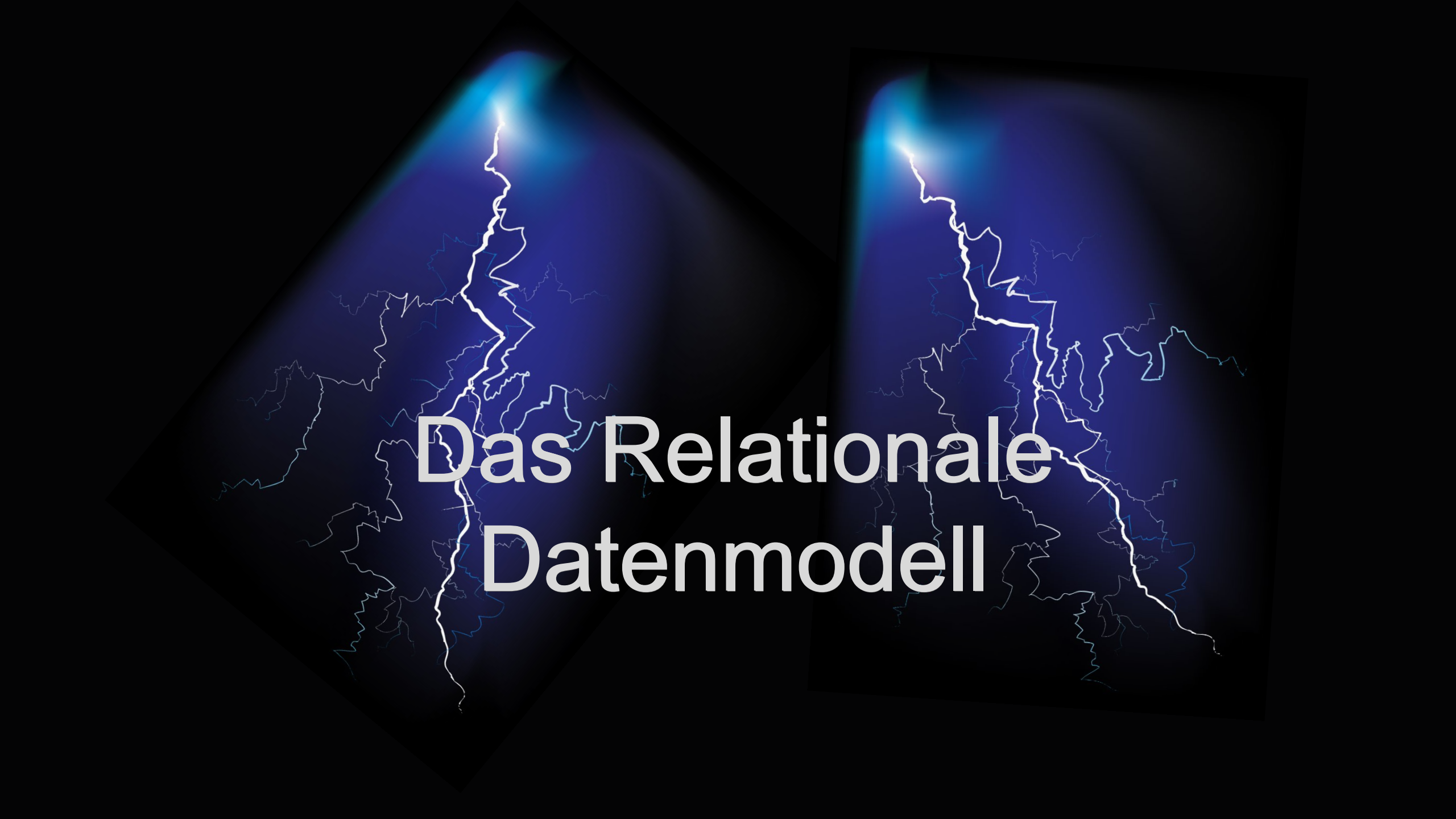


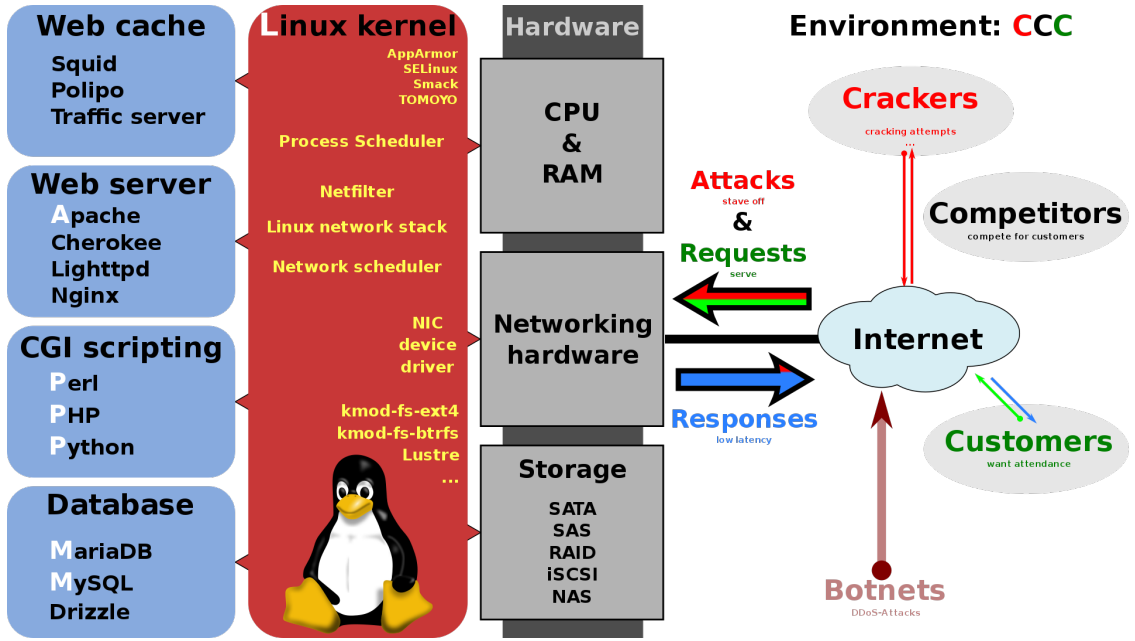
Das Relationale Datenmodell

3. Das Relationale Datenmodell

1. **Das Datenmodell**
2. Transformation von ER-Diagrammen in das Relationale Modell
3. Relationale Algebra
4. Relationaler Kalkül

Beispiele für Relationale Datenbanken-Anwendungen: Teil des LAMP Stacks

- LAMP (Linux, Apache, MySQL, PHP/Perl/Python)
  - Verbreiteter Software-Stack für Webanwendungen
- MySQL: relationales Datenbankmanagementsystem (RDBMS) im LAMP-Stack
  - Andere RDBMS-Optionen: PostgreSQL, MariaDB
- Anwendungsbeispiele:
  - Content-Management-Systeme (z.B. WordPress, Joomla)
  - E-Commerce-Plattformen (z.B. Magento, PrestaShop)
  - Webanwendungen mit Datenbankzugriff und -verwaltung (z.B. Kundenportale, CRM-Systeme)



Datenbanken – Wie machen es die “Großen”?
Beispiel: Google Spanner und Facebook TAO

Google Spanner [1][2]:

- Verteilte “NewSQL” Datenbank mit “multi-version concurrency control”.
  - “Semi-relationales” Datenmodel.
  - Unterstützt SQL Anfragen und Transaktionen.

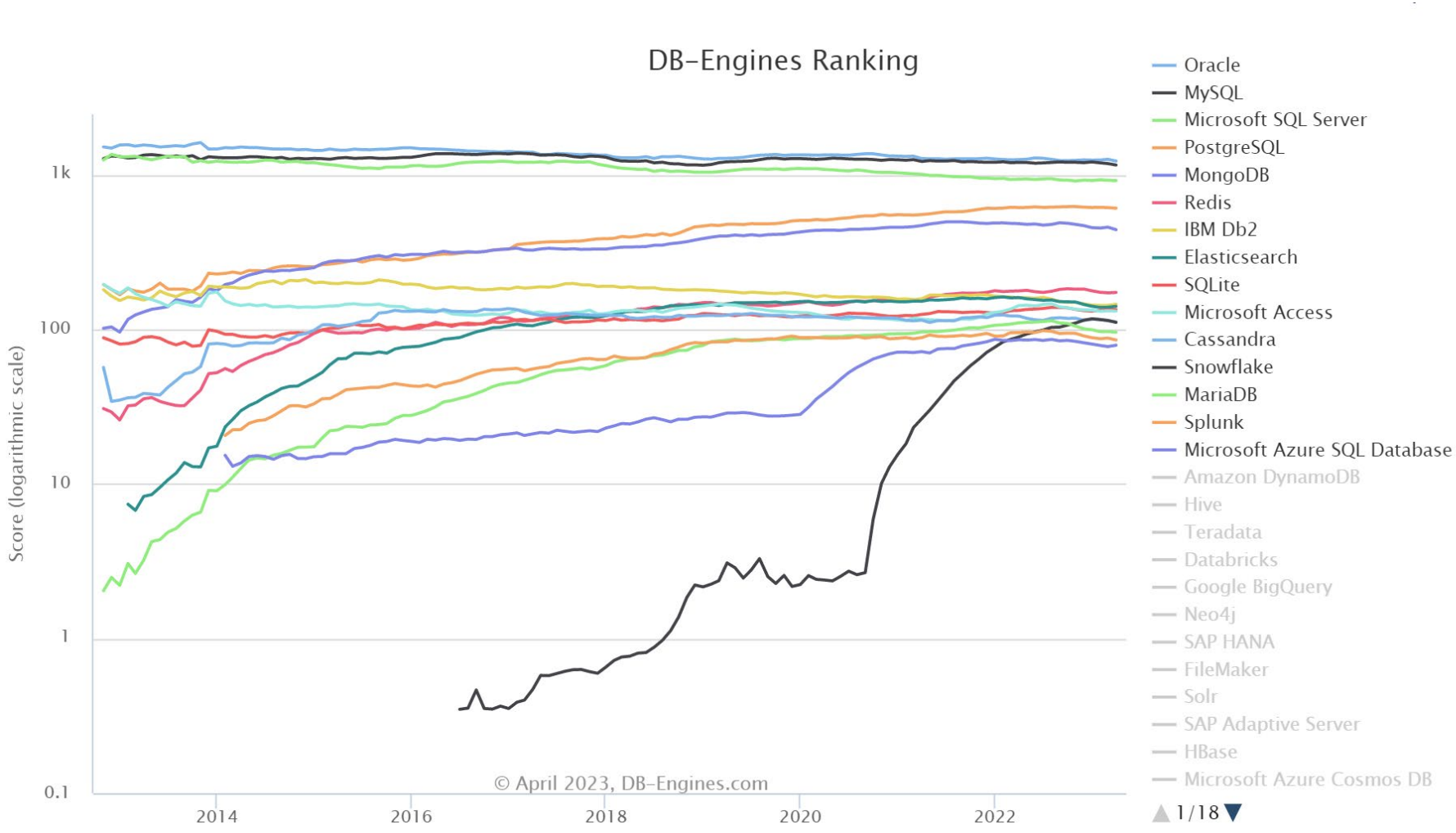
Facebook TAO [3]:

- Verteilte Graphdatenbank – verwendet MySQL zur Persistenz.
  - Optimiert für Lesen.
  - Wertet Effizienz und Verfügbarkeit höher als Konsistenz.

TAO

[1] J.C. Corbett et al.: Spanner: Google's Globally-Distributed Database. OSDI 2012: 251-264
[2] D. F. Bacon et al: Spanner: Becoming a SQL System. SIGMOD Conference 2017: 331-343
[3] N. Bronson, et al.: TAO: Facebook's Distributed Data Store for the Social Graph. USENIX Annual Technical Conference 2013: 49-60

Datenbankranking



https://db-engines.com/de/ranking_trend

Einführung

- Das relationale Datenmodell nutzt das einfache Strukturierungsprinzip “Tabelle” zur Modellierung sowohl von Objekten als auch von Beziehungen.

- Das Modell geht auf *Codd* (1970) zurück und hat sich seither auf breiter Basis durchgesetzt.
  - Edgar F. Codd: *A Relational Model of Data for Large Shared Data Banks*. Commun. ACM 13(6): 377-387 (1970), see http://dblp.dagstuhl.de/pers/hd/c/Codd:E=_F=

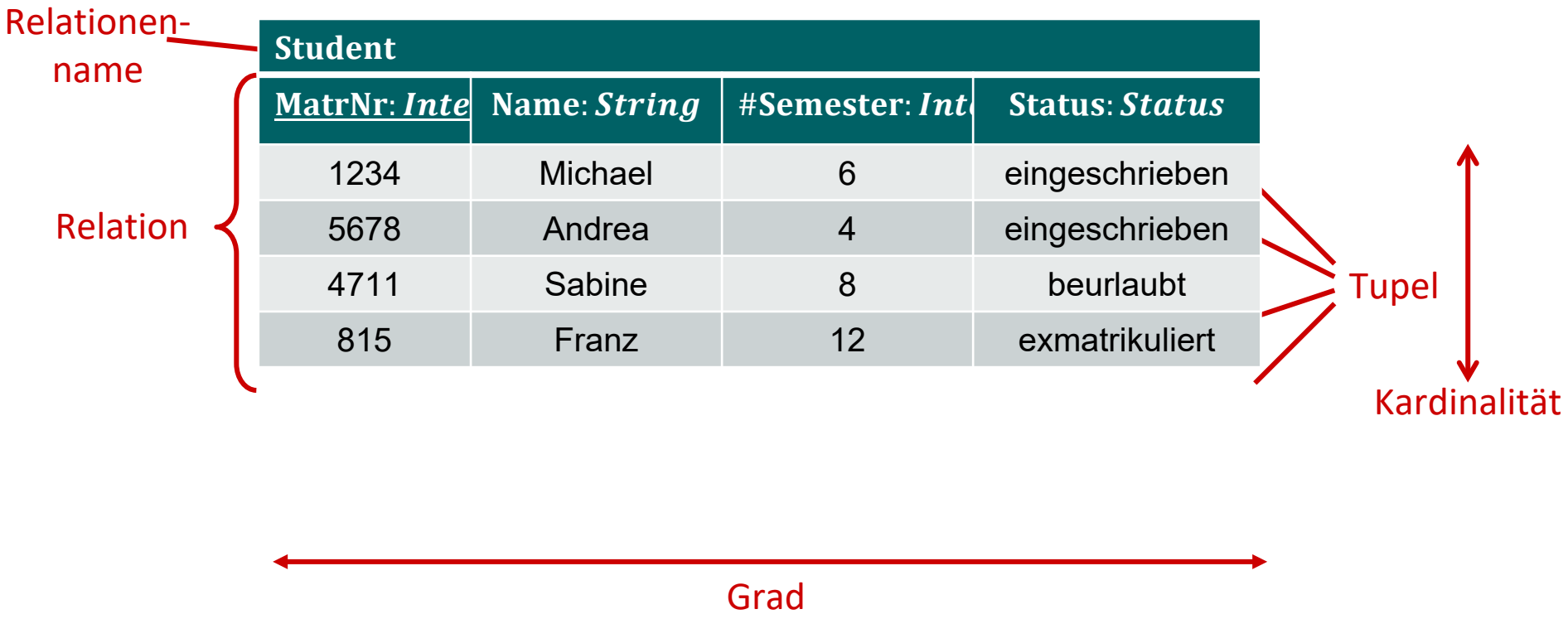
- Heute ist es Grundlage vieler kommerzieller Datenbanksysteme (Oracle, IBM DB/2, Informix, Ingres, Sybase, MS SQL-Server, MS Access, usw.) und damit wichtiger Bestandteil vieler großer Anwendungssysteme.

- Im naturwissenschaftlich-technischen Bereich dient es vielfach als Grundlage für komplexere Datenmodelle, insbesondere für sogenannte “Nichtstandard-Anwendungen”.

