













































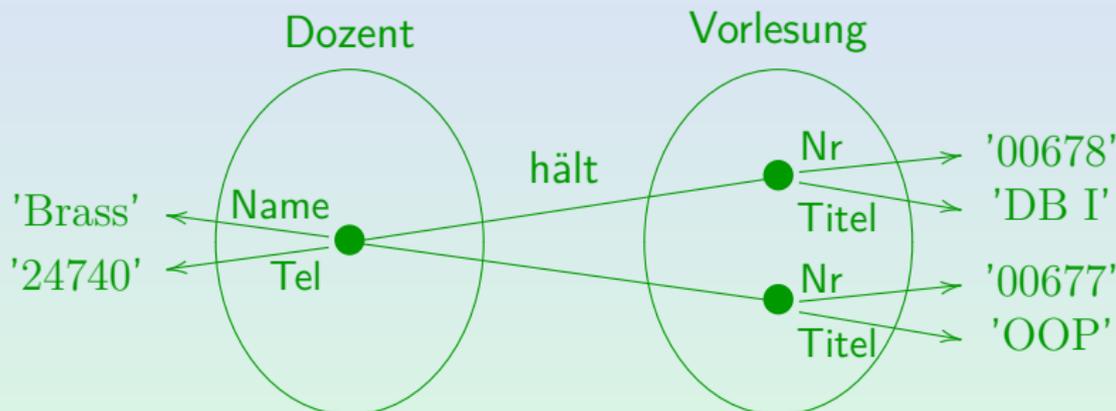






# Beispiel (2)

## Möglicher Zustand:



Es gibt in diesem Zustand ein Entity (Objekt) vom Typ "Dozent", dies hat z.B. den Wert "Brass" für das Attribut "Name". Entsprechend gibt es zwei Entities vom Typ "Vorlesung". Das vorhandene Dozent-Entity steht zu beiden Vorlesungs-Entities in der Beziehung "hält".







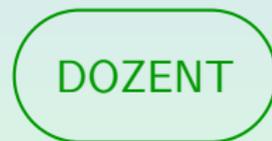






# Entity-Typen: Notation

- In der Barker-Notation wird für Entity-Typen eine “Softbox” verwendet, d.h. ein Rechteck mit abgerundeten Kanten.
- In das Rechteck wird oben zentriert der Name des Entity-Typs geschrieben:



Es ist üblich, als Entity-Typ-Name die Singular-Form eines Nomens zu benutzen. Im Buch von Barker und im Werkzeug “Oracle Designer” werden Entity-Typen immer ganz in Großbuchstaben geschrieben.



# Attribute: Notation (2)

- Jedem Attribut wird genau eins der folgenden drei Symbole vorangestellt:

#: Dieses Attribut ist Teil des Primärschlüssels.

Der Primärschlüssel des Entity-Typs ist die Kombination aller mit # markierten Attribute. Man kann im Diagramm also nur einen Schlüssel pro Entity-Typ darstellen.

\*: Dieses Attribut erlaubt keine Nullwerte.

Primärschlüssel erlauben natürlich auch keine Nullwerte. In der Original-Notation im Buch von Barker wurde "# \*" zusammen angegeben. Beim Werkzeug "Oracle Designer" wurde dies vereinfacht.

o: Das Attribut ist optional (erlaubt Nullwerte).

# Attribute: Notation (3)

- Im neueren Werkzeug “Oracle SQL Developer Data Modeler” gibt es zwei Spalten von Symbolen:
  - In der ersten Spalte steht “#” um die Primärschlüsselattribute zu markieren, bzw. nichts, wenn das Attribut nicht zum Primärschlüssel gehört.

Man kann Alternativschlüssel abspeichern, aber sie werden nicht graphisch dargestellt.
  - In der zweiten Spalte steht “\*” für Attribute, die nicht Null sein können, bzw. “o”, falls das Attribut Nullwerte erlaubt.

