Natürlicher Verbund (1)

- **Eingabe**
  - Zwei Relationen r_1 und r_2 die gleiche Attribute besitzen.
    - Durch die Verwendung von **Primärschlüsseln als Fremdschlüssel** ist dies oft der Fall.
    - Ziel ist es, die Daten aus beiden Relation miteinander zu verknüpfen.
- **Ausgabe (konstruktive Beschreibung)**
  1. Wir erzeugen das kartesische Produkt (ohne dabei zu fordern, dass die Schemata disjunkt sind).
     - Zur Unterscheidung bekommen gleiche Attribute als Präfix den Relationenname (und einem Punkt).
  2. Es werden alle Tupel eliminiert, die in den „gleichen“ Attributpaaren unterschiedliche Werte haben.
  3. Danach wird noch für jedes dieser Attributpaare eins der Attribute eliminiert.

Natürlicher Verbund (2)

- **Beispiele**
  - Gegeben folgende Relationen r_1 und r_2.

r_1

A	B	C
1	2	3
2	3	4

r_2

B	C	D
2	3	4
2	3	5
3	5	2

  - Natürlicher Verbund zwischen der Relation PMZuteilung und Maschine.

1. Kartesisches Produkt

A	r_1.B	r_1.C	r_2.B	r_2.C	D
1	2	3	2	3	4
...	...	...	...	...	...

2. Selektion
   - Erhalte nur die Tupel mit $r_1.B = r_2.B$ und $r_1.C = r_2.C$.
3. Projektion
   - Entferne die „doppelten“ Spalte $r_2.B$ und $r_2.C$

Formale Definition (Natürlicher Verbund)

- **Gegeben zwei Relationen $r_1 \in REL(RS_1)$, $r_2 \in REL(RS_2)$. Dann ist**

$$r_1 \bowtie r_2 := \{t \mid t[RS_1] \in r_1 \text{ und } t[RS_2] \in r_2\}$$

- **Man beachte, dass bei dieser Definition kein Relationenschema für t angegeben wurde. Dieser ergibt sich implizit aus der kleinsten Menge von Attributen ($RS_1 \cup RS_2$), so dass die Bedingung syntaktisch korrekt ist.**

$$r_1 \bowtie r_2$$

$RS_1 - RS_2$			$RS_1 \cap RS_2$			$RS_2 - RS_1$		
A_1	. . .	A_m	B_1	. . .	B_k	C_1	. . .	C_n
				gemeinsame Attribute				

Unterschiede zwischen Theta Join und Natural Join

- Beim Theta-Join ist es erforderlich, dass die Attribute der Relationen unterschiedlich sind.
  - ➔ Es gibt **keine** automatische Selektion attributgleicher Tupel

- Beim Theta-Join findet **keine** automatische Projektion auf "relevante" Spalten statt.

Semi-Join

- **Informell**
  - Der Semi-Join von Relationen r und s berechnet alle Tupel der Relation r, die am Join mit der Relation s beteiligt sind.
- **Beispiel**
  - Berechne alle Personen, die mindestens eine Maschine bedienen können.
- **Varianten**
  - Linker und rechter Semi Join
    - Teilrelationenbildung eines der beiden Join-Operanden
    - Nur diejenigen Tupel des ausgewählten Join-Operanden werden ausgewählt, die "einen Joinpartner" besitzen.
  - Der Semi-Join kann für alle Join-Varianten definiert werden.

Semi-Join (Definition)

Seien $r_1 \in REL(RS_1)$, $r_2 \in REL(RS_2)$.

$$r_1 \ltimes r_2 := \pi_{RS_1}(r_1 \bowtie r_2)$$

- Beispiel

r_1

A	B
1	2
3	1
2	5

$\ltimes$

r_2

B	C
5	2
2	1
2	3

$=$

A	B
1	2
2	5

Anti-Join

- "vergessene" Operation (von Codd nicht eingeführt und in Lehrbüchern meist nicht erwähnt): Anti-Join (auch Complement (Semi) Join)

- Tupel aus einem der beiden Operanden werden ausgewählt, die **keinen** Join-partner besitzen.

- Definition: $r_1 \overline{\ltimes} r_2 := r_1 - (r_1 \ltimes r_2)$

- Beispiel:

r_1

A	B
1	2
3	1
2	5

$\overline{\ltimes}$

r_2

B	C
5	2
2	1
2	1

=

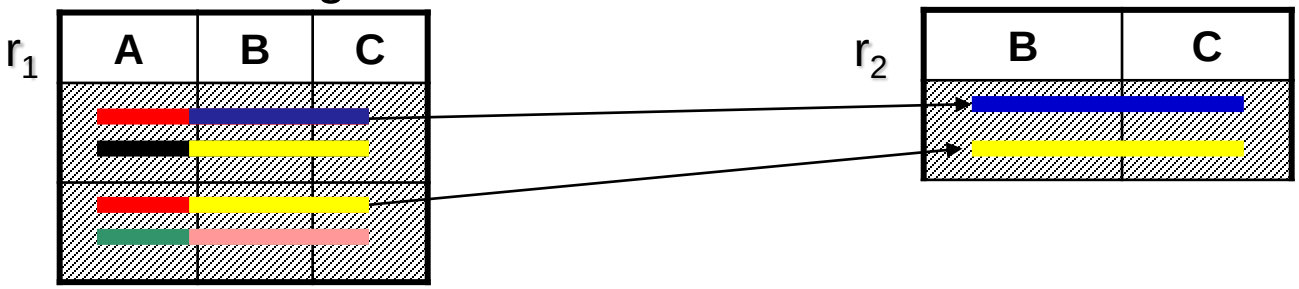
A	B
3	1

- Semi Join und Anti Join sind - wie alle Joinvarianten - ableitbar.

