# 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**

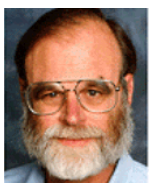
***DB-Anwendungsprogrammierung: ->DBS2***



# 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

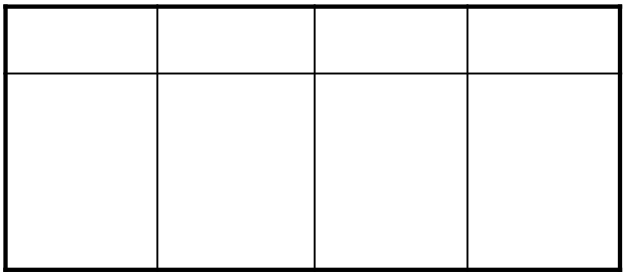
- ab ca.1975: erste Prototypen relationaler DBS
  - System R (IBM Research, San Jose), u.a. Query-Sprache SEQUEL / SQL (D. Chamberlin), ACID-Techniken (Jim Gray et al.), Query-Optimierung etc.



  - 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 …

# 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
Wertebereich
Primärschlüssel
} wie im ERM

Entity-Menge (Klassen) → Relation
**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 Attribute (atomare Werte), also keine Unterstützung für mehrwertige/zusammengesetzte Attribute!

- Darstellungsmöglichkeit für R: n-spaltige Tabelle
  - *Grad* der Relation: n
  - *Kardinalität:* Anzahl der Sätze (Tupel)

- Relation ist Menge
  - Garantie der Eindeutigkeit der Zeilen/Tupel über Primärschlüssel

# Normalisierte Relationen in Tabellendarstellung

FACULTY

| FNO | FNAME | ... |
|---|---|---|
| WI | Wirtschaftswiss. | ... |
| MI | Math./Informatik | ... |

STUDENT

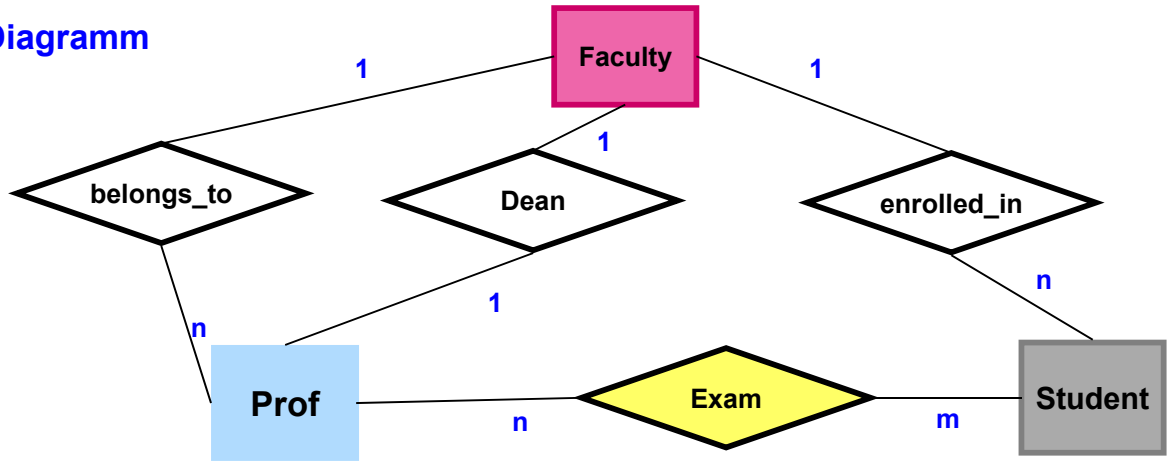
| MATNO | SNAME | FNO | CITY |
|---|---|---|---|
| 123 766 | Coy | MI | Halle |
| 654 711 | Abel | WI | Leipzig |
| 196 481 | Maier | MI | Delitzsch |
| 226 302 | Schulz | MI | Leipzig |

- 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 Wertebereich)



