

# Kapitel DB:II

## II. Konzeptueller Datenbankentwurf

- ❑ Entwurfsprozess
- ❑ Datenbankmodelle
- ❑ Einführung in das Entity-Relationship-Modell
- ❑ ER-Konzepte und ihre Semantik
- ❑ Charakterisierung von Beziehungstypen
- ❑ Existenzabhängige Entity-Typen
- ❑ Abstraktionskonzepte
- ❑ Konsolidierung, Sichtenintegration
- ❑ Konzeptuelle Modellierung mit UML

# Einführung in das Entity-Relationship-Modell

## Historie:

- ❑ Entity-Relationship-Modell kurz: ER-Modell bzw. ERM
- ❑ 1976 von Peter Chen vorgeschlagen
- ❑ Standardmodell für frühe Entwurfsphasen in der Datenbankentwicklung
- ❑ mittlerweile existieren viele Varianten und Erweiterungen
- ❑ eingesetzt auch in anderen Bereichen der Informatik

## Eigenschaften:

- ❑ unabhängig von einem bestimmten Datenbanksystem
- ❑ Grundkonzepte „Entity“ und „Relationship“ werden als natürlich und – trotz ihrer Einfachheit – als ausreichend für viele Situationen empfunden
- ❑ unterstützt die drei wichtigsten Abstraktionsmechanismen:  
*Klassifikation, Aggregation, Verallgemeinerung*
- ❑ starre Informationsstruktur, d.h., Ausrichtung auf große Datenmengen
- ❑ Definition von statischen Eigenschaften und Integritätsbedingungen

# Einführung in das Entity-Relationship-Modell

## Elemente

### 1. Entity(-Instanz)

Ein Objekt oder Ding (*Entity*) der realen oder einer virtuellen Welt, für das Informationen zu speichern sind. Jedes Entity ist von einem bestimmten Entity-Typ.

### 2. Beziehung(sinstanz)

Ein n-Tupel mit Entities. Jede Beziehung (*Relationship*) ist von einem bestimmten Beziehungstyp. Beziehungen zwischen Entities derselben Typen gehören zu demselben Beziehungstyp.

### 3. Attribut

Definiert eine Eigenschaft von Entity-Typen oder Beziehungstypen.

### 4. Rolle

Dokumentiert die Rolle eines Entities in einer Beziehung.

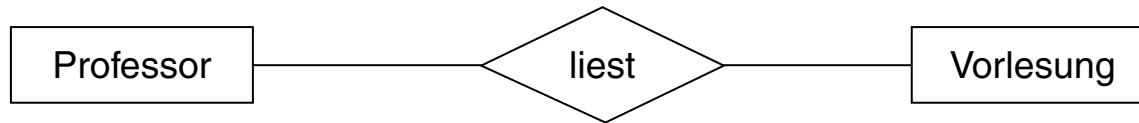
# Einführung in das Entity-Relationship-Modell

Die Beschreibung der Informationsstruktur eines Anwendungsbereichs im Entity-Relationship-Modell heißt Entity-Relationship-Diagramm (ER-Diagramm) oder Entity-Relationship-Schema (ER-Schema).

*" Ein Professor liest eine Vorlesung "*

# Einführung in das Entity-Relationship-Modell

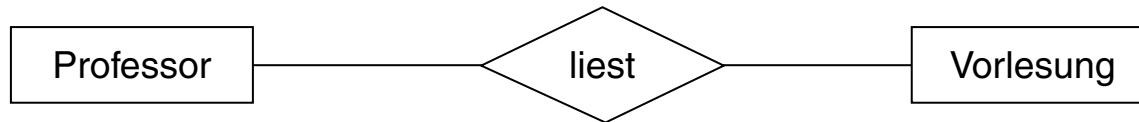
Die Beschreibung der Informationsstruktur eines Anwendungsbereichs im Entity-Relationship-Modell heißt Entity-Relationship-Diagramm (ER-Diagramm) oder Entity-Relationship-Schema (ER-Schema).



