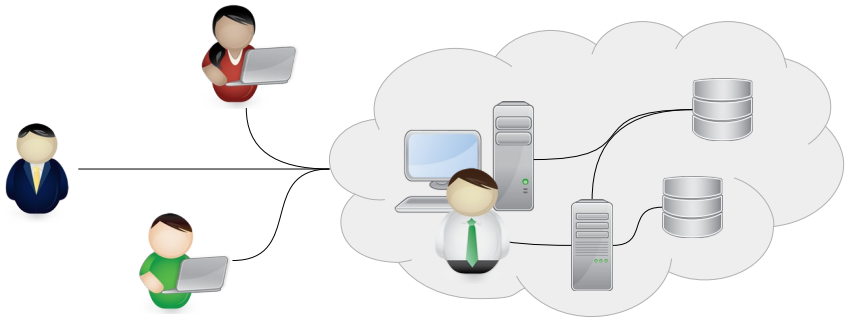
Kapitel 1:

Informationssysteme: ERM + DBM + SQL

Informationssysteme

- „Informationssystem“ (IS)

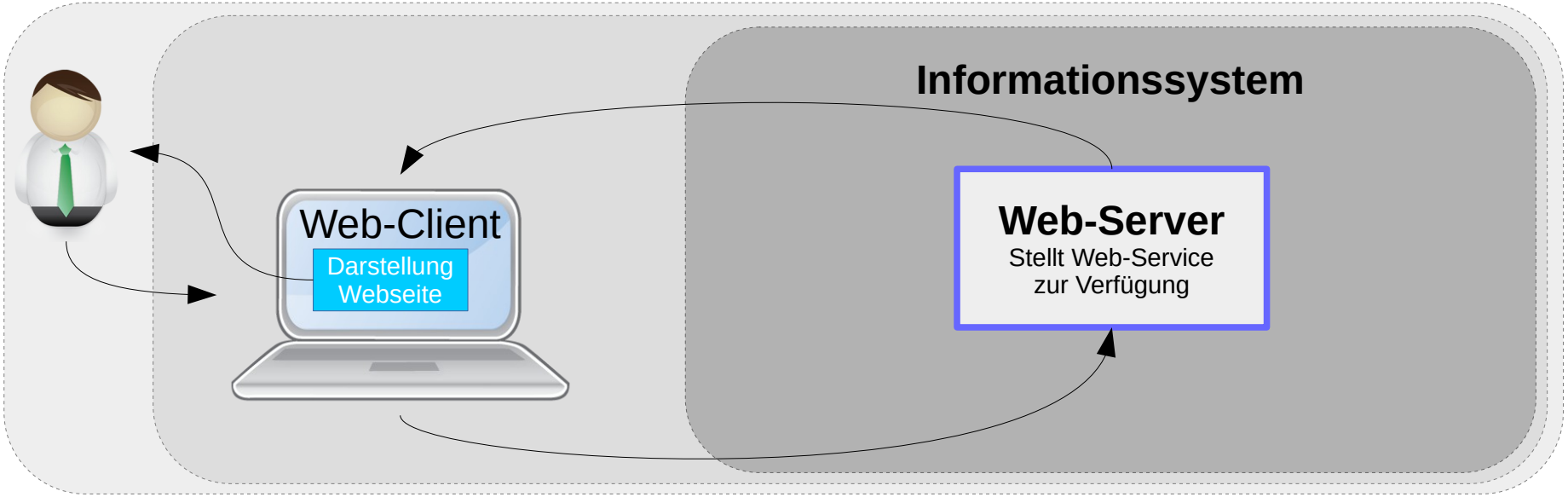


  - Eigenschaften und Ziele
    - produziert, beschafft, verteilt und verarbeitet **Daten** (bzw. **Informationen**)
    - Ziel: Deckung von Informationsnachfrage
    - **Mensch-** / **Aufgabe-** / **Technik-**System (soziotechnisches System)
  - Technische Aspekte
    - Dienste, Server, Kommunikation, Protokolle, Programmiersprachen, …
  - Rolle des Mensch
    - Nutzer von Diensten (innerhalb oder außerhalb des Systems)
    - Funktionsträger (Anbieter von Diensten, Verarbeiter von Daten, …)
  - Abgrenzung des Systembegriffs ist unscharf

Informationssysteme

- **Beispiel: Webservices (bzw. Webserver)**
  - Liefern Informationen auf Basis von eigenen Daten
    - z.B. GET-Request „gib mir die Liste der Vorlesungen“
  - Verarbeiten Informationen, die sie aus Requests erhalten
    - z.B. POST-Request „melde mich für die Vorlesung XYZ an“
  - Informationen werden „*produziert, beschafft, verteilt und verarbeitet*“

