Datenbanksysteme
4 - Die Relationale Algebra

Prof. Dr. Katja Zeume

katja.zeume@w-hs.de

WHS Gelsenkirchen

Inhalt

In dieser Vorlesung

- Die Relationale Algebra
- Operationen in der Relationalen Algebra
- Rechengesetze und Eigenschaften
- Join-Operationen
- Anfragebäume und Anfrageoptimierung (kurz)

Inhalt

In dieser Vorlesung

- **Die Relationale Algebra**
- Operationen in der Relationalen Algebra
- Rechengesetze und Eigenschaften
- Join-Operationen
- Anfragebäume und Anfrageoptimierung (kurz)

Die Relationale Algebra

Bis jetzt: Grundlagen Daten(-bank)entwurf

Die Relationale Algebra

Bis jetzt: Grundlagen Daten(-bank)entwurf

Jetzt: Grundlagen der Datenverarbeitung

Die Relationale Algebra

Bis jetzt: Grundlagen Daten(-bank)entwurf

Jetzt: Grundlagen der Datenverarbeitung

Die **Relationale Algebra** ist über Relationen definiert und wird benutzt um Anfragen über relationale Daten (Tabellen) zu beschreiben und zu optimieren.

Die Relationale Algebra

Bis jetzt: Grundlagen Daten(-bank)entwurf

Jetzt: Grundlagen der Datenverarbeitung

Die **Relationale Algebra** ist über Relationen definiert und wird benutzt um Anfragen über relationale Daten (Tabellen) zu beschreiben und zu optimieren.

Viele praktische DBMS Anwendungen verwenden auch intern die Relationale Algebra um Anfragen zu optimieren und zu berechnen.

$\rightarrow$ Es lohnt sich die Relationale Algebra zu verstehen!

Die Relationale Algebra

Algebra

- Eine **Algebra** allgemein wird über einer Menge von **Operanden** und einer **Operatorenmenge** definiert.
- Die Operationen sind dabei abgeschlossen über der gegebenen Menge, d.h. die Anwendung eines Operators auf einem Element der Menge resultiert wieder in einem Element der Menge.

Die Relationale Algebra

Algebra

- Eine **Algebra** allgemein wird über einer Menge von **Operanden** und einer **Operatorenmenge** definiert.
- Die Operationen sind dabei abgeschlossen über der gegebenen Menge, d.h. die Anwendung eines Operators auf einem Element der Menge resultiert wieder in einem Element der Menge.

- In unserem Kontext betrachten wir Operationen auf Relationen.

Die Relationale Algebra

Algebra

- Eine **Algebra** allgemein wird über einer Menge von **Operanden** und einer **Operatorenmenge** definiert.
- Die Operationen sind dabei abgeschlossen über der gegebenen Menge, d.h. die Anwendung eines Operators auf einem Element der Menge resultiert wieder in einem Element der Menge.

- In unserem Kontext betrachten wir Operationen auf Relationen.
- In der Relationalen Algebra sind die üblichen Mengenoperationen **Vereinigung**, **Schnitt**, und **Mengendifferenz** erlaubt.

Die Relationale Algebra

Algebra

- Eine **Algebra** allgemein wird über einer Menge von **Operanden** und einer **Operatorenmenge** definiert.
- Die Operationen sind dabei abgeschlossen über der gegebenen Menge, d.h. die Anwendung eines Operators auf einem Element der Menge resultiert wieder in einem Element der Menge.

- In unserem Kontext betrachten wir Operationen auf Relationen.
- In der Relationalen Algebra sind die üblichen Mengenoperationen **Vereinigung**, **Schnitt**, und **Mengendifferenz** erlaubt.
- Außerdem dürfen die folgenden Operationen angewendet werden:
  Selektion, Projektion, kartesisches Produkt, Umbenennung.

Inhalt

In dieser Vorlesung

- Die Relationale Algebra
- **Operationen in der Relationalen Algebra**
- Rechengesetze und Eigenschaften
- Join-Operationen
- Anfragebäume und Anfrageoptimierung (kurz)

Mengenoperationen in der Relationalen Algebra

Beispiele für Vereinigung ($\cup$), Schnitt ($\cap$) und Mengendifferenz (-):

Mengenoperationen in der Relationalen Algebra

Beispiele für Vereinigung ($\cup$), Schnitt ($\cap$) und Mengendifferenz (-):

Seien $A = \{(1,1), (1,3), (2,5)\}$ und $B = \{(1,1), (1,2), (3,1)\}$.

- $A \cup B =$
- $A \cap B =$
- $A \setminus B =$

Mengenoperationen in der Relationalen Algebra

Beispiele für Vereinigung ($\cup$), Schnitt ($\cap$) und Mengendifferenz (-):

Seien $A = \{(1,1), (1,3), (2,5)\}$ und $B = \{(1,1), (1,2), (3,1)\}$.

- $A \cup B = \{(1,1), (1,2), (1,3), (2,5), (3,1)\}$
- $A \cap B = \{(1,1)\}$
- $A \setminus B = \{(1,3), (2,5)\}$

Mengenoperationen in der Relationalen Algebra

Beispiele für Vereinigung ($\cup$), Schnitt ($\cap$) und Mengendifferenz (-):

Seien $A = \{(1,1), (1,3), (2,5)\}$ und $B = \{(1,1), (1,2), (3,1)\}$.

- $A \cup B = \{(1,1), (1,2), (1,3), (2,5), (3,1)\}$
- $A \cap B = \{(1,1)\}$
- $A \setminus B = \{(1,3), (2,5)\}$

Frage: Was müssen wir in dem Fall von Tabellenrelationen beachten?

Mengenoperationen in der Relationalen Algebra

- In der Relationalen Algebra müssen die Relationen dasselbe Schema besitzen.

Mengenoperationen in der Relationalen Algebra

- In der Relationalen Algebra müssen die Relationen dasselbe Schema besitzen.
- Nehmen wir an:

Mengenoperationen in der Relationalen Algebra

- In der Relationalen Algebra müssen die Relationen dasselbe Schema besitzen.
- Nehmen wir an:

Tabelle A:

Film-ID	Titel
1	Wonka
2	Dune 2
4	Oppenheimer

Tabelle B:

Film-ID	Titel
1	Wonka
3	Barbie
5	John Wick 4

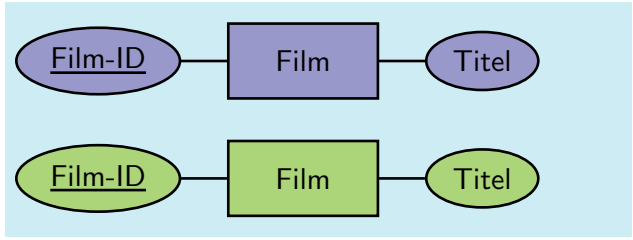
Tabelle C:

Serien-ID	Titel
1	Euphoria
2	Sherlock

- Da die Schemata von `A` & `B` identisch sind, können die Mengenoperationen auf diese Mengen angewendet werden.
- Das Schema von `C` unterscheidet sich von den anderen beiden, daher sind Mengenoperationen wie bsp. $A \cup C$ **undefiniert**.

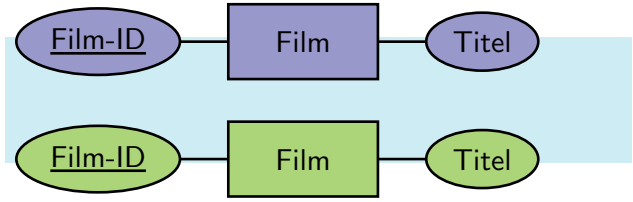
Mengenoperationen in der Relationalen Algebra

- $sch(A \cup B) \rightarrow$



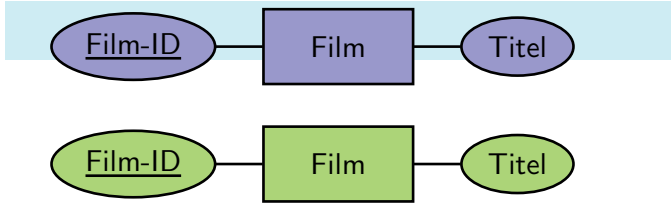
  - Die Mengen A und B sind komplett in der Zielmenge enthalten.

- $sch(A \cap B) \rightarrow$



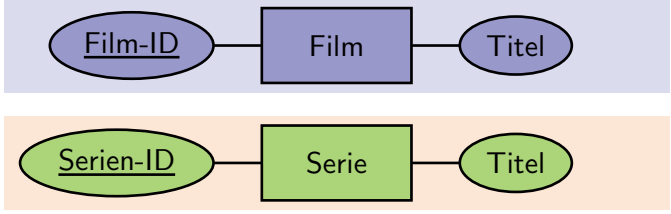
  - Die Zielmenge beinhaltet einen Teil von A und einen Teil B.

- $sch(A \setminus B) \rightarrow$



  - Die Zielmenge beinhaltet den Teil von A, der nicht in B enthalten ist.

- $sch(A \cup C) \rightarrow$



  - Es kann keine Zielmenge definiert werden, da die Schemata nicht übereinstimmen.

Mengenoperationen in der Relationalen Algebra

Tabelle A:

Film-ID	Titel
1	Wonka
2	Dune 2
4	Oppenheimer

Tabelle B:

Film-ID	Titel
1	Wonka
3	Barbie
5	John Wick 4

Mengenoperationen in der Relationalen Algebra

Tabelle A:

Film-ID	Titel
1	Wonka
2	Dune 2
4	Oppenheimer

Tabelle B:

Film-ID	Titel
1	Wonka
3	Barbie
5	John Wick 4

– Für A ∪ B, müssen A und B nicht disjunkt sein.

Tabelle A ∪ B:

Mengenoperationen in der Relationalen Algebra

Tabelle A:

Film-ID	Titel
1	Wonka
2	Dune 2
4	Oppenheimer

Tabelle B:

Film-ID	Titel
1	Wonka
3	Barbie
5	John Wick 4

– Für A ∪ B, müssen A und B nicht disjunkt sein.

Tabelle A ∪ B:

Film-ID	Titel
1	Wonka
2	Dune 2
3	Barbie
4	Oppenheimer
5	John Wick 4

Mengenoperationen in der Relationalen Algebra

Tabelle A:

Film-ID	Titel
1	Wonka
2	Dune 2
4	Oppenheimer

Tabelle B:

Film-ID	Titel
1	Wonka
3	Barbie
5	John Wick 4

Mengenoperationen in der Relationalen Algebra

Tabelle A:

Film-ID	Titel
1	Wonka
2	Dune 2
4	Oppenheimer

Tabelle B:

Film-ID	Titel
1	Wonka
3	Barbie
5	John Wick 4

- Für A \B, muss B keine Teilmenge von A sein.

Tabelle A \B:

Mengenoperationen in der Relationalen Algebra

Tabelle A:

Film-ID	Titel
1	Wonka
2	Dune 2
4	Oppenheimer

Tabelle B:

Film-ID	Titel
1	Wonka
3	Barbie
5	John Wick 4

- Für A \B, muss B keine Teilmenge von A sein.