Die theoretisch mögliche Kombination “# o” ist ausgeschlossen.





## Relationships: Notation (2)

- Ein Relationship hat immer zwei Namen, einen für jede Richtung.

Man schreibt immer den Namen für die Beziehung von  $A$  nach  $B$  in die Nähe von  $A$ , und den Namen für die Beziehung von  $B$  nach  $A$  in die Nähe von  $B$ . Bei horizontalen Verbindungen ist es üblich, den Beziehungsnamen links über die Verbindungslinie zu schreiben, und rechts unter die Verbindungslinie. Das ist aber keine Vorschrift.

- Rekursive Relationships (Beziehungen von einem Entity-Typ zum gleichen Entity-Typ) sind zulässig.
  - Beispiel: Die “ist Voraussetzung für”-Beziehung zwischen zwei Vorlesungen.







# ER-Schemata: Semantik (1)

Ein DB-Zustand  $\mathcal{I}$  interpretiert die Symbole im Schema durch Definition einer

- endlichen Menge  $\mathcal{I}[E]$  für jeden Entity-Typ  $E$ ,
- Abbildung  $\mathcal{I}[A]: \mathcal{I}[E] \rightarrow \text{val}(D)$  für jedes Attribut  $A$  eines Entity-Typs  $E$ , wobei  $D$  der Datentyp von  $A$  und  $\text{val}(D)$  die Domain (Wertemenge) von  $D$  ist,

Wenn das Attribut Nullwerte erlaubt (Markierung mit "o"), handelt es sich um eine partielle Abbildung. Alternativ kann man eine totale Abbildung in  $\text{val}(D) \cup \{\text{null}\}$  verwenden.

- Relation  $\mathcal{I}[R] \subseteq \mathcal{I}[E_1] \times \mathcal{I}[E_2]$  für jedes Relationship  $R$  zwischen Entity-Typen  $E_1$  and  $E_2$ ,

# ER-Schemata: Semantik (2)

- Dies ist ein Spezialfall der logischen Interpretation, die in Kapitel 5 vorgestellt wurde.

Die logische Interpretation enthält formal auch die Interpretation der Datentypen, diese ist aber fest (gehört nicht zum Zustand).

- Die Haupteinschränkungen des ER-Modells sind:

- Es können nur neue Sorten mit endlichen Domains eingeführt werden (Entity-Typen).

Die Datentypen sind ja vorab fest gegeben (ins DBMS eingebaut).

- Neue Funktionen nur von Entity-Typen zu Datentypen (Attribute).
- Neue Prädikate müssen zwei Entity-Typen als Argumente haben (Relationships).

# ER-Schemata: Semantik (3)

## Beispiel-Zustand:

- $\mathcal{I}[\text{Dozent}] = \{sb, ms\}$
- $\mathcal{I}[\text{Name}] = f_1$ , mit  
 $f_1(sb) = \text{'Brass'}$ ,  $f_1(ms) = \text{'Spring'}$ .
- $\mathcal{I}[\text{Tel}] = f_2$ , mit  
 $f_2(sb) = 49404$ ,  $f_2(ms) = 49429$ .
- $\mathcal{I}[\text{Vorlesung}] = \{db, ds, dp\}$
- $\mathcal{I}[\text{Nr}] = g_1$ , mit  
 $g_1(db) = 20727$ ,  $g_1(ds) = 42232$ ,  $g_1(dp) = 40492$ .
- $\mathcal{I}[\text{Titel}] = g_2$ , mit  
 $g_2(db) = \text{'DB'}$ ,  $g_2(ds) = \text{'Data Str.'}$ ,  $g_2(dp) = \text{'Doc. Proc.'}$ .
- $\mathcal{I}[\text{hält}] = \{(sb, db), (sb, ds), (ms, ds), (ms, dp)\}$ .





