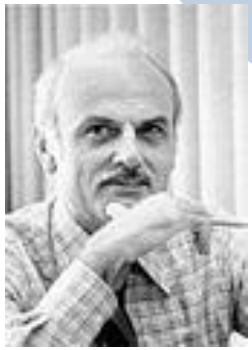
3. GRUNDLAGEN DES RELATIONALEN DATENMODELLS

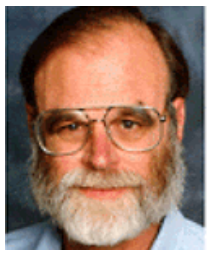
- Grundkonzepte
- Relationale Invarianten
  - Primärschlüsselbedingung, Fremdschlüsselbedingung (referentielle Integrität)
  - Wartung der referentiellen Integrität
- Abbildung ERM / UML → RM
- Nachbildung von Generalisierung und Aggregation im RM
-
- Kapitel 4: Relationenalgebra
- Kapitel 5: Standard-Anfragesprache 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



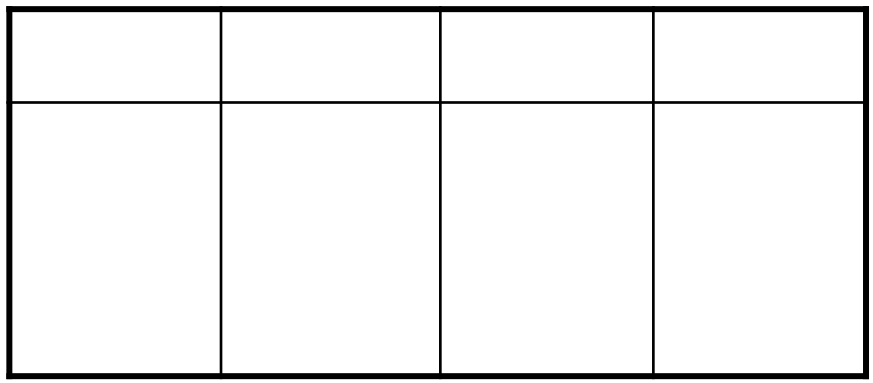


- ab ca.1975: erste Prototypen relationaler DBS
  - System R (IBM Research, San Jose), u.a. Query-Sprache SEQUEL / SQL (D. Chamberlin), ACID-Techniken (Jim Gray et al.), Query-Optimierung etc.
  - Ingres (Berkeley Univ.) unter Leitung von Michael Stonebraker, Query-Sprache QUEL



- 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)
Relationship-Menge
(Assoziationen)
} Relation

- Relationsschema: R (A_1, A_2,…, A_i,..., A_n) Name der Relation und Attributnamen mit jeweiligen Wertebereichen W(A_i)
- Relation: r(R) $\subseteq$ W (A_1) $\times$ W (A_2) $\times$ ... $\times$ W (A_n)
  - Relation = Untermenge des kartesischen Produktes der Attributwertebereiche
  - nur einfache Attribute (atomare Werte), also keine Unterstützung für mehrwertige/zusammengesetzte Attribute!
- Darstellungsmöglichkeit für r(R): n-spaltige Tabelle
  - **Grad** der Relation: n
  - **Kardinalität**: Anzahl der Sätze (Tupel)

NORMALISIERTE RELATIONEN IN TABELLENDARSTELLUNG

- 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 oder Attributkombination, das in Bezug auf den Primärschlüssel einer anderen (oder derselben) Relation definiert ist (gleicher Wertebereich)

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

RELATIONALE INVARIANTEN

- inhärente Integritätsbedingungen des Relationenmodells (Modellbedingungen)

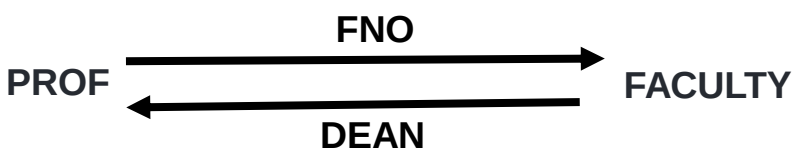
1. **Primärschlüsselbedingung** (Entity-Integrität)
   - Eindeutigkeit des Primärschlüssels / Minimalität
   - keine Nullwerte!
2. **Fremdschlüsselbedingung** (referentielle Integrität):
   - zugehöriger Primärschlüssel muss existieren
   - d.h. zu jedem Wert (ungleich Null) eines Fremdschlüsselattributs 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 (<u>PNO</u>, 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,
     FOREIGN KEY(DEAN)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 S 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)`



WARTUNG DER REFERENTIELLEN INTEGRITÄT (3)

