



Datenbanken und Informationssysteme

3. Das Relationale Datenmodell

Prof. Dr. Stefan Decker

📍 RWTH Aachen

Von Konzepten zu Tabellen: Die logische Datenbankstruktur

Agenda

1. **Grundlagen**
Relationen, Tupel, Attribute, Domänen, Schlüssel
2. **Transformation**
Vom ER-Modell zum relationalen Schema
3. **Relationale Algebra**
Operationen auf Tabellen
4. **Relationaler Kalkül**
Deklarative Anfragen

Lernziele

- ▶ Grundbegriffe des relationalen Modells sicher unterscheiden
- ▶ ER-Modelle in relationale Schemata übersetzen
- ▶ Grundoperationen der relationalen Algebra anwenden
- ▶ Kalkül und Bezug zu SQL fachlich einordnen

Rückblick & Einordnung: Wo stehen wir im Entwurfsprozess?

1. Rückblick: Konzeptioneller Entwurf

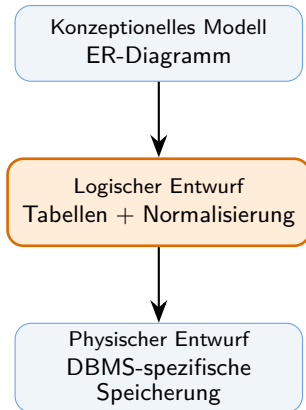
- ▶ Das **Entity-Relationship-Modell (ERM)** beschreibt die Datenstruktur implementierungsunabhängig.

2. Warum reicht das noch nicht?

- ▶ **Problem:** Ein ER-Modell ist für ein DBMS noch zu abstrakt.
- ▶ **Ziel:** Überführung in eine **logische Struktur**, die das Datenbanksystem direkt verarbeiten kann.

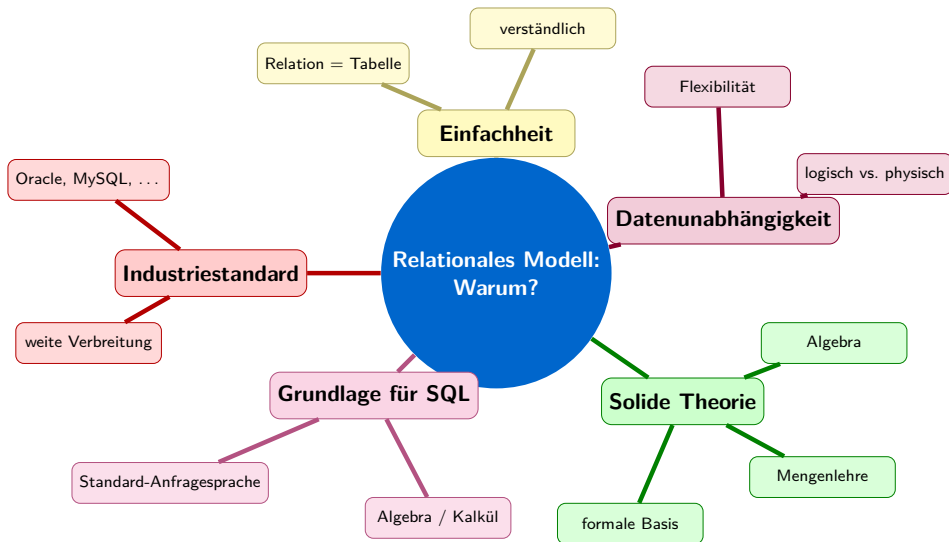
3. Die Lösung: Das relationale Modell

- ▶ Seit Codd (1970) die Brücke zwischen Fachmodell und Tabellenwelt



Edgar F. Codd: *A Relational Model of Data for Large Shared Data Banks*. Commun. ACM 13(6), 377–387 (1970).

Motivation: Warum das Relationale Modell?



Student

MatrNr <i>Integer</i>	Name <i>String</i>	#Semester <i>Integer</i>	Status <i>Statuswert</i>
1234	Michael	6	eingeschrieben
5678	Andrea	4	eingeschrieben
4711	Sabine	8	beurlaubt
815	Franz	12	exmatrikuliert

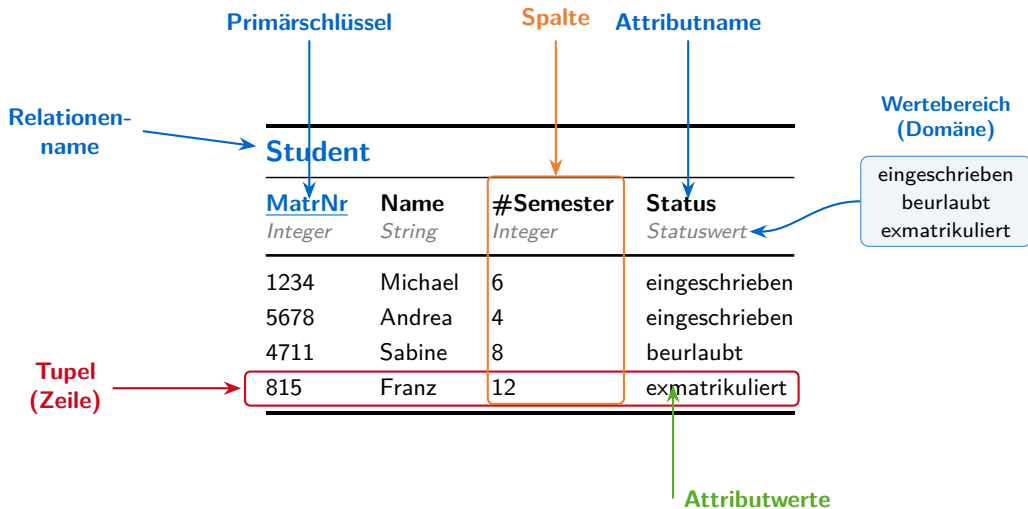
Beispielrelation: Alle Tupel besitzen dieselbe Attributstruktur; die konkreten Werte unterscheiden sich von Zeile zu Zeile.

Grundlagen: Die Relation und ihre Bausteine

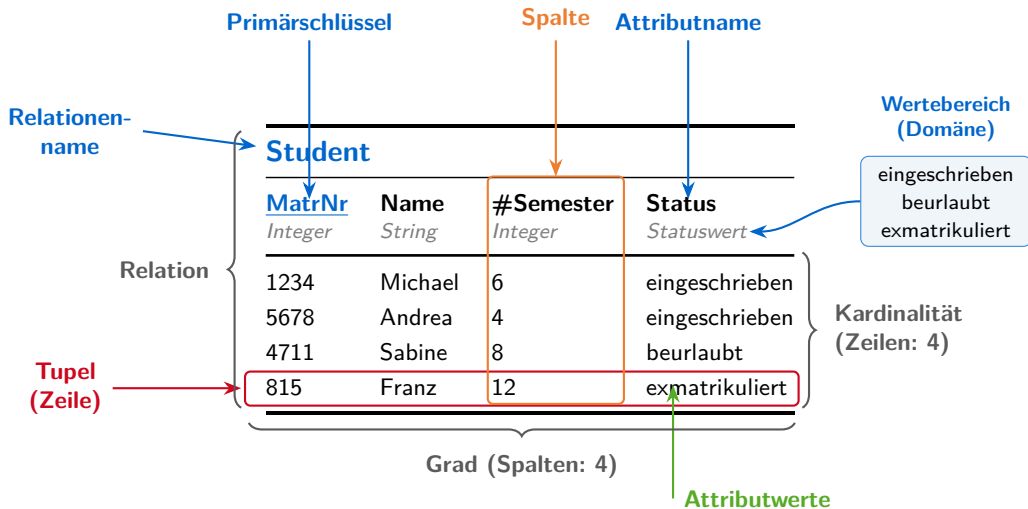
The diagram illustrates the components of a database relation. The relation is named **Student**. Its primary key is **MatrNr** (underlined in blue). The attributes are **MatrNr**, **Name**, **#Semester**, and **Status**. The data types are *Integer*, *String*, *Integer*, and *Statuswert* respectively. The value domain for **Status** is shown in a box on the right, containing the values **eingeschrieben**, **beurlaubt**, and **exmatrikuliert**.

Student			
<u>MatrNr</u>	Name	#Semester	Status
<i>Integer</i>	<i>String</i>	<i>Integer</i>	<i>Statuswert</i>
1234	Michael	6	eingeschrieben
5678	Andrea	4	eingeschrieben
4711	Sabine	8	beurlaubt
815	Franz	12	exmatrikuliert

Grundlagen: Die Relation und ihre Bausteine



Grundlagen: Die Relation und ihre Bausteine



Grundlagen: Relationsschema (Die Struktur der Tabelle)

Was ist ein Relationsschema?

- ▶ Es definiert die **Struktur** einer Relation, also ihren Bauplan.
- ▶ Es legt fest, welche Attribute gespeichert werden dürfen.
- ▶ Es ist relativ stabil, während sich die konkrete Relation laufend ändern kann.

Bestandteile und Notation

- ▶ **Relationsname**: eindeutiger Name der Tabelle
- ▶ **Attribute**: Liste der Attributnamen
- ▶ **Domänen**: Wertebereiche bzw. Datentypen je Attribut
- ▶ **Schlüssel**: ausgezeichnete Attribute zur Identifikation

Notation:

`Relationsname(Attribut1:Domäne1, ..., Attributn:Domänen)`

Beispiel:

`Student(MatrNr:INTEGER, Name:STRING, Semester:INTEGER, Status:STATUS)`

Grundlagen: Domänen, Relationsschemata und Relationen als Mengen

1. Domänen (Wertebereiche)

- ▶ Eine **Domäne** D ist eine Menge **atomarer** (nicht weiter zerlegbarer) Werte.
- ▶ Jedem Attribut A ist eine Domäne $\text{dom}(A)$ zugeordnet.
- ▶ In SQL approximiert durch Datentypen: INTEGER, VARCHAR(100), BOOLEAN, DATE.
- ▶ Aufzählung: STATUS = {eingeschrieben, beurlaubt, exmatrikuliert}.

2. Schema und Relation

- ▶ **Schema** $R(A_1, \dots, A_k)$ mit $\text{dom}(A_i) = D_i$ (*Intension*) (auch geschrieben als $R(A_1:D_1, \dots, A_k:D_k)$)
- ▶ **Relation** r zum Schema R : *endliche* Teilmenge (*Extension*):

$$r = \{t_1, \dots, t_n\} \subseteq D_1 \times \dots \times D_k.$$

- ▶ Kardinalität: $|r| = n$.

Beispiel: $R(\text{Farbe}, \text{Zahl})$ mit $D_{\text{Farbe}} = \{\text{rot}, \text{gelb}, \text{blau}\}$, $D_{\text{Zahl}} = \{1, 2\}$. $r_1 = D_{\text{Farbe}} \times D_{\text{Zahl}}$ ($|r_1| = 6$),
 $r_2 = \{(\text{rot}, 1), (\text{gelb}, 1), (\text{blau}, 1), (\text{blau}, 2)\}$ ($|r_2| = 4$), $r_3 = \emptyset$.

Grundlagen: Schema, Relation, Tupel

Formale Definitionen

- ▶ **Relationsschema** $R(A_1, \dots, A_k)$: Relationsname und Liste paarweise verschiedener Attribute.
- ▶ Der **Grad** ist die Anzahl der Attribute, also k .
- ▶ Eine **Relation** r über R ist eine endliche Menge von k -Tupeln über den zugehörigen Domänen:

$$r = \{t_1, \dots, t_n\} \subseteq D_1 \times \dots \times D_k$$

- ▶ Ein **Tupel** ist ein geordneter Wertvektor $t = (v_1, \dots, v_k)$ mit $v_i \in D_i$.

Zwei Sichtweisen auf Tupel

- ▶ **Geordnete Sicht**: Zugriff über Positionen $t[i] = v_i$.
- ▶ **Mapping-Sicht**: Zugriff über Namen $t(A_i) = v_i$.

