Das Relationen-Modell

Einführung

- Geht auf klassische Arbeit von Codd zurück (1970)
- Meistgenutztes Datenmodell
  - Anbieter: IBM (DB2), Informix, Microsoft (SQL-Server), Sybase, Oracle
- Legacy-Systeme in älteren Modellen (z.B. Netzwerk-Datenmodell, hierarch. Datenmodell)
  - z.B. IMS von IBS
- Konkurrenz durch objektorientierte Modelle
  - ObjectStore, Versant
- Zur Zeit: Synthese des relationalen Modells mit objektorientierten Konzepten:
  - Informix Universal Server
  - Oracle 8i
  - UniSQL
  - Sprachstandards: SQL-99

Relationenmodell - Übersicht

- Datenstruktur: Relation (Tabelle)
  - Alle Informationen durch Werte dargestellt
  - Integritätsbedingungen auf/zwischen Relationen
- Operatoren:
  - Vereinigung, Differenz
  - Kartesisches Produkt
  - Projektion
  - Selektion
  - Zusätzlich: Grundoperationen: Einfügen, Löschen, Ändern
  - Verknüpfung zu komplexeren Operationen möglich
- Entwurfstheorie
  - Normalformenlehre
  - Synthese von Relationen
  - Sagt was “gute“ Relationen sind (sichere Ausdrücke)

Relationenmodell: Grundkonzepte

- Relationale Datenbank: Menge von Relationen
- Relation (mathematisch)

  $R(A_1, A_2 \ldots A_n) \subseteq W(A_1) \times W(A_2) \ldots \times W(A_n)$

  Teilmenge des kartesischen Produkts einer Menge von Wertebereichen
- Relation besteht aus 2 Teilen:

  Instanz: Tabelle mit Zeilen (Tupel) und Spalten

  Kardinalität: Anzahl der Tupel

  Grad einer Relation: Anzahl der Spalten

  Schema: Name der Relation und Name und Typ aller Spalten

  Beispiel:

  Student (sid:string, name:string, login:string, alter:integer, durchsch:real)

  Relationen sind Mengen von Tupeln (d.h. alle Tupel sind distinkt)

Beispiel: Instanz einer Relation *Student*

sid	name	login	alter	durchsch
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

- Kardinalität = 3, Grad = 5
- alle Tupel sind distinkt

Primärschlüssel

- Ein Schlüssel einer Relation ist eine Menge von Attributen, für die gilt:
  1. Es gibt keine zwei unterschiedlichen Tupel, die in allen Schlüsselattributen den gleichen Wert haben (Eindeutigkeits-eigenschaft)
  2. Es gibt keine Teilmenge von Attributen des Schlüssels, die Eigenschaft 1 erfüllt
- Anmerkungen:
  - Wenn es mehrere Attribute / Attributkombinationen gibt, die die Eigenschaften 1 und 2 erfüllen (Schlüsselkandidaten), wird darunter ein Primärschlüssel ausgewählt und als solcher definiert
  - Eine Menge von Attributen (die einen Schlüsselkandidaten enthält) mit identifizierender Eigenschaft in einer Relation wird auch *Superkey* genannt

Fremdschlüssel

- Definition

  Ein Fremdschlüssel bezüglich einer Relation R1 ist ein (ggf. zusammengesetztes) Attribut FS einer Relation R2, für das zu jedem Zeitpunkt gilt: zu jedem Wert (ungleich NULL) von FS muß ein gleicher Wert des Primärschlüssels PS oder eines Schlüsselkandidaten SK in in irgendeinem Tupel von R1 enthalten sein.

  Ein Fremdschlüssel ist wie ein “logischer Pointer“

- Bemerkungen

  Fremdschlüssel und zugehöriger Primärschlüssel gestatten die Realisierung von Beziehungen.

  Fremdschlüssel können Nullwerte aufweisen, wenn sie nicht Teil eines Primärschlüssels sind

  Schlüsselkandidaten können Nullwerte aufweisen, wenn nicht explizit NOT NULL definiert ist

  Eine Relation kann mehrere Fremdschlüssel besitzen, die die gleiche oder verschiedene Relationen realisieren

  Referenzierte und referenzierende Relationen sind nicht notwendig verschieden (Selbstreferenz)

  .