- Lösch- und Änderungsregeln
  - 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 (nicht zulässig, wenn Attribut nicht `NULL` sein darf)
  - SET DEFAULT: Fremdschlüssel wird auf einen benutzerdefinierten Default-Wert gesetzt

```
CREATE TABLE S(
     A2 INT PRIMARY KEY)
CREATE TABLE R(
     A1 INT PRIMARY KEY,
     A2 INT,
     FOREIGN KEY(A2)REFERENCES S ON(DELETE|UPDATE)
         (NO ACTION|CASCADE|SET NULL|SET DEFAULT))
```

BEISPIEL WARTUNG REFERENTIELLE INTEGRITÄT

STUDENT

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

EXAM

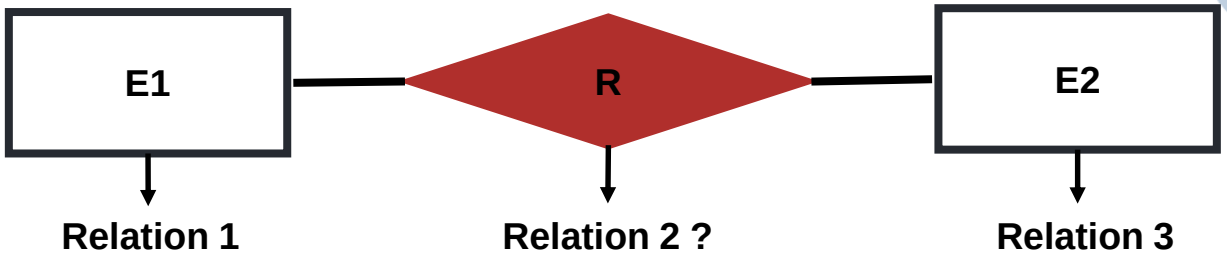
EID	MATNO	TOPIC	DATE	MARK
1780	6511	FA	19.9.	2
1781	1981	DBS	15.10.	1
1782	6511	DBS	17.4.	2
1783	1981	KI	25.3.	3

```
DELETE FROM STUDENT
WHERE MATNO=1981
```

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

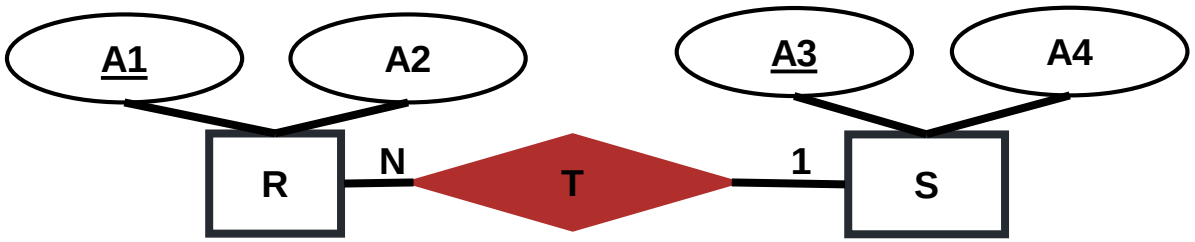
- Auswirkungen von ***<Action>***
  - NO ACTION:
  - CASCADE:
  - SET NULL:

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

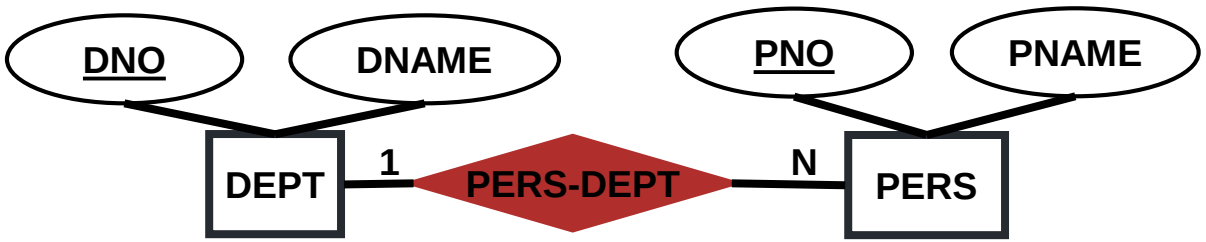




- Regel: n:1-Beziehungen lassen sich ohne eigene Relation darstellen.
  - in Relation R, der pro Tupel maximal 1 Tupel der anderen Relation S zugeordnet ist, wird Primärschlüssel von S als Fremdschlüssel verwendet
  - Fremdschlüssel in R `NOT NULL`, wenn genau 1 Tupel aus S ein Tupel aus R zugeordnet ist
  - Ausnahme: Relationship-Menge ist durch Attribute spezifiziert →

