3. Grundlagen des Relationalen Datenmodells

- Grundkonzepte
- Relationale Invarianten
  - Primärschlüsselbedingung, Fremdschlüsselbedingung (referentielle Integrität)
  - Wartung der referentiellen Integrität
- Abbildung ERM / UML → RM
- Nachbildung von Generalisierung und Aggregation im RM

Kapitel 4: **Relationenalgebra**

Kapitel 5: Standard-Anfragesprache SQL

Kapitel 6: **Logischer DB-Entwurf (Normalformenlehre)**

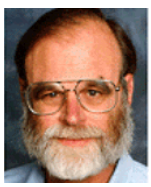
Kapitel 7/8: **Datendefinition und -kontrolle**

DB-Anwendungsprogrammierung: ->DBS2

Relationenmodell - Entwicklung

- 1969/1970: Vorschlag von Edgar F. Codd (IBM)
  - ***A Relational Model of Data for Large Shared Data Banks***. Commun. ACM 13(6): 377-387 (1970)
  - Konzept, Relationenalgebra

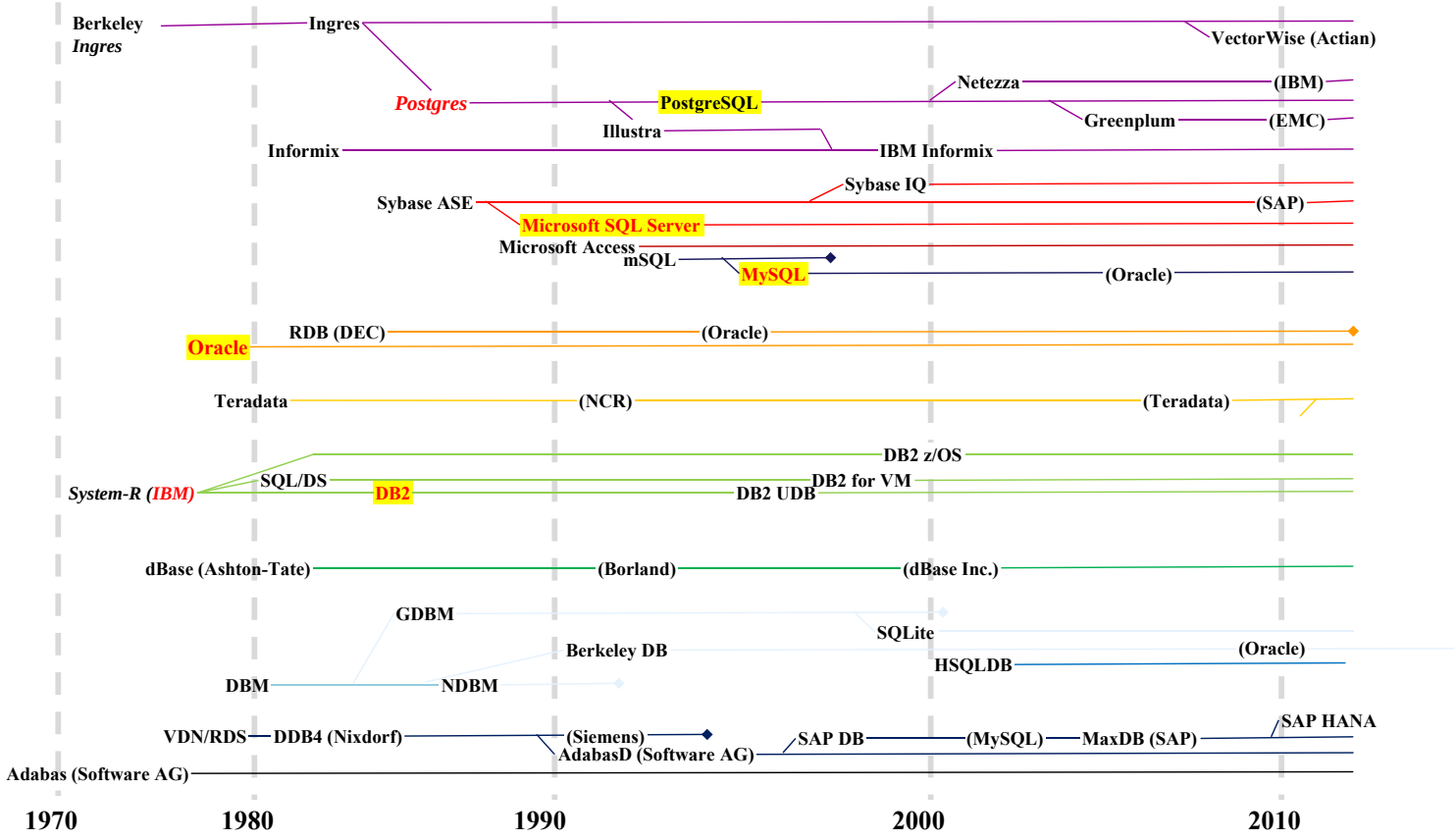
- ab ca.1975: erste Prototypen relationaler DBS
  - System R (IBM Research, San Jose), u.a. Query-Sprache SEQUEL / SQL (D. Chamberlin), ACID-Techniken (Jim Gray et al.), Query-Optimierung etc.



