Datenbanken und Informationssysteme

VL 05, Objektorientiertes Datenmodell

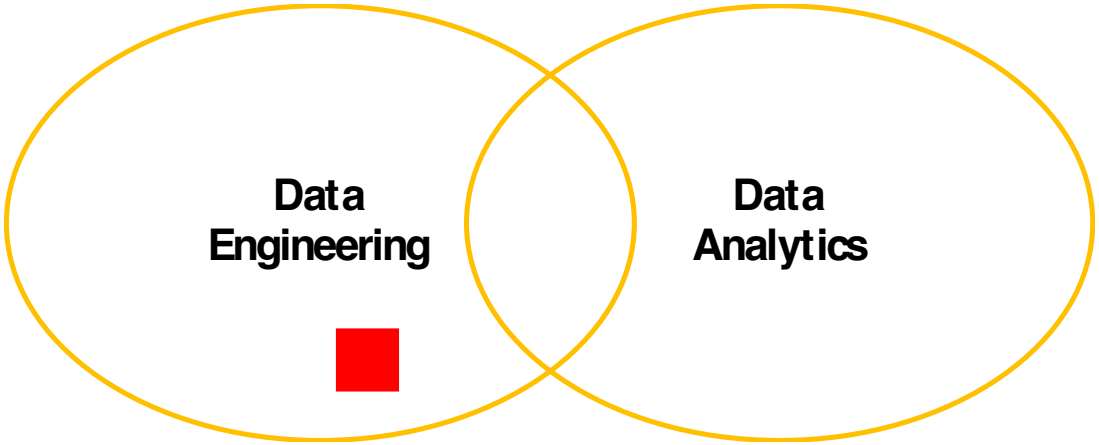
SoSe 2022

Univ.-Prof. Dr.–Ing. habil. Norbert Gronau
Lehrstuhlinhaber | Chairholder

Mail August-Bebel-Str. 89 | 14482 Potsdam | Germany
Visitors Digitalvilla am Hedy-Lamarr-Platz, 14482 Potsdam
Tel +49 331 977 3322

E-Mail ngronau@lswi.de
Web lswi.de

Kapitel 5: Objektorientiertes Datenmodell



Grundlagen objektorientierter Datenbanken

- Objektorientierung
  - > Beschreibung eines System durch das Zusammenspiel kooperierender Objekte
  - > Alles kann ein Objekt sein: Personen, Abteilungen, Bücher,...
    - » Auswahl der Objekte problemadäquat
- Elemente eines Objekts
  - > Eigenschaften: durch Attributwerte beschrieben
  - > Verhalten: durch Methoden (Operationen) ausgedrückt wird
- Grundprinzipien der Objektorientierung
  - > Datenkapselung
  - > Klassenbildung und Vererbung
  - > Nachrichtenkommunikation und Polymorphismus

Grundprinzipien der Objektorientierung

- Datenkapselung (einschließlich Objektbildung)
  - > Objekte besitzen neben Eigenschaften zusätzlich ein Verhalten
  - > Veränderung der Attributwerte nur durch die Methoden des Objekts möglich (nicht von außen)
- Klassenbildung und Vererbung
  - > Klassenbildung durch Zusammenfassung von Objekten mit denselben Attributen und demselben Verhalten
  - > Vererbung durch Erfassung von Gemeinsamkeiten verschiedener, logisch zusammengehöriger Klassen
- Nachrichtenkommunikation und Polymorphismus
  - > Nachrichtenkommunikation zwischen Objekten mit jeweils einem Sender und einem Empfänger
  - > Polymorphismus: Nachricht, die an Objekte verschiedener Klassen gesendet wird, kann jeweils unterschiedliche Reaktionen auslösen

[StHa04]

Relationale vs. objektorientierte DBS

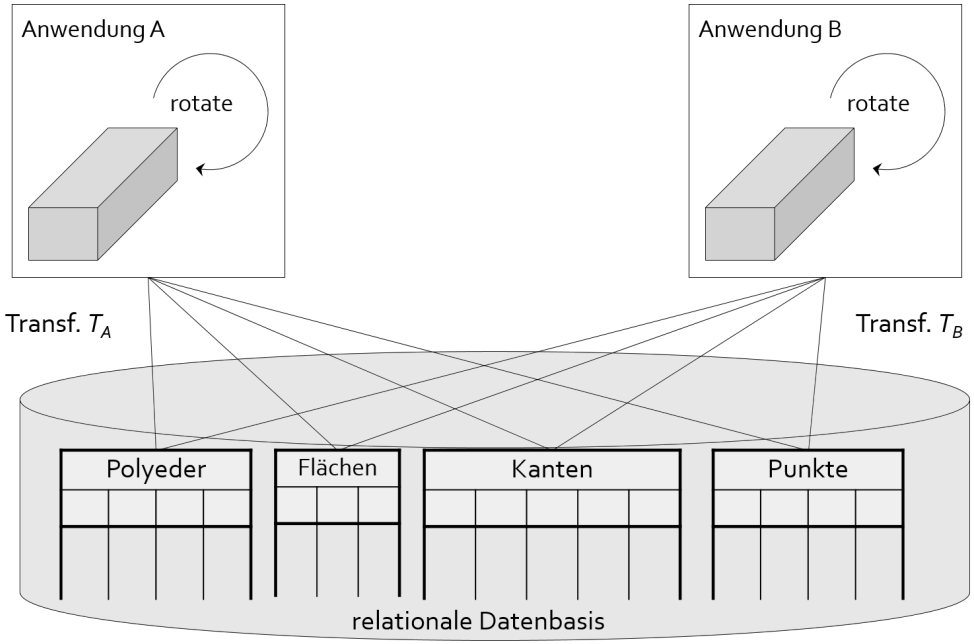
- Relationale DBS
  - > (Daten-)Objekte enthalten lediglich Attributwerte
  - > Programmoperationen der Anwendung laufen außerhalb des DBS ab
  - > mit Abfragesprachen wie SQL ist ein direkter Zugriff auf die Attributwerte möglich
- Objektorientierte DBS
  - > in Objekten werden lediglich Methoden namentlich festgehalten, Codierung selbst erfolgt innerhalb des jeweiligen Anwendungsprogramms (z.B. in C++)
  - > vgl. Grundprinzipien der Objektorientierung

[StHa04]

Nachteile des relationalen Datenmodells

- Segmentierte Daten müssen mühsam aus den Datenspalten zu Datensätzen zusammengesetzt werden (Stichwort: Join)
- Anwendungsspezifisches Verhalten wird nicht berücksichtigt

