

Voraussichtlicher Vorlesungsablauf

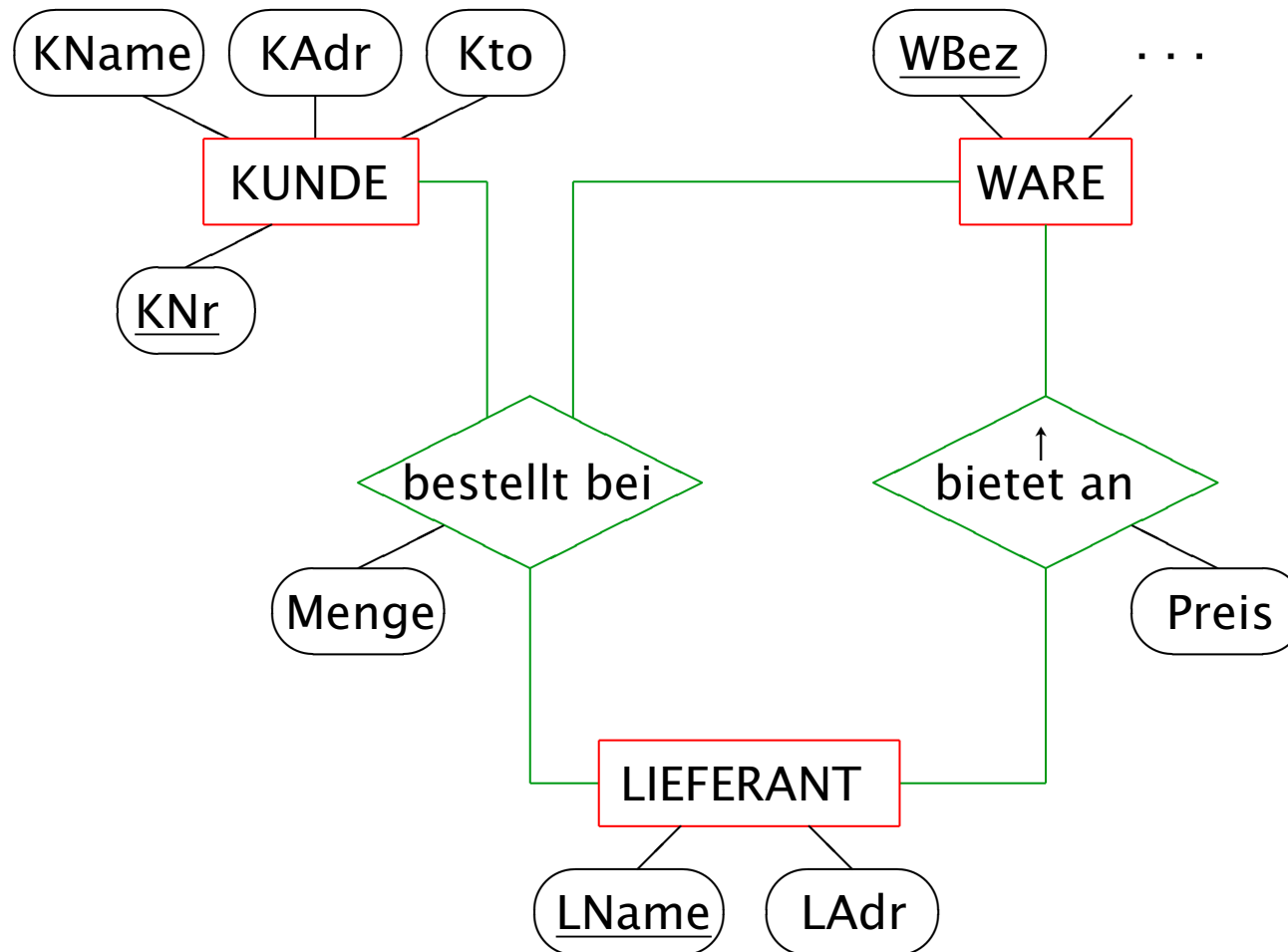
Wochen	Kapitel
(1)	1. Einführung
(1-4)	2. Datenmodellierung: Entity-Relationship-Modell Relationen-Modell
(5-8)	3. Relationale Anfragen: Relationenalgebra SQL
(9)	4. SQL-DML und -DDL
(10-11)	5. Datenbankprogrammierung: in PL/SQL (und JDBC)
(11-12)	6. Integritätssicherung
(13)	7. Weitere Datenbank-Objekte
	8. Ausblicke

Kapitel 2

Datenmodellierung

2.1 Das Entity-Relationship (ER)-Modell

Beispiel: Konzeptionelles Schema eines Warenmarktes im ER-Modell



In einer DB sollen Informationen über Kunden, Waren, Lieferanten und deren Beziehungen in einem gemeinsamen Warenmarkt gespeichert werden (z.B. Web-Portal).

Von jedem Kunden werden Name und Adresse benötigt; außerdem erhält jeder Kunde eine eindeutige Nummer und ein Konto. ...

Kunden bestellen Waren in bestimmten Mengen bei Lieferanten. Lieferanten bieten Waren zu bestimmten Preisen an.

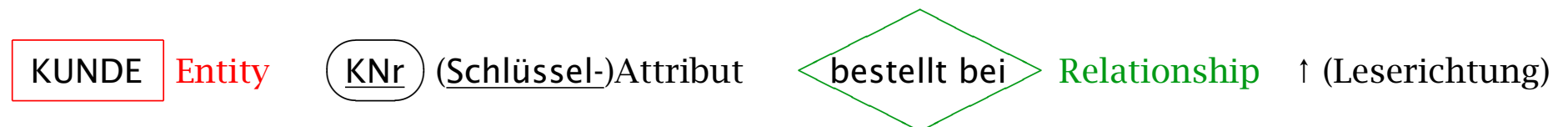
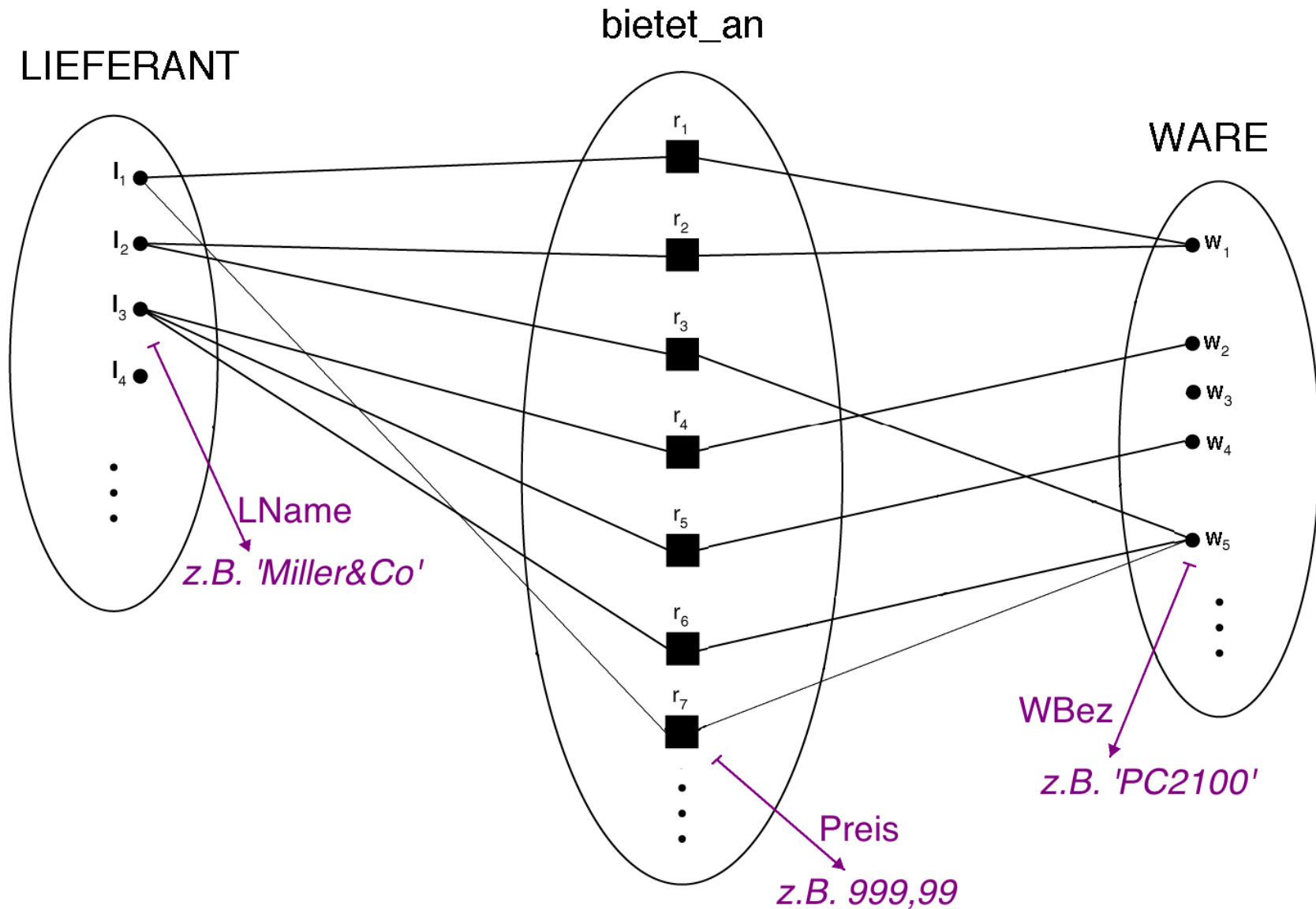
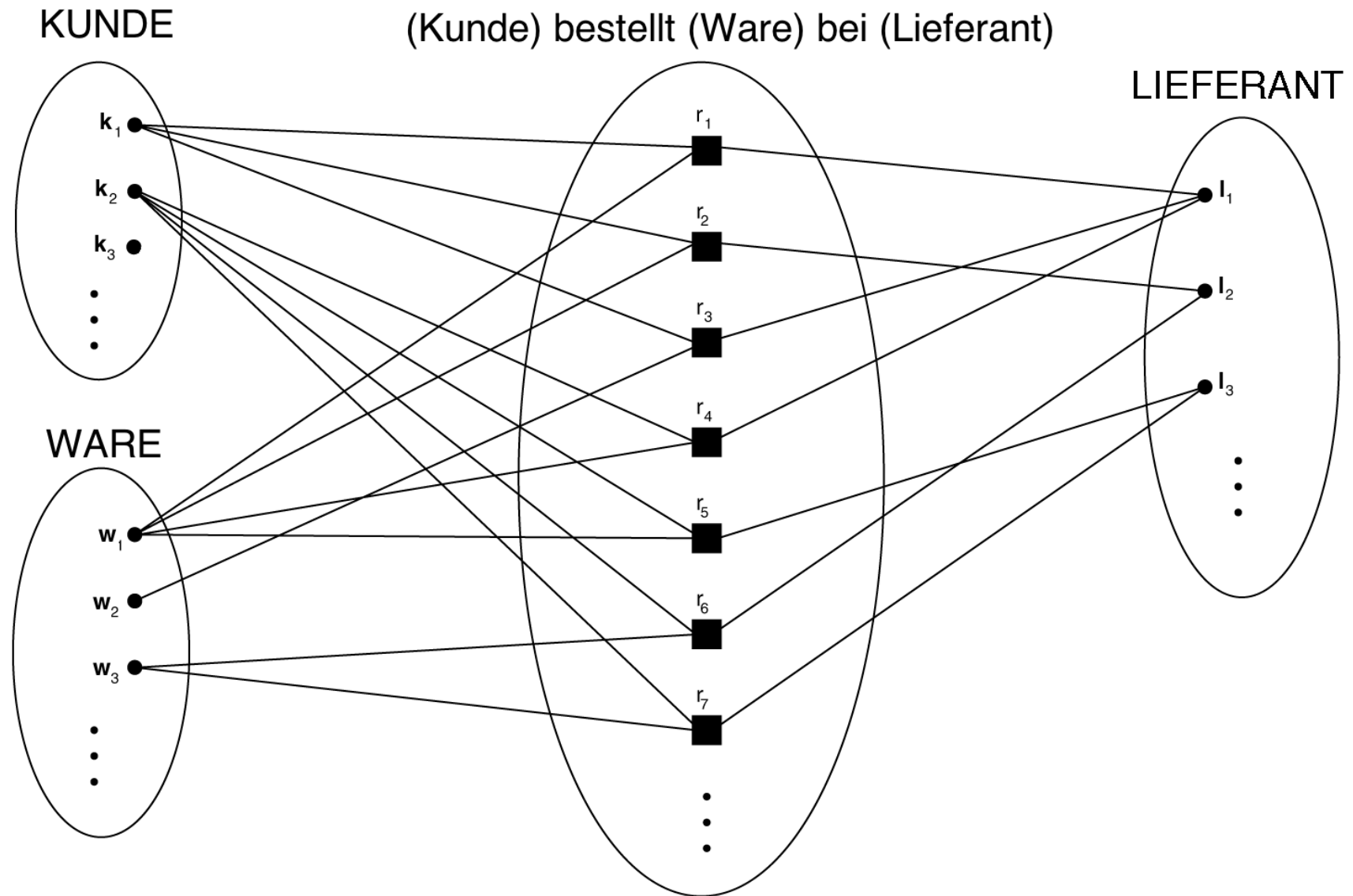


Illustration von Entites, Relationships und Attributen

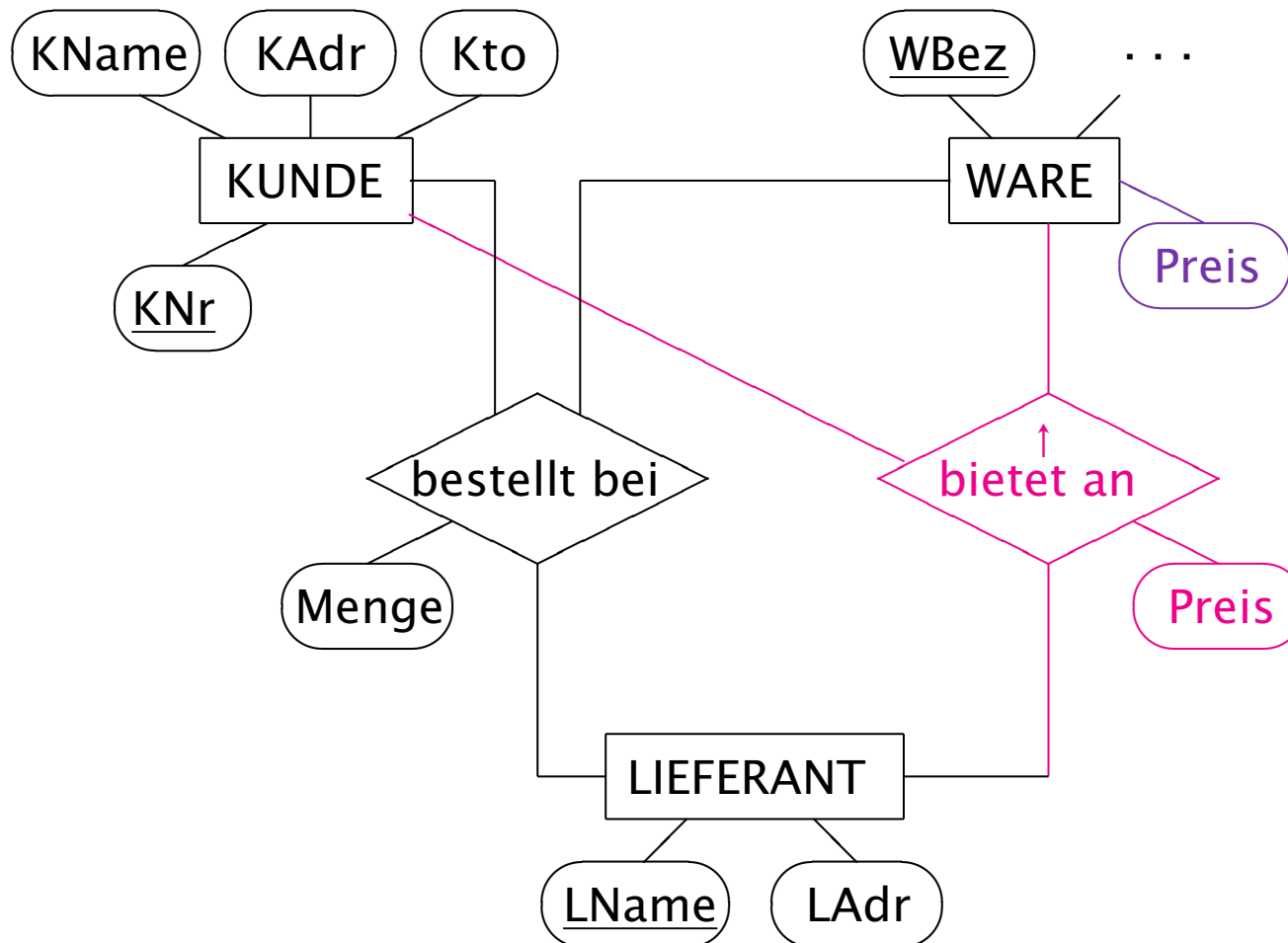


¹beachte: jede Kombination (□) von Entities darf nur einmal gebildet werden

Illustration von Relationships (Forts.)