# Schlüssel (3)

## Graphische Syntax:

- Ein Entity-Typ kann mehrere Schlüssel haben, aber im Diagramm kann man nur einen angeben, den sogenannten Primärschlüssel.
- Die Attribute, die den Primärschlüssel bilden, werden mit “#” markiert:



# Schlüssel (4)

## Spezifikation mit Logik:

- Schlüsselbedingungen können als logische Formeln spezifiziert werden. Z.B. bedeutet der Schlüssel "Vorname, Nachname" von "Dozent":

$\forall$  Dozent X, Dozent Y:

$X.Vorname = Y.Vorname \wedge$

$X.Nachname = Y.Nachname \rightarrow X = Y.$

- Äquivalente Formulierung:

$\neg \exists$  Dozent X, Dozent Y:

$X.Vorname = Y.Vorname \wedge$

$X.Nachname = Y.Nachname \wedge X \neq Y.$

# Schlüssel (5)

## Implikation von Schlüsselbedingungen:

- Schlüsselbedingungen werden schwächer (mehr Zustände gültig), wenn Attribute hinzugefügt werden.
- Z.B. betrachten wir die Professoren Stefan Posch, Nina Brass, Stefan Brass. Dieser Zustand
  - verletzt die Schlüsselbedingung für **Vorname**,  
Es gibt zwei "Stefans".
  - verletzt die Schlüsselbedingung für **Nachname**,
  - erfüllt die Schlüsselbedingung für die Kombination von **Vorname, Nachname**.





# Schlüssel (8)

## Auswahl des Primärschlüssels:

- Wenn es mehrere Schlüssel gibt, sollte man als Primärschlüssel möglichst einen wählen,
  - der nur aus einem Attribut besteht, und
  - möglichst nie geändert wird.

Nach der Übersetzung ins relationale Modell wird der Primärschlüssel in anderen Tabellen verwendet, die sich auf Entities dieses Typs beziehen. In manchen Systemen kann der Zugriff über den Primärschlüssel besonders schnell sein. Ansonsten ist die Wahl nicht sehr wichtig.

- Die übrigen Schlüssel werden Alternativ-, Sekundär- oder UNIQUE-Schlüssel genannt.

Da sie nicht im Diagramm angegeben werden können, müssen sie in zusätzlicher Dokumentation spezifiziert werden.







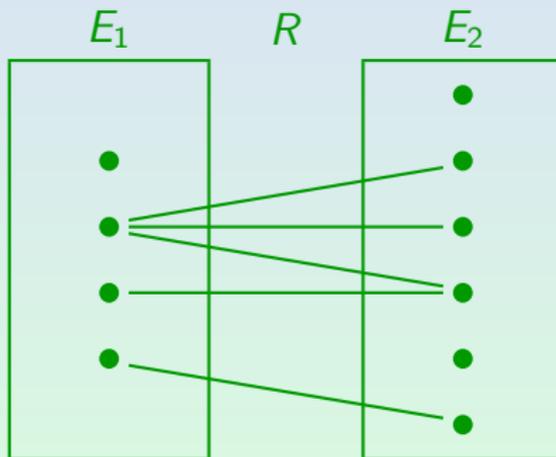


# Inhalt

- 1 DB-Entwurf
- 2 Integritätsbedingungen
- 3 Grundlegende ER-Konstrukte
- 4 Kardinalitäten**
- 5 Schwache Entities
- 6 Schema-Qualität

# Kardinalitäten: Motivation (1)

Allgemeines Relationship:







## Kardinalitäten: Maximum (2)

- In der Barker-Notation wird die Klassifizierung der Relationships in “1:1”, “1:n” und “n:m” über sogenannte “Krähenfüße” an den Enden ausgedrückt:



- Dies ist eine “Eins-zu-viele Beziehung”:
  - Ein Dozent kann viele Vorlesungen halten.
  - Jede Vorlesung wird nur von einem Dozenten gehalten (weil auf Dozentenseite kein Krähenfuß).
- Der Krähenfuß drückt intuitiv die Aufspaltung aus.

