- Semi Join, Anti Join sind nicht kommutativ !

Division

- **Informell**
  - Eingabe
    - Zwei Relationen r_1 und r_2, wobei das Schema von r_1 das Schema von r_2 enthält.
  - Ausgabe
    - Tupel aus r_1, die mit allen Tupeln der Relation r_2 in Beziehung stehen.



- **Beispiel**
  - Berechne die Angestellten, die alle Maschinen bedienen können.

Division (Formale Definition)

- **Definition**

  Seien $r_1 \in REL(RS_1)$ und $r_2 \in REL(RS_2)$ mit $\mathbf{RS_1 \supseteq RS_2}$.

  $r_1 \div r_2 := \{t \mid \forall u \in r_2\ \exists\, v \in r_1 : t = v[RS_1 - RS_2] \wedge u[RS_2] = v[RS_2]\}$

  - Welche Attribute sind in dem Schema von $r_1 \div r_2$?

- **Beispiel**

r_1

A	B
a	1
a	2
a	3
b	2
b	3

$\div$

r_2

B
1
2

$=$

A
a

Division (Beispiel)

Das Beispiel zeigt schon die Bedeutung der Division bei der Formulierung allquantifizierter Aussagen:

"Welcher Angestellter ist für alle Maschinen ausgebildet?"

bzgl. dieser Tabellen der ERP-Datenbank:

PMZuteilung

pnr	mnr	Note
67	**84**	**3**
67	**93**	**2**
67	**101**	**3**
73	**84**	**5**
114	**93**	**5**
114	**101**	**3**
51	**93**	**2**

. . .

Maschine

mnr	MName
84	Presse
93	Füllanlage
101	Säge

Eigenschaften der Division

- **Man kann die Division als Umkehrfunktion des kartesischen Produkts ansehen.**
  - Ist $s = r_1 \div r_2$. Dann gilt nämlich folgende Eigenschaft:

    $$s \times r_2 \subseteq r_1 \qquad (*)$$

  - Tatsächlich ist s die **größte** Relation, die diese Eigenschaft (*) erfüllt.

- **Aus Eigenschaft (*) lässt sich herleiten, wie die Division auf die Grundoperationen der RA zurückgeführt werden kann.**

Beispiele (1)

- **Datenbankschema siehe Folie S. 49**
  - Finde alle Namen von Angestellten, die an einer Maschine ausgebildet sind.

Beispiele (2)

- **Datenbankschema siehe Folie S. 49**
  - Finde alle Namen der Angestellten, die an keiner Maschine genügend gut ausgebildet sind (Note ist immer schlechter als 4).

Beispiele (3)

- **Datenbankschema siehe Folie S. 49**
  - Finde die Namen der Angestellten aus Abteilung “Suppen”, die an der Maschine mit mnr = 93 ausgebildet sind.

Beispiele (4)

- **Datenbankschema siehe Folie S. 49**
  - Finde die Personalnummern der Angestellten, die an der gleichen Maschine ausgebildet sind wie der Angestellte mit pnr = 114.

Zusammenfassung

- **Relationale Algebra**
  - Menge von 6 Operatoren
    - Selektion, Projektion, Kartesisches Produkt, Vereinigung, Differenz und Umbenennung
  - Jeder Operator berechnet zu einer Relation eine neue (temporäre) Relation
- **Wichtige davon abgeleitete Operatoren**
  - Join
    - Natürlicher Join, Theta-Join, Semi-Join, Anti-Join
  - Division
    - Allquantifizierte Anfragen
- **Nachteile**
  - **Imperative Formulierung** durch explizite Vorschrift zur schrittweisen Berechnung → Komplexe aufgebaute Anfragen
  - Einschränkung auf die **Mengen**verarbeitung
    - Hoher Aufwand für die Duplikateliminierung
  - **Keine Aggregationsmöglichkeiten** für die Daten

2.3 Das Relationenkalkül

- **Bisher**
  - Benutzung einer prozeduralen Anfragesprache
  - Explizite Beschreibung, wie das Ergebnis berechnet wird.
    - Hoher Aufwand bei der Formulierung komplexer Anfragen

- **Zugrundeliegende Idee beim Relationenkalkül (RK)**
  - **Deklarative** Beschreibung der Ergebnisrelation ohne dabei explizit eine Vorschrift für die Konstruktion der Ergebnisrelation zu geben.

- **Zwei bekannte Kalküle**
  - **Tupelkalkül** und Domänenkalkül

Anfragesprachen

- **Entwicklung einer Anfragesprache basierend auf dem Relationenkalkül**
  ➔Erheblich einfachere und kompaktere Formulierung komplexer Anfragen.
- **SQL**
  - Sprache ist eine Mischung zwischen Relationale Algebra (RA) und dem Tupelkalkül (TK)
- **Anfragesprachen auf Basis des Relationenkalküls**
  - Graphische Anfragesprache **QBE** (Query by Example)
    - Anfragesprache ähnlich zu der von Microsoft Access
  - **Quel** – Anfragesprache im Ingres-System
  - **Datalog**
    - Basiert auf Konzepten der Sprache Prolog
    - Unterstützung von Rekursion

Grundlagen für TK

- **Prädikatenlogik erster Stufe**
  - Konzept der mathematischen Logik → siehe Modul Logik

- **Übertragung auf Datenbanken unter folgenden Annahmen (Closed World Assumption):**
  - Tupel in der Datenbank sind elementare Aussagen, die wahr sind.
  - Anfragen sind abgeleitete Aussagen, die aus anderen Aussagen hergeleitet werden können.
  - Aussagen, die **nicht** in der Datenbank sind und **nicht** hergeleitet werden können, **sind falsch**!