  - Ingres (Berkeley Univ.) unter Leitung von Mike Stonebraker, Query-Sprache QUEL

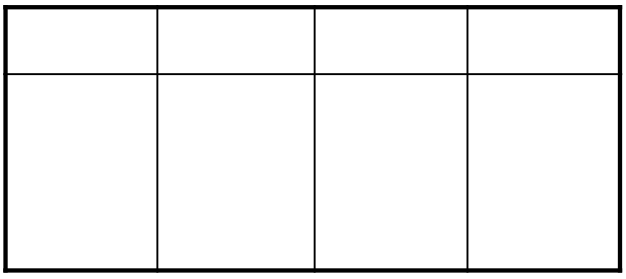
- seit ca. 1980: kommerzielle relationale DBS
  - zunächst Oracle (Larry Ellison): 1979 „Version 2“
  - IBM DB2 …

Genealogie ausgewählter relationaler DBS



Quelle: Hasso Plattner Institut, Potsdam.

Relationenmodell - Übersicht



- Datenstruktur: Relation (Tabelle)
  - einzige Datenstruktur (neben atomaren Werten)
  - alle Informationen ausschließlich durch Werte dargestellt
  - Integritätsbedingungen auf/zwischen Relationen: *relationale Invarianten*
- Operatoren auf (mehreren) Relationen
  - Vereinigung, Differenz
  - kartesisches Produkt
  - Projektion
  - Selektion
  - zusätzlich: Änderungsoperationen (Einfügen, Löschen, Ändern)

Relationenmodell - Grundkonzepte

Attribut
Definitionsbereich
Primärschlüssel

} wie im ERM

Entity-Menge (Klassen)

Relationship-Menge (Assoziationen)

→ Relation

- normalisierte Relation

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

D_i D_j D_k

  - Relation = Untermenge des kartesischen Produktes der Attributwertebereiche
  - nur einfache Attibute (atomare Werte) !
- Darstellungsmöglichkeit für R:
  n-spaltige Tabelle (*Grad* der Relation: n)
  - *Kardinalität:* Anzahl der Sätze (Tupel)
- Relation ist eine Menge: Garantie der Eindeutigkeit der Zeilen/Tupel über Primärschlüssel (ggf. mehrere Schlüsselkandidaten)

Normalisierte Relationen in Tabellendarstellung

FAK

FNR	FNAME	...
WI	Wirtschaftswiss.	...
MI	Math./Informatik	...

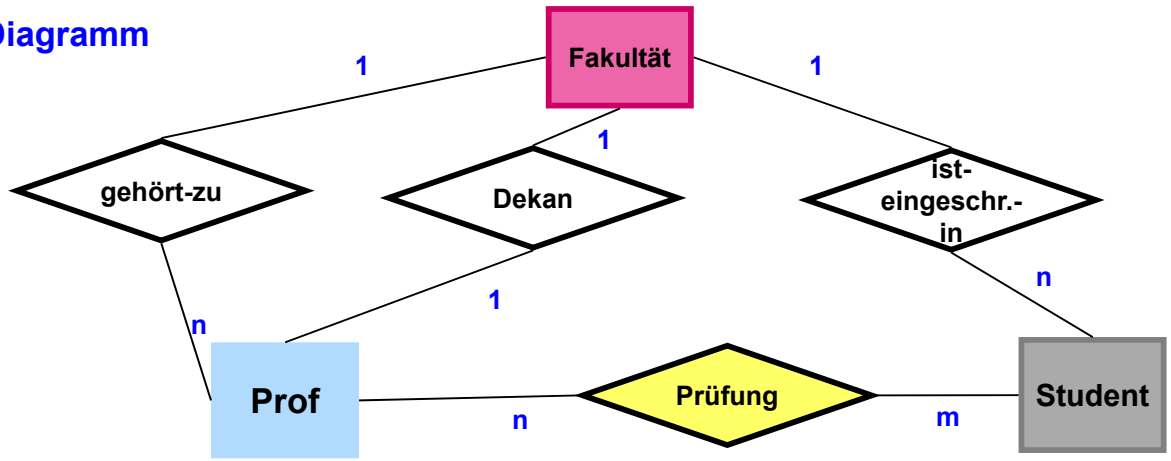
STUDENT

MATNR	SNAME	FNR	W-ORT
123 766	Coy	MI	Halle
654 711	Abel	WI	Leipzig
196 481	Maier	MI	Delitzsch
226 302	Schulz	MI	Leipzig