*" Ein Professor liest eine Vorlesung "*

# Einführung in das Entity-Relationship-Modell

Die Beschreibung der Informationsstruktur eines Anwendungsbereichs im Entity-Relationship-Modell heißt Entity-Relationship-Diagramm (ER-Diagramm) oder Entity-Relationship-Schema (ER-Schema).

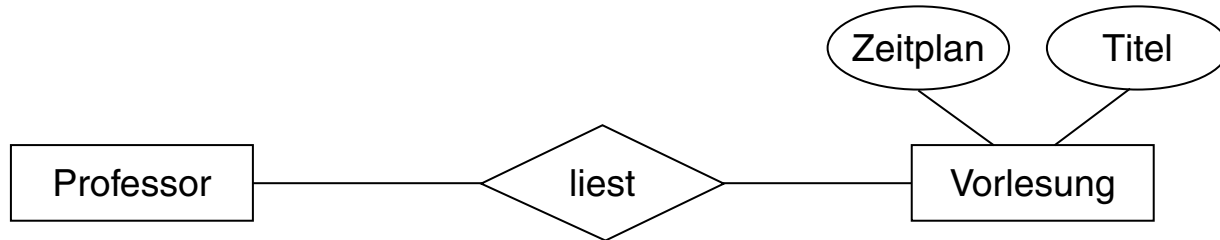


*" Ein Professor liest eine Vorlesung "*

*" Eine Vorlesung hat einen Zeitplan und einen Titel "*

# Einführung in das Entity-Relationship-Modell

Die Beschreibung der Informationsstruktur eines Anwendungsbereichs im Entity-Relationship-Modell heißt Entity-Relationship-Diagramm (ER-Diagramm) oder Entity-Relationship-Schema (ER-Schema).

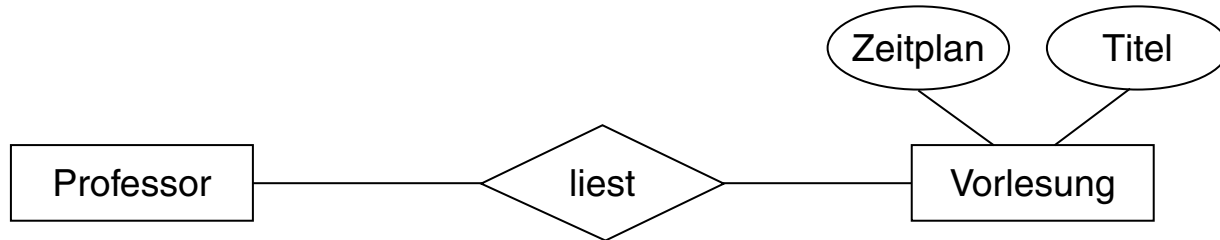


*" Ein Professor liest eine Vorlesung "*

*" Eine Vorlesung hat einen Zeitplan und einen Titel "*

# Einführung in das Entity-Relationship-Modell

Die Beschreibung der Informationsstruktur eines Anwendungsbereichs im Entity-Relationship-Modell heißt Entity-Relationship-Diagramm (ER-Diagramm) oder Entity-Relationship-Schema (ER-Schema).



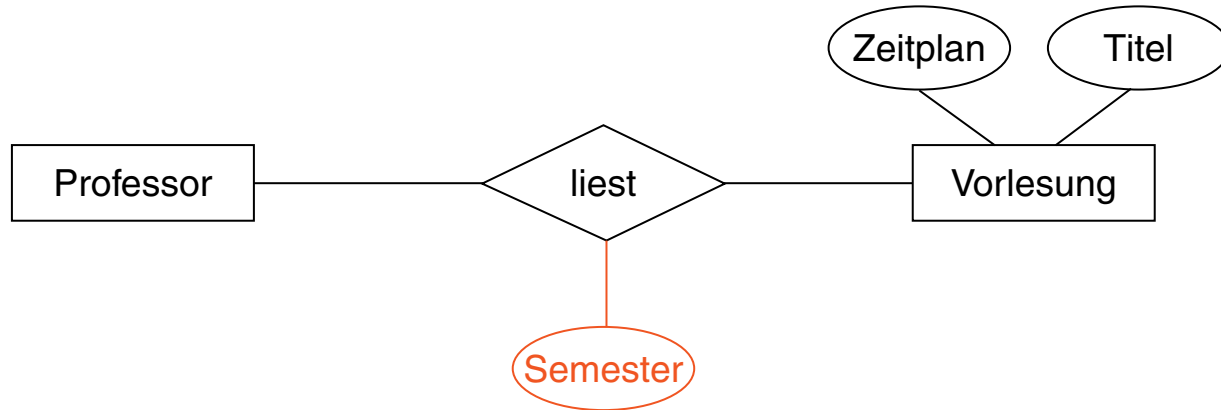
*" Ein Professor liest eine Vorlesung in einem Semester "*