Das Tupelkalkül

- **Tupelkalkül (TK)**
  - Variablen, die einem Tupel einer Relation entsprechen.

    → **Tupelvariable**

    Das Konzept der Tupelvariable ist auch Teil der Anfragesprache SQL.
- **Beispiele für Anfragen im Tupelkalkül**

Tupelvariablen

Logische Operatoren

$$\{x \mid \exists\, u: \text{PMZuteilung}(u) \wedge u[pnr] = x[pnr]\}$$

Quantoren

Atome

- **Einführung einer Sprache**
  - Syntax
  - Semantik

2.3.1 Syntax des TK

- Eine Formel setzt sich zusammen aus **Atomen** der Form
  - **r(t)**

    r ist eine Relation und t ist eine Tupelvariable mit $t \in r$.
  - **t[A] θ u[B]**

    t und u sind Tupelvariablen, A und B sind Attribute, θ ein relationaler Operator.
  - **t[A] θ c**

    t ist eine Tupelvariable, A ein Attribut, c eine Konstante aus dom(A) und θ ein relationaler Operator

- **Beispiele:**
  - Personal(t), t[Note] > 4, t[abtnr] = u[abtnr]

Zusammengesetzte Formeln

- **Eine Formel ist entweder**
  - ein Atom

  oder rekursiv durch folgende Ausdrücke definiert (Ann.: *f und g sind Formeln):*
  - **$f \wedge g$, $f \vee g$, $\neg f$, (f)**
  - **$\forall t: f(t)$, $\exists t: f(t)$**
    - t sind (freie) Variablen in der Formel f

- **Beispiele**
  - $\neg t[Note] > 4$
  - $(t[pnr] = u[pnr]) \vee \neg t[Note] > 4$
  - $PMZuteilung(t) \wedge (t[pnr] = u[pnr]) \vee \neg t[Note] > 4$
  - $\exists t: PMZuteilung(t) \wedge (t[pnr] = u[pnr]) \vee \neg v[Note] > 4$

Freie und gebundene Variablen

- **Informell**
  - Durch Angabe eines Quantors direkt vor einer freien Variablen wird diese gebunden.
- **Formale Definition (freie / gebundene Variablen)**
  - Das Auftreten einer Tupelvariablen in einem Atom ist stets frei.
  - Für f := $\neg$g und f := (g) sind alle freien Variablen von g auch frei in *f.*
  - Für f := g $\wedge$ h und f := g $\vee$ h sind die Variablen in *f frei, falls sie in g oder h frei sind.*
  - Sei *t eine freie Variable in g. Dann wird durch f :=* $\forall$t: g(t), f:= $\exists$t: g(t) t *zu einer* ***gebundenen Variable*** *in f.*

- **Beispiel**

  $\forall$ t: $\neg$PMZuteilung(t) $\vee$ t[Note] > 4

Ausdruck im TK

- **Definition**
  - Ein **Ausdruck im Tupelkalkül** hat die Form

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

    wobei t die einzig freie Variable in der Formel f ist.

- Das Schema einer Tupelvariable t ergibt sich implizit dadurch, dass genau die Attribute dazugehören, die von t benötigt werden.
  - Beispiel
    - $\{x \mid \exists\ u\text{: PMZuteilung}(u) \wedge u[pnr] = x[pnr]\}$
      Das Schema von x ist {pnr} und von u ist {mnr,pnr,Note}.

  - Man darf auch ein Schema RS explizit zu einer Variable hinzufügen.

    $$\{t(RS) \mid f(t)\}$$

Substitution in einer Formel

- Sei f eine Formel mit einer freien Variable x. Dann ist

  $$f(x/t)$$

  die **Substitution** der Tupelvariable x in f durch das Tupel t. In jedem Atom, das ein freies Auftreten von x enthält, gehen wir folgendermaßen vor:
  - r(x) wird ersetzt durch “wahr”, falls $t \in r$. Andernfalls durch “falsch”.
  - $x[A]\ \theta\ u[B]$ wird ersetzt durch $c\ \theta\ u[B]$ mit $c = t[A]$
  - $x[A]\ \theta\ c$ wird ersetzt durch “wahr”, falls $t[A]\ \theta\ c$ gilt. Andernfalls durch “falsch”.

- Durch Substitution gewinnt man eine Formel, die nur noch die Konstanten “wahr” und “falsch” sowie Atome mit gebundenen Variablen enthält.

Beispiele

- **Beispiel**
  - Ein Ausdruck des TK mit Formel:

    $\{t \mid \forall u: \neg PMZuteilung(u) \vee \neg u[pnr] = t[pnr] \vee u[Note] < t[Note]\}$
  - In der Formel ist Variable u gebunden und Variable t frei.
  - Ersetzen von Variable t mit t[pnr] = 73 und t[Note] = 5:

    $\forall u: \neg PMZuteilung(u) \vee \neg u[pnr] = 73 \vee u[Note] < 5$

- **Beispiel**
  - Ein Ausdruck des TK mit Formel:

    $\{u \mid \neg PMZuteilung(u) \vee \neg u[pnr] = 73 \vee u[Note] < 5\}$
  - Die Variable u ist frei in der Formel.
  - Ersetzen von u durch u[pnr] = 51, u[mnr] = 93 und u[Note] = 2:

    $\neg$ wahr $\vee$ $\neg$ falsch $\vee$ wahr

2.3.2 Interpretation einer Formel

- Sei f eine **Formel ohne freie Variablen**. Die **Interpretation** I(f) ist wie folgt definiert:
  - Sei f = “wahr”. Dann ist I(f) := true. Sei f = “falsch”. Dann ist I(f) := false.
  - Sei f = (g). Dann ist I(f) := I(g).
  - Sei f = $\neg$ g. Dann ist I(f) := true genau dann, falls I(g) = false.
  - Sei f = g $\wedge$ h. Dann ist I(f) := true genau dann, falls I(g) = true und I(h) = true.
  - Sei f = g $\vee$ h. Dann ist I(f) := true genau dann, falls I(g) = true oder I(h) = true.
  - Sei f = $\exists$ x: g(x). Dann ist I(f) := true, falls es mindestens ein Tupel t *gibt, so dass I(g(x/t)) = true ist. Andernfalls, I(f) := false*
  - Sei f = $\forall$ x: g(x). Dann ist I(f) := true genau dann, falls für alle t *I(g(x/t))= true gilt. Andernfalls, I(f) := false.*
- Der **Wert des Ausdrucks** { x | f(x) } des Tupelkalküls besteht aus allen Tupeln t, die I(f(x/t)) = true erfüllen.

Beispiel (Anfrage ohne Quantoren)

- **Anfrage**
  - Finde alle Datensätze aus PMZuteilung, so dass die Note mindestens 2 ist.

    $\{x \mid PMZuteilung(x) \wedge x[Note] < 3\}$
- **Auswertung der Formel**
  - x ist eine Variable mit dem Schema {pnr, mnr,Note} und der Wertebereich von pnr und mnr ist die Menge der positiven ganzen Zahlen und von Note die Menge {1,2,3,4,5,6}
  - Iterative Substitution
    - x = (1,1,1)
      - Ist PMZuteilung(1,1,1) $\wedge$ 1 < 3 erfüllt ?
    - x = ….
      - …..
    - …
    - x = (51,93,2)
      - Ist PMZuteilung(51,93,2) $\wedge$ 2 < 3 erfüllt ? ➔ (51,93,2) ist Ergebnis
    - …

Beispiel (mit ∃)

- **Anfrage mit Existenzquantor**
  - Finde alle Personalnummern von Angestellten, die an mindestens einer Maschine ausgebildet sind.

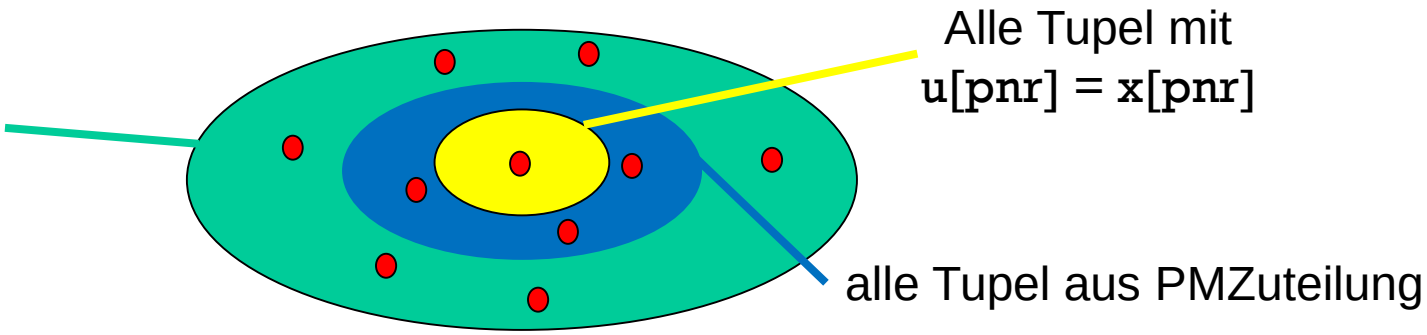
    {x | ∃ u: PMZuteilung(u) ∧ u[pnr] = x[pnr] }
- **Auswertung der Formel**
  - x ist eine Variable mit dem Schema {pnr} und der Wertebereich von pnr ist die Menge der positiven ganzen Zahlen.
  - Iterative Substitution
    - x = 1
      - Ist **∃ u: PMZuteilung(u) ∧ u[pnr] = 1** erfüllt ?
    - x = 2
      - .....
    - …
    - x = 51
      - Ist **∃ u: PMZuteilung(u) ∧ u[pnr] = 51** erfüllt ? ➔ 51 ist Ergebnis
    - …

Beispiel (mit ∀)

- **Formulierung der Anfrage mit Allquantor**
  - Finde alle Personalnummern **der Angestellten**, die an keiner Maschine genügend gut ausgebildet sind.

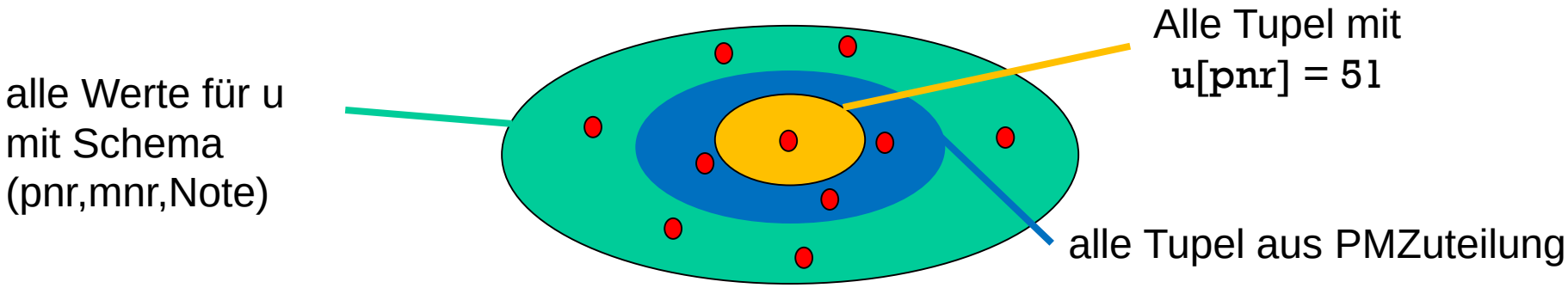
$$\{x \mid (\exists\, u\colon PMZuteilung(u) \wedge u[pnr] = x[pnr]) \wedge (\forall\, u\colon \neg\, PMZuteilung(u) \vee \neg\, u[pnr] = x[pnr] \vee u[Note] > 4)\}$$