→ ***Frage**: Was ist hier ein IS?* → *Viele Sichtweisen!*

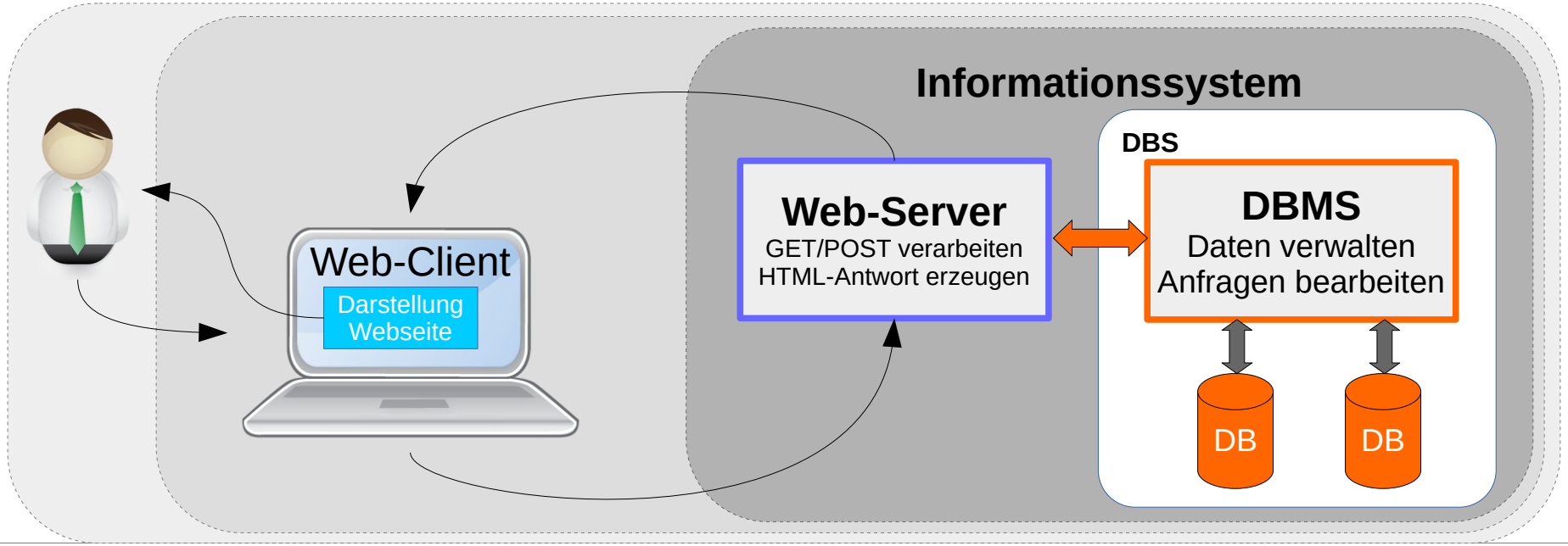


Informationssysteme

- **Webservices**
  - Verarbeiten also **Daten**
    - liefern Informationen, speichern Änderungen, ...
  - Webserver sind aber doch **zustandslos** *(HTTP)*

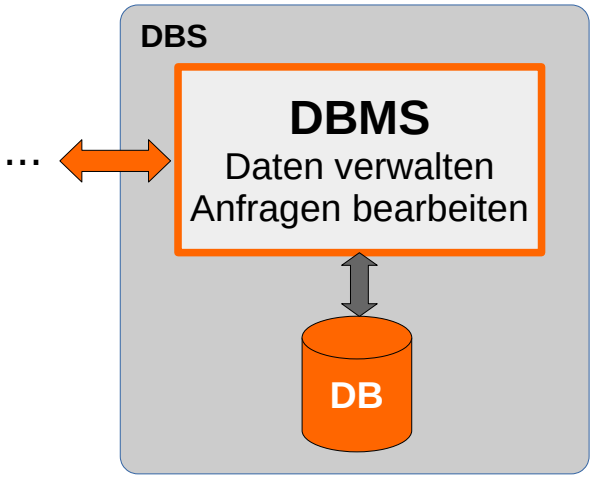
    → Wo kommen die Informationen her?

    → Wo gehen die Änderungen der Informationen hin?



Datenbanksysteme

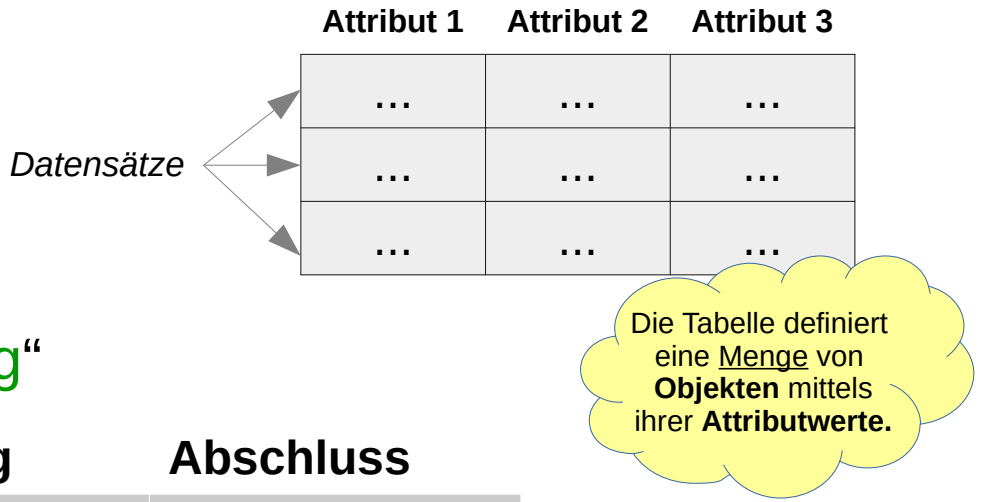
- **Datenbanksystem (DBS)**
  - System zur elektronischen Datenverwaltung
  - Zwei Teile:
    - **Datenbank** (**DB**)
      - also der zu verwaltende Datenbestand
    - **Datenbankmanagementsystem** (**DBMS**)
      - Hard- und Software zur Datenverwaltung
  - Aufgaben:
    - große Datenmengen ***speichern***
      - effizient, konsistent und dauerhaft
    - Daten für Nutzung in der benötigten Form ***bereitstellen***
      - ggf. auch (örtlich) verteilt (effizient, konsistent)



Datenbankmanagementsysteme

- **Wesentliche Funktionen eines DBMS**
  - **Speicherung** / **Änderung** / **Löschen** von Daten
  - Effiziente **Bereitstellung** von Daten über **Anfragesprachen**
    - Bei relationalen Datenbanken meist **SQL** (s.u.)
  - **Datensicherheit** / **Datenschutz** sichern
    - Daten dürfen nicht verloren gehen oder (ungewollt / unberechtigt) verändert werden (Datensicherheit)
    - Daten dürfen nicht „*in falsche Hände*“ geraten (Datenschutz)
      - was sind dann „*die richtigen Hände*“? → benötigt **Autorisationsmodell**
  - **Datenintegrität** sichern
    - Daten müssen „zueinander passen“ (Integritätsbedingungen erfüllen)
      - z.B. keine Einschreibung in einen Studiengang, der gar nicht existiert
  - **Mehrbenutzerbetrieb** ermöglichen
    - Es sollen mehrere Anfragen zeitgleich ausgeführt werden können, ohne dass es zu unerwarteten Wechselwirkungen kommt