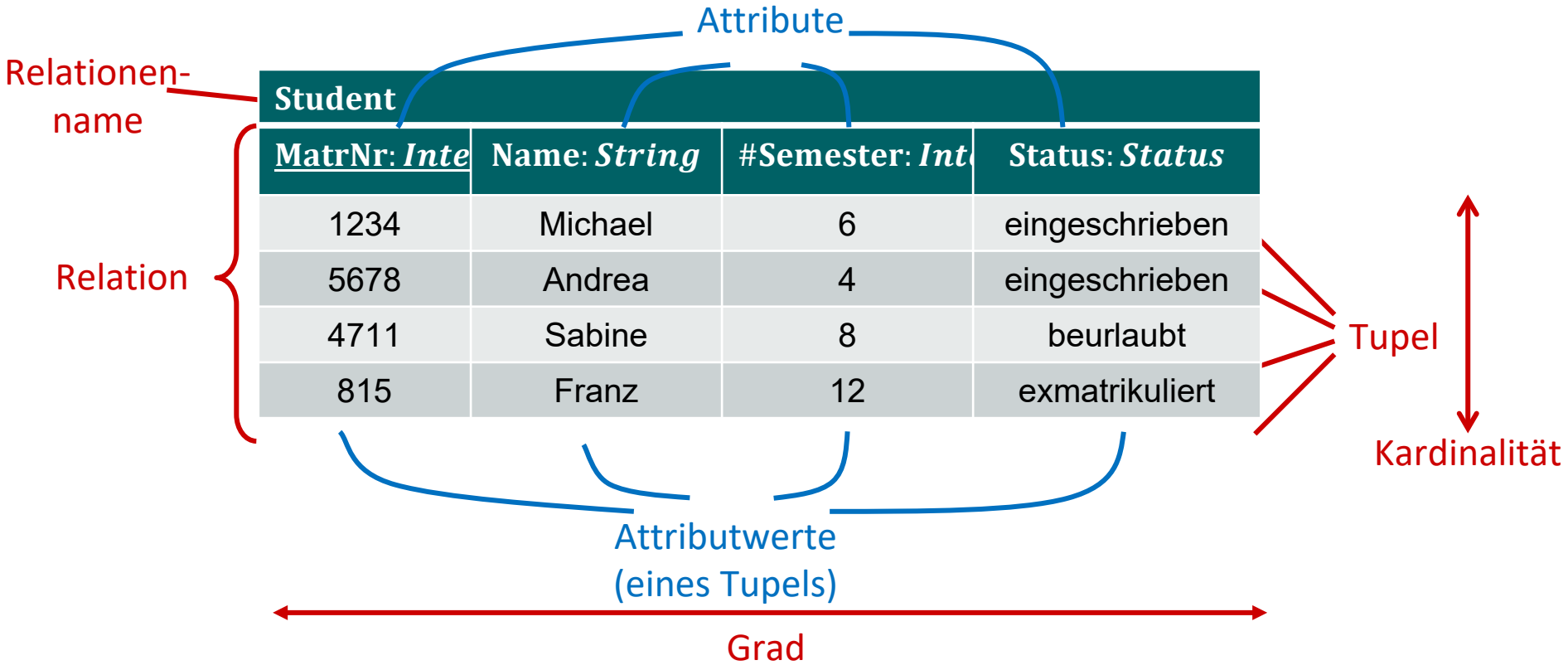
Relationenschemata Terminologie

Student			
MatrNr: *Inte*	Name: *String*	#Semester: *Int*	Status: *Status*
1234	Michael	6	eingeschrieben
5678	Andrea	4	eingeschrieben
4711	Sabine	8	beurlaubt
815	Franz	12	exmatrikuliert

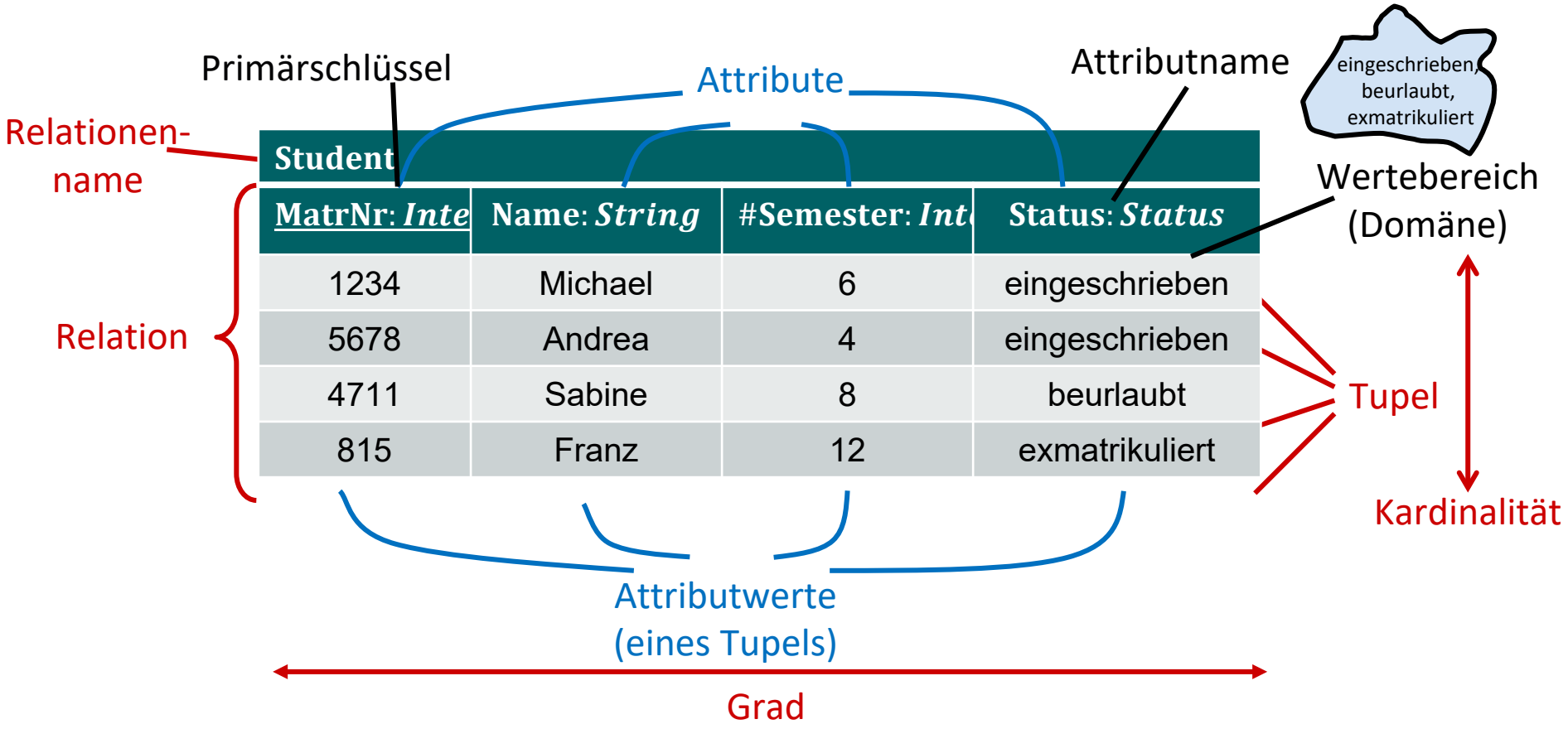
Relationenschemata Terminologie



Relationenschemata Terminologie



Relationenschemata Terminologie



Relationen, Domänen

- Ein *Wertebereich (Domäne, Domain)* ist eine (logisch zusammengehörige) Menge von Werten, z.B. INTEGER, STRING, DATUM, {1, … , 10}, etc. Eine Domäne kann *endliche* oder *unendliche* Kardinalität haben. Die Funktion dom bildet Attribute auf ihren Wertebereich ab.

- Ist $k \geq 1$ und $\{A_1, \ldots, A_k\}$ eine Folge von Attributnamen mit den Wertebereichen $D_i = dom(A_i), 1 \leq i \leq k$, dann ist eine *Relation r* Teilmenge des kartesischen Produktes der Wertebereiche $D_1, \ldots, D_k$:

$$r \subseteq D_1 \times \cdots \times D_k$$

- *Beispiel*
  - Gegeben seien die Wertebereiche $D_1 = \{a, b, c\}$ und $D_2 = \{0,1\}$. Dann gibt es unter anderem die Relationen:
    - $r_1 = D_1 \times D_2 = \{(a,0), (a,1), (b,0), (b,1), (c,0), (c,1)\}$
    - $r_2 = \{(a,0), (b,0), (c,0), (c,1)\}$
    - $r_3 = \emptyset$

Tupel, Tabellen

- Einzelne Elemente einer Relation heißen *Tupel*.
- Für $r \subseteq D_1 \times D_2 \times \cdots \times D_k$ heißt k der *Grad* oder die *Stelligkeit* der Relation; alle Tupel in r haben k Komponenten.

- Relationen kann man als *Tabellen* verstehen und darstellen. Die Zeilen einer Tabelle entsprechen den Tupeln. Die Spalten heißen Attribute; sie können Namen tragen, im Beispiel *MatNr* und *Name*.

- Bemerkung: Im Datenbankbereich betrachtet man gewöhnlich nur endliche Relationen. Unendliche Relationen können nicht materialisiert werden, sie treten z.B. im Bereich der deduktiven Datenbanksysteme auf.

$$R_2 = \{(30487, Bob), (30548, Ede), (30375, Anna), (30455, Pia)\}$$

R_2	
$MatNr: D_1$	$Name: D_2$
30487	Bob
30548	Ede
30375	Anna
30455	Pia

$D_1 = \text{Integer} = \{1,2, \dots\}$,
$D_2 = \text{String}$

Relationenschemata und Relation

- **Geordnetes Relationenschema**
  - *Geordnetes einfaches Relationenschema* $R' = R(A_1, \ldots, A_k)$
    - Relationsname R
    - Liste von (voneinander verschiedenen) Attributen $A_1, \ldots, A_k$. Jedes Attribut A_i, $1 \leq i \leq k$ ist der Name einer Rolle, die von einem Wertebereich (Domain) D im *Relationenschema* R' gespielt wird..D_i wird die Domain von A_i genannt und wird mit $dom(A_i)$ bezeichnet (also $\mathrm{dom}(A_i) = D_i$).
    - Der Grad einer Relation ist die Anzahl der Attribute k ihres Relationenschemas.
- **Relation**:
  - Eine Relation r des Relationenschemas $R(A_1, \ldots, A_k)$ (auch $r(\mathrm{R}')$), ist eine Menge von k-Tupeln $\mathrm{r} = \{t_1, \ldots, t_n\}$.
  - Jedes k -Tupel t ist eine geordnete Liste von k Werten $t = (v_1, \ldots, v_k)$ wobei jeder Wert $v_i, 1 \leq i \leq k$, ein Element von $dom(A_i)$ ist oder ist.
  - Der i-te Wert im Tupel t, der dem Wert des Attribut A_i entspricht, wird als $\mathrm{t}(\mathrm{A_i})$ oder $\mathrm{t}.A_i$ bezeichnet (oder $\mathrm{t}[i]$ bei Verwendung der Positionsnotation).

- Hinweise:
  - Auf die Attributwerte eines Tupels t kann man also über die Namen zugreifen: $t(A)$ bzw. $t.A$ oder $t(B)$ bzw. $t.B$ – d.h., man kann die Attribute als Abbildungen betrachten. Im folgenden verwenden wir jeweils die einfacher anzuwendende Alternative.
  - Wenn es aus dem Kontext klar wird, verwenden wir den Buchstaben R sowohl für das Relationenschema als auch für den Namen der Relation
  - Das Relationenschema einer Relation r bezeichnen wir auch manchmal mit $\mathrm{Sch}(r)$. Wir verwenden (später) auch Audrücke der Form $(A_1 : D_1, \ldots, A_k : D_k)$ als Notation für das Schema einer Relation. Gemeint ist $\mathrm{dom}(A_i) = D_i$.
  - Wir erlauben uns einige Freiheiten bei der Verwendung von Groß- und Kleinbuchstaben. Was gemeint ist sollte eindeutig aus dem Kontext hervorgehen.

Relationen, Reihenfolgen, Datenbanken

- Reihenfolgeabhängigkeiten
  - Die Reihenfolge der Zeilen (= Tupel) spielt keine Rolle (Relation ist Menge).
  - Im geordneten Relationenschema ist die Reihenfolge der „Spalten“ wichtig:

$$\{(a, 0), (b, 1)\} \neq \{(0, a), (1, b)\}$$

- **Relation, Datenbank**
  - Eine *Relation* ist eine Ausprägung eines Relationenschemas.
  - Ein *Datenbankschema* ist eine Menge von Relationenschemata.
  - Eine *Datenbank* ist eine Menge der aktuellen Relationen (Ausprägungen)

Beispiel: Relation Städte

Städte		
Name	*Einwohner*	*Land*
München	1.211.617	Bayern
Bremen	535.058	Bremen

- Als *"geordnetes Relationenschema"*:
  - Schema: Städte(Name, Einwohner, Land)
  - dom(Name) = STRING, dom(Einwohner) = INTEGER, dom(Land) = STRING
  - Ausprägung: {(München, 1.211.617, Bayern), (Bremen, 535.058, Bremen)}

- *Tupel als Abbildung*:
  - Ausprägung: $\{t_1 , t_2\}$ mit:
    - t_1(Name) = München, t_1 (Einwohner) = 1.211.617, t_1 (Land) = Bayern
    - t_2 (Name) = Bremen, t_2 (Einwohner) = 535.058, t_2 (Land) = Bremen

Schlüssel

- Eine minimale Teilmenge der Attribute eines Relationenschemas, anhand der alle Tupel einer (möglichen) Relation unterscheidbar sind, heißt Schlüssel.

- Definition *(Schlüssel):*
  - Eine Teilmenge S der Attribute eines Relationenschemas R ist ein *Schlüssel* (genauer *Schlüsselkandidat*), wenn gilt:
    - (i) **Eindeutigkeit:** Keine Ausprägung von R kann zwei verschiedene Tupel enthalten, die sich in allen Attributen von S gleichen.
    - (ii) **Minimalität:** Keine echte Teilmenge von S erfüllt die Bedingung (i).

- Formal:
  - Sei r eine Relation über dem Schema R und $S \subseteq R$ eine Teilmenge der Attribute von R; $t[S]$ bezeichne die Einschränkung (Projektion) des Tupels t auf die Attribute in S.
  - S ist ein Schlüssel der Relation r, wenn gilt:
    - (i) Für alle möglichen Ausprägungen r und Tupel $t_1, t_2 \in r$ gilt: $t_1 \neq t_2 \Rightarrow t_1[S] \neq t_2[S]$
    - (ii) Für alle Attributmengen T, die (i) erfüllen, gilt: $T \subseteq S \Rightarrow T = S$

Beispiel für Schlüssel

- *Wichtig:*
  - Die Schlüsseleigenschaft ist abhängig von der Semantik des Schemas, nicht von der aktuellen Ausprägung einer Relation!

- *Beispiel*: Relation mit Personalnummer und Gehalt

Personal	
PNr	*Gehalt*
1	1.700 €
2	2.172 €
3	3.189 €
4	2.167 €

  - In der Relation *Personal* erfüllen sowohl die Attributmengen $\{PNr\}$ als auch $\{Gehalt\}$ die Bedingung (*i*). Schlüssel ist jedoch nur $\{PNr\}$, da Personalnummern logisch stets eindeutig sind, Gehälter jedoch nicht.
  - Auch $\{PNr, Gehalt\}$ erfüllt Bedingung (*i*), kann jedoch auf $\{PNr\}$ reduziert werden und ist deshalb kein Schlüssel.

- Gelegentlich gibt es mehrere Schlüsselkandidaten für eine Relation, unter denen dann einer als *Primärschlüssel* ausgewählt wird (üblicherweise der mit den kürzesten Attributswerten).
- Wie bereits im ER-Diagramm werden die zum Schlüssel gehörigen Attribute im Schema unterstrichen: Mitarbeiter (<u>PNr</u>, Gehalt)

Das Relationale Datenmodell

1. Das Datenmodell
2. **Transformation von ER-Diagrammen in das Relationale Modell**
3. Relationale Algebra
4. Relationaler Kalkül

Informatik 5 -
Informationssysteme
und Datenbanken

RWTH AACHEN
UNIVERSITY

Datenabhängigkeiten

- Im relationalen Modell:
  - Objekte (Entities) und alle Arten von Beziehungen (Relationships) werden durch Relationen dargestellt
- keine Einschränkungen: an den Relationen können beliebige Mengen von Attributen beliebiger Objekte beteiligt sein
  - auch „ungültige“ Kombinationen können auftreten
- Integritätsbedingungen um Konsistenz der Datenbank zu gewährleisten:
  - intra- und interrelationale Abhängigkeiten
  - wichtig zur Modellierung von 1:n-, Is-A-Beziehungen, uvm.