Beispielvariationen



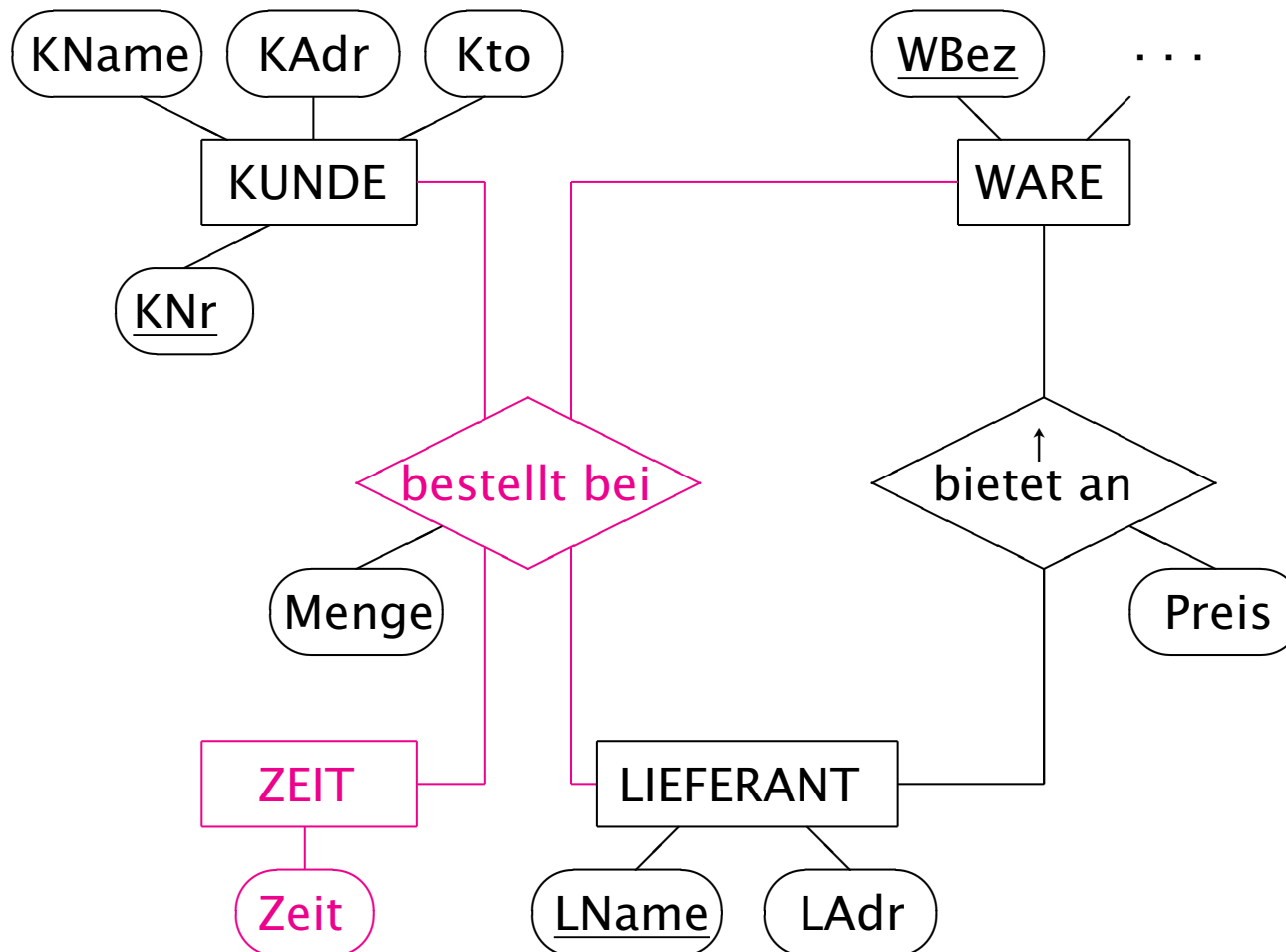
In der Miniwelt der Datenbank soll gelten:

Der Preis hängt nur von der Ware ab (wie bei der Preisbindung von Büchern.)

oder:

Lieferanten bieten ihre Waren nur manchen Kunden an. Der Preis kann abhängig von Lieferant, Ware und Kunde variieren (was individuelle Rabatte erlaubt).

Beispielvariationen (Forts.)



Die bisher entworfene DB kann nur eine Bestellung eines Kunden über eine Ware bei einem Lieferanten speichern (nur die letzte?, Mengen aufsummieren?, mehrere Bestellvorgänge verhindern?).

alternativ:

Die DB kann mehrere Bestellungen des gleichen Kunden über die gleiche Ware beim gleichen Lieferanten zu verschiedenen Zeiten mit jeweiliger Menge speichern.

Datenmodelle

Ein **Datenmodell** ist eine Sammlung von Konzepten zur Datenbeschreibung und -strukturierung. Diese bestimmen:

- die Syntax von DB-Schemata,
z. B. von Entity-Relationship-Diagrammen
- und deren Semantik, d. h. die zugehörigen DB-Zustände.

Beispiele:

- *“semantische”*² *Datenmodelle*: **Entity-Relationship-Modell (ER-Modell)**, Erweiterungen des ER-Modells, objektorientierte Modelle einschl. UML
- *implementierte Datenmodelle*:
 - *heute marktführend*: **Relationen-Modell**
 - *früher verbreitet*: Netzwerk-Modell, Hierarchisches Modell
 - *neuer*: objektorientierte und objekt-relationale Modelle

Die Aufstellung von DB-Schemata ist Gegenstand des **DB-Entwurfs**.

²“semantisch” im Sinne von: “der Anwendungssemantik näherstehend”

Modellierungskonzepte des ER-Modells

1. **Werte** = darstellbare Daten:

eingeteilt in **Datentypen** D_1, D_2, \dots

z.B. integer, string, time, ...

Semantik: (nicht zustandsabhängig)

Datentyp $D \rightsquigarrow$ (**Wertemenge** $|D|$, zugeh. Operationen)

z.B. integer \rightsquigarrow (\mathbb{Z} bzw. $[\text{minint}, \text{maxint}]$, $\{+, -, <, >, \dots\}$)

evtl. auch Nicht-Standard-Datentypen wie

z.B. point (Punkt im Koordinatensystem), interval, rectangle (Rechteck),

line (Linienzug), region (Fläche), image (Rasterbild), ...

Modellierungskonzepte des ER-Modells (Forts.)

2. Objekte (Entities):

Ein **Objekt** ist ein individuelles und identifizierbares Exemplar von Dingen, Personen oder Begriffen der realen oder der Vorstellungswelt, über das Informationen zu speichern sind.

eingeteilt in **Objekttypen** E_1, E_2, \dots

z. B. KUNDE, LIEFERANT, WARE, ...

Graphische Notation:

E

Semantik:

Objekttyp $E \rightsquigarrow$


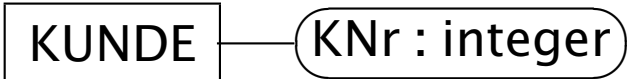
- $\mu(E)$ = Menge der möglichen Objekte (vom Typ E)
- $\zeta(E)$ = Menge der aktuellen Objekte vom Typ E im Zustand ζ ,
so dass $\zeta(E) \subseteq \mu(E)$ und $\zeta(E)$ endlich

Modellierungskonzepte des ER-Modells (Forts.)

3. (Objekt-) **Attribute** = Eigenschaften von Objekten

eingeteilt in **Attributtypen** $A_1, A_2 \dots$

Zu jedem Attribut[typ] A gehören ein Objekttyp E u. ein Datentyp D.

Notation:  z.B.: 

(Standard-Datentypen werden in ER-Diagrammen meist weggelassen.)

Semantik (im Zustand ζ):

Attribut A wie oben \rightsquigarrow **Funktion** $\zeta(A): \zeta(E) \rightarrow |D|$

z. B. $\zeta(\text{KNr}): \zeta(\text{KUNDE}) \rightarrow \mathbb{Z}$

Jedem Entity e vom Typ E wird genau ein Attributwert A vom Typ D zugeordnet.

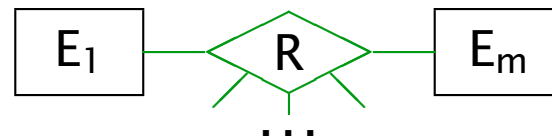
Modellierungskonzepte des ER-Modells (Forts.)

4. **Beziehungen (Relationships)** zwischen Objekten:

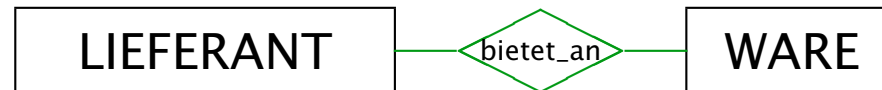
eingeteilt in **Beziehungstypen** R_1, R_2, \dots

Zu jedem m-stelligen Beziehungstyp R gehören m Objekttypen E_1, \dots, E_m .

Notation:



z.B.:



Semantik (im Zustand ζ):

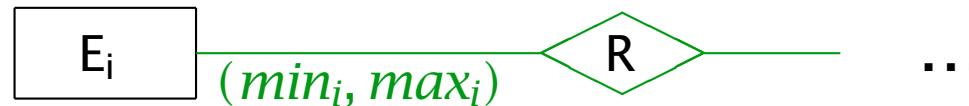
R wie oben \rightsquigarrow **Relation** $\zeta(R) \subseteq \zeta(E_1) \times \dots \times \zeta(E_m)$

z. B. $\zeta(\text{bietet_an}) \subseteq \zeta(\text{LIEFERANT}) \times \zeta(\text{WARE})$

Ein Relationship r vom Typ R entspricht einem Tupel (e_1, \dots, e_m) von Entities der Typen E_1, \dots, E_m .

Modellierungskonzepte des ER-Modells (Forts.)

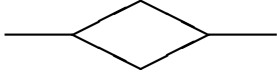
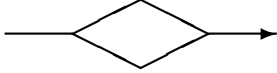
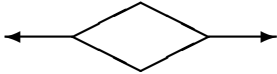
Zusatzangaben an Kanten: **Kardinalitäten**



Für alle $e_i \in \zeta(E_i)$ gilt: $min_i \leq |\{ r \in \zeta(R) \mid r = (\dots, e_i, \dots) \}| \leq max_i$

D.h.: Jedes E_i -Objekt ist an min_i bis max_i R -Beziehungen beteiligt.

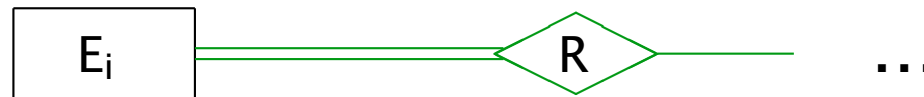
Voreinstellung: $(0,*)$, d. h. keine Beschränkung

Ausgewählte binäre Beziehungsarten	(min_1, max_1)	(min_2, max_2)	(alternative) graphische Darstellung
allgemein ("n:m")	frei wählbar	frei wählbar	(1)  (2)
funktional ("n:1")	$max_1 = 1$	frei wählbar	
"1:1"	$max_1 = 1$	$max_2 = 1$	

Modellierungskonzepte des ER-Modells (Forts.)

Alternative Notation: **Totale Beteiligung** (an einer Beziehung):

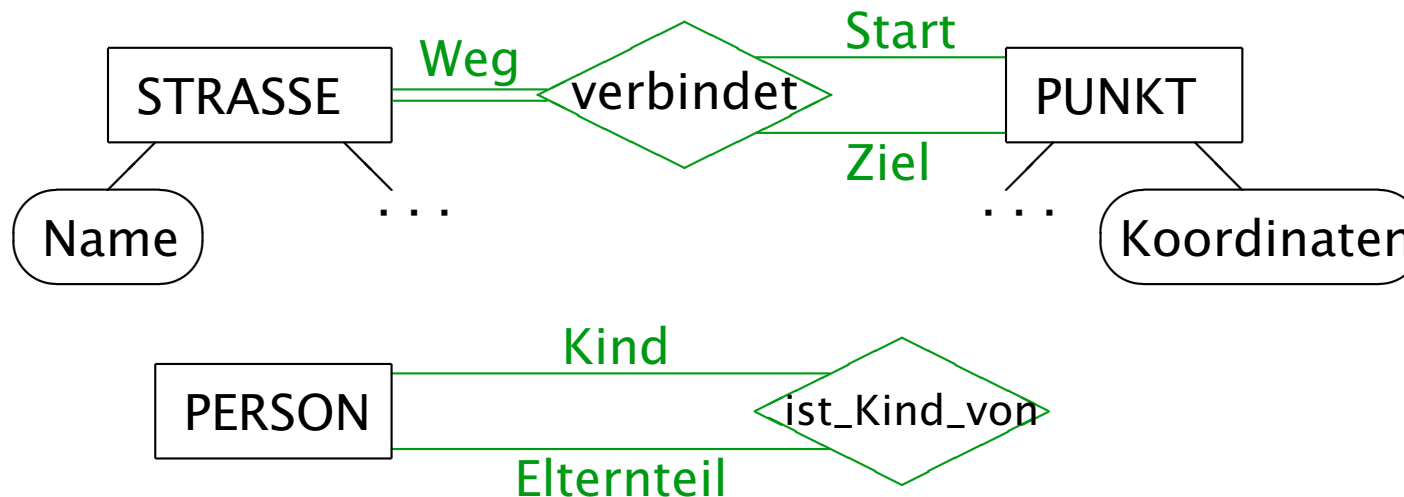
Eine Mindestkardinalität von 1 kann auch graphisch notiert werden:



Die E_i -Objekte heißen dann auch **existenzabhängige** Objekte, weil sie nur existieren können, wenn Beziehungspartner existieren.

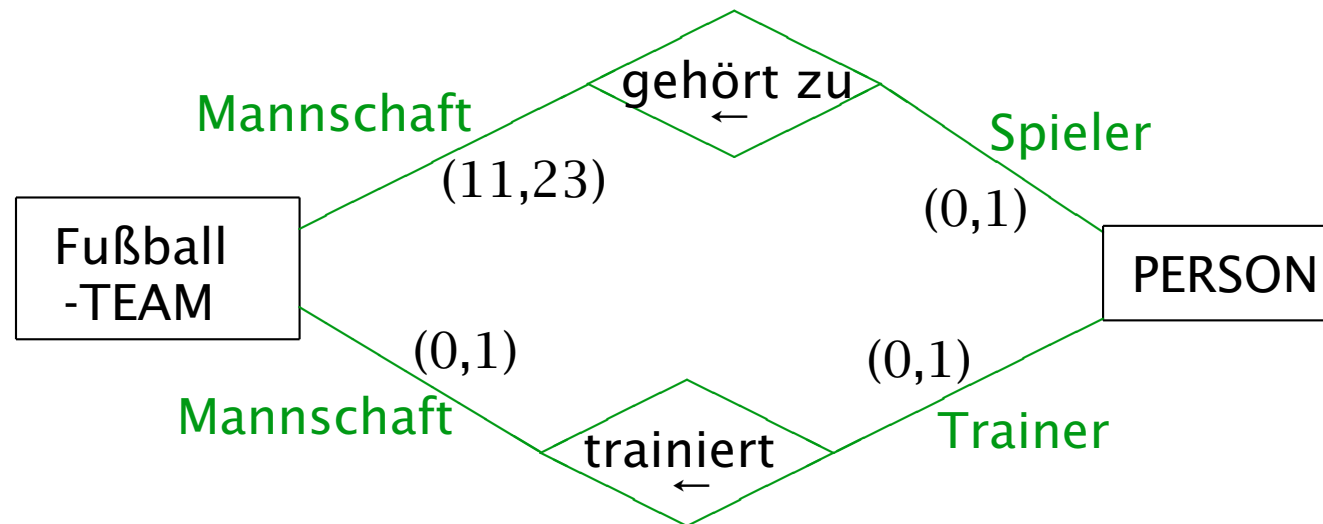
Weitere Zusatzangaben: **Rollen**,

d. h. Namen für die an Beziehungen teilnehmenden Entities, z. B.:



Modellierungskonzepte des ER-Modells (Forts.)

Weiteres Beispiel:



Modellierungskonzepte des ER-Modells (Forts.)

