



Prof. Dr. Peer Küppers

Abstract

Dieses Dokument beschreibt begleitend zu den Vorlesungen und Übungen die Inhalte der Veranstaltung BW342, Datenbanken I mit Praktikum. Somit ist es als komplementär zu den Präsenzveranstaltungen zur Vor- und Nachbereitung anzusehen, ersetzt diese jedoch nicht. Darüber hinaus wird im Selbststudium auch die Lektüre der angegebenen Quellen (relevante Ausschnitte) und externer Materialien, sowie die eigenständige praktische Auseinandersetzung mit dem Datenbanksystem PostgreSQL erwartet.

Inhalt

1	Einführung und Motivation.....	3
2	Datenbankgrundlagen	5
2.1	Definitionen	5
2.2	Datenabstraktion	6
2.2.1	Externe Ebene	7
2.2.2	Konzeptionelle Ebene	7
2.2.3	Interne Ebene.....	7
2.3	Komponenten eines DBMS	7
2.4	Arten von DBMS	9
3	Datenbankmodellierung.....	10
3.1	Grundlagen	10
3.1.1	Zweistufiger Modellierungsansatz	10
3.1.2	Konzeptionelle Modellierung	11
3.1.3	Logische Modellierung	11
4	Konzeptionelle Modellierung	12
4.1	Zielgruppe, Notationen und Elemente	12
4.2	Fallbeispiel Online-Lebensmittelhandel	13
4.3	Geschäftsobjekte	14
4.4	Sub- bzw. Supertypen	14
4.5	Attribute und Schlüssel.....	14
4.6	Beziehungen und Beziehungstypen	15
4.7	Entity-Relationship-Modell nach Chen.....	16

4.7.1	Allgemein	16
4.7.2	Geschäftsobjekte	16
4.7.3	Sub- bzw. Supertypen (=Generalisierung bzw. Spezialisierung).....	17
4.7.4	Attribute.....	17
4.7.5	Beziehungen und Beziehungstypen	18
4.7.6	Hinweis zu Kardinalitäten	18
4.7.7	Hierarchien.....	19
4.7.8	Strukturen	19
4.7.9	N:M-Beziehungen / Uminterpretation	19
4.7.10	Modellkonsolidierung.....	20
5	Literaturverzeichnis	23

1 Einführung und Motivation

Bis Ende der 1960er Jahre wurde für Anwendungen der direkte Zugriff auf das Dateisystem aus Programmen heraus zum Speichern von Daten verwendet. Jede Anwendung eines Unternehmens speicherte in seinen eigenen Dateien die Daten, die es benötigte. Dabei entwickelten sich entsprechend unterschiedliche „Datenformate“ je nach Programmierer, Applikation, usw.

Die separate Abspeicherung von unter Umständen zueinander in Beziehung stehenden Daten durch verschiedene Anwendungen führt zu diversen Problemen:

- **Redundanz:** Dieselben Informationen werden mehrfach gespeichert.
Herr Müller speichert die Datei an einem anderen Ort als Herr Schmidt, womit ein Duplikat erschaffen wurde. Für "korrekte" Daten müsste künftig alles in beiden Dateien gespeichert werden
- **Integrität / Inkonsistenz:** Die Einhaltung von Integritätsbedingungen fällt schwer und es existieren unterschiedliche Versionen derselben Informationen.
Wird nicht in beiden Dateien gespeichert, laufen die Daten „auseinander“ und sind nicht mehr konsistent
- **Verknüpfungseinschränkung:** Logisch verwandte Daten sind schwer zu verknüpfen, wenn sie in isolierten Dateien liegen.
- **Mehrbenutzerprobleme:** Das gleichzeitige Bearbeiten derselben Datei führt zu Anomalien („lost update“).
Wenn zwei Mitarbeiter die Datei gleichzeitig bearbeiten wollen, besteht die Gefahr des gegenseitigen Überschreibens von Daten.
- **Verlust von Daten:** Begrenzt auf komplettes Backup der einzelnen Dateien. Kein spezieller „Recover-Mechanismus“
- **Sicherheitsprobleme:** Zugriffsrechte können nicht in abgestufter Form implementiert werden.
Nicht jeder darf alles sehen!
- **Hohe Entwicklungskosten:** Für jedes Anwendungsprogramm müssen die Fragen zur Dateiverwaltung erneut gelöst werden.
Sollte das „Datenformat“ geändert werden, können u. U. Programme die Daten nicht mehr korrekt „nutzen“.

Beispiel: verschiedene Anwendungen verwalten in ihren eigenen Dateien die Kundendaten eines spezifischen Kunden von einem Unternehmen; die gleiche Adresse wird also in verschiedenen Dateien gespeichert. Bei einer Änderung der Adresse mussten alle Dateien gleichzeitig angepasst werden. → Redundanz, Gefahr der Inkonsistenz usw.

Somit bietet es sich an, Anwendungen Zugriff auf eine gemeinsame Datenbasis mit Hilfe eines Datenbanksystems zu ermöglichen, wie in Abbildung 1 dargestellt:

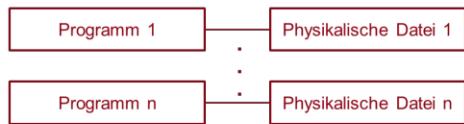
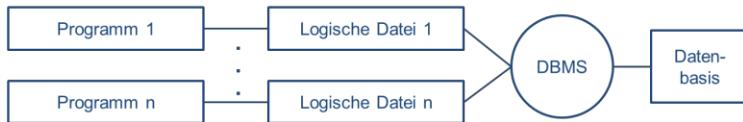
Übergang von isolierten Dateien:**Zu einer zentralen Datenbasis:**

Abbildung 1: Übergang von isolierten Dateien zu einer zentralen Datenbasis

All diese Gründe erfordern es, die Daten unter die „Obhut“ eines eigenen Softwareprogrammes zu geben, das die Daten verwaltet.



Empfohlene weiterführende Lektüre: (Elmasri und Navathe 2014), Kapitel 1.6

Datenbanken und Datenbankmanagementsysteme sind heute allgegenwärtig.

- Tagsüber prüfen wir mehrfach unseren digitalen Terminkalender
- In der App der Deutschen Bahn informieren wir uns über die für uns perfekte Verbindung zur Hochschule
- Per Online-Banking prüfen wir unseren Kontobetrag und tätigen Überweisungen
- Bei Amazon kaufen wir ein Smartphone
- ...

Jede dieser Aktionen erfordert das Vorhandensein und Zugreifen auf (mindestens eine) Datenbank.

Unternehmen können heute nur noch erfolgreich sein, wenn sie über qualitativ und quantitativ hochwertige Daten verfügen (vgl. Abbildung 2). Hierzu werden teilweise sehr spezielle Datenbanken entwickelt (bspw. Google BigTable).

Forbes The World's Most Valuable Brands							2017
Rank	Brand	Brand Value	1-Yr Value Change	Brand Revenue	Company Advertising	Industry	
#1	Apple	\$170 B	10%	\$214.2 B	\$1.8 B	Technology	
#2	Google	\$101.8 B	23%	\$80.5 B	\$3.9 B	Technology	
#3	Microsoft	\$87 B	16%	\$85.3 B	\$1.6 B	Technology	
#4	Facebook	\$73.5 B	40%	\$25.6 B	\$310 M	Technology	
#5	Coca-Cola	\$56.4 B	-4%	\$23 B	\$4 B	Beverages	
#6	Amazon	\$54.1 B	54%	\$133 B	\$5 B	Technology	
#7	Disney	\$43.9 B	11%	\$30.7 B	\$2.9 B	Leisure	
#8	Toyota	\$41.1 B	-2%	\$168.8 B	\$4.3 B	Automotive	

Abbildung 2: Marken-Ranking 2017 - Technologieunternehmen führend

2 Datenbankgrundlagen

2.1 Definitionen

Ein Datenbankmanagementsystem (DBMS) ist eine Software, die große Mengen von persistenten Daten für die Speicherung und den Zugriff durch mehrere Benutzer verwaltet.

Dies soll unter folgenden Anforderungen erfolgen: sicher (In Bezug auf Systemausfälle und Nutzerberechtigungen), effizient (Akzeptable Antwortzeiten, Anzahl gleichzeitiger Transaktionen), zweckmäßig (für den Anwendungsfall optimierter Zugriff (bspw. einfache Abfragesprache, Erstellung komplexer Abfragen)), für den parallelen Zugriff vieler Anwender und Anwendungen (insbesondere Handling von gleichzeitigen Änderungen (Transaktionen, Synchronisation), verschiedene Sichten und skalierbare Performance bei Multi-User Zugriff).

Die Aufgaben von DBMS bestehen somit in der Beseitigung der oben aufgeführten Probleme und der Kontrolle der Daten, die sich in einer Datenbank (DB) befinden. Die DB stellt einen zusammengehörigen Datenbestand dar (z.B. alle Daten der Abteilung Personal). Ein DBMS kann mehrere Datenbanken verwalten:



Abbildung 3: DBMS können mehrere Datenbanken enthalten

Neben den eigentlichen Datenbanken enthält das DBMS auch eine Datenbasis zum Speichern von **Metadaten** ("Daten über Daten"), wie Informationen über die Datenstruktur, Zugriffsrechte, etc.

—
Datenbankmanagementsystem und Datenbanken bezeichnet man als **Datenbanksystem** (kurz: **DBS**).

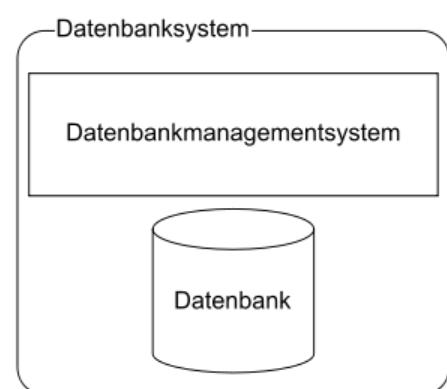


Abbildung 4: DBMS und DBs bilden zusammen das Datenbanksystem (DBS)

2.2 Datenabstraktion

Man unterscheidet drei Abstraktionsebenen im Datenbanksystem (Abbildung 5):

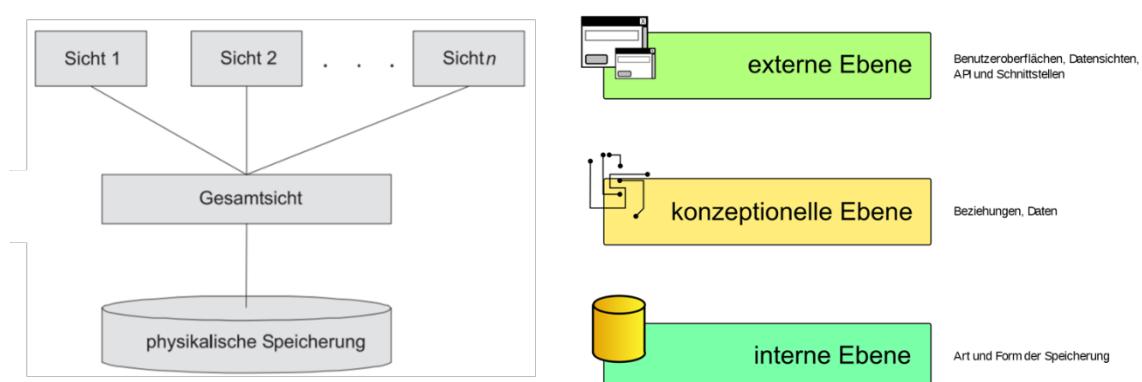


Abbildung 5: Abstraktionsebenen in Datenbanksystemen (ANSI/SPARC-Architektur)

2.2.1 Externe Ebene

In dieser Ebene wird für jede Benutzergruppe eine spezielle anwendungsbezogene Sicht der Daten („View“) spezifiziert. Die Beschreibung erfolgt mittels einer DDL, Änderungen in den Daten durch Benutzer erfolgt durch eine DML (data manipulation language). Ergebnis ist das externe Schema.

2.2.2 Konzeptionelle Ebene

In dieser Ebene wird unabhängig von allen Applikationen die Gesamtheit aller Daten, deren Strukturierung sowie Beziehungen spezifiziert. Die Formulierung erfolgt mittels einer DDL (data definition language). Das Ergebnis ist das konzeptionelle Schema (auch Datenbankschema genannt).

2.2.3 Interne Ebene

In dieser Ebene wird festgelegt, in welcher Form die logisch beschriebenen Daten im Speicher abgelegt werden. Geregelt werden Record-Aufbau, Darstellung der Datenbestandteile, Dateiorganisation, Zugriffspfade etc. Für einen effizienten Entwurf werden statistische Informationen über die Häufigkeit der Zugriffe benötigt. Ergebnis ist das interne Schema.

Das Datenbankschema legt also die Struktur der Daten fest und sagt noch nichts über die individuellen Daten aus. Die Datenbankausprägung stellt den momentan gültigen Zustand der Datenbasis, die den im Schema festgelegten Strukturbeschreibungen folgen muss, dar.



Empfohlene weiterführende Lektüre: (Elmasri und Navathe 2014), Kapitel 2.2

2.3 Komponenten eines DBMS

Abbildung 6 zeigt die typischen Komponenten von DBMS:

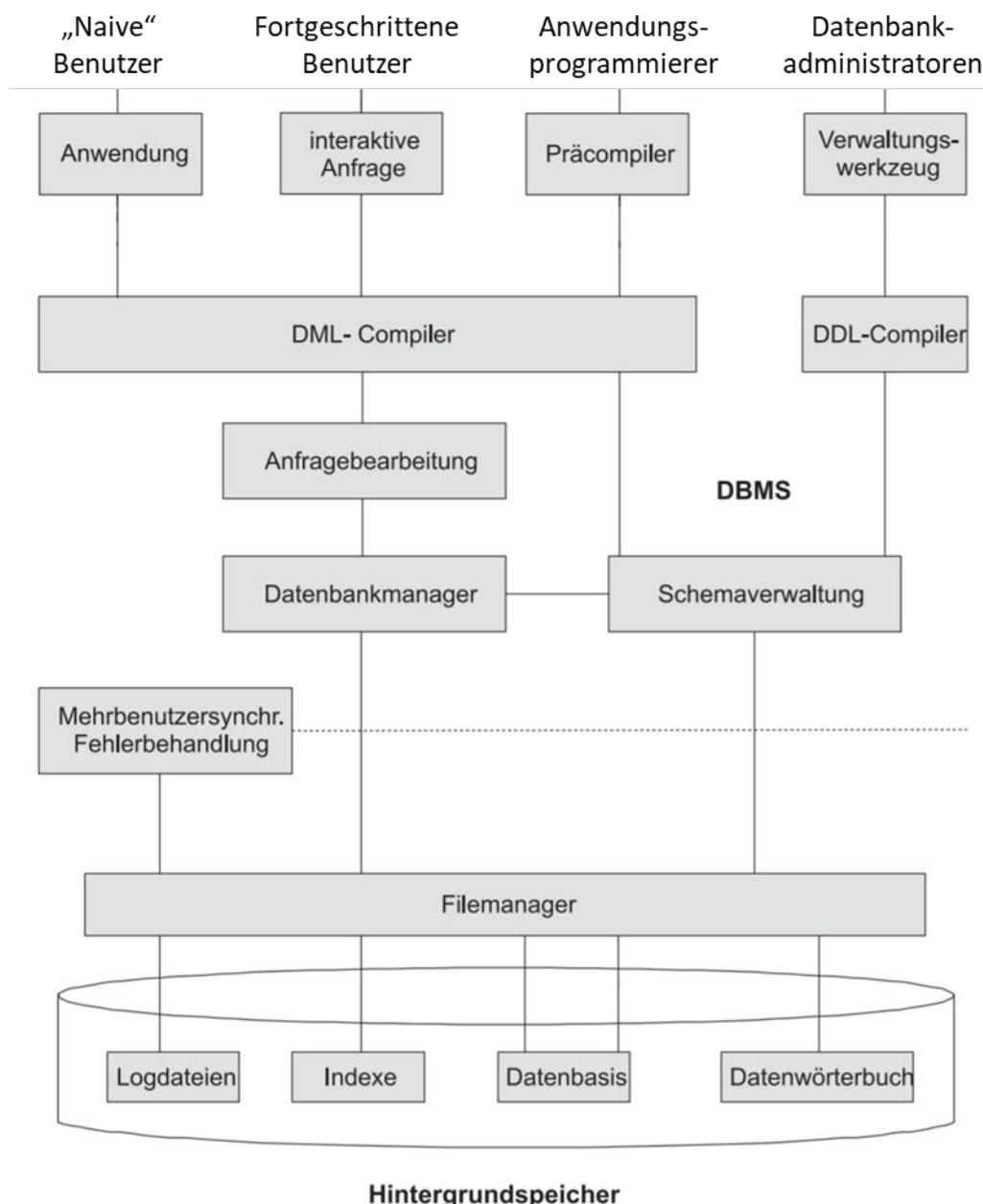


Abbildung 6: Komponenten von DBMS

Quelle: (Vornberger und Haldenwang 2015, S. 10)

Hinweis: Unterschied Casual / Parametric Users:

“Casual end users occasionally access the database, but they may need different information each time. They use a sophisticated database query language to specify their requests and are typically middle- or high-level managers or other occasional browsers.”

Naive or parametric end users make up a sizable portion of database end users. Their main job function revolves around constantly querying and updating the database, using standard types of queries and updates—called canned transactions—that have been carefully programmed and tested.” (Elmasri und Navathe 2014, S. 15)



Empfohlene weiterführende Lektüre: (Elmasri und Navathe 2014, S. 40–43)

2.4 Arten von DBMS

Es gibt sehr viele verschiedene Arten von DBMS. Hier ein paar typische Beispiele:

- Netzwerk-DBMS und Hierarchische DBMS (veraltet, nur noch in Legacy-Systemen)
- Relationale DBMS (auch RDBMS), marktbeherrschend, z. B. als Grundlage aller wichtigen ERP-Systeme.
- Objektorientierte DBMS (auch OODBMS) haben sich bislang kaum durchgesetzt.
- Manche Hersteller sprechen von "objektrelationalen DBMS". Dabei handelt es sich um relationale DBMS, die um Funktionen und Schnittstellen für die Objektverarbeitung erweitert wurden. Sie sind keine vollwertigen OODBMS, dafür aber meist viel performanter als echte OODBMS.
- Sogenannte NoSQL-Datenbanken kommen immer häufiger zum Einsatz. Sie werden insbesondere bei Anwendungen mit sehr großen Datenmengen verwendet, um horizontale Skalierung zu erreichen (z.B. beim Verarbeiten von Sensordaten).
- Multidimensionale Datenbanken, z. B. als Basis für Data Warehouse und Business Intelligence Anwendungen
- In-Memory-Datenbanken, die den Arbeitsspeicher des Computers als Datenspeicher nutzen und damit wesentlich höhere Zugriffsgeschwindigkeiten bieten
- Graphdatenbanken, die darauf ausgelegt sind, Informationen über Graphen optimal abzulegen und durchsuchbar zu machen (bspw. für soziale Netzwerke); werden oft auch zu NoSQL gezählt
- ... und viele mehr

Abbildung 7 zeigt eine beispielhafte Systematisierung dieser verschiedenen Arten von DBMS:

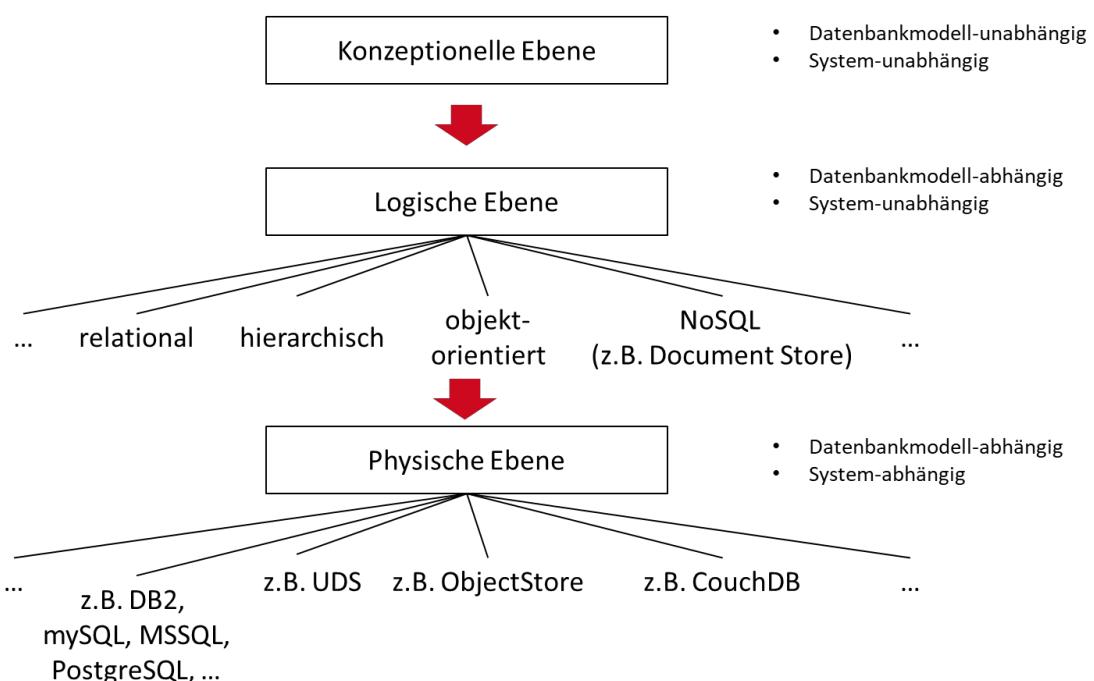


Abbildung 7: Systematisierung der verschiedenen Arten von DBMS

Wir befassen uns in dieser Veranstaltung hauptsächlich mit relationalen DBMS. Daher werden wir uns in den folgenden Kapiteln anschauen, wie zum einen die konzeptionelle Ebene modelliert werden kann, wie diese Modelle in die logische Ebene überführt werden

können und wie letztlich die logische Ebene in einem realen Datenbanksystem auf der physischen Ebene umgesetzt wird (mit PostgreSQL).

3 Datenbankmodellierung

3.1 Grundlagen

3.1.1 Zweistufiger Modellierungsansatz

Ein Datenbankdesigner muss gemeinsam mit dem Kunden zunächst den Zweck der zu erstellenden Datenbank und zukünftiger Anwendungssoftware diskutieren. Damit wird festgelegt, welche Daten als relevant anzusehen sind. Beispielsweise ist es technisch gesehen kein Problem, die Schuhgröße eines Kunden zu speichern. Doch macht dies in den abzubildenden Anwendungsfällen Sinn? Fachlich muss sich deshalb ein Datenbankdesigner immer wieder in die unterschiedlichsten Themengebiete eines Unternehmens einarbeiten, um die Datenbank und zuvor das Modell, auf dessen Grundlage die Datenbank angelegt wird, erstellen zu können. Bevor er also einen „Bauplan“ entwirft, führt er Gespräche mit den zukünftigen Endanwendern, sichtet Dokumente des Unternehmens, ermittelt Informationen anhand von Fragebögen u.ä. Ziel des Datenbank-Designers in dieser ersten Phase ist es also, das Geschäftsumfeld des Kunden mit den jeweiligen Funktionen und anfallenden Daten kennenzulernen.

Bei der darauffolgenden Datenbankmodellierung wird ein zweistufiges Vorgehen wie in Abbildung 8 dargestellt unterschieden.

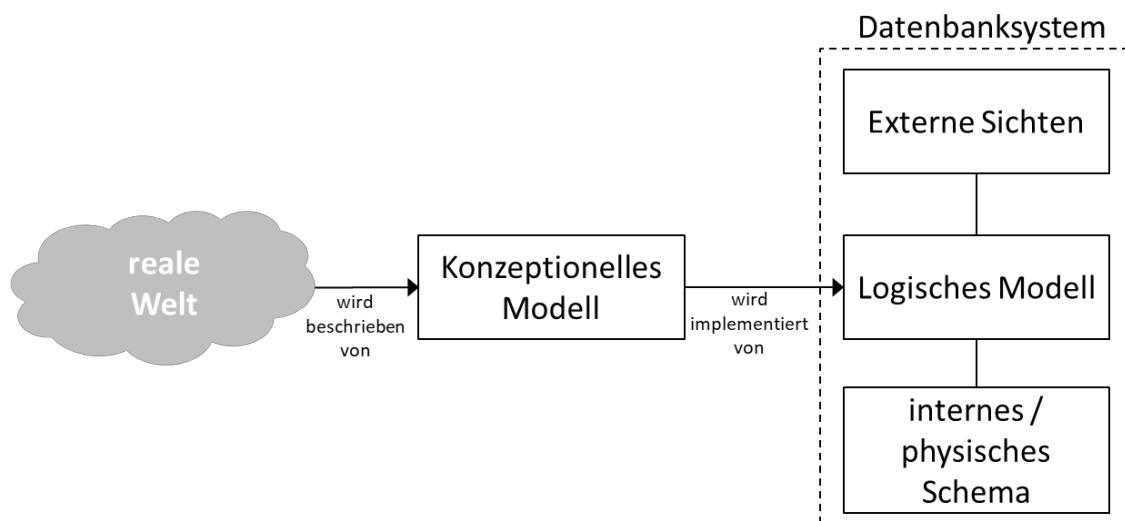


Abbildung 8: Zweistufiger Ansatz der Datenbankmodellierung

Das konzeptionelle Modell soll sowohl die reale Welt unabhängig von Datenverarbeitungs (DV)-Aspekten mit angemessener Genauigkeit beschreiben, als auch die Grundlage für das interne Datenbankschema bilden. Das interne / physische Schema ist stark DV-abhängig. Diese Herausforderung wird gelöst, indem dem konzeptionellen Modell ein logisches Modell zur Seite gestellt wird. Dieses beschreibt die Gesamtheit der Daten zwar hardwareunabhängig, berücksichtigt jedoch bestimmte Gesichtspunkte der Implementierung. Das logische Modell wird daher auch als implementiertes konzeptionelles Modell bezeichnet. Es übernimmt die Rolle des konzeptionellen Modells, das nicht Teil des eigentlichen

Datenbanksystems ist. Dieses steht also „daneben“ und kann auch aufgestellt werden, wenn kein Datenbanksystem zum Einsatz kommt.

3.1.2 Konzeptionelle Modellierung

Nachdem der Datenbankdesigner die Geschäftsanforderungen verstanden hat fängt er vorbereitend für das zweistufige Vorgehen also an, einen ersten „Bauplan“ für das zukünftige System zu erstellen. Dazu identifiziert er die Daten der Geschäftsobjekte, die für das Geschäftsumfeld relevant sind, und deren Beziehungen untereinander.

Beispiel: bei dem einfachen Geschäftsumfeld „Kunde kauft Auto“ stellen „Kunde“ und „Auto“ Geschäftsobjekte dar und „kauft“ beschreibt die Beziehung zwischen diesen beiden Geschäftsobjekten.

Der erste Entwurf muss erneut mit dem Kunden diskutiert werden (alle Akteure erfasst, Beziehungen richtig verstanden usw.). So können Missverständnisse schnell ausgemacht werden. Es scheitern viele IT-Projekte (bzw. dauern deutlich länger und werden teurer) daran, dass diese Missverständnisse erst spät oder am Ende des Projektes auffallen.

Um eine eindeutige „Sprache“ für diese Diskussion mit Kunden zu finden, wird auf die **konzeptionelle Modellierung** zurückgegriffen. Bei dieser hat sich als grafische Notation das Entity-Relationship-Modell (ERM) etabliert. Das ERM wird heutzutage in der Regel für den Entwurf eines Datenmodells verwendet und hat sich in der Praxis durchgesetzt, da es ohne viel Einarbeitungsaufwand schnell verstanden werden kann. Gerade dieser Punkt ist wichtig, um sich bei den Gesprächen mit dem Kunden auf die fachspezifischen Inhalte konzentrieren zu können und Missverständnisse bei den grafischen Notationen zu vermeiden.¹

Das im ersten Schritt erstellte Datenmodell wird wie bereits dargestellt **konzeptionelles Modell (oder Schema)** genannt und ist unabhängig von bestimmten Datenbankmanagementsystemen. Es muss allerdings ausreichend detailliert sein, um von Datenbankdesigner in ein logisches und letztlich physisches Modell umgewandelt werden zu können.

3.1.3 Logische Modellierung

Nach dem Entwurf des konzeptionellen Datenmodells ist ein weiterer „Bauplan“ notwendig, das so genannte logische Datenmodell. Dieses stellt die Grundlage für die Umsetzung des konzeptionellen Modells in eine physische Datenstruktur dar. Das logische Datenmodell legt fest, wie die im konzeptionellen Modell definierten Geschäftsobjekte und Beziehungen vom Datenbankmodell (relational, hierarchisch, objektorientiert) her abgebildet werden sollen. Es ist also unabhängig von einem konkreten Datenbankmanagementsystem, aber abhängig von der Art der Datenspeicherung. In relationalen Datenbanksystemen erfolgt die Speicherung von Geschäftsobjekten und Beziehungen in Form von Tabellen.

Im physischen Datenmodell können noch Zugriffsmechanismen, Datei- und Indexstrukturen usw. festgelegt werden. Dies wird im Rahmen der Veranstaltung nicht weiter behandelt.

¹ Aus diesem Grund hat sich die UML für die abstrakte Modellierung von Datenbankentwürfen nicht so stark durchgesetzt. Im Gegensatz zur UML hat das ERM allerdings den Nachteil, dass es nicht standardisiert wurde. Dadurch existieren zurzeit verschiedene „Dialekte“.

Zusammengefasst ergibt sich folgender Ablauf der Aufgaben eines Datenbankdesigners. Die einzelnen Schritte sollten iterativ wiederholt werden, um zu verhindern, dass Fehler erst in späten Projektphasen auffallen.

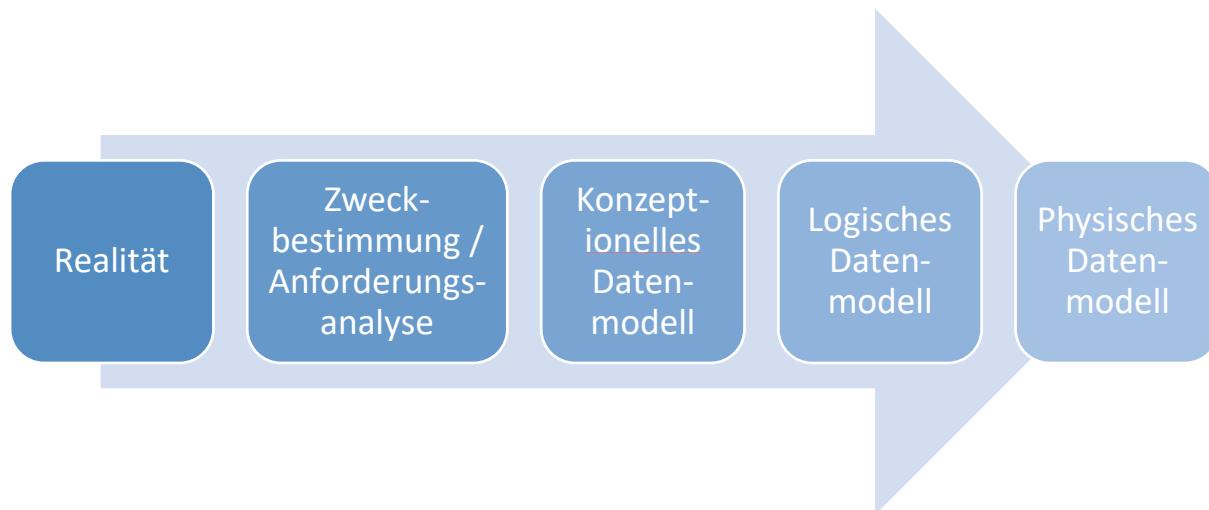


Abbildung 9: Vorgehen in der Datenbankmodellierung

4 Konzeptionelle Modellierung

4.1 Zielgruppe, Notationen und Elemente

Der konzeptionelle Datenbankentwurf wird für zwei Personengruppen erstellt:

- Den **Analytiker**. Dieser verwendet den „Bauplan“ für die Besprechung mit dem Kunden.
- Den **Datenbankdesigner**. Dieser nutzt den konzeptionellen Entwurf als Grundlage für die Umsetzung in ein logisches und anschließend in ein physisches Modell (nicht relevant für uns).

Das Datenmodell besteht aus:

- Strukturiertem Text,
- grafischer Notation:
 - o Geschäftsobjekte,
 - o Attribute,
 - o eindeutige Identifizierer (Schlüssel),
 - o Beziehungen,
 - o Beziehungstypen,
 - o Sub- bzw. Supertypen.

In der Praxis existieren verschiedene Notationen, um den konzeptionellen Datenbankentwurf umzusetzen:

- Entity-Relationship-Model (ERM) nach Chen,
- Entity-Relationship-Model (ERM) nach Barker,
- Unified Modeling Language (UML), kennen Sie bereits aus der Modellierungs-Vorlesung
- ...

Im Folgenden widmen wir uns ausschließlich dem ERM nach Chen. Dies erfolgt auf Basis eines durchgehenden Fallbeispiels.

4.2 Fallbeispiel Online-Lebensmittelhandel

Ein Unternehmen möchte in den Markt der Online-Supermärkte einsteigen und von Ihnen einen entsprechenden Online-Shop entwickeln lassen. Grundlage hierfür stellt eine entsprechende Datenbank dar.

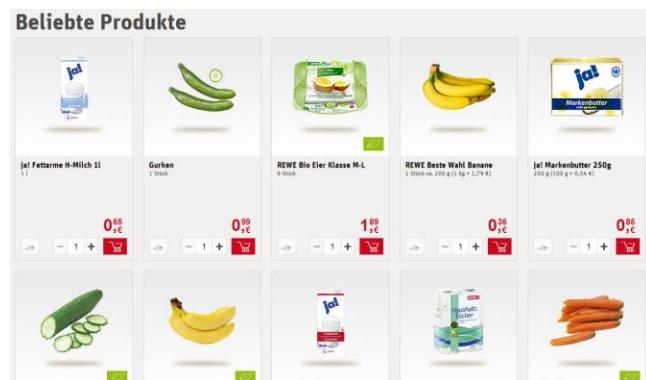


Abbildung 10: Fallbeispiel Online-Lebensmittelhandel - User-Interface

Für den Online-Shop stellt das Unternehmen die folgenden Anforderungen:

- Das System wird von Benutzern (Kunden & Mitarbeiter), die durch ihre E-Mailadresse identifiziert werden, bedient.
 - o Alle Benutzer registrieren sich mit Name und Vorname.
 - o Kunden sollen darüber hinaus in der Lage sein, bei der Registrierung ihre Adresse (Str., Hausnummer, PLZ, Ort) und ein Geburtsdatum zu hinterlegen. Für Kunden wird automatisch eine ID angelegt.
 - o Mitarbeiter hinterlegen ihre eindeutige Personal- und Tel.-Nr. Zusätzlich sollen Sie ihrer Abteilung in einer Abteilungshierarchie zugeordnet werden.
- Registrierte Kunden sollen Produkte in ihren Warenkorb „legen“ können (auch mehr als 1x).
 - o Dabei bekommt jeder Warenkorb eine eindeutige ID und das Erstelldatum wird auch gespeichert.
 - o Produkte haben eine eindeutige Nummer und in der Datenbank sollen zusätzlich die Bezeichnung, die aktuelle Verfügbarkeit, der Listenpreis und das Anlagedatum des Produkts in den Stammdaten gespeichert werden.
 - o Produkte gehören zu Kategorien; diese wiederum haben eine Hierarchie (bspw. Banane → Obst → Lebensmittel).
- Wenn ein Kunde einen Warenkorb bestellen möchte, wird eine entsprechende Bestellung im System angelegt und dem Warenkorb zugeordnet.
 - o So kann eine Historie der Bestellungen aller Kunden aufgebaut werden. Bestellungen setzen einen Warenkorb voraus, in dem die zu bestellenden Produkte und deren Bestellmenge hinterlegt werden.
 - o Bestellungen werden von einem Mitarbeiter bearbeitet und der Kunde kann sich über die Website über den aktuellen Status seiner Bestellung informieren.

Ziel der Veranstaltung ist es, die Datenbank für den Online-Shop zu designen (konzeptioneller und logischer Entwurf), per SQL anzulegen, zu befüllen und die Daten anzeigen zu lassen (um betriebswirtschaftliche Fragestellungen zu beantworten). Abschließend soll mit JDBC und Java die Benutzerverwaltung des Online-Shops prototypisch entwickelt werden.

4.3 Geschäftsobjekte

Wichtiger Bestandteil eines ERM sind die Geschäftsobjekte (allgemeiner: Objekte). Diese sind Elemente, die **eindeutig identifiziert** und von anderen Objekten anhand ihrer **Eigenschaften** unterschieden werden können (ähnlich Klassen im Klassendiagramm). Ein Geschäftsobjekt stellt somit eine „**Schablone**“ dar, um gleichartige Dinge und deren Informationen zusammenzufassen. Dies kann z. B. sein:

- Etwas real Existierendes: Person, Auto, Buch
- ein Ereignis: Artikel bestellen, Gespräch führen
- eine Rolle oder Person: Arzt, Mitarbeiter,
- eine Organisation,
- eine Transaktion: Kaufvertrag, Warenausgang, Wareneingang (vgl. Ereignis).

In unserem **Fallbeispiel** können wir die folgenden Geschäftsobjekte definieren:

- Benutzer
- Kunde
- Mitarbeiter
- Produkt
- Bestellung
- Warenkorb
- Kategorie
- Abteilung
- Status

4.4 Sub- bzw. Supertypen

Diese kommen zum Einsatz, wenn das Vererbungsprinzip (bekannt aus Programmieren 2) angewandt wird.

Ein Hund ist ein Lebewesen.

Ein Mensch ist ein Lebewesen.



Subtyp Supertyp

4.5 Attribute und Schlüssel

Mit Attributen können Objekttypen **beschrieben** werden, soweit notwendig.

In unserem Fallbeispiel könnten die folgenden Attribute zum Einsatz kommen (nur ein Auszug, eine beispielhafte Musterlösung wird in der Veranstaltung erarbeitet):

Geschäftsobjekt	Attribute
Nutzer	E-Mail, Passwort, Vorname, Nachname
Kunde	Kundennummer, Straße, PLZ, Ort, Geburtsdatum
Mitarbeiter	Vorname, Nachname, E-Mail, Passwort
Artikel	Artikelnummer, Name, Beschreibung, Gewicht, Preis
Bestellung	Datum
Warenkorb	Erstelldatum

Nach der Bestimmung der Attribute für die Geschäftsobjekte ist es wichtig herauszufinden, welche Attribute bzw. auch Kombinationen aus Attributen notwendig sind, um diese **eindeutig zu identifizieren**. Dieses identifizierende Attribut bezeichnet man als **Schlüssel**. Gibt es mehrere Schlüssel, spricht man von **Schlüsselkandidaten**. In diesem Falle muss ein Schlüsselkandidat ausgewählt werden, der als so genannter **Primärschlüssel** fungieren soll und über den ein Objekt eines Objekttyps immer eindeutig identifiziert wird. Ein Schlüssel kann auch aus mehreren Attributen **zusammengesetzt** sein, generell gilt jedoch, dass ein Primärschlüssel eine minimale Anzahl von Attributen enthalten sollte.

In Bezug auf unser Fallbeispiel könnte der Schlüssel für einen Mitarbeiter die E-Mail sein, für den Artikel die Artikelnummer. Ist kein (zusammengesetztes) Attribut ein geeigneter Schlüsselkandidat, kann auch ein künstlicher Schlüssel (ID) vergeben werden (z. B. als Kundennummer für den Kunden):

Geschäftsobjekt	Attribute & <u>Schlüssel</u>
Filiale	<u>Straße, PLZ, Ort</u>
Online-Shop	<u>ID</u> , Name
Benutzer	<u>E-Mail</u> , Vorname, Nachname, Passwort
Kunde	Kundennummer, Straße, PLZ, Ort, <u>E-Mail</u>
Mitarbeiter	<u>E-Mail</u> , Tel.-Nr.
Artikel	<u>Artikelnummer</u> , Name, Beschreibung, Gewicht, Preis
Bestellung	<u>ID</u> , Datum
Warenkorb	<u>ID</u> , Erstelldatum
Abteilung	<u>AbteilungID</u> , Bezeichnung
Kategorie	<u>KategorieID</u> , Bezeichnung

4.6 Beziehungen und Beziehungstypen

Beziehungen sind **Assoziationen** zwischen Objekttypen (z. B. Mann <-> Frau). Hierbei spielt (wie auch bei Klassendiagrammen) die Beziehungsrichtung eine wichtige Rolle.

In unserem Fallbeispiel ergeben sich die folgenden Beziehungen:

- Bestellung <-> Warenkorb

- Warenkorb <-> Artikel
- Warenkorb <-> Kunde

Bei Beziehungen ist auch die sogenannte „**Kardinalität**“ zu definieren. Diese gibt an, wie viele Objekte eines Objekttyps in Beziehung zu einem Objekt des anderen Objekttyps stehen. Beispiele:

Objekt 1	Objekt 2	Kardinalität
Abteilung , die aus 1 Mitarbeiter besteht	Mitarbeiter , der nur in 1 Abteilung arbeitet	1:1 = $(0, 1) : (1, 1)$
Abteilung , die aus mehreren Mitarbeitern besteht	Mitarbeiter , der nur in 1 Abteilung arbeitet	1:N $(0, n) : (1, 1)$
Kunde kauft beliebig viele Artikel	Artikel wird von beliebig vielen Kunden gekauft	N:M $(0, n) : (0, m)$

Für unser Fallbeispiel ergeben sich die folgenden Kardinalitäten (nur ein Auszug!):

Objekt 1	Objekt 2	Kardinalität
Bestellung hat einen zugeordneten Warenkorb	Ein Warenkorb gehört zu maximal einer Bestellung	1:1
Kunde hat u.U. mehrere Warenkörbe	Ein Warenkorb gehört zu genau einem Kunden	N:1
Warenkorb enthält mehrere Artikel	Artikel ist in mehreren Warenkörben	N:M
Warenkorb gehört zu genau einem Kunden	Ein Kunde hat genau einen Warenkorb	1:1

4.7 Entity-Relationship-Modell nach Chen

4.7.1 Allgemein

Das Modell von Chen ist leicht zu verstehen, begnügt sich mit den wichtigsten grafischen Elementen und hat sich gerade deshalb in der Kommunikation mit dem Kunden durchgesetzt. Außerdem ist es auch leicht in ein logisches Modell für relationale Datenbanken umzusetzen.

4.7.2 Geschäftsobjekte

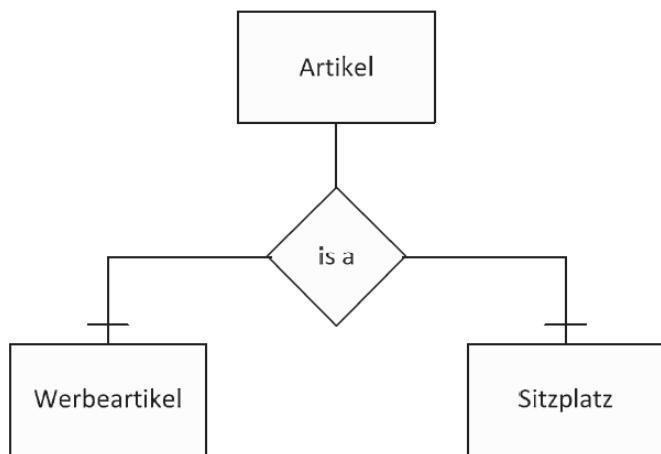
Geschäftsobjekte werden nach Chen folgendermaßen dargestellt:



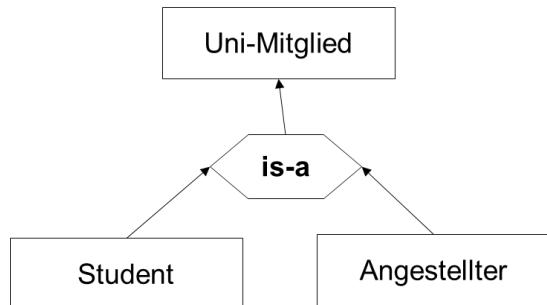
4.7.3 Sub- bzw. Supertypen (=Generalisierung bzw. Spezialisierung)

Für Sub- bzw. Supertypen gab es ursprünglich keine grafische Notation. Sie wurde erst im Laufe der Jahre nachträglich hinzugefügt.

Möglichkeit 1:

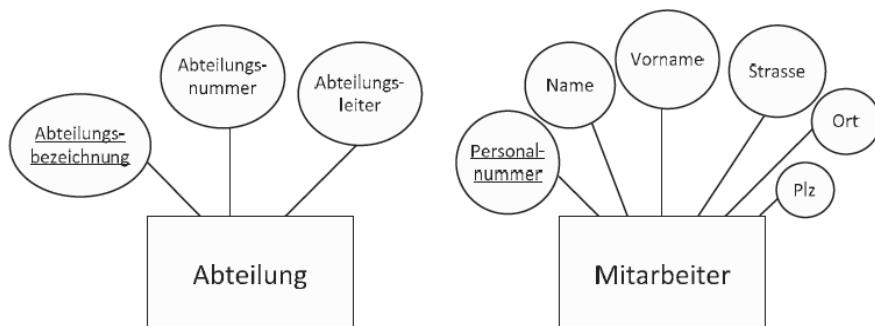


Möglichkeit 2:



4.7.4 Attribute

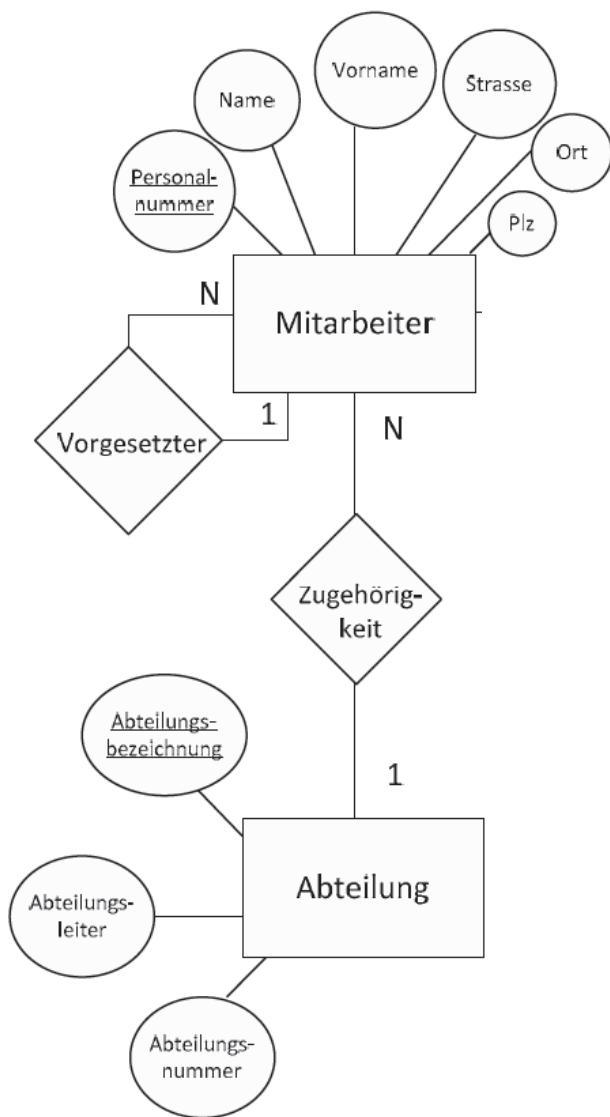
Attribute werden in Form eines Kreises an den jeweiligen Entitätstyp gehängt:



Die unterstrichenen Attribute stellen Primärschlüssel dar.

4.7.5 Beziehungen und Beziehungstypen

Beziehungen werden über eine Raute dargestellt. Beziehungen können theoretisch auch Attribute zugeordnet bekommen. Die Kardinalitäten einer Beziehung werden durch die Zeichen 1, N und M gekennzeichnet:



Besonderheiten: Mitarbeiterhierarchie / Vorgesetzte (s.u.)

4.7.6 Hinweis zu Kardinalitäten

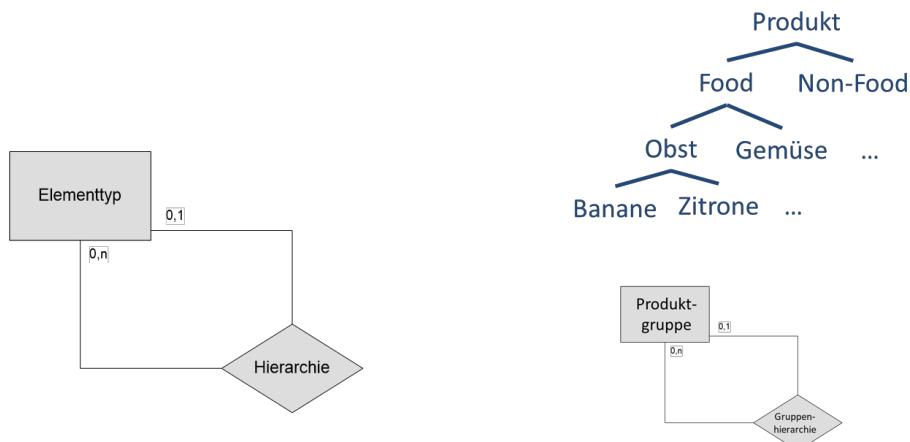
Die Chen-Notation ist semantisch ärmer als die min,max-Notation. Sie lässt keine Aussagen über die minimale Anzahl korrespondierender Entitäten zu. Dies ist bei der Umwandlung in ein logisches Modell u.U. von Bedeutung.

4.7.7 Hierarchien

Hierarchien setzen Entities in eine Über- /Unterordnungsbeziehung, d.h. es werden Baumstrukturen abgebildet. Somit darf einem Element maximal ein anderes Element (gleichen Typs) übergeordnet werden, einem Element können aber beliebig viele Elemente (gleichen Typs) untergeordnet werden.

→ Kardinalität (0,1) – (0,n)

Beispiel „Produktgruppenhierarchie“:

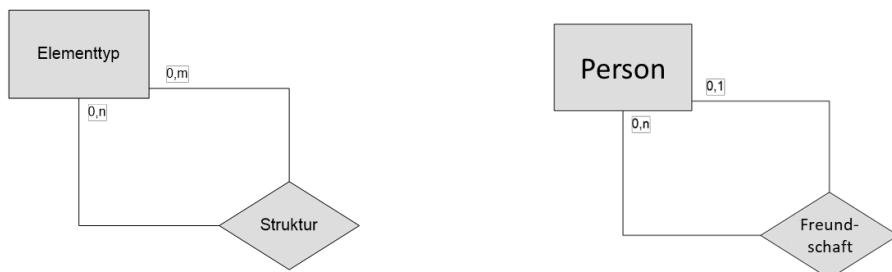


4.7.8 Strukturen

Strukturen im ERM setzen Entities mit beliebig vielen Entities gleichen Typs in Verbindung. Keine klare Richtung der Beziehungen notwendig.

→ Kardinalität: (0, n) – (0, m)

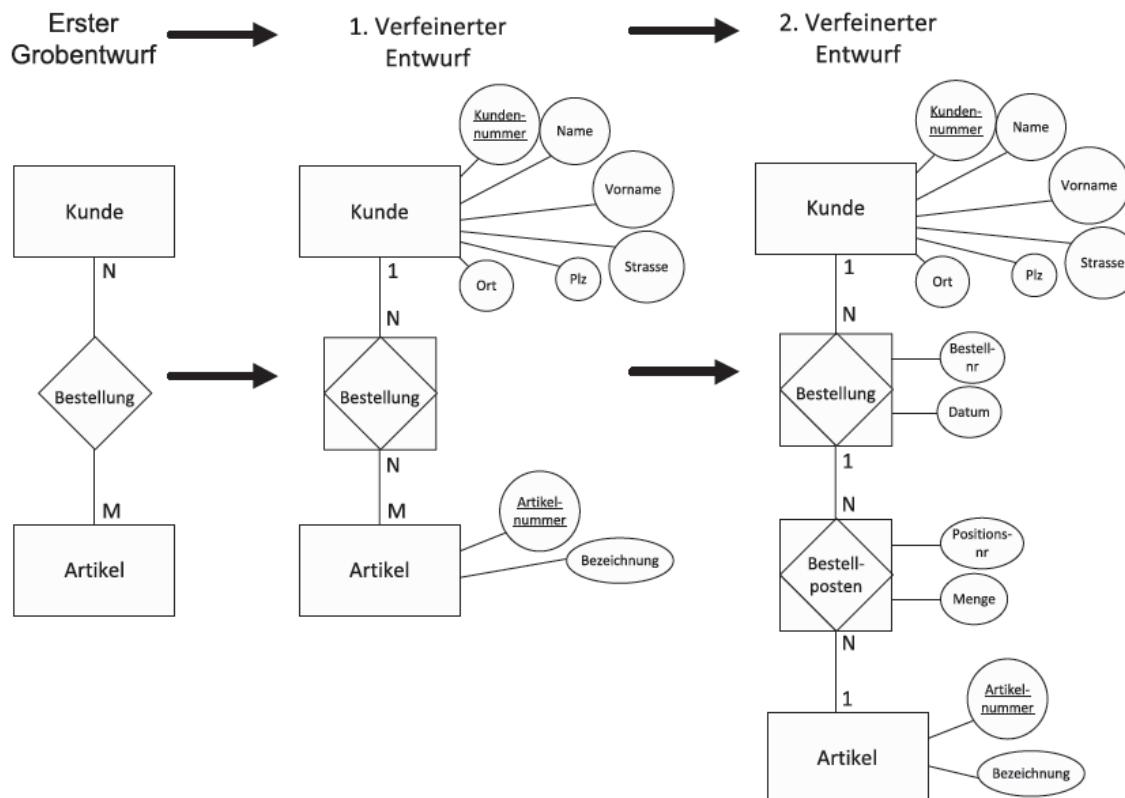
Beispiel „soziales Netzwerk“



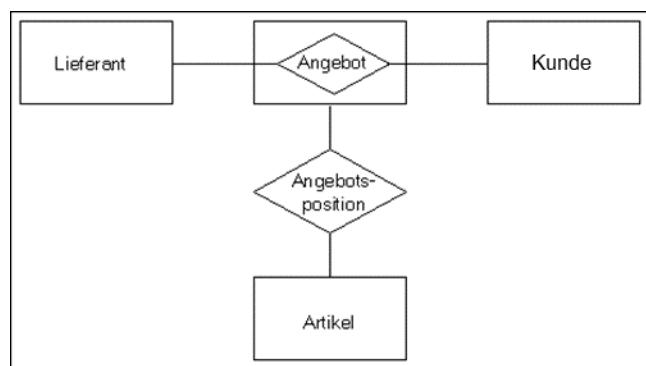
4.7.9 N:M-Beziehungen / Uminterpretation

Ein **Sonderfall** können **N:M-Beziehungen** darstellen. Im unten aufgeführten Beispiel soll abgebildet werden, dass ein Kunde eine oder mehrere Bestellungen aufgeben kann. Darüber hinaus besteht eine Bestellung aus mehreren Bestellpositionen. Sowohl für die Bestellung, als auch für die Bestellposition sollen Attribute hinterlegt werden (Datum, Menge). Dementsprechend müssen, insbesondere je detaillierter man modelliert bzw. je näher man sich der logischen Modellierung nähert, N:M-Beziehungen in zwei 1:N-Beziehungen umdefiniert werden. Dies geschieht grafisch, indem man um die Beziehungsraute ein

Rechteck zeichnet. Dadurch wird erkennbar, dass es sich nun um einen als Entitätstypen uminterpretierten Relationship-Typen, der aus einer N:M-Beziehung hervorgegangen ist.