Städte		
Name	Einwohner	Land
München	1.211.617	Bayern
Bremen	535.058	Bremen

Beispielinterpretation

Schema:

Städte(Name, Einwohner, Land)

Domänen: STRING, INTEGER, STRING

Ausprägung:

- ▶ (München, 1211617, Bayern)
- ▶ (Bremen, 535058, Bremen)

Drei Begriffe, drei Ebenen

- ▶ Eine **Relation** ist die aktuelle **Ausprägung** eines Relationsschemas, also die momentan gespeicherten Tupel.
- ▶ Ein **Datenbankschema** ist eine Menge von Relationsschemata, also $\mathcal{R} = \{R_1, \dots, R_k\}$.
- ▶ Eine **Datenbank** ist die Menge der aktuellen Relationen, also die Gesamtheit aller gegenwärtigen Ausprägungen.

Merke: Schema = Bauplan, Datenbank = aktueller Gesamtzustand.

1. Relationsschema

Student(MatrnNr, Name, Semester)



2. Relation

konkrete Tupel zu Student



3. Datenbank

Menge aktueller Relationen zu allen Schemata

Definition der Tupelprojektion

- ▶ Sei t ein Tupel über der Attributmenge $attr$.
- ▶ Sei $S \subseteq attr$ eine Teilmenge der Attribute von t .
- ▶ Die **Projektion** von t auf S , geschrieben als $t[S]$, enthält nur die Werte von t zu den Attributen in S .
- ▶ Für $S = \{A_{j_1}, \dots, A_{j_m}\}$ gilt:

$$t[S] = (t(A_{j_1}), t(A_{j_2}), \dots, t(A_{j_m}))$$

- ▶ Die Schreibweisen $t(A_j)$ und $t.A_j$ sind hier äquivalent.
- ▶ Die Domänen der gewählten Attribute bleiben erhalten.
- ▶ Anschaulich: Wir wählen bestimmte Spalten aus einer Zeile aus.
- ▶ **Zweck:** Die Tupelprojektion isoliert relevante Teilinformationen eines einzelnen Datensatzes.

Beispiel

Gegebenes Tupel:

$t = (26120, \text{Fichte},$
 $10, \text{eingeschrieben})$

Wähle $S = \{\text{Name}, \text{Semester}\}$

Dann ergibt sich:

$$t[S] = (\text{Fichte}, 10)$$

Name	Semester
Fichte	10

Motivation, Definition und Primärschlüssel

Warum brauchen wir Schlüssel?

- ▶ Jedes Tupel einer Relation muss eindeutig unterscheidbar sein.
- ▶ Nur so sind Referenzierung, Verknüpfungen und Updates zuverlässig möglich.

Schlüssel (Schlüsselkandidat):

Eine **minimale** Attributmenge S eines Schemas R , deren Werte ein Tupel **eindeutig** identifizieren.

Primärschlüssel:

Ein Schlüsselkandidat wird als bevorzugte Identifikation gewählt.

Formale Eigenschaften

Sei r eine Relation über R und $t[S]$ die Projektion eines Tupels t auf S .

1. Eindeutigkeit

$$\forall t_1, t_2 \in r : t_1 \neq t_2 \Rightarrow t_1[S] \neq t_2[S]$$

2. Minimalität

$$\forall T \subseteq S : T \text{ erfüllt 1.} \Rightarrow T = S$$

Interpretation:

- ▶ Kein Schlüsselwert kommt doppelt vor.
- ▶ Keine echte Teilmenge von S genügt.

Beispiel und Interpretation

Die Schlüsseleigenschaft hängt von der Semantik ab.

- ▶ In dieser kleinen Relation sind {PNr} und {Gehalt} zufällig beide eindeutig.
- ▶ Fachlich ist nur {PNr} ein **Schlüssel**, weil Personalnummern dauerhaft eindeutig sein sollen.
- ▶ {Gehalt} ist kein Schlüssel, da gleiche Gehälter in realen Daten vorkommen können.
- ▶ {PNr, Gehalt} ist zwar eindeutig, aber nicht minimal.

Schema: Personal(PNr, Gehalt)

Zum Schlüssel gehörigen Attribute werden im Schema unterstrichen.

Personal	
<u>PNr</u>	Gehalt
1	1.700 €
2	2.172 €
3	3.189 €
4	2.167 €

Transformation von ER-Diagrammen in das relationale Modell

Vom konzeptionellen Entwurf zu Tabellenstrukturen, Schlüsseln und Integritätsbedingungen.

Kapitelüberblick

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

Worauf wir jetzt achten

- ▶ Wie Semantik aus dem ER-Modell in Tabellen überführt wird
- ▶ Welche Rollen Schlüssel und Fremdschlüssel dabei spielen
- ▶ Warum Integritätsbedingungen für konsistente Datenbanken nötig sind

Datenabhängigkeiten (Integritätsbedingungen)

Warum brauchen wir Integritätsbedingungen?

- ▶ Objekte und Beziehungen aus dem ER-Modell werden als Relationen gespeichert.
- ▶ Ohne Regeln wären beliebige und damit auch inkonsistente Zustände möglich.
- ▶ Integritätsbedingungen beschränken die Datenbank auf fachlich sinnvolle Zustände.
- ▶ Ziel ist Datenkonsistenz und eine korrekte Abbildung der Semantik.

Arten von Abhängigkeiten

- ▶ **Intrarelationale Abhängigkeiten:**
Regeln innerhalb einer Relation
- ▶ Beispiel: Ein Schlüsselwert darf in `Student` nicht doppelt vorkommen.
- ▶ **Interrelationale Abhängigkeiten:**
Regeln zwischen Relationen
- ▶ Beispiel: Eine `MatrNr` in `Hört` muss in `Student` existieren.
- ▶ Weitere Fälle: Disjunktheit, Kardinalitätsregeln.

Konzept

- ▶ Intrarelationale Abhängigkeiten betreffen die Gültigkeit **einer einzelnen Relation**.
- ▶ Sie beziehen sich nicht auf andere Tabellen.
- ▶ Typisches Beispiel: die **Schlüsselabhängigkeit**.

Formale Definition

Sei X die Attributmenge eines Schemas R' und $\text{Rel}(X)$ die Menge aller Relationen über X .

Eine **intrarelationale Abhängigkeit** σ über X ist eine Abbildung

$$\sigma : \text{Rel}(X) \rightarrow \{\text{true}, \text{false}\}$$

- ▶ $\sigma(r) = \text{true}$: Die Relation r erfüllt die Regel.
- ▶ $\sigma(r) = \text{false}$: Die Relation r verletzt die Regel.
- ▶ Beispiel: Kein Schlüsselwert darf doppelt vorkommen.

1. Schlüsselabhängigkeit σ_K

Sei $K \subseteq X$. Die Schlüsselabhängigkeit σ_K prüft, ob K in einer gegebenen Relation r tatsächlich ein Schlüssel ist:

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

2. Erweitertes Relationsschema

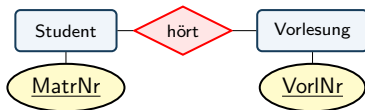
Ein Schema beschreibt nicht nur Struktur, sondern auch die zugehörigen Regeln.

$$R'' = (R', \Sigma_X)$$

- ▶ R' ist ein geordnetes Relationsschema.
- ▶ Σ_X ist eine Menge intrarelationaler Abhängigkeiten über den Attributen X von R' .
- ▶ Eine Relation r ist **gültig** bzw. **konsistent**, wenn für alle $\sigma \in \Sigma_X$ gilt: $\sigma(r) = \text{true}$.

1. Motivation: Vom ER-Modell zur Konsistenz

- ▶ Beziehungen aus dem ER-Modell führen zu Regeln **zwischen** Tabellen.
- ▶ Ziel ist Konsistenz über die gesamte Datenbank.



- ▶ Beispiel: Jeder Eintrag in Hört verweist auf existierende Studenten und Vorlesungen.

2. Referentielle Integrität

Szenario:

Student(MatrNr, ...), Hört(MatrNr, VorlNr)

Dann muss gelten:

$$\text{Hört}[\text{MatrNr}] \subseteq \text{Student}[\text{MatrNr}]$$

Bei Verletzung ist die Datenbank inkonsistent.

3. Formale Definition

Sei $\mathcal{R} = \{R_1, \dots, R_k\}$ eine Menge erweiterter Schemata.

$\text{Dat}(\mathcal{R})$ bezeichnet die Menge aller **Datenbankzustände** über $\mathcal{R}$, also aller Tupel $(r_1, \dots, r_k)$, bei denen r_i eine Relation zum Schema R_i ist.

Eine **interrelationale Abhängigkeit** ist eine Abbildung $\sigma : \text{Dat}(\mathcal{R}) \rightarrow \{\text{true}, \text{false}\}$ und bewertet somit ganze Datenbankzustände statt einzelner Relationen.

Interrelationale Abhängigkeiten: Inklusionsabhängigkeit

Rückbezug und Motivation

- ▶ Referentielle Integrität ist der wichtigste Spezialfall:

$$\text{Hört}[\text{MatrNr}] \subseteq \text{Student}[\text{MatrNr}]$$

- ▶ Bedeutung: Ein Wert in einer Tabelle muss auch in einer anderen Tabelle existieren.
- ▶ Ziel: Dieses Prinzip als allgemeinen Typ einer interrelationalen Abhängigkeit formulieren.

Definition

- ▶ **Notation:** $R_i[V] \subseteq R_j[W]$
- ▶ R_i, R_j : zwei Relationsschemata mit Attributmengen X_i und X_j
- ▶ $V \subseteq X_i, W \subseteq X_j$ mit $|V| = |W|$
- ▶ $u : V \rightarrow W$: bijektive Zuordnung der betrachteten Attribute

Bedingung

$$\sigma_{R_i[V] \subseteq R_j[W]} : \text{Dat}(\mathcal{R}) \rightarrow \{\text{true}, \text{false}\}$$

$$\{t[V] \mid t \in r_i\} \subseteq \{s[W] \circ u \mid s \in r_j\}$$

$s[W] \circ u$ zieht die Projektion auf W über u auf die Attributmenge V zurück.

Beispiel: $\text{Hört}[\text{MatrNr}] \subseteq \text{Student}[\text{MatrNr}]$. Die Menge der MatrNr-Werte in *Hört* muss also Teilmenge der MatrNr-Werte in *Student* sein.

Interrelationale Abhängigkeiten: Exklusionsabhängigkeit

Motivation und Konzept

- ▶ Exklusionsabhängigkeiten sind das Gegenstück zu Inklusionsabhängigkeiten.
- ▶ Sie fordern disjunkte Projektionen, also $R_i[V] \cap R_j[W] = \emptyset$.
- ▶ Typischer ER-Kontext: disjunkte Spezialisierungen, etwa Professor **oder** Assistent.

Definition

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

R_i, R_j : zwei Relationsschemata mit $V \subseteq X_i, W \subseteq X_j, |V| = |W|$

$u : V \rightarrow W$: bijektive Zuordnung gleichartiger Attribute

Bedingung

$\sigma_{R_i[V] \cap R_j[W] = \emptyset} : \text{Dat}(\mathcal{R}) \rightarrow \{\text{true}, \text{false}\}$

$\{t[V] \mid t \in r_i\} \cap \{s[W] \circ u \mid s \in r_j\} = \emptyset$

$s[W] \circ u$ macht beide Projektionen vergleichbar.

Beispiel: $\text{Assistent}[\text{PersNr}] \cap \text{Professor}[\text{PersNr}] = \emptyset$. Damit wird disjunkte isA-Semantik relational abgesichert.

Das vollständige Datenbankschema

Definition des Datenbankschemas

$$D = (\mathcal{R}, \Sigma_{\mathcal{R}})$$

- ▶ $\mathcal{R}$: Menge erweiterter Relationsschemata R_i''
- ▶ Jedes $R_i'' = (R_i', \Sigma_{X_i})$ enthält Struktur **und** interne Regeln einer Tabelle
- ▶ $\Sigma_{\mathcal{R}}$: Menge der Regeln zwischen Tabellen
- ▶ Beispiele: Inklusions- und Exklusionsabhängigkeiten

Gültigkeit einer Instanz

$$d = \{r_1, \dots, r_k\}$$

- ▶ d ist **gültig**, wenn alle Regeln aus D erfüllt sind.
- ▶ **Intra-Konsistenz**: Jede Tabelle erfüllt ihre internen Regeln Σ_{X_i} .
- ▶ **Inter-Konsistenz**: Die gesamte Datenbank erfüllt die Regeln aus $\Sigma_{\mathcal{R}}$.

Merke: Ein Datenbankzustand ist nur dann gültig, wenn Tabellen intern und zwischen Tabellen konsistent sind.