- **Relationale Datenbanken**
  - basieren auf **Tabellen**
    - Spalten = **Attribute**
    - Zeile = **Datensätze** / Objekte



  - **Beispiel**: Tabelle „Einschreibung“

Matrikelnummer	Studiengang	Abschluss
123456	Informatik	Bachelor
121212	Mathematik	Master
133333	Mathematik	Bachelor

    - Jede Zeile beschreibt einen **Datensatz**
    - Jeder Datensatz hat die angegebenen **Attributwerte**
    - Die Namen der Spalten und Typen der Attribute sind Teil der **Metadaten**.
      - Metadaten beschreiben Strukturen und Eigenschaften der Daten.
  - Die Tabellen repräsentieren **Relationen** zwischen den Attributen

Relationale Datenbanken

- **Schlüssel**
  - Ein **Schlüssel** einer Tabelle ist eine **Menge von Attributen**, mit der jeder Datensatz eindeutig identifiziert werden kann.
    - Häufig wird auch gefordert, dass diese Attributmenge minimal sein muss
  - **Beispiel**: Tabelle „Studierende“

Matrikelnummer	Vorname	Nachname	Emailadresse
123456	Peter	Müller	pm@gmail.de
121212	Karin	Müller	km@web.de
133333	Peter	Schmitt	ps@gmx.net

    - { Matrikelnummer } und { Emailadresse } sind jeweils (minimale) Schlüssel
    - { Matrikelnummer, Nachname } ist ein Schlüssel (aber nicht minimal)
    - Ungeeignet als Schlüssel ist z.B. { Nachname } *(warum?)*
    - **Frage**: Kann { Vorname, Nachname } hier ein Schlüssel sein?
      - Warum ja? Warum nein? Wenn ja: Ist das eine gute Idee?

- **Primärschlüssel**
  - Einer der Schlüssel einer Tabelle muss als **Primärschlüssel** (*Primary Key*, ***PK***) gewählt werden.
    - Um einen Datensatz zu identifizieren, geben wir meist den PK an.
      - Man kann aber auch jeden anderen Schlüssel der Tabelle nutzen
    - Der PK sollte **gut zu handhaben** sein (kurz, möglichst nur ein Attribut)
    - Der PK sollte sich nur möglichst **selten ändern**
  - **Beispiel**: Tabelle „Studierende“

Matrikelnummer	Vorname	Nachname	Emailadresse
123456	Peter	Müller	pm@gmail.de
121212	Karin	Müller	km@web.de
133333	Peter	Schmitt	ps@gmx.net

    - { Matrikelnummer } ist hier ein günstiger Primärschlüssel
    - { Emailadresse } ist zwar Schlüssel, ändert sich aber möglicherweise
      - Auch: Eindeutigkeit der Schreibweise, z.B. Ist Groß-Kleinschreibung eindeutig bei Emailaddr,?

Relationale Datenbanken

- **Natürliche Schlüssel** (sprechende Schlüssel)
  - Ergeben sich aus den Attributen des modellierten Objekts
    - z.B. um ein Buch zu identifizieren genügt vielleicht { Autor, Buchtitel, Jahr }
  - Beispiel: Personenliste

Vorname	Nachname	Geburtstag	Geburtsort	...
Peter	Müller	28.02.2000	Berlin	
Karin	Schmitt	31.12.1999	München	
Peter	Müller	28.02.2000	Kaiserslautern	

    - Genügt { Vorname, Nachname } als Schlüssel?
    - Genügt { Vorname, Nachname, Geburtstag, Geburtsort } als Schlüssel?
    - Selbst wenn: Ist dieser Schlüssel **handlich**?

Relationale Datenbanken

- **Künstliche Schlüssel**
  - Zusätzliches Attribut (z.B. „id“) mit **künstlicher** „Durchnummerierung“ um eindeutige Schlüssel zu erhalten
    - z.B. aufsteigende ganze Zahlen (id = 1, 2, 3, …)
      - Kann aber jede (kollisionsfreie) Sequenz von beliebigen Werten sein
      - z.B.: **UUID** (siehe Wikipedia), z.B. „`69cdd0f2-10f3-4104-aa39-0bd449cf68a0`“
    - **unvermeidlich**, wenn es keine natürlichen Schlüssel gibt
    - manchmal **sinnvoll**, wenn vorhandene natürliche Schlüssel zu unhandlich
  - Beispiel:

id	Vorname	Nachname	Geburtstag	Geburtsort	...
1	Peter	Müller	28.02.2000	Berlin	
2	Karin	Schmitt	31.12.1999	München	
42	Peter	Müller	28.02.2000	Kaiserslautern	

- **Frage:** Ist die { Matrikelnummer } in der früher betrachteten Tabelle „Studierende“ prinzipiell ein *natürlicher* oder ein *künstlicher* Schlüssel? (Oder beides?)

Relationale Datenbanken

- **Beziehungen zwischen Tabellen**
  - Eine Tabelle kann auf eine andere Tabelle **verweisen**
    - Der Verweis besteht aus den Attributenwerten ihres (Primär-) Schlüssels.
  - **Beispiel**: Tabellen „Studierende“ und „Fachstudium“

Matrikel-nummer	**Vor-name**	**Nach-name**
123456	Peter	Müller
121212	Karin	Müller
133333	Peter	Schmitt

Matrikel-nummer	**Studien-gang**	**Ab-schluss**
123456	Informatik	Bachelor
123456	Mathematik	Bachelor
133333	Mathematik	Master

    - (Primär-) Schlüssel in der Tabelle „Studierende“ ist { Matrikelnummer }
    - Die Tabelle „Fachstudium“ verweist mit dem Attribut { Matrikelnummer } auf die Tabelle „Studierende“
    - Diese Datensätze mit Matrikelnummer = „123456“ stehen also in Beziehung:
      - „Peter Müller“ studiert „Informatik“ mit dem Abschluss „Bachelor“
      - „Peter Müller“ studiert „Mathematik“ mit dem Abschluss „Bachelor“
    - *Achtung: Die verweisenden Attribute müssen nicht die selben Namen haben.*

- **Fremdschlüssel**
  - Verweist eine Tabelle auf den (Primär-) Schlüssel einer anderen, so nennt man diese Attribute **Fremdschlüssel** (Foreign Key)
    - Beispiel: Leiht eine Studentin ein Buch aus, so könnte in der Tabelle „Ausleihen“ durch den Fremdschlüssel { Matrikelnummer } auf den entsprechenden Studierenden-Datensatz verwiesen werden.
  - Eine Tabelle kann auch Fremdschlüssel auf sich selbst enthalten
    - **Beispiel**: In der Tabelle „Personal“ mit dem Primärschlüssel { Personalnummer } gibt es das Attribut { Vorgesetzter }, das auf die Personalnummer der selben Tabelle verweist.