*" Eine Vorlesung hat einen Zeitplan und einen Titel "*



# Einführung in das Entity-Relationship-Modell

Die Beschreibung der Informationsstruktur eines Anwendungsbereichs im Entity-Relationship-Modell heißt Entity-Relationship-Diagramm (ER-Diagramm) oder Entity-Relationship-Schema (ER-Schema).

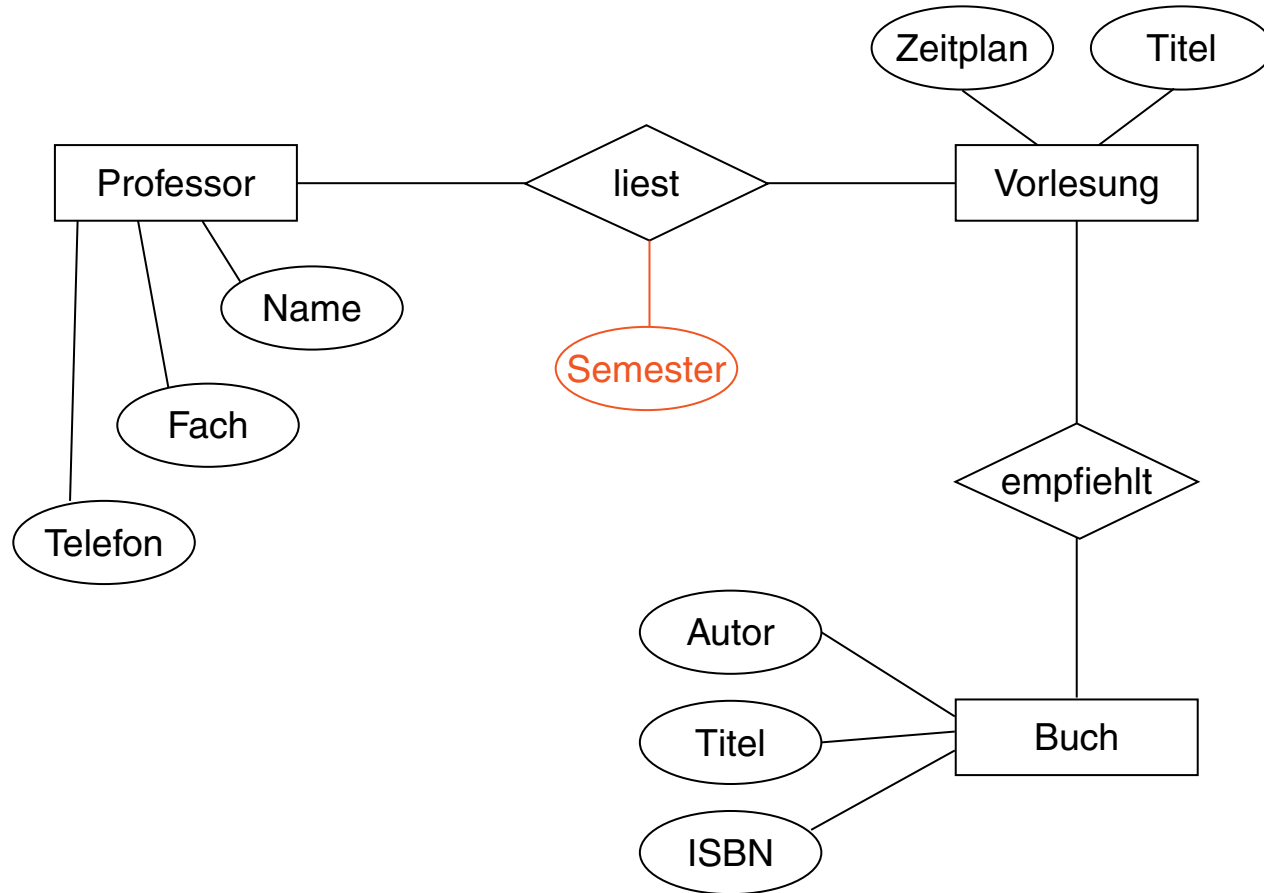


*" Ein Professor liest eine Vorlesung in einem **Semester** "*

*" Eine Vorlesung hat einen Zeitplan und einen Titel "*

# Einführung in das Entity-Relationship-Modell

Die Beschreibung der Informationsstruktur eines Anwendungsbereichs im Entity-Relationship-Modell heißt Entity-Relationship-Diagramm (ER-Diagramm) oder Entity-Relationship-Schema (ER-Schema).



# ER-Konzepte und ihre Semantik

## Datentypen

Datentypen, hier insbesondere Entity-, Beziehungs- und Attributtypen, dienen zur Definition von Wertebereichen. Ein Datentyp  $X$  ist gekennzeichnet durch:

1.  $dom(X)$  Domain, Definitionsbereich.  
Die möglichen Werte, die ein Objekt vom Typ  $X$  annehmen kann.
2. Operationen, Funktionen, Prädikate, die für die Bearbeitung von Werten und für Aussagen über Werte zur Verfügung stehen.
3.  $state(X)$  Zustand.  
Alle zu einem bestimmten Zeitpunkt bekannten Werte vom Typ  $X$ .

# ER-Konzepte und ihre Semantik

## Datentypen

Datentypen, hier insbesondere Entity-, Beziehungs- und Attributtypen, dienen zur Definition von Wertebereichen. Ein Datentyp  $X$  ist gekennzeichnet durch:

1.  $dom(X)$  Domain, Definitionsbereich.  