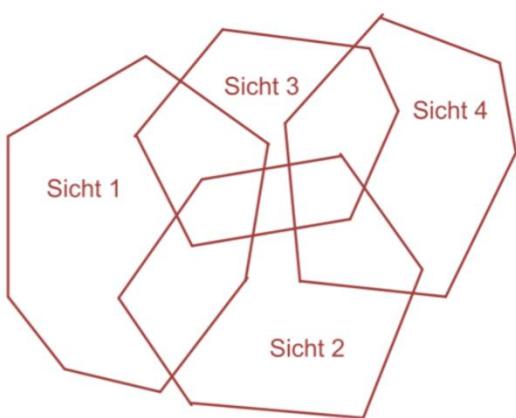
Eine derartige Uminterpretation der Relationship-Typen kann im ERM auch auf diese Weise modelliert werden:



Dies erlaubt nicht nur die Nutzung von Attributen an Relationship-Typen, sondern auch deren „In-Beziehung-Setzen“ zu anderen Entity-Typen. Dies werden wir aber nicht vertiefen.

4.7.10 Modellkonsolidierung

Bei der Modellierung komplexer Domänen bietet es sich an, den konzeptuellen Entwurf zunächst aus verschiedenen Anwenderperspektiven (Sichten) zu modellieren. Nachdem diese modelliert sind, müssen sie zu einem globalen Schema zusammengefasst werden () .



Globales Schema

- Ohne Redundanzen
- Ohne Widersprüche
- Synonyme bereinigt
- Homonyme bereinigt

Abbildung 11: Konsolidierung einzelner Sichten

Die Datenbestände der verschiedenen Anwender überlappen sich teilweise, weshalb es nicht genügt, die einzelnen konzeptionellen Modelle der Sichten zu vereinen. Stattdessen müssen diese konsolidiert werden, d.h. Redundanzen und Widersprüche entfernt werden. Widersprüche entstehen insbesondere durch Synonyme (gleiche Sachverhalte wurden unterschiedlich benannt), durch Homonyme (unterschiedliche Sachverhalte wurden gleich benannt) und durch unterschiedliches Modellieren desselben Sachverhalts. Letzteres entsteht bspw. über Beziehungen, zum anderen über Attribute. Bei der Zusammenfassung von ähnlichen Entity-Typen zu einem Obertyp bietet sich die Generalisierung an.

Eine beispielhafte Konsolidierung ist in Abbildung 12 und Abbildung 13 dargestellt:

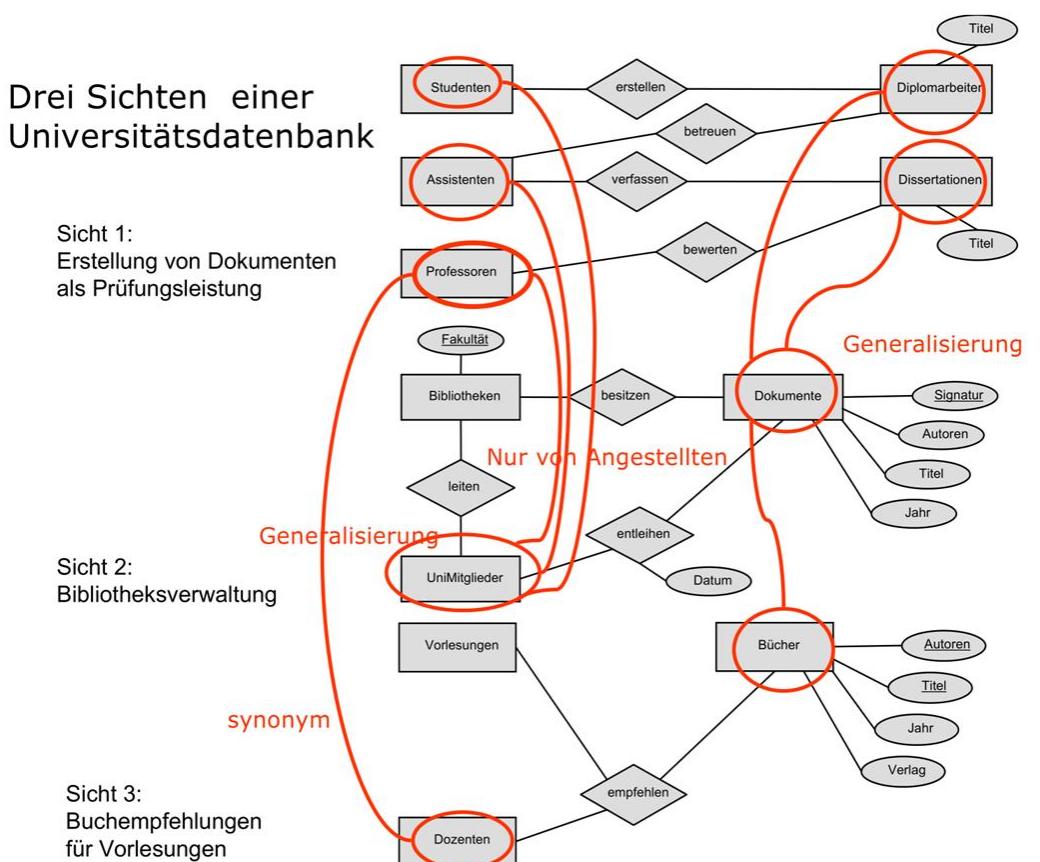


Abbildung 12: Nicht-konsolidiertes Beispiel-ERM

Quelle: (Vornberger und Haldenwang 2015, S. 18)

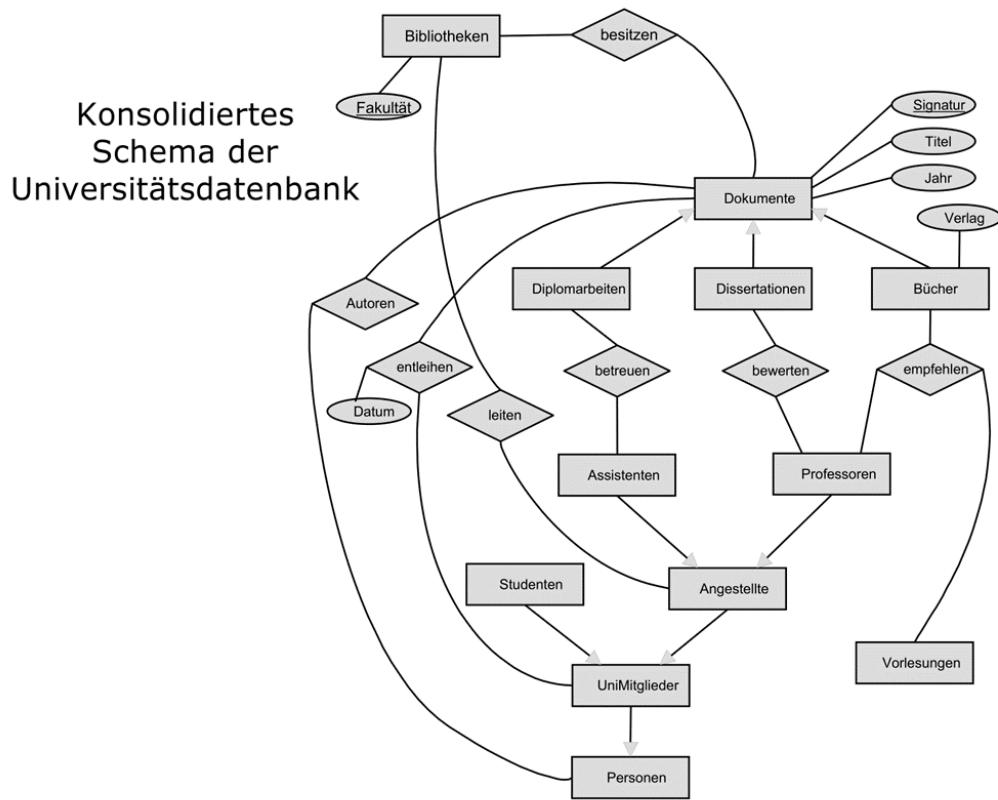


Abbildung 13: Nicht-konsolidiertes Beispiel-ERM

Quelle: (Vornberger und Haldenwang 2015, S. 18)

5 Literaturverzeichnis

Elmasri, Ramez; Navathe, Sham (2014): Fundamentals of database systems. 6. ed., Pearson new international ed. Harlow: Pearson Education.

Vornberger, Oliver; Haldenwang, Nils (2015): Datenbanksysteme. Vorlesungsskript. Universität Osnabrück, Osnabrück. Institut für Informatik. Online verfügbar unter <http://www-lehre.inf.uos.de/~dbs/2015/PDF/dbs-2015-media2mult.pdf>.

**BW342 – Datenbanken I
(mit Praktikum)**
Sommersemester 2019

Veranstaltungsüberblick

Überblick & Ablauf der Veranstaltung „Datenbanken“



Prof. Dr. Peer Küppers

E-Mail: peer.kueppers@hs-lu.de

Tel.: +49 (0)621 5203 - 416, Raum E1020

Veranstaltungszeiten (3 SWS):

Donnerstag, 10:00-11:30 Uhr UND 12:30-14:00, Raum E1106

Ausnahmen: siehe OLAT

Vorlesung & Praktikum (Übungen)

Bei Fragen:

- Nutzen Sie bitte das Forum in OLAT
- Schreiben Sie mir eine E-Mail
- Vereinbaren Sie einen Termin per E-Mail

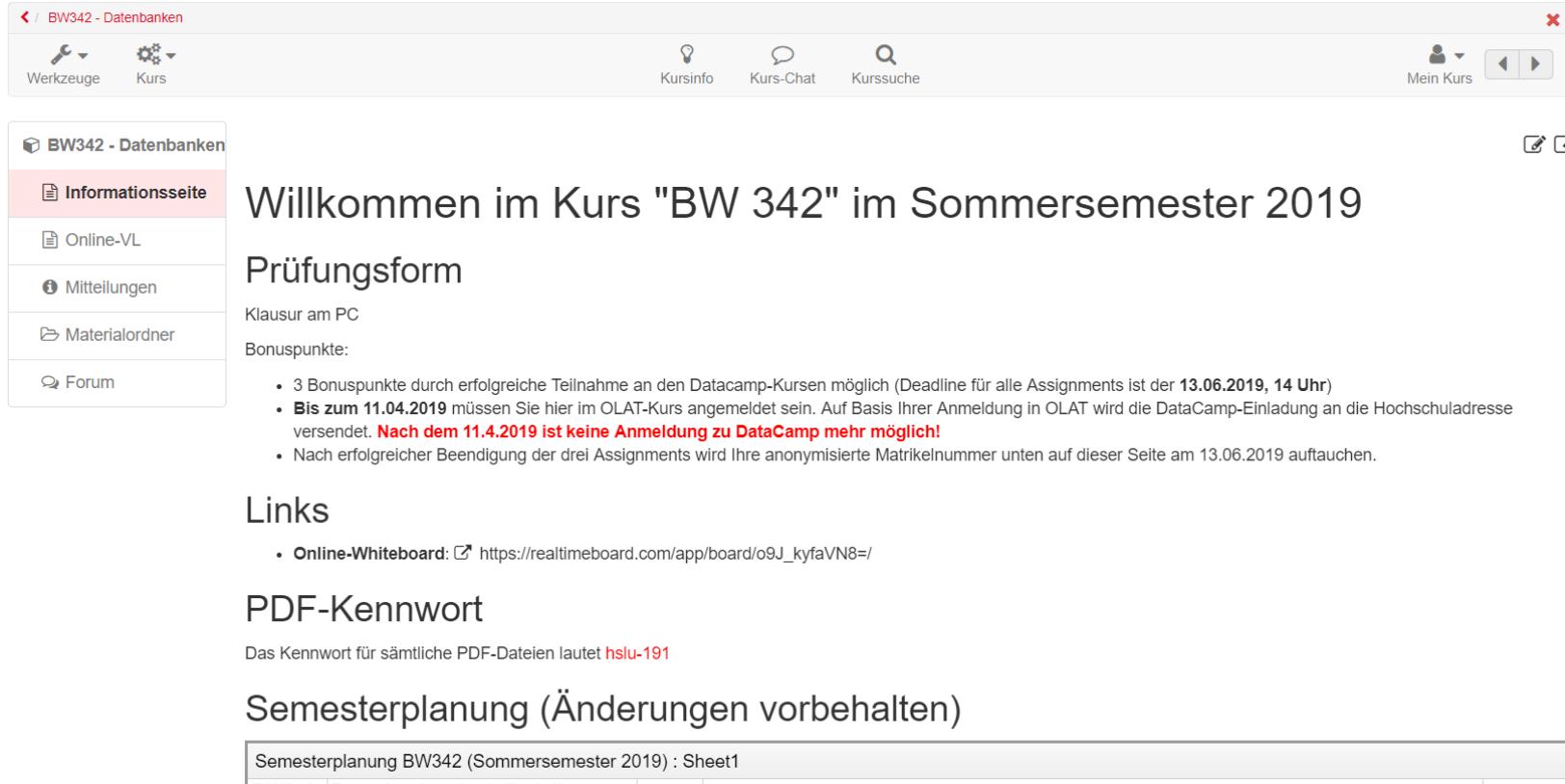
Überblick & Ablauf der Veranstaltung „Datenbanken“

Terminübersicht siehe OLAT, (Änderungen vorbehalten)

12 x 3 SWS = 36 SWS insgesamt

Semesterplanung (Änderungen vorbehalten)

Semesterplanung BW342 (Sommersemester 2019) : Sheet1						
BW342	Datenbanken I mit Praktikum					
Stunden	KW	Datum	Tag	Typ	Nr.	Thema
4	12	21.03.2019	Do	VL & UE	1&2	Überblick und Einführung, konzeptionelle Datenmodellierung
4	13	28.03.2019	Do	VL & UE	2&3	Rest VL 2, Logische Datenmodellierung (Relationales Modell), Normalisierung, Datenintegrität (Teil 1)
	14	04.04.2019	Do			Fällt aus (Ausgleich, da nur 3SWS-Veranstaltung)
4	15	11.04.2019	Do	VL & UE	3&4	Logische Datenmodellierung (Relationales Modell), Normalisierung, Datenintegrität (Teil 2)
4	16	18.04.2019	Do	VL & UE	5	Structured Query Language - SQL (Grundlagen & DDL)
4	17	25.04.2019	Do	VL & UE	6	Structured Query Language - SQL (DML)
	18	02.05.2019	Do	VL & UE		Fällt aus (Ausgleich, da nur 3SWS-Veranstaltung)
4	19	09.05.2019	Do	VL & UE	7	Structured Query Language - SQL (DQL1)
4	20	16.05.2019	Do	VL & UE	7&8	Structured Query Language - SQL (DQL2)
4	21	23.05.2019	Do	VL & UE	8	Structured Query Language - SQL (DQL3)
	22	30.05.2019	Do			Feiertag
4	23	06.06.2019	Do	VL & UE	10	Einbindung von DBMS in Anwendungen mit JDBC + Klausurvorbereitung
	24	13.06.2019	Do	VL & UE	11	Fällt aus (Ausgleich, da nur 3SWS-Veranstaltung)
	25	20.06.2019	Do			Feiertag
26	27.06.2019	Do		Start Prüfungen		
27	04.07.2019	Do				
28	11.07.2019	Do		Ende Prüfungen		
36	GESAMT					



The screenshot shows the OLAT course interface for "BW342 - Datenbanken". The top navigation bar includes links for "Werkzeuge" and "Kurs", and icons for "Kursinfo", "Kurs-Chat", "Kurssuche", and "Mein Kurs". On the left, a sidebar menu lists "Informationsseite" (highlighted in pink), "Online-VL", "Mitteilungen", "Materialordner", and "Forum". The main content area displays a welcome message: "Willkommen im Kurs \"BW 342\" im Sommersemester 2019" and "Prüfungsform Klausur am PC". It also mentions bonus points: "3 Bonuspunkte durch erfolgreiche Teilnahme an den Datacamp-Kursen möglich (Deadline für alle Assignments ist der **13.06.2019, 14 Uhr**)", "Bis zum **11.04.2019** müssen Sie hier im OLAT-Kurs angemeldet sein. Auf Basis Ihrer Anmeldung in OLAT wird die DataCamp-Einladung an die Hochschuladresse versendet. **Nach dem 11.4.2019 ist keine Anmeldung zu DataCamp mehr möglich!**", and "Nach erfolgreicher Beendigung der drei Assignments wird Ihre anonymisierte Matrikelnummer unten auf dieser Seite am 13.06.2019 auftauchen."

Links

- Online-Whiteboard: https://realtimeboard.com/app/board/o9J_kyfaVN8=/

PDF-Kennwort

Das Kennwort für sämtliche PDF-Dateien lautet **hslu-191**

Semesterplanung (Änderungen vorbehalten)

Semesterplanung BW342 (Sommersemester 2019) : Sheet1
BW342 - Datenbanken I mit Praktikum

Zugangskennwort: **db191**

PDF-Kennwort: **hslu-191**

Link: <https://olat.vcrp.de/url/RepositoryEntry/1870037000>

Ziele und Themen aus dem Modulhandbuch

Qualifikationsziele

„[...] Erlernen und Anwenden der Datenbankgrundlagen, Datenmodellierung, Implementierung von Datenmodellen, Datenabfragen und Datenmanipulation, Erlernen von grundlegenden Kenntnissen der Datenbanksicherheit, Erlernen und Anwenden der Einbindung von Datenbanken in betriebswirtschaftliche Anwendungssysteme“

Themen

- Datenbankmanagementsysteme
- Datenbankentwurf, Normalisierung, Integrität
- SQL: DDL, DML, DQL (*Data Definition Language, Data Manipulation Lang., Data Query Lang.*)
- Access Control und Transaktionen
- Katalogabfragen
- Einbindung von Datenbanken in Anwendungssysteme, z.B. mit JDBC

Workload BW342

126 h Gesamtworkload
36 h Kontaktzeit
90 h Selbststudium

Ziele der gesamten Veranstaltung (1/2)



- Das notwendige **theoretische Wissen** bezüglich **Datenbanken und Datenmodellierung** erwerben
 - Verschiedene Arten der konzeptionellen Modellierung
 - Überführung konzeptioneller in logische Datenbankmodelle (relationales Modell)
 - Normalisierung: Notwendigkeit und Anwendung
 - Datenintegrität und Transaktionen
- Fähigkeiten aufbauen, konzeptionelle Datenmodelle in ein **Datenbanksystem** zu überführen und mit diesem zu arbeiten (SQL)
 - Anlegen von Datenbankstrukturen (Data Definition Language, DDL)
 - Einfügen und Ändern von Datenbankeinträgen (Data Manipulation Language, DML)
 - Abfrage, Aggregation und Analyse von Datenbanken (Data Query Language, DQL)
 - → Beispielhafte Nutzung vom PostgreSQL Datenbankmanagementsystem (DBMS)

Ziele der gesamten Veranstaltung (2/2)

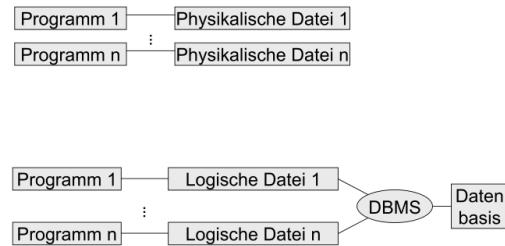


- Möglichkeiten der **Nutzung von Datenbanken** in (geschäftlichen) Anwendungen verstehen
 - Erste Schritte im Programmieren mit Datenbanken
 - → Beispielhafte Nutzung von JDBC (Java Database Connectivity)
 - Objektrelationales Mapping (ORM)
 - → Beispielhafte Nutzung in Java mit JPA (Java Persistence API)
- Festigung sämtlicher Inhalte durch deren **praktische Anwendung**
 - Übungen während der Vorlesung
 - Übungsblätter

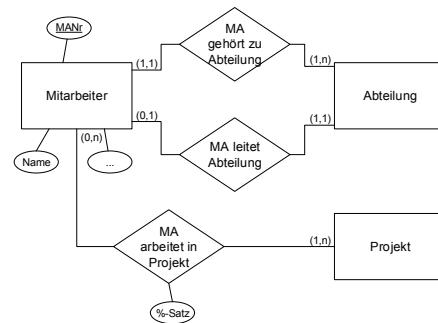
Themenüberblick



A Datenbankgrundlagen



B Konzeptionelle Modellierung



C Logische Modellierung

MID	Name	Adresse	Gehalt
1234	Müller	Hofstraße 7	23.000 €
1235	Meyer	Hauptstraße 13b	45.678 €
1236	Schmidt	Kurze Straße 1	55.555 €
1237	Heinze	Neue Straße 34	44.333 €
1238	Mustermann	Musterstraße 8	80.000 €
1239

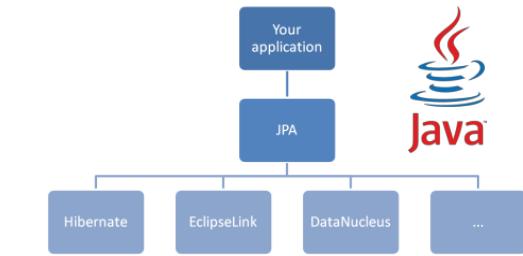
D Structured Query Language

```
-- filter your columns  
SELECT col1, col2, col3, ... FROM table1  
-- filter the rows  
WHERE col4 = 1 AND col5 = 2  
-- aggregate the data  
GROUP by ...  
-- limit aggregated data  
HAVING count(*) > 1  
-- order of the results  
ORDER BY col2
```

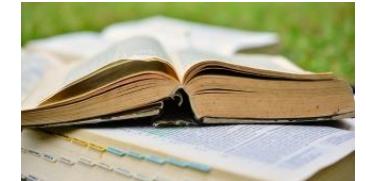
E Anwendungsanbindung (JDBC)



F Anwendungsanbindung (JPA) - optional

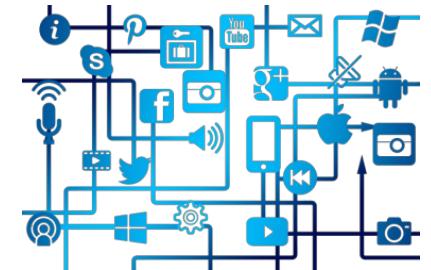


Literatur



- Kemper, Alfons; Eickler, André (2015): Datenbanksysteme. Eine Einführung. 10., aktualisierte und erweiterte Auflage. Berlin, Boston: De Gruyter Oldenbourg (De Gruyter Oldenbourg Studium).
 - Begleitende Website: <https://db.in.tum.de/teaching/bookDBMSeinf/index.shtml?lang=de>
 - Auf diesen Materialien basieren einige Teile der Veranstaltung.
- Kemper, Alfons; Wimmer, Martin (2012): Übungsbuch Datenbanksysteme. 3., aktualisierte und erweiterte Auflage. München: Oldenbourg.
- Elmasri, Ramez; Navathe, Shamkant B. (2017): Fundamentals of database systems. Seventh edition. Pearson.
 - 6. Auflage (engl.) verfügbar unter <http://iips.icci.edu.iq/images/exam/databases-ramaz.pdf>.
 - Deutsche Auflage: Elmasri, Ramez A.; Navathe, Shamkant B.; Shafir, Angelika (2011): Grundlagen von Datenbanksystemen. München: Pearson Studium (IT - Informatik).
- Vornberger, Oliver; Haldenwang, Nils (2015): Datenbanksysteme. Vorlesungsskript. Universität Osnabrück, Osnabrück. Institut für Informatik.
 - Für die Veranstaltung nicht vollständig relevante Kapitel: 4, 5, 10, 11, 15, 16, 18
 - Online verfügbar unter <http://www-lehre.inf.uos.de/~dbs/2015/PDF/dbs-2015-media2mult.pdf>.

Begleitmaterialien



- Vorlesungsunterlagen
 - In der Regel vor der Veranstaltung (mit „Lücken“) bereitgestellt.
 - Kleinere Übungen finden sich in den Vorlesungsfolien.
- Vorlesungsskript
 - Führend sind die VL-Unterlagen, diese enthalten u.U. weitere Inhalte!
- Übungsblätter
 - Diese sind vorzubereiten und werden in der Veranstaltung besprochen. **Dabei wird erwartet, dass Sie die Aufgaben vorstellen bzw. erklären können.**
- Datenbankmanagementsystem (DMBS) PostgreSQL
 - Nutzung über PC-Pool oder eigenen Rechner (empfohlen!).
 - DBMS: <https://www.postgresql.org/download/>
 - Administration, Abfragen etc. empfohlen über pgAdmin
 - <https://www.pgadmin.org/download/>
 - Dieses Programm wird auch verwendet, um sich zur HS-LU Datenbank zu verbinden.
 - JDBC Download: <https://jdbc.postgresql.org/download.html>

**BW342 – Datenbanken I
(mit Praktikum)**

Vorlesungen 1 & 2
Einführung und konzeptionelle
Datenmodellierung



Ziele von VL 1 & 2

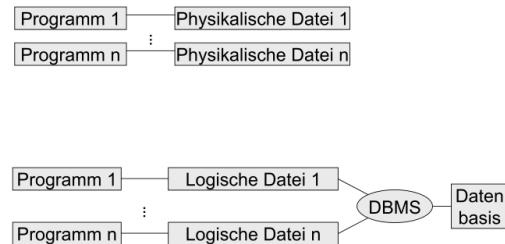


- **Notwendigkeit** für Datenbanken verstehen und einen Überblick über die **Terminologie** erhalten
- Unterschiedliche **Daten-Modellierungsebenen** kennenlernen
- Konzeptionelle Datenmodellierung: **Entity-Relationship-Modellen (ERM)** im Detail kennenlernen und verinnerlichen

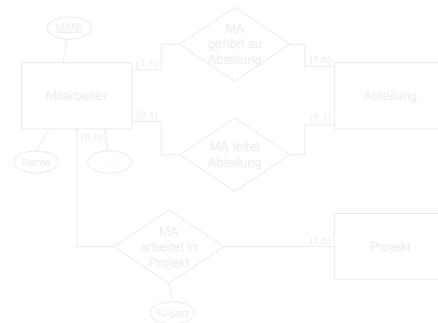
Themenüberblick



A Datenbankgrundlagen



B Konzeptionelle Modellierung



C Logische Modellierung

MID	Name	Adresse	Gehalt
1234	Müller	Hofstraße 7	23.000 €
1235	Meyer	Hauptstraße 13b	45.678 €
1236	Schmidt	Kurze Straße 1	55.555 €
1237	Heinze	Neue Straße 34	44.333 €
1238	Mustermann	Musterstraße 8	80.000 €
1239

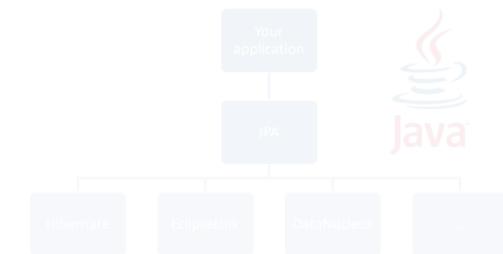
D Structured Query Language

- ~ filter your columns
`SELECT col1, col2, col3, ... FROM table1`
- ~ filter the rows
`WHERE col4 = 1 AND col5 = 2`
- ~ aggregate the data
`GROUP by ...`
- ~ limit aggregated data
`HAVING count(*) > 1`
- ~ order of the results
`ORDER BY col2`

E Anwendungsanbindung (JDBC)



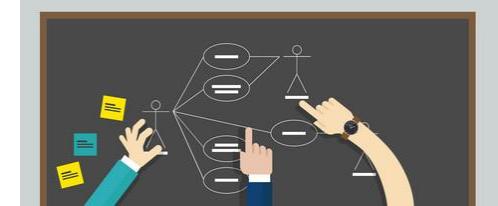
F Anwendungsanbindung (JPA)



■ Motivation

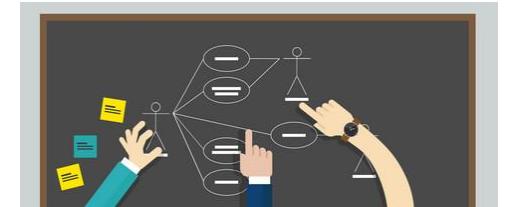
- Datenbankgrundlagen und Definitionen
- Grundlagen der Datenbankmodellierung
- Konzeptionelle Modellierung
- Zusammenfassung und Ausblick

Anwendungsbeispiel



- Wir möchten einen Web-Shop zum Lebensmittel-Verkauf eröffnen.
- Die selbst programmierte Anwendung muss Daten speichern:
Artikel, Kunden, Aufträge usw.
 - Wir fangen bei der Programmierung so an, dass die Daten in einzelnen Dateien im Dateisystem abgelegt werden, also:
 - eine Datei für Artikel,
 - eine Datei für Kunden,
 - eine Datei für Bestellungen usw.
- Welche Probleme können bei dieser Lösung auftreten?

Anwendungsbeispiel



■ Beispielhafte Probleme

- Herr Müller speichert die Datei an einem anderen Ort als Herr Schmidt
- Wird nicht in beiden Dateien gespeichert, laufen die Daten „auseinander“ und sind nicht mehr konsistent.
- Wird ein Kunde in der Kundendatei gelöscht, dann ist nicht sichergestellt, dass dieser auch wirklich gelöscht werden durfte (bspw. bei noch offenen Aufträgen)
- Sollte das „Datenformat“ geändert werden, können u. U. Programme die Daten nicht mehr korrekt „nutzen“
- Wenn zwei Mitarbeiter die Datei gleichzeitig bearbeiten wollen, besteht die Gefahr des gegenseitigen Überschreibens von Daten
- Unterschiedliche „Sichten“ auf Daten, z. B. Marketing vs. Vertrieb
- Nicht jeder darf alles sehen, z. B. Gehaltsdaten

Datenbanken als Lösung

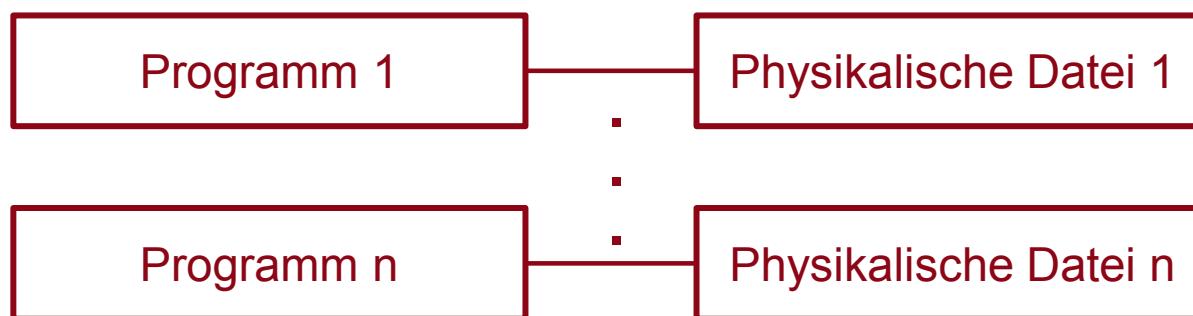
- Das Problem wurde schon vor langer Zeit erkannt:
 - *“Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). ... Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.” – E.F. Codd*

Quelle: E. F. Codd: A Relational Model of Data for Large Shared Data Banks,
Communications of the ACM Juni 1970

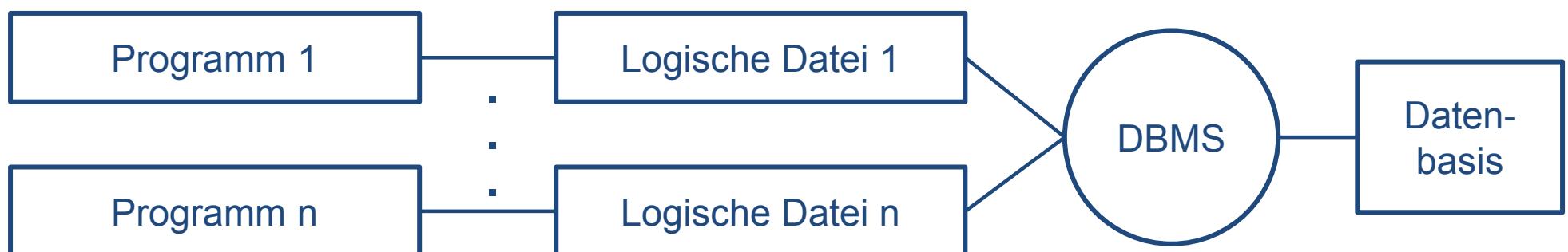
- Es muss gelten: keine Kenntnis der Anwendung über die interne
Daten-Repräsentation notwendig!
- Datenbankmanagementsysteme (DBMS) als Lösung

DBMS als Lösung

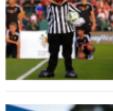
Übergang von isolierten Dateien:



Zu einer zentralen Datenbasis:



Relevanz der effizienten und effektiven Nutzung von Daten

Forbes The World's Most Valuable Brands							2017
Rank	Brand	Brand Value	1-Yr Value Change	Brand Revenue	Company Advertising	Industry	
#1	Apple	\$170 B	10%	\$214.2 B	\$1.8 B	Technology	
#2	Google	\$101.8 B	23%	\$80.5 B	\$3.9 B	Technology	
#3	Microsoft	\$87 B	16%	\$85.3 B	\$1.6 B	Technology	
#4	Facebook	\$73.5 B	40%	\$25.6 B	\$310 M	Technology	
#5	Coca-Cola	\$56.4 B	-4%	\$23 B	\$4 B	Beverages	
#6	Amazon	\$54.1 B	54%	\$133 B	\$5 B	Technology	
#7	Disney	\$43.9 B	11%	\$30.7 B	\$2.9 B	Leisure	
#8	Toyota	\$41.1 B	-2%	\$168.8 B	\$4.3 B	Automotive	

Cookies on Forbes

Quelle: <https://www.forbes.com/powerful-brands/list/>

Übung



- In einer Bibliothek werden die zu verleihenden Bücher in einer einzigen Datei verwaltet. Die folgende Tabelle zeigt, wie die Daten gespeichert werden:

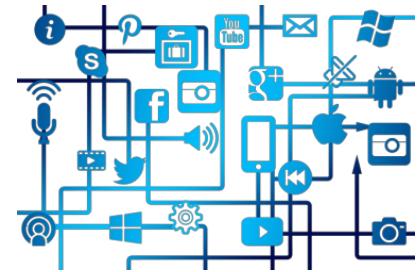
ISBN	Autor	Titel	Verlag
3834812226	H. Neuendorf	Java-Grundkurs für Wirtschaftsinformatiker	Vieweg+Teubner
3834813826	G. Blakowski	Datenbanken für Wirtschaftsinformatiker	Vieweg+Teubner
3834812226	H. Neuendorf	Java-Grundkurs für Wirtschaftsinformatiker	Vieweg+Teubner
3834813816	G. Brosius	Datenbanken für Wirtschaftsinformatiker	Teubner+Vieweg
3834813826	S. Cordts	Datenbanken für Wirtschaftsinformatiker	Springer Verlag
3540372172	G. Fandel	Produktionsmanagement	Springer Verlag

- In der Dateistruktur sind mehrere Entwurfsfehler enthalten, die zu Problemen bei der Speicherung geführt haben. Um welche Probleme handelt es sich und wie könnten diese gelöst werden?
 - Bearbeitungszeit: 5min individuell, 5min Kleingruppendiskussion (2-3 Studierende)
 - Im Anschluss: Vorstellung und gemeinsame Diskussion

Probleme bei isolierter Datenhaltung

- **Redundanz:** Dieselben Informationen werden mehrfach gespeichert.
- **Inkonsistenz:** Es existieren unterschiedliche Versionen derselben Informationen.
- **Integrität:** Die Einhaltung von Integritätsbedingungen fällt schwer.
- **Verknüpfungseinschränkung:** Logisch verwandte Daten sind schwer zu verknüpfen, wenn sie in isolierten Dateien liegen.
- **Mehrbenutzerprobleme:** Das gleichzeitige Bearbeiten derselben Datei führt zu Anomalien („lost update“).
- **Verlust von Daten:** Begrenzt auf komplettes Backup der einzelnen Dateien. Kein spezieller „Recovery-Mechanismus“
- **Sicherheitsprobleme:** Zugriffsrechte können nicht in abgestufter Form implementiert werden.
- **Hohe Entwicklungskosten:** Für jedes Anwendungsprogramm müssen die Fragen zur Dateiverwaltung erneut gelöst werden.

Weiterführendes Material



- Video zur Motivation / Notwendigkeit von Datenbanksystemen
https://www.youtube.com/watch?v=l0z3OqCII_E

Beispiel:

13.04 Warum Datenbanksysteme?
Jens Dittrich • 15.000 Aufrufe • vor 4 Jahren

Komplette Liste der Videos und zusätzliches Material auf <http://datenbankenlernen.de> Informatik, Uni Saarland:

- Motivation**
- Datenbankgrundlagen und Definitionen**
- Grundlagen der Datenbankmodellierung**
- Konzeptionelle Modellierung**
- Zusammenfassung und Ausblick**

Terminologie im Kontext von DBMS

- Daten
 - bekannte Tatsachen über die interessierende Domäne
- Datenbank
 - strukturierte Sammlung von Daten über eine Domäne, d.h.
 - logisch zusammenhängend
 - systematisch aufgezeichnet
 - gespeichert und gepflegt
 - zweckmäßig für (evtl. verschiedene) Anwender
- Datenbankmanagementsystem (DBMS)
 - Software zum Erstellen und Pflegen von Datenbanken; **generisch**, d.h. unabhängig von einem bestimmten Anwendungsgebiet
- Datenbanksystem
 - Einsatz eines DBMS für eine bestimmte Datenbank und bestimmte Anwendungen

Datenbankmanagementsystem – Definition

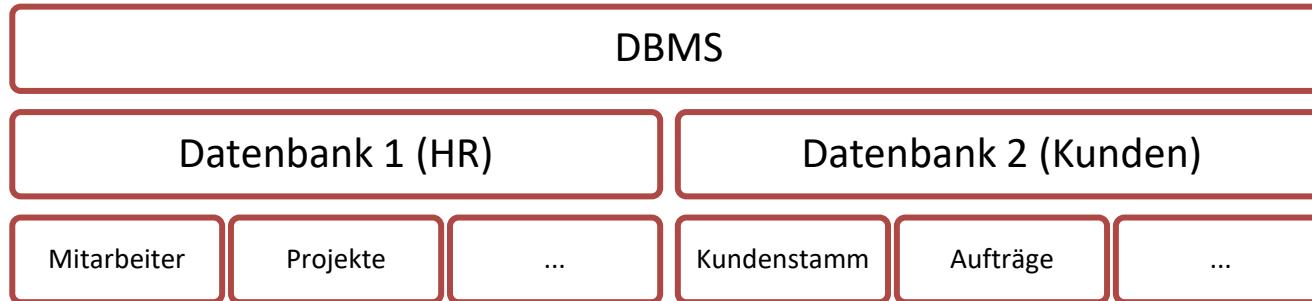
- Ein Datenbankmanagementsystem (DBMS) ist eine Software, die **große** Mengen von **persistenten** Daten für Speicherung und Zugriff durch **mehrere Benutzer** verwaltet.
 - Zu groß für den Hauptspeicher
 - Daten bleiben erhalten, auch wenn Anwendungen, die sie verwenden, beendet werden
 - Synchronisation von Zugriffen, Verschiedene Sichten für unterschiedliche Nutzer und Anwendungen
- Dies soll unter folgenden Anforderungen erfolgen
 - sicher
 - In Bezug auf Systemausfälle und Nutzerberechtigungen
 - effizient
 - Akzeptable Antwortzeiten, Anzahl gleichzeitiger Transaktionen
 - zweckmäßig
 - Für den Anwendungsfall optimierter Zugriff (bspw. einfache Abfragesprache, Erstellung komplexer Abfragen)
 - für den parallelen Zugriff vieler Anwender und Anwendungen
 - Insbesondere Handling von gleichzeitigen Änderungen (Transaktionen, Synchronisation), verschiedene Sichten und skalierbare Performance bei Multi-User Zugriff

Datenbankmanagementsystem – weitere Merkmale

- Gemeinsamer Datenbestand
 - Integrierte Daten für verschiedene Anwendungen mit gemeinsamer Nutzung derselben Daten
 - Kontrolle von Redundanzen
 - Überwachung der Konsistenz
 - Sicherung der Persistenz
- Physische Datenunabhängigkeit = Immunität von Anwendungen in Bezug auf Änderungen der physischen Repräsentation der Daten und von Zugriffstechniken
- Unterstützung spezifischer Sichten und verschiedener Berechtigungen auf die Daten
- Steuerung des Mehrbenutzerbetriebs = Synchronisation konkurrierender Zugriffe
- Einhalten von Standards (bspw. in Bezug auf die Abfragesprache)

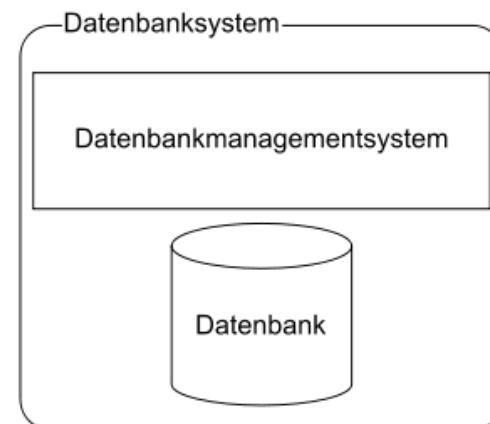
DBMS und Datenbanksysteme

- DBMS können mehr als eine Datenbank enthalten:



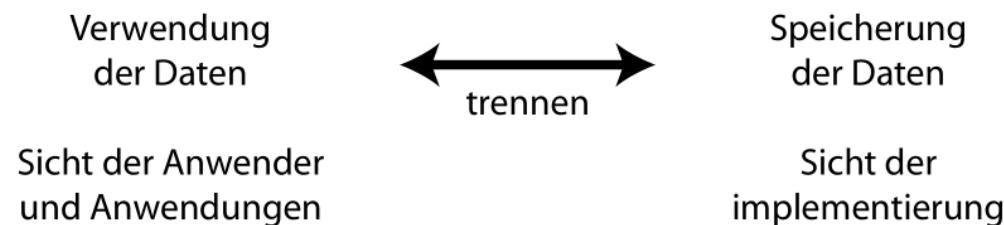
Neben den eigentlichen Datenbanken enthält das DBMS auch eine Datenbasis zum Speichern von **Metadaten** ("Daten über Daten"), wie Informationen über die Datenstruktur, Zugriffsrechte, etc.

- Datenbankmanagementsystem und Datenbanken bezeichnet man als **Datenbanksystem** (kurz: **DBS**).



Datenunabhängigkeit

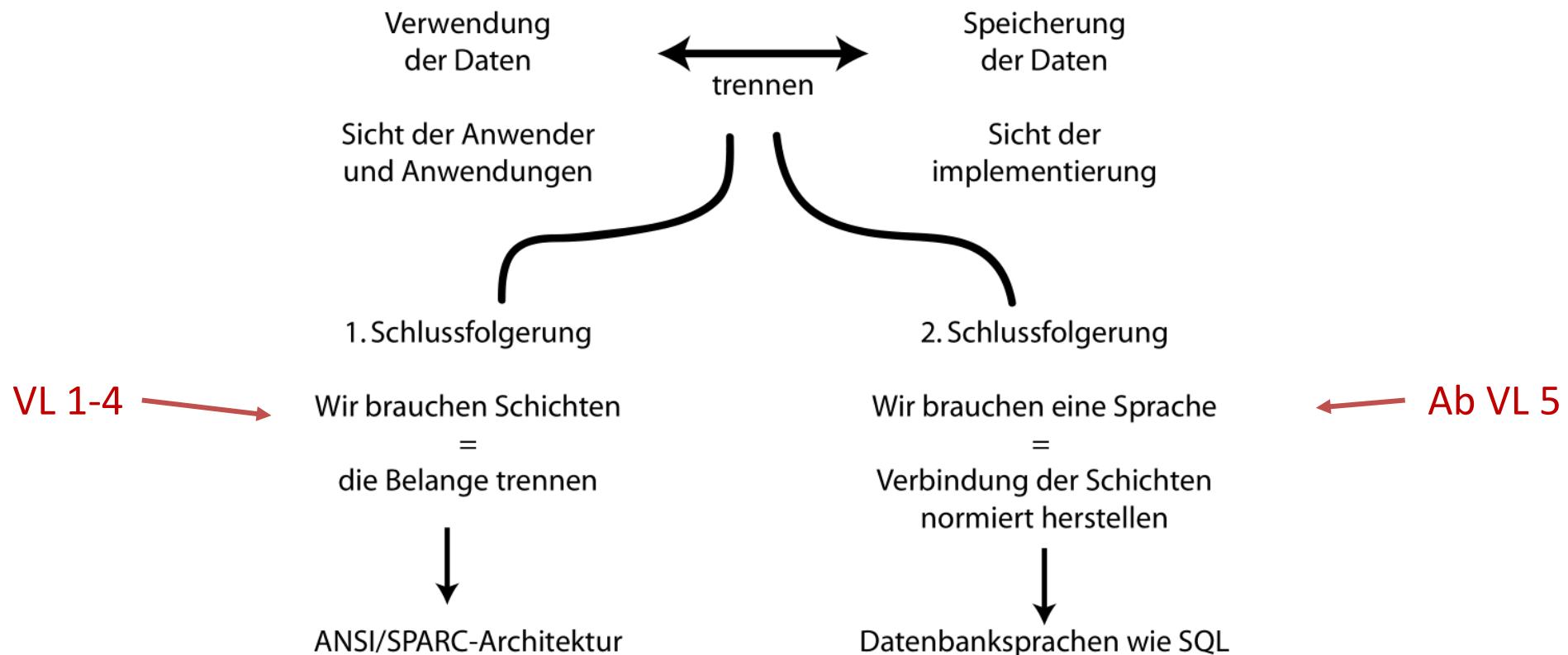
- Wie erreicht man die gewünschte Unabhängigkeit zwischen Verwendung der Daten und Speicherung der Daten?



Quelle: <https://homepages.thm.de/~hg11260/mat/dbs-einf-bh.pdf>

Datenunabhängigkeit

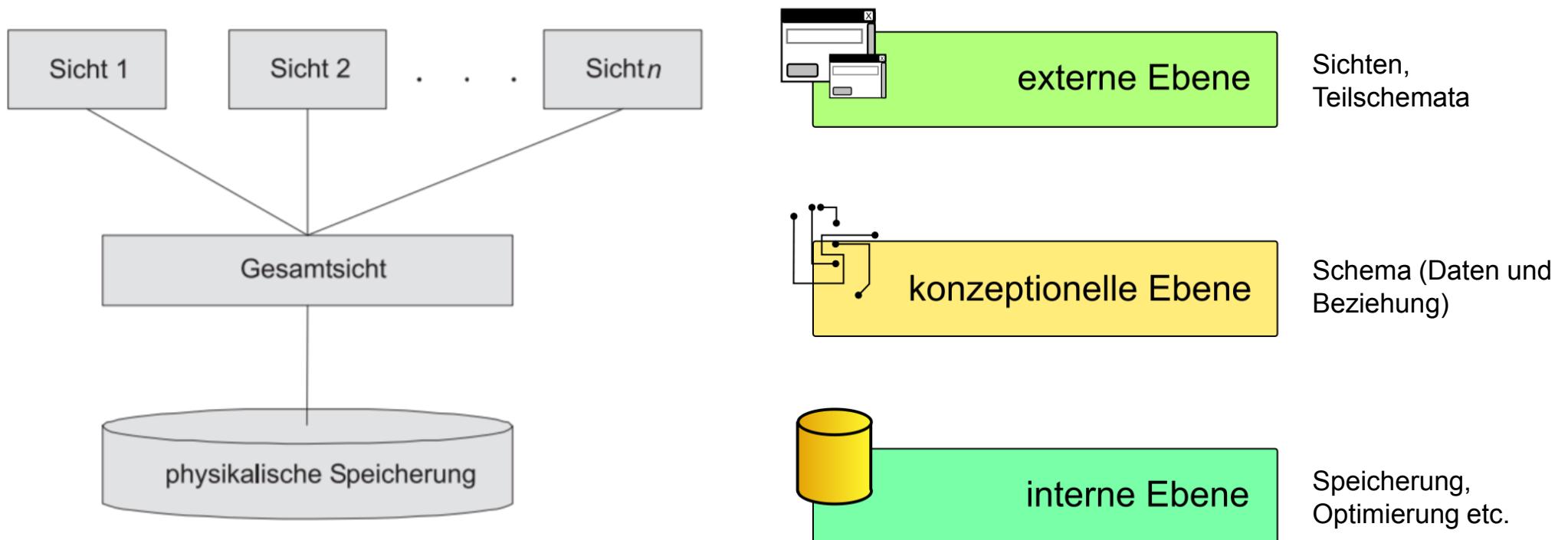
- Wie erreicht man die gewünschte Unabhängigkeit zwischen Verwendung der Daten und Speicherung der Daten?



Quelle: <https://homepages.thm.de/~hg11260/mat/dbs-einf-bh.pdf>

Datenabstraktion nach der ANSI/SPARC-Architektur

- Ein Ansatz ist die Unterscheidung zwischen drei Abstraktionsebenen:



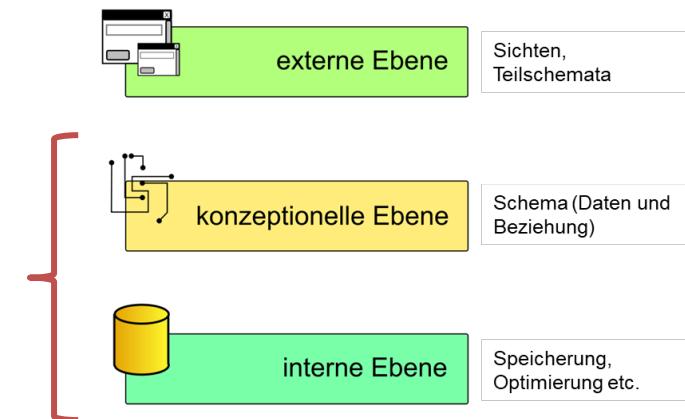
ANSI = American National Standards Institute

SPARC = Standards Planning and Requirements Committee

Datenunabhängigkeit

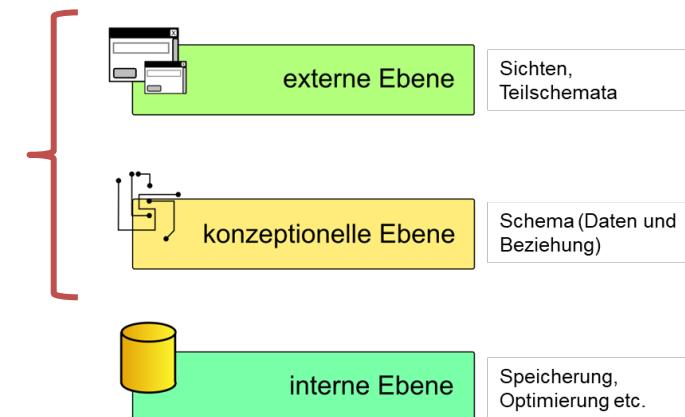
Physische Datenunabhängigkeit

- Bei Änderung der internen Ebene keine Änderung der konzeptionellen Ebene
 - In den meisten heutigen DBMS erreicht. Allerdings kann die Notwendigkeit für standardisierte Schnittstellen zwischen den Ebenen Ineffizienzen hervorrufen.