Tabelle A \B:

Film-ID	Titel
2	Dune 2
4	Oppenheimer

Mengenoperationen in der Relationalen Algebra

Tabelle A:

Film-ID	Titel
1	Wonka
2	Dune 2
4	Oppenheimer

Tabelle B:

Film-ID	Titel
1	Wonka
3	Barbie
5	John Wick 4

Mengenoperationen in der Relationalen Algebra

Tabelle A:

Film-ID	Titel
1	Wonka
2	Dune 2
4	Oppenheimer

Tabelle B:

Film-ID	Titel
1	Wonka
3	Barbie
5	John Wick 4

– Noch ein Beispiel:

Tabelle A ∩ B:

Mengenoperationen in der Relationalen Algebra

Tabelle A:

Film-ID	Titel
1	Wonka
2	Dune 2
4	Oppenheimer

Tabelle B:

Film-ID	Titel
1	Wonka
3	Barbie
5	John Wick 4

– Noch ein Beispiel:

Tabelle A ∩ B:

Film-ID	Titel
1	Wonka

Selektion

Definition

Sei R eine Relation und φ eine beliebige Bedingung. Die **Selektion** $\sigma_{\varphi}(R)$ bildet R auf eine Teilmenge R' ab, in der alle Tupel die Bedingung φ erfüllen.

Selektion

Definition

Sei R eine Relation und φ eine beliebige Bedingung. Die **Selektion** $\sigma_\varphi(R)$ bildet R auf eine Teilmenge R' ab, in der alle Tupel die Bedingung φ erfüllen.

Beispiel:

- Sei R $= \{(1,1),(1,2),(2,1),(2,2),(2,3)\}$ und $sch(R) = (A, B)$. Dann ist $\sigma_{A>1}(R) =$
- $\sigma_{Dauer<120}($

Titel	Dauer
Wonka	117
Dune 2	166
Barbie	114
Oppenheimer	180
John Wick 4	169

$) =$

Selektion

Definition

Sei R eine Relation und φ eine beliebige Bedingung. Die **Selektion** $\sigma_\varphi(R)$ bildet R auf eine Teilmenge R' ab, in der alle Tupel die Bedingung φ erfüllen.

Beispiel:

- Sei R $= \{(1,1),(1,2),(2,1),(2,2),(2,3)\}$ und $sch(R) = (A, B)$. Dann ist $\sigma_{A>1}(R) = \{(2,1),(2,2),(2,3)\}$.
- $\sigma_{Dauer<120}($

Titel	Dauer
Wonka	117
Dune 2	166
Barbie	114
Oppenheimer	180
John Wick 4	169

$) =$

Titel	Dauer
Wonka	117
Barbie	114

Selektion

Die Bedingung φ ist ein boolescher Ausdruck aus folgenden Operanden und Operatoren:

- Konstanten
- Attribute
- arithmetische Operatoren $+$, -, *, ...
- Vergleiche $=, >, <$, ...
- Boolesche Operatoren $\vee, \wedge, /$

Selektion

Die Bedingung φ ist ein boolescher Ausdruck aus folgenden Operanden und Operatoren:

- Konstanten
- Attribute
- arithmetische Operatoren $+$, -, *, ...
- Vergleiche $=, >, <$, ...
- Boolesche Operatoren $\vee, \wedge, /$

φ wird für jedes Tupel einzeln ausgewertet.

Selektion

Die Bedingung φ ist ein boolescher Ausdruck aus folgenden Operanden und Operatoren:

- Konstanten
- Attribute
- arithmetische Operatoren $+$, -, *, ...
- Vergleiche $=, >, <$, ...
- Boolesche Operatoren $\vee, \wedge, /$

φ wird für jedes Tupel einzeln ausgewertet.

Es sind keine Quantoren $(\exists, \forall)$ oder verschachtelte Algebraausdrücke erlaubt.

Projektion

Definition

Sei R eine Relation und L eine Attributsliste. Die **Projektion** $\pi_L(R)$ bildet R auf eine Relation R' ab, in der alle Tupel aus R enthalten in denen die Attribute, die nicht in L sind jeweils entfernt wurden.

Projektion

Definition

Sei R eine Relation und L eine Attributsliste. Die **Projektion** $\pi_L(R)$ bildet R auf eine Relation R' ab, in der alle Tupel aus R enthalten in denen die Attribute, die nicht in L sind jeweils entfernt wurden.

Beispiel:

- Sei R $= \{(1,1,1),(1,2,3),(2,2,2),(3,3,3)\}$ und $sch(R) = (A,B,C)$
  Dann ist $\pi_{A,C}(R) = \{(1,1),(1,3),(2,2),(3,3)\}$.

Projektion

Definition

Sei R eine Relation und L eine Attributsliste. Die **Projektion** $\pi_L(R)$ bildet R auf eine Relation R' ab, in der alle Tupel aus R enthalten in denen die Attribute, die nicht in L sind jeweils entfernt wurden.

Beispiel:

- Sei R $= \{(1,1,1),(1,2,3),(2,2,2),(3,3,3)\}$ und $sch(R) = (A,B,C)$
  Dann ist $\pi_{A,C}(R) = \{(1,1),(1,3),(2,2),(3,3)\}$.

- π kann auch dazu genutzt werden die Spalten einer Tabelle neu zu sortieren.

Projektion

Definition

Sei R eine Relation und L eine Attributsliste. Die **Projektion** $\pi_L(R)$ bildet R auf eine Relation R' ab, in der alle Tupel aus R enthalten in denen die Attribute, die nicht in L sind jeweils entfernt wurden.

Beispiel:

- Sei R $= \{(1,1,1),(1,2,3),(2,2,2),(3,3,3)\}$ und $sch(R) = (A,B,C)$
  Dann ist $\pi_{A,C}(R) = \{(1,1),(1,3),(2,2),(3,3)\}$.

- π kann auch dazu genutzt werden die Spalten einer Tabelle neu zu sortieren.

Intuitiv: σ entfernt Zeilen, π entfernt Spalten.

Projektion

Achtung:
Durch eine Projektion kann sich auch die Anzahl der Spalten in der Tabelle ändern.

Projektion

Achtung:
Durch eine Projektion kann sich auch die Anzahl der Spalten in der Tabelle ändern.

- Sei $R = \{(1,1,1),(1,2,1),(2,2,2),(2,3,2)\}$ und $sch(R) = (A,B,C)$.
  Dann ist $\pi_{A,C}(R) = \{(1,1),(2,2)\}$.

Projektion

Achtung:
Durch eine Projektion kann sich auch die Anzahl der Spalten in der Tabelle ändern.

- Sei $R = \{(1,1,1),(1,2,1),(2,2,2),(2,3,2)\}$ und $sch(R) = (A,B,C)$.
  Dann ist $\pi_{A,C}(R) = \{(1,1),(2,2)\}$.

- Nicht vergessen! Relationen sind hier immer auf Mengen (und nicht auf Multimengen) definiert.

Projektion

Achtung:
Durch eine Projektion kann sich auch die Anzahl der Spalten in der Tabelle ändern.

- Sei $R = \{(1,1,1),(1,2,1),(2,2,2),(2,3,2)\}$ und $sch(R) = (A,B,C)$.
  Dann ist $\pi_{A,C}(R) = \{(1,1),(2,2)\}$.

- Nicht vergessen! Relationen sind hier immer auf Mengen (und nicht auf Multimengen) definiert.

Später: In **SQL** werden diese Duplikate nicht automatisch entfernt, sondern müssen mit `DISTINCT` aus dem Ergebnis gelöscht werden.

Das kartesische Produkt

Definition

Sei R_1 eine n-stellige und R_2 eine m-stellige Relation für beliebige $n, m \in \mathbb{N}$. Das **kartesische Produkt** (oder **Kreuzprodukt**) $R_1 \times R_2$ ist definiert als:

$$R_1 \times R_2 = \{(x_1, \ldots, x_n, y_1, \ldots, y_m) \mid (x_1, \ldots, x_n) \in R_1 \wedge (y_1, \ldots, y_m) \in R_2\}$$

Das kartesische Produkt

Definition

Sei R_1 eine n-stellige und R_2 eine m-stellige Relation für beliebige $n, m \in \mathbb{N}$. Das **kartesische Produkt** (oder **Kreuzprodukt**) $R_1 \times R_2$ ist definiert als:

$$R_1 \times R_2 = \{(x_1, \ldots, x_n, y_1, \ldots, y_m) \mid (x_1, \ldots, x_n) \in R_1 \wedge (y_1, \ldots, y_m) \in R_2\}$$

- R_1 und R_2 müssen dabei keine gemeinsamen Attribute haben.

Das kartesische Produkt

Definition

Sei R_1 eine n-stellige und R_2 eine m-stellige Relation für beliebige $n, m \in \mathbb{N}$. Das **kartesische Produkt** (oder **Kreuzprodukt**) $R_1 \times R_2$ ist definiert als:

$$R_1 \times R_2 = \{(x_1, \ldots, x_n, y_1, \ldots, y_m) \mid (x_1, \ldots, x_n) \in R_1 \wedge (y_1, \ldots, y_m) \in R_2\}$$

- R_1 und R_2 müssen dabei keine gemeinsamen Attribute haben.
- In **SQL**, wird der Verbund wie folgt definiert.

```
SELECT * FROM A,B
```

Das kartesische Produkt

Beispiel:

R:

A	B
1	1

S:

C	D
4	4
4	5

T:

E
8
9

Das kartesische Produkt

Beispiel:

R:

A	B
1	1

S:

C	D
4	4
4	5

T:

E
8
9

$R \times S$:

A	B	C	D
1	1	4	4
1	1	4	5

Das kartesische Produkt

Beispiel:

R:

A	B
1	1

S:

C	D
4	4
4	5

T:

E
8
9

$R \times S$:

A	B	C	D
1	1	4	4
1	1	4	5

$(R \times S) \times T$:

A	B	C	D	E
1	1	4	4	8
1	1	4	5	8
1	1	4	4	9
1	1	4	5	9

Umbennenung

Umbennung

Seien A und B Attribute und R eine Relation. Die **Umbennung** $\beta_{B \leftarrow A}(R)$ benennt die Spalte A in R um nach B.

Umbennenung

Umbennung

Seien A und B Attribute und R eine Relation. Die **Umbennung** $\beta_{B \leftarrow A}(R)$ benennt die Spalte A in R um nach B.

Beispiel:

$$- \beta_{ID \leftarrow Film-ID}($$

Film-ID	Titel
1	Wonka
2	Dune 2
3	Barbie
4	Oppenheimer
5	John Wick 4

$) =$

ID	Titel
1	Wonka
2	Dune 2
3	Barbie
4	Oppenheimer
5	John Wick 4

Die Relationale Algebra

Definition

Die Relationale Algebra ist definiert über den folgenden Operatoren:

1. σ_{φ} Selektion
2. π_{L} Projektion
3. $\beta_{B \leftarrow A}$ Umbenennung
4. $\times$ Kartesisches Produkt
5. $\cup$ Vereinigung
6. $\setminus$ Mengendifferenz

Die Relationale Algebra

Definition

Die Relationale Algebra ist definiert über den folgenden Operatoren:

1. σ_{φ} Selektion
2. π_{L} Projektion
3. $\beta_{B \leftarrow A}$ Umbenennung
4. $\times$ Kartesisches Produkt
5. $\cup$ Vereinigung
6. $\setminus$ Mengendifferenz

- Die Schnittmenge können wir aus $\cup$ und $\setminus$ ableiten und darf deshalb benutzt werden.