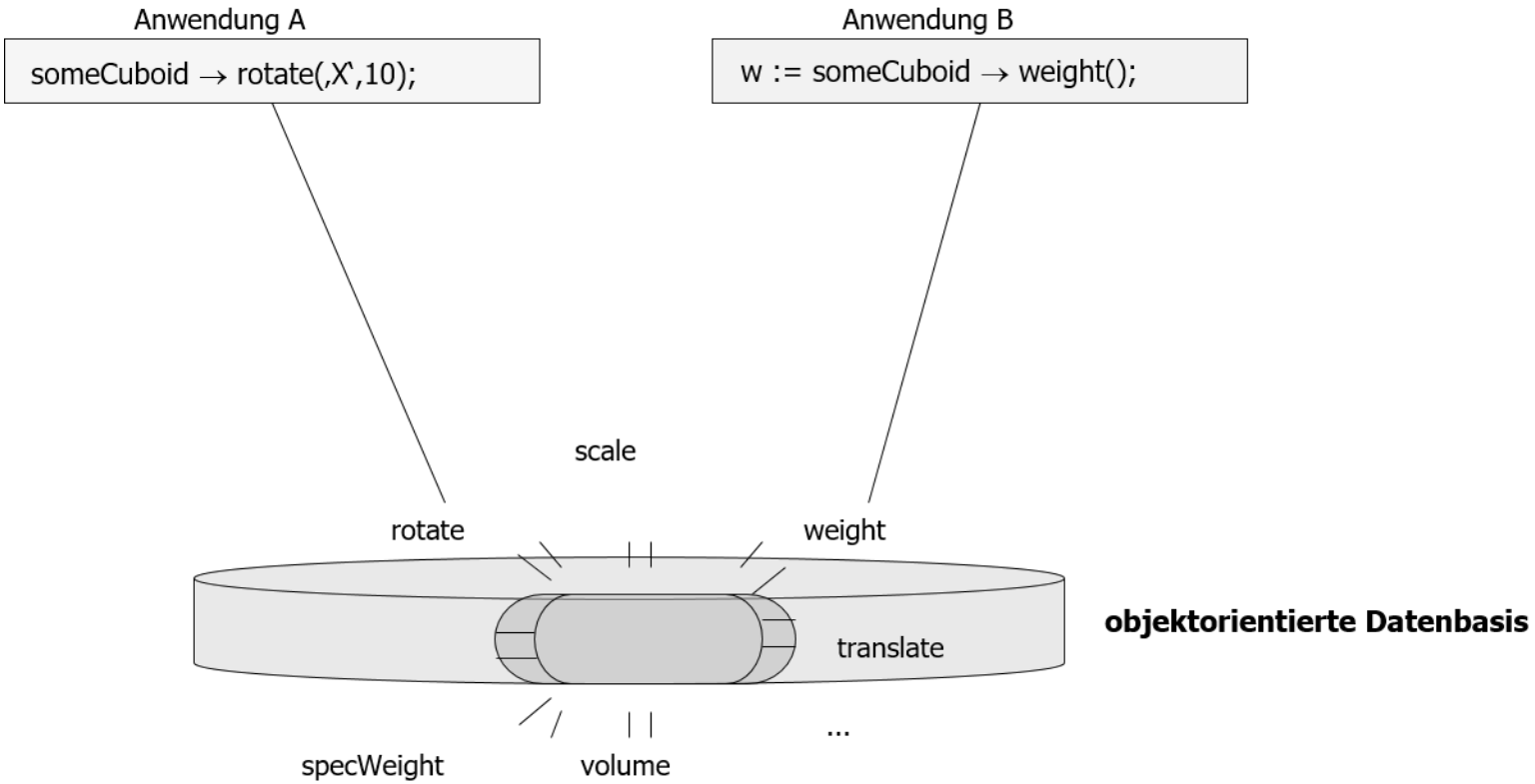
➢ Daten können nur aufwändig in Programmiersprachen mit anderem Verarbeitungsparadigma (mengen- vs. satzorientiert) eingebunden werden



[KeEi15]

Vorteile des objektorientierten Datenmodells

- Objektkapselung
- Wiederverwendbarkeit
- Operationen in der Sprache des Objektmodells



[KeEi15]

Objektorientierte und objektrelationale DBMS

- Objektorientierte Datenbankmanagementsysteme (ODBMS)
  - > Datenkapselung, Vererbungs-, Ganz/Teil- und Nachrichtenbeziehungen
  - > Object Query Language (OQL)
    - » Anfragesprache, mit der sowohl Klassen und Objekte definiert/abgefragt als auch Programme entwickelt werden
  - > Mehrfachzugriff, Datensicherheit + Datenintegrität schwer umsetzbar

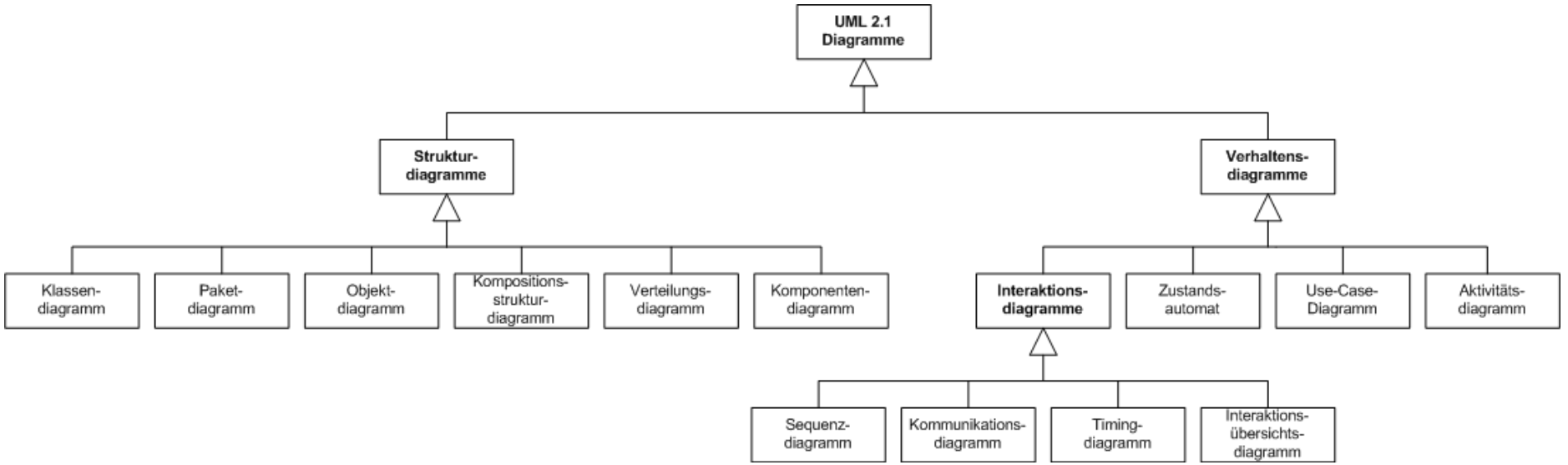
- Objektrelationale Datenbankmanagementsysteme (ORDBMS)
  - > relationale DBMS mit Erweiterungen der Objektorientierung
  - > Vorstufe: relationale DBS mit objektorientierten Schnittstellen für externe Realisierung objektorientierten Prinzipien
  - > 1. Stufe: Klassenbildung und Vererbung beschränkt auf Attribut
  - > 2. Stufe: Speicherung von Methoden zusammen mit Attributen und Ausdehnung des Vererbungsprinzips auf die Methoden

[StHa04]

Unified Modeling Language (UML)

- Grafische Modellierungssprache als einheitliches Hilfsmittel in den Phasen Spezifikation, Entwurf und Realisierung
- Objektorientierte Vorgehensweise
  - > Notation, um Realweltsituationen mit dem Ziel der Entwicklung von IT-Systemen zu analysieren und zu modellieren
- Entwicklung
  - > Vielzahl von objektorientierten Methoden in den 90er Jahren
    - » Problem: Widersprüchlichkeit und Inkompatibilität der Methoden
  - > 1997 Zusammenschluss der Methoden von Booch, Jacobson und Rumbaugh zur UML
  - > Seit 1998 Weiterentwicklung und Standardisierung
    - » Object Management Group (OMG)
    - » Einführung des Standards ISO/IEC 19501

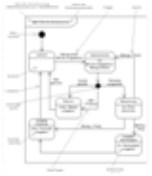
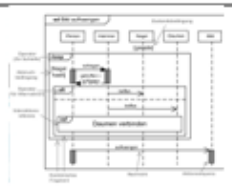
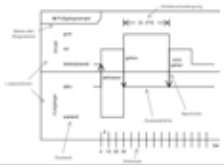
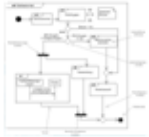
Überblick: UML-Diagrammarten