- Grundregeln:
  - jede Zeile (Tupel) ist eindeutig und beschreibt ein Objekt (Entity) der Miniwelt
  - Reihenfolge der Zeilen ist ohne Bedeutung
  - Reihenfolge der Spalten ist ohne Bedeutung, da sie eindeutigen (Attribut-) Namen tragen
  - jeder Datenwert innerhalb einer Relation ist ein atomares Datenelement
  - alle für Benutzer relevanten Informationen sind ausschließlich durch Datenwerte ausgedrückt
- Darstellung von Beziehungen durch *Fremdschlüssel (foreign key)*
  - Attribut, das in Bezug auf den Primärschlüssel einer anderen (oder derselben) Relation definiert ist (gleicher Definitionsbereich)

Anwendungsbeispiel

ER-Diagramm



Relationales Schema

FAK

FNR	FNAME	DEKAN

STUDENT

MATNR	SNAME	FNR	W-Ort

PROF

PNR	PNAME	FNR	FACHGEB

PRÜFUNG

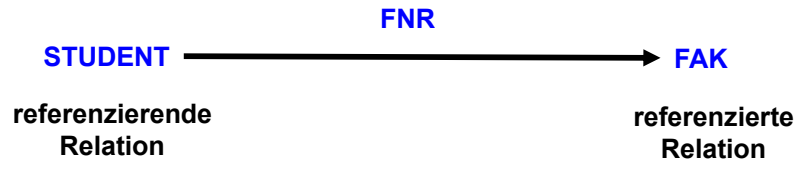
PRID	PNR	MATNR	FACH	DATUM	NOTE

Relationale Invarianten

- inhärente Integritätsbedingungen des Relationenmodells (Modellbedingungen)

1. Primärschlüsselbedingung (Entity-Integrität)
   - Eindeutigkeit des Primärschlüssels / Minimalität
   - keine Nullwerte!
2. Fremdschlüsselbedingung (referentielle Integrität):
   - zugehöriger Primärschlüssel muss existieren
   - d.h. zu jedem Wert (ungleich Null) eines Fremdschlüsselattributs einer Relation R2 muss ein gleicher Wert des Primärschlüssels in irgendeinem Tupel von Relation R1 vorhanden sein

- graphische Notation:



Relationale Invarianten (2)

- Fremdschlüssel und zugehöriger Primärschlüssel tragen wichtige interrelationale (bzw. intrarelationale) Informationen
  - gleicher Wertebereich
  - gestatten Verknüpfung von Relationen
- Fremdschlüssel
  - können Nullwerte aufweisen, wenn sie nicht Teil eines Primärschlüssels sind.
  - Fremdschlüssel ist „zusammengesetzt“, wenn zugehöriger Primärschlüssel „zusammengesetzt“ ist
- eine Relation kann mehrere Fremdschlüssel besitzen, die die gleiche oder verschiedene Relationen referenzieren
- Zyklen sind möglich (***geschlossener referentieller Pfad***)
- eine Relation kann zugleich referenzierende und referenzierte Relation sein (selbstreferenzierende Tabelle)

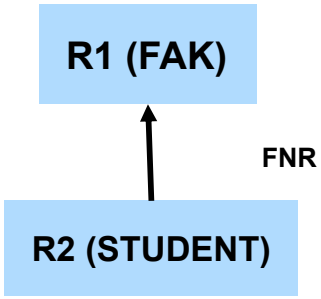
Relationale Invarianten (3)

- DDL-Spezifikation in SQL bei CREATE TABLE
  - Primary KEY- und FOREIGN KEY-Klauseln
  - Angaben für Attributdefinition :
    - Attributname sowie Datentyp
    - Default-Werte
    - Eindeutigkeit (UNIQUE bzw. PRIMARY KEY)
    - FOREIGN-KEY-Klausel
    - Verbot von Nullwerten (NOT NULL)

```
CREATE  TABLE STUDENT
     (MATNR    INT,
      SNAME    VARCHAR (50)   NOT NULL,
      FNR      INT,
      PRIMARY KEY (MATNR),
      FOREIGN KEY (FNR) REFERENCES FAK )

CREATE  TABLE FAK
    (FNR    INT     PRIMARY KEY,
     FNAME  VARCHAR(50) NOT NULL,
     DEKAN  INT     REFERENCES PROF ... )
```