# Anwendungsbeispiel

**ER-Diagramm**




**relationales Schema**

*FACULTY* (FNO, FNAME, DEAN)

*PROF* (PNO, PNAME, FNO, TOPIC)

*STUDENT* (MATNO, SNAME, FNO, CITY)

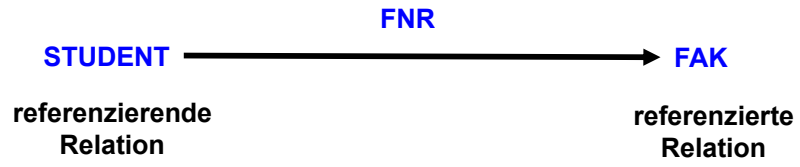
*EXAM* (EID, PNO, MATNO, FACH, DATUM, NOTE)

# 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 R muss ein gleicher Wert des Primärschlüssels in irgendeinem Tupel von Relation S vorhanden sein

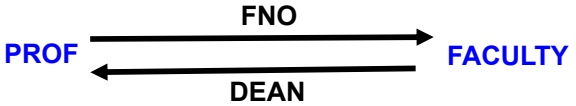
- graphische Notation:






# 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)




PERS (PNO, Name, …, MGR )

# 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)

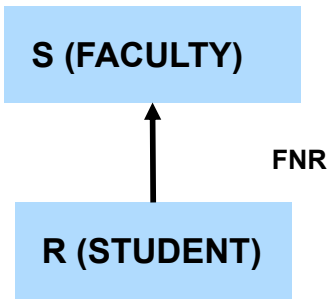
```
CREATE  TABLE STUDENT
      (MATNO    INT,
       SNAME    VARCHAR (50)   NOT NULL,
       FNO      INT,
       PRIMARY KEY (MATNO),
       FOREIGN KEY (FNO) REFERENCES FACULTY )

CREATE  TABLE FACULTY
     (FNO    INT     PRIMARY KEY,
      FNAME  VARCHAR(50) NOT NULL,
      DEAN   INT     REFERENCES PROF ... )
```



# Wartung der referentiellen Integrität

- Gefährdung bei INSERT, UPDATE, DELETE




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

  **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



# Beispiel Wartung referentielle Integrität

**STUDENT**

| MATNO | SNAME | CITY |
|---|---|---|
| 1266 | Coy | Halle |
| 6511 | Abel | Leipzig |
| 1981 | Maier | Delitzsch |
| 2202 | Schulz | Leipzig |

**EXAM**

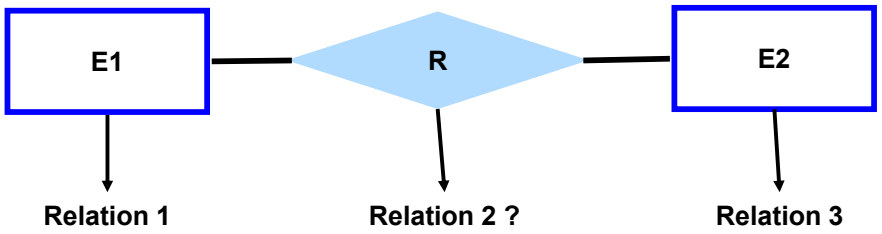
| EID | MATNO | TOPIC | DATE | MARK |
|---|---|---|---|---|
| 1780 | 6511 | FA | 19.9. | 2 |
| 1781 | 1981 | DBS | 15.10. | 1 |
| 1782 | 6511 | DBS | 17.4. | 2 |
| 1783 | 1981 | KI | 25.3. | 3 |

DELETE FROM STUDENT
WHERE MATNO=1981

CREATE TABLE EXAM (…
MATNO INT, NOT NULL, REFERENCES STUDENT
ON DELETE OF STUDENT **<… Action …>**

- Auswirkungen von <Action>
  - NO ACTION:
  - CASCADE:
  - SET NULL:

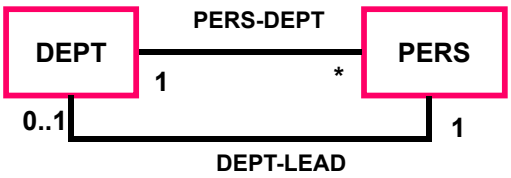
# Abbildung ERM / UML -> RM




- 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

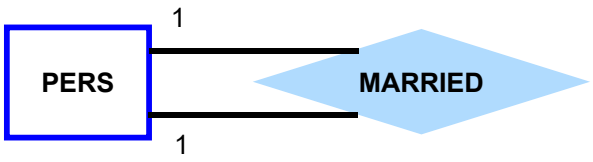
```
DEPT   (DNO,   DNAME,   ...)
PERS   (PNO,   PNAME,   ...)
PERS-DEPT (PNO, ANO, )
```

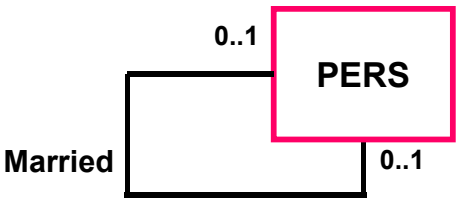
2.) besser: Verwendung von zwei Relationen

```
DEPT   (DNO,   DNAME,   ...)
PERS   (PNO,   PNAME,   ...,   DNO)
```

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

# 1 Entity-Menge mit 1:1 Verknüpfung




1.) Verwendung von zwei Relationen

```
PERS   (PNO,  PNAME,  ...)
MARRIED(PNO, PARTNER,   ...)
```

2.) Verwendung von einer Relation

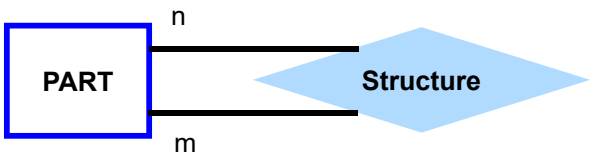
```
PERS   (PNO,  PNAME,  ...,   PARTNER)
          27                    39
```

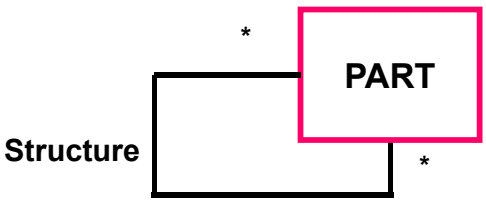
- Unterscheidung zu n:1 ?



# 1 Entity-Menge mit m:n-Verknüpfung

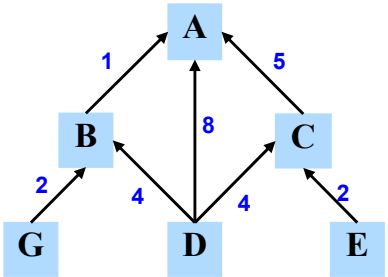
*Stückliste (Bill of material)*




*Darstellungsmöglichkeit im RM:*

**PART** (PNO, NAME, MAT, STOCK)
**STRUCTURE** (TOP, COMP, PCOUNT)




**PART**

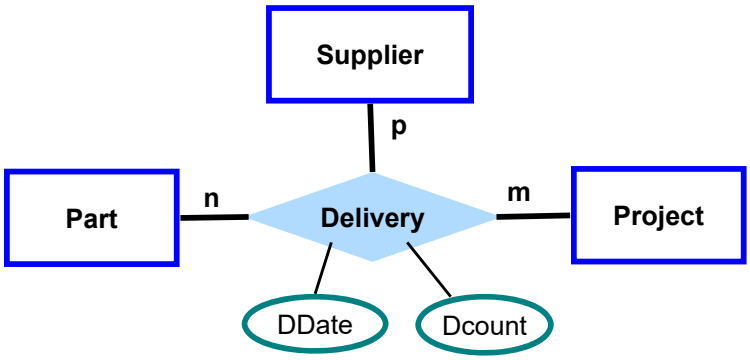
| PNO | NAME | MAT | STOCK |
|---|---|---|---|
| A | Getriebe | - | 10 |
| B | Gehäuse | Alu | 0 |
| C | Welle | Stahl | 100 |
| D | Schraube | Stahl | 200 |
| E | Kugellager | Stahl | 50 |
| F | Scheibe | Blei | 0 |
| G | Schraube | Chrom | 100 |