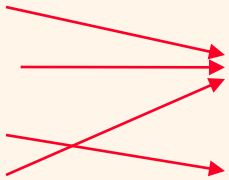
Beispiel: Referentielle Integrität

Nur Studenten, die in der Tabelle *Student* erfaßt sind, dürfen sich in Kurse einschreiben.

```
CREATE TABLE Einschreibung
  (sid CHAR(20),
   kid CHAR(20),
   note CHAR(2),
    PRIMARY KEY  (sid,kid),
    FOREIGN KEY (sid) REFERENCES Student )
```

Einschreibung

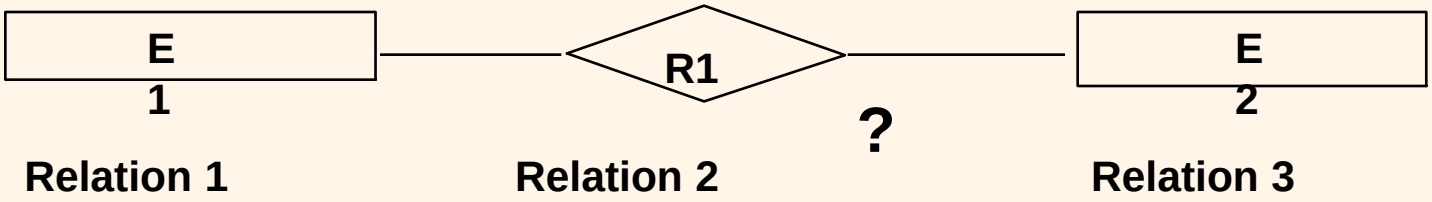
sid	kid	note
53666	Datenbanken	3
53666	Reggae	2
53650	Topologie	1
53666	Geschichte	2



Student

sid	name	login	alter	durchsch
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Abbildung vom ERM in Relationen



- Kriterien
  - Informationserhaltung
  - Minimierung der Redundanz
  - Minimierung des Verknüpfungsaufwandes
  - Natürlichkeit der Abbildung
  - Keine Vermischung von Objekten
  - Verständlichkeit

Zwei Entity-Mengen mit 1:n-Beziehung

Darstellungsmöglichkeiten im RM

1. Verwendung von drei Relationen

   ABT (ANR, ANAME, ...)

   PERS (PNR, PNAME, ...)

   ABT-ZUGEH (ANR, PNR)

   – Normalerweise wird der 1:n-Beziehungstyp nur dann auf eine eigene Relation abgebildet, wenn er beschreibende Attribute besitzt. Minimierung der Redundanz

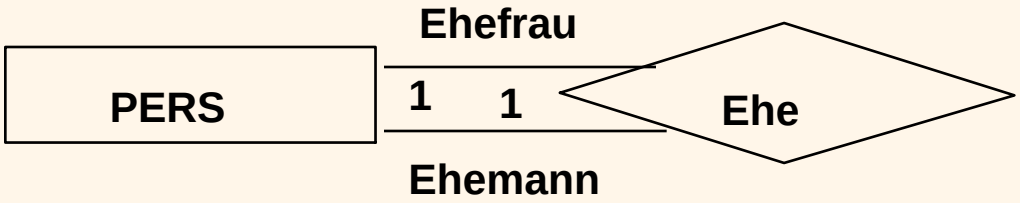
2. Verwendung von zwei Relationen

   ABT (ANR, ANAME, ...)

