IMDb (Teil 2)
Relationale Algebra

VL Big Data Engineering
(aka Informationssysteme)

Prof. Dr. Jens Dittrich

bigdata.uni-saarland.de

21. Mai 2020

IMDb (Teil 2)

1. Konkrete Anwendung: IMDb

- Das haben wir schon letztes Mal angeschaut.

IMDb (Teil 2)

2. Was sind die Datenmanagement und -analyseprobleme dahinter?

letzte Woche:

Frage 1

Wie werden in IMDb die Daten zu Filmen, Schauspielern, Regisseuren usw. modelliert und abgelegt?

Frage 2

Wie werden in IMDb Verknüpfungen dieser Daten modelliert und abgelegt?

diese Woche:

Frage 3

Wie stellen wir Anfragen an diese Daten?

...

IMDb (Teil 2)

3. Grundlagen, um diese Probleme lösen zu können
   (a) Folien
   (b) Jupyter/Python/SQL Hands-on

- Die relationale Algebra

Wichtigste Lernziele

Relationale Algebra

Operatoren (unär und binär), Selektion (nicht zu verwechseln mit SELECT in SQL), Projektion, Vereinigung, Differenz, Kreuzprodukt, Schnitt, Theta Join, Gruppierung vs Aggregation

Die relationale Algebra

- modulare Anfragesprache zur Umwandlung einer oder mehrerer Eingaberelationen in eine Ausgaberelation
- besteht aus einer Menge von Operatoren:

  unär: $\text{op}([\text{R}_{\text{Eingabe}}]) \rightarrow [\text{R}_{\text{Ausgabe}}]$

  binär: $\text{op}([R_{\text{Eingabe1}}], [\text{R}_{\text{Eingabe2}}]) \rightarrow [\text{R}_{\text{Ausgabe}}]$

- aus historischen Gründen nur unäre und binäre Operatoren (dies ist eigentlich eine unnötige Einschränkung)

Achtung

Die Relationale Algebra beschreibt nur abstrakt **WAS** berechnet werden soll, aber nicht **WIE** diese Berechnung algorithmisch umgesetzt wird!

Stellenwert für Datenanalyse und -verarbeitung:

Vergleichbar mit dem Periodensystem der Elemente in der Chemie

Die relationale Algebra

- gibt es in verschiedenen Varianten/Dialekten
- der Kern der Operatoren (wie im Folgenden erklärt) ist aber immer gleich
- kann sehr leicht erweitert werden
- es gibt viele verschiedene Implementierungen der relationalen Algebra, die aktuell Bekannteste ist Apache Spark.
- wird intern von Systemen zur Anfrageoptimierung verwendet, z.B. in relationalen Datenbanksystemen
- oder: wird vom Nutzer spezifiziert als Datenflussprogramm, z.B. in Apache Spark

Anmerkungen zu Domänen

Wir hatten bereits für Entity-Relationship-Modellierung Domänen (Wertebereiche) eingeführt (siehe IMDb 01, Folie 17):

Im Folgenden gilt:

Domäne (Wertebereich)

Eine Domäne ist eine Menge **atomarer** Werte. Diese Werte dürfen nicht strukturiert (d.h. weiter aufteilbar sein). Domänen werden notiert als D, z.B. integer, float, String, etc.

Vorsicht

Diese Einschränkung ist veraltet und führt zu sehr viel Verwirrung, z.B. der sogenannten ersten Normalform. In der Praxis wird diese Einschränkung meist abgeschwächt. Wir werden darauf zurückkommen, wenn wir (modernes) SQL-99 diskutieren.

Welche atomaren Domänen sind erlaubt?

Aber was für atomare Domänen sind **konkret** erlaubt in ER, im Relationalen Modell, in der Relationalen Algebra, ...?

Erlaubte Domänen in Werkzeug/Modellierungstechnik X

Die erlaubten Domänen werden meist durch das Werkzeug vorgegeben und dann näher spezifiziert. Falls nicht näher spezifiziert nehmen wir kanonische Wertebereiche an wie int/integer, float, string/str/varchar/text, bool/boolean. Die Details dieser Domänen (insbesondere Überlauf) spielen im Moment noch keine Rolle.

Das Periodensystem der Relationalen Algebra

Symbol	Name		Einteilung	
	Deutsch	Englisch	Klasse	unär/binär
σ	Selektion	selection	Basisoperatoren	unär
π	Projektion	projection		unär
$\cup$	Vereinigung	union		binär
$-$	Differenz	minus		binär
$\times$	kartesisches Produkt (oder Kreuzprodukt)	cartesian product		binär
ρ	Umbenennung	rename		unär
$\cap$	Schnitt	intersection	abgeleitete Operatoren	binär
$\bowtie$	Verbund	join		
$\bowtie_\theta$	Theta-Verbund	theta join		
$\bowtie_{[L],[R]}$	Equi-Verbund	equi join		
$\ltimes$	Linker Pseudo-Verbund	left semi join		
$\rtimes$	Rechter Pseudo-Verbund	right semi join		
$\triangleright$	Linker Anti-Pseudo Verbund	left anti semi join		
$\triangleleft$	Rechter Anti-Pseudo Verbund	right anti semi join		
$⟕$	Linker äußerer Verbund	left outer join		
$⟖$	Rechter äußerer Verbund	right outer join		
$⟗$	Äußerer Verbund	full outer join		
γ	Gruppierung (mit Aggregation)	grouping (group by)	Erweiterungen	unär
Γ	Co-Gruppierung (**ohne** Aggregation)	co-grouping		binär
	...			

