Prof. DI Dr. Erich Gams

Relationenmodell

Vertiefung

informationssysteme htl-wels

To start….

Please, close your laptops

and just

Übersicht ☛ Was lernen wir?

- Kurze Wiederholung
- Generalisierung/Spezialisierung
- Die einzelnen MC Beziehungstypen und deren Auflösung bzw. Transformation
- Umwandlung rekursiver Relationship Types
- Beispiele

Transformationsregeln in a nut shell

- Wandle alle Entity Types in Relationen um.
- 1:N
  - Primärschlüssel des Entity Type auf der 1(oder C) Seite wandert als Fremdschlüssel zu Entity Type auf der N Seite
- 1:1
  - Zusammenlegen in 2 Relationen (1 Primärschlüssel wandert als Fremdschlüssel in die „andere“ Relation)
  - (Eventuell: nur 1 Tabelle)
- N:M
  - Relationship Type in eigene Tabelle umwandeln, Primärschlüssel beider Entity Types wandern in diese Tabelle

Generalisierung/Spezialisierung: 4 Varianten

- eine Relation für Supertyp und je eine pro Subtyp
  - Typisch für Spezialisierung: Relationen für Subtypen enthalten zusätzlich Primär-Schlüssel des Supertypen.
- nur eine Relation pro Subtyp
  - Typisch für Generalisierung: Relationen der Subtypen enthalten zusätzlich die Attribute des Supertypen.
- nur eine Relation Supertyp
  - Relation für Supertyp enthält alle Attribute der Subtypen und einen Diskriminator. Geeignet, falls die Anzahl der unterschiedlichen Attribute der Subtypen sehr klein ist bzw. überlappende Spezialisierung / Generalisierung.
- nur eine Relation Supertyp
  - Relation für Supertyp enthält alle Attribute der Subtypen und pro Subtyp ein Flag ob entsprechende Attribute Anwendung finden. Geeignet, falls die Anzahl der unterschiedlichen Attribute der Subtypen sehr klein ist und disjunkte Spezialisierung / Generalisierung.

1)

Person

P#	Vorname	Nachname
1212	Susi	Kompetent
1213	Bodo	Bach

Arzt

P#	Fachgebiet
1212	Virologie

Pflegepersonal

P#	Zusatzausbildung
1213	Intensivpflege

2)

Arzt

P#	Fachgebiet	Vorname	Nachname
1212	Virologie	Susi	Kompetent

Pflegepersonal

P#	Zusatzausbildung	Vorname	Nachname
1213	Intensivpflege	Bodo	Bach

3)

Person

P#	Vorname	Nachname	Fachgebiet	Zusatzausbildung	Typ
1212	Susi	Kompetent	Virologie	null	Arzt
1213	Bodo	Bach	null	Intensivpflege	Pflegepersonal

=Diskriminator

4)

Person

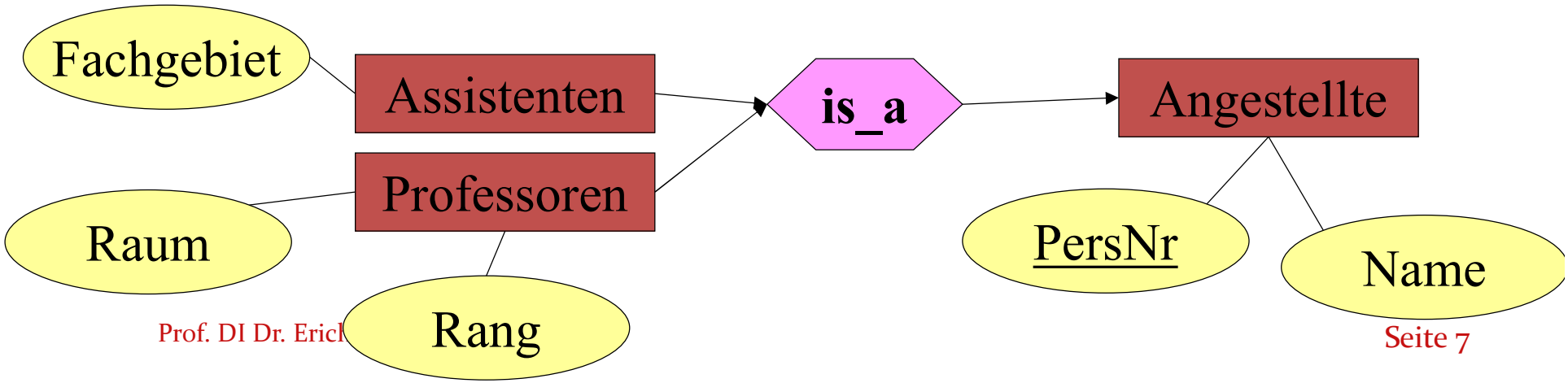
P#	Vorname	Nachname	Fachgebiet	Zusatzausbildung	Arzt	Pflegepersonal
1212	Susi	Kompetent	Virologie	null	ja	nein
1213	Bodo	Bach	null	Intensivpflege	nein	ja