   PERS (PNR, PNAME, ... , ... ANR)

   – Standardabbildung des 1:n-Beziehungstyps mit Hilfe von Primär- und Fremdschlüssel

Eine Entity-Menge mit 1:1-Beziehung



Darstellungsmöglichkeiten im RM

1. Verwendung von zwei Relationen

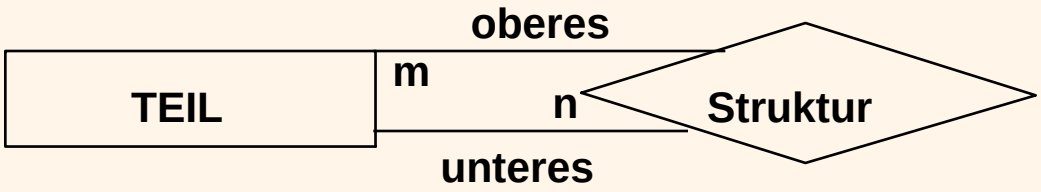
   PERS (PNR, PNAME, ...)

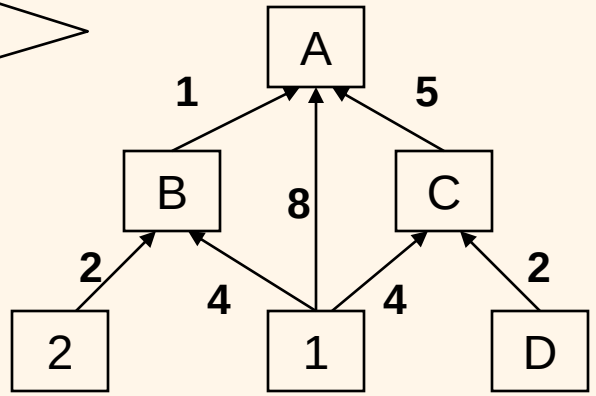
   EHE (MPNR, FPNR)

2. Verwendung von einer Relation

   PERS (PNR, PNAME, ... , ... GÁTTE)

Eine Entity-Menge mit m:n-Beziehung



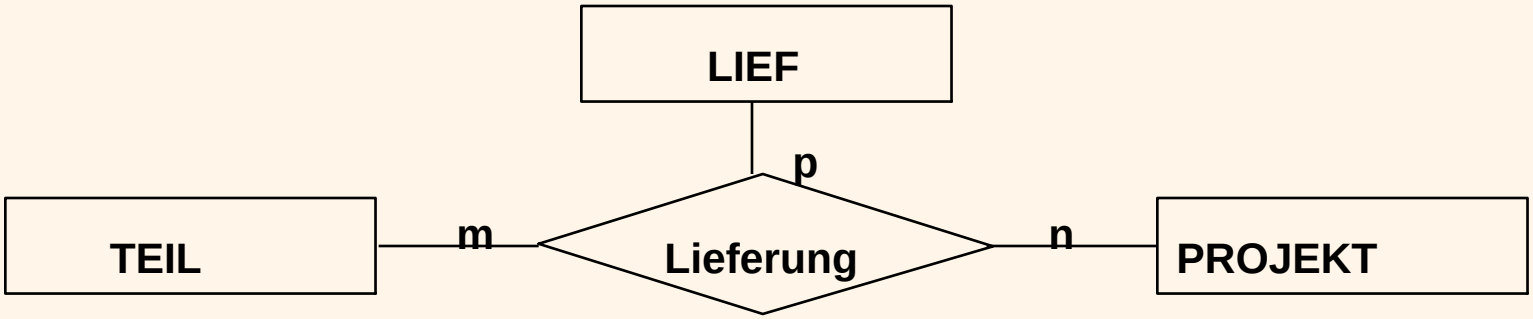


Darstellungsmöglichkeiten im RM

TEIL (TNR, TBEZ, ...
STRUKTUR (OTNR, UTNR, ANZAHL)

