# 6. Modellierung von Strukturen

In Kapitel 4 haben wir bereits <u>Graphen</u> und <u>Bäume</u> als Möglichkeiten kennengelernt, mit denen sich Objekte sowie Beziehungen zwischen je 2 Objekten gut modellieren lassen.

In Kapitel 5.2 wurden Verallgemeinerungen davon eingeführt, die so genannten $\sigma$-Strukturen, wobei $\sigma$ eine Signatur ist. Abgesehen von Graphen und Bäumen kann man damit beispielsweise auch die natürlichen (oder die rationalen) Zahlen mit arithmetischen Operationen $+, \times$ etc., modellieren oder – wie in Kapitel 5.6 gesehen – auch relationale Datenbanken, die z.B. Informationen über Kinofilme und das aktuelle Kinoprogramm enthalten.

In Kapitel 6 werden wir nun zwei weitere Kalküle kennenlernen, mit denen man strukturelle Eigenschaften von Systemen beschreiben kann: das Entity-Relationship-Modell und kontextfreie Grammatiken.

# 6.1 Das Entity-Relationship-Modell

Das Entity-Relationship-Modell (kurz: ER-Modell) geht zurück auf einen grundlegenden Artikel von P.P. Chen aus dem Jahr 1976:

P.P. Chen: "The Entity-Relationship-Model — Towards a Unified View of Data". ACM Transactions on Database Systems, Band 1, Nr. 1, Seiten 9-36, 1976.

Es wird heute praktisch als Standardmodell für frühe Entwurfsphasen in der Datenbankentwicklung eingesetzt. Darüber hinaus basiert auch die Spezifikationssprache UML ("Unified Modeling Language"), die zur Spezifikation von Strukturen und Beziehungen in Software-Systemen eingesetzt wird, auf dem ER-Modell.

In dieser Vorlesung werden wir einige grundlegende Züge des ER-Modells vorgestellt. Details können Sie z.B. in der Veranstaltung "Datenbanksysteme I" kennenlernen.

Das ER-Modell basiert auf den 3 Grundkonzepten

- <u>Entity</u>: "zu modellierende Informationseinheit" (deutsch: "Objekt", "Ding", "Entität")
- <u>Relationship</u>: zur Modellierung von Beziehungen zwischen Entities (deutsch: "Beziehung", "Relation")
- <u>Attribut</u>: Eigenschaft von einem Entity oder einer Beziehung.

<u>Genauer</u>:

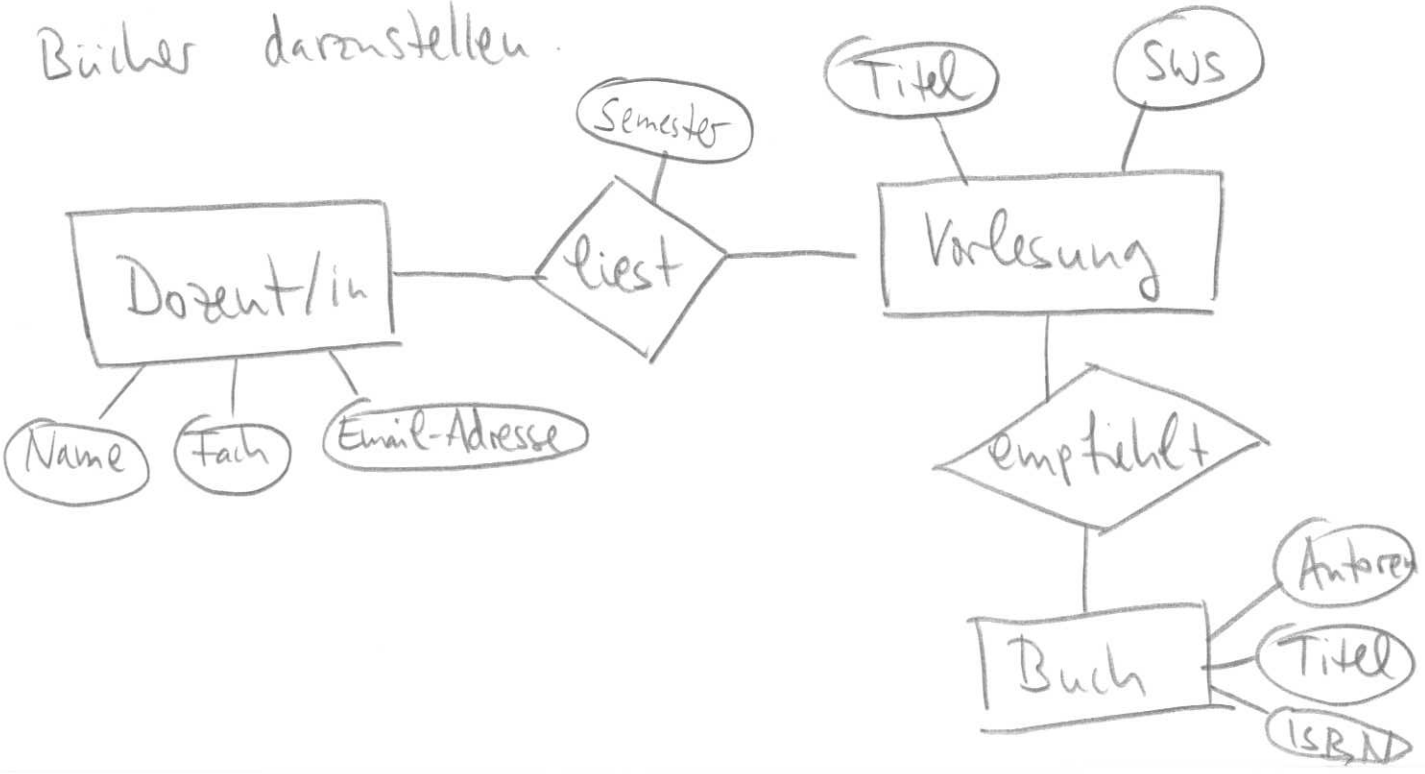
- <u>Entity</u>: Objekt der realen oder der Vorstellungswelt, über das Informationen zu speichern sind
  z.B. eine Vorlesungsveranstaltung, ein Buch oder eine/n Dozent/in.
  Auch Informationen über Ereignisse wie Klausuren können Objekte im Sinne des ER-Modells sein.
- <u>Entity-Typ</u> (bzw. <u>Entity-Menge</u>): eine Zusammenfassung von Entities, die im Modell als "gleichartig" angesehen werden.
  z.B.: "Vorlesung", "Buch", "Dozent/in"

Im Modell steht ein Entity-Typ für die Menge aller in Frage kommenden Objekte dieser Art.

- <u>Relationship</u>: Beziehung zwischen Entities, z.B.: welche Dozenten/innen welche Vorlesungen halten
- <u>Attribut</u>: Eigenschaft von Entities oder Relationships, z.B. die ISBN eines Buchs, der Titel einer Vorlesung oder die Semester, in denen Vorlesung X von Dozent/in Y gehalten wird.

<u>Beispiel 6.1</u>:

Graphische Darstellung für eine Modellierung im ER-Modell — im Beispiel geht es darum, Vorlesungen, Dozenten und für die Vorlesung empfohlene Bücher darzustellen.



- Entity-Typen werden als Rechtecke dargestellt (hier: Dozent/in, Vorlesung, Buch)

- Eigenschaften von Entities, sog. Attribute, werden durch Ellipsen dargestellt, die mit dem Rechteck des zugehörigen Entity-Typs verbunden sind (im Beispiel hat jede/r Dozent/in die Attribute "Name", "Fach" und "Email-Adresse")

Ein Attibut ordnet jeder Entity des entsprechenden Entity-Typs einen Wert zu.

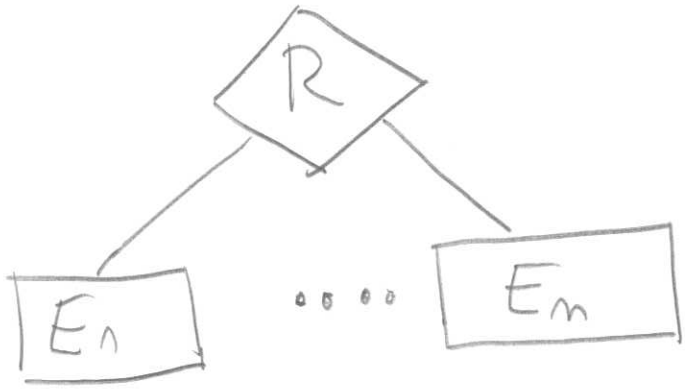
Ein Attibut, dessen Wert jede Entity eindeutig bestimmt (z.B. die ISBN von Büchern), heißt Schlüsselattribut.
Um Schlüsselattribute im ER-Modell explizit zu kennzeichnen, werden sie unterstrichen.
Auch mehrere Attribute zusammen können einen Schlüssel bilden, z.B.

- Typen von Relationships, sog. <u>Relationen-Typen</u>, werden durch Rauten dargestellt, die mit den betreffenden Entity-Typen durch Striche verbunden sind.
  (z.B. ist in Bsp 6.1 "liest" ein Relationen-Typ, der angibt, welche/r Dozent/in welche Vorlesung liest).

<u>Allgemein gilt</u>: Ein Relationen-Typ modelliert Beziehungen zwischen den Entities der betroffenen Entity-Typen.

Ein <u>n-stelliger Relationen-Typ</u> R (für $n \geq 2$) verknüpft Entities aus n Entity-Typen $E_1, \ldots, E_n$.

Er wird graphisch repräsentiert durch

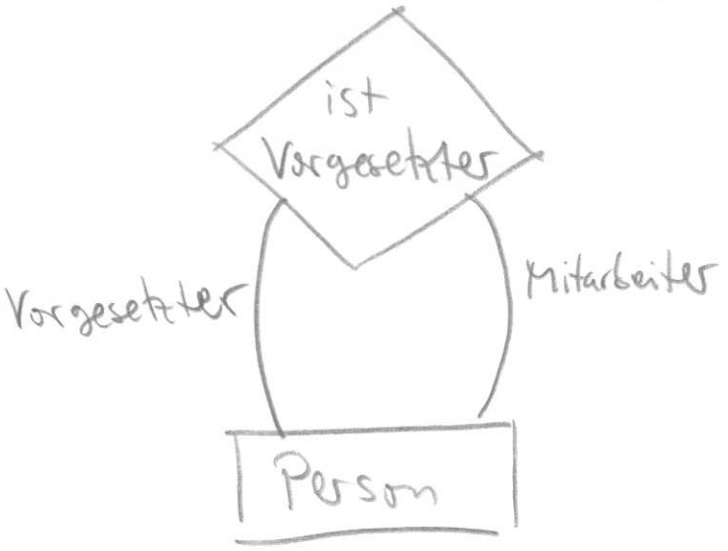



Eine <u>konkrete Ausprägung des Relationen-Typs R</u> ist eine Menge von n-Tupeln $(e_1, \ldots, e_n)$, wobei für jedes $i \in \{1, \ldots, n\}$ gilt: $e_i$ ist ein Entity des Entity-Typs $E_i$.

- Auch Relationen-Typen können Attribute haben (z.B. hat der Relationen-Typ "liest" in Bsp 6.1 ein Attribut "Semester", das angibt, in welchen Semestern Dozent/in X die Vorlesung Y hält).

<u>Allgemein gilt:</u> Ein Attribut ordnet jedem Tupel des entsprechenden Relationen-Typs einen Wert zu.
Beispielsweise ordnet das Attribut "Semester" jedem Tupel (X, Y) der "liest"-Relation die Liste aller Semester zu, in denen Dozent/in X die Vorlesung Y hält.

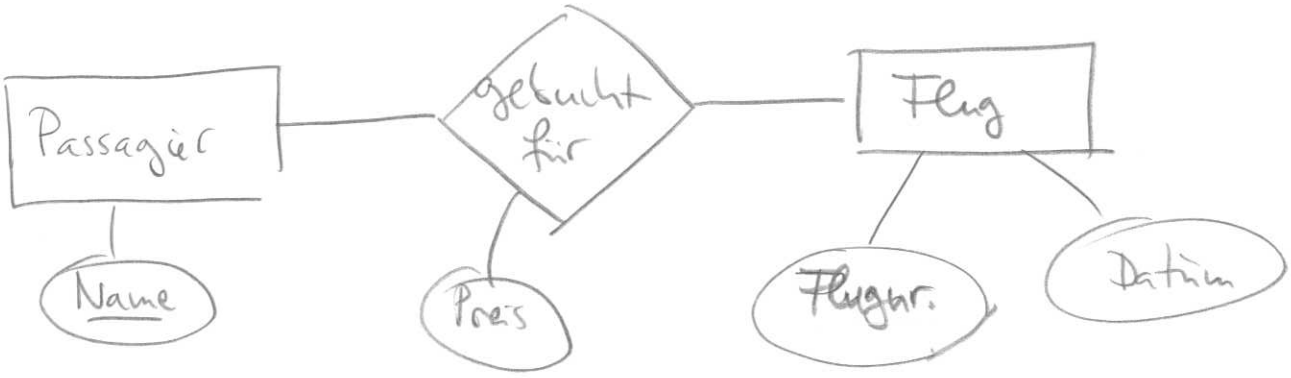
- Für manche Relationen-Typen wird aus ihrem Namen und der graphischen Darstellung zunächst nicht klar, welche Bedeutung die einzelnen Entity-Typen in der Relation haben — insbes. wenn ein Entity-Typ mehrfach am Relationen-Typ beteiligt ist. Es können dann <u>Rollennamen</u> vergeben werden, etwa um die Beziehung "Person X ist Vorgesetzter von Person Y" darzustellen:




Beispiel 6.2

Man beachte die Auswirkungen von Modellierungsentscheiden beim Entwickeln eines ER-Modells:

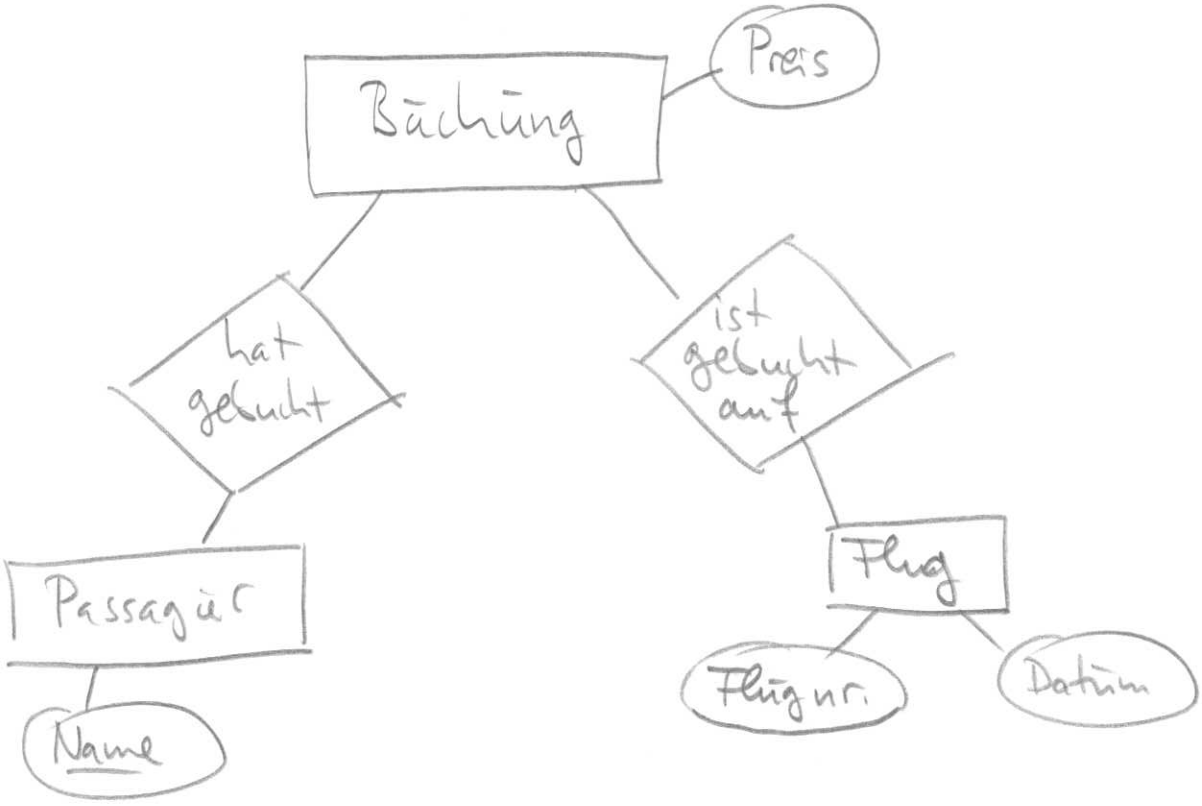
Nutzt ein Reisebüro das ER-Modell




So besteht eine konkrete Ausprägung des Relationen-Typs "gebucht für" aus einer Menge von Tupeln (X, Y), die angibt, dass Person X ein Ticket für Flug Y gebucht hat – und zwar zum Preis Preis(X, Y). Insbesondere heißt dies aber, dass

Passagier X für Flug Y nicht zwei verschiedene Buchungen getätigt haben kann.

Wenn man solche "Mehrfachbuchungen" zulassen will, kann man das folgende ER-Modell benutzen:

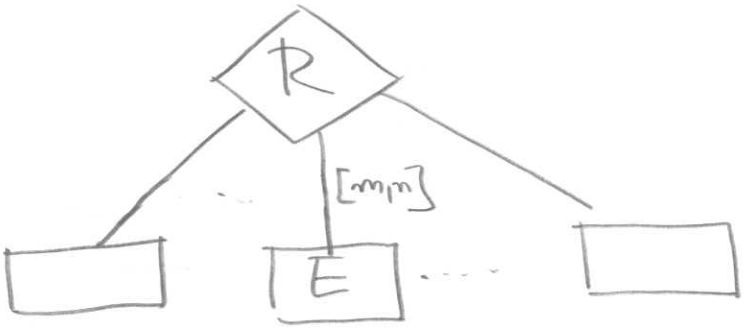


# Ein weiterer Bestandteil von ER-Modellen

## Kardinalität von Relationen-Typen:

Relationen-Typen in der Form, wie wir sie bisher eingeführt haben, sagen über konkrete Ausprägungen nur aus, dass einige Entities aus den beteiligten Entity-Typen in der angegebenen Beziehung stehen können. Oft will man aber genauere Angaben (bzw. Einschränkungen) machen — z.B., dass jeder Angestellte durch eine Relation des Relationen-Typs "arbeitet in" mit <u>genau einer</u> Abteilung verbunden ist. Dies kann graphisch folgendermaßen dargestellt werden:


Allgemein besagt ein Relationen-Typ der Form



dass für jede konkrete Ausprägung dieses Typs gelten muss:

Jedes Entity e der konkreten Ausprägung des Entity-Typs E kommt in mindestens m und höchstens n Tupeln vor.

Spezialfälle für [m,n]:

- [1,1] bedeutet: "in genau einem Tupel"
- [0,1] bedeutet: "in höchstens einem Tupel"
- [0,*] bedeutet: "in beliebig vielen Tupeln"
  Die Angabe [0,*] wird oft auch einfach weggelassen.

Kurznotation für 2-stellige Relationen-Typen:


bedeutet

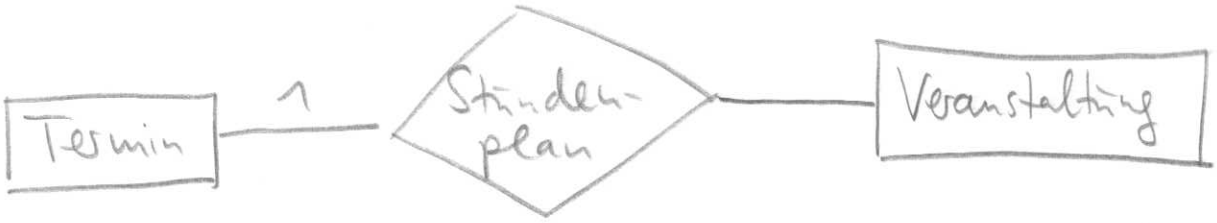

d.h: "jedes Entity a des Typs A kommt in höchstens einem Tupel von R vor, und jedes Entity b des Typs B kommt in beliebig vielen Tupeln von R vor."

Beispiel 6.3:

(a)

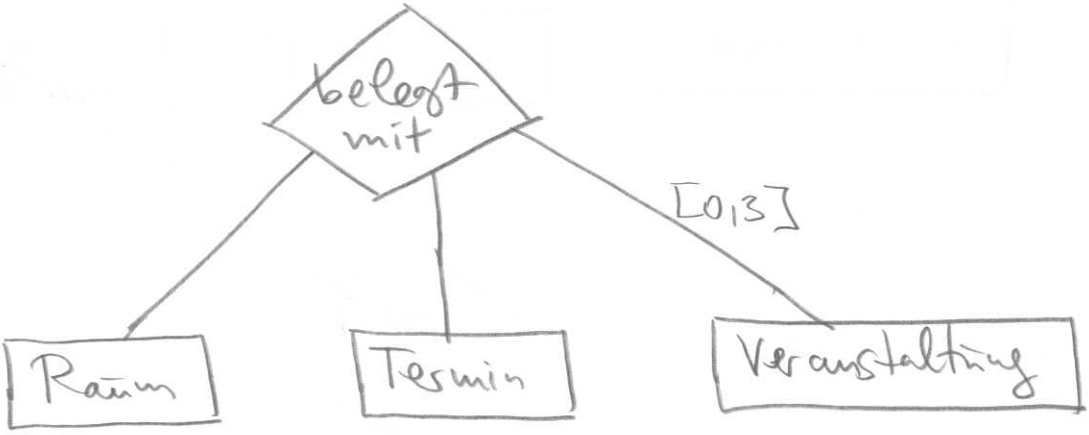

bedeutet: "Jedes Buch wird von mindestens einer Person geschrieben".

(b)




bedeutet: "Jeder Termin im Stundenplan ist mit höchstens einer Veranstaltung belegt"

(c)



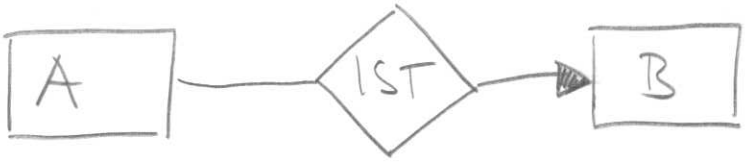

bedeutet: "Jede Veranstaltung wird höchstens dreimal (pro Woche) angeboten.

# Noch ein Bestandteil von ER-Modellen:

## Die IST-Beziehung (englisch "is-a"):

Der spezielle Relationen-Typ IST definiert eine Spezialisierungs-Hierarchie.

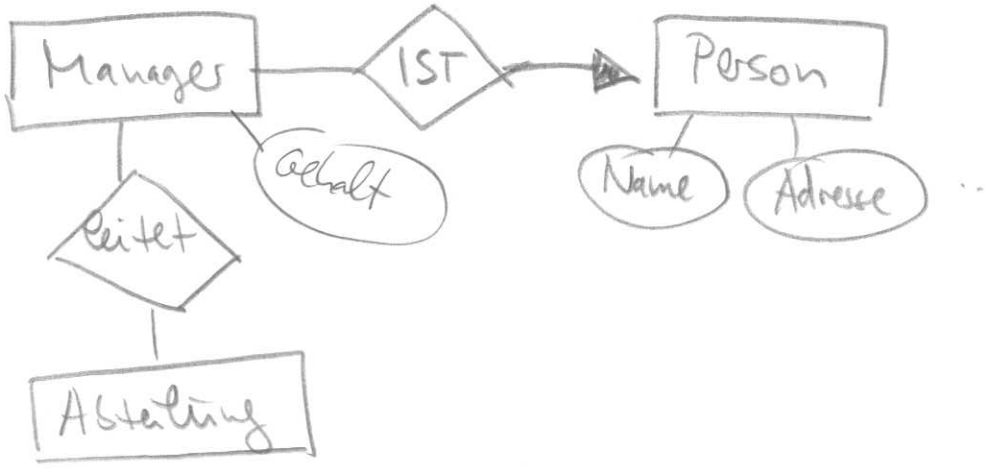
Graphische Darstellung:




Bedeutung: Einige Entities des Typs B sind auch Entities des Typs A
(d.h A ist eine Spezialisierung des Typs B).

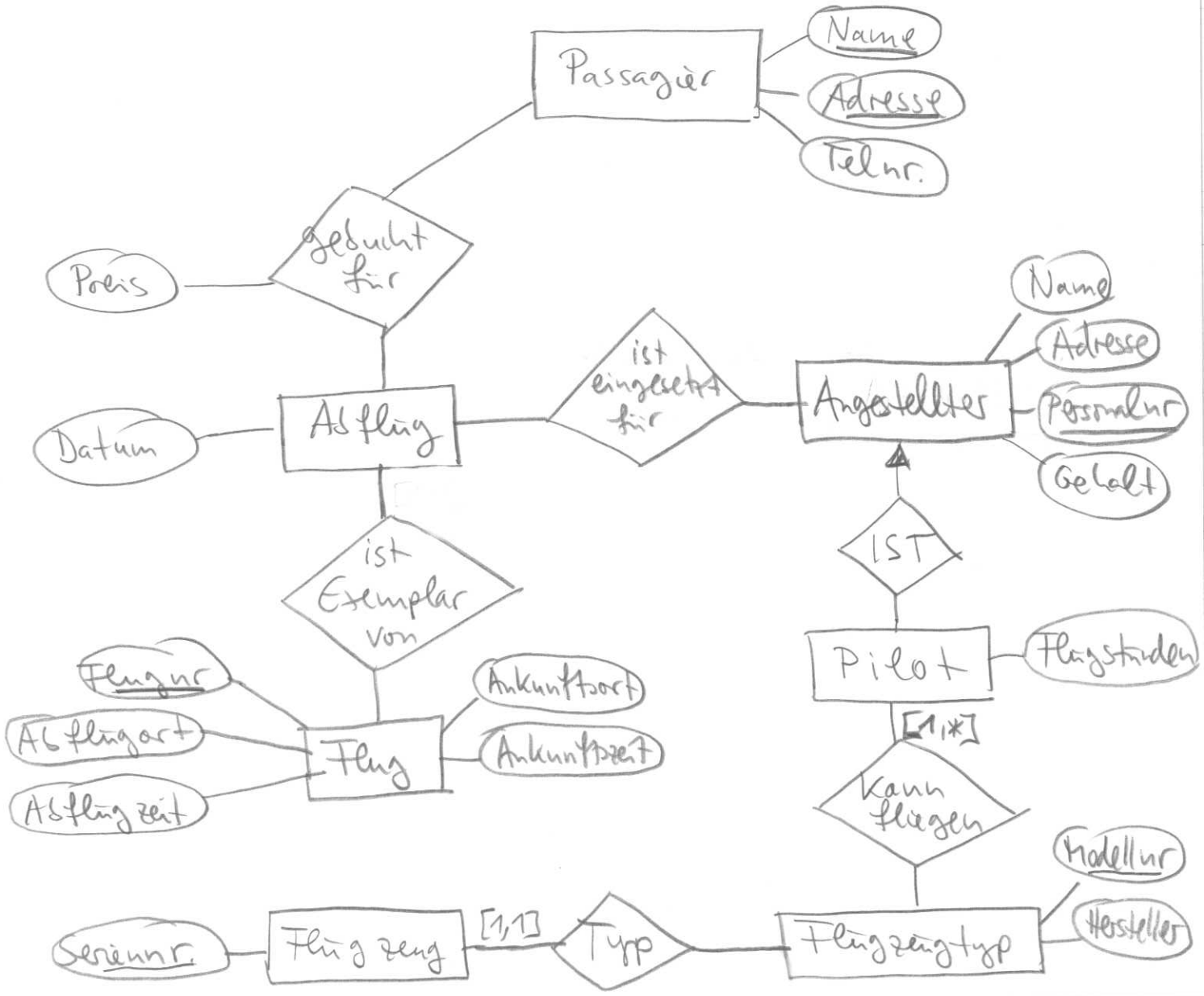
Andere Formulierung: Jedes Entity des Typs A ist auch ein Entity des Typs B.

Beispiel:



Allgemein gilt:
Die Entities des Typs A "erben" alle Attribute von B und können außerdem noch weitere Attribute haben, die spezielle "A-Eigenschaften" beschreiben.
Auch Schlüsselattribute werden als solche geerbt.

<u>Beispiel 6.4</u>: Ein umfangreiches ER-Modell, das einige Aspekte einer Fluggesellschaft modelliert:



In diesem ER-Modell wird u. a. folgendes modelliert:

1) Es kommt eine IST-Spezialisierung vor, die besagt, dass Piloten spezielle Angestellte sind.
Dies wird insbes. benötigt, um den Relationen-Typ "kann fliegen" hinreichend präzise formulieren zu können und das Attribut "Flugstunden" nicht allen Angestellten zuordnen zu müssen.

2) Entities aller hier aufgeführten Typen (bis auf Abflug) werden durch Schlüsselattribute eindeutig identifiziert. Bei den Passagieren wird angenommen, dass Name und Adresse den jeweiligen Passagier eindeutig festlegen. Die Namen der übrigen Schlüsselattribute deuten an, dass man jeweils eine Nummerierung eingeführt hat, um die Eindeutigkeit zu erreichen (z.B. Personalnr, Flugnr etc).

3) Der Relationen-Typ "Typ" verbindet konkrete Flugzeuge mit Flugzeugtypen, indem sie jedem Flugzeug genau einen Flugzeugtypen zuordnet.
Solche Unterscheidungen zwischen "Typ" und "Exemplar" werden oft in Modellierungen verwendet und sind wichtig, damit Relationen und Attribute sachgerecht zugeordnet werden können. Beispielsweise sind die

Fähigkeiten eines Piloten dadurch bestimmt, welche Flugzeugtypen er fliegen kann – und nicht durch die konkreten Flugzeuge(exemplare), die er fliegen kann.

Vorsicht: Beim ersten Hinsehen mag es verlockend erscheinen, Typ-Exemplar-Beziehungen durch die IST-Spezialisierung zu modellieren.
Dies ist aber ein schwerer Entwurfsfehler:
"Typen" und "Exemplare" bezeichnen verschiedenartige Entity-Typen, zwischen denen i.d.R. keine Teilmengen-Beziehung (wie bei der IST-Spezialisierung) bestehen kann.

# 6.2 Kontextfreie Grammatiken

Kontextfreie Grammatiken (kurz: KFGs) eignen sich besonders gut zur Modellierung von beliebig tief geschachtelten baum-artigen Strukturen.

KFGs können gleichzeitig

- hierarchische Baumstrukturen und
- Sprachen und deren textuelle Notation

spezifizieren.

KFGs werden z.B. angewendet zur Definition von

- Programmen einer Programmiersprache und deren Struktur, z.B. Java, C, Pascal (KFGs spielen z.B. beim "Compilerbau" eine wichtige Rolle)
- Datenaustauschformaten, d.h. Sprachen als Schnittstelle zwischen Software-Werkzeugen, z.B. HTML, XML
- Bäumen zur Repräsentation strukturierter Daten, z.B. XML
- Strukturen von Protokollen beim Austausch von Nachrichten zwischen Prozessen oder Geräten

KFGs sind ein grundlegender Kalkül für die formale Definition von Sprachen eingesetzt wird

In dieser Veranstaltung werden nur die Grundbegriffe und einige Beispiele vorgestellt; im Detail werden KFGs in der Veranstaltung "GL-2" behandelt.

<u>Definition des Begriffs "Kontextfreie Grammatik"</u>:

Es gibt 2 Sichtweisen auf KFGs:

1) Eine KFG ist ein spezielles <u>Ersetzungssystem</u>. Seine Regeln geben an, auf welche Art man ein Symbol durch eine Folge von Symbolen ersetzen kann. Auf diese Weise definiert eine KFG eine <u>Sprache</u>, dh eine <u>Menge von Worten</u> über einem bestimmten Alphabet, die mit den durch die KFG gegebenen Regeln erzeugt werden können

2) Gleichzeitig definiert eine KFG eine <u>Menge von Baumstrukturen</u>, die sich durch schrittweises Anwenden der Regeln erzeugen lassen.

Für die Modellierung von Strukturen ist die zweite Sichtweise besonders interessant. Aber es ist oft sehr nützlich, dass dasselbe Kalkül auch gleichzeitig

eine textuelle Notation für die Baumstrukturen liefern kann und dass Eigenschaften der zugehörigen Sprache untersucht werden können.

<u>Definition 6.5</u>: (KFG)

Eine kontextfreie Grammatik $G = (T, N, S, P)$ besteht aus

- einer endlichen Menge $T$, der so genannten Menge der <u>Terminalsymbole</u> (die Elemente aus $T$ werden auch <u>Terminale</u> genannt)
- einer endlichen Menge $N$, der sog. Menge der <u>Nichtterminalsymbole</u> (die Elemente aus $N$ werden auch <u>Nichtterminale</u> genannt).

  Die Mengen $T$ und $N$ sind disjunkt, d.h. $T \cap N = \emptyset$.

  Die Menge $V := T \cup N$ heißt <u>Vokabular</u>; die Elemente in $V$ nennt man auch <u>Symbole</u>.
- einem Symbol $S \in N$, dem sog. <u>Startsymbol</u>
- einer endlichen Menge $P \subseteq N \times V^*$, der sog. Menge der <u>Produktionen</u>.

  Für eine Produktion $(A, x) \in P$ schreiben wir meistens $A \to x$ (bzw. $A ::= x$).

Beispiel 6.6:

Als erstes Beispiel betrachten wir eine KFG für "wohlgeformte Klammerausdrücke":

$G_K := (T, N, S, P)$ mit

- $T := \{ (, ) \}$, d.h.
  $G_K$ hat als Terminale die Symbole "(" ("öffnende Klammer") und ")" ("schließende Klammer")

- $N := \{$ Klammerung, Liste $\}$, d.h.
  $G_K$ hat als Nichtterminale die Symbole "Klammerung" und "Liste"

- $S :=$ Klammerung, d.h.
  "Klammerung" ist das Startsymbol

- $P := \{$ Klammerung $\rightarrow$ ( Liste ),
  Liste $\rightarrow$ Klammerung Liste,
  Liste $\rightarrow$ $\varepsilon$ $\}$

(Zur Erinnerung: $\varepsilon$ bezeichnet das leere Wort).