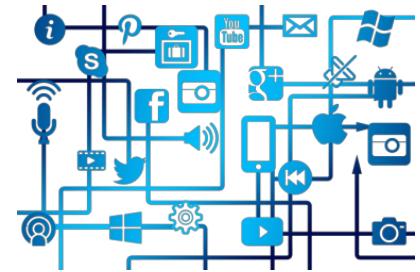


Logische Datenunabhängigkeit

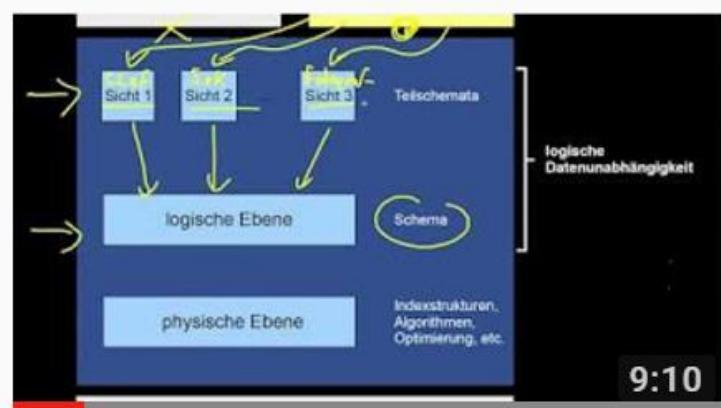
- Bei Änderungen der konzeptionellen Ebene keine Änderung der externen Ebene notwendig.
 - Anwendungen gegen Änderungen des Datenbankschemas „immun“
 - Schwierig zu erreichen, da externe Sichten oft 1:1 Ausschnitte des konzeptionellen Schemas abbilden und Änderungen sich somit auswirken.



Weiterführendes Material



- Video zur physischen und logischen Unabhängigkeit
<https://www.youtube.com/watch?v=Flb3SeEcqGc>



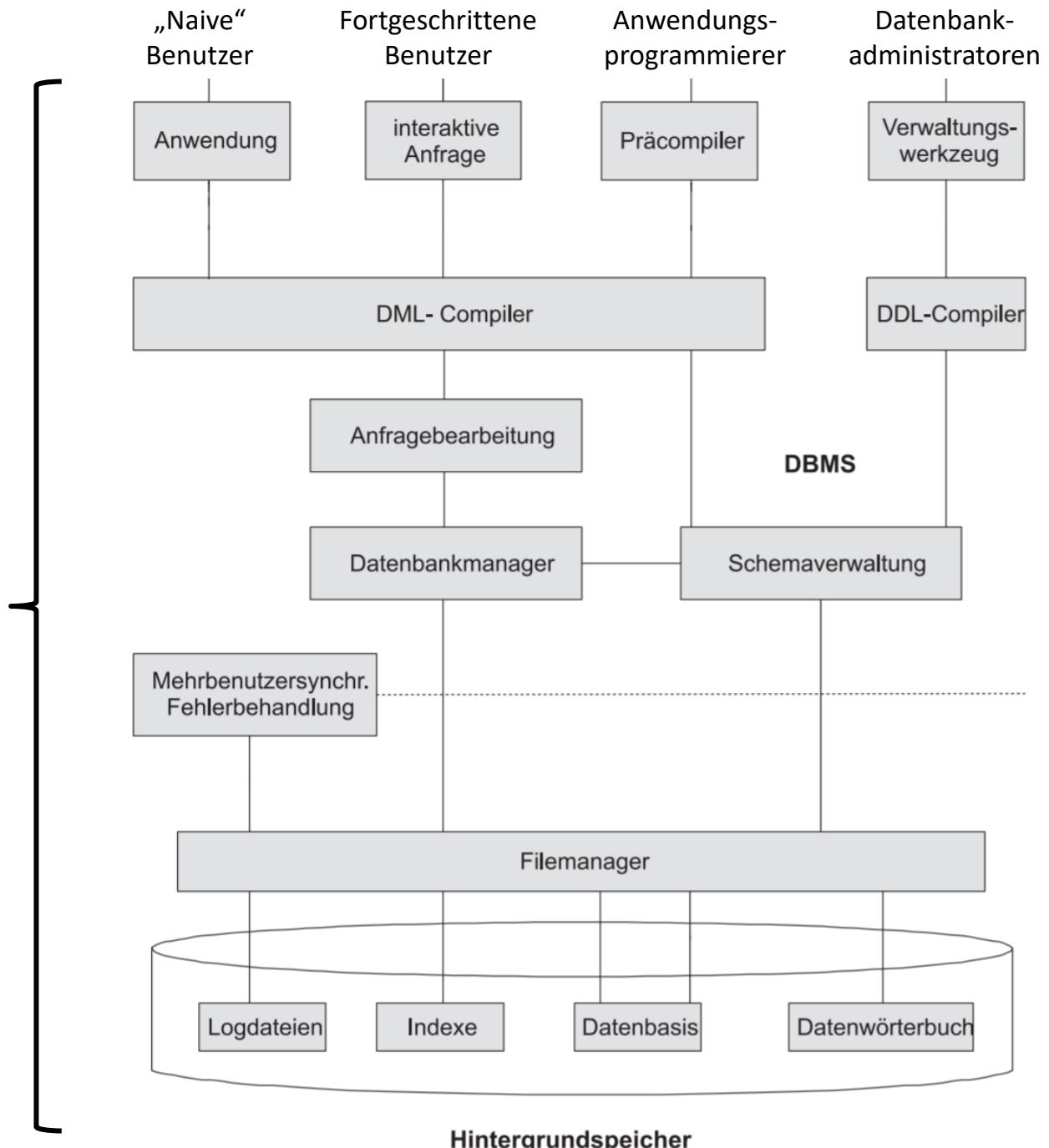
13.06 Physische und Logische Datenunabhängigkeit

Jens Dittrich • 10.000 Aufrufe • vor 4 Jahren

Komplette Liste der Videos und zusätzliches Material auf
<http://datenbankenlernen.de> Informatik, Uni Saarland:

Typische Komponenten eines DBMS

Zusammen Abbildung der drei Ebenen
(externe Schemata, konzeptionelles Schema, internes Schema)



DDL: Data Definition Language

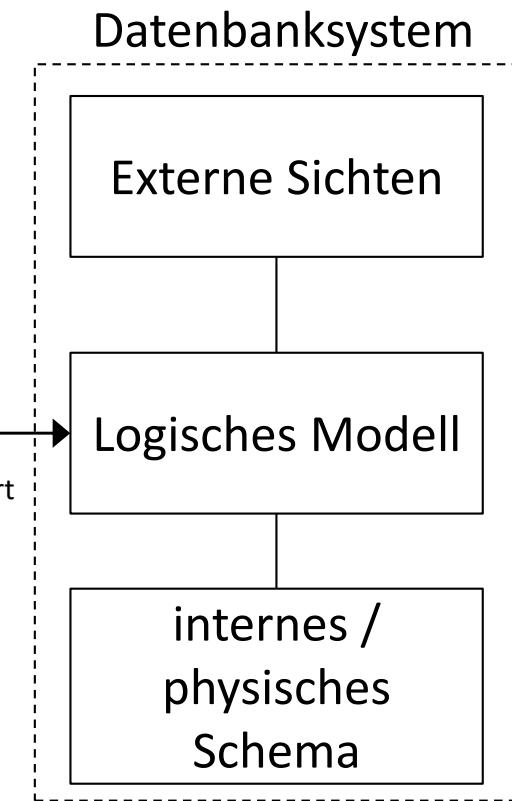
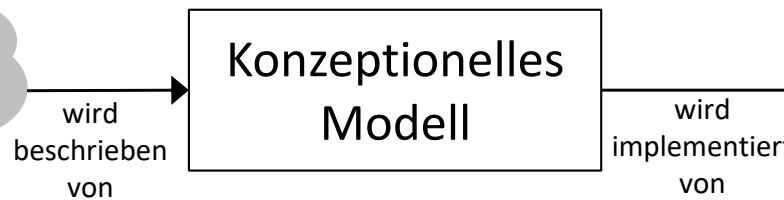
DML: Data Manipulation Language

Quelle: Vornberger 2015

- Motivation**
- Datenbankgrundlagen und Definitionen**
- Grundlagen der Datenbankmodellierung**
- Konzeptionelle Modellierung**
- Zusammenfassung und Ausblick**

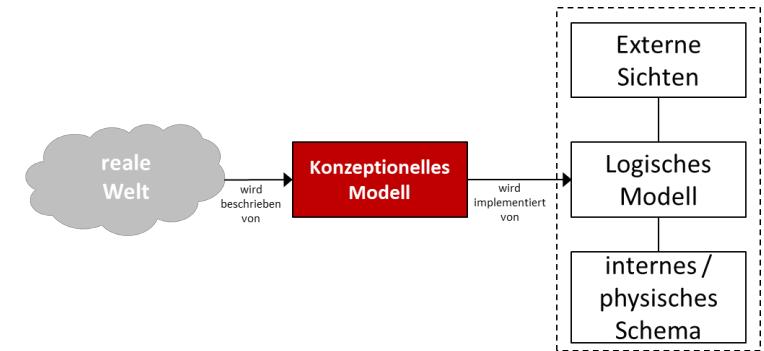
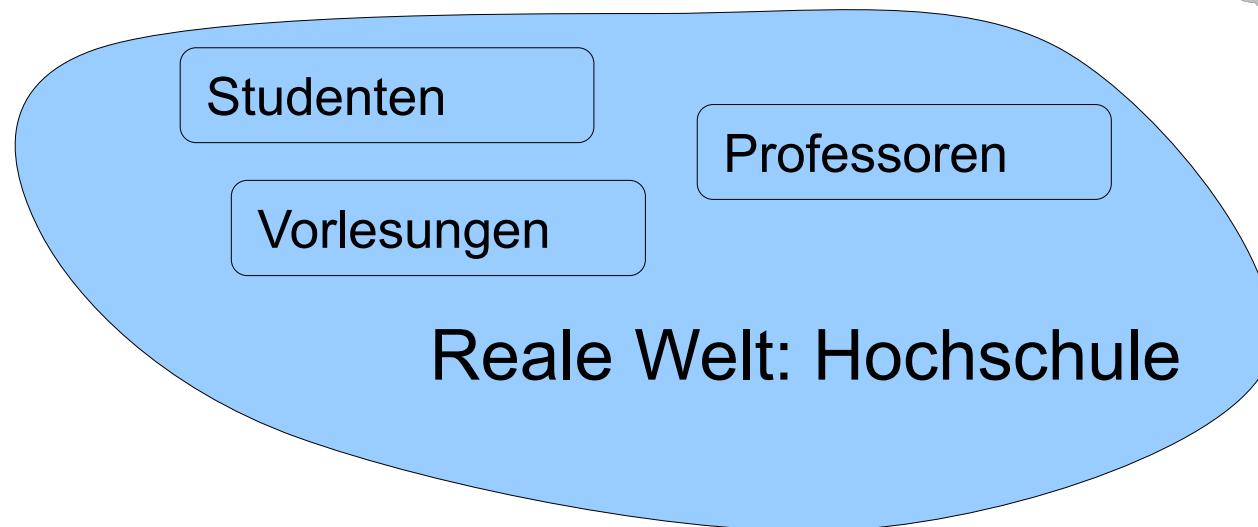
Modellierungskonzepte

■ Zweistufiger Abstraktionsansatz

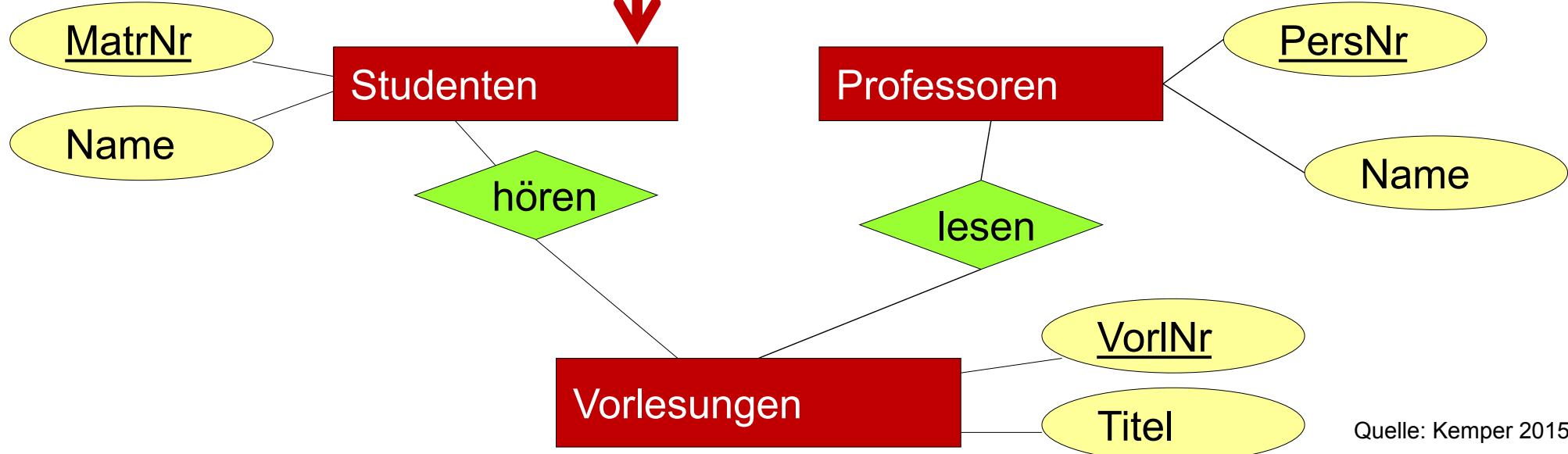


- Konzeptionelles Modell:
 - Welt unabhängig von DV-Gesichtspunkten beschreiben
 - Grundlage für das interne (DV-abhängige) Schema bilden (oftmals verwendet: Entity-Relationship-Model, ERM)
- → dem konzeptionellen Modell wird ein logisches Modell zur Seite gestellt
 - Abbildung der Gesamtheit der Daten Hardware-unabhängig, aber dennoch unter Berücksichtigung von bestimmten Implementierungsgesichtspunkten (implementiertes konzeptionelles Schema)
 - Bspw. kann ein und dasselbe ERM als Netzwerkmodell oder relationales Modell oder durch Verwendung von NoSQL-Ansätze logisch abgebildet werden

Beispiel

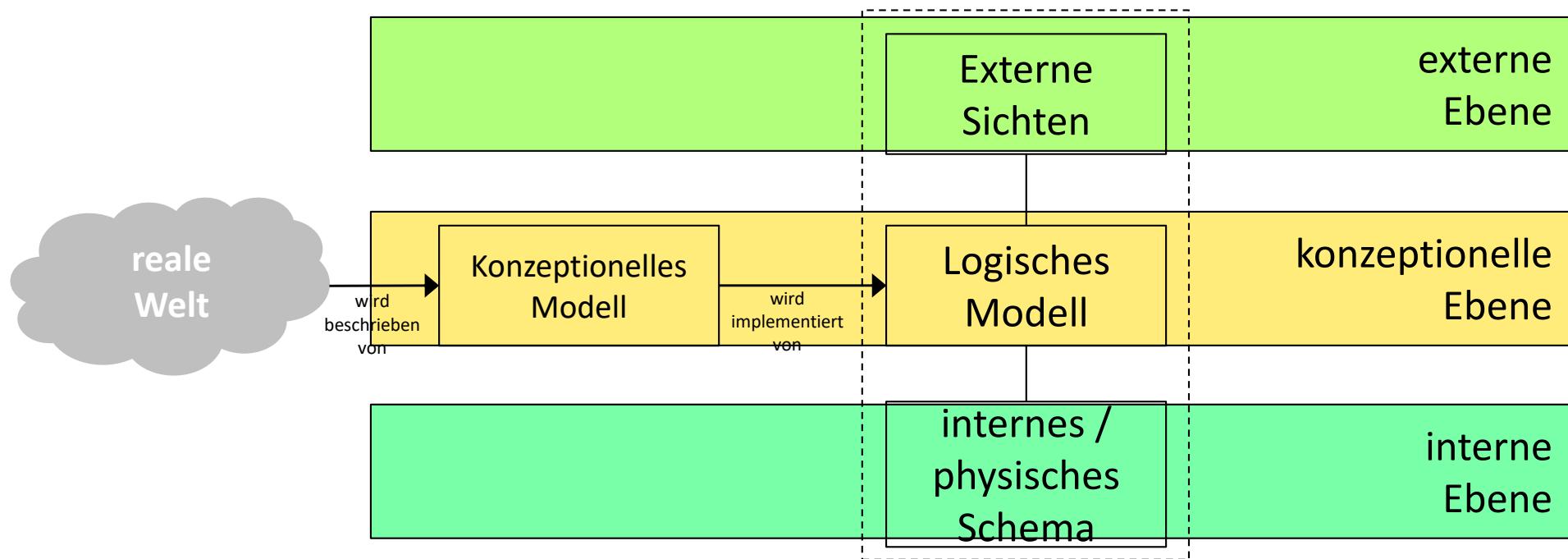
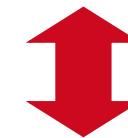
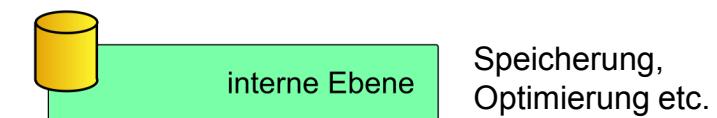
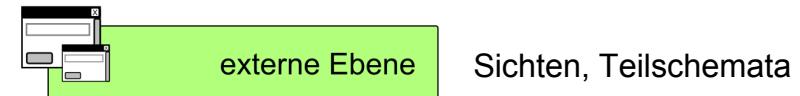


Konzeptionelle Modellierung

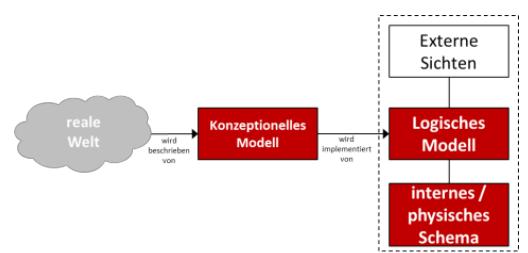


Quelle: Kemper 2015

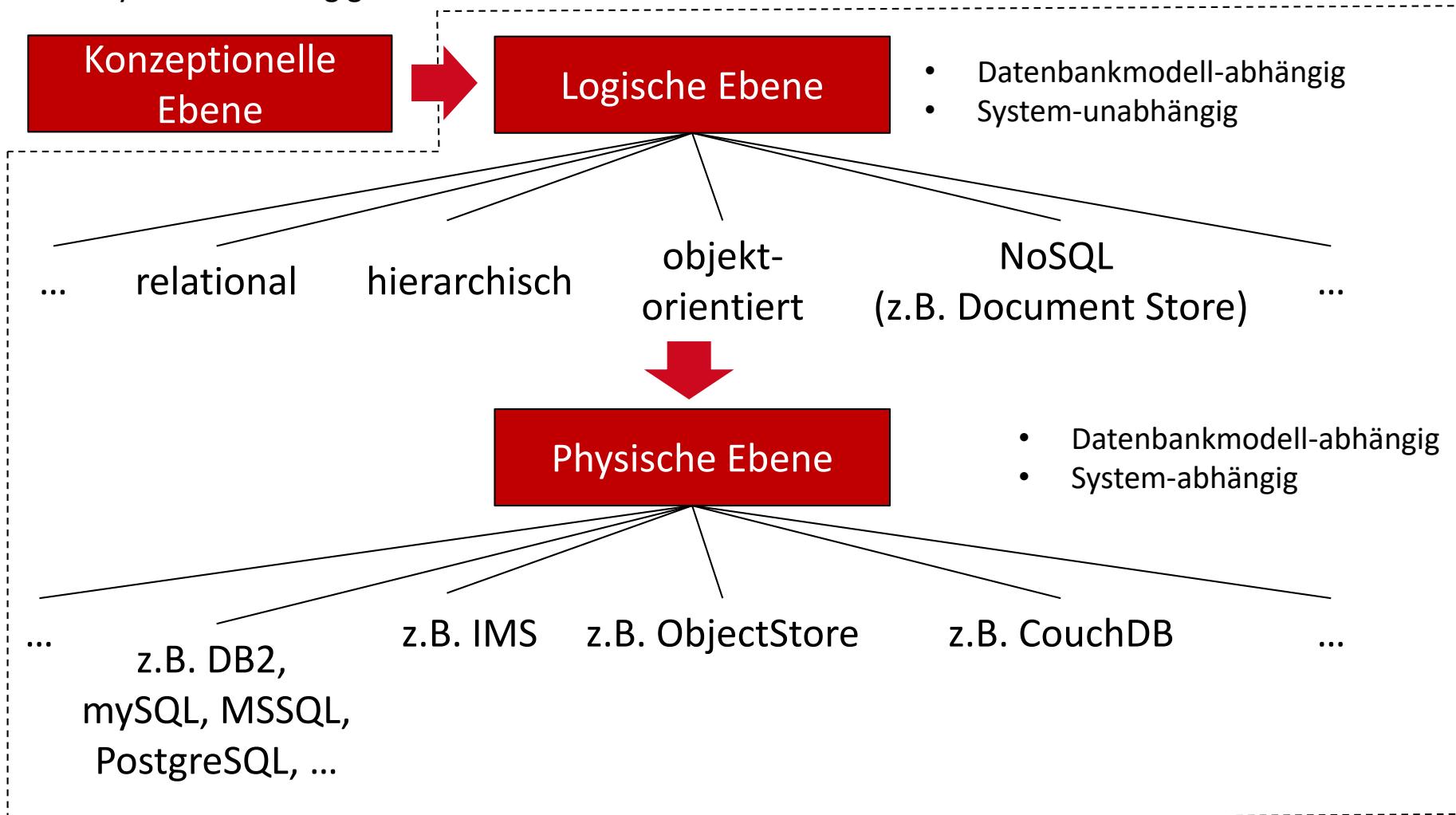
Zusammenhänge



Modellierung und Arten von DBMS



- Datenbankmodell-unabhängig
- System-unabhängig

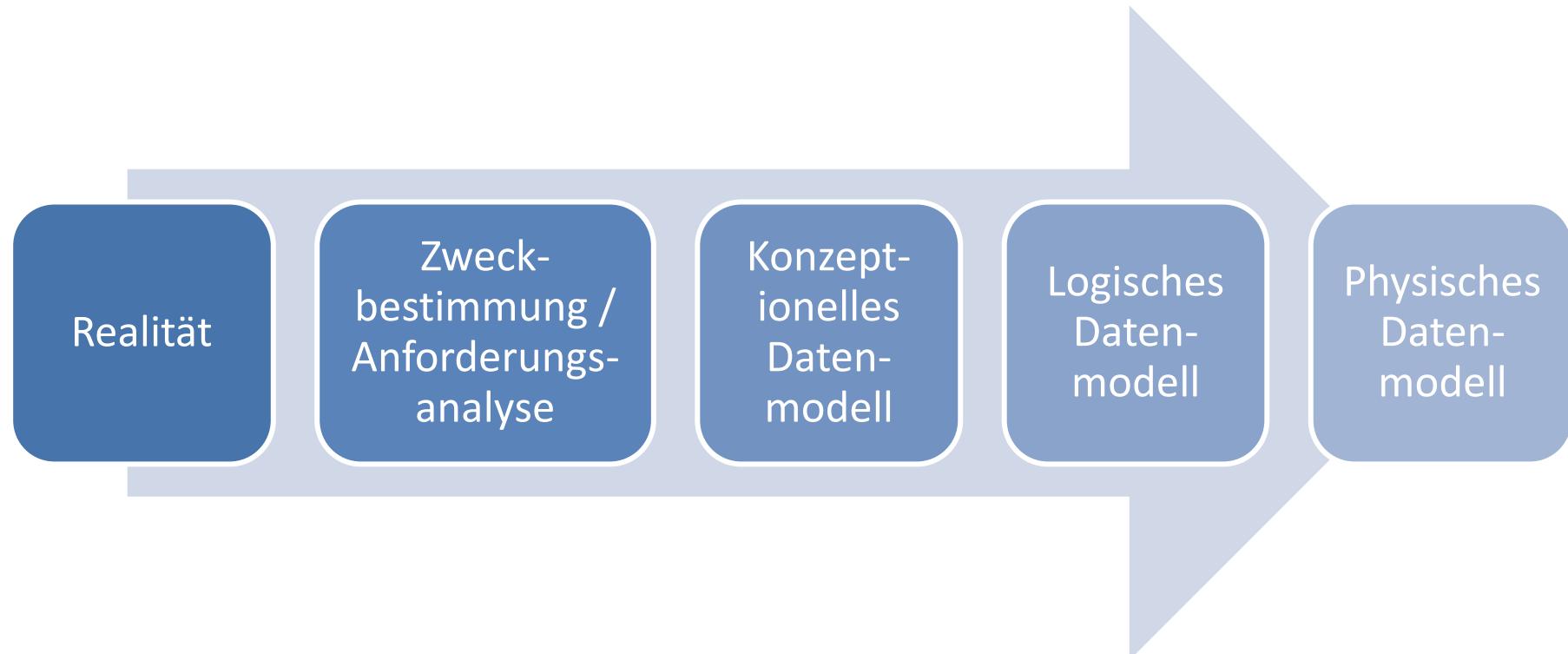


Arten von DBMS

- Netzwerk-DBMS und Hierarchische DBMS
 - veraltet, nur noch in Legacy-Systemen
- Relationale DBMS (auch RDBMS)
 - marktbeherrschend, z. B. als Grundlage aller wichtigen ERP-Systeme.
- Objektorientierte DBMS (auch OODBMS)
- NoSQL-Datenbanken
 - insbesondere bei Anwendungen mit sehr großen Datenmengen, um horizontale Skalierung zu erreichen (z.B. beim Verarbeiten von Sensordaten)
- Multidimensionale Datenbanken
 - z. B. als Basis für Data Warehouse und Business Intelligence Anwendungen
- In-Memory-Datenbanken für hohe Zugriffsgeschwindigkeiten
- Graphdatenbanken
- ... und viele mehr

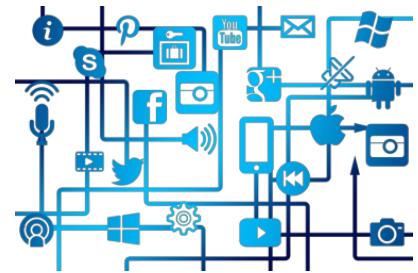
Vorgehen in der Modellierung

- Phasen:



Beispiel: einfaches Geschäftsumfeld „Kunde kauft Auto“

Weiterführendes Material



- Video zu Modellierungsschritten

<https://www.youtube.com/watch?v=nOmSmTGy39o>



13.07 Übersicht über die
Modellierungsschritte: von der
Jens Dittrich • 16.000 Aufrufe • vor 4 Jahren

Komplette Liste der Videos und zusätzliches Material auf
<http://datenbankenlernen.de> Informatik, Uni Saarland:

- Motivation
- Datenbankgrundlagen und Definitionen
- Datenbankmodellierung
- Konzeptionelle Modellierung
- Zusammenfassung und Ausblick

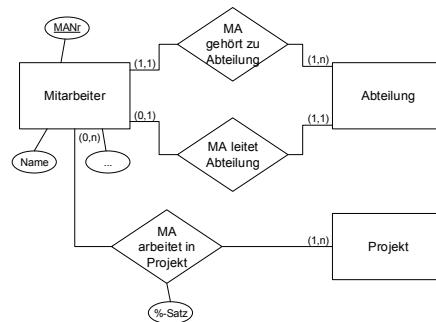
Themenüberblick



A Datenbankgrundlagen



B Konzeptionelle Modellierung



C Logische Modellierung

MID	Name	Adresse	Gehalt
1234	Müller	Hofstraße 7	23.000 €
1235	Meyer	Hauptstraße 13b	45.678 €
1236	Schmidt	Kurze Straße 1	55.555 €
1237	Heinze	Neue Straße 34	44.333 €
1238	Mustermann	Musterstraße 8	80.000 €
1239

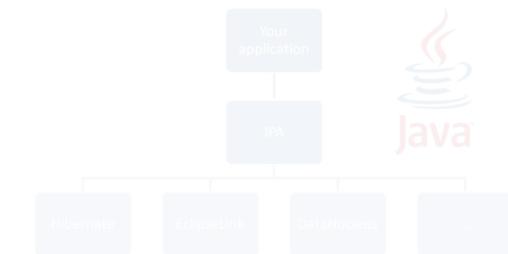
D Structured Query Language

- filter your columns
`SELECT col1, col2, col3, ... FROM table1`
- filter the rows
`WHERE col4 = 1 AND col5 = 2`
- aggregate the data
`GROUP by ...`
- limit aggregated data
`HAVING count(*) > 1`
- order of the results
`ORDER BY col2`

E Anwendungsanbindung (JDBC)

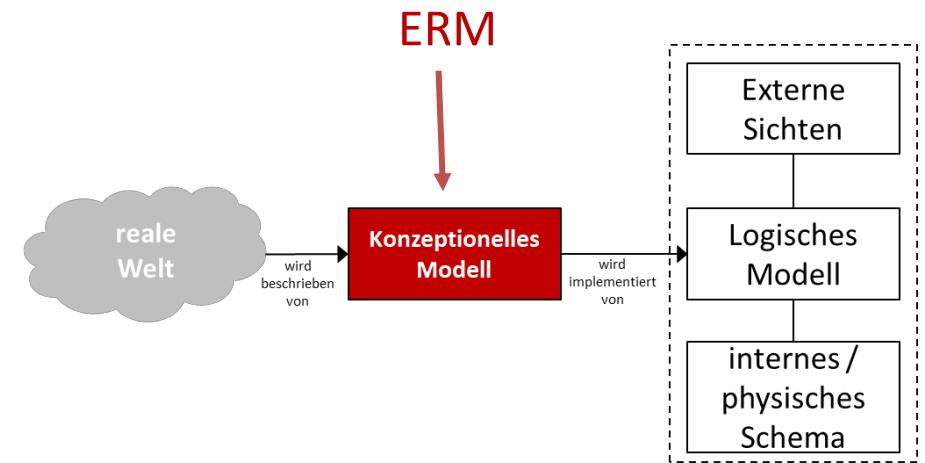


F Anwendungsanbindung (JPA)



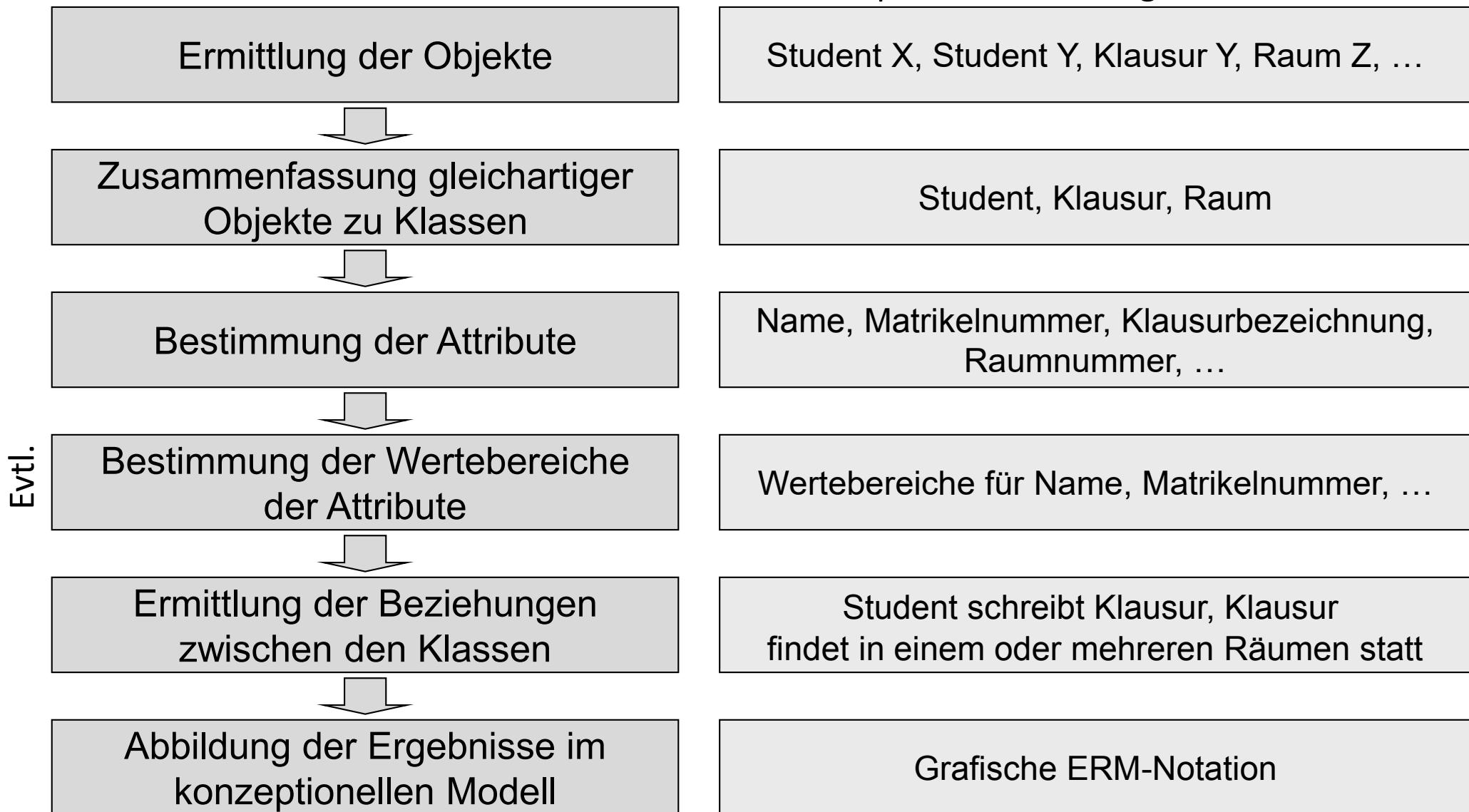
Entity Relationship Model (ERM)

- Grafische Notation zur formellen Darstellung von konzeptionellen Datenbankmodellen.
- Unabhängig von einem bestimmten Datenbankmanagementsystem und dessen grundlegendem Ansatz der logischen Modellierung.
- Hält die im Betrachtungsbereich (Domäne) relevanten Objekte (Entities) und Beziehungen (Relationships) zwischen diesen formal fest.
- Objekte und Beziehungen können durch Eigenschaften (Attributes) näher beschrieben werden.
- Beziehungen zwischen Objekten können genau spezifiziert werden.



ERM-Modellierung: Schrittweises Vorgehen

Beispiel: Modellierung von Klausuren

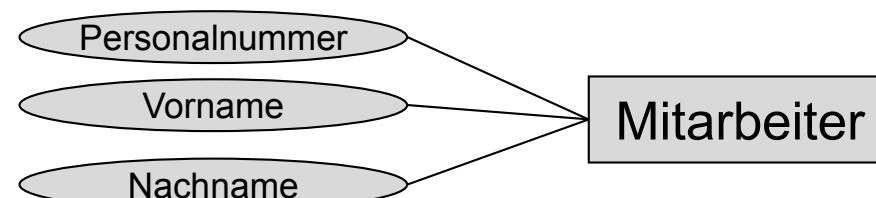


ERM: Entitytypen und Attribute

- Entity (Geschäftsobjekt)
 - Etwas real Existierendes: Person, Auto, Buch
 - ein Ereignis: Artikel bestellen, Gespräch führen
 - eine Rolle oder Person: Arzt, Mitarbeiter,
 - eine Organisation,
 - eine Transaktion: Kaufvertrag, Warenausgang, Wareneingang (vgl. Ereignis)
- Attribut
 - Weist einem Entity- oder Relationshiptypen Eigenschaften zu.
- Verbindung
 - Ungerichtete Kante
- Beispiel:

Entitytyp

Attribut



ERM: Namenskonventionen für Entitytypen

- Mehrere gleichartige Entities werden zu Entity-Typen zusammengefasst (z. B. „Person“)

1. Substantiv, Singular

- „Artikel“
- „Student“
- „Organisationseinheit“
- „Rechnung“, „Bestellung“, ...

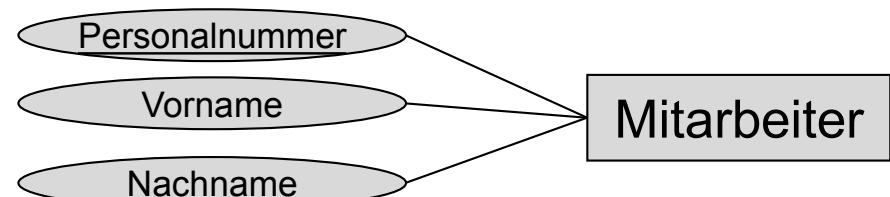
2. Näher spezifiziertes Substantiv, Singular

- „Person, weiblich“ (oder: „weibliche Person“)
- „Person, männlich“ (oder: „männliche Person“)
- „Rechnung, Streckengeschäft“
- „Bestellung, Filiale“, „Bestellung, Web-Shop“, ...
- ...

ERM: Schlüsselattribute

■ Schlüssel

- Welche Attribute bzw. Kombinationen aus Attributen sind notwendig, um ein Geschäftsobjekt (eine Entität) eindeutig zu identifizieren?
- Gibt es mehrere Attribute?
→ als Schlüsselkandidaten bezeichnet
- **Primärschlüssel**
 - Identifiziert ein Objekt eindeutig
 - (oft künstlicher) Schlüssel, der auch aus mehreren Attributen zusammengesetzt sein kann
 - unterstrichen dargestellt





Fallbeispiel „Vereinsportal“

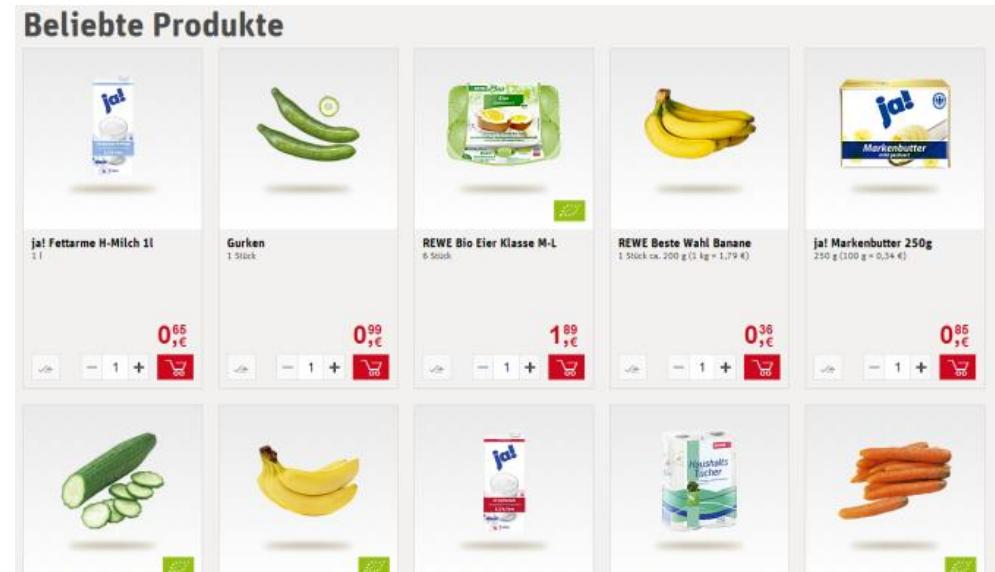
Bitte helfen Sie der Verwaltung eines Vereins bei der Errichtung eines Mitglieder-Portals. Der Vereinsvorstand beschreibt Ihnen die Anforderungen wie folgt:

- In unserem Portal müssen wir die Mitglieder speichern können. Zu jedem Mitglied soll zunächst nur der Name, Geburtsdatum und eine Telefonnummer erfasst werden. Eindeutig identifizieren wir Mitglieder über den Namen und das Geburtsdatum.
- Wir sind ein lokaler Verein und nehmen bisher nur Mitglieder aus unserer Stadt auf. Daher soll zu jedem Mitglied aus einer Liste der Stadtteil zugeordnet werden können (eindeutig über den Namen des Stadtteils identifizierbar).
- Ein Mitglied übernimmt eine oder mehrere Rollen im Verein, muss dies jedoch nicht tun. Bei den Rollen haben wir uns RollenIDs und Bezeichnungen ausgedacht. Es gibt auch für jeden Stadtteil jeweils eine Rolle für den Stadtteilverantwortlichen, d.h. eine Rolle kann einem Stadtteil zugeordnet werden.
- Zusätzlich haben wir auch noch eine Rollenhierarchie.

→ **Welche Entitytypen (Geschäftsobjekte) mit welchen Attributen können Sie im Fallbeispiel identifizieren? Wie lauten die Primärschlüssel?** Ignorieren Sie zunächst den Hinweis auf die Hierarchie!

Fallbeispiel Online-Lebensmittelhandel

- Ein Unternehmen möchte in den Markt der Online-Supermärkte einsteigen und von Ihnen einen entsprechenden Online-Shop entwickeln lassen.
 - Grundlage hierfür stellt eine entsprechende Datenbank dar.



- Anforderungen
 - Ein Kunde soll in der Lage sein, sich vor dem Einkauf zu registrieren und dabei die für den Einkauf relevanten Daten (z. B. Adresse) zu hinterlegen.
 - Registrierte Kunden sollen Produkte in den Warenkorb übertragen können.
 - Nur verfügbare Produkte sollen dem Kunden angezeigt werden.
 - Eine Historie aller Bestellungen soll aufgebaut werden.
 - Der Kunde soll über Statusänderungen seiner Bestellung informiert werden können.

Übung

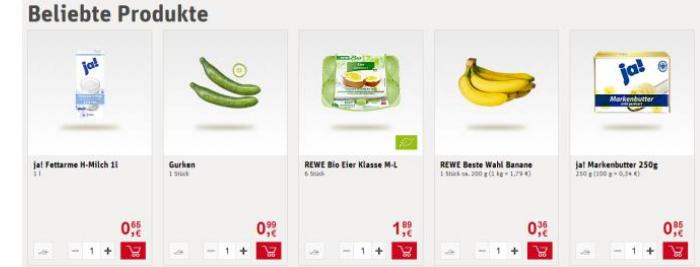


Anforderungen

- Das System wird von Benutzern (Kunden & Mitarbeiter), die durch ihre E-Mailadresse identifiziert werden, bedient.
 - Alle Benutzer registrieren sich mit Name und Vorname.
 - Kunden sollen darüber hinaus in der Lage sein, bei der Registrierung ihre Adresse (Str., Hausnummer, PLZ, Ort) und ein Geburtsdatum zu hinterlegen. Für Kunden wird automatisch eine ID angelegt.
 - Mitarbeiter hinterlegen ihre eindeutige Personal- und Tel.-Nr. Zusätzlich sollen Sie ihrer Abteilung in einer Abteilungshierarchie zugeordnet werden.
- Registrierte Kunden sollen Produkte in ihren Warenkorb „legen“ können (auch mehr als 1x).
 - Dabei bekommt jeder Warenkorb eine eindeutige ID und das Erstelldatum wird auch gespeichert.
 - Produkte haben eine eindeutige Nummer und in der Datenbank sollen zusätzlich die Bezeichnung, die aktuelle Verfügbarkeit, der Listenpreis und das Anlagedatum des Produkts in den Stammdaten gespeichert werden.
 - Produkte gehören zu Kategorien; diese wiederum haben eine Hierarchie (bspw. Banane → Obst → Lebensmittel).
- Wenn ein Kunde einen Warenkorb bestellen möchte, wird eine entsprechende Bestellung im System angelegt und dem Warenkorb zugeordnet.
 - So kann eine Historie der Bestellungen aller Kunden aufgebaut werden. Bestellungen setzen einen Warenkorb voraus, in dem die zu bestellenden Produkte und deren Bestellmenge hinterlegt werden.
 - Bestellungen werden von einem Mitarbeiter bearbeitet und der Kunde kann sich über die Website über den aktuellen Status seiner Bestellung informieren.

■ Welche Entitytypen (Geschäftsobjekte) mit welchen Attributen können Sie im Fallbeispiel identifizieren? Wie lauten die Primärschlüssel?

- Bearbeitungszeit: 10min individuell, 5min Kleingruppendiskussion (2-3 Studierende)
- Im Anschluss: Vorstellung und gemeinsame Diskussion



ERM: Beziehungstypen und Kardinalitäten

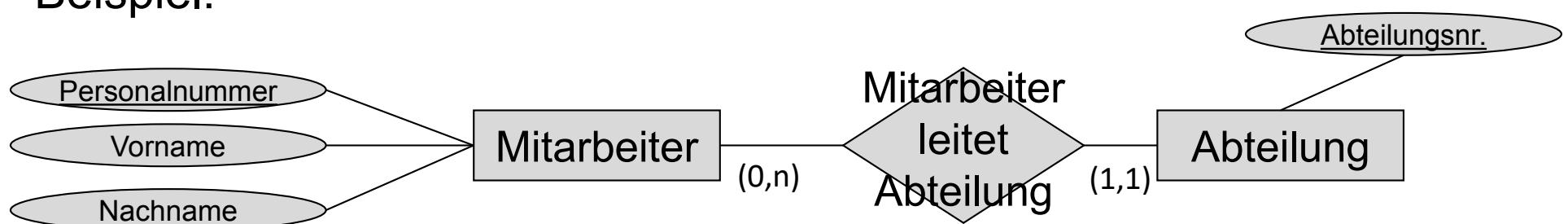
- Relationship (Beziehung)
 - Logische Zuordnungen bzw. Beziehung zwischen Entitytypen, z.B. Kunde kauft Produkt.



- Kardinalität:
 - Gibt an, wie oft ein Entity eines Entitytypen A eine Beziehung mit einem Entity eines Entitytypen B eingehen kann (minimal / maximal).

(n,m)

- Beispiel:

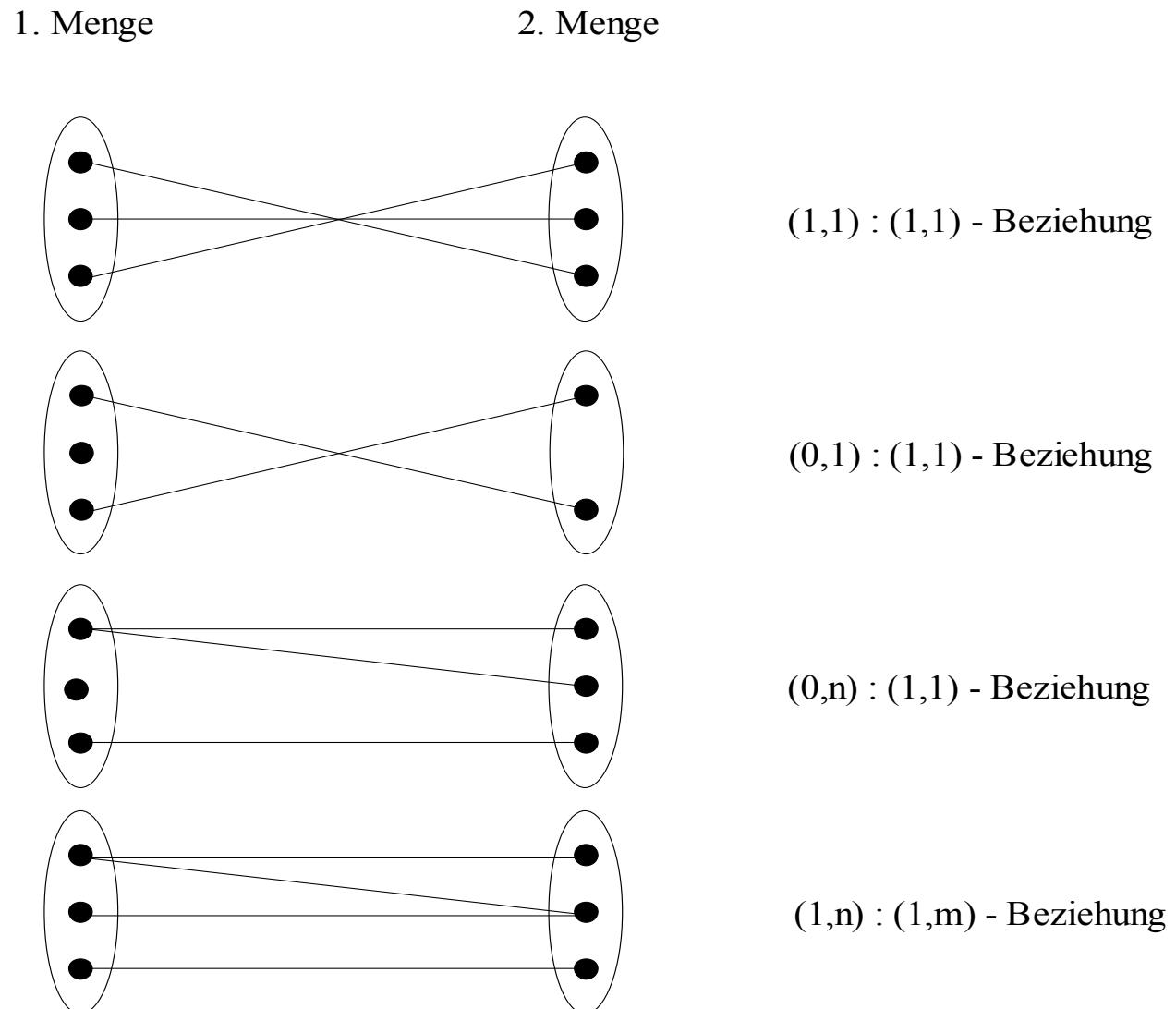


ERM: Namenskonventionen für Relationshiptypen

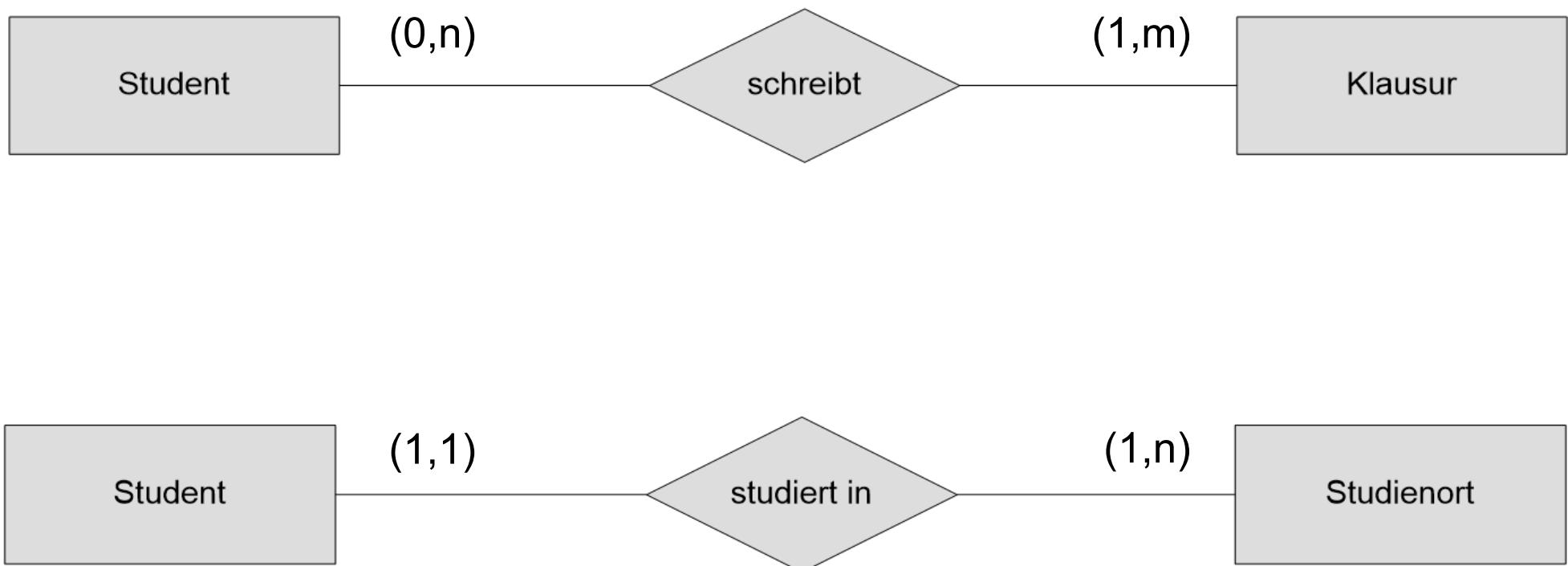
- 1.** <Substantiv, Singular> <Verb, 3. Person Singular>
<Substantiv, Singular>
 - Student hört Vorlesung
 - Kunde kauft Artikel
 - ...
- 2.** Einzelnes, „sprechendes“ Substantiv
 - Kauf
 - Belegung
 - Rechnungskopf
 - ...