Wartung der referentiellen Integrität

- Gefährdung bei INSERT, UPDATE, DELETE



- **Fall 0:** INSERT auf R1, DELETE auf R2
  - keine Auswirkungen für referentielle Integrität
- **Fall 1:** INSERT in der referenzierenden (abhängigen) Relation R2 bzw. UPDATE auf Fremdschlüssel in R2
  - Ablehnung falls kein zugehöriger Primärschlüssel-Wert in referenzierter Relation R1 besteht
- **Fall 2:** DELETE auf referenzierter Relation R1 bzw. UPDATE von Primärschlüssel von R1
  - unterschiedliche Folgeaktionen auf referenzierender Relation R2 möglich, um referentielle Integrität zu wahren

Wartung der referentiellen Integrität (2)

- SQL-Standard erlaubt Spezifikation der referentiellen Aktionen für jeden Fremdschlüssel
- sind Nullwerte verboten?
  - **NOT NULL**
- Löschregel für Zielrelation (referenzierte Relation R1):

  ON DELETE {NO ACTION | CASCADE | SET NULL | SET DEFAULT }
- Änderungsregel für Ziel-Primärschlüssel (Primärschlüssel oder Schlüsselkandidat):

  ON UPDATE {NO ACTION | CASCADE | SET NULL | SET DEFAULT}
- dabei bedeuten:
  - **NO ACTION** (Voreinstellung): Operation wird nur zugelassen, wenn keine zugehörigen Sätze (Fremdschlüsselwerte) vorhanden sind. Es sind folglich keine referentiellen Aktionen auszuführen
  - **CASCADE**: Operation „kaskadiert“ zu allen zugehörigen Sätzen
  - **SET NULL**: Fremdschlüssel wird in zugehörigen Sätzen zu “Null” gesetzt
  - **SET DEFAULT**: Fremdschlüssel wird auf einen benutzerdefinierten Default-Wert gesetzt

Anwendungsbeispiel

```
CREATE TABLE    TEIL (    TNR  INT  PRIMARY KEY,
                          BEZEICHNUNG . . .  )

CREATE TABLE    LIEFERANT (LNR INT  PRIMARY KEY,
                           LNAME  . . . )

CREATE TABLE    LIEFERUNG (TNR INT, LNR INT, DATUM ...
                PRIMARY KEY (TNR, LNR, DATUM),
                FOREIGN KEY (TNR)      REFERENCES TEIL,  NOT NULL,
                                ON DELETE OF TEIL NO ACTION
                                ON UPDATE OF TEIL.TNR CASCADE,
                FOREIGN KEY (LNR)      REFERENCES LIEFERANT, NOT NULL,
                                ON DELETE OF LIEFERANT NO ACTION,
                                ON UPDATE OF LIEFERANT.LNR CASCADE )
```

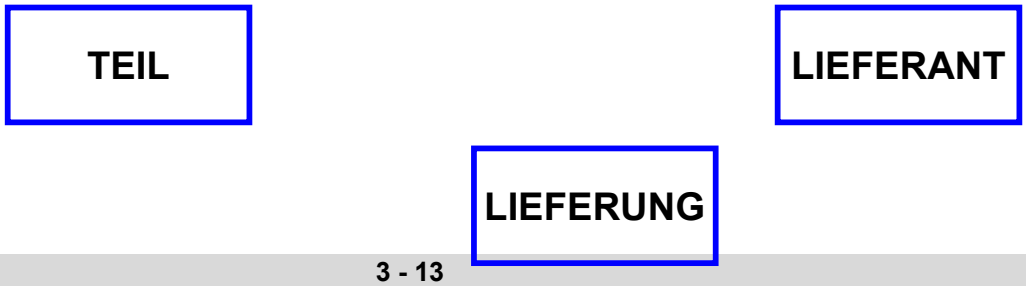
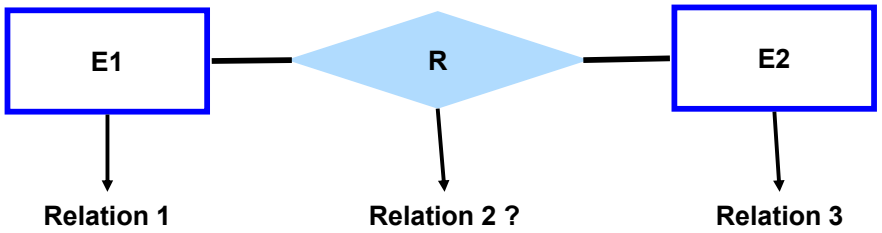


Abbildung ERM / UML -> RM



- Kriterien
  - Informationserhaltung
  - Minimierung der Redundanz
  - Minimierung des Verknüpfungsaufwandes
  - Natürlichkeit der Abbildung
  - keine Vermischung von Objekten
  - Verständlichkeit
- Regeln:
  - jede Entity-Menge *muss* als eigenständige Relation (Tabelle) mit einem eindeutigen Primärschlüssel definiert werden.
  - Relationship-Mengen können als eigene Relationen definiert werden, wobei die Primärschlüssel der zugehörigen Entity-Mengen als Fremdschlüssel zu verwenden sind.