Die möglichen Werte, die ein Objekt vom Typ  $X$  annehmen kann.
2. Operationen, Funktionen, Prädikate, die für die Bearbeitung von Werten und für Aussagen über Werte zur Verfügung stehen.
3.  $state(X)$  Zustand.  
Alle zu einem bestimmten Zeitpunkt bekannten Werte vom Typ  $X$ .

Beispiele:

- ❑ abstrakter Datentyp `Integer`.  
 $dom(Integer) = \mathbb{Z}$  mit Operationen  $+$   $-$   $*$   $/$   $<$   $>$   $=$
- ❑ abstrakter Datentyp `String`.  
 $dom(String) = \Sigma^*$  für ein Alphabet  $\Sigma$  mit Operationen  $+$  `length`

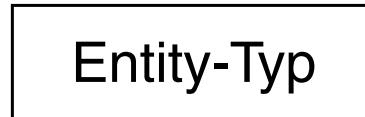
## Bemerkungen:

- ❑ Die Operationen auf einem Datentyp sind hier nur informell genannt. Implementierungen von Datenbanken verwenden hauptsächlich aus Programmiersprachen bekannte, primitive Datentypen sowie Nicht-Standard-Datentypen wie `Date` und `Time`. Operationen über diesen Datentypen werden teilweise in der Anfragesprache, die ein DBMS unterstützt, zur Verfügung gestellt.
- ❑ Der Zustand eines Datentyps ist nicht Teil der Definition eines Datentyps. Er dient uns hier als Konzept, um den Zustand einer Datenbank ermitteln zu können.

# ER-Konzepte und ihre Semantik

## Entity-Typen

Ein Entity-Typ deklariert eine Menge von Objekten mit bestimmten Eigenschaften, die (in der Datenbank) repräsentiert werden sollen. Graphische Darstellung:



Bezeichner für Entity-Typen sind  $E, E_1, E_2, \dots$  oder wie im Beispiel *Professor*, *Vorlesung* und *Buch*.