Personalnummer	**Vorname**	**Nachname**	**Vorgesetzter**
101	Peter	Obermotz	
121	Karin	Mittelmeyer	101
133	Peter	Kleinschmidt	101

      - Herr Obermotz ist also Vorgesetzter von Frau Mittelmeyer und Herrn Kleinschmidt

Relationale Datenbanken

- **Integritätsbedingungen (1)**
  - **Integritätsbedingungen** definieren zulässige Zustände der DB
  - **Fremdschlüssel** unterliegen einer strikten Integritätsbedingung:
    - Jeder referenzierte Datensatz muss existieren („**Referentielle Integrität**“)

      ⇒ Wird ein referenzierter Datensatz z.B. **entfernt**, so dürfen die darauf verweisenden (referenzierenden) Fremdschlüssel nicht bestehen bleiben.
    - Es gibt verschiedene Optionen zur **Lösung** des Problems:
      - Fremdschlüssel auf **NULL** setzen (falls das erlaubt ist)
      - Fremdschlüssel auf einen anderen (existierenden) Datensatz umsetzen
      - Den referenzierenden Datensatz (der den Fremdschlüssel enthält) ebenfalls löschen
    - Wird keine Lösung (s.u.) gefunden, so wird die auslösende Löschung des referenzierten Objekts **rückgängig** gemacht (Transaktions-Abbruch, s.u.).
  - Es gibt noch diverse andere Integritätsbedingungen, z.B.
    - Datentypen und Bereichsbeschränkungen bei Attributen
    - Die Eindeutigkeit von Attributen oder Attributmengen (u.a. Primärschlüssel)
  - **Das DBMS garantiert die Einhaltung aller Integritätsbedingungen**

Relationale Datenbanken

- **Integritätsbedingungen (2)**
  - Beispiel: „Studierende“ und „Fachstudium“ (s.o.)

Matrikel-nummer	**Vor-name**	**Nach-name**
~~123456~~	~~Peter~~	~~Müller~~
121212	Karin	Müller
133333	Peter	Schmitt

Matrikel-nummer	**Studien-gang**	**Ab-schluss**
123456	Informatik	Bachelor
123456	Mathematik	Bachelor
133333	Mathematik	Master

Löscht man den Studenten „123456“ (Peter Müller) aus der Tabelle „Studierende“, so können die beiden Datensätze in „Fachstudium“, die sich auf ihn beziehen, nicht weiter darauf verweisen.

1. Lösung: Die beiden betroffenen „Fachstudium“-Datensätze löschen.

2. Lösung: Die Fremdschlüssel auf **NULL** (also undefiniert) setzten.

3. Lösung: Den Fremdschlüssel auf einen anderen (erlaubten) Wert setzen

4. Lösung: Das Löschen wird verweigert.

  - Frage: Was ist hier sinnvoll?

- **Integritätsbedingungen (3)**
  - Beispiel: „Personal“ (s.o.)

Personalnummer	**Vorname**	**Nachname**	**Vorgesetzter**
~~101~~	~~Peter~~	~~Obermotz~~	
121	Karin	Mittelmeyer	101
133	Peter	Kleinschmidt	101

    - Wird der Mitarbeiter „101“ (Peter Obermotz) aus „Personal“ gelöscht, so können die beiden anderen Datensätze, die sich mit dem Fremdschlüssel „Vorgesetzter“ auf ihn beziehen, nicht weiter darauf verweisen.
    - **1. Lösung**: Die beiden betroffenen „Personal“-Datensätze löschen.
    - **2. Lösung**: Die Fremdschlüssel auf **NULL** (also undefiniert) setzen.
    - **3. Lösung**: Den Fremdschlüssel auf einen anderen (erlaubten) Wert setzen
    - **4. Lösung**: Das Löschen wird verweigert.
  - Frage: Was ist hier sinnvoll? Welche Folgen hat es jeweils?

- **Modellierung von Tabellenstrukturen (1)**
  - Oft sind für gegebene Probleme mehrere Modellierungen möglich
  - Beispiel: „Studierende“ und „Fachstudium“ (s.o.)

Matrikel-nummer	**Vor-name**	**Nach-name**
123456	Peter	Müller
121212	Karin	Müller
133333	Peter	Schmitt

Matrikel-nummer	**Studien-gang**	**Ab-schluss**
123456	Informatik	Bachelor
123456	Mathematik	Bachelor
133333	Mathematik	Master

  - Statt der beiden Tabellen wäre auch eine große vorstellbar:

Matrikelnummer	**Vorname**	**Nachname**	**Studiengang**	**Abschluss**
123456	Peter	Müller	Informatik	Bachelor
123456	Peter	Müller	Mathematik	Bachelor
121212	Karin	Müller		
133333	Peter	Schmitt	Mathematik	Master

  - **Nachteile**: Gefahr von Wiederholungen (Redundanz, Änderungen aufwändig), riesige Tabellen, viele leere Felder, ...

Modellierung von Tabellenstrukturen (2)

- Beispiel: „Studierende“ (s.o.)