Nächster Schritt: Fremdschlüssel setzen Inklusionsabhängigkeiten praktisch um.

Zweck

- ▶ **Problem:** Wie wird die Inklusionsabhängigkeit $R_1[F] \subseteq R_2[K]$ im DBMS praktisch umgesetzt?
- ▶ **Lösung:** Durch das Konzept des **Fremdschlüssels** (FK).
- ▶ **Ziel:** Referentielle Integrität deklarieren und automatisch erzwingen.

Formale Definition

- ▶ Kontext: $D = (\mathcal{R}, \Sigma_{\mathcal{R}})$, Relationen R_1, R_2
- ▶ K : Primärschlüssel von R_2
- ▶ F ist Fremdschlüssel in R_1 auf R_2 , wenn:
 - ▶ **Domänenkompatibilität:** Attribute in F und K passen typ- und bereichsmäßig zusammen
 - ▶ **Inklusionsabhängigkeit:** $R_1[F] \subseteq R_2[K]$ ist als Regel definiert

Fremdschlüssel: Bedeutung und Beispiel

Bedeutung und Konsistenz erzwingung

- ▶ Ein Wert in den FK-Spalten F muss entweder einem existierenden PK-Wert in der referenzierten Tabelle entsprechen oder NULL sein, falls erlaubt.
- ▶ Das DBMS überwacht und erzwingt diese Regel bei Änderungen, um Dateninkonsistenzen zu verhindern.

Vorlesung			
Typ	Attribut	Rolle	Bedeutung
int	VorlNr	PK	Vorlesungsnummer
string	Titel		Titel
int	SWS		Semesterwochenstunden
int	gelesenVon	FK	Professor.PersNr



FK-
Beziehung

Professor			
Typ	Attribut	Rolle	Bedeutung
int	PersNr	PK	Personalnummer
string	Name		Name
string	Rang		Rang
string	Raum		Raum

Abhängigkeit (aus $\Sigma_{\mathcal{R}}$): $\text{Vorlesung}[\text{gelesenVon}] \subseteq \text{Professor}[\text{PersNr}]$

Erklärung: Vorlesung.gelesenVon ist FK auf Professor.PersNr. Jeder Wert außer NULL muss auf eine existierende Personalnummer verweisen.

Transformation: Vom ER-Modell zum relationalen Modell

Der Weg vom Konzept zur Tabelle

- ▶ Nach der Definition des relationalen Modells folgt nun die systematische Übersetzung eines ER-Diagramms in Tabellen.
- ▶ DBMS benötigen explizite Tabellenstrukturen und klar formulierte Regeln.

Gegenüberstellung der Modelle

Merkmal	ER-Modell	Relationales Modell
Grundbaustein	Entity-Typ, Beziehungstyp	Relation (Tabelle)
Beschreibt	Objekte und Beziehungen	Strukturierte Daten in Tabellen
Regeln durch	Kardinalitäten, Attribute, isA	PK, FK, Inklusion, Exklusion
Ziel	Verständnis der Miniwelt	Grundlage für DBMS

Konzeptuelles Modell
(ER-Diagramm)

Logisches Modell
(relationale Tabellen)

Physisches Modell
(DBMS-spezifisch)

Ausblick: Als Nächstes folgen die Abbildungsregeln für konkrete ER-Konstrukte.

Kernaufgaben der ER-zu-relational-Transformation

1. Struktur-Abbildung

- ▶ ER-Strukturelemente werden in relationale Konstrukte überführt.
- ▶ Entity-Typen → Tabellen
- ▶ Attribute → Spalten mit Domänen
- ▶ Beziehungstypen → Tabellen und/oder Fremdschlüssel

2. Regel-Abbildung und Semantik-Erhalt

- ▶ Bedeutung und Konsistenzregeln bleiben im relationalen Modell erhalten.
- ▶ ER-Schlüssel → Primärschlüssel
- ▶ Kardinalitäten → Fremdschlüssel und Abhängigkeiten
- ▶ Weitere Regeln → Nicht-Null-Bedingungen, Disjunktheit

Nächster Schritt: Abbildungsregeln für konkrete ER-Konstrukte.

ER → Relational: Agenda der Abbildungsregeln

Im Folgenden behandeln wir die wichtigsten Abbildungsregeln für ER-Konstrukte.

Agenda: Abbildung von ...

1. Entitätstypen und Attribute

- ▶ Einfache Entity-Typen → Tabellen
- ▶ Attribute → Spalten
- ▶ Umgang mit zusammengesetzten Attributen
- ▶ Umgang mit mehrwertigen Attributen

2. Binäre Beziehungen

- ▶ $n:m$ -Beziehungen
- ▶ $1:n$ -Beziehungen
- ▶ $1:1$ -Beziehungen

3. Rekursive Beziehungen

- ▶ Beziehung eines Typs mit sich selbst
- ▶ Abhängig von $1:1$, $1:n$, $n:m$

4. Generalisierung / Spezialisierung

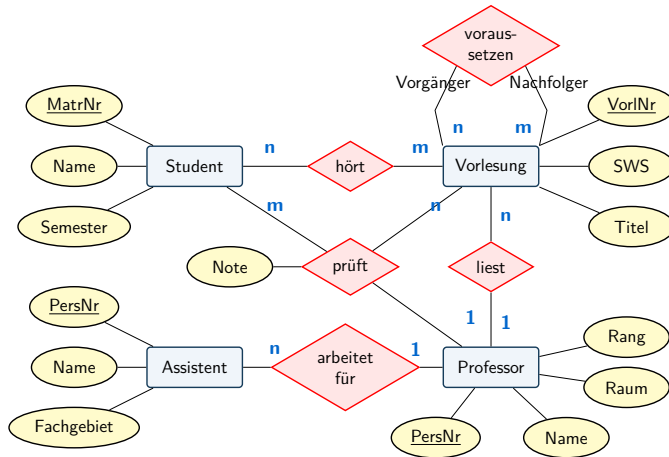
- ▶ Grundlegende Strategien
- ▶ Constraints: Total/Partiell, Disjunkt/Überlappend

5. Weitere Konstrukte

- ▶ Schwache Entity-Typen
- ▶ n -äre Beziehungen

Zu jedem Konstrukt betrachten wir Abbildungsregel und resultierendes relationales Schema mit PK, FK und Abhängigkeiten.

Durchgehendes Beispiel: ER-Diagramm „Universität“



Agenda: Abbildung von ...

1. Entitätstypen und Attribute

- ▶ Einfache Entity-Typen → Tabellen
- ▶ Attribute → Spalten
- ▶ Umgang mit zusammengesetzten Attributen
- ▶ Umgang mit mehrwertigen Attributen

2. Binäre Beziehungen

- ▶ $n:m$ -Beziehungen
- ▶ $1:n$ -Beziehungen
- ▶ $1:1$ -Beziehungen

3. Rekursive Beziehungen

Beziehung eines Typs mit sich selbst
Abhängig von $1:1$, $1:n$, $n:m$

4. Generalisierung / Spezialisierung

- ▶ Grundlegende Strategien
- ▶ Constraints: Total/Partiell, Disjunkt/Überlappend

5. Weitere Konstrukte

- ▶ Schwache Entity-Typen
- ▶ n -äre Beziehungen

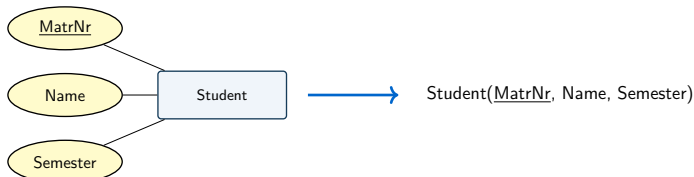
Wir starten mit den einfachsten Bausteinen: Entitäten, Attributen und den grundlegenden binären Beziehungen.

Abbildungsregel 1.a: Starke Entity-Typen und einfache Attribute

Regel

- ▶ Jeder **starke Entity-Typ** E des ER-Modells wird zu einer eigenen Relation R .
- ▶ **Einfache Attribute** von E werden zu Spalten von R .
- ▶ Der **Primärschlüssel** des Entity-Typs wird zum Primärschlüssel der Relation.

Beispiel

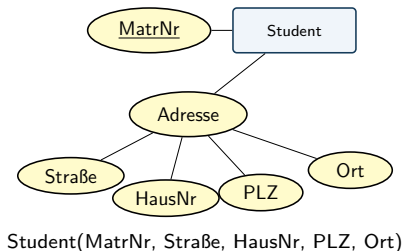


Die einfachste Abbildung lautet also: **Entity-Typ** → **Tabelle** und **Attribut** → **Spalte**.

Abbildungsregel 1.b: Zusammengesetzte und mehrwertige Attribute

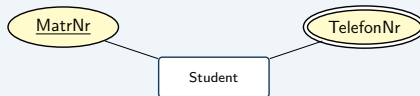
Zusammengesetzte Attribute

- ▶ Die **Teilattribute** werden zu eigenen Spalten der Hauptrelation.
- ▶ Beispiel: Adresse → Straße, HausNr, PLZ, Ort
- ▶ Nur wenn das Attribut nie zerlegt ausgewertet wird, kann man es zusammengefasst speichern.



Mehrwertige Attribute

- ▶ Ein **mehrwertiges Attribut** wird in eine eigene Relation ausgelagert.
- ▶ Diese Relation enthält den Primärschlüssel der Hauptrelation als Fremdschlüssel.
- ▶ Der Schlüssel besteht meist aus Primärschlüssel und Attributwert.

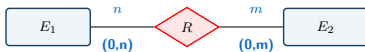


Relational:

`Student(MatNr, ...)`
`Student_Telefon(MatNr, TelefonNr)`
`Student_Telefon[MatrNr] ⊆ Student[MatrNr]`

Abbildungsregel 2.a: $n:m$ -Beziehungen

Abbildungsregel



- ▶ Für den Beziehungstyp R wird eine **eigene Relation** erzeugt.
- ▶ Sie enthält PK_1 aus T_1 und PK_2 aus T_2 als **Fremdschlüssel**.
- ▶ Beziehungsattribute von R werden ebenfalls übernommen.
- ▶ Primärschlüssel: meist beide Fremdschlüssel zusammen.

Konsequenz

Zuordnung: $E_1 \rightarrow T_1(\underline{PK_1})$, $E_2 \rightarrow T_2(\underline{PK_2})$, $R \rightarrow T_R(\underline{FK_1}, \underline{FK_2}, \dots)$

$$T_R[FK_1] \subseteq T_1[PK_1]$$
$$T_R[FK_2] \subseteq T_2[PK_2]$$

Beispiel: hört



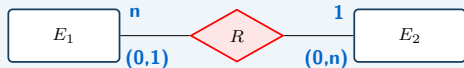
$Student(\underline{MatrNr})$
 $Vorlesung(\underline{VorlNr})$
 $Hört(\underline{MatrNr}, \underline{VorlNr})$

$Hört[MatrNr] \subseteq Student[MatrNr]$

$Hört[VorlNr] \subseteq Vorlesung[VorlNr]$

Abbildungsregel 2.b: 1:n-Beziehungen als eigene Relation

Ausgangspunkt



Die Objekte von E_1 nehmen wegen der 1:n-Struktur nur einmal an R teil.

Formale Möglichkeit

- ▶ Auch hier wird R zu einer Relation T_R .
- ▶ T_R enthält beide Fremdschlüssel und ggf. Beziehungsattribute.
- ▶ Der FK auf E_1 ist dann zugleich der Primärschlüssel von R .
- ▶ Die referenziellen Abhängigkeiten sind wie beim $n:m$ -Fall.

Bewertung

- ▶ Diese Lösung ist meist **ineffizient**, weil eine zusätzliche Tabelle und zusätzliche Joins entstehen.
- ▶ Verhindert viele leere Felder (Nullwerte), wenn die Beziehung nur selten genutzt wird.
- ▶ Im Normalfall ist es besser, die Beziehung direkt in die Relation der n -Seite zu integrieren.

Merke: Formal möglich, aber meist nicht Standard.

Abbildungsregel 2.b: 1:n-Beziehungen als Standardmethode

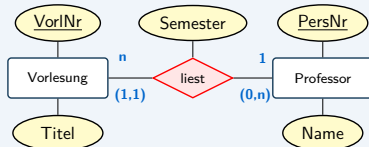
Regel

- ▶ Die Relation der n -Seite erhält einen Fremdschlüssel auf den Primärschlüssel der 1-Seite.
- ▶ Beziehungsattribute werden ebenfalls in die Relation der n -Seite übernommen.
- ▶ Ist die Teilnahme der n -Seite verpflichtend, wird dieser Fremdschlüssel **NOT NULL**.