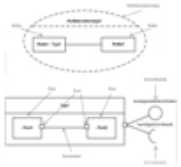
UML im IT-Projekt

- **1. Schritt:** Anwendungsnahe Modellierung der erforderlichen Systemfunktionalität
  - > Ermittlung der Anwendungsfälle (Use-Case-Diagramm)
  - > Spezifikation der Anwendungsfälle (Aktivitätsdiagramm)
- **2. Schritt:** Spezifikation der Systemstruktur
  - > Ziel: statisches Systemmodell
  - > Beispiele: Datenstruktur, Verteilung von Komponenten
  - > Strukturdiagramme
- **3. Schritt:** Definition des Systemverhaltens
  - > Ziel: laufzeitbezogenes Systemmodell
  - > Beispiele: Kommunikations- und Interaktionsverhalten von Akteuren
  - > Verhaltensdiagramme

UML: Verhaltensdiagramme

Diagramm-Bezeichnung	Darstellungsform	Zentrale Frage	Stärken
❶ Use-Case-Diagramm		Was leistet mein System für seine Umwelt (Nachbarsysteme, Stakeholder)?	Präsentiert die Außenansicht auf das System. Geeignet zur Kontextabgrenzung. Hohes Abstraktionsniveau, einfache Notationsmittel.
❷ Aktivitätsdiagramm		Wie läuft ein bestimmter flussorientierter Prozess oder ein Algorithmus ab?	Sehr detaillierte Visualisierung von Abläufen mit Bedingungen, Schleifen, Verzweigungen. Parallelisierung und Synchronisation möglich. Darstellung von Daten- und Kontrollflüssen.
❸ Zustandsautomat		Welche Zustände kann ein Objekt, eine Schnittstelle, ein Use Case, etc. bei welchen Ereignissen annehmen?	Präzise Abbildung eines Zustandsmodells mit Zuständen, Ereignissen, Nebenläufigkeiten, Bedingungen, Ein- und Austrittsaktionen. Schachtelung möglich.
❹ Sequenzdiagramm		Wer tauscht mit wem welche Informationen in welcher Reihenfolge aus?	Stellt detailliert den Informationsaustausch zwischen den Kommunikationspartnern dar. Sehr präzise Darstellung der zeitlichen Abfolge auch mit Nebenläufigkeit. Schachtelung und Flusssteuerung(Bedingungen, Schleifen, Verzweigungen) möglich.
❺ Kommunikations-diagramm		Wer kommuniziert mit wem? Wer „arbeitet" im System zusammen?	Stellt den Informationsaustausch zwischen Kommunikationspartnern dar. Überblick steht im Vordergrund (Details und zeitliche Abfolge weniger wichtig).
❻ Timing-Diagramm		Wann befinden sich verschiedene Interaktionspartner in welchem Zustand?	Visualisiert das exakte zeitliche Verhalten von Klassen, Schnittstellen, … Geeignet für die Detailbetrachtungen, bei denen es überaus wichtig ist, dass ein Ereignis zum richtigen Zeitpunkt eintritt.
❼ Interaktionsübersichts-diagramm		Wann läuft welche Interaktion ab?	Verbindet Interaktionsdiagramme (Sequenz-, Kommunikationsdiagramme) auf Top-Level-Ebene. Hohes Abstraktionsniveau. Leseeinstieg für Interaktionsdiagramme.

UML: Strukturdiagramme

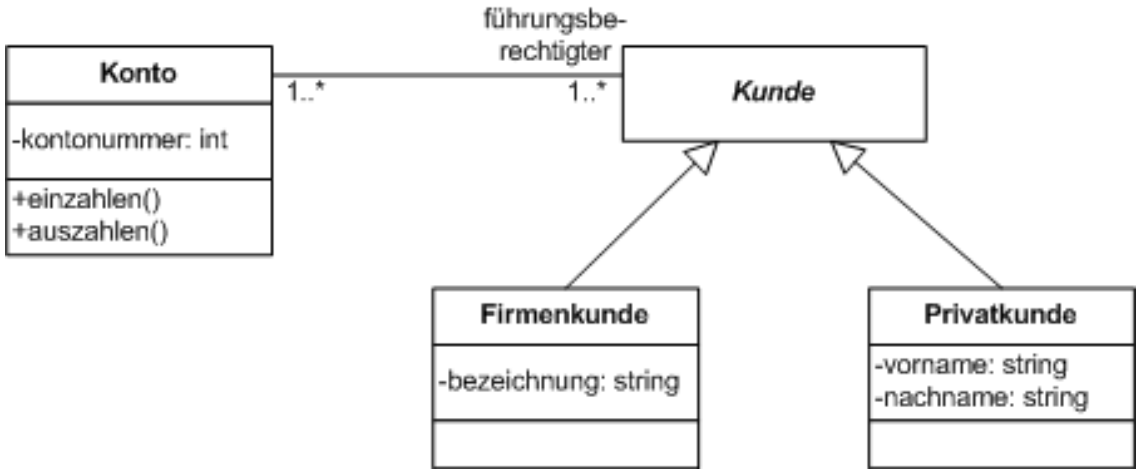
Diagramm-Bezeichnung	Darstellungsform	Leitfrage	Stärken
① Klassendiagramm		Aus welchen Klassen besteht mein System und wie stehen diese untereinander in Beziehung?	Beschreibt die statische Struktur des zu entwerfenden oder auszubildenden Systems. Enthält alle relevanten Strukturzusammenhänge und Datentypen. Bildet die Brücke zu den dynamischen Diagrammen.
② Verteilungsdiagramm		Wie sieht das Einsatzumfeld (Hardware, Server, DB…) des Systems aus? Wie werden die Komponenten zur Laufzeit wohin verteilt?	Zeigt das Langzeitumfeld des Systems mit den greifbaren Systemteilen (meist Hardware). Darstellung von „Softwareservern" möglich. Hohes Abstraktionsniveau, kaum Notationselemente.
③ Objektdiagramm		Welche innere Struktur besitzt mein System zu einem bestimmten Zeitpunkt der Laufzeit (Klassen-diagrammschnappschuss)?	Zeigt Objekte und deren Attributbelegungen zu einem bestimmten Zeitpunkt. Dient zur beispielhaften Veranschaulichung. Sehr gute Darstellung von Mengenverhältnissen.
④ Kompositionsstruktur-diagramm		Wie sieht das Innenleben einer Klasse, einer Komponente, eines Systemteils aus?	Ideal für die Top-Down-Modellierung des Systems. Präzise Modellierung der Teile-Beziehungen über spezielle Schnittstellen (Ports) möglich.
⑤ Komponentendiagramm		Wie werden meine Klassen zu wieder verwendbaren, verwalt-baren Komponenten zusammen gefasst und wie stehen diese miteinander in Beziehung?	Zeigt Organisation und Abhängigkeiten einzelner technischer Systemkomponenten. Modellierung angebotener und benötigter Schnittstellen möglich.
⑥ Paketdiagramm		Wie kann ich mein Modell so schneiden, dass ich den Überblick bewahre?	Organisiert das Systemmodell in größere Einheiten durch logische Zusammenfassung von Modellelementen. Modellierung von Abhängigkeiten und Inklusion möglich.