Matrikel-nummer	**Vor-name**	**Nach-name**
123456	Peter	Müller
121212	Karin	Müller
133333	Peter	Schmitt

- Man könnte die Tabellen aber auch weiter aufteilen:

Matrikel-nummer	**Vor-name**
123456	Peter
121212	Karin
133333	Peter

Matrikel-nummer	**Nach-name**
123456	Müller
121212	Müller
133333	Schmitt

- Nachteile: Sehr viele Tabellen, Einsammeln von Attributen ggf. sehr mühsam und wenig performant (viele Verknüpfungen und Leseoperationen nötig um alle Attribute zu erhalten)

- **Modellierung von Tabellenstrukturen (3)**
  - Es gilt hier zu einen Kompromiss zwischen gegenläufigen Anforderungen zu finden.
  - Es gibt im Bereich der Relationalen Datenbanken dazu eine eigene Theorie: Die **Normalformenlehre**
    - Das Erreichen der 5 (oder 6) Normalformen ist ein inkrementeller Prozess. Die Tabellenstruktur wird dabei immer abstrakter und redundanzärmer.
      - **1. Normalform**: Attribute aufteilen bis sie atomar sind
      - **2. bis 5. Normalform**: Tabellen immer weiter aufspalten um Wiederholungen zu vermeiden und Abhängigkeiten zwischen Attributen zu verringern
    - Die unteren Normalform-Stufen sind fast immer sinnvoll anzuwenden
    - Die höheren Normalform-Stufen zersplittern die Tabellenstruktur oft über das sinnvolle Maß hinaus.

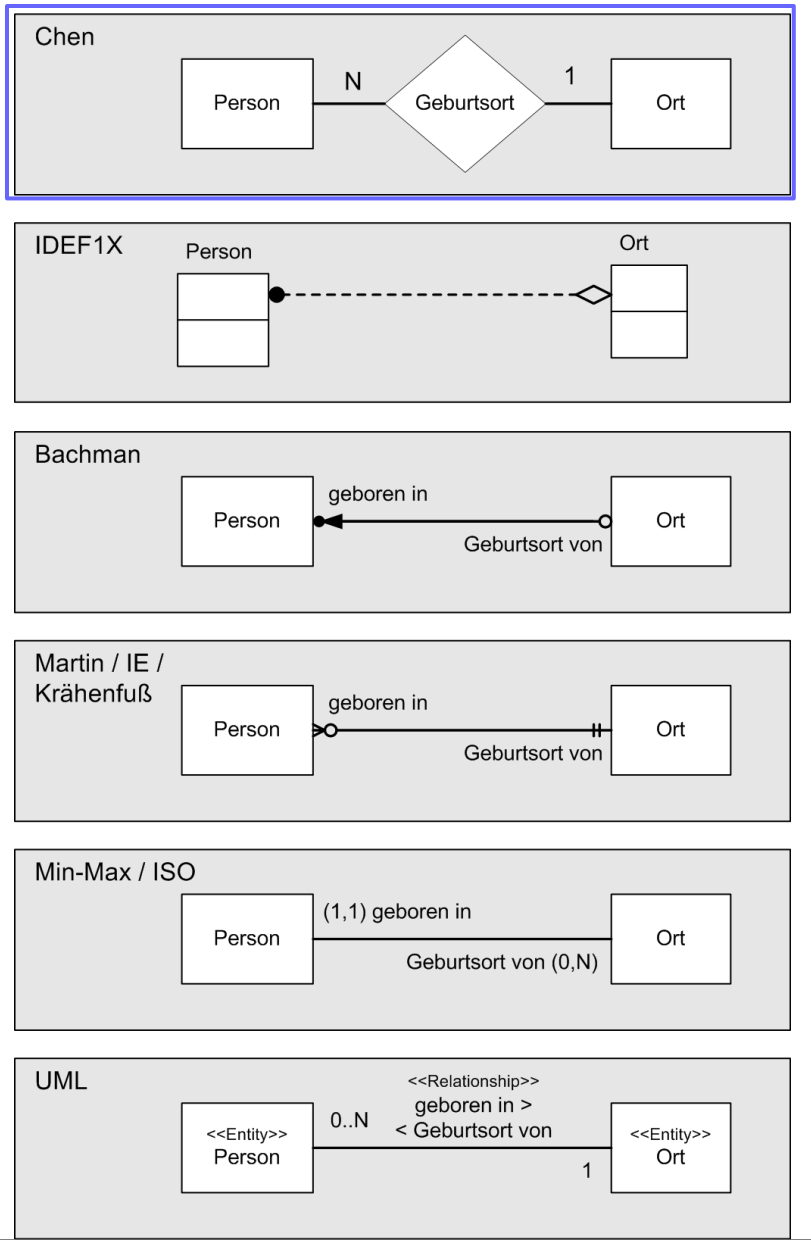
    - siehe Wikipedia: Artikel "Normalisierung (Datenbank)"
    - *Empfehlung: Schauen sie sich zumindest einmal die Beispiele im Wikipedia-Artikel an, um einen Eindruck zu erhalten.*

Entity-Relationship-Modell

- **Das Entity-Relationship-Modell (ERM)**
  - Grafische Notation zur Datenmodellierung (Chen, 1976)
  - De-facto-Standard bei der Modellierung von relationalen Modellen
  - **Gegenstände**:
    - **Entity** (Entität) = (konkreter) Gegenstand der realen Welt
      - z.B. Student „Peter Müller“, Hörsaal „42-105“, ...
    - **Relationship** (Beziehung) = Beziehung zwischen 2 oder mehr Entitäten
      - z.B. „Student ABC besucht Vorlesung XYZ in 42-105“,
    - **Eigenschaft** = Eigenschaft einer konkreten Entität (z.B. Vorname)
  - Darauf Aufbauende **Typen** (Klassen / Mengen)
    - **Entitätstyp** (z.B. Studierende, Hörsäle, Vorlesungen, …)
    - **Beziehungstyp** = Beziehung zwischen den (Elementen der) Entitätstypen
      - z.B. „besucht Vorlesung“
    - **Attribut** = Typisierung der Eigenschaft eines Entitätstyps (z.B. Vorname)