$dom(E)$ :

Menge der möglichen Entities (Objekte) vom Typ  $E$ . Diese Menge wird meist nicht explizit festgelegt, sondern als genügend groß, z.B. isomorph zu  $\mathbb{N}$  angenommen. Man kann  $dom(E)$  als die Menge der Namen der Entities auffassen.

# ER-Konzepte und ihre Semantik

## Entity-Typen (Fortsetzung)

$state(E)$ :

Menge der aktuellen Entities vom Typ  $E$  im Zustand  $state$  der Datenbank. Durch Operationen auf der Datenbank kann sich  $state(E)$  verändern. Insbesondere gilt:

- $state(E) \subseteq dom(E)$
- $state(E)$  ist endlich

Die Anzahl der zum Typ  $E$  gespeicherten Entities kann beliebig sein, ist aber in jedem Zustand  $state$  endlich.

Beispiel:

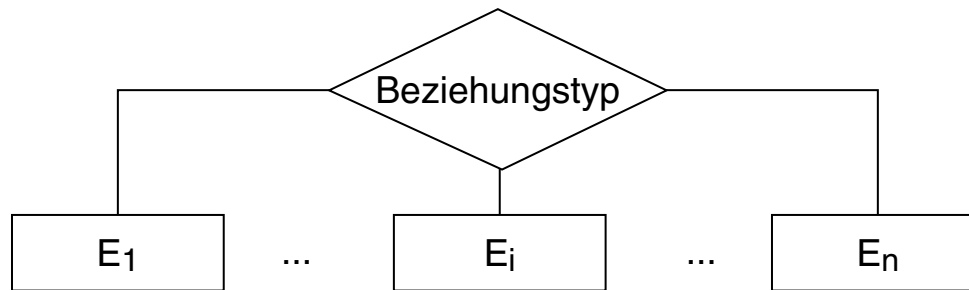
- Entity-Typ *Professor*
- $dom(Professor) = \Sigma^*$  mit  $\Sigma = \{A, \dots, Z, a, \dots, z\}$
- $state(Professor) = \{\text{Froehlich, Lucks, Wuethrich}\}$

# ER-Konzepte und ihre Semantik

## Beziehungstypen

Ein Beziehungstyp deklariert eine Beziehung zwischen Entity-Typen. Es kann eine beliebige Anzahl  $n \geq 2$  von Entity-Typen an einem Beziehungstyp teilhaben.

Graphische Darstellung:



Bezeichner für Beziehungstypen sind  $R, R_1, R_2, \dots$ , oder wie im [Beispiel](#) *empfiehl*t und *liest*.

$dom(R)$ :

Menge der möglichen Beziehungen (Entity-Tupel) vom Typ  $R$  zwischen Entities der Typen  $E_1, \dots, E_n$ .

$$dom(R) = dom(E_1) \times \dots \times dom(E_n)$$



# ER-Konzepte und ihre Semantik

## Beziehungstypen (Fortsetzung)

$state(R)$ :

Menge der aktuellen Beziehungen vom Typ  $R$  im Zustand  $state$  der Datenbank.  
Durch Operationen auf der Datenbank kann sich  $state(R)$  verändern.

$$state(R) \subseteq state(E_1) \times \dots \times state(E_n)$$

Schreibweise:

$$R(E_1, \dots, E_n)$$

## Beispiel:

- ❑ Beziehungstyp *liest*
- ❑  $state(Professor) = \{\text{Froehlich, Lucks, Wuethrich}\}$
- ❑  $state(Vorlesung) = \{\text{Algorithmen, Computergrafik, Kryptographie, HCI, DB}\}$
- ❑  $state(liest) = \{(\text{Wuethrich, Algorithmen}), (\text{Luck, Kryptographie})\}$

