

Gliederung

1.	Einführung
2.	Datenbankentwurf
3.	Datenbankimplementierung
4.	Physische Datenorganisation
5.	Anfrageoptimierung
6.	Transaktionsverwaltung
7.	Datensicherheit und Wiederherstellung
8.	Business Intelligence

Gliederung

2.

Datenbankentwurf

Definition Datenbankentwurf

Datenmodellierung

Phasen des Datenbank-Entwurfsprozesses

Konzeptioneller Entwurf mit dem Entity-Relationship-Modell

Logischer Entwurf mit dem Relationalen Datenmodell

Transformation eines Entity-Relationship-Modells
in ein Relationales Datenmodell


Definition eines Datenbankentwurfs

Die Aufgabe des Datenbankentwurfs ist der Entwurf der logischen und physischen Struktur einer Datenbank so, dass die Informationsbedürfnisse der Benutzer in einer Organisation für bestimmte Anwendungen adäquat befriedigt werden können.

(Vossen, 2008)

Darunter sind folgende **Teilaspekte** zu verstehen:

- Ermittlung einer **logischen** Datenbankstruktur, d. h. welche Informationen werden in welchen Informationseinheiten (meist: DB-Tabellen) gespeichert?
- Entwurf einer **physischen** Datenbankstruktur, d. h. Aspekte der Datenspeicherung (auf Speichermedien) unter Berücksichtigung von effizienten Zugriffsmöglichkeiten.
- Berücksichtigung von **Randbedingungen** der zu realisierenden Anwendung, d. h. Fehlerprüfungen, Abhängigkeiten zwischen Daten.



Der Datenbankentwurf ist eine durchaus anspruchsvolle Aufgabe, die in verschiedenen **Teilschritten** unter Zuhilfenahme spezieller **Techniken** durchgeführt wird.

Gliederung

2.

Datenbankentwurf



Definition Datenbankentwurf



Datenmodellierung



Phasen des Datenbank-Entwurfsprozesses



Konzeptioneller Entwurf mit dem Entity-Relationship-Modell

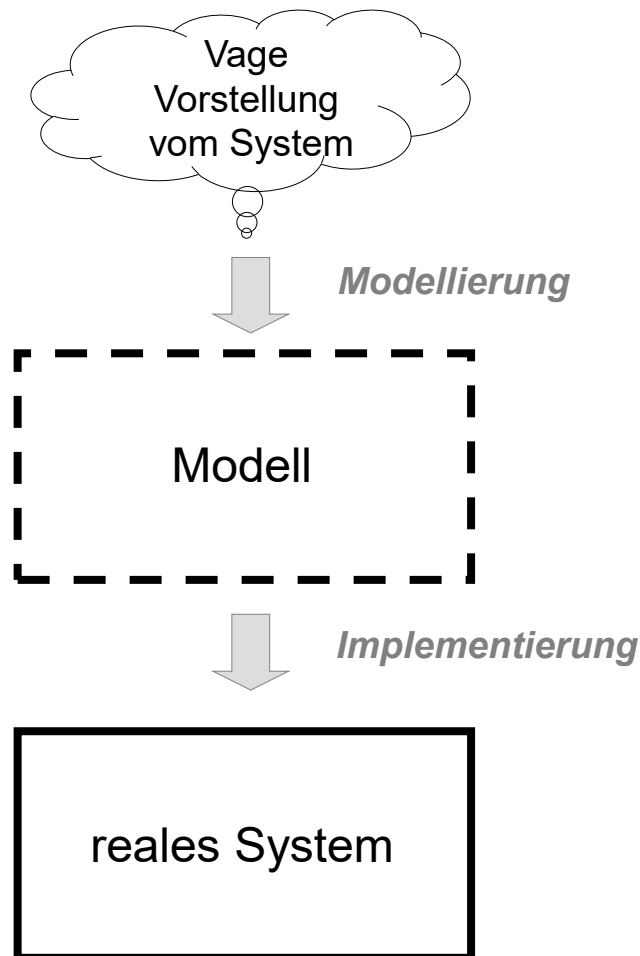


Logischer Entwurf mit dem Relationalen Datenmodell



Transformation eines Entity-Relationship-Modells
in ein Relationales Datenmodell

Datenmodellierung

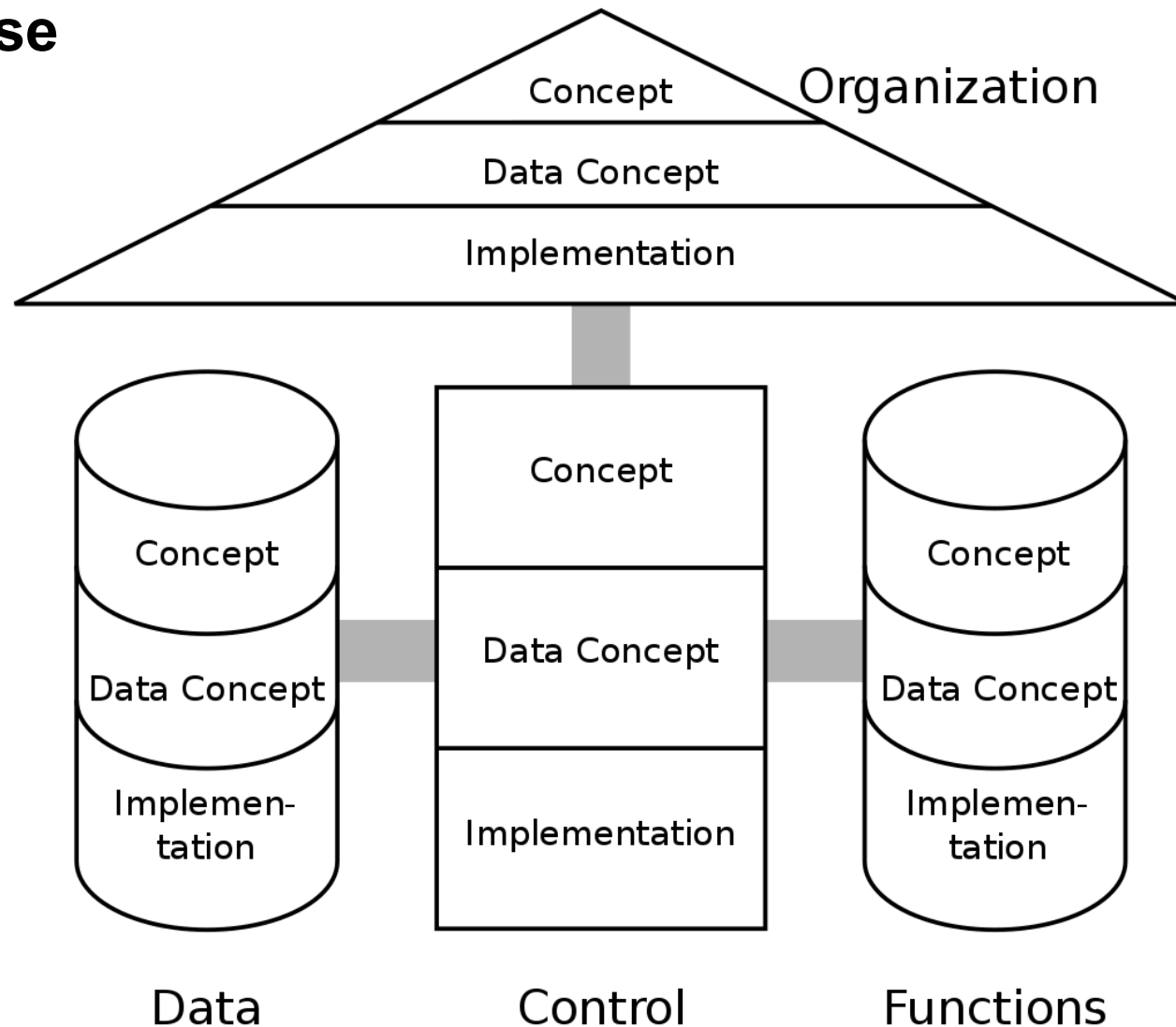


Vorteile:

- ✓ **Reduktion der Komplexität:**
Nur relevante Zusammenhänge werden abgebildet (Abstraktion)
- ✓ **Präzision und Vermeidung von Missverständnissen:**
Einsatz formaler Modellierungssprachen mit eingeschränkten Ausdrucksmitteln und genau definierter Syntax und Semantik
- ✓ **Analyse und Test des Systemverhaltens noch vor der Implementierung möglich**

Datenmodellierung

ARIS House (Scheer)

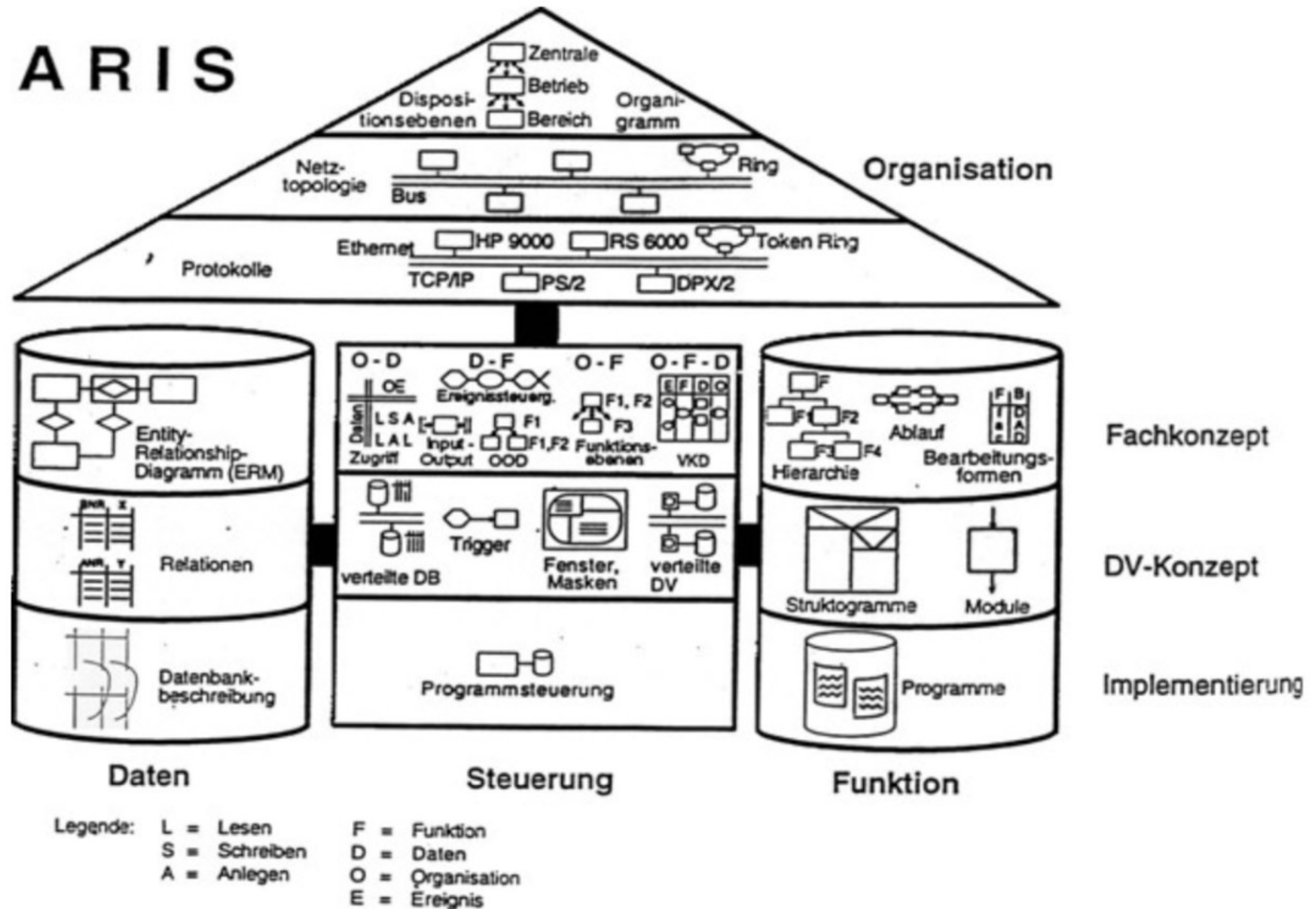


1. Einführung
2. Datenbankentwurf
3. Datenbankimplementierung

4. Physische Datenorganisation
5. Anfrageoptimierung
6. Transaktionsverwaltung

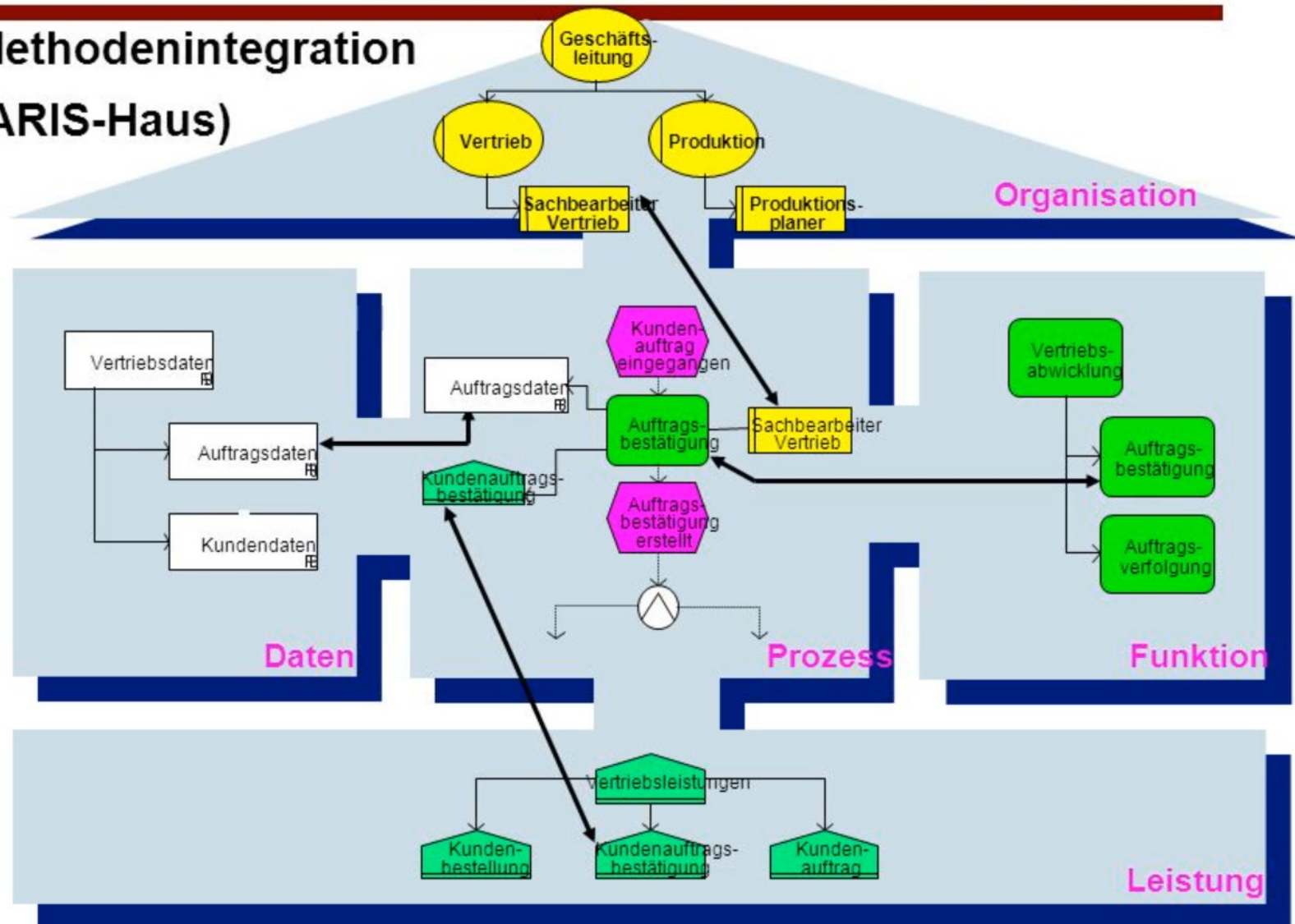
7. Datensicherheit und Wiederherstellung
8. Business Intelligence

Datenmodellierung

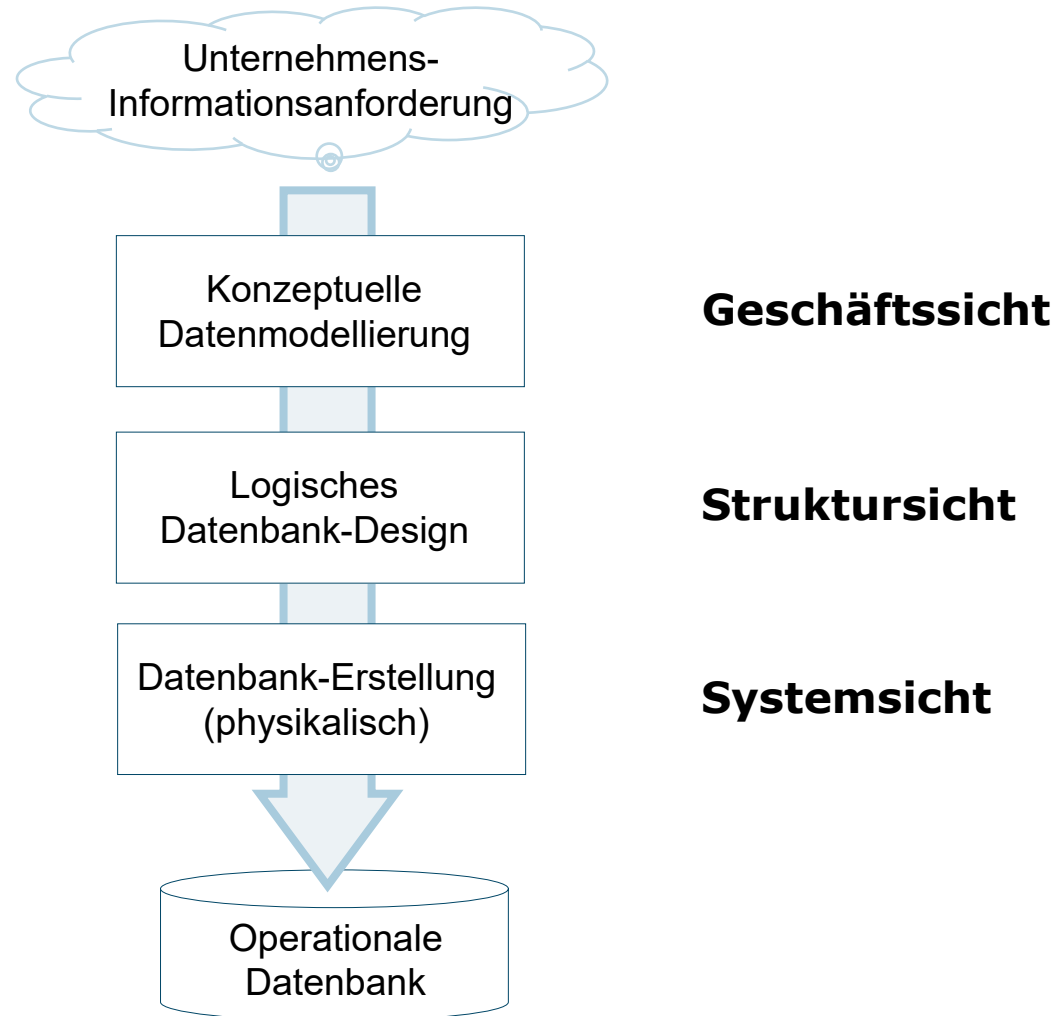


Datenmodellierung

Methodenintegration (ARIS-Haus)



Schrittweise Erstellung einer Datenbank



Allgemeine Abstraktionskonzepte

- ▶ Zur Modellierung von Realwelt-Zusammenhängen auf einer abstrakten Ebene.
- ▶ **Abstraktion** ist ein mentaler Prozess und dient zur Auswahl **relevanter** Eigenschaften zur **Beschreibung** von Objekten.
- ▶ Abstraktionsmechanismen:
 1. **Klassifikation**
(Klassendefinitionen durch Mengenbildung der Attribute),
 2. **Aggregation**
(neue Klassendefinitionen aus bestehenden Klassen durch Bildung kartesischer Produkte),
 3. **Generalisierung bzw. Spezialisierung**
(Teilmengenbeziehungen mit Vererbungseigenschaften)

Aspekte der Qualitätssicherung bei der Modellierung

Vollständigkeit

- ▶ Alle relevanten Eigenschaften und Aspekte des Anwendungsbereiches müssen erfasst sein.

Prüfung:

- Sind alle Anforderungen im DB-Schema repräsentiert?
- Werden alle vorkommenden Konzepte innerhalb des Schemas auch wirklich in der Anwendung verlangt?

Korrektheit

- ▶ Ist das Modell syntaktisch korrekt?

Minimalität

- ▶ Jeder Aspekt der Anforderung kommt nur einmal vor.
Ohne Informationsverlust kann kein Aspekt entfernt werden.
(Redundanzvermeidung vs. Denormalisierung!)

Lesbarkeit

- ▶ Verständliche Beschreibung der abzubildenden Eigenschaften und optisch klare Diagramme.
Selbsterklärend: Unversierte Leser müssen das Schema auch ohne Dokumentation verstehen können.

Modifizierbarkeit

- ▶ Dynamische Nutzeranforderungen und Anwendungsverschiebungen müssen durch flexible Schema-Modifikationen umgesetzt werden können.
(Modularer Aufbau!)

Gliederung

1.

Datenbankentwurf



Definition Datenbankentwurf



Datenmodellierung



Phasen des Datenbank-Entwurfsprozesses



Konzeptioneller Entwurf mit dem Entity-Relationship-Modell

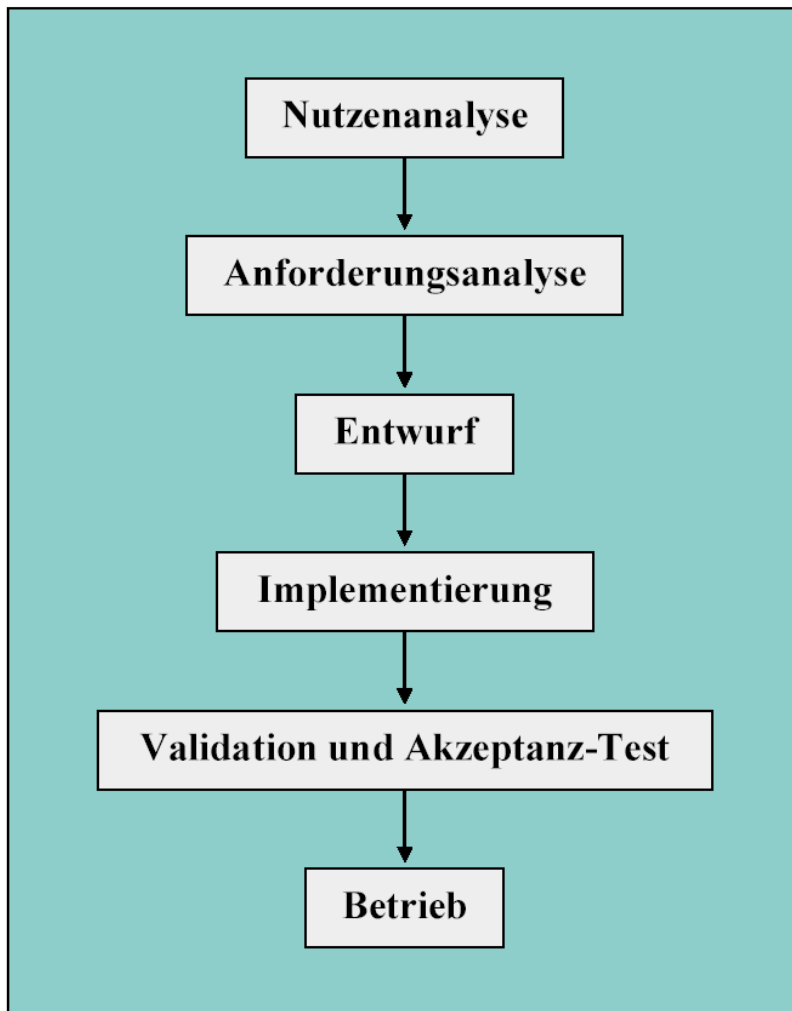


Logischer Entwurf mit dem Relationalen Datenmodell

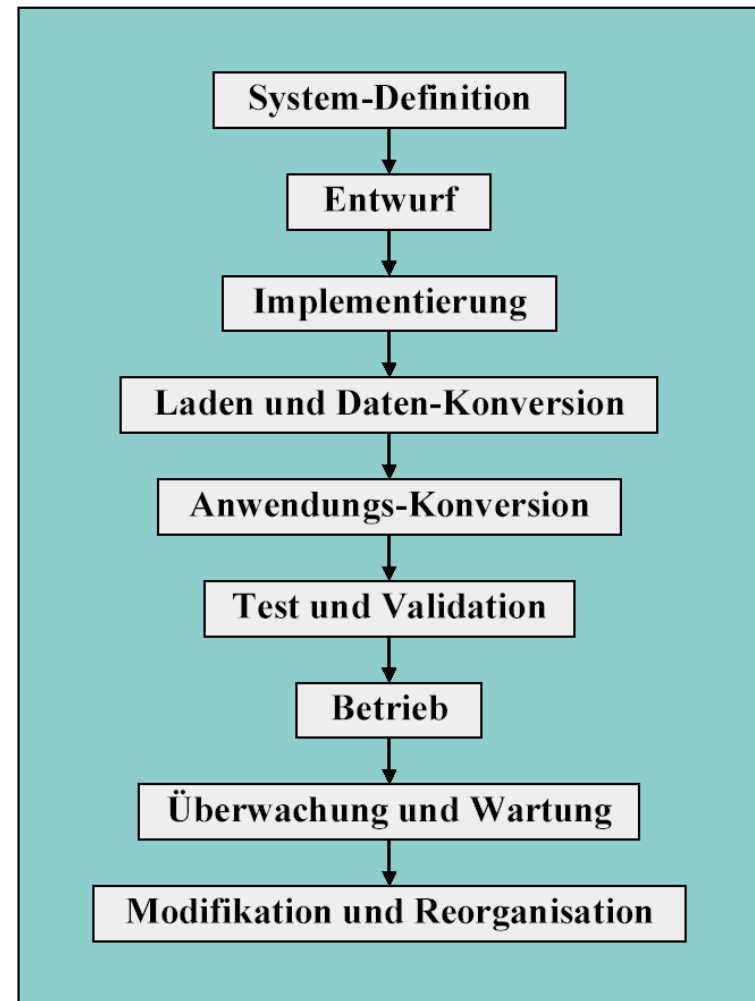


Transformation eines Entity-Relationship-Modells
in ein Relationales Datenmodell

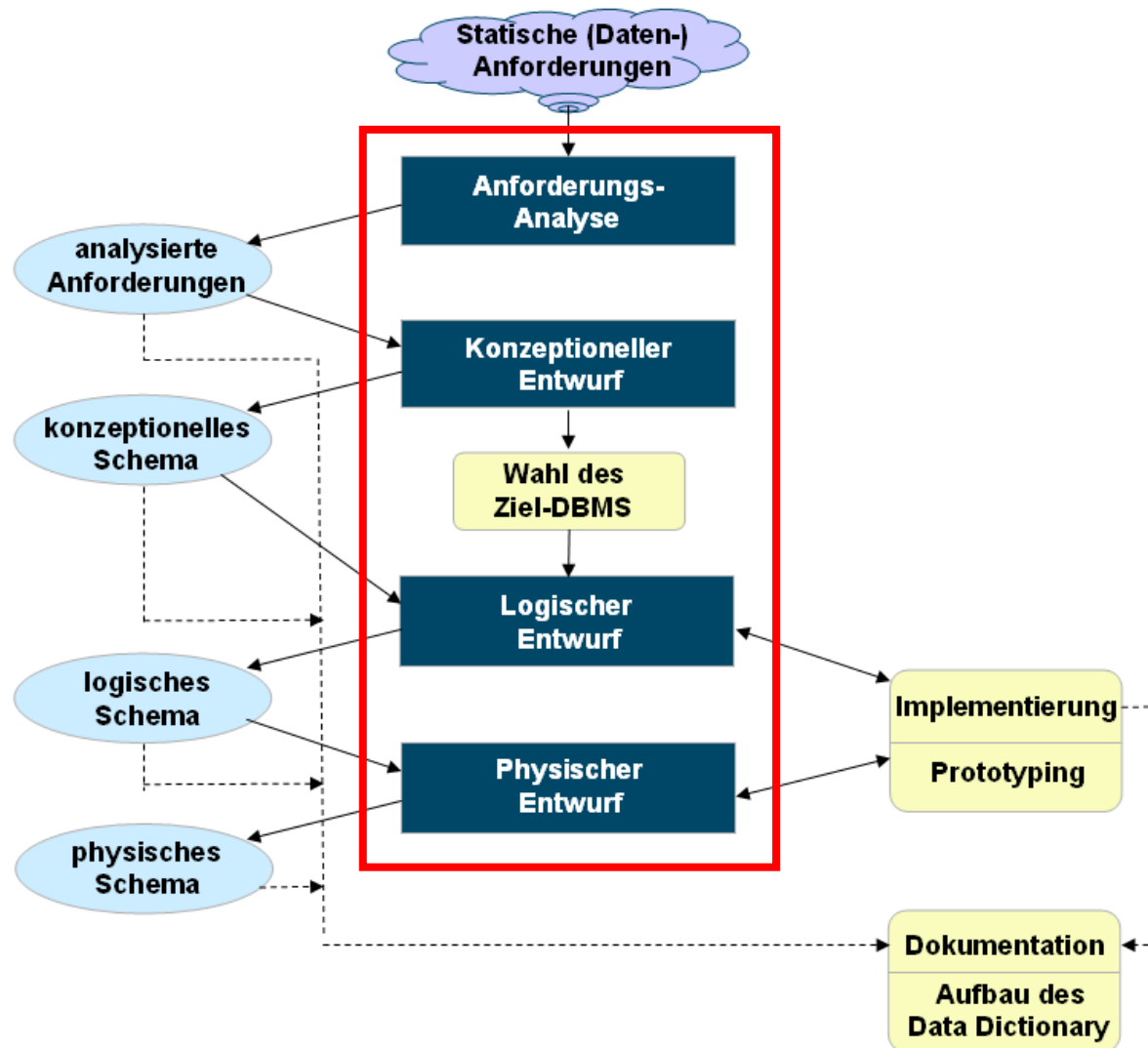
Lebenszyklus eines Informationssystems



Lebenszyklus einer Datenbankanwendung



Phasen des Entwurfsprozesses



Anforderungsanalyse (1)

Typische Aktivitäten

Vorgehensweise:

Sammlung des Informationsbedarfs

1. Identifikation der Benutzergruppen und Anwendungsbereiche der zu entwerfenden DB
2. Sichtung existierender Dokumentationen
Prozessbeschreibungen, interne Handbücher, Softwaremanuals, Organigramme, Stellenbeschreibungen, Schema-Beschreibungen bereits vorhandener DB
3. Fragebögen und Interviews
Fragen zu Datenstrukturen und Prioritäten der zu entwickelnden Applikationen

Ergebnisse:

Analyse des Datenbestandes

- Ergebnisse werden primär in natürlicher Sprache abgefasst
(Texte, tabellarische Aufstellungen, Formblätter, usw.)
- Trennen der Informationen über Daten (Datenanalyse) von den Informationen über Funktionen (Funktionsanalyse)

- erste verbale Beschreibung des Fachproblems -

Anforderungsanalyse (2)

Analyse des Datenbestandes

Informations- anforderungen:

Statische Informationen

Welche Daten sollen gespeichert werden?

- Daten,
deren Eigenschaften, Attribute und Abhängigkeiten,
Identifikatoren (Schlüssel) und Integritätsbedingungen für die
Konsistenzdefinition → Syntax und Semantik

Funktions- anforderungen:

Aktivitäten und Prozesse, die auf der DB ablaufen

Wie sollen diese Daten bearbeitet werden?

Welche Benutzer treten auf?

- Anfragen oder Änderungen
- Verfügbarkeitsaspekte
(Häufigkeit und Reihenfolge der Prozesse, Datenvolumen)
- Datenschutz und Datensicherheit
(Zugriffsrechte und Sicherungsmechanismen)

Konzeptioneller Entwurf

Erstellung einer konzeptionellen Globalsicht der Anwendung, unabhängig vom DB-System.

Input:

Anforderungsspezifikation

Durchführung:

Modellierung und Formalisierung.

Strategien:

- TOP-DOWN (schrittweise Verfeinerung) vs. BOTTOM-UP (schrittweise Verallgemeinerung).
- ZENTRALISIERT (globalsicht-orientiert) vs. DEZENTRALISIERT (einzelsicht-orientiert).

Output:

Konzeptionelles, zielsystem-unabhängiges Datenbankschema (semantisches Datenmodell)

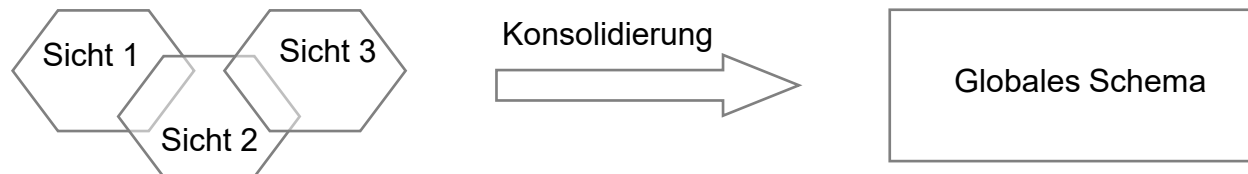
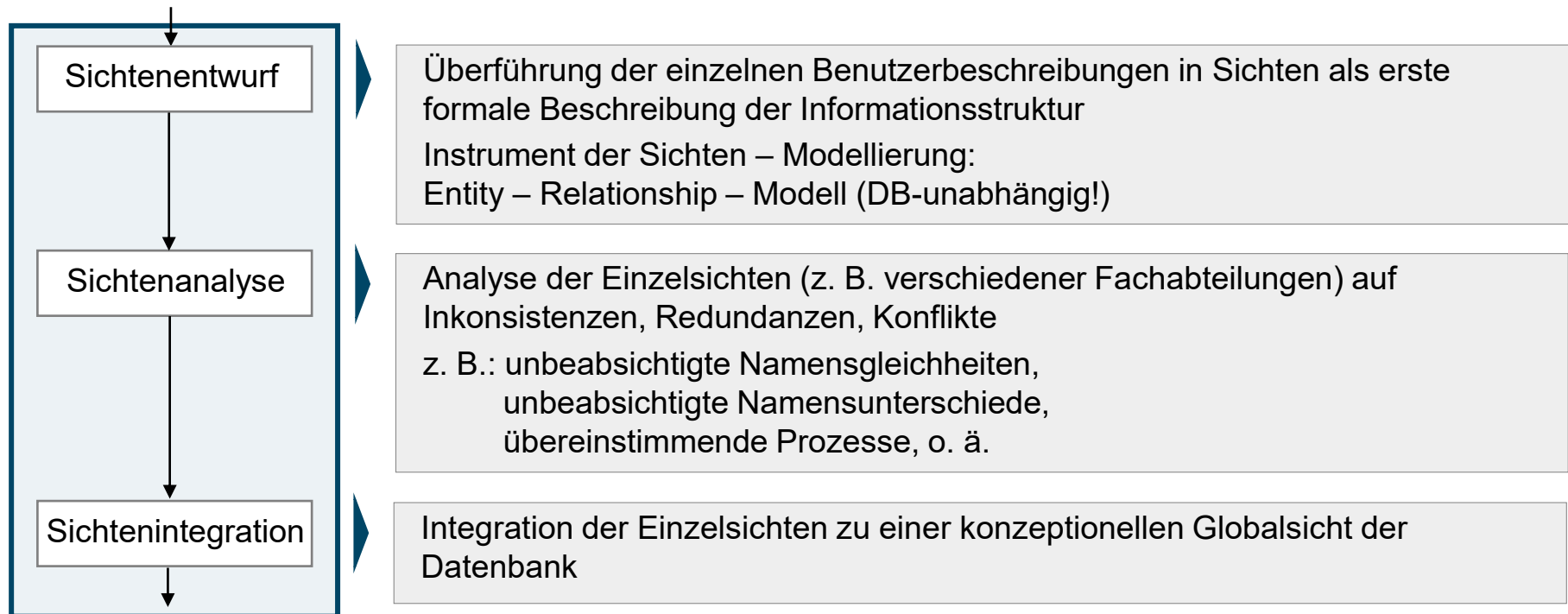
Instrument der Implementierung:

Entity-Relationship-Modell

- erste formale Beschreibung des Fachproblems -

Einzelsicht-orientiertes BOTTOM-UP - Vorgehen

Beispielvorgehen



Auswahl des Ziel – DBS

Nach Ende des konzeptionellen Entwurfs muss die Auswahl des zu verwendeten Datenbanksystems erfolgen. Hierbei sind folgende Aspekte zu berücksichtigen:

Technische Aspekte:

- Angemessenheit des zu verwendeten Datenmodells,
- Verfügbare Typen von Abfragesprachen,
- Benötigte Schnittstellen zu Programmiersprachen,
- Leistungsmessungen vergleichbarer Systeme (Geschwindigkeit laut benchmark – Tests)

Ökonomische und Organisatorische Aspekte:

- Herstellerbindungen des Unternehmens,
- Vorhandene Hard- und Software, eingesetzte Betriebssysteme, bisher im Betrieb genutzte DB-Systeme, Kosten für Neubeschaffungen,
- Beschaffungs- und Wartungskosten des DBMS,
- Kosten für Migration vom Altsystem,
- Kosten für zusätzlich benötigten Personals und für Schulungsmaßnahmen

Logischer Entwurf

Umsetzung (Transformation) des konzeptionellen Schemas in das logische Schema des eingesetzten DBMS.

Input:

Konzeptionelles, zielsystem-unabhängiges Datenbankschema
(Semantisches Datenmodell)

Durchführung:

Modellierung und Formalisierung.

Qualitätsprüfung und ggf. Qualitätsverbesserung

z. B.: bei relationalen DB-Systemen durch Normalisierungsschritte; Details später

Output:

Systemnahes, zielsystem-abhängiges Datenbankschema
(Relationales Datenmodell)

**Instrument der
Implementierung:**

Relationales Datenmodell

- systemnahe Beschreibung des Fachproblems -

Physischer Entwurf

Umsetzung (Transformation) des logischen Schemas in das konkrete Schema des eingesetzten DBMS.

Input:

Systemnahes, zielsystem-abhängiges Datenbankschema
(Relationales Datenmodell)

Durchführung:

Datendefinition im System und Customizing der Systemparameter.

Daten:

- ✓ Objekte des Modells und derer Beziehungen
- ✓ Integritätsbedingungen
- ✓ Benutzersichten

Systemparameter:

1. Verwendete Datei-Formate
(Record-Längen, Baum- oder Hash – Zugriffsverfahren)
2. Block- und Seitenzuweisungen auf Platte
3. Cluster-Bildung
(Optimierung der Zugriffe und Zugriffszeiten durch Verteilung der Datenobjekte)
4. Indexauswahl für Attribute oder Attributkombinationen
(Effizienter Datenzugriff)
5. Denormalisierung

Output:

Quellcode-Erzeugung im System

**Instrument der
Implementierung:**

SQL

- Implementierung des Fachproblems -

Gliederung

2.

Datenbankentwurf



Definition Datenbankentwurf



Datenmodellierung



Phasen des Datenbank-Entwurfsprozesses



Konzeptioneller Entwurf mit dem Entity-Relationship-Modell



Logischer Entwurf mit dem Relationalen Datenmodell



Transformation eines Entity-Relationship-Modells
in ein Relationales Datenmodell

Das Entity – Relationship – Modell als Entwurfswerkzeug

Entwickelt 1976 von Peter Chen

Weithin akzeptiertes Modellierungswerkzeug

- Hohe Bedeutung nicht nur in der Datenbankwelt, sondern auch für sonstige Modellierungsansätze in der Informatik

Ideal geeignet zur Umsetzung in relationale Datenbanksysteme

- Speicherung / Bearbeitung der Daten in Tabellenform

Unabhängig von konkreten Datenbanksystemen

Hohe Aussagekraft durch die Grundkonstrukte „Entity“ und „Relationship“ als natürliche und ausreichende Ausdrucksmittel

Unterstützt die drei Abstraktionskonzepte

- Klassifikation, Aggregation, Verallgemeinerung bzw. Spezialisierung

Erlaubt eine Darstellung konzeptioneller Entwürfe in einer grafischen Notation

Entities und Attribute

Entity



Ein Entity ist ein Objekt der realen oder der Vorstellungswelt, das existiert und von anderen unterscheidbar ist und über das Informationen zu speichern sind

Beispiele:

Person, Firma, Stadt, Auto, Artikel, Buch, Wein

aber auch Informationen über Ereignisse, wie z. B. Bestellung

Entity - Typ



Die Menge aller einander ähnlichen vergleichbaren Entities bildet einen Entity – Typ (Entity – Set). In ER-Diagrammen werden demnach **Entity-Typen modelliert**.

Beispiele:

Personen, Firmen, Städte, Autos, Artikel, Bücher

Darstellung: Name eines Entity-Sets (**Singular**) im Rechteck

Buch

Entities und Attribute

Attribut

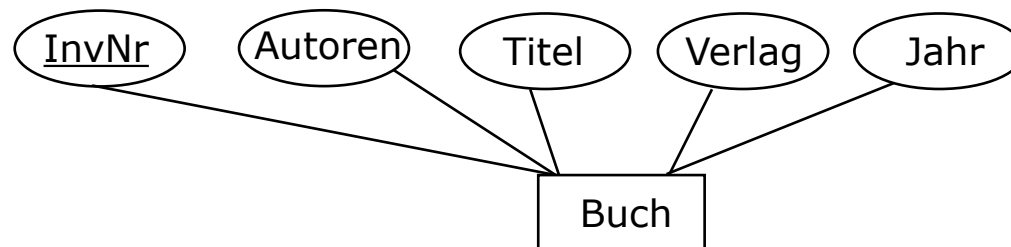
► Entities besitzen Eigenschaften (Merkmale). Ein Attribut repräsentiert eine Eigenschaft von einem Entity oder einer Beziehung. Alle Entities eines Entity-Typs haben dieselben Eigenschaften; **Attribute werden somit für Entity-Typen deklariert.**

Beispiele:

Farbe (eines Weines),
Datum (einer Bestellung),
Adresse (eines Kunden)

Darstellung: Dazugehörige Attribute als mit dem Rechteck (Entity-Typ) verbundene Kreise oder Ovale.

Entity-Diagramm für Buchinformationen:



Entities und Attribute

Wertebereich

▶ Die **Ausprägungen der Eigenschaften** werden als **Werte (Values)** bezeichnet.

Die Zusammenfassung aller möglichen bzw. zugelassenen Werte für eine Eigenschaft bezeichnet man als Wertebereich (Domain oder Value Set).

Wertebereiche sind beschrieben durch Datentypen, die neben einer Wertemenge auch die Grundoperationen auf diesen Werten charakterisieren.

Beispiele für Wertebereiche:

- Ganze Zahlen
- Reelle Zahlen
- Zeichenketten in einer vorgegebener (Maximal-) Länge
- Datumsformate
- Menge fest vorgegebener Werte (z.B. m, w bei Geschlecht)

Entities und Attribute

Ein konkretes Entity erhält man, indem man jedem Attribut einen Wert zuordnet.

Einwertige Attribute

Das Attribut in einem Entity nimmt genau einen Wert aus dem Wertebereich an.

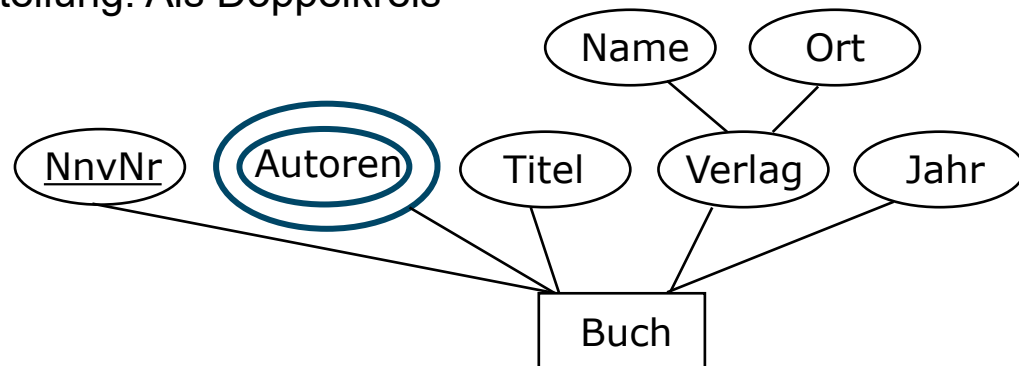
Beispiel: Geburtsdatum

Mehrwertige Attribute

Das Attribut in einem Entity kann einen oder mehrere Werte aus dem Wertebereich annehmen.

Beispiel: Mehrere Autoren zu einem Buch

Darstellung: Als Doppelkreis

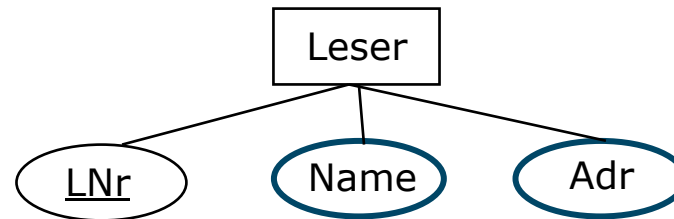


Entities und Attribute

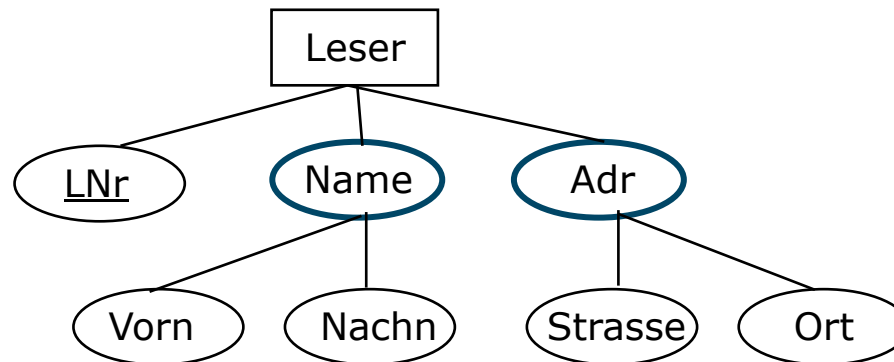
Zusammengesetzte Attribute

► Ein Attribut muss **unteilbar** sein.

Beispiel für zusammengesetzte Attribute:



Zusammengesetzte Attribute ausformulieren,
d. h. Attribute werden unteilbar gemacht.



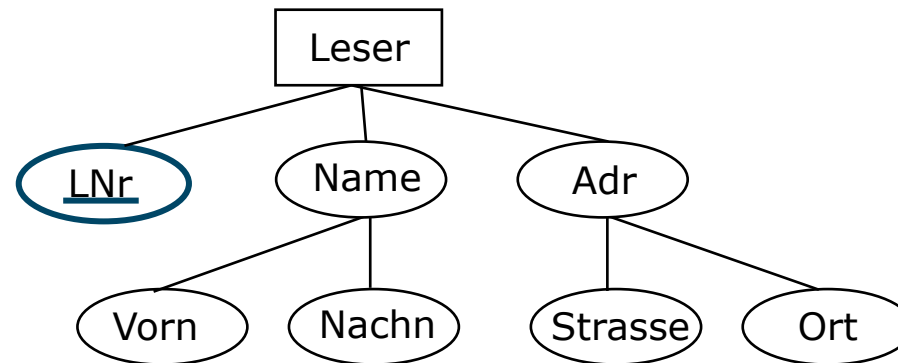
Entities und Attribute

Schlüsselattribute

► Attribute mit Identifizierungsfunktion / Schlüsselfelder (Keys).

Jedes Entity im Entity-Set muss über ein Schlüsselfeld eindeutig wiederfindbar sein.

Darstellung: Durch Untestreichung



Entities und Attribute

Schlüsselattribute

- ▶ Eine minimale Menge von einwertigen Attributen, deren Werte das zugeordnete Entity eindeutig innerhalb aller Entities eines Typs identifiziert, werden als Schlüssel bezeichnet (z.B. Inventarnummer, Serien-Nr.).

Oft werden einzelne Attribute als Schlüssel „künstlich“ eingebaut (z.B. Personalnummer).

- ▶ Manchmal gibt es mehrere Schlüsselkandidaten: Dann wird einer dieser Kandidaten-Schlüssel als Primärschlüssel ausgewählt und die weiteren werden als Sekundärschlüssel bezeichnet.
- ▶ Redundanzfreiheit: Wenn K_1 und K_2 beide jeweils ein Entity im Sinne eines Schlüssels identifizieren können und für die gilt $K_1 \supset K_2$, dann nennt man K_2 einen Superkey (Obermenge eines Schlüssels).
- ▶ Ein Schlüssel K ist stets eine minimale, identifizierende Attributkombination.

' Entity – Typ ' – Deklaration

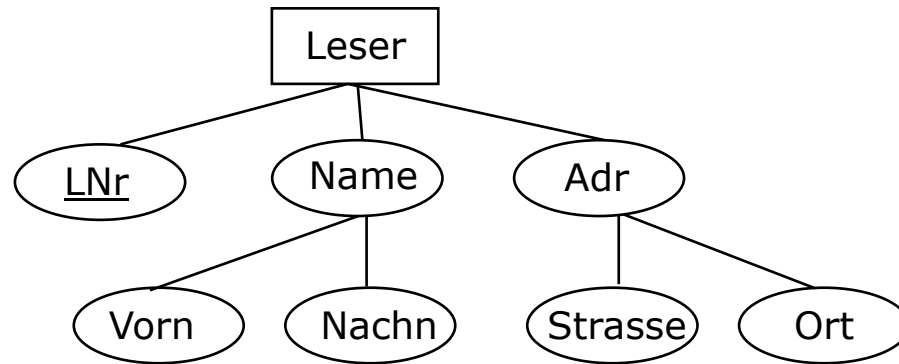
Definition:

- ▶ Eine **Entity-Deklaration** hat die **Form $E = (X, K)$** ;
sie besteht aus einem Namen E ,
einem Format X und
einem Primärschlüssel K , welcher aus (einwertigen) Elementen von X zusammengesetzt ist.
- ▶ **$\text{attr}(E)$** bezeichnet die Menge aller in X vorkommenden Attributnamen,
mit jedem $A \in X$ sei eine Wertemenge **$W(A)$** assoziiert.
- ▶ Wir bezeichnen diese Deklaration sprachlich als „**Entity-Typ**“.
- ▶ Die Elemente eines Formats X werden dabei wie folgt notiert:

Attribut	Notation	Notation unter Angabe der Domains (Langbezeichnung)	Notation unter Angabe der Domains (Kurzbezeichnung)
Einwertiges Attribut	A	„ Titel : char(30) “	$\text{dom}(A) := \begin{cases} W(A) \\ 2^{W(A)} \\ W(B_1) \times \dots \times W(B_k) \end{cases}$
Mehrwertiges Attribut	$\{A\}$	$\{\text{Autor}\} : \{\text{char}(20)\}$	
Zusammengesetztes Attribut als Kartesisches Produkt	$A(B_1, \dots, B_k)$	Adresse(Strasse, Ort) : (char(20), char(15))	

' Entity – Typ ' – Deklaration

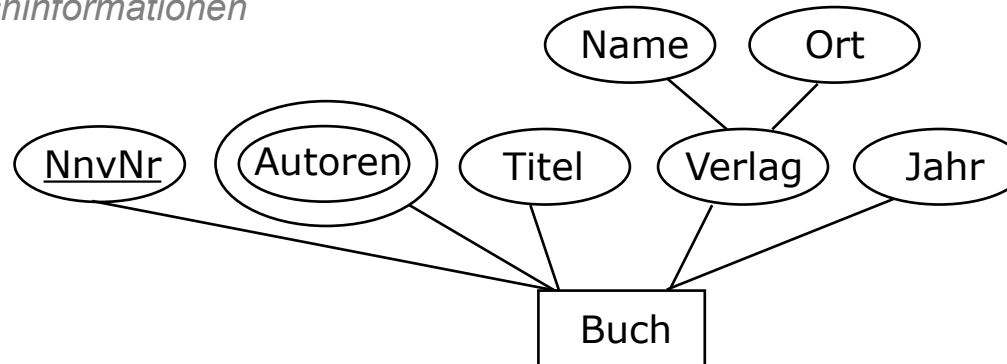
Entity-Diagramm für Leserinformationen



Entity-Typ Leser

Leser=({LNr,Name(Vorname,Nachname),Adresse(Strasse,Ort)},{LNr})

Entity-Diagramm für Buchinformationen



Entity-Typ Buch

Buch=({InvNr,{Autor},Titel,Verlag(Name,Ort),Jahr},{InvNr})

' Entity – Typ ' – Deklaration

Entity-Typ Buch

$\text{Buch} = (\{\text{InvNr}, \{\text{Autor}\}, \text{Titel}, \text{Verlag}(\text{Name}, \text{Ort}), \text{Jahr}\}, \{\text{InvNr}\})$

Zu einem Zeitpunkt t könnte das Entity-Set $\text{Buch}^t = \{b_1, b_2, b_3\}$ vorliegen mit

$b_1 = (123, \{\text{'Vossen'}, \text{'Witt'}\}, \text{'DB2 Handbuch'}, (\text{'Adisson-Wesley'}, \text{'Bonn'}), 1990)$

$b_2 = (125, \{\text{'Vossen'}, \text{'Witt'}\}, \text{'SQL/DS Handbuch'}, (\text{'Adisson-Wesley'}, \text{'Bonn'}), 1988)$

$b_3 = (130, \{\text{'Witt'}\}, \text{'OO Programmierung'}, (\text{'Oldenburg'}, \text{'München'}), 1992)$

Man beachte, dass in Buch^t das Entity

$b_4 = (123, \{\text{'Vossen'}\}, \text{'Transaktionsverarbeitung'}, (\text{'Hüthing'}, \text{'Heidelberg'}), 1990)$

nicht vorkommen darf, da es den Schlüssel **InvNr** verletzt.

' Relationship – Typ ' – Deklaration

Relationship

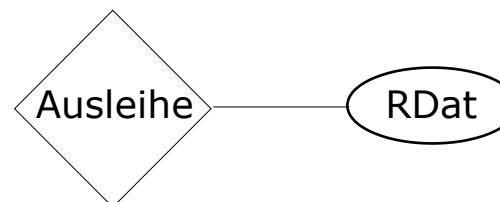
- ▶ Entities verschiedener Entity-Typen stehen miteinander in Beziehung. Bücher werden bspw. von Lesern entliehen, d. h. ein Buch steht in Beziehung (dem **Entleihvorgang bzw. Ausleihe**) zu einem Leser. Diese Beziehungen (Vorgänge) können wiederum eigene (einwertige, mehrwertige oder zusammengesetzte) **Attribute** besitzen.

Relationship - Typ

- ▶ Beziehungen zwischen den Entities verschiedener Entity-Typen werden zu „Relationship-Typen“ zusammengefasst.

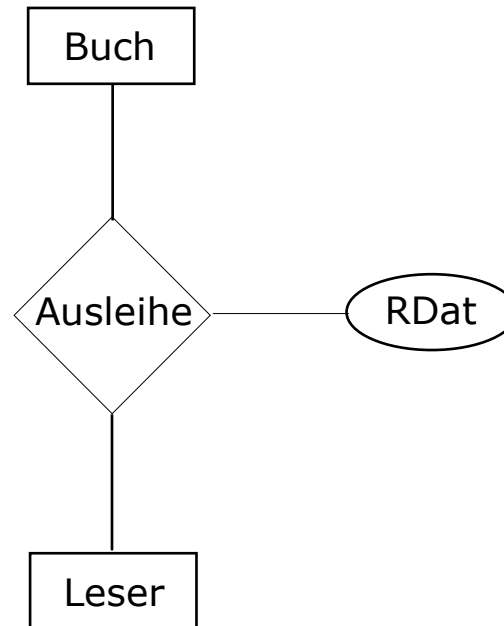
Auch bei den Beziehungen unterscheidet man **zeitinvariante** Beschreibungen und **zeitveränderliche** Inhalte.

Darstellung: Als Raute mit dem Namen der Relationship-Typ Deklaration verbunden durch (ungerichtete) Kanten zu Entity-Typen, ggf. Kreise mit eigenen Attributen



' Relationship – Typ ' – Deklaration

Entity-Relationship-Diagramm für Beziehung Ausleihe



Relationship-Typ Ausleihe

Ausleihe = ((Buch, Leser), (RDat))

' Relationship – Typ ' – Deklaration

Definition:

(i) Eine Relationship-Deklaration hat die Form:

$$R = (\text{Ent}, Y)$$

- Dabei ist **R** der Name der Deklaration
(in ungenauer Sprechweise auch der „Name der Beziehung“),
- **Ent** bezeichnet die Folge der Namen der Entity-Deklarationen,
zwischen denen eine Beziehung definiert werden soll,
- und **Y** ist eine (möglicherweise leere) Folge von Attributen (der Beziehung).

(ii) Sei $\text{Ent} = (E_1, \dots, E_k)$, und für beliebiges, aber festes t (zur Zeit t) sei E_i^t der Inhalt der Entity-Deklaration E_i ,
 $1 \leq i \leq k$.

Ferner sei $Y = (B_1, \dots, B_n)$.

Ein Relationship r ist ein Element des Kartesischen Produktes aus allen E_i^t und den Domains der B_j , d. h.

$$r \in E_1^t \times \dots \times E_k^t \times \text{dom}(B_1) \times \dots \times \text{dom}(B_n)$$

bzw.

$$r = (e_1, \dots, e_k, b_1, \dots, b_n)$$

mit

$$e_i \in E_i^t \text{ für } 1 \leq i \leq k \text{ und } b_j \in \text{dom}(B_j) \text{ für } 1 \leq j \leq n.$$

Die Domains der Attribute sind dabei wie im letzten Abschnitt definiert.

(iii) Ein Relationship-Set R^t (zur Zeit t) ist eine Menge von Relationships, d. h.

$$R^t \in E_1^t \times \dots \times E_k^t \times \text{dom}(B_1) \times \dots \times \text{dom}(B_n).$$

' Relationship – Typ ' – Deklaration

Relationship-Typ Ausleihe

Ausleihe = ((Buch, Leser), (RDat))

Zu einem Zeitpunkt t könnte das Entity-Set **Buch^t = {b₁, b₂, b₃}** vorliegen mit

b₁ = (123, {'Vossen', 'Witt'}, 'DB2 Handbuch', ('Adisson-Wesley', 'Bonn'), 1990)

b₂ = (125, {'Vossen', 'Witt'}, 'SQL/DS Handbuch', ('Adisson-Wesley', 'Bonn'), 1988)

b₃ = (130, {'Witt'}, 'OO Programmierung', ('Oldenburg', 'München'), 1992)

Zu einem Zeitpunkt t könnte das Entity-Set **Leser^t = {l₁, l₂}** vorliegen mit

l₁ = (500, {'Peter', 'Müller'}, ('Waldstrasse', 'Köln'))

l₂ = (550, {'Franz', 'Meier'}, ('Feldweg', 'Bonn'))

Damit lässt sich das Relationship-Set **Ausleihe^t = {a₁, a₂}** bilden mit

a₁ = (123, {'Vossen', 'Witt'}, 'DB2 Handbuch', ('Adisson-Wesley', 'Bonn'), 1990, 500, {'Peter', 'Müller'}, ('Waldstrasse', 'Köln'), 31-07-08)

a₂ = (130, {'Witt'}, 'OO Programmierung', ('Oldenburg', 'München'), 1992, 550, {'Franz', 'Meier'}, ('Feldweg', 'Bonn'), 30-06-07)

' Relationship – Typ ' – Deklaration

Vereinfachungen in der Notation:



Unter der Voraussetzung der Deklaration von (Primär-) Schlüsseln reicht die **Angabe des Schlüsselwertes** zur Identifikation der Entities.

Bibliotheksanwendung:

Relationship-Typ Ausleihe

Ausleihe = ((Buch, Leser), (RDat))

Entities des Relationship-Sets **Ausleihe^t = {a₁, a₂}** mit folgenden Werten

a₁ = (123, {'Vossen', 'Witt'}, 'DB2 Handbuch', ('Adisson-Wesley', 'Bonn'), 1990, 500, {'Peter', 'Müller'}, ('waldstrasse', 'Köln'), 31-07-08)

a₂ = (130, {'Witt'}, 'OO Programmierung', ('Oldenburg', 'München'), 1992, 550, {'Franz', 'Meier'}, ('Feldweg', 'Bonn'), 30-06-07)

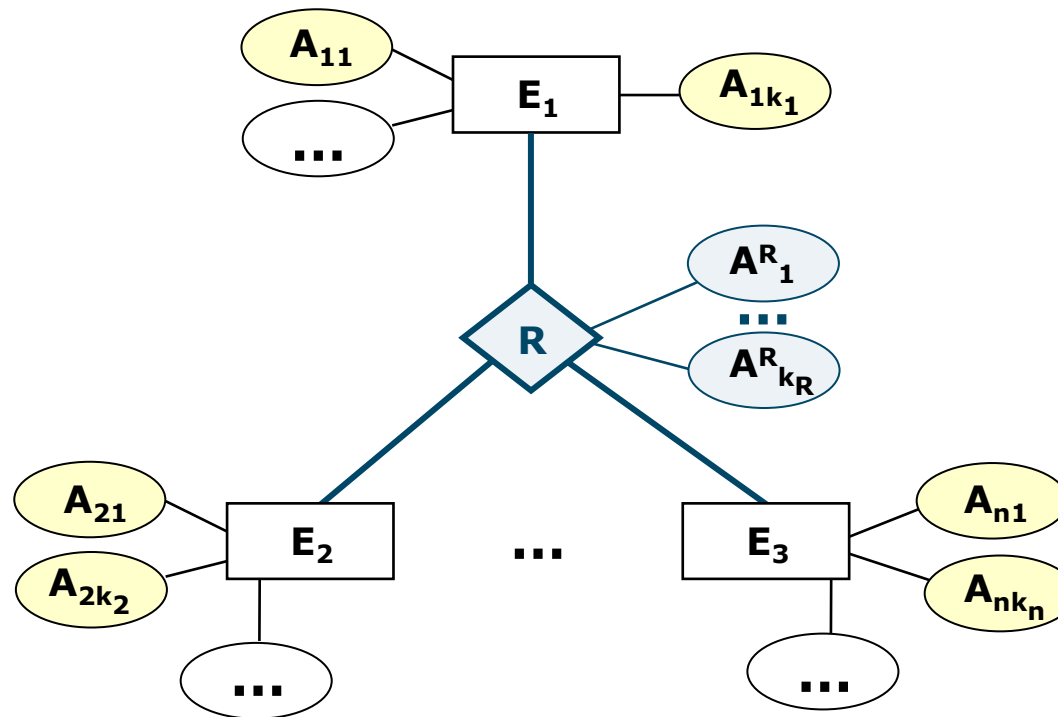


a₁ = (123, 500, 31-07-08)

a₂ = (130, 550, 30-06-07)

' Relationship – Typ ' – Deklaration

Relationale Darstellung einer Attributdeklaration bei einem Beziehungstyp



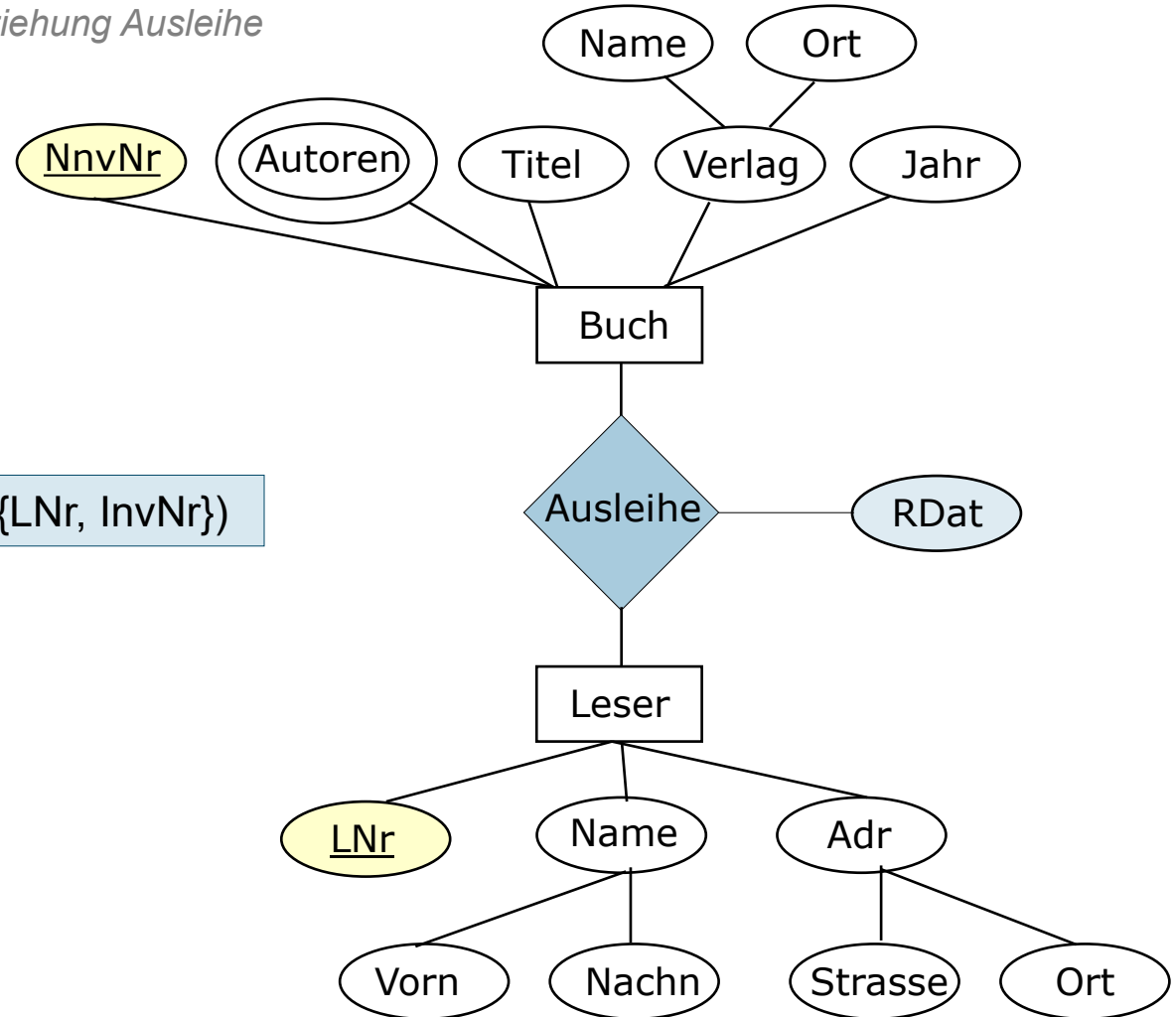
Textuelle Notation einer Attributdeklaration bei einem Beziehungstyp

Relationship-Typ R

$$R = (\underbrace{\{A_{11}, \dots, A_{1k_1}\}}_{\text{Schlüssel von } E_1}, \underbrace{\{A_{21}, \dots, A_{2k_2}\}}_{\text{Schlüssel von } E_2}, \dots, \underbrace{\{A_{n1}, \dots, A_{nk_n}\}}_{\text{Schlüssel von } E_n}, \underbrace{\{A^R_1, \dots, A^R_{k_R}\}}_{\text{Attribute von } R})$$

' Relationship – Typ ' – Deklaration

Entity-Relationship-Diagramm für Beziehung Ausleihe

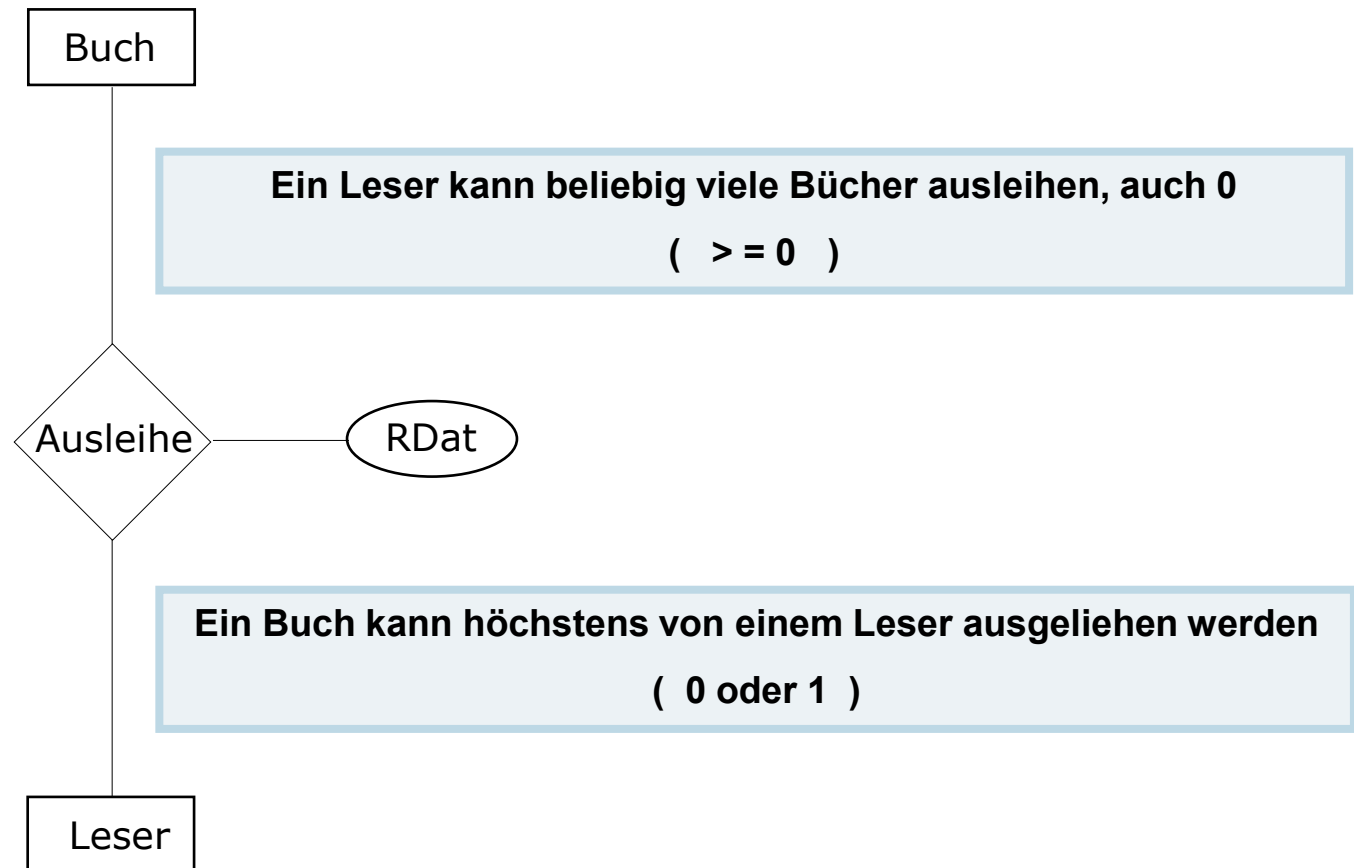


Relationship-Typ Ausleihe

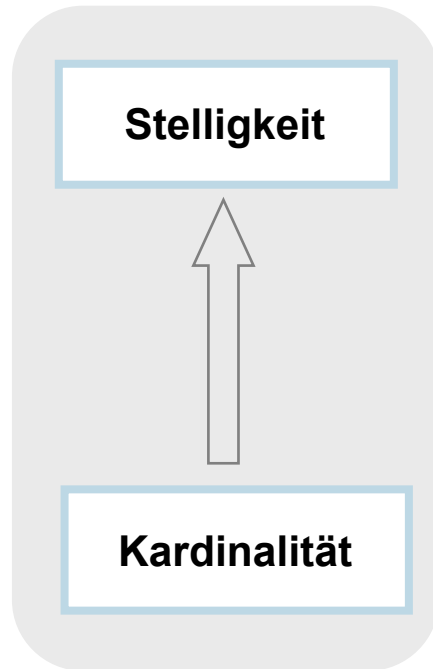
Ausleihe = ({LNr, InvNr, RDat}, {LNr, InvNr})

Merkmale einer ' Relationship – Typ ' – Deklaration

Entity-Relationship-Diagramm für Beziehung Ausleihe



Merkmale einer ' Relationship – Typ ' – Deklaration



Stelligkeit oder Grad:

- Anzahl der an einem Beziehungs-Typ beteiligten Entity-Typen
- häufig: zweistellig
- Beispiel: Lieferant liefert Produkt

Kardinalität oder Komplexität:

- Anzahl der in eine Beziehung eingehender Instanzen eines Entity-Typs (Einschränkung der Teilnahme von Entity-Typ - Instanzen an Beziehung)
- Formen: 1 – 1, 1 – n, n – m
- Stellt Integritätsbedingung dar
- Beispiel: maximal 5 Produkte pro Bestellung

Merkmale einer ' Relationship – Typ ' – Deklaration

Sei $R = (Ent, Y)$ eine Relationship – Deklaration mit $Ent = (E_1, \dots, E_k)$.

Dann heißt k die Stelligkeit oder der Grad von R , kurz k oder $\text{grad}(R)$

Zweistellige Beziehungen ($k=2$) kommen am häufigsten vor!

Einer Relationship-Deklaration wird dabei eine **Komplexität** (Kardinalität) zugeordnet, welche folgende Angabe ermöglicht:

Mit wie vielen Entities des ersten Typs kann, darf oder sogar muss ein bestimmtes Entity des zweiten Typs in einer konkreten Beziehung stehen.

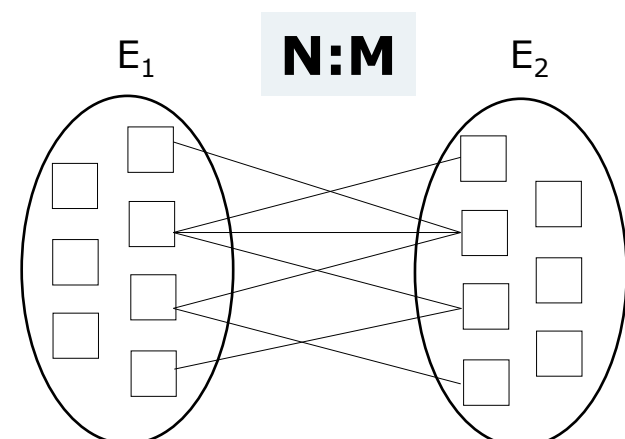
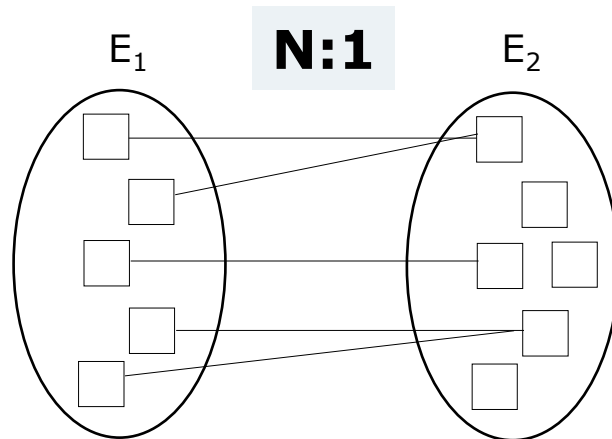
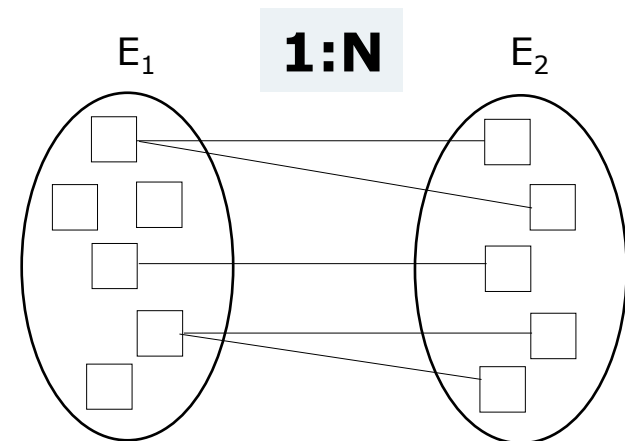
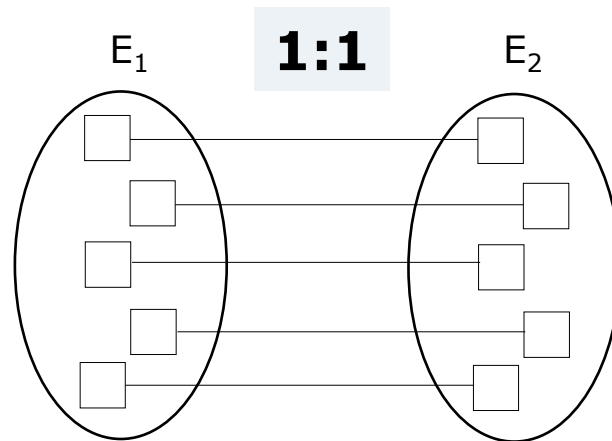
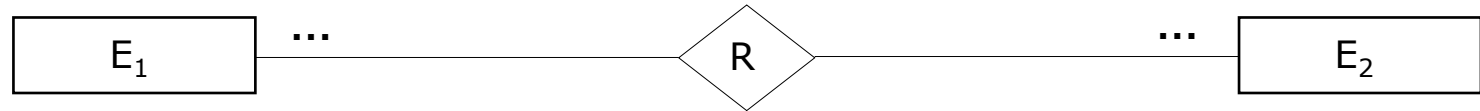
Ist R eine Relationship – Deklaration mit $Ent = (E_1, \dots, E_k)$, so tritt ein spezielles Entity $e_i \in E_i$ (zum Zeitpunkt t) in einem Relationship-Set R^t mindestens m -mal, aber höchstens n -mal auf; dies wird durch die Schreibweise

„ $\text{comp}(R, E_i) = (m, n)$ “ („complexity“ von R bzgl. E_i)

zum Ausdruck gebracht. Die formale Definition der Komplexität einer Beziehung lautet:

$$\text{comp}(R, E_i) = (m, n) : \iff (\forall t)(\forall e_i \in E_i^t) \mid \{r \in R^t \mid r[E_i] = e_i\} \begin{cases} \geq m \\ \leq n \end{cases}$$

Kardinalitäten einer 2-stelligen ' Relationship – Typ ' – Deklaration



Kardinalitäten einer 2-stelligen ' Relationship – Typ ' – Deklaration

Die verschiedenen Beziehungstypen unterschiedlicher Komplexitäten lassen sich am besten an einem konkreten Beispiel darstellen:



Ausgehend von dem Entity-Set Person ergeben sich die folgenden Möglichkeiten, im Fachausdruck **Assoziationstypen**:

Eine Person hat **genau ein** Auto

in Zahlen: **1**

Eine Person hat **höchstens ein** Auto

in Zahlen: **0 oder 1**

Eine Person hat **mindestens ein** Auto

in Zahlen: **> = 1**

Eine Person hat **beliebig viele** Autos

in Zahlen: **> = 0**

Analog müssen natürlich auch die Assoziationstypen ausgehend von Entity-Set Auto betrachtet werden – hierbei wird untersucht die Anzahl der Besitzer eines Autos.

Erst die Betrachtung der beiden Assoziationen ergibt die Beziehung zwischen den beiden Entity-Sets.

Kardinalitäten einer N-stelligen ' Relationship – Typ ' – Deklaration

► Möglichkeit zur Darstellung von Kardinalitäten

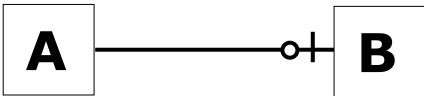
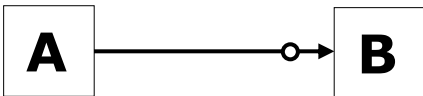
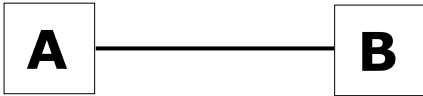
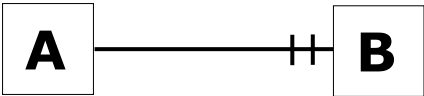
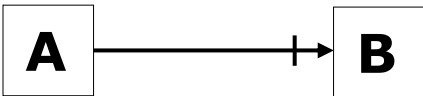
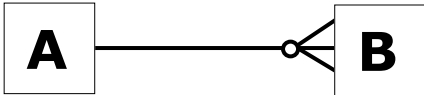
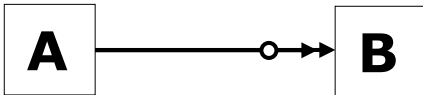
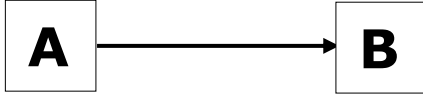
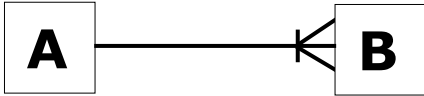
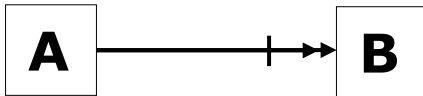
Abkürzung	Assoziationstyp	Anzahl Entities in Entity-Set Auto (in Worten)	Anzahl Entities in Entity-Set Auto (in Zahlen)
1	einfache Assoziation	genau ein Entity	1
c	konditionelle Assoziation	höchstens ein Entity	1 oder 0
m	multiple Assoziation	mindestens ein Entity	> = 1
mc	multipel-konditionelle Assoziation	beliebig viele Entities, auch 0	> = 0

► Für zweistellige Beziehungen lässt sich die Kombination der möglichen Beziehungstypen in folgende Kategorien einteilen:

E₂ \ E₁	1	c	m	mc	Kategorie
1	1 - 1	c - 1	m - 1	mc - 1	hierarchisch
c	1 - c	c - c	m - c	mc - c	konditionell
m	1 - m	c - m	m - m	mc - m	netzwerkförmig
mc	1 - mc	c - mc	m - mc	mc - mc	

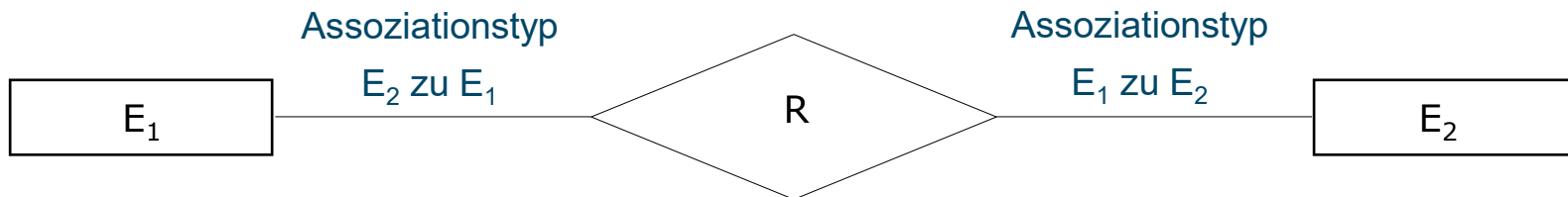
Kardinalitäten einer N-stelligen ' Relationship – Typ ' – Deklaration

Alternative Möglichkeit zur Darstellung von Kardinalitäten

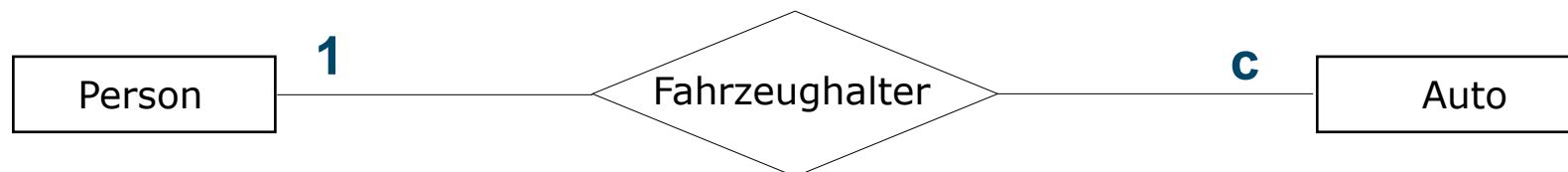
Numerische Notation	Krähenfußnotation	Pfeilnotation	Bachmannnotation
(0,1)			
(1,1)			
(n,0)			
(n,1)			

Kardinalitäten einer N-stelligen ' Relationship – Typ ' – Deklaration

Kardinalitäten einer Beziehung werden an den Kanten der jeweiligen Entity-Typen wie folgt dargestellt:



Der Assoziationstyp wird als ein Wert aus dem Wertebereich 1, c, m und mc angegeben, davor müsste eigentlich immer noch „1 zu“ stehen!



Eine Person hat höchstens ein Auto
Ein Auto hat genau einen Besitzer



Formal: Der Assoziationstyp zwischen Auto und Person steht links und bedeutet „(1 zu) 1“
Der Assoziationstyp zwischen Person und Auto steht rechts und bedeutet „(1 zu) c“

IS – A Beziehungen

► Spezialisierungs-/Generalisierungsbeziehung oder auch IST-Beziehung (engl. is-a relationship)

Für Beziehung E_1 IST E_2 gilt immer: $IST(E_1 [1,1], E_2 [0,1])$

Jede Instanz von E_1 nimmt genau einmal an der IST-Beziehung teil,
während Instanzen des Entity-Typs E_2 nicht teilnehmen müssen

Beispiel: Angestellte einer Fluggesellschaft

Entity-Typ Angestellter

Angestellter=((AngNr,Name,Adresse,Beruf,Gehalt),(AngNr))

Weitere Attribute gelten nur für bestimmte Angestellte der Gesellschaft:

Entity-Typ Pilot

Pilot=((AngNr,Std,Liz),(AngNr))

Entity-Typ Techniker

Techniker=((AngNr,TeamNr),(AngNr))

Die Attribute für Angestellter haben auch Gültigkeit für Pilot und Techniker!

So werden alle Attribute von Angestellter an dessen Spezialisierungen **vererbt**!

IS – A Beziehungen

Definition:

Sind $E_1 = (X_1, K_1)$ und $E_2 = (X_2, K_2)$ zwei Entity-Deklarationen, so besteht zwischen diesen eine IS-A-Beziehung (der Form E_1 IS-A E_2), falls gilt:

- (i) Alle Elemente von X_2 kommen in X_1 vor;
- (ii) Zu jedem Zeitpunkt t gilt:
Für jedes $e_1 \in E_1^t$ existiert ein $e_2 \in E_2^t$ mit $e_1(A) = e_2(A)$ für jedes Attribut $A \in X_2$.

Schreibweise kurz $E_1 \in E_2$.

Die Bedingung „Alle X_2 kommen in X_1 vor“ implizit:

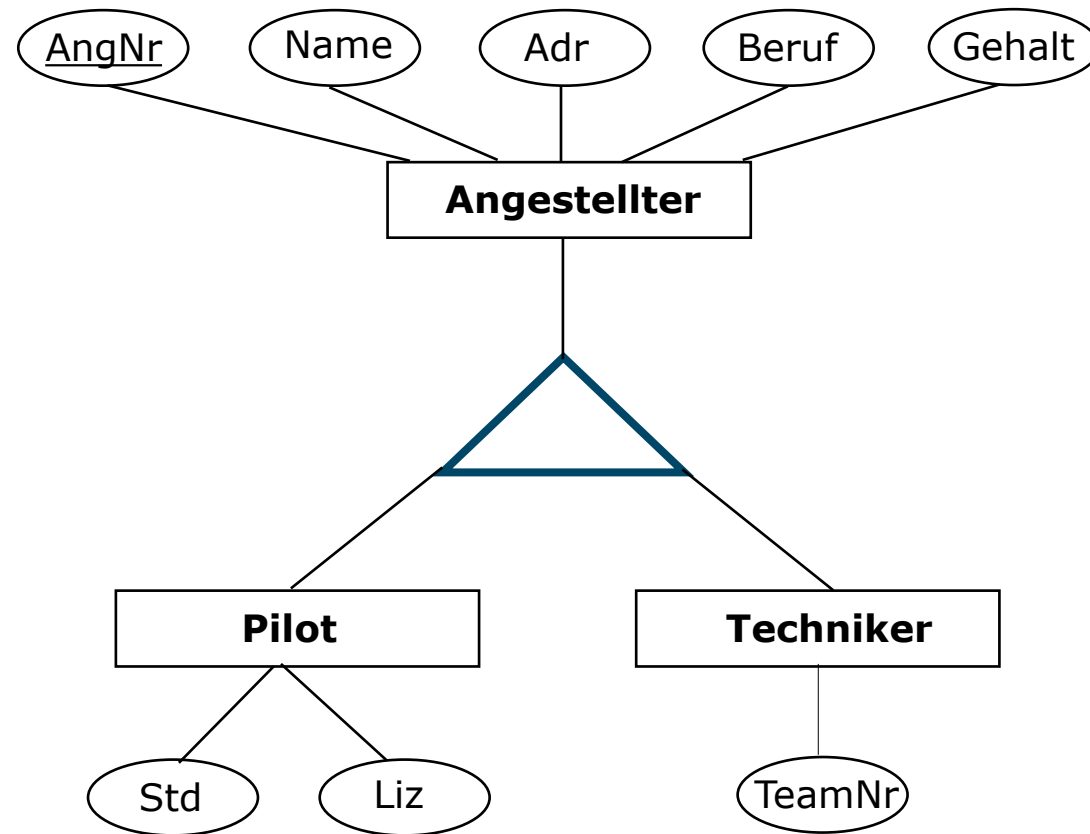
- (1) Es gilt $K_1 = K_2$; dies ist der bei weitem häufigste Fall.
- (2) K_2 ist eine Teilfolge von K_1 ; in diesem Fall gibt es für die Spezialisierung E_1 neue Attribute, welche zur Identifikation eines Entities benötigt werden.
- (3) K_1 und K_2 besitzen keine gemeinsamen Attribute; in diesem Fall sichert die Schlüsseleigenschaft von K_1 , dass dieser auch K_2 und damit ein E_2 -Entity identifizieren kann.

IS – A Beziehungen

Darstellung:

Dreiecke, die auf die Generalisierung zeigen.

Verbindungen zu den Spezialisierungen mittels Kanten.



Spezialisierungen der IS-A Beziehungen

Definition:

Es seien E, E_1, \dots, E_k Entity-Deklarationen, $k \geq 2$, und es gelte, dass alle E_i , $1 \leq i \leq k$ zu derselben Spezialisierung von E gehören (also insbesondere E_i IS-A E für $1 \leq i \leq k$). Die IS-A-Beziehung heißt

(i) *total*, falls gilt: $(\forall t) \bigcup_{i=1}^k E_i^t = E^t$

(ii) *disjunkt*, falls gilt: $(\forall t) (\forall i, j \in \{1, \dots, k\}, i \neq j) E_i^t \cap E_j^t = \emptyset$

Spezialisierungen der IS-A Beziehungen

Spezialformen der IS – A Beziehung

total

Alle Entities der Verallgemeinerung gehören zu einer Spezialisierung

Darstellung: **t** im Dreieck der IS-A-Beziehung

Beispiel:

Verallgemeinerung: Person

Spezialisierung: Mann, Frau, Divers

partiell

Es gibt Entities der Verallgemeinerung, die nicht zu einer der vorgegebenen Spezialisierungen gehören

Darstellung: **p** im Dreieck der IS-A-Beziehung

Beispiel:

Verallgemeinerung: Angestellter

Spezialisierung: Pilot, Techniker, usw.

Es gibt Angestellte, die weder Piloten, noch Techniker sind.

Spezialisierungen der IS-A Beziehungen

Spezialformen der IS – A Beziehung

disjunkt

Es gibt keine Entities, die zu mehreren Spezialisierung gehören

Darstellung: Pfeile ausgehend von Verallgemeinerungs-Entity-Typ

Beispiel:

Verallgemeinerung: Fahrzeug

Spezialisierung: Auto **oder** Fahrrad

nicht disjunkt

Es gibt Entities, die zu mehreren Spezialisierungen gehören

Darstellung: Pfeile ausgehend von Spezialisierungs-Entity-Typen

Beispiel:

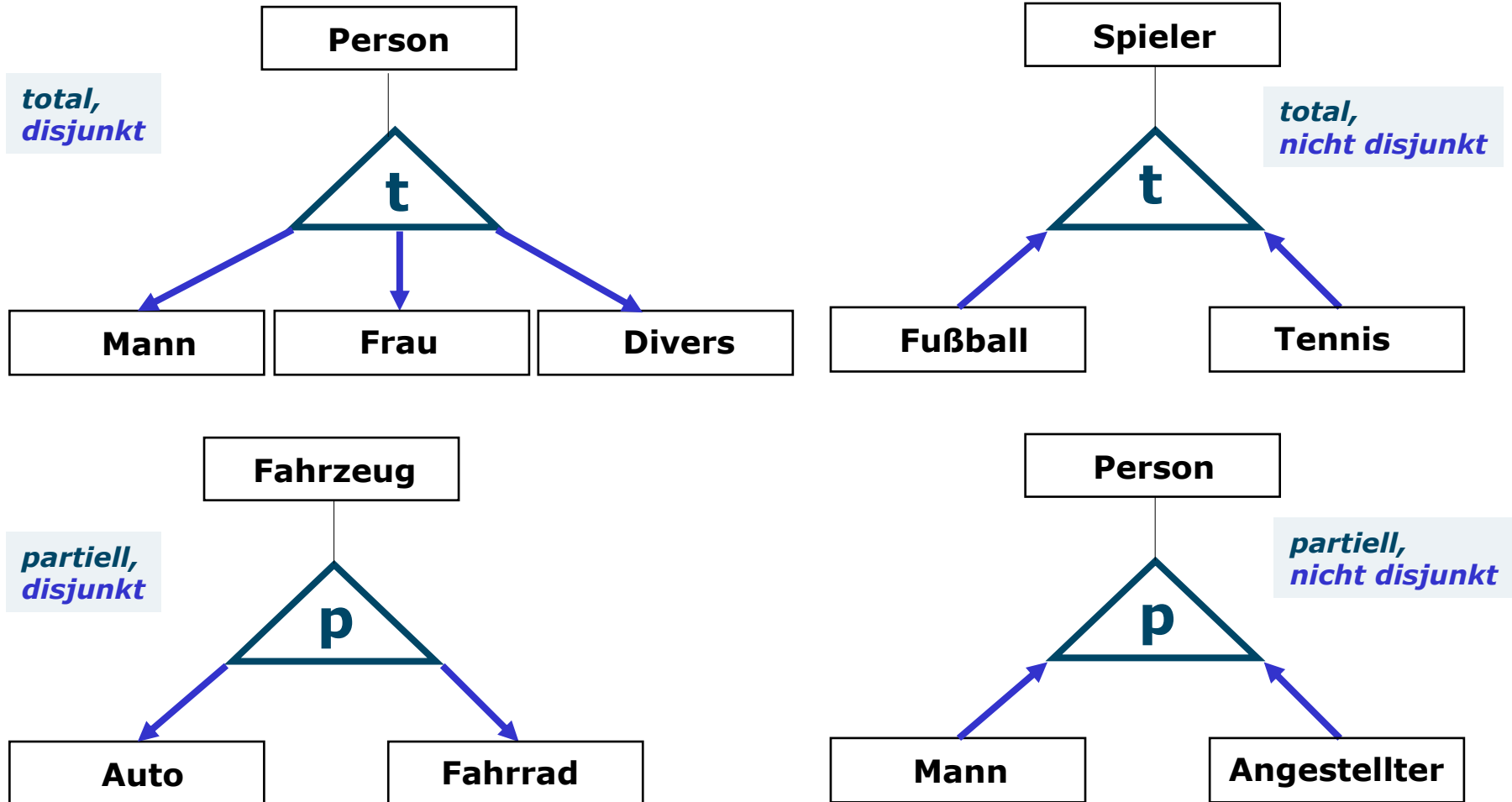
Verallgemeinerung: Person

Spezialisierung: Mann **und** Angestellter

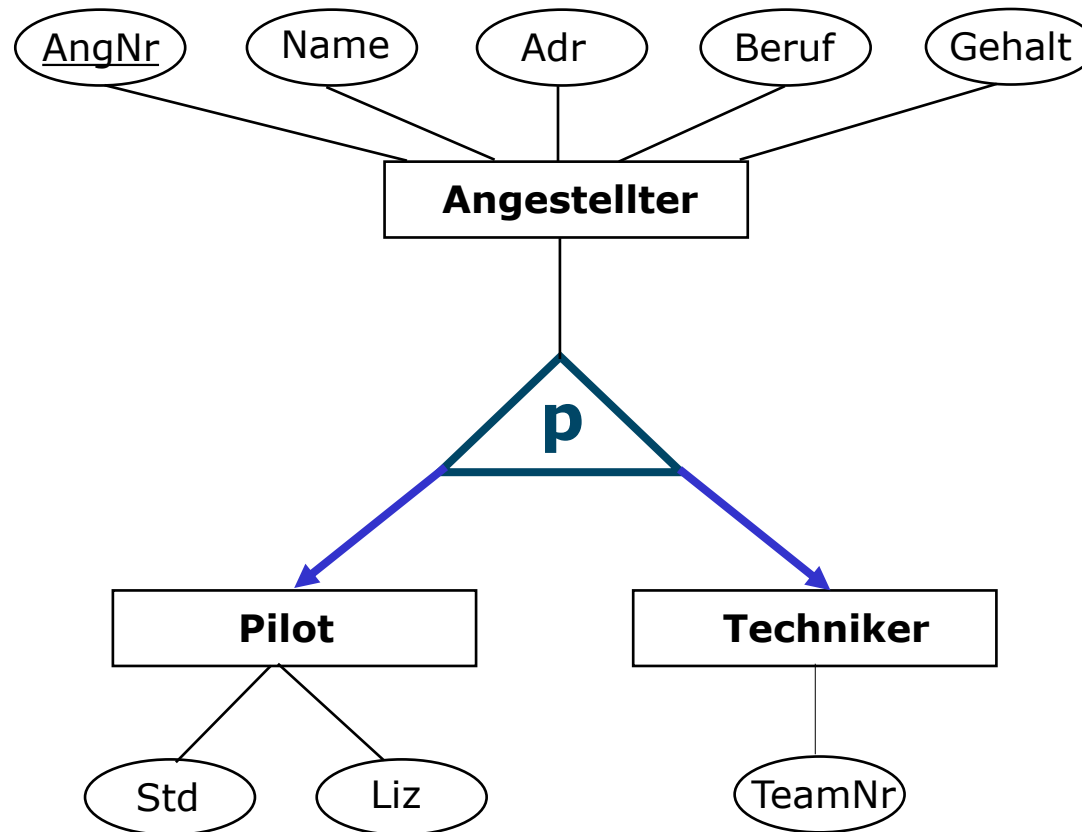
Es gibt eine gemeinsame Schnittmenge unten.

Spezialisierungen der IS-A Beziehungen

Insgesamt sind 4 Kombinationen der genannten Spezialisierungen möglich:

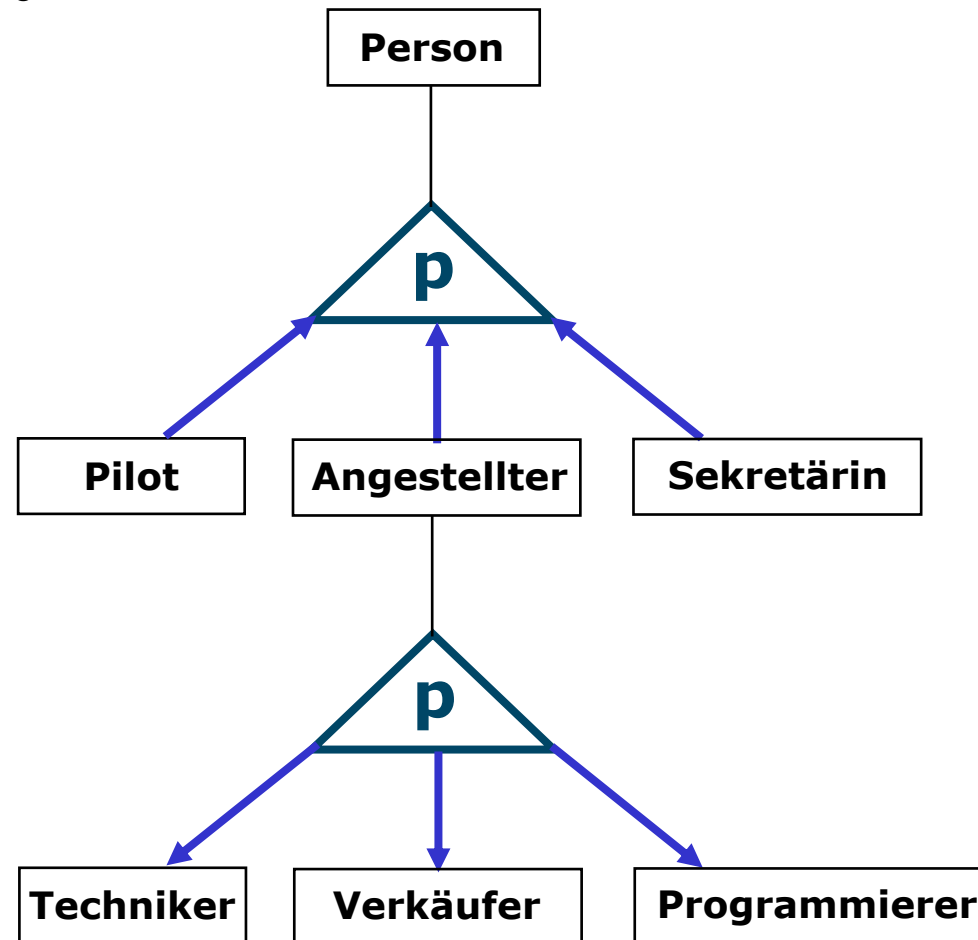


Spezialisierungen der IS-A Beziehungen



Spezialisierungen der IS-A Beziehungen

Spezialisierungs- / Vererbungshierarchien



Zusammenfassung Entity-Relationship-Modell

1. Entity -Typ Deklarationen mit

- a) Namen,
- b) einwertigen, mehrwertigen und zusammengesetzten Attributen und deren Wertebereichen,
- c) Primärschlüsseln

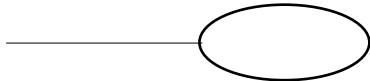
2. Relationship -Typ Deklarationen mit

- a) Namen,
- b) beteiligten Entity-Typ Deklarationen,
- c) gegebenenfalls eigenen Attributen und deren Wertebereichen,
- d) Komplexitätsfestlegung,
- e) Spezialisierungen in Form von IS-A-Beziehungen (partiell / total, disjunkt / nicht disjunkt)

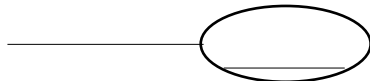
Grafische Darstellung der ER-Konstrukte



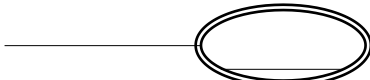
Entity bzw. Entität (Objekt) → Entity - Typ



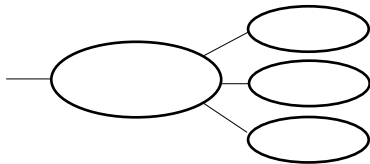
Attribut (Eigenschaft, Merkmal)



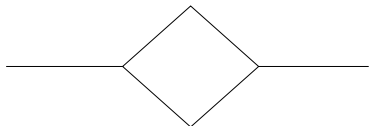
Schlüsselattribut



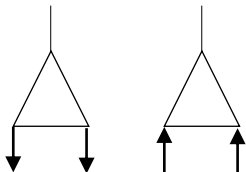
mehrwertiges Attribut



zusammengesetztes Attribut



Relationship (Beziehung)



IS – A Beziehung, disjunkt / nicht disjunkt

Konzeptioneller Entwurf mit dem Entity-Relationship-Modell

Datenbank – Entwurfsprozess (systemunabhängig)

- ▶ Anforderungsanalyse ist abgeschlossen ✓
- ▶ Konzeptionelle Globalsicht der Anwendung
- ▶ Vorgehensweise: Top – Down – Modellierung

Zunächst große Informationsblöcke,
dann schrittweise Verfeinerung des konzeptionellen Entwurfs

Verfeinerung

Entity

Relationship

Attribut

Konzeptioneller Entwurf mit dem Entity-Relationship-Modell

Entity - Verfeinerung

- a) Existierende Entities werden durch neue Typen mit relevanten Beziehungen ersetzt.
- b) Existierende Entities werden spezialisiert in Subtypen.
- c) Ein existierender Entity wird in voneinander unabhängige Typen zerlegt, welche nicht in Beziehung zueinander stehen noch Spezialisierungen darstellen.
- d) Ein Entity-Typ wird mit Attributen versehen, und unter diesen wird der Primärschlüssel ausgezeichnet.

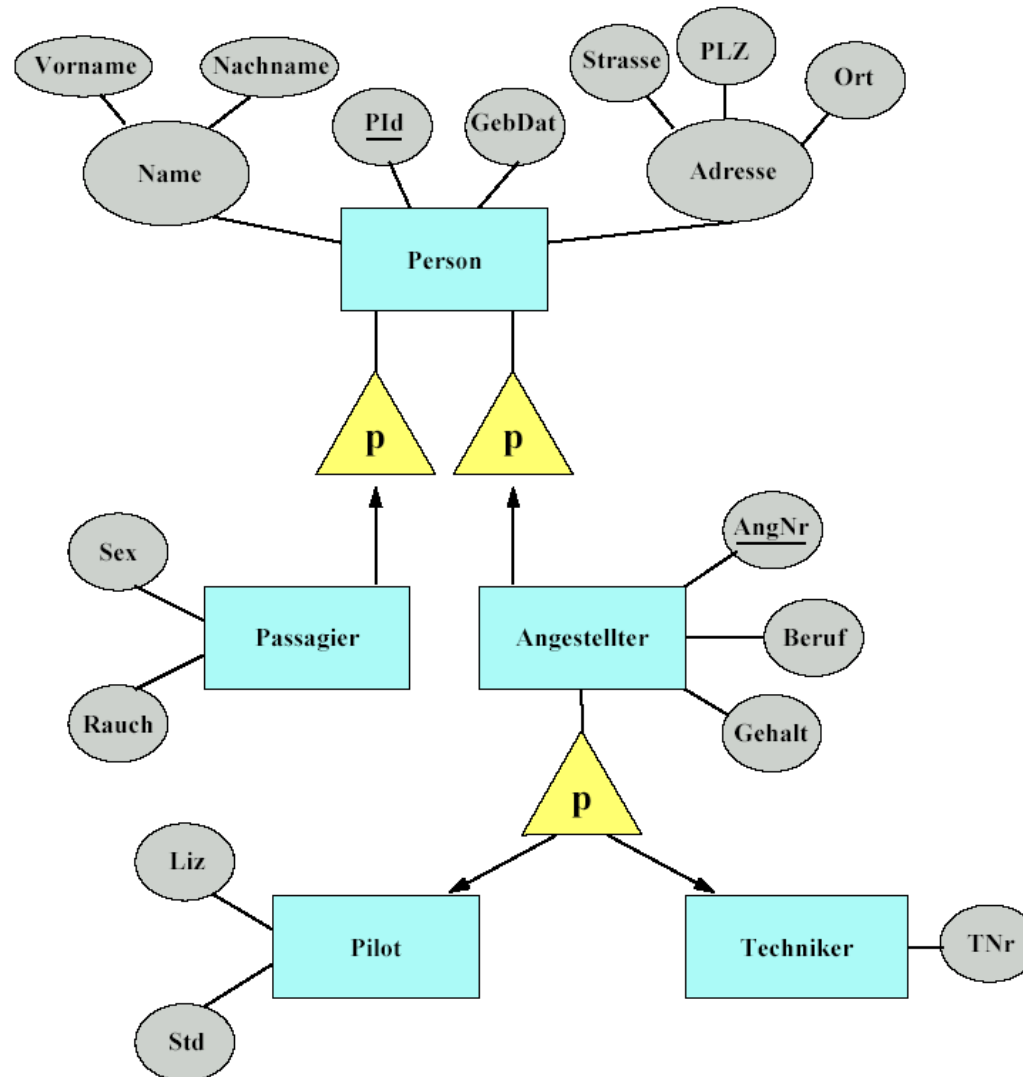
Relationship - Verfeinerung

- a) Existierende Relationships werden in zwei oder mehr Relationships zwischen den beteiligten Entitäten zerlegt.
- b) Ein existierender Relationship wird durch eine Folge von Beziehungen ersetzt (ggf. durch Hinzuziehung weiterer Entity-Typen).
- c) Ein Relationship wird mit Attributen versehen.

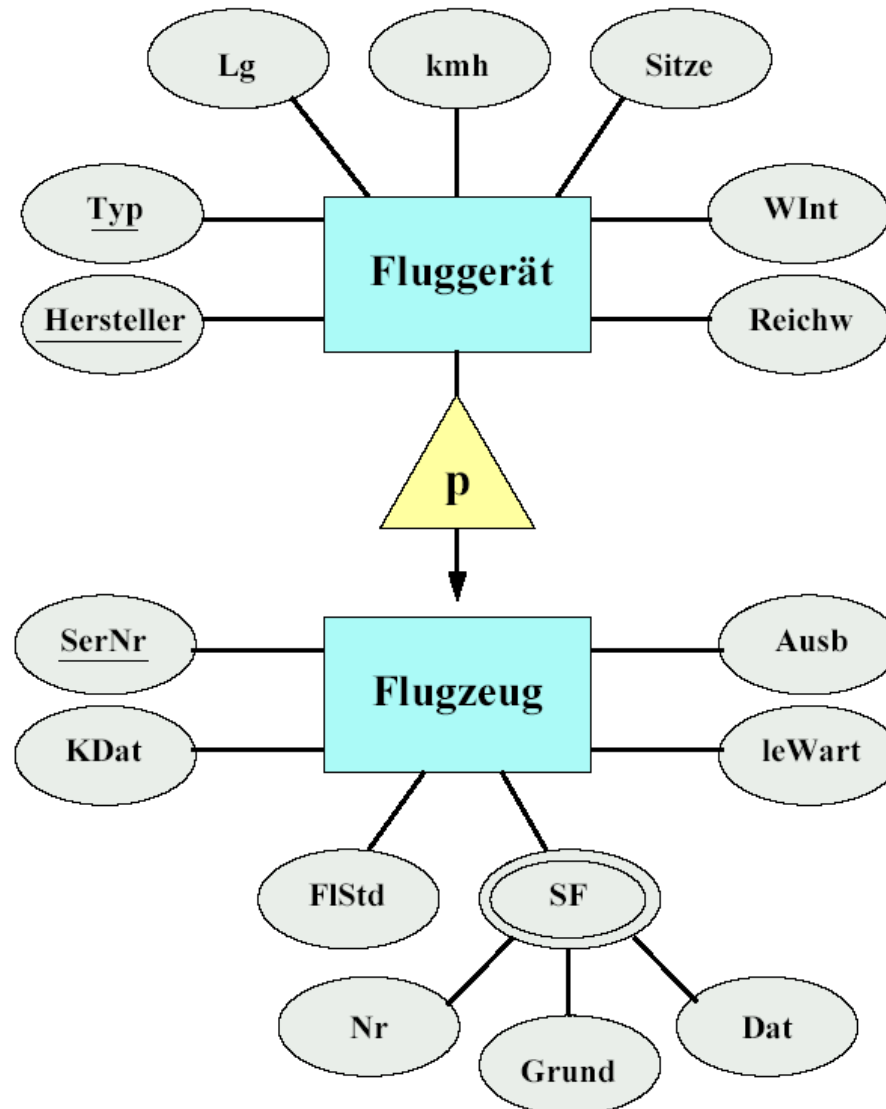
Attribut - Verfeinerung

- a) Ein Attribut einer Entität bzw. eines Relationships wird durch mehrere Attribute ersetzt.
- b) Ein Attribut wird durch ein zusammengesetztes Attribut ersetzt.

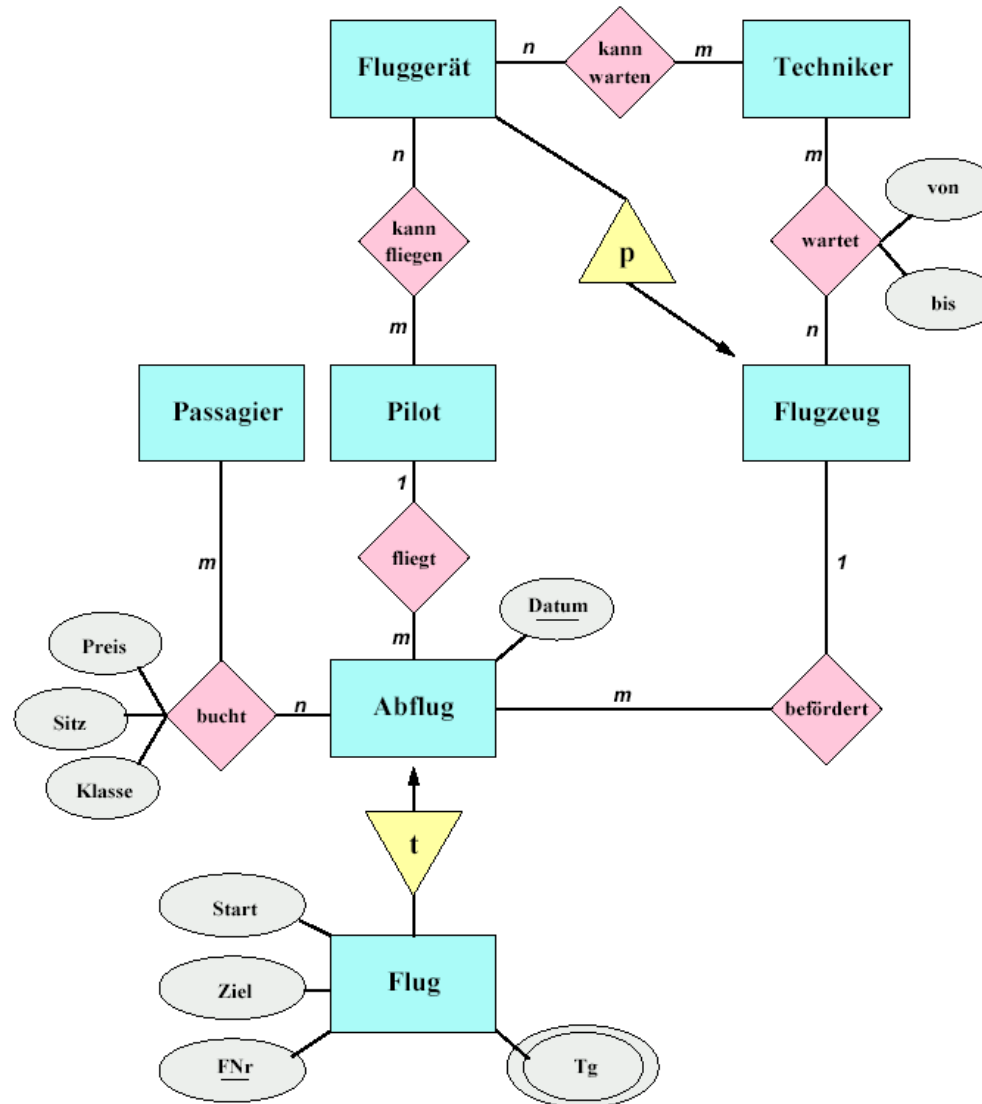
Fluggesellschaft - Teildiagramm



Fluggesellschaft - Teildiagramm



Fluggesellschaft - Teildiagramm



Qualitätsmerkmale für ER-Diagramme / Modelle

Vollständigkeit

▶ Abgleich mit Anforderungsanalyse.

Korrektheit

▶ **Syntaktisch:** Einhaltung der Modell – Definitionen.

- keine zyklischen Spezialisierungen,
- Spezialisierungen nur zwischen Entities

▶ **Semantisch:** Inhalte der Typen gem. Definition.

Häufige Fehler:

- ⊖ Verwendung von Attributen anstelle von Entitäten,
- ⊖ Vergessen von Spezialisierungen oder Verallgemeinerungen,
- ⊖ Übersehen von Vererbungseigenschaften einer Spezialisierung,
- ⊖ Relationships mit unzutreffender Anzahl beteiligter Entitäten,
- ⊖ Vertauschen von Entitäten und Relationships,
- ⊖ Vergessen von Schlüsselangaben, Kardinalitäten.

Qualitätsmerkmale für ER-Diagramme / Modelle

Minimalität

► Informale Prüfung.

Beispiel: Attribut Anzahl_Angestellte ist nicht notwendig, wenn die Anzahl am Attribut Arbeitet_an_Projekt abgeleitet werden kann.

Lesbarkeit

► Erfüllt durch graphische Darstellung.

Allerdings sollten ästhetische Kriterien berücksichtigt werden:

- ✓ Diagramm auf Gitterstruktur zeichnen,
- ✓ Rechtecke und Rauten in gleicher Größe,
- ✓ Kanten möglichst horizontal oder vertikal,
- ✓ Spezialisierungen von oben (allgemein) nach unten (speziell),
- ✓ Sichtbare und kenntlich gemachte Symmetrien,
- ✓ Möglichst kreuzungsfreie Zeichnung.

Modifizierbarkeit

► Modularer Aufbau und gute Dokumentation, Bildung größerer logischer Einheiten, Teildiagramme mit wohldefinierten Schnittstellen.

Gliederung

1.

Datenbankentwurf



Definition Datenbankentwurf



Datenmodellierung



Phasen des Datenbank-Entwurfsprozesses



Konzeptioneller Entwurf mit dem Entity-Relationship-Modell



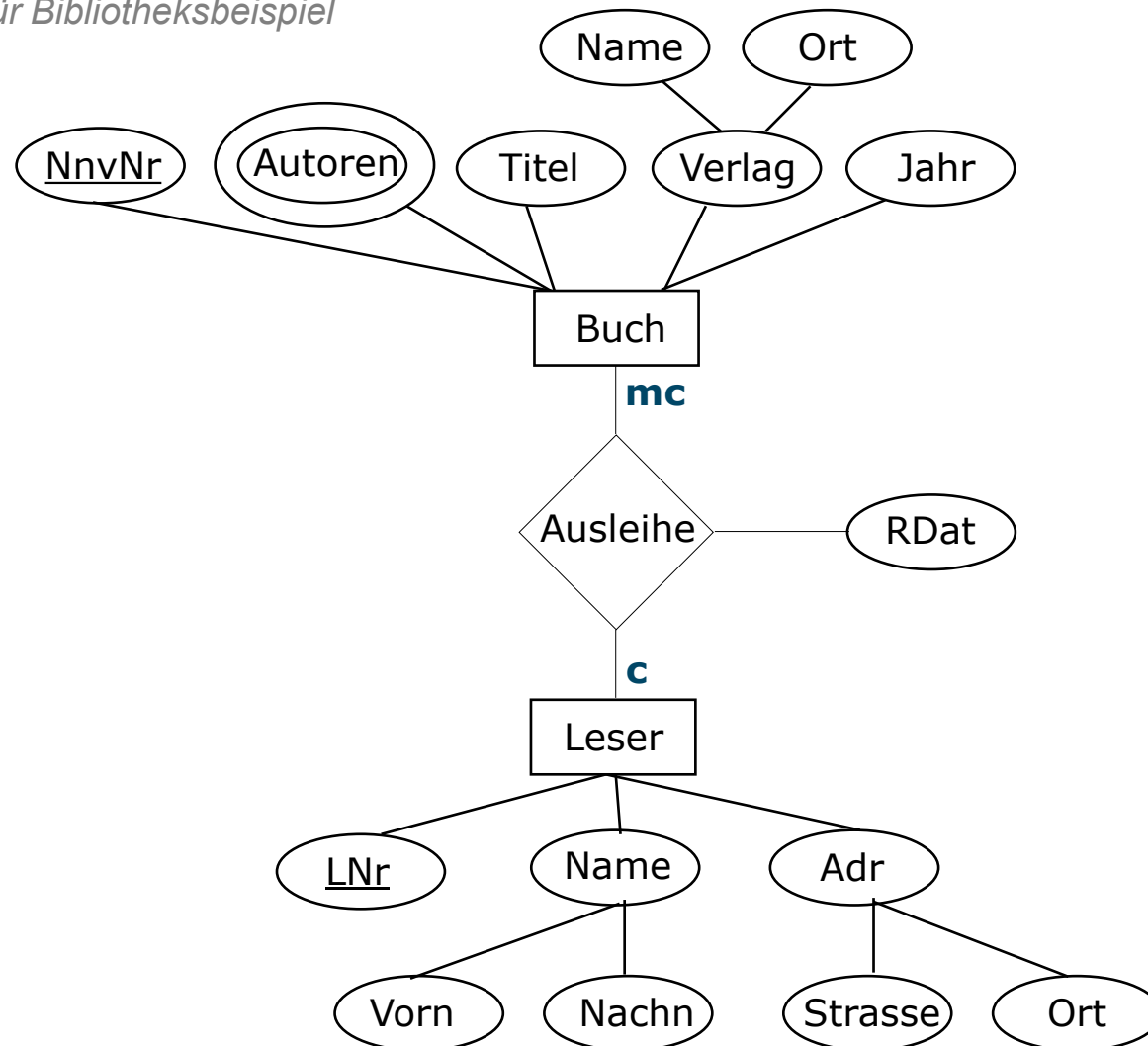
Transformation eines Entity-Relationship-Modells
in ein Relationales Datenmodell



Logischer Entwurf mit dem Relationalen Datenmodell

Konzeptioneller Entwurf mit dem Entity-Relationship-Modell

ERM-Diagramm für Bibliotheksbeispiel



Das relationale Datenmodell – Daten in Tabellenform

Relationenmodell für Bibliotheksbeispiel

Buch

<u>InvNr</u>	Autor1	Autor2	Titel	V_Name	V_Ort	Jahr
123	Vossen	Witt	DB2 Handbuch	Adisson-Wesley	Bonn	1990
125	Vossen	Witt	SQL/DS Handbuch	Adisson-Wesley	Bonn	1988
130	Witt		OO Programmierung	Oldenburg	München	1992



Ausleihe

<u>InvNr</u>	<u>LNr</u>	RDat
123	500	31-07-08
130	550	30-06-07



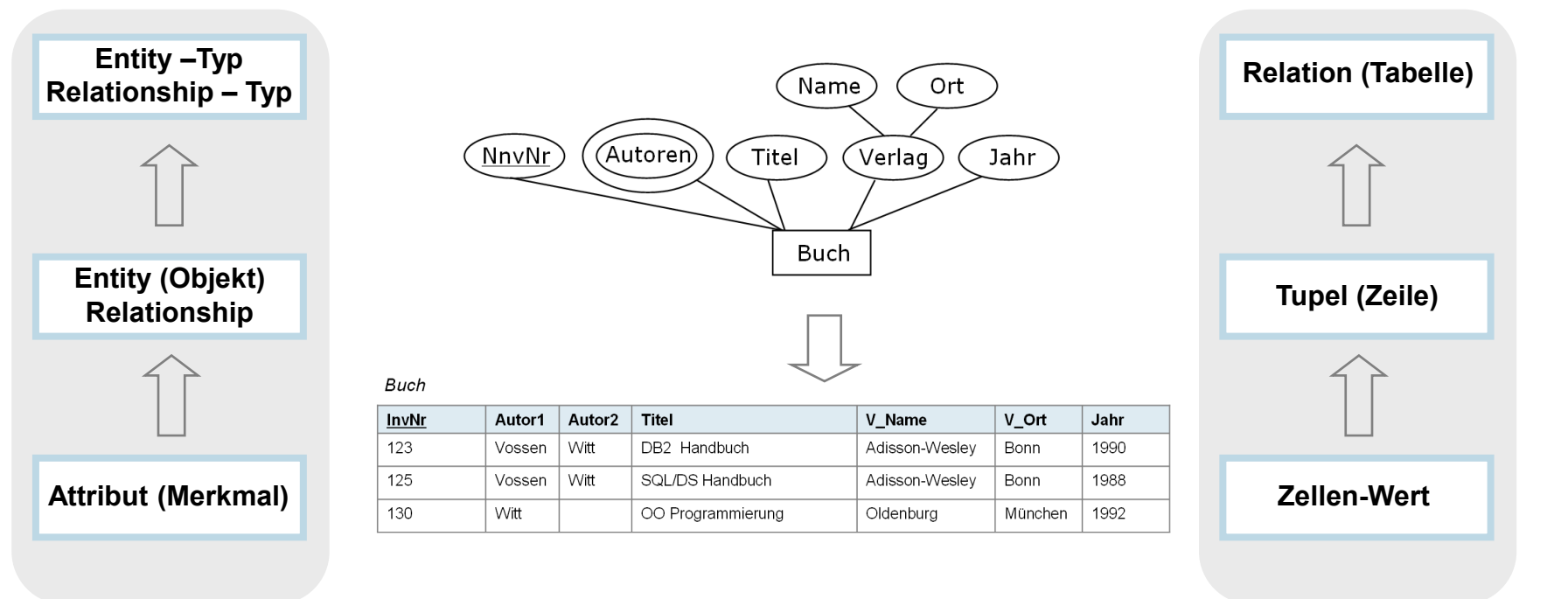
Leser

<u>LNr</u>	Vorn	Nachn	Strasse	Ort
500	Peter	Müller	Walstrasse	Köln
550	Franz	Meier	Feldweg	Bonn

Entity – Relationship – Modell vs. Relationenmodell

Das Relationenmodell nach **E.F. Codd** (1970)

- als de facto Standard seit Mitte der 80er Jahre
- überwiegende Mehrheit aktueller DBMS sind relationale DBMS → RDBMS
- Relational, weil... abgeleitet vom mathematischen Konzept der Relationenalgebra
- Im relationalen Datenmodell werden Eigenschaftswerte sowohl von Entitäten (Objekten) als auch von Beziehungen zwischen Entitäten in Form von (Datenbank-) Tabellen mit einzelnen Zeilen dargestellt



Relation

Definition

(i) Ein *Tupel* über X ist eine (bis auf weiteres totale) Abbildung

$$\mu : X \rightarrow \text{dom}(X),$$

für die gilt:

$$(\forall A \in X) \mu(A) \in \text{dom}(A).$$

Zu gegebener Attributmenge X bezeichne dann $\text{Tup}(X)$ die Menge aller Tupel über X ; man beachte, daß es sich bei $\text{Tup}(X)$ um eine Menge (ohne doppelte Elemente) handelt.

(ii) Eine *Relation* r über X ist eine (endliche) Menge von Tupeln über X , d.h. $r \subseteq \text{Tup}(X)$; mit $\text{Rel}(X)$ bezeichnen wir die Menge aller Relationen über X .

Transformationen eines ER-Modells auf ein Relationenmodell

Erklärung der Konstrukte des Relationenmodells:

Relationenname (Entity-Typ bzw. Relationship-Typ Bezeichnung)

Spaltenüberschrift (Attribut)

Relationenschema (Entity-Typ bzw. Relationship-Typ Deklaration)

Relation (Entity-Typ bzw. Relationship-Typ)

Spaltenwert (Attributwert)

Tupel (Entität bzw. Relationship)

Buch

<u>InvNr</u>	Autor1	Autor2	Titel	V_Name	V_Ort	Jahr
123	Vossen	Witt	DB2 Handbuch	Adisson-Wesley	Bonn	1990
125	Vossen	Witt	SQL/DS Handbuch	Adisson-Wesley	Bonn	1988
130	Witt		OO Programmierung	Oldenburg	München	1992

Falls für ein Attribut in einem Datensatz kein Wert vorliegt, spricht man von einem NULL – Wert, nicht zu verwechseln mit dem Wert 0 (konkreter Wert!). Mögliche Gründe für NULL - Werte in Attributen:

- a) Der Wert des Objektes ist zu diesem Zeitpunkt unbekannt,
- b) dieses Attribut trifft für das Objekt nicht zu oder
- c) eine Kombination aus a) und b).

Transformationen eines ER-Modells auf ein Relationenmodell

Erklärung der Konstrukte des Relationenmodells:

ERM-Konstrukt	RM-Konstrukt	Erklärung
Entity-Typ bzw. Relationship-Typ Bezeichnung	Relationenname	Bezeichnung der Tabelle
Entity-Typ bzw. Relationship-Typ Deklaration	Relationenschema	Überschriftenzeile beinhaltet Spaltenüberschriften, Schema der Tabelle besteht auf fester Anzahl von Spalten
Attribut	Spaltenüberschrift	Spalten repräsentieren Eigenschaften – haben Namen und festgelegten Datentyp
Entität bzw. Relationship	Tupel	Zeilen (Überschriftenzeile ausgeschlossen) repräsentieren eigentliche Daten bzw. Datensätze
Attribut-Wert	Spaltenwert	Zellenwert einer Zeile
Entity-Typ bzw. Relationship-Typ	Relation	Gesamtheit an Zeilen
Schlüssel – Attribut (Kombination)	Primärschlüssel	Einzelne Spalten (Kombination), deren Wert (-kombination) innerhalb der Tabelle einmalig ist

Das relationale Datenmodell – Daten in Tabellenform

Eigenschaften von Relationen

Eine Relation

Name

hat einen eindeutigen Namen

0 bis viele Tupel

besteht aus 0 bis vielen Tupeln (d.h. eine Relation kann somit auch leer sein)
Die Reihenfolge / Position der Tupel in der Relation ist bedeutungslos,
weil ein Tupel aufgrund seiner Werte in der Relation angesprochen wird.

ein bis viele Attribute

hat mindestens ein Attribut.
Die Anzahl der Attribute ist festgelegt und damit nicht variabel.
Die Reihenfolge der Attribute ist bedeutungslos, weil ein Attribut auf Grund
seines Namens und nicht auf Grund seiner Position angesprochen wird.

Schlüssel

hat genau einen Schlüssel (Primärschlüssel).
Er ist ein Attribut, mit dessen Werten die Tupeln der Relation eindeutig zu
identifizieren sind. Unter Umständen lässt sich die eindeutige Identifikation
nur mit einem zusammengesetzten Schlüssel erzielen, der aus mehreren
Attributen (Teilschlüsseln) besteht.
Ein Schlüsselwert kommt nur einmal in einer Relation vor, somit kann
auch ein Tupel nur einmal in einer Relation vorkommen.
Schlüssel können durch Unterstreichen gekennzeichnet werden.

Das relationale Datenmodell – Daten in Tabellenform

Eigenschaften von Relationen

Ein Attribut

Attributwerte

Ein Attribut enthält Attributwerte, die aus einer Domäne stammen.
Eine Domäne ist die Menge aller zulässigen Werte eines oder mehrerer Attribute.

Namen

Innerhalb einer Relation hat jedes Attribut einen eindeutigen Namen.
Dieser entspricht normalerweise dem Namen der Domäne, aus der die Attributwerte stammen.

Rollename

Um auch dann eindeutige Attributnamen zu erhalten, wenn mehrere Attribute einer Relation ihre Werte aus derselben Domäne beziehen, qualifiziert man den Domänennamen mit einem Rollennamen und erhält somit den eindeutigen Attributnamen.

Beispiel:	Domänenbezeichnung	Name
	Rollenbezeichnung	Vor
	Attributbezeichnung	Vorname

Das relationale Datenmodell – Daten in Tabellenform

Eigenschaften von Relationen

Relationsschlüssel

Schlüsselkandidat

Es ist möglich, dass mehrere Attribute einer Relation jeweils in der Lage sind, die Identifikationsfunktion zu übernehmen, und somit Relationenschlüssel sein könnten. Sie werden als Schlüsselkandidaten bezeichnet.

Beispiel: Relation Mitarbeiter
 Schlüsselkandidaten Personalnummer und
 Sozialversicherungsnummer

Primärschlüssel

Den Schlüsselkandidaten, dem die Identifikationsaufgabe zugeordnet wird, bezeichnet man als Primärschlüssel.

Sekundärschlüssel

Die restlichen Schlüsselkandidaten, die nicht als Primärschlüssel gewählt wurden, nennt man Sekundärschlüssel.

Entitätsintegrität

Beim Einfügen (INSERT) eines Tupels in eine Relation muss mindestens dessen Primärschlüsselwert bekannt sein. Die restlichen Attributwerte können durch eine Veränderungsoperation an dem betroffenen Tupel nachgereicht werden. Diesen Sachverhalt bezeichnet man als Entitätsintegrität. Integrität bedeutet Richtigkeit. Mit Entitätsintegrität drückt man den Sachverhalt aus, dass eine Entität in der Datenhaltung korrekt abgebildet ist. Die Minimalanforderung hierfür ist, dass der Entitätsschlüssel, der Repräsentant der Entität, immer bekannt ist.

Das relationale Datenmodell – Daten in Tabellenform

Eigenschaften eines Primärschlüssels

Primärschlüssel

- ▶ Er ist Schlüsselkandidat (er identifiziert).
- ▶ Er ist prinzipiell unveränderlich.
- ▶ Bei einem zusammengesetzten Primärschlüssel ist jeder Teil zur Identifikation des Tupels erforderlich.
- ▶ Er bildet die angemessene gedankliche Brücke zur abgebildeten Entität oder Beziehung.
- ▶ Er ist immer vorhanden. Das bedeutet, dass mit der Aufnahme der Daten einer Entität in eine Relation mindestens der Wert des Schlüsselattributes angegeben werden muss.
- ▶ Er ist wirtschaftlich zu handhaben (kurz und einprägsam).



Abbilden von Beziehungen

► Hierarchische Beziehungen

Verschmelzen von Entity- und Relationship-Typen auf Relationen (bei 2-stelligen Beziehungen: 2 Relationen)

- ✓ Aufnahme der Attribute der Entity-Typ-Deklarationen in die jeweilige Relation
- ✓ Aufnahme der Attribute des Relationship-Typs in eine der beiden Relationen
- ✓ Primär-Schlüsselbildung in Relationen durch Übernahme der Schlüssel aus den Entity-Typ-Deklarationen

► Konditionelle und netzförmige Beziehungen

Abbilden jeder einzelnen Typ-Deklaration auf eine Relation (bei 2-stelligen Beziehungen: 3 Relationen)

- ✓ Aufnahme der Attribute der Typ-Deklarationen in die jeweilige Relation
 - ✓ Bei „Relationship-Typ“-Relation: Zusätzlich Übernahme aller Primärschlüssel der beteiligten Entity-Typ-Deklarationen
- ✓ Primär-Schlüsselbildung in Relationen durch Übernahme der Schlüssel aus den Typ-Deklarationen



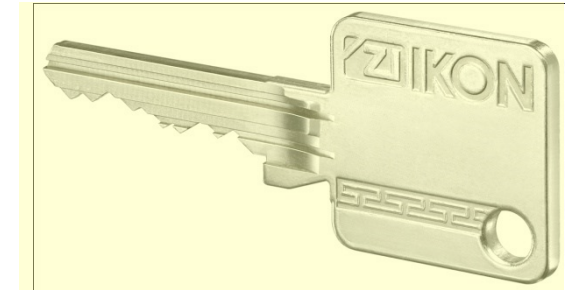
Darstellung von Beziehungen durch **Fremd - Schlüsselbildungen!**



Abbilden von Beziehungen

Eigenschaften eines Fremdschlüssels

Fremdschlüssel

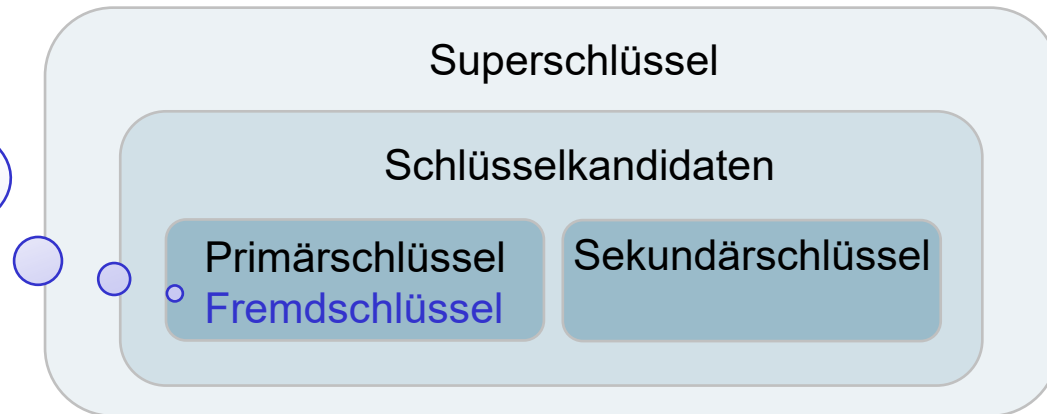


- ▶ Ein Merkmal, das in einer Tabelle als Schlüsselattribut fungiert und das in eine andere Tabelle aufgenommen wird, wird als Fremdschlüssel bezeichnet.
- ▶ Er dient zur Verknüpfung (Verweis) zusammengehörender Tupel zwischen Relationen, d. h. er zeigt an, welche Tupel der Relationen inhaltlich miteinander in Verbindung stehen.

Referentielle Integrität

Abgrenzung diverser Schlüsselbegriffe

Ein Fremdschlüssel verweist auf einen Primärschlüssel, damit wird eine Beziehung zwischen Relationen dargestellt!



Superschlüssel

➤ Menge von Attributen in einer Relation, die die Tupel in dieser Relation eindeutig identifizieren. Trivialer Superschlüssel: Die Menge aller Attribute einer Relation.

Schlüsselkandidaten

➤ Eine minimale Teilmenge der Attribute eines Superschlüssels, welche die Identifizierung der Tupel ermöglicht.

Primärschlüssel

➤ Ein ausgewählter Schlüsselkandidat. Die Werte dieses Schlüssels können in einer anderen Tabelle als **Fremdschlüssel** verwendet werden.

➤ **Sprechender Schlüssel** (natürlicher Schlüssel):
Ein Schlüsselkandidat, der im Tupel auf natürliche Weise vorhanden ist. Ein solcher Schlüssel besitzt auch in der realen Welt eine Bedeutung z. B. RNr = Rechnungsnummer.

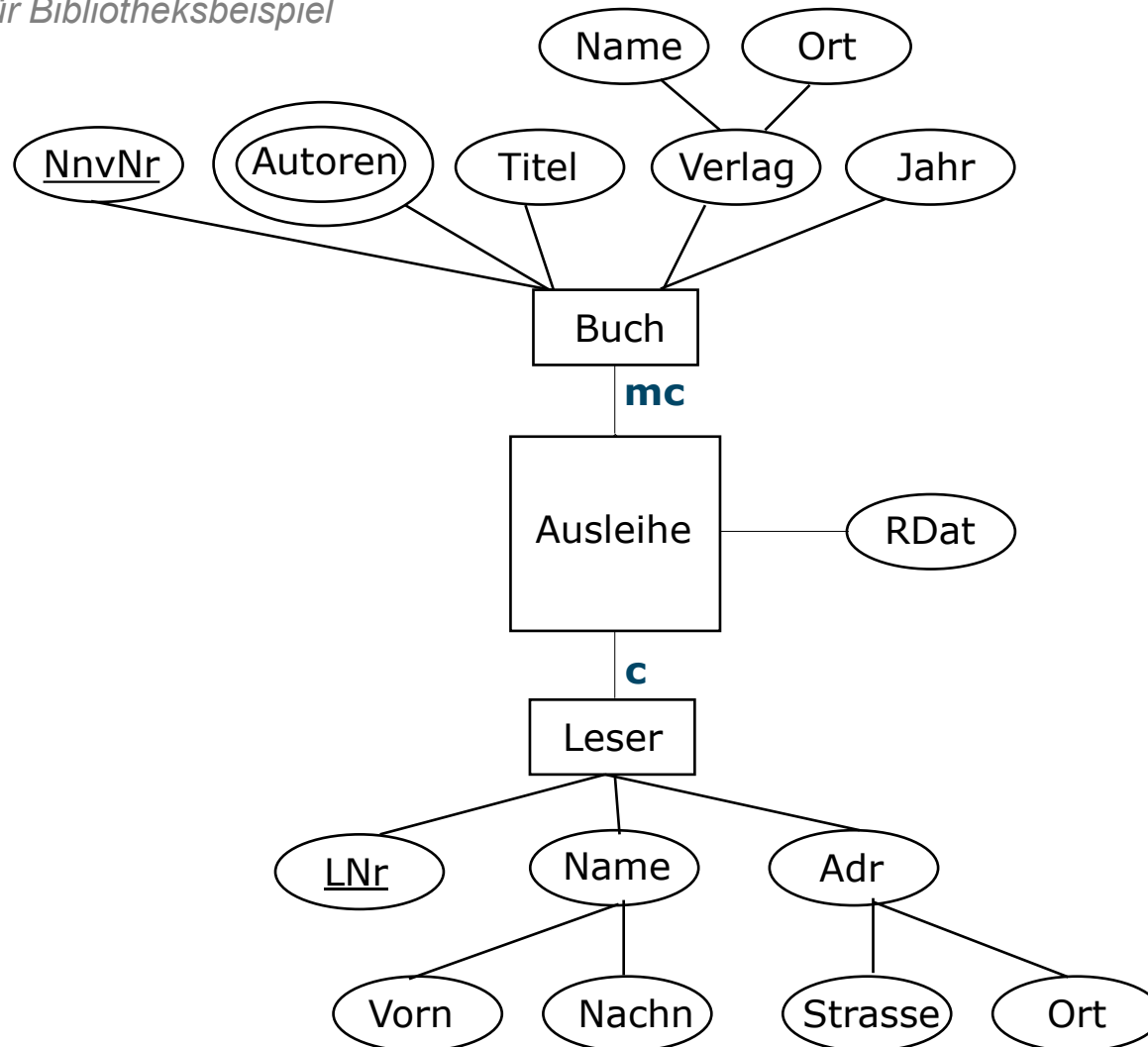
➤ **Stellvertretender Schlüssel** (Surrogatschlüssel):
Ein künstlich erzeugtes im Tupel zuvor gar nicht vorkommendes Attribut, das die Tupel der Relation identifiziert z. B. „Laufende Nummer“

Sekundärschlüssel

➤ Ein nicht ausgewählter Schlüsselkandidat.

Abbilden von Beziehungen - Beispiel

ERM-Diagramm für Bibliotheksbeispiel



Gliederung

1.

Datenbankentwurf



Definition Datenbankentwurf



Datenmodellierung



Phasen des Datenbank-Entwurfsprozesses



Konzeptioneller Entwurf mit dem Entity-Relationship-Modell



Transformation eines Entity-Relationship-Modells
in ein Relationales Datenmodell



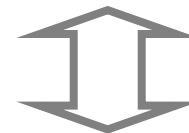
Logischer Entwurf mit dem Relationalen Datenmodell

Normalisierung

Zielsetzung

Entity – Relationship- Modell

Entwicklung eines **unternehmensweiten, globalen** Datenmodells mit geringem Aufwand.



Relationen – Modell

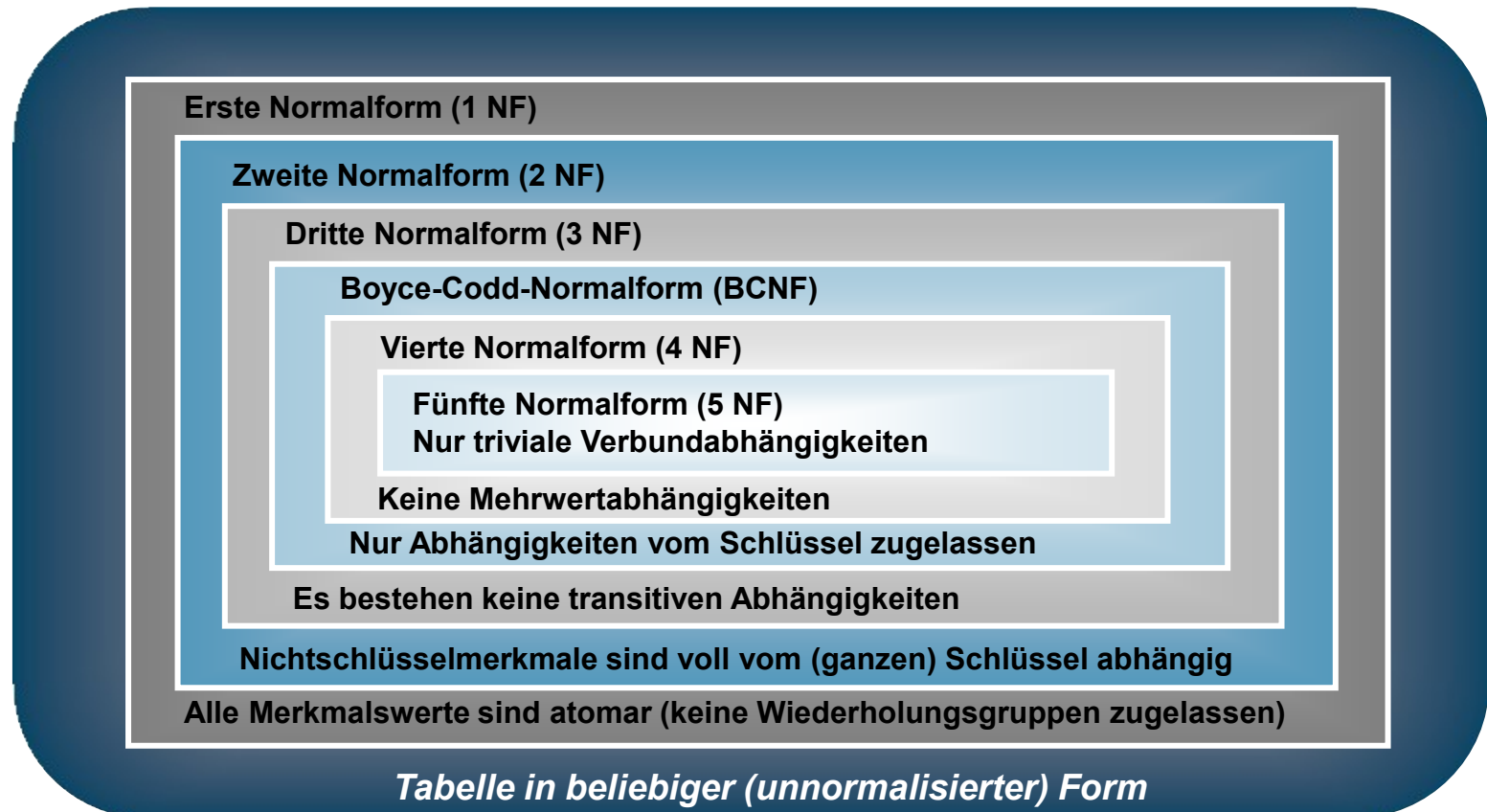
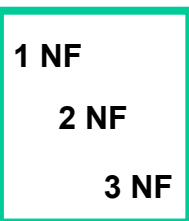
Entwicklung eines **stabilen** Datenbankmodells mit der ihm zu Grunde liegenden Methoden der Normalisierung und damit Befreiung von Defekten.

Normalisierung

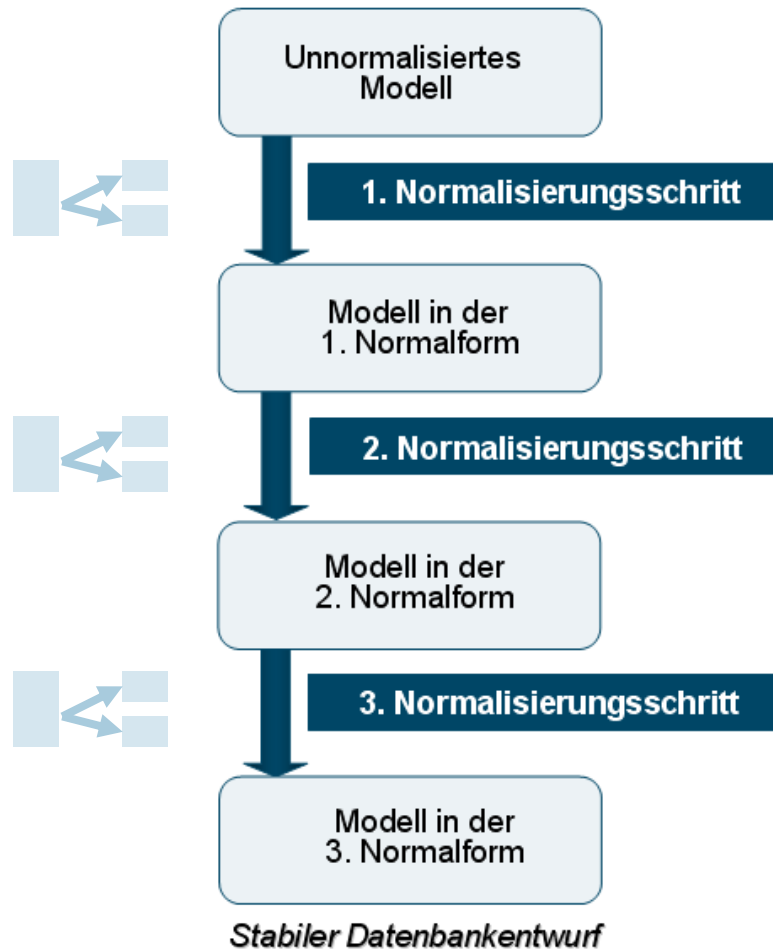
- ▶ Unter Normalisieren (Anwenden von Normalformen) versteht man das systematische Untersuchen von Relationen mit dem Zweck zur Überprüfung der Qualität eines Datenbankentwurfs und zur evtl. Verbesserung.
- ▶ Qualitativ hochwertige Relationen weisen folgende Eigenschaften auf:
 - Sie sind redundanzfrei.
 - Sie weisen keine Defekte (Anomalien) bei DB-Operationen auf (INSERT, UPDATE, DELETE).
 - Sie beschreiben einen Ausschnitt der Realität angemessen und richtig.
- ▶ Es kann für jeden Entwurf einer Datenbanktabelle entschieden werden, in welcher Normalform sie sich befindet.

Normalisierung

- ▶ Es gibt insgesamt 5 Normalformen, die aufeinander aufbauen.
Das bedeutet beispielsweise, dass die 2. Normalform nur dann erfüllt sein kann, wenn die 1. Normalform erfüllt ist.
- ▶ In der Praxis sind nur die ersten drei Normalformen von Bedeutung.



Normalisierung



Ziel des jeweiligen Normalisierungsschritts

- ✓ Nichtvorhandensein von Wiederholungsfeldern.
- ✓ Nichtvorhandensein von zusammengesetzten Merkmalen.
- ✓ Nichtvorhandensein von Teilschlüsselabhängigkeiten.
- ✓ Nichtvorhandensein von Abhängigkeiten zwischen Nichtschlüselfeldern.

Prinzip:

Aufspaltung einer größeren Tabelle zur Redundanzvermeidung in mehrere kleinere Tabellen und deren Rekonstruktion durch Joins!

Normalisierung

Rechnungsbeispiel: Dem Unternehmen liegen drei Rechnungen von zwei Kunden vor

Herrn
Hugo Müller
Gartenstr. 4a

69123 Heidelberg

Rechnungsnummer: **R001**
Kundennummer: K001
Datum: 04.04.2004
Rechnungsbetrag: 13.000,00 Euro

Arti-Nr.	Bezeichnung	Anzahl	E-Preis	G-Preis
A001	Computer	2	5.000,00	10.000,00
A002	Drucker	3	1.000,00	3.000,00

Herrn
Hugo Müller
Gartenstr. 4a

69123 Heidelberg

Rechnungsnummer: **R002**
Kundennummer: K001
Datum: 05.04.2004
Rechnungsbetrag: 2.000,00 Euro

Arti-Nr.	Bezeichnung	Anzahl	E-Preis	G-Preis
A002	Drucker	1	1.000,00	1.000,00
A003	Bildschirm	2	500,00	1.000,00

Herrn
Georg Mayer
Neckarstraße 1

69123 Heidelberg

Rechnungsnummer: **R003**
Kundennummer: K002
Datum: 05.04.2004
Rechnungsbetrag: 5.000,00 Euro

Arti-Nr.	Bezeichnung	Anzahl	E-Preis	G-Preis
A001	Computer	1	5.000,00	5.000,00

Normalisierung

Rechnungsbeispiel: Dem Unternehmen liegen drei Rechnungen von zwei Kunden vor

Setzt man an Stelle der Eigenschaftswerte die Eigenschaftsnamen in die Rechnung ein, so erhält man das Rechnungsformular wie folgt:

<Anrede>
<Vorname> <Zuname>
<Straße mit Hausnummer>

<Postleitzahl> <Ort>

Rechnungsnummer: <R#>

Kundennummer: <K#>

Datum: <Datum>

Rechnungsbetrag: <Rechnungsbetrag> Euro

Arti-Nr.	Bezeichnung	Anzahl	E-Preis	G-Preis
<A#>	<Bezeichnung>	<Anzahl>	<E-Preis>	<G-Preis>

Normalisierung

Unnormalisierte Relation

Durchführung einer Formularanalyse

1. Alle Attribute des Formulars werden gesammelt und mit Eigenschaftsnamen belegt.
2. Benennung der Sammlung mit einem Relationsnamen.
3. Auswahl des Primärschlüssels.

Darstellung in Tabellenform:

Rechnung

RNr	Datum	Betrag	KNr	VName	ZName	Str	PLZ	Ort	ANr	AName	APreis	Anzahl	GPreis
R001	04.04.2004	13.000	K001	Hugo	Müller	Garten str. 4a	69123	Heidelberg	A001	Computer	5.000	2	10.000
									A002	Drucker	1.000	3	3.000
R002	05.04.2004	2.000	K001	Hugo	Müller	Garten str. 4a	69123	Heidelberg	A002	Drucker	1.000	1	1.000
									A003	Kabel	500	2	1.000
R003	05.04.2004	5.000	K003	Georg	Mayer	Neckar str. 1	69123	Heidelberg	A001	Computer	5.000	1	5.000

Normalisierung

Darstellung auf der Typ-Ebene (Relationenmodell in UML-Darstellung)

Rechnung (unnormalisiert)

RNr
Datum
Betrag
KNr
VName
ZName
Str
PLZ
Ort
ANr[1..*]
AName[1..*]
APreis[1..*]
Anzahl[1..*]
GPreis[1..*]

Multiplizitätsangaben [1..] als Indikatoren für
Wiederholungsgruppenwerte der entsprechenden Attribute*

Normalisierung

Nachteile einer unnormalisierten Relation

Kommen an einem Kreuzungspunkt von Attribut (Spalte) und Tupel (Zeile) mehrere Attributwerte vor, so ist das Attribut kein skalarer, sondern ein zusammengesetzter Datentyp und hat die Form einer **Wiederholungsgruppe**.



Zur **Identifikation** eines speziellen Attributwertes innerhalb der Werte der Wiederholungsgruppe ist der Primärschlüssel des Tupels nicht mehr ausreichend.

Normalisierung

Nachteile einer unnormalisierten Relation

Datenredundanz



Integritätsprobleme (Defekte, Anomalien)
bei Veränderungsoperationen (INSERT,DELETE,UPDATE)
an der Relation (Tabelle)

Einfügungsdefekt

Unfakturierte Artikel sind nicht existent und führen zu einer nicht realitätsgerechten Datenhaltung.

Löschdefekt

Das Löschen einer Rechnung kann ggf. den Artikel löschen, damit verschwindet seine Aufzeichnung auch in der Datenhaltung (Lagerbestand?).

Veränderungsdefekt

Artikeländerungen (z.B. Bezeichnung) führen zu temporären Inkonsistenzen.

Normalisierung

Definition der 1. Normalform,

Eine Relation ist dann in der ersten Normalform (1NF), wenn sie an den Kreuzungspunkten der Tupel (Zeilen) und der Attribute (Spalten) jederzeit höchstens einen Wert, d. h. einen skalaren Wert oder kurz ein Skalar aufweist.

1. Normalform verlangt,

- ✓ Nichtvorhandensein von Wiederholungsfeldern (Mehrfachabhängigkeiten vom Schlüssel)
- ✓ Nichtvorhandensein von zusammengesetzten Merkmalen

Vorgehensweise bei der Bildung der 1. Normalform

(1. Normalisierungsprozess)

1. Kennzeichnen der Attribute, die zusammengehörige Wiederholungsgruppen (Redundanz) darstellen.
2. Entfernung dieser Wiederholungsgruppen aus der Ausgangsrelation.
3. Erstellung neuer Relationen für die Wiederholungsgruppen unter Verwendung des Schlüssels der Ausgangsrelation.
4. Deklaration (Bestimmung) des Schlüssels der neuen Relation(en).
5. Benennung der neuen Relation(en).

Normalisierung

Ausgangsrelation mit Wiederholungsgruppen

Rechnung (unnormalisiert)

RNr	Datum	Betrag	KNr	VName	ZName	Str	PLZ	Ort	ANr	AName	APreis	Anzahl	GPreis
R001	04.04.2004	13.000	K001	Hugo	Müller	Garten str. 4a	69123	Heidelberg	A001	Computer	5.000	2	10.000
									A002	Drucker	1.000	3	3.000
R002	05.04.2004	2.000	K001	Hugo	Müller	Garten str. 4a	69123	Heidelberg	A002	Drucker	1.000	1	1.000
									A003	Kabel	500	2	1.000
R003	05.04.2004	5.000	K003	Georg	Mayer	Neckar str. 1	69123	Heidelberg	A001	Computer	5.000	1	5.000

Von den Wiederholungsgruppen bereinigte Ausgangsrelation

Rechnung (NF1)

RNr	Datum	Betrag	KNr	VName	ZName	Str	PLZ	Ort
R001	04.04.2004	13.000	K001	Hugo	Müller	Garten str. 4a	69123	Heidelberg
R002	05.04.2004	2.000	K001	Hugo	Müller	Garten str. 4a	69123	Heidelberg
R003	05.04.2004	5.000	K003	Georg	Mayer	Neckar str. 1	69123	Heidelberg

Abgespaltete Relation

Artikel (unnormalisiert)

RNr	ANr	AName	APreis	Anzahl	GPreis
R001	A001	Computer	5.000	2	10.000
R001	A002	Drucker	1.000	3	3.000
R002	A002	Drucker	1.000	1	1.000
R002	A003	Kabel	500	2	1.000
R003	A001	Computer	5.000	1	5.000

Normalisierung

Abgespaltete Relation

Artikel (unnormalisiert)

RNr	ANr	AName	APreis	Anzahl	GPreis
R001	A001	Computer	5.000	2	10.000
R001	A002	Drucker	1.000	3	3.000
R002	A002	Drucker	1.000	1	1.000
R002	A003	Kabel	500	2	1.000
R003	A001	Computer	5.000	1	5.000

Schlüsselbildung



Position (1NF)

PNr	<u>RNr</u>	<u>ANr</u>	AName	APreis	Anzahl	GPreis
P001	R001	A001	Computer	5.000	2	10.000
P002	R001	A002	Drucker	1.000	3	3.000
P001	R002	A002	Drucker	1.000	1	1.000
P002	R002	A003	Kabel	500	2	1.000
P001	R003	A001	Computer	5.000	1	5.000

Eigentlich würde die Kombination aus Attributen „RNr“ und „ANr“ zur Identifikation der jeweiligen Zeilen genügen. In der Praxis wird oft stattdessen ein „künstlicher“ Schlüssel eingeführt!

Normalisierung

Darstellung auf der Typ-Ebene (Relationenmodell in UML-Darstellung)



Normalisierung

Position (1NF)

PNr	RNr	ANr	AName	APreis	Anzahl	GPreis
P001	R001	A001	Computer	5.000	2	10.000
P002	R001	A002	Drucker	1.000	3	3.000
P001	R002	A002	Drucker	1.000	1	1.000
P002	R002	A003	Kabel	500	2	1.000
P001	R003	A001	Computer	5.000	1	5.000

Eine Relation in der 1. Normalform ist immer noch für Defekte anfällig, da sie noch Redundanzen aufweisen kann.

Grund:

Attribute *AName* und *APreis* sind nicht nur vom gesamten (zusammengesetzten) Schlüssel *RNr* und *ANr* funktional abhängig, sondern bereits von einem Teil hiervon, dem Teilschlüssel *ANr*.

Normalisierung

Definition der 2. Normalform,

Eine Relation befindet sich in der zweiten Normalform (2NF), wenn alle Nichtschlüsselattribute vom gesamten Schlüsselattribut abhängig sind.

2. Normalform verlangt,

- ✓ Nichtvorhandensein von Teilschlüsselabhängigkeiten

Vorgehensweise bei der Bildung der 2. Normalform

(2. Normalisierungsprozess)

1. Kennzeichnen der Teilschlüssel und der davon bereits abhängigen Nichtschlüsselattribute in der Ausgangsrelation.
2. Entfernen der gekennzeichneten Nichtschlüsselattribute und Kopieren des gekennzeichneten Teilschlüssels.
3. Bildung einer neuen Relation aus den entfernten Nichtschlüsselattributen und dem kopierten Teilschlüsselattribut. Benennung der Relation.
4. Der kopierte Teilschlüssel ist der Schlüssel der neuen Relation.

Normalisierung

Position (1NF)

PNr	<u>RNr</u>	<u>ANr</u>	AName	APreis	Anzahl	GPreis
P001	R001	A001	Computer	5.000	2	10.000
P002	R001	A002	Drucker	1.000	3	3.000
P001	R002	A002	Drucker	1.000	1	1.000
P002	R002	A003	Kabel	500	2	1.000
P001	R003	A001	Computer	5.000	1	5.000

Position (2NF)

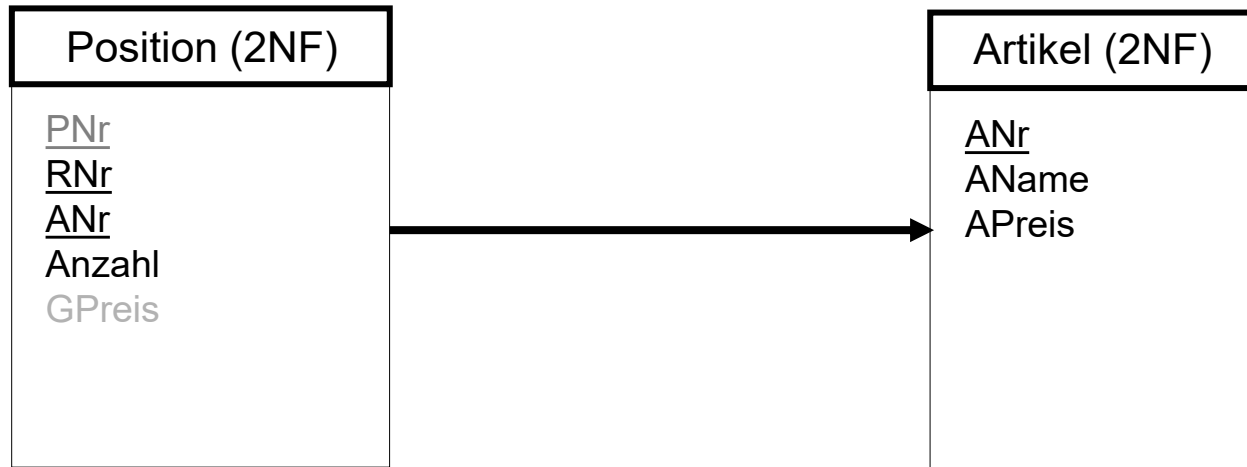
PNr	<u>RNr</u>	<u>ANr</u>	Anzahl	GPreis
P001	R001	A001	2	10.000
P002	R001	A002	3	3.000
P001	R002	A002	1	1.000
P002	R002	A003	2	1.000
P001	R003	A001	1	5.000

Artikel (2NF)

<u>ANr</u>	AName	APreis
A001	Computer	5.000
A002	Drucker	1.000
A003	Kabel	500

Normalisierung

Darstellung auf der Typ-Ebene (Relationenmodell in UML-Darstellung)



Normalisierung

Von den Wiederholungsgruppen bereinigte Ausgangsrelation

Rechnung (NF2)

RNr	Datum	Betrag	KNr	VName	ZName	Str	PLZ	Ort
R001	04.04.2004	13.000	K001	Hugo	Müller	Garten str. 4a	69123	Heidelberg
R002	05.04.2004	2.000	K001	Hugo	Müller	Garten str. 4a	69123	Heidelberg
R003	05.04.2004	5.000	K003	Georg	Mayer	Neckar str. 1	69123	Heidelberg

Eine Relation in der 2. Normalform ist immer noch für Defekte anfällig, denn obwohl sie keine Wiederholungsgruppen enthält und ihr Schlüssel skalar ist, d. h. nicht zusammengesetzt, weist sie Redundanzen im Bereich der Kundendaten auf.

In diesem Fall liegen so genannte transitive (indirekte) Abhängigkeiten vor:

Transitive Abhängigkeiten = Abhängigkeiten zwischen Nichtschlüselfeldern

Rechnungsnummer → Kundennummer → Name

Normalisierung

Definition der 3. Normalform,

Eine Relation ist dann in der dritten Normalform (3NF), wenn sie keine transitiven Abhängigkeiten aufweist.

3. Normalform verlangt,

- ✓ Nichtvorhandensein von Abhängigkeiten zwischen Nichtschlüsselattributen

Vorgehensweise bei der Bildung der 3. Normalform

(3. Normalisierungsprozess)

1. Markierung des Nichtschlüsselattributes und die davon abhängigen Nichtschlüsselattribute.
2. Entfernung der markierten Attribute aus der Ausgangsrelation.
3. Erstellung einer neuen Relation aus den entfernten Attributen.
4. Kennzeichnung des Schlüsselattributes der neuen Relation, von dem die anderen Attribute abhängig sind.
5. Benennung der neuen Relation.
6. Übertragung des Schlüsselattributes der neuen Relation als Fremdschlüssel in die bereinigte Ausgangsrelation.

Normalisierung

Von den Wiederholungsgruppen bereinigte Ausgangsrelation

Rechnung (NF2)

<u>RNr</u>	Datum	Betrag	KNr	VName	ZName	Str	PLZ	Ort
R001	04.04.2004	13.000	K001	Hugo	Müller	Garten str. 4a	69123	Heidelberg
R002	05.04.2004	2.000	K001	Hugo	Müller	Garten str. 4a	69123	Heidelberg
R003	05.04.2004	5.000	K003	Georg	Mayer	Neckar str. 1	69123	Heidelberg



Rechnung (NF3)

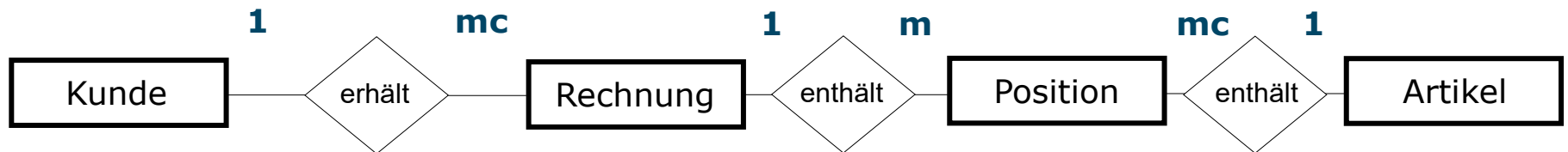
<u>RNr</u>	KNr	Datum	Betrag
R001	K001	04.04.2004	13.000
R002	K001	05.04.2004	2.000
R003	K003	05.04.2004	5.000

Kunde (NF3)

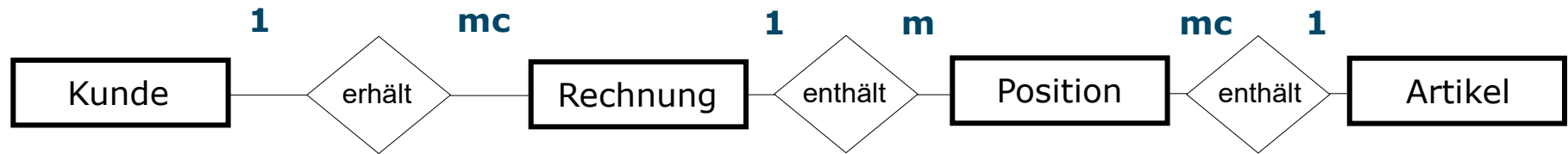
<u>KNr</u>	VName	ZName	Str	PLZ	Ort
K001	Hugo	Müller	Garten str. 4a	69123	Heidelberg
K003	Georg	Mayer	Neckar str. 1	69123	Heidelberg

Normalisierung / Normalformen

Vereinfachtes ERM-Diagramm für Rechnungsbeispiel



Normalisierung / Normalformen

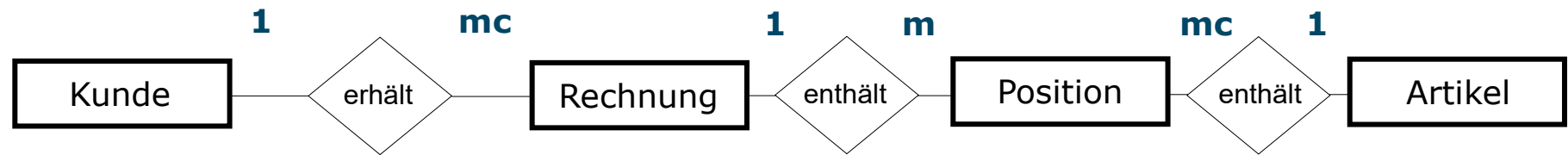


*Relation
unnormalisiert*

Rechnung

RNr
 Datum
 Betrag
 KNr
 VName
 ZName
 Str
 PLZ
 Ort
 ANr[1..*]
 AName[1..*]
 APreis[1..*]
 Anzahl[1..*]
 GPreis[1..*]

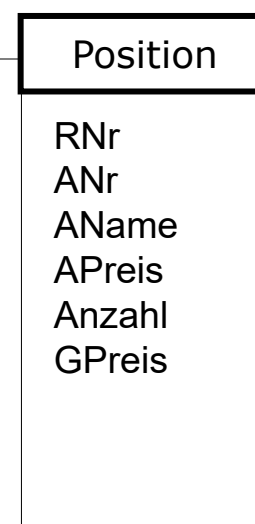
Normalisierung / Normalformen



*Relation
1 NF*

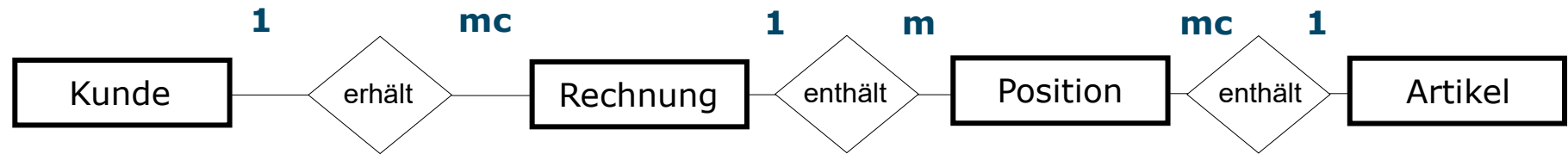


*Relation
1 NF*

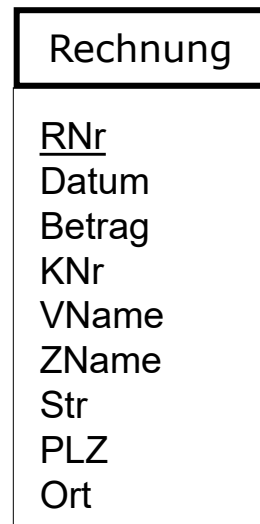


enthält

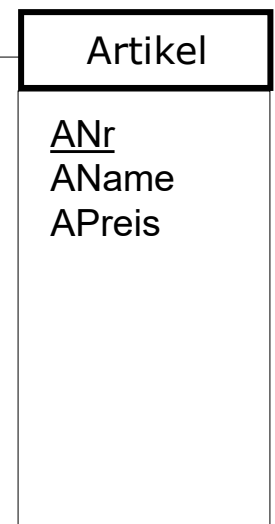
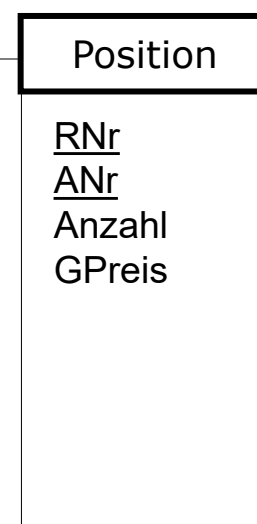
Normalisierung / Normalformen



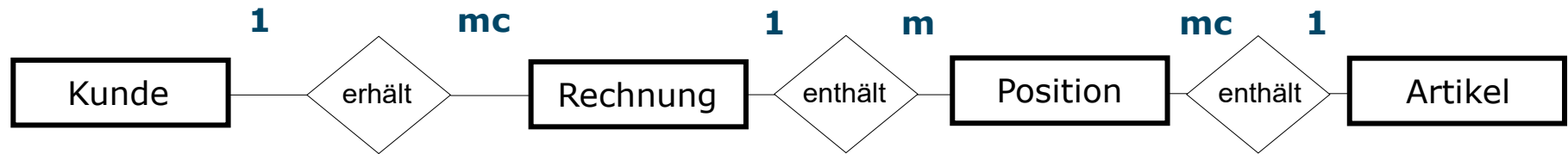
*Relation
2 NF*



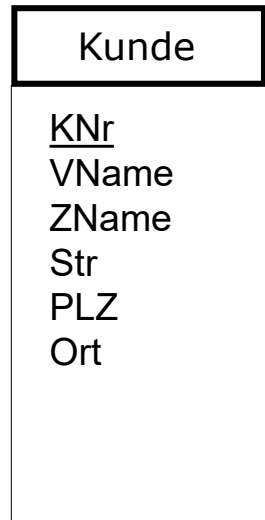
*Relation
2 NF*



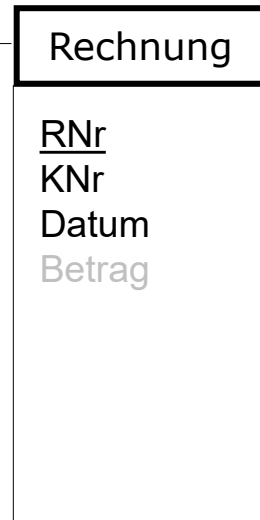
Normalisierung / Normalformen



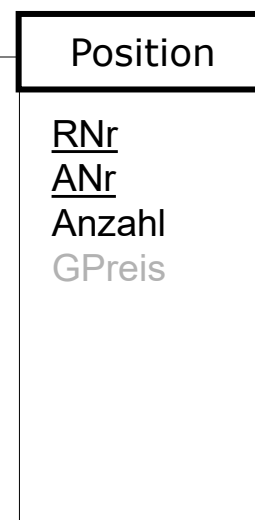
Relation
3 NF



Relation
3 NF



Relation
3 NF



Relation
3 NF



Normalisierung / Normalformen

In welcher Normalform befindet sich die Tabelle Bestellung?

Tabelle: Bestellung

<u>B ID</u>	Lfd ID	Kunden ID	<u>Artikel ID</u>	Artikelbezeichnung	Menge	Bestelldatum
1	1	12	1	Computer	5	12.02.2009
1	2	12	2	Video-Recorder	12	10.03.2009
3	3	5	1	Computer	3	08.04.2009
4	4	16	1	Computer	1	06.05.2009
5	5	17	3	Akku	12	07.06.2009

Wie wird diese Tabelle in die nächste NF überführt?

Tabelle: Bestellung

<u>B ID</u>	Lfd ID	Kunden ID	<u>Artikel ID</u>	Menge	Bestelldatum
1	1	12	1	5	12.02.2009
1	2	12	2	12	10.03.2009
3	3	5	1	3	08.04.2009
4	4	16	1	1	06.05.2009
5	5	17	3	12	07.06.2009



Tabelle: Artikel

<u>Artikel ID</u>	Artikelbezeichnung
1	Computer
2	Video-Recorder
3	Akku



Vermeidung von Anomalien: Insert / Update / Delete !!!

Fragen?

Danke für Ihre Aufmerksamkeit !

F r a g e n