Entity-Relationship-Modell

- **ER-Diagramm (ERD)**
  - Verschiedene Grafische Darstellungen möglich
  - Ziele:
    - Entitätstypen und Beziehungstypen übersichtlich darzustellen
    - Rollennamen für Beziehungen zu vergeben
    - Kardinalitäten zu vergeben
  - Wir benutzen in der Vorlesung eine spezielle *Chen*-Notation
    - Mit modifizierten Kardinalitäten
    - Alternativen: Siehe Wikipedia-Artikel zum ER-Modell



Entity-Relationship-Modell

- **Beispiel für ER-Diagramm:** Personen und Geburtsorte

  - Das Diagramm beschreibt die Beziehung „Geburtsort“ zwischen den **Entitäts-Typen** (Entitäten-Mengen) „Person“ und „Ort“
    - Idee: *„Jede Person wird **verknüpft** mit genau einem Ort (aus der Menge aller Orte). Diese Verknüpfung beschreibt den Geburtsort der Person.“*
  - Die Zahlenangaben sind **Kardinalitäten**
    - „1“ = „Es gibt zu jeder Person genau 1 Ort, an dem sie geboren ist.“
    - „0..*“ oder „0..N“ = „Zu jedem Ort gibt es eine beliebige Anzahl von Personen, die dort geboren sind (einschließlich 0).“
    - **Beachten Sie die Anordnung:** „1“ steht bei „Ort“, „0..*“ bei „Person“.
      - Idee: Aus Sicht der Person gibt es „1“ Orte,
        aus Sicht des Ortes „0..*“ (*beliebig viele*) Personen

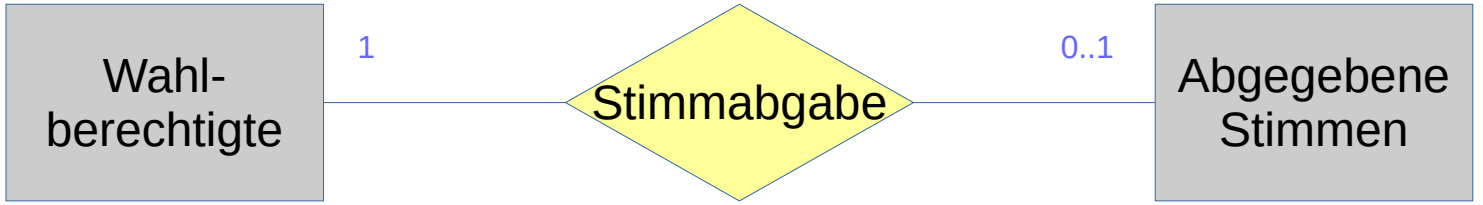
Entity-Relationship-Modell

- **Weitere Kardinalitäten**

  - Es sind auch andere Kardinalitäten möglich
    - „1..*“ oder „1..N“ = „Zu jedem Ort gibt es eine beliebige Anzahl von Personen, die dort geboren sind, mindestens jedoch einer.“
      - z.B. sinnvoll, wenn ich nur Orte in der Menge haben will, die auch als Geburtsort verwendet werden.
    - „0..1“ = „Es gibt zu jeder Person höchstens 1 Ort, an dem sie geboren ist.“
      - z.B. hier sinnvoll, wenn ich nicht zu jeder Person den Geburtsort kenne
  - Man bezeichnet die Kardinalitäten außerhalb der Diagramme **vereinfacht** (**abstrahiert**) als „**1:1**“, „**1:n**“ oder „**n:m**“
    - Die „1“ bedeutet in „1:1“ bzw. „1:n“ ein „*höchstens 1*“, schließt also 0 mit ein
    - „n“ und „m“ sind (wie oben „N“ und „*“) keine konkreten Werte (= „*beliebig*“)

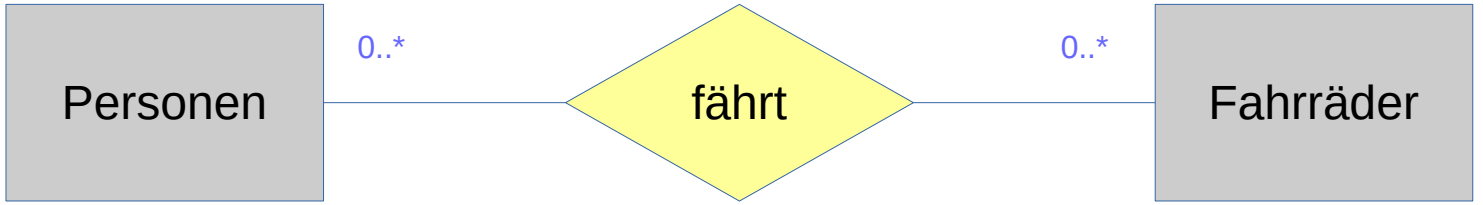
Entity-Relationship-Modell

- **Abstrahierte Kardinalitäten** (charakterisierend)
  - **1:1** („One man, one vote“ – man muss aber nicht wählen)



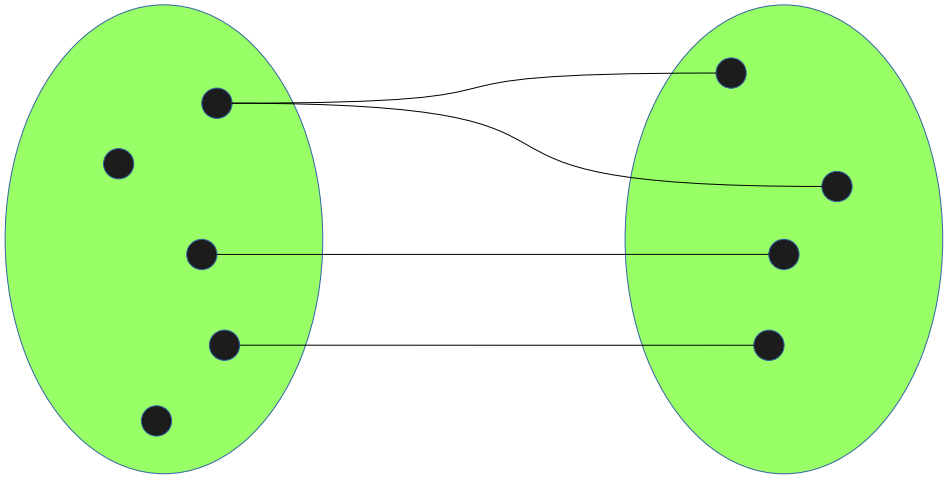
  - **1:n** (Jedes KFZ hat einen Halter, aber nicht jeder hat ein KFZ)