Generalisierung/Spezialisierung

- 1:1 bzw. 1:c Beziehung
- Angestellte: {[*PersNr, Name, Jobtitle*]}

  (eventuell Kategorisierungsattribut, Performance)

  (jobtitle NOT NULL =1:1, bei erlaubten Nullwerten = 1:c)
- Professoren: {[*PersNr, Rang, Raum*]}
- Assistenten: {[*PersNr, Fachgebiet*]}



Generalisierung/Spezialisierung (addon)

- 1:m bzw 1:cm Beziehung
  - Angenommen, ein Mitarbeiter kann sowohl Projektleiter und Programmierer sein.
- Lösung: Eine zusätzliche Tabelle aus Performancegründen

  Mitarbeiter (pers#, Vorname, Nachname)

  Projektleiter (pers#, Fremdsprache)

  Programmierer (pers#, Programmiersprache)

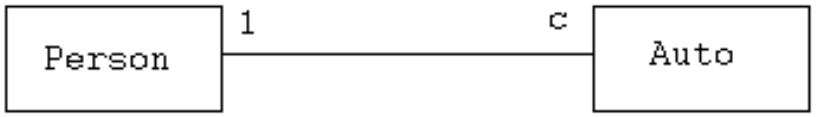
  Qualifikation (pers#, jobtitle)

Modifizierte Chen-Notation (MC-Notation)

- Die **Modifizierte Chen-Notation** (*Modified Chen Notation*, *MC-Notation*) ist eine Erweiterung der Chen-Notation.

Notation	Kardinalität (Vorkommen)
MC	0,1, …, n mal
M	1,….,n mal
C	0 oder 1
1	genau 1 mal

Die 1:c Beziehung

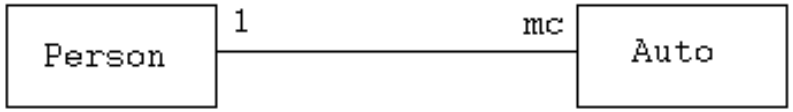


- Jede Person hat entweder ein oder kein Auto, und jedes Auto ist genau einer Person zugeordnet.
- Lösung:
- In die Tabelle mit dem Assoziationstyp „c“ wird der PK der anderen Tabelle als FK aufgenommen.

```
Person (Pers#, VN, NN)
Auto (Auto#, Marke, Bezeichnung, Baujahr, Pers#)
```

- In der Definition der Tabelle in SQL muss das FK-Attribut „Pers#“ noch zusätzlich
- die Eigenschaft „**UNIQUE**“ erhalten.
  - **UNIQE:**
    Dies bedeutet, dass jede Attributsausprägung nur einmal vorkommen darf.
- Der FK darf keinen Nullwert annehmen.

Die 1:mc Beziehung



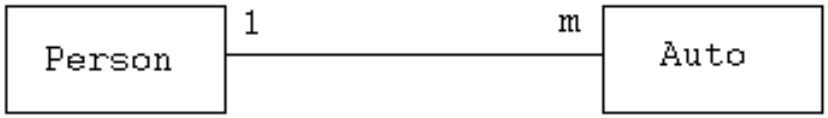
- Jede Person kann beliebig viele Autos besitzen (>=0), wobei jedes Auto genau einer Person zugeordnet ist.

- Auflösung: In der Tabelle mit der Assoziation „mc“ wird der PK der anderen Tabelle als FK übernommen.

  ```
  Person (Pers#, VN, NN)
  Auto (Auto#, Marke, Bezeichnung, Baujahr, Pers#)
  ```

- Im FK darf derselbe Attributwert mehrmals vorkommen (Unterschied zur Lösung 1:c mit UNIQUE), und keinen Nullwert annehmen.