  - Die Formel besteht aus zwei mit logischem und verknüpften Teilen.
    - Der erste Teil schränkt die potentiellen Ergebnisse für x auf die Personalnummern aus PMZuteilung ein.
    - Der zweiten Teil der Formel braucht nur noch für diese Personalnummern ausgewertet werden.



Beispiel (mit ∀)

- **Auswertung der Formel mit Allquantor**
  - x ist eine Variable mit dem Schema {pnr} und der Wertebereich von pnr ist die Menge der positiven ganzen Zahlen.
    - x = 1, …, 50 ➔ erster Teil der Formel ist false
    - x = 51 ➔ erster Teil der Formel ist true
      - Ist auch der zweite Teil der Formel erfüllt?
        Ist ∀ u: ¬ **PMZuteilung(u)** ∨ ¬ **u[pnr] = 51** ∨ **u[Note] > 4** erfüllt?



      - Gilt für den einen Datensatz in der gelben Menge, dass u[Note] > 4 ist?
        - Dies ist nicht der Fall, da (51, 93, 2) der einzige Datensatz für die Person mit pnr = 51 ist.

Kurzschreibweisen

- **Einführung von Kurzschreibweisen:**
  - Formeln f, g:

    $$\mathbf{f \Rightarrow g := \neg f \vee g}$$

  - Relation r und Formel f:

    $$\mathbf{\exists\, t\colon r(f(t)) := \exists\, t\colon r(t) \wedge f(t)}$$

  - Relation r und Formel f:

    $$\mathbf{\forall\, t\colon r(f(t)) := \forall\, t\colon \neg r(t) \vee f(t) \Leftrightarrow \forall\, t\colon r(t) \Rightarrow f(t)}$$

Beachte hier das logische oder.

Verständlicher mit Implikation

- **Beispiel**
  - Berechne die pnr der Angestellten, die an keiner Maschine genügend gut ausgebildet sind.

    $$\{\, u \mid (\exists\, u\colon \mathrm{PMZuteilung}(u) \wedge u[pnr] = x[pnr]) \wedge (\forall\, u\colon \mathrm{PMZuteilung}(u) \Rightarrow (u[pnr] = x[pnr] \Rightarrow u[Note] > 4))\}$$

    }

Ausdrucksstärke der Sprache

- **Satz**
  - Das Tupelkalkül (TK) ist ausdrucksstärker als die RA.

- **Beweisidee**
  - Zeige für die Operatoren der relationalen Algebra, dass diese sich durch TK ausdrücken lassen.
    - Projektion, Filter, Vereinigung, Differenz, Kartesisches Produkt, Umbenennung
  - Finde eine Anfrage, die sich in TK ausdrücken lässt, aber nicht in RA.
    - $\{ t \mid \neg \text{PMZuteilung}(t) \}$

TK und RA (1)

- Alle Ausdrücke der Relationenalgebra lassen sich äquivalent auch durch Anfragen im Tupelkalkül ausdrücken:

- Projektion:

RA	TK
$\pi_A(r)$	$\{t \mid \exists\, u: r(u) \wedge t = u[A]\}$

- Selektion:

$\sigma_F(r)$	$\{ v \mid r(v) \wedge F(v)\}$

- Vereinigung (Durchschnitt, Differenz und Produkt analog):

$r \cup s$	$\{ v \mid r(v) \vee s(v) \}$

TK und RA (2)

- Natürlicher Verbund
  - Gegeben zwei Relationen $r_1 \in REL(RS_1), r_2 \in REL(RS_2)$.

$$r_1 \bowtie r_2 = \{t \mid t[RS_1] \in r_1 \text{ und } t[RS_2] \in r_2\}$$

- Division
  - Gegeben zwei Relationen $r_1 \in REL(RS_1), r_2 \in REL(RS_2)$ mit $RS_1 \supseteq RS_2$

$$r_1 \div r_2 = \{t \mid \forall u \in r_2\ \exists\, v \in r_1: \\ t = v[RS_1 - RS_2] \wedge u[RS_2] = v[RS_2]\}$$

Praktische Relevanz ?

- Es können im TK Anfragen mit unendlich große Ergebnismenge formuliert werden.
  - Benutzer sind an solchen Ergebnissen nicht interessiert.
  - Solche Anfragen sind nicht effizient zu berechnen.
  - Leider ist das Problem, ob eine Anfrage eine endliche Ergebnismenge liefert, **nicht entscheidbar**.