Relationenschema

- Intrarelationale Abhängigkeiten
  - Beispiel: Schlüssel einer Relation
  - Erlaubt Aussagen über die Konsistenz/Gültigkeit einer Relation
    - Ein Schlüssel einer Relation darf in der Relation nicht doppelt vorkommen

- Formal: Erweiterung der Relationenschemata
  - Es bezeichne X die Menge der Attribute eines geordneten Relationschemas R'
  - $\mathrm{Tup}(X)$ bezeichnet die Menge aller Tupel über X
  - $\mathrm{Rel}(X) = \{ r \mid r \subseteq \mathrm{Tup}(X) \}$ ist die Menge aller Relationen über X
  - eine intrarelationale Abhängigkeit über X ist eine Abbildung

    $$\sigma : \mathrm{Rel}(X) \to \{\mathbf{true}, \mathbf{false}\},$$

    formal spezifiziert sie, welche der möglichen Relationen eine gegebene Bedingung erfüllen

Relationenschema

- Intrarelationale Abhängigkeiten
  - Beispiel: Schlüssel einer Relation
  - Sei $K \subseteq X$. Eine Schlüsselabhängigkeit σ_K bezeichnet folgende intrarelationale Abhängigkeit

$$\sigma_K : \mathrm{Rel}(X) \to \{true, false\}, \qquad r \mapsto \begin{cases} true, & \text{falls } K \text{ Schlüssel von } r \text{ ist} \\ false, & \text{sonst} \end{cases}$$

- $R'' = (R', \Sigma_X)$ ist ein erweitertes *Relationenschema*, dabei sind
  - R' ein geordnetes Relationenschema
  - Σ_X eine Menge intrarelationaler Abhängigkeiten (Schlüsselabhängigkeiten) über X, dabei ist X die Menge der Attribute in R'
  - eine Relation r mit erweitertem Relationenschema $R'' = (R', \Sigma_X)$ heißt *gültig* oder *konsistent*, falls sie alle intrarelationalen Abhängigkeiten Σ_X erfüllt

Datenbankschema

- Interrelationale Abhängigkeiten
  - Beispiel 2 Tabellen:
    - Student(MatrNr, Name, Semester)
    - Hört(MatrNr, VorlNr)
    - $\text{Hört}[\text{MatrNr}] \subseteq \text{Student}[\text{MatrNr}]$
  - Erlaubt Aussagen über die Konsistenz/Gültigkeit einer Datenbank
    - jede Matrikelnummer die in der Relation Hört enthalten ist, muss auch in einem Eintrag in Student existieren

- Formal: Erweiterung der Datenbankschemata
  - Es bezeichne $\mathcal{R} = \{R_1, \dots, R_k\}$ eine endliche Menge von erweiterten Relationenschemata
  - Eine Datenbank $D = \{r_1, \dots, r_k\}$ ist eine Menge von Relationen (Ausprägungen)
  - $\text{Dat}(\mathcal{R})$ bezeichnet die Menge aller Datenbanken über $\mathcal{R}$
  - eine interrelationale Abhängigkeit über $\mathcal{R}$ ist eine Abbildung

$$\sigma : \text{Dat}(\mathcal{R}) \to \{\mathbf{true}, \mathbf{false}\},$$

formal spezifiziert sie, welche der möglichen Datenbanken die gegebene Bedingung erfüllen

Datenbankschema

- Interrelationale Abhängigkeiten
  - Beispiel: $\text{Hört}[\text{MatrNr}] \subseteq \text{Student}[\text{MatrNr}]$
  - Inklusionsabhängigkeit: Sei $\mathcal{R} = \{R_1, \dots, R_k\}$ eine endliche Menge von Relationenschemata

    und $V \subseteq X_i, W \subseteq X_j$,$|V| = |W|, u: V \mapsto W$ eine bijektive Abbildung sowie $r_i \in Rel(X_i), r_j \in Rel(X_j)$.

Eine Inklusionsabhängigkeit:

$$R_i[V] \subseteq R_j[W]$$

bezeichnet folgende interrelationale Abhängigkeit

$$\sigma_{R_i[V] \subseteq R_j[W]} : \text{Dat}(\mathcal{R}) \to \{true, false\}, \qquad D \longmapsto \begin{cases} true, & \text{falls } \{ t[V] \mid t \in r_i \} \subseteq \{ s[W] \circ u \mid s \in r_j \} \\ false, & \text{sonst} \end{cases}$$

Anschaulich: jede Matrikelnummer die in der Relation Hört enthalten ist, muss auch in einem Eintrag in Student existieren. u ist bei gleichen Attributnamen die Identitätsfunktion, ansonsten wird die Definition von u aus dem Kontext klar, da u die Attributnamen aufeinander abbildet.

Beachte: Attribute werden hier als Abbildungen behandelt – daher die Verkettung mit u!

Datenbankschema

- Interrelationale Abhängigkeiten
  - Beispiel: $\text{Assistent}[\text{PersNr}] \cap \text{Professor}[\text{PersNr}] = \emptyset$
  - Zweiter Typ, Exklusionsabhängigkeit: Sei wieder $\mathcal{R} = \{R_1, \ldots, R_k\}$ eine endliche Menge von Relationenschemata, $V \subseteq X_i, W \subseteq X_j$,$|V| = |W|, u: V \mapsto$ W eine bijektive Abbildung sowie $r_i \in Rel(X_i), r_j \in Rel(X_j)$ .Eine Exklusionsabhängigkeit

$$R_i[V] \cap R_j[W] = \emptyset$$

bezeichnet die Abhängigkeit

$$\sigma_{R_i[V] \cap R_j[W] = \emptyset} : \text{Dat}(\mathcal{R}) \to \{true, false\}, D \mapsto \begin{cases} true, \text{falls } \{ t[V] \mid t \in r_i \} \cap \{ s[W] \circ u \mid s \in r_j \} = \emptyset \\ false, \text{sonst} \end{cases}$$

Datenbankschema

- $D = (\mathcal{R}, \Sigma_{\mathcal{R}})$ ist ein *Datenbankschema*, dabei sind
  - $\mathcal{R}$ eine Menge von Relationenschemata
  - $\Sigma_{\mathcal{R}}$ eine Menge interrelationaler Abhängigkeiten über $\mathcal{R}$
- Eine Datenbank $\mathrm{d} \in Dat(\mathcal{R})$ mit Datenbankschema $D = (\mathcal{R}, \Sigma_{\mathcal{R}})$ heißt *konsistent* oder *gültig*, falls alle intra- und interrelationalen Abhängigkeiten erfüllt sind

Fremdschlüssel

- Inklusionsabhängigkeiten: Modellierung von $1{:}n$ – und Is-A – Beziehungen

- Sei $D = (\mathcal{R}, \Sigma_{\mathcal{R}})$ ein Datenbankschema. Eine Attributmenge $F \subseteq X$ der Attributmenge eines Relationenschemas $R_1 \in \mathcal{R}$ heißt Fremdschlüssel, bezogen auf eine Relation r_2, falls
  - F den selben Wertebereich besitzt wie der Schlüssel von r_2
  - die interrelationale Abhängigkeit $R_1[F] \subseteq R_2[K]$, wobei K der Schlüssel von r_2 ist, gilt in D, d.h.

    $\sigma_{R_1[F] \subseteq R_2[K]} \in \Sigma_{\mathcal{R}}$

- Die zweite Bedingung nennt man auch *referentielle Integrität* des Fremdschlüssels
- Anschaulich: jedes Tupel in den Fremdschlüsselattributen verweist auf ein existierendes Tupel der referenzierten Relation oder die Fremdschlüsselattribute sind auf Null gesetzt

ER Modell und Relationales Modell

- Ausgangslage nach konzeptueller Entwurf: ER-Modell
  - Entity-Typen
  - Beziehungs-Typen

- Wo wollen wir hin: Relationales Modell
  - nur ein Strukturierungskonzept: Relationen

Unsere Aufgaben:

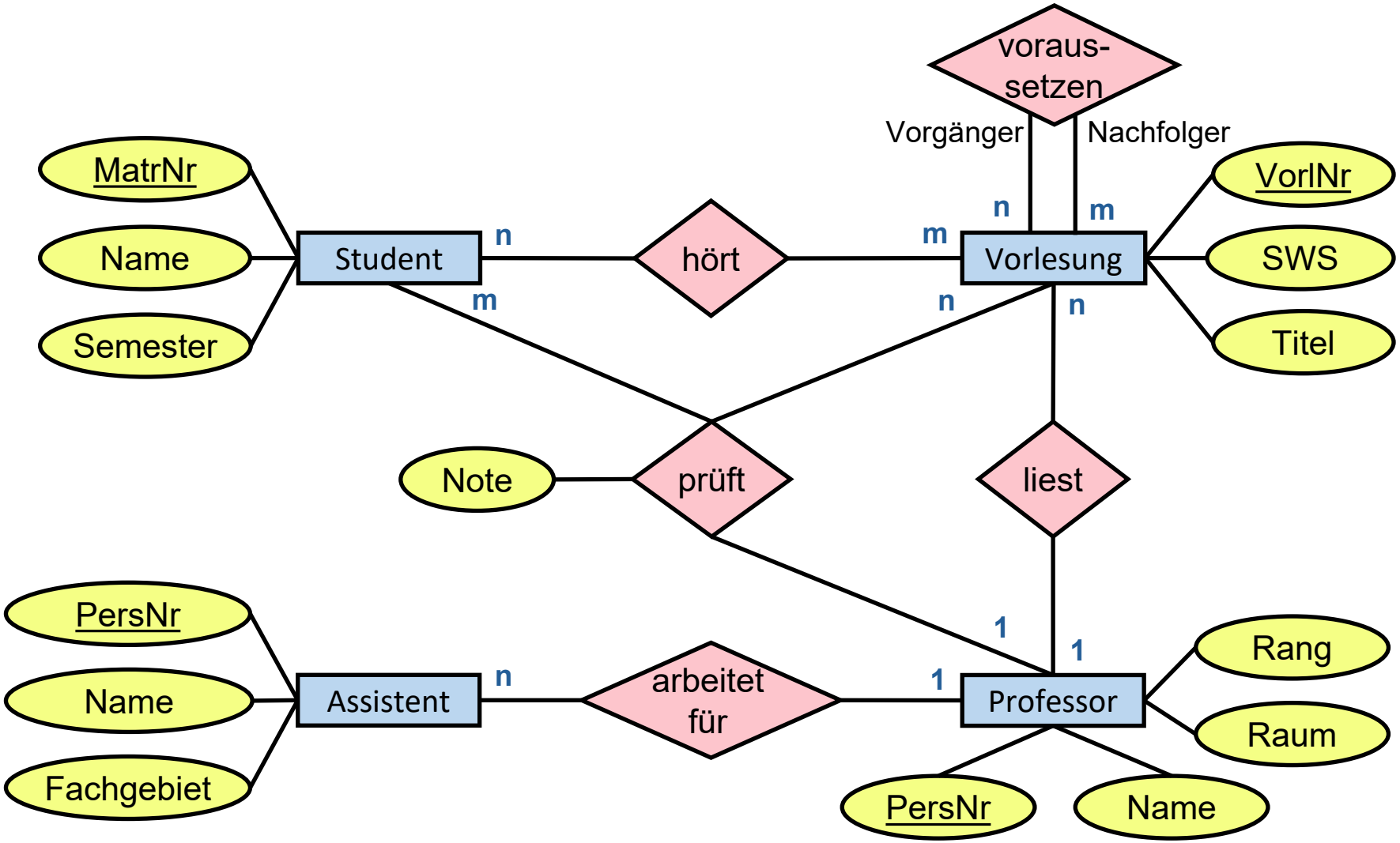
- Entities und Beziehungen müssen auf Relationen abgebildet werden
- Intrarelationale und interrelationale Abhängigkeiten ableiten

Unsere Todo-Liste

Wir müssen abbilden:

- Entitytypen mit Attributen
- Beziehungen
  - n:m, 1:n, 1:1
  - Rekursive Beziehungen
- Generalisierung / Spezialisierung
  - Partiell, total
- Spezialfälle
  - Schwache Entity-Typen
  - n-stellige Beziehungen

Beispiel : ER-Diagramm (Universität)



Unsere Todo-Liste

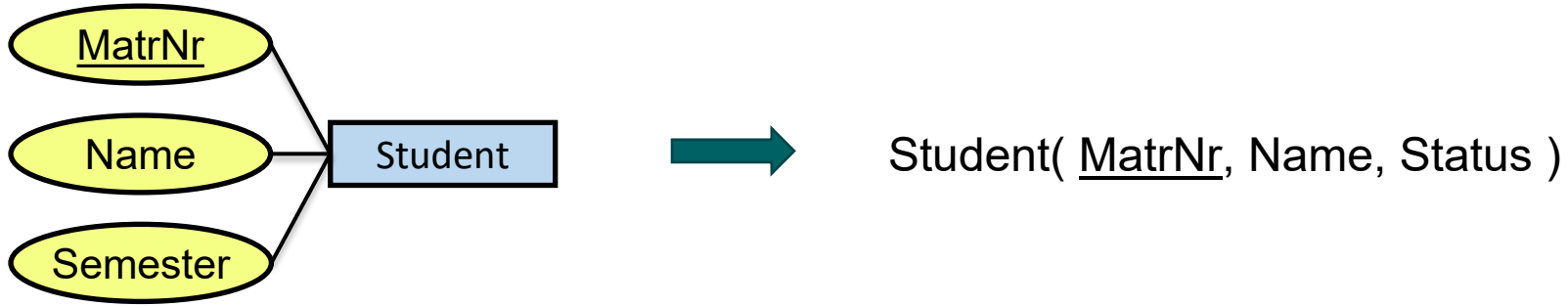
Wir müssen abbilden:

- **Entitytypen mit Attributen**
- Beziehungen
  - n:m, 1:n, 1:1
  - Rekursive Beziehungen
- Generalisierung / Spezialisierung
  - Partiell, total
- Spezialfälle
  - Schwache Entity-Typen
  - n-stellige Beziehungen

Entitytyp mit Attributen

- Jeder Entity-Typ wird zu einer Tabelle/Relation mit entsprechenden Attributen

- Spezialfälle
  - Zusammengesetzte Attribute
    - Jedes (Teil-) Attribut als einzelnes Attribut
    - Oder das zusammengesetzte Attribute als Ganzes
  - Mehrwertige Attribute
    - Neue Relation mit zusammengesetztem Schlüssel (bestehend aus eigenem Schlüssel und Fremdschlüssel auf den Entity-Typ)
    - Ähnlich zu einer 1:n – Beziehung



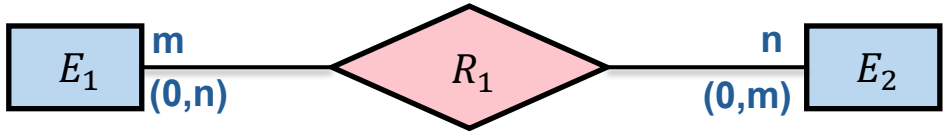
Unsere Todo-Liste

Wir müssen abbilden:

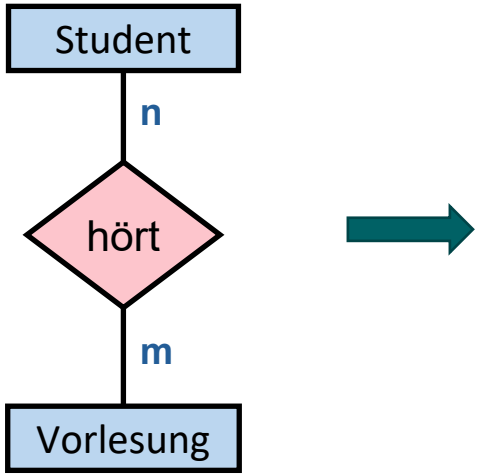
- Entitytypen mit Attributen
- **Beziehungen**
  - n:m, 1:n, 1:1
  - Rekursive Beziehungen
- Generalisierung / Spezialisierung
  - Partiell, total
- Spezialfälle
  - Schwache Entity-Typen
  - n-stellige Beziehungen

n:m – Beziehungen

- werden durch eigene Relationen dargestellt



  - Schlüssel: besteht aus zwei Fremdschlüsseln auf die Relationen E_1 und E_2

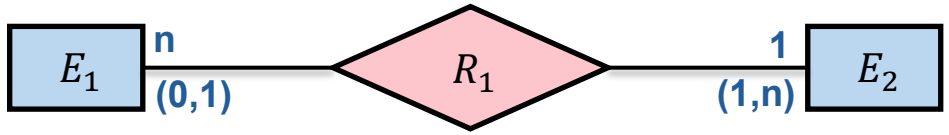


Entitäten:
Student(<u>MatrNr</u>)
Vorlesung(<u>VorlNr</u>)
Hört(<u>MatrNr</u>, <u>VorlNr</u>)

Interrelationale Abhängigkeiten:
Hört[MatrNr] ⊆ Student[MatrNr]
Hört[VorlNr] ⊆ Vorlesung[VorlNr]

1:n – Beziehungen

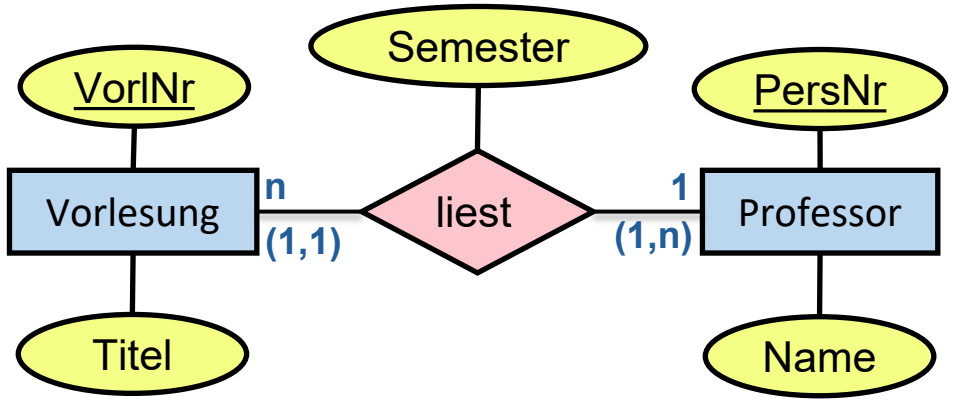
- 1:n – Spezialfall von n:m
  - Schlüssel der Relation ist ein Fremdschlüssel auf E_1 (E_2 ist dann durch E_1 bestimmt)



- Praktikabel falls nur wenige Objekte aus E_1 an R_1 teilnehmen, da die hinzugefügte Relation R_1 dann klein ist.

1:n – Beziehungen

- Effizienter: die Beziehung wird in die Relation des n-Entity-Typs integriert
  - E_1 bekommt Schlüssel von E_2 als Fremdschlüssel



Professor(PersNr, Name)

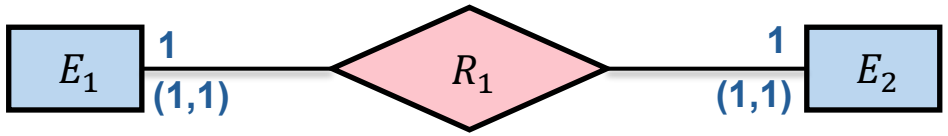
Vorlesung(VorlNr, Titel, Semester,
ProfessorPersNrLiest)

Vorlesung[ProfessorPersNrLiest] ⊆
Professor[PersNr]

  - Vorteil: Eine zusätzliche Relation ist nicht notwendig
  - Nachteil: falls nur wenige Objekte aus E_1 an R_1 teilnehmen enthält E_2 viele null Werte
    - im Beispiel: falls Vorlesungen meistens nicht von einem Professor gehalten werden (andere (min,max)-Notation nötig)

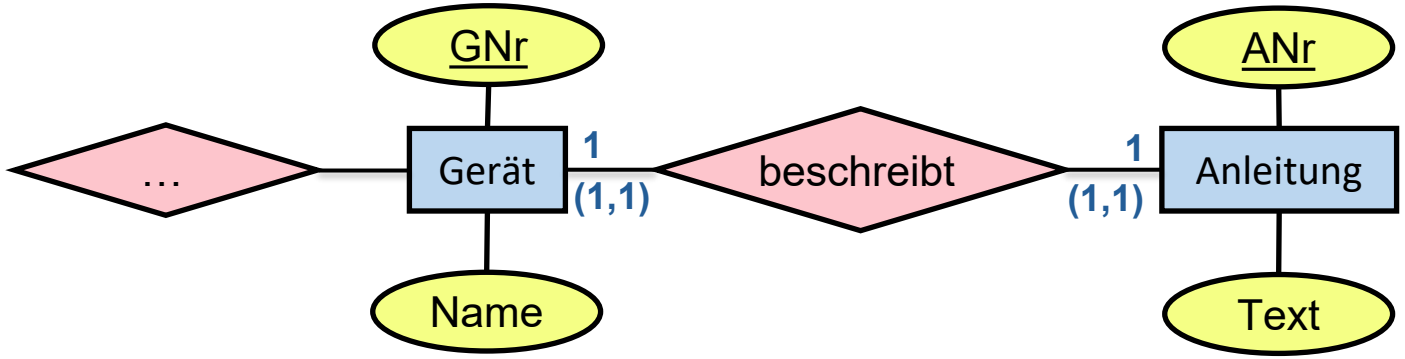
1:1 – Beziehungen

- Mögliche Lösungen:
  - wie 1:n – Beziehung, Fremdschlüssel in E_1 verweist auf E_2 oder eigene Relation



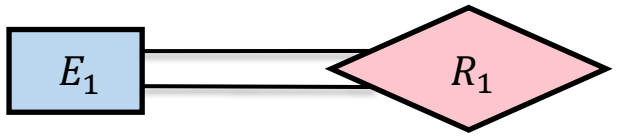
  - Effizienter: Verschmelzen der beteiligten Relationen zu einer einzigen
    - nur falls Teilnahme an R_1 total ist und E_1 und/oder E_2 nicht an anderen Beziehungen teilnehmen

- Beispiel:



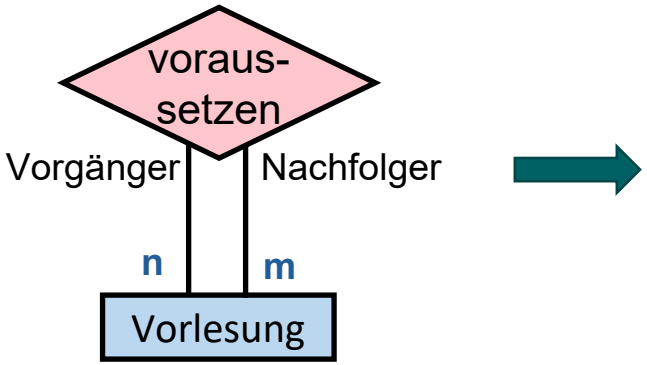
➡ Gerät(GNr, Name, ANr, Text)

Rekursive Beziehungen



- Möglichkeit für 1:1 und 1:n – Beziehung:
  - E_1 hat Fremdschlüssel auf sich selbst
  - Attribute von R_1 werden zu E_1 hinzugefügt

- m:n – Beziehung:
  - eigene Relation für R_1



Voraussetzen(<u>Vorgänger</u>, <u>Nachfolger</u>)

Voraussetzen[Vorgänger] ⊆ Vorlesung[VorlNr]

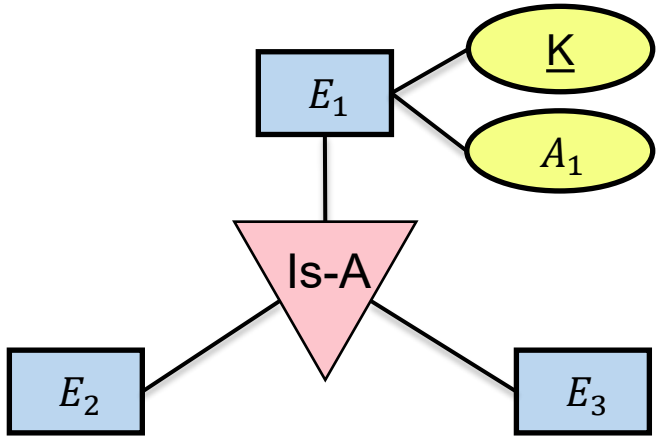
Voraussetzen[Nachfolger] ⊆ Vorlesung[VorlNr]

Unsere Todo-Liste

Wir müssen abbilden:

- Entitytypen mit Attributen
- Beziehungen
  - n:m, 1:n, 1:1
  - Rekursive Beziehungen
- **Generalisierung / Spezialisierung**
  - Partiell, total
- Spezialfälle
  - Schwache Entity-Typen
  - n-stellige Beziehungen

Generalisierung / Spezialisierung



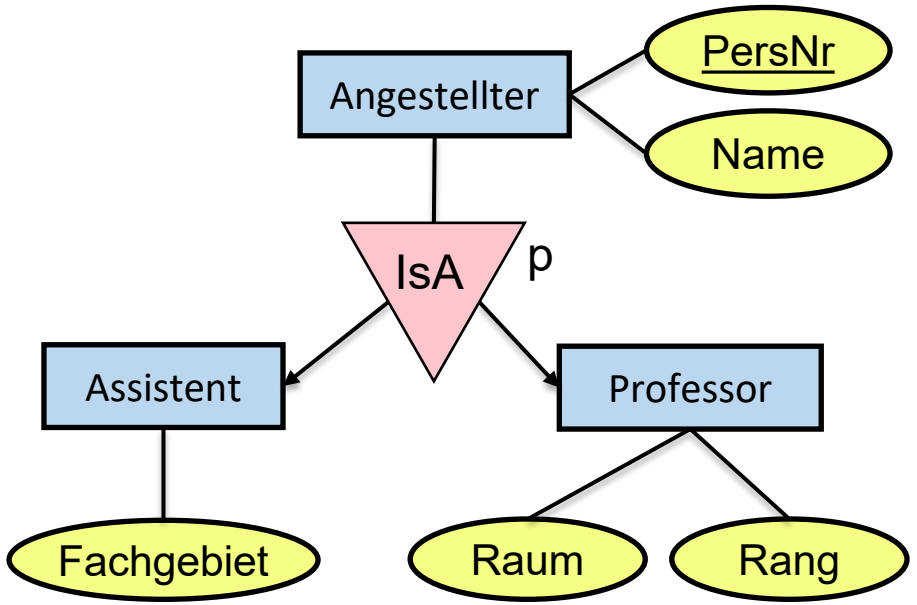
- jede Spezialisierung besitzt einen Fremdschlüssel auf die Generalisierung
  - ist die Beziehung disjunkt: Exklusionsbeziehungen
  - ist die Beziehung zusätzlich total: keine eigene Relation für E_1

- Nachteil (außer bei totalen, disjunkten Beziehungen):
  - Volle Informationen zu E_2 und E_3 können nur durch join mit E_1 abgerufen werden
  - Views/Sichten relationaler DBMS verschaffen Abhilfe

Möglichkeiten zur Übersetzung von isA-Beziehungen

- Erzeuge Tabellen für **jeden** Entity-Typ
  - Für partielle isA-Beziehungen
- Erzeuge **nur** Tabellen **für Subtypen**
  - Bei totalen isA-Beziehungen
- Erzeuge **eine einzige Tabelle**, die alle Attribute aller Entity-Typen enthält, sowie ein **Typ-Attribut**
  - Für disjunkte isA-Beziehungen
- Erzeuge **eine einzige Tabelle**, die alle Attribute aller Entity-Typen enthält, sowie ein **boole'sches Typ-Attribut** für jeden der Entity-Typen
  - falls die Subtypen überlappen (nicht disjunkt)

Generalisierung / Spezialisierung (Partielle Beziehung)

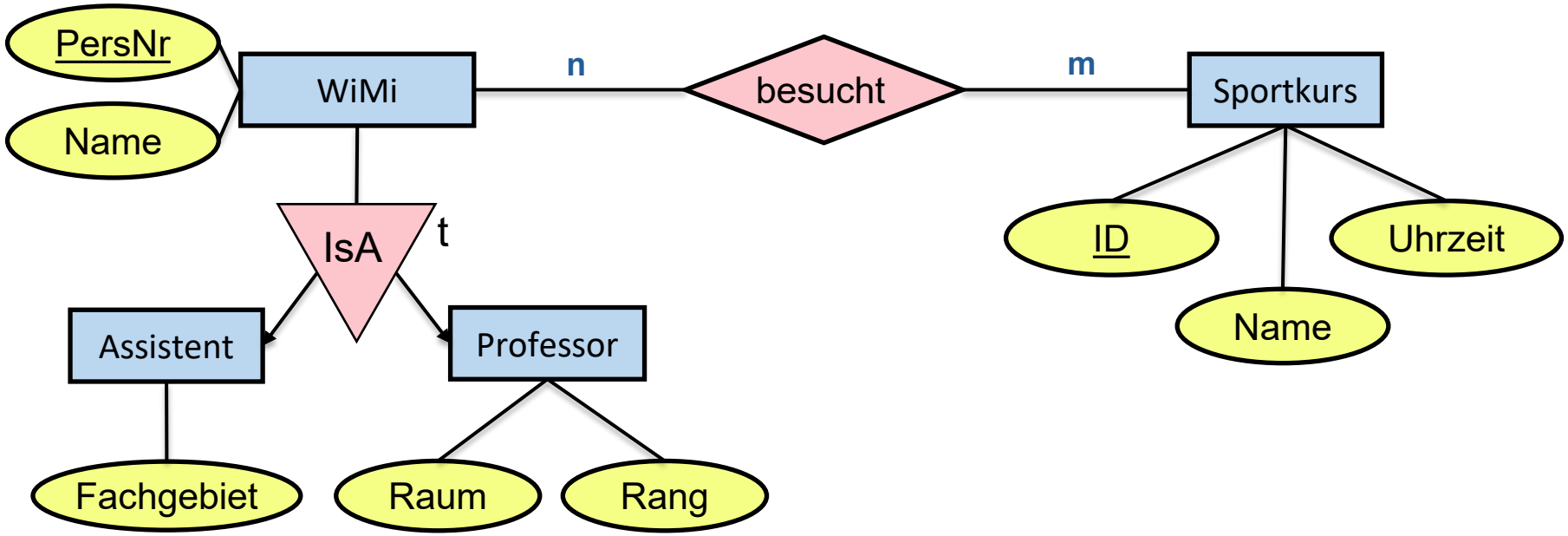


Angestellter(PersNr, Name)
Professor(PersNr, Rang, Raum)
Assistent(PersNr, Fachgebiet)

$Professor[PersNr] \subseteq Angestellter[PersNr]$
$Assistent[PersNr] \subseteq Angestellter[PersNr]$

$Professor[PersNr] \cap Assistent[PersNr] = \emptyset$

Generalisierung / Spezialisierung (totale Beziehung)



Professor(PersNr, Name, Raum, Rang)
Assistent(PersNr, Name, Fachgebiet)
Sportkurs(ID, Name, Uhrzeit)
Besucht(PersNr, SportkursID)

$Professor[PersNr] \cap Assistent[PersNr] = \emptyset$

$Besucht[SportkursID] \subseteq Sportkurs[ID]$
$Besucht[PersNr] \subseteq Assistent[PersNr] \cup Professor[PersNr]$