**STRUCTURE**

| TOP | COMP | PCOUNT |
|---|---|---|
| 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

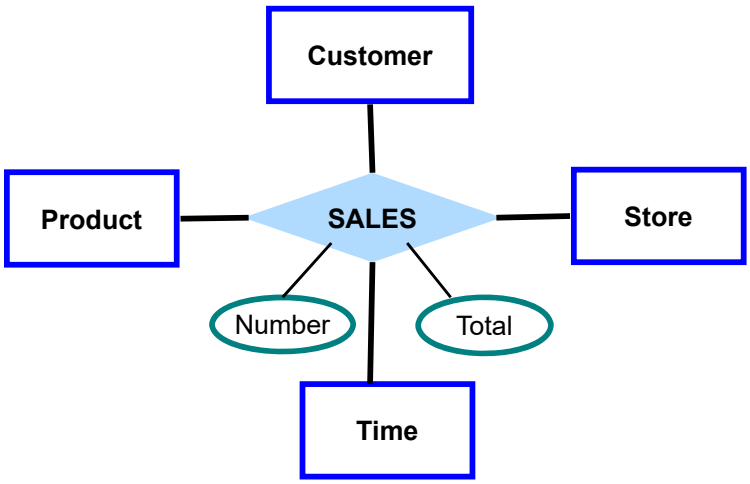



```
SUPPLIER     (SNO,   SNAME,  SCITY, ...)
PROJECT      (PRONO, PRONAME,PCITY,  ...)
PART         (PNO,   PNAME,  WEIGHT,  ... )

DELIVERY     (D-NO, SNO, PRONO, PNO, DDate, Dcount)
```



# M Entity-Mengen (Stern-Schema)

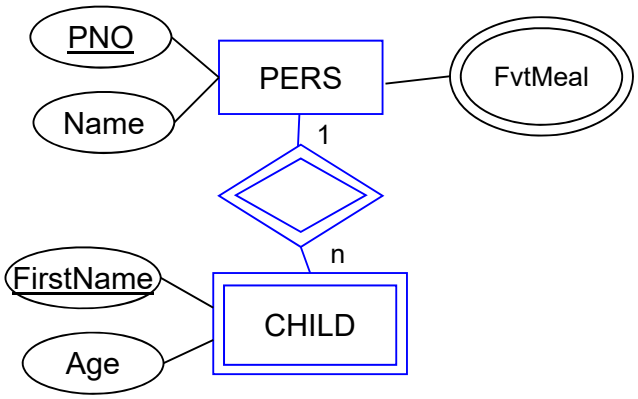


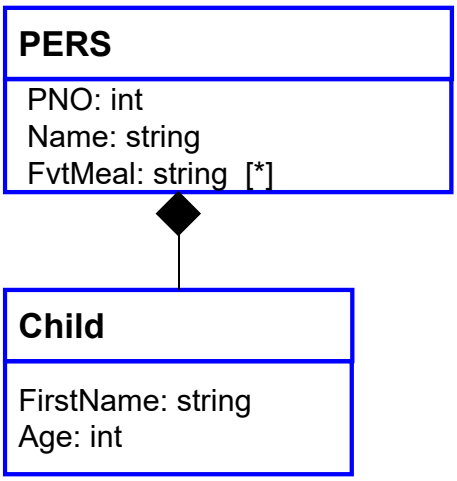

```
CUSTOMER(CNO,  CNAME,  Gender, ...)
PRODUCT (PNO,  PNAME, PType,  ...)
STORE   (STNO,  City,  State,  ... )
TIME    (TNO,  Day,  Month, Year, Quarter ...)
SALES   (SNO, CNO, PNO, STNO, TNO, Number, Total)
```

Typischer Einsatz in Data Warehouses für OLAP

- *Faktentabelle* Verkauf + 4 *Dimensionstabellen*

# Abbildung mehrwertiger Attribute bzw. schwacher Entity-Mengen




**Darstellungsmöglichkeit im RM**

PERS (PNO, NAME ...)