Inhalt

In dieser Vorlesung

- Die Relationale Algebra
- Operationen in der Relationalen Algebra
- **Rechengesetze und Eigenschaften**
- Join-Operationen
- Anfragebäume und Anfrageoptimierung (kurz)

Die Relationale Algebra - Rechengesetze

Die Relationale Algebra erfüllt einige allgemeine Rechengesetze.

Die Relationale Algebra - Rechengesetze

Die Relationale Algebra erfüllt einige allgemeine Rechengesetze.

- Das kartesische Produkt ist **assoziativ:**

$$R \times (S \times T) \equiv (R \times S) \times T$$

Die Relationale Algebra - Rechengesetze

Beispiel:

R:

A	B
1	1

S:

C	D
4	4
4	5

T:

E
8
9

Die Relationale Algebra - Rechengesetze

Beispiel:

R:

A	B
1	1

S:

C	D
4	4
4	5

T:

E
8
9

$S \times T$:

C	D	E
4	4	8
4	5	8
4	4	9
4	5	9

$R \times (S \times T)$:

A	B	C	D	E
1	1	4	4	8
1	1	4	5	8
1	1	4	4	9
1	1	4	5	9

$R \times S$:

A	B	C	D
1	1	4	4
1	1	4	5

$(R \times S) \times T$:

A	B	C	D	E
1	1	4	4	8
1	1	4	5	8
1	1	4	4	9
1	1	4	5	9

Die Relationale Algebra - Rechengesetze

Die kartesische Produkt ist **nicht kommutativ** außer in Kombination mit einer Sortierung der Spalten:

$$\pi_L(R \times S) \equiv \pi_L(S \times R)$$

Beispiel:

R:

A	B
1	1

S:

C	D
4	4
4	5

Die Relationale Algebra - Rechengesetze

Die kartesische Produkt ist **nicht kommutativ** außer in Kombination mit einer Sortierung der Spalten:

$$\pi_L(R \times S) \equiv \pi_L(S \times R)$$

Beispiel:

R:

A	B
1	1

S:

C	D
4	4
4	5

$R \times S$:

A	B	C	D
1	1	4	4
1	1	4	5

$S \times R$:

C	D	A	B
4	4	1	1
4	5	1	1

Die Relationale Algebra - Rechengesetze

Die kartesische Produkt ist **nicht kommutativ** außer in Kombination mit einer Sortierung der Spalten:

$$\pi_L(R \times S) \equiv \pi_L(S \times R)$$

Beispiel:

R:

A	B
1	1

S:

C	D
4	4
4	5

$R \times S$:

A	B	C	D
1	1	4	4
1	1	4	5

$S \times R$:

C	D	A	B
4	4	1	1
4	5	1	1

aber $R \times S \equiv \pi_{A,B,C,D}(S \times R)$

Die Relationale Algebra - Rechengesetze

Außerdem gilt: Gegeben σ_φ enthält nur Attribute von R, dann

$$\sigma_\varphi(R \times S) \equiv \sigma_\varphi(R) \times S$$

Beispiel:

R:

A	B
1	1

S:

C	D
4	4
4	5

$R \times S$:

A	B	C	D
1	1	4	4
1	1	4	5

Die Relationale Algebra - Rechengesetze

Außerdem gilt: Gegeben σ_φ enthält nur Attribute von R, dann

$$\sigma_\varphi(R \times S) \equiv \sigma_\varphi(R) \times S$$

Beispiel:

R:

A	B
1	1

S:

C	D
4	4
4	5

$R \times S$:

A	B	C	D
1	1	4	4
1	1	4	5

$\sigma_{A=1}(R)$:

A	B
1	1

Die Relationale Algebra - Rechengesetze

Außerdem gilt: Gegeben σ_φ enthält nur Attribute von R, dann

$$\sigma_\varphi(R \times S) \equiv \sigma_\varphi(R) \times S$$

Beispiel:

R:

A	B
1	1

S:

C	D
4	4
4	5

$R \times S$:

A	B	C	D
1	1	4	4
1	1	4	5

$\sigma_{A=1}(R)$:

A	B
1	1

$$\sigma_{A=1}(R) \times S = \sigma_{A=1}(R \times S) = \begin{array}{cc} \text{A} & \text{B} \\ \hline 1 & 1 \end{array}$$

Die Relationale Algebra - Rechengesetze

Außerdem gilt: Gegeben σ_φ enthält nur Attribute von R, dann

$$\sigma_\varphi(R \times S) \equiv \sigma_\varphi(R) \times S$$

Beispiel:

R:

A	B
1	1

S:

C	D
4	4
4	5

$R \times S$:

A	B	C	D
1	1	4	4
1	1	4	5

$\sigma_{A=1}(R)$:

A	B
1	1

$$\sigma_{A=1}(R) \times S = \sigma_{A=1}(R \times S) = \begin{array}{cc} \text{A} & \text{B} \\ \hline 1 & 1 \end{array}$$

$\rightarrow$ Duplikate fallen durch die Mengensemantik automatisch weg.

Die Relationale Algebra - Rechengesetze

Die Relationale Algebra erfüllt einige allgemeine Rechengesetze.

- Das kartesische Produkt ist **assoziativ:**

$$R \times (S \times T) \equiv (R \times S) \times T$$

- Das kartesische Produkt ist **nicht kommutativ** außer in Kombination mit einer Sortierung der Spalten:

$$\pi_L(R \times S) \equiv \pi_L(S \times R)$$

- Außerdem gilt: Gegeben σ_φ enthält nur Attribute von R, dann

$$\sigma_\varphi(R \times S) \equiv \sigma_\varphi(R) \times S$$

Die Relationale Algebra - Rechengesetze

Die Relationale Algebra erfüllt einige allgemeine Rechengesetze.

- Das kartesische Produkt ist **assoziativ:**

$$R \times (S \times T) \equiv (R \times S) \times T$$

- Das kartesische Produkt ist **nicht kommutativ** außer in Kombination mit einer Sortierung der Spalten:

$$\pi_L(R \times S) \equiv \pi_L(S \times R)$$

- Außerdem gilt: Gegeben σ_φ enthält nur Attribute von R, dann

$$\sigma_\varphi(R \times S) \equiv \sigma_\varphi(R) \times S$$

- Grundlegende Eigenschaften der Relationalen Algebra können bei der Anfrageoptimierung ausgenutzt werden.

Die Relationale Algebra - Eigenschaften

Eine weitere wichtige Eigenschaft im Kontext von Datenbanken ist die **Monotonie** von Anfragen.

Die Relationale Algebra - Eigenschaften

Eine weitere wichtige Eigenschaft im Kontext von Datenbanken ist die **Monotonie** von Anfragen.

Monotonie

In der Relationalen Algebra verhalten sich alle Operatoren außer der Mengendifferenz **monoton**. Seien R_1 und R_2 Relationen und f eine Anfrage der Relationalen Algebra (ohne Mengendifferenz), dann gilt:

$$R_1 \subseteq R_2 \rightarrow f(R_1) \subseteq f(R_2)$$

Die Relationale Algebra - Eigenschaften

Eine weitere wichtige Eigenschaft im Kontext von Datenbanken ist die **Monotonie** von Anfragen.

Monotonie

In der Relationalen Algebra verhalten sich alle Operatoren außer der Mengendifferenz **monoton**. Seien R_1 und R_2 Relationen und f eine Anfrage der Relationalen Algebra (ohne Mengendifferenz), dann gilt:

$$R_1 \subseteq R_2 \rightarrow f(R_1) \subseteq f(R_2)$$

Beispiel:

- $R_1 = \{(1,1),(1,2)\}$ mit $sch(R_1) = (A,B)$ und
  $R_2 = \{(1,1),(1,2),(1,3),(2,1)\}$ mit $sch(R_2) = (A,B)$.
- $\sigma_{A=1}(R_1) = \{(1,1),(1,2)\}$
  $\sigma_{A=1}(R_2) = \{(1,1),(1,2),(1,3)\}$.

Die Relationale Algebra - Eigenschaften

Monotonie

In der Relationalen Algebra verhalten sich alle Operatoren außer der Mengendifferenz **monoton**. Seien R_1 und R_2 Relationen und f eine Anfrage der Relationalen Algebra (ohne Mengendifferenz), dann gilt:

$$R_1 \subseteq R_2 \rightarrow f(R_1) \subseteq f(R_2)$$

- Das heißt, mit mehr Daten in der DB, werden die Ergebnisse dergleichen Anfrage nur größer.
- Es folgt daraus auch, dass ein neu eingefügter Wert eine korrekte Anfrageantwort nicht invalidiert.
- Inklusive der Mengendifferenz verhält sich die Algebra nicht mehr monton.

Inhalt

In dieser Vorlesung

- Die Relationale Algebra
- Operationen in der Relationalen Algebra
- Rechengesetze und Eigenschaften
- **Join-Operationen**
- Anfragebäume und Anfrageoptimierung (kurz)

Kreuzprodukt vs. Join

– Zuvor haben wir gesehen wie wir das **kartesische Produkt** in der Relationalen Algebra und in SQL deklarieren können.

Film x spielt_in:

Film-ID	Titel	Genre	Dauer	Film-ID	Splr-ID
1	Wonka	Family	117	1	4
2	Dune 2	Action	166	1	4
3	Barbie	Comedy	114	1	4
1	Wonka	Family	117	2	1
2	Dune 2	Action	166	2	1
3	Barbie	Comedy	114	2	1

Kreuzprodukt vs. Join

- Zuvor haben wir gesehen wie wir das **kartesische Produkt** in der Relationalen Algebra und in SQL deklarieren können.

Film x spielt_in:

Film-ID	Titel	Genre	Dauer	Film-ID	Splr-ID
1	Wonka	Family	117	1	4
2	Dune 2	Action	166	1	4
3	Barbie	Comedy	114	1	4
1	Wonka	Family	117	2	1
2	Dune 2	Action	166	2	1
3	Barbie	Comedy	114	2	1

- In der Praxis wird der Operator allerdings in den meisten Anfragen durch Gleichheitsbedingungen weitereingeschränkt um Anfragen effizienter zu gestalten.

Kreuzprodukt vs. Join

- Zuvor haben wir gesehen wie wir das **kartesische Produkt** in der Relationalen Algebra und in SQL deklarieren können.

Film x spielt_in:

Film-ID	Titel	Genre	Dauer	Film-ID	Splr-ID
1	Wonka	Family	117	1	4
2	Dune 2	Action	166	1	4
3	Barbie	Comedy	114	1	4
1	Wonka	Family	117	2	1
2	Dune 2	Action	166	2	1
3	Barbie	Comedy	114	2	1

- In der Praxis wird der Operator allerdings in den meisten Anfragen durch Gleichheitsbedingungen weitereingeschränkt um Anfragen effizienter zu gestalten.
  Können Sie das in der relationalen Algebra ausdrücken?