5. **Beziehungsattribute:** analog zu Objektattributen

z.B. **Preis** als Attribut zum Beziehungstyp `bietet_an`
zwischen `LIEFERANT` und `WARE`: `number(6,2)`

Semantik: **Funktion** $\zeta(\text{Preis})$: $\zeta(\text{bietet_an}) \rightarrow |\text{number}(6, 2)|$

Jedem Relationship wird genau ein Attributwert zugeordnet.

Ein Beziehungsattribut wird benötigt, wenn die Attributwerte nicht von den beteiligten Entities einzeln abhängen (z.B. hängt der Preis nicht nur von betrachteten `WARE` ab), sondern von der Beziehung, also von allen daran beteiligten Entities (hier `WARE` und `LIEFERANT`).

Modellierungskonzepte des ER-Modells (Forts.)

6. **Schlüssel:** zur (sichtbaren) Identifizierung von Objekten

z. B. KUNDE (KNr, KName, KAdr, Kto)

MdB (Name, Ort, Vorname, Partei, ...)

aber nicht AUTO (Farbe) !

Def.: Gegeben sei ein Objekttyp E mit Attributen A_1, \dots, A_n .

Eine Teilmenge $\{B_1, \dots, B_k\} \subseteq \{A_1, \dots, A_n\}$ heißt genau dann

Schlüssel von E, wenn

- (i) sich beliebige verschiedene Objekte $e, e' \in \mu(E), e \neq e'$, jeweils in mindestens einem Attribut B_j unterscheiden³
- (ii) und $\{B_1, \dots, B_k\}$ eine *minimale* Menge mit der Eigenschaft (i) ist.

Zu jedem Objekttyp muss ein Schlüssel angegeben werden. In ER-Diagrammen werden das Schlüsselattribut bzw. alle Attribute einer Schlüsselattributkombination durch Unterstreichen gekennzeichnet⁴.

³Idealerweise sollten sich Schlüsselattributwerte nicht von Zustand zu Zustand ändern.

⁴Oft wird eine systemvergebene, eindeutige, unveränderliche, nicht wiederverwendbare Nummerierung verwendet. Mehrere alternative Schlüsselkandidaten kennzeichnen wir in unseren ER-Diagrammen nicht.

Modellierungskonzepte des ER-Modells (Forts.)

Durch die Aufstellung eines DB-Schemas, hier eines ER-Diagramms, also durch die Strukturen und weiteren Angaben des Datenmodells, sind schon vielfältige Bedingungen (**inhärente Integritätsbedingungen**) festgelegt, die die Zustände der zukünftigen DB erfüllen müssen. Z.B. gilt für den Warenmarkt aufgrund der Modellierung von Preis als Beziehungsattribut:

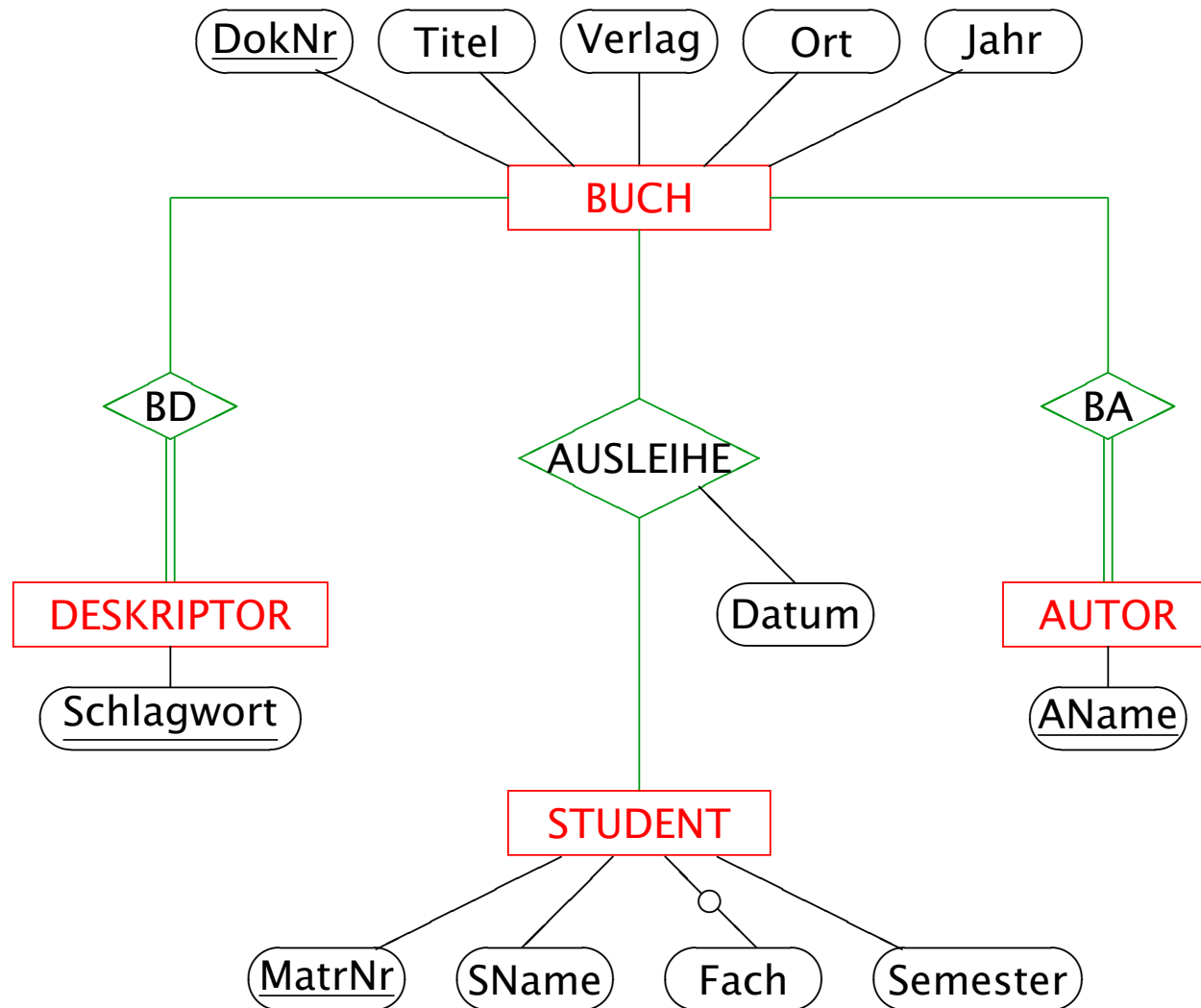
Jeder Lieferant bietet jede Ware zu genau einem Preis an.

Damit wären kundenabhängige Rabatte in der DB nach dem Schema von S. 2.1 nicht darstellbar; würde das gewünscht, wäre der Entwurf zu revidieren.

Oft reichen die Modellierungskonzepte eines Datenmodells nicht aus, um die in der DB möglichen Zustände an die in der Realität (bzw. “Mini-Welt”) möglichen Situationen anzupassen. Dann muss das Schema um **explizite Integritätsbedingungen** (IBen) erweitert werden, um Zustände auf **zulässige** Zustände einzuschränken, z.B. für den Warenmarkt:

Jede bei einem Lieferanten bestellte Ware muss von diesem auch angeboten werden.

Weiteres Beispiel: ER-Schema für eine Bibliotheks-DB



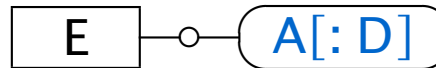
Schlagworte und Autorennamen werden nur zu (in der DB vorhandenen) Büchern gespeichert.

Pro Buch/Student-Kombination wird max. eine Ausleihe gespeichert. Welches Datum meint der Entwerfer?

*Mögliche explizite IB:
Das Datum einer Ausleihe liegt nicht vor dem Erscheinungsjahr des zugehörigen Buches.*

Erweiterungen des ER-Modells

- **Optionale Attribute:**

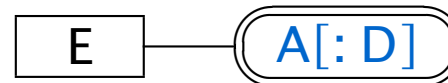


z. B. zum Objekttyp STUDENT das Attribut **Fach**

Semantik: Funktion⁵ mit Wertebereich $|D|^{\perp} := (|D| \cup \{\perp\})$

d. h. der Datentyp wird als um einen **Nullwert** \perp erweitert angenommen ('undefiniert', engl. *null*; kann auch für 'unbekannt' oder 'unzutreffend' stehen) ⁶.

- **Mehrwertige Attribute:**

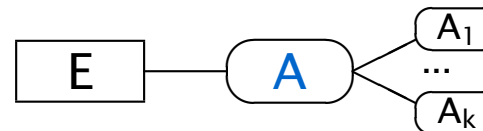


z. B. zum Objekttyp BUCH das Attribut **Schlagwort**

Semantik: Funktion mit Wertebereich $\mathcal{P}_{\text{endlich}}(|D|)$

d. h. Werte sind endliche Teilmengen von $|D|$.

- **Zusammengesetzte Attribute:**



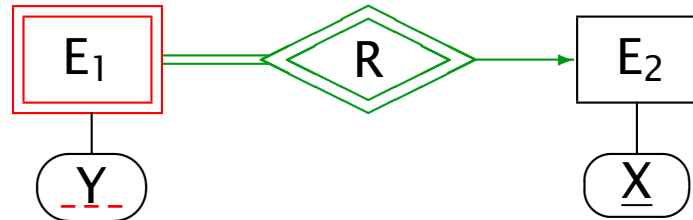
z. B. Attribut **Adresse** (mit Teilattributen PLZ, Ort, Straße, Hausnr) zu KUNDE

⁵eigentlich eine "partielle Funktion" $\zeta(A): \zeta(E) \rightarrow |D|$

⁶Nullwerte erfordern bei Verknüpfungen spezielle Rechenregeln, z. B. $\perp + 3 = \perp$

Erweiterungen des ER-Modells (Forts.)

- **Schwache Entities** und zugehörige **identifizierende Beziehungen**:



Die Objekte des Typs E_1 können nur durch eigene Attribute (Y) zusammen mit der Angabe des zugeordneten E_2 -Objekts (bzw. dessen Schlüsselattribute X) identifiziert werden. Die Beziehung R ist quasi am Schlüssel von E_1 beteiligt. Es kann auch mehrere identifizierende Beziehungen geben.

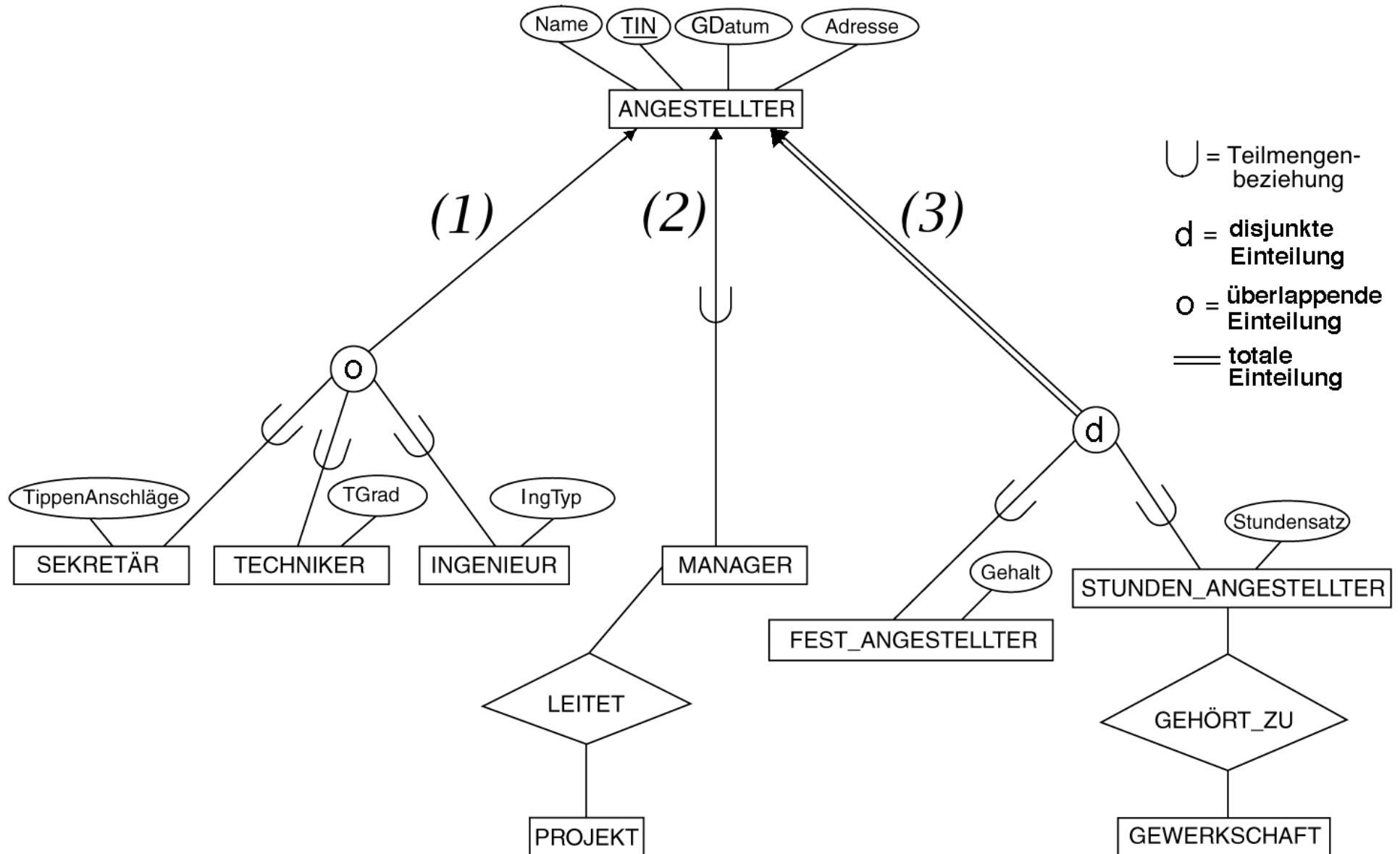
Die mitidentifizierenden E_1 -lokalen Attribute (Y) bilden einen “partiellen Schlüssel” und werden nur durch gestrichelte Unterstreichung gekennzeichnet !!!

Eine identifizierende Beziehung R muss, falls binär (wie im Bild), total und funktional sein. Äquivalent bzw. auf mehrstellige Beziehungen verallgemeinerbar ist die Anforderung, dass R auf der Seite des schwachen Objekttyps die Kardinalität (1,1) tragen muss.

z. B. $E_1 = \text{FILIALE}$, $\underline{Y} = \underline{\text{Ort}}$ — $R = \text{von}$ — $E_2 = \text{FIRMA}$, $\underline{X} = \underline{\text{FName}}$

Erweiterungen des ER-Modells (Forts.)

- **Spezialisierungen** bzw. **Subklassen-Beziehungen** (am Beispiel)



Erweiterungen des ER-Modells: Spezialisierungen (Forts.)

Erläuterungen:

Spezialisierungsbeziehungen sind semantisch (Gruppen von) Teilmengenbeziehungen. Zu den hier dargestellten Arten von Einteilungen:

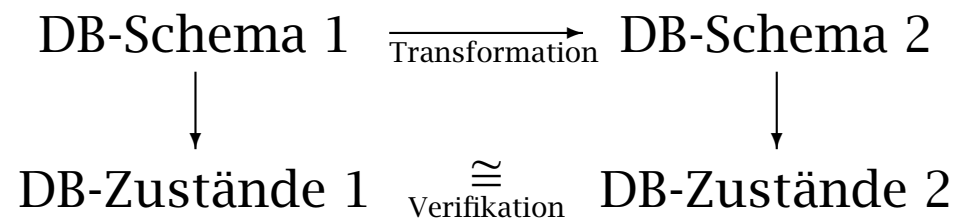
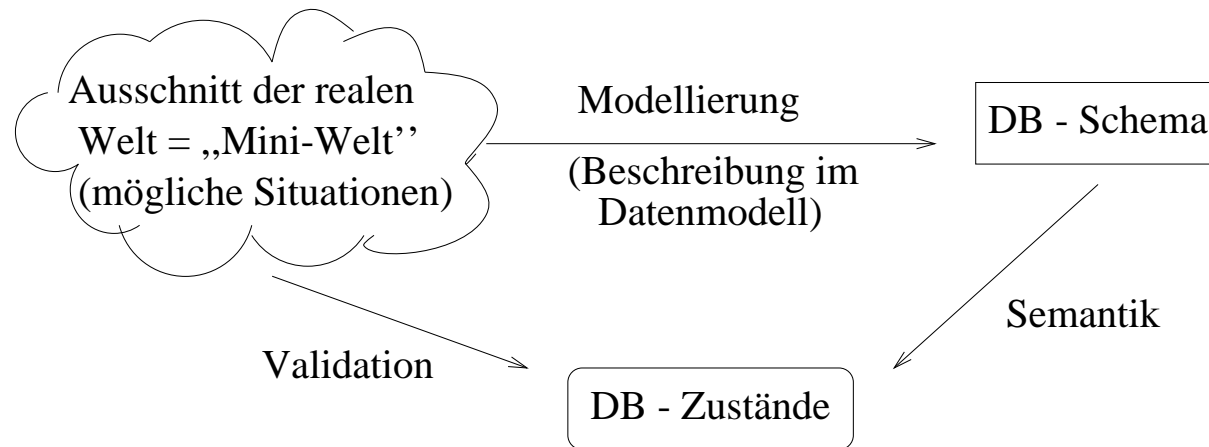
- (1) Angestellte können tätig sein als Sekretär, Techniker oder Ingenieur oder als etwas anderes (nicht totale Einteilung) oder als mehreres davon (überlappende Einteilung). Zu den explizit genannten Spezialisierungen werden spezifische Attribute festgehalten. [wäre auch durch mehrere einfache Spezialisierungen darstellbar]*
- (2) Einige Angestellter sind Manager (einfache Spezialisierung, nicht total); nur diese leiten evtl. Projekte.*
- (3) Jeder Angestellter ist entweder Fest- oder Stundenangestellter (totale, disjunkte Einteilung), wiederum mit spezifischen Attributen und Relationships.*

Spezialisierungen (Subklassen) eines Entity-Typs brauchen nicht unbedingt eigene Schlüssel; sie erben implizit den Schlüssel ihrer Superklasse, hier “TIN” (*Steuer-Identifikations-Nummer*). Aber sie sind prädestiniert, eigene Attribute und Relationships zu tragen, die nicht für alle Objekte der Oberklasse sinnvoll sind.