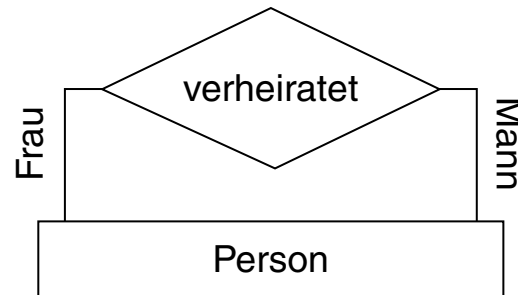
# ER-Konzepte und ihre Semantik

## Beziehungstypen (Fortsetzung)

Sind einzelne Entity-Typen mehrfach in einem Beziehungstyp eingebunden, legen Rollenbezeichner die Rollen der beteiligten Objekte in der Beziehung fest. Man nennt solche Beziehungstypen **reflexiv**.

Beispiel:

*verheiratet(Frau : Person, Mann : Person)*



## Bemerkungen:

- ❑ In der textuellen Notation sind Rollenbezeichner nicht erforderlich: die Rolle kann über die Position im Argumentetupel (Signatur) festgelegt werden: *verheiratet(Person, Person)*
- ❑ In der graphischen Darstellung sind Rollenbezeichner erforderlich.
- ❑ In der Datenbankliteratur werden reflexive Beziehungstypen mitunter auch *rekursive Beziehungstypen* genannt.

# ER-Konzepte und ihre Semantik

## Attribute und Attributtypen

Ein Attribut definiert eine Eigenschaft für alle Instanzen eines Entity- oder Beziehungstyps. Ein Attribut kann deshalb als *Funktion* verstanden werden, die jeder Instanz eine Eigenschaftsausprägung zuordnet. Graphische Darstellung:



Bezeichner für Attribute sind  $A, A_1, A_2, \dots$  oder wie im [Beispiel](#) *Name, ISBN, Titel* oder *Semester*.

Der Attributtyp  $T$  bezeichnet den Typ der Eigenschaft und ist üblicherweise ein Standard-Datentyp wie `Integer` oder `String`.

$dom(T)$ :

Menge der möglichen Werte des Attributtyps  $T$ .

## Bemerkungen:

- ❑ Nicht-Standard-Datentypen für Attribute sind
  - von DBMS zur Verfügung gestellte Datentypen wie `Date`, `Time`, `Timestamp`, `Blob`.
  - Potenzmengen von Attributtypen für *mehrwertige* Attribute.  
Beispiel: Attribut *Titel* vom Attributtyp  $\mathcal{P}(\text{dom}(T))$  für  $\text{dom}(T) = \{ \text{BSc, Dipl.-Inform., Dr., MSc, Prof., ...} \}$ .
  - (hierarchische) Kompositionen von Standard-Datentypen.  
Beispiel: Attribut *Adresse* vom gleichnamigen Attributtyp `Adresse`, das Adressbestandteile vom Typ `String` und `Integer` zusammenfasst.

# ER-Konzepte und ihre Semantik

## Attribute von Entity-Typen

$dom(A)$ :

Menge der möglichen Abbildungen von  $dom(E)$  nach  $dom(T)$ .

$state(A)$ :

Eine Abbildung  $state(E) \rightarrow dom(T)$  im Zustand  $state$  der Datenbank, die jedem aktuellen Entity aus  $state(E)$  einen Wert aus  $dom(T)$  zuordnet.

Schreibweise (Entity-Typ mit Attributen  $A_1, \dots, A_n$ ):

$$E(A_1 : T_1, \dots, A_n : T_n) \quad \text{bzw.} \quad E(A_1, \dots, A_n)$$

Beispiel:  $\rightsquigarrow \mathcal{TAFEL}$

## Bemerkungen:

- ❑ Modellierungstechnisch gibt es zwei Blickwinkel, unter denen Attribute betrachtet werden.  $dom(A)$  und  $dom(T)$  spiegeln diese Unterscheidung wider:

### 1. Aus Sicht des ER-Modells.

Hier wird ein Attribut als eine Abbildung über dem Wertebereich eines Entity-Typs bzw. Beziehungstyps aufgefasst.  $dom(A)$  ist die Menge der möglichen Abbildungen.