Unsere Todo-Liste

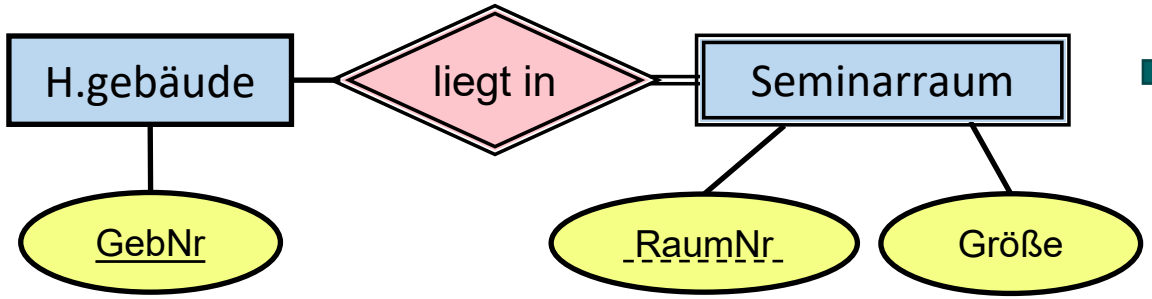
Wir müssen abbilden:

- Entitytypen mit Attributen
- Beziehungen
  - n:m, 1:n, 1:1
  - Rekursive Beziehungen
- Generalisierung / Spezialisierung
  - Partiell, total
- **Spezialfälle**
  - Schwache Entity-Typen
  - n-stellige Beziehungen

Schwache Entity-Typen

- Schwacher Entity-Typ E_1 mit
  - Teilschlüssel K_1
  - Übergeordneter Entity-Typ E_2 mit Schlüssel K_2

- Transformation
  - Relationenschema für E_1: E_1($\underline{K_1}$, $\underline{K_2}$, ...)
  - Schlüssel wird gebildet aus K_1 und K_2
  - K_2 ist ein Fremdschlüssel auf E_1



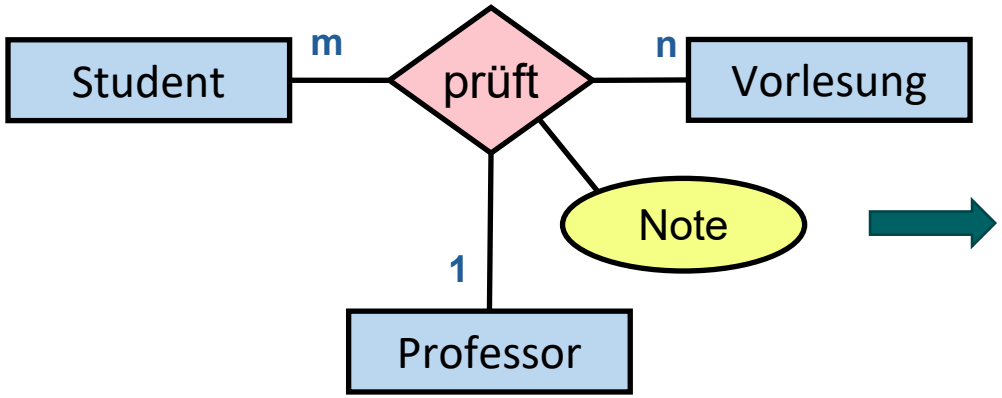
Seminarraum($\underline{\text{RaumNr}}$, $\underline{\text{GebNr}}$, Größe)

Seminarraum[GebNr] ⊆ H.gebäude[GebNr]

n-stellige Beziehungen

- können wie üblich durch eine eigene Relation dargestellt werden
  - Kardinalitäten geben an, wie sich der Schlüssel zusammensetzt

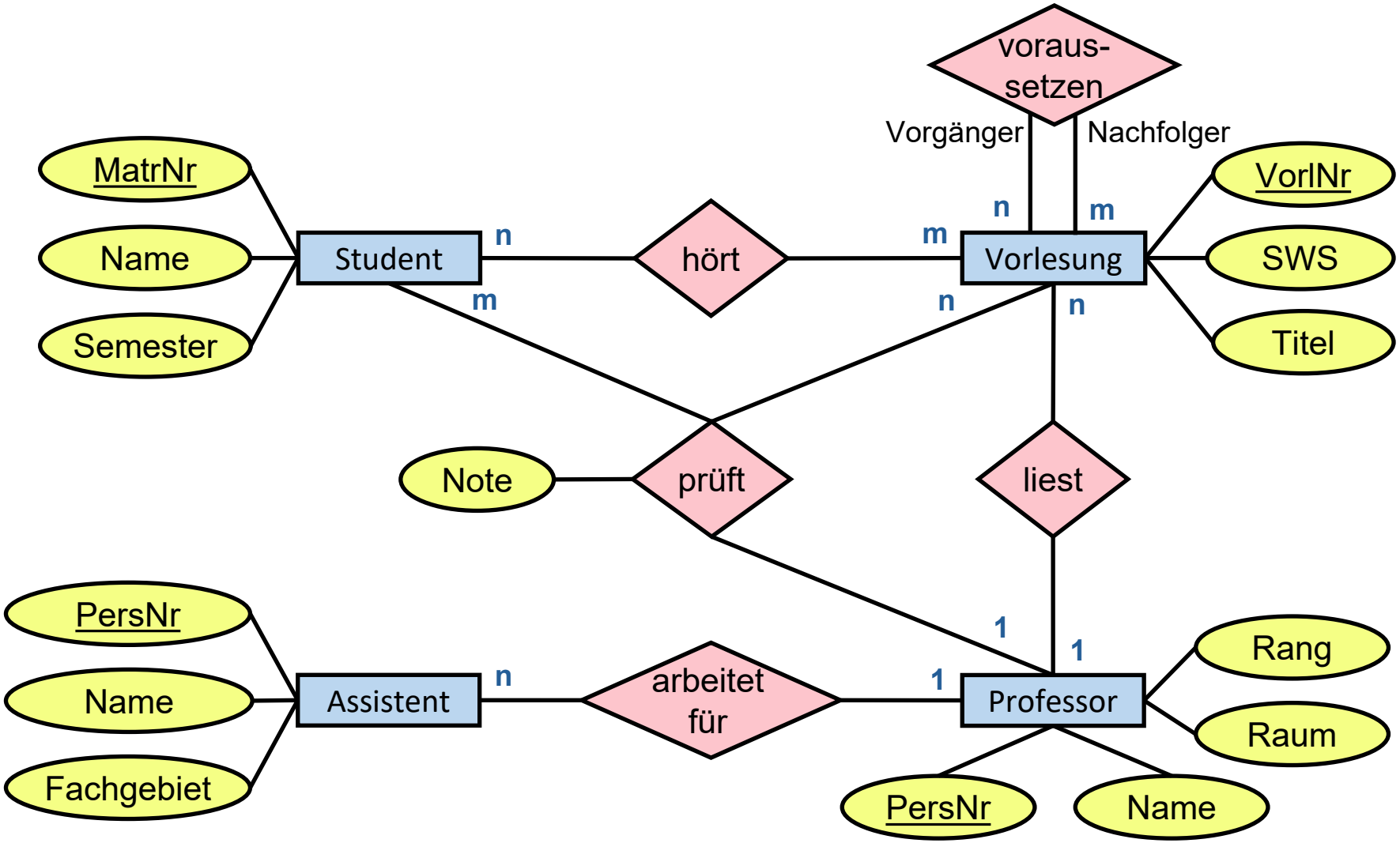
- Beispiel:



Prüft(<u>MatrNr</u>, <u>VorlNr</u>,
PersNr, Note)

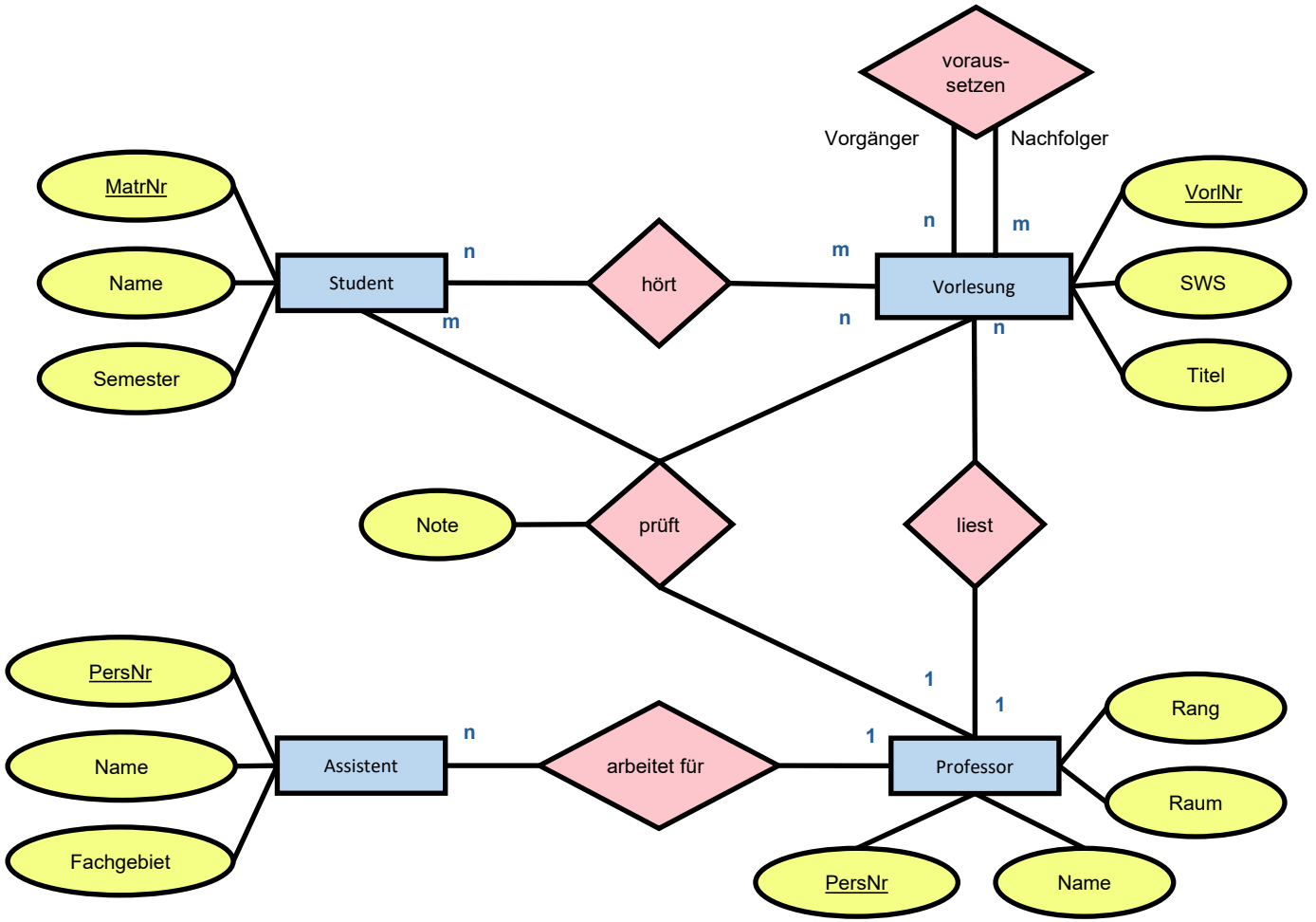
Prüft[MatrNr] ⊆ Student[MatrNr]
Prüft[VorlNr] ⊆ Vorlesung[VorlNr]
Prüft[PersNr] ⊆ Professor[PersNr]

Beispiel : ER-Diagramm (Universität)



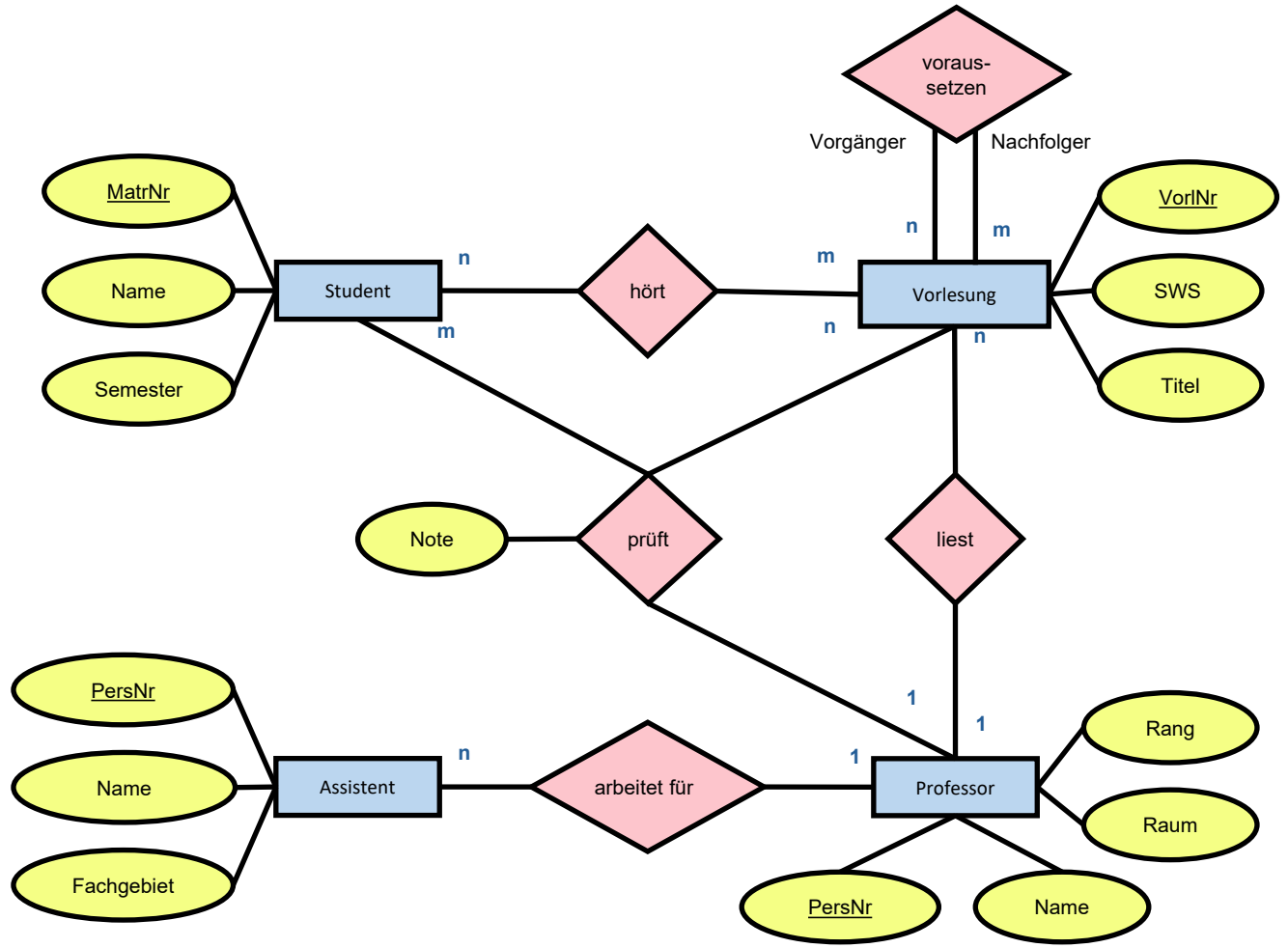
Beispiel für ein Datenbankschema

- **Relationsschema R:**
  - Student(MatrNr, Name, Semester)
  - Professor(PersNr, Name, Rang, Raum)
  - Vorlesung(VorlNr, Titel, SWS, gelesenVon)
  - Assistent(PersNr, Name, Fachgebiet, Boss)
  - Voraussetzen(Vorgänger, Nachfolger)
  - Hört(MatrNr, VorlNr)
  - Prüft(MatrNr, VorlNr, PersNr, Note)