Bitte schreiben Sie keine Kardinalitäten an solche Einteilungen (können verwirren). Die Pfeile deuten (redundant) an, dass Teilmengenbeziehungen spezielle “ist-ein”-funktionale Beziehungen sind.

Wozu eine mathematische Semantik von Datenmodellen ?

Es sollen möglichst rigorose **Validationen** von Schemata und **Verifikationen** von Schema-Transformationen erleichtert werden:



Zwei Schemata heißen **äquivalent** (\cong) gdw. sich die zugehörigen Mengen von Zuständen bijektiv aufeinander abbilden lassen.

2.2 Datenbank-Entwurf: Phasen, Beispiele, Methoden, Kriterien

A) Informationsbedarfsanalyse

Anforderungssammlung
(*natürlichsprachlich, tabellarisch,
evtl. schon graphisch*)

Beschreiben der Informationen, die durch das DBS verwaltet werden sollen (u.a. aus existierenden Dokumenten/Programmen /Dateien und Interviews)

B) Konzeptioneller Entwurf (Objektmodellierung)

konzeptionelles DB-Schema,
z. B. EER-Diagramm

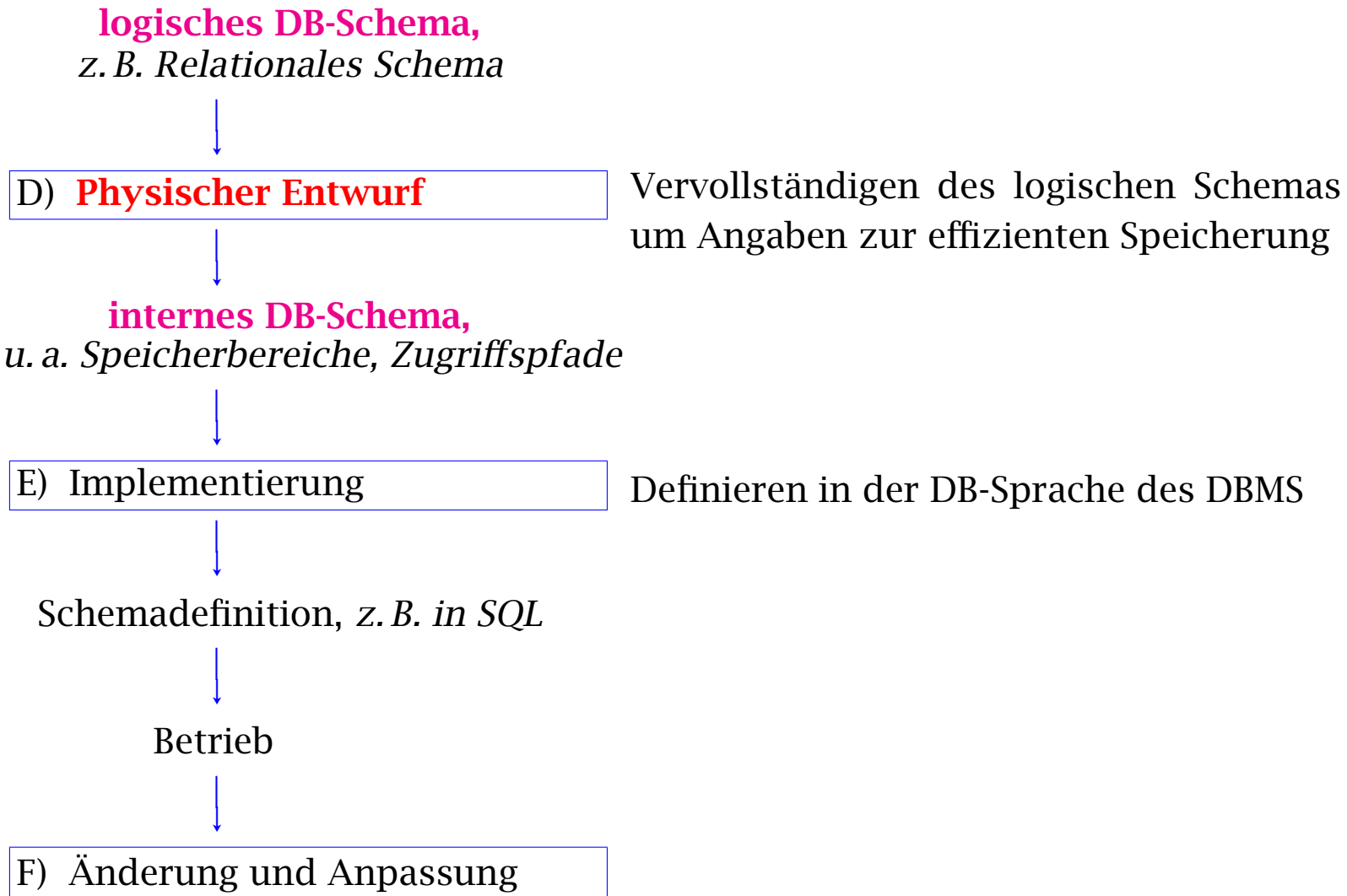
Spezifizieren von DB-Inhalten (und DB-Entwicklungen) mit einem “semantischen” Datenmodell, noch unabhängig vom implementierten Datenmodell/ DBMS

C) Logischer Entwurf (Datenmodellierung)

logisches DB-Schema,
z. B. Relationales Schema

Abbilden des konzeptionellen Schemas auf ein “implementiertes” Datenmodell, möglichst noch unabhängig vom DBMS

Entwurfsphasen (Forts.)



Zum konzeptionellen Entwurf

Die Hauptaufgabe dieser Entwurfsphase ist es, alle in den Anforderungsdokumenten gesammelten Informationsanforderungen in eine einheitliche *Spezifikation*, das unternehmensweite “Datenmodell” bzw. das **konzeptionelle DB-Schema**, zu integrieren, und damit für alle Anwendungen eine einheitliche Schnittstelle für den Zugriff auf die DB zu schaffen.

Eine DB muss den aus der Sicht der zu bedienenden Anwendungen relevanten Realweltausschnitt (“Mini-Welt”) abdecken.

Dafür braucht man als Modellierungskonzepte Ausdrucksmittel, mit denen die Realwelt so vereinfacht dargestellt werden kann,

- dass die resultierende Darstellung **abstrakt** genug ist, um unwichtige Details weglassen bzw. verstecken zu können,
- aber **ausdrucksstark** genug ist, um alle erwünschten Merkmale und Beziehungen ausdrücken zu können.

Gleichzeitig sollte eine **möglichst direkte Transformation in implementierte Datenmodelle** unterstützt werden (→ Logischer Entwurf, vgl. Abschnitt 2.3).

Das **erweiterte ER- (EER-) Modell** ist im Sinne aller dieser Kriterien geeignet.

Beispiel 1:

Informationsanforderungen an eine Bestellabwicklungs-DB

Von einem Handelsunternehmen gehen (Bestell-)Aufträge an Lieferanten. Aufträge enthalten Auftragspositionen, die wiederum auf genau einen bestellten Artikel verweisen. Nach Durchführung der Lieferung legt der Lieferant eine Rechnung, die aus unterschiedlichen Rechnungspositionen besteht. In der hier verwendeten vereinfachten Darstellung sind Aufträge durch eine eindeutige Auftragsnummer und durch das Auftrags-Datum beschrieben. Jede Auftragsposition bezeichnet den bestellten Artikel über seine eindeutige Nummer, die Artikel-Bezeichnung und seinen Preis. Des weiteren enthält jede Auftragsposition noch die Bestellmenge. Über Lieferanten soll in der Datenbank der Name, eine fortlaufende Nummer als Identifikator, ihre Anschrift und ihre Kontonummer gespeichert werden. Rechnungen haben eine eindeutige Rechnungsnummer, tragen ein Datum, weisen einen Endbetrag und einen Steuersatz auf und geben eine Zahlungsart vor. Jede Rechnungsposition muss einer Auftragsposition entsprechen. Über die bestellten Artikel soll die Artikelnummer, ihre Bezeichnung und der Standort des Artikels im Lager gespeichert werden.

Beispiel 1 (Forts.)

Konzeptioneller Entwurf: Sprachliche Analyse der Inf'anforderungen

Von einem Handelsunternehmen **gehen** (**Bestell-**)Aufträge **an** Lieferanten. **Aufträge** **enthalten** Auftragspositionen, die wiederum **auf** genau einen bestellten Artikel **verweisen**. Nach Durchführung der Lieferung "**legt**" der **Lieferant** eine Rechnung, die aus unterschiedlichen Rechnungspositionen **besteht**.

In der hier verwendeten vereinfachten Darstellung sind **Aufträge** durch eine eindeutige Auftragsnummer und durch das Auftrags-Datum beschrieben.

Jede **Auftragsposition** bezeichnet den bestellten **Artikel** über seine eindeutige Nummer, die Artikel-Bezeichnung und seinen Preis. Des weiteren enthält jede Auftragsposition noch die Bestellmenge.

Über **Lieferanten** soll in der Datenbank der Name, eine fortlaufende Nummer als Identifikator, ihre Anschrift und ihre Kontonummer gespeichert werden.

Rechnungen haben eine eindeutige Rechnungsnummer, tragen ein Datum, weisen einen Endbetrag und einen Steuersatz auf und geben eine Zahlungsart vor.

Jede **Rechnungsposition** muss einer **Auftragsposition** **entsprechen**.

Über die bestellten **Artikel** soll die Artikelnummer, ihre Bezeichnung und der Standort des Artikels im Lager gespeichert werden.

Beispiel 1 (Forts.)

Konzeptioneller Entwurf: Identifikation von Entities, Attr'en, Rel'ships

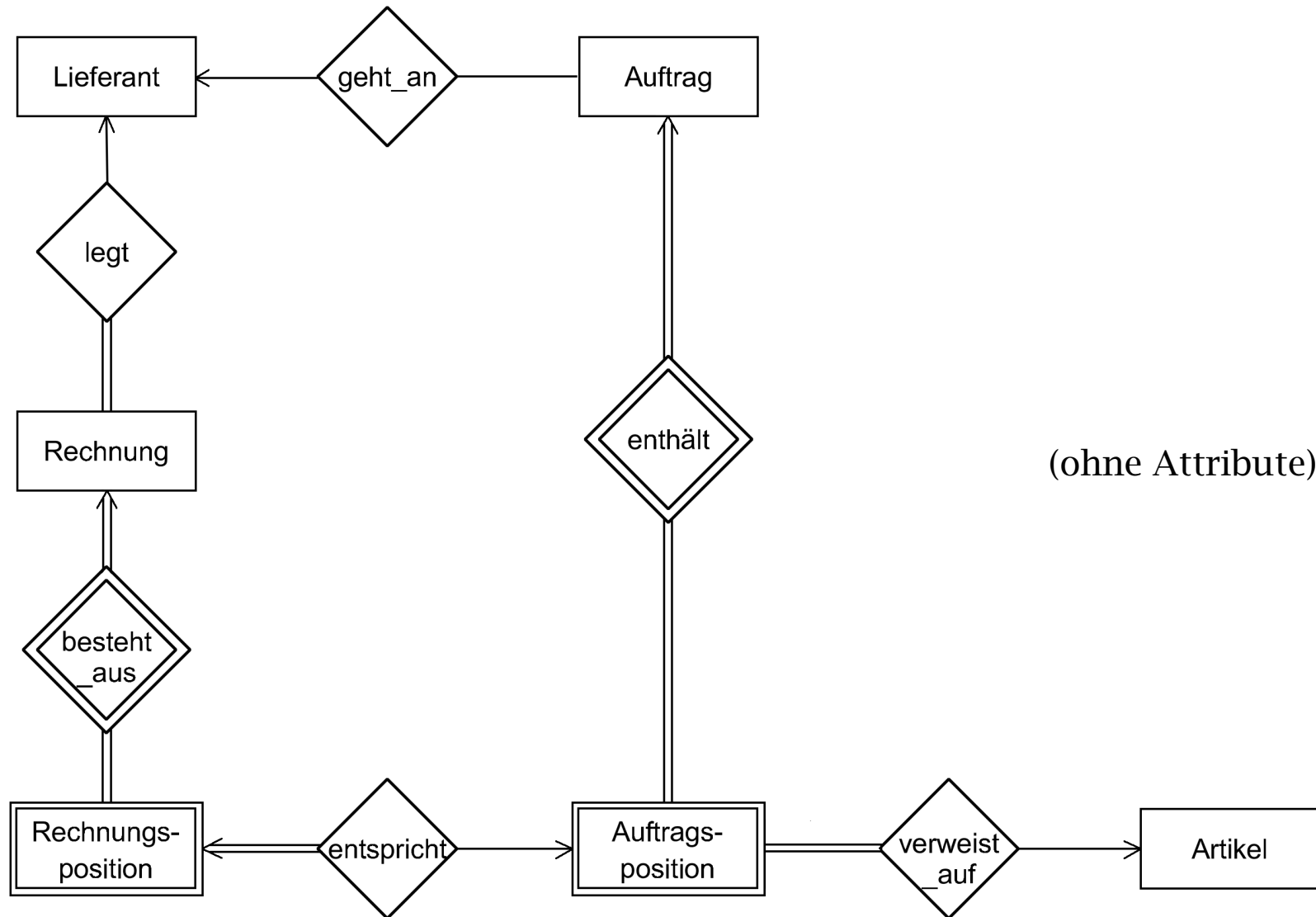
Substantive \rightsquigarrow Kandidaten für *Entitytypen* und *Attribute*:

Auftrag:	Auftrags-Nr, Auftrags-Datum
Auftragsposition:	Artikel-Nr, Artikel-Bezeichnung , Menge, Preis
Lieferant:	Lief-Nr, Lief-Name, Anschrift, Konto-Nr
Rechnung:	Re-Nr, Re-Datum, Re-Betrag, Steuersatz, Zahlungsart
Rechnungsposition:	<i>⟨wie Auftragsposition⟩</i>
Artikel:	Artikel-Nr, Artikel-Bezeichnung, Standort

Verben \rightsquigarrow Kandidaten für *Beziehungstypen*:

“geht_an”	Auftrag : Lieferant (funktional \rightarrow)
“enthält”	Auftrag : Auftragsposition (funktional \leftarrow)
“verweist_auf”	Auftragsposition : Artikel (funktional \rightarrow)
“legt”	Lieferant : Rechnung (funktional \leftarrow)
“besteht aus”	Rechnung : Rechnungsposition (funktional \leftarrow)
“entspricht”	Rechnungsposition : Auftragsposition (1:1)

Beispiel 1 (Forts.): EER-Schema der Bestellabwicklungs-DB



Beispiel 1 (Forts.):