  3 Relationen möglich
- `R(A1, A2, A3(FS auf S))  S(A3, A4)`

2 ENTITY-MENGEN MIT N:1 - VERKNÜPFUNG





Ohne Attribute (Standard)

DEPT(DNO, DNAME)

PERS(PNO, PNAME, DNO(FS auf DEPT))

Mit Attribut Since für PERS-DEPT

DEPT(DNO, DNAME)

PERS(PNO, PNAME,)

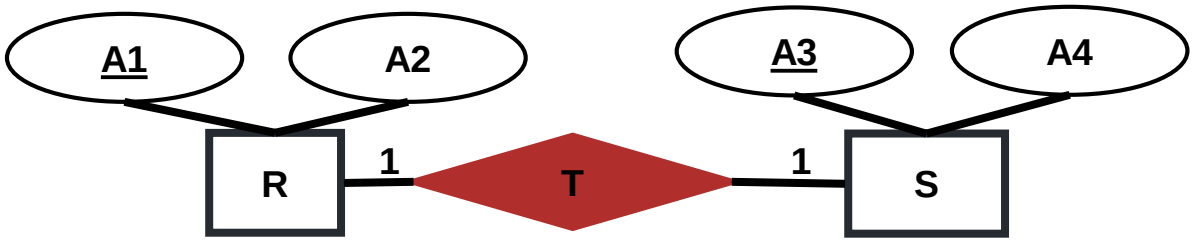
PERS-DEPT(PNO(FS auf PERS), DNO(FS auf DEPT), Since)

Obligatorisch - Multiplizität 1

DEPT(DNO, DNAME)

PERS(PNO, PNAME, DNO(FS auf DEPT, NOT NULL))

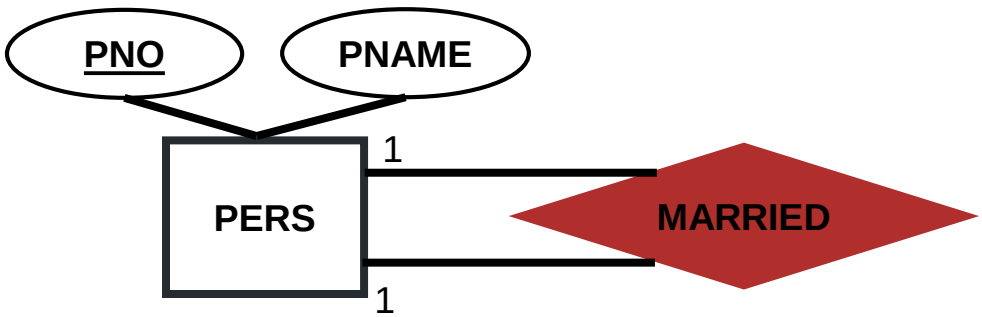
1 ENTITY-MENGE MIT 1:1 VERKNÜPFUNG





- Regel: 1:1-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
  - Fremdschlüssel in R `NOT NULL`, wenn genau 1 Tupel aus S ein Tupel aus R zugeordnet ist
  - Fremdschlüssel in R muss eindeutig sein, da einem Tupel aus S nur maximal ein Tupel aus R zugeordnet sein kann
- `R(A1, A2, A3(FS auf S, UNIQUE))  S(A3, A4)`

1 ENTITY-MENGE MIT 1:1 VERKNÜPFUNG





Allgemein

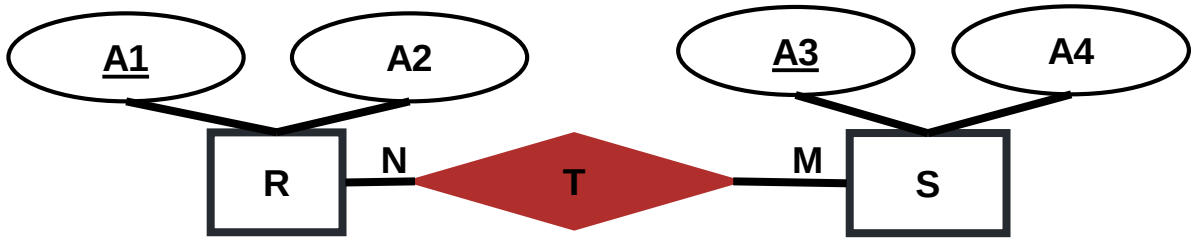
PERS(PNO, PNAME, PARTNER(FS auf PERS,UNIQUE))

Obligatorisch - Multiplizität 1

PERS(PNO, PNAME)

DEPT(DNO, DNAME, MGR(FS auf PERS, UNIQUE, NOT NULL))