Beispiel für ein Datenbankschema

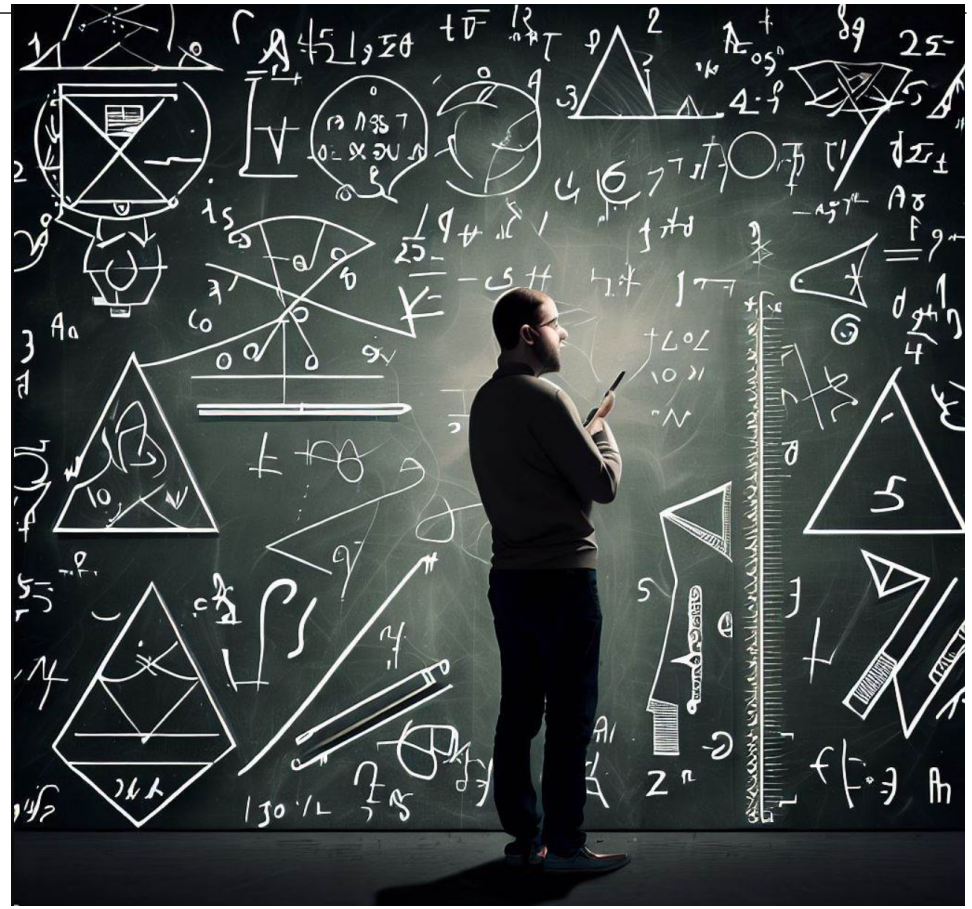
- **Interrelationale Abhängigkeiten:**
  - Vorlesung[gelesenVon] ⊆ Professor[PersNr]
  - Assistent[Boss] ⊆ Professor[PersNr]
  - Voraussetzen[Vorgänger] ⊆ Vorlesung[VorlNr]
  - Voraussetzen[Nachfolger] ⊆ Vorlesung[VorlNr]
  - Hört[MatrNr] ⊆ Student[MatrNr]
  - Hört[VorlNr] ⊆ Vorlesung[VorlNr]
  - Prüft[MatrNr] ⊆ Student[MatrNr]
  - Prüft[VorlNr] ⊆ Vorlesung[VorlNr]
  - Prüft[PersNr] ⊆ Professor[PersNr]
  - Assistent[PersNr] ∩ Professor[PersNr] = ∅



3. Das Relationale Datenmodell

1. Das Datenmodell
2. Transformation von ER-Diagrammen in das Relationale Modell
3. **Relationale Algebra**
4. Relationaler Kalkül

Was ist eine Algebra?



Definition:
- Mathematisches System
- Kombination von Symbolen und Regeln
- Darstellung und Manipulation abstrakter Objekte

Komponenten einer Algebra:
- Menge von Elementen (z.B., Zahlen, Vektoren, Matrizen)
- Operationen (z.B., Addition, Multiplikation, Subtraktion)
- Axiome (z.B., Assoziativität, Kommutativität, Distributivität)

Anwendungen:
- Mathematik (z.B., Lineare Algebra, Boolesche Algebra)
- Informatik (z.B., Relationale Algebra in Datenbanken)
- Physik und Ingenieurwissenschaften (z.B., Tensoralgebra, Lie-Algebra)

Warum Relationale Algebra?

- Teilung der Aufgaben (erste Vorlesung: Schichtenbildung!):
  - Verwendung der Algebra: z.B., Entwurf von Abfragesprachen mit relationalen Algebraoperatoren
  - Implementierung der Algebra
    - Effiziente Algorithmen und Datenstrukturen für Algebraoperationen
      - Zugriffsmethoden (z.B., B-Bäume, Hash-Indizes, Bitmap-Indizes)
      - Abfrageoptimierung (Kostenabschätzung, Auswahl der besten Pläne)
      - Abfrageausführung (z.B., geschachtelte Schleifenverknüpfungen, Hash-Verknüpfungen, Sortierungs-Zusammenführungsverknüpfungen)
      - Nebenläufigkeitskontrolle (Datenkonsistenz und Integrität)
      - Wiederherstellung (z.B., Protokollierung und Kontrollpunkte)

Relationale Algebra

- Eine Algebra ist allgemein gegeben durch
  - Menge von Objekten (Wertebereich)
  - Operationen zur Verknüpfung von Objekten

- Relationale Algebra
  - *Relationen* als Wertebereich
  - *relationale Operationen*, die Relationen als Argumente haben und wiederum Relationen als Resultate liefern

- Anfragen
  - Formulierung von Anfragen als Ausdrücke der relationalen Algebra, d.h. durch (rekursive) Anwendung von Operationen auf Relationen. Die Bearbeitung der Anfragen ist durch tatsächliche Auswertung der Operationen auf den Relationen einer Datenbank möglich.
  - Maß für die Ausdruckskraft einer Anfragesprache:
    - Sprache L heißt relational vollständig: jeder Ausdruck der Relationenalgebra kann in L simuliert werden kann (es gibt Ausdrücke in L, mit dem selben Ergebnis)
    - Eine Sprache L heißt streng relational vollständig: jeder Ausdruck der Relationenalgebra kann durch einen einzigen Ausdruck von L simuliert

Sechs Grundoperationen der Relationalen Algebra

Für die relationale Algebra gibt es sechs Grundoperationen, aus denen sich alle anderen Operationen nachbilden lassen:

- Vereinigung
- Differenz
- Kartesisches Produkt
- Selektion
- Projektion
- Umbenennung

Unsere Aufgabe im folgenden:

- Identifikation der Voraussetzungen für die Anwendung der Operatoren hat
- Identifizieren was die Operatoren mit den Tupeln **und** dem Schema der Relationen machen

Sechs Grundoperationen - Vereinigung

- *Vereinigung* $R \cup S$
  - Die beiden Relationenschema $R(A_1, \ldots, A_k)$ und $S(A_1, \ldots, A_k)$ müssen die gleichen Attribute besitzen. Dann ist $R \cup S$ definiert als die mengentheoretische Vereinigung:

$$R \cup S = \{t \mid t \in R \vee t \in S\}$$

  - Das neue Relationenschema ist $Q(A_1, \ldots, A_k)$. Dabei ist Q ein neuer Relationenname.
  - Anschaulich: Packe alle Tupel zusammen in eine Relation (lösche dabei Duplikate)
  - Beispiel: $\{(a, 0), (a, 1), (b, 1)\} \cup \{(a, 2), (b, 1)\} = \{(a, 0), (a, 1), (a, 2), (b, 1)\}$

Sechs Grundoperationen - Differenz

- *Differenz* $R - S$
  - Die beiden Relationenschema $R(A_1, \ldots, A_k)$ und $S(A_1, \ldots, A_k)$ müssen die gleichen Attribute besitzen. Dann ist $R - S$ definiert als die mengentheoretische Differenz:

$$R - S = \{t \mid t \in R \wedge t \notin S\}$$

  - Das neue Relationenschema ist $Q(A_1, \ldots, A_k)$. Dabei ist Q ein neuer Relationenname.
  - Anschaulich: Nimm alle Tupel aus R heraus, die es auch in S gibt.
  - Beispiel: $\{(a, 0), (a, 1), (b, 1)\} - \{(a, 1), (b, 2)\} = \{(a, 0), (b, 1)\}$

Sechs Grundoperationen – Kartesisches Produkt

- *Kartesisches Produkt* $R \times S$ (Kreuzprodukt)
  - Seien r und s der Grad von R bzw. S. Die Attributmenge der Relationenschema $R(A_1, \ldots, A_r)$ und $S(B_1, \ldots, B_s)$ seien disjunkt.
    Dann ist $R \times S$ die Menge aller $(r+s)$-Tupel, deren erste r Komponenten ein Tupel in R und deren letzte s Komponenten ein Tupel in S bilden, d.h.

$$R \times S = \{(a_1, \ldots, a_r, b_1, \ldots, b_s) \mid (a_1, \ldots, a_r) \in R \wedge (b_1, \ldots, b_s) \in S\}$$

  - Das neue Relationenschema ist $Q(A_1, \ldots, A_r, B_1, \ldots, B_s)$. Dabei ist Q ein neuer Relationenname.
  - Für die Anzahl der Tupel gilt: $|R \times S| = |R| \cdot |S|$
  - Anschaulich: verknüpfe jedes Tupel aus R mit jedem Tupel aus S
  - Beispiel: $\{(a,0),(b,1)\} \times \{(c,h),(d,v)\} = \{(a,0,c,h),(a,0,d,v),(b,1,c,h),(b,1,d,v)\}$

Sechs Grundoperationen – Selektion

- *Selektion* $\sigma_{\boldsymbol{F}}(R)$
  - Mit der Selektion $\sigma_{\boldsymbol{F}}(R)$ werden diejenigen Tupel aus der Relation R mit *Relationenschema* $R(A_1, \ldots, A_k)$ ausgewählt, die eine durch die logische Formel $\boldsymbol{F}$ ausgedrückte Eigenschaft erfüllen.
  - Das Relationschema der neuen Relation ist $Q(A_1, \ldots, A_k)$. Dabei ist Q ein neuer Relationenname.
  - Die Formel $\boldsymbol{F}$ besteht aus
    - **Operanden:** Attributsnamen $A_1, \ldots, A_k$ oder Konstanten
    - **Vergleichsoperatoren:** $=, \neq, <, \leq, >, \geq$
    - **Boolesche Operatoren:** $\wedge, \vee, \neg$ (and, or, not)

- Beispiel (Universität):
  - Zur Bestimmung von $\sigma_{\boldsymbol{F}}(R)$ wird die Formel $\boldsymbol{F}$ für jedes Tupel t betrachtet. Jedes Attribut A in $\boldsymbol{F}$ wird durch den Wert $t.A$ ersetzt.
    - eine naive Auswertungsstrategie überprüft jedes Tupel; die Verwendung von Indexstrukturen ermöglicht hier in manchen Fällen Einsparungen.

$\sigma_{\text{Semester}>11}(\text{Student})$

MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12

Sechs Grundoperationen – Projektion

- *Projektion* $\pi_{A_{i_1},\ldots,A_{i_m}}(R)$
  - Die Projektion erlaubt es Spalten (Attribute) aus einer Relation mit Relationenschema $R(A_1, \ldots, A_k)$ auszuwählen.
  - Sei k der Grad von R und $A_{i_1}, \ldots, A_{i_m}$ eine Auswahl von m paarweise verschiedenen Attributen (oder entsprechenden Indizes) aus $A_1, \ldots, A_k$ .Dann gilt:

$$\pi_{A_{i_1},\ldots,A_{i_m}}(R) = \{(a_{i_1}, \ldots, a_{i_m}) \mid (a_1, \ldots, a_k) \in R\}$$

  - Das Relationschema der neuen Relation ist $Q(A_{i_1}, \ldots, A_{i_m})$. Dabei ist Q ein neuer Relationenname.
  - Beispiel:

π_{Rang} (Professor)
Rang
W3
W2

  - Die Anzahl der Tupel kann sich durch implizite Elimination von Duplikaten verringern, solange in der Projektion keine (vollständigen) Schlüssel enthalten sind.

Sechs Grundoperationen – Umbenennung

- *Umbenennung $\rho_S(R)$ oder $\rho_{A' \leftarrow A}(R)$*
  - Die Umbenennung kann sowohl Relationen (R nach S) als auch Attribute einer Relation (A nach A' in der Relation R) umbenennen
  - Sei $R(A_1, \ldots, A_k)$ ein Relationschema. Nach Anwendung von $\rho_S(R)$ ist das neue Relationenschema $S(A_1, \ldots, A_k)$. Dabei ist S ein neuer Relationenname.
  - Sei $R(A_1, \ldots, A, \ldots A_k)$ ein Relationschema. Nach Anwendung von $\rho_{A' \leftarrow A}(R)$ ist das neue Relationenschema $Q(A_1, \ldots, A', \ldots A_k)$ für einen neuen Relationennamen Q.

Beispiel Umbenennung

– Beispiel: Wir wollen die Voraussetzungen 2. Stufe (also die Voraussetzungen der Voraussetzungen) der Vorlesung 5216 bestimmen

$$\pi_V(\sigma_{N=\text{Vorgänger} \wedge \text{Nachfolger}=5216}(\rho_{V \leftarrow Vorgänger}(\rho_{N \leftarrow Nachfolger}(voraussetzen)) \times voraussetzen))$$

- Zwischenergebnis:

Q		voraussetzen	
V	N	Vorgänger	Nachfolger
5001	5041	5001	5041
…	…	…	…
5001	**5041**	**5041**	**5216**
…	…	…	…
5052	5259	5052	5259

- Ergebnis: 5001 ist Vorgänger von 5041, diese ist wiederum Vorgänger von 5216

Zur Übung - wir wiederholen:

- Vereinigung
- Differenz
- Kartesisches Produkt
- Selektion
- Projektion
- Umbenennung

Beispiele - Grundoperationen

- Zwei Relationen (R, S) und ihre Verknüpfung mit relationalen Operationen

R	A	B	C
	a	b	c
	d	a	f
	c	b	d

S	A	B	C
	b	g	a
	d	a	f

$R \cup S$	A	B	C
	a	b	c
	d	a	f
	c	b	d
	b	g	a

$R - S$	A	B	C
	a	b	c
	c	b	d

$R \times S$	R.A	R.B	R.C	S.A	S.B	S.C
	a	b	c	b	g	a
	a	b	c	d	a	f
	d	a	f	b	g	a
	d	a	f	d	a	f
	c	b	d	b	g	a
	c	b	d	d	a	f

$\pi_{A, C}(R)$	A	C
	a	c
	d	f
	c	d

$\sigma_{B = b}(R)$	A	B	C
	a	b	c
	c	b	d

- Mit den vorgestellten Grundoperationen lassen sich nun alle Operationen der relationalen Algebra (induktiv) definieren.

Die Ausdrücke der relationalen Algebra

Induktive Definition der Audrücke der relationalen Algebra:

- ein Basisausdruck ist:
  - eine konstante Relation c (für ein beliebiges Attribute A und einen Wert c $\in dom(A)$ ist die Relation $\{(c)\}$ mit dem Relationenschema $Q(A)$ mit einem neuen Relationennamen Q ein Ausdruck).
  - eine Relation R der Datenbank

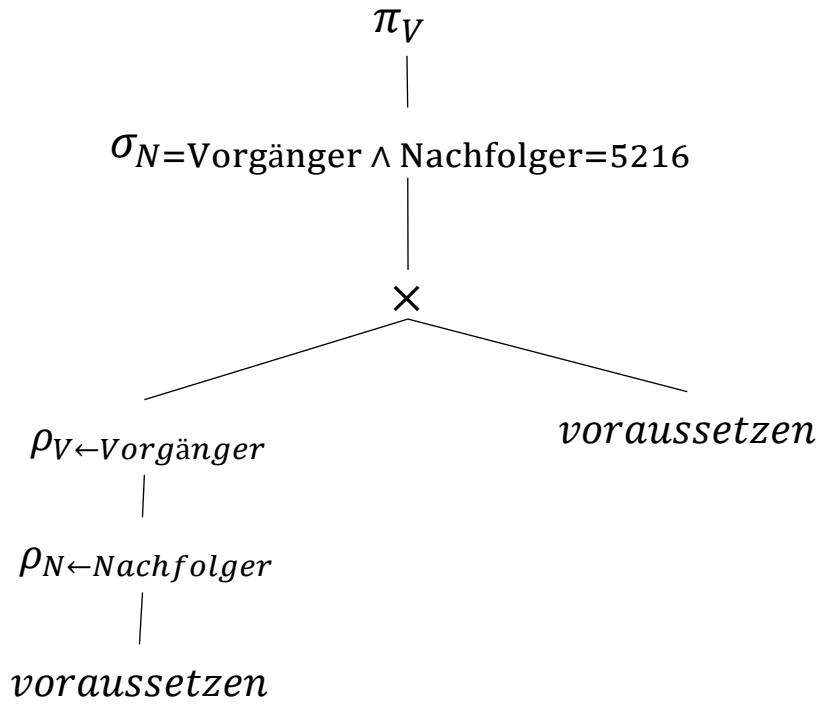
- seien E und F Relationenalgebra-Ausdrücke, A' und A Attribute, T, S und V wie bei den Algebra-Operatoren definiert. Dann sind folgendes auch gültige Ausdrücke, wobei die Anforderungen der Algebraoperatoren an die Relationenschemata gelten.
  - $E \cup F,\ E - F, E \times F, \sigma_T(E), \pi_S(E),\ \rho_V(E)$ und $\rho_{A' \leftarrow A}(E)$
  - Das sind alle (Minimalität)
- Mit der vorgestellten Algebra lassen sich beliebig komplexe Anfragen (Queries) formulieren, zum Beispiel die im Weiteren vorgestellten zusätzlichen Operationen

Operatorbäume

- Jeder Ausdruck einer Algebra lässt sich auch als *Operatorbaum* darstellen.
- Beispiel:

$$\pi_V(\ \sigma_{N=\text{Vorgänger} \wedge \text{Nachfolger}=5216}(\ \rho_{V \leftarrow Vorgänger}(\ \rho_{N \leftarrow Nachfolger}(voraussetzen)) \times voraussetzen))$$

  - Die Blätter enthalten *Relationen*.
  - Die inneren Knoten repräsentieren die *Operationen*.