[RuQZ07]

Klassendiagramm

- Einführendes Beispiel:

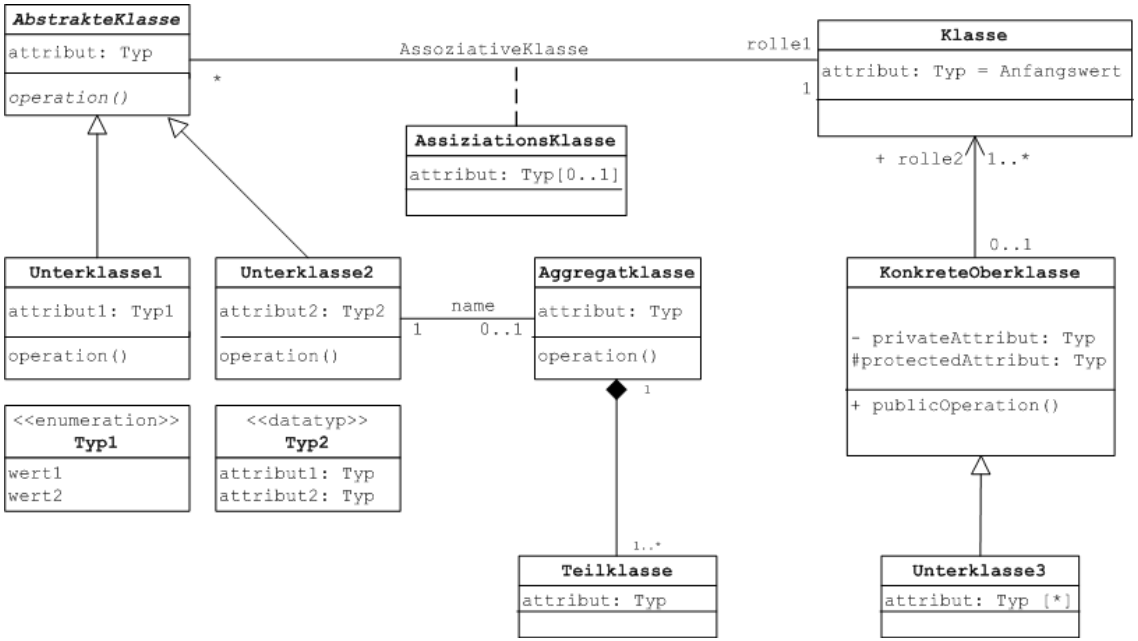
Ein Konto hat eine ganzzahlige Kontonummer. Auf einem Konto sind Einzahlungen und Auszahlungen möglich. Ein Kunde kann ein Firmenkunde oder ein Privatkunde sein. Ein Privatkunde hat einen Vor- und einen Nachnamen. Ein Firmenkunde hat eine Bezeichnung. Ein Kunde kann mehrere Konten besitzen. Ebenso kann ein Konto mehrere Kunden als Kontoführungsberechtigte haben.



[Balz10]

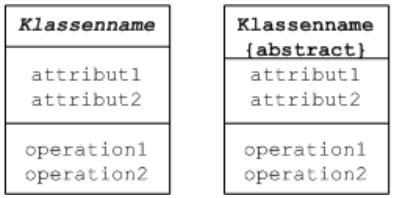
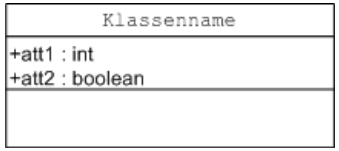
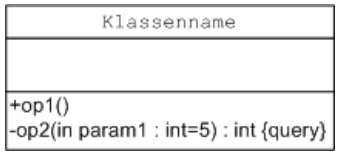
Klassendiagramm

- Leitfrage
  - > Aus welchen Klassen besteht mein System und wie stehen diese untereinander in Beziehung?
- Lösungsansatz
  - > Beschreibung der statischen Struktur des Systems
  - > Darstellung von Strukturzusammenhängen und Datentypen

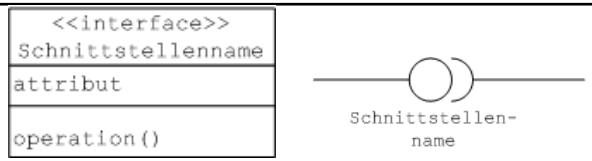
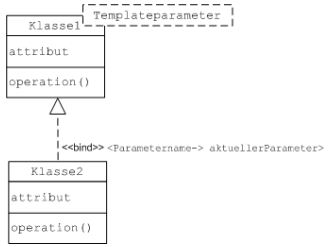
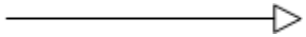
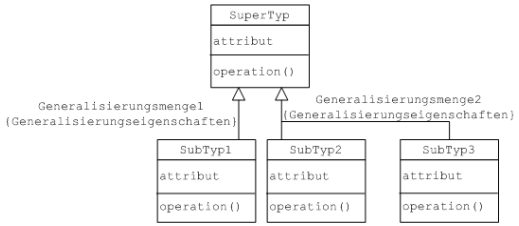
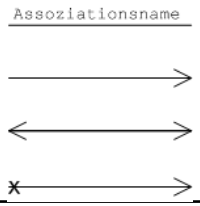


[Balz10]

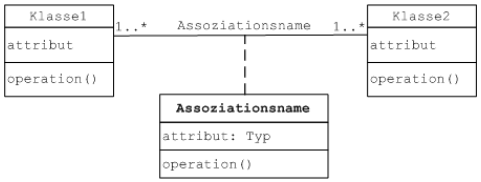
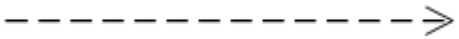
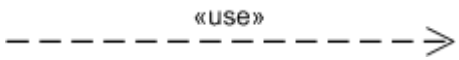
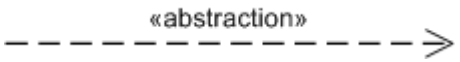
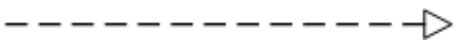
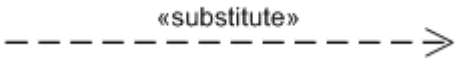
Klassendiagramm: Notationselemente (1)

Bezeichnung	Notation	Beschreibung
Klasse		Menge von Objekten mit gleichen Eigenschaften (Attribute), Verhalten (Operationen) und Beziehungen (Assoziationen und Generalisierungsstrukturen) ➢ Klassenname: Substantiv, Singular
Abstrakte Klasse		Klasse, von der keine Exemplare angelegt werden können
Aktive Klasse		Besitzt ausführenden Prozess/Task ➢ Alle Instanzen sind aktive Objekte
Attribut		Strukturelle Eigenschaften von Klassen ➢ Alle Objekte einer Klasse besitzen dieselben Attribute, aber unterschiedliche Attributwerte ➢ Eindeutigkeit des Namens innerhalb der Klasse
Operation		Verhaltensmerkmal ➢ Jede Operation kann auf alle Attribute eines Objektes in einer Klasse direkt zugreifen ➢ Stellen Interaktion mit Objekten der Klasse dar ➢ Einzige Möglichkeit zur Zustandsveränderung eines Objekts

Klassendiagramm: Notationselemente (2)

Bezeichnung	Notation	Beschreibung
Schnittstelle		Liste von Attributen, Operationen und Signalempfängern, die öffentlich sichtbar sind ➢ Unterscheidung zwischen angebotenen und benötigten Schnittstellen ➢ Alle aufgeführten Operationen müssen realisiert werden
Parametrisierte Klasse		Argument eines Verhaltensbestandteils (z. B. einer Operation) ➢ Beschreibt Klasse mit einem oder mehreren formalen Parametern (definiert Familie von Klassen) ➢ Um parametrisierte Klasse zu benutzen, müssen formale Parameter an aktuelle Parameter gebunden werden ➢ Bindung einer konkreten an eine parametrisierte Klasse durch Generalisierungspfeil
Generalisierung		Mittel zur Abstraktion durch Verallgemeinerung von einem speziellen zu einem allgemeineren Element
Generalisierungs-menge		Generalisierungsstrategie ➢ Partitionierung in Unterklassen ➢ Einschränkungen: complete, incomplete, disjoint, overlapping (z.B. Geschlecht {complete, disjoint})
Aggregation		Teil-Ganzes-Beziehung ➢ Teilobjekt (z.B. Kapitel) kann in mehreren Aggregatobjekten (z.B. Büchern) enthalten sein
Komposition		Teil-Ganzes-Beziehung ➢ Teilobjekt (z.B. Datei) kann nur einem Aggregatobjekt (z.B. Verzeichnis) angehören ➢ Das Ganze ist verantwortlich für das Erzeugen/Löschen von Teilen
Assoziation		Beziehung zwischen zwei oder mehr gleichartigen Klassen ➢ Unspezifiziert ➢ Unidirektional ➢ Bidirektional ➢ Navigierbarkeit ausgeschlossen