Bewertung

- ▶ **Vorteil:** keine zusätzliche Relation und damit weniger Join-Aufwand.
- ▶ **Nachteil:** Bei optionaler Teilnahme können in der n -Seite Nullwerte entstehen.
- ▶ Insgesamt ist dies die **übliche Standardabbildung** für 1:n-Beziehungen.

Beispiel: liest



Professor(PersNr, Name)
Vorlesung(VorlNr, Titel, Semester, ProfessorPersNr)

FK: ProfessorPersNr
übernimmt Professor.PersNr.

Vorlesung[ProfessorPersNr] $\subseteq$ Professor[PersNr]

Abbildungsregel 2.c: 1:1-Beziehungen

Methodenüberblick

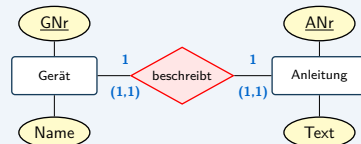
1. **Eigene Relation für R :** formal korrekt, aber meist ineffizient.
2. **Fremdschlüssel in einer Tabelle:** wie bei 1:n; sinnvoll, wenn eine Seite optional ist.
3. **Verschmelzen:** beide Entity-Typen und die Beziehung werden zu einer Relation zusammengefasst.

Faustregel: Verschmelzen ist attraktiv, wenn beide Seiten total an der Beziehung teilnehmen und nicht unabhängig voneinander benötigt werden.

Bewertung

- ▶ Eigene Relation: zusätzlicher Join ohne großen Mehrwert
- ▶ Fremdschlüssel: flexibel, aber asymmetrisch
- ▶ Verschmelzen: effizient, solange keine unerwünschten Nullwerte entstehen

Beispiel: Verschmelzen



Gerät(GNr, Name, ANr, Text)

Hinweis: ANr bleibt bei echter 1:1-Semantik eindeutig und ist damit ein alternativer Schlüssel.

Übergang: Rekursive Beziehungen und Hierarchien

Nach den binären Standardfällen folgen jetzt die strukturell anspruchsvolleren ER-Konstrukte.

Jetzt stehen Rollen, Selbstbezüge und Hierarchien im Modell im Mittelpunkt.

Was jetzt neu wird

- ▶ **Rekursive Beziehungen:** Ein Entity-Typ tritt in unterschiedlichen Rollen mehrfach auf.
- ▶ Diese Rollen müssen im relationalen Schema getrennt abgebildet werden.
- ▶ **Generalisierung / Spezialisierung:** isA-Hierarchien erlauben mehrere sinnvolle Abbildungsstrategien.

Nächster Fokus

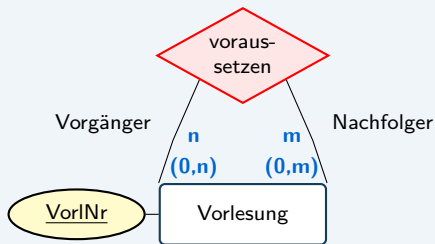
1. Rekursive Beziehungen
2. Generalisierung / Spezialisierung
3. Danach: schwache Entity-Typen und n -äre Beziehungen

Abbildungsregel 3: Rekursive Beziehungen

Regel

- ▶ Rekursive Beziehungen folgen denselben Grundregeln wie nicht-rekursive Beziehungen; entscheidend ist die **Kardinalität**.
- ▶ Bei **rekursiven 1:1- oder 1:n-Beziehungen** genügt ein Fremdschlüssel **auf dieselbe Relation**.
- ▶ Bei **rekursiven n:m-Beziehungen** entsteht eine eigene Relation mit **zwei rollenbezogenen Fremdschlüsseln** auf die Ursprungstabelle.
- ▶ Beziehungsattribute werden jeweils in die gewählte relationale Form übernommen.

Beispiel: voraussetzen



Relationale Form

Voraussetzen(Vorgänger, Nachfolger)

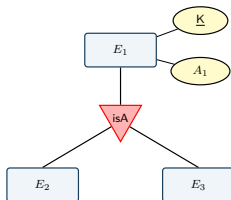
Voraussetzen[Vorgänger] $\subseteq$ Vorlesung[VorlNr]

Voraussetzen[Nachfolger] $\subseteq$ Vorlesung[VorlNr]

Rollen: Vorgänger und Nachfolger sind zwei Rollen derselben Relation Vorlesung.

Abbildungsregel 4: Generalisierung / Spezialisierung

isA-Struktur



Noch offen: total/partiell und
disjunkt/überlappend

Constraints bestimmen die Strategie

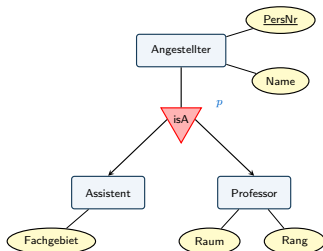
- ▶ **Partiell** oder **total**: Muss jedes Objekt der Generalisierung in einer Spezialisierung vorkommen?
- ▶ **Disjunkt** oder **überlappend**: Darf ein Objekt gleichzeitig mehreren Spezialisierungen angehören?
- ▶ Diese Constraints entscheiden, welche relationale Abbildungsstrategie günstig ist.

Zentrale Idee

- ▶ Spezialisierungen übernehmen den Schlüssel der Generalisierung.
- ▶ Bei **disjunkten** Spezialisierungen kommen Exklusionsbedingungen zwischen Untertypen hinzu.
- ▶ Bei **totalen** Spezialisierungen kann man ggf. auf eine eigene Generalisierungsrelation verzichten.
- ▶ Außer in günstigen Spezialfällen erfordern vollständige Abfragen meist **Joins**.

isA-Abbildung: Strategie 1 - Tabelle pro Typ

ER-Beispiel



Relationale Form

Angestellter(PersNr, Name)

Professor(PersNr, Rang, Raum)

Assistent(PersNr, Fachgebiet)

$\text{Professor}[\text{PersNr}] \subseteq \text{Angestellter}[\text{PersNr}]$

$\text{Assistent}[\text{PersNr}] \subseteq \text{Angestellter}[\text{PersNr}]$

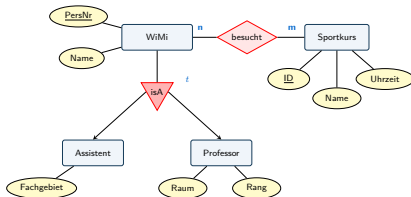
$\text{Professor}[\text{PersNr}] \cap \text{Assistent}[\text{PersNr}] = \emptyset$

Eigenschaften

- ▶ Standardmethode: eine Tabelle für die Generalisierung, je eine weitere für jede Spezialisierung
- ▶ Primärschlüssel der Spezialisierung ist gleichzeitig Fremdschlüssel auf die Generalisierung
- ▶ Flexibel für partielle sowie disjunkte oder überlappende Spezialisierungen
- ▶ Nachteil: vollständige Informationen erfordern häufig einen Join mit der Generalisierung

isA-Abbildung: Strategie 2 - Nur Tabellen für Spezialisierungen

ER-Beispiel



Relationale Form

Professor(PersNr, Name, Raum, Rang)

Assistent(PersNr, Name, Fachgebiet)

Sportkurs(ID, Name, Uhrzeit)

Besucht(PersNr, SportkursID)

$\text{Professor}[\text{PersNr}] \cap \text{Assistent}[\text{PersNr}] = \emptyset$

$\text{Besucht}[\text{SportkursID}] \subseteq \text{Sportkurs}[\text{ID}]$

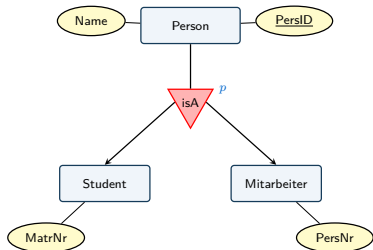
$\text{Besucht}[\text{PersNr}] \subseteq \text{Assistent}[\text{PersNr}] \cup \text{Professor}[\text{PersNr}]$

Einordnung

- ▶ Regel: Attribute des Obertyps werden in die Spezisierungstabellen kopiert; eine eigene Obertyp-Tabelle entfällt.
- ▶ Eignung: nur sinnvoll bei totaler und disjunkter Spezialisierung.
- ▶ Vorteil: Für vollständige Objekte ist kein Join mit der Generalisierung nötig.
- ▶ Nachteil: Redundante gemeinsame Attribute; ungeeignet für partielle oder überlappende Spezialisierungen.

isA-Abbildung: Strategie 3 - Eine einzige Tabelle

ER-Beispiel



Beispiel: **partiell** und **disjunkt**

Relationale Form

Person(PersID, Name, MatrNr, PersNr, Typ)

Typ $\in \{\text{Person, Student, Mitarbeiter}\}$

ODER

Person(PersID, Name, MatrNr, PersNr, istStudent, istMitarbeiter)

Regel und Einordnung

Regel: Eine einzige Tabelle enthält Obertyp- und Spezialattribute aller Typen.

- ▶ Disjunkt: Ein Typ-Attribut genügt, z.B. Typ $\in \{\text{Person, Student, Mitarbeiter}\}$.
- ▶ Überlappend und auch disjunkt: Boolesche Indikatorattribute pro Entity-Typ.

Benötigte Constraints: Konsistenzregeln für Typ- oder Indikatorattribute.

Eignung: Kein Join, aber viele NULL-Werte und zusätzliche Konsistenzregeln.

isA-Strategien im Überblick

Vergleich der drei Abbildungsstrategien

Strategie	Tabellen	Joins?	NULLs?	Geeignet für	Vorteil	Nachteil
Tabelle pro Typ	1+ Anzahl Spez.	Ja, oft	Wenig	Alle Fälle: partiell/total, dis- junkt/überlappend	Flexibel, normalisiert	Mehr Joins
Nur Tabellen für Spezialisierungen	Anzahl Spez.	Nein	Wenig, intern	Nur total + disjunkt	Kein Join zum Obertyp	Redundanz
Eine einzige Tabelle	1	Nein	Viele möglich	Vor allem disjunkt; partiell oder total	Keine Joins	Viele NULLs, Logik

Merksatz: Es gibt keine universell beste Strategie. Die Wahl hängt von den semantischen Randbedingungen der Spezialisierung und vom gewünschten Abfrage- und Speicherverhalten ab.

Abbildungsregel 5: Schwache Entity-Typen

Ausgangspunkt und Regel

Ausgangspunkt

- ▶ schwacher Entity-Typ E_2 mit Teilschlüssel K_2
- ▶ besitzender Entity-Typ E_1 mit Schlüssel K_1
- ▶ identifizierende Beziehung zwischen E_1 und E_2

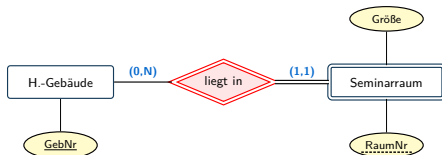
Abbildungsregel

$$E_2(\underline{K_1}, \underline{K_2}, \dots)$$

$$E_2[K_1] \subseteq E_1[K_1]$$

- ▶ Primärschlüssel von E_2 : Kombination aus K_1 und K_2
- ▶ K_1 bleibt zugleich Fremdschlüssel auf E_1
- ▶ Die identifizierende Beziehung braucht keine eigene Tabelle.

Beispiel: Hörsaalgebäude und Seminarraum

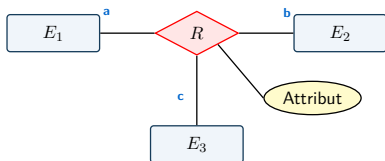


Seminarraum(GebNr, RaumNr, Größe)

Seminarraum[GebNr] $\subseteq$ H.-Gebäude[GebNr]

Abbildungsregel 5: n-stellige Beziehungen

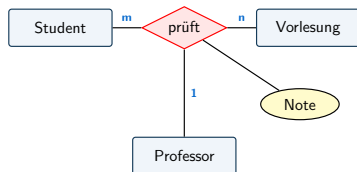
ER-Konstrukt und Regel



Regel

- ▶ Für R wird eine eigene, separate Relation erstellt.
- ▶ Sie enthält Fremdschlüssel auf die Primärschlüssel aller beteiligten Entity-Typen ($E_1, E_2, E_3, \dots$) sowie die Beziehungsattribute von R .
- ▶ Primärschlüssel: Fremdschlüssel der Seiten mit Kardinalität $n, m, \dots$
- ▶ Inklusionsbedingungen: eine für jeden Fremdschlüssel.