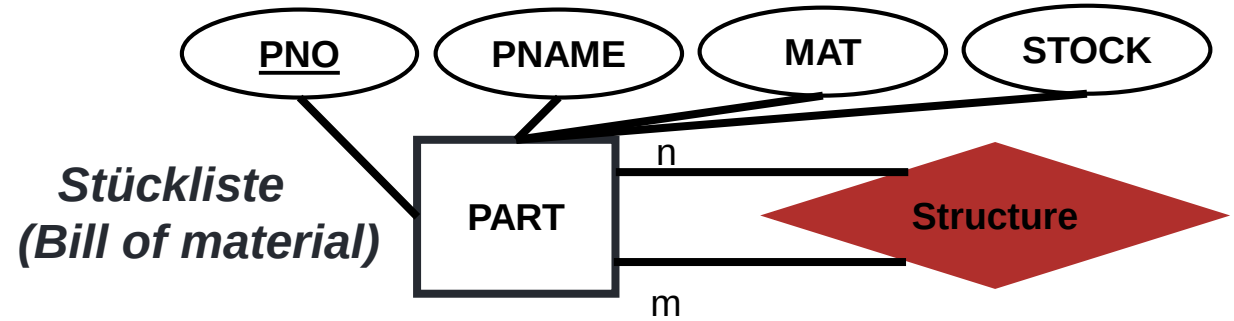
2 ENTITY-MENGEN MIT N:M - VERKNÜPFUNG



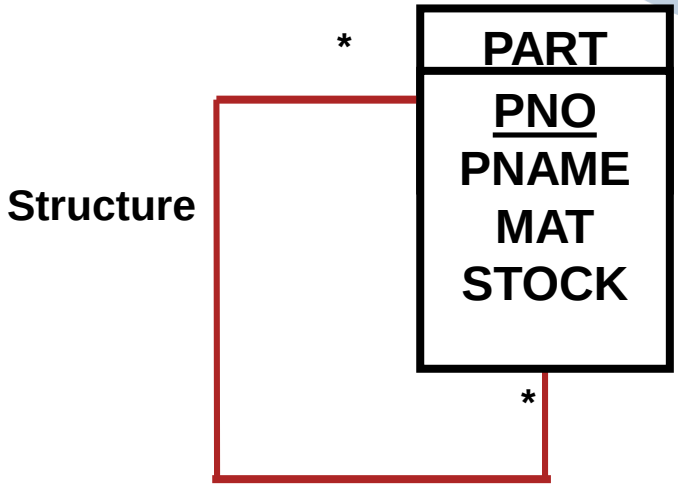
- Regel: n:m-Beziehungen benötigen eigenständige Relation
  - Primärschlüssel der dazugehörigen Entitätsmengen sind Fremdschlüssel in der eigenständigen Relation T und referenzieren die jeweiligen Relationen
  - Primärschlüssel der Relation T ist zusammengesetzt aus den Primärschlüssel der beteiligten Entitätsmengen
    - Eventuell Attribute der Relationship-Menge mit in den Primärschlüssel aufnehmen, falls Beziehung zwischen Entitäten mehrfach auftreten kann
    - In der Praxis generierte PS
- `R(A1, A2)  S(A3, A4)`

  `T(A1(FS auf R), A3(FS auf S))`

1 ENTITY-MENGE MIT M:N-VERKNÜPFUNG



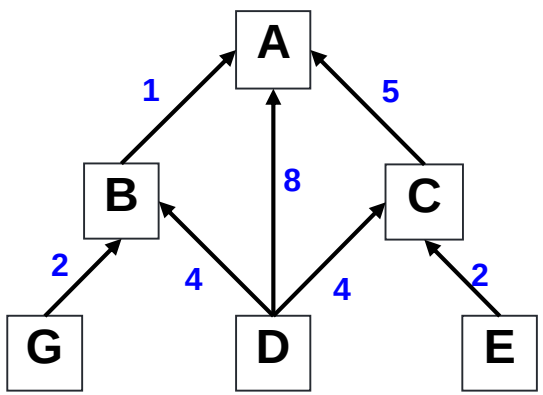
Stückliste
(Bill of material)



Darstellungsmöglichkeit im RM:

PART (PNO, PNAME, MAT, STOCK)