STRUKTUR	OTNR	UTNR	ANZAHL
	A	B	1
	A	C	5
	A	1	8
	B	1	4
	B	2	2
	C	1	4
	C	D	2

Drei Entity-Mengen mit (m:n:p)-Beziehung



Darstellungsmöglichkeiten im RM

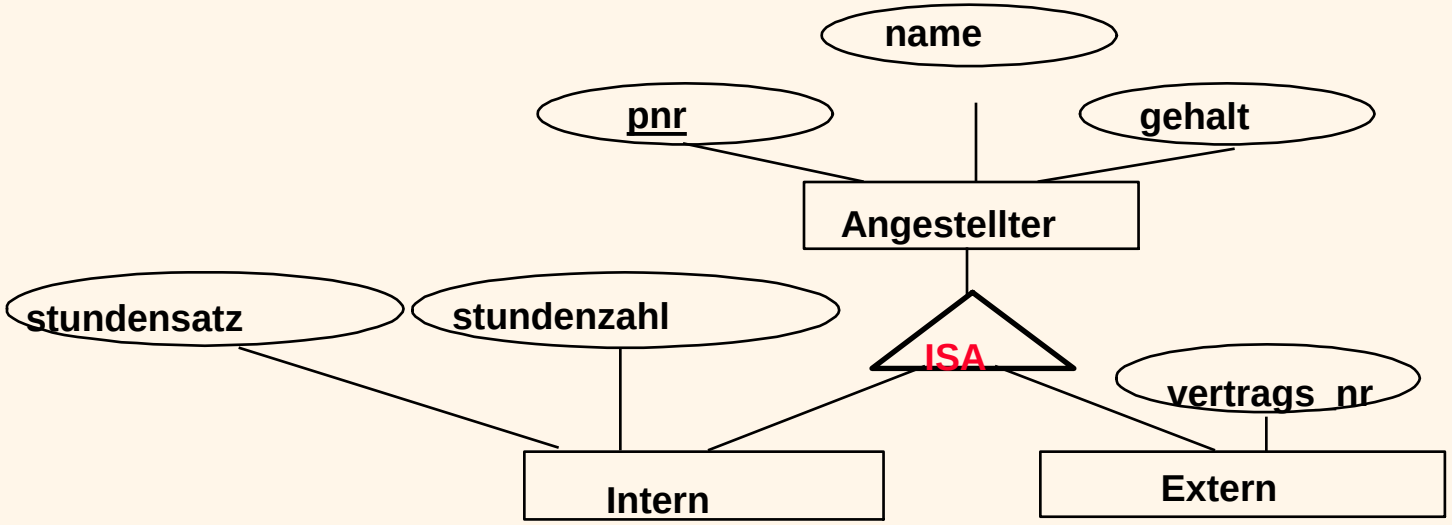
LIEF (LNR, LNAME,L-ORT ...)

PROJEKT (PRONR,PRONAME, P-ORT ...)

TEIL (TNR, TBEZ, GEWICHT ...)

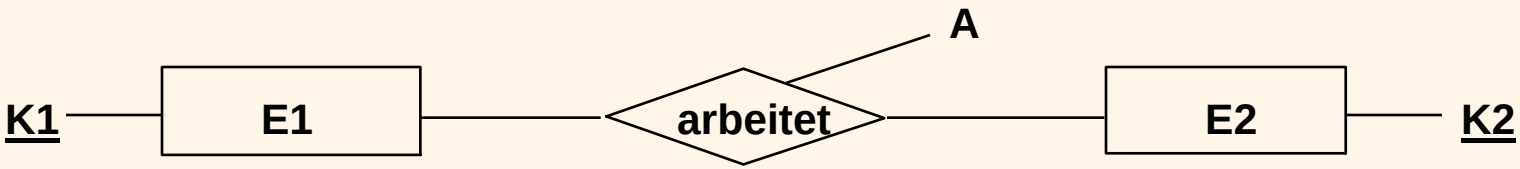
LIEFERUNG (LNR, PRONR, TNR, ANZAHL, DATUM)

Abbildung von ISA-Hierarchien



- 3 Relationen: Angestellter, Intern, Extern
  - Intern: Jeder Angestellte ist in ANGESTELLTER. Für interne Angestellte sind zusätzliche Infos in INTERN (stundensatz, stundenzahl,pnr), Löschabhängigkeit zum referenzierten Tupel in ANGESTELLTER
  - Anfragen auf allen Angestellten einfach, für zusätzliche Infos Join erforderlich
- Alternative: 2 Relationen Intern und Extern (“Flachklopfen“)
  - INTERN (pnr, name, gehalt, stundensatz, stundenzahl)
  - EXTERN (pnr, name, gehalt, vertrags_nr)
  - Jeder Beschäftigte gehört in eine der beiden Relationen