Weitere Operationen – Durchschnitt

- Neben den sechs Grundoperationen existiert eine Reihe anderer nützlicher relationaler Operationen.
- *Durchschnitt* $R \cap S$
  - Die beiden Relationen R und S müssen das gleiche Schema besitzen. Dann ist $R \cap S$ definiert als die Schnittmenge der beiden Relationen, d.h.

$$R \cap S = \{t \mid t \in R \wedge t \in S\} \qquad = R - (R - S)$$

  - Beispiel: Finde die PersNr der W3-Professoren die mindestens eine Vorlesung halten.

$$\pi_{\text{PersNr}}\left(\rho_{\text{PersNr} \leftarrow \text{gelesenVon}}(\text{Vorlesungen})\right) \cap \pi_{\text{PersNr}}(\sigma_{\text{Rang=W3}}(\text{Professoren}))$$

    - Ergebnis

PersNr

Vorlesung

VorlNr	Titel	SWS	gelesenVon
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

Professor

PersNr	Name	Rang	Raum
2125	Sokrates	W3	226
2126	Russel	W3	232
2127	Kopernikus	W2	310
2133	Popper	W2	52
2134	Augustinus	W2	309
2136	Curie	W3	36
2137	Kant	W3	7

Weitere Operationen – Durchschnitt

- Neben den sechs Grundoperationen existiert eine Reihe anderer nützlicher relationaler Operationen.
- *Durchschnitt* $R \cap S$
  - Die beiden Relationen R und S müssen das gleiche Schema besitzen. Dann ist $R \cap S$ definiert als die Schnittmenge der beiden Relationen, d.h.

$$R \cap S = \{t \mid t \in R \wedge t \in S\} \qquad = R - (R - S)$$

  - Beispiel: Finde die PersNr der W3-Professoren die mindestens eine Vorlesung halten.

$$\pi_{\text{PersNr}}\left(\rho_{\text{PersNr} \leftarrow \text{gelesenVon}}(\text{Vorlesungen})\right) \cap \pi_{\text{PersNr}}(\sigma_{\text{Rang=W3}}(\text{Professoren}))$$

    - Ergebnis

PersNr
2125
2126
2137

Vorlesung

VorlNr	Titel	SWS	gelesenVon
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

Professor

PersNr	Name	Rang	Raum
2125	Sokrates	W3	226
2126	Russel	W3	232
2127	Kopernikus	W2	310
2133	Popper	W2	52
2134	Augustinus	W2	309
2136	Curie	W3	36
2137	Kant	W3	7

Weitere Operationen – Natürlicher Verbund (natural join)

- Natürlicher Verbund $R \bowtie S$
  - Idee: Selektiere aus dem Kreuzprodukt $R \times S$ nur zueinander „passende“ Tupel. Es sind nicht immer alle Tupel eines Kreuzprodukts relevant.

  - Sei $|\mathrm{Sch}(R) \cap \mathrm{Sch}(S)| = k$,d.h. R und S haben k gemeinsame Attribute. Wir schreiben die $m + k$ Attribute von R als $A_1, \ldots, A_m, B_1, \ldots, B_k$ und die $n + k$ Attribute von S als $B_1, \ldots, B_k, C_1, \ldots, C_n$. Weiter sind $D_1, \ldots, D_k$ neue Attributenamen. Dann ist

$$R \bowtie S = \pi_{A_1,\ldots,A_m,B_1,\ldots,B_k,C_1,\ldots,C_n}\left(\sigma_{D_1=B_1 \wedge \ldots \wedge D_k=B_k}\left(\rho_{B_1 \leftarrow D_1}(\ldots(\rho_{B_k \leftarrow D_k}(R))) \times S\right)\right)$$

  - Beispiel: Zugriff auf die Informationen einer m:n-Beziehung, wie zum Beispiel

$$\mathrm{Student} \bowtie \mathrm{hört} \bowtie \mathrm{Vorlesung}$$

Weitere Operationen – Natürlicher Verbund (natural join) (2)

– Beispiel: $\text{Student} \bowtie \text{hört} \bowtie \text{Vorlesung}$

Student		
MatrNr	Name	Sem.
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
27550	Carnap	8
…	…	…

hört	
MatrNr	VorlNr
25403	5022
26120	5001
27550	5001
27550	4052
…	…

Vorlesung			
VorlNr	Titel	SWS	gelesenVon
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5049	Mäeutik	2	2125
4052	Logik	4	2125
…	…	…	

$(\text{Student} \bowtie \text{hört}) \bowtie \text{Vorlesung}$						
MatrNr	Name	Semester	VorlNr	Titel	SWS	gelesenVon
25403	Jonas	12	5022	Glaube und Wissen	2	2134
26120	Fichte	10	5001	Grundzüge	4	2137
27550	Carnap	8	5001	Grundzüge	4	2137
…	…	…	…	…	…	…

Weitere Operationen – Natürlicher Verbund (natural join) (3)

- Der natürliche Join ist sowohl **assoziativ**, d.h.

$$(\text{Student} \bowtie \text{hört}) \bowtie \text{Vorlesung} = \text{Student} \bowtie (\text{hört} \bowtie \text{Vorlesung}),$$

  als auch **kommutativ** bis auf Vertauschung der Attributreihenfolge, zum Beispiel ist

$$\text{Student} \bowtie \text{hören} = \text{hören} \bowtie \text{Student}.$$

- Da der natürliche Join nur auf Attributen mit gleichen Attributnamen arbeitet, kann es notwendig sein Attribute umzubenennen.
  - Wollen wir die Relationen *Vorlesung* und *Professor* miteinander verbinden, müssen wir eines der Attribute *Vorlesung.gelesenVon* oder *Professor.PersNr* umbenennen.

$$\text{Vorlesung} \bowtie \rho_{\text{gelesenVon} \leftarrow \text{PersNr}}(\text{Professor})$$

- Wird der natürliche Join auf zwei Relationen ohne gleiche Attributnamen angewandt, dann entspricht des Ergebnis dem Ergebnis des kartesischen Produktes der beiden Relationen.

Weitere Operationen – Theta-Join

- Der Theta-Join $R \bowtie_\theta S$:
  - Idee: Selektiere aus dem Kreuzprodukt $R \times S$ nur zueinander „passende“ Tupel. „Passend“ wird hierbei durch die Formel θ festgelegt.
  - Seien $\mathrm{Sch}(R) = A_1, \ldots, A_n$ und $\mathrm{Sch}(S) = B_1, \ldots, B_m$ und θ eine Formel über $A_1, \ldots, A_n, B_1, \ldots, B_m$ .Dann ist

$$R \bowtie_\theta S = \sigma_\theta(R \times S)$$

  - Doppelte Attribute werden nicht aussortiert – d.h., müssen vorher umbenannt werden! Wir können dies implizit verlangen indem wir den Relationennamen mit angegeben.
  - Beispiel:

$$\text{Student} \bowtie_{\text{S.MatrNr=h.MatrNr} \wedge \text{ S.Semester>9}} \text{hört}$$

steht für

$$\text{Student} \bowtie_{\text{MatrNr=}ma \wedge \text{ Semester>9}} \rho_{\text{ma} \leftarrow MatrNr}(hört)$$

Student. MatrNr	Name	Semester	hört. MatrNr	VorlNr
25403	Jonas	12	25403	5022
26120	Fichte	10	26120	5001
…	…	…		…

Weitere Operationen – Äußere Join-Operatoren

- Im Gegensatz zum natürlichen Join bleiben bei äußeren Join-Operatoren auch Tupel erhalten, die keinen "Joinpartner" gefunden haben.
  - left outer join $⟕$: Die Tupel der linken Relation bleiben erhalten.
  - right outer join $⟖$: Die Tupel der rechten Relation bleiben erhalten.
  - (full) outer join $⟗$: Die Tupel beider Relationen bleiben erhalten.

- Darüber hinaus gibt es die Semi-Join-Operatoren $\ltimes$ und $\rtimes$. Der linke Semi-Join von R mit L – in Zeichen $L \ltimes R$ – ist definiert als

$$L \ltimes R = \pi_{\mathcal{L}}(L \bowtie R),$$

  wobei $\mathcal{L}$ die Menge der Attribute von L ist.

Der rechte Semi-Join $(L \rtimes R)$ analog dazu (mit $\mathcal{L}$ dann Menge der Attribute von R)

Join-Operatoren – Beispiele

natural join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

$\bowtie$

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

$=$

Resultat				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1

left outer join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

$\bowtie$

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

$=$

Resultat				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_2	b_2	c_2	-	-

full outer join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

$\bowtie$

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

$=$

Resultat				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_2	b_2	c_2	-	-
-	-	c_3	d_2	e_2

left semi-join L with R

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

$\ltimes$

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

$=$

Resultat		
A	B	C
a_1	b_1	c_1

3. Das Relationale Datenmodell

1. Das Datenmodell
2. Transformation von ER-Diagrammen in das Relationale Modell
3. Relationale Algebra
4. **Relationaler Kalkül**

Relationaler Kalkül

- **Relationale Algebra**
  - Algebra: Menge mit Operationen (hier: Relationen und relationale Operatoren).
  - Anfragen werden als Operationen auf den Relationen einer Datenbank ausgedrückt.
  - Die Antwortmenge wird durch die tatsächliche Anwendung der Operationen berechnet.
  - *konstruktives* (**prozedurales**) Vorgehen: Es wird angegeben, **WIE** eine Anfrage zu bearbeiten ist

- **Relationenkalkül(e)**
  - Kalkül: Logischer Formalismus zum Ableiten von Ergebnissen.
  - Anfragen werden durch Formeln der Prädikatenlogik erster Stufe ausgedrückt.
  - Die Antwortmenge enthält genau diejenigen Tupel, welche die Formel erfüllen.
  - *deskriptives* (**deklaratives**) Vorgehen: Vorgabe des **WAS**, aber nicht des WIE einer Anfrage
  - Für den Relationenkalkül unterscheiden wir zwei Arten: **Tupel- und Bereichskalkül**

- **Äquivalenz**: Ausdrücke der Relationenalgebra, sichere Ausdrücke des Tupelkalküls und sichere Ausdrücke des Bereichskalküls äquivalent in ihrer Mächtigkeit.
  - Ein Ausdruck des Tupelkalküls bzw. des Bereichskalküls heißt sicher, wenn sich aus der syntaktischen Struktur erkennen lässt, dass bei der Auswertung einer Anfrage nur endlich viele Tupelkandidaten überprüft werden müssen.
  - Das bedeutet, zu jedem Ausdruck der in der Relationenalgebra formuliert ist, gibt es einen Ausdruck im Relationenkalkül der die gleiche Rückgabe liefert, und andersherum.

Relationaler Kalkül

- Relationenkalkül: Prädikatenlogik erster Stufe.
  - *Syntax*: wie können gültige Ausdrücke des Kalküls gebildet werden. Dabei treten Konstanten, Variablen, Operatoren und Quantoren auf (symbolische Ebene).
  - *Semantik*: beschreibt die Bedeutung von syntaktisch korrekt gebildeten Ausdrücken. Dabei geht es um Relationen und die Erfüllung von Bedingungen (Bedeutungsebene).

- Zwei ähnliche Arten des Relationenkalküls:
  - *Tupelkalkül*: Variablen repräsentieren die Tupel einer Relation
  - *Domänenkalkül*: Variablen repräsentieren mögliche Werte eines Attributs

- Zunächst: Beispiele zu Relationenkalkül-Anfragen und Quantoren.
- Später: Formale Definition von Syntax und Semantik

Der Tupelkalkül – Beispiele

- Form der Ausdrücke im Tupelkalkül:

$$\{ t \mid F(t) \}$$

  Hier ist t eine *Tupelvariable* und F eine *Formel*.

  – die neue Relation besteht aus den Tupeln die die Formel F erfüllen

  – die Variable t muss in F eine *freie Variable* sein (= nicht quantifiziert, dazu gleich mehr)

  – Beispiel: Finde alle Professoren mit Rang ‘W3’.

$$\{ p \mid p \in \text{Professor} \wedge p.\text{Rang} = 'W3' \}$$

    - Auswertung:
      - p wird an jedes Tupel der Relation Professor gebunden
      - dann wird für jedes Tupel die $\text{Rang} = 'W3'$ Bedingung überprüft

Der Tupelkalkül – Beispiele

- Form einer Aunfrage im Tupelkalkül:

$$\{ t \mid F(t) \}$$

  Hier ist t eine *Tupelvariable* und F eine *Formel*.