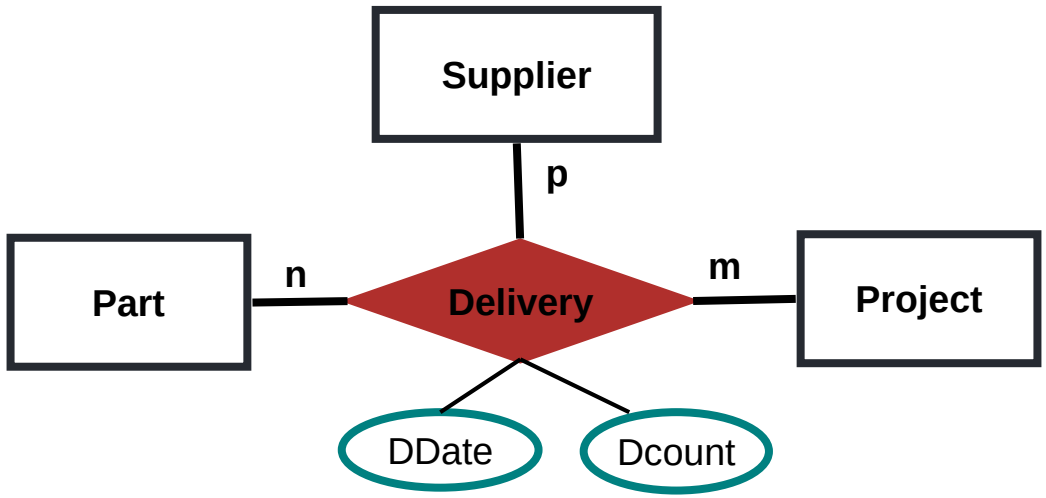
STRUCTURE (TOP(FS auf PART), COMP(FS auf PART), PCOUNT)



PNO	PNAME	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

TOP	COMP	PCOUNT
A	B	1
A	C	5
A	D	8
B	D	4
B	G	2
C	D	4
C	E	2

3 ENTITY-MENGEN MIT M:N-VERKNÜPFUNG

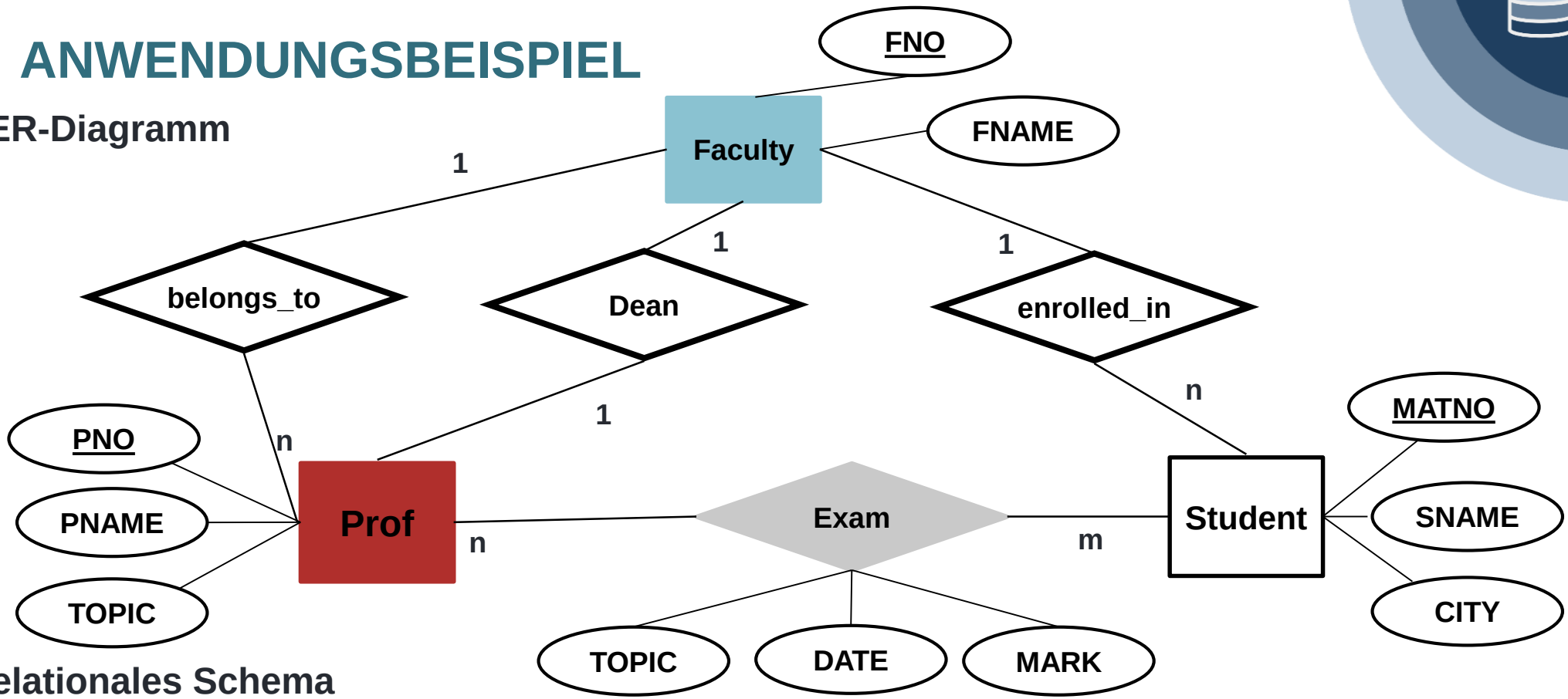



```
SUPPLIER(SNO, SNAME, SCITY,...)
PROJECT(PRONO, PRONAME,PCITY,...)
PART(PNO, PNAME, WEIGHT,... )
DELIVERY(SNO(FS auf SUPPLIER),
         PRONO(FS auf PROJEKT),
          PNO(FS auf PART), DDate, Dcount)
```