Die 1:m Beziehung



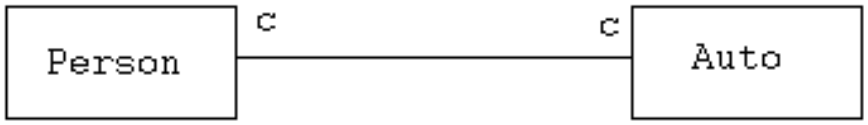
- Jede Person besitzt mindestens ein Auto, und jedes Auto gehört genau einer Person.
- Auflösung: In der Tabelle mit der Assoziation „m“ wird der PK der anderen Tabelle als FK übernommen.

```
Person (Pers#, VN, NN)
Auto (Auto#, Marke, Bezeichnung, Baujahr, Pers#)
```

- Im FK darf derselbe Attributwert mehrmals vorkommen (wie 1:cm), und keinen Nullwert annehmen.

- Hinweis: Durch diese Tabellendefinition kann aber die Konsistenzbedingung, dass jedes Auto genau einer Person zugeordnet und das jede Person genau über ein Auto verfügt nicht gewährleistet werden => **Progammierung**!

Die c:c Beziehung



- Jede Person kann ein Auto besitzen (d.h. besitzt kein oder genau ein Auto) und jedes Auto kann einer (Zahlwort) Person zugeordnet sein.
- Auflösung:

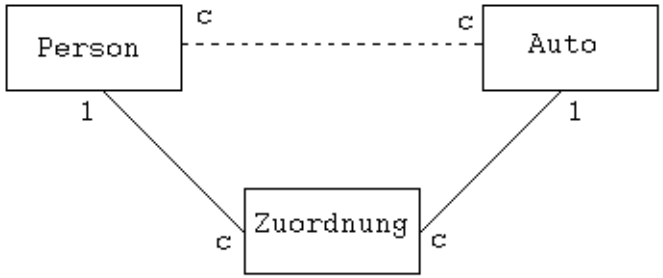
```
Person (Pers#, VN, NN, Auto#)
Auto (Auto#, Marke, Bezeichnung, Baujahr, Pers#)
```

- Die FK-Attribute müssen noch zusätzlich die Eigenschaft „UNIQUE“ erhalten, damit ein Auto bzw. eine Person nicht öfters zugeordnet werden kann.
- Diese Lösung führt aber zu *Nullwerten* im FK, was nicht gestattet ist. Daher erfolgt eine andere Lösung mittels *Transformation*!
- Transformation:

Person	(Pers#, VN, NN)
Auto	(Auto#, Marke, Bezeichnung, Baujahr)
Zuordnung	(Pers#, Auto#)

------- -------

Die c:c Beziehung

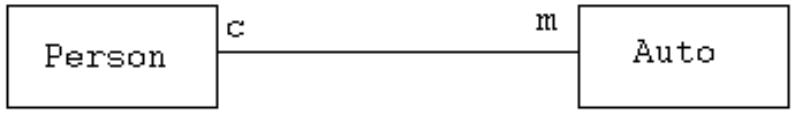


- Transformation: gestrichelt dargestellte c-c Beziehung -> zwei 1-c Beziehungen
  -> Eine neue Zuordnungstabelle/Transformationstabelle entsteht
- Der PK der Transformationstabelle besteht aus den FKs der beteiligten Tabellen. Zusätzlich müssen diese Attribute noch die Eigenschaft „**UNIQUE**“ erhalten!

- Transformationstabellenname:
  - „Fahrzeughalter“ oder
  - Schema „Tabelle1-Tabelle2-Zuordnung“ -> „Person-Auto-Zuordnung“.

- Hinweis: In der Praxis kann die Lösung mittels zweier FKs sehr wohl angetroffen werden. Man erspart sich dadurch die Transormationstabelle was die Performance erhöhen kann.

