# Kapitel 14: Informationsmodellierung mit Semantischen Datenmodellen

**Ziel der Informationsmodellierung (des konzeptionellen Datenbankentwurfs):**

Modellierung von Sachverhalten der realen Welt mittels weniger Grundkonzepte
mit dem Zweck, eine computergestützte Datenbank aufzubauen.
Dazu sind notwendig:

- eine *Modellierungssprache* (ein "semantisches" Datenmodell)
- eine *Entwurfsmethodologie*
- computergestützte *Entwurfswerkzeuge* (für große Entwürfe)

**Rolle der Informationsmodellierung bei der Entwicklung von Informationssystemen:**

Informationssystemen werden in mehreren Stufen entwickelt:

1) Analyse der Anwendungswelt (Informationsbedarf, zu automatisierende Abläufe)
2) Informationsmodellierung (sowie Prozeßmodellierung)
3) Abbildung des Informationsmodells auf ein (relationales) Datenbankschema

**Notwendigkeit einer systematischen Informationsmodellierung (Entwurfstheorie):**

Relationale Datenbankschemata können "gut" oder "schlecht" entworfen sein.
Es sollte Konstruktionsrichtlinien für (komplexere) Datenbankschemata geben.

Beispiel eines schlechten bzw. debattierbaren Entwurfs:
Bestellungen (Datum, KNr, PNr, Bez, Preis, Gewicht, Menge, Summe, Status)
Produkte (PNr, Bez, Lagerort, Vorrat)
Kunden (KNr, Name, Adresse, Rabatt, Saldo)

→ Probleme: Datenredundanz, potentielle Inkonsistenz,
Nichtdarstellbarkeit bestimmter Informationen

Beispiel:
Wie wäre das Schema der Bestellungs−DB aufzubauen, wenn mit einer Bestellung mehrere Produkte auf einmal bestellt werden können, wenn diese u.U. über mehrere Lieferungen verteilt geliefert werden, entsprechend mehrere Teilzahlungen eingehen, etc. ?

# 14.1 Grundkonzepte Semantischer Datenmodelle am Beispiel des (erweiterten) Entity−Relationship−Modells

## Entities und Entity−Mengen (Entitäten, Objekte, Gegenstände)

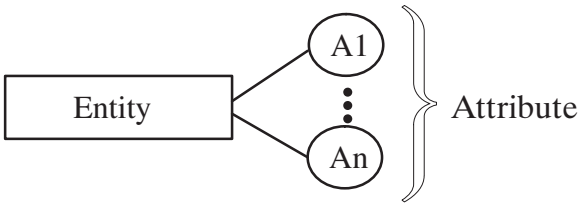
Ein Entity ist ein eindeutig identifizierbarer Gegenstand der realen Welt oder unserer Vorstellung (eine Person, ein Kunde, ein Buch, ein Bankkonto, etc.).
Gleichartige Entities werden zu einem *Entity−Typ* zusammengefaßt (z.B. alle Bücher, alle Bankkonten). Alle Entities desselben Entity−Typs werden durch eine einheitliche Menge von *Attributen* (Eigenschaften, Merkmalen) beschrieben, die jeweils einem Entity einen Wert eines bestimmten Wertebereichs (Domains) zuordnen.
Im Gegensatz zum Relationenmodell sind die Wertebereiche von Attributen nicht auf primitive Datentypen beschränkt, sondern können beispielsweise mengenwertig sein (z.B. Informatikkenntnisse als Attribut von Studenten oder Angestellten).

Beispiele:
- ISBN, Titel, Autoren, Verlag, Erscheinungsjahr als Attribute von Büchern
- Kontonummer, Kontoinhaber, Kontostand, Zinssatz als Attribute von Bankkonten
- Matrikelnummer, Name, Adresse, Semester, Prüfungen als Attribute von Studenten

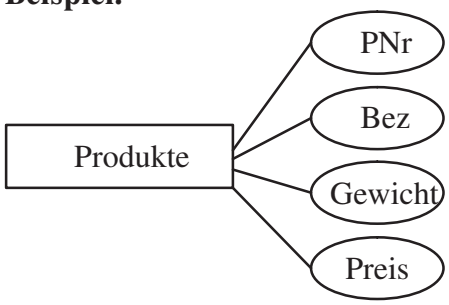
Eine Menge von Entities desselben Typs heißt *Entity−Menge* (Objektmenge, Objektklasse). Es kann mehrere, voneinander verschiedene Entity−Mengen desselben Entity−Typs geben (z.B. Studenten der Informatik und Studenten der Elektrotechnik, Bücher der Informatikbibliothek und Bücher der Universitätsbibliothek). Entity−Mengen müssen nicht disjunkt sein (z.B. Ärzte und Patienten, beide vom Entity−Typ Personen).

Bemerkungen:
- Bei vielen Anwendungen gibt es pro Entity−Typ genau eine relevante Entity−Menge. In diesen Fällen wird in der Regel nicht explizit zwischen Entity−Typ und Entity−Menge unterschieden.
- Der (Analyse−) Schritt von der Betrachtung einzelner Entities zur Betrachtung von Entity−Mengen wird auch als Abstraktion oder Klassifikation bezeichnet. Bei der Informationsmodellierung ist nur diese (Schema−) Ebene von Bedeutung.