- Einfügen eines künstlichen Primärschlüssels D-NO in Delivery möglich

ANWENDUNGSBEISPIEL

ER-Diagramm



relationales Schema

FACULTY

FNO	FNAME	DEAN

PROF

PNO	PNAME	FNO	TOPIC

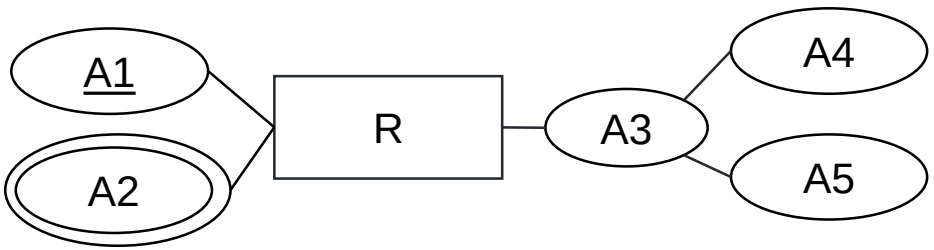
STUDENT

MATNO	SNAME	FNO	CITY

EXAM

MATNO	PNO	TOPIC	DATE	MARK

ABBILDUNG MEHRWERTIGER BZW. ZUSAMMENGESETZTER ATTRIBUTE



Darstellungsmöglichkeit im RM

`R(A1, A4, A5)`

`R_A2(A1(FS auf R),A2)`

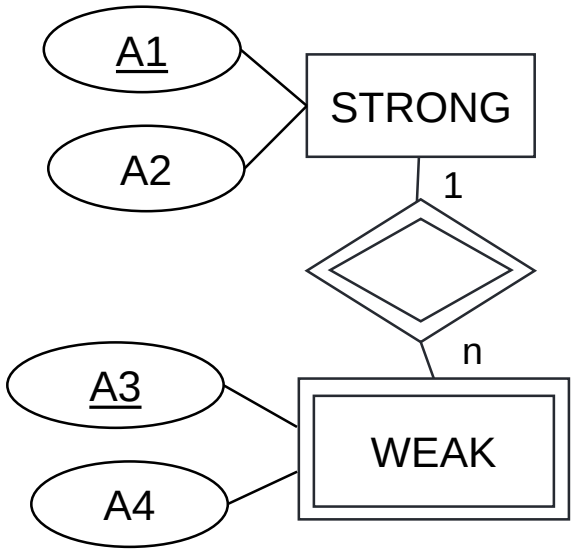
Mehrwertig:

- eigenständige 1:N Relation mit Primärschlüssel als Fremdschlüssel auf die Relation R
- PS zusammengesetzt aus Attributen des mehrwertigen Attributs und PS der Relation R

Zusammengesetzt

- Atomare Attribute des zusammengesetzten Attributs in Relation R übernehmen

ABBILDUNG SCHWACHER ENTITY-MENGEN



Darstellungsmöglichkeit im RM

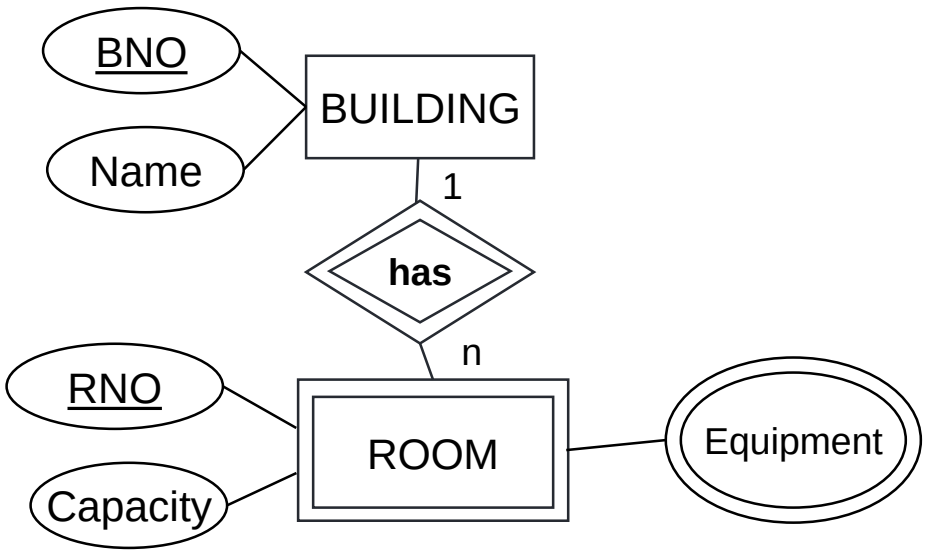
```
STRONG(A1,A2)

WEAK(A1(FS auf STRONG ON {DELETE, UPDATE}
    CASCADE), A3, A4)
```