FVTMEAL (PNO, Meal)

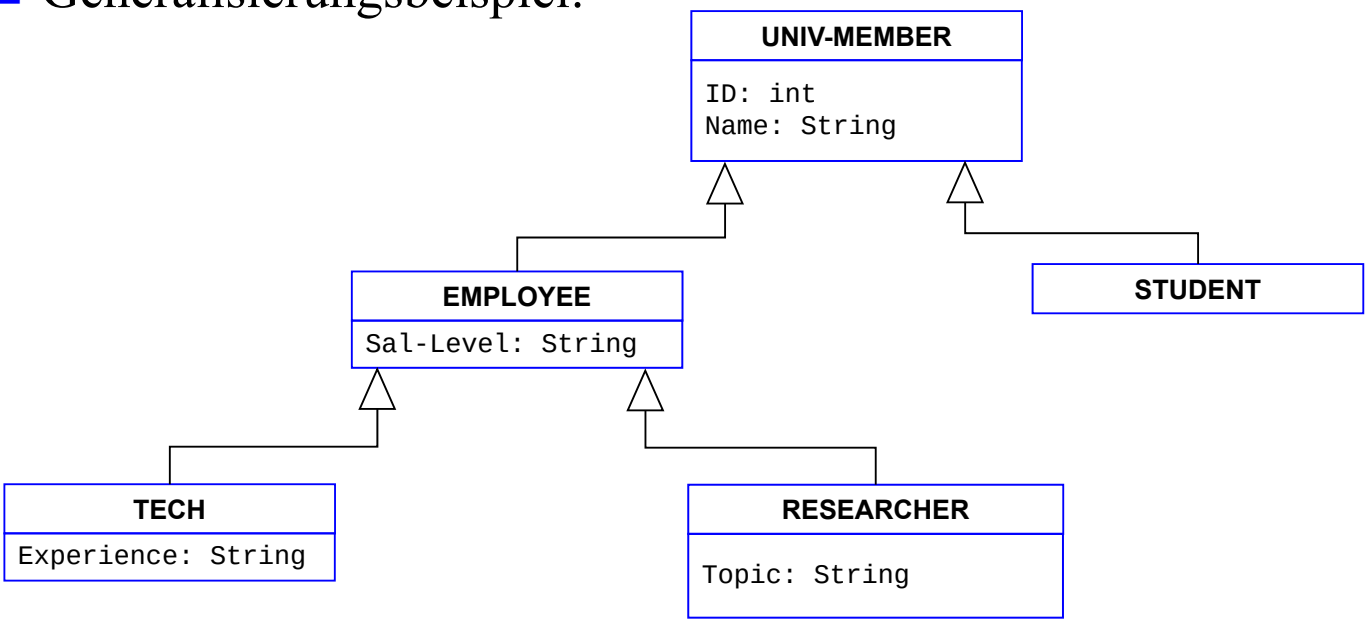
| | |
|---|---|
| 43 | Pizza |
| 43 | Schnitzel |
| 44 | Pizza |

CHILD (PNO, FirstName, Age)