# Kardinalitäten: Minimum (3)

- Selbstverständlich können auch beide Hälften der Verbindungslinie gestrichelt oder beide durchgezogen sein:



- Hier ist nicht verlangt, dass ein Student mindestens eine Vorlesung hört. Es ist auch nicht verlangt, dass eine Vorlesung mindestens einen Teilnehmer hat.

Eine Vorlesung ohne Teilnehmer wird gestrichen, wenn das Semester begonnen hat. Aber wenn sie angekündigt wird, ist völlig normal, dass sie zunächst keine Teilnehmer hat (temporäre Situationen bedenken!).



























# Schwache Entities (4)

- Zusätzlich besteht eine Existenzabhängigkeit:  
Wird ein Gebäude abgerissen (gelöscht),  
verschwinden die Räume darin automatisch.
- Existenz-Abhängigkeiten gibt es aber schon bei der  
Minimalkardinalität 1 (durchgezogene Linie):  
Jedes Entity muss dann eine Beziehung zum Entity des  
anderen Typs haben.

Es kann also nicht getrennt existieren. Die automatische Löschung ist eine Form der aktiven Integritätssicherung, die später im Kapitel 14 besprochen wird. Sie ist nicht zwingend mit schwachen Entities verbunden (obwohl recht typisch).

- Die Existenzabhängigkeit sollte also nicht das Erste/Einzige sein, was Ihnen zu schwachen Entities einfällt!



# Schwache Entities (6)

- In der Barker-Notation wird ein schwacher Entity-Typ durch einen Querstrich an der Beziehung zum übergeordneten Entity gekennzeichnet:



- Das Schlüsselattribut "RNr" des übergeordneten Entity-Typs wird beim schwachen Entity-Typ nicht angegeben.

Über die Beziehung ist aber immer genau eine Rechnung erreichbar, die dann eine bestimmte Rechnungsnummer hat. Es gibt also die gleiche Information wie vorher, nur nicht mehr redundant.

# Schwache Entities (7)

- Die mit “#” markierten Attribute reichen jetzt nicht mehr zur Identifikation, sondern die mit dem Querstrich markierte Beziehung wird Teil des Schlüssels.
- Die Bedingung ist hier also, dass es nie zwei verschiedene POSITION-Entities gibt, die mit dem gleichen RECHNUNG-Entity verbunden sind, und den gleichen Wert für das Attribut “Pos” haben:

$\forall$  RECHNUNG R, POSITION P1, POSITION P2:  
 $\text{hat}(R, P1) \wedge \text{hat}(R, P2) \wedge P1.\text{Pos} = P2.\text{Pos} \rightarrow P1 = P2$

# Schwache Entities (8)

- Man beachte, dass es immer ein eindeutig bestimmtes Entity geben muss, das den Kontext definiert.
- Ein Krähenfuß auf der anderen Seite ist ein Syntaxfehler:



- Querbalken auf gestrichelten Linien sind ebenso ein Fehler (entspricht Nullwerten im Primärschlüssel):