2 Entity-Mengen mit n:1 - Verknüpfung

1.) Verwendung von drei Relationen

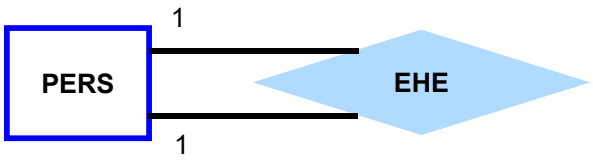
```
ABT   (ANR,   ANAME,   ...)
PERS   (PNR,   PNAME,   ...)
ABT-ZUGEH (PNR,   ANR, )
```

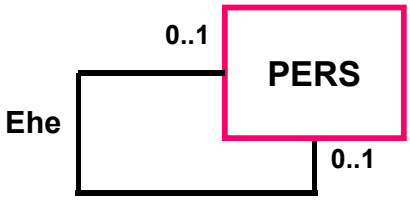
2.) Besser: Verwendung von zwei Relationen

```
ABT   (ANR,   ANAME,   ...)
PERS   (PNR,   PNAME,   ...,   ANR)
```

- Regel: n:1-Beziehungen lassen sich ohne eigene Relation darstellen.
  - dazu wird in der Relation, der pro Tupel maximal 1 Tupel der anderen Relation zugeordnet ist, der Primärschlüssel der referenzierten Relation als Fremdschlüssel verwendet

1 Entity-Menge mit 1:1 Verknüpfung





1.) Verwendung von zwei Relationen

```
PERS   (PNR,   PNAME,   ...)
EHE   ( PNR , GATTE,    ...)
```

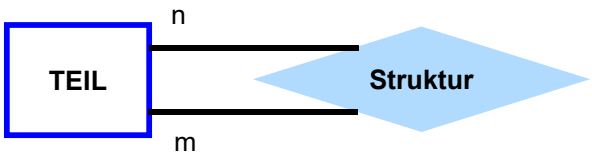
2.) Verwendung von einer Relation

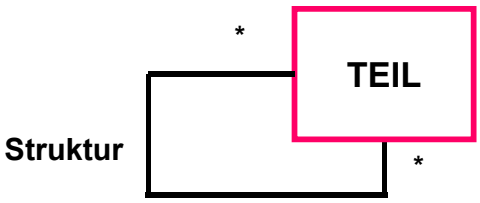
```
PERS   (PNR,   PNAME,   ...,    GATTE)
```

- Unterscheidung zu n:1 ?

1 Entity-Menge mit m:n-Verknüpfung

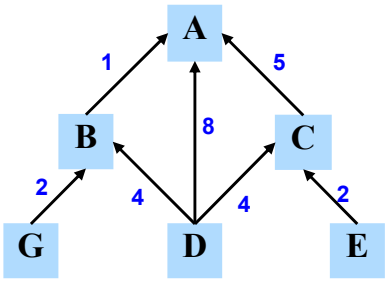
Stückliste





Darstellungsmöglichkeit im RM:

TEIL (TNR, BEZ, MAT, BESTAND)
STRUKTUR (OTNR, UTNR, ANZAHL)



Teil

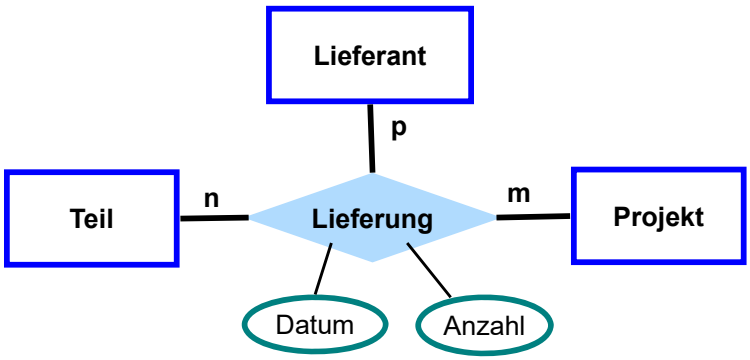
TNR	BEZ	MAT	BESTAND
A	Getriebe	-	10
B	Gehäuse	Alu	0
C	Welle	Stahl	100
D	Schraube	Stahl	200
E	Kugellager	Stahl	50
F	Scheibe	Blei	0
G	Schraube	Chrom	100