Sichere Ausdrücke

- **Syntaktische Einschränkung auf Anfragen mit endlicher Ergebnismenge.**
  - Diese Anfragen werden als **sichere Ausdrücke** bezeichnet.
    - Definition ist relativ komplex
      → siehe das Buch „Datenmodelle …“ von Vossen
  - Sichere Ausdrücke kann man durch Anbinden der Tupelvariablen an eine Relation r erhalten.
    - $\forall t: r(f(t))$
    - $\exists t: r(f(t))$

  Dabei muss f ebenfalls eine sichere Formel sein.

Satz

- **Das TK eingeschränkt auf sichere Ausdrücke hat die gleiche Ausdrucksstärke wie RA.**

RA und SQL

- SQL: Prinzipien werden genauer im nachfolgenden Kapitel behandelt.

- Basiskonzept 'SELECT-FROM_WHERE'-Block

```
SELECT    ID, Kfz, Einwohner, Land
FROM      stadt , stadt_in_land
WHERE     Einwohner  >=  1000 AND Land = 'BY';
```

- Dieser Block kann direkt in ein Ausdruck der RA überführt werden:

$$\pi_{\text{ID, Kfz, Einwohner, Land}} (\sigma_{\text{Einwohner} \geq 1000 \wedge \text{Land = 'BY'}} (\text{stadt} \times \text{stadt_in_land}))$$

- Weitere "RA-Erbschaft": Operatoren UNION, INTERSECT und MINUS zur Verknüpfung von SFW-Blöcken und von komplexeren Anfragen

TK und SQL

- Aber SQL kann auch als TK-Sprache bezeichnet werden.

- Korrespondenz zum TK beim Grundkonzept SFW-Block:

```
SELECT   ID, Kfz, Einwohner, Land
FROM     stadt x, stadt_in_land y
WHERE    x.Einwohner >= 1000 AND y.Land = 'BY';
```

$$\{x \mid stadt(x) \wedge stadt_in_land(y) \wedge x[Einwohner] \geq 1000 \wedge y[Land] = 'BY' \}$$

- Mögliche Verwendung von Quantoren im WHERE-Teil einer SQL-Anfrage zeigt, dass wirklich TK gemeint ist und nicht etwa die (syntaktisch ähnlichen) Selektionsbedingungen der RA.

2.4 Erweiterung der relationalen Algebra

- **Viele interessante Anfragen sind leider weder in RA noch in TK ausdrückbar.**
  - Beispiele:
    - Berechne zu jeder Maschine die Anzahl von Personen, welche die Maschine bedienen können.
    - Liefere die Ergebnisse sortiert.

- **Was ist unser Problem?**
  - In der Mengensemantik gibt es **keine Ordnung**.
  - **Verdichten** der Datensätze ist nicht möglich.
  - **Erhaltung von Duplikaten** wird nicht unterstützt.

Anforderung für zusätzliche Funktionen

- **Bewahrung von Duplikaten (z. B. die durch Projektion entstehen)**
  - Unterstützung von **Multimengen** (engl.: bags)
- **Verdichtung der Daten einer Relation durch Aggregation**
  - Zusätzliche Operatoren in der relationalen Algebra
- **Unterstützung für das Sortieren der Daten**
  - Unterstützung von Folgen statt nur Mengen

2.4.1 M-Relationen

- **Informelle Vorgehensweise**
  - In jeder Relation speichern wir uns zu jedem Tupel noch zusätzlich die Vielfachheit des Tupels ab.
  - Operationen der RA müssen noch erweitert, so dass jetzt die **Multimengensemantik** unterstützt wird.

- **Formale Definition**
  - Sei V $\notin$ U (U ist das Universum der Attribute) mit dom(V) = $\mathbb{N}$. Zu einem Relationenschema RS ist die **M-Relation** r = r(RS) eine endliche Menge von totalen Abbildungen t mit

$$t: RS \cup \{V\} \rightarrow \bigcup_{A \in RS} dom(A) \cup \mathbb{N}$$

    und t(A)$\in$ dom(A) für alle A $\in$ RS und t(V) $\in$ $\mathbb{N}$.
  - Im Folgenden bezeichnet **MREL(RS)** die Menge aller M-Relationen über dem Schema RS.

Kartesisches Produkt

- **Informell**
  - Die Häufigkeit im Ergebnis ergibt sich durch das Produkt der Häufigkeiten in den beiden M-Relationen.

- **Formale Definition**
  - Seien $r_1 \in MREL(RS_1)$, $r_2 \in MREL(RS_2)$ zwei M-Relationen mit $RS_1 \cap RS_2 = \emptyset$. Die Ausgabe von $r_1 \times r_2$ ist eine (temporäre) Relation s mit
    - $REL_s = RS_1 \cup RS_2$
    - $s = \{ t \mid t[RS_1] \in r_1[RS_1]$ und $t[RS_2] \in r_2[RS_2]$ und $\mathbf{t[V] = r_1[V]*r_2[V]}\}$

Kartesisches Produkt

- **Eingabe**
  - Zwei M-Relationen r und s.
- **Ausgabe**
  - Eine M-Relation, in der jedes Tupel aus r mit jedem Tupel aus s verknüpft ist.

r

A	B	V
		2
		3

s

C	D	V
		5
		7

r × s

A	B	C	D	V
				10
				14
				15
				21

Summenvereinigung

- **Informell**
  - Die Zeilen von zwei Relationen werden vereinigt und die Häufigkeit ergibt sich aus der Summe der Einzelhäufigkeiten

- **Formale Definition**
  - Seien $r_1, r_2 \in$ MREL(RS) zwei Relationen über dem gleichen Schema RS. Die Ausgabe von $r_1 \cup r_2$ ist eine (temporäre) Relation s mit
    - $REL_s = RS$
    - $s = \{t \mid t[RS] \in r_1$ oder $t[RS] \in r_2$ und $\mathbf{t[V] = r_1[V] + r_2[V]}\}$

Multimengensemantik!