Kardinalitäten in der min-max Notation (1/2)

- Relationship-Typen werden durch die Angabe ergänzt, wie oft jedes Entity darin vertreten sein kann

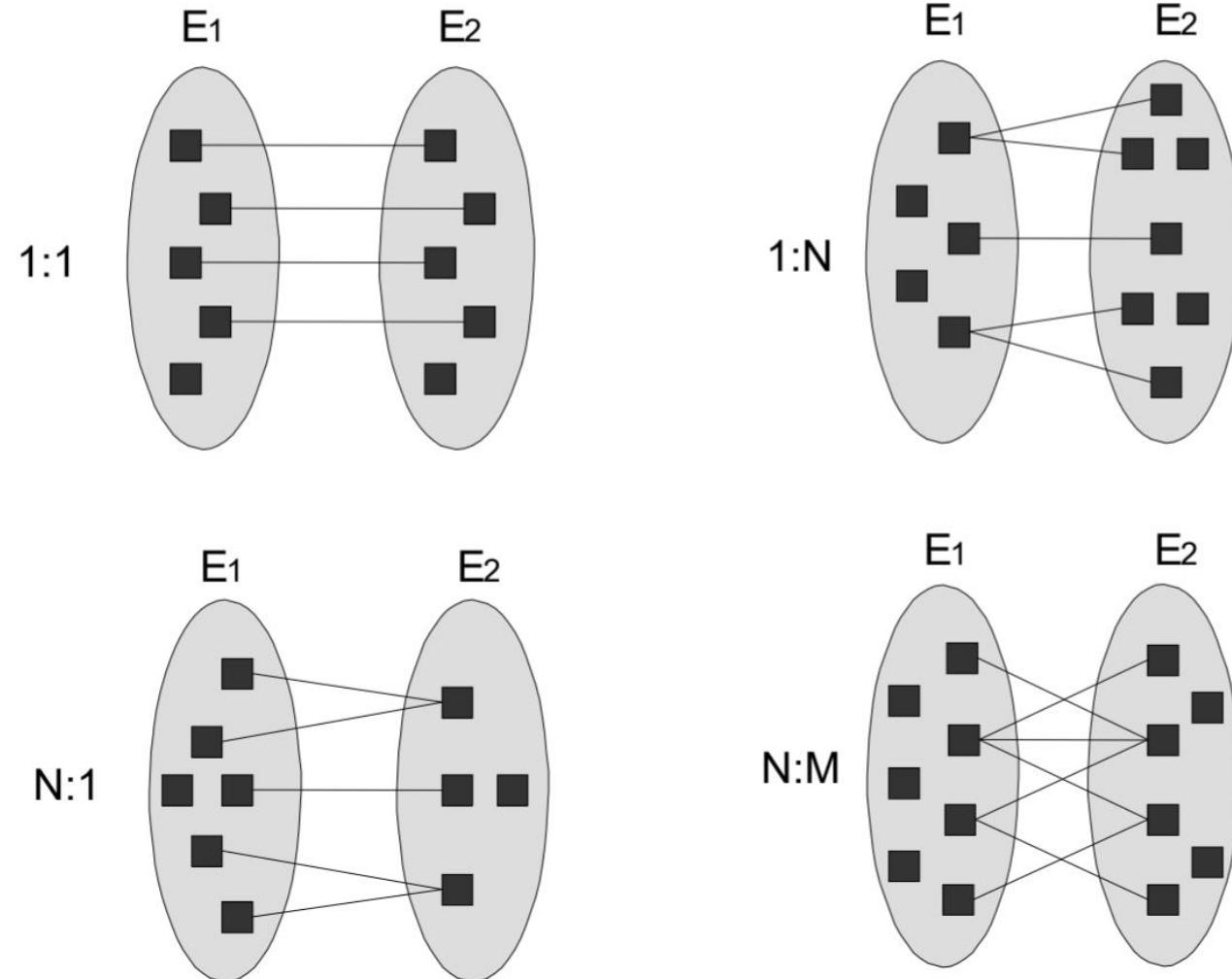


Kardinalitäten in der min-max Notation

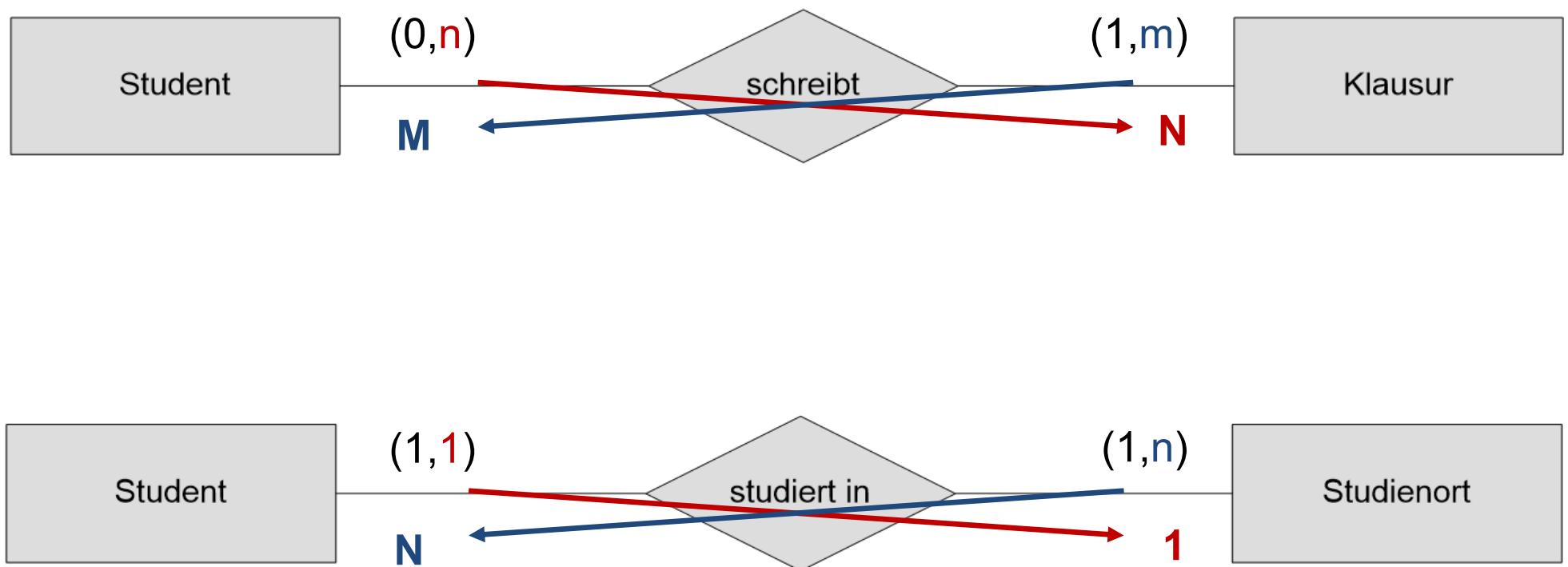


Kardinalitäten in der vereinfachten Notation nach Chen (1/3)

- 1:1 - Beziehung
(one-one)
- 1:N - Beziehung
(one-many)
- N:1 - Beziehung
(many-one)
- N:M - Beziehung
(many-many)

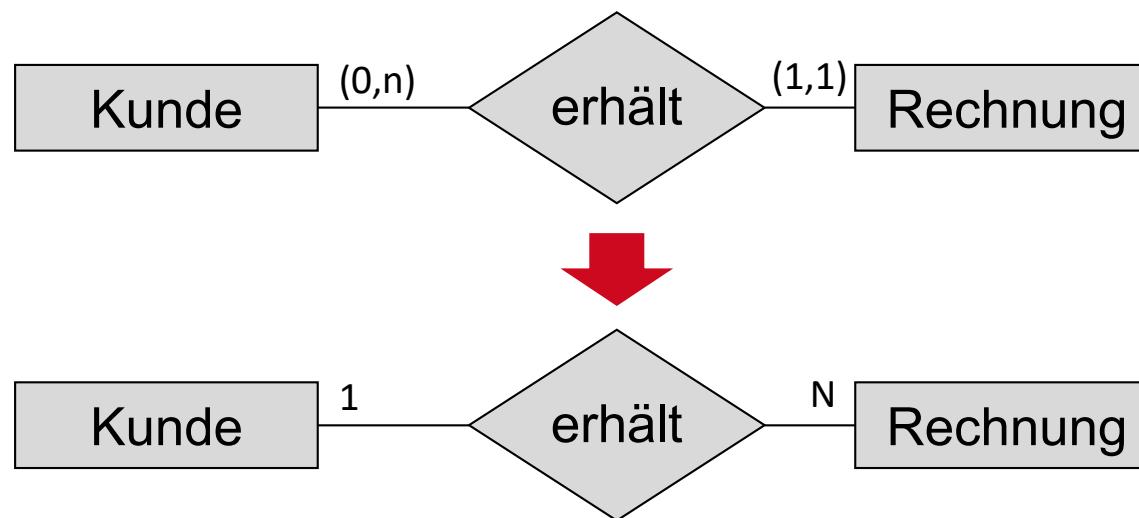


Kardinalitäten in der vereinfachten Notation nach Chen (2/3)



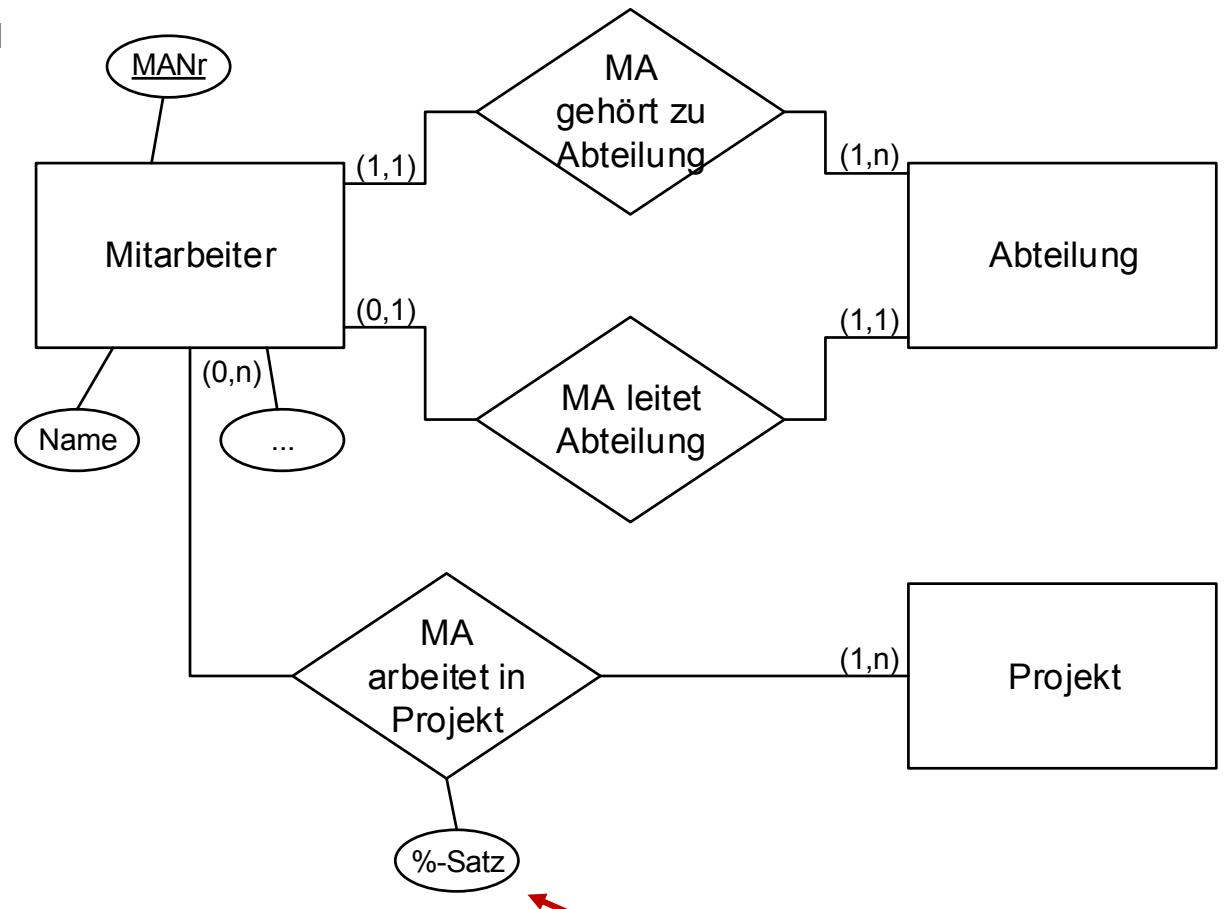
Kardinalitäten in der vereinfachten Notation nach Chen (3/3)

- Die Chen-Notation ist semantisch ärmer als die min,max-Notation. Sie lässt keine Aussagen über die minimale Anzahl korrespondierender Entitäten zu. Dies ist bei der Umwandlung in ein logisches Modell u.U. von Bedeutung.



Kardinalitäten: Beispiel (min-max Notation)

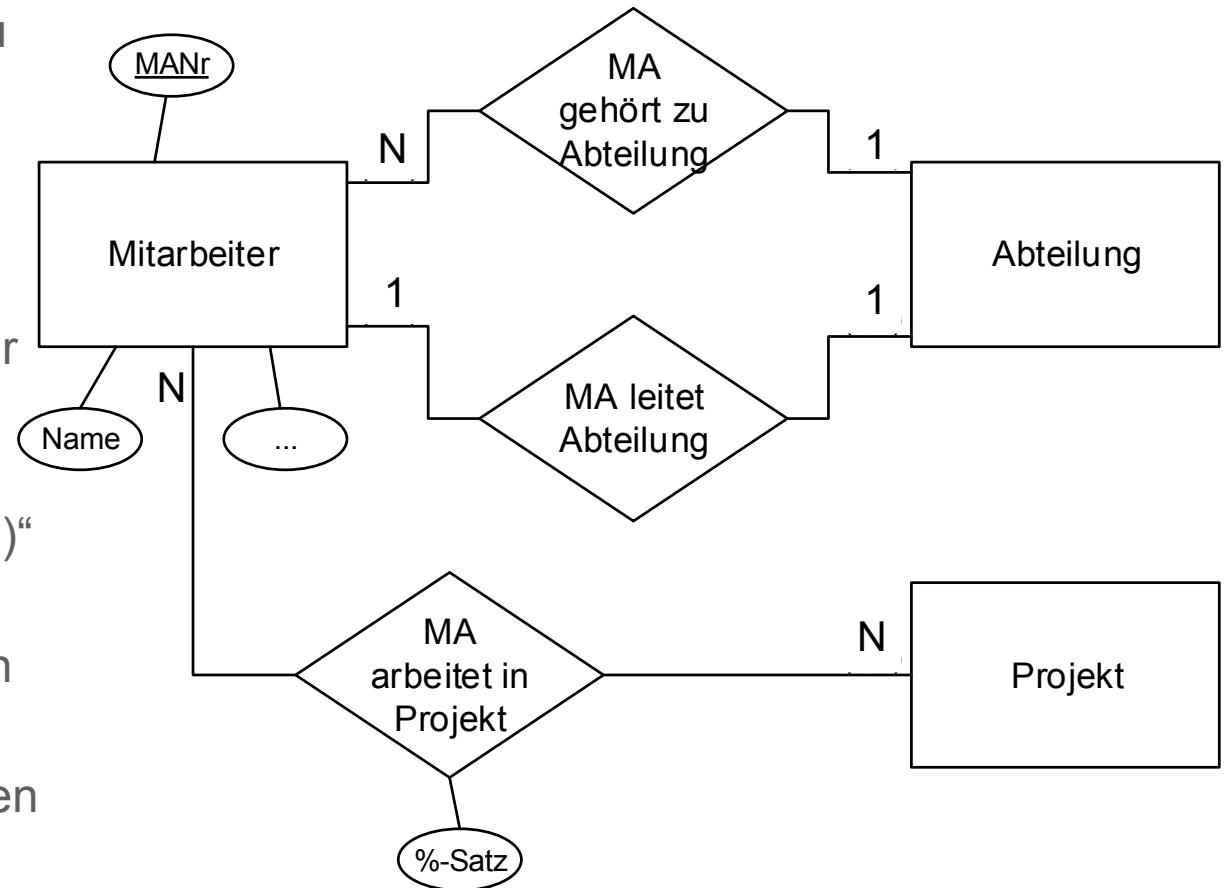
- Ein Mitarbeiter gehört zu genau einer Abteilung: „(1,1)“
- In einer Abteilung arbeiten mindestens ein Mitarbeiter, maximal n Mitarbeiter: „(1,n)“
- Ein Mitarbeiter kann Leiter einer Abteilung sein: „(0,1)“
- Eine Abteilung wird von genau einem Mitarbeiter geleitet: „(1,1)“
- Ein Mitarbeiter kann in keinem, einem oder mehreren Projekten arbeiten: „(0,n)“
- Ein Projekt hat mindestens einen aktiven Mitarbeiter: „(1,n)“



Hinweis: Es sind auch Attribute an Relationship-Typen möglich!

Kardinalitäten: Beispiel (Notation nach Chen)

- Ein Mitarbeiter gehört zu genau einer Abteilung: „(1,1)“
- In einer Abteilung arbeiten mindestens ein Mitarbeiter, maximal n Mitarbeiter: „(1,n)“
- Ein Mitarbeiter kann Leiter einer Abteilung sein: „(0,1)“
- Eine Abteilung wird von genau einem Mitarbeiter geleitet: „(1,1)“
- Ein Mitarbeiter kann in keinem, einem oder mehreren Projekten arbeiten: „(0,n)“
- Ein Projekt hat mindestens einen aktiven Mitarbeiter: „(1,n)“





Fallbeispiel „Vereinsportal“

Bitte helfen Sie der Verwaltung eines Vereins bei der Errichtung eines Mitglieder-Portals. Der Vereinsvorstand beschreibt Ihnen die Anforderungen wie folgt:

- In unserem Portal müssen wir die Mitglieder speichern können. Zu jedem Mitglied soll zunächst nur der Name, Geburtsdatum und eine Telefonnummer erfasst werden. Eindeutig identifizieren wir Mitglieder über den Namen und das Geburtsdatum.
- Wir sind ein lokaler Verein und nehmen bisher nur Mitglieder aus unserer Stadt auf. Daher soll zu jedem Mitglied aus einer Liste der Stadtteil zugeordnet werden können (eindeutig über den Namen des Stadtteils identifizierbar).
- Ein Mitglied übernimmt eine oder mehrere Rollen im Verein, muss dies jedoch nicht tun. Bei den Rollen haben wir uns RollenIDs und Bezeichnungen ausgedacht. Es gibt auch für jeden Stadtteil jeweils eine Rolle für den Stadtteilverantwortlichen, d.h. eine Rolle kann einem Stadtteil zugeordnet werden.
- Zusätzlich haben wir auch noch eine Rollenhierarchie.

→ **Welche Beziehungen (Relationship-Typen) können Sie im Fallbeispiel identifizieren? Wie lauten die Kardinalitäten?** Ignorieren Sie zunächst die Hinweise auf Hierarchien!

Übung



Anforderungen

- Das System wird von Benutzern (Kunden & Mitarbeiter), die durch ihre E-Mailadresse identifiziert werden, bedient.
 - Alle Benutzer registrieren sich mit Name und Vorname.
 - Kunden sollen darüber hinaus in der Lage sein, bei der Registrierung ihre Adresse (Str., Hausnummer, PLZ, Ort) und ein Geburtsdatum zu hinterlegen. Für Kunden wird automatisch eine ID angelegt.
 - Mitarbeiter hinterlegen ihre eindeutige Personal- und Tel.-Nr. Zusätzlich sollen Sie ihrer Abteilung in einer Abteilungshierarchie zugeordnet werden.
- Registrierte Kunden sollen Produkte in ihren Warenkorb „legen“ können (auch mehr als 1x).
 - Dabei bekommt jeder Warenkorb eine eindeutige ID und das Erstelldatum wird auch gespeichert.
 - Produkte haben eine eindeutige Nummer und in der Datenbank sollen zusätzlich die Bezeichnung, die aktuelle Verfügbarkeit, der Listenpreis und das Anlagedatum des Produkts in den Stammdaten gespeichert werden.
 - Produkte gehören zu Kategorien; diese wiederum haben eine Hierarchie (bspw. Banane → Obst → Lebensmittel).
- Wenn ein Kunde einen Warenkorb bestellen möchte, wird eine entsprechende Bestellung im System angelegt und dem Warenkorb zugeordnet.
 - So kann eine Historie der Bestellungen aller Kunden aufgebaut werden. Bestellungen setzen einen Warenkorb voraus, in dem die zu bestellenden Produkte und deren Bestellmenge hinterlegt werden.
 - Bestellungen werden von einem Mitarbeiter bearbeitet und der Kunde kann sich über die Website über den aktuellen Status seiner Bestellung informieren.

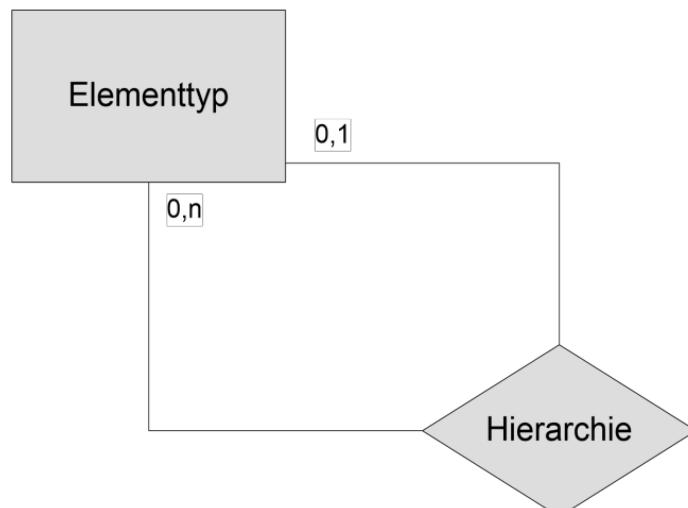


■ Welche Beziehungen (Relationship-Typen) können Sie im Fallbeispiel identifizieren? Wie lauten die Kardinalitäten? Ignorieren Sie zunächst die Hinweise auf Hierarchien!

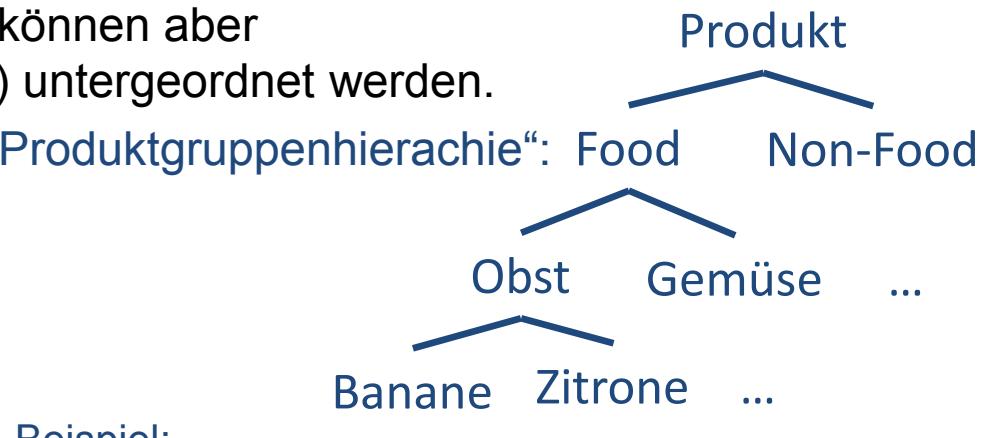
- Bearbeitungszeit: 15min individuell, 10min Kleingruppendiskussion (2-3 Studierende)
- Im Anschluss: Vorstellung und gemeinsame Diskussion

Hierarchien im ERM

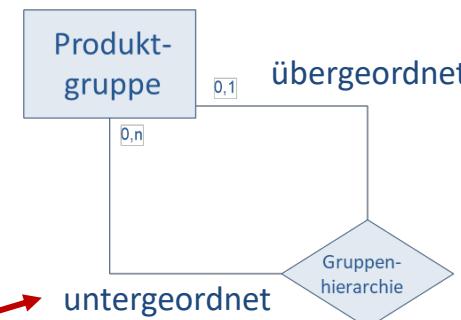
- Hierarchien setzen Entities in eine Über- /Unterordnungsbeziehung.
- Damit werden Baumstrukturen abgebildet.
 - D.h. einem Element darf maximal ein anderes Element (gleichen Typs) übergeordnet werden, einem Element können aber beliebig viele Elemente (gleichen Typs) untergeordnet werden.
 - → Kardinalität (0,1) – (0,n) Beispiel „Produktgruppenhierarchie“:



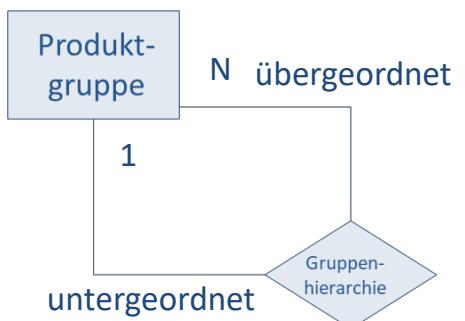
Hinweis: derartige Rollenbezeichner
können die Lesbarkeit erhöhen!



Beispiel:
in (min,max)-Notation:



in Chen-Notation:





Fallbeispiel „Vereinsportal“

Bitte helfen Sie der Verwaltung eines Vereins bei der Errichtung eines Mitglieder-Portals. Der Vereinsvorstand beschreibt Ihnen die Anforderungen wie folgt:

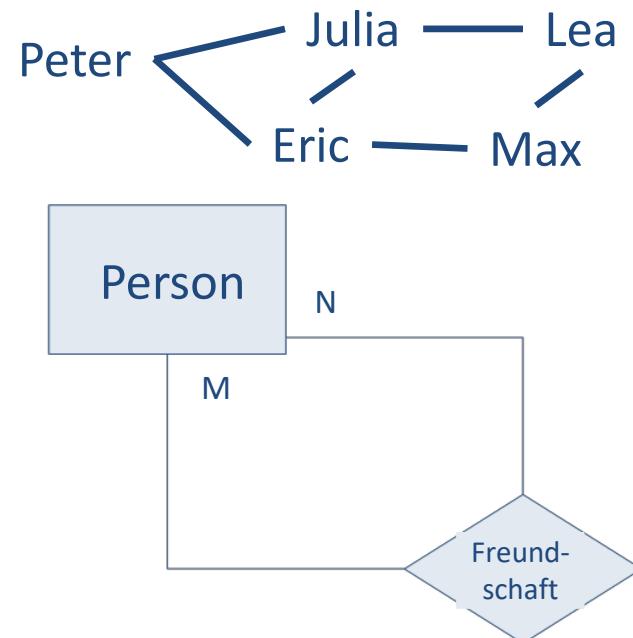
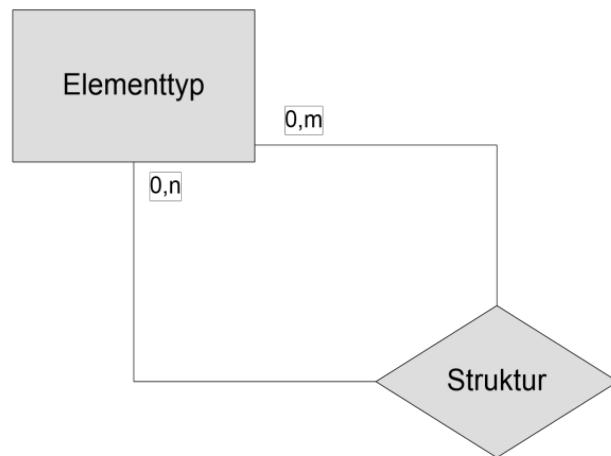
- In unserem Portal müssen wir die Mitglieder speichern können. Zu jedem Mitglied soll zunächst nur der Name, Geburtsdatum und eine Telefonnummer erfasst werden. Eindeutig identifizieren wir Mitglieder über den Namen und das Geburtsdatum.
- Wir sind ein lokaler Verein und nehmen bisher nur Mitglieder aus unserer Stadt auf. Daher soll zu jedem Mitglied aus einer Liste der Stadtteil zugeordnet werden können (eindeutig über den Namen des Stadtteils identifizierbar).
- Ein Mitglied übernimmt eine oder mehrere Rollen im Verein, muss dies jedoch nicht tun. Bei den Rollen haben wir uns RollenIDs und Bezeichnungen ausgedacht. Es gibt auch für jeden Stadtteil jeweils eine Rolle für den Stadtteilverantwortlichen, d.h. eine Rolle kann einem Stadtteil zugeordnet werden.
- Zusätzlich haben wir auch noch eine Rollenhierarchie.

→ **Modellieren Sie die Rollenhierarchie?**

Strukturen im ERM

- Strukturen im ERM setzen Entitiy-Typen mit beliebig vielen Entity-Typen gleichen Typs in Verbindung.
- Keine klare Richtung der Beziehungen notwendig.
 - → Kardinalität: (0, n) – (0, m)

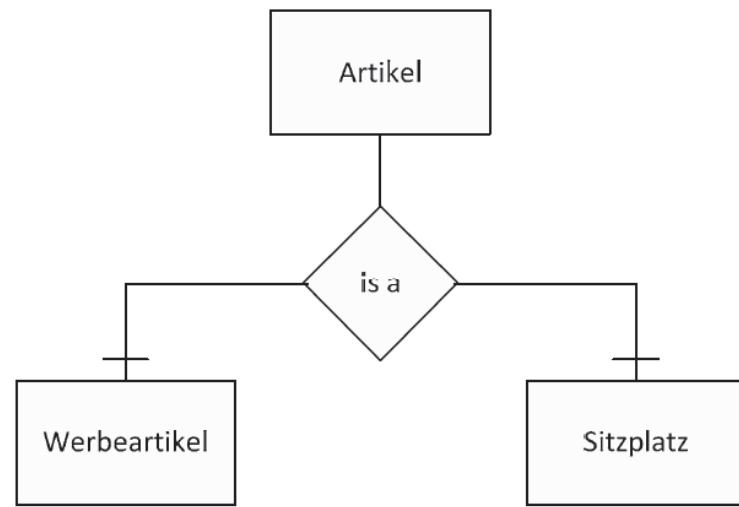
Beispiel „soziales Netzwerk“:



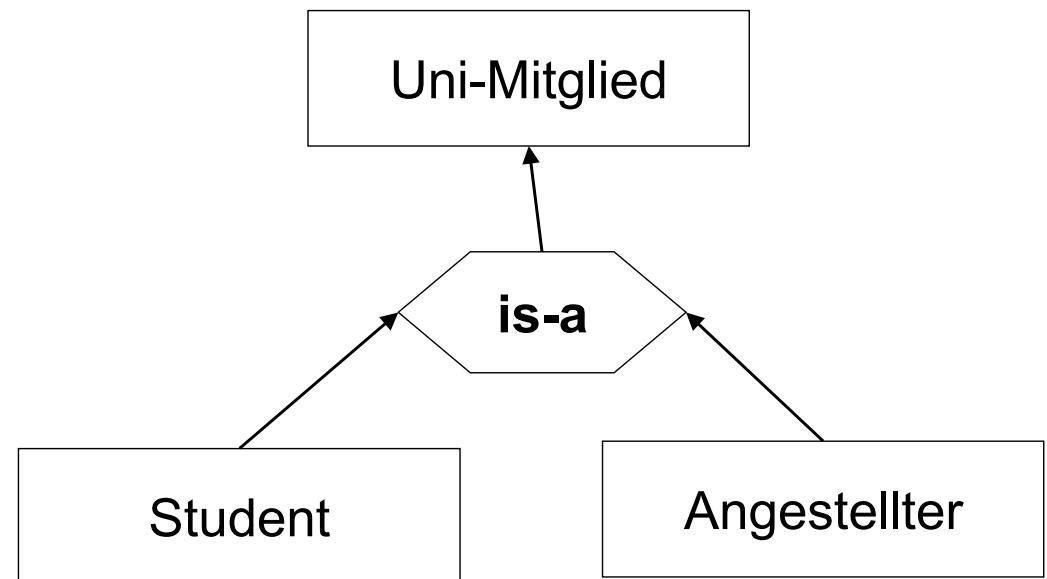
Generalisierung / Spezialisierung im ERM

- Für Sub- bzw. Supertypen gab es ursprünglich keine grafische Notation. Sie wurde erst im Laufe der Jahre nachträglich hinzugefügt.

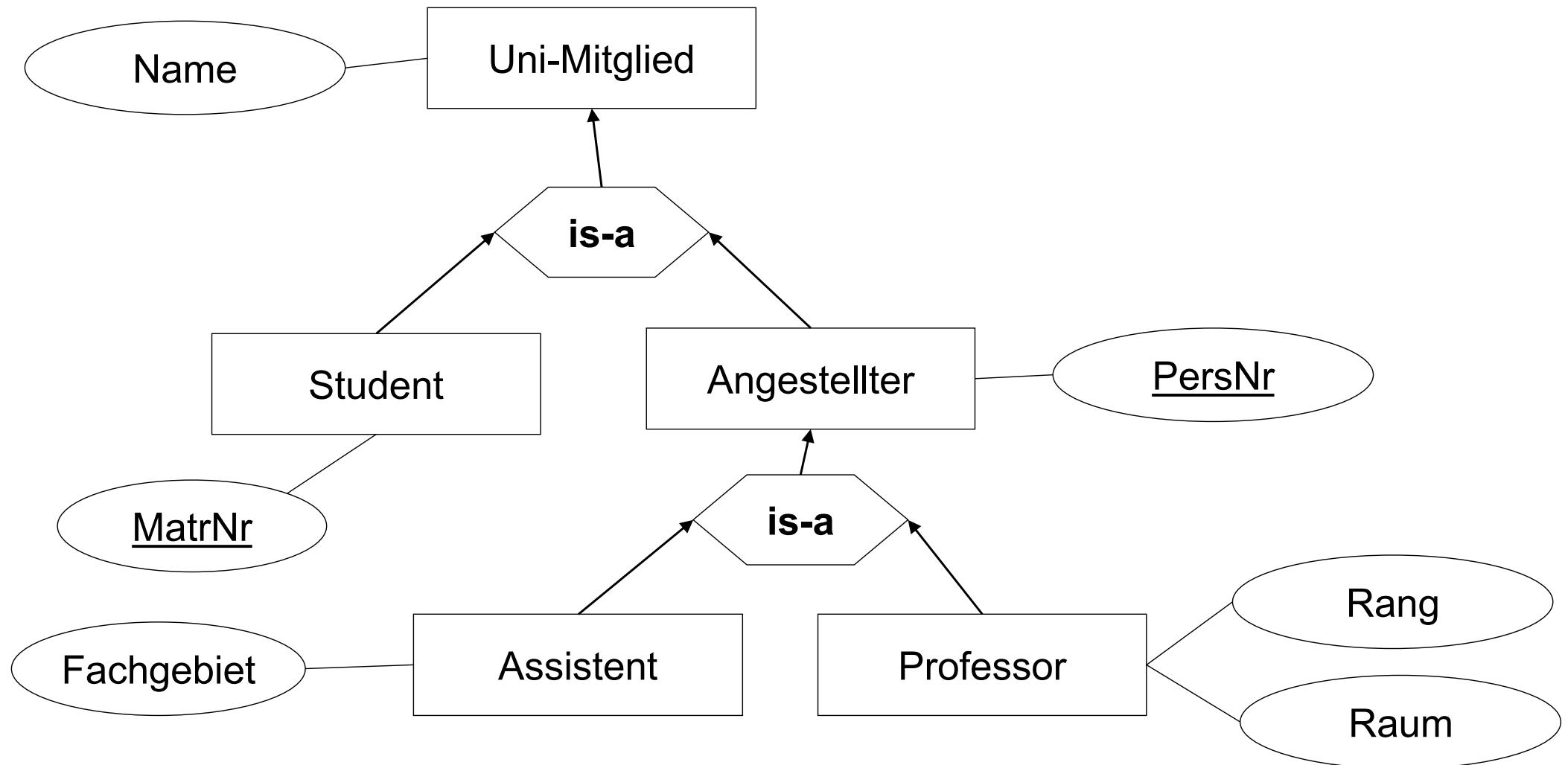
Möglichkeit 1:



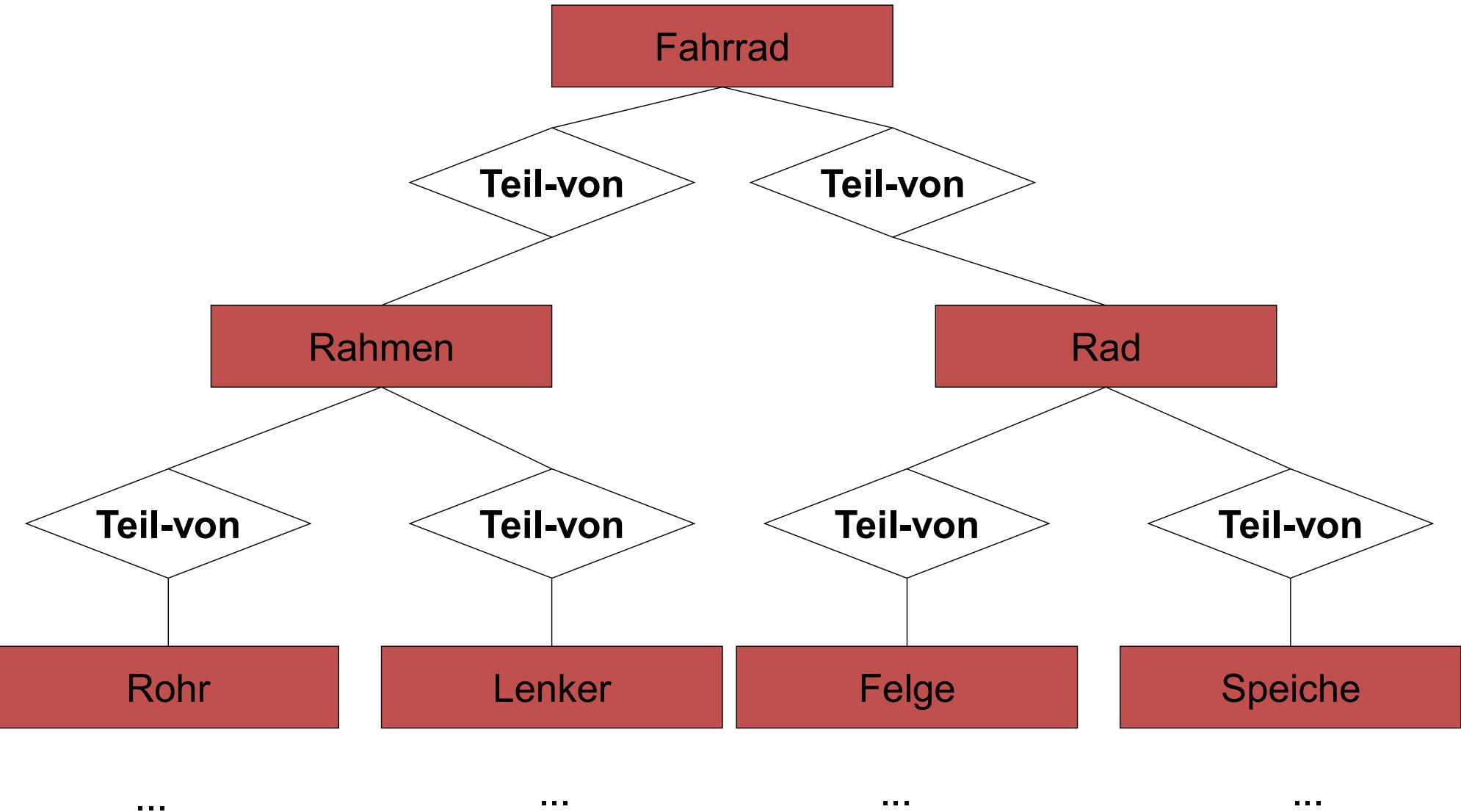
Möglichkeit 2:



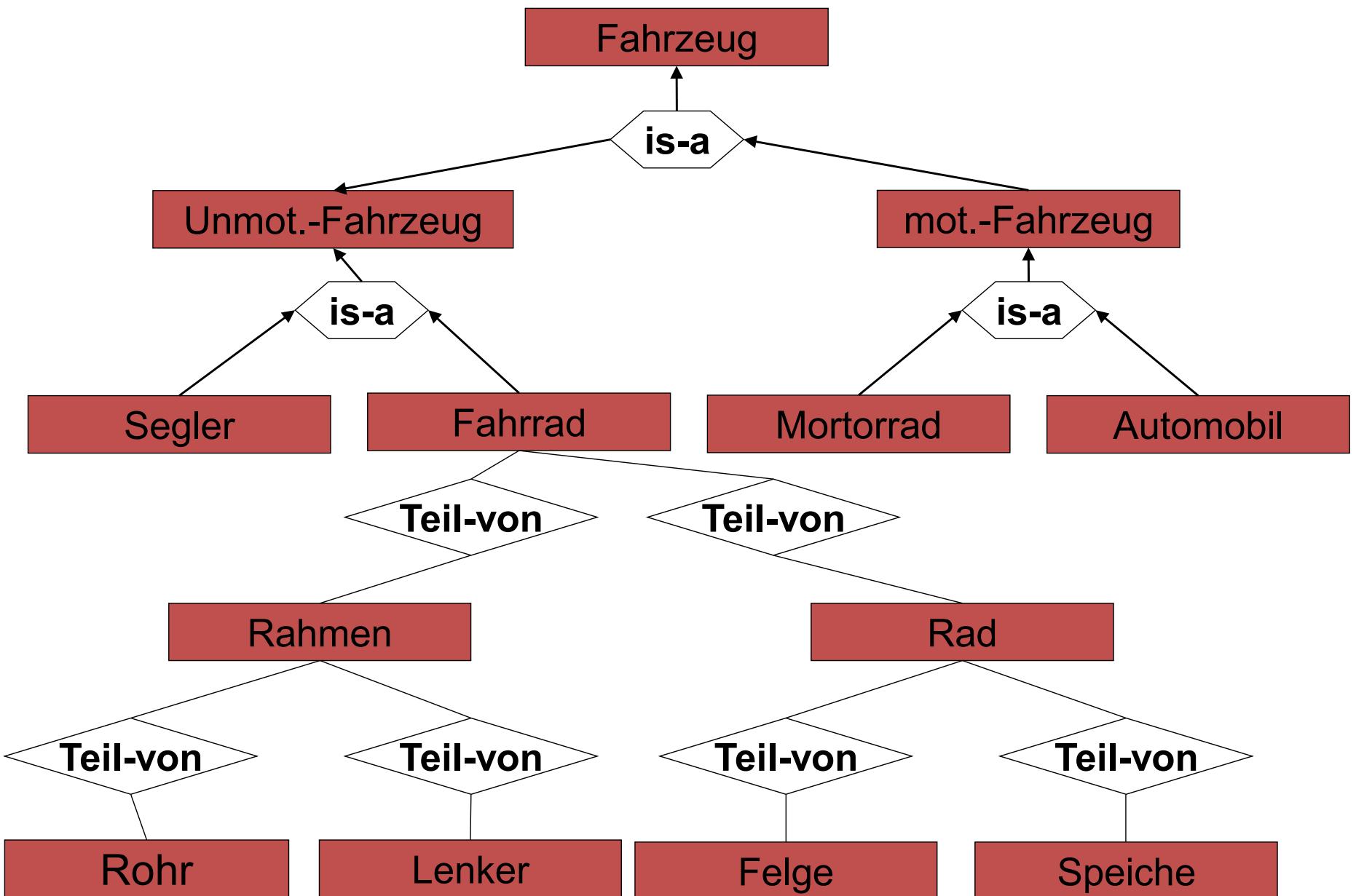
Generalisierung / Spezialisierung im ERM: Beispiel



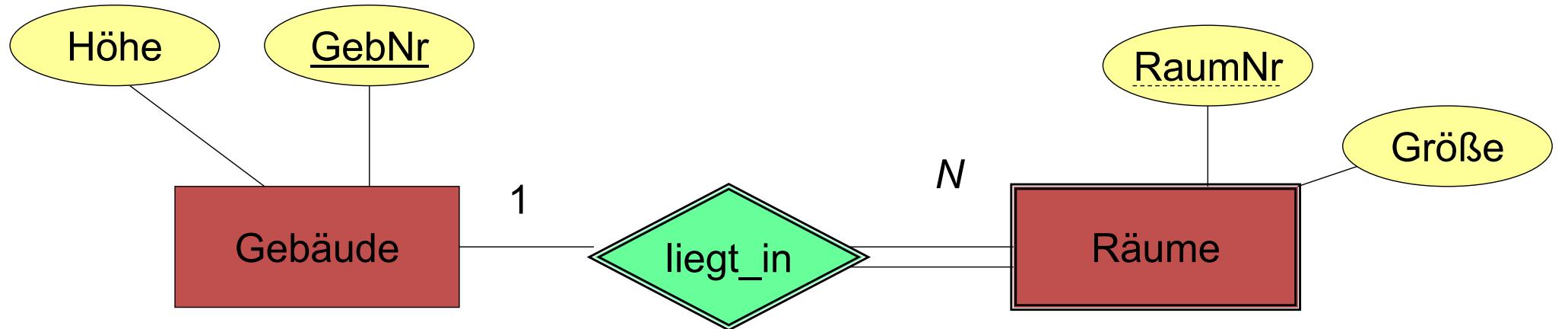
Aggregation im ERM



Aggregation und Generalisierung im ERM



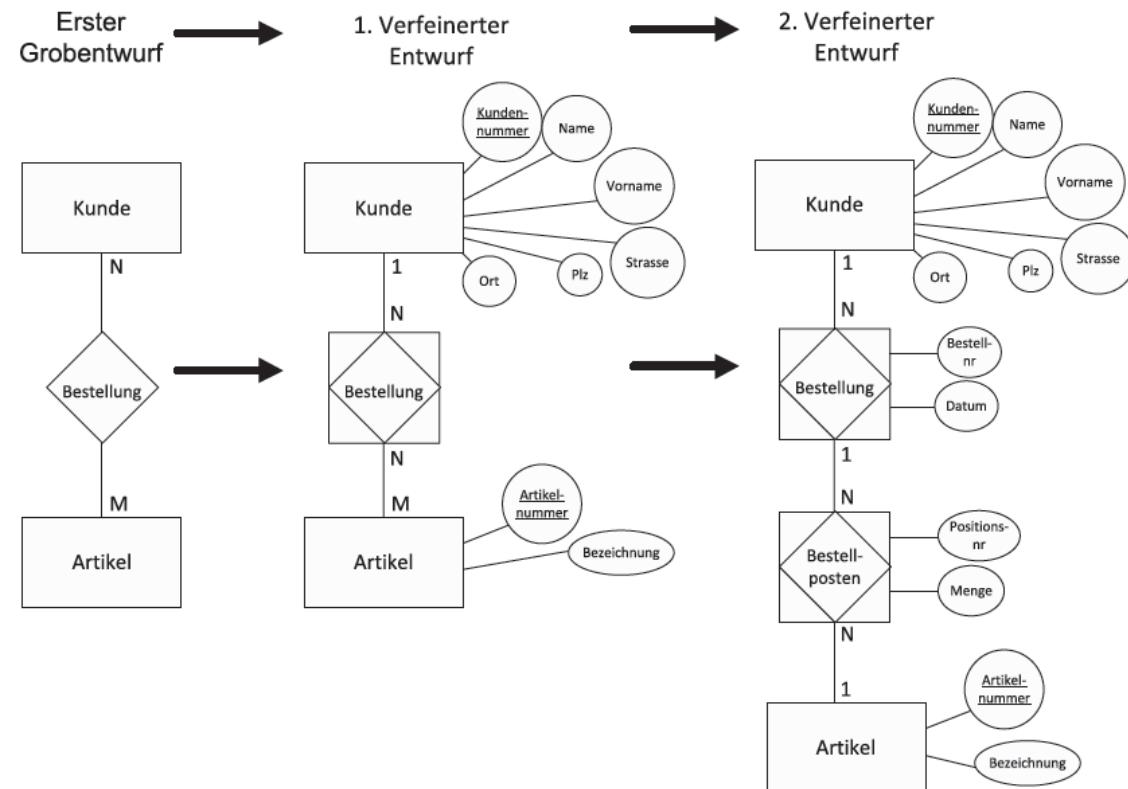
Schwache Entitytypen



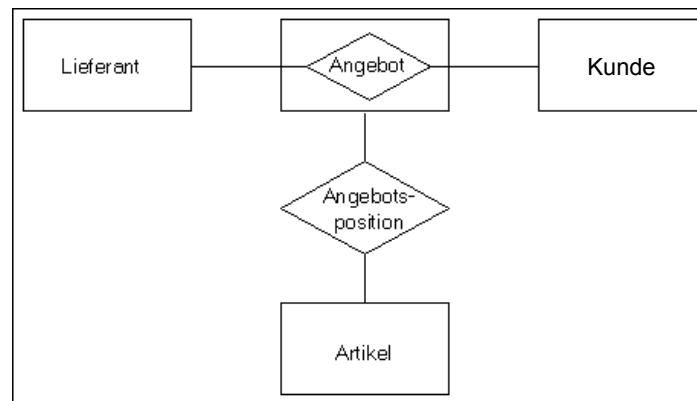
- Sogenannte schwache Entities können nicht autonom existieren.
 - Sie sind in ihrer Existenz von einem anderen, übergeordneten Entitytyp abhängig.
 - Sie sind nur in Kombination mit dem Schlüssel des übergeordneten Entity eindeutig identifizierbar.
 - Schwache Entities werden durch doppelt gerahmte Rechtecke repräsentiert und ihre Beziehung zum übergeordneten Entity-Typ durch eine Verdoppelung der Raute und der von dieser Raute zum schwachen Entity-Typ ausgehenden Kante markiert.
- Im Beispiel:
 - RaumNr ist nur innerhalb eines Gebäudes eindeutig (gestrichelt-unterstrichen!)
 - Schlüssel ist: GebNr und RaumNr

Sonderfall N:M-Beziehungen / Uminterpretation

- Wenn man sich der logischen Modellierung durch ein relationales Schema nähert, können N:M-Beziehungen folgendermaßen angepasst werden:



- Diese Uminterpretation von Relationship-Typen kann auch so genutzt werden:

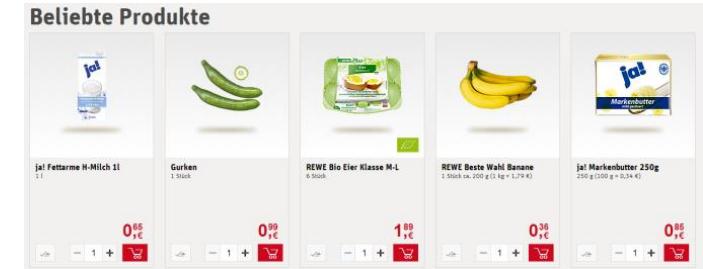


Übung



Anforderungen

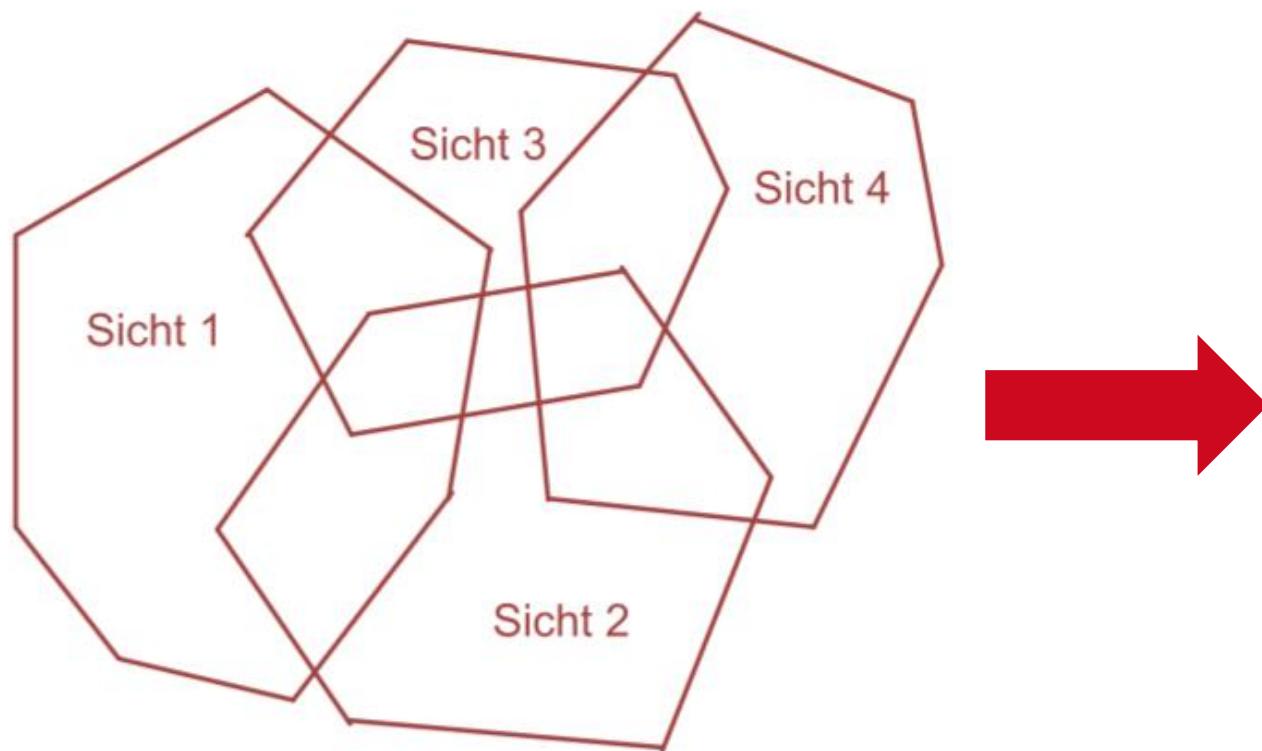
- Das System wird von Benutzern (Kunden & Mitarbeiter), die durch ihre E-Mailadresse identifiziert werden, bedient.
 - Alle Benutzer registrieren sich mit Name und Vorname.
 - Kunden sollen darüber hinaus in der Lage sein, bei der Registrierung ihre Adresse (Str., Hausnummer, PLZ, Ort) und ein Geburtsdatum zu hinterlegen. Für Kunden wird automatisch eine ID angelegt.
 - Mitarbeiter hinterlegen ihre eindeutige Personal- und Tel.-Nr. Zusätzlich sollen Sie ihrer Abteilung in einer Abteilungshierarchie zugeordnet werden.
- Registrierte Kunden sollen Produkte in ihren Warenkorb „legen“ können (auch mehr als 1x).
 - Dabei bekommt jeder Warenkorb eine eindeutige ID und das Erstelldatum wird auch gespeichert.
 - Produkte haben eine eindeutige Nummer und in der Datenbank sollen zusätzlich die Bezeichnung, die aktuelle Verfügbarkeit, der Listenpreis und das Anlagedatum des Produkts in den Stammdaten gespeichert werden.
 - Produkte gehören zu Kategorien; diese wiederum haben eine Hierarchie (bspw. Banane → Obst → Lebensmittel).
- Wenn ein Kunde einen Warenkorb bestellen möchte, wird eine entsprechende Bestellung im System angelegt und dem Warenkorb zugeordnet.
 - So kann eine Historie der Bestellungen aller Kunden aufgebaut werden. Bestellungen setzen einen Warenkorb voraus, in dem die zu bestellenden Produkte und deren Bestellmenge hinterlegt werden.
 - Bestellungen werden von einem Mitarbeiter bearbeitet und der Kunde kann sich über die Website über den aktuellen Status seiner Bestellung informieren.



■ Wie können im Anwendungsbeispiel die Hierarchien sowie Spezialisierung modelliert werden?