Struktur

OTNR	UTNR	Anzahl
A	B	1
A	C	5
A	D	8
B	D	4
B	G	2
C	D	4
C	E	2

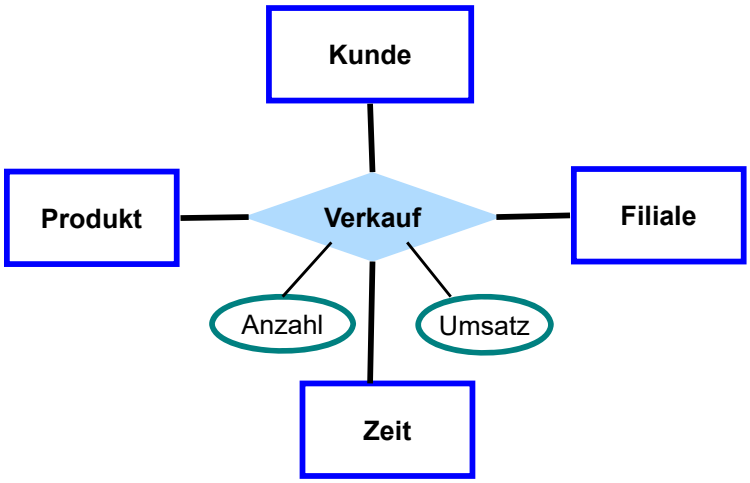
- Regel
  - n:m-Beziehungen müssen durch eigene Relation dargestellt werden.
  - die Primärschlüssel der zugehörigen Entity-Mengen treten als Fremdschlüssel auf.

3 Entity-Mengen mit m:n-Verknüpfung



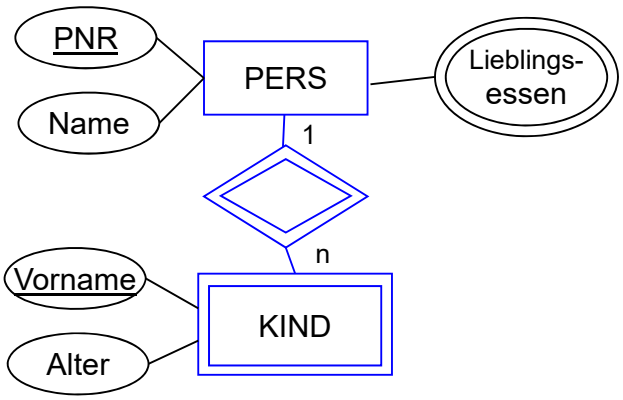

```
LIEFERANT     (LNR,    LNAME,   LORT, ...)
PROJEKT       (PRONR,  PRONAME, PORT,  ...)
TEIL          (TNR,    TBEZ,    GEWICHT,  ... )

LIEFERUNG     (
```

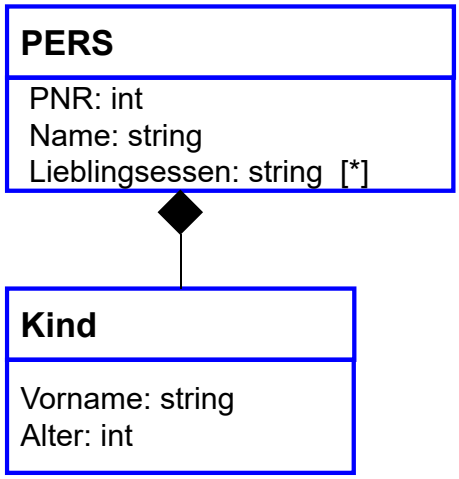
M Entity-Mengen (Stern-Schema)




```
KUNDE   (KNR,  KNAME,  Geschlecht, ...)
PRODUKT (PNR,  PNAME, PTyp,  ...)
FILIALE (FNR,  Ort,  Land,  ... )
ZEIT    (ZNR,  Tag,  Monat, Jahr, Quartal ...)
```