Diese Sichtweise ist durch die Operationen auf der Datenbank motiviert: Zu jedem Zeitpunkt  $t$  ist der Zustand der Datenbank durch die Attribute (= den entsprechenden Abbildungen) festgelegt. [[DB:II Datenbankmodelle » Datenbankzustand](#)]

Beispiel: Zum Zeitpunkt  $t$  ordnet Attribut *ISBN* jedem Buch eine ganze Zahl zu.

### 2. Aus Sicht des relationalen Modells.

Hier wird ein Attribut lediglich als Bezeichner des Datentyps einer Eigenschaft verstanden.  $dom(A)$  im relationalen Modell ist die Menge der möglichen Attributwerte und entspricht der Menge  $dom(T)$  im ER-Modell. [[DB:IV Das Relationale Modell](#)]

Beispiel: Das Attribut *ISBN* ist vom Typ „ganze Zahl“.

# ER-Konzepte und ihre Semantik

## Attribute von Beziehungstypen

$dom(A)$ :

Menge der möglichen Abbildungen von  $dom(R)$  nach  $dom(T)$ .

$state(A)$ :

Eine Abbildung  $state(R) \rightarrow dom(T)$  im Zustand  $state$  der Datenbank, die jeder aktuellen Beziehung aus  $state(R)$  einen Wert des Datentyps  $T$  aus  $dom(T)$  zuordnet.

Schreibweise:

$$R(E_1, \dots, E_m; A_1 : T_1, \dots, A_n : T_n) \quad \text{bzw.} \quad R(E_1, \dots, E_m; A_1, \dots, A_n)$$



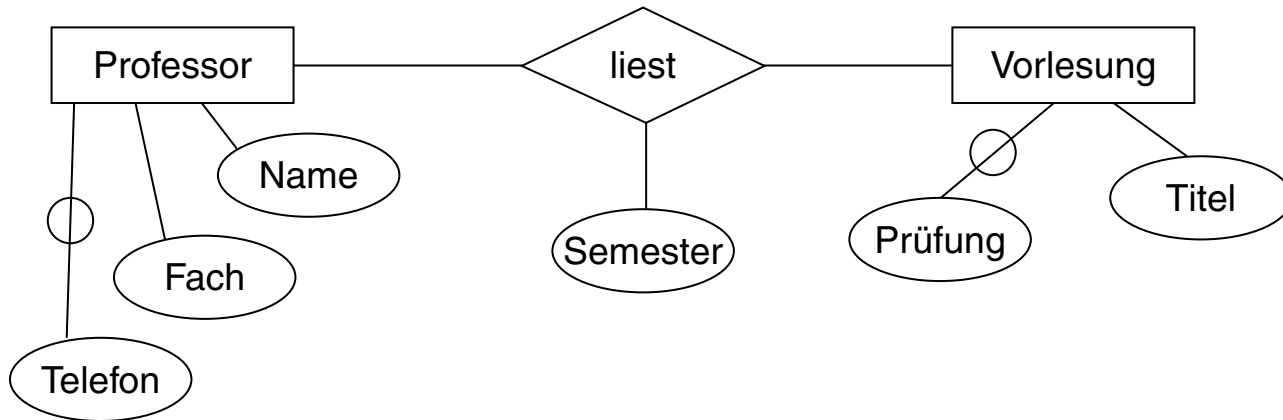
# ER-Konzepte und ihre Semantik

## Optionale Attribute

Optionale Attribute müssen nicht für alle Entities einen definierten Wert annehmen.  
Graphische Darstellung:



Beispiel:



## Bemerkungen:

- Eine Kennzeichnung von  $A$  als optionales Attribut zeigt also an, dass es sich bei der Funktion  $state(A)$  um eine *partielle* Funktion handelt.

# ER-Konzepte und ihre Semantik

## Schlüssel

### Definition 1 (Schlüssel im ER-Modell, Schlüsselattribut)

Sei  $\{A_1, \dots, A_n\}$  die Menge der Attribute eines Entity-Typs  $E$ . Eine nicht verkleinerbare Menge von Attributen  $\{A_{i_1}, \dots, A_{i_k}\} \subseteq \{A_1, \dots, A_n\}$  nennt man Schlüssel, gdw. ( $\leftrightarrow$ ) deren Werte das zugeordnete Entity eindeutig innerhalb aller Entities seines Typs identifiziert. Die  $\{A_{i_1}, \dots, A_{i_k}\}$  heißen Schlüsselattribute.