Beispiel: ternäre Beziehung prüft



Relation:

Prüft(MatrNr, VorlNr, PersNr, Note)

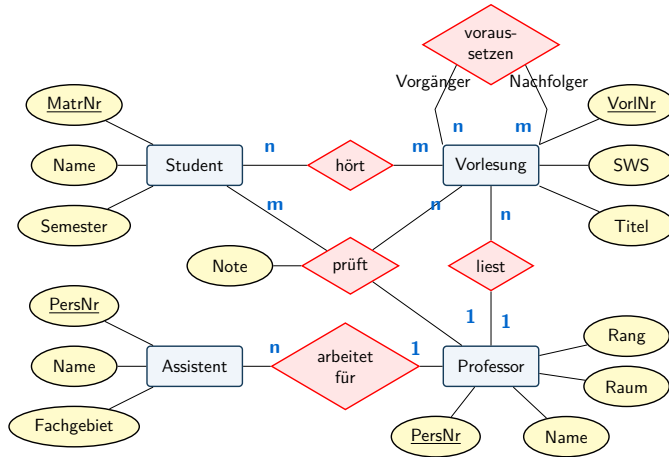
Inklusionsbedingungen:

Prüft[MatrNr] $\subseteq$ Student[MatrNr]

Prüft[VorlNr] $\subseteq$ Vorlesung[VorlNr]

Prüft[PersNr] $\subseteq$ Professor[PersNr]

Laufendes Beispiel: ER-Diagramm Universität



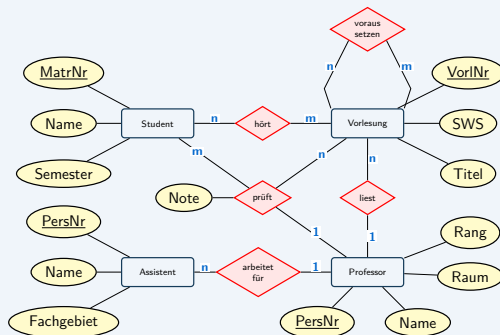
Beispiel für ein Datenbankschema zum ER-Diagramm

Universität

Relationsschema $\mathcal{R}$

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)

ER-Referenz



Interrelationale Abhängigkeiten zum ER-Diagramm

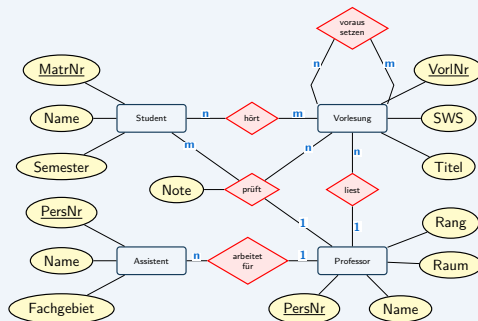
Universität

Fremdschlüssel- und Integritätsbedingungen

$\text{Vorlesung}[\text{gelesenVon}] \subseteq \text{Professor}[\text{PersNr}]$
 $\text{Assistent}[\text{Boss}] \subseteq \text{Professor}[\text{PersNr}]$
 $\text{Voraussetzen}[\text{Vorgänger}] \subseteq \text{Vorlesung}[\text{VorINr}]$
 $\text{Voraussetzen}[\text{Nachfolger}] \subseteq \text{Vorlesung}[\text{VorINr}]$
 $\text{Hört}[\text{MatrNr}] \subseteq \text{Student}[\text{MatrNr}]$
 $\text{Hört}[\text{VorINr}] \subseteq \text{Vorlesung}[\text{VorINr}]$
 $\text{Prüft}[\text{MatrNr}] \subseteq \text{Student}[\text{MatrNr}]$
 $\text{Prüft}[\text{VorINr}] \subseteq \text{Vorlesung}[\text{VorINr}]$
 $\text{Prüft}[\text{PersNr}] \subseteq \text{Professor}[\text{PersNr}]$

$$\text{Assistent}[\text{PersNr}] \cap \text{Professor}[\text{PersNr}] = \emptyset$$

ER-Referenz



3. Das Relationale Datenmodell

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

Nächster Schritt: Formale Anfragen über Relationen ausdrücken und systematisch umformen.

Agenda und Lernziele

Agenda

- ▶ Ausgangspunkt: Von Relationen zu Anfragen
- ▶ Relationale Algebra: „Wie?“
- ▶ Grundoperationen und algebraische Ausdrücke
- ▶ Abgeleitete Operationen, insbesondere Joins
- ▶ Relationaler Kalkül: Tupel- und Domänenkalkül
- ▶ Vergleich, Ausdrucksmächtigkeit und SQL-Bezug

Lernziele

- ▶ Die Rolle formaler Anfragesprachen erläutern
- ▶ Operatoren der relationalen Algebra anwenden
- ▶ Anfragen prozedural mit Algebra formulieren
- ▶ Tupel- und Domänenkalkül einordnen
- ▶ Algebra und Kalkül vergleichen
- ▶ Den Bezug zu SQL herstellen

Leitidee: Das relationale Modell beschreibt nicht nur Datenstrukturen, sondern auch formale Sprachen zur präzisen Formulierung gewünschter Ergebnisse.

Ausgangspunkt: Relationen und Anfragen

Unsere Datenbasis

Daten liegen in Relationen mit festem Schema vor.
Beispiel:

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)

Die Aufgabe: Gezielte Anfragen

Wie beantworten wir Fragen **präzise** und **eindeutig**?

- ▶ Welche Studierenden hören „Ethik“?
- ▶ Finde alle W3-Professoren.
- ▶ Gib Vorlesungstitel und Dozentennamen aus.
- ▶ Welche Studierenden sind mindestens im 5. Semester?

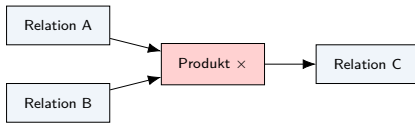
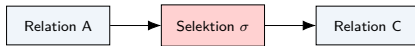
Die Werkzeuge: Formale Sprachen

- ▶ **Relationale Algebra**: beschreibt das **Wie** einer Anfrage.
- ▶ **Relationaler Kalkül**: beschreibt das **Was** einer Anfrage.
- ▶ Beide bilden die formale Grundlage von SQL.

Ansatz 1: Relationale Algebra - Das Wie

Was ist die relationale Algebra?

- ▶ Eine Algebra besteht aus **Objekten** und **Operationen**.
- ▶ Hier sind die Objekte **Relationen**.
- ▶ Die Operationen sind **relationale Operatoren** wie σ , π oder $\times$.
- ▶ Eingabe: eine oder zwei Relationen.
- ▶ Ausgabe: immer wieder eine neue Relation.



Das prozedurale Prinzip

- ▶ Die relationale Algebra beschreibt **Schritt für Schritt**, wie ein Ergebnis konstruiert wird.
- ▶ Eine Anfrage ist eine **Sequenz** oder ein **Baum** von Operationen.
- ▶ Der Fokus liegt auf dem **Weg** beziehungsweise Verfahren.
- ▶ Beim relationalen Kalkül liegt der Fokus dagegen auf dem gewünschten Ergebnis.

Nächste Schritte

- ▶ Anfragen werden aus wenigen Grundoperationen aufgebaut.
- ▶ Wir betrachten nun die **sechs fundamentalen Operatoren** im Detail.

Sechs Grundoperationen der Relationalen Algebra

Basis der relationalen Algebra

#	Operation	Symbol	Wirkung
1	Selektion	σ	Wählt Tupel anhand einer Bedingung aus.
2	Projektion	π	Wählt Attribute aus und entfernt Dubletten.
3	Vereinigung	$\cup$	Kombiniert Tupel zweier schema-kompatibler Relationen.
4	Differenz	$-$	Entfernt Tupel der zweiten Relation aus der ersten.
5	Kartesisches Produkt	$\times$	Bildet alle Tupel-Kombinationen aus zwei Relationen.
6	Umbenennung	ρ	Ändert Namen von Relationen oder Attributen.

Warum Grundoperationen? Weil sich wichtige weitere Operatoren wie Schnitt, Join oder Division aus diesen sechs Bausteinen ableiten lassen.

Grundoperation 1: Selektion (σ)

Notation und Wirkung

$$\sigma_G(R)$$

- ▶ σ steht für **Sigma**.
- ▶ Die Selektion wählt genau die Tupel aus R , die die Bedingung G erfüllen.
- ▶ G ist ein logischer Ausdruck mit Attributen, Konstanten, Vergleichen und Boolescher Logik.
- ▶ Das **Schema bleibt unverändert**; nur die Tupelmenge wird verkleinert.

Beispiel: Studierende ab dem 12. Semester

Anfrage: Finde alle Studierenden mit Semester > 11 .

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

Eingabe $R = \text{Student}$

MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10

Ausgabe Q

MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12

Beobachtung: Das Schema bleibt gleich; nur die Tupelmenge wird kleiner.

Grundoperation 2: Projektion (π)

Notation und Wirkung

$$\pi_{A_1, \dots, A_m}(R)$$

- ▶ π steht für **Pi**.
- ▶ Die Projektion wählt nur bestimmte Attribute aus einer Relation aus.
- ▶ Das Ergebnis erhält ein **neues Schema**, das genau aus diesen Attributen besteht.
- ▶ Wichtig: Dubletten werden entfernt, weil Relationen mengenbasiert sind.

Beispiel: Unterschiedliche Professorenränge

Anfrage: Welche unterschiedlichen Ränge gibt es bei Professoren?

$$P := \pi_{\text{Rang}}(\text{Professor})$$

Eingabe $R = \text{Professor}$

PersNr	Name	Rang	Raum
24002	Xenokrates	W3	226
25403	Jonas	W3	232
26120	Fichte	W2	310
27550	Carnap	W2	052

Ausgabe P

Rang
W3
W2

Beobachtung: Nur Rang bleibt; Dubletten entfallen.

Grundoperation 3: Vereinigung ($\cup$)

Notation und Wirkung

$$R \cup S$$

- ▶ R und S müssen **schema-kompatibel** sein.
- ▶ Das bedeutet insbesondere: gleiche Attribute in gleicher Struktur.
- ▶ Das Ergebnis enthält alle Tupel aus R oder S .
- ▶ Dubletten erscheinen nur einmal, da Relationen mengenbasiert sind.
- ▶ Das Schema bleibt dasselbe wie bei R und S .

Beispiel: Zwei Studierendengruppen

Anfrage: Welche Studierenden sind in Gruppe 1 oder Gruppe 2?

$$M := \text{Gruppe1} \cup \text{Gruppe2}$$

Gruppe 1

MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10

Gruppe 2

MatrNr	Name	Semester
26120	Fichte	10
25403	Jonas	12
26830	Aristoxenos	8

Ausgabe M

MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8

Beobachtung: Das Schema bleibt gleich; gelb markierte Tupel erscheinen in M nur einmal.

Grundoperation 4: Differenz (−)

Notation und Wirkung

$$R - S$$

- ▶ Auch hier müssen R und S **schema-kompatibel** sein.
- ▶ Das Ergebnis enthält genau die Tupel, die in R , aber **nicht** in S vorkommen.
- ▶ Das Schema bleibt dasselbe wie bei den Eingaberelationen.

Merksatz: Gesucht ist nicht $R \cap S$, sondern der Teil von R , der nach Abzug von S übrig bleibt.

Beispiel: Zwei Studierendengruppen

Anfrage: Welche Studierenden sind nur in Gruppe 1?

$$N := \text{Gruppe1} - \text{Gruppe2}$$

Gruppe 1

MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10

Gruppe 2

MatrNr	Name	Semester
26120	Fichte	10
25403	Jonas	12
26830	Aristoxenos	8

Ausgabe N

MatrNr	Name	Semester
24002	Xenokrates	18

Beobachtung: Nur Tupel aus Gruppe 1 bleiben; gelbe Tupel entfallen.