Klassendiagramm: Notationselemente (3)

Bezeichnung	Notation	Beschreibung
Assoziationsklasse		Assoziation, die zusätzlich Eigenschaften einer Klasse besitzt ➢ Zusätzliche Attribute, Operationen, Assoziationen zu anderen Klassen ➢ Besitzt Eigenschaften einer Klasse und einer Assoziation
Abhängigkeits-beziehung		Strukturelle oder semantische Abhängigkeiten
Verwendungs-beziehung		Verbindet ein oder mehrere abhängige Modellelemente mit einem oder mehreren Elementen, die für die Realisierung der abhängigen Modellelemente erforderlich sind
Abstraktions-beziehung		Verbindet zwei oder mehr Elemente, die sich auf unterschiedlichen Abstraktionsebenen befinden oder unterschiedliche Sichten darstellen ➢ Stereotypen: <<derive>>, <<refine>>, <<trace>>
Realisierungs-beziehung		Abstraktionsbeziehung zwischen einzelnen Elementen oder Mengen von Elementen
Substitutions-beziehung		Abstraktionsbeziehung, die zu ersetzendes Element kennzeichnet ➢ Verbindet nur Classifier (z.B. Klassen, Komponenten)
Informationsfluss		Informationsübertragung
Informationseinheit		Information

Klassendiagramm: Attribut (1)

- Attributname
  - > Substantiv, Einzahl
- Klassenattribut
  - > nur ein Attributwert für alle Ausprägungen der Klasse
  - > z.B. bei Klasse Artikel: `preis`
- Abgeleitete Attribute
  - > Inhalt wird aus anderen im System vorliegenden Daten zur Laufzeit berechnet; unveränderlich
  - > Beispiel: Ableitung des Alters aus Geburtsdatum: `/alter`
- Sichtbarkeiten
  - > + = public, - = private, # = protected, ~ = package
- Multiplizität
  - > Festlegung der Ober- und Untergrenze unter einem Attributnamen ablegbarer Instanzen
  - > Veränderung von Inhaltswerten
  - > [0..1], [1..1], [1..*], [n..m]

[Balz10]

Klassendiagramm: Attribut (2)

- Eigenschaften
  - > readOnly, z.B. `kontonr {readOnly}`
  - > subsets, z.B. `gerade Ziffern {subsets ziffern}`
  - > union, z.B. `ziffern {union}`
  - > ordered, z.B. `vorname {ordered}`
  - > unique, z.B. `lottozahlen [6] {unique}`
  - > nonunique, z.B. `noten [1..8] {nonunique}`
  - > redefines, z.B. `personalnr: Integer {redefines nr}`
- Einschränkungen
  - > können beliebig spezifiziert werden und sich auf mehrere Attribute beziehen
  - > Beispiel: `anzahl {anzahl > 0}`
- Attributtyp
  - > Datentyp
    - » Primitiver Datentyp <<primitive>>
    - » Aufzählungstyp <<enumeration>>

[Balz10]

Klassendiagramm: Operation (1)

- Operationsname
  - > Verb, Präsens
- Klassenoperation
  - > Operation ist für alle Objekte einer Klasse zugehörig
  - > Beispiel: bei Klasse Artikel: `artikellisteausgeben`
- Abstrakte Operation
  - > Besteht nur aus der Signatur und besitzt keine Implementierung
  - > Beispiel: bei Klasse Box: *`erledigeBoxenstopp()`*
- Sichtbarkeiten
  - > + = public, - = privat; # = protected, ~ = package
- Multiplizität
  - > Ausführung von Operationen
  - > [0..1]; [1..1]; [1..*]; [n..m]

[Balz10]

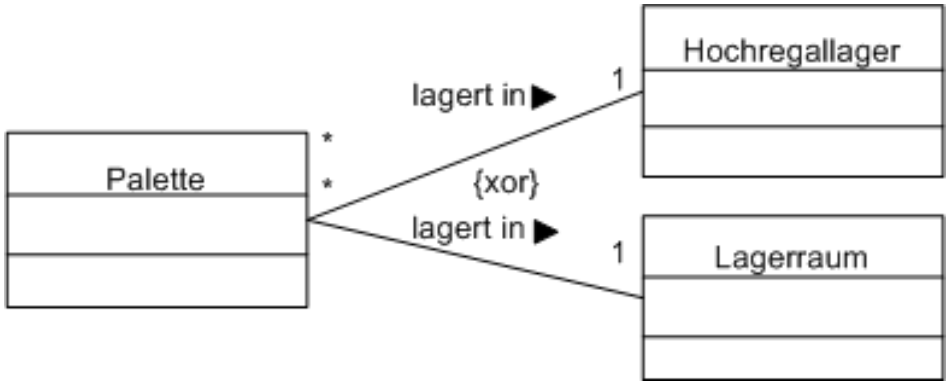
Klassendiagramm: Operation (2)

- Parameter
  - in, out, inout, return
- Eigenschaften
  - redefines `operation`
  - query
  - ordered
  - unique
- Syntax
  - `operationsname :: = [Sichtbarkeit] name ([Parameterliste])[: [Rückgabetyp] {eigenschaftswert [,eigenschaftswert]*}]`
  - `Parameterliste ::= [Übergaberichtung] name : Typ [Multiplizität]] [Vorgabewert][{eigenschaftswert [, eigenschaftswert]*}]`

[Balz10]

Klassendiagramm: Assoziation (1)

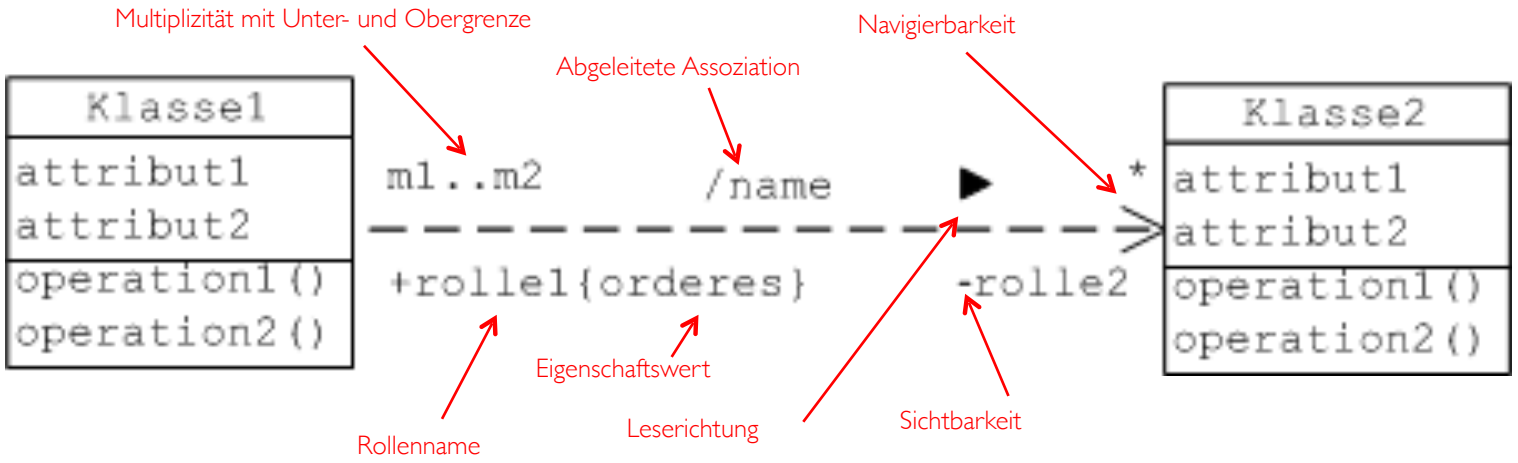
- Rollenname
  - > Information über die Bedeutung der Klasse in der Assoziation
- Assoziationsname
  - > beschreibt Richtung der Assoziation
- Sichtbarkeit
  - > nur bei navigierbaren Assoziationen zulässig
- Multiplizität
  - > Angabe wie viele Objekte ein bestimmtes Objekt kennen