Vereinigung (Beispiel)

- **Eingabe**
  - Zwei M-Relationen r und s mit gleichem Schema.
- **Ausgabe**
  - Eine M-Relation, in der jedes Tupel aus r und jedes Tupel aus s vorkommt (Vielfachheit: Summe).

r

A	B	V
		2
		3

s

A	B	V
		5
		7

$r \cup s$

A	B	V
		9
		3
		5

Verallgemeinerung der Projektion

- **Informell**
  - Statt einer Projektion sollen beliebige Funktionen erlaubt sein, die ein Tupel in ein anderes Tupel überführen.

    → Map-Operator funktionaler Sprachen

- **„Formale“ Definition**
  - Sei RS ein Schema und r $\in$ MREL(RS) eine M-Relation. Weiterhin sei X ein Relationenschema und μ: RS → X eine Funktion, die zu einem Tupel aus r ein Tupel mit dem Schema X erzeugt. *Dann gilt:*
    - $RS_s = X$
    - $s = \{ t \mid t[X] = \mu(u) \text{ mit } u \in r \text{ und } t[V] = \sum_{t[X]=\mu(u) \wedge u \in r} u[V] \}$

Projektion mit Duplikaterhaltung

- **Relation PMZuteilung projiziert auf Attribut pnr**
  - Man beachte dabei, dass die Definition sich auf mengenwertige Relationen direkt übertragen lässt.

PMZuteilung

pnr	mnr	Note
67	84	3
67	93	2
67	101	3
73	84	5
114	93	5
114	101	3
51	93	2
69	101	2
333	84	3
701	84	2
701	101	2
82	101	2

π_{pnr}(PMZuteilung)

pnr	V
67	3
73	1
114	2
51	1
69	1
333	1
701	2
82	1

Projektion mit arithmetischem Ausdruck

- **Relation PMZuteilung auf Attribut pnr.**
  - Umwandlung Note in Notenpunkte mit folgender Formel
    - return (Note == 6) ? 0 : 14 - (Note - 1) * 3

PMZuteilung

pnr	mnr	Note
67	84	3
67	93	2
67	101	3
73	84	5
114	93	5
114	101	3
51	93	2
69	101	2
333	84	3
701	84	2
701	101	2
82	101	2

$\pi_{pnr,mnr,np \leftarrow (note == 6)\ ?\ 0 : 14-(note-1)*3}$ (PMZuteilung)

pnr	mnr	np	V
67	84	8	1
67	93	11	1
67	101	8	1
73	84	2	1
114	93	2	1
114	101	8	1
51	93	11	1
69	101	11	1
333	84	8	1
701	84	11	1
701	101	11	1
82	101	11	1

Multimengen in Flink

- Die Klasse Table unterstützt sowohl Mengen als auch Multimengen.
  - Es gibt zusätzliche Operatoren für die Berechnung von Multimengen.
    - Table unionAll(Table right)
      - Diese Methode implementiert die Summenvereinigung.
    - Table minusAll(Table right)
      - Diese Methode implementiert die Summendifferenz.
- Projektionsoperator select liefert eine Multimenge (ohne Duplikateliminierung)
    - Sollen die Ergebnisse ohne Duplikate geliefert werden, muss noch zusätzlich der Operator Table distinct() angewendet werden.
    - Zudem kann mit as ein Ausdruck an einen neuen Attributnamen gebunden werden.
- Beispiele
  - Berechne alle Personen, die eine Maschine bedienen können.

    ```
    pmz.select("pnr").distinct();
    ```
  - Berechne eine um 1 bessere Note in der Tabelle pmz.

    ```
    Pmz.filter("Note > 1").select("Note-1 as nn");
    ```

4.4.2 Aggregation

- **Ziel**
  - Berechnung wichtiger Kennzahlen einer Multimenge/Menge
    - Summe (*sum*), *Durchschnitt* (*avg*), *Anzahl* (*count*), *Minimum* (*min*) *und Maximum* (*max*) *von Bedeutung.*

- ***Zwei Arten der Aggregation***
  - *Skalare Aggregation*
    - *Berechnung eines Wertes für eine Multimenge/Menge*
  - *Vektoraggregate*
    - *Berechnung eines Vektors mit Aggregaten für eine Multimenge/Menge.*
      - *Dies erfordert zunächst eine disjunkte Aufteilung in Gruppen.*

Skalare Aggregate

- **Eine Aggregationsfunktion *agg:MREL(RS) → D berechnet zu einer M-Relation r ∈ MREL(RS) einen Wert aus einem Wertebereich D.***
  - Die Funktion count(r) liefert die Anzahl der Tupel in r.
  - Aggregationsfunktionen $avg_A(r)$, $sum_A(r)$, $count_A(r)$ liefern einen numerischen Wert zu einem Attribut A ∈ RS.
  - Aggregationsfunktionen $min_A(r)$, $max_A(r)$ den kleinsten bzw. größten Wert zu dem Attribut A ∈ RS.