Abbildungsregeln Beziehungen - Relationen



... nur “E0“	(1,1) (1,1)	E0 (<u>K1</u>, K2, A) oder E0 (<u>K2</u>, K1, A)
bleibt E1 + E2	(0,1) (1,1) (1,*) (1,1) (0,*) (1,1)	E1 (<u>K1</u>, ...) E2 (<u>K2</u>, ..., A, *K1*)
entsteht ein neues E3	(0,1) (0,1)	E3 (<u>*K1, K2*</u>, A) oder E3 (<u>*K2, K1*</u>, A)
	(1,*) (0,1) (0,*) (0,1)	E3 (<u>*K2, K1*</u>, A)
	(1,*) (1,*) (1,*) (0,*) (0,*) (0,*)	E3 (<u>*K1, K2*</u>, A)

Abbildung von Beziehungen

Darstellung einer 1:n-Beziehung

ABT (ABTNR ...,
....
PRIMARY KEY(ABTNR))

PERS (PNR ...,
ANR ...,
PRIMARY KEY(PNR)
FOREIGN KEY (ANR)
REFERENCE ABT)

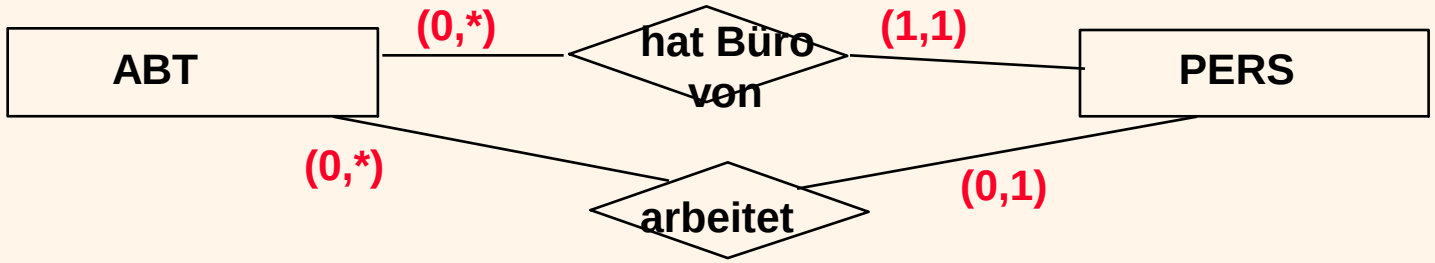
Jeder Angestellte PERS muß in einer Abteilung beschäftigt sein (1,1)

→ PERS.ANR ... NOT NULL

Ein (1,*)-Constraint kann in SQL2 nicht spezifiziert werden.

Abbildung von Beziehungen (2)

Darstellung mehrerer 1:n-Beziehungen




```
ABT (ABTNR ...,
  ....
  PRIMARY KEY(ABTNR))
```

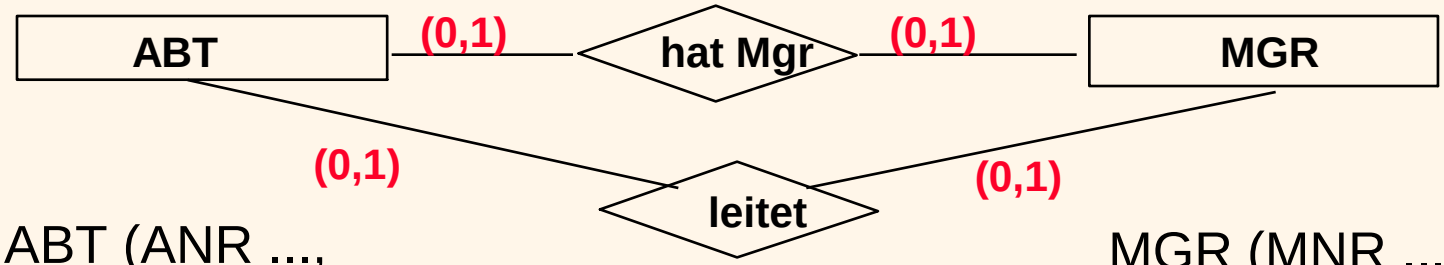
```
PERS (PNR ...,
  ANRB ... NOT NULL,
  ANRA ...,
  PRIMARY KEY(PNR)
  FOREIGN KEY (ANRA)
   REFERENCES ABT),
  FOREIGN KEY (ANRB)
   REFERENCES ABT)
```