  – die neue Relation besteht aus den Tupeln die die Formel F erfüllen

  – die Variable t muss in F eine *freie Variable* sein (= nicht quantifiziert, dazu gleich mehr)

  – Beispiel: Finde alle Professoren mit Rang ‘W3’.

$$\{ p \mid p \in \text{Professor} \wedge p.\text{Rang} = 'W3' \}$$

    - Auswertung:
      - p wird an jedes Tupel der Relation Professor gebunden
      - dann wird für jedes Tupel die $\text{Rang} = 'W3'$ Bedingung überprüft

Der Tupelkalkül – Quantifizierung

- Existenz- und Allquantoren $\exists, \forall$
  - $\exists t \in R\ \big(P(t)\big)$: es existiert ein t in R, sodass $P(t)$ gilt.
  - $\forall t \in R\ \big(P(t)\big)$: für alle t aus R gilt $P(t)$.
  - hierbei muss t in P eine freie Variable sein

  - Beispiel: Finde die Studenten, die mindestens eine Vorlesung bei der Professorin Curie hören.

    s ist in dem folgenden Prädikat die einzige freie Tupelvariable!

$$\{ s \mid s \in \text{Student} \ \wedge\ \exists h \in \text{hört}$$
$$(s.\text{MatrNr} = h.\text{MatrNr} \ \wedge\ \exists v \in \text{Vorlesung}$$
$$(h.\text{VorlNr} = v.\text{VorlNr} \ \wedge\ \exists p \in \text{Professor}$$
$$(p.\text{PersNr} = v.\text{gelesenVon} \ \wedge p.\text{Name} = \text{'Curie'}\ ))) \}$$

Der Tupelkalkül – Quantifizierung

- Existenz- und Allquantoren $\exists, \forall, \nexists$

  – Beispiel: Finde die Studenten, die alle von Professor Sokrates (PersNr 2125) angebotenen Vorlesungen besuchen.

$$\{ s \mid s \in \text{Student} \wedge \forall v \in \text{Vorlesung}$$
$$(v.\text{gelesenVon} = 2125 \rightarrow \exists h \in \text{hören}$$
$$(h.\text{VorlNr} = v.\text{VorlNr} \wedge h.\text{MatrNr} = s.\text{MatrNr}))\}$$

„Finde die Studierenden, für die alle Vorlesungen v, die von 2125 gelesen werden, impliziert, dass eine Beziehung „hören“ existiert, für die die Vorlesungsnummer der Vorlesung v und die Matrikelnummer der Studierenden übereinstimmt. „

Auch hier ist s die einzige freie Variable des Prädikats.

$$F \rightarrow G := \neg F \vee G$$

Der Tupelkalkül – Konstruktion neuer Relationen

- Schema einer Tupelvariable:
  - Eine Tupelvariable t besitzt ein Schema $(A_1: D_1, \ldots, A_k: D_k)$

$$\{\, t \in (A_1: D_1, \ldots, A_k: D_k) \mid F(t) \,\}$$

  - ist das Schema $\mathrm{Sch}(t)$ aus dem Zusammenhang klar, so wird auf die Angabe verzichtet

$$\{\, p \mid p \in \mathrm{Professor} \wedge p.\mathrm{Rang} = \text{'}W3\text{'} \,\}$$

  - über das Schema können neue Tupel konstruiert werden (vgl. Projektion und Join):
    - PersNr aller Professoren mit $\mathrm{Rang} =$ 'W3'

$$\{\, t \in (\mathrm{PersNr}: \mathrm{String}) \mid \exists p \in \mathrm{Professor}\ (t.\mathrm{PersNr} = \mathrm{p}.\mathrm{PersNr} \wedge p.\mathrm{Rang} = \text{'}W3\text{'}) \,\}$$

    - Kurzschreibweise (Tupelkonstruktor, nächste Seite):

$$\{\, [p.\mathrm{PersNr}] \mid p \in \mathrm{Professor} \wedge p.\mathrm{Rang} = \text{'}W3\text{'} \,\}$$

Der Tupelkalkül – Konstruktion neuer Relationen

- Konstruktion neuer Tupel:
  - Tupelkonstruktor [...]: $\{ [t_1.A_1, \ldots, t_n.A_n] \mid P(t_1, \ldots, t_n) \}$
    - die Attribute $A_1, \ldots, A_n$ müssen in den Schemata der Relationen enthalten sein, an die die $t_1, \ldots, t_n$ gebunden werden
  - Beispiel (Projektion und Join):
    - Ordne jedem Professor (Name) den ihm zugeordneten Assistenten (PersNr) zu

$$\{ [p.\text{Name}, a.\text{PersNr}] \mid p \in \text{Professor} \wedge a \in \text{Assistent} \wedge p.\text{PersNr} = a.\text{Boss} \}$$

    - Kurzschreibweise für:

$$\{ t \in (\text{Name: String}, \text{PersNr: Int}) \mid \exists p \in \text{Professor}\ \exists a \in \text{Assistent}$$
$$(t.\text{Name} = p.\text{Name} \wedge t.\text{PersNr} = a.\text{PersNr} \wedge p.\text{PersNr} = a.\text{Boss} \}$$

Der Tupelkalkül – Formale Definition (Syntax)

- Syntax: **Tupelvariable** t ⇒ Formel $F(t)$ ⇒ Ausdruck $\{ t \mid F(t) \}$
  - Idee: Ein Ausdruck liefert als Ergebnis die Relation aller Tupel, die die Formel F erfüllen.
  - Konstruktion von gültigen Tupelkalkül-Ausdrücken
  - <u>Tupelvariablen:</u>
    - Eine Tupelvariable t besitzt ein Schema $\mathrm{Sch}(t) = (A_1 : D_1, \ldots, A_k : D_k)$

      $$\{ t \in (A_1 : D_1, \ldots, A_k : D_k) \mid F(t) \}$$

    - Das Schema $\mathrm{Sch}(t)$ ist oft implizit festgelegt durch
      - eine Formel der Form $t \in R$, oder
      - die Struktur des Tupelkonstruktors $[t_1.A_1, \ldots, t_n.A_n]$
    - Eine Tupelvariable kann *frei* oder *gebunden* auftreten (siehe Syntax: Formel)

Der Tupelkalkül – Formale Definition (Syntax)

- Syntax: Tupelvariable t ⇒ **Formel** $F(t)$ ⇒ Ausdruck $\{ t \mid F(t) \}$
  - Idee: Ein Ausdruck liefert als Ergebnis die Relation aller Tupel, die die Formel F erfüllen.
  - Die Grundbausteine der Formeln sind die Atome
  - Tupelvariablen kommen in Atomen grundsätzlich nur *frei* vor
  - Es gibt drei Arten von Atomen:
    - $t \in R$ ist ein Atom, wobei t eine Tupelvariable und R eine Relation ist
    - $t.A\ \theta\ s.B$ ist ein Atom, wobei t, s Tupelvariablen sind, A und B Attributnamen von t bzw. s und $\theta \in \{=, \neq, <, \leq, >, \geq\}$ ein Vergleichsoperator. $t.A$ und $t.B$ müssen bzg. θ vergleichbar sein
    - $t.A\ \theta\ c$, $c\ \theta\ t.A$ wobei im Unterschied zu oben c eine Konstante ist

Der Tupelkalkül – Formale Definition (Syntax)

- Syntax: Tupelvariable t ⇒ **Formel** $F(t)$ ⇒ Ausdruck $\{ t \mid F(t) \}$
  - Idee: Ein Ausdruck liefert als Ergebnis die Relation aller Tupel, die die Formel F erfüllen.
  - Der Aufbau von Formeln ist rekursiv definiert:
    - Atome: Jedes Atom ist eine Formel
    - Verknüpfungen: Seien F, G Formeln. Dann sind auch $\neg F$, $F \wedge G$ und $F \vee G$ Formeln
    - Quantoren: Sei F eine Formel in der t als *freie* Variable auftritt. Dann sind auch $\exists t \in R\ (F)$ und $\forall t \in R\ (F)$ Formeln. t ist in dieser Formel dann eine *gebundene* Variable
  - Beispiel: In der folgenden Formel ist s eine freie Variable, h eine gebundene.

$$s \in \text{Student} \wedge \exists h \in \text{hören}\ (h.\text{MatrNr} = s.\text{MatrNr})$$

Der Tupelkalkül – Formale Definition (Syntax)

- Syntax: Tupelvariable t ⇒ Formel $F(t)$ ⇒ **Ausdruck** $\{ t \mid F(t) \}$
  - Idee: Ein Ausdruck liefert als Ergebnis die Relation aller Tupel, die die Formel F erfüllen.
  - Ausdruck: Ein Ausdruck des Tupelkalküls hat die Form
    - $\{ t \mid F(t) \}$, t ist die einzige freie Tupelvariable in der Formel F

Der Tupelkalkül – Formale Definition (Semantik)

- Semantik: Tupelvariable t ⇒ Formel $F(t)$ ⇒ Ausdruck $\{ t \mid F(t) \}$
  - Idee: Interpretation der Tupelkalkül-Ausdrücke
  - Analoge drei Schritte:
    - Tupelvariablen ⇒ konkrete Tupel
    - Formeln (Atome, Operatoren & Quantoren) ⇒ true, false
    - Ausdrücke ⇒ Relationen

Der Tupelkalkül – Formale Definition (Semantik)

- Semantik: **Tupelvariable** t ⇒ Formel $F(t)$ ⇒ Ausdruck $\{ t \mid F(t) \}$
  - Idee: Interpretation der Tupelkalkül-Ausdrücke
  - Belegung von Tupelvariablen: Seien
    - t eine Tupelvariable mit Schema $\mathrm{Sch}(t) = (A_1 : D_1, \ldots, A_k : D_k)$,
    - $F(t)$ eine Formel, die (i.A. nicht nur) t als freie Tupelvariable enthält
    - $r \in D_1 \times \cdots \times D_k$ ein beliebiges Tupel (muss i.A. nicht zu geg. Relation gehören)

    Bei der Belegung von t mit r wird jedes freie Vorkommen von t in $F(t)$ durch r ersetzt:

    $$F(r|t)$$

  - Beispiel: Sei $F(t) = t \in \mathrm{Professor} \wedge t.\mathrm{Rang} = 'W3'$ mit $\mathrm{Sch}(t) = \mathrm{Sch}(\mathrm{Professor})$.

    Für $r_1 = (2125, 'Sokrates', 'W3', 226)$ ist

    $$F(r_1|t) = r_1 \in \mathrm{Professor} \wedge 'W3' = 'W3'$$

Der Tupelkalkül – Formale Definition (Semantik)

- Semantik: Tupelvariable t ⇒ **Formel** $F(t)$ ⇒ Ausdruck $\{ t \mid F(t) \}$
  - Idee: Interpretation der Tupelkalkül-Ausdrücke
  - Interpretation $I(F)$ einer Formel F (analog zu syntaktischem Aufbau):
    - Die Formel F darf keine freien Variablen mehr enthalten.
    - Atome haben dann nur noch zwei Formen:
      - $I(r \in R) = \mathbf{true}$,falls r in R enthalten ist; sonst $\mathbf{false}$
      - $I(c_1 \theta c_2) = \mathbf{true}$,falls c_1 in Beziehung θ zu c_2 steht (Bsp.: $3 < 7$)
  - Beispiel:

$$I\big((2125, \text{'Sokrates'}, \text{'}W3\text{'}, 226) \in \text{Professor}\big) = \mathbf{true}$$

$$I(\text{'}W3\text{'} = \text{'}W3\text{'}) = \mathbf{true}$$

Der Tupelkalkül – Formale Definition (Semantik)

- Semantik: Tupelvariable t ⇒ **Formel** $F(t)$ ⇒ Ausdruck $\{ t \mid F(t) \}$
  - Idee: Interpretation der Tupelkalkül-Ausdrücke
  - Interpretation $I(F)$ einer Formel F (analog zu syntaktischem Aufbau):
    - Logische Operatoren:
      - $I(\neg F) = \mathbf{true}$, falls $I(F) = \mathbf{false}$ ist und umgekehrt
      - $I(F_1 \wedge F_2) = \mathbf{true}$ genau dann, wenn $I(F_1) = I(F_2) = \mathbf{true}$
      - $I(F_1 \vee F_2) = \mathbf{true}$, falls mindestens eines von $I(F_1), I(F_2) = \mathbf{true}$ ist
  - Beispiel:

$$I\big((2125, \text{'Sokrates'}, \text{'W3'}, 226) \in \text{Professor} \wedge \text{'W3'} = \text{'W3'}\big) = \mathbf{true}$$

Der Tupelkalkül – Formale Definition (Semantik)

- Semantik: Tupelvariable t ⇒ **Formel** $F(t)$ ⇒ Ausdruck $\{ t \mid F(t) \}$
  - Idee: Interpretation der Tupelkalkül-Ausdrücke
  - Interpretation $I(F)$ einer Formel F (analog zu syntaktischem Aufbau):
    - Quantoren:
      - Zur Interpretation von $\exists t \in R\ (F(t))$ und $\forall t \in R\ (F(t))$ darf nur t in F frei sein
      - $I\left(\exists t \in R\ (F(t))\right) = \mathbf{true}$ gdw ein $r \in R$ existiert, sodass $I(F(r|t)) = \mathbf{true}$ ist
      - $I\left(\forall t \in R\ (F(t))\right) = \mathbf{true}$ gdw für alle $r \in R$ gilt: $I(F(r|t)) = \mathbf{true}$
  - Beispiel:

$$I(\forall t \in \text{Vorlesung}\ (t.\text{gelesenVon} = '2125')) = \mathbf{false}$$

Der Tupelkalkül – Formale Definition (Semantik)

- Semantik: Tupelvariable t ⇒ Formel $F(t)$ ⇒ **Ausdruck** $\{ t \mid F(t) \}$
  - Idee: Interpretation der Tupelkalkül-Ausdrücke
  - Interpretation eines Ausdrucks:

    Sei $E = \{t \mid F(t)\}$ oder $E = \{[t_1.A_1, \ldots, t_n.A_n] \mid F(t_1, \ldots, t_n)\}$ ein Ausdruck und $D_1 \times \cdots \times D_k$ das Schema von t bzw. $[t_1.A_1, \ldots, t_n.A_n]$.
    - t bzw. $t_1, \ldots, t_n$ dürfen die einzigen freien Variabeln in F sein.
    - Der Wert von E ist die Menge aller Tupel $r \in D_1 \times \cdots \times D_k$ für die gilt

$$I\big(F(r|t)\big) = \mathbf{true}$$

Der Domänenkalkül

- Im Unterschied zum Tupelkalkül: *Bereichsvariablen*
  - Ein Ausdruck mit k Bereichsvariablen $x_1 : D_1, \ldots, x_k : D_k$ hat die Form

$$\{x_1, \ldots, x_k \mid F(x_1, \ldots, x_k)\}$$

- Definitionen:
  - Atome:
    - $R(x_1, \ldots, x_k)$ Bedeutung: ist wahr, falls nach Belegung der x_i mit $u_i \in D_i$ das Tupel $(u_1, \ldots, u_k)$ in R enthalten ist
    - $x\ \theta\ y$ Bedeutung: ist wahr, falls nach Belegung x in der Beziehung θ zu y steht
  - Formeln: Analog zum Tupelkalkül, Quantoren $\exists, \forall$ beziehen sich auf Domänen
  - Ausdruck: siehe oben

Der Domänenkalkül

- Beispiele:
  - Finde MatNr und Name der Studenten, die mindestens eine Prüfung bei Professor Curie abgelegt haben.

$$\{ m, n \mid \exists s \, (Student(m, n, s) \; \wedge \exists v, p, g \, (prüfen(m, v, p, g)$$
$$\wedge \exists a, r, b \, (Professor(p, a, r, b) \wedge a = \text{'Curie'}))) \; \}$$

  - Operationen der Relationalen Algebra:

$$R \cup S = \{x_1, \ldots, x_n \mid R(x_1, \ldots, x_n) \vee S(x_1, \ldots, x_n)\}$$
$$R - S = \{x_1, \ldots, x_n \mid R(x_1, \ldots, x_n) \wedge \neg S(x_1, \ldots, x_n)\}$$
$$P \bowtie Q = \{x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+k}, \ldots, x_{n+k+m} \mid$$
$$P(x_1, \ldots, x_{n+k}) \wedge Q(x_{n+1}, \ldots, x_{n+k+m}) \}$$

Beispiele Tupelkalkül vs. Domänenkalkül

- MatNr und Name der Studenten, die mindestens eine Prüfung bei Professor Curie abgelegt haben.

- Tupelkalkül

$$\{ [s.\text{MatrNr}, s.\text{Name}] \mid s \in \text{Student} \\ \wedge \exists h \in \text{hört}\ (s.\text{MatrNr} = h.\text{MatrNr} \\ \wedge \exists v \in \text{Vorlesung}\ (h.\text{VorlNr} = v.\text{VorlNr} \\ \wedge \exists p \in \text{Professor}\ \ (p.\text{PersNr} = v.\text{gelesenVon} \\ \wedge p.\text{Name} = \text{'Curie'}\ ))) \}$$

- Domänenkalkül

$$\{ m, n \mid \exists s\ (Student(m, n, s)\ \wedge \exists v, p, g\ (prüfen(m, v, p, g) \\ \wedge \exists r, b\ (Professor(p, \text{'Curie'}, r, b))\ )\ ) \}$$

Kurzschreibweise

Beispiele Tupelkalkül vs. Domänenkalkül

- PersNr aller Professoren mit $\text{Rang} = \text{'W3'}$

- Tupelkalkül

$$\{ [p.\text{PersNr}] \mid p \in \text{Professor} \wedge p.\text{Rang} = 'W3' \}$$

- Domänenkalkül

$$\{p | \exists n, b\ (Professor(p, n, 'W3', b))\}$$

Beispiele Tupelkalkül vs. Domänenkalkül

- Ordne jedem Professor (Name) den ihm zugeordneten Assistenten (PersNr) zu

- Tupelkalkül

$$\{ [p.\text{Name}, a.\text{PersNr}] \mid p \in \text{Professor} \wedge a \in \text{Assistent} \wedge p.\text{PersNr} = a.\text{Boss} \}$$

- Domänenkalkül

$$\{ \text{np}, \text{pa} \mid \exists \text{p}, \text{r}, \text{b}, \text{na}, \text{f}(\text{Professor}(\text{p}, \text{np}, \text{r}, \text{b}) \wedge \text{Assistent}(\text{pa}, \text{na}, \text{f}, p)) \}$$

Folie 1