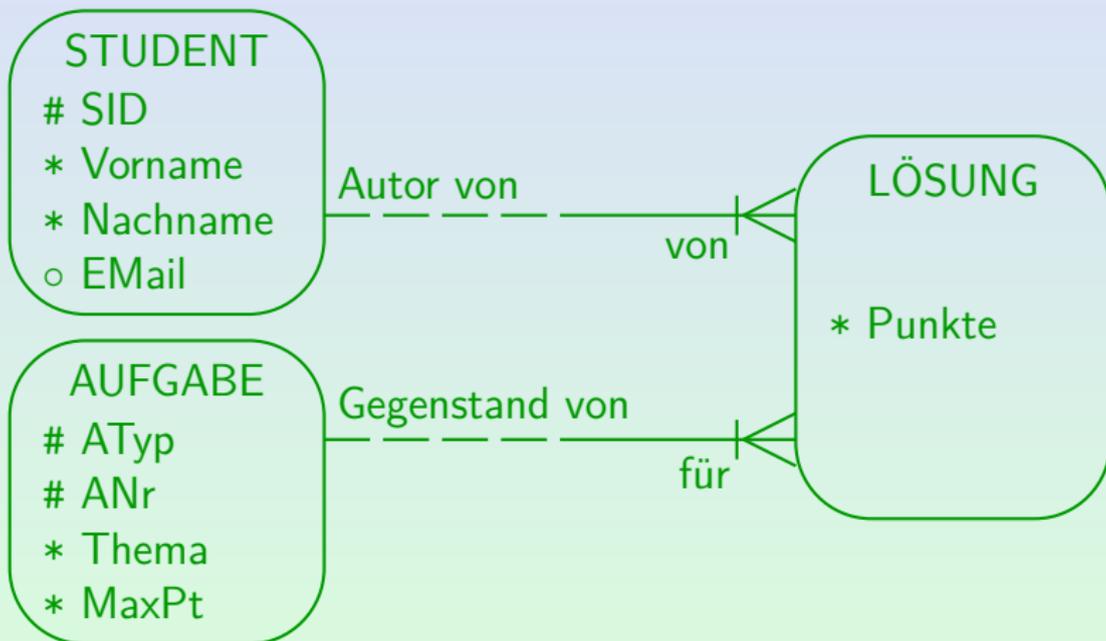








# Association-Entities (4)



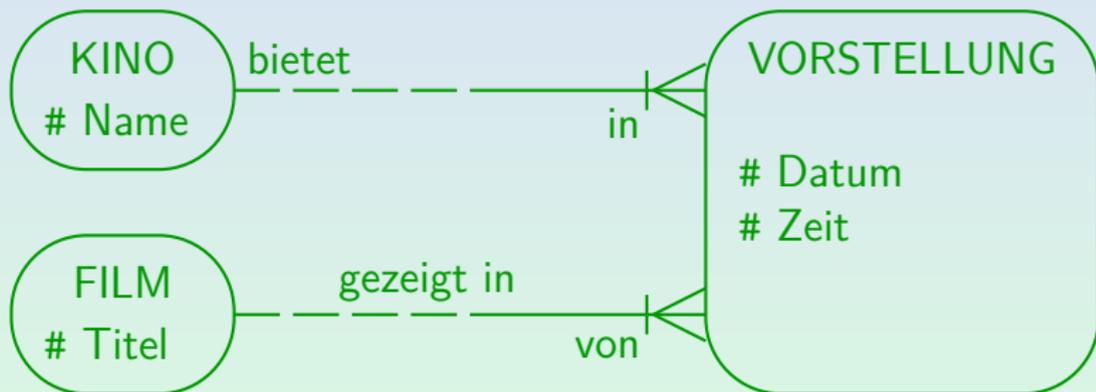
# Association-Entities (5)

- Ein schwacher Entity-Typ kann also mehrere “Master” haben (d.h. mehrere Beziehungen, die zur Identifikation beitragen).
- Solche Entity-Typen heißen “Association-Entities”, weil sie Beziehungen zwischen den Master-Typen entsprechen.

Sie sind natürlich auch weiter “schwache Entity-Typen”, weil Beziehungen an ihrer Identifikation beteiligt sind, und sie nicht über global eindeutige eigene Schlüsselattribute verfügen.

# Association-Entities (6)

- Auch ein Association Entity kann noch zusätzlich weitere Schlüsselattribute haben:



Ein Kino zeigt den gleichen Film mehrfach (zu verschiedenen Zeiten).  
Beziehungsattribute würden hier nicht ausreichen, da diese nicht Teil des Schlüssels sein können (würde Semantik der Beziehung ändern).