Für jede FS-Beziehung benötigt man ein separates FS-Attribut

Mehrere FS-Attribute können auf dasselbe PS/SK-Attribut verweisen

Abbildung von Beziehungen (3)

Darstellung einer 1:1-Beziehung



ABT (ANR ...,
MNR .... UNIQUE
...
PRIMARY KEY(ANR)
FOREIGN KEY(MNR)
REFERENCES MGR)

MGR (MNR ...,
ANR ... UNIQUE,
...
PRIMARY KEY(MNR)
FOREIGN KEY
(ANR)
REFERENCES ABT),

Es sind symmetrische Lösungen möglich.

Zusätzlich:

Jede Abteilung hat einen Manager

→ ABT.MNR ... UNIQUE NOT NULL

Jeder Manager leitet eine Abteilung

→ MGR.ANR ... UNIQUE NOT NULL

Abbildung von Beziehungen (4)

Darstellung einer 1:1-Beziehung


```
ABT (ANR ...,
  MNR .... UNIQUE NOT NULL
  ...
  PRIMARY KEY(ANR)
  FOREIGN KEY(MNR)
    REFERENCES MGR)
```

```
MGR (MNR ...,
  ...
  PRIMARY KEY(MNR)
  FOREIGN KEY (MNR)
    REFERENCES ABT(MNR))
```

Es sind symmetrische Lösungen möglich. Die Nutzung des MNR-Attributs für beide FS-Beziehungen gewährleistet 1:1.

Der Fall (0,1)(0,1) ist so nicht darstellbar.

Abbildung von Beziehungen (5)

Darstellung einer m:n-Beziehung

PERS (PNR ...,
...
PRIMARY KEY(PNR))

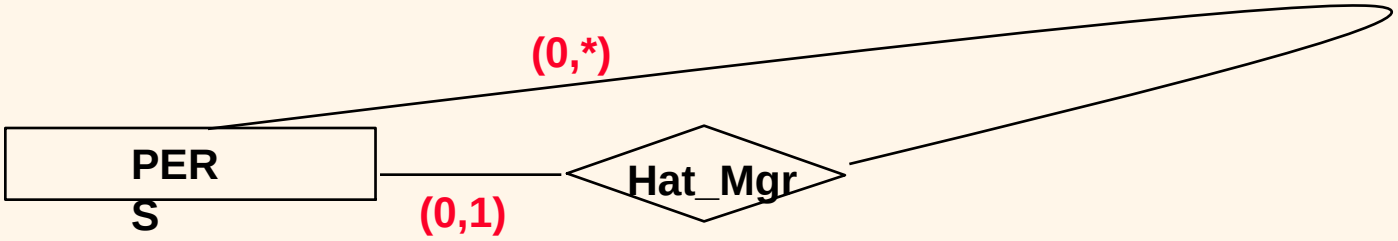
PROJ (JNR ...,
...
PRIMARY KEY(JNR))

MITARBEIT (PNR ...,
JNR ...,
PRIMARY KEY(PNR,JNR)
FOREIGN KEY (PNR) REFERENCES PERS)
FOREIGN KEY (JNR) REFERENCES PROJ)

Diese Standardlösung erzwingt eine Existenzabhängigkeit von MITARBEIT. Soll dies vermieden werden, dürfen die Fremdschlüssel von MITARBEIT nicht als Teil des Primärschlüssels spezifiziert werden.