Basisoperatoren: Selektion σ

Die nachfolgenden Operatoren heißen die **Basisoperatoren** der relationalen Algebra. Viele andere Operatoren bauen hierauf auf bzw. sind *syntaktischer Zucker (syntactic sugar)*.

Selektion (selection/filter) σ

Die Selektion selektiert aus der Eingaberelation R eine Teilmenge R'$\subseteq$ R anhand eines Prädikats P : [R]$\mapsto$ *bool*.
Das Prädikat muss wohledfiniert sein auf [R]. R'=$\{r \in R \mid P(r)\}$.

Beispiele:
$[R_1]$: {[id:int]}, $R_1 = \{(1), (2)\}$,
$[R_2]$: {[id:int]}, $R_2 = \{(2), (3), (4)\}$

P:= id$\leq$ 2

$\sigma_P(R_1) = \{(1), (2)\}$, $\sigma_P(R_2) = \{(2)\}$

directors		
id	**first_name**	**last_name**
78273	Quentin	Tarantino
43095	Stanley	Kubrick
11652	James (I)	Cameron

$\sigma_{id>45000}(directors)$:

id	first_name	last_name
78273	Quentin	Tarantino

$\sigma_{first\_name='Stanley'}(directors)$:

id	first_name	last_name
43095	Stanley	Kubrick

Beispiele (siehe Notebook „Relational Algebra“)

In [11]:
```
exp2 = Projection_ScanBased(newmovies, ['id', 'year'])
```

In [12]:
```
print(exp2)
```

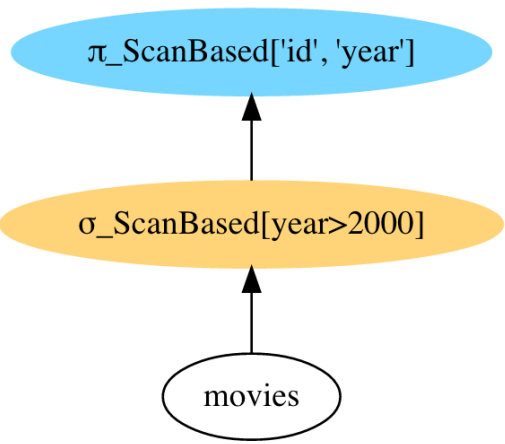
```
π_ScanBased_['id', 'year'](σ_ScanBased_[year>2000](movies))
```

In [13]:
```
exp2.evaluate().print_set()
```

```
[Result] : {[ id:int, year:int ]}
{
        (96779, 2001),
        (393538, 2003),
        (176712, 2004),
        (176711, 2003),
        (127297, 2003),
        (159665, 2006),
        (105938, 2002),
        (10934, 2005)
}
```

In [14]:
```
graph = exp2.get_graph()
Source(graph)
```

Out[14]:



github: Relational Algebra.ipynb

Projektion π

Projektion (projection) π

Die Projektion projiziert die Eingaberelation R auf eine Teilmenge der Attribute der Eingaberelation.
Sei $[R'] \subseteq [R]$ eine beliebige nichtleere Teilmenge der Attribute von R.
Dann ist das Ergebnis der Projektion $R' := \pi_{[R']}(R) := \{r_{[R']} \mid r \in R\}$.
$r_{[R']}$ ist ein Tupel, das nur die Attribute enthält, die in [R'] ethalten sind.

Beispiele:

[R] : {[a:int, b:int]}, R = {(1,3), (2,4), (2,7), (3,3), (4,2)}

$[R'] = \{[a:int]\}$

$\pi_{[R']}(R) = \pi_a(R) = \{ (1), (2), (3), (4) \}$

$[R''] = \{[b:int]\}$

$\pi_{[R'']}(R) = \pi_b(R) = \{ (3), (4), (7), (2) \}$

Projektion und Duplikate

Beachte: Projektion und Duplikate

Es gilt, dass $|\pi_{[R']}(R)| \leq |R|$.

Warum?

Durch die Projektion können Duplikate entstehen. Da eine Relation eine Menge ist, werden die Duplikate nicht mehrfach repräsentiert.

Beispiel:

genre
Comedy
Mystery
Action
Romance
Fantasy
Horror
Music
Film-Noir
Sci-Fi
Drama
Short
Documentary
Adventure
Crime
War
Family
Thriller

$|\pi_{genre}(\text{movies\_genres})| = 17$

aber:

$|\text{movies\_genres}| = 102$

Vereinigung ∪

Vereinigung (union) ∪:

Die Vereinigung vereinigt die Tupel aus zwei Relationen R_1 und R_2 zu einer neuen Relation R'. Es muss gelten: $[R_1] = [R_2]$. Dann ist das Ergebnis der Vereinigung $R' := R_1 \cup R_2$.
Die Anzahl der Tupel in R' ist beschränkt durch
$max(|R_1|, |R_2|) \leq |R'| \leq |R_1| + |R_2|$.