# 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)
- 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

UNIV-MEMBER

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

EMPLOYEE

| ID | Sal |
|---|---|
| 007 | E14 |
| 123 | E11 |
| 333 | E9 |
| 765 | E13 |

RESEARCHER

| ID | TOPIC |
|---|---|
| 007 | NOSQL |
| 765 | Cloud |

TECH

| ID | Experience |
|---|---|
| 123 | Linux |

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



# Generalisierung – relationale Sicht (2)

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

UNIV-MEMBER

| ID | Name |
|---|---|
| 111 | Ernie |

RESEARCHER

| ID | Topic | Name | Sal |
|---|---|---|---|
| 007 | NOSQL | Garfield | E14 |
| 765 | Cloud | Grouch | E13 |

EMPLOYEE

| ID | Name | Sal |
|---|---|---|
| 333 | Daisy | E9 |

TECH

| ID | Experience | Name | Sal |
|---|---|---|---|
| 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 ($\pi$, $\cup$ in Relationenalgbra)

=> Beispiel: Finde alle EMPLOYEE

# 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

**UNIV-MEMBER**

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

**EMPLOYEE**

| ID | Name | Sal |
|---|---|---|
| 007 | Garfield | E14 |
| 123 | Donald | E11 |
| 333 | Daisy | E9 |
| 765 | Grouch | E13 |

**RESEARCHER**

| ID | Name | Sal | Topic |
|---|---|---|---|
| 007 | Garfield | E14 | NOSQL |
| 765 | Grouch | E13 | Cloud |

**TECH**

| ID | Name | Sal | Experience |
|---|---|---|---|
| 123 | Donald | E11 | Linux |

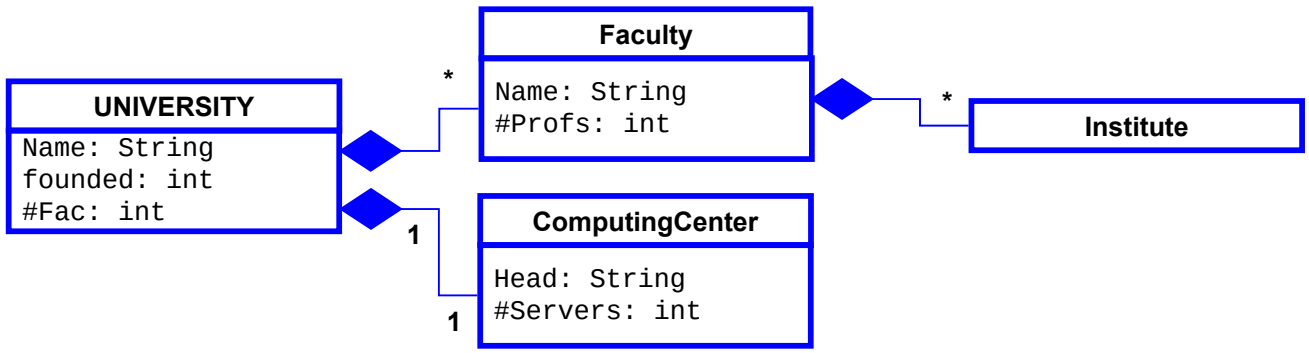
- Eigenschaften
  - hoher Speicherplatzbedarf und Auftreten von Änderungsanomalien.
  - einfaches Retrieval, da nur Zielklasse (z. B. EMPLOYEE) aufzusuchen



# Generalisierung: Verfahrensvergleich

| | vertikale Partitionierung | horizontale Partitionierung | volle Redundanz |
|---|---|---|---|
| **Änderungen** | | | |
| **Lesen** | | | |

# Aggregation – relationale Sicht




University

| ID | Name | founded | #Fac |
|---|---|---|---|
| UL | Leipzig Univ. | 1409 | 14 |
| TUD | TU Dresden | 1828 | 14 |

Faculty

| FID | Uni | Name | #Profs |
|---|---|---|---|
| 123 | UL | Theology | 14 |
| 132 | UL | Math/Comp. Science | 28 |

Institut

| ID | FID | Name |
|---|---|---|
| 1234 | 123 | New Testament Science |
| 1235 | 123 | Old Testament Science |
| 1236 | 123 | Practical Theology |
| 1322 | 132 | Computer Science |

- *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