Die c:m Beziehung

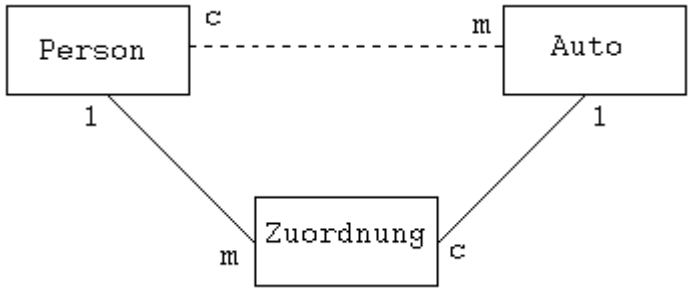


- Interpretation: Jede Person besitzt mindestens ein Auto (>=1) und jedes Auto kann einer Person zugeordnet sein (d.h. ist genau einer oder keiner Person zugeordnet).
- Auflösung:

  Person (Pers#, VN, NN)
  Auto (Auto#, Marke, Bezeichnung, Baujahr, Pers#)
- Im FK „Pers#“darf derselbe Werte mehrmals vorkommen. Diese Lösung führt aber gleichfalls zu *Nullwerten* im FK, was nicht gestattet ist. Daher erfolgt eine andere Lösung mittels *Transformation*!
- Transformation:

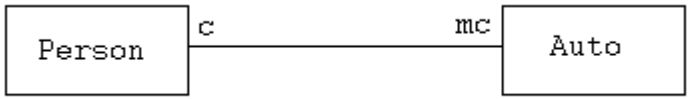
  Person (Pers#, VN, NN)
  Auto (Auto#, Marke, Bezeichnung, Baujahr)
  Zuordnung (Pers#, Auto#)

Die c:m Beziehung



- Transformation:
  - gestrichelt dargestellte konditionelle c-m Beziehung
  - ➔ 1-m und 1:c Beziehung
  - ➔ neue Zuordnungstabelle/Transformationstabelle entsteht.
- Zu jeder Person existiert mindestens ein Tupel in der Zuordnungstabelle. Ein Auto hingegen kann, muss aber nicht, einen Fahrzeughalter haben.
- Die Konsistenzbedingung der 1:m Beziehung („Zu jeder Person existiert mindestens ein Tupel in der Zuordnungstabelle“) lässt sich aber nicht durch eine Tabellendefinition realisieren -> Programmierung

Die c:mc Beziehung



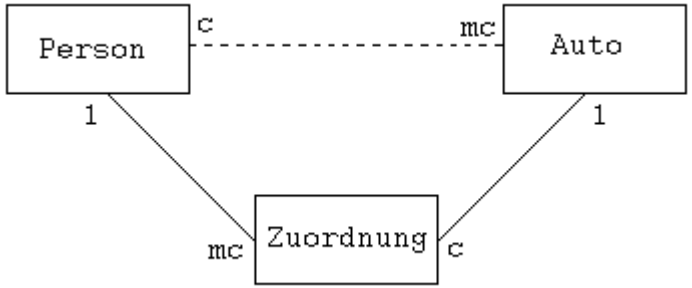
- Interpretation: Jede Person kann beliebig viele Autos besitzen, und jedes Auto hat entweder einen oder keinen Besitzer (d.h. kann aber muss keinen Besitzer haben).
- Auflösung

  Person (Pers#, VN, NN)

  Auto (Auto#, Marke, Bezeichnung, Baujahr, Pers#)
- Auch hier gilt wieder: Diese Lösung führt zu Nullwerten im FK => Transformationstabelle

Die c:mc Beziehung

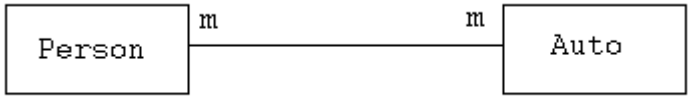
Transformation



Person (P[illegible])
Auto (Auto#, Marke, Bezeichnung, Baujahr)
Zuordnung (Pers#, Auto#)

- Im FK „Auto#“ der Zuordnungstabelle darf jeder Attributwert nur einmal vorkommen (**UNIQUE**), während im FK „Pers#“ derselbe Werte öfters vorkommen darf.
- Die c:mc Beziehung lässt sich demnach nach einer Transformation auf Tabellendefinitionsebene realisieren.

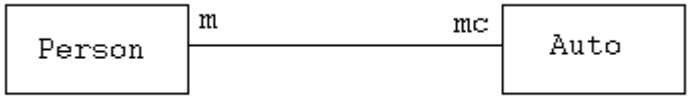
Die m:m Beziehung



- Interpretation: Jede Person verfügt über mindestens ein Auto, und jedes Auto ist mindestens einer Person zugeordnet.
- => Im Falle der m:m Beziehung muss auf jeden Fall eine „Transformationstabelle“ verwendet werden.