- Bearbeitungszeit: 5min individuell, 5min Kleingruppendiskussion (2-3 Studierende)
- Im Anschluss: Vorstellung und gemeinsame Diskussion

Konsolidierung von Sichten (1/3)



Globales Schema

- Ohne Redundanzen
- Ohne Widersprüche
- Synonyme bereinigt
- Homonyme bereinigt

Synonym: Dozent und Professor
Homonym: Bank

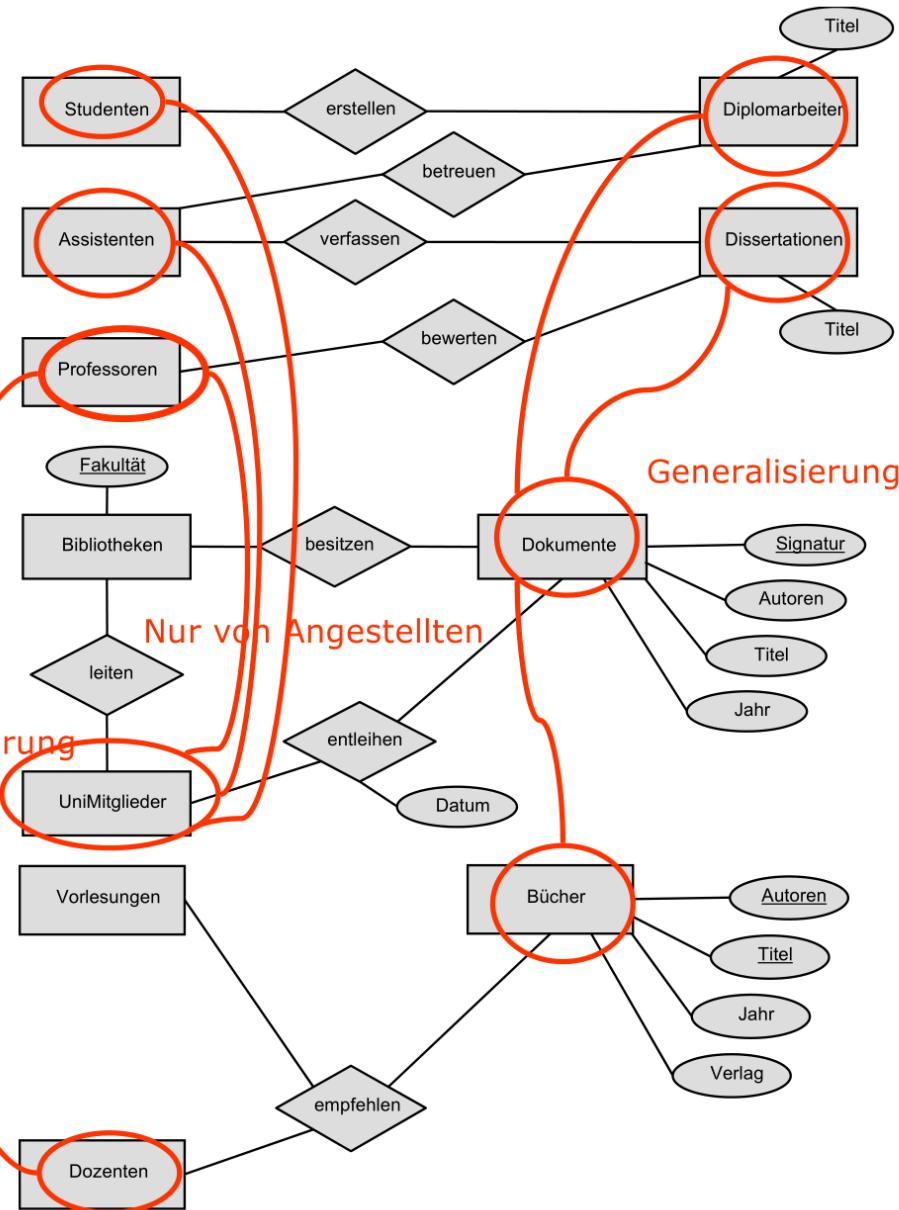
Konsolidierung von Sichten (2/3)

Drei Sichten einer Universitätsdatenbank

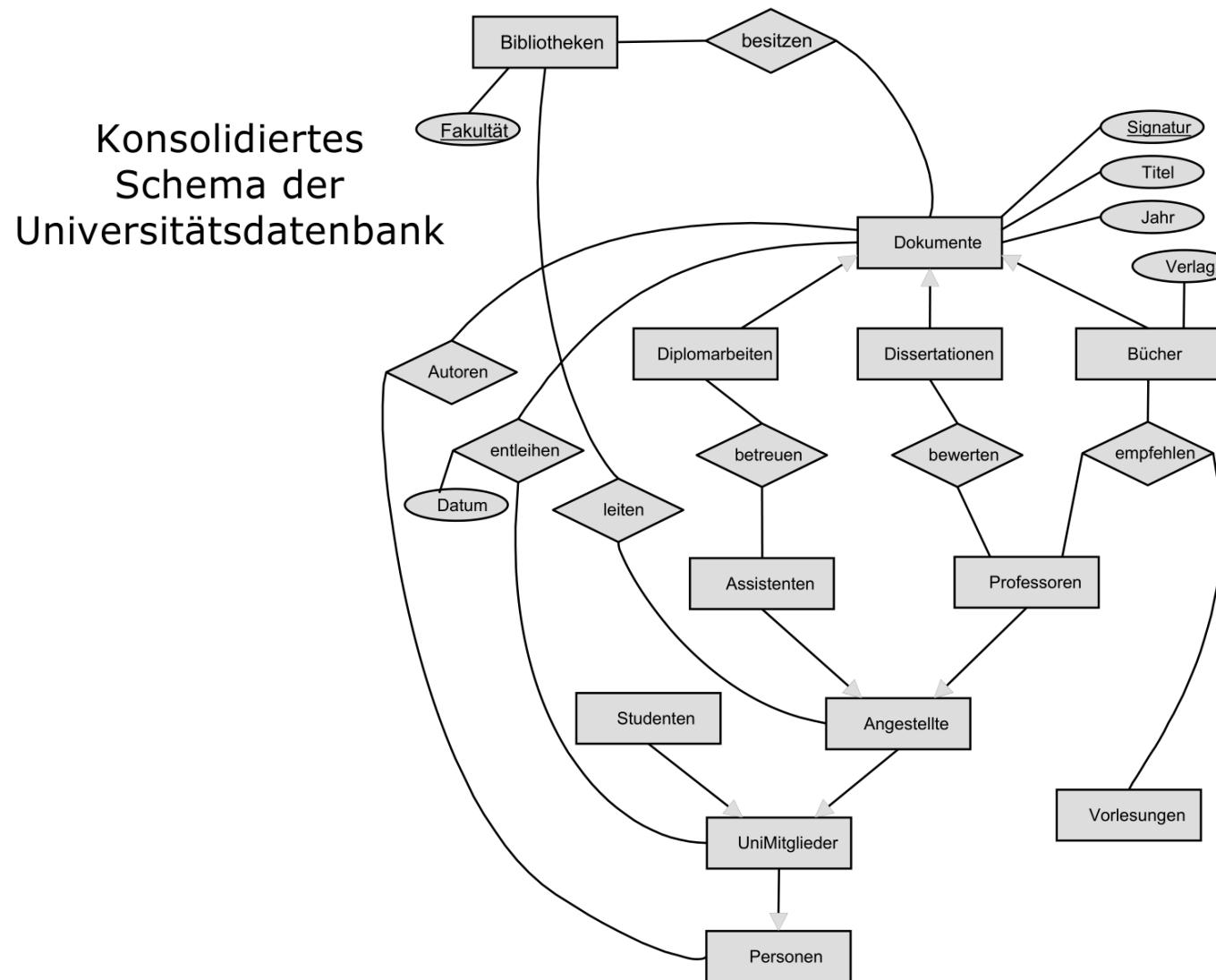
Sicht 1:
Erstellung von Dokumenten
als Prüfungsleistung

Sicht 2: Bibliotheksverwaltung

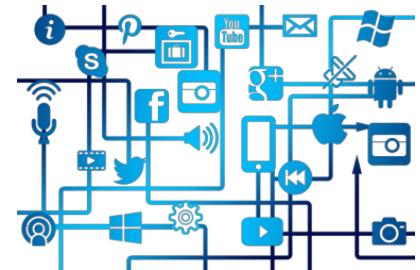
Sicht 3: Buchempfehlungen für Vorlesungen



Konsolidierung von Sichten (3/3)

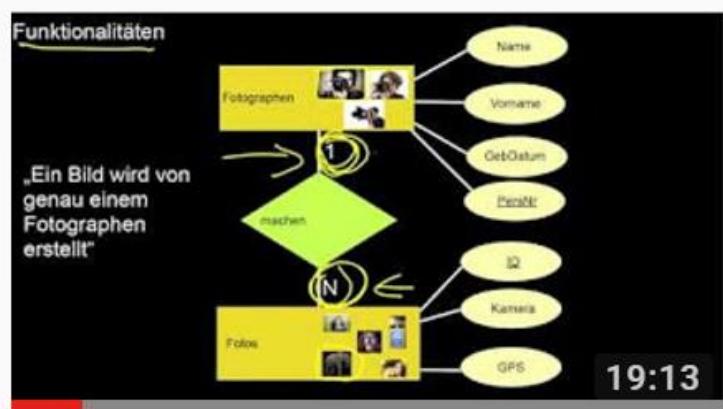


Weiterführendes Material



- Video zu ERM

<https://www.youtube.com/watch?v=F5rTvnbjPq8>



13.08 Entity Relationship Modellierung: Grundlagen,

Jens Dittrich • 82.000 Aufrufe • vor 4 Jahren

Komplette Liste der Videos und zusätzliches Material auf
<http://datenbankenlernen.de> Informatik, Uni Saarland:

- Hinweis: die Rollenzuweisung zu Beziehungstypen nutzen wir nicht.

**BW342 – Datenbanken I
(mit Praktikum)**

**Vorlesungen 3 & 4
Logische Datenmodellierung**



Ziele von VL 3 & 4



- Das **relationale Modell** als eine Möglichkeit der logischen Datenmodellierung kennenlernen und anwenden können.
- Konzeptionelle Modelle (ERM) in relationale Modelle **überführen** können.
- Die **Normalisierung** von relationalen Modellen kennenlernen und selbst durchführen können.
- Einen Überblick über Aspekte und Herausforderungen in Bezug auf **Datenintegrität** kennenlernen.

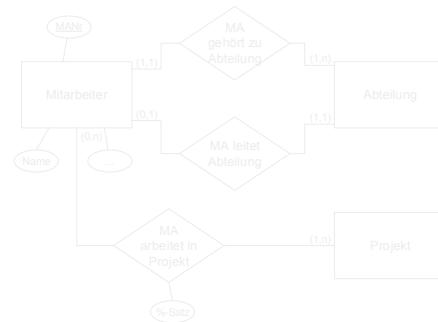
Themenüberblick



A Datenbankgrundlagen



B Konzeptionelle Modellierung



C Logische Modellierung

MID	Name	Adresse	Gehalt
1234	Müller	Hofstraße 7	23.000 €
1235	Meyer	Hauptstraße 13b	45.678 €
1236	Schmidt	Kurze Straße 1	55.555 €
1237	Heinze	Neue Straße 34	44.333 €
1238	Mustermann	Musterstraße 8	80.000 €
1239

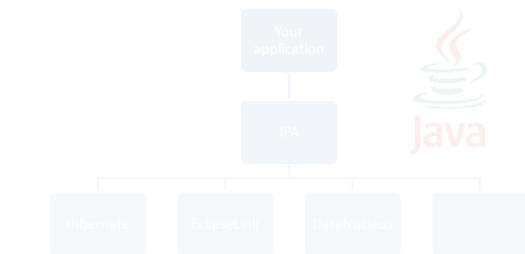
D Structured Query Language

- filter your columns
`SELECT col1, col2, col3, ... FROM table1`
- filter the rows
`WHERE col4 = 1 AND col5 = 2`
- aggregate the data
`GROUP by ...`
- limit aggregated data
`HAVING count(*) > 1`
- order of the results
`ORDER BY col2`

E Anwendungsanbindung (JDBC)



F Anwendungsanbindung (JPA)



■ Relationales Datenbankmodell

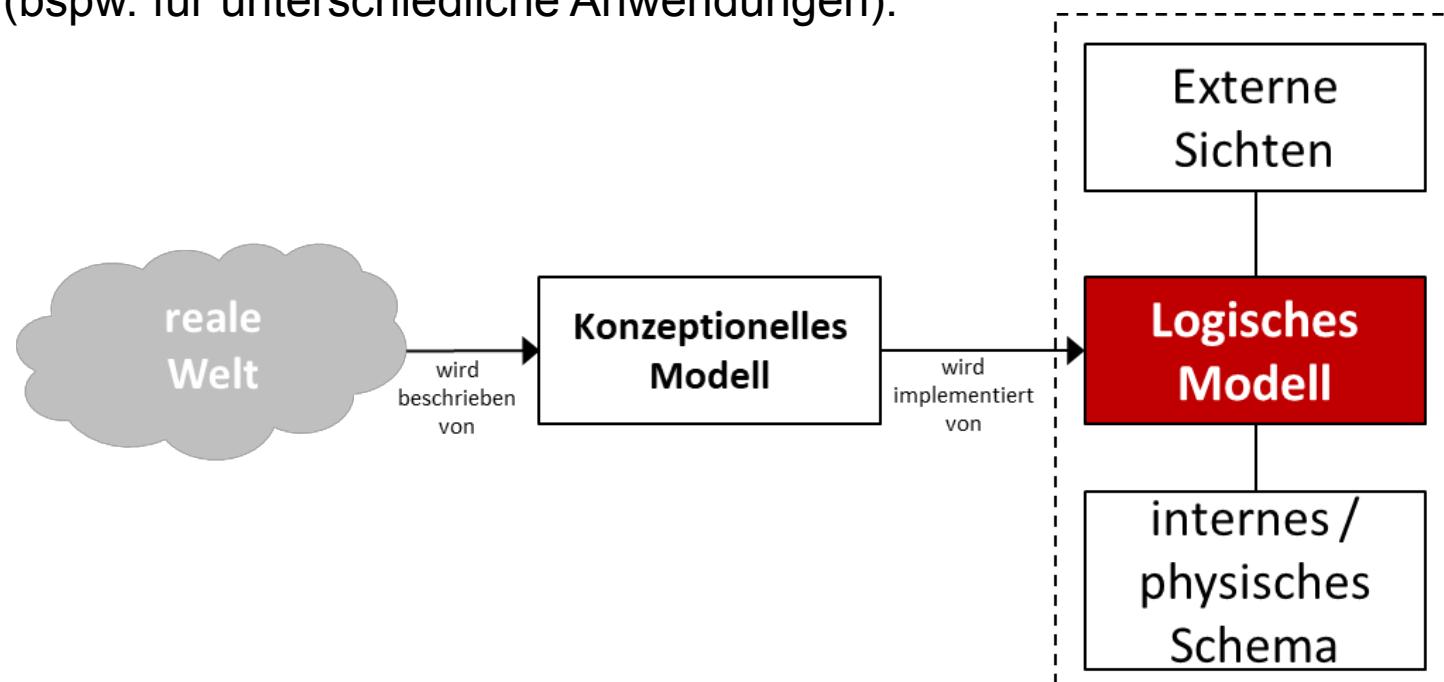
Normalisierung

Datenintegrität

Zusammenfassung und Ausblick

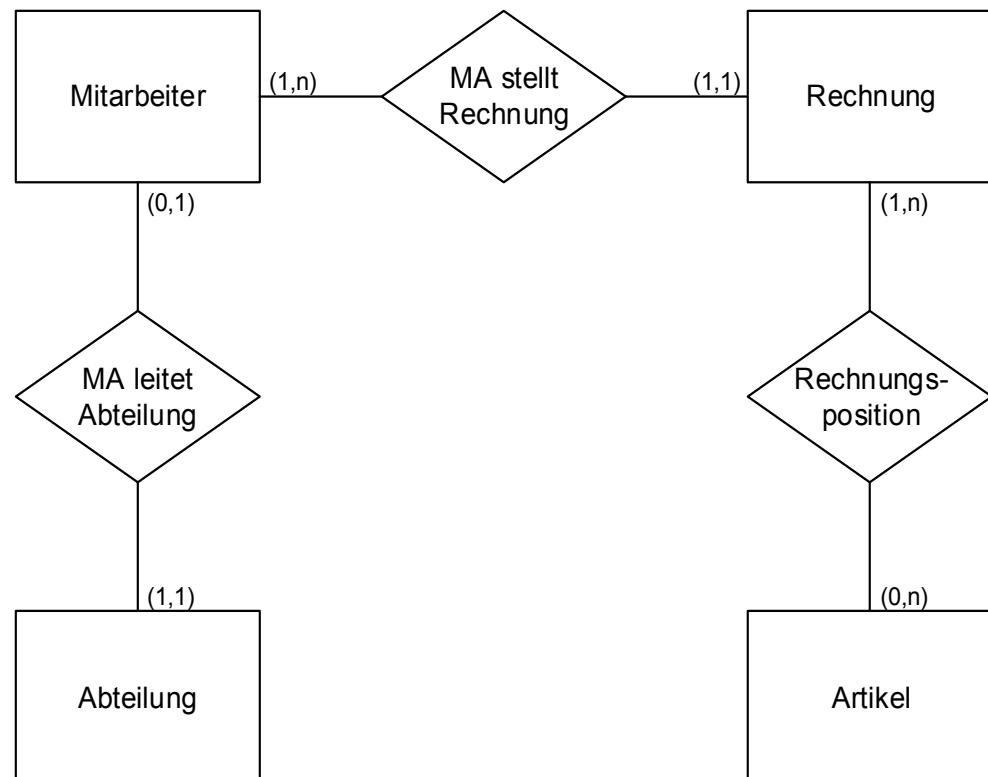
Einordnung

- Wir befassen uns nun mit der logischen Datenmodellierung.
 - Diese basiert auf dem konzeptionellen Modell der betrachteten Domäne.
 - *Wir betrachten nur das relationale Modell als logisches Datenmodell.*
 - Das logische Modell kann in ein physisches Schema überführt werden. Dies übernimmt dann das konkrete DBMS.
 - Aus dem logischen Gesamtmodell können einzelne externe Sichten erzeugt werden (bspw. für unterschiedliche Anwendungen).



Vom Modell zur Datenbank

- Wie können aus einem ERM direkt Strukturen in der Datenbank erzeugt werden?



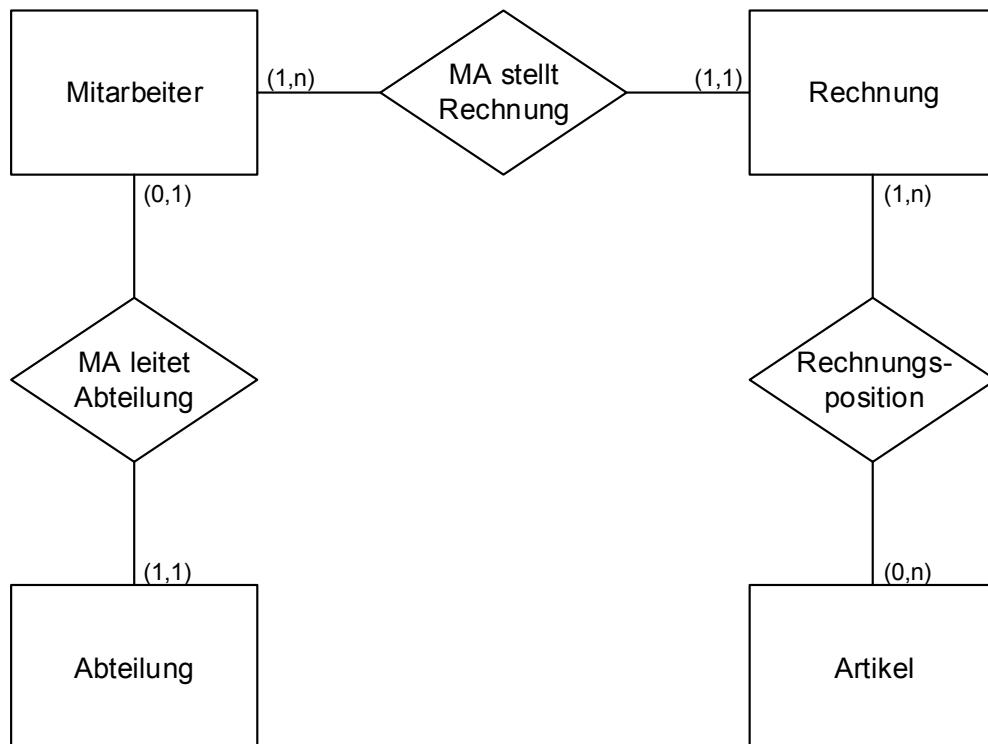
Wie viele Tabellen
werden benötigt?

Relationale Datenbankmodelle

- Darstellung in Tabellenform
 - Zeilen enthalten Objekte (Entitäten)
 - Spalten enthalten Attribute
 - Attribute verfügen über einen Wertebereich (Domäne)
 - Schlüsselattribute werden ausgezeichnet (Unterstrichen)
- Relationale DBMS (RDBMS) beherrschen derzeit den Markt
 - Z.B. Oracle, IBM DB2, MS SQL Server, MySQL, PostgreSQL, Sybase, Informix, CA Ingres.
 - Die meisten neuen DB-Projekte nutzen RDBMS.
 - Es gibt noch Systeme, die auf einem Netzwerk- oder Hierarchie-DBMS beruhen, z.B. das hierarchische System IMS von IBM (vgl. VL1&2).
 - Seit ein paar Jahren gibt es darüber hinaus auch sogenannte NoSQL-Datenbanken (vgl. Vorlesungen 1&2). Diese bieten eine moderne Alternative zu relationalen Datenbanken, werden in dieser Veranstaltung aber nicht behandelt (→ MW243)

Entity-Typ

- Ein Entity-Typ wird in einer Tabelle abgebildet.
- Seine Attribute werden als Feldnamen verwendet.
- Schlüssel festlegen (unterstreichen)!



Notwendige Tabellen

Abteilung	(<u>Abt.-ID</u>)
Mitarbeiter	(<u>MID</u>)
Rechnung	(<u>RID</u>)
Artikel	(<u>AID</u>)

Erste Tabellen

Mitarbeiter			
<u>MID</u>	Name	Adresse	Gehalt
1234	Müller	Hofstraße 7	23.000 €
1235	Meyer	Hauptstraße 13b	45.678 €
1236	Schmidt	Kurze Straße 1	55.555 €
1237	Heinze	Neue Straße 34	44.333 €
1238	Mustermann	Musterstraße 8	80.000 €
1239

Rechnung	
<u>RID</u>	Datum
1	12.03.2009
2	13.03.2009
3	13.03.2009
4	15.03.2009
5	15.03.2009
6	16.03.2009
7	16.03.2009
8	...

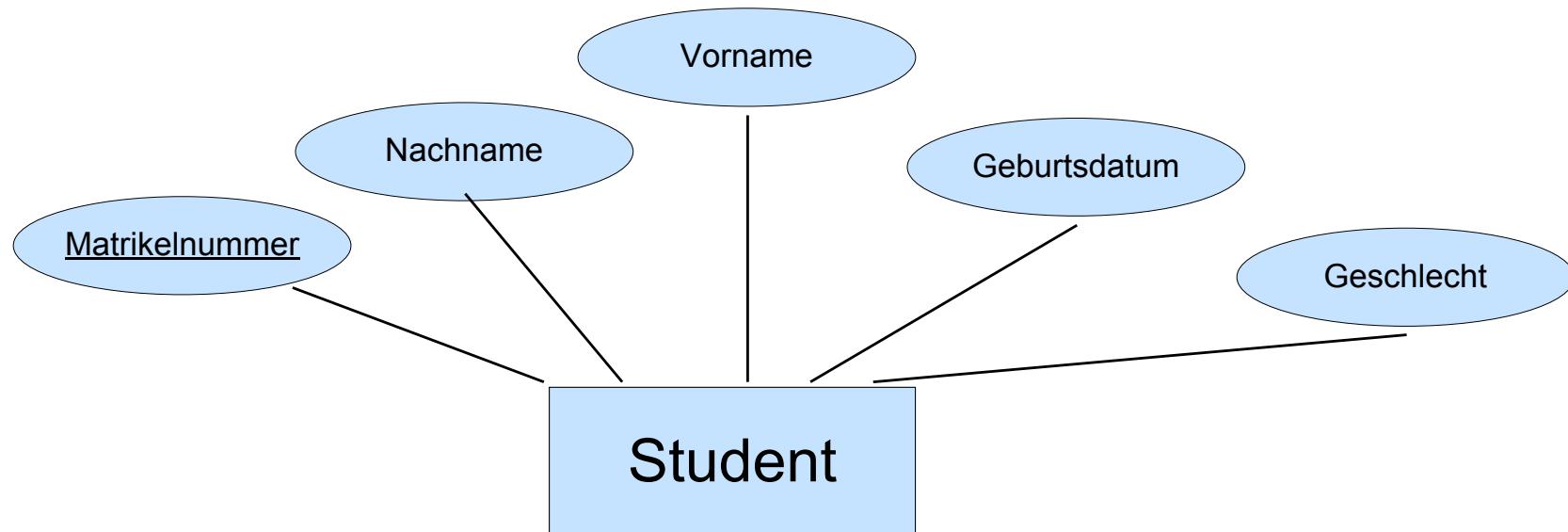
Abteilung	
<u>Abt.-ID</u>	Bezeichnung
1	Vertrieb
2	Produktion
3	Einkauf
4	Forschung
5	...

Artikel		
<u>AID</u>	Bezeichnung	Preis
1	Teller (blau)	11,95 €
2	Teller (weiß)	11,95 €
3	Tasse (blau)	3,95 €
4	Tasse (gelb)	3,95 €
5	Tasse (weiß)	3,95 €
6	Untertasse (rot)	2,95 €
7

Beispiel für eine Transformation

Relation STUDENT (Matrikelnummer, Name, Vorname, Geburtsdatum, Geschlecht)

<u>MATRIKEL-NUMMER</u>	NAME	VORNAME	GEBURTS-DATUM	GESCHLECHT
297645	Meier	Klaus	1980-05-23	M
297668	Müller	Claudia	1981-06-12	W
298902	Richter	Michael	1983-07-28	M



Herausforderung beim Transformieren

Problem

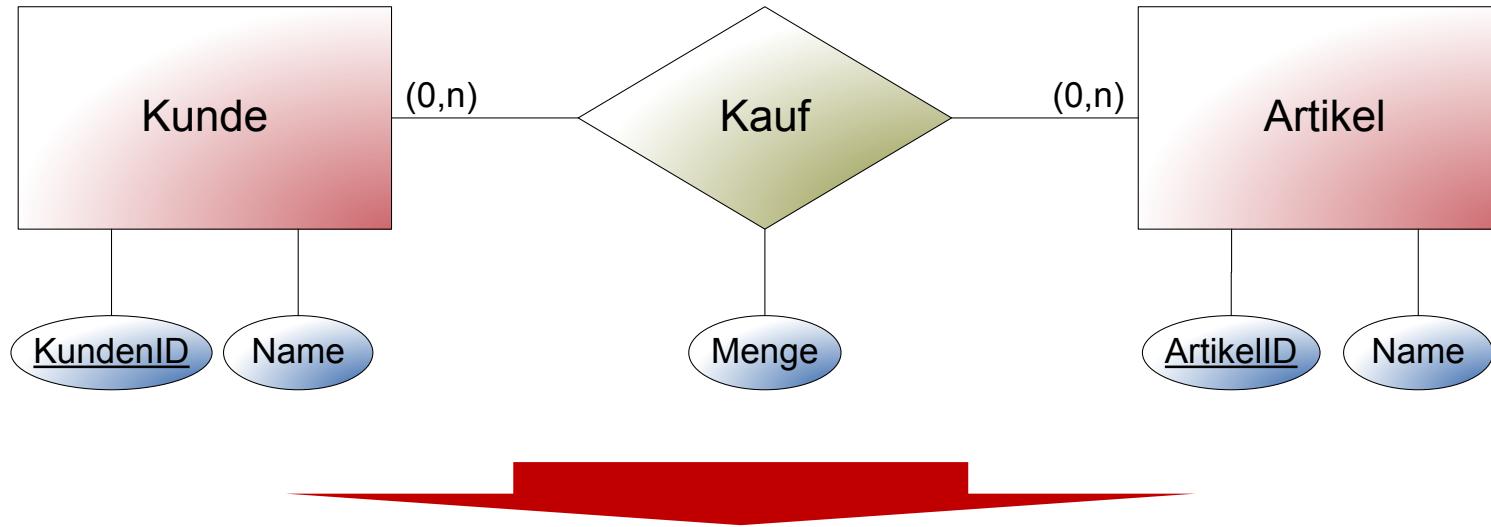
- Kein Bezug zwischen Tabellen
- Vier alleinstehende Tabellen
- Informationen, die in mehreren Tabellen stehen, können nicht zusammengefügt werden.

Ziel

- Datenbank mit zusammenhängenden Tabellen, die die Relationen im ERM korrekt abbilden.



Bespiel



Relation <i>Kunde</i>	
<u>KundenID</u>	Name
...	...
...	...

Relation <i>Kauf</i>		
<u>#KundenID</u>	<u>#ArtikelID</u>	Menge
...
...

Relation <i>Artikel</i>	
<u>ArtikelID</u>	Name
...	...
...	...

Verschiedene Beziehungstypen

Drei verschiedene Beziehungstypen
(jeweils nur der max-Wert)

1:N-Beziehung

- $(0,n) - (0,1)$
- $(1,n) - (1,1)$
- $(0,n) - (1,1)$
- $(1,n) - (0,1)$
- ...

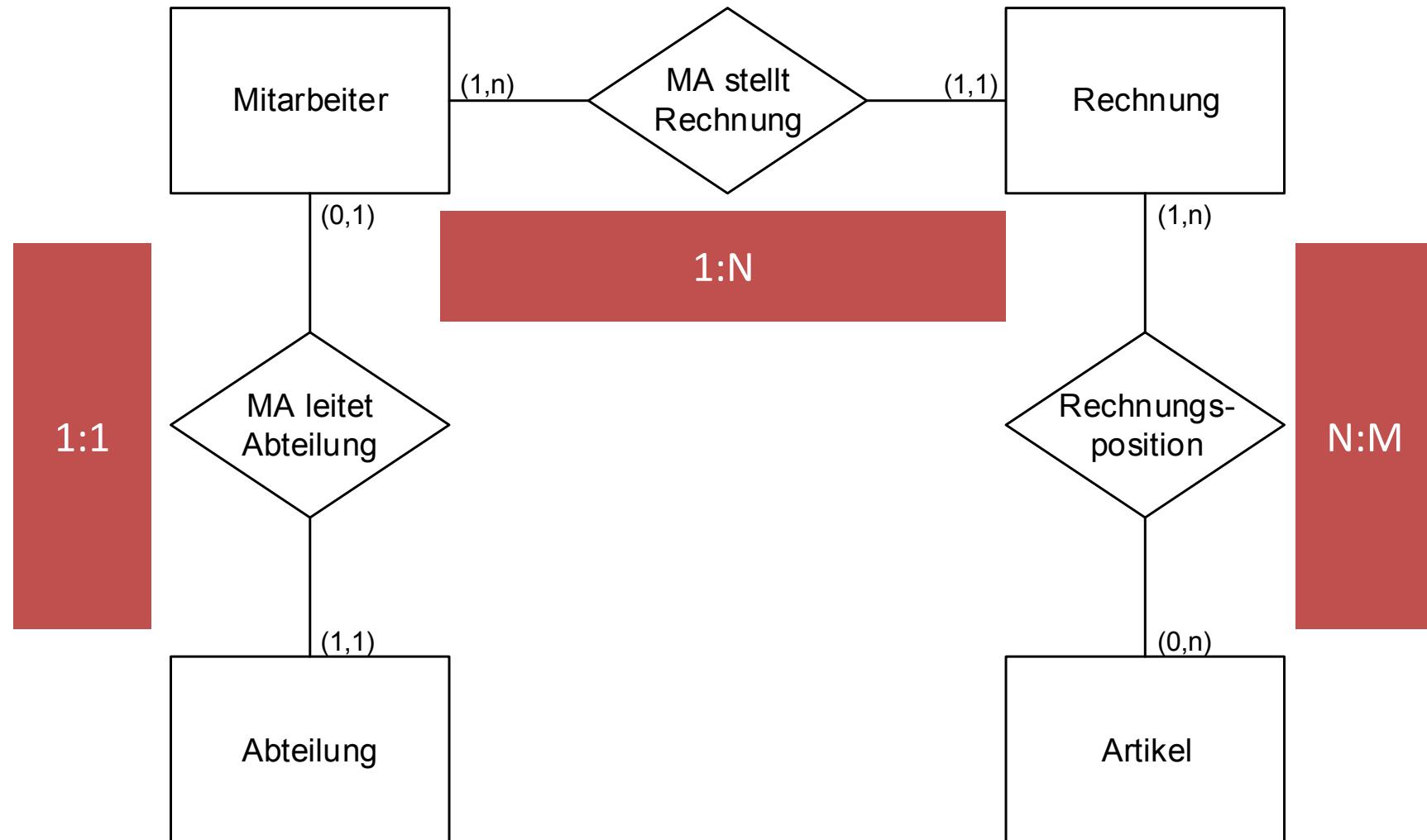
1:1-Beziehung

- $(0,1) - (0,1)$
- $(1,1) - (0,1)$
- ...

N:M-Beziehung

- $(0,m) - (1,n)$
- $(1,m) - (1,n)$
- ...

Maximale Kardinalitäten (vgl. Modellierung nach Chen)

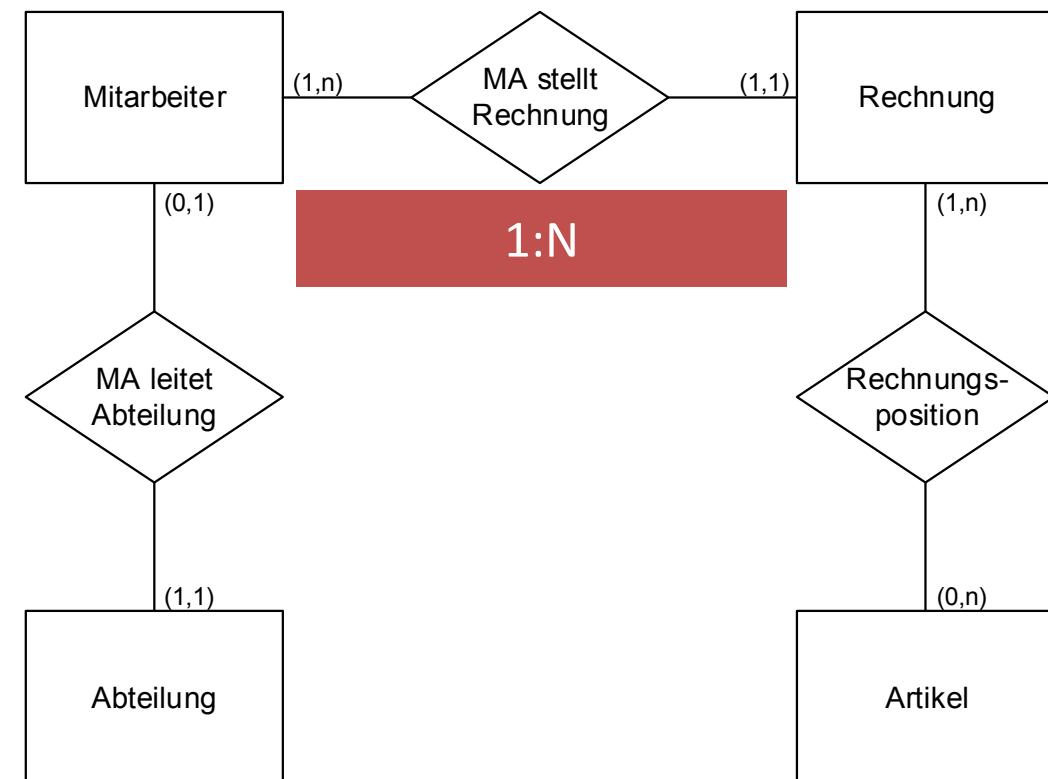


1:n Beziehung (1/3)

1:N-Beziehung

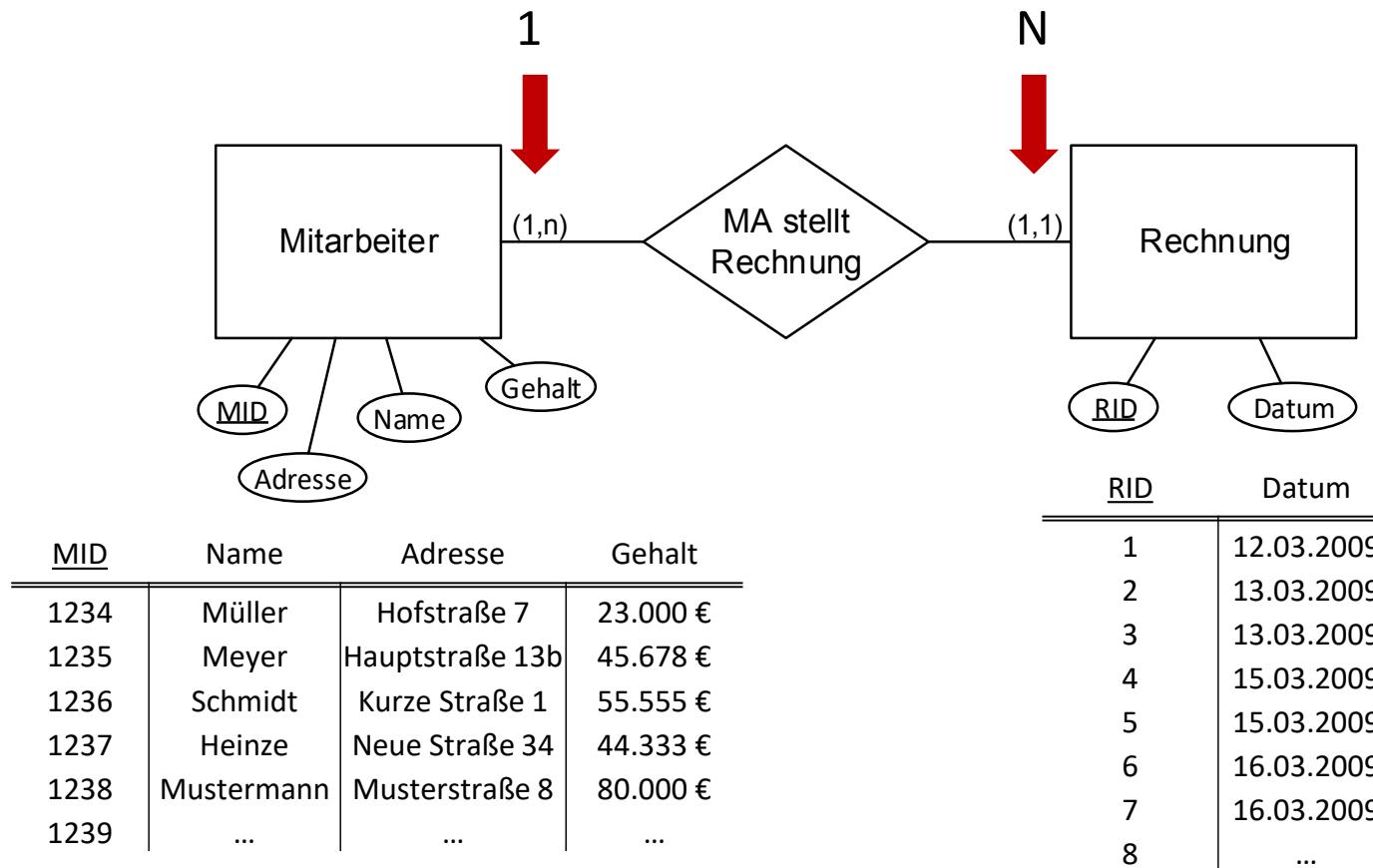
1:1-Beziehung

n:m-Beziehung



1:n Beziehung (2/3)

Relationen-Darstellung
Mitarbeiter : (MID, Name, Adresse, Gehalt)
Rechnung : (RID, Datum)



Transformation von 1:n-Beziehungen in das relationale Modell:

Ein weiteres Feld in der n-Tabelle anlegen (sogenannter Fremdschlüssel), das auf den entsprechenden Eintrag der 1-Tabelle verweist (also den Primärschlüssel der 1-Tabelle)

1:n Beziehung (3/3)

Relationen-Darstellung
Mitarbeiter : (MID, Name, Adresse, Gehalt)
Rechnung : (RID, Datum, #MID)

Im Beispiel

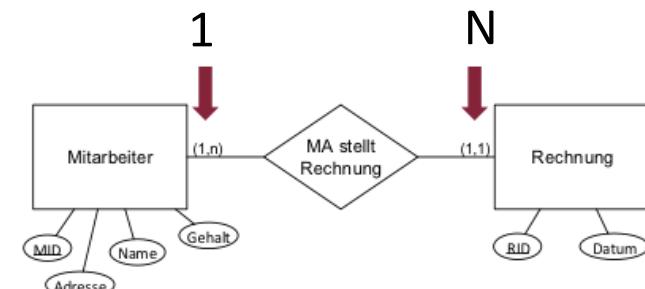
- Fremdschlüssel aus der 1-Tabelle (Mitarbeiter) in der n-Tabelle (Rechnung) anlegen
- MID zusätzlich als Fremdschlüssel in der n-Tabelle (Rechnung)

Mitarbeiter:

MID	Name	Adresse	Gehalt
1234	Müller	Hofstraße 7	23.000 €
1235	Meyer	Hauptstraße 13b	45.678 €
1236	Schmidt	Kurze Straße 1	55.555 €
1237	Heinze	Neue Straße 34	44.333 €
1238	Mustermann	Musterstraße 8	80.000 €
1239

Rechnung:

RID	Datum	#MID
1	12.03.2009	1234
2	13.03.2009	1234
3	13.03.2009	1237
4	15.03.2009	1238
5	15.03.2009	1235
6	16.03.2009	1237
7	16.03.2009	1237
8



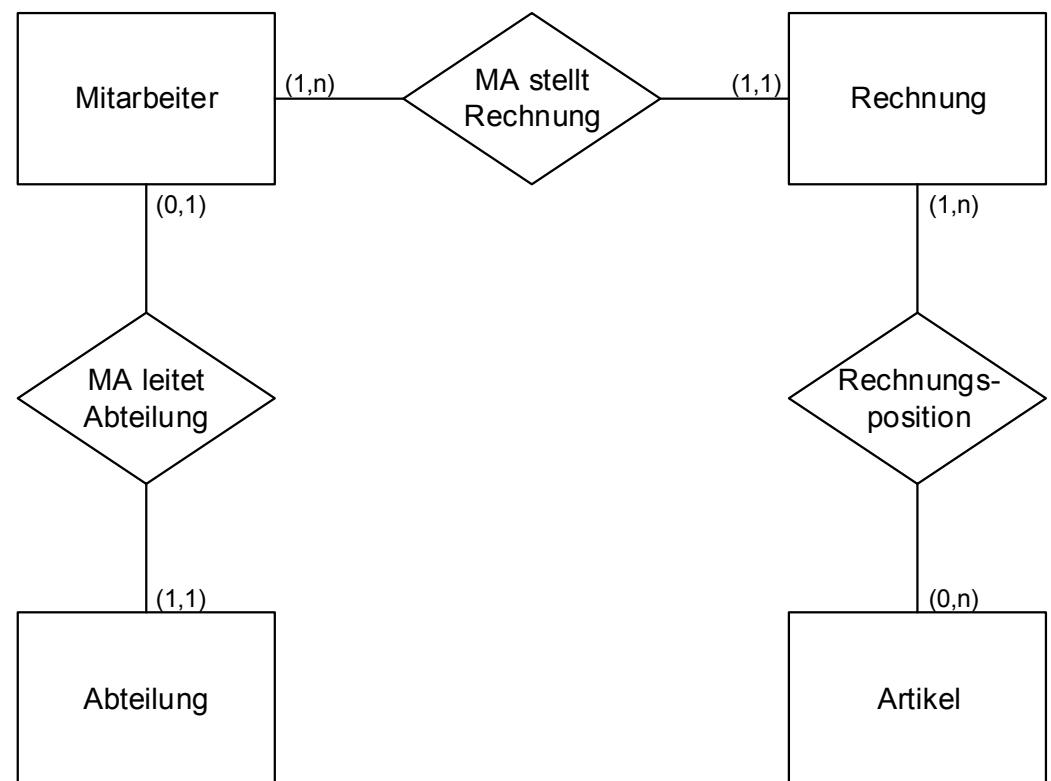
1:1-Beziehung (1/6)

1:N-Beziehung

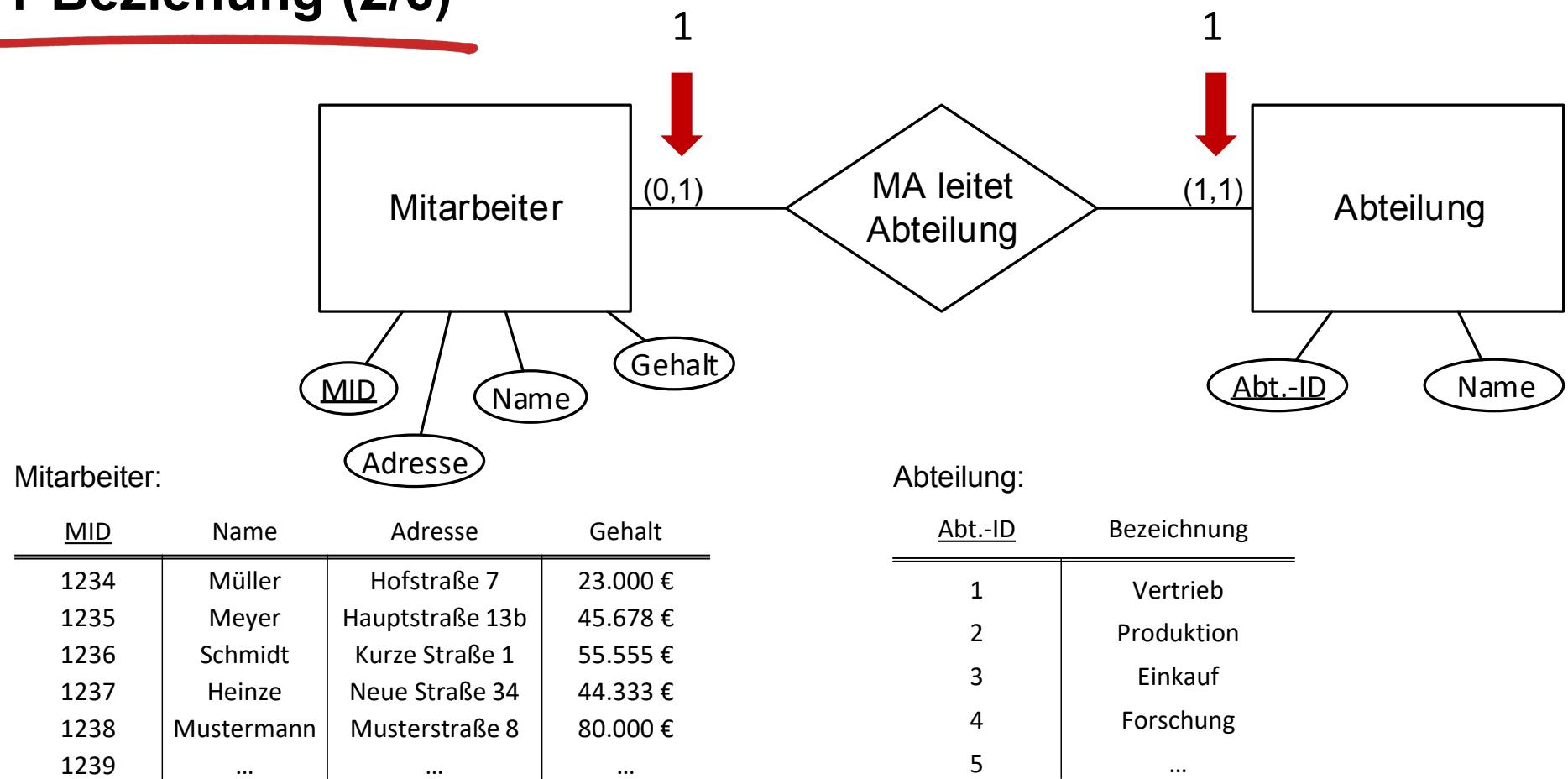
1:1-Beziehung

N:M-Beziehung

1:1



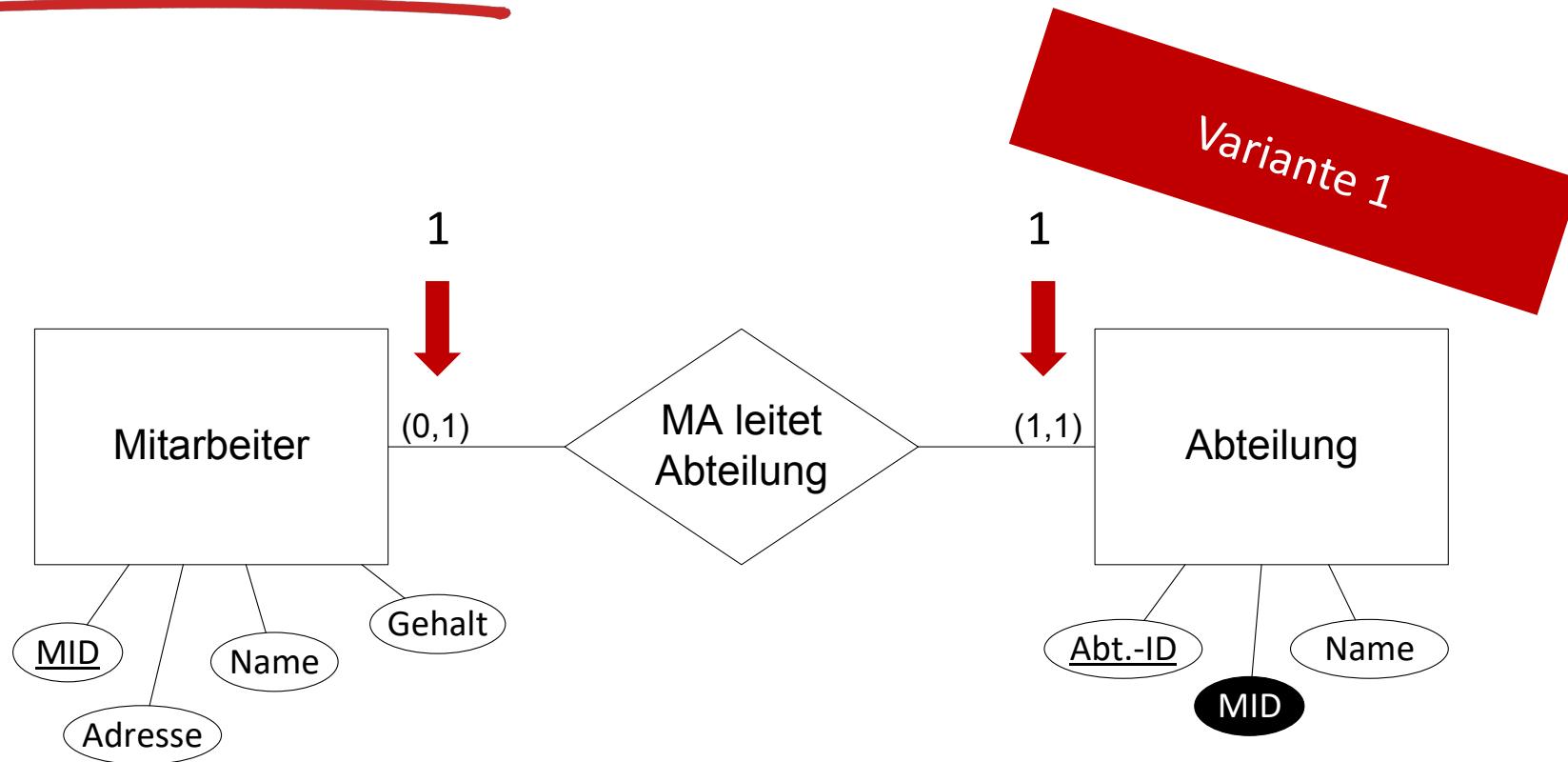
1:1-Beziehung (2/6)



Transformation von 1:1-Beziehungen in das relationale Modell:

Fremdschlüssel als weiteres Feld in einer 1-Tabelle, das auf den Eintrag der anderen 1-Tabelle verweist. Dies ist also ein Sonderfall einer 1:n-Beziehung. Alternativ können die Felder beider Entity-Typen auch in einer Tabelle zusammengefasst werden.