[Balz10]

Klassendiagramm: Assoziation (2)

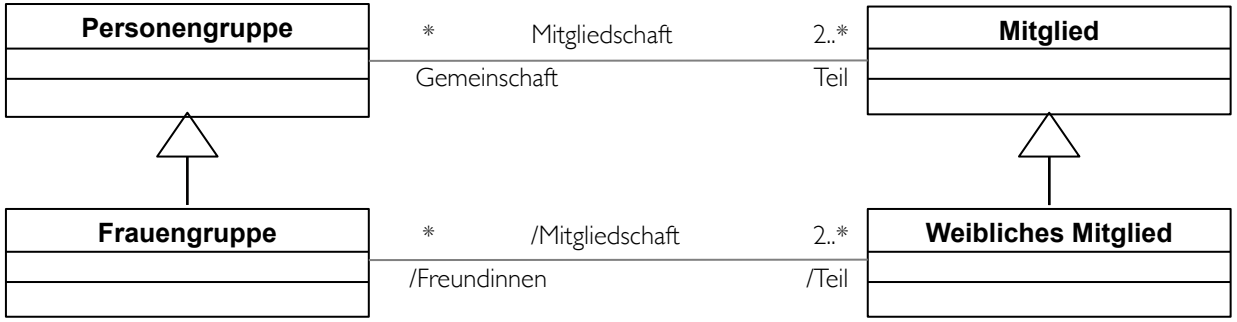
- Eigenschaftswerte
  - > subsets `Eigenschaft`
  - > union
  - > redefines `Rollenname`
  - > ordered
  - > nonunique
  - > sequence



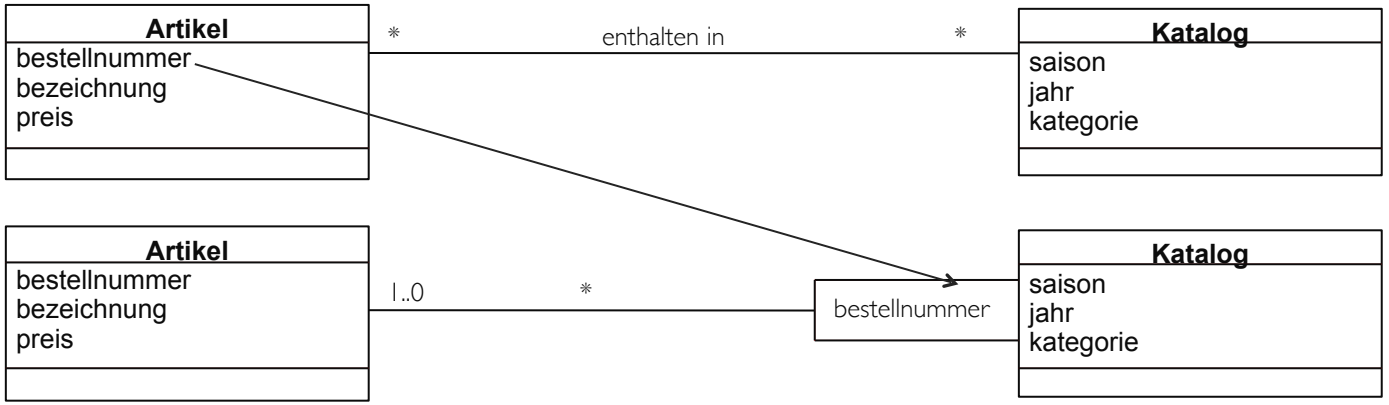
[Balz10]

Klassendiagramm: Assoziation (3)

- Abgeleitete Assoziation
  - > Beschreibung gleicher Abhängigkeiten bereits durch andere Assoziationen



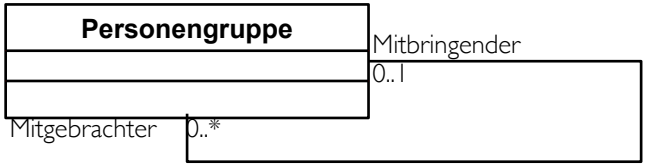
- Qualifizierte Assoziation
  - > Zwischen zwei Klassen
  - > Zerlegt Objekte der gegenüberliegenden Klasse in Partitionen



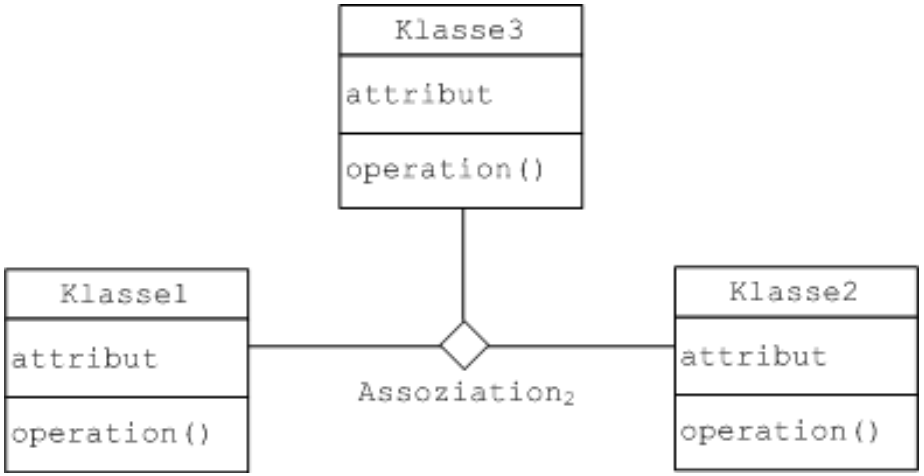
[Balz10]

Klassendiagramm: Assoziation (4)