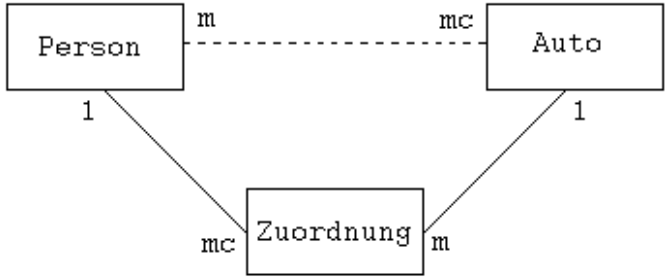
  Person (Pers#, VN, NN)
  Auto (Auto#, Marke, Bezeichnung, Baujahr)
  Zuordnung (Pers#, Auto#)
- Durch diese Tabellendefintion kann aber die Konsistenzbedingung dass jede Person über mindesten ein Auto verfügt, und jedes Auto mindestens einer Person zugeordnet ist nicht sichergestellt werden => Programmierung!

Die m:mc Beziehung



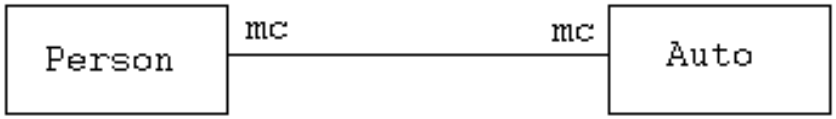
- Interpretation: Jede Person kann über mehrere Autos verfügen (=>0) und jedes Auto ist mindestens einem Verfügungsberechtigten zugeordnet (>=1).

  Person (Pers#, VN, NN)
  Auto (Auto#, Marke, Bezeichnung, Baujahr)
  Zuordnung (Pers#, Auto#)



- Auch hier gilt wieder, dass die Konsistenzbedingung der 1:m Beziehung zwischen Auto und Zuordnung auf Tabellendefinitionsebene nicht gewährleistet werden kann => Programmierung! Die Konsistenzbedingung der 1:mc Beziehung wird auf Tabellendefinitionsebene hingegen berücksichtigt.

Die mc:mc Beziehung

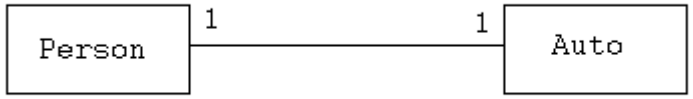


- Interpretation: Jede Person kann über mehrere Autos verfügen (>=0) und jedes Auto kann mehreren Personen gleichzeitig zugeordnet sein (>=0)

- Transformation:
- Auch hier ist wieder eine „Transformationstabelle“ **zwingend** erforderlich. (In den Fällen c:c und c:mc könnte darauf verzichtet werden, wenn Nullwerte im FK zugelassen bzw. in Kauf genommen werden.)

  Person (Pers#, VN, NN)
  Auto (Auto#, Marke, Bezeichnung, Baujahr)
  Zuordnung (Pers#, Auto#)
- Die Konsistenzbedingung der mc:mc Bedingung kann auf Tabellendefintionsebene realisiert werden.

Die 1:1 Beziehung 1/2



- Interpretation: Jede Person besitzt genau ein Auto, und jedes Auto gehört genau einer Person.
- Transformation:
  - In eine der beiden Tabelle wird der PK der anderen Tabelle als FK übernommen.

    Person (Pers#, VN, NN)

    Auto (Auto#, Marke, Bezeichnung, Baujahr, Pers#)
  - oder

    Person (Pers#, VN, NN, Auto#)

    Auto (Auto#, Marke, Bezeichnung, Baujahr)

Die 1:1 Beziehung 2/2

- In der Definition der Tabelle in SQL kann das FK-Attribut „Auto#“ noch zusätzlich die Eigenschaft „UNIQUE“ erhalten.
- Diese Lösung ist ident mit der Transformation der 1:c Beziehung. Auf Ebene der Tabellendefinition sind **keine** weiteren Konsistenzbedingungen möglich => Programmierung

› ODER: die beiden Tabellen werden zu einer Tabelle zusammengefasst.

- In diesem Fall wird einer der beiden PK der PK der neuen zusammengefassten Tabelle. Dies ist aber nur dann erlaubt, wenn der „wegfallende“ PK nicht als FK in einer anderen Tabelle verwendet wird.