1:1-Beziehung (3/6)



- Primärschlüssel aus einer 1-Tabelle (Mitarbeiter) in der anderen 1-Tabelle als Fremdschlüssel (Abteilung)
 - MID und Abt.-ID als Primärschlüssel
 - MID zusätzlich als Fremdschlüssel

1:1-Beziehung (4/6)

Relationen-Darstellung
Mitarbeiter : (MID, Name, Adresse, Gehalt)
Rechnung : (RID, Datum, #MID)

Variante 1

Mitarbeiter:

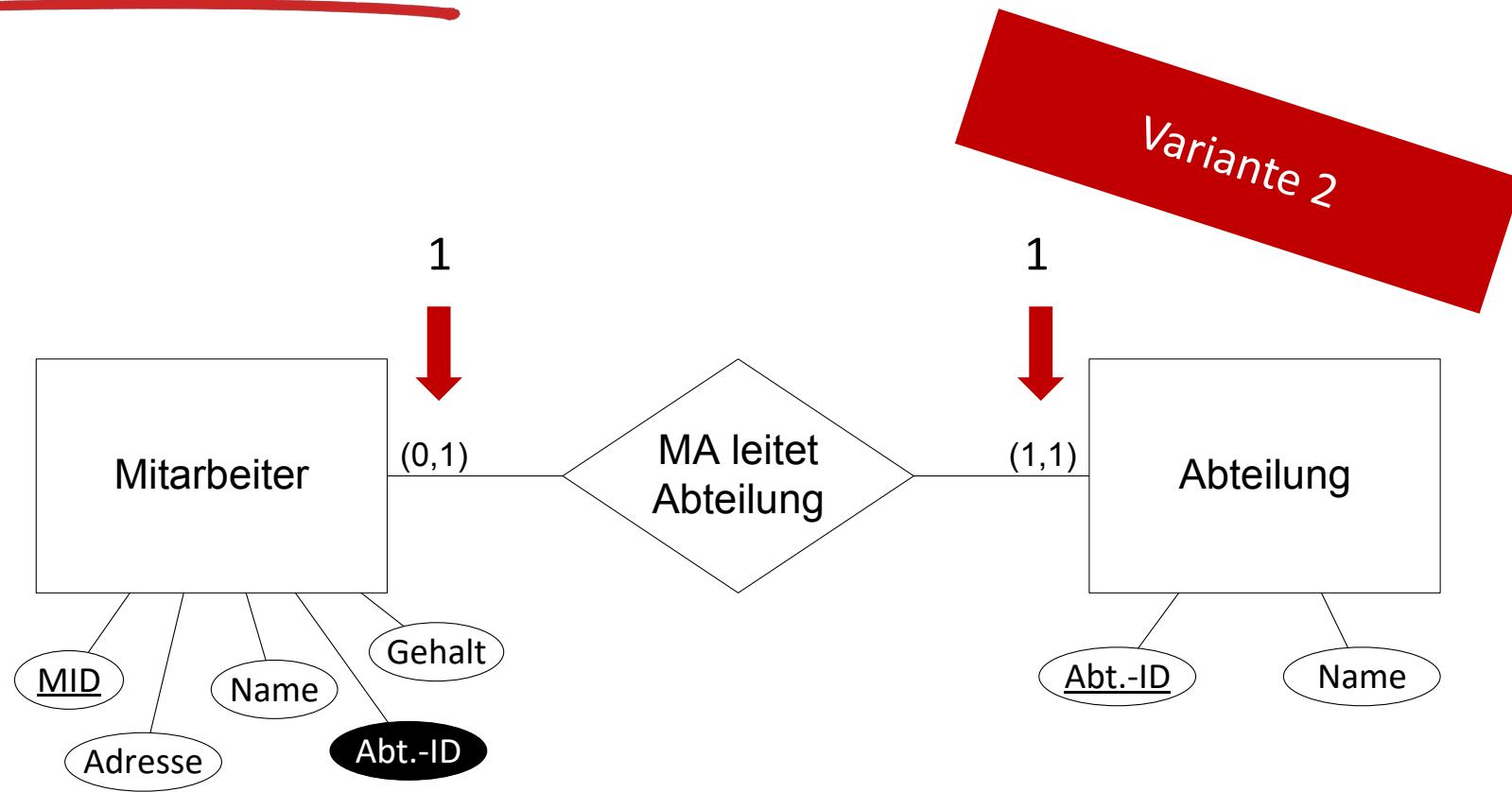
<u>MID</u>	Name	Adresse	Gehalt
1234	Müller	Hofstraße 7	23.000 €
1235	Meyer	Hauptstraße 13b	45.678 €
1236	Schmidt	Kurze Straße 1	55.555 €
1237	Heinze	Neue Straße 34	44.333 €
1238	Mustermann	Musterstraße 8	80.000 €
1239

Abteilung:

<u>Abt.-ID</u>	Bezeichnung	# <u>MID</u>
1	Vertrieb	1235
2	Produktion	1237
3	Einkauf	1234
4	Forschung	1238
5

- Primärschlüssel aus einer 1-Tabelle (Mitarbeiter) in der anderen 1-Tabelle als Fremdschlüssel (Abteilung)
 - MID und Abt.-ID als Primärschlüssel
 - MID zusätzlich als Fremdschlüssel

1:1-Beziehung (5/6)



- Primärschlüssel aus der anderen 1-Tabelle (**Abteilung**) in der 1-Tabelle als Fremdschlüssel (**Mitarbeiter**)

1:1-Beziehung (6/6)

Relationen-Darstellung

Mitarbeiter : (MID, Name, Adresse, Gehalt, #Abt.-ID)

Rechnung : (RID, Datum)

Variante 2

Mitarbeiter:

<u>MID</u>	Name	Adresse	Gehalt	#Abt.-ID
1234	Müller	Hofstraße 7	23.000 €	1
1235	Meyer	Hauptstraße 13b	45.678 €	
1236	Schmidt	Kurze Straße 1	55.555 €	4
1237	Heinze	Neue Straße 34	44.333 €	
1238	Mustermann	Musterstraße 8	80.000 €	2
1239

Abteilung:

<u>Abt.-ID</u>	Bezeichnung
1	Vertrieb
2	Produktion
3	Einkauf
4	Forschung
5	...

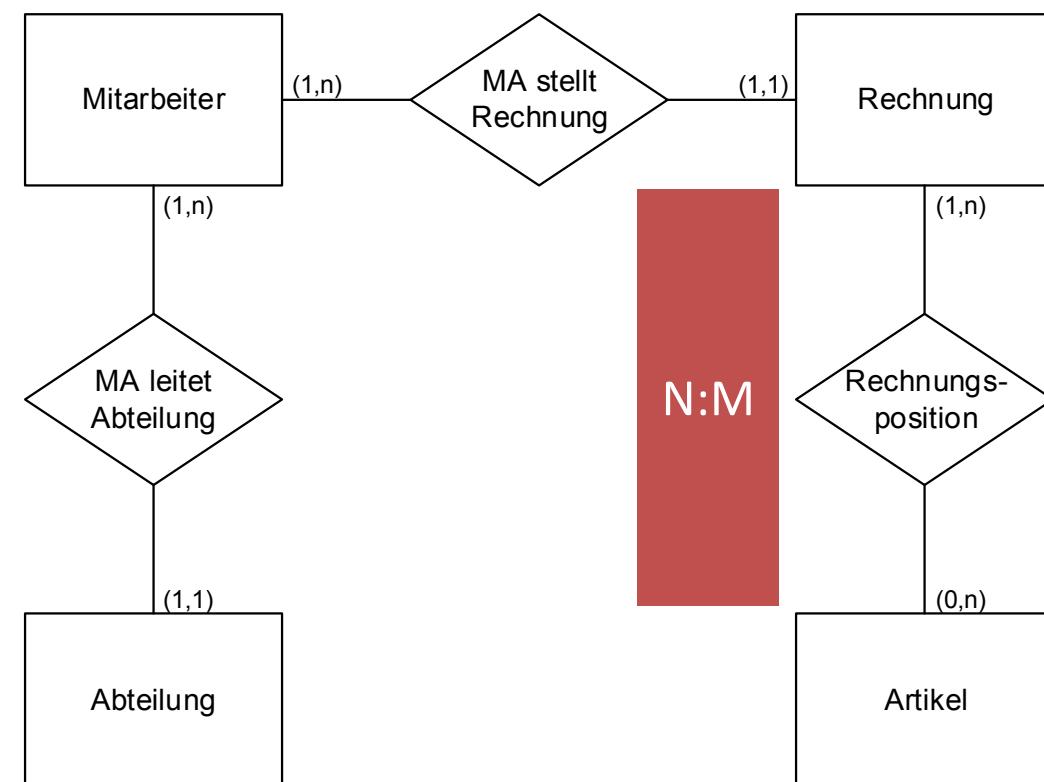
- Primärschlüssel aus der anderen 1-Tabelle (Abteilung) in der 1-Tabelle als Fremdschlüssel (Mitarbeiter)
 - MID und Abt.-ID als Primärschlüssel
- Abt.-ID zusätzlich als Fremdschlüssel in der Mitarbeiter-Tabelle

n:m-Beziehung (1/4)

1:n-Beziehung

1:1-Beziehung

N:M-Beziehung

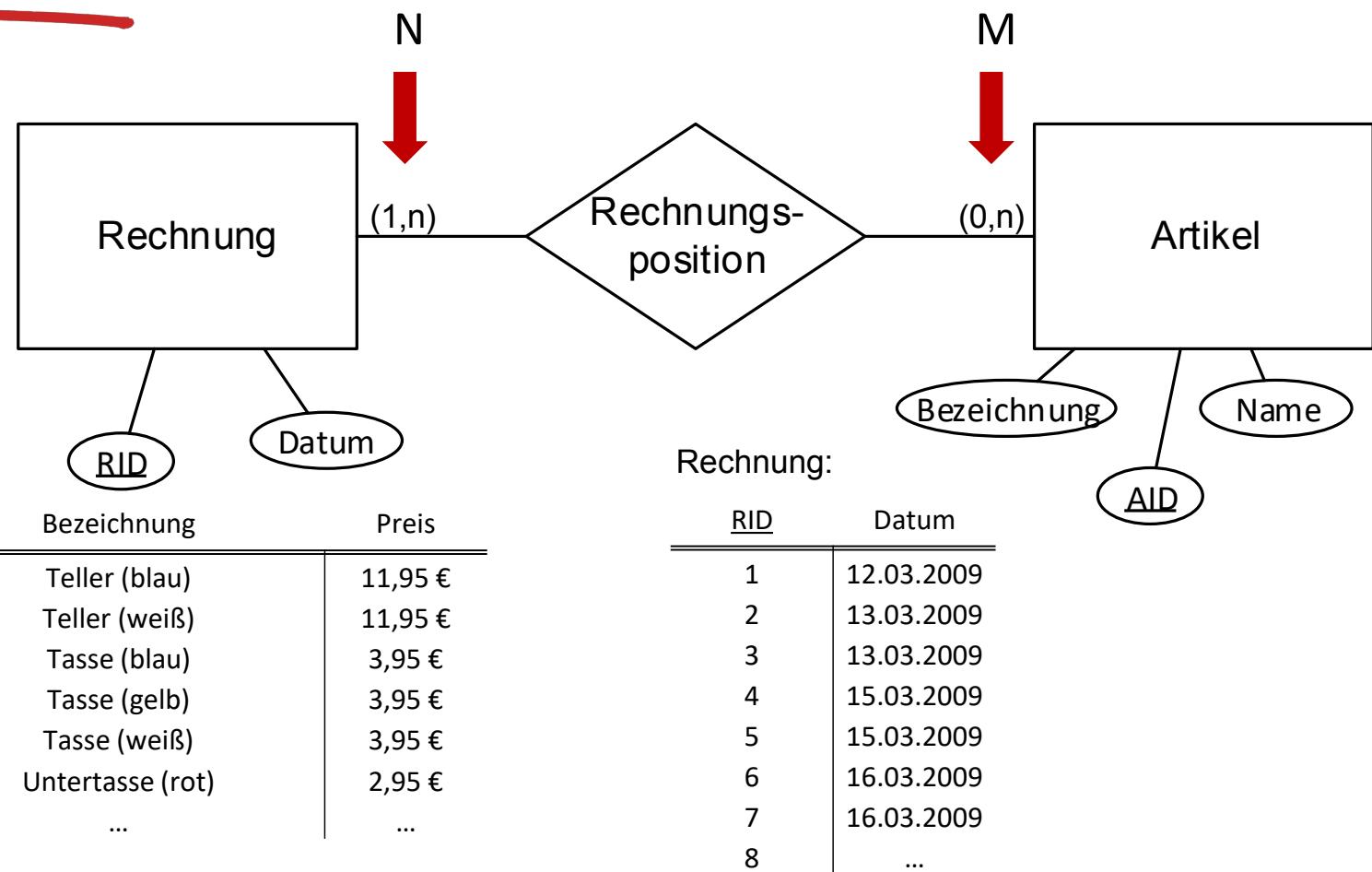


n:m-Beziehung (2/4)

Achtung: Wenn die ERM-Umformung wie in Kapitel 4.7.9 des Skripts beschrieben – vorgenommen wurde, dann sind diese Schritte nicht nötig (sondern die 1:n-Schritte). Am Ende sollte aber das gleiche relationale Schema herauskommen.

Artikel:

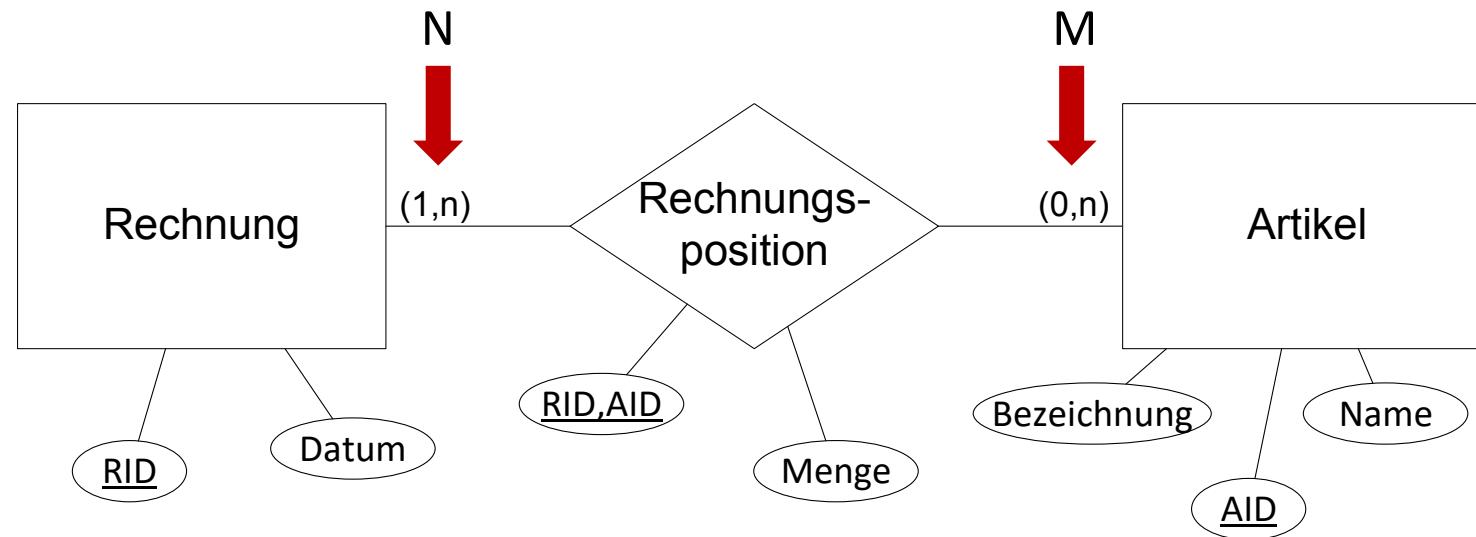
AID	Bezeichnung	Preis
1	Teller (blau)	11,95 €
2	Teller (weiß)	11,95 €
3	Tasse (blau)	3,95 €
4	Tasse (gelb)	3,95 €
5	Tasse (weiß)	3,95 €
6	Untertasse (rot)	2,95 €
7



Transformation von n:m-Beziehungen in das relationale Modell:

- Beziehung wird in zusätzlicher Rechnungs-Artikel-Zuordnungstabelle abgebildet
- Tabelle enthält für jeden Beziehungsfall eine Zeile
- Deren Primärschlüssel wird zusammengesetzt aus den Primärschlüsseln beider Tabellen

n:m-Beziehung (3/4)



- Die Primärschlüssel aus beiden Tabellen (Rechnung, Artikel) bilden den zusammengesetzten und eindeutigen Primärschlüssel für die Zuordnungstabelle (zusammengesetzt als Fremdschlüssel)
 - Attribute an der Relation sind möglich (bspw. Menge der Position)!

n:m-Beziehung (4/4)

Relationen-Darstellung

Artikel: (AID, Bezeichnung, Preis)

Rechnung : (RID, Datum)

Rechnungsposition: (#RID, #AID, Menge)

Rechnung:

RID	Datum
1	12.03.2009
2	13.03.2009
3	13.03.2009
4	15.03.2009
5	15.03.2009
6	16.03.2009
7	16.03.2009
8	...

Rechnungsposition:

#RID	#AID	Menge
1	8	1
1	7	3
1	1	3
2	1	5
2	9	7
2	14	12
3	3	10
4	8	5
...

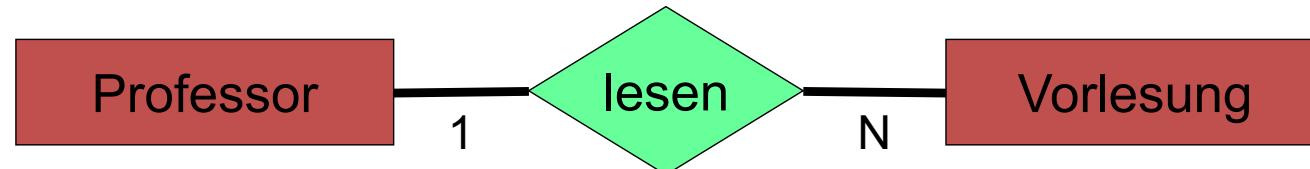
Artikel:

AID	Bezeichnung	Preis
1	Teller (blau)	11,95 €
2	Teller (weiß)	11,95 €
3	Tasse (blau)	3,95 €
4	Tasse (gelb)	3,95 €
5	Tasse (weiß)	3,95 €
6	Untertasse (rot)	2,95 €
7

- Zusätzliche Rechnung-Artikel-Zuordnungstabelle
- Zusammengesetzter Fremdschlüssel aus beiden Tabellen (RID, AID)
 - Evtl. weitere Attribute, z. B. Menge

Zusammenhang 1:N und N:M-Beziehung

- Beispiel



- Initial-Entwurf (entsprechend N:M-Beziehung)

- **Vorlesung** : (VorlNr, Titel, SWS)
- **Professor** : (PersNr, Name, Rang, Raum)
- **lesen**: (VorlNr, PersNr)

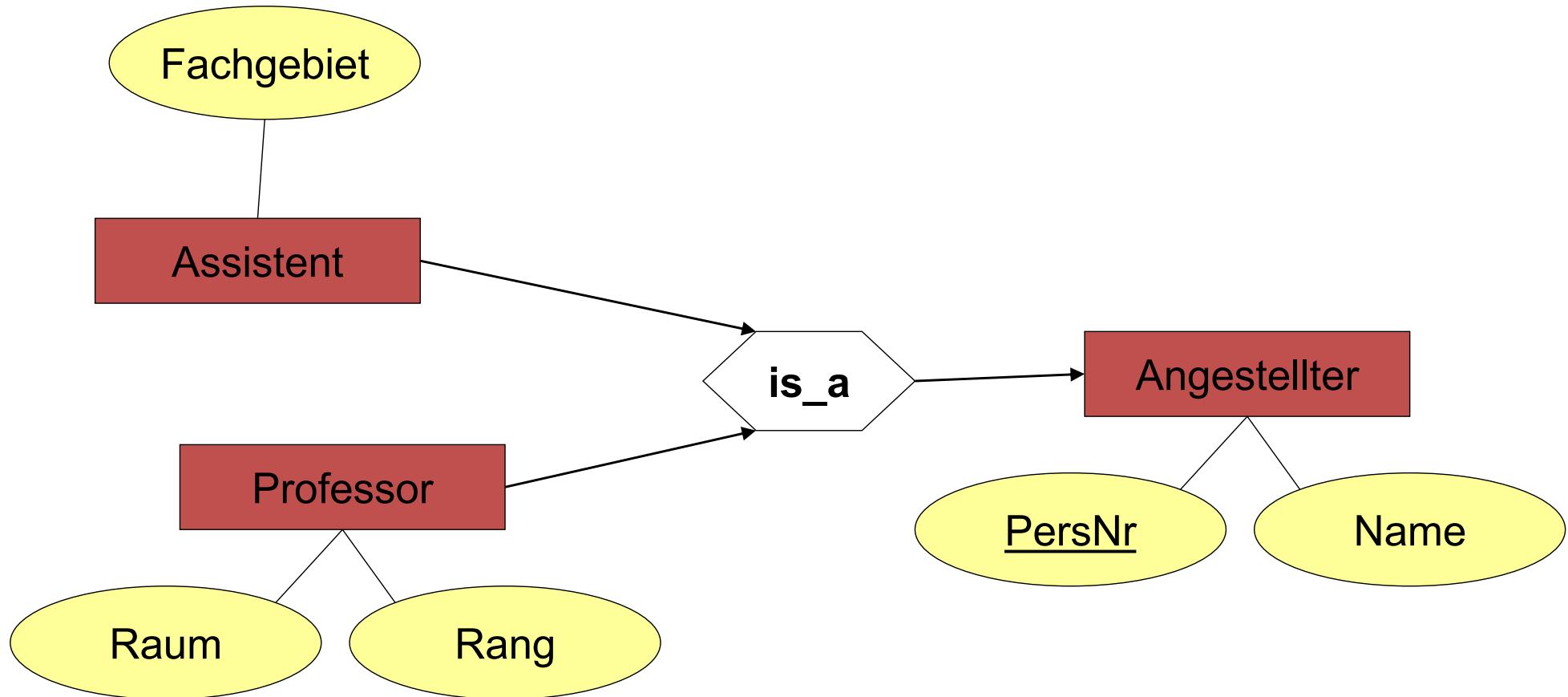
- Verfeinerung durch Zusammenfassung

- **Vorlesung** : (VorlNr, Titel, SWS, **gelesenVon**)
- **Professor** : (PersNr, Name, Rang, Raum)

Professoren			
PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Vorlesungen			
VorlNr	Titel	SWS	Gelesen Von
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

Relationale Modellierung der Generalisierung

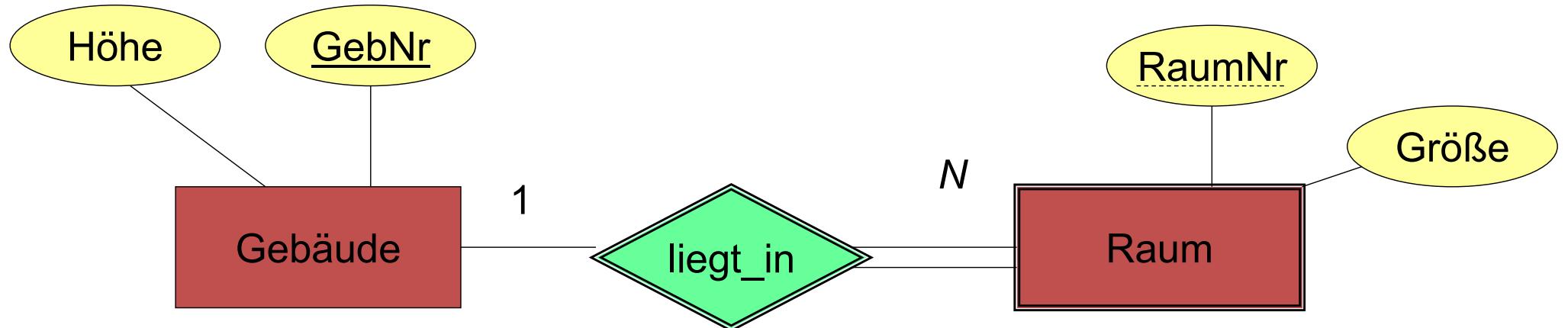


Angestellter: (PersNr, Name)

Professor: (#PersNr, Rang, Raum)

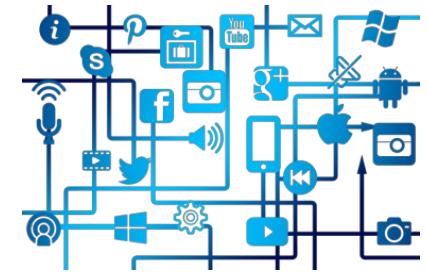
Assistent: (#PersNr, Fachgebiet)

Relationale Modellierung schwacher Entitytypen



- Relation `liegt_in`: der schwache Entity-Typ Räume wird dem Entity-Typ Gebäude untergeordnet.
- Wegen der 1 : N -Beziehung zwischen Gebäude und Räume kann die Beziehung `liegt_in` verlagert werden in die Relation Räume:
- Raum : (#GebNr, RaumNr, Grösse) }

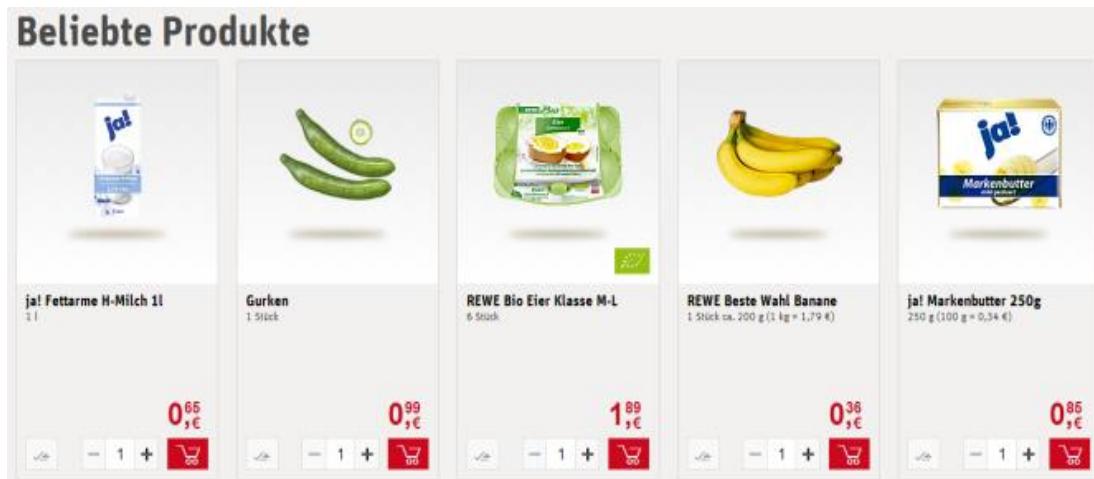
Weiterführendes Material



- Es empfiehlt sich die Lektüre von Kapitel 6.2 des Skripts Vornberger 2015

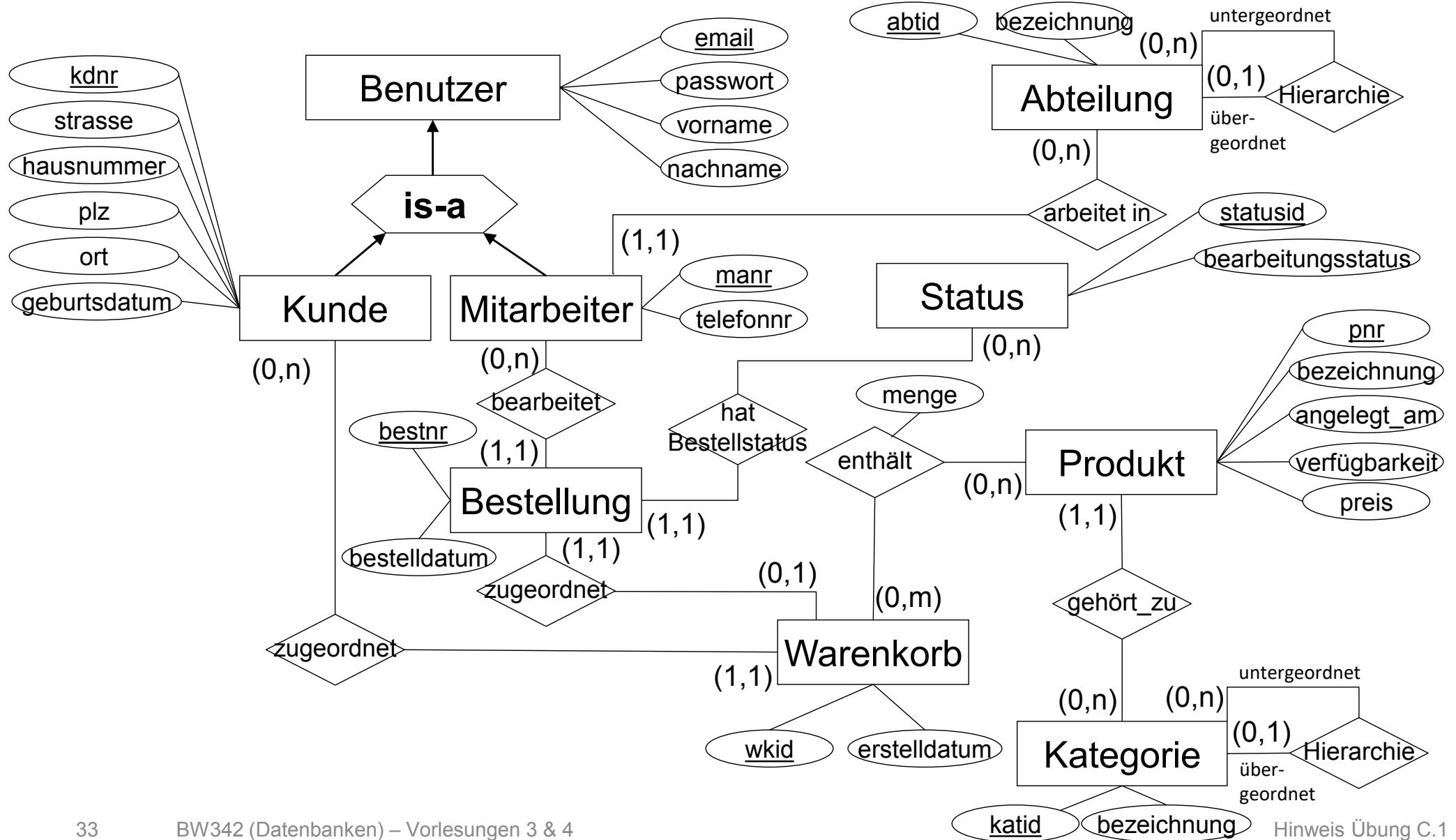
Übung

- Überführen Ihr konzeptionelle Modell (ERM) des Online-Shops in ein relationales Modell.



- Nutzen Sie das folgende ERM.

Bitte nutzen Sie dieses ERM



- Relationales Datenbankmodell**
- Normalisierung**
- Datenintegrität**
- Zusammenfassung und Ausblick**

Normalisierung: 1. Normalform

■ Definition 1. Normalform (1. NF):

Eine Relation befindet sich in erster Normalform, wenn jede Attributausprägung atomar ist.

- Damit darf ein Datensatz keine Attributausprägungen besitzen, die z. B. Listen oder andere Datenstrukturen (bspw. Wiederholungsgruppen) enthalten und jedes Attribut muss einen atomaren Wertebereich haben.
- Um eine Tabelle in die erste Normalform zu überführen, sind folglich sämtliche **nicht-atomaren Attributausprägungen aufzulösen**.

Beispielszenario

Achtung: der logische Fehler, dass dem selben Ski im selben Genre zwei Eignungen zugeordnet wurden ignorieren wir hier! Wir nehmen also an, dies wäre so gewünscht.

- Ein Sportgeschäft führt eine (**nicht normalisierte**) Tabelle über seine angebotenen Wintersportgeräte.
- Es wird jeweils angegeben, von welchem Hersteller der Ski ist und für welche Geländeformen er wie gut geeignet ist.

HNR	Hersteller	SkiNr	SkiName	GenreID	Genre	EID	Eignung
F0012	Fischer	FRC4R	RC4 Race	001	Kurzschwung	2	gut
				002	Carving	2	gut
				002	Carving	3	mittel
				003	Freeride	1	sehr gut
	Fischer	FRC4W	RC4 Worldcup	001	Kurzschwung	1	sehr gut
				002	Carving	1	sehr gut
				003	Freeride	2	gut
				003	Freeride	1	sehr gut
S0001	Salomon	SR	Rocker	003	Freeride	1	sehr gut
				004	Buckelpiste	1	sehr gut
				004	Buckelpiste	2	gut
	Salomon	ST	Threat	005	Freestyle	1	sehr gut
				003	Freeride	3	mittel
S0002	Stöckli	SS	Sinox	001	Kurzschwung	5	mangelhaft
				001	Kurzschwung	2	gut
				002	Carving	2	gut

1. Normalform – Vorgehen

- Falls die Tabelle keine nicht-atomaren Attributausprägungen enthält, ist sie bereits in erster Normalform.
- Alle übrigen Tabellen sind wie folgt zu untersuchen:
 1. Nicht-atomare Attributausprägungen identifizieren.
 2. Nicht-atomare Datensätze auflösen (bspw. durch „Auffüllen der Zeilen“).
 3. Primärschlüssel identifizieren/festlegen, so dass jeder Datensatz eindeutig identifiziert werden kann.

1. Normalform – Vorgehen

1. Nicht-atomare Attributausprägungen identifizieren

HNR	Hersteller	SkiNr	SkiName	GenreID	Genre	EID	Eignung
F0012	Fischer	FRC4R	RC4 Race	001	Kurzschwung	2	gut
				002	Carving	2	gut
				002	Carving	3	mittel
				003	Freeride	1	sehr gut
	FRC4W	RC4 Worldcup	RC4 Worldcup	001	Kurzschwung	1	sehr gut
				002	Carving	1	sehr gut
				003	Freeride	2	gut
				003	Freeride	1	sehr gut
S0001	Salomon	SR	Rocker	003	Freeride	1	sehr gut
				004	Buckelpiste	1	sehr gut
				004	Buckelpiste	2	gut
		ST	Threat	005	Freestyle	1	sehr gut
				003	Freeride	3	mittel
S0002	Stöckli	SS	Sinox	001	Kurzschwung	5	mangelhaft
				001	Kurzschwung	2	gut
				002	Carving	2	gut

1. Normalform – Vorgehen

2. Nicht-atomare Datensätze auflösen durch „Auffüllen der Zeilen“

HNR	Hersteller	SkiNr	SkiName	GenreID	Genre	EID	Eignung
F0012	Fischer	FRC4R	RC4 Race	001	Kurzschwung	2	gut
				002	Carving	2	gut
				002	Carving	3	mittel
				003	Freeride	1	sehr gut
F0012	Fischer	FRC4W	RC4 Worldcup	001	Kurzschwung	1	sehr gut
				002	Carving	1	sehr gut
				003	Freeride	2	gut
				003	Freeride	1	sehr gut
S0001	Salomon	SR	Rocker	003	Freeride	1	sehr gut
				004	Buckelpiste	1	sehr gut
				004	Buckelpiste	2	gut
S0001	Salomon	ST	Threat	005	Freestyle	1	sehr gut
				003	Freeride	3	mittel
				001	Kurzschwung	5	mangelhaft
S0002	Stöckli	SS	Sinox	001	Kurzschwung	2	gut
				002	Carving	2	gut

1. Normalform – Vorgehen

2. Nicht-atomare Datensätze auflösen durch „Auffüllen der Zeilen“

HNR	Hersteller	SkiNr	SkiName	GenreID	Genre	EID	Eignung
F0012	Fischer	FRC4R	RC4 Race	001	Kurzschwung	2	gut
F0012	Fischer	FRC4R	RC4 Race	002	Carving	2	gut
F0012	Fischer	FRC4R	RC4 Race	002	Carving	3	mittel
F0012	Fischer	FRC4R	RC4 Race	003	Freeride	1	sehr gut
F0012	Fischer	FRC4W	RC4 Worldcup	001	Kurzschwung	1	sehr gut
F0012	Fischer	FRC4W	RC4 Worldcup	002	Carving	1	sehr gut
F0012	Fischer	FRC4W	RC4 Worldcup	003	Freeride	2	gut
F0012	Fischer	FRC4W	RC4 Worldcup	003	Freeride	1	sehr gut
S0001	Salomon	SR	Rocker	003	Freeride	1	sehr gut
S0001	Salomon	SR	Rocker	004	Buckelpiste	1	sehr gut
S0001	Salomon	SR	Rocker	004	Buckelpiste	2	gut
S0001	Salomon	ST	Threat	005	Freestyle	1	sehr gut
S0001	Salomon	ST	Threat	003	Freeride	3	mittel
S0001	Salomon	ST	Threat	001	Kurzschwung	5	mangelhaft
S0002	Stöckli	SS	Sinox	001	Kurzschwung	2	gut
S0002	Stöckli	SS	Sinox	002	Carving	2	gut

1. Normalform – Vorgehen

Dabei gilt: Minimale Anzahl von Schlüsselattributen ist das Ziel!

Dort, wo IDs (also SkiNr, GenreID, EID usw.) vorhanden sind, sollten diese als Schlüssel verwendet werden und nicht die jeweilige Bezeichnung).

3. Primärschlüssel identifizieren (mit einem Primärschlüssel kann jeder Datensatz einer Relation eindeutig identifiziert werden)

HNR	Hersteller	SkiNr	SkiName	GenreID	Genre	EID	Eignung
F0012	Fischer	FRC4R	RC4 Race	001	Kurzschwung	2	gut
F0012	Fischer	FRC4R	RC4 Race	002	Carving	2	gut
F0012	Fischer	FRC4R	RC4 Race	002	Carving	3	mittel
F0012	Fischer	FRC4R	RC4 Race	003	Freeride	1	sehr gut
F0012	Fischer	FRC4W	RC4 Worldcup	001	Kurzschwung	1	sehr gut
F0012	Fischer	FRC4W	RC4 Worldcup	002	Carving	1	sehr gut
F0012	Fischer	FRC4W	RC4 Worldcup	003	Freeride	2	gut
F0012	Fischer	FRC4W	RC4 Worldcup	003	Freeride	1	sehr gut
S0001	Salomon	SR	Rocker	003	Freeride	1	sehr gut
S0001	Salomon	SR	Rocker	004	Buckelpiste	1	sehr gut
S0001	Salomon	SR	Rocker	004	Buckelpiste	2	gut
S0001	Salomon	ST	Threat	005	Freestyle	1	sehr gut
S0001	Salomon	ST	Threat	003	Freeride	3	mittel
S0001	Salomon	ST	Threat	001	Kurzschwung	5	mangelhaft
S0002	Stöckli	SS	Sinox	001	Kurzschwung	2	gut
S0002	Stöckli	SS	Sinox	002	Carving	2	gut

1. Normalform – Vorgehen

- Die Relation befindet sich jetzt in erster Normalform (1. NF)

HNR	Hersteller	<u>SkiNr</u>	SkiName	<u>GenreID</u>	Genre	<u>EID</u>	Eignung
F0012	Fischer	FRC4R	RC4 Race	001	Kurzschwung	2	gut
F0012	Fischer	FRC4R	RC4 Race	002	Carving	2	gut
F0012	Fischer	FRC4R	RC4 Race	002	Carving	3	mittel
F0012	Fischer	FRC4R	RC4 Race	003	Freeride	1	sehr gut
F0012	Fischer	FRC4W	RC4 Worldcup	001	Kurzschwung	1	sehr gut
F0012	Fischer	FRC4W	RC4 Worldcup	002	Carving	1	sehr gut
F0012	Fischer	FRC4W	RC4 Worldcup	003	Freeride	2	gut
F0012	Fischer	FRC4W	RC4 Worldcup	003	Freeride	1	sehr gut
S0001	Salomon	SR	Rocker	003	Freeride	1	sehr gut
S0001	Salomon	SR	Rocker	004	Buckelpiste	1	sehr gut
S0001	Salomon	SR	Rocker	004	Buckelpiste	2	gut
S0001	Salomon	ST	Threat	005	Freestyle	1	sehr gut
S0001	Salomon	ST	Threat	003	Freeride	3	mittel
S0001	Salomon	ST	Threat	001	Kurzschwung	5	mangelhaft
S0002	Stöckli	SS	Sinox	001	Kurzschwung	2	gut
S0002	Stöckli	SS	Sinox	002	Carving	2	gut

2. Normalform – Definition

- **Definition 2. Normalform (2. NF):**
Eine Relation befindet sich in zweiter Normalform,
 - wenn sie sich in erster Normalform befindet
 - und alle Nichtschlüsselattribute vollfunktional vom Primärschlüssel abhängig sind.
- Es dürfen keine Nichtschlüsselattribute existieren, die nur von einem **Teil** des Primärschlüssels abhängen.
- Um eine Relation von der ersten in die zweite Normalform zu überführen, sind solche Attributgruppen in eigene Relationen auszulagern.

2. Normalform – Vorgehen

- Falls der Primärschlüssel einer Relation aus nur einem Attribut besteht, ist sie bereits in zweiter Normalform.
- Falls eine Relation keine Nichtschlüsselattribute enthält, ist sie bereits in zweiter Normalform.
- Alle übrigen Relationen sind wie folgt zu untersuchen:
 1. Alle Gruppen von Nichtschlüsselattributen identifizieren, die nur von einem Teil des Schlüssels funktional abhängig sind und diese jeweils mit diesem Schlüsselteil in eine eigene Tabelle überführen.
 2. Der ausgegliederte Schlüsselteil wird der Schlüssel der neuen Tabelle.
 3. Die ausgegliederten Nichtschlüsselattribute werden aus der Ursprungstabelle entfernt.

2. Normalform – Vorgehen

- Der Primärschlüssel der Relation besteht nicht nur aus einem Attribut; Nichtschlüsselattribute existieren ebenfalls.
- Die Erfüllung der 2. NF ist zu prüfen

HNR	Hersteller	SkiNr	SkiName	GenreID	Genre	EID	Eignung
F0012	Fischer	FRC4R	RC4 Race	001	Kurzschwung	2	gut
F0012	Fischer	FRC4R	RC4 Race	002	Carving	2	gut
F0012	Fischer	FRC4R	RC4 Race	002	Carving	3	mittel
F0012	Fischer	FRC4R	RC4 Race	003	Freeride	1	sehr gut
F0012	Fischer	FRC4W	RC4 Worldcup	001	Kurzschwung	1	sehr gut
F0012	Fischer	FRC4W	RC4 Worldcup	002	Carving	1	sehr gut
F0012	Fischer	FRC4W	RC4 Worldcup	003	Freeride	2	gut
F0012	Fischer	FRC4W	RC4 Worldcup	003	Freeride	1	sehr gut
S0001	Salomon	SR	Rocker	003	Freeride	1	sehr gut
S0001	Salomon	SR	Rocker	004	Buckelpiste	1	sehr gut
S0001	Salomon	SR	Rocker	004	Buckelpiste	2	gut
S0001	Salomon	ST	Threat	005	Freestyle	1	sehr gut
S0001	Salomon	ST	Threat	003	Freeride	3	mittel
S0001	Salomon	ST	Threat	001	Kurzschwung	5	mangelhaft
S0002	Stöckli	SS	Sinox	001	Kurzschwung	2	gut
S0002	Stöckli	SS	Sinox	002	Carving	2	gut

2. Normalform – Vorgehen

- Welche Nichtschlüsselattribute hängen nur von einem Teil des Schlüssels ab?

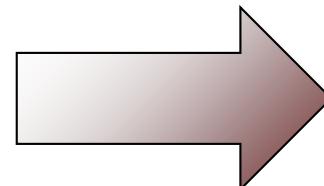
HNR	Hersteller	<u>SkiNr</u>	SkiName	<u>GenreID</u>	Genre	<u>EID</u>	Eignung
F0012	Fischer	FRC4R	RC4 Race	001	Kurzschwung	2	gut
F0012	Fischer	FRC4R	RC4 Race	002	Carving	2	gut
F0012	Fischer	FRC4R	RC4 Race	002	Carving	3	mittel
F0012	Fischer	FRC4R	RC4 Race	003	Freeride	1	sehr gut
F0012	Fischer	FRC4W	RC4 Worldcup	001	Kurzschwung	1	sehr gut
F0012	Fischer	FRC4W	RC4 Worldcup	002	Carving	1	sehr gut
F0012	Fischer	FRC4W	RC4 Worldcup	003	Freeride	2	gut
F0012	Fischer	FRC4W	RC4 Worldcup	003	Freeride	1	sehr gut
S0001	Salomon	SR	Rocker	003	Freeride	1	sehr gut
S0001	Salomon	SR	Rocker	004	Buckelpiste	1	sehr gut
S0001	Salomon	SR	Rocker	004	Buckelpiste	2	gut
S0001	Salomon	ST	Threat	005	Freestyle	1	sehr gut
S0001	Salomon	ST	Threat	003	Freeride	3	mittel
S0001	Salomon	ST	Threat	001	Kurzschwung	5	mangelhaft
S0002	Stöckli	SS	Sinox	001	Kurzschwung	2	gut
S0002	Stöckli	SS	Sinox	002	Carving	2	gut

2. Normalform – Vorgehen

■ Ausgliederung der Relation „Eignung“

- Der Schlüssel wird aus der Ursprungsrelation übernommen
- Doppelte Einträge werden aus der neuen Relation entfernt

EID	Eignung
2	gut
2	gut
3	mittel
1	sehr gut
1	sehr gut
1	sehr gut
2	gut
1	sehr gut
1	sehr gut
1	sehr gut
2	gut
1	sehr gut
3	mittel
5	mangelhaft
2	gut
2	gut



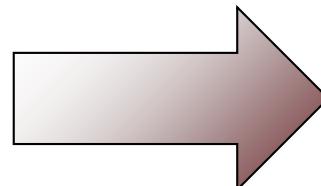
EID	Eignung
1	sehr gut
2	gut
3	mittel
5	mangelhaft

2. Normalform – Vorgehen

■ Ausgliederung der Relation „Genre“

- Der Schlüssel wird aus der Ursprungsrelation übernommen
- Doppelte Einträge werden aus der neuen Relation entfernt

GenreID	Genre
001	Kurzschwung
002	Carving
002	Carving
003	Freeride
001	Kurzschwung
002	Carving
003	Freeride
003	Freeride
003	Freeride
004	Buckelpiste
004	Buckelpiste
005	Freestyle
003	Freeride
001	Kurzschwung
001	Kurzschwung
002	Carving



GenreID	Genre
001	Kurzschwung
002	Carving
003	Freeride
004	Buckelpiste
005	Freestyle

2. Normalform – Vorgehen

■ Ausgliederung der Relation „Ski“

- Der Schlüssel wird aus der Ursprungsrelation übernommen
- Doppelte Einträge werden aus der neuen Relation entfernt

HNR	Hersteller	SkiNr	SkiName
F0012	Fischer	FRC4R	RC4 Race
F0012	Fischer	FRC4R	RC4 Race
F0012	Fischer	FRC4R	RC4 Race
F0012	Fischer	FRC4R	RC4 Race
F0012	Fischer	FRC4W	RC4 Worldcup
F0012	Fischer	FRC4W	RC4 Worldcup
F0012	Fischer	FRC4W	RC4 Worldcup
F0012	Fischer	FRC4W	RC4 Worldcup
S0001	Salomon	SR	Rocker
S0001	Salomon	SR	Rocker
S0001	Salomon	SR	Rocker
S0001	Salomon	ST	Threat
S0001	Salomon	ST	Threat
S0001	Salomon	ST	Threat
S0002	Stöckli	SS	Sinox
S0002	Stöckli	SS	Sinox



SkiNr	SkiName	HNR	Hersteller
FRC4R	RC4 Race	F0012	Fischer
FRC4W	RC4 Worldcup	F0012	Fischer
SR	Rocker	S0001	Salomon
ST	Threat	S0001	Salomon
SS	Sinox	S0002	Stöckli

2. Normalform – Vorgehen

- Relationenschema nach Herstellen der 2. Normalform

Relation „Skieignung für Genre“

<u>SkiNr</u>	<u>GenreID</u>	<u>EID</u>
FRC4R	001	2
FRC4R	002	2
FRC4R	002	3
FRC4R	003	1
FRC4W	001	1
FRC4W	002	1
FRC4W	003	2
FRC4W	003	1
SR	003	1
SR	004	1
SR	004	2
ST	005	1
ST	003	3
ST	001	5
SS	001	2
SS	002	2

Relation „Ski“

<u>SkiNr</u>	SkiName	HNR	Hersteller
FRC4R	RC4 Race	F0012	Fischer
FRC4W	RC4 Worldcup	F0012	Fischer
SR	Rocker	S0001	Salomon
ST	Threat	S0001	Salomon
SS	Sinox	S0002	Stöckli

Relation „Genre“

<u>GenreID</u>	Genre
001	Kurzschwung
002	Carving
003	Freeride
004	Buckelpiste
005	Freestyle

Relation
„Eignung“

<u>EID</u>	Eignung
1	sehr gut
2	gut
3	mittel
5	mangelhaft

3. Normalform – Definition

- **Definition 3. Normalform (3. NF):**
Eine Relation befindet sich in dritter Normalform,
 - wenn sie sich in zweiter Normalform befindet
 - und kein Nichtschlüsselattribut transitiv vom Primärschlüssel abhängig ist.
- Es dürfen keine Nichtschlüsselattribute existieren, die nur indirekt über andere Nichtschlüsselattribute vom Primärschlüssel abhängen.
- Um eine Relation von der zweiten in die dritte Normalform zu überführen, sind solche Attributgruppen in eigene Relationen auszulagern.

3. Normalform – Vorgehen

- Alle Tabellen, die keine Nichtschlüsselattribute oder nur ein Nichtschlüsselattribut enthalten, sind bereits in dritter Normalform.
- Alle übrigen Tabellen sind wie folgt zu untersuchen:
 1. Alle Gruppen von Nichtschlüsselattributen identifizieren, die vom Schlüssel nur transitiv (d. h. indirekt über ein anderes Nichtschlüsselattribut – die sog. Determinante) abhängig sind.
 2. Diese Nichtschlüsselattribute werden in eigene Tabellen ausgegliedert. Die Determinante wird zum Primärschlüssel der neuen Tabelle.
 3. Die transitiv abhängigen Nichtschlüsselattribute werden aus der Ursprungstabelle entfernt.
 4. Die Determinante verbleibt in der Ursprungstabelle als Nichtschlüsselattribut.

3. Normalform – Vorgehen

- Prüfen, welche Relationen bereits in 3. Normalform sind

<u>SkiNr</u>	<u>GenreID</u>	<u>EID</u>
FRC4R	001	2
FRC4R	002	2
FRC4R	002	3
FRC4R	003	1
FRC4W	001	1
FRC4W	002	1
FRC4W	003	2
FRC4W	003	1
SR	003	1
SR	004	1
SR	004	2
ST	005	1
ST	003	3
ST	001	5
SS	001	2
SS	002	2



<u>SkiNr</u>	SkiName	HNR	Hersteller
FRC4R	RC4 Race	F0012	Fischer
FRC4W	RC4 Worldcup	F0012	Fischer
SR	Rocker	S0001	Salomon
ST	Threat	S0001	Salomon
SS	Sinox	S0002	Stöckli

<u>GenreID</u>	Genre
001	Kurzschwung
002	Carving
003	Freeride
004	Buckelpiste
005	Freestyle

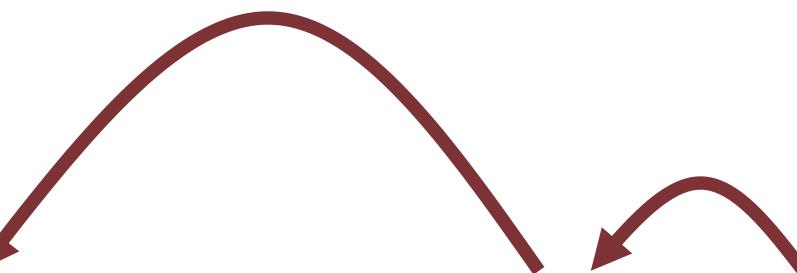


<u>EID</u>	Eignung
1	sehr gut
2	gut
3	mittel
5	mangelhaft



3. Normalform – Vorgehen

- Überprüfung auf transitive Abhängigkeiten

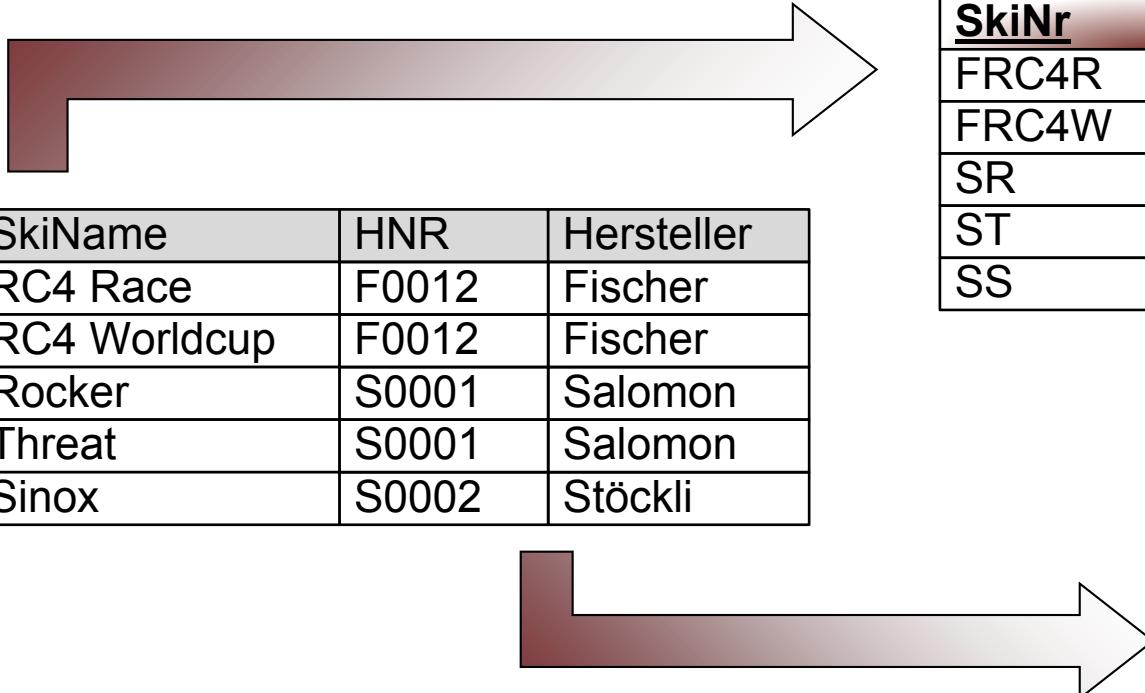


SkiNr	SkiName	HNR	Hersteller
FRC4R	RC4 Race	F0012	Fischer
FRC4W	RC4 Worldcup	F0012	Fischer
SR	Rocker	S0001	Salomon
ST	Threat	S0001	Salomon
SS	Sinox	S0002	Stöckli

3. Normalform – Vorgehen

■ Ausgliederung der Relation „Hersteller“

- Die Determinante wird als Schlüssel aus der Ursprungsrelation übernommen.
- Doppelte Einträge werden aus der neuen Relation entfernt.
- Die Determinante verbleibt in der Ursprungsrelation als Nichtschlüsselattribut.