- Zirkuläre Assoziation
  - > Objekte gehören derselben Klasse an

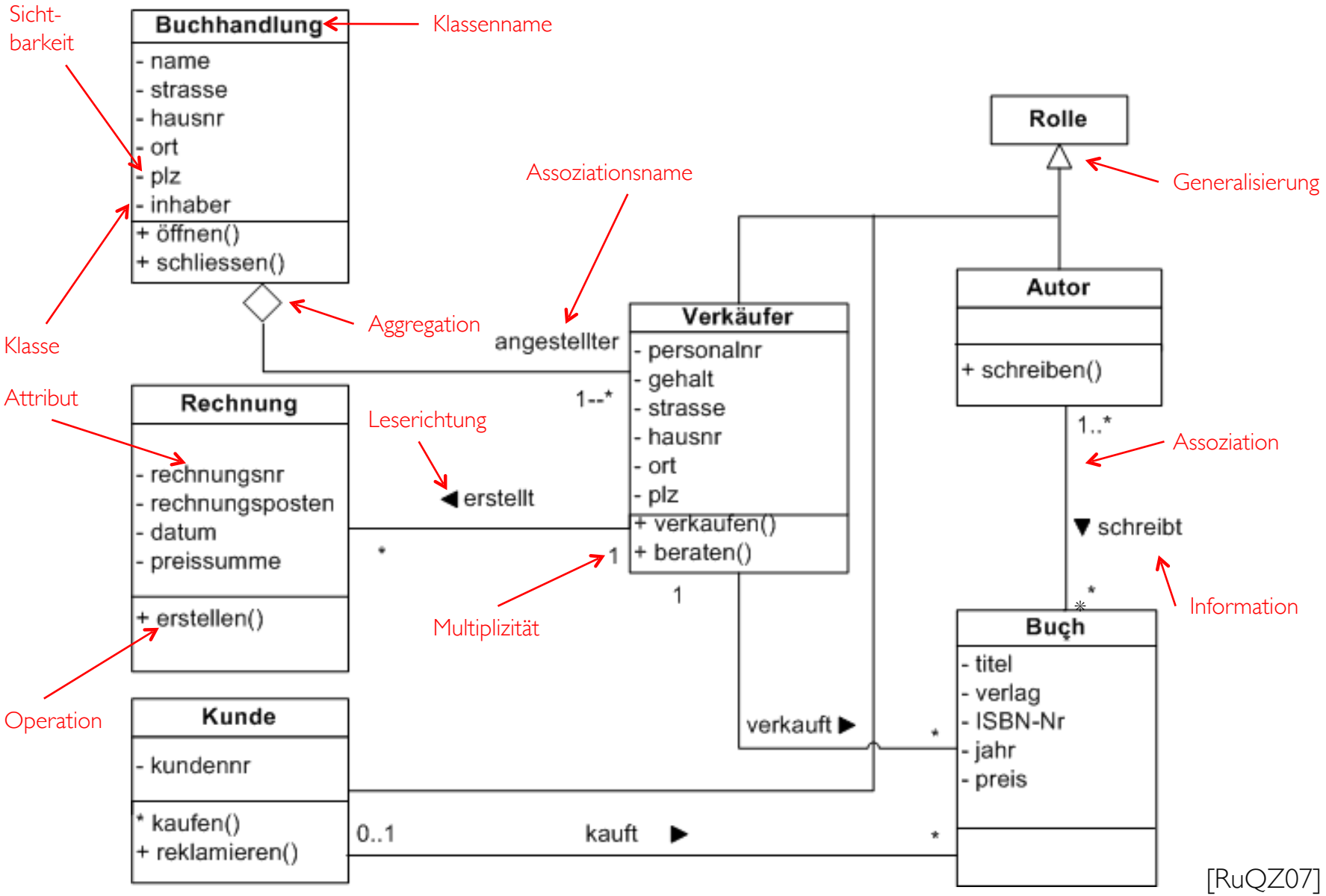


- n-äre Assoziationen



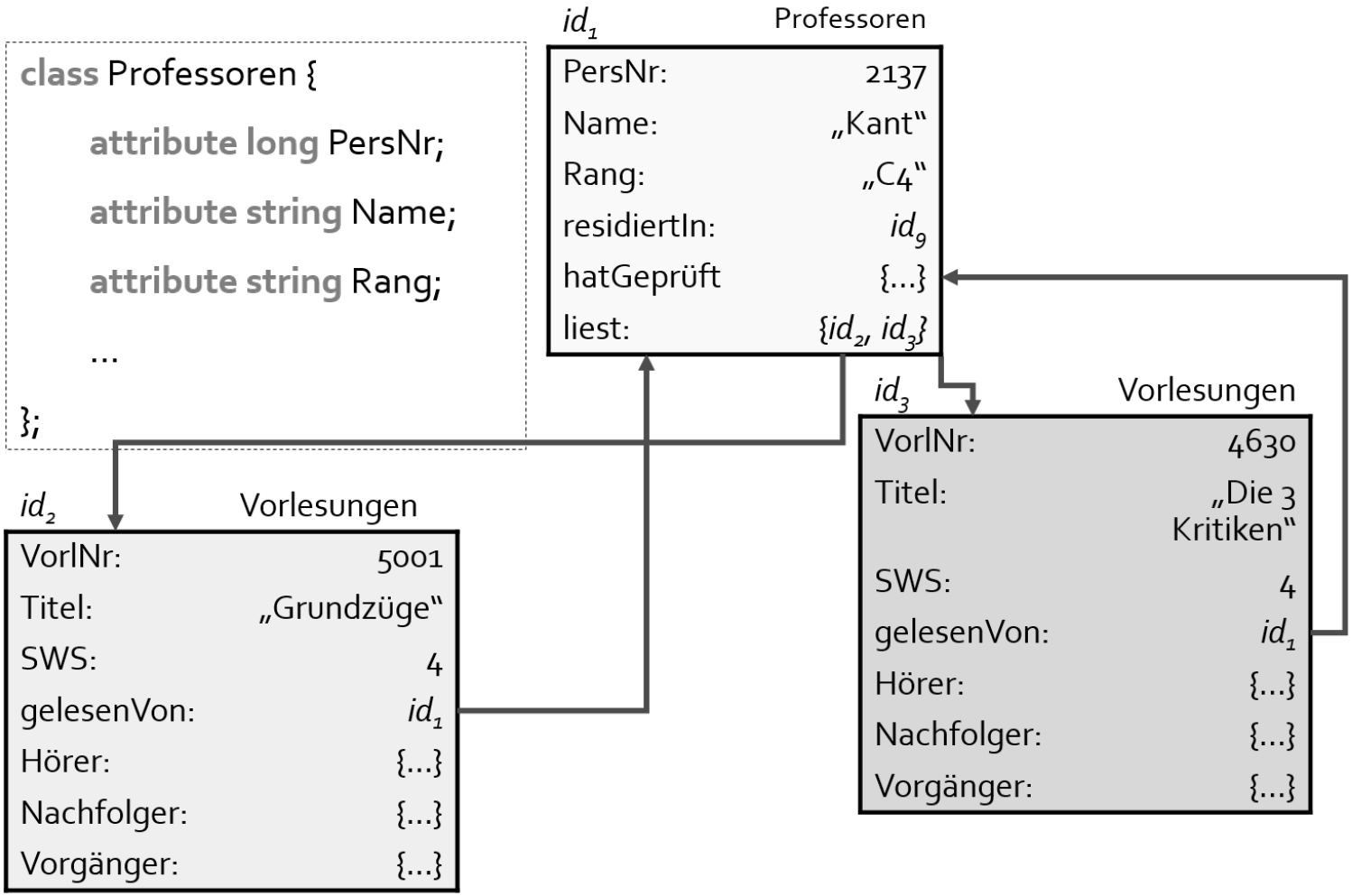
[Balz10]

Beispiel: Buchhandlung



[RuQZ07]

Beispiel: Universitätswelt



[KeEi15]

Beispiel: Universitätsschema

```
class Professoren {
        attribute long PersNr;
        attribute string Name;
        attribute string Rang:
        relationship Räume residiertIn inverse Räume::beherbergt;
        relationship set(Vorlesungen) liest inverse Vorlesungen::gelesenVon
        relationship set(Prüfungen) hatGeprüft inverse Prüfungen::Prüfer;
};
class Vorlesungen {
        attribute long VorlNr;
        attribute string Titel;
        attribute short SWS;
        relationship Professoren gelesenVon
        inverse Professoren::liest;
        relationship set(Studenten) Hörer inverse Studenten::hört;
        relationship set(Vorlesungen) Nachfolger inverse Vorlesungen::Vorgänger;
        relationship set(Vorlesungen) Vorgänger inverse Vorlesungen::Nachfolger;
        relationship set(Prüfungen) wurdeAbgeprüft inverse Prüfungen::Inhalt;
};
class Studenten {
        ...
        relationship set(Prüfungen) wurdeGeprüft inverse Prüfungen::Prüfling;
};
```

[KeEi15]

Objekt-Identität (OID)

- Charakteristikum objektorientierter Datenmodellierung
  - > Verweise werden über die OID realisiert
- Realisierungsformen
  - > Physische OIDs
    - » enthalten den Speicherort des Objekts
    - » im wesentlichen entsprechen diese den TIDs
  - > Logische OIDs
    - » unabhängig vom Speicherort der Objekte
    - » d.h. Objekte können verschoben werden
    - » Indirektion über eine „Mapping"-Struktur
      - B-Baum
      - Hash-Tabelle
      - Direct Mapping

[KeEi15]

Beispiel: Klassen-Definition von Operationen in ODL

- Beispiel-Operationen

```
class Professoren {
        exception hatNochNichtGeprüft {};
        exception schonHöchsteStufe {};
        ...
        float wieHartAlsPrüfer() raises (hatNochNichtGeprüft);
         void befördert() raises (schonHöchsteStufe);
};
```