Beispiele:

- ❑ Ein/e Student/in wird eindeutig durch eine Matrikelnummer identifiziert.
- ❑ Ein Buch wird eindeutig durch eine ISBN identifiziert.

# ER-Konzepte und ihre Semantik

## Schlüssel

### Definition 1 (Schlüssel im ER-Modell, Schlüsselattribut)

Sei  $\{A_1, \dots, A_n\}$  die Menge der Attribute eines Entity-Typs  $E$ . Eine nicht verkleinerbare Menge von Attributen  $\{A_{i_1}, \dots, A_{i_k}\} \subseteq \{A_1, \dots, A_n\}$  nennt man Schlüssel, gdw. ( $\leftrightarrow$ ) deren Werte das zugeordnete Entity eindeutig innerhalb aller Entities seines Typs identifiziert. Die  $\{A_{i_1}, \dots, A_{i_k}\}$  heißen Schlüsselattribute.

Beispiele:

- ❑ Ein/e Student/in wird eindeutig durch eine Matrikelnummer identifiziert.
- ❑ Ein Buch wird eindeutig durch eine ISBN identifiziert.

Beispiel für eine Folgerung auf Basis von Definition 1 :

Sei  $\{A_2\}$  die Menge der Schlüsselattribute von Entity-Typ  $E(A_1, A_2, A_3)$ .

⇒

$\forall e_1, e_2 \in \text{state}(E) :$

(Un)gleiche Werte für  $A_2$  implizieren die (Un)gleichheit von  $e_1$  und  $e_2$ .

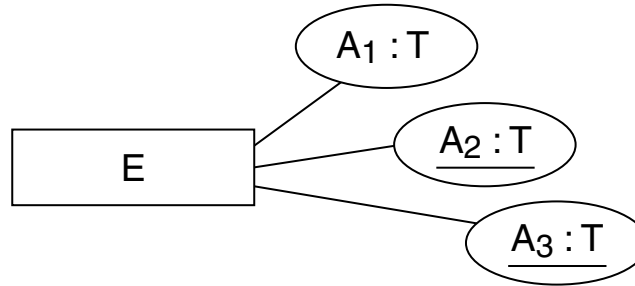
## Bemerkungen:

- ❑ Es kann mehrere *Schlüsselkandidaten* geben. Von ihnen ist einer auszuwählen; dieser wird als *Primärschlüssel* bezeichnet.
- ❑ Verschiedene Schlüsselkandidaten können eine unterschiedliche Anzahl von Attributen besitzen.

# ER-Konzepte und ihre Semantik

## Schlüssel (Fortsetzung)

Graphische Darstellung:



Schreibweise:

$$E(A_1, \underline{A_2}, \underline{A_3})$$

# ER-Konzepte und ihre Semantik

## Datenbankzustand

Für jeden Zustand *state* einer Datenbank besitzt ein ER-Schema folgende Zuordnungen:

$$E \mapsto \mathit{state}(E) \subseteq \mathit{dom}(E)$$

$$R(E_1, \dots, E_n) \mapsto \mathit{state}(R) \subseteq \mathit{state}(E_1) \times \dots \times \mathit{state}(E_n)$$

$$E(\dots, A_i : T, \dots) \mapsto \mathit{state}(A_i) : \mathit{state}(E) \rightarrow \mathit{dom}(T)$$

$$R(\dots; \dots, A_j : T, \dots) \mapsto \mathit{state}(A_j) : \mathit{state}(R) \rightarrow \mathit{dom}(T)$$

Vorausgesetzt sind passende Interpretationen für  $\mathit{dom}(E)$  und  $\mathit{dom}(T)$ : Mengen möglicher Entities für Entity-Typen und Wertebereiche für Datentypen.