Regel

- wie n:1 Beziehung mit Lösch-und Updateregel `CASCADE`
- PS der WEAK Relation ist Kombination aus PS der STRONG Relation und partiellen PS der eigenen Relation

ABBILDUNG MEHRWERTIGER ATTRIBUTE BZW. SCHWACHER ENTITY-MENGEN - BEISPIEL



Darstellungsmöglichkeit im RM

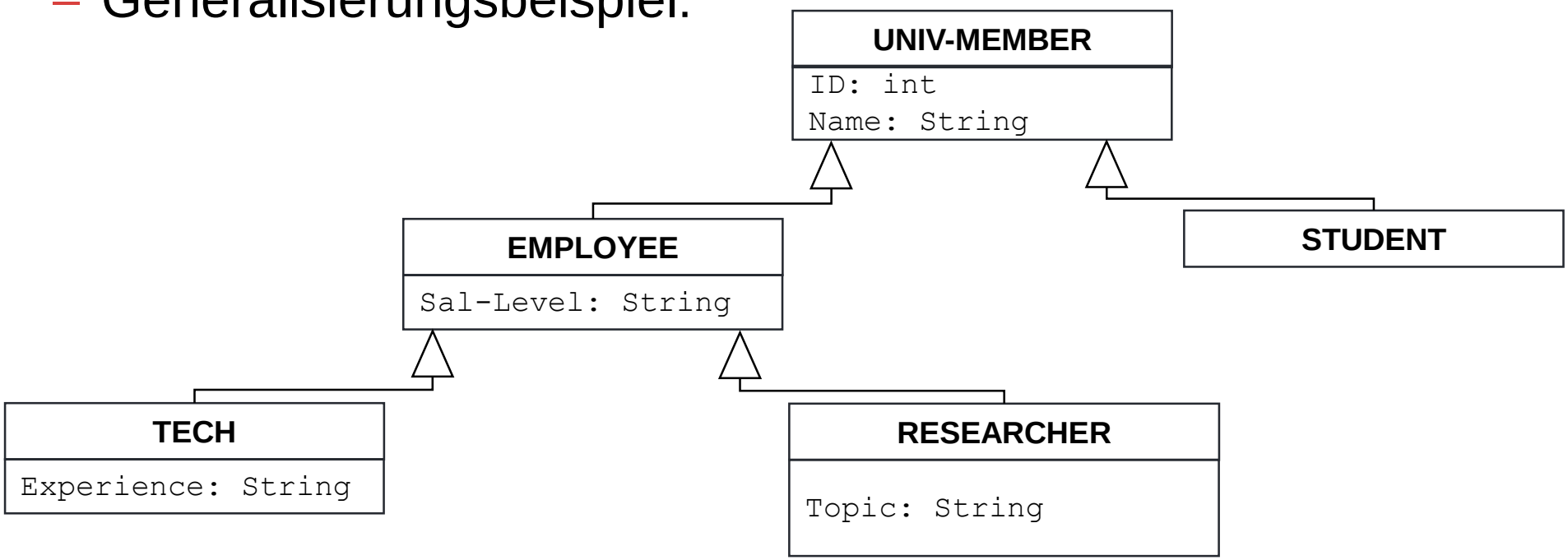
```
BUILDING(BNO,NAME)

ROOM(BNO(FS auf Building ON {DELETE, UPDATE} CASCADE), RNO,
    Capacity)

EQUIPMENT_ROOM(BNO, RNO, EQUIPMENT) (BNO, RNO) FS auf ROOM
```

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
    - Fremdschlüssel auf Super-Relation
- Eigenschaften
  - geringfügig erhöhte Speicherungskosten, aber hohe Aufsuch- und Aktualisierungkosten
  - Integritätsbedingungen: TECH.ID $\subseteq$ EMPLOYEE.ID, usw.
  - Instanzenzugriff erfordert Verbundoperationen, z.B. Bestimme alle TECH-Daten