Abbildung mehrwertiger Attribute bzw. schwacher Entity-Mengen



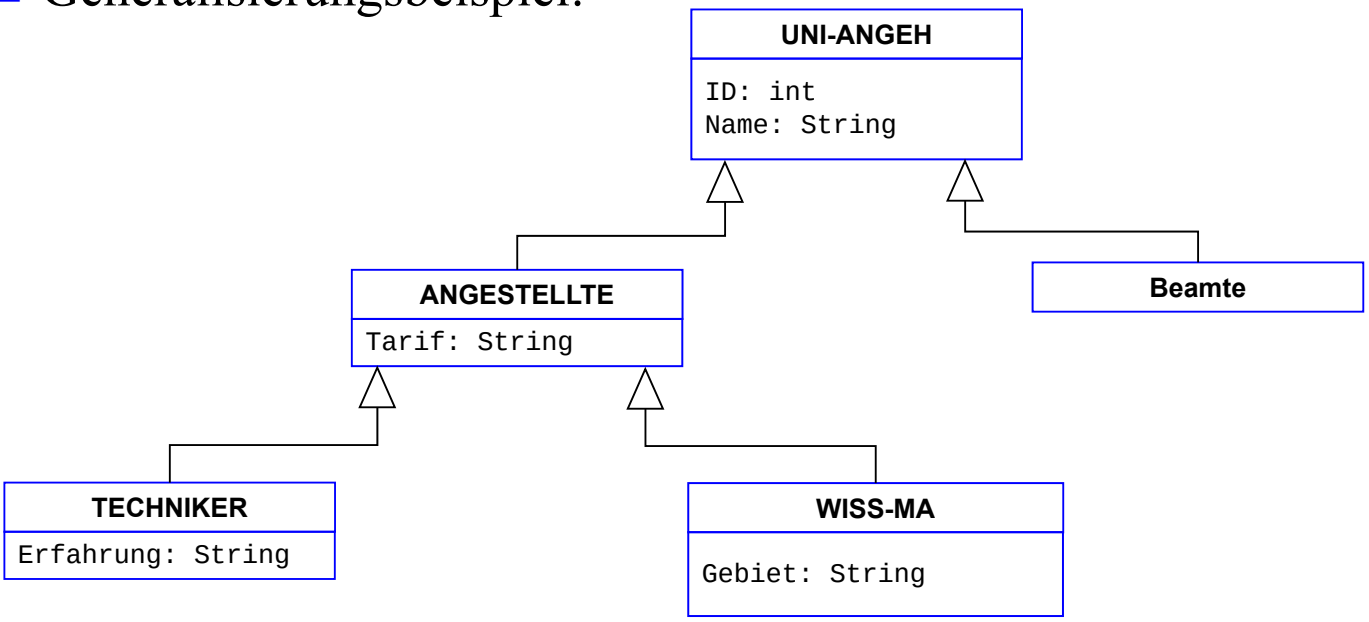


Darstellungsmöglichkeit im RM

```
PERS        (PNR,  NAME  ...)
```

Abbildungen von Generalisierung und Aggregation im RM

- RM sieht keine Unterstützung der Abstraktionskonzepte vor
  - keine Maßnahmen zur Vererbung (von Struktur, Integritätsbedingungen, Operationen)
  - „Simulation“ der Generalisierung und Aggregation eingeschränkt möglich
- Generalisierungsbeispiel:



Generalisierung – relationale Sicht

- pro Klasse 1 Tabelle (3 Varianten) oder insgesamt nur 1 Tabelle
- Lösungsmöglichkeit 1: vertikale Partitionierung
  - jede Instanz wird entsprechend der Klassenattribute in der IS-A-Hierarchie zerlegt und in Teilen in den zugehörigen Klassen (Relationen) gespeichert.
  - nur das ID-Attribut wird dupliziert

UNI-ANGEH

ID	Name
007	Garfield
123	Donald
333	Daisy
765	Grouch
111	Ernie

ANGESTELLTE

ID	Tarif
007	E14
123	E11
333	E9
765	E13

WISS-MA

ID	Gebiet
007	XML
765	Cloud

TECHNIKER

ID	Erfahrung
123	Linux

- Eigenschaften
  - geringfügig erhöhte Speicherungskosten, aber hohe Aufsuch- und Aktualisierungkosten
  - Integritätsbedingungen: *TECHNIKER.ID* $\subseteq$ *ANGESTELLTE.ID*, usw.
  - Instanzenzugriff erfordert implizite oder explizite Verbundoperationen
  - Beispiel: Finde alle TECHNIKER-Daten

Generalisierung – relationale Sicht (2)

- Lösungsmöglichkeit 2: horizontale Partitionierung
  - jede Instanz ist genau einmal und vollständig in ihrer „Hausklasse“ gespeichert.
  - keinerlei Redundanz

UNI-ANGEH

ID	Name
111	Ernie

WISS-MA

ID	Gebiet	Name	Tarif
007	XML	Garfield	E14
765	Cloud	Grouch	E13

ANGESTELLTE

ID	Name	Tarif
333	Daisy	E9

TECHNIKER

ID	Erfahrung	Name	Tarif
123	Linux	Donald	E11

- Eigenschaften
  - niedrige Speicherungskosten und keine Änderungsanomalien
  - Eindeutigkeit von ID zwischen Relationen aufwändiger zu überwachen
  - Retrieval kann rekursives Suchen in Unterklassen erfordern.
  - explizite Rekonstruktion durch Relationenoperationen (π, $\cup$ in Relationenalgbra)

  => Beispiel: Finde alle ANGESTELLTE

Generalisierung – relationale Sicht (3)

- Lösungsmöglichkeit 3: volle Redundanz
  - eine Instanz wird wiederholt in jeder Klasse, zu der sie gehört, gespeichert.
  - sie besitzt dabei die Werte der Attribute, die sie geerbt hat, zusammen mit den Werten der Attribute der Klasse