Beispiel: sowohl neue als auch gute Filme in einer Relation

$\sigma_{year>2000}(movies) \cup \sigma_{rank \geq 7.5}(movies)$

id	name	year	rank
10934	Aliens of the Deep	2005	6.5
92616	Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb	1964	8.7
105938	Expedition: Bismarck	2002	7.5
387728	ER	1994	7.7
96779	Earthship.TV	2001	5.6
393538	Jimmy Kimmel Live!	2003	6.7
267038	Pulp Fiction	1994	8.7
121538	Full Metal Jacket	1987	8.2
193519	Lolita	1962	7.6
127297	Ghosts of the Abyss	2003	6.7
328285	Terminator, The	1984	7.9
..	..	..	..

Differenz –

Differenz (minus/except) –

Die Differenz entfernt die Tupel der Relation R_2 aus Relation R_1. Es muss gelten: $[R_1] = [R_2]$. Dann ist das Ergebnis der Differenz $R' := R_1 - R_2$.
Die Anzahl der Tupel in R' ist beschränkt durch $0 \leq |R'| \leq |R_1|$.

Beispiel:

nur neue Filme, die kein schlechtes Ranking haben, in einer Relation:
mit anderen Worten: nur neue, gute Filme

$\sigma_{year>2000}(movies) - \sigma_{rank<7.5}(movies)$

id	name	year	rank
105938	Expedition: Bismarck	2002	7.5
176711	Kill Bill: Vol. 1	2003	8.4
176712	Kill Bill: Vol. 2	2004	8.2
159665	Inglorious Bastards	2006	8.3

Kreuzprodukt $\times$

Kreuzprodukt (cross product, cross join) $\times$

Das Kreuzprodukt bildet das kartesische Produkt[a] von zwei Relationen. D.h. jedes Tupel aus der linken Eingabe wird mit jedem Tupel aus der rechten Relation zu einem neuen Tupel kombiniert.
Falls $[R_1] \cap [R_2] \neq \varnothing$, siehe unten: Eindeutigkeit von Attributnamen. Es gilt: $[R'] = [R_1] \cup [R_2]$. Das Ergebnis des Kreuzproduktes ist $R' := R_1 \times R_2$. $|R'| = |R_1| \cdot |R_2|$.

[a] Die Begriffe Kartesisches- und Kreuzprodukt werden für die relationale Algebra synonym verwendet.

Beispiel:
$[R_1]$: {[a:int]}, $R_1 = \{(1), (2)\}$,
$[R_2]$: {[b:int]}, $R_2 = \{(2), (3), (4)\}$

$R_1 \times R_2 = \{ (1, 2), (1,3), (1,4), (2, 2), (2,3), (2,4)\}$

$|R_1| = 2$, $|R_2| = 3$, $|R_1 \times R_2| = |R_1| \cdot |R_2| = 2 \cdot 3 = 6$.

$[R_1] \cap [R_2] = \varnothing$

Umbenennung ρ

Umbenennung (rename) ρ

Die Umbenennung ändert den Namen einer Relation oder den Namen eines Attributs einer Relation. Für das Umbenennen einer Relation von R zu R_1 gilt $[R'] = [R_1], R' := \rho_{R'}(R)$.
Für das Umbenennen eines Attributs $A \in [R]$ zu A' gilt: $R' := \rho_{A' \leftarrow A}(R)$.

Beispiele:
siehe Python-Notebook

Das Problem mit doppelten Attributnamen

Beispiel:

$[R_1] : \{[\text{id:int}]\}$, $R_1 = \{(1), (2)\}$,
$[R_2] : \{[\text{id:int}]\}$, $R_2 = \{(3), (4)\}$
$\Rightarrow [R_1 \times R_2] = \{[\text{id:int, id:int}]\}$, $R_1 \times R_2 = \{(1,3), (1,4), (2,3), (2,4)\}$

- Wie referenzieren wir im Ergebnis des Kreuzproduktes das Attribut ‘id‘?
- Das ist nicht eindeutig!

Eindeutigkeit von Attributnamen

Falls für die Eingaben eines binären Operators $[R_1] \cap [R_2] \neq \emptyset$, dürfen die Attribute $[R_1] \cap [R_2]$ für Prädikate und Projektionslisten nur qualifiziert referenziert werden. Dies erfolgt durch Voranstellen von '<Relationenname>.'. Ist der Relationenname nicht eindeutig, erfolgt das Voranstellen durch '<Relationenname>$^{\text{<index>}}$.'.

D.h. für das Beispiel:
$\Rightarrow [R_1 \times R_2] = \{[R_1.\text{id:int}, R_2.\text{id:int}]\}$

Das Problem mit doppelten Attributnamen

Beispiel:

Kreuzprodukt auf derselben Relation (ein Selbstkreuzprodukt):

$[R] : \{[\text{id:int}]\}$
$[R \times R] = \{[R^0.\text{id:int}, R^1.\text{id:int}]\}