  - **n:m** (Beliebig viele Fahrer pro Fahrrad und umgekehrt)



Entity-Relationship-Modell

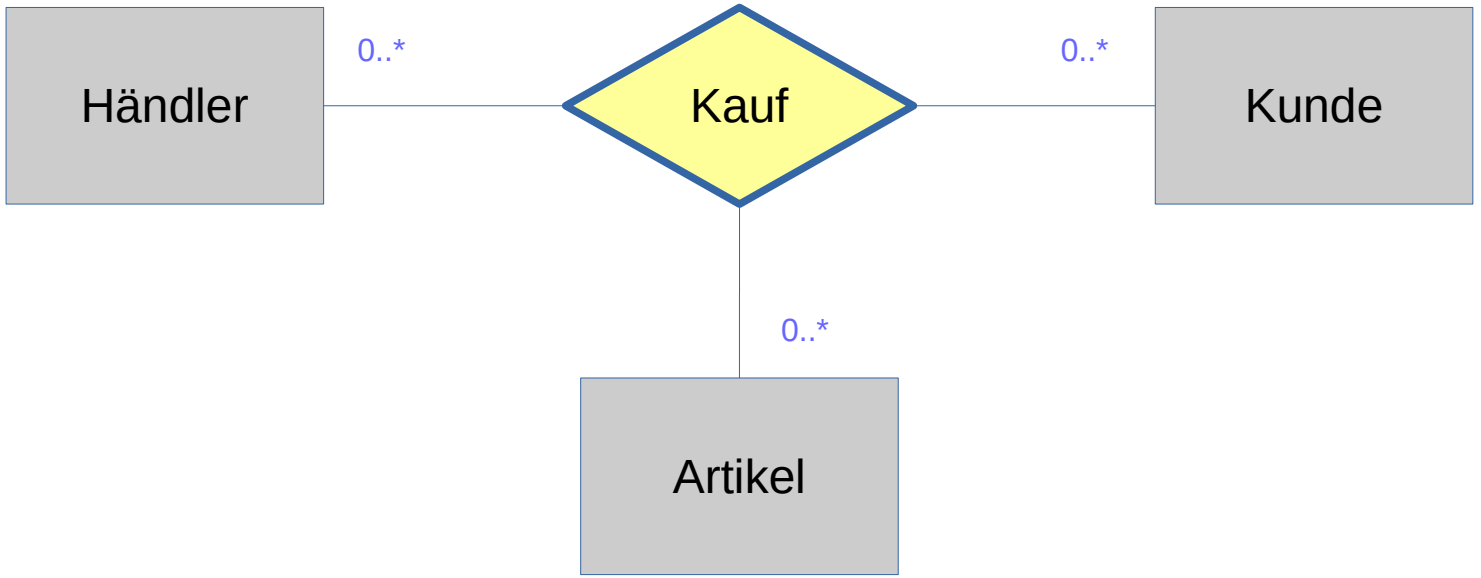
- **Es gibt auch Diagramme für Entities**
  - Diese sind nützlich um Kardinalitäten klar zu machen
- **Beispiel:**
  - **1:n**



- Zwei Entity-Mengen (Person, KFZ),
- jeweils mehrere Entities (Punkte) und ihre Beziehungen (Linien)
- Hier: *Jedes KFZ hat einen Halter, aber nicht jeder hat ein KFZ*