Kreuzprodukt vs. Join

- Zuvor haben wir gesehen wie wir das **kartesische Produkt** in der Relationalen Algebra und in SQL deklarieren können.

Film x spielt_in:

Film-ID	Titel	Genre	Dauer	Film-ID	Splr-ID
1	Wonka	Family	117	1	4
2	Dune 2	Action	166	1	4
3	Barbie	Comedy	114	1	4
1	Wonka	Family	117	2	1
2	Dune 2	Action	166	2	1
3	Barbie	Comedy	114	2	1

- In der Praxis wird der Operator allerdings in den meisten Anfragen durch Gleichheitsbedingungen weitereingeschränkt um Anfragen effizienter zu gestalten.
  Können Sie das in der relationalen Algebra ausdrücken?

Beispiel

$$\sigma_{Film.Film-ID=spielt\_in.Film-ID}(Film \times spielt\_in)$$

Kreuzprodukt vs. Join

Diese Art von Operation definieren wir als **Join**:

Join

Seien R_1 und R_2 Relationen und φ eine beliebige Bedingung. Der **Join** zwischen R_1 und R_2 ist definiert als:

$$R_1 \bowtie_\varphi R_2 = \sigma_\varphi(R_1 \times R_2)$$

In SQL wird ein Join wie folgt deklariert:

Beispiel

```
SELECT *
FROM Film INNER JOIN spielt_in
  ON Film.Film-ID = spielt_in.Film-ID
```

Kreuzprodukt vs. Join

Beispiel

```
SELECT *
FROM Film INNER JOIN spielt_in
  ON Film.Film-ID = spielt_in.Film-ID
```

Kreuzprodukt vs. Join

Beispiel

```
SELECT *
FROM Film INNER JOIN spielt_in
  ON Film.Film-ID = spielt_in.Film-ID
```

Film-ID	Titel	Genre	Dauer	Film-ID	Splr-ID
1	Wonka	Family	117	1	4
2	Dune 2	Action	166	2	1

Kreuzprodukt vs. Join

Beispiel

```
SELECT *
FROM Film INNER JOIN spielt_in
  ON Film.Film-ID = spielt_in.Film-ID
```

Film-ID	Titel	Genre	Dauer	Film-ID	Splr-ID
1	Wonka	Family	117	1	4
2	Dune 2	Action	166	2	1

→ Das Schlüsselwort `INNER` ist optional, verbessert jedoch oft die Lesbarkeit.

Theta Join und Equi Join

Allgemein kann der Join-Operator $\bowtie_\varphi$ in der Relationalen Algebra mit einer beliebigen Bedingung φ benutzt werden.

Theta Join und Equi Join

Allgemein kann der Join-Operator $\bowtie_\varphi$ in der Relationalen Algebra mit einer beliebigen Bedingung φ benutzt werden.
In der Praxis werden jedoch eher zwei Arten von Join-Bedingungen betrachtet.

Theta Join

Seien R_1 und R_2 Relationen und A und B Attribute. Der **Theta Join** zwischen R_1 und R_2 ist definiert als:

$$R_1 \bowtie_\varphi R_2, \text{ wobei } \varphi = A\delta B \text{ und } \delta \in \{=, <>, <, >, <=, >=\}$$

Theta Join und Equi Join

Allgemein kann der Join-Operator $\bowtie_\varphi$ in der Relationalen Algebra mit einer beliebigen Bedingung φ benutzt werden.
In der Praxis werden jedoch eher zwei Arten von Join-Bedingungen betrachtet.

Theta Join

Seien R_1 und R_2 Relationen und A und B Attribute. Der **Theta Join** zwischen R_1 und R_2 ist definiert als:

$$R_1 \bowtie_\varphi R_2, \text{ wobei } \varphi = A\delta B \text{ und } \delta \in \{=, <>, <, >, <=, >=\}$$

Equi Join

Seien R_1 und R_2 Relationen und A und B Attribute. Der **Equi Join** zwischen R_1 und R_2 ist definiert als:

$$R_1 \bowtie_{A=B} R_2.$$

Theta Join und Equi Join Beispiele

Theta Join mit $\neq$ Beispiel

```
SELECT *
FROM Film INNER JOIN spielt_in
  ON Film.Film-ID <> spielt_in.Film-ID
```

Theta Join und Equi Join Beispiele

Theta Join mit $\neq$ Beispiel

```
SELECT *
FROM Film INNER JOIN spielt_in
  ON Film.Film-ID <> spielt_in.Film-ID
```

Film-ID	Titel	Genre	Dauer	Film-ID	Splr-ID
2	Dune 2	Action	166	1	4
3	Barbie	Comedy	114	1	4
1	Wonka	Family	117	2	1
3	Barbie	Comedy	114	2	1

Theta Join und Equi Join Beispiele

Theta Join mit $\neq$ Beispiel

```
SELECT *
FROM Film INNER JOIN spielt_in
  ON Film.Film-ID <> spielt_in.Film-ID
```

Film-ID	Titel	Genre	Dauer	Film-ID	Splr-ID
2	Dune 2	Action	166	1	4
3	Barbie	Comedy	114	1	4
1	Wonka	Family	117	2	1
3	Barbie	Comedy	114	2	1

Equi Join Beispiel

```
SELECT *
FROM Film INNER JOIN spielt_in
  ON Film.Film-ID = spielt_in.Film-ID
```

Theta Join und Equi Join Beispiele

Theta Join mit $\neq$ Beispiel

```
SELECT *
FROM Film INNER JOIN spielt_in
  ON Film.Film-ID <> spielt_in.Film-ID
```

Film-ID	Titel	Genre	Dauer	Film-ID	Splr-ID
2	Dune 2	Action	166	1	4
3	Barbie	Comedy	114	1	4
1	Wonka	Family	117	2	1
3	Barbie	Comedy	114	2	1

Equi Join Beispiel

```
SELECT *
FROM Film INNER JOIN spielt_in
  ON Film.Film-ID = spielt_in.Film-ID
```

Film-ID	Titel	Genre	Dauer	Film-ID	Splr-ID
1	Wonka	Family	117	1	4
2	Dune 2	Action	166	2	1

Natural Join

Natural Join

Seien R_1 und R_2 Relationen. Der **Natural Join** zwischen R_1 und R_2 ist definiert als:

$$R_1 \bowtie R_2,$$

wobei $\bowtie$ ein Equi Join über **allen Spalten mit dem gleichen Namen** ist. (Die Spalten, die im Join verwendet werden, werden nur einmal zu dem Ergebnis hinzugefügt.)

- Ein Natural Join wird oft genutzt um eine Fremdschlüsselbeziehung aufzulösen.
- **Hinweis zum Datenbankentwurf:**
  Nutzen Sie dieselben Namen für gleiche Attribute.
- In **SQL** wird das Schlüsselwort `NATURAL JOIN` benutzt.

Natural Join - Beispiel

Film:

Film-ID	Titel	Erscheinungsdatum	Genre
1	Wonka	2023-12-07	Family
2	Dune:Part Two	2024-03-02	Action
3	Barbie	2023-07-20	Comedy
4	Oppenheimer	2023-07-20	Thriller
5	John Wick 4	2023-03-23	Thriller

spielt_in:

Film-ID	Splr-ID
2	1
1	4
5	5

Natural Join - Beispiel

Film:

Film-ID	Titel	Erscheinungsdatum	Genre
1	Wonka	2023-12-07	Family
2	Dune:Part Two	2024-03-02	Action
3	Barbie	2023-07-20	Comedy
4	Oppenheimer	2023-07-20	Thriller
5	John Wick 4	2023-03-23	Thriller

spielt_in:

Film-ID	Splr-ID
2	1
1	4
5	5

Anfrage:

Selektiere Attribute von `Film` und die zugehörige `Splr-ID` die zu dem Film gehört, aus der Tabelle `spielt_in`.

Natural Join - Beispiel

Film:

Film-ID	Titel	Erscheinungsdatum	Genre
1	Wonka	2023-12-07	Family
2	Dune:Part Two	2024-03-02	Action
3	Barbie	2023-07-20	Comedy
4	Oppenheimer	2023-07-20	Thriller
5	John Wick 4	2023-03-23	Thriller

spielt_in:

Film-ID	Splr-ID
2	1
1	4
5	5

Anfrage:

Selektiere Attribute von `Film` und die zugehörige `Splr-ID` die zu dem Film gehört, aus der Tabelle `spielt_in`.

```
SELECT Film-ID, Titel, Genre, Splr-ID
  FROM Film NATURAL JOIN spielt_in
```

Natural Join - Beispiel

Film:

Film-ID	Titel	Erscheinungsdatum	Genre
1	Wonka	2023-12-07	Family
2	Dune:Part Two	2024-03-02	Action
3	Barbie	2023-07-20	Comedy
4	Oppenheimer	2023-07-20	Thriller
5	John Wick 4	2023-03-23	Thriller

spielt_in:

Film-ID	Splr-ID
2	1
1	4
5	5

Anfrage:

Selektiere Attribute von `Film` und die zugehörige `Splr-ID` die zu dem Film gehört, aus der Tabelle `spielt_in`.

```
SELECT Film-ID, Titel, Genre, Splr-ID
  FROM Film NATURAL JOIN spielt_in
```

$Film \bowtie spielt\_in$:

Film-ID	Titel	Genre	Splr-ID
1	Wonka	Family	4
2	Dune:Part Two	Action	1
5	John Wick 4	Thriller	5

Natural Join - Beispiel und Semi-Join

```
SELECT Film-ID, Titel, Genre
  FROM Film NATURAL JOIN spielt_in
```

Natural Join - Beispiel und Semi-Join

```
SELECT Film-ID, Titel, Genre
  FROM Film NATURAL JOIN spielt_in
```

– Sind die angegebenen Attribute nur Teil der linken Tabelle, so wird ein solcher Join auch **Semi-Join** genannt.

Natural Join - Beispiel und Semi-Join

```
SELECT Film-ID, Titel, Genre
  FROM Film NATURAL JOIN spielt_in
```

- Sind die angegebenen Attribute nur Teil der linken Tabelle, so wird ein solcher Join auch **Semi-Join** genannt.
- Der **Semi-Join** gibt die Attribute der linken Spalte zurück, die übereinstimmende Spalten in der rechten Tabelle besitzen.

Natural Join - Beispiel und Semi-Join

Semi-Join

Seien R_1 und R_2 Relationen. Der **Semi-Join** zwischen R_1 und R_2 ist definiert als:

$$R_1 \ltimes R_2 = \{x \mid x \in R_1 \wedge (x, y) \in R_1 \bowtie R_2 \text{ für ein } y\}$$

Natural Join - Beispiel und Semi-Join

Semi-Join

Seien R_1 und R_2 Relationen. Der **Semi-Join** zwischen R_1 und R_2 ist definiert als:

$$R_1 \ltimes R_2 = \{x \mid x \in R_1 \wedge (x, y) \in R_1 \bowtie R_2 \text{ für ein } y\}$$

In **SQL**:

- Seien R_1 und R_2 Relationen mit $sch(R_1) = (A, B, C)$, dann

```
SELECT A, B, C
  FROM R1 NATURAL JOIN R2
```

Natural Join - Beispiel und Semi-Join

Semi-Join

Seien R_1 und R_2 Relationen. Der **Semi-Join** zwischen R_1 und R_2 ist definiert als:

$$R_1 \ltimes R_2 = \{x \mid x \in R_1 \wedge (x, y) \in R_1 \bowtie R_2 \text{ für ein } y\}$$

In **SQL**:

- Seien R_1 und R_2 Relationen mit $sch(R_1) = (A, B, C)$, dann

```
SELECT A, B, C
  FROM R1 NATURAL JOIN R2
```

$Film \bowtie spielt\_in$:

Film-ID	Titel	Genre
1	Wonka	Family
2	Dune:Part Two	Action
5	John Wick 4	Thriller

Outer Joins

In unserem Beispiel wurden bisher nur Film und ihre zugehörigen Schauspieler aus `spielt_in` gefiltert.