<u>SkiNr</u>	SkiName	HNR	Hersteller
FRC4R	RC4 Race	F0012	Fischer
FRC4W	RC4 Worldcup	F0012	Fischer
SR	Rocker	S0001	Salomon
ST	Threat	S0001	Salomon
SS	Sinox	S0002	Stöckli

<u>SkiNr</u>	SkiName	HNR
FRC4R	RC4 Race	F0012
FRC4W	RC4 Worldcup	F0012
SR	Rocker	S0001
ST	Threat	S0001
SS	Sinox	S0002

<u>HNR</u>	Hersteller
F0012	Fischer
S0001	Salomon
S0002	Stöckli

3. Normalform – Vorgehen

- Relationenschema nach Herstellen der 3. Normalform

Relation „Skieignung für Genre“

SkiNr	GenreID	EID
FRC4R	001	2
FRC4R	002	2
FRC4R	002	3
FRC4R	003	1
FRC4W	001	1
FRC4W	002	1
FRC4W	003	2
FRC4W	003	1
SR	003	1
SR	004	1
SR	004	2
ST	005	1
ST	003	3
ST	001	5
SS	001	2
SS	002	2

Relation „Ski“

SkiNr	SkiName	HNR
FRC4R	RC4 Race	F0012
FRC4W	RC4 Worldcup	F0012
SR	Rocker	S0001
ST	Threat	S0001
SS	Sinox	S0002

Relation
„Eignung“

EID	Eignung
1	sehr gut
2	gut
3	mittel
5	mangelhaft

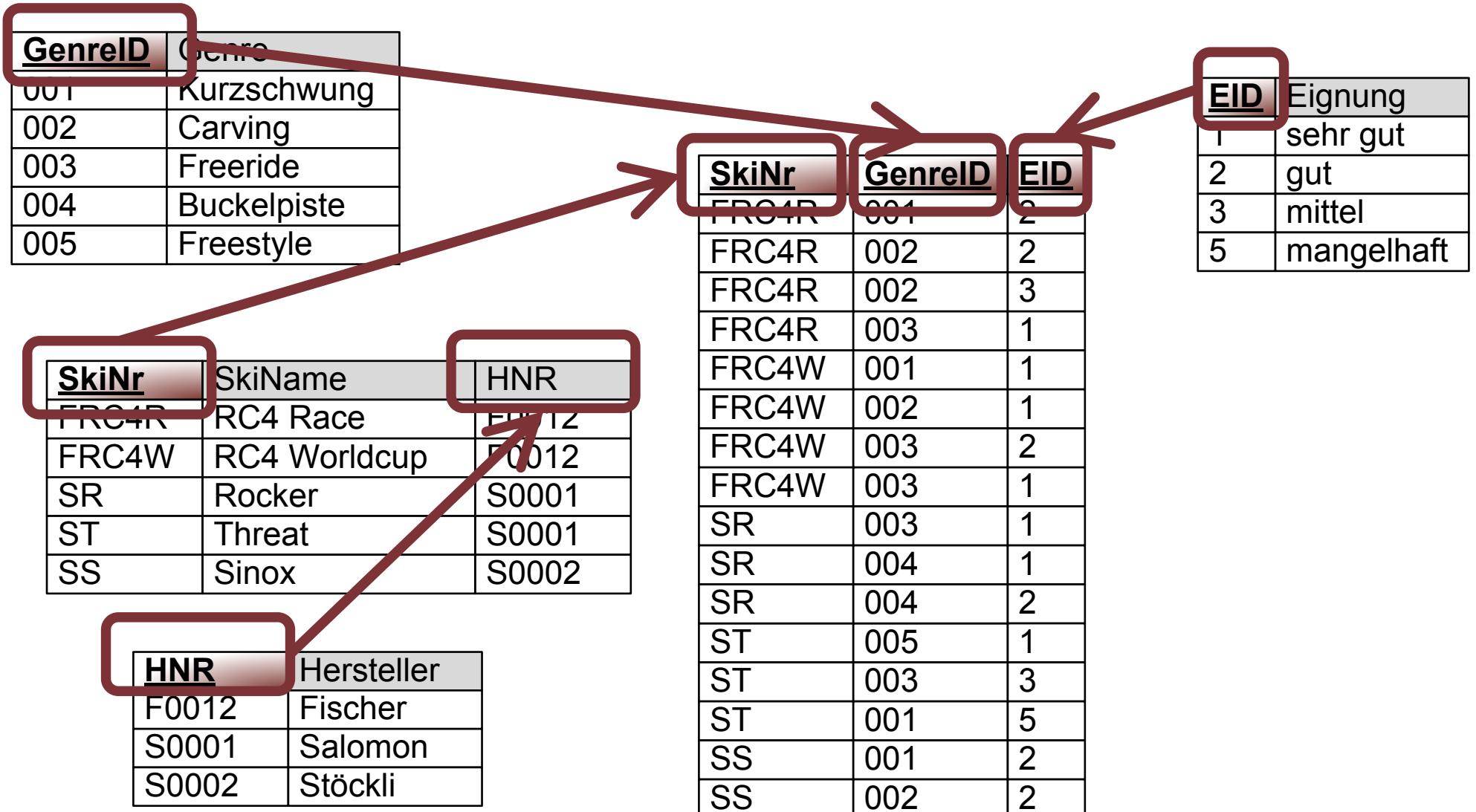
Relation „Genre“

GenreID	Genre
001	Kurzschwung
002	Carving
003	Freeride
004	Buckelpiste
005	Freestyle

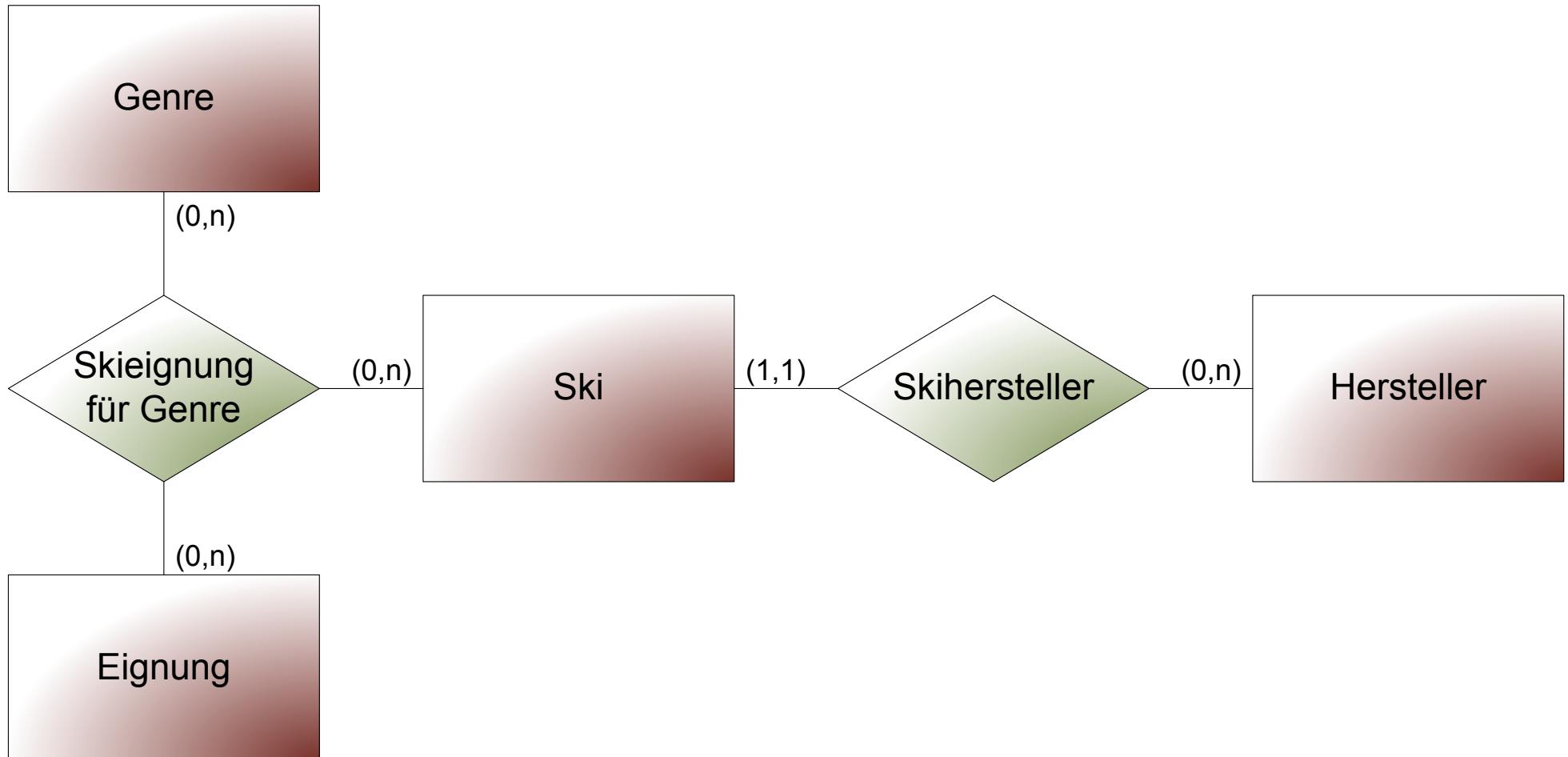
Relation
„Hersteller“

HNR	Hersteller
F0012	Fischer
S0001	Salomon
S0002	Stöckli

Finales Relationenschema

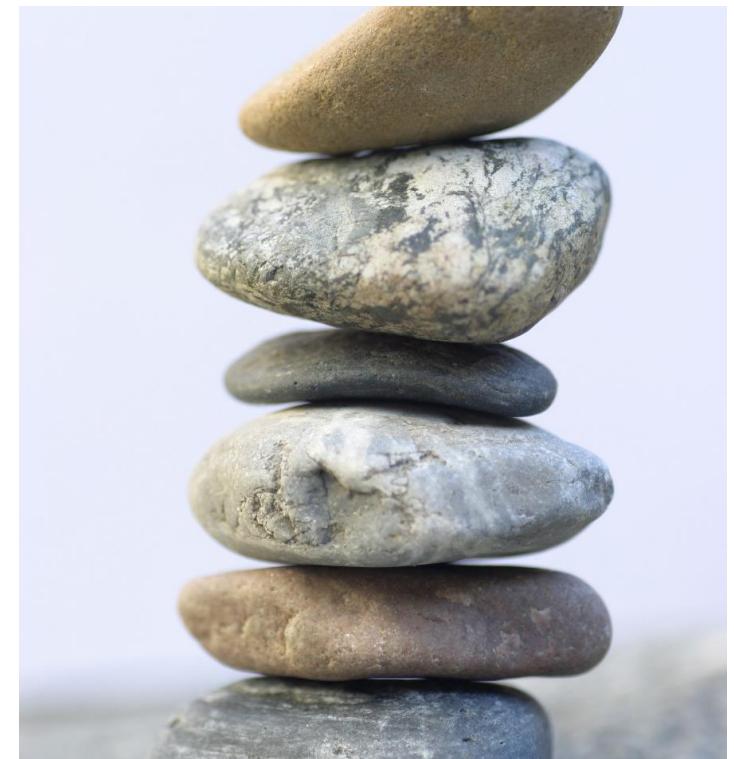


Ein mögliches ERM für das Szenario



Normalisierung – Merksatz

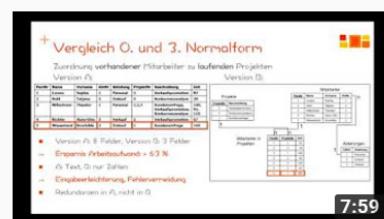
- Im relationalen Datenbankschema sind Attribute abhängig von ...
 - ... dem Schlüssel ... (1. NF)
 - ... dem ganzen Schlüssel ... (2. NF)
 - ... und von nichts als dem Schlüssel. (3. NF)



Weiterführendes Material



- Videos zu Normalisierung
 - <https://www.youtube.com/watch?v=sFG5pR5016k>



Datenbanken - Normalisierung - 1. Normalform

BildungInteraktiv • 61.000 Aufrufe • vor 2 Jahren

In diesem Video zeige ich, warum die Optimierung der Datenstrukturen einer **Datenbank** wichtig ist und wie man dabei vorgeht ...

- <https://www.youtube.com/watch?v=kAuZ-v-HBtA>

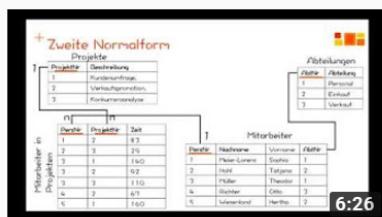


Datenbanken - Normalisierung - 2. Normalform

BildungInteraktiv • 53.000 Aufrufe • vor 2 Jahren

In diesem Video zeige ich, was man unter der zweiten **Normalform** versteht. Anhand einer einfachen Beispieldatenbank, die auch ...

- <https://www.youtube.com/watch?v=x183UkdsPQ8>



Datenbanken - Normalisierung - 3. Normalform

BildungInteraktiv • 45.000 Aufrufe • vor 2 Jahren

In diesem Video zeige ich, was man unter der dritten **Normalform** versteht. Anhand einer einfachen Beispieldatenbank, die auch ...

Übung

P#	P_Name	Abt#	Abt-Name	Pj#	Pj-Name	Pj-Std
101	Müller	1	Motoren	11, 12	A, B	60, 40
102	Meier	2	Karosserie	13	C	100
103	Krause	2	Karosserie	11, 12, 13	A, B, C	20, 50, 30
104	Schmidt	1	Motoren	11, 13	A, C	80, 20

- Überführen Sie die Tabelle zur Verwaltung von Personalinformationen bis in die 3. Normalform.
 - Angelegt wurde eine „naive“ Tabelle „PERSONAL-PROJEKT“. Mitarbeiterinnen und Mitarbeiter können an mehreren Projekten beteiligt sein. Es wird jeweils notiert, wie viele Stunden sie in einem Projekt geleistet haben.

- Relationales Datenbankmodell**
- Normalisierung**
- Datenintegrität**
- Zusammenfassung und Ausblick**

Motivation (1/2)

Datenkonsistenz / Datenintegrität:

- „In der Datenbankorganisation (...) die Korrektheit der gespeicherten Daten im Sinn einer widerspruchsfreien und vollständigen Abbildung der relevanten Aspekte des erfassten Realitätsausschnitts.“
 - Quelle: <http://wirtschaftslexikon.gabler.de/Archiv/57132/datenintegritaet-v6.html>
- → logisch korrekter Zustand der Daten

Inkonsistente Daten / Daten ohne Integrität: welche Fehler erkennen Sie (abgesehen von Verletzungen der Normalform)?

kunden				
<u>idKunde</u>	name	postleitzahl	ortName	arbeitgeber
1	Schmitt	10000	Musterhausen	Bäckerei Zimmermann GmbH
2	Müller	10000	Musterausen	Schlüter & Co KG
	Müller	79132	Coburg	Bäckerei Zimmermann

Quelle: http://www.informatikzentrale.de/_files/06datenbanken/datenmodellierung05_datenintegritaet.pdf

Motivation (2/2)

Inkonsistente Daten / Daten ohne Integrität: welche Fehler erkennen Sie?

kunden

<u>idKunde</u>	<u>name</u>	<u>postleitzahl</u>	<u>arbeitgeberFK</u>
1	Schmitt	10000	1
2	Müller	10000	14
3	Maier	79312	7

orte

<u>postleitzahl</u>	<u>name</u>
10000	Musterhausen
79098	Freiburg

Quelle: http://www.informatikzentrale.de/_files/06datenbanken/datenmodellierung05_datenintegritaet.pdf

Integritätsregeln

Datenintegrität gewährleistet durch Integritätsbedingungen:

■ Domänenintegrität / Bereichsintegrität

- Attribute sind nur gültig, wenn sie einen bestimmten Wertebereich haben

id	name	geburtsdatum
1	Smith	hihihi

■ Entitätsintegrität

- Jeder Datensatz ist eindeutig definiert (z.B. durch PRIMARY KEY).

id	name	geburtsdatum
1	Smith	2012-12-12
1	Sponz	2012-12-13

■ Referentielle Integrität

- Beziehungen zwischen Tabellen müssen synchronisiert bleiben. (s.u.)

■ Benutzerdefinierte Integrität

- Sonstige vom Benutzer festgelegte Regeln (z.B.: Datum nicht vor 01.01.2000)

Referentielle Integrität (1/3)

artist_id	artist_name
1	Bono
2	Cher
3	Nuno Bettencourt

artist_id	album_id	album_name
3	1	Schizophrenic
4	2	Eat the rich
3	3	Crave (single)

- Primärschlüssel-Fremdschlüssel-Beziehungen müssen intakt sein.
- Auch Beziehungsintegrität genannt: Attributwerte eines Fremdschlüssels müssen als Attributwert des Primärschlüssels vorhanden in der entsprechenden Tabelle vorhanden sein.
 - Fremdschlüssel müssen also immer auf existierende Datensätze verweisen.

kunden

<u>idKunde</u>	<u>name</u>	<u>postleitzahl</u>	<u>arbeitgeberFK</u>
1	Schmitt	10000	1
2	Müller	10000	14
3	Maier	79312	7

Keine Tabelle zu arbeitgeberFK gefunden.

orte

<u>postleitzahl</u>	<u>name</u>
10000	Musterhausen
79098	Freiburg

Kein Datensatz in orte gefunden.

Quelle: http://www.informatikzentrale.de/_files/06datenbanken/datenmodellierung05_datenintegritaet.pdf

Referentielle Integrität (2/3)

artist_id	artist_name
1	Bono
2	Cher
3	Nuno Bettencourt

artist_id	album_id	album_name
3	1	Schizophrenic
4	2	Eat the rich
3	3	Crave (single)

- Referentielle Integrität muss beim **Löschen eines Datensatzes** beachtet werden:

kunden

<u>id</u>	Kunde	name	postleitzahl
1	Schmitt	10000	
2	Müller	10000	
3	Maier	79312	

orte

<u>postleitzahl</u>	name
10000	Musterhausen
79312	Emmendingen

Auch eine Änderung des Datensatzes (PLZ 10000 ersetzen durch 99999) würde zu Inkonsistenzen führen.

→ Mögliche Lösung: Änderungsweitergabe (im Beispiel: Ändern aller Datensätze mit PLZ 10000 auf 99999 in kunden)

Löschen dieses Datensatzes würde zu Inkonsistenzen führen.

→ Mögliche Lösung: Löschweitergabe (im Beispiel: Löschen aller Datensätze mit PLZ 10000 in kunden)

Referentielle Integrität (3/3)

artist_id	artist_name
1	Bono
2	Cher
3	Nuno Bettencourt

artist_id	album_id	album_name
3	1	Schizophrenic
4	2	Eat the rich
3	3	Crave (single)

- Referentielle Integrität muss beim **Einfügen eines Datensatzes** beachtet werden:

kunden

<u>id</u>	Kunde	name	postleitzahl
1	Schmitt	10000	
2	Müller	10000	
3	Maier	79312	
4	Huber	80985	

orte

postleitzahl	name
10000	Musterhausen
79098	Freiburg

Einfügen dieses Datensatzes würde zu Inkonsistenzen führen.

→ Mögliche Lösung: Erst die PLZ in orte anlegen (kann durch die Anwendung, die auf das DBMS zugreift, bspw. unterstützt werden)

Konsistente Transformation: Commit / Rollback

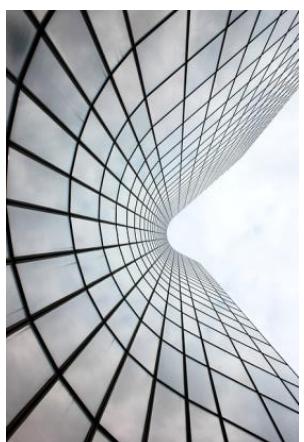
- Eine **Transaktion** muss eine Datenbank von einem Zustand mit Integrität in einen anderen Zustand mit Integrität überführen.
 - Während der Verarbeitung kann die Integrität kurzfristig verletzt werden.
 - Nach jeder durch eine Transaktion definierten Reihe von Veränderungen (Einfügen, Löschen oder Ändern von Daten) – d.h. nach dem commit – wird die Datenbank auf Integritätsbedingungen geprüft.
 - Sind diese nicht erfüllt, muss die gesamte Transaktion so zurück abgewickelt werden, dass der vorige Zustand wiederhergestellt wird („Rollback“).
- Transaktionen schützen auch vor DBMS-Abstürzen: erst wenn alle Veränderungen abgeschlossen sind, wird das commit gesetzt. Stürzt die Datenbank vorher ab, wird automatisch ein Rollback gestartet.

Agenda – VL 3 & 4

- Relationales Datenbankmodell
- Normalisierung
- Datenintegrität
- Zusammenfassung und Ausblick

Zusammenfassung (VL 3 & 4) und Ausblick

- In den letzten beiden Vorlesungen haben wir uns mit der logischen Datenbankmodellierung befasst.
- Sie sollten nun
 - logische Datenbankmodelle auf Basis des relationalen Modells erzeugen können,
 - konzeptionelle Modelle (ERM) in logische (relationale) Modelle überführen können,
 - relationale Modelle normalisieren können und
 - ein erstes Grundverständnis für Datenintegritätsbedingungen erlangt haben.
- In den folgenden Veranstaltungen werden wir uns mit der konkreten Umsetzung von Datenmodellen in einem DBMS (PostgreSQL) befassen und im Detail die entsprechende DDL, DML und DQL (alles auf Basis der Structured Query Language, SQL) kennenlernen.



Fragen

Vielen Dank für Ihre Aufmerksamkeit!



Lösungs- vorschlag

Relationen-Darstellung

Benutzer: (email, passwort, vorname, nachname)

Kunde: (email, kdnr, strasse, hausnummer, plz, ort, geburtsdatum)

Mitarbeiter: (email, manr, telefonnr, #arbeitet_in abtid)

Abteilung: (abtid, bezeichnung, #uebergeordnet abtid)

Produkt: (pnr, bezeichnung, angelegt_am, verfuegbarkeit, preis, #kategorie)

Kategorie: (katid, bezeichnung, #uebergeordnet katid)

Warenkorb: (wkid, erstelldatum, #kunde_email)

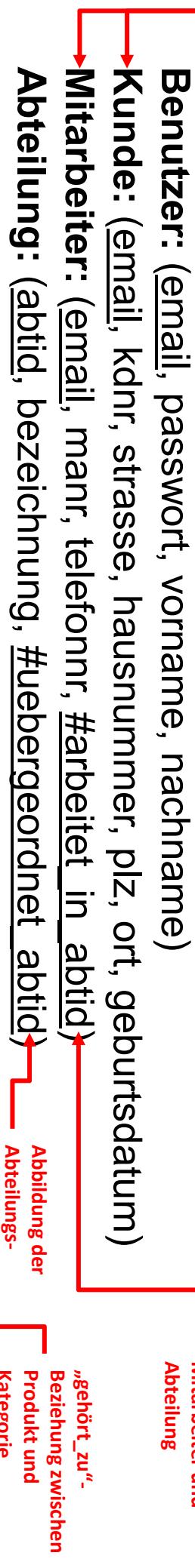
Warenkorb_Prodkt_Zuordnung: (wkid, pnr, menge)

Bestellung: (bestnr, bestelldatum, #wkid, #bestellstatus, #bearbeitet_email)

Status: (statusid, bearbeitungsstatus)

Hinweise

Durch die Generalisierung wurden die Primärschlüssel hier auf email gesetzt (geerbt von Benutzer). In der Datenbank sorgen wir dann für eindeutige kdnr & manr (UNIQUE!).
Achtung: andere Lösungsmöglichkeiten existieren (werden in BW342 nicht behandelt)!



„zugeordnet“-Beziehung zwischen Bestellung und Warenkorb.
Achtung: diese 1:1-Beziehung könnte auch in der Warenkorb-Relation abgebildet werden

Bestellung: (bestnr, bestelldatum, #wkid, #bestellstatus, #bearbeitet_email)

Status: (statusid, bearbeitungsstatus)

„zugeordnet“-Beziehung zwischen Bestellung und Warenkorb.
Achtung: diese 1:1-Beziehung könnte auch in der Warenkorb-Relation abgebildet werden

Lösungs- vorschlag

Schritt 1: Herstellen der 1.NF

P#	P_Name	Abt#	Abt-Name	Pi#	Pj-Name	Pj-Std
101	Müller	1	Motoren	11	A	60
101	Müller	1	Motoren	12	B	40
102	Meier	2	Karosserie	13	C	100
103	Krause	2	Karosserie	11	A	20
103	Krause	2	Karosserie	12	B	50
104	Schmidt	1	Motoren	11	A	80
104	Schmidt	1	Motoren	13	C	20

Die Angabe der Projekte war nicht atomar, sondern eine Liste atomarer Werte (bei Pj# Liste von int, bei Pi-Name Liste von String).
 Herstellung der 1.NF durch Einfügen der entsprechenden Zeilen.

Achtung: der Primärschlüssel ändert sich!
 Durch P# und Pj# wird jeder Datensatz eindeutig identifiziert.

Schritt 2: Prüfen, ob die 2.NF auch schon vorliegt:

Die zweite Normalform liegt noch nicht vor, da P_Name von einem Teil (einer Untermenge) des Primärschlüssels (P#, Pi#) abhängig ist: von P#. Das gilt auch für Abt# und Abt-Name.

Weiterhin ist auch Pj-Name von nur einem Teil des Primärschlüssels abhängig (nämlich von Pi#).

Achtung: Dass Abt-Name von Abt# abhängig ist, ist an dieser Stelle noch nicht wichtig. Abt# ist nicht Teil vom Primärschlüssel!

Lösungs- vorschlag

Schritt 3: Herstellen der 2.NF



P#	P_Name	Abt#	Abt-Name	Pi#
101	Müller	1	Motoren	11
101	Müller	1	Motoren	12
102	Meier	2	Karosserie	13
103	Krause	2	Karosserie	11
103	Krause	2	Karosserie	12
103	Krause	2	Karosserie	13
104	Schmidt	1	Motoren	11
104	Schmidt	1	Motoren	13

Pj#	Pj-Name	Pj-Std
A	60	
B	40	
C	100	
A	20	
B	50	
C	30	
A	80	
C	20	

Notwendige Schritte:
 (1) Alle Gruppen von Nichtschlüsselattributen identifizieren, die nur von einem Teil des Schlüssels funktional abhängig sind und diese jeweils mit diesem Schlüsselteil in eine eigene Tabelle überführen.

Gruppe 1: P_Name, Abt#, Abt-Name
Gruppe 2: Pj-Name

P#	Pj#
101	A
101	B
102	C
103	A
103	B
103	C
104	A
104	B
104	C

Personal-Projekt:

Gruppe 1

Personal:

Gruppe 2

Projekt:

P#	Pj#	Pj-Std
101	11	60
101	12	40
102	13	100
103	11	20
103	12	50
103	13	30
104	11	80
104	13	20

P#	P_Name	Abt#	Abt-Name
101	Müller	1	Motoren
102	Meier	2	Karosserie
103	Krause	2	Karosserie
104	Schmidt	1	Motoren

Lösungs- vorschlag

Schritt 4: Prüfen der 3.NF

Personal-Projekt:

P#	Pj#	Pj-Std
101	11	60
101	12	40
102	13	100
103	11	20
103	12	50
103	13	30
104	11	80
104	13	20

Personal:

P#	P_Name	Abt#	Abt-Name
101	Müller	1	Motoren
102	Meier	2	Karosserie
103	Krause	2	Karosserie
104	Schmidt	1	Motoren

Projekt:

Pj#	Pj-Name
11	A
12	B
13	C

Prüfung:

Die Relation enthält drei Nicht-Schlüsselattribute (P_Name, Abt#, Abt-Name).

Abt# hängt vom Schlüssel ab (da ja jeder Mitarbeiter einer Abteilung zugeordnet ist, d.h. abhängig vom Primärschlüssel-Wert ändert sich auch die Abt#).

Der Abt-Name hängt wiederum von Abt# ab, was aber indirekt über ein anderes Nicht-Schlüsselattribut existiert, das indirekt über ein anderes Nicht-Schlüsselattribut vom Primärschlüssel (transitiv) abhängig ist.

→ Personal ist noch nicht in der 3.NF

Prüfung:
 Die Relation enthält nur ein Nicht-Schlüsselattribut (Pj-Std). Also kann weiteres Nicht-Schlüsselattribut nicht existieren, das indirekt über ein anderes Nicht-Schlüsselattribut vom Primärschlüssel (transitiv) abhängig ist.

→ Projekt ist in der 3.NF

→ Personal-Projekt ist in der 3.NF

Notwendige Schritte zur Überführung von Personal in 3.NF:

- (1) Abt# und Abt-Name werden in eine eigene Tabelle aus gegliedert. Die Determinante (also Abt#) wird zum Primärschlüssel der neuen Tabelle.
- (2) Die transitiv abhängigen Nichtschlüsselattribute werden aus der Ursprungstabelle entfernt (also Abt-Name).
- (3) Die Determinante verbleibt in der Ursprungstabelle als Nichtschlüsselattribut (also Abt#).

Lösungs- vorschlag

Schritt 5: Überführung von Personal in die 3.NF

Personal-Projekt:

P#	Pj#	Pj-Std
101	11	60
101	12	40
102	13	100
103	11	20
103	12	50
103	13	30
104	11	80
104	13	20

Personal:

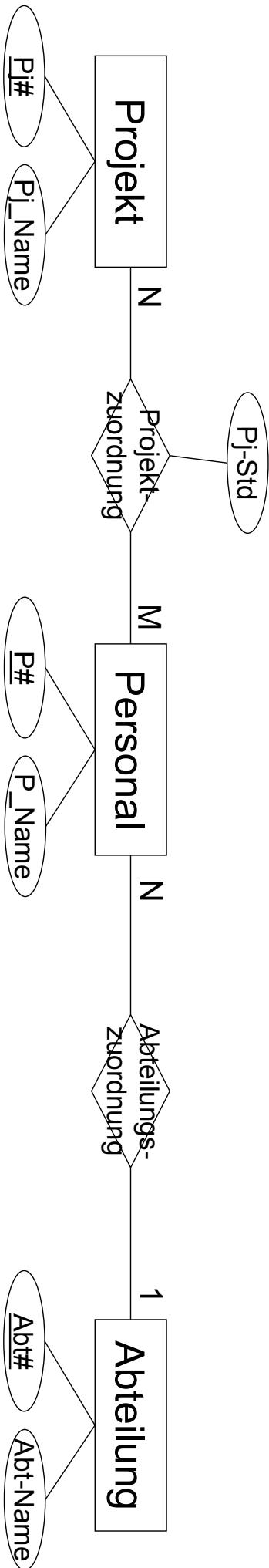
P#	P_Name	Abt#
101	Müller	1
102	Meier	2
103	Krause	2
104	Schmidt	1

Projekt:

Pj#	Pj-Name
11	A
12	B
13	C

Abteilung:

Abt#	Abt-Name
1	Motoren
2	Karosserie



**BW342 – Datenbanken I
(mit Praktikum)**

**Vorlesung 5
SQL-Grundlagen und DDL**



Ziele von VL 5



- Die Grundlagen von SQL kennen.
- Relationale Modelle mittels der Data Definition Language (DDL) in konkrete Tabellen des DBMS PostgreSQL überführen können.
- Spezielle Aspekte in SQL hinsichtlich der Definition von Attributen (Spalten) sowie in Bezug auf Datenintegrität kennenlernen und anwenden können.

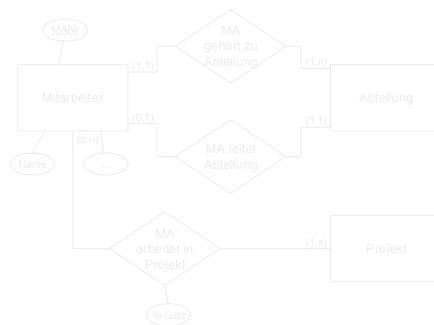
Themenüberblick



A Datenbankgrundlagen



B Konzeptionelle Modellierung



C Logische Modellierung

MID	Name	Adresse	Gehalt
1234	Müller	Hofstraße 7	23.000 €
1235	Meyer	Hauptstraße 13b	45.678 €
1236	Schmidt	Kurze Straße 1	55.555 €
1237	Heinze	Neue Straße 34	44.333 €
1238	Mustermann	Musterstraße 8	80.000 €
1239

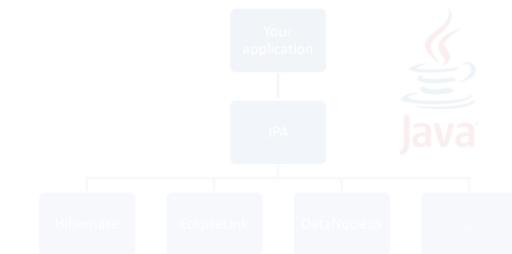
D Structured Query Language

```
-- filter your columns
SELECT col1, col2, col3, ... FROM table1
-- filter the rows
WHERE col4 = 1 AND col5 = 2
-- aggregate the data
GROUP by ...
-- limit aggregated data
HAVING count(*) > 1
-- order of the results
ORDER BY col2
```

E Anwendungsanbindung (JDBC)



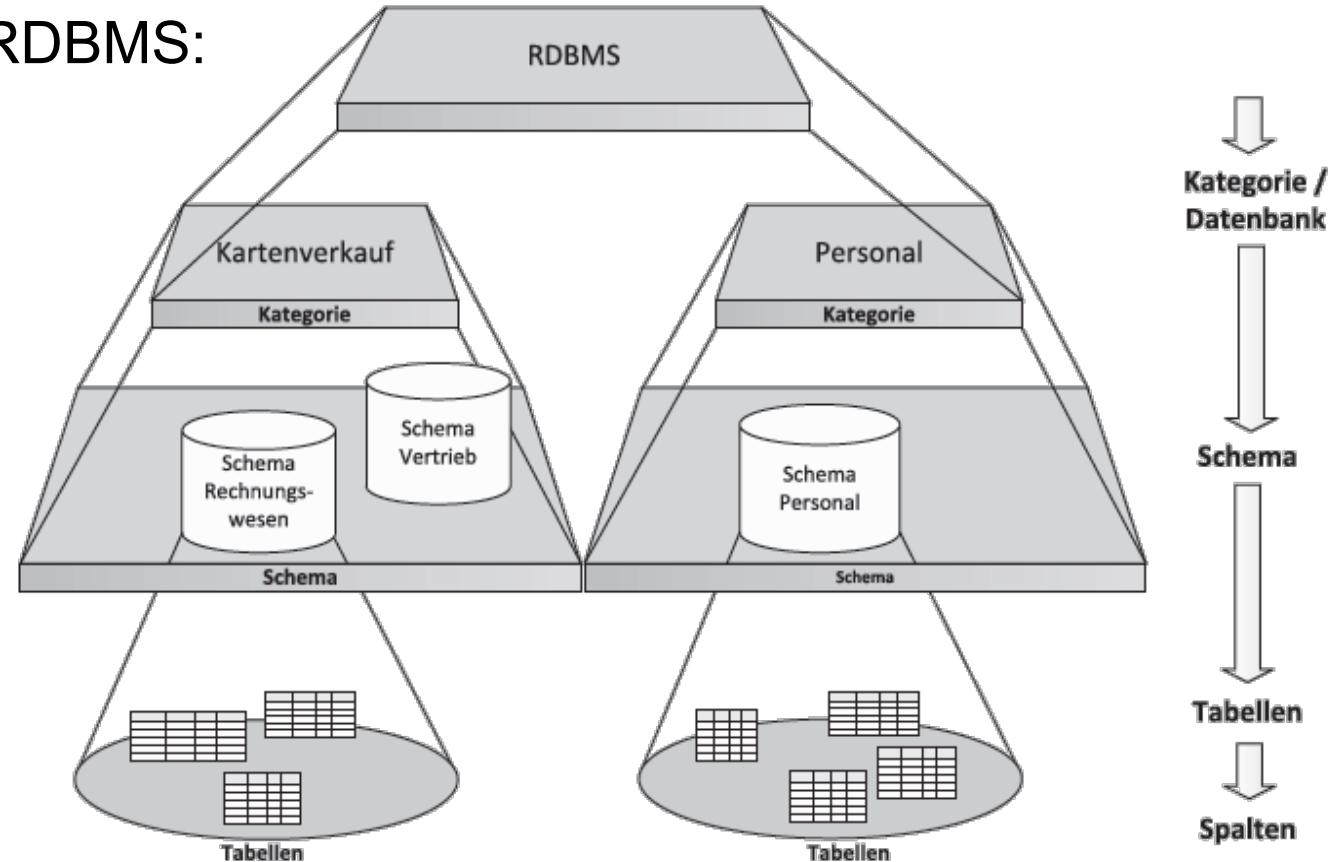
F Anwendungsanbindung (JPA)



- **Grundlagen zu RDBMS und SQL**
- Data Definition Language in PostgreSQL (CREATE, DROP)
- Data Definition Language in PostgreSQL (ALTER)
- Zusammenfassung und Ausblick

Relationales DBMS

- Bisher: konzeptionelle und logische Modellierung
- Jetzt: Umsetzung in RDBMS (relational DBMS)
 - Verwaltung und Nutzung des RDBMS mittels der Structured Query Language (SQL)
- Strukturierung eines RDBMS:



Structured Query Language

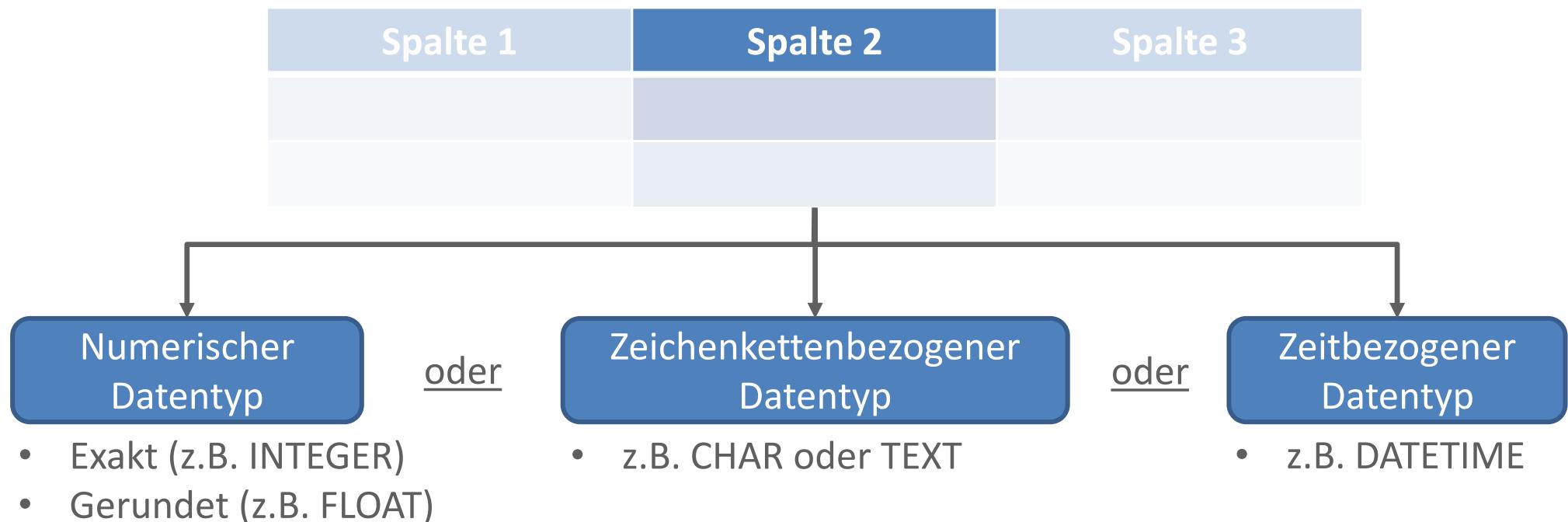
- SQL = Structured Query Language
- Deklarative Programmiersprache (im weiteren Sinne)
 - Zweck: Datenmanagement in relationalen Datenbanken
 - Algorithmische Lösung eines Problems nicht im Vordergrund
 - Beschreibung des Problems
 - Trennung von Arbeits- und Steuermechanismen
- Beispiel:
 - `SELECT Name FROM Kunde;`
- Standardsprache für relationales Datenbankmodell
 - Einige „Dialekte“ zur Erweiterung oder bzgl. unterschiedlicher Syntax

Historie von SQL

- 1975: IBM entwickelt SEQUEL für System R
- 1986: SQL-86 von ANSI offiziell standardisiert
- 1987: SQL-86 von ISO offiziell standardisiert
- 1992: SQL-92 / SQL2
 - Neue Datentypen, leicht veränderte Syntax
- 1999: SQL:1999 / SQL3
 - Reg-Ex-Matching, rekursive Abfragen, Trigger
- 2003: SQL:2003
 - Spalten mit automatisch generierten Werten
- 2006: SQL:2006
 - Import, Speicherung und Manipulation von XML-Daten in SQL-DB
 - Integration von XQuery- und SQL-Code
- 2008: SQL:2008
 - Weitere Trigger, Truncate-Statement
- 2011: SQL:2011
 - Temporale Definitionen und Manipulationen
- 2016: SQL:2016
 - Row Pattern Matching, Polymorphic Table Functions, JSON

SQL-Datentypen

- Jede Spalte hat **einen** Datentyp



Numerische Datentypen

<https://www.postgresql.org/docs/10/static/datatype-numeric.html>

Numerische
Datentypen

Name	Größe	Beschreibung	Wertebereich
INTEGER	2 bytes	small-range integer	-32768 to +32767
integer	4 bytes	typical choice for integer	-2147483648 to +2147483647
bigint	8 bytes	large-range integer	-9223372036854775808 to +9223372036854775807
decimal	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
numeric	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
real	4 bytes	variable-precision, inexact	6 decimal digits precision
double precision	8 bytes	variable-precision, inexact	15 decimal digits precision
smallserial	2 bytes	small autoincrementing integer	1 to 32767
serial	4 bytes	autoincrementing integer	1 to 2147483647
bigserial	8 bytes	large autoincrementing integer	1 to 9223372036854775807

Zeichenkettenbasierte Datentypen

Zeichenkettenbezogene
Datentypen

<https://www.postgresql.org/docs/10/static/datatype-character.html>

Name	Beschreibung
character varying(n), varchar(n)	variable-length with limit
character(n), char(n)	fixed-length, blank padded
text	variable unlimited length

Zeitbezogene Datentypen

Zeitbezogene
Datentypen

<https://www.postgresql.org/docs/10/static/datatype-datetime.html>

Name	Größe	Beschreibung	Unterer Wert	Oberer Wert	Auflösung
timestamp [(p)] [without time zone]	8 bytes	both date and time (no time zone)	4713 BC	294276 AD	1 microsecond
timestamp [(p)] with time zone	8 bytes	both date and time, with time zone	4713 BC	294276 AD	1 microsecond
date	4 bytes	date (no time of day)	4713 BC	5874897 AD	1 day
time [(p)] [without time zone]	8 bytes	time of day (no date)	00:00:00	24:00:00	1 microsecond
time [(p)] with time zone	12 bytes	time of day (no date), with time zone	00:00:00+1459	24:00:00-1459	1 microsecond
interval [fields] [(p)]	16 bytes	time interval	-178000000 years	178000000 years	1 microsecond

Sprachelemente von SQL

Bereich	Anwendung	Befehle	Objekte
DDL (Data Definition Language)	Anlegen, ändern und löschen von Datenbanken oder Tabellen	<ul style="list-style-type: none">• CREATE• ALTER• DROP	<ul style="list-style-type: none">• Database• Table
DML (Data Manipulation Language)	Anlegen, ändern und löschen von Datensätzen	<ul style="list-style-type: none">• INSERT• UPDATE• DELETE	
DQL (Data Query Language)	Abfrage von Datensätzen	<ul style="list-style-type: none">• SELECT	
DCL (Data Control Language)	Verwaltung von Zugriffsberechtigungen	<ul style="list-style-type: none">• GRANT• REVOKE	

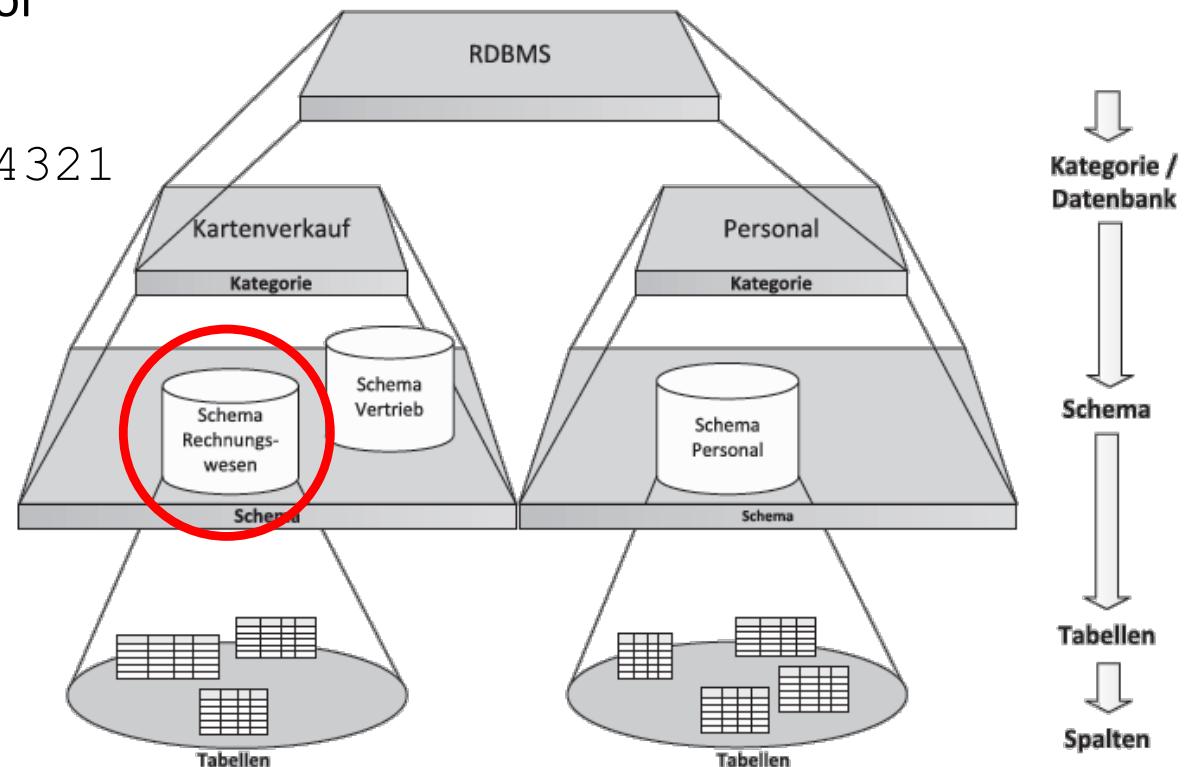
- Grundlagen zu RDBMS und SQL**
- Data Definition Language in PostgreSQL (CREATE, DROP)**
- Data Definition Language in PostgreSQL (ALTER)**
- Zusammenfassung und Ausblick**

Erzeugen von Schemata

pgAdmin → Tools / Query Tool

CREATE SCHEMA bw342_654321

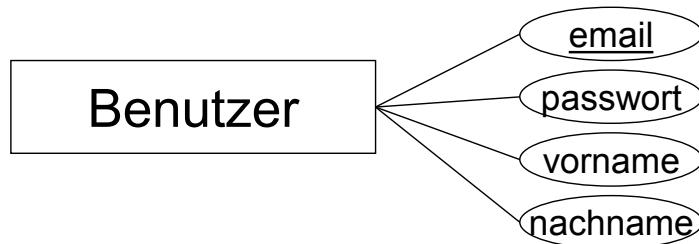
Achtung: Sie haben keine Rechte,
ein Schema zu erstellen.
Dies wurde für Sie bereits
gemacht.
Bitte setzen Sie das aktuelle
Schema wie folgt:



SET SCHEMA 'bw342_654321'

Erzeugen von Tabellen

Im ERM:



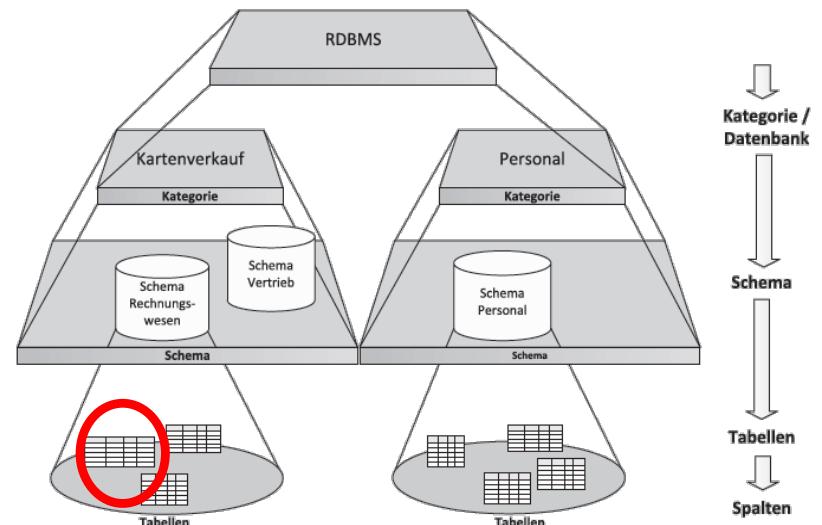
Im Relationenschema:

Benutzer: {[email], passwort, vorname, nachname]}

CREATE TABLE

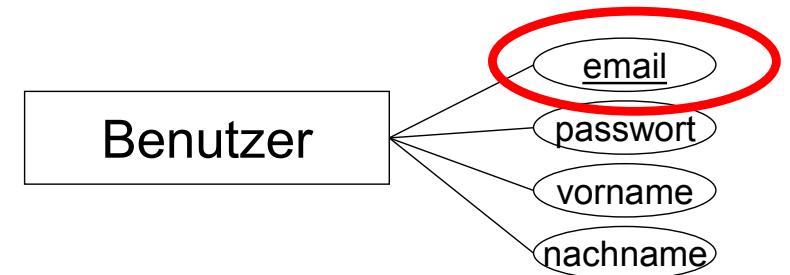
```
bw342_654321.BENUTZER (
    email          VARCHAR(100) ,
    vorname        VARCHAR(50)  ,
    nachname       VARCHAR(50)  ,
    passwort       VARCHAR(100)
)
```

DROP TABLE IF EXISTS bw342_654321.BENUTZER



IF EXISTS vermeidet, dass Ihre SQL-Scripte abstürzen, wenn die Tabelle nicht existiert beim DROP.

Erzeugen von Tabellen

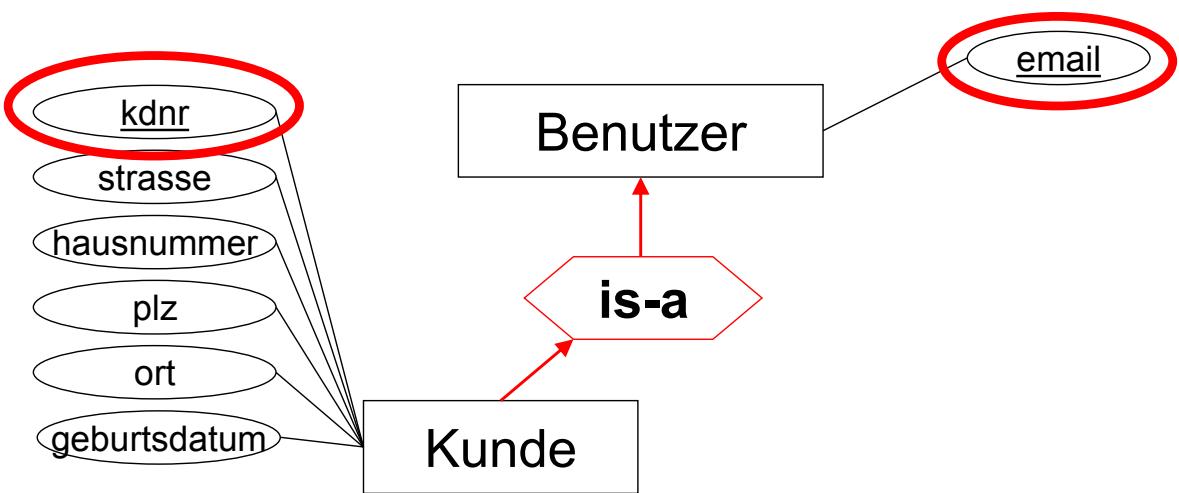


Pflichtfelder und Primärschlüssel:

```
CREATE TABLE bw342_654321.BENUTZER(
    email      VARCHAR(100) NOT NULL PRIMARY KEY ,
    vorname    VARCHAR(50)  NOT NULL ,
    nachname   VARCHAR(50)  NOT NULL ,
    passwort   VARCHAR(100) NOT NULL
)
```

```
CREATE TABLE bw342_654321.BENUTZER(
    email      VARCHAR(100) NOT NULL ,
    vorname    VARCHAR(50)  NOT NULL ,
    nachname   VARCHAR(50)  NOT NULL ,
    passwort   VARCHAR(100) NOT NULL ,
    PRIMARY KEY(email)
)
```

Erzeugen von Tabellen

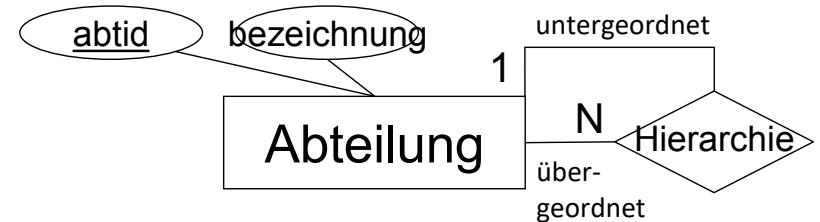


Eindeutige Spalten (nicht zwingend Primärschlüssel):

```
CREATE TABLE bw342_654321.KUNDE (
    email          VARCHAR(100)      NOT NULL PRIMARY KEY,
    kdnr           INTEGER          NOT NULL UNIQUE,
    strasse        VARCHAR(50)       NOT NULL ,
    hausnummer     VARCHAR(10)        NOT NULL ,
    plz            VARCHAR(5)         NOT NULL ,
    ort            VARCHAR(50)       NOT NULL ,
    geburtsdatum   DATE             NOT NULL ,
)
```

Erzeugen von Tabellen

Fremdschlüssel bei Hierarchie:



```
CREATE TABLE bw342_654321.ABTEILUNG (
    abtid          INTEGER      NOT NULL      PRIMARY KEY,
    bezeichnung    VARCHAR(50)  NOT NULL ,
    uebergeordnet_abtid  INTEGER REFERENCES bw342_654321.ABTEILUNG(abtid)
)
```

Erzeugen von Tabellen



Fremdschlüssel und Steuerung der Integrität:

```
CREATE TABLE bw342_654321.MITARBEITER (
    email          VARCHAR(100)      NOT NULL PRIMARY KEY,
    manr           INTEGER          NOT NULL UNIQUE,
    telefonnummer VARCHAR(50)       NOT NULL,
    arbeitet_in_abt INTEGER          REFERENCES bw342_654321.ABTEILUNG(abtid)
                                    ON DELETE SET NULL
)
```

- **RESTRICT**
 - Das Löschen/Ändern wird abgelehnt
- **SET NULL**
 - Der Fremdschlüssel würde auf „NULL“ gesetzt werden
- **SET DEFAULT**
 - Entspricht dem Setzen auf „NULL“, nur dass ein Standard festgelegt werden kann
- **CASCADE**
 - Datensatz wird aus der Eltern- und Kindtabelle gelöscht

Erzeugen von Tabellen



Auto-Inkrement:

```
CREATE TABLE bw342_654321.MITARBEITER(
    email          VARCHAR(100)      NOT NULL PRIMARY KEY,
    manr          SERIAL            NOT NULL UNIQUE,
    telefonnummer VARCHAR(50)      NOT NULL,
    arbeitet_in_abt INTEGER          REFERENCES bw342_654321.ABTEILUNG(abtid)
                                    ON DELETE SET NULL
)
```

Alternative mit FOREIGN KEY-Definition

```
CREATE TABLE bw342_654321.MITARBEITER(
    email          VARCHAR(100)      NOT NULL PRIMARY KEY,
    manr          SERIAL            NOT NULL UNIQUE,
    telefonnummer VARCHAR(50)      NOT NULL,
    arbeitet_in_abt INTEGER          NOT NULL,
    FOREIGN KEY (arbeitet_in_abt) REFERENCES bw342_654321.ABTEILUNG(abtid) ON DELETE SET NULL
)
```

Sie können alternativ also auch den Foreign Key am Ende des Befehls definieren (so wie beim Primary Key). Dies ist insbesondere bei zusammengesetzten Fremdschlüsseln zu verwenden.