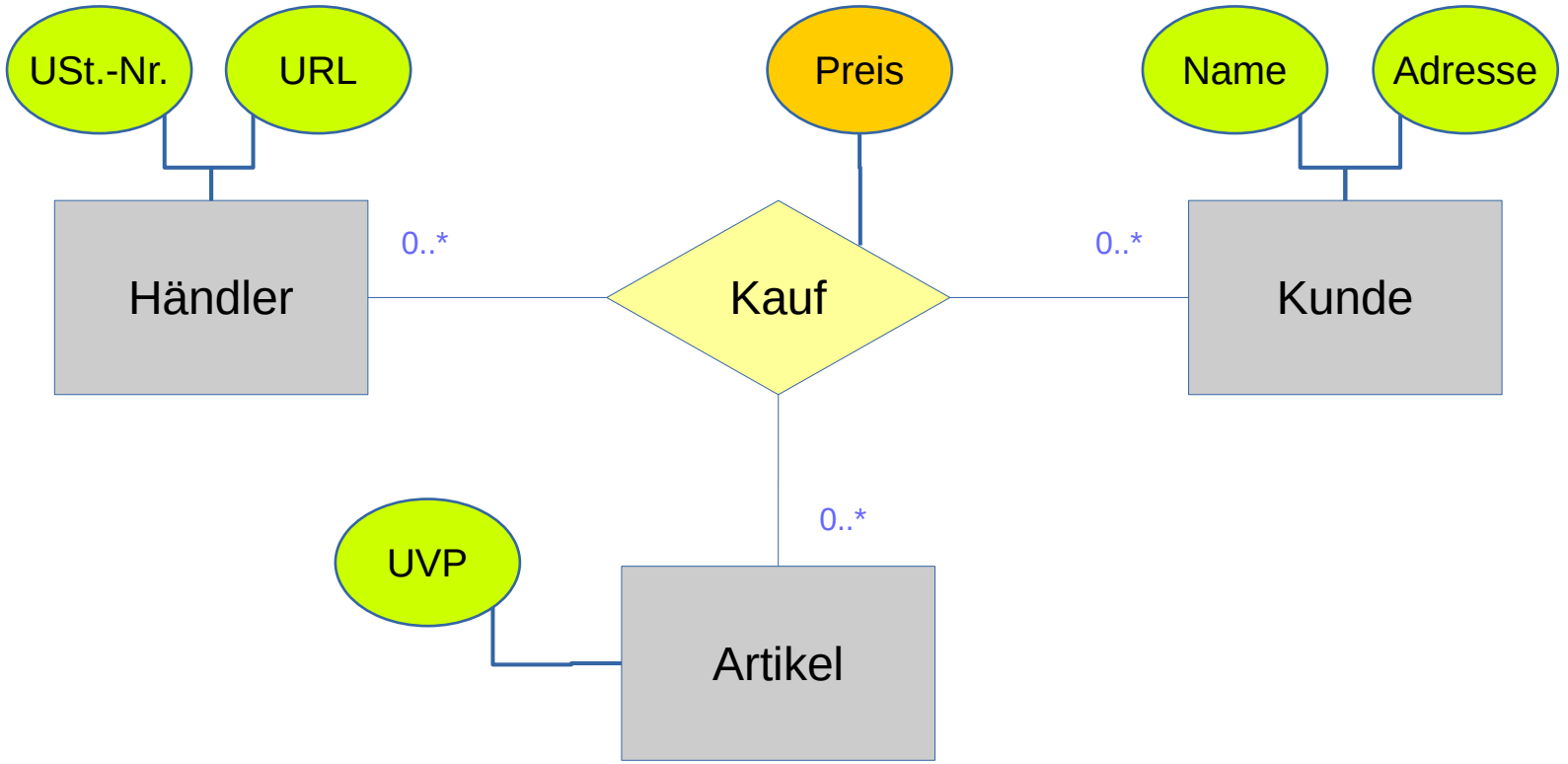
Entity-Relationship-Modell

- **Es gibt neben binären auch n-äre Relationen**
  - Beispiel: n=3



Entity-Relationship-Modell

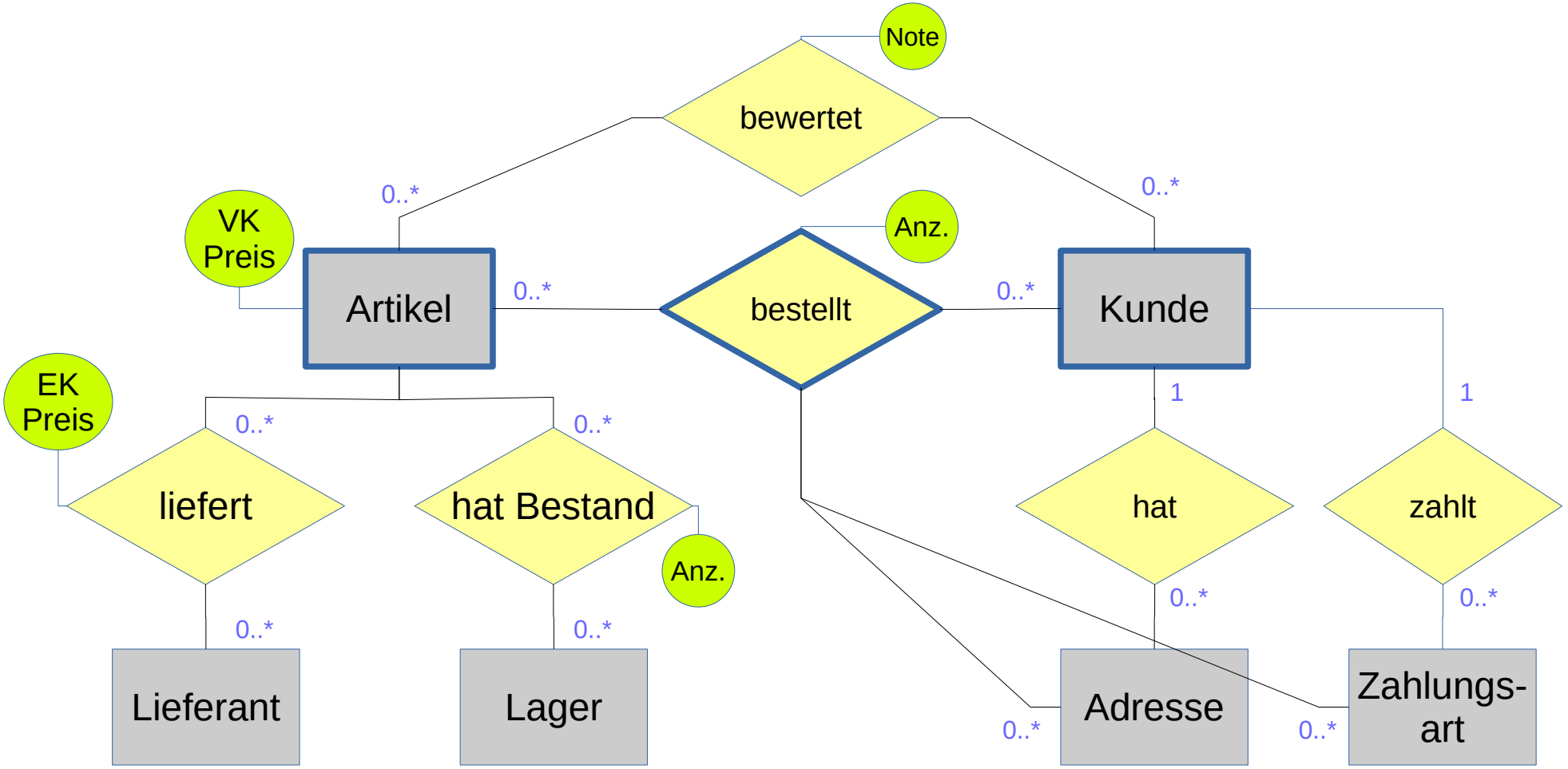
- **Man kann auch Attribute zuordnen**
  - zu Entitätstypen oder auch zu Relationen



  - Das wird aber schnell unübersichtlich

Entity-Relationship-Modell

- **Das ER-Modell ist nützlich zur Datenmodellierung**



  - Es liefert uns Übersicht bei komplexen Datenstrukturen