Outer Joins

In unserem Beispiel wurden bisher nur Film und ihre zugehörigen Schauspieler aus `spielt_in` gefiltert.

Was ist mit den Schauspielern die in diesen Filmen nicht mitspielen?

Outer Joins

In unserem Beispiel wurden bisher nur Film und ihre zugehörigen Schauspieler aus `spielt_in` gefiltert.

Was ist mit den Schauspielern die in diesen Filmen nicht mitspielen?

Tatsächlich gibt es weitere Join-Operatoren, die hier helfen:

Outer Joins

In unserem Beispiel wurden bisher nur Film und ihre zugehörigen Schauspieler aus `spielt_in` gefiltert.

Was ist mit den Schauspielern die in diesen Filmen nicht mitspielen?

Tatsächlich gibt es weitere Join-Operatoren, die hier helfen:

```
SELECT * FROM Schauspieler NATURAL LEFT OUTER JOIN
spielt_in
```

Outer Joins

```
SELECT * FROM Schauspieler NATURAL LEFT OUTER JOIN
spielt_in
```

Outer Joins

```
SELECT * FROM Schauspieler NATURAL LEFT OUTER JOIN
spielt_in
```

Schauspieler $\bowtie$ *spielt_in* :

Splr-ID	Vorname	Nachname	Geburtsdatum	Film-ID
1	Zendaya	Coleman	1996-09-01	2
2	Cillian	Murphy	1976-05-25	4
3	Margot	Robbie	1990-07-02	3
4	Timothee	Chalamet	1995-12-27	2
5	Keanu	Reeves	1964-09-02	5
6	Millie Bobby	Brown	2004-02-19	NULL
7	Robert	Pattinson	1986-05-13	NULL

Outer Joins

```
SELECT * FROM Schauspieler NATURAL LEFT OUTER JOIN
spielt_in
```

Schauspieler $\bowtie$ *spielt_in* :

Splr-ID	Vorname	Nachname	Geburtsdatum	Film-ID
1	Zendaya	Coleman	1996-09-01	2
2	Cillian	Murphy	1976-05-25	4
3	Margot	Robbie	1990-07-02	3
4	Timothee	Chalamet	1995-12-27	2
5	Keanu	Reeves	1964-09-02	5
6	Millie Bobby	Brown	2004-02-19	NULL
7	Robert	Pattinson	1986-05-13	NULL

– Das Schlüsselwort `OUTER` ist optional.

Outer Joins

```
SELECT * FROM Schauspieler NATURAL LEFT OUTER JOIN
spielt_in
```

Schauspieler $\bowtie$ *spielt_in* :

Splr-ID	Vorname	Nachname	Geburtsdatum	Film-ID
1	Zendaya	Coleman	1996-09-01	2
2	Cillian	Murphy	1976-05-25	4
3	Margot	Robbie	1990-07-02	3
4	Timothee	Chalamet	1995-12-27	2
5	Keanu	Reeves	1964-09-02	5
6	Millie Bobby	Brown	2004-02-19	NULL
7	Robert	Pattinson	1986-05-13	NULL

- Das Schlüsselwort `OUTER` ist optional.
- `LEFT OUTER JOIN`, `NATURAL RIGHT OUTER JOIN` und `LEFT OUTER JOIN` sind analog definiert.

Outer Joins

```
SELECT * FROM Schauspieler NATURAL LEFT OUTER JOIN
spielt_in
```

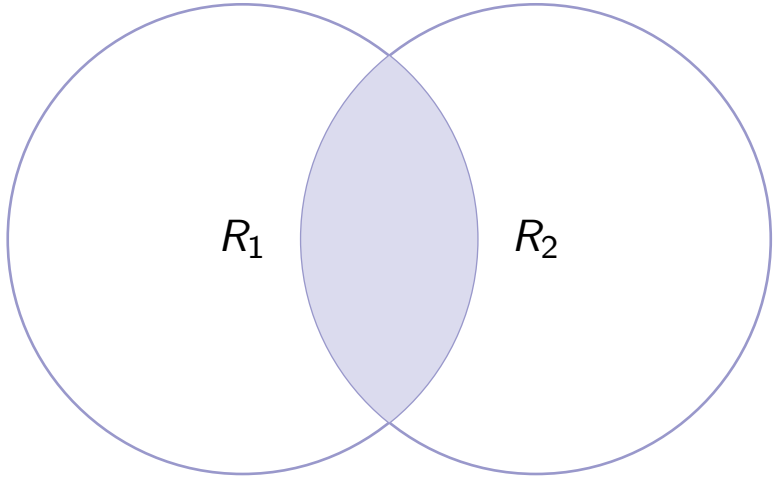
Schauspieler $\bowtie$ *spielt_in* :

Splr-ID	Vorname	Nachname	Geburtsdatum	Film-ID
1	Zendaya	Coleman	1996-09-01	2
2	Cillian	Murphy	1976-05-25	4
3	Margot	Robbie	1990-07-02	3
4	Timothee	Chalamet	1995-12-27	2
5	Keanu	Reeves	1964-09-02	5
6	Millie Bobby	Brown	2004-02-19	NULL
7	Robert	Pattinson	1986-05-13	NULL

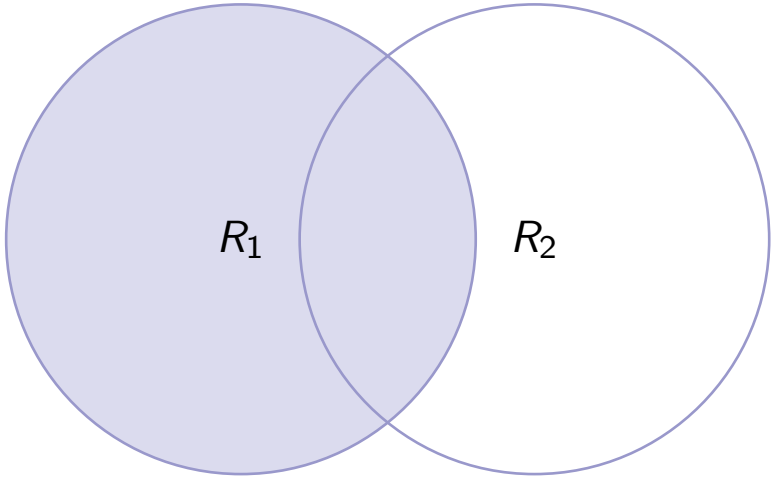
- Das Schlüsselwort `OUTER` ist optional.
- `LEFT OUTER JOIN`, `NATURAL RIGHT OUTER JOIN` und `LEFT OUTER JOIN` sind analog definiert.

→ **Wie soll man sich das merken?**

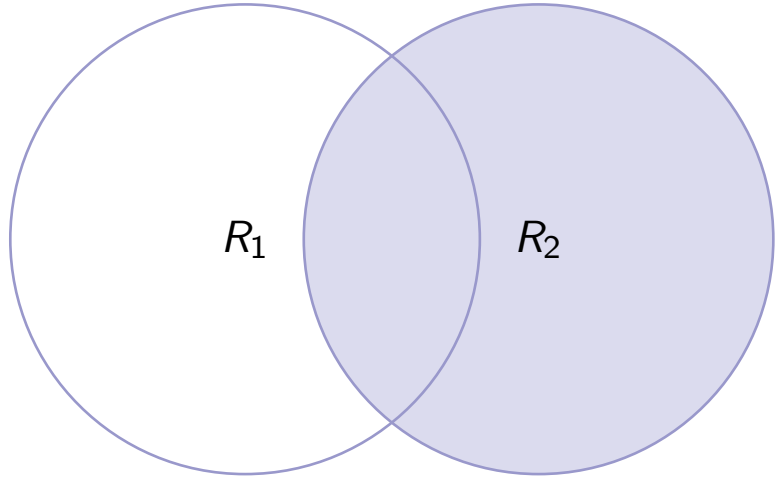
Join Operatoren Übersicht



INNER JOIN
JOIN



LEFT OUTER JOIN
LEFT JOIN



RIGHT OUTER JOIN
RIGHT JOIN

Inhalt

In dieser Vorlesung

- Die Relationale Algebra
- Operationen in der Relationalen Algebra
- Rechengesetze und Eigenschaften
- Join-Operationen
- **Anfragebäume und Anfrageoptimierung (kurz)**

Die Relationale Algebra - Anfragebäume

Bis jetzt:

- Mit Hilfe der relationalen Algebra können SQL Anfragen mathematisch formalisiert werden.

Die Relationale Algebra - Anfragebäume

Bis jetzt:

- Mit Hilfe der relationalen Algebra können SQL Anfragen mathematisch formalisiert werden.
- Diese formalen Darstellungen werden im DBMS benutzt um Anfragen zu optimieren.

Die Relationale Algebra - Anfragebäume

Bis jetzt:

- Mit Hilfe der relationalen Algebra können SQL Anfragen mathematisch formalisiert werden.
- Diese formalen Darstellungen werden im DBMS benutzt um Anfragen zu optimieren.

Jetzt:

- Dafür schauen wir uns im folgenden Anfragebäume an.

Anfragebäume

Anfrage:
Selektiere die Titel aller Filme, in denen "Chalamet" mitspielt.

In SQL:

```
SELECT Titel
  FROM Film, Schauspieler, spielt_in
  WHERE Film.Film-ID = spielt_in.Film-ID
  AND spielt_in.Splr-ID = Schauspieler.Splr-ID
  AND Nachname = 'Chalamet'
```

Anfragebäume

Anfrage:
Selektiere die Titel aller Filme, in denen "Chalamet" mitspielt.

