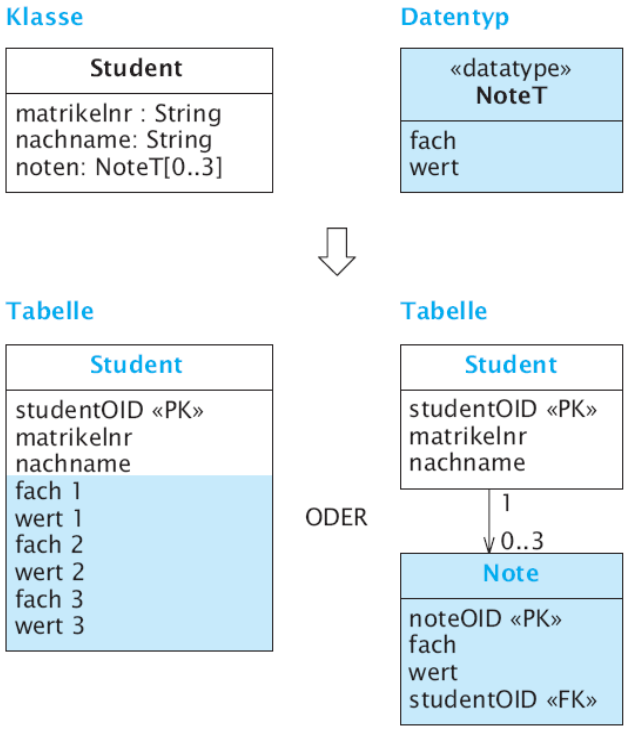
Objektrelationale Abbildung

Datentypen

Listenattribute

- Feste Obergrenze bekannt:
  Attribute können in die Tabelle der Klasse eingetragen werden

- Obergrenze variabel oder besitzen meist nur wenige Elemente der Tabelle Werte:
  Liste auf eigene Tabelle abbilden



Struktur

Weitere Fragen