**Graphische Darstellung**




**Beispiel:**



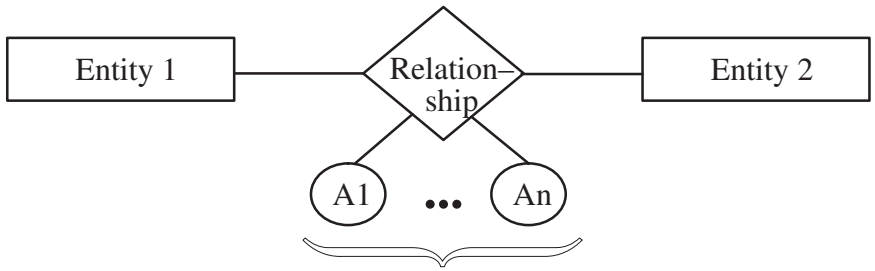

Eine Attributmenge K einer Entity−Menge E heißt *Schlüsselkandidat*, wenn zu jedem Zeitpunkt für je zwei verschiedene Entities e1, e2 ∈ E gelten muß, daß sich e1 und e2 bezüglich K unterscheiden, und es keine echte Teilmenge von K gibt, die diese Eigenschaft hat (z.B. Kontonummer bei Bankkonten einer bestimmten Bank, Kontonummer und Name der Bank bei allen Bankkonten, Personalnummer bei Angestellten).
Der *Primärschlüssel* einer Entity−Menge ist ein explizit ausgewählter Schlüsselkandidat.

## Relationships und Relationship−Mengen (Beziehungen, Assoziationen)

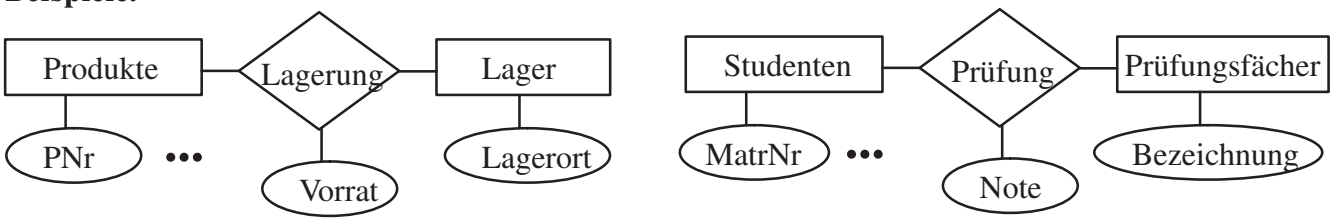
Eine *Relationship−Menge* R zwischen Entity−Mengen E1, ..., En ist eine Teilmenge des kartesischen Produkts E1 × ... × En (meist ist n=2). Ein Element <e1, ..., en> einer Relationship−Menge bildet eine Relationship zwischen den Entities e1, ..., en.
Alle möglichen Relationship−Mengen zwischen denselben Entity−Typen bilden zusammen einen Relationship−Typ.
Die Elemente einer Relationship−Menge können durch Attribute zusätzlich charakterisiert werden (z.B. die Vorratsmenge, die von einem Produkt in einem Lager vorhanden ist, oder die Note, die ein Student in einem Prüfungsfach erreicht hat). Die Primärschlüssel der an einer Relationship−Menge beteiligten Entity−Mengen gelten als implizite Attribute der Relationship−Menge.

Bemerkungen:
- Bei vielen Anwendungen gibt es pro Relationship−Typ genau eine relevante Relationship−Menge. In diesen Fällen wird in der Regel nicht explizit zwischen Relationship−Typ und Relationship−Menge unterschieden.
- Die Modellierung von Relationships zwischen Entities wird auch als (spezielle Form der) Aggregation bezeichnet.

**Graphische Darstellung (ERM−Diagramm):**




**Beispiele:**

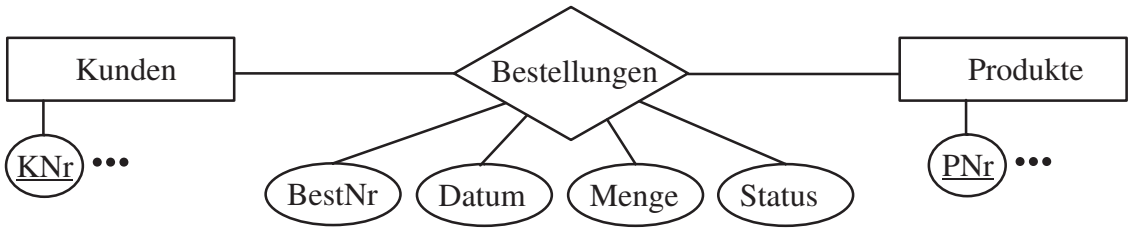



Eine Teilmenge K der Vereinigung $\bigcup_{i=1}^{n} Ki$ der Primärschlüssel K1, ..., Kn von E1, ..., En heißt *Schlüsselkandidat* der Relationship−Menge R, wenn sich zu jedem Zeitpunkt je zwei verschiedene Elemente r=<e1, ..., en> und r'=<e1', ..., en'> von R bezüglich K unterscheiden müssen und es keine echte Teilmenge von K gibt, die diese Eigenschaft hat.
Der *Primärschlüssel* einer Relationship−Menge ist ein explizit ausgewählter Schlüsselkandidat.

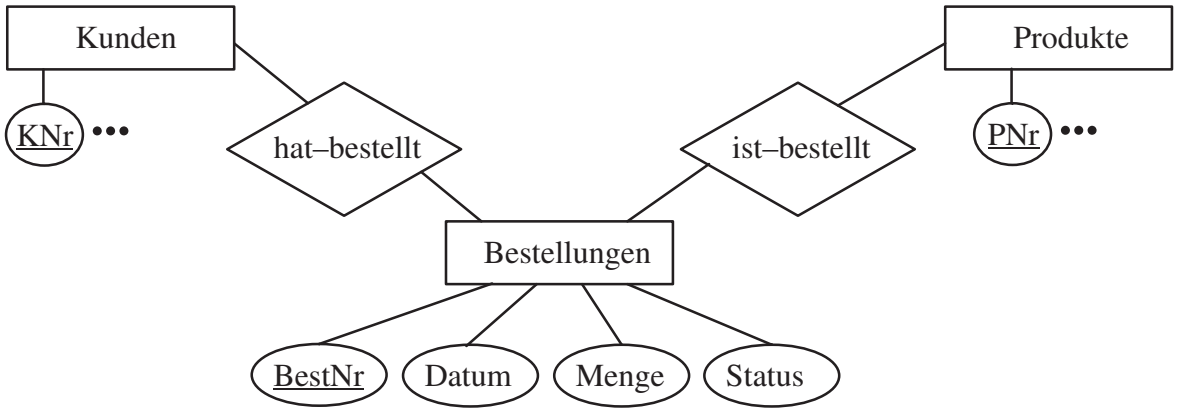
Beispiel:
PNr ist ein Schlüsselkandidat der Relationship−Menge Lagerung, falls ein Produkt immer nur an einem Ort gelagert wird. Andernfalls ist {PNr, Lagerort} ein Schlüsselkandidat von Lagerung.