Abbildung von Beziehungen (6)

Darstellung einer 1:n-Beziehung als Selbstreferenz



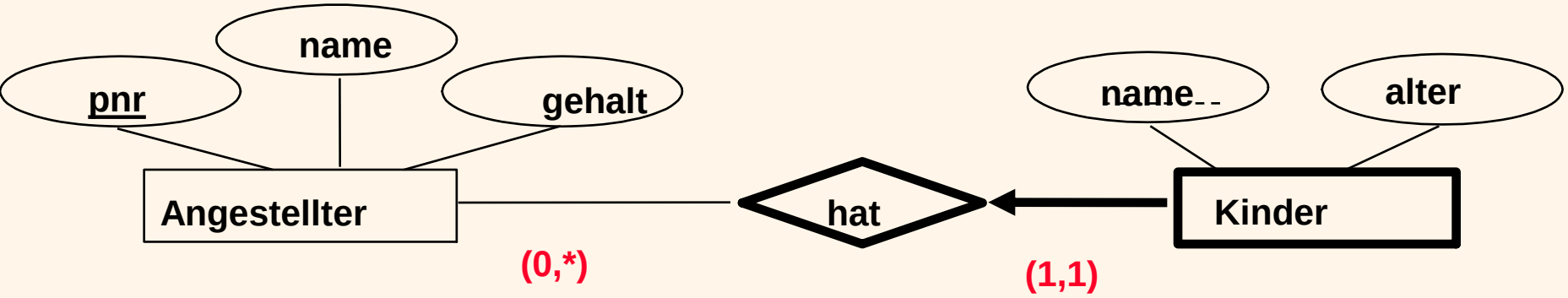
PERS (PNR ...,
MNR ...,
...
PRIMARY KEY(PNR)
FOREIGN KEY (MNR) REFERENCES PERS(PNR))

Erlaubt die Darstellung der Personalhierarchie eines Unternehmens. Ist (0,1), weil die obersten Manager einer Hierarchie keinen Manager haben.

MNR ... NOT NULL nur realisierbar, wenn die obersten Manager als ihre eigenen Manager realisiert werden. Verursacht jedoch andere Probleme (z.B. Konsistenzprüfung)

Rückblick: Schwache Entities

- Schwaches Entity (weak entity) kann eindeutig identifiziert werden nur über den Primärschlüssel einer anderen (Owner) Entity.
- Owner Entity und Weak Entity müssen in einer 1:n-Beziehung stehen (ein Owner, mehrere Weak Entities)



Jedes Entity aus Kinder **muß** an der Beziehung teilnehmen (*total Participation Constraint*)

Übersetzung schwacher Entity-Menge

- Schwache Entity-Menge und identifizierende Beziehung werden in eine einzige Tabelle übersetzt
  - Wenn das Owner-Entity (z.B. der Angestellte) gelöscht wird, müssen auch alle davon abhängigen schwachen Entities gelöscht werden (Existenzabhängigkeit).

```
CREATE TABLE Abhängig (
   name CHAR(20),
   alter INTEGER,
   pnr CHAR(11) NOT NULL,
   PRIMARY KEY (name, pnr),
   FOREIGN KEY (pnr) REFERENCES Angestellter,
      ON DELETE CASCADE)
```

Zusammenfassung Relationen-Modell

- Tabellarische Darstellung von Daten
- Einfach und intuitiv, zur Zeit meistverbreitetes Modell
- Grundlage relationaler Anfragesprachen
- Integritätsbedingungen können durch den Modellierer spezifiziert werden, basierend auf der Applikationssemantik
  - 2 bedeutende Constraints: Primär- und Fremdschlüsselbedingungen
  - zusätzlich immer Wertbereichsbedingungen (Domain Constraints)
- Regeln zur Transformation ER-Modell in relationales Modell
  - Behandlung der unterschiedlichen Beziehungstypen (1:1, 1:n, m:n)
  - Alle Beziehungstypen müssen im Prinzip durch (n:1)-Beziehungen dargestellt werden.
  - Es ist nur eine eingeschränkte Nachbildung von Kardinalitäts-restriktionen möglich.