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

GENERALISIERUNG – RELATIONALE SICHT (2)

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

UNIV-MEMBER

ID	Name
111	Ernie

EMPLOYEE

ID	Name	Sal
333	Daisy	E9

RESEARCHER

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

TECH

ID	Experience	Name	Sal
123	Linux	Donald	E11

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

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

GENERALISIERUNG: VERFAHRENSVERGLEICH

	vertikale Partitionierung	horizontale Partitionierung	volle Redundanz
Änderungen			
Lesen			

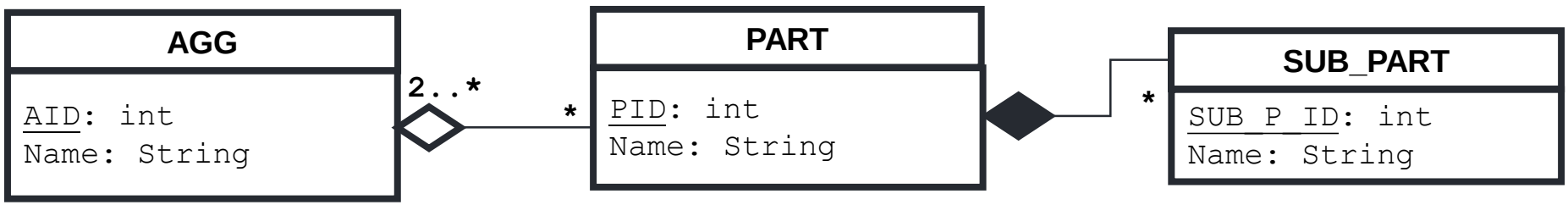
AGGREGATION UND KOMPOSITION – RELATIONALE SICHT

Aggregation

– Multiplizitäten ≥ 1 bei der Aggregat-Klasse erfordern n:m Relation ansonsten 1:N Modellierung

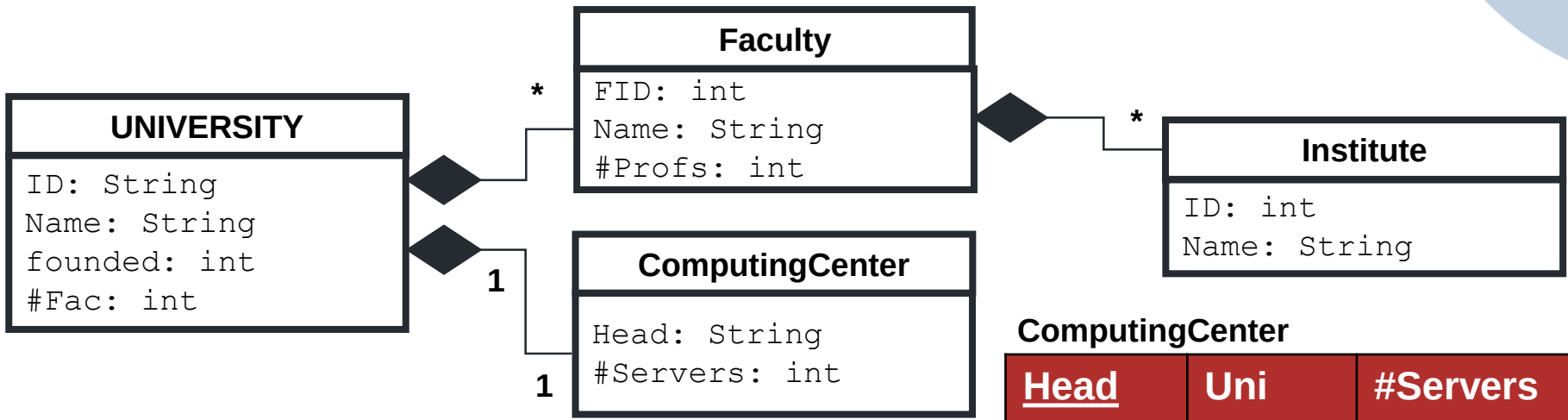
Komposition

– Wie 1:N Beziehung mit NOT NULL Bedingung für Fremdschlüssel



AGG(AID,Name)
PART(PID,Name)
AGG_PART(AID(FS auf AGG), PID(FS auf PART))
SUB_PART(SUB_P_PID, PID(FS auf PART, NOT NULL), Name)

AGGREGATION UND KOMPOSITION – RELATIONALE SICHT - BEISPIEL



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

ComputingCenter

Head	Uni	#Servers
Kühne	UL	120
Müller	TUD	130

Institute

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

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