Grundoperation 5: Kartesisches Produkt ($\times$)

Notation und Wirkung

$$R \times S$$

- ▶ Jede Zeile aus R wird mit **jeder** Zeile aus S kombiniert.
- ▶ Das Schema enthält alle Attribute von R und zusätzlich alle Attribute von S .
- ▶ Die Größe des Ergebnisses ist $|R| \cdot |S|$.

Achtung: Das kartesische Produkt wächst sehr schnell. In Anfragen wird es daher oft nur als Zwischenschritt vor Selektion oder Join verwendet.

Beispiel: Studierende und Bier-Vorlieben

Anfrage: Bilde alle Kombinationen aus Studierenden und Bier-Vorlieben.

$$P := \text{Student} \times \text{magBier}$$

Student

MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10

magBier

IDNr	Sorte
25403	Kölsch
26120	Hefeweizen

Ausgabe P mit 3 Studierenden $\cdot$ 2 Bier-Vorlieben = 6 Tupeln

MatrNr	Name	Sem.	IDNr	Sorte
24002	Xenokrates	18	25403	Kölsch
24002	Xenokrates	18	26120	Hefeweizen
25403	Jonas	12	25403	Kölsch
25403	Jonas	12	26120	Hefeweizen
26120	Fichte	10	25403	Kölsch
26120	Fichte	10	26120	Hefeweizen

Beobachtung: Jede Zeile aus Student wird mit jeder Zeile aus magBier kombiniert.

Grundoperation 6: Umbenennung (ρ)

Notation, Funktion und Zweck

$$\rho_S(R) \quad \rho_{B \leftarrow A}(R)$$

- ▶ ρ steht für **Rho**.
- ▶ $\rho_S(R)$ benennt die **Relation** R in S um.
- ▶ $\rho_{B \leftarrow A}(R)$ benennt das **Attribut** A in B um.
- ▶ Die Tupel selbst bleiben dabei unverändert.
- ▶ Umbenennung vermeidet Namenskonflikte.



Frage:

Warum braucht man die Umbenennung?

Beispiel: Änderung Studenten

Eingabe $R = \text{Student}$: Relation Student

MatrNr	Name	Semester
24002	Xenokrates	18
...	...	...

$$\rho_{\text{Studierende}}(\text{Student})$$

Ausgabe: Relation Studierende

MatrNr	Name	Semester
24002	Xenokrates	18
...	...	...

$$\rho_{\text{IDNr} \leftarrow \text{MatrNr}}(\text{Studierende})$$

Ausgabe: Relation Studierende mit Attribut IDNr

IDNr	Name	Semester
24002	Xenokrates	18
...	...	...

Formale Definition: Algebraische Ausdrücke

Induktive Definition

1. Basisausdrücke

- ▶ Für ein Attribut A und $c \in \text{dom}(A)$ ist die konstante Relation $\{(c)\}$ über A ein gültiger Algebra-Ausdruck.

2. Induktiver Schritt

- ▶ Sind E und F gültige Algebra-Ausdrücke, dann auch:

$$\begin{array}{ll} E \cup F & \sigma_G(E) \\ E - F & \pi_{A_1, \dots, A_m}(E) \\ E \times F & \rho_{A' \leftarrow A}(E) \\ & \rho_S(E) \end{array}$$

- ▶ Die Voraussetzungen der Operatoren müssen gelten.
- ▶ Mehrfache Attribut-Umbenennung schreibt man geschachtelt.

3. Abschluss (Minimalität)

- ▶ Gültig sind nur Ausdrücke, die nach 1 und 2 gebildet werden.

Konsequenz: Jeder gültige Algebra-Ausdruck beschreibt eine Relation.



Frage:
Warum Ausdrücke
statt Relationen?

Beispiel: Indirekte Voraussetzungen von Vorlesung 5216

- ▶ Gegeben ist die Relation *voraussetzen*(*Vorgänger*, *Nachfolger*).
- ▶ Für 5216 kennen wir die direkte Voraussetzung (5041, 5216).
- ▶ Gesucht ist die zweite Stufe: Welche Vorlesungen liegen vor 5041?

Frage: Wie finden wir alle Vorlesungen, die indirekt vor 5216 liegen?

Wir brauchen zwei umbenannte Rollen derselben Relation: einmal für $5041 \rightarrow 5216$ und einmal für den Schritt davor.

Relation *voraussetzen*

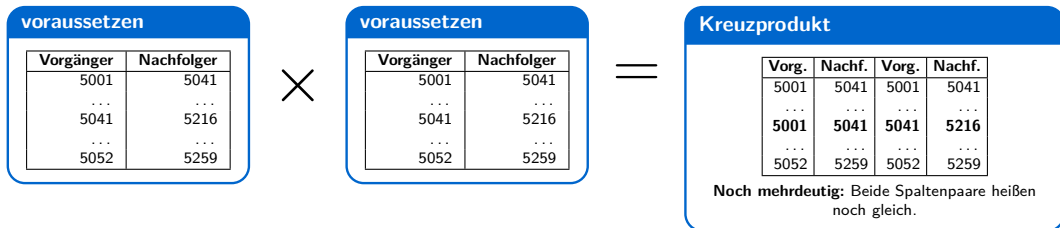
Vorgänger	Nachfolger
5001	5041
...	...
5041	5216
...	...
5052	5259

Die markierten Tupel bilden die Kette **5001** $\rightarrow$ **5041** $\rightarrow$ **5216**.

Rollen: 5041 ist einmal **Nachfolger** und einmal **Vorgänger**.

Beispiel: Kreuzprodukt für die 2. Stufe von 5216

- ▶ Wir suchen Voraussetzungen zweiter Stufe für 5216.
- ▶ Ein naheliegender erster Schritt ist *voraussetzen* $\times$ *voraussetzen*.

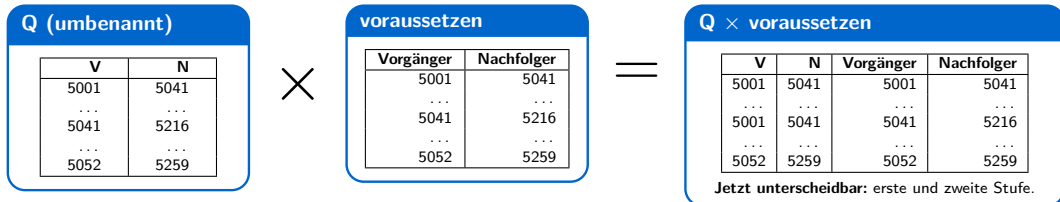


Problem: Doppelte Attributnamen verhindern die Selektion.

Beispiel: 2. Stufe der Voraussetzungen von 5216

1. Attribute umbenennen 2. Kreuzprodukt bilden

Die linke Kopie erhält neue Attributnamen, damit beide Rollen unterscheidbar werden.



$$Q = \rho_{V \leftarrow \text{Vorgänger}} \left(\rho_{N \leftarrow \text{Nachfolger}} (\text{voraussetzen}) \right)$$

Beispiel: Selektion für die 2. Stufe von 5216

1. Attribute umbenennen 2. Kreuzprodukt bilden 3. Selektion

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

Q (umbenannt)

V	N
5001	5041
...	...
5041	5216
...	...
5052	5259

×

voraussetzen

Vorgänger	Nachfolger
5001	5041
...	...
5041	5216
...	...
5052	5259

=

Nach der Selektion

V	N	Vorgänger	Nachfolger
5001	5041	5041	5216

Ergebnis: Genau ein passendes Tupel bleibt übrig.

Die Bedingung erzwingt genau die Kette $V \rightarrow N \rightarrow 5216$.

Beispiel: Projektion auf die 2. Stufe von 5216

1. Attribute umbenennen 2. Kreuzprodukt bilden 3. Selektion 4. Projektion

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

Nach der Selektion steht die richtige Kette fest.
Für das Ergebnis brauchen wir jetzt nur noch
den gesuchten Vorgänger **V**.

Interpretation: Die Projektion auf **V** entfernt alle Hilfsattribute.
Übrig bleibt nur die gesuchte Vorlesung.

Nach der Selektion

V	N	Vorgänger	Nachfolger
5001	5041	5041	5216

⇓ π_V

Ergebnis

V
5001

Nur V bleibt übrig.

Operatorbäume: Visualisierung von Ausdrücken

Zweck

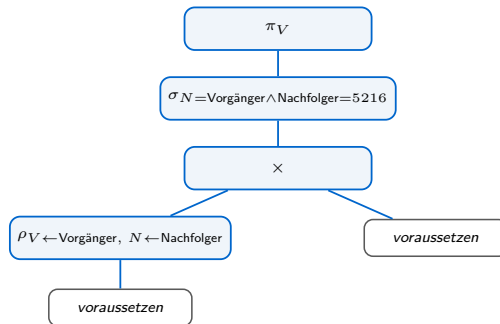
- ▶ Operatorbäume zeigen die Berechnungsstruktur eines Algebra-Ausdrucks.
- ▶ Sie machen die Reihenfolge der Teilschritte sichtbar.
- ▶ Für Verständnis und Optimierung sind sie ähnlich wichtig wie Syntaxbäume.

Aufbau

- ▶ **Blätter:** Eingaberelationen
- ▶ **Innere Knoten:** Operatoren wie ρ , $\times$, σ , π
- ▶ **Kanten:** Übergabe von Zwischenergebnissen
- ▶ **Wurzel:** Endergebnis der Anfrage

Beispiel: 2. Stufe der Voraussetzungen von 5216

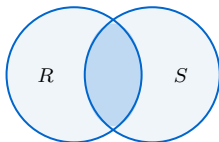
$$\pi_V \left(\sigma_{N=\text{Vorgänger} \wedge \text{Nachfolger}=5216} \left(\rho_{V \leftarrow \text{Vorgänger}, N \leftarrow \text{Nachfolger}} (\text{voraussetzen}) \times \text{voraussetzen} \right) \right)$$



Weitere Operation: Durchschnitt ($\cap$)

- ▶ **Notation:** $R \cap S$
- ▶ **Funktion:** Liefert genau die Tupel, die in R und in S vorkommen.
- ▶ **Bedingung:** R und S müssen schema-kompatibel sein.
- ▶ **Ableitung:** Keine Grundoperation; es gilt $R \cap S = R - (R - S)$.
- ▶ Das Ergebnisschema ist identisch zum Schema von R und S .

Aufgabe: Erkläre mit dem Venn-Diagramm, warum $R - (R - S)$ genau die Schnittmenge liefert.



Beispiel: Zwei Studentengruppen

Gruppe 1

MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10

Gruppe 2

MatrNr	Name	Semester
26120	Fichte	10
25403	Jonas	12
26830	Aristoxenos	8

Algebra-Ausdruck: $Gruppe1 \cap Gruppe2$

Ausgabe

MatrNr	Name	Semester
25403	Jonas	12
26120	Fichte	10

Weitere Operation: Natürlicher Verbund ($\bowtie$)

- ▶ **Notation (Natural Join):** $R \bowtie S$
- ▶ Tupel werden kombiniert, wenn alle gleich benannten Attribute dieselben Werte haben.
- ▶ Gemeinsame Attribute erscheinen im Ergebnis nur **einmal**.
- ▶ Das Ergebnis enthält die übrigen Attribute aus R und S ; die Verknüpfung ergibt sich automatisch aus den gemeinsamen Namen.

Beispiel: Studenten hören Vorlesungen

Student

MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10

hört

MatrNr	VorlNr
26120	5001
25403	5022
26120	5041

Algebra-Ausdruck: $Student \bowtie hört$

Ausgabe: StudentHört

MatrNr	Name	Semester	VorlNr
25403	Jonas	12	5022
26120	Fichte	10	5001
26120	Fichte	10	5041

Beobachtung: Die gemeinsame Spalte MatrNr erscheint nur einmal.

? Frage: Wie bildet man $R \bowtie S$ aus Basisoperationen?

Natürlicher Verbund: Beispiel mit drei Relationen

Student ⋈ *hört* ⋈ *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
...	...	...	...

(Student ⋈ hört) ⋈ Vorlesung

MatrNr	Name	Sem.	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
...	...	...	...	...	...	...

Fragen

- ▶ Ändert die Klammerung das Ergebnis?
- ▶ Ist *Student* ⋈ *hört* dasselbe wie *hört* ⋈ *Student*?
- ▶ Warum oder warum nicht?

Weitere Operation: Theta-Join ($\bowtie_{\theta}$)

- ▶ **Notation:** $R \bowtie_{\theta} S$
- ▶ **Funktion:** Verknüpft Tupel aus R und S gemäß θ .
- ▶ **Definition:** $R \bowtie_{\theta} S = \sigma_{\theta}(R \times S)$
- ▶ θ kann beliebige Vergleiche enthalten, z. B. $=$, $>$, $<$ oder $\neq$.
- ▶ Für $\theta ==$ spricht man auch von einem **Equijoin**; das Ergebnisschema enthält zunächst alle Spalten aus R und S .

Gleichnamige Attribute werden vor dem Theta-Join mit ρ umbenannt.
Gleichbedeutende Attribute bleiben doppelt und werden später projiziert.

Beispiel: Welche Studenten mögen Bier?

Student

MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10

magBier

IDNr	Sorte
25403	Kölsch
26120	Hefeweizen

Algebra-Ausdruck: $Student \bowtie_{MatrNr=IDNr} magBier$

Ausgabe: P

MatrNr	Name	Semester	IDNr	Sorte
25403	Jonas	12	25403	Kölsch
26120	Fichte	10	26120	Hefeweizen

Frage: Wie können wir die doppelte Identifikationsinformation $MatrNr/IDNr$ wieder loswerden?

Weitere Operationen: Outer Joins (Äußere Verbünde)

- ▶ **Problem:** Innere Joins ($\bowtie$, $\bowtie_{\theta}$) verwerfen Tupel ohne Join-Partner.
- ▶ **Lösung:** Outer Joins behalten diese Tupel und ergänzen fehlende Werte mit **NULL**.
- ▶ **Left Outer Join:** $R \bowtie_{\theta} S$ erhält alle Tupel von R .
- ▶ **Right Outer Join:** $R \bowtie_{\theta} S$ erhält alle Tupel von S .
- ▶ **Full Outer Join:** $R \bowtie_{\theta} S$ erhält alle Tupel aus R und S .
- ▶ Das Ergebnisschema enthält wie beim Theta Join die Attribute beider Relationen. Ohne θ : Natural Outer Join (Ergebnisrelation enthält nur ein Join-Attribut).

Outer Joins haben wie Theta Joins eine explizite Bedingung θ . Gleichnamige Attribute werden vorher mit ρ umbenannt.

Beispiel: Studenten hören Vorlesungen

Student

MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10

hört' (umbenannt)

HMatrNr	VorlNr
26120	5001
25403	5022
26120	5041
29555	5022

Algebra-Ausdruck:

$Student \bowtie_{MatrNr=HMatrNr \ \rho HMatrNr \leftarrow MatrNr} (hört')$

Ausgabe: StudentHört

MatrNr	Name	Semester	HMatrNr	VorlNr
24002	Xenokrates	18	NULL	NULL
25403	Jonas	12	25403	5022
26120	Fichte	10	26120	5001
26120	Fichte	10	26120	5041

Weitere Operationen: Semi-Joins ($\bowtie$, $\ltimes$)

- ▶ **Left Semi Join:** $R \ltimes_{\theta} S$
- ▶ **Right Semi Join:** $R \bowtie_{\theta} S$
- ▶ Semi-Joins filtern eine Relation anhand passender Join-Partner in der anderen.
- ▶ Die Attribute der zweiten Relation werden **nicht** in das Ergebnis übernommen.
- ▶ $R \ltimes_{\theta} S$ hat das Schema von R , $R \bowtie_{\theta} S$ das von S . Ohne θ : Natural Semi Join.

$$R \ltimes_{\theta} S = \pi_{\text{attr}(R)}(R \bowtie_{\theta} S)$$

$$R \bowtie_{\theta} S = \pi_{\text{attr}(S)}(R \bowtie_{\theta} S)$$

Gleichnamige Attribute werden vor dem Semi-Join (mit θ) mit ρ umbenannt; die Join-Bedingung bleibt explizit sichtbar.

Beispiel: Studenten hören Vorlesungen

Student

MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10

hört' (umbenannt)

HMatrNr	VorlNr
26120	5001
25403	5022
26120	5041
29555	5022

Algebra-Ausdruck:

$$Student \ltimes_{MatrNr=HMatrNr \wedge PHMatrNr \leftarrow MatrNr} (h\ddot{o}rt')$$

oder

$$Student \ltimes h\ddot{o}rt$$

Ausgabe: StudentMitVorlesung

MatrNr	Name	Semester
25403	Jonas	12
26120	Fichte	10

Nur Tupel aus *Student* mit passendem Join-Partner bleiben erhalten.

Join-Operatoren – Überblick

Theta Join ($\bowtie_{\theta}$)

Nur Tupel, die θ erfüllen, werden verbunden; die Attribute beider Relationen bleiben zunächst erhalten.

Left Outer Join ($\bowtie_{\theta}^{\leftarrow}$)

Wie bei $\bowtie_{\theta}$, aber alle linken Tupel bleiben erhalten; fehlende rechte Werte werden mit NULL ergänzt.

Left Semi-Join ($\ltimes_{\theta}$)

Nur linke Tupel mit Join-Partner bleiben erhalten; das Schema bleibt das von R .

Right Outer Join ($\bowtie_{\theta}^{\rightarrow}$)

Wie bei $\bowtie_{\theta}$, aber alle rechten Tupel bleiben erhalten; fehlende linke Werte werden mit NULL ergänzt.

Full Outer Join ($\bowtie_{\theta}^{\leftrightarrow}$)

Passende Tupel werden verbunden; alle übrigen Tupel bleiben erhalten und werden auf der fehlenden Seite mit NULL ergänzt.

Right Semi-Join ($\rtimes_{\theta}$)

Nur rechte Tupel mit Join-Partner bleiben erhalten; das Schema bleibt das von S .



Frage: Welche dieser Operatoren sind kommutativ oder assoziativ, welche nicht?

3. Das Relationale Datenmodell

Relationaler Kalkül

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

Wir wechseln von der prozeduralen Algebra zur logischen Anfragebeschreibung:
vom **WIE** zum **WAS**.

Von der Algebra zum Kalkül: Zwei Sichten auf Anfragen

Bisher: Relationale Algebra

Ansatz: prozedural

- ▶ Beschreibt, **wie** das Ergebnis durch Operatoren konstruiert wird.
- ▶ Typische Operatoren sind σ , π , $\times$, $\bowtie$.
- ▶ Analogie: Kochrezept oder Algorithmus.

$\sigma_{\text{Rang} = 'W3'}(Professor)$

Alternative: Relationaler Kalkül

Ansatz: deklarativ

- ▶ Beschreibt, **was** das Ergebnis erfüllen muss.
- ▶ Grundlage ist die Prädikatenlogik.
- ▶ Analogie: Zielbeschreibung oder Spezifikation.

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

Kernunterschied: Algebra beschreibt das **WIE**, Kalkül das **WAS**.

Kalkül-Varianten: Tupelkalkül (TRC) mit Tupelvariablen, Domänenkalkül (DRC) mit Attributwerten.

Wichtige Erkenntnis: Relationale Algebra und **sicherer** relationaler Kalkül sind gleich mächtig.

Relationaler Kalkül: Tupelkalkül (TRC)

Grundidee und Form

- ▶ Variablen wie t, p, s stehen für ganze **Tupel**.
- ▶ Allgemeine Form: $\{ t \mid F(t) \}$
- ▶ t ist die freie Variable und beschreibt das Ergebnis.
- ▶ $F(t)$ ist eine logische Formel, die t erfüllen muss.
- ▶ Typische Bausteine sind Relationszugehörigkeit, Attributvergleiche, Logik und später Quantoren.

Lesart: Das Ergebnis enthält genau die Tupel t , für die $F(t)$ wahr ist.

Beispiel: Finde alle W3-Professoren

- ▶ Gesucht sind Tupel p mit:
- ▶ $p \in Professor$ und $p.Rang = 'W3'$.

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

TRC: Existenzquantor $\exists$ („Es gibt...“)

Bedeutung

- ▶ $\exists t \in R(P(t))$ bedeutet: Es gibt ein Tupel t in R , für das $P(t)$ wahr ist.
- ▶ Ein Quantor ($\exists$ / $\forall$) vor $P(t)$ bindet t nur innerhalb von $P(t)$.
- ▶ Existenzquantoren modellieren typische „mindestens ein“-Bedingungen.

Frei vs. gebunden: Die Variable der äußeren $\{\}$ -Klammer bleibt frei; Variablen nach $\exists$ sind nur lokale Hilfsvariablen.

Beispiel: Studierende mit einer Curie-Vorlesung

```
{ s | s ∈ Student ∧ ∃ h ∈ hört (
    s.MatrNr = h.MatrNr ∧
    ∃ v ∈ Vorlesung (
        h.VorlNr = v.VorlNr ∧
        ∃ p ∈ Professor (
            p.PersNr = v.gelesenVon ∧
            p.Name = 'Curie'
        )
    )
}
```

Lesart: Ein Student gehört ins Ergebnis, wenn es eine Kette Student $s \rightarrow h$ (*hört*) $\rightarrow$ Vorlesung $v \rightarrow$ Professor Curie gibt.

Bedeutung

- ▶ $\forall t \in R(P(t))$ bedeutet: Jedes Tupel aus R muss $P(t)$ erfüllen.
- ▶ Allquantoren modellieren Bedingungen vom Typ **für alle**.
- ▶ In Anfragen prüft $\forall$ meist nur Tupel mit Vorbedingung; deshalb steht innen fast immer eine Implikation.

Wichtige Struktur: Nur Sokrates-Vorlesungen lösen die Existenzprüfung aus; andere Tupel erfüllen die Formel sofort.

Beispiel: Alle Sokrates-Vorlesungen hören

$$\left\{ s \mid s \in Student \wedge \forall v \in Vorlesung (\right. \\ \left. \begin{array}{l} v.gelesenVon = 2125 \Rightarrow \exists h \in hört (\\ \quad h.VorlNr = v.VorlNr \wedge h.MatrNr = s.MatrNr \\ \quad) \\ \quad) \end{array} \right\}$$

Lesart: Ein Student gehört ins Ergebnis, wenn zu jeder Sokrates-Vorlesung ein passender Eintrag in *hört* existiert.

? Frage: Warum brauchen wir hier die Implikation ($\Rightarrow$)? Was würde mit $\wedge$ passieren?

TRC: Ergebnisschema und Tupelkonstruktion

Grundidee

- ▶ Eine Tupelvariable besitzt ein Schema $(A_1 : D_1, \dots, A_k : D_k)$.
- ▶ Ist das Schema aus dem Zusammenhang klar, kann es weggelassen werden; siehe Beispiel 1.
- ▶ Über das Schema lassen sich auch **neue Ergebnistupel** konstruieren; siehe Beispiel 2.
- ▶ Die Kurzform beschreibt dieselbe Konstruktion kompakter; siehe Beispiel 3.

Merke: Beispiel 1 liefert Professor-Tupel. Beispiel 2 und 3 konstruieren neue Tupel mit nur PersNr.

Drei Varianten mit unterschiedlichem Ergebnis

1. Ganze Professor-Tupel

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

2. Neue Tupel mit explizitem Schema

$$\left\{ t \in (\text{PersNr} : \text{Integer}) \mid \exists p \in Professor (\begin{array}{l} t.PersNr = p.PersNr \wedge \\ p.Rang = 'W3' \end{array}) \right\}$$

3. Neue Tupel in Kurzform

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

TRC: Tupelkonstruktor in Kurz- und Langform

Tupelkonstruktor

$$[t_1.A_1, \dots, t_n.A_n]$$

- ▶ Die referenzierten Attribute müssen in den Schemata der gebundenen Variablen vorkommen.
- ▶ Auswahl und Reihenfolge der Komponenten bestimmen das Ergebnisschema.
- ▶ Fachlich entspricht das einer frei formulierten Projektion.

Beispielziel: Liefere für jede Boss-Beziehung ein Ergebnistupel aus Professurname und Assistenten-PersNr.

Beispiel: Kurz- und Langform

Kurzform

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

Direkt mit Tupelkonstruktor formuliert.

Langform

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

Dieselbe Anfrage mit explizitem Ergebnisschema.

TRC: Formale Definition Syntax: Tupelvariablen

Syntax: **Tupelvariable** $t \implies$ Formel $F(t) \implies$ Ausdruck $\{ t \mid F(t) \}$

Idee: Ein Ausdruck liefert als Ergebnis die Relation aller Tupel, die die Formel F erfüllen.