In SQL:

```
SELECT Titel
  FROM Film, Schauspieler, spielt_in
  WHERE Film.Film-ID = spielt_in.Film-ID
  AND spielt_in.Splr-ID = Schauspieler.Splr-ID
  AND Nachname = 'Chalamet'
```

Als **Algebra-Ausdruck**:

$$\pi_{Titel}(\sigma_{Nachname='Chalamet'}((Schauspieler \bowtie Film) \bowtie spielt\_in))$$

Beispiel Datenbank

Film:

Film-ID	Titel	Erscheinungsdatum	Genre
1	Wonka	2023-12-07	Family
2	Dune:Part Two	2024-03-02	Action
3	Barbie	2023-07-20	Comedy
4	Oppenheimer	2023-07-20	Thriller
5	John Wick 4	2023-03-23	Thriller

Schauspieler:

Splr-ID	Vorname	Nachname
1	Zendaya	Coleman
2	Cillian	Murphy
3	Margot	Robbie
4	Timothee	Chalamet
5	Keanu	Reeves

spielt_in:

Film-ID	Splr-ID
2	1
1	4
5	5
2	4

Algebra-Ausdruck:

$$\pi_{Titel}(\sigma_{Nachname='Chalamet'}((Schauspieler \bowtie Film) \bowtie spielt\_in))$$

Anfragebäume

Algebra-Ausdruck:

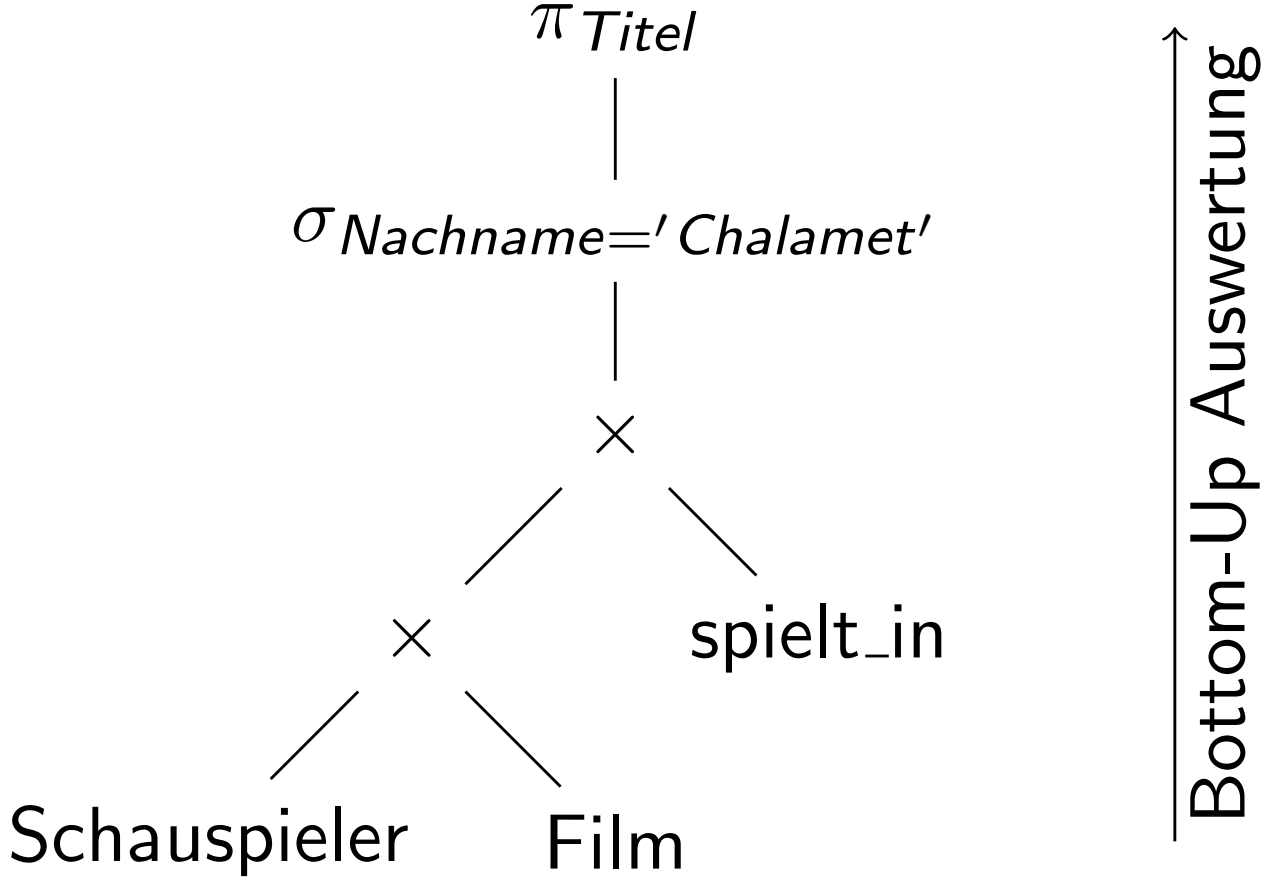
$$\pi_{Titel}(\sigma_{Nachname='Chalamet'}((Schauspieler \bowtie Film) \bowtie spielt\_in))$$

Anfragebäume

Algebra-Ausdruck:

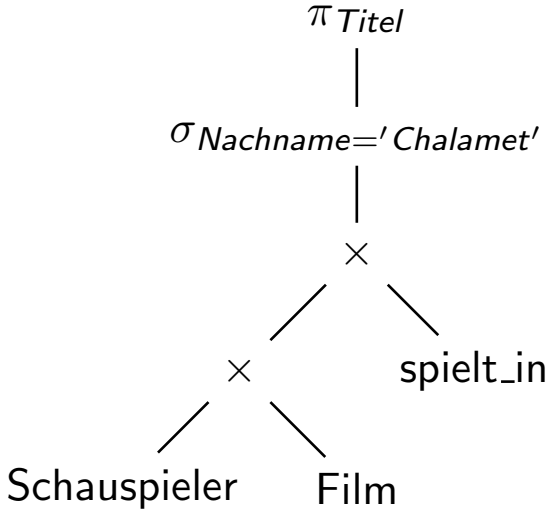
$$\pi_{Titel}(\sigma_{Nachname='Chalamet'}((Schauspieler \bowtie Film) \bowtie spielt\_in))$$

Anfragebaum:



Anfragebäume

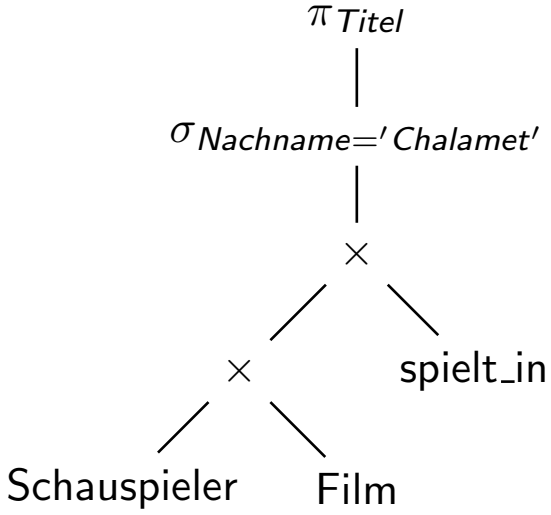
Baum 1:



- Der Anfragebaum gibt eine Evaluationsstrategie (Reihenfolge der Auswertung) vor.
- Diese Strategie führt zu einem Execution Plan im DBMS, der optimiert wird.

Anfragebäume

Baum 1:

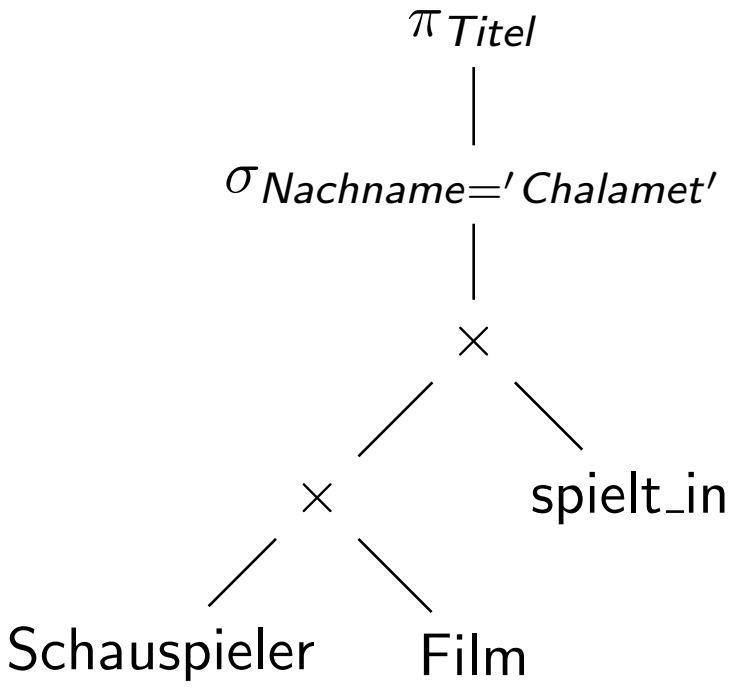


- Der Anfragebaum gibt eine Evaluationsstrategie (Reihenfolge der Auswertung) vor.
- Diese Strategie führt zu einem Execution Plan im DBMS, der optimiert wird.

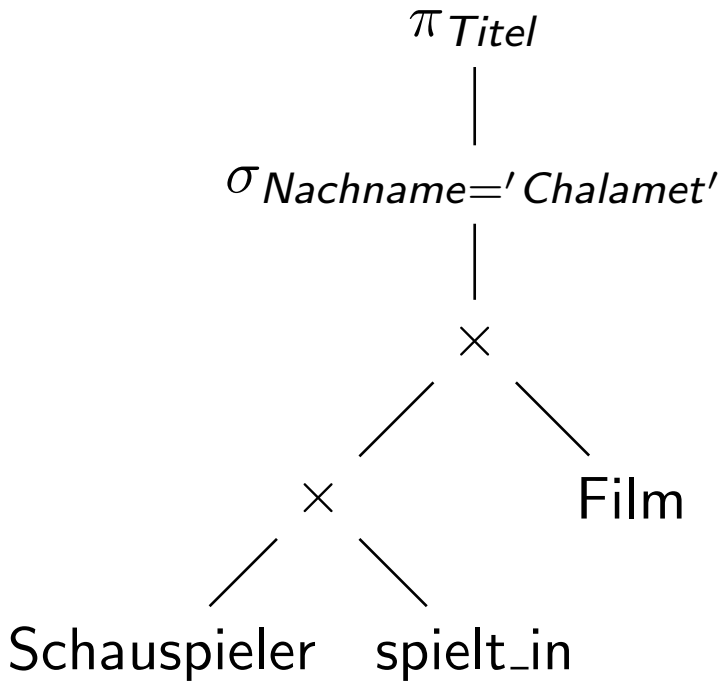
Notiz: Durch die Übersetzung in die Algebra wird auch die deklarative (was) Beschreibung der SQL Anfrage in eine prozedurale Beschreibung (wie) für die Ausführung durch den Computer übersetzt.