UNI-ANGEH

ID	Name
007	Garfield
123	Donald
333	Daisy
765	Grouch
111	Ernie

ANGESTELLTE

ID	Name	Tarif
007	Garfield	E14
123	Donald	E11
333	Daisy	E9
765	Grouch	E13

WISS-MA

ID	Name	Tarif	Gebiet
007	Garfield	E14	XML
765	Grouch	E13	Cloud

TECHNIKER

ID	Name	Tarif	Erfahrung
123	Donald	E11	Linux

- Eigenschaften
  - hoher Speicherplatzbedarf und Auftreten von Änderungsanomalien.
  - einfaches Retrieval, da nur Zielklasse (z. B. ANGESTELLTE) aufzusuchen

Generalisierung: Verfahrensvergleich

	Vertikale Partitionierung	Horizontale Partitionierung	Volle Redundanz
Änderungen			
Lesen			

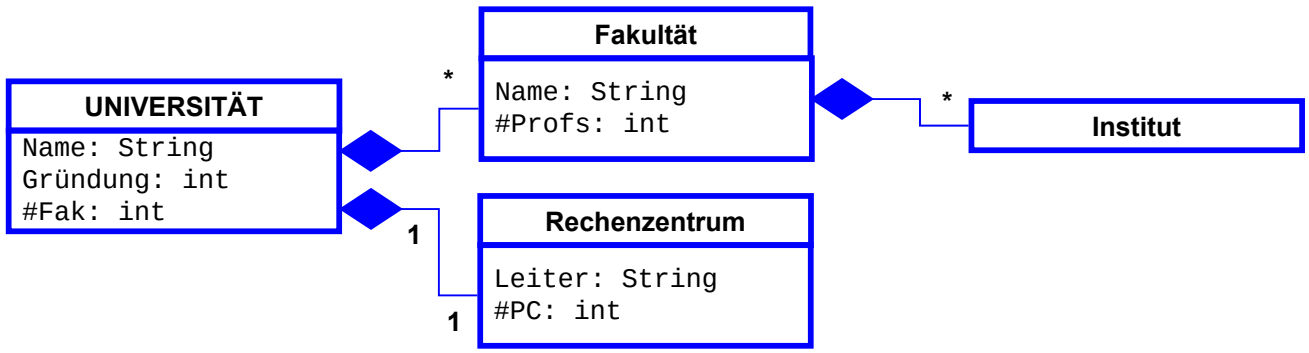
„Wide Table“-Realisierung

- Verwendung einer gemeinsamen Tabelle pro Klassenhierarchie
  - Verwendung eines *Diskrimatorattributs* zur Angabe der Klasse
- minimale Redundanz ermöglicht effiziente Verwaltung
- keine explizite Modellierung der Klassenhierarchie und Is-A-Semantik
  - Nutzer müssen Abhängigkeiten bei Lese- und Änderungsoperationen selbst beachten

UNI-ANGEH

ID	Name	Typ	Tarif	Gebiet	Erfahrung	…
007	Garfield	WISS-MA	E14	XML	*NULL*	
123	Donald	TECHNIKER	E11	*NULL*	Linux	
333	Daisy	ANGESTELLTE	E9	*NULL*	*NULL*	
765	Grouch	WISS-MA	E13	Cloud	*NULL*	
111	Ernie	UNI-ANGEH	*NULL*	*NULL*	*NULL*	

Aggregation – relationale Sicht



Universität

ID	Name	Gründung	#Fak
UL	Univ. Leipzig	1409	14
TUD	TU Dresden	1828	14

Fakultät

FID	Uni	Name	#Profs
123	UL	Theologie	14
132	UL	Mathe/Informatik	28

Institut

ID	FID	Name
1234	123	Neutestamentliche Wissenschaft
1235	123	Alttestamentliche Wissenschaft
1236	123	Praktische Theologie
1322	132	Informatik

- *komplexe Objekte* erfordern Zerlegung über mehrere Tabellen

Zusammenfassung

- RM: einheitliche Datenrepräsentation durch normalisierte Relationen (Tabellen)
  - nur einfache (atomare) Attributwerte
  - eindeutiger Primärschlüssel pro Relation
  - Realisierung von Beziehungen über Fremdschlüssel
- Wartung der referentiellen Integrität
  - automatisch durch DBMS auf Basis von Nutzerspezifikation für Löschungen / PK-Updates referenzierter Sätze
- Abbildung ERM ins Relationenmodell
  - eigene Tabellen für n:m-Beziehungen sowie mehrwertige Attribute
- Nachbildung von Is-A-Beziehungen und Aggregation
  - nur teilweise mit Fremdschlüsseln realisierbar (keine Vererbungssemantik)
  - mehrere Varianten für Generalisierung mit eigener Relation pro (Sub-)Klasse sowie Partitionierung bzw. Replikation von Attributen
  - Wide Table: implementierungsnahe Verwendung einer Tabelle pro Hierarchie