1. Tupelvariablen

Konstruktion von gültigen Tupelkalkül-Ausdrücken

- ▶ Eine Tupelvariable t besitzt ein Schema $(A_1:D_1, \dots, A_k:D_k)$.

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

- ▶ Das Schema ist oft implizit festgelegt, z. B. durch eine Formel der Form $t \in R$.
- ▶ Es kann auch aus der Struktur eines Tupelkonstruktors wie $[t_1.A_1, \dots, t_n.A_n]$ hervorgehen.
- ▶ Eine Tupelvariable kann in Formeln **frei** oder **gebunden** auftreten; die Bindung folgt in Schritt 2.

TRC: Formale Definition Syntax: Formeln

Syntax: Tupelvariable $t \implies$ **Formel** $F(t) \implies$ Ausdruck $\{t \mid F(t)\}$

Idee: Ein Ausdruck liefert als Ergebnis die Relation aller Tupel, die die Formel F erfüllen.

2. Formeln

1. Atome (kleinste wahr/falsch-auswertbare Bausteine)

$$\begin{aligned} & t \in R \\ & t.A \theta s.B \quad \theta \in \{=, \neq, <, \leq, >, \geq\} \\ & t.A \theta c \end{aligned}$$

Diese Atome bilden die kleinsten wahr/falsch-auswertbaren Bausteine.

2. Rekursive Bildung

1. Jedes Atom ist eine Formel.
2. Sind F und G Formeln, dann auch $\neg F$, $F \wedge G$ und $F \vee G$.
3. Ist F eine Formel, in der t frei vorkommt, dann sind auch $\exists t \in R(F)$ und $\forall t \in R(F)$ Formeln.

Quantoren binden die Variable t innerhalb von F .

TRC: Formale Definition Syntax: Ausdruck

Syntax: Tupelvariable $t \implies$ Formel $F(t) \implies$ **Ausdruck** $\{t \mid F(t)\}$

Idee: Ein Ausdruck liefert als Ergebnis die Relation aller Tupel, die die Formel F erfüllen.

3. Ausdrücke

Ein Ausdruck des Tupelkalküls hat die Form

$$\{t \mid F(t)\} \quad \text{oder} \quad \{t \in (A_1:D_1, \dots, A_k:D_k) \mid F(t)\}$$

- ▶ In F darf nur **eine** Tupelvariable frei bleiben: t .
- ▶ Das Ergebnisschema stammt entweder direkt von t oder wird explizit angegeben.
- ▶ Der Ausdruck beschreibt genau die Tupel, für die $F(t)$ wahr wird.

TRC: Formale Definition Semantik

Semantik: Tupelvariable $t \implies$ Formel $F(t) \implies$ Ausdruck $\{ t \mid F(t) \}$

Idee: Interpretation der Tupelkalkül-Ausdrücke.

Analoge drei Schritte

- ▶ Tupelvariablen $\implies$ konkrete Tupel
- ▶ Formeln (Atome, Operatoren und Quantoren) $\implies$ true oder false
- ▶ Ausdrücke $\implies$ Relationen

Die Semantik folgt also denselben drei Ebenen wie die Syntax: Tupelvariablen, Formeln und Ausdrücke erhalten jeweils eine präzise Bedeutung.

TRC: Formale Definition Semantik: Tupelvariablen

Semantik: **Tupelvariable** $t \implies$ Formel $F(t) \implies$ Ausdruck $\{t \mid F(t)\}$

Idee: Eine Tupelvariable wird semantisch durch ein konkretes Tupel belegt.

1. Tupelvariablen

Belegung von Tupelvariablen

- ▶ t sei eine Tupelvariable mit Schema $(A_1:D_1, \dots, A_k:D_k)$.
- ▶ $F(t)$ sei eine Formel, die t frei enthalten kann.
- ▶ $r \in D_1 \times \dots \times D_k$ sei ein beliebiges Tupel.

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

$$F(r \mid t)$$

Beispiel:

$$\begin{aligned} F(t) &= t \in Professor \wedge t.Rang = 'W3' \\ r_1 &= (2125, 'Sokrates', 'W3', 226) \\ F(r_1 \mid t) &= r_1 \in Professor \wedge 'W3' = 'W3' \end{aligned}$$

Semantik: Tupelvariable $t \implies$ **Formel** $F(t) \implies$ Ausdruck $\{t \mid F(t)\}$

Idee: Formeln werden unter einer Belegung als true oder false interpretiert.

2. Formeln

Atome

- ▶ $I(r \in R) = \text{true}$, falls $r \in R$; sonst false
- ▶ $I(c \theta d) = \text{true}$, falls c in Beziehung θ zu d steht

Logische Operatoren

- ▶ $I(\neg F) = \text{true}$ gdw. $I(F) = \text{false}$
- ▶ $I(F_1 \wedge F_2) = \text{true}$ gdw. beide Formeln wahr sind
- ▶ $I(F_1 \vee F_2) = \text{true}$ gdw. mindestens eine der beiden Formeln wahr ist

Quantoren (nur t darf in F frei sein)

- ▶ $I(\exists t \in R (F(t))) = \text{true}$ gdw. ein $r \in R$ existiert mit $I(F(r \mid t)) = \text{true}$
- ▶ $I(\forall t \in R (F(t))) = \text{true}$ gdw. für alle $r \in R$ gilt: $I(F(r \mid t)) = \text{true}$

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

TRC: Formale Definition Semantik: Ausdruck

Semantik: Tupelvariable $t \implies$ Formel $F(t) \implies$ **Ausdruck** $\{ t \mid F(t) \}$

Idee: Ein Ausdruck wird semantisch als Relation interpretiert.

3. Ausdrücke

Sei

$$E = \{ t \mid F(t) \} \quad \text{oder} \quad E = \{ t \in (A_1 : D_1, \dots, A_k : D_k) \mid F(t) \}$$

ein Ausdruck und $D_1 \times \dots \times D_k$ die Wertebereiche des Schemas von t .

- ▶ t ist die einzige freie Tupelvariable in F .
- ▶ Der Wert von E ist die Menge aller Tupel $r \in D_1 \times \dots \times D_k$, für die $I(F(r \mid t)) = \text{true}$ gilt.

$$\text{Wert}(E) = \{ r \in D_1 \times \dots \times D_k \mid I(F(r \mid t)) = \text{true} \}$$

TRC: Sicherheit – Endliche Ergebnisse garantieren

Problem

$$\{t \mid \neg(t \in \textit{Student})\}$$

- ▶ Der Ausdruck beschreibt alle Tupel außerhalb von *Student*.
- ▶ Das Ergebnis kann unendlich sein und hängt nicht nur von der Datenbankinstanz ab.
- ▶ Solche Anfragen sind praktisch nicht sinnvoll auswertbar.

Intuition: Sicher heißt: Alle Variablen bleiben an tatsächlich vorkommende Datenbankwerte gebunden; nur dann bleiben TRC-Anfragen endlich.

Beispiele

Sicher:

$$\{t \mid t \in \textit{Student} \wedge t.\textit{Semester} > 2\}$$

Unsicheres Gegenbeispiel:

Links ist der Ausdruck unsicher, weil *t* an keine endliche Relation gebunden wird.



Frage:

Ist der Ausdruck sicher oder unsicher?

$$\left\{ t \mid (t \in \textit{Student} \wedge t.\textit{Semester} > 10) \vee (\forall s \in \textit{Student} (s.\textit{Semester} \geq t.\textit{Semester})) \right\}$$

Domänenkalkül (DRC): Idee und Grundform

DRC-Block: Wertvariablen statt Tupelvariablen

Wertvariablen $x_1, \dots, x_k \Rightarrow$ Formel $F(x_1, \dots, x_k) \Rightarrow$ Ausdruck $\{x_1, \dots, x_k \mid F(x_1, \dots, x_k)\}$

1. Grundform

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

- ▶ Die Belegungen der Variablen sind Werte aus Attributdomänen.
- ▶ Das Ergebnis besteht genau aus den ausgegebenen Werten $x_1, \dots, x_k$.

2. Atome und Formeln

$$R(x_1, \dots, x_n)$$

$$x \theta y \quad \theta \in \{=, \neq, <, \leq, >, \geq\}$$

$$x \theta c$$

- ▶ Atome werden unter einer Belegung zu **wahr** oder **falsch**.
- ▶ Formeln entstehen analog zum TRC; Quantoren binden Werte aus Domänen statt Tupel aus Relationen.

TRC und DRC sind für sichere Anfragen gleich ausdrucksstark; sie unterscheiden sich nur in der Ebene der Variablen: Tupel im TRC, Einzelwerte im DRC.

Domänenkalkül (DRC): Drei typische Beispiele

Einordnung: Nach der Grundform der vorhergehenden Folie folgen nun drei typische DRC-Anfragen.

Beispiel 1: Studierende im 3. Semester

Ergebnis: MatNr m und Name n .

$$\{ m, n \mid \exists s (Student(m, n, s) \wedge s = 3) \}$$

Beispiel 3: Algebra im DRC

Schematisch lassen sich auch algebraische Operationen im DRC ausdrücken:

$$R \cup S = \{ \vec{x} \mid R(\vec{x}) \vee S(\vec{x}) \}$$

$$R - S = \{ \vec{x} \mid R(\vec{x}) \wedge \neg S(\vec{x}) \}$$

$$P \times Q = \{ \vec{x}, \vec{y} \mid P(\vec{x}) \wedge Q(\vec{y}) \}$$

Hier stehen $\vec{x}$ und $\vec{y}$ für passende Wertelisten.

Beispiel 2: Prüfungen bei Curie

Ergebnis: MatNr m und Name n .

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

TRC vs. DRC – Beispiel 1

Anfrage

Gib **MatNr** und **Name** aller Studierenden aus, die mindestens eine Prüfung bei Professor 'Curie' abgelegt haben.

TRC: Tupelvariablen

$$\left\{ [s.\text{MatrNr}, s.\text{Name}] \mid s \in \text{Student} \wedge \right. \\ \left. \begin{array}{l} \exists h \in \text{prüfen} \exists v \in \text{Vorlesung} \\ \exists p \in \text{Professor} (\\ \quad s.\text{MatrNr} = h.\text{MatrNr} \wedge \\ \quad h.\text{VorlNr} = v.\text{VorlNr} \wedge \\ \quad p.\text{PersNr} = v.\text{gelesenVon} \wedge \\ \quad p.\text{Name} = \text{'Curie'}) \end{array} \right\}$$

DRC: Wertvariablen

$$\left\{ m, n \mid \exists s, v, p, g, r, b (\right. \\ \left. \begin{array}{l} \text{Student}(m, n, s) \wedge \\ \text{prüfen}(m, v, p, g) \wedge \\ \text{Professor}(p, \text{'Curie'}, r, b) \end{array} \right\}$$

Merke: Im DRC sind nur die Ausgabewerte freie Variablen.

TRC vs. DRC – Beispiel 2

Anfrage

Gib die **PersNr** aller W3-Professoren aus.

TRC: Tupelvariablen

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

- ▶ Variable p steht für ein ganzes Professor-Tupel.
- ▶ Die Ausgabe entsteht über den Tupelkonstruktor $[p.\text{PersNr}]$.

DRC: Wertevariablen

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

- ▶ Variable p steht direkt für den auszugebenden Wert.
- ▶ Weitere Attribute werden über Existenzvariablen gebunden.

Beide Ausdrücke sind gleichwertig: Sie liefern die Personalnummern aller W3-Professoren.

TRC vs. DRC – Beispiel 3 & Fazit

Anfrage

Ordne jedem Professor den **Namen des Professors** und die **PersNr seines Assistenten** zu.

TRC: Tupelvariablen

$$\left\{ [p.Name, a.PersNr] \mid \begin{array}{l} p \in Professor \wedge \\ a \in Assistent \wedge \\ p.PersNr = a.Boss \end{array} \right\}$$

DRC: Wertvariablen

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

Fazit

- ▶ **TRC:** Variablen stehen für Tupel.
- ▶ **DRC:** Variablen stehen für Werte.
- ▶ Bei sicheren Ausdrücken sind beide **gleich mächtig**.
- ▶ Dieselben Anfragen gibt es auch in relationaler Algebra.

Sicherer TRC $\iff$ sicherer DRC $\iff$ relationale Algebra