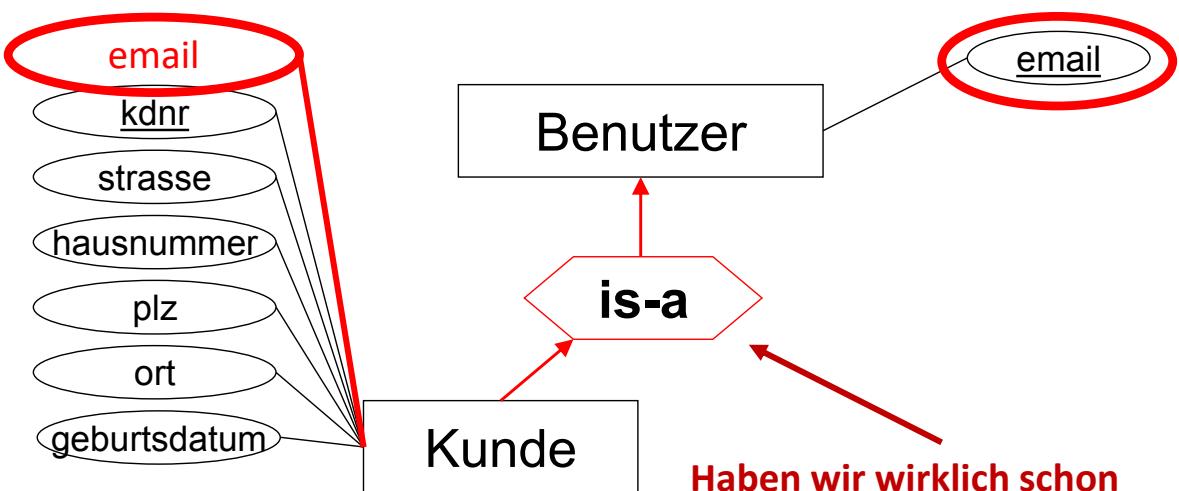


Übung

- 1. Definieren Sie das automatische Hochzählen auch für das Attribut kdnr in KUNDE.

Generalisierung

Gleichzeitig Primär- und Fremdschlüssel!



Haben wir wirklich schon diese Beziehung vollständig umgesetzt?

Es könnten Kunden angelegt werden, die gar keine Benutzer sind!

```
CREATE TABLE bw342_654321.KUNDE (
    email          VARCHAR(100)      NOT NULL PRIMARY KEY
    REFERENCES bw342_654321.BENUTZER(email),
    kdnr           SERIAL            NOT NULL UNIQUE,
    strasse        VARCHAR(50)       NOT NULL ,
    hausnummer    VARCHAR(10)        NOT NULL ,
    plz            VARCHAR(5)         NOT NULL ,
    ort            VARCHAR(50)       NOT NULL ,
    geburtsdatum   DATE             NOT NULL ,
```

)



Übung

- 2. Legen Sie auch für Mitarbeiter die Generalisierungs-/Spezialisierungsbeziehung in der Datenbank an.



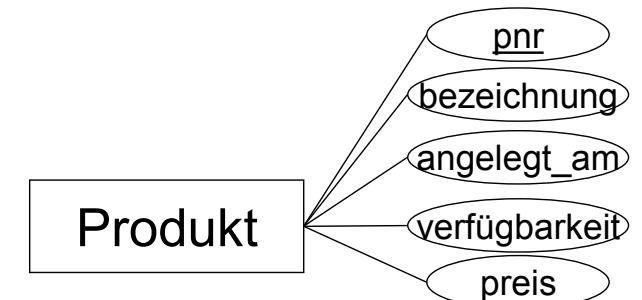
Übung

- 3. Definieren Sie in Analogie zur Abteilungshierarchie die Tabelle KATEGORIE.

Erzeugen von Tabellen

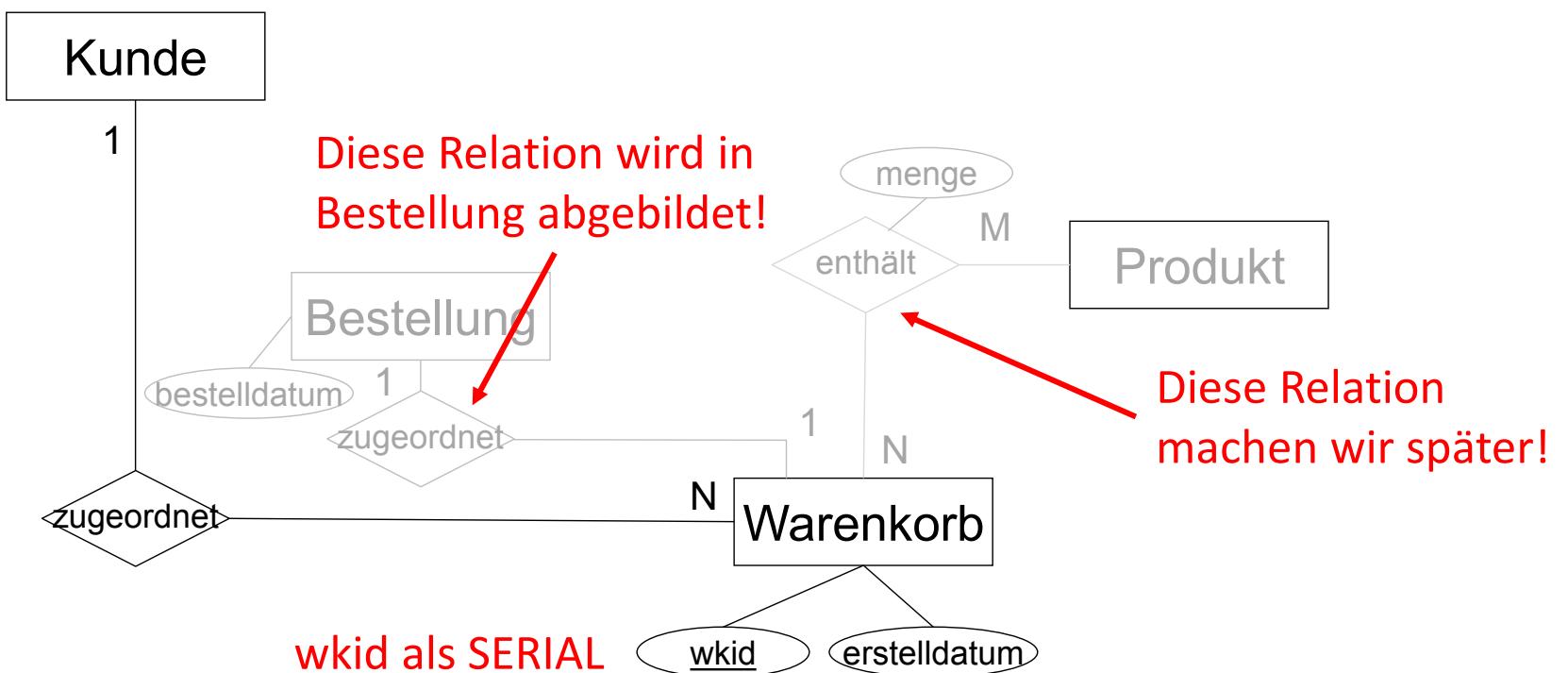
Default-Werte:

```
CREATE TABLE bw342_654321.PRODUKT (
    pnr                INTEGER      NOT NULL PRIMARY KEY ,
    bezeichnung        VARCHAR(50)  NOT NULL ,
    preis              DECIMAL(6, 2) NOT NULL DEFAULT 2.0 ,
    verfuegbarkeit     INTEGER      NOT NULL DEFAULT 0 ,
    angelegt_am       DATE ,
    kategorie          INTEGER      NOT NULL
                                REFERENCES bw342_654321.KATEGORIE(katid)
)
```



Übung

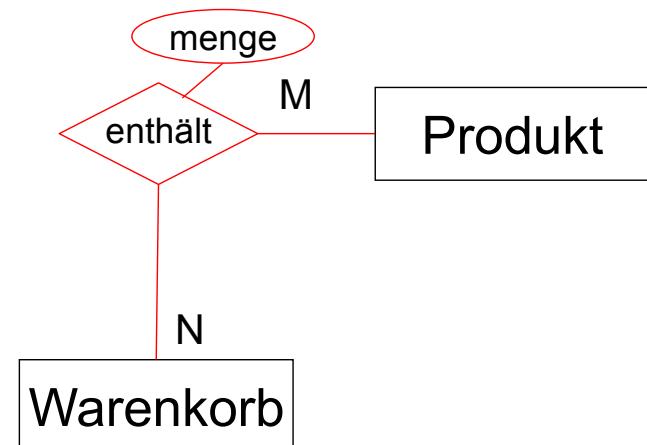
- 4. Definieren Sie die Tabelle Warenkorb wie im Relationenschema angegeben. Achten Sie auf den Fremdschlüssel zu Kunde! Dieser kann entweder über das geerbte Attribut email definiert werden, oder – weil wir kdnr als UNIQUE spezifiziert haben – auch über kdnr.



Erzeugen von Tabellen

N:M-Zuordnungstabellen:

```
CREATE TABLE bw342_654321.WARENKORB_PRODUKT_ZUORDNUNG (
    wkid          INTEGER      NOT NULL REFERENCES bw342_653421.WARENKORB(wkid),
    pnr          INTEGER      NOT NULL REFERENCES bw342_654321.PRODUKT(pnr),
    menge        INTEGER      NOT NULL,
    PRIMARY KEY (wkid, pnr)
)
```



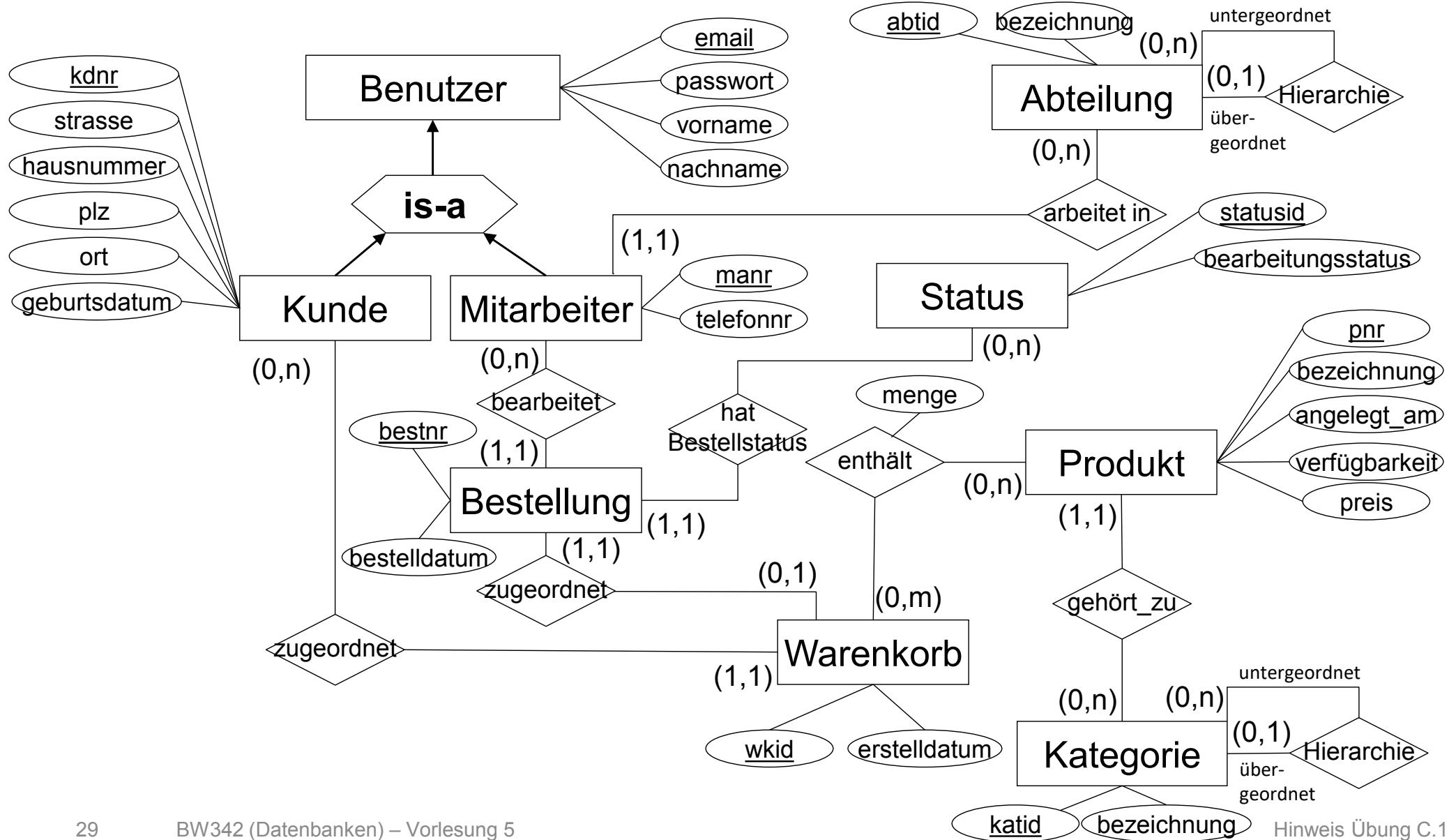
Übung



- Erstellen Sie sämtliche Tabellen des Fallbeispiels. Nutzen Sie dafür das unten angegebene Relationenschema. Löschen Sie ggf. bereits angelegte Tabellen, die nicht den Vorgaben entsprechen. Berücksichtigen Sie dabei eine korrekte Auswahl der Datentypen (wie sie für Sie sinnvoll jeweils erscheinen), die Festlegung von Primär- und Fremdschlüssels, die Einstellung der referentiellen Integrität (wie sie für Sie jeweils sinnvoll erscheint), sinnvolle DEFAULT-Werte und eine sinnvolle Nutzung von NOT NULL.
- **Relationenschema:**
 - **Benutzer:** {[email], passwort, vorname, nachname]}
 - **Kunde:** {[email], kdnr, strasse, hausnummer, plz, ort, geburtsdatum]}
 - **Mitarbeiter:** {[email], manr, telefonnr, #arbeitet_in_abtid]}
 - **Abteilung:** {[abtid], bezeichnung, #uebergeordnet_abtid]}
 - **Produkt:** {[pnr], bezeichnung, angelegt_am, verfuegbarkeit, preis, #kategorie]}
 - **Kategorie:** {[katid], bezeichnung, #uebergeordnet_katid]}
 - **Warenkorb:** {[wkid], erstelldatum, #kunde_email]}
 - **Warenkorb_Produkt_Zuordnung:** {[wkid, pnr, menge]}
 - **Bestellung:** {[bestnr, bestelldatum, #wkid, #bestellstatus, #bearbeitet_email]}
 - **Status:** {[statusid, bearbeitungsstatus]}

Definieren Sie die Tabellen Status und Bestellung so, dass statusid bzw. bestnr automatisch hochgezählt werden.

Bitte nutzen Sie dieses ERM



- Grundlagen zu RDBMS und SQL**
- Data Definition Language in PostgreSQL (CREATE, DROP)**
- Data Definition Language in PostgreSQL (ALTER)**
- Zusammenfassung und Ausblick**

Anpassung von Tabellenstrukturen

```
ALTER TABLE [ IF EXISTS ] [ ONLY ] name [ * ] action [, ... ]
```

... dabei kann „action“ sehr viel sein:

```
ADD [ COLUMN ] [ IF NOT EXISTS ] column_name data_type Spalte hinzufügen
```

```
DROP [ COLUMN ] [ IF EXISTS ] column_name [ RESTRICT | CASCADE ] Spalte löschen
```

```
ALTER TABLE distributors ADD PRIMARY KEY (dist_id); Primärschlüssel setzen
```

... und sehr viel mehr.

<https://www.postgresql.org/docs/current/static/sql-altertable.html>

Anpassung von Tabellenstrukturen

Ein paar Beispiele:

```
ALTER TABLE benutzer ALTER COLUMN vorname TYPE VARCHAR(200)
```

Ändern vom Typ der Spalte vorname

```
ALTER TABLE benutzer ALTER COLUMN passwort DROP NOT NULL
```

Löschen von Constraint „NOT NULL“ in Spalte passwort

```
ALTER TABLE benutzer DROP passwort
```

Löschen einer Spalte

```
ALTER TABLE benutzer ADD passwort VARCHAR(50)
```

Hinzufügen einer Spalte

- Grundlagen zu RDBMS und SQL**
- Data Definition Language in PostgreSQL (CREATE, DROP)**
- Data Definition Language in PostgreSQL (ALTER)**
- Zusammenfassung und Ausblick**

Zusammenfassung (VL 5) und Ausblick



- In der letzten Vorlesung haben wir intensiv mit der Data Definition Language anhand von PostgreSQL auseinandergesetzt und Sie haben ein Relationenschema in ein konkretes DBMS überführt.
- Sie sollten nun
 - Tabellen in PostgreSQL anlegen können,
 - die existierenden Datentypen kennen und wissen, wann sie anzuwenden sind,
 - spezielle Schlüsselwörter z.B. zur Definition von Schlüsseln, eindeutiger Werte, sowie weiterer Constraints kennen,
 - die unterschiedlichen Optionen der referentiellen Integrität in PostgreSQL kennen und einstellen können, sowie
 - existierende Tabellen hinsichtlich ihrer Definition anpassen können.
- In der folgenden Veranstaltung werden wir unsere Datenbank „mit Leben füllen“. Dazu nutzen wir die Data Manipulation Language von SQL.

Fragen



Vielen Dank für Ihre Aufmerksamkeit!

**BW342 – Datenbanken I
(mit Praktikum)**

**Vorlesung 6
SQL: DML**



Ziele von VL 6



- Relationale Datenbanken, die per DDL angelegt wurden, mittels der Data Manipulation Language (DML) mit Daten befüllen können.
- Existierende Daten in der relationalen Datenbank verändern können.

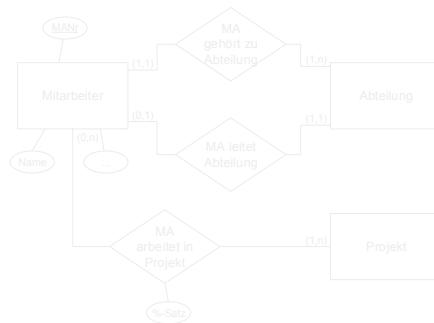
Themenüberblick



A Datenbankgrundlagen



B Konzeptionelle Modellierung



C Logische Modellierung

MID	Name	Adresse	Gehalt
1234	Müller	Hofstraße 7	23.000 €
1235	Meyer	Hauptstraße 13b	45.678 €
1236	Schmidt	Kurze Straße 1	55.555 €
1237	Heinze	Neue Straße 34	44.333 €
1238	Mustermann	Musterstraße 8	80.000 €
1239

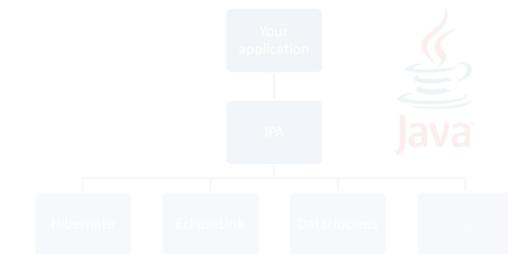
D Structured Query Language

```
-- filter your columns
SELECT col1, col2, col3, ... FROM table1
-- filter the rows
WHERE col4 = 1 AND col5 = 2
-- aggregate the data
GROUP by ...
-- limit aggregated data
HAVING count(*) > 1
-- order of the results
ORDER BY col2
```

E Anwendungsanbindung (JDBC)

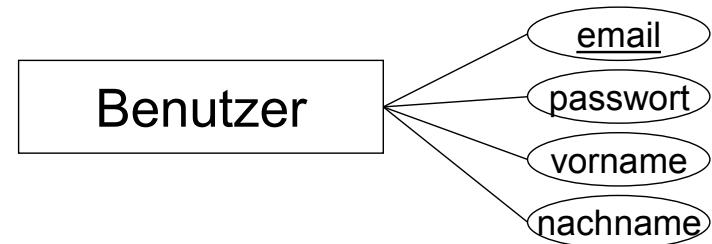


F Anwendungsanbindung (JPA)



- Data Manipulation Language in PostgreSQL – INSERT
- Data Manipulation Language in PostgreSQL – DELETE
- Data Manipulation Language in PostgreSQL – UPDATE
- Zusammenfassung und Ausblick

Einfügen von Datensätzen



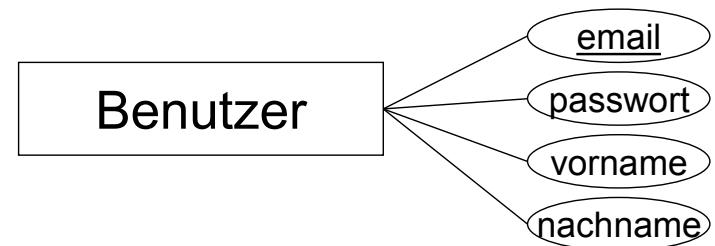
Einfügen mit Angabe aller Werte:

```
INSERT INTO bw342_654321.BENUTZER  
VALUES ('max.mustermann@abc.de', 'Max', 'Mustermann', 'geheim_pw')
```

Wir erinnern uns an die Definition von Benutzer:

```
CREATE TABLE bw342_654321.BENUTZER(  
    email      VARCHAR(100) NOT NULL ,  
    vorname    VARCHAR(50)  NOT NULL ,  
    nachname   VARCHAR(50)  NOT NULL ,  
    passwort   VARCHAR(100) NOT NULL ,  
    PRIMARY KEY(email)  
)
```

Einfügen von Datensätzen



Funktioniert dies?

```
INSERT INTO bw342_654321.BENUTZER  
VALUES ('max.mustermann@abc.de', 'Max', 'Mustermann', null)
```

Wir erinnern uns an die Definition von Benutzer:

```
CREATE TABLE bw342_654321.BENUTZER(  
    email      VARCHAR(100) NOT NULL ,  
    vorname    VARCHAR(50)  NOT NULL ,  
    nachname   VARCHAR(50)  NOT NULL ,  
    passwort   VARCHAR(100) NOT NULL ,  
    PRIMARY KEY(email)  
)
```



Übung

■ 1. Ändern der Tabellendefinition von BENUTZER

Funktioniert die oben dargestellt Einfügeoperation mit dem Null-Wert?

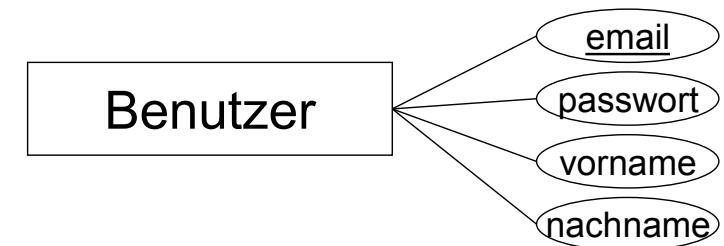
Falls nein, ist in der Definition des Schemas BENUTZER für die Spalte „NOT NULL“ gesetzt worden (wie gefordert). Ändern Sie mittels ALTER TABLE die Spaltendefinition von `passwort` in der Tabelle BENUTZER so, dass Null-Werte erlaubt sind.

Hilfe zu dem Befehl finden Sie z.B. hier:

- Vorlesungsunterlagen (VL5)
- <https://www.postgresql.org/docs/current/static/sql-altertable.html>
- Geben Sie bei Google bspw. `postgres alter table nullable` als Suchbegriffe ein. Die Seite Stackoverflow hat oft gute Antworten zu typischen Problemen.

Alternativ können Sie natürlich auch mittels DROP TABLE zunächst die Tabelle löschen und dann mit CREATE TABLE und geändertem Schema anlegen. Dies hat – neben dem Mehraufwand – den Nachteil, dass Sie alle eventuell in der Tabelle vorhandenen Daten verlieren. Es lohnt sich also, den Befehl ALTER TABLE zu verstehen und anwenden zu können.

Einfügen von Datensätzen



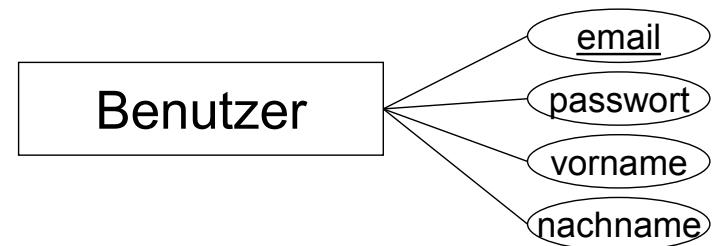
Bisher haben wir immer der Reihenfolge der Felddefinition folgend eingefügt:

```
INSERT INTO bw342_654321.BENUTZER  
VALUES ('max.mustermann@abc.de', 'Max', 'Mustermann', null)
```

Wir erinnern uns an die nun aktualisierte Definition von Benutzer:

```
CREATE TABLE bw342_654321.BENUTZER(  
    email      VARCHAR(100) NOT NULL ,  
    vorname    VARCHAR(50)  NOT NULL ,  
    nachname   VARCHAR(50)  NOT NULL ,  
    passwort   VARCHAR(100) ,  
    PRIMARY KEY(email)  
)
```

Einfügen von Datensätzen



Man kann aber auch die Felder, die man befüllen möchte, vorher angeben.

```
INSERT INTO bw342_654321.BENUTZER (email, vorname, nachname, passwort)  
VALUES ('max.mustermann@abc.de', 'Max', 'Mustermann', null)
```

So kann man auch Felder weglassen

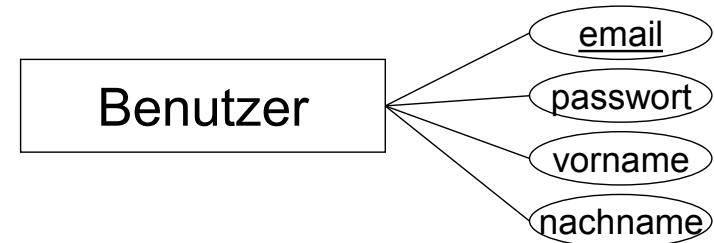
```
INSERT INTO bw342_654321.BENUTZER (email, vorname, nachname)  
VALUES ('max.mustermann@abc.de', 'Max', 'Mustermann')
```

Oder auch in einer anderen Reihenfolge Daten übergeben

```
INSERT INTO bw342_654321.BENUTZER (email, nachname, vorname)  
VALUES ('max.mustermann@abc.de', 'Mustermann', 'Max')
```

- Data Manipulation Language in PostgreSQL – INSERT
- Data Manipulation Language in PostgreSQL – DELETE
- Data Manipulation Language in PostgreSQL – UPDATE
- Zusammenfassung und Ausblick

Löschen von Datensätzen



Man kann alle Datensätze aus einer Tabelle mittels DELETE löschen:

Exportieren Sie vorher bitte die Tabelle Benutzer in eine csv-Datei (pgAdmin).

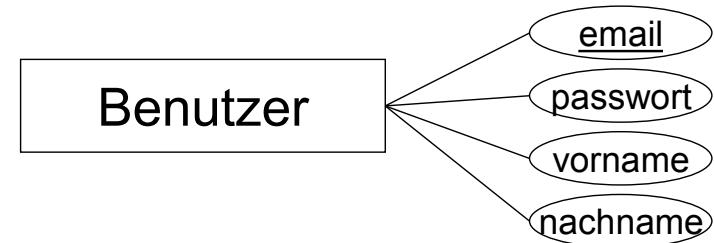
```
DELETE FROM bw342_654321.BENUTZER
```

Lassen Sie sich in pgAdmin den Tabelleninhalt von Benutzer anzeigen (es sollten 0 Zeilen sein).

Importieren Sie bitte die vorher exportierten Daten wieder in die Benutzer-Tabelle.

Oder: führen Sie das gespeicherte SQL-Statement noch einmal aus (Folie 9)

Löschen von Datensätzen



Mittels WHERE-Statement kann man einzelne Datensätze zum Löschen selektieren:

```
DELETE FROM bw342_654321.BENUTZER WHERE email = 'max.mustermann@abc.de'
```

Lassen Sie sich in pgAdmin den Tabelleninhalt von Benutzer anzeigen, um die Löschoperation zu prüfen.

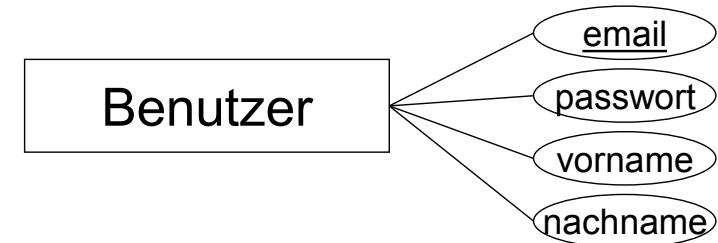
Nutzung logischer Operatoren und String-Vergleiche:

```
DELETE FROM bw342_654321.BENUTZER WHERE vorname = 'Max' AND nachname = 'Mustermann'
```

```
DELETE FROM bw342_654321.BENUTZER WHERE email LIKE '%mustermann%'
```

- Data Manipulation Language in PostgreSQL – INSERT
- Data Manipulation Language in PostgreSQL – DELETE
- Data Manipulation Language in PostgreSQL – UPDATE
- Zusammenfassung und Ausblick

Ändern von Datensätzen



Man kann alle Datensätze aus einer Tabelle mittels UPDATE aktualisieren:

Exportieren Sie vorher bitte die Tabelle Benutzer in eine csv-Datei (pgAdmin).

```
UPDATE bw342_654321.BENUTZER
```

```
SET passwort = 'geheim123'
```

Lassen Sie sich in pgAdmin den Tabelleninhalt von Benutzer anzeigen.

Auch hier können Sie wieder mittels des WHERE-Operators einzelne Datensätze für die Aktualisierung selektieren:

```
UPDATE bw342_654321.BENUTZER
```

```
SET passwort = '123geheim'
```

```
WHERE nachname LIKE 'Me%'
```



Übung

■ 2. Einfügen in Status-Tabelle

Legen Sie die folgenden Status an:

- Offen
- In Bearbeitung
- Auf dem Weg
- Geliefert

Fügen Sie im Anschluss noch den Status „Retoure“ ein.

Übung

■ 3. Anlegen von Kunden

Fügen Sie die folgenden Kundendatensätze ein. Bitte beachten Sie, dass hier sowohl die Benutzer- als auch die Kundentabelle gefüllt werden muss (in welcher Reihenfolge?).

Email	kdnr	Vorname	Nachname	Strasse	Hnr	PLZ	Ort	Geburtsdatum
m.l@abc.de	auto	Michael	Lindhausen	Schlossallee	4	67227	Frankenthal	22.06.2005
j.b@abc.de	auto	Jennifer	Bauer-bacher	Elisabethen-straße	1	67061	Ludwigshafen	11.03.1993
a.k@abc.de	auto	Alexander	Kalten-brunner	Bahnhofsweg	38	67061	Ludwigshafen	28.09.1994
m.b@abc.de	auto	Maya	Bäcker	Odenwaldstraße	3	76227	Karlsruhe	21.05.2006
m.w@abc.de	auto	Mike	Walddorf	Flughafenstraße	93	68159	Mannheim	10.07.2006

Weisen Sie nun sämtlichen Kunden das Passwort „geheim“ zu.



Übung

- 4. Anlegen von Abteilungen
- Bitte fügen Sie folgende Abteilungen ein:
 - ID=1, Bezeichnung="Shop", Übergeordnet=NULL
 - ID=2, Bezeichnung="Verkauf", Übergeordnet=1
 - ID=3, Bezeichnung="Einkauf", Übergeordnet=1



Übung

■ 5. Anlegen von Mitarbeitern

Email	manr	Vor-name	Nach-name	Telefon	Abteilung
h.m@abc.de	auto	Herbert	Meyer	06233 987123	Verkauf
a.b@abc.de	auto	Astrid	Bauer	0721 123987	Einkauf



Übung

■ 6. Anlegen von Kategorien

- Bitte fügen Sie folgende Kategorien ein:
 - ID=1, Bezeichnung="Alle Produkte", Übergeordnet=NULL
 - ID=2, Bezeichnung="Food", Übergeordnet=1
 - ID=3, Bezeichnung="Non-Food", Übergeordnet=1

Übung

■ 7. Anlegen von Produkten

pnr	Bezeichnung	Preis	Verfügbarkeit	Angelegt_am	Kategorie
1	Snickers	0.8	30	Heutiges Datum	Food
2	Banane	1.8	20	Heutiges Datum minus 1 Woche	Food
3	Zitrone	0.7	40	Heutiges Datum minus 1 Woche	Food
4	Wasser	0.3	100	Heutiges Datum	Food
5	Rotwein	15.25	50	Heutiges Datum	Food
6	Grillbratwurst	2.3	40	Heutiges Datum minus 2 Wochen	Food
7	Grillkohle	2.7	80	Heutiges Datum	Non-Food
8	Grillanzünder	1.8	80	Heutiges Datum	Non-Food

Übung

■ 8. Anlegen von Warenkörben und „Einlegen“ von Produkten

Legen Sie für die angegebenen Kunden Warenkörbe an und „legen“ Sie die angegebenen Produkte mit der Anzahl in diesen hinein (Verwendung der Tabelle Warenkorb-Produkt-Zuordnung):

Kunde	Produkt	Anzahl
m.l@abc.de	1	5
j.b@abc.de	2	40
	3	1
a.k@abc.de	6	10
	7	2
	8	2
	5	5
	1	2
m.b@abc.de	5	1
	2	2
m.w@abc.de	3	10
	4	2

Hinweis: um die jeweils richtige wkid für den Eintrag in
WARENKORB_PRODUKT_ZUORDNUNG herauszufinden, kann folgender Befehl helfen:

```
SELECT * FROM bw342_654321.WARENKORB
```

Das Erstelltdatum soll für die ersten beiden Kunden heute sein. Für die letzten drei Kunden heute minus 1 Woche.

Übung



■ 9. Anlegen Bestellungen & Zuweisung zu Warenkorb, Status und Bearbeiter

Nehmen Sie an, dass die Kunden m.l@abc.de, m.b@abc.de und a.k@abc.de den für sie eben angelegten Warenkorb bestellten möchten bzw. bestellt haben. Wir legen nun die entsprechenden Bestellungen an und ordnen diese den Warenkörben zu.

Fügen Sie folgende Bestellungen in die Datenbank ein:

- Kunde m.l@abc.de
 - Bestelldatum entspricht dem Warenkorb-Erstelldatum
 - wkid: ID des in Aufgabe 8 erstellten Warenkorbs des Kunden
 - Status: Offen
 - Bearbeitet von: Herbert Meyer
- Kunde m.b@abc.de
 - Bestelldatum entspricht dem Warenkorb-Erstelldatum + 1 Tag
 - wkid: ID des in Aufgabe 8 erstellten Warenkorbs des Kunden
 - Status: Auf dem Weg
 - Bearbeitet von: Herbert Meyer
- Kunde a.k@abc.de
 - Bestelldatum entspricht dem Warenkorb-Erstelldatum
 - wkid: ID des in Aufgabe 8 erstellten Warenkorbs des Kunden
 - Status: Geliefert
 - Bearbeitet von: Astrid Bauer

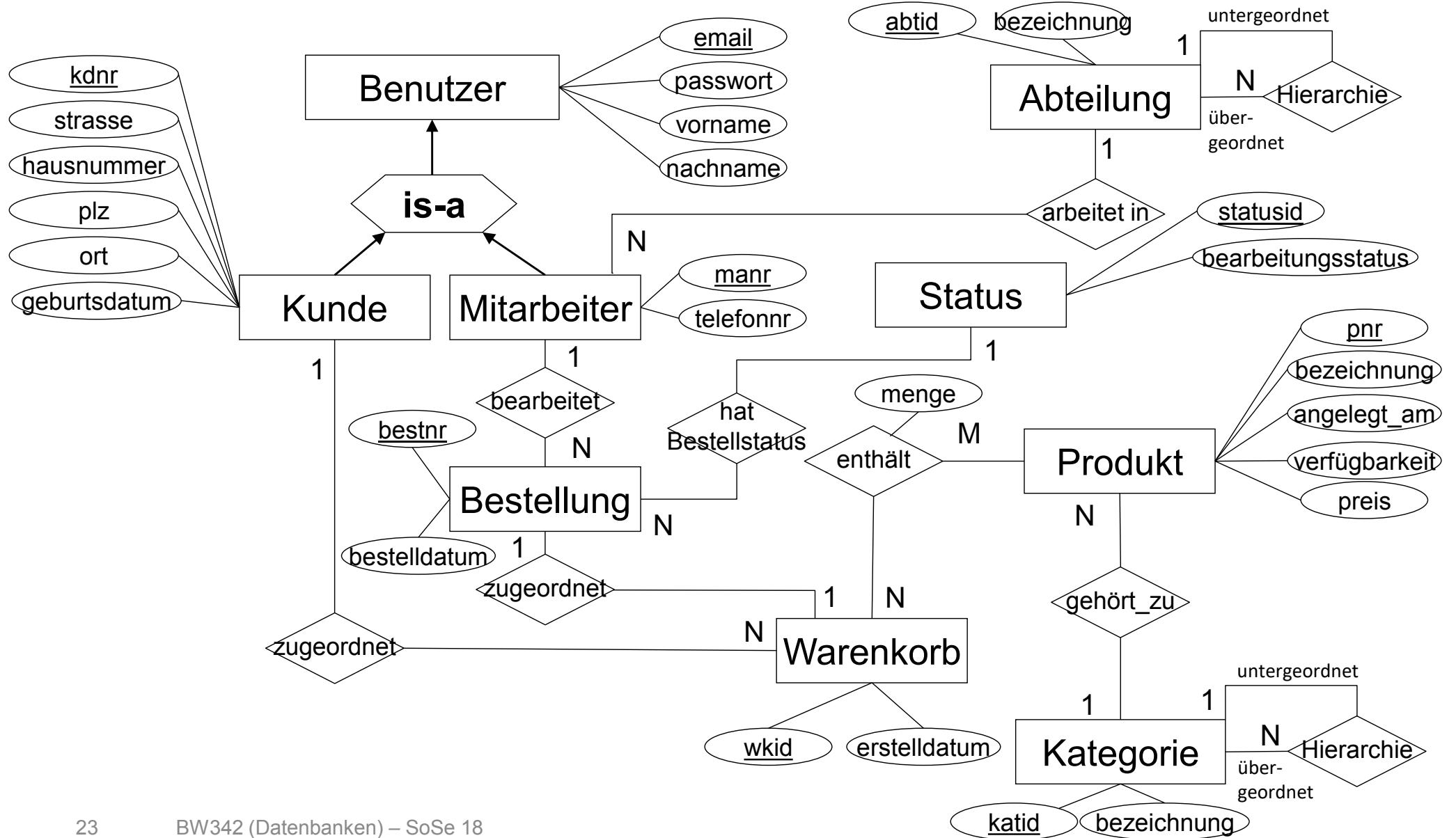
Hinweis: um die jeweils richtige wkid in WARENKORB zu finden, führen Sie folgenden Befehl aus:

```
SELECT * FROM bw342_654321.WARENKORB
```

Hinweis: um die jeweils richtige statusid in der Tabelle Status für den Fremdschlüssel in Bestellung herauszufinden, führen Sie folgenden Befehl aus:

```
SELECT * FROM bw342_654321.STATUS
```

Unser ERM



- Data Manipulation Language in PostgreSQL – INSERT
- Data Manipulation Language in PostgreSQL – DELETE
- Data Manipulation Language in PostgreSQL – UPDATE
- Zusammenfassung und Ausblick

Zusammenfassung (VL 6) und Ausblick



- In der letzten Vorlesung haben wir uns intensiv mit der Data Manipulation Language anhand von PostgreSQL auseinandergesetzt und Sie haben in ein Relationenschema eines konkreten DBMS Daten eingefügt.

- Sie sollten nun
 - Datensätze mittels INSERT einfügen können,
 - alle Datensätze einer Tabelle oder auch nur Teile davon mittels DELETE löschen können,
 - existierende Datensätze mittels UPDATE ändern können.

Fragen



Vielen Dank für Ihre Aufmerksamkeit!

**BW342 – Datenbanken I
(mit Praktikum)**

**Vorlesungen 7&8
SQL: DQL**

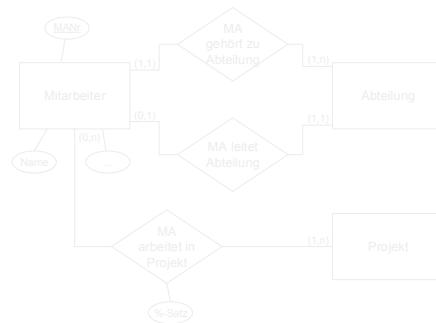
Themenüberblick



A Datenbankgrundlagen



B Konzeptionelle Modellierung



C Logische Modellierung

MID	Name	Adresse	Gehalt
1234	Müller	Hofstraße 7	23.000 €
1235	Meyer	Hauptstraße 13b	45.678 €
1236	Schmidt	Kurze Straße 1	55.555 €
1237	Heinze	Neue Straße 34	44.333 €
1238	Mustermann	Musterstraße 8	80.000 €
1239

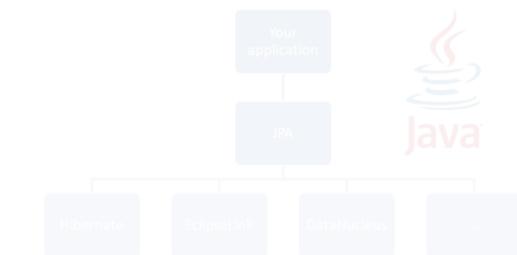
D Structured Query Language

```
-- filter your columns  
SELECT col1, col2, col3, ... FROM table1  
-- filter the rows  
WHERE col4 = 1 AND col5 = 2  
-- aggregate the data  
GROUP by ...  
-- limit aggregated data  
HAVING count(*) > 1  
-- order of the results  
ORDER BY col2
```

E Anwendungsanbindung (JDBC)



F Anwendungsanbindung (JPA)

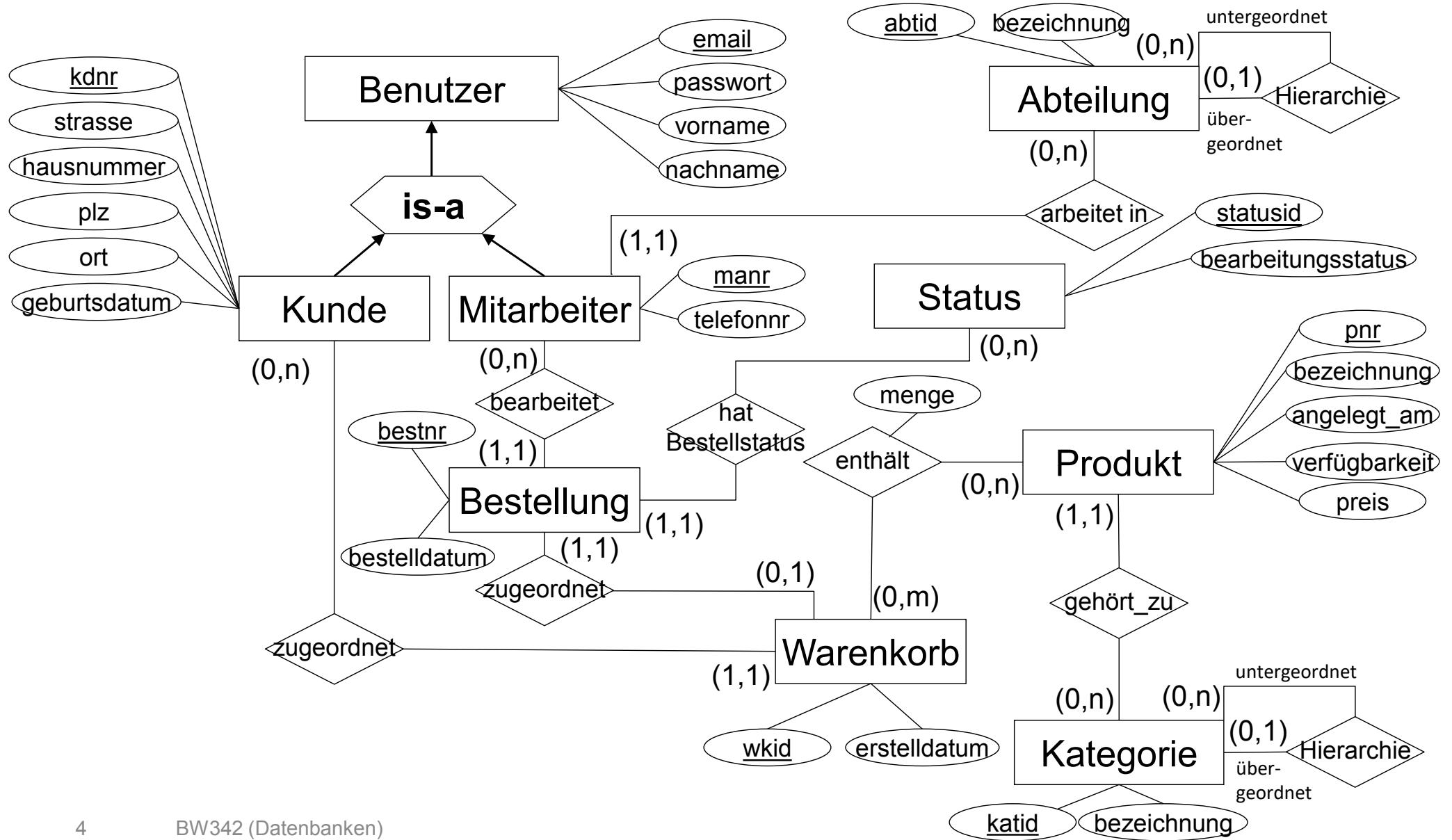


Ablauf DQL

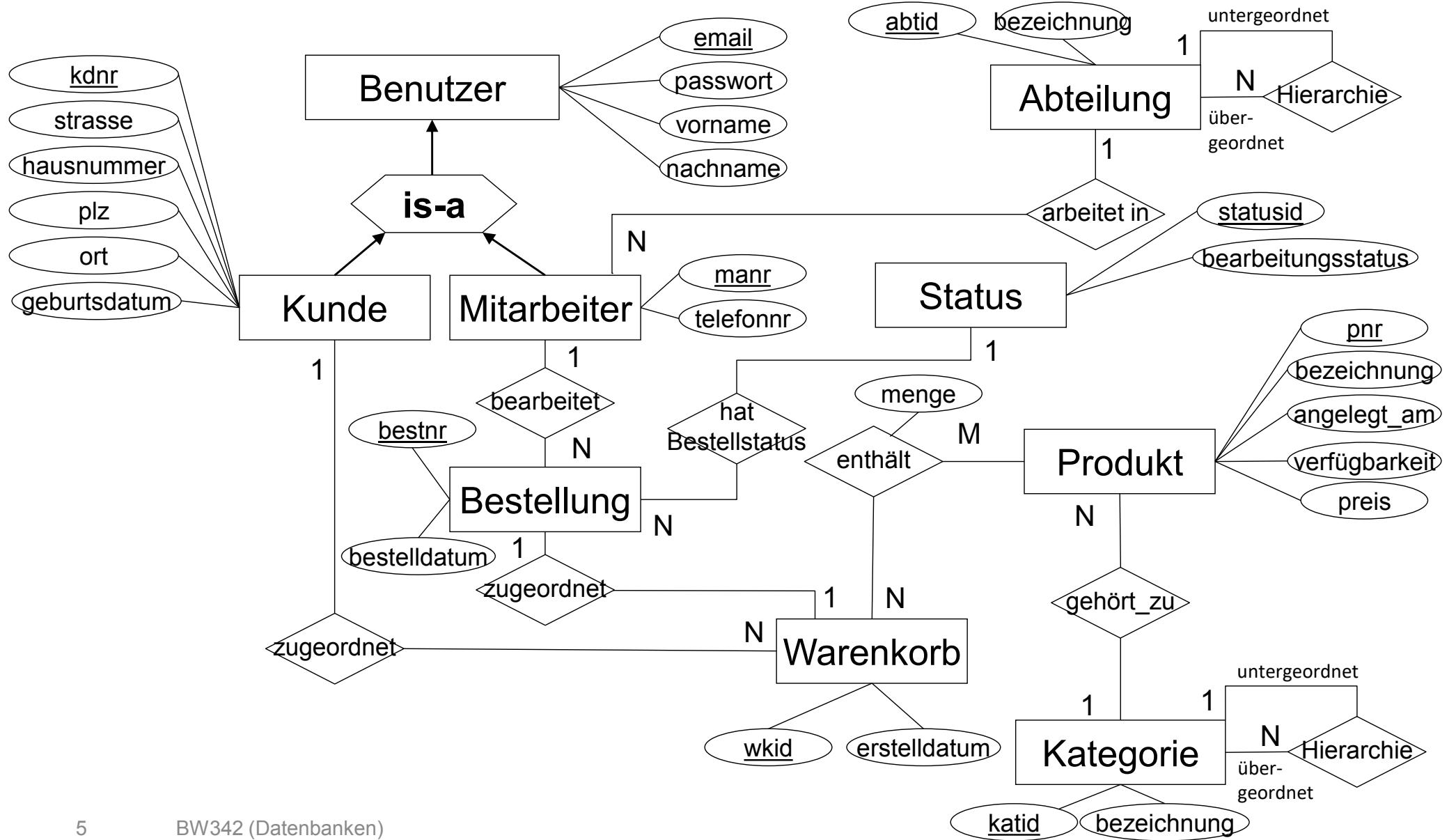


- Wir werden ab jetzt anhand des Skripts die Data Query Language gemeinsam besprechen.
- In diesen Slides finden Sie noch einmal das ERM sowie das Relationenmodell, das wir in der Veranstaltung verwenden.

Unser ERM



Unser ERM



Relationales Modell



Relationen-Darstellung

Benutzer: {[email], passwort, vorname, nachname]}

Kunde: {[email], kdnr, strasse, hausnummer, plz, ort, geburtsdatum]}

Mitarbeiter: {[email], manr, telefonnr, #arbeitet_in_abtid]}

Abteilung: {[abtid], bezeichnung, #uebergeordnet_abtid]}

Produkt: {[pnr], bezeichnung, angelegt_am, verfuegbarkeit, preis, #kategorie]}

Kategorie: {[katid], bezeichnung, #uebergeordnet_katid]}

Warenkorb: {[wkid], erstelldatum, #kunde_email]}

Warenkorb_Produkt_Zuordnung: {[wkid, pnr, menge]}

Bestellung: {[bestnr, bestelldatum, #wkid, #bestellstatus, #bearbeitet_email]}

Status: {[statusid, bearbeitungsstatus]}

Zusammenfassung (VL 7&8) und Ausblick



- In den letzten Vorlesungen haben wir uns intensiv mit der Data Query Language anhand von PostgreSQL auseinandergesetzt.
- In der folgenden Veranstaltung werden wir die PostgreSQL-Datenbank aus Java heraus mittels JDBC anbinden.

Fragen



Vielen Dank für Ihre Aufmerksamkeit!