Übe

Auf Tabellendefintionsebene sind darstellbar

Ohne Nullwerte im FK

1:c	Person (Pers#, VN, NN) Auto (Auto#, Marke, Bezeichnung, Baujahr, Pers#)	FK UNIQUE
1:mc	Person (Pers#, VN, NN) Auto (Auto#, Marke, Bezeichnung, Baujahr, Pers#)	

Mit Nullwerten im FK

c:c	Person (Pers#, VN, NN, Auto#) Auto (Auto#, Marke, Bezeichnung, Baujahr, Pers#)	Beide FK UNIQUE
c:mc	Person (Pers#, VN, NN) Auto (Auto#, Marke, Bezeichnung, Baujahr, Pers#)	

Mit einer Transformationstabelle (ohne Nullwerte im FK)

c:c	Person (Pers#, VN, NN) Auto (Auto#, Marke, Bezeichnung, Baujahr) Zuordnung (Pers#, Auto#)	Beide FK UNIQUE
c:mc	Person (Pers#, VN, NN) Auto (Auto#, Marke, Bezeichnung, Baujahr) Zuordnung (Pers#, Auto#)	FK Auto# UNIQUE
mc:mc	Person (Pers#, VN, NN) Auto (Auto#, Marke, Bezeichnung, Baujahr) Zuordnung (Pers#, Auto#)	*Anmerkung: Transformationstabelle ist **zwingend***

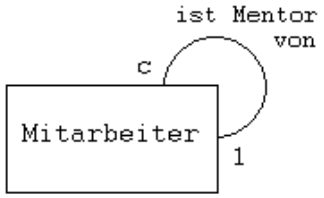
Konsistenzbedingungen nicht auf Tabellendefintionsebene lösbar => Programmierung

1:m	Person (Pers#, VN, NN) Auto (Auto#, Marke, Bezeichnung, Baujahr, Pers#)	Programmierung
1:1	Person (Pers#, VN, NN) Auto (Auto#, Marke, Bezeichnung, Baujahr, Pers#)	FK UNIQUE

Zusätzlich eine Transformationstabelle erforderlich machen

m:mc	Person (Pers#, VN, NN) Auto (Auto#, Marke, Bezeichnung, Baujahr) Zuordnung (Pers#, Auto#)	Programmierung
m:m	Person (Pers#, VN, NN) Auto (Auto#, Marke, Bezeichnung, Baujahr) Zuordnung (Pers#, Auto#)	Programmierung
c:m	Person (Pers#, VN, NN) Auto (Auto#, Marke, Bezeichnung, Baujahr) Zuordnung (Pers#, Auto#)	Programmierung

Rekursive Strukturen



- **Rekursiv c:c**

  Mitarbeiter (Pers#, Vorname, Nachname, Partner#)

  - Partner darf **Null** sein, **Unique**
  - Anmerkung: Partner wäre ein FK aus derselben Tabelle, das können aber manche Datenbanken nicht
- **Rekursiv 1:c**

  Mitarbeiter (Pers#, Vorname, Nachname, Mentor#)

  - Mentor= **NOT NULL**; **Unique**
- **Rekursiv 1:n**

  Mitarbeiter (Pers#, Vorname, Nachname, Vorgesetzter#)

  - Vorgesetzter .... **Nullwert** nicht erlaubt, **kein Unique**

Anmerkung: rot eingefärbt bedeutet Fremdschlüssel

Falls DB keine FK innerhalb einer Tabelle erlaubt …

Mitarbeiter (Pers#, Vorname, Nachname)

Vorgesetzter (Pers#, Knecht#)

- Knecht ist Teil PK, ein FK, und Unique !

Mitarbeiter (Pers#, Vorname, Nachname)

Mentor (Pers#, Schützling)

- (Schützling ist FK und UNIQUE)
- *Anm.: In der Praxis (d.h. betriebwirtschaftlichen Anwendungssystemen haben diese rekursiven Beziehungen aber keine allzu große Bedeutung)*

Rekursive Strukturen

- Rekursiv n:m

  Teil (*Teil#,* Bezeichnung)

  Teilstruktur (Teil#, Komponente#, Menge)

  Kurs (Kursbez, ECTS, Institut)

  Vorraussetzung (Kursbez, Vorkursbez)

  - Anm.: Diese rekursive Beziehung **ist** in der Praxis relevant!