- Aufruf der Operationen
  - > im Anwendungsprogramm:

    meinLieblingsProf→befördert();

  - > in OQL:

```
select p.wieHartAlsPrüfer()
  from p in AlleProfessoren
  where p.Name = "Curie";
```

[KeEi15]

Abfragesprache Object Query Language (OQL)

- An SQL angelehnte Abfragesprache für objektorientierte Datenbanken
- Beispiel: Einfache Anfragen
  - > Finde die Namen der W3-Professoren

```
select p.Name
  from p in AlleProfessoren
  where p.Rang = „W3“;
```

  - > Generiere Namen- und Rang-Tupel der W3-Professoren

```
select struct(n: p.Name, r: p.Rang)
  from p in AlleProfessoren
  where p.Rang = „W3“;
```

- Beispiel: Geschachtelte Anfragen und Partitionierung

```
select struct(n: p.Name, a: sum(select v.SWS
  from v in p.liest))
  from p in Alle Professoren
  where avg(select v.SWS from v in p.liest) > 2;
```

[KeEi15]

Beispiele für objektorientierte Datenbanken

- Versant
  - > Versant Object Database
    Mit C++ API
  - > Versant JPA
    Mit Java API
  - > Versant FastObjekts
    Mit .NET API
- Db4o (database for objects)
  - > Nicht-kommerzielle Open-Source Datenbank für Java und .NET
- Poet
  - > Unterstützt C++ und Java
  - > Varianten: FastObjects t7/e7/j2

Literaturverzeichnis

Literaturverzeichnis

[Bapp14] Bappalige, O. (2014). Introduction to Apache Hadoop. URL: https://opensource.com/life/14/8/intro-apache-hadoop-big-data

[BEPW16] Backhaus, K., Erichson, B., Plinke, W., Weiber, R. (2015). Multivariate Analysemethoden: Eine anwendungsorientierte Einführung. Springer-Verlag.

[Dave14] Davenport, T. (2014). Big data at work: dispelling the myths, uncovering the opportunities. Harvard Business Review Press.

[EFH+11] Edlich, S., Friedland, A., Hampe, J., Brauer, B., Brückner, M. (2011). NoSQL: Einstieg in die Welt nichtrelationaler Web 2.0 Datenbanken. Hanser.

[FaEg05] Fassott, G., Eggert, A. (2005). Zur Verwendung formative und reflektiver Indikatoren in Strukturgleichungsmodellen, in: Handbuch PLS- Pfadmodellierung. Schäffer Poeschel.

[FaPS96] Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P. (1996, August). Knowledge Discovery and Data Mining: Towards a Unifying Framework. In KDD (Vol. 96, pp. 82-88).

[FHKo17] Fachhochschule Köln (2011). Datenbanken Online Lexikon. URL: http://wikis.gm.fh-koeln.de/wiki_db/Datenbanken

Literaturverzeichnis

[GaUW08] Garcia-Molina, H., Ullman, J. D., Widom, J. (2008). Database Systems: The Complete Book. Pearson International.

[GMPB+09] Ginsberg, J., Mohebbi, M. H., Patel, R. S., Brammer, L., Smolinski, M. S., & Brilliant, L. (2009). Detecting influenza epidemics using search engine query data. Nature, 457 (7232), 1012-1014.

[HaKa06] Han, J., Kamber, M. (2006). Data mining: concepts and techniques. Morgan Kaufmann.

[Harr15] Harrison, G. P. (2015). Next Generation Databases - NoSQL and Big Data. Apress.

[JZFF11] Jannach, D., Zanker, M., Felfernig, A., & Friedrich, G. (2010). Recommender Systems: An Introduction. Cambridge University Press.

[KeBM10] Kemper, H. G., Baars, H., Mehanna, W. (2010). Business Intelligence: Grundlagen und praktische Anwendungen. Vieweg.

Literaturverzeichnis

[KeEi15] Kemper, A., Eickler, A. (2015). Datenbanksysteme. Oldenbourg.

[KrGL05] Kraft, M., Götz, O., Liehr-Gobbers, K. (2005). Die Validierung von Strukturgleichungsmodellen mit Hilfe des Partial-Least-Squares-Ansatzes, in: Handbuch PLS-Pfadmodellierung. Schäffer Poeschel.

[KuJo13] Kuhn, M., Johnson, K. (2013): Applied Predictice Modeling. Springer.

[LKKV14] Lazer, D., Kennedy, R., King, G., Vespignani, A. (2014). The parable of Google Flu: traps in big data analysis. Science, 343 (6176), 1203-1205.

[MBK+12] Mertens, P., Bodendorf, F., König, W., Schumann, M., Hess, T. (2012). Grundzüge der Wirtschaftsinformatik. Springer.

[Muel15] Müller, E. (2015). Vorlesungsskript: Big Data Analytics. Hasso-Plattner- Institut Potsdam.

[Naum15] Naumann, F. (2015). Vorlesungsskript: Datenbanksysteme II. Hasso-Plattner-Institut Potsdam.

Literaturverzeichnis

[O'Nei16] O'Neil, C. (2016). Weapons of math destruction: How big data increases inequality and threatens democracy. Crown Books.

[PäPa15] Pääkkönen, P., Pakkala, D. (2015). Reference architecture and classification of technologies, products and services for big data systems. Big Data Research, 2 (4), 166-186.

[Plat13] Plattner, H. (2013). A Course in In-Memory Data Management. Springer.

[PoBe13] Becker, L., Pousttchi, K. (2013). Requirements for personalized m-commerce - what drives consumers' use of social networks?. In: Journal of Electronic Commerce in Organizations 11 (2013) 4, S. 19-36.

[RaSS15] Rahm, E., Saake, G., & Sattler, K. U. (2015). Verteiltes und paralleles Datenmanagement: von verteilten Datenbanken zu Big Data und Cloud. Springer.

[RePi91] Reichwald, R., Picot, A. (1991). Informationswirtschaft, in: Heinen, E. (Hrsg.). Industriebetriebslehre: Entscheidungen im Industriebetrieb. Springer.

[Rupa16] Ruparelia, N. B. (2016). Cloud Computing. MIT Press.

Literaturverzeichnis

[RuQZ07] Rupp, C., Queins, S., Zengler, B. (2007). UML 2 glasklar: Praxiswissen für die UML-Modellierung. Carl Hanser.

[SaSH11] Saake, G., Sattler, K. U., Heuer, A. (2011). Datenbanken: Implementierungstechniken. mitp.

[SaSH13] Saake, G., Sattler, K. U., Heuer, A. (2013). Datenbanken: Konzepte und Sprachen. mitp.

[Sche97] Scheer, A. W. (1997). ARIS—vom Geschäftsprozess zum Anwendungssystem. Springer.

[StHa04] Stahlknecht, P., Hasenkamp, U. (2004). Einführung in die Wirtschaftsinformatik. Springer.

[Sull15] Sullivan, D. (2015). NoSQL for mere mortals. Addison-Wesley Professional.

[Thor13] Thor, A. (2013). Vorlesungsskript: Datenmanagement. Universität Leipzig.

Literaturverzeichnis

[Vige16] Vigen, T. (2016). Spurious Correlations. Hachette Books.

[Voss08] Vossen, G. (2008). Datenmodelle, Datenbanksprachen und Datenbank-Management-Systeme. Oldenbourg.

[WeKa16] Wedel, M., Kannan, P. K. (2016). Marketing analytics for data-rich environments. Journal of Marketing, 80(6), 97-121.

[Wies15] Wiese, L. (2015). Advanced Data Management: For SQL, NoSQL, Cloud and Distributed Databases. de Gruyter.

[Zehn13] Zehnder, C. A. (2013). Informationssysteme und Datenbanken. Springer.

[Zoss15] Zoss, A. (2015). Practical Data Visualization. Lecture at Duke University.