- **Das Ergebnis dieser Operation ist keine Tabelle, sondern ein skalarer Wert.**

Beispiele

- **Berechne die Anzahl der Tupel in der Relation PMZuteilung.**
  - count(PMZuteilung)

- **Berechne die Anzahl der Angestellten, die eine Maschine bedienen können.**
  - $count_{pnr}$(PMZuteilung)

- **Berechne die durchschnittlichen Noten der Mitarbeiter**
  - avg_{Note}(PMZuteilung)

Skalare Aggregate in Flink

- **Unterstützung von Aggregaten in der Projektion**
  - Zur Erinnerung
    - Die Methode select unterstützt als Argumente eine Liste von Attributnamen.
  - Zusätzlich kann jetzt noch ein skalares Aggregat angegeben werden.
    - Hierbei ist es zwingend erforderlich mit “as” noch ein Attributname der Ausgabe zu definieren.
    - Beispiele
      - Table res = pmz.select("avg(Note) as a");
      - Table res = pmz.select(“sum(Note) as b”);

Vektoraggregate

- **Zunächst wird die Tabelle in eine möglichst kleine Anzahl von Gruppen aufgeteilt.**
  - Die Aufteilung erfolgt an Hand eines oder mehrerer Attribute der Relation.

1. Partitionierung von PMZuteilung unter Verwendung von {pnr} in Gruppen.
   - In einer Gruppe sind die Tupel, die in dem ausgewählten Attribute gleich sind.

pnr	mnr	Note	V
67	84	3	1
67	93	2	1
67	101	3	1
73	84	5	1
114	93	5	1
114	101	3	1
51	93	2	1
69	101	2	1
333	84	3	1
701	84	2	1
701	101	2	1
82	101	2	1

2. Danach wird für jede Gruppe unter Verwendung eines Aggregats ein Tupel erzeugt.

pnr	avg(Note)
67	8/3
73	5
114	4
51	2
69	2
333	3
701	2
82	2

Das Ergebnis ist eine normale Relation!

Formale Definition

- Eine **Gruppe** ist eine Relation, die das gleiche Schema wie die Eingaberelation hat.
  - Sei $X \subseteq RS$ und $r \in MREL(RS)$. Für zwei Tupel $t_1, t_2 \in r$ mit $t_1[X] = t_2[X]$ gilt, dass diese in der gleichen Gruppe von r liegen.

- Zu einer Aggregation (über einem ausgezeichneten Attribut) wird nun für jede Gruppe **eine** Kennzahl berechnet. Diese Kennzahl wird zusammen mit den Werten der Gruppierungsattribute in der Ergebnisrelation eingetragen.
  - Man kann auch mehrere Aggregate pro Gruppe berechnen.

- Operator: $\gamma_{X,\ A1 \leftarrow agg1,\ A2 \leftarrow agg2,\ \ldots,\ An \leftarrow aggN}(r)$

Beispiele

- **Berechne für jede Maschine die Anzahl der Angestellten, welche die Maschine bedienen können.**

$$\gamma_{mnr,\ C \leftarrow count()}\ (PMZuteilung)$$

- **Berechne für jeden Angestellten seine durchschnittliche Note eine Maschine zu bedienen.**

$$\gamma_{\ pnr,\ C \leftarrow avg(Note)}(PMZuteilung)$$

- **Berechne den Notenspiegel.**

Vektoraggregate in Flink

- **Beispiel**
  - Table counts = orders

    .groupBy("a")

    .select("a, b.count as cnt");
- **Erklärungen**
  - In der groupBy-Methode wird das Gruppierungsattribut bzw. die Gruppierungsattribute angegeben.
  - Danach **muss** eine select-Methode folgen, in der für jede Gruppe genau ein Tupel berechnet wird.
    - Dabei **muss** das Gruppierungsattribut in der Liste sein.
    - Ansonsten können noch für die anderen Attribute Aggregate berechnet werden.
      - Das Aggregat wird ähnlich wie ein Methodenaufruf an das Attribut gehängt.

Duplikatbeseitigung

- **Bei der Duplikatbeseitigung transformiert man eine M-Relation in eine normale Relation.**
  - Streichen der Spalte V.
- **Betrachten wir hierzu die M-Relation von Folie 132, die durch π_{pnr}(PMZuteilung) erzeugt wurde.**
  - Durch Beseitigung der Duplikate erzeugen wir eine mengenwertige Relation mit einer Spalte.
- **Der Operator bekommt wegen seiner Bedeutung das Symbol δ.**

$\delta(\pi_{pnr}(\text{PMZuteilung}))$

pnr
67
73
114
51
69
333
701
82

Sortieren

- **Zusätzlich gibt es den Sortieroperator τ, der die Tupel einer Relation / M-Relation als Folge ausgibt.**
  - Hierzu wird eine Liste L = (A_1, A_2, ...,A_k) mit Attributen aus dem Schema der Relation verwendet. Die Tupel werden lexikographisch bezgl. der Attribute sortiert.

- **Beispiel**
  - $\tau_{\text{Note, pnr}}$(PMZuteilung)

- **Der Sortieroperator wird bei einer Anfrage immer zuletzt angewendet.**
  - Danach kann also kein weiterer Operator folgen.

Sortieren in Flink

- **Beispiel**
  - Table t = pmz.orderBy("mnr, Note.asc");
  - In der Methode orderBy wird als Parameter noch die Liste der Sortierattribute und ggf. noch mitgeteilt, ob absteigend sortiert werden soll.
- **Zusätzlich kann noch angegeben werden, ab welcher Position und wie viele Tupel in der sortierten Folge geliefert werden sollen.**
  - Table t = pmz.orderBy("mnr, Note.asc").fetch(5);
  - Table t = pmz.orderBy("mnr, Note.asc").offset(3);

Zusammenfassung

- **RA & TK**
  - Verarbeitung von Mengen
  - Keine Berechnung von Aggregaten

- **Erweiterung der RA**
  - Unterstützung einer Multimengensemantik
    - Neudefinition der Operationen der RA
  - Unterstützung von weitergehenden Operationen
    - Aggregatberechnung
    - Partitionierung in Gruppen
    - Sortieren