## Modellierung eines Sachverhalts als Entity oder als Relationship?

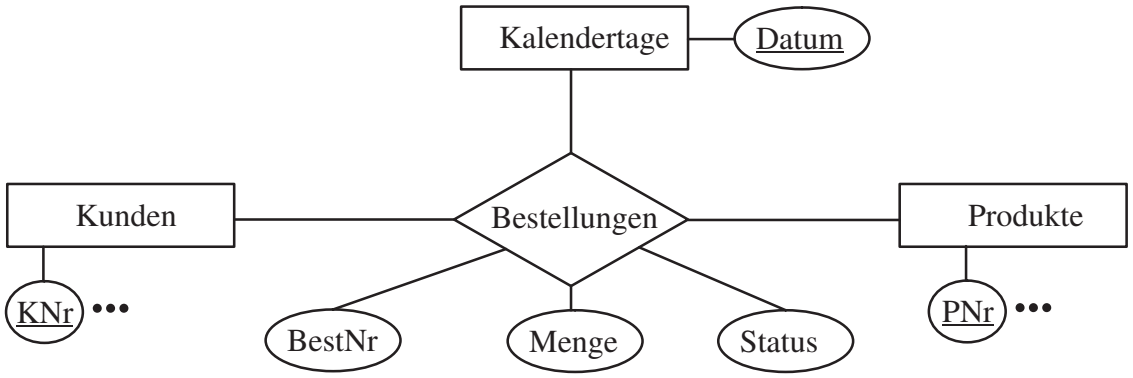
**Variante 1:**




**Variante 2:**




**Variante 3:**



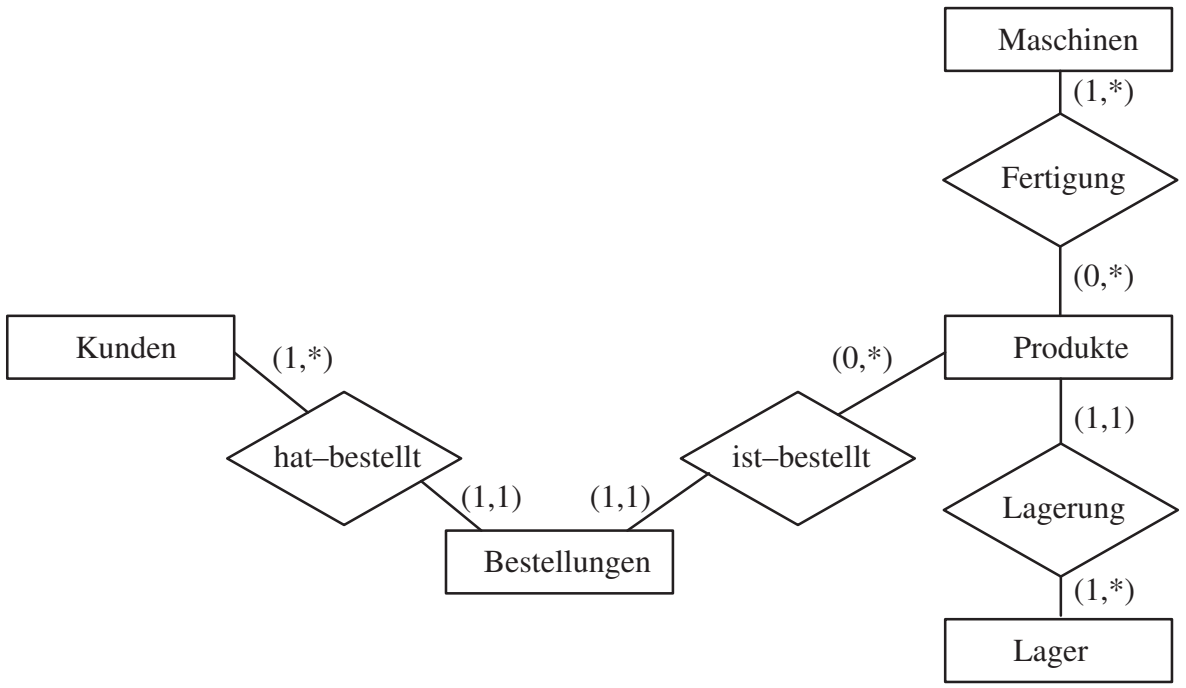

Variante 1 ist inadäquat bzw. inkorrekt, da es mehrere Bestellungen mit demselben Wert für <KNr, PNr> geben kann.
Variante 3 gibt dem Kalender eine unangemessen prominente Position.
Variante 2 ist hier wohl die angemessenste Modellierung.

## Kardinalitätsbedingungen für (Entities in) Relationship−Mengen

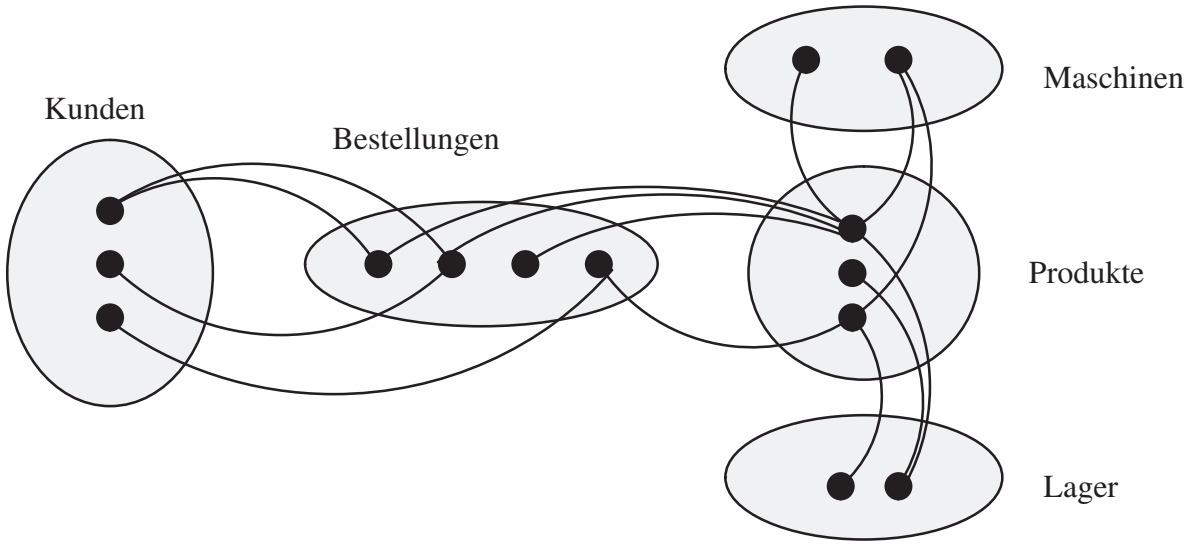
Eine Kardinalitätsbedingung für die Rolle (Bindung, Assoziation) einer Entity−Menge E in einer Relationship−Menge R spezifiziert, wie häufig ein Entity von E in R mindestens vorkommen muß und/oder wie häufig es höchstens vorkommen darf.

Notation:
Die Kante zwischen E und R in einem ERM−Diagramm wird mit (x,y) annotiert, wobei x, y natürliche Zahlen sind und für y auch das Symbol "*" (für "don't care") stehen darf.
x bezeichnet die Untergrenze für die Anzahl der Beziehungen eines Entities, y die Obergrenze. Falls für y das Symbol "*" angegeben ist, ist keine Obergrenze spezifiziert; in diesem Fall wird statt "*" gelegentlich auch "N" geschrieben.

**Beispiel:**

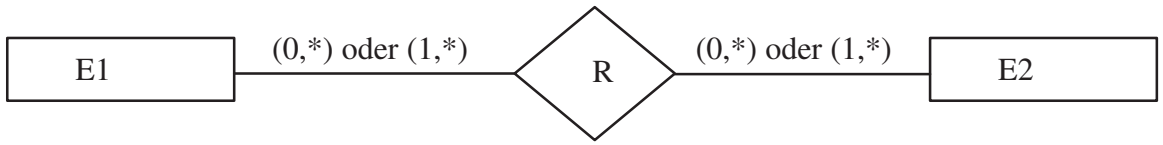



**Beispielausprägung:**

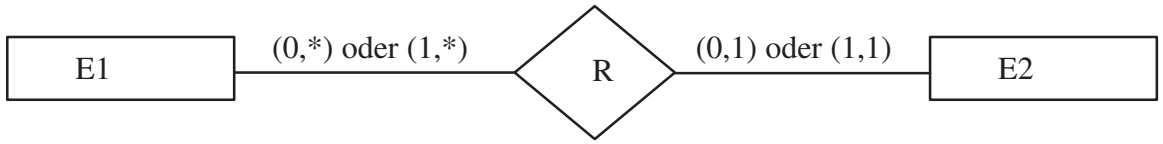


## Verschiedene Arten von Relationships je nach Art der Kardinalitätsbedingung

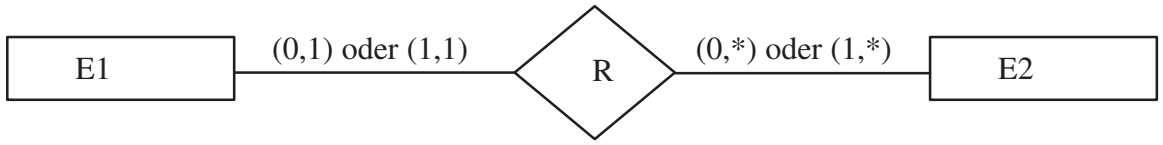
**M:N−Relationship (Viele−zu−viele−Beziehung, komplexe Beziehung)**



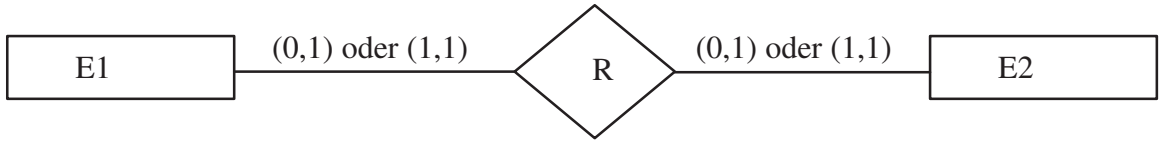

**1:N−Relationship (Eins−zu−viele−Beziehung, hierarchische Beziehung)**



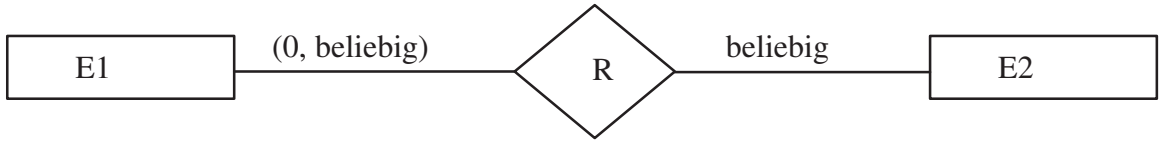

**N:1−Relationship (Viele−zu−eins−Beziehung, hierarchische Beziehung)**



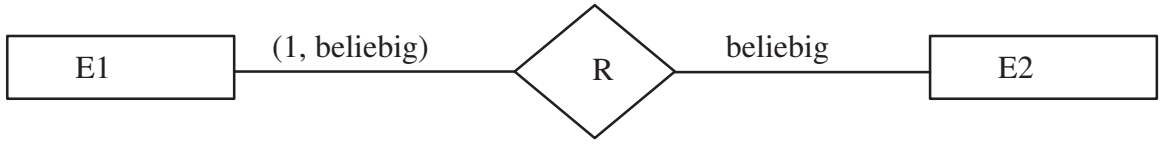

**1:1−Relationship (Eins−zu−eins−Beziehung, injektive Beziehung)**




**Optionale Rolle eines Entities E1 in einer Relationship R**



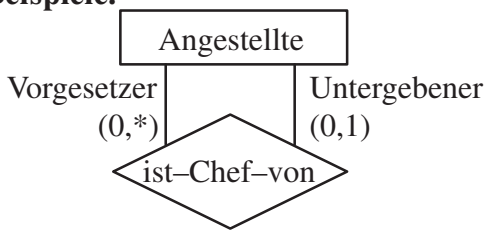

**Verpflichtende Rolle eines Entities E1 in einer Relationship R**

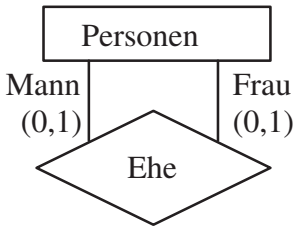


## Rekursive Relationship−Mengen

Eine Relationship−Menge zwischen einer Entity−Menge E und sich selbst heißt *rekursiv*. Um Kardinalitätsbedingungen richtig ausdrücken zu können, werden bei rekursiven Relationship−Mengen die verschiedenen "Rollen" von E benannt. Im ERM−Diagramm werden die entsprechenden Kanten mit den Rollennamen annotiert.

**Beispiele:**

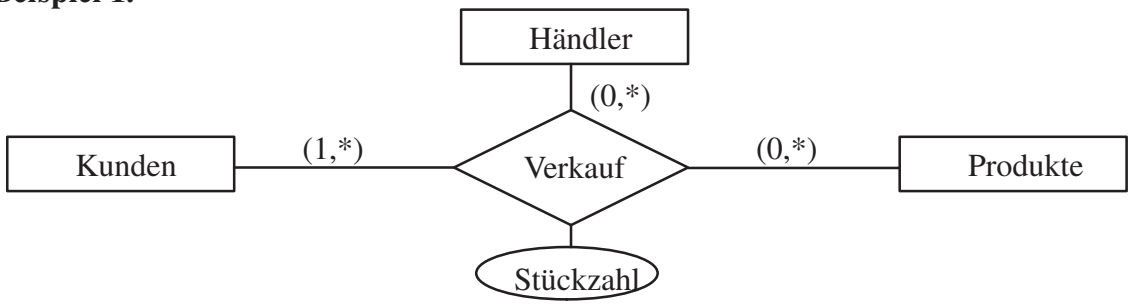



## Relationship−Mengen zwischen mehr als zwei Entity−Mengen

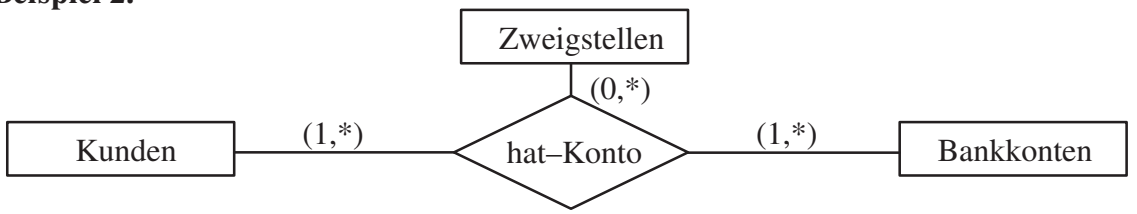
Relationship−Mengen sind nicht notwendigerweise binär. In vielen Anwendungen kommen ternäre Relationship−Mengen vor. Einige davon lassen sich in mehrere binäre Relationship−Mengen zerlegen, bei anderen ist eine solche Zerlegung nicht möglich.

**Beispiel 1:**

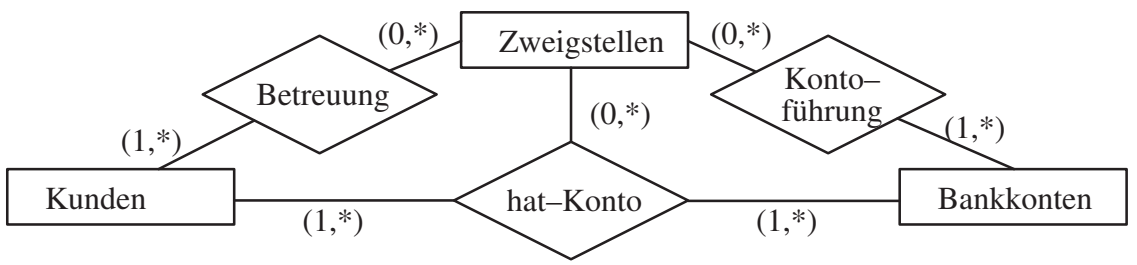



Verkauf ist nicht in mehrere binäre Relationship−Mengen zerlegbar.

**Beispiel 2:**




Die Relationship−Menge hat−Konto ist zerlegbar in drei binäre Relationship−Mengen. Achtung: Die Zerlegung verändert aber die Semantik der Relationships in subtiler Weise. Beispielsweise ist die Betreuung jetzt unabhängig davon, ob der Kunde ein Konto bei einer Zweigstelle hat.

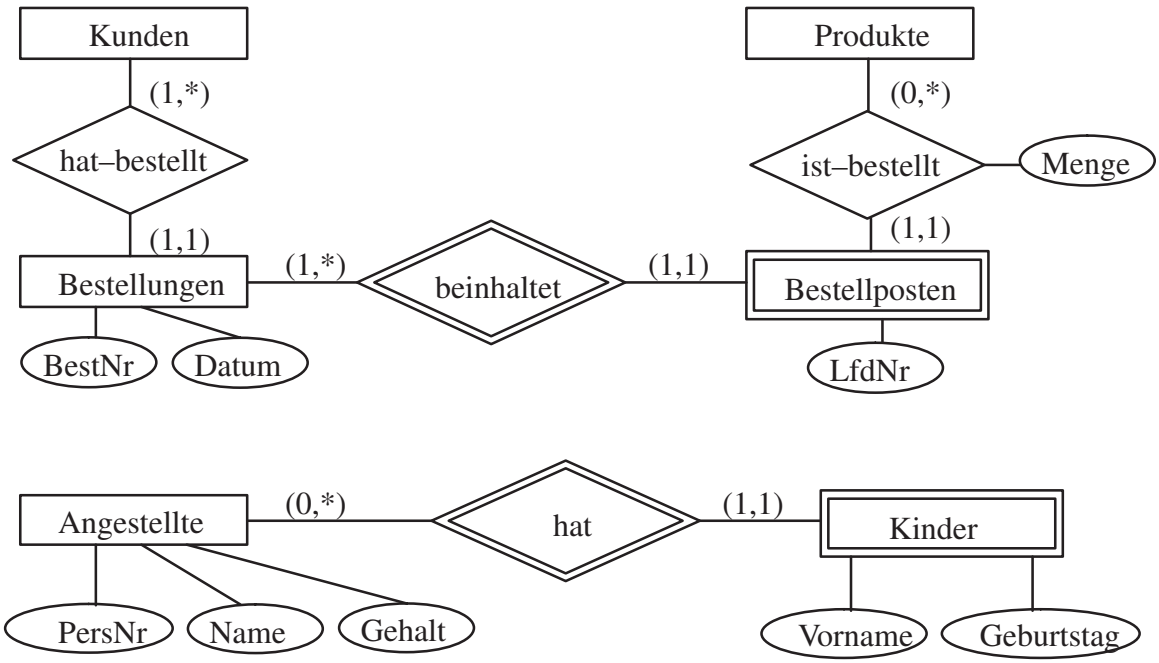


## Schwache Entity−Mengen und Schwache Relationships

Eine schwache Entity−Menge S ist eine Entity−Menge, die in einer hierarchischen N:1−Relationship R zu einer anderen (nichtschwachen) Entity−Menge E steht, so daß der Primärschlüssel von S den Primärschlüssel von E enthält. R heißt dann auch schwache Relationship, wobei S eine verpflichtende Rolle in R hat.
Ein Entity einer schwachen Entity−Menge ist nur in Verbindung mit dem entsprechenden nichtschwachen Entity für die modellierte Anwendungswelt interessant.

Darstellung:
Die schwache Entity−Menge S und die schwache Relationship R werden in ERM−Diagrammen speziell markiert. Der Primärschlüssel von E wird bei S nicht explizit aufgeführt.

**Beispiele:**

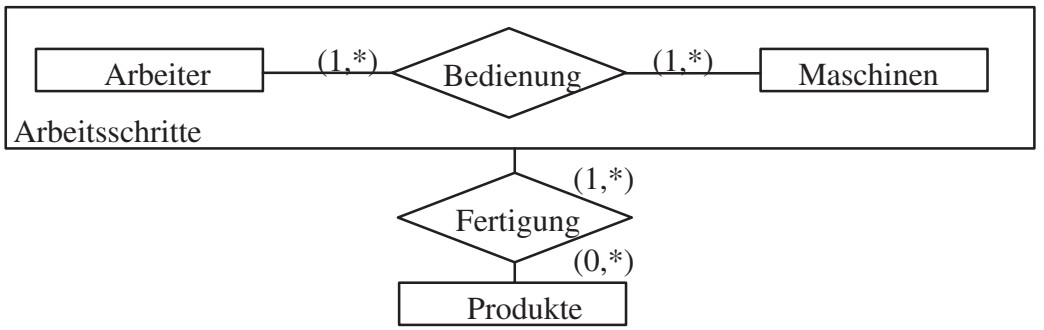



## Aggregation

Mehrere Entities mit ihren Relationships können selbst wiederum zu einem *zusammengesetzten Entity* (komplexen Objekt) zusammengefaßt werden. Die entsprechende Entity−Menge heißt aggregierte Entity−Menge.

Bemerkung: Aggregation ist in vielen Semantischen Datenmodellen nur in eingeschränkter Form vorgesehen.

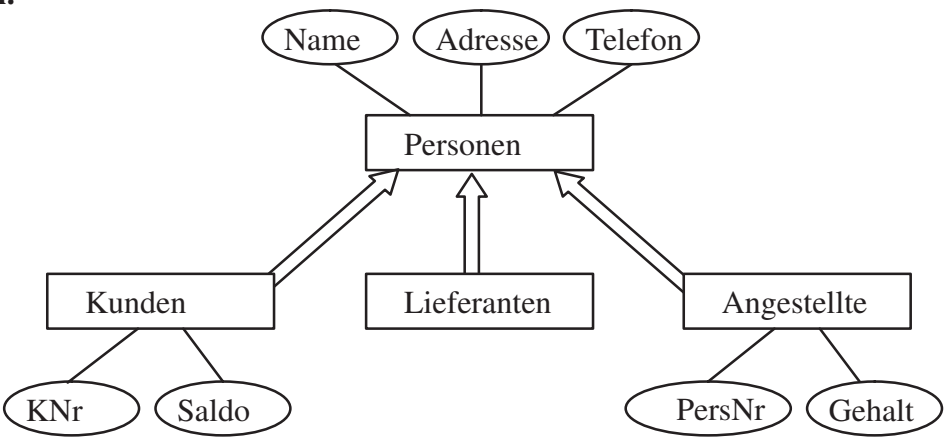
**Beispiel:**



## ISA−Relationships (Generalisierungsbeziehungen)

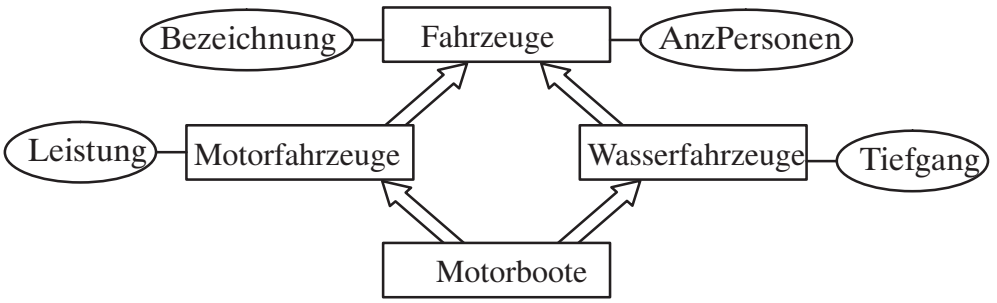
Eine binäre 1:1−Relationship−Menge R zwischen zwei Entity−Mengen E und F heißt *ISA−Relationship−Menge,* wenn die Attributmenge von F eine Obermenge der Attributmenge von E ist und die Teilmengenbeziehung F $\subseteq$ E gilt.
Man sagt dann "F is a E" und bezeichnet F als *Spezialisierung* (Subklasse) von E und E als *Generalisierung* (Superklasse) von F. Ferner bezeichnet man den Entity−Typ von F als einen Subtyp des Entity−Typs von E und den Entity−Typ von E als einen Supertyp des Entity−Typs von F.

**Beispiel:**




Mit der Generalisierung lassen sich ganze Hierarchien aufbauen, die häufig die morphologische Gliederung eines Anwendungsfelds modellieren.
Beispiele:
1) Wirbeltiere − Säugetiere − Katzen − Tiger − Sibirische Tiger
2) Bankkonten − Sparkonten − Jugendsparkonten
3) Finanzinstrumente − Derivative Instrumente − Optionen − Aktienindexoptionen − DAX−Optionen

Generalisierungshierarchien müssen nicht notwendigerweise Bäume sein.
Beispiel:




**Verschiedene Arten von Generalisierungsbeziehungen**

Seien F1, ..., Fn Spezialisierungen der Entity−Menge E.
Die Generalisierungsbeziehungen zwischen F1, ..., Fn und E heißen *disjunkt*, wenn für je zwei verschiedene Spezialisierungen Fi, Fj gilt: Fi $\cap$ Fj = $\emptyset$.

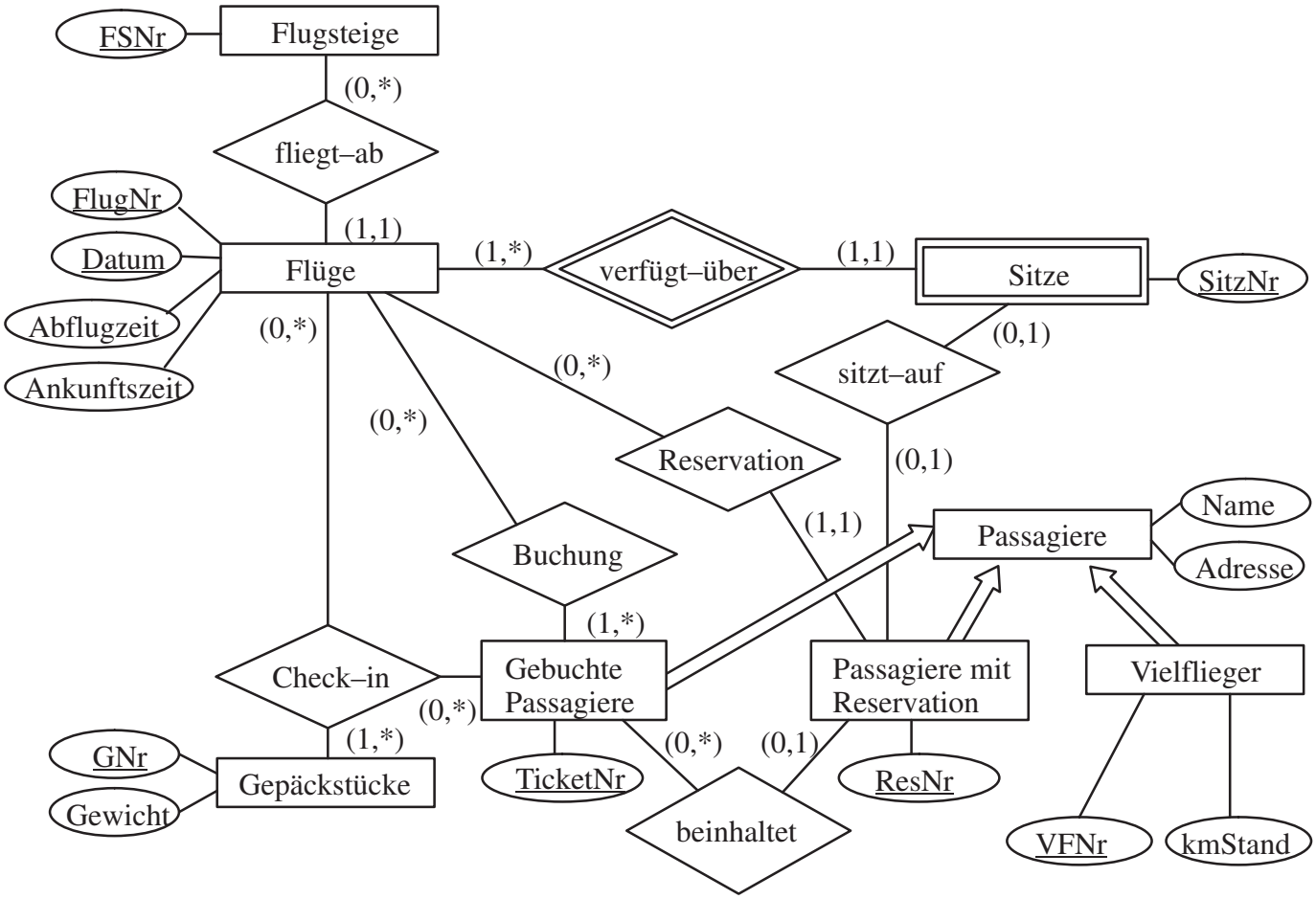
Sie heißen *vollständig−überdeckend*, wenn gilt: $\bigcup_{i=1}^{n}$ Fi = E.

## Entwurfsmethodologie (grob)

1) Abstecken des Problemrahmens (Analyse des Informationsbedarfs)
2) Spezifikation von Entity−Mengen mit ihren Attributen
3) Festlegung von Primärschlüssel für die Entity−Mengen
4) Bildung von Generalisierungsbeziehungen
5) Spezifikation der relevanten Relationship−Mengen
6) Spezifikation von Kardinalitätsbedingungen und ggf. weiteren Integritätsbedingungen

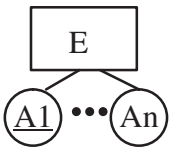
**Beispiel:**

Es sollen die Informationszusammenhänge für ein Flugbuchungssystem einer Fluggesellschaft modelliert werden.
Flüge werden durch eine Flugnummer identifiziert, die für Flüge am selben Tag eindeutig ist.
Passagiere können einen Flug reservieren, was durch eine Reservationsnummer bestätigt wird.
Eine Reservation wird zu einer festen Buchung, indem man ein Ticket käuft.
Bei der Reservation oder später können Passagiere auch eine Sitzplatzreservierung vornehmen.
Für Teilnehmer des Vielfliegerprogramms ist die gesamte mit der Fluggesellschaft geflogene Kilometerzahl von Bedeutung.
Flüge fliegen von einem bestimmten Flugsteig ab. Passagiere müssen vor dem Abflug eine Check−in−Prozedur durchlaufen. Dabei können sie auch Gepäckstücke aufgeben.



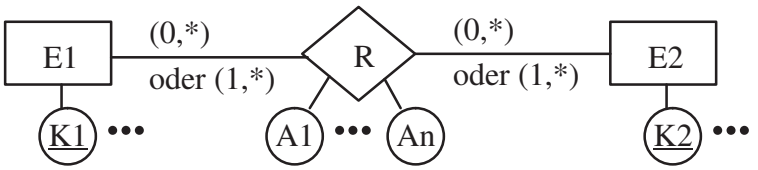
# 14.2 Abbildung eines ERM−Entwurfs auf Relationen

**Regel 1: Jede Entity−Menge bildet eine Relation**



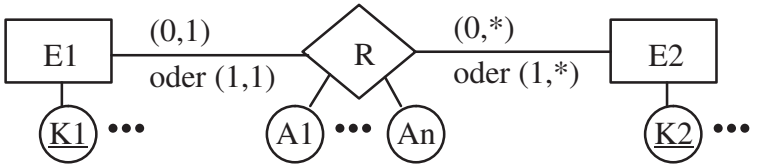

→ **E (A1, ..., An)**

**Regel 2: M:N−Relationships bilden eigene Relationen**



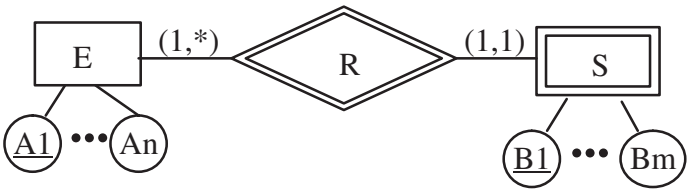

→ **R (K1, K2, A1, ..., An)**

**Regel 3: N:1−, 1:N und 1:1−Relationships können in eine "Entity−Relation" integriert werden**



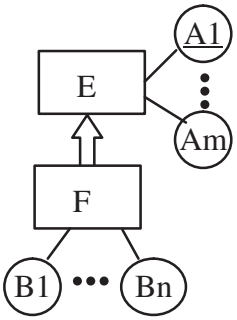

→ **E1 (K1, ..., A1, ..., An, K2)**
**E2 (K2, ...)**

**Regel 4: Eine schwache Entity−Menge bildet eine eigene Relation**




→ **E (A1, ..., An)**
**S (A1, B1, ..., Bm)**

**Regel 5: Generalisierung und Spezialisierung bilden jeweils eine eigene Relation**




→ **E (A1, ..., Am)**
**F (A1, B1, ..., Bn)**

**oder**

→ **E (A1, ..., Am)**
**F (A1, ..., Am, B1, ..., Bn)**

**wobei die Relation E dann nur Tupel enthält, die Entities aus E − F entsprechen**

**Flugbuchungs−Beispiel:**

Flüge (FlugNr, Datum, Abflugzeit, Ankunftszeit, FSNr)
Flugsteige (FSNr)
Sitze (FlugNr, SitzNr)
PassagiereMitReservation (ResNr, Name, Adresse, FlugNr, Datum, SitzNr, TicketNr)
GebuchtePassagiere (TicketNr, Name, Adresse)
Buchungen (TicketNr, FlugNr, Datum)
Vielflieger (VFNr, Name, Adresse, kmStand)
Gepäckstücke (GNr, Gewicht)
Check−In (FlugNr, Datum, TicketNr, GNr)

# Ergänzende Literatur zu Kapitel 14:


Batini, C., Ceri, S., Navathe, S.B., Conceptual Database Design, Benjamin/Cummings, 1992

Teorey, T.J., Database Modelling and Design, Morgan Kaufmann, 1990

P. Chen, The Entity Relationship Model − Toward a Unified View of Data, ACM Transactions on Database Systems Vol.1 No.1, March 1976, pp. 9−36

L.A. Maciaszek, Database Design and Implementation, Prentice Hall, 1990

S.M. Lang, P.C. Lockemann, Datenbankeinsatz, Springer−Verlag, 1995