Explizite Integritätsbedingungen an die Bestellabwicklungs-DB


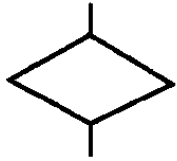




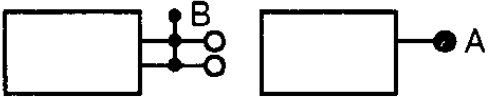
- umgangssprachlich: *Rechnungen werden von dem Lieferanten gelegt, der den zugehörigen Auftrag bekommen hat.*
- genauer: *Für beliebige Aufträge A, Lieferanten L und Rechnungen R gilt: Wenn ein Auftrag A an einen Lieferanten L geht, und wenn eine Rechnung R aus Rechnungspositionen besteht, die Auftragspositionen aus dem Auftrag A entsprechen, dann wird die Rechnung R von dem gleichen Lieferanten L gelegt.*
- äquivalent: *Wenn eine Auftragsposition zu einem Auftrag an einen Lieferanten L gehört, dann gehört eine dazu entsprechende Rechnungsposition zu einer Rechnung des gleichen Lieferanten L.*
- *Das Datum einer Rechnung kann nicht vor dem Datum eines über entsprechende Positionen verbundenen Auftrags liegen.*
- *Die Menge einer Rechnungsposition muss kleiner/gleich der Menge der entsprechenden Auftragsposition sein; ebenso der Preis.*
- *Der Betrag einer Rechnung ergibt sich als Summe der Menge*Preis-Produkte aller Rechnungspositionen, erhöht um den Steuersatz der Rechnung.*

— zur Ergänzung der Vorlesung —

Exkurs: Alternative ER-Notation

[nach Batini/ Ceri/
Navathe 1992]

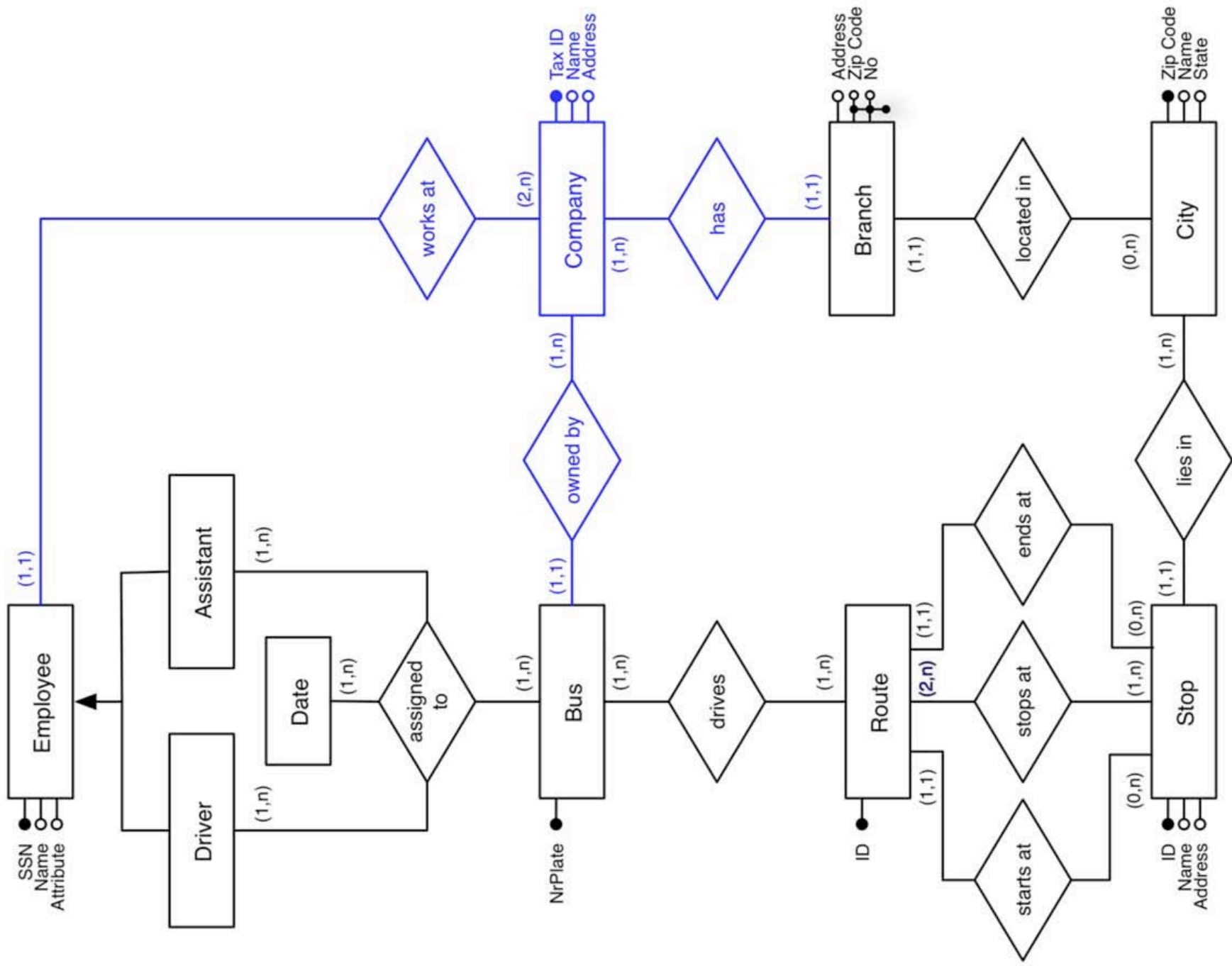
Die folgenden Beispiele
(aus dem gleichen Buch)
verwenden diese leicht
modifizierte Notation.

Entity	
Relationship	
Attribute	
Composite attribute	
Generalization hierarchy	
Subset	
Identifier	

Beispiel 2: Informationsanforderungen an eine Busunternehmens-DB

Design a database system for managing information about routes supported by a bus company. Each route served by the company has a starting place and an ending place, but it can go through several intermediate stops. The company is distributed over several branches. Not all the cities where the buses stop have a branch; however, each branch must be at a city located along the bus routes. There can be multiple branches in the same city and also multiple stops in the same city. One bus is assigned by the company to one route; some routes can have multiple buses. Each bus has a driver and an assistant, who are assigned to the bus for the day.

→ auf Folgeseite: ausgearbeitetes EER-Diagramm
für ein (oder mehrere) Busunternehmen



Beispiel 2 (Forts.):

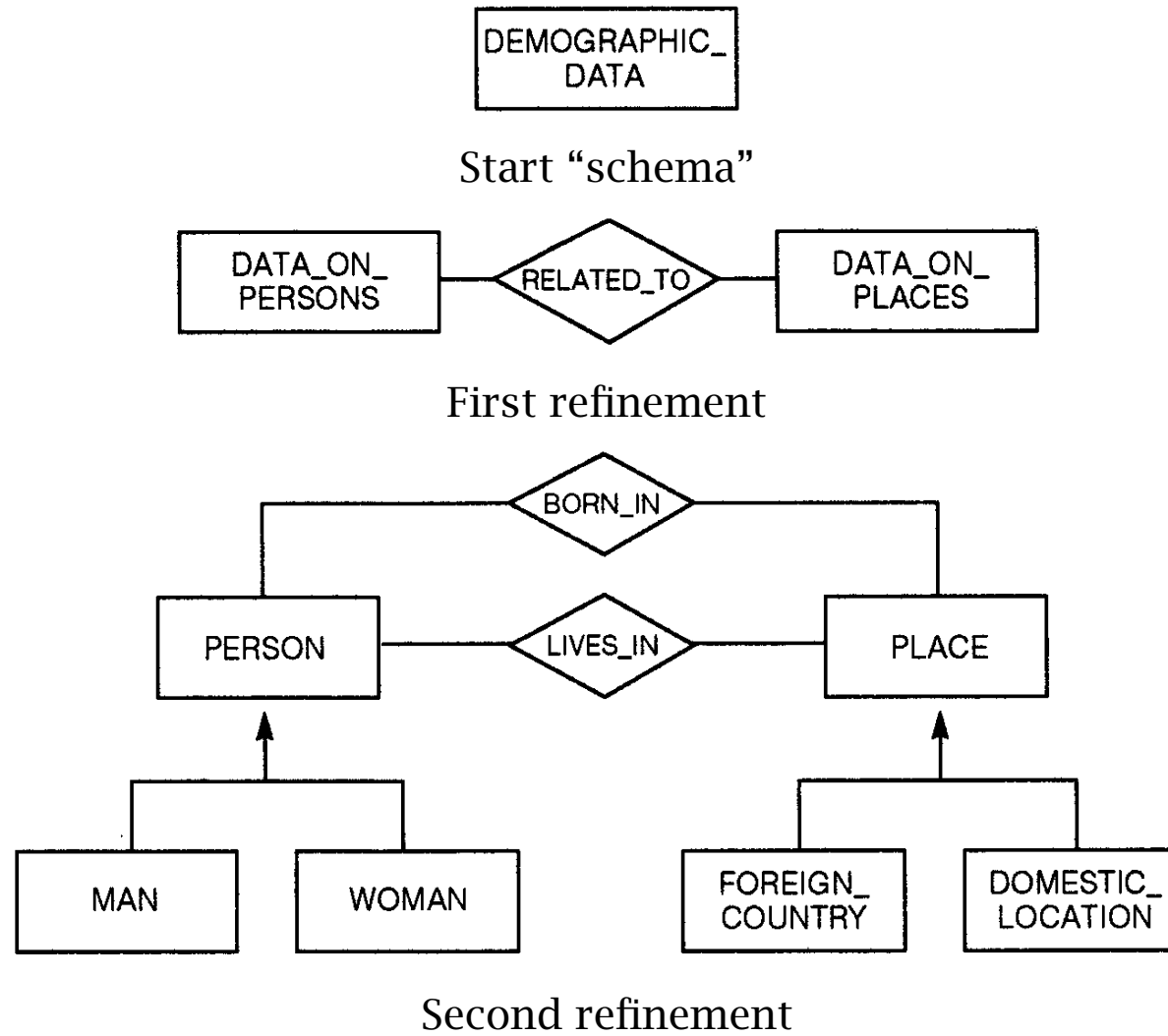
Explizite Integritätsbedingungen an die Busunternehmens-DB

- beachte: Die Integritätsbedingung *“each branch must be at a city located along the bus routes”* wird durch die EER-Modellierung bereits inhärent abgedeckt ! Denn: *In jeder Stadt (City) [die in der Datenbank gespeichert wird] liegt mind. ein Stop einer Bus-Route.*
- *Start- und Ende-Beziehungen (starts at, ends at) einer Bus-Route müssen auch Stop-Beziehungen (stops at) sein.*
- *Jedem Bus wird pro Datum genau ein Fahrer und genau ein Assistent (Driver/Assistant) zugeordnet (assigned); diese müssen verschieden sein. Kein Angestellter (Employee) darf am gleichen Datum mehreren Bussen zugeordnet sein.*
- *Falls mehrere Busunternehmen in der DB gespeichert sind, etwa weil sie gemeinsam Routen betreiben: Wenn ein Bus einem Unternehmen (Company) C gehört (owned by), dann müssen auch alle zugeordneten Angestellten für C arbeiten (works at).*

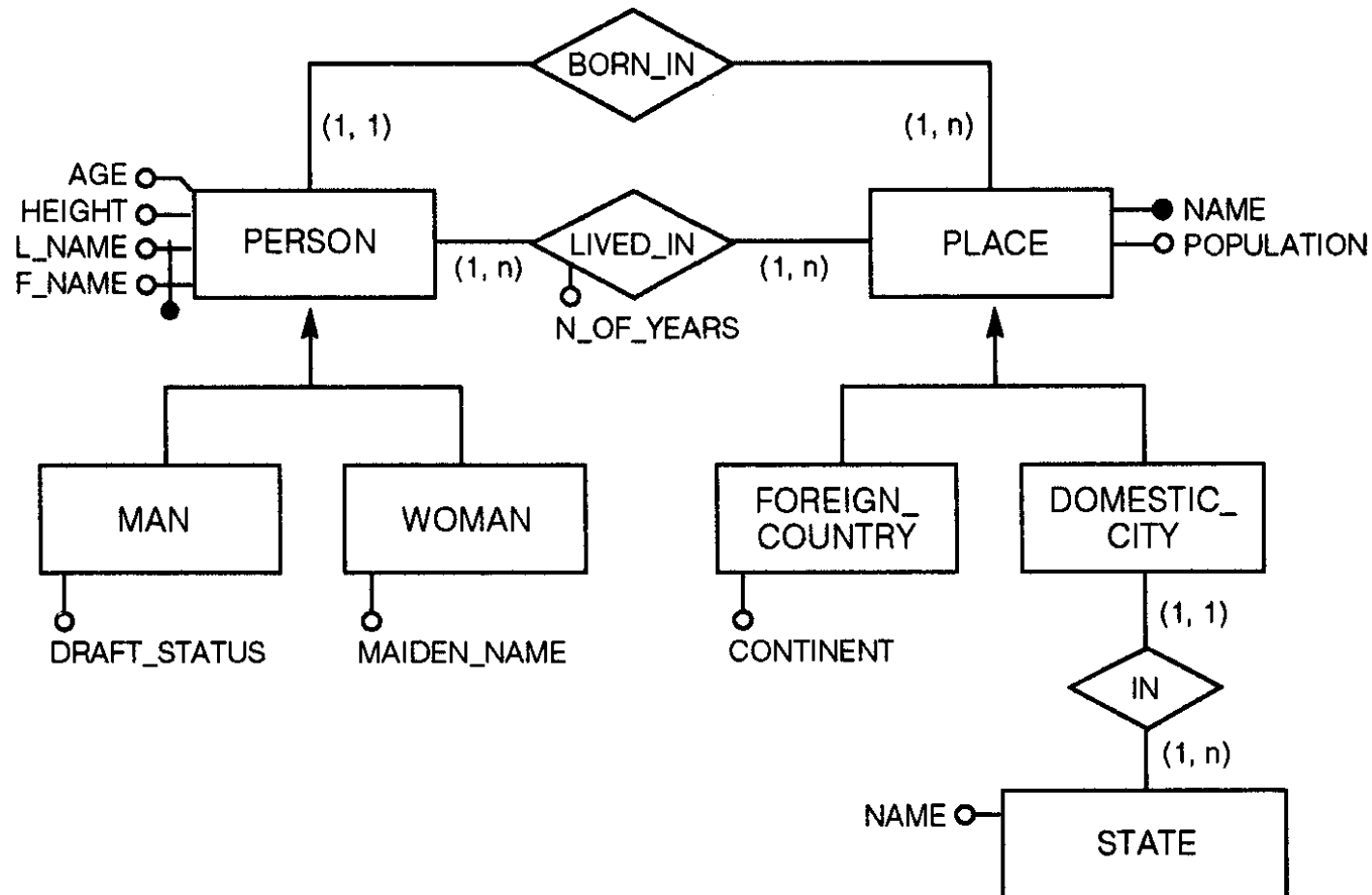
Methoden zum Entwurf von EER-Schemata

- **Top-down** (“Entity-Analyse”):
zuerst [aufgrund struktureller Vorkenntnisse] Wahl der Entities, der identifizierenden Attribute und der Relationships, ggf. Spezialisierungen, dann Anreicherung um weitere Attribute;
allgemeiner: schrittweise Verfeinerung von Schemata
- **Bottom-up** (“Attribut-Synthese”):
zuerst Sammlung von Attributen, dann Zusammenfassung zu Entities und Relationships [etwa aufgrund gemeinsamer Verwendung], schließlich Aufstellung weiterer Relationships, ggf. Generalisierung (Zusf. von Spezialisierungen), und Zuordnung restlicher Attribute;
allgemeiner: schrittweise Bildung von Abstraktionen
- **Inside-out:**
vom wichtigsten Konzept ausgehen und ergänzen
- oder **gemischt** (“Jo-jo”-Methode)

A design session using the top-down strategy



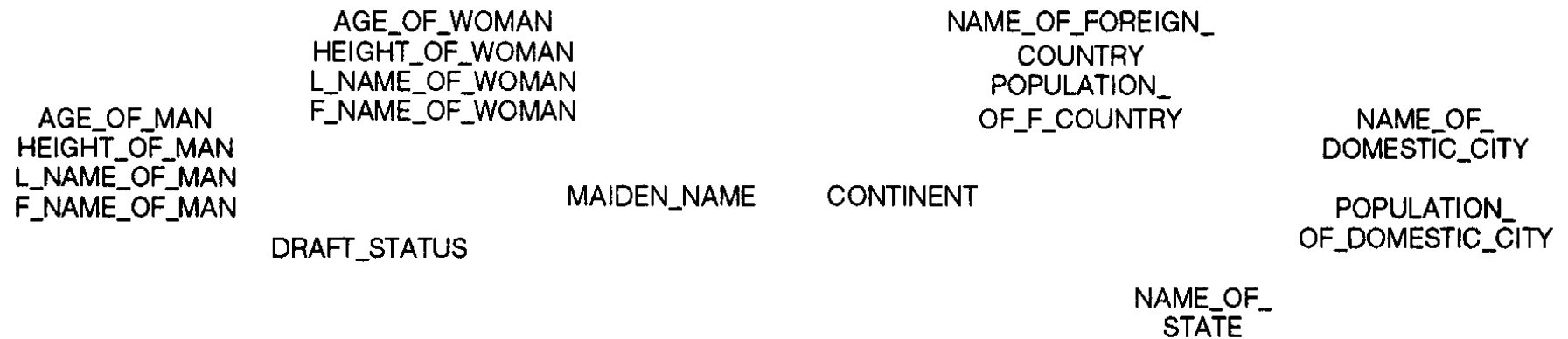
A design session using the top-down strategy (Forts.)



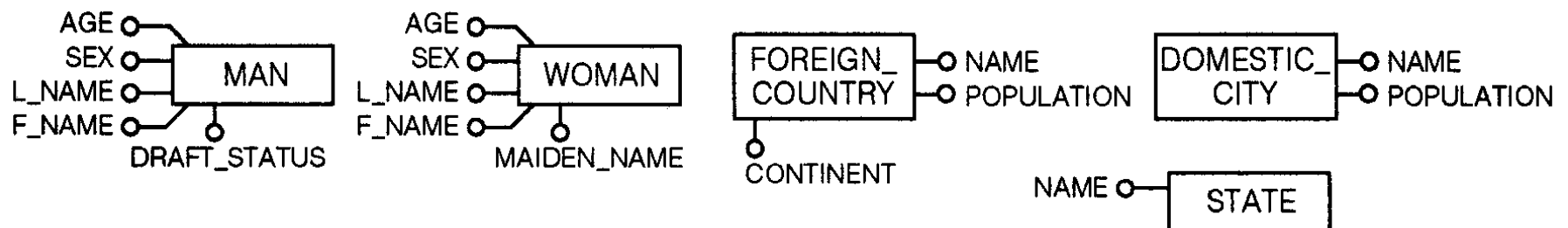
Further refinements / Final schema

— zur Ergänzung der Vorlesung —

A design session using the bottom-up strategy

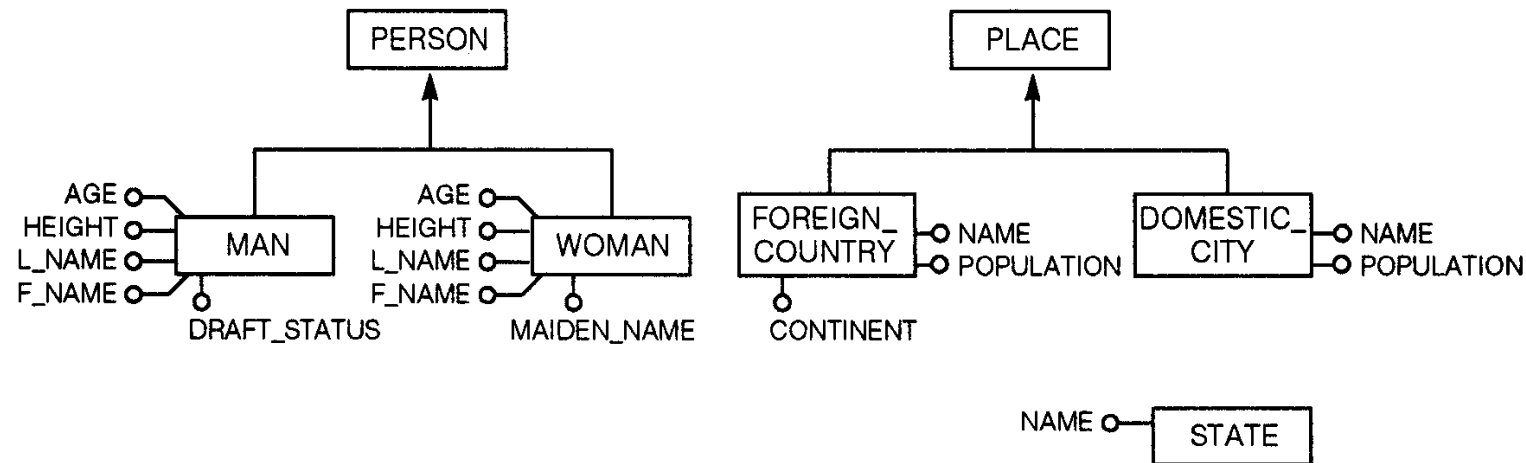


First “schema” (attributes only)

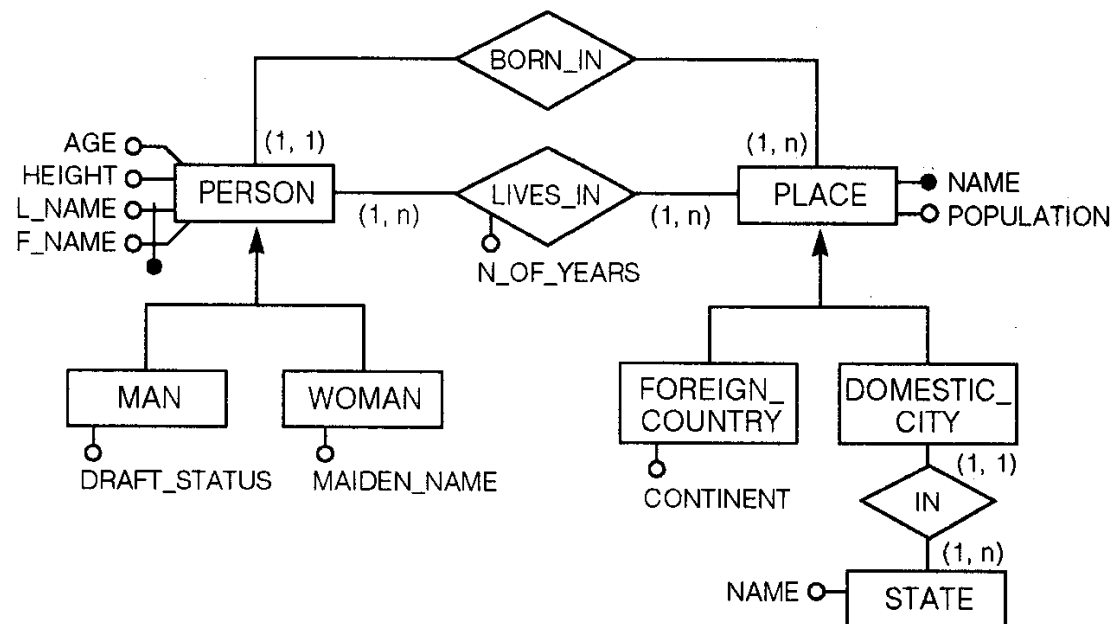


Second schema

A design session using the bottom-up strategy (Forts.)



Third schema

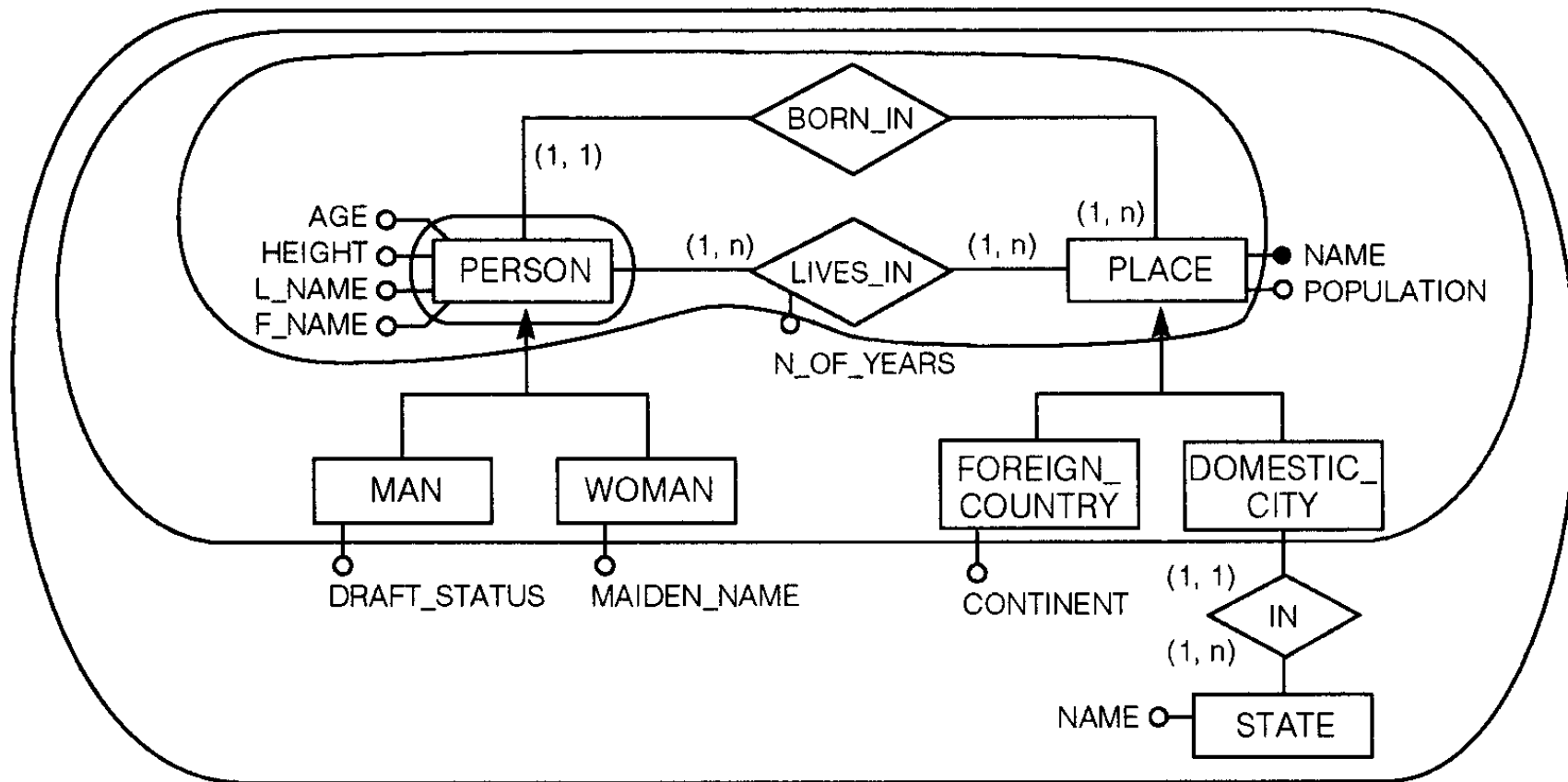


Final schema⁷

⁷Man beachte die Migration von Attributen und die Festlegung von Schlüsseln bei den Generalisierungen.

— zur Ergänzung der Vorlesung —

A design session using the inside-out strategy⁸



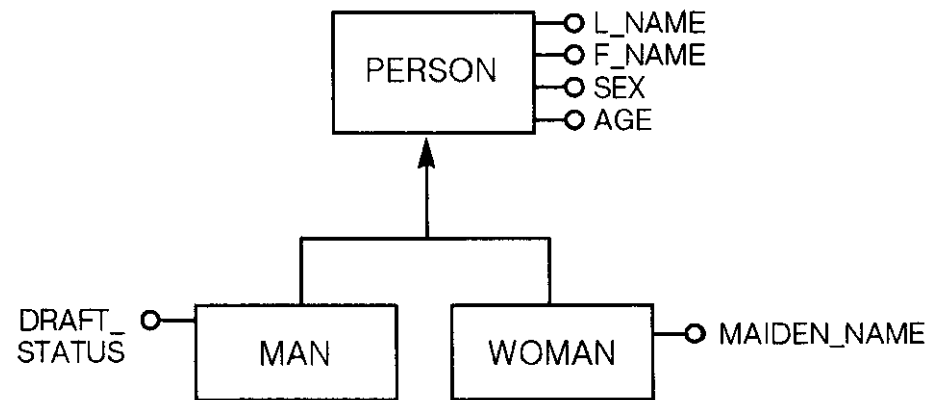
⁸ausgehend vom wichtigsten oder bekanntesten Konzept, hier PERSON

— zur Ergänzung der Vorlesung —

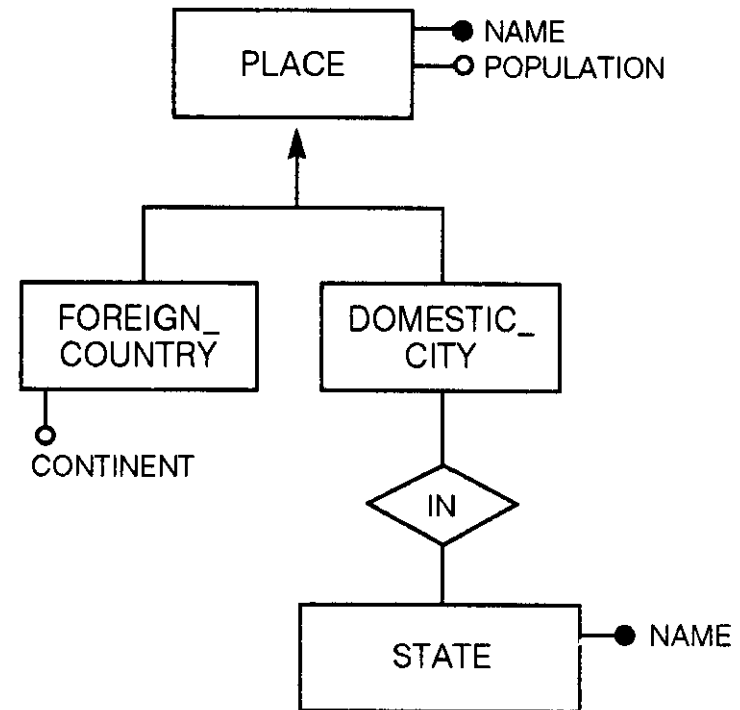
A design session using the mixed strategy



Skeleton schema

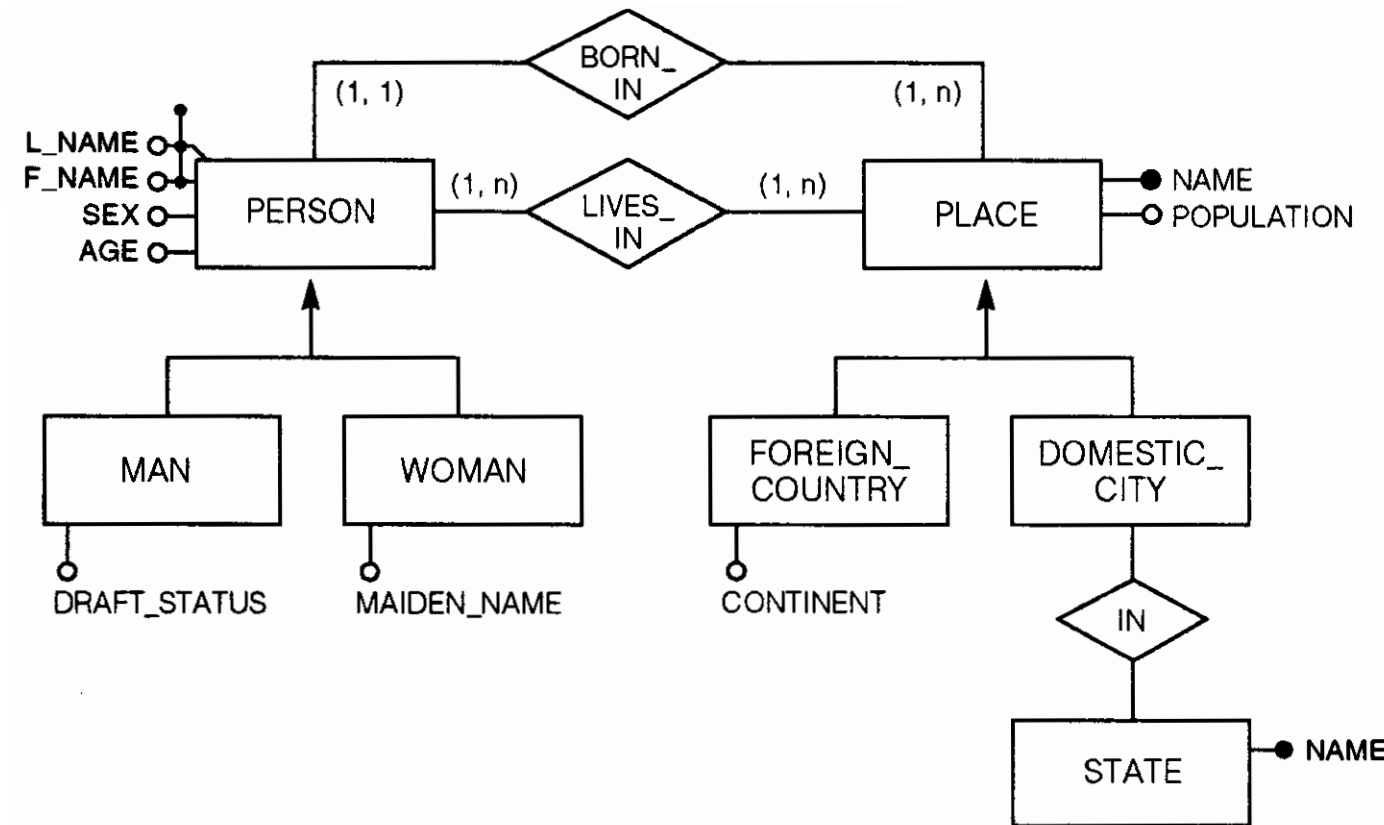


PERSON schema



PLACE schema

A design session using the mixed strategy (Forts.)



Integrated schema

Comparison of the design strategies

Strategy	Description	Advantages	Disadvantages
Top-down	Concepts are progressively refined	No undesired side effects	Requires a capable designer with high abstraction ability at the very beginning
Bottom-up	Concepts are built from elementary components	Ease of local design decisions No burden on initial designer	Need of restructuring after applying each bottom-up primitive
Inside-out	Concepts are built with an oil-stain approach	Ease of discovering new concepts close to previous ones No burden on the initial designer	A global view of the application domain is built only at the end
Mixed	Top-down partitioning of requirements; bottom-up integration using a skeleton schema	Divide-and-conquer approach	Requires critical decisions about the skeleton schema at the beginning of the design process

Qualitätskriterien für gute Modellierungen

- **Korrektheit**

Die syntaktische Korrektheit wird eingehalten, wenn ein Schema den Notationsregeln des gewählten Datenmodells genügt.

Die semantische Korrektheit bewertet den “Inhalt” des Schemas: inwieweit entspricht das Schema dem abzubildenden Realweltausschnitt?

- **Vollständigkeit**

Im Schema sollten keine relevanten Informationen fehlen.

- **Klarheit**

Übersichtlichkeit, Strukturiertheit, Lesbarkeit, Verständlichkeit

- **Erweiterbarkeit/Anpassbarkeit**

... für zukünftige Anforderungen

Qualitätskriterien für gute Modellierungen (Forts.)

- **Selbsterklärung**

Das Schema sollte sich in der gewählten (z.B. graphischen) Notation weitgehend selbst erklären; es sollten also möglichst wenig Zusatz-erklärungen, Kommentare usw. nötig sein.

Konkret: Es ist besser, die strukturellen bzw. inhärenten Konzepte des Datenmodells auszunutzen als viele explizite Integritätsbedingungen aufstellen zu müssen.

- Evtl. **Minimalität** bzw. **Nichtredundanz**

Das Schema sollte möglichst keine überflüssigen oder ableitbaren Informationen enthalten. Wenn doch, dann sind Redundanzen durch Integritätsbedingungen explizit zu dokumentieren.

Typische Anlässe für IBen liefern jegliche Mehrfachwege in Diagrammen, temporale Attribute, und verwandte aber verstreute Attribute.

- **“Normalisierung”**

Die Schema-Strukturen sollten zu den erwarteten Updates passen: Unabhängig manipulierbare Einheiten bzw. Einheiten mit spezifischen (Attribut/Rel'ship-) Kontexten sollten durch eigene Entities oder Relationships dargestellt werden.

2.3 Das Relationenmodell

Modellierungskonzepte

Relationen (= Mengen von Tupeln)

|
Tupel (= Wertekombinationen,
stellen Objekte oder Beziehungen dar)

|
Werte ($\hat{=}$ Attribute)
Objektbezüge?: durch Schlüsselattribute darstellen!

Relationenschema $R(A_1 : D_1, \dots, A_n : D_n)$

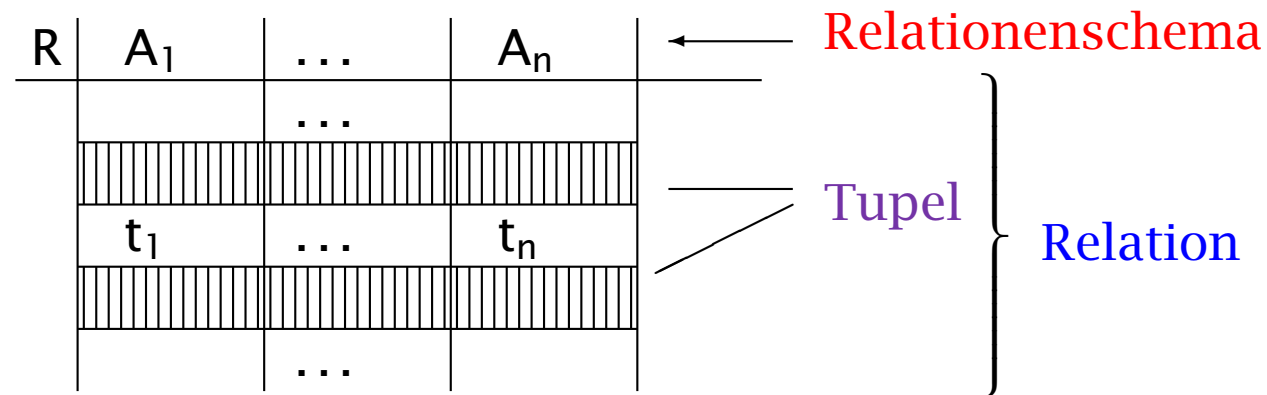
entspricht in der Regel einem Objekt- oder Beziehungstyp R,
hat **Attribute** A_i (vom Datentyp D_i), $i = 1, \dots, n$

Relationales DB-Schema:

Sammlung von mehreren Relationenschemata

Jedes Relationenschema legt eine endliche feste Auswahl von Attributen fest. Die Attribute eines Relationenschemas müssen paarweise verschiedene Namen haben; ebenso die Relationenschemata innerhalb eines DB-Schemas.

Repräsentation von Relationen durch Tabellen



Sprachgebrauch:

<i>in Theorie</i>	<i>in SQL-Programmierung</i>
Relation	Tabelle (table)
Attribut	Spalte (column)
Tupel	Zeile (row)

Beachte: Reihenfolge und Mehrfachvorkommen von Zeilen sind im Relationenmodell irrelevant; in SQL sind Mehrfachvorkommen zu beachten.

Beispiel-Zustand der Bibliotheks-Datenbank:

(mehr zum Schema s.u.)

BUCH	DokNr	Titel	Verlag	Ort	Jahr
	B8501	Datenbanksysteme
	A8442	Compiler
	D8333	Betriebssysteme

STUDENT	MatrNr	SName	Fach	Semester
	3141593	K. Klug	Math.	7
	2718300	N. Neugierig	Elek.	9
	1010111	P. Praktisch	Inf.	4
	1618034	W. Weise		11

AUSLEIHE	DokNr	MatrNr	Datum
	B8501	3141593	11.10.2013
	B8501	2718300	18.10.2014
	A8442	2718300	28.03.2015
	B8501	1010111	30.04.2015

Beispiel-Zustand (Forts.)

AUTOREN	DokNr	AName
	B8501	Ullman
	A8442	Ullman
	A8442	Aho
	D8333	Tanenbaum

DESKRIPTOREN	DokNr	Schlagwort
	B8501	Datenbank
	B8501	Anfragesprache
	A8442	Programmiersprache
	A8442	Anfragesprache
	D8333	Programmiersprache
	D8333	Scheduling

Modellierungskonzepte (Forts.)

Semantik:

$R \rightsquigarrow$ **endliche Relation** $\zeta(R) \subseteq \mu(R) := |D_1| \times \dots \times |D_n|$,
d. h. eine Menge von **Tupeln** t der Form (t_1, \dots, t_n) , $t_i \in |D_i|$

Die im Zustand ζ “aktuellen Tupel” $\zeta(R)$ bilden wieder eine Teilmenge der “möglichen Tupel” $\mu(R)$. — Wenn das i -te Attribut Nullwerte annehmen darf, ist der Wertebereich nicht $|D_i|$, sondern $|D_i|^\perp := D_i \cup \{\perp\}$.

$A_i \rightsquigarrow$ Funktion, die von jedem Tupel t den Wert **$t.A_i = t_i$** liefert

Die Notation “ $t.A_i$ ” macht den Zugriff auf Attributwerte unabhängig von der Position des Attributs.

Beispiel⁹:

STUDENT(MatrNr: number(7), SName: varchar(20),
Fach: varchar(8)^{null}, Semester: number(2))

$\mu(\text{STUDENT}) = \{0, \dots, 9999999\} \times C^{[20]} \times C^{[8]^\perp} \times \{0, \dots, 99\}$

enthält **(1010111, 'P.Praktisch', 'Inf.', 4)**, aber auch **(0, 'xyz', 'Fun', 99)**

⁹ C bezeichne die Zeichenmenge $\{A, \dots, Z, a, \dots, z, 0, \dots, 9, \sqcup, \dots\}$ und $C^{[k]}$ die Menge aller Wörter über C bis zur Länge k .

Beispiel-Tabellendefinitionen

*Relationenschema: (Notation in Theorie und beim logischen DB-Entwurf)*¹⁰

STUDENT(MatrNr: number(7), SName: varchar(20),
Fach: varchar(8)^{null}, Semester: number(2))

*Tabellendefinition in SQL:*¹¹

```
create table STUDENT
(MatrNr    number(7) primary key,
 SName     varchar2(20) not null,
 Fach      varchar2(8),
 Semester  number(2) not null);
```

Ein aktueller Tabelleninhalt:

STUDENT	MatrNr	SName	Fach	Semester
	3141593	K. Klug	Math.	7
	2718300	N. Neugierig	Elek.	9
	1010111	P. Praktisch	Inf.	4
	1618034	W. Weise		11

¹⁰Attribute **not null**, soweit nicht anders angegeben

¹¹In SQL ist **null** voreingestellt.

Modellinhärente Integritätsbedingungen

- Für ein Attribut A können **Nullwerte** als Attributwerte
 - erlaubt sein (**null**), d. h. der zugehörige Datentyp wird um ' \perp ' ("undefiniert") erweitert
 - oder ausgeschlossen sein (**not null**).

Voreinstellung:

in SQL: **null**; beim logischen Entwurf: jeweils angeben.

Modellinhärente Integritätsbedingungen (Forts.)

- Zu einer Relation R kann eine Attributkombination \bar{A} (oder ein Attribut) als **eindeutig (unique)** spezifiziert werden, d. h. es soll in jedem DB-Zustand gelten¹² :

$$\forall (t, t' \in R) (t \neq t' \Rightarrow ((\text{is_null}(t.\bar{A}) \wedge \text{is_null}(t'.\bar{A})) \vee t.\bar{A} \neq t'.\bar{A}))$$

Für je zwei Tupel t, t' gilt: Wenn die Tupel verschieden sind, dann müssen sie sich bereits in einem Attribut aus \bar{A} unterscheiden, außer beide haben Nullwerte in \bar{A} .

Eine minimale¹³ eindeutige Attributkombination, in der Nullwerte verboten sind, heißt auch **Schlüsselkandidat**.

- Zu jedem Relationenschema muss ein Schlüsselkandidat als **(Primär-) Schlüssel (primary key)** ausgezeichnet werden¹⁴.

Kurznotation: Unterstreichen der Schlüsselattribute

¹²Notation: Für gleichlange Attributkombinationen $\bar{A} = \langle A_{i1}, \dots, A_{ik} \rangle$ und $\bar{B} = \langle B_{j1}, \dots, B_{jk} \rangle$ und Tupel t, t' stehe eine Gleichung $t.\bar{A} = t'.\bar{B}$ für die Konjunktion $(t.A_{i1} = t'.B_{j1} \wedge \dots \wedge t.A_{ik} = t'.B_{jk})$ d. h. Übereinstimmung in allen Attributpositionen, und eine Ungleichung $t.\bar{A} \neq t'.\bar{B}$ dementsprechend für Abweichung in mindestens einer Attributposition.

Für den Test von Attributkombinationen auf Nullwerte bedeutet $\text{is_null}(t.\bar{A}) : \Leftrightarrow t.A_{i1} \equiv \perp \vee \dots \vee t.A_{ik} \equiv \perp$ (Nullwert in einer Position).

Die Zeichen ' \wedge ', ' \vee ' und ' \neg ' stehen für die logischen Verknüpfungen 'und', 'oder' bzw. 'nicht'; ' \forall '/' \exists ' bedeuten 'für alle'/'es gibt'.

¹³Minimalität bzgl. der Eindeutigkeitsbedingung besagt, dass nach Weglassen eines Attribut die Bedingung nicht mehr erfüllt wäre.

¹⁴Beachte: **primary key** beinhaltet **not null**. In Oracle/SQL ist die Angabe eines Primärschlüssels leider nicht unbedingt nötig.

Beispiel-Tabellendefinitionen (Forts.)

```
create table HOERSAAL
  (Interneld      number(10) primary key,           (column constraints)
   GebaeudeNr     number(4) not null,
   GebaeudeName   varchar2(50) not null,
   Trakt          varchar2(1),
   RaumNr         number(3) not null,
   SaalName       varchar2(30) null unique,
   AnzPlaetze     number(3) not null,
   unique (GebaeudeNr,Trakt,RaumNr),                 (table constraints)
   unique (GebaeudeName,Trakt,RaumNr)
);
```

oder wahlweise als benutzer-benannte IBen (für lesbarere Fehlermeldungen):

```
...
Interneld      number(10) constraint HOERSAAL_PK primary key, ...
AnzPlaetze     number(3) constraint PLAETZE_NN not null,
constraint HOERSAAL_SECOND_ID unique (GebaeudeNr,Trakt,RaumNr),
unique (GebaeudeName,Trakt,RaumNr)
...
```


Modellinhärente Integritätsbedingungen (Forts.)

- Ein Attribut oder eine Attributkombination \bar{B} einer Relation R_1 kann als **Referenz** auf eine in Länge und Datentypen passende Attributkombination \bar{A} einer Relation R_2 spezifiziert werden. In diesem Fall muss die folgende Bedingung (**referentielle Integrität**) gelten:

$$\forall (t_1 \in R_1) (\neg \text{is_null}(t_1.\bar{B}) \Rightarrow \exists (t_2 \in R_2) (t_1.\bar{B} = t_2.\bar{A}))$$

Für jedes Tupel t_1 aus R_1 muss es, wenn t_1 keinen Nullwert in \bar{B} hat, ein Tupel t_2 in R_2 geben, dessen \bar{A} -Wert der \bar{B} -Wert von t_1 ist.

oder äquivalent: $\{t_1.\bar{B} \mid t_1 \in R_1, \neg \text{is_null}(t_1.\bar{B})\} \subseteq \{t_2.\bar{A} \mid t_2 \in R_2\}$

Alle \bar{B} -Attributwerte in R_1 (außer Nullwerte) müssen – im gleichen DB-Zustand – als \bar{A} -Attributwerte in R_2 vorkommen (“dynamischer Wertebereich”).

z.B.: $\{a.\text{DokNr} \mid a \in \text{AUSLEIHE}\} \subseteq \{b.\text{DokNr} \mid b \in \text{BUCH}\}$

Kurznotation: $R_1(\dots, \bar{B} \rightarrow R_2.\bar{A}, \dots)$, z.B. $\text{AUSLEIHE}(\text{DokNr} \rightarrow \text{BUCH.DokNr}, \dots)$

- Eine Attributkombination \bar{B} von R_1 kann als **Fremdschlüssel (foreign key)** bzgl. R_2 spezifiziert werden, wenn \bar{B} eine Referenz auf den Primärschlüssel von R_2 bildet. *Kurznotation: $R_1(\dots, \bar{B} \rightarrow R_2, \dots)$*

Beispiel-Tabellendefinitionen (Forts.)

Relationenschema: Attribute **not null**

AUSLEIHE(DokNr: varchar(5) → BUCH,
 MatrNr: number(7) → STUDENT, Datum: date)

Tabellendefinition in SQL:

```
create table AUSLEIHE  
  (DokNr  varchar2(5) references BUCH,  
   MatrNr number(7),  
   Datum  date not null,  
   constraint AUSLEIHE_PK primary key (DokNr,MatrNr),  
   constraint AUSLEIHE_STUD_FK  
     foreign key (MatrNr) references STUDENT (MatrNr) )
```

Ein aktueller Tabelleninhalt:

AUSLEIHE	DokNr	MatrNr	Datum
	B8501	3141593	11.10.2013
	B8501	2718300	18.10.2014
	A8442	2718300	28.03.2015
	B8501	1010111	30.04.2015

Logischer DB-Entwurf: Transformation von ER-Schemata in äquivalente Relationale DB-Schemata

Prinzipielle Transformationsschritte:

1. Ein **Objekttyp E** mit Schlüsselattribut(kombination) X , weiteren Attributen A_1, \dots, A_k und optionalen Attributen B_1, \dots, B_m wird transformiert in ein Relationenschema

$$E (\underline{X}, A_1^{\text{not null}}, \dots, A_k^{\text{not null}}, B_1^{\text{null}}, \dots, B_m^{\text{null}})$$

2. Ein **allgemeiner (nicht-funktionaler) Beziehungstyp R** zwischen Objekttypen E_1, \dots, E_p , ggf. mit eigenen Attributen A_{\dots}, B_{\dots} wie oben, wird transformiert in ein Relationenschema

$$R (\underline{X_1} \rightarrow E_1, \dots, \underline{X_p} \rightarrow E_p, A_1^{\text{not null}}, \dots, B_1^{\text{null}}, \dots)$$

wobei X_i für die Schlüsselattribut(kombination) von E_i steht.

Ggf. sind gleichnamige Schlüsselattribute von verschiedenen Objekttypen oder verschiedenen Rollen vorher eindeutig umzubenennen.

Transformation von ER- in Relationale DB-Schemata (Forts.)

3. Für einen **funktionalen Beziehungstyp** $E \text{ --- } \langle R \rangle \text{ --- } F$ (Kardinalität (... ,1) an E-Kante), bei dem E und F als Schlüsselattribut(kombination)e(n) X bzw. Y haben, wird stattdessen das Schema $E(\underline{X}, \dots)$ zu
- $$E(\underline{X}, \dots, Y \rightarrow F)$$

ergänzt. — Falls die funktionale Beziehung total ist (Kardinalität (1,1) an E-Kante), darf Y keinen Nullwert annehmen ($Y^{\text{not null}}$).

Sollte auch die F-Kante die Kardinalität (... ,1) tragen (1:1-Beziehung), wird eine explizite IB benötigt (selbst wenn man $F(\dots)$ in Gegenrichtung erweitern würde).

Auch **mehrstellige Beziehungstypen** $E \text{ --- } \langle R \rangle \text{ --- } \dots$ (mit maximaler Kardinalität 1 an einem “Quell”-Objekttyp E) sind analog zu funktionalen Beziehungen durch Erweitern des Relationenschemas $E(\dots)$ um die Schlüssel aller “Ziel”-Objekttypen zu transformieren; diese dürfen nur gemeinsam den Nullwert annehmen oder nicht (explizite IB!). Bei Kardinalität (1,1) sind Nullwerte zu verbieten.

Außer in den oben erwähnten (1,1)-Fällen erfordern jegliche totale Beteiligungen bzw. Beziehungskanten mit Kardinalität (1,...) die Aufstellung expliziter IBen.

Transformation von ER- in Relationale DB-Schemata (Forts.)

4. Weitere **Kardinalitäten und Integritätsbedingungen** sind (in dieser Prioritätenfolge) durch
 - Zusammenfassen und Streichen von Relationen
 - Nutzen von modellinhärenten Integritätsbedingungen (Fremdschlüssel, Nullwerte erlaubt?, Eindeutigkeiten)
 - Ergänzen expliziter Integritätsbedingungenzu berücksichtigen, um Äquivalenz zu garantieren.
5. Außerdem sind in allen Schritten passende Umbenennungen von Attributen und Relationen durch den Entwerfer erlaubt und oft sogar empfehlenswert.

Beispiel: Transformation des ER-Schemas zur Bibliotheks-DB¹⁵

gemäß Schritt 1 (*Objekttypen*):

BUCH(DokNr, Titel, Verlag, Ort, Jahr)

STUDENT(MatrNr, SName, Fach^{null}, Semester)

AUTOR(AName)

DESKRIPTOR(Schlagwort)

Schritt 4 (*Kardinalitäten*):

} streichen, da alle Autoren
und Schlagwörter an BA, BD
beteiligt sind.

gemäß Schritt 2 (*Beziehungstypen*):

AUSLEIHE(DokNr → BUCH, MatrNr → STUDENT, Datum)

BA(DokNr → BUCH, AName → AUTOR)

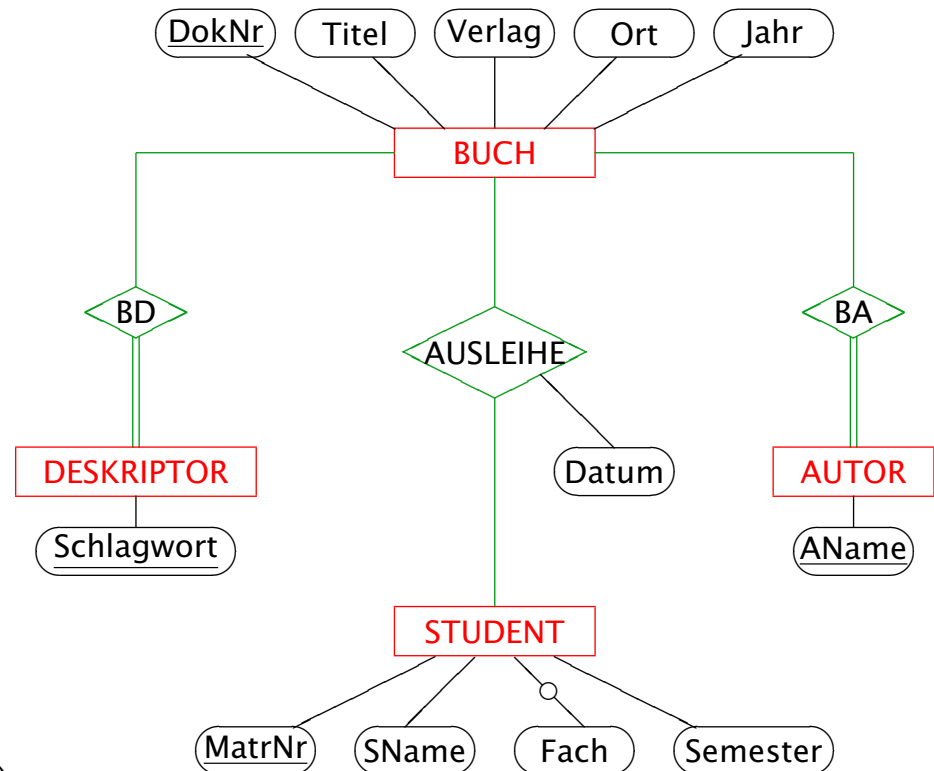
BD(DokNr → BUCH, Schlagwort → DESKRIPTOR)

} Schritt 5:
umbenennen
zu (*)

$(*) = \begin{cases} \text{AUTOREN} (\underline{\text{DokNr}} \rightarrow \text{BUCH}, \underline{\text{AName}}) \\ \text{DESKRIPTOREN} (\underline{\text{DokNr}} \rightarrow \text{BUCH}, \underline{\text{Schlagwort}}) \end{cases}$

¹⁵Die Attribute seien voreingestellt als **not null** spezifiziert.

Beispiel: Transformation des ER-Schemas zur Bibliotheks-DB (Forts.)



Ergebnisschema:

BUCH (DokNr, Titel, Verlag, Ort, Jahr)

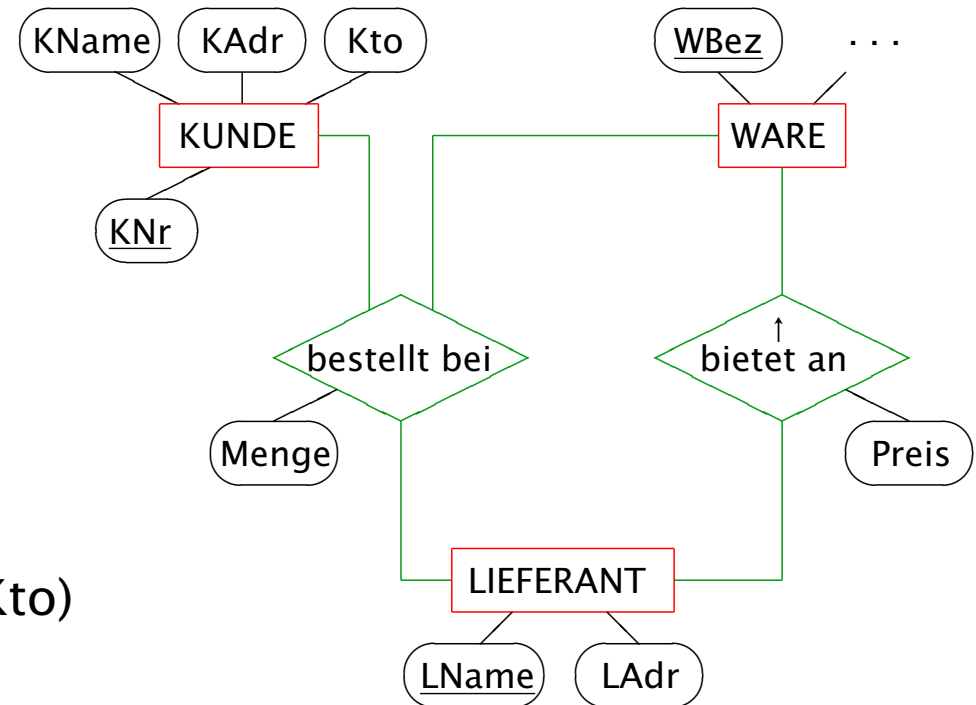
STUDENT (MatrNr, SName, Fach^{null}, Semester)

AUSLEIHE (DokNr → BUCH, MatrNr → STUDENT, Datum)

AUTOREN (DokNr → BUCH, AName)

DESKRIPTOREN (DokNr → BUCH, Schlagwort)

Beispiel: Transformation des ER-Schemas zum Warenmarkt¹⁶



KUNDE (KNr, KName, KAdr, Kto)

WARE (WBez, . . .)

LIEFERANT (LName, LAdr)

ANGEBOT (LName → LIEFERANT, Ware → WARE, Preis)

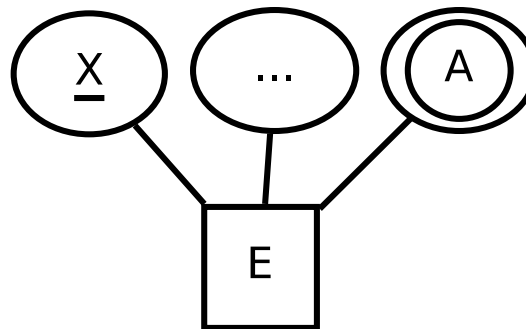
BESTELLUNG (KNr → KUNDE, LName → LIEFERANT,
 Ware → WARE, Menge;
 (LName, Ware) → ANGEBOT¹⁷)

¹⁶nur **not null**-Attribute

¹⁷Hier kann man eine ER-explizite IB (vgl. S. 2.16u) durch eine relational inhärente IB, und zwar eine Fremdschlüsselbedingung, ausdrücken!

Transformation von ER-Erweiterungen

- **Mehrwertige Attribute** sind wie zusätzliche, funktional rückverbundene Objekte zu behandeln (vgl. die Schlagworte von Büchern).



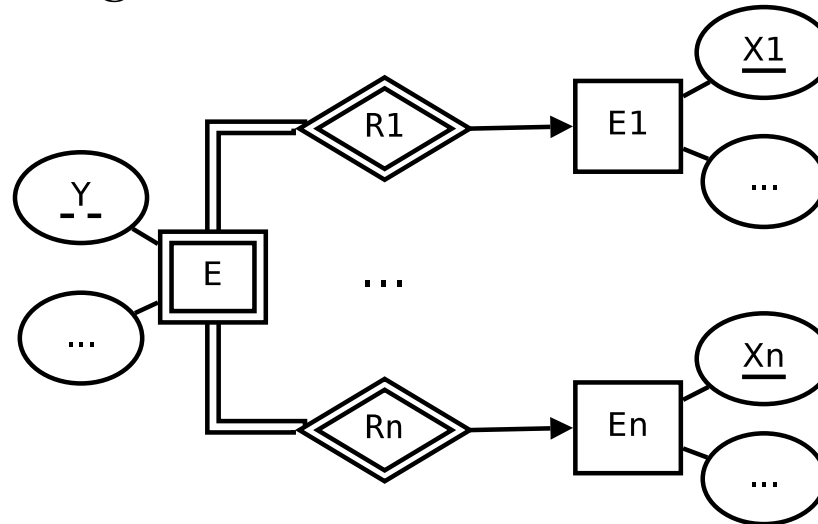
z.B.: $E \equiv \text{BUCH}$, $X \equiv \text{DokNr}$, $A \equiv \text{Schlagwort}$, ...

Ein **mehrwertiges Attribut A** eines Objekttyps E mit Schlüsselattribut(kombination) X wird transformiert in ein eigenes Relationenschema mit den Attributen X (Fremdschlüssel) und A, die auch zusammen den Primärschlüssel bilden:

$E(\underline{X}, \dots)$ $A(\underline{X} \rightarrow E, \underline{A})$

Transformation von ER-Erweiterungen (Forts.)

- **Schwache Entities** sind durch ihre zusammengesetzten Schlüssel zu repräsentieren. **Identifizierende Beziehungen** sind wie spezielle funktionale Beziehungen zu behandeln.



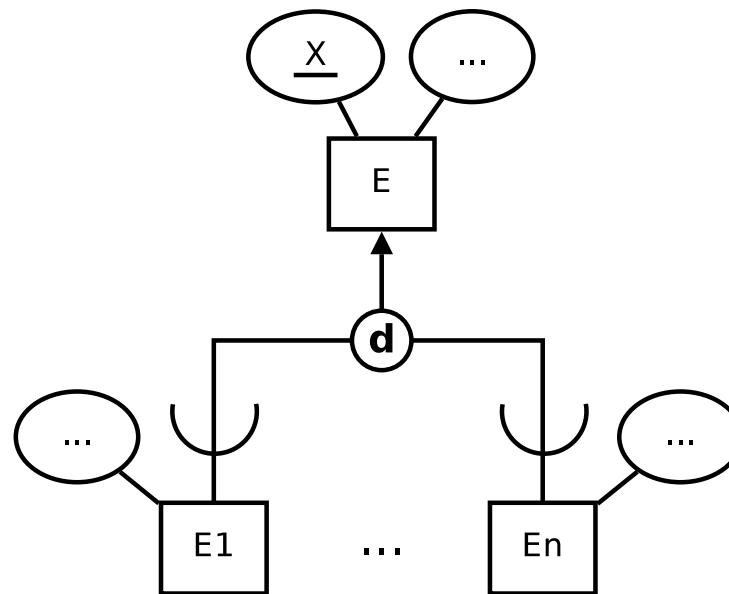
z.B.: $E \equiv \text{VERKAUF}$, $Y \equiv \text{LfdNr}$, $R_1 \equiv \text{was?}$, $E_1 \equiv \text{PRODUKT}$, $X_1 \equiv \text{PNr}$,
 $R_2 \equiv \text{wo?}$, $E_2 \equiv \text{LADEN}$, $X_2 \equiv \text{LNr}$, ...

- Bei der Transformation eines **schwachen Objekttyps** E mit partiellem Schlüssel Y werden die Schlüssel X_1, \dots, X_n bzgl. der über identifizierende Beziehungen verbundenen Objekttypen E_1, \dots, E_n zum Schlüssel des Relationenschemas E hinzugefügt, und zwar als Fremdschlüssel:

$E_1(\underline{X_1}, \dots) \dots E_n(\underline{X_n}, \dots) \quad E(\underline{X_1} \rightarrow E_1, \dots, \underline{X_n} \rightarrow E_n, \underline{Y}, \dots)$

Transformation von ER-Erweiterungen (Forts.)

- Subklassen ohne eigene Schlüssel (in Spezialisierungen) erben die Schlüssel ihrer Superklasse. **Teilmengen-Beziehungen** in Spezialisierungen sind wie spezielle funktionale Beziehungen zu behandeln.



z.B.: $E \equiv \text{ANGESTELLTE}$, $X \equiv \text{TIN}$, $E1 \equiv \text{FEST_ANG}$, $E2 \equiv \text{ZEIT_ANG}$, ...

- In den Relationenschemata der **spezialisierten Objekttypen (Subklassen)** E_1, \dots, E_n ist der Schlüssel X der Superklasse E sowohl Primär- als auch Fremdschlüssel:

$$E(\underline{X}, \dots) \quad E_1(\underline{X} \rightarrow E, \dots) \quad \dots \quad E_n(\underline{X} \rightarrow E, \dots)$$

Transformation von ER-Erweiterungen (Forts.)

- Eine **disjunkte [und totale] Einteilung** erfordert eine explizite IB:
Jeder X-Wert in E kommt in höchstens [und mindestens]
einer der Relationen E_1, \dots, E_n vor.
- Die Alternative, eine einzige “breite” Relation E für alle Spezialisierungsattribute zu bauen, von denen oft viele **null** sind, würde ein zusätzliches Attribut **Subklasse** sowie explizite IBen zur Kontrolle der Nullwerte erfordern.