Anfragebäume

Baum 1:

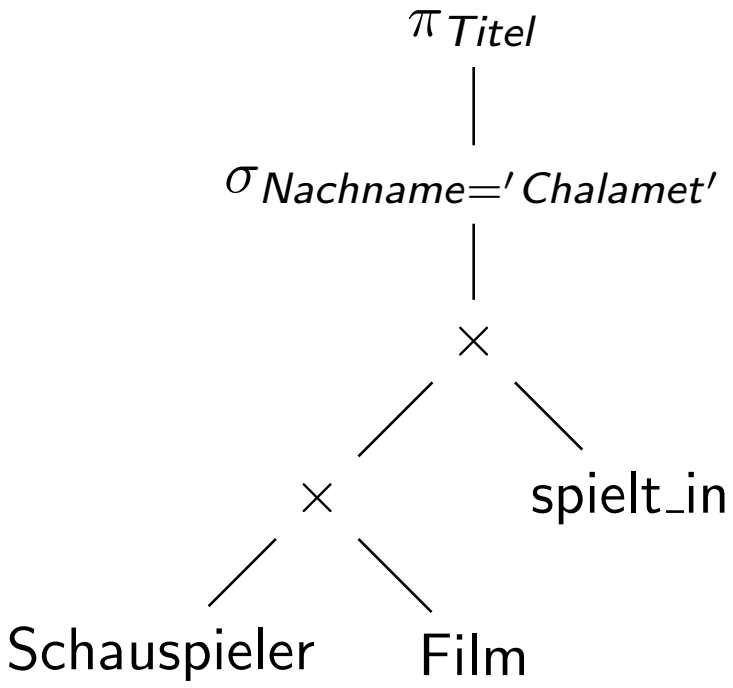


Baum 2:

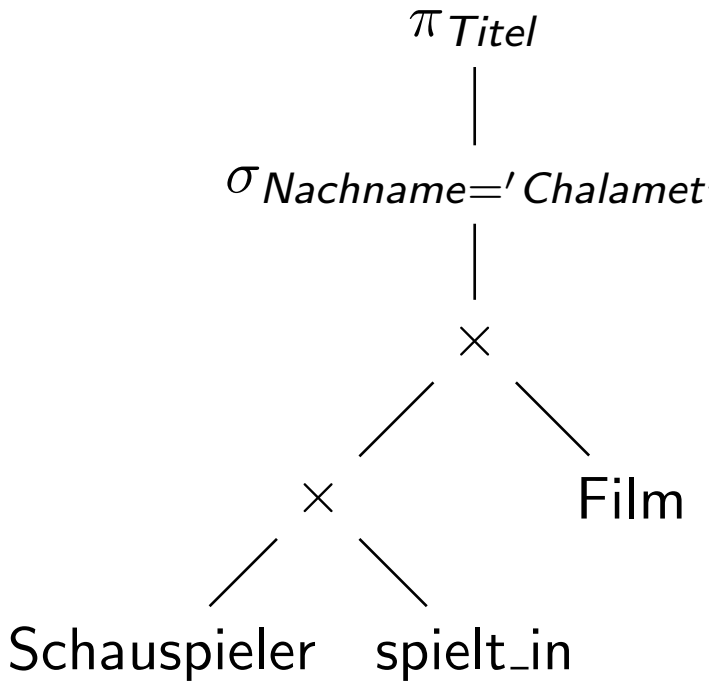


Anfragebäume

Baum 1:



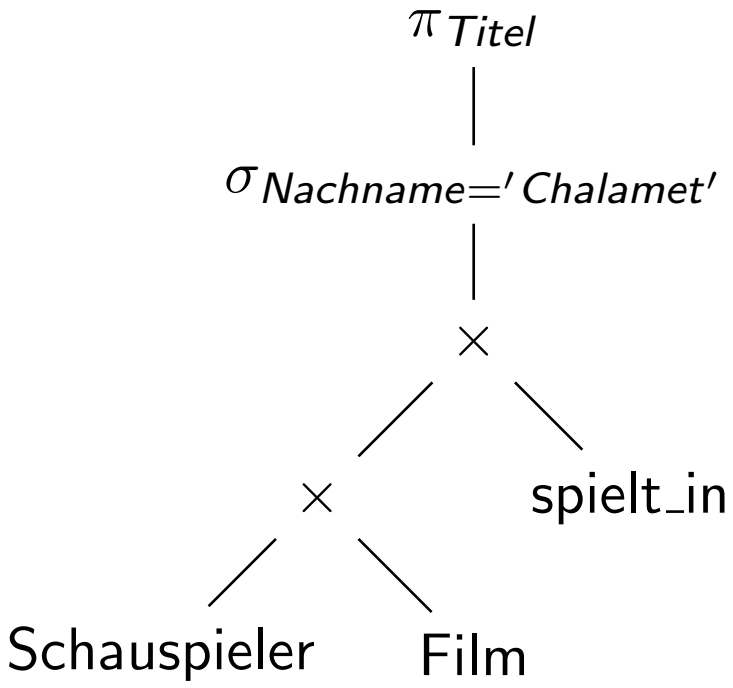
Baum 2:



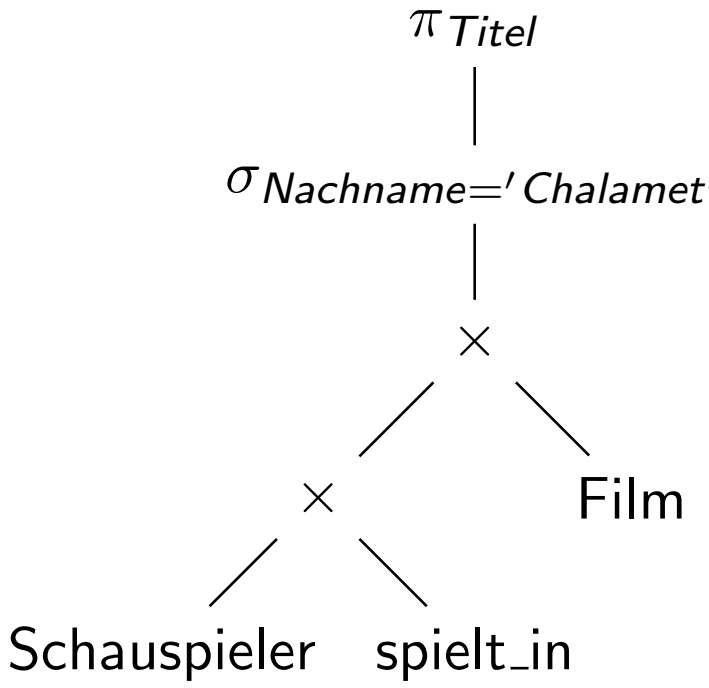
In dem linken Baum ist der Join zwischen Schauspieler und Film wegen dem fehlenden Join-Attribut uneffizient.

Anfragebäume

Baum 1:



Baum 2:

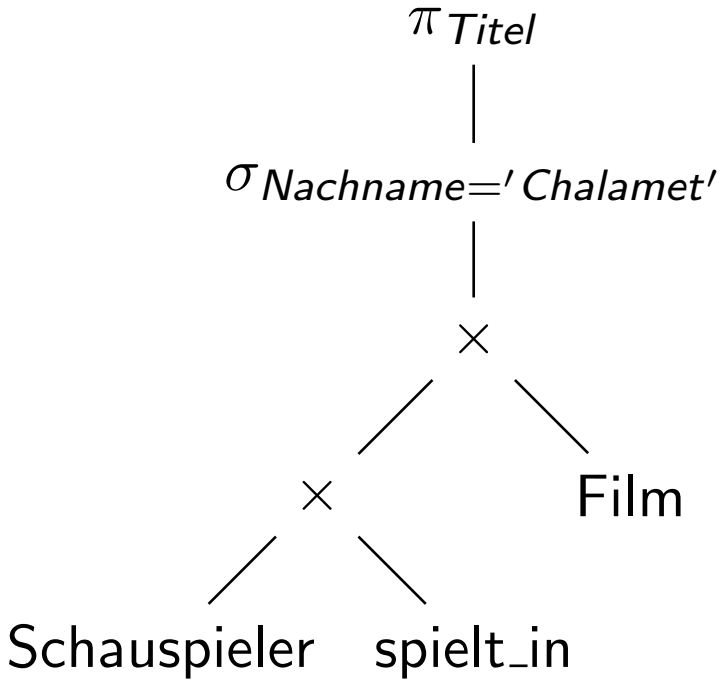


In dem linken Baum ist der Join zwischen Schauspieler und Film wegen dem fehlenden Join-Attribut uneffizient.

Im rechten Baum ist der Join effizienter, **aber geht es noch besser?**

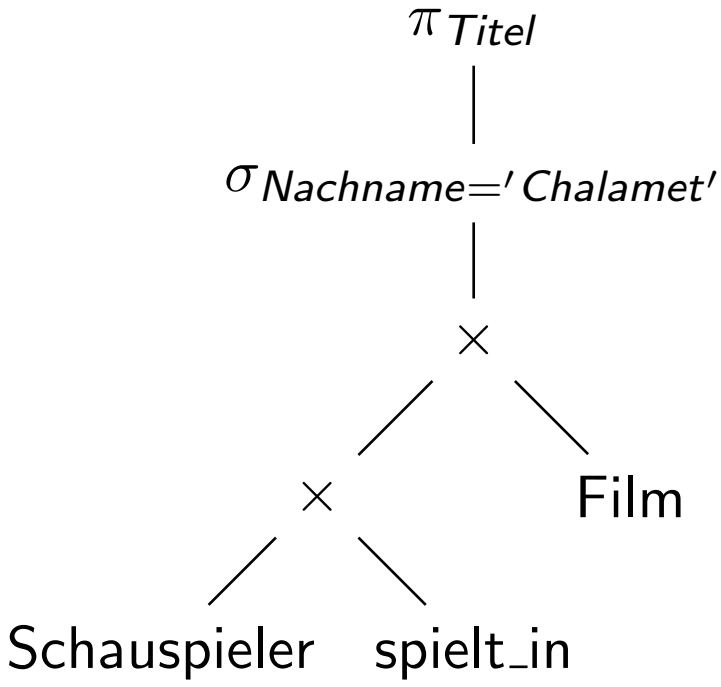
Anfragebäume

Baum 2:

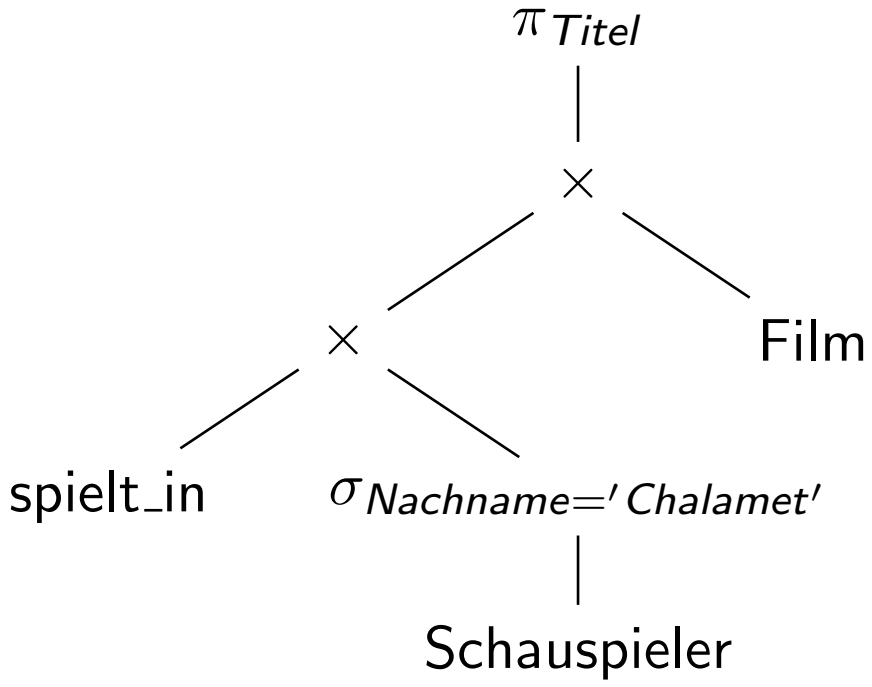


Anfragebäume

Baum 2:

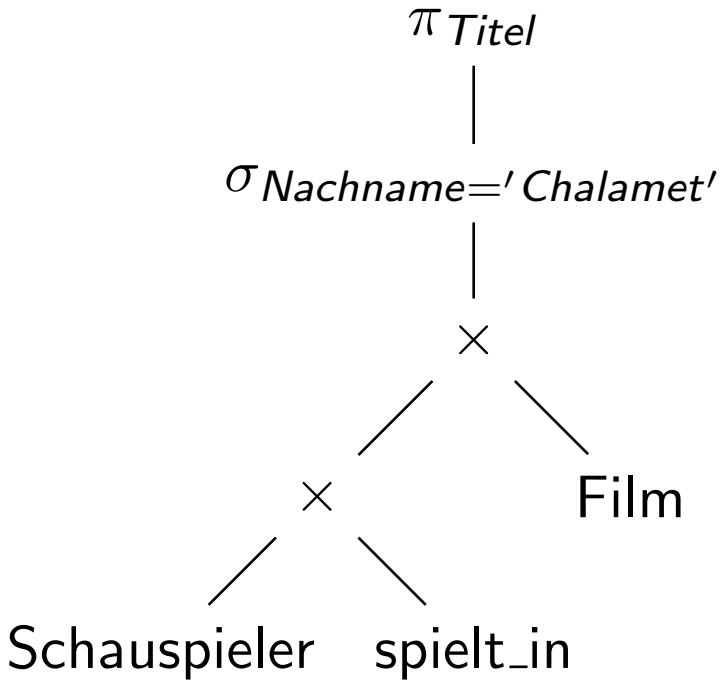


Baum 3:

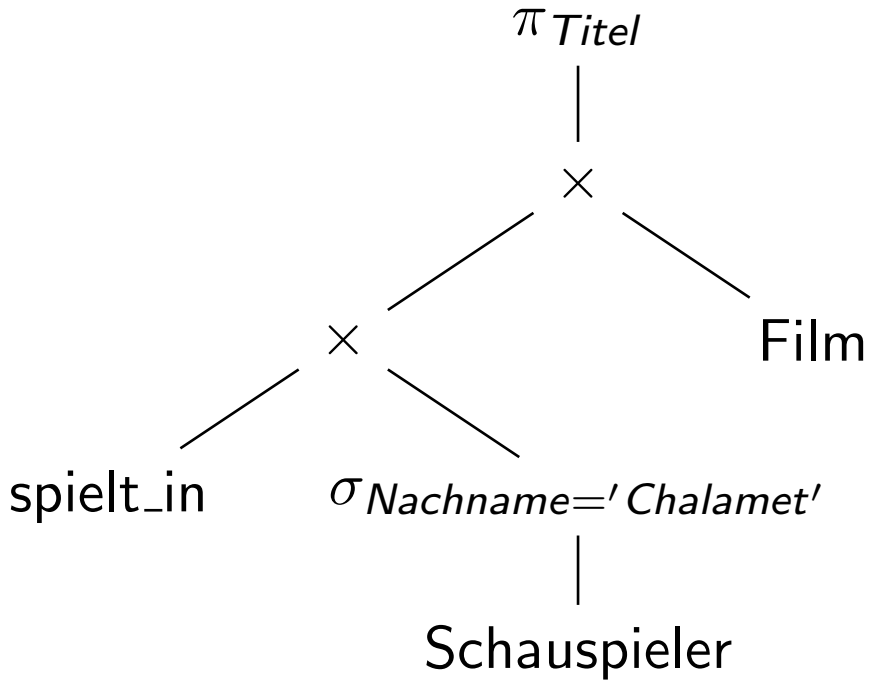


Anfragebäume

Baum 2:



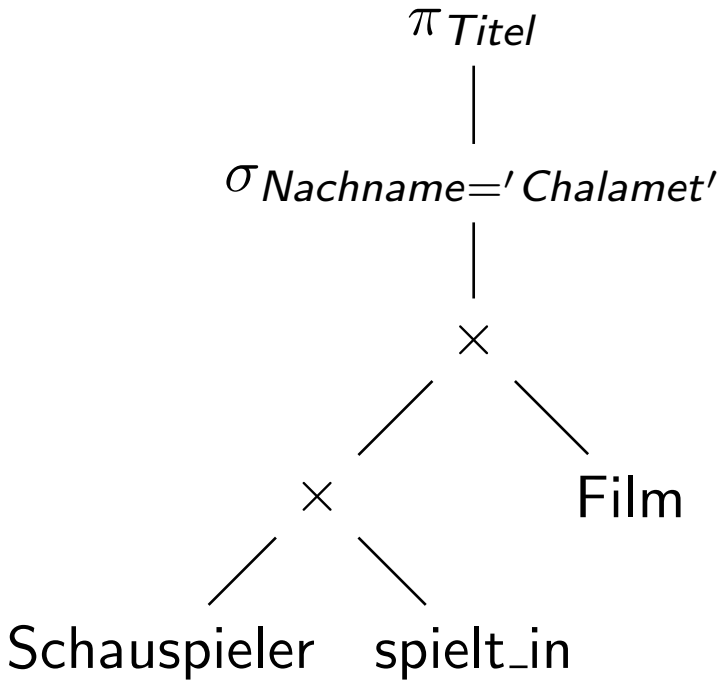
Baum 3:



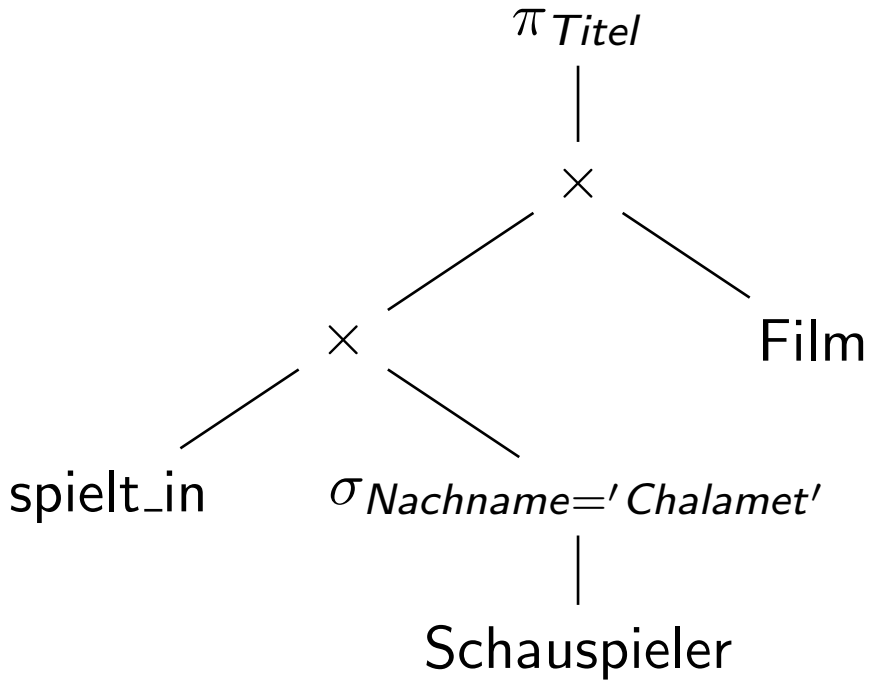
Der linke Baum enthält aufwändige Joins (über großen Tabellen) für ein relativ kleines Anfrageergebnis.

Anfragebäume

Baum 2:



Baum 3:



Der linke Baum enthält aufwändige Joins (über großen Tabellen) für ein relativ kleines Anfrageergebnis.

Frühzeitige Anwendung bekannter Filter reduziert Speicherzugriffe.

Anfrageoptimierung - Zwischenfazit

- Die Auswertungsreihenfolge der einzelnen Operatoren in Anfragen ist essentiell für eine effiziente Auswertung.

Anfrageoptimierung - Zwischenfazit

- Die Auswertungsreihenfolge der einzelnen Operatoren in Anfragen ist essentiell für eine effiziente Auswertung.
- Der Anfrageentwickler kann entscheiden welche SQL Anweisungen und Operatoren verwendet werden. Das DBMS optimiert danach.

Anfrageoptimierung - Zwischenfazit

- Die Auswertungsreihenfolge der einzelnen Operatoren in Anfragen ist essentiell für eine effiziente Auswertung.
- Der Anfrageentwickler kann entscheiden welche SQL Anweisungen und Operatoren verwendet werden. Das DBMS optimiert danach.
- Es gibt Möglichkeiten der internen Optimierung "Hinweise" (Stichwort HINT) zur gewünschten Auswertung mitzugeben.

Anfrageoptimierung - Zwischenfazit

- Die Auswertungsreihenfolge der einzelnen Operatoren in Anfragen ist essentiell für eine effiziente Auswertung.
- Der Anfrageentwickler kann entscheiden welche SQL Anweisungen und Operatoren verwendet werden. Das DBMS optimiert danach.
- Es gibt Möglichkeiten der internen Optimierung "Hinweise" (Stichwort HINT) zur gewünschten Auswertung mitzugeben.
- Anmerkung: Je häufiger eine Anfrage ausgewertet werden soll, desto mehr Aufwand sollte für die Optimierung aufgewendet werden.

Anfrageoptimierung - Zwischenfazit

- Die Auswertungsreihenfolge der einzelnen Operatoren in Anfragen ist essentiell für eine effiziente Auswertung.
- Der Anfrageentwickler kann entscheiden welche SQL Anweisungen und Operatoren verwendet werden. Das DBMS optimiert danach.
- Es gibt Möglichkeiten der internen Optimierung "Hinweise" (Stichwort HINT) zur gewünschten Auswertung mitzugeben.
- Anmerkung: Je häufiger eine Anfrage ausgewertet werden soll, desto mehr Aufwand sollte für die Optimierung aufgewendet werden.
- Das Thema Anfrageoptimierung wird in einem späteren Kapitel noch genauer betrachtet.

Zusammenfassung

In dieser Vorlesung

- Die Relationale Algebra
- Operationen in der Relationalen Algebra
- Rechengesetze und Eigenschaften
- Join-Operationen
- Anfragebäume und Anfrageoptimierung (kurz)

- Heuer, Sattler, Saake. Datenbanken: Konzepte und Sprachen. mitp-Verlag
- Saake, Heuer: Datenbanken - Implementierungstechniken, mitp-Verlag, 2005
- Serge Abiteboul, Rick Hull, Victor Vianu. Foundations of Databases. 1995
- Beispiele und Folien basieren teilweise auf Folien von Prof. Schwentick
- Beispiele und Folien basieren teilweise Folien von Prof. Teubner TU Dortumund
  http://dbis.cs.tu-dortmund.de/cms/de/lehre/ss17/infosys/index.html
- Beispiele und Folien basieren teilweise auf Folien von Prof. Naumann HPI
  https://hpi.de/naumann/teaching/teaching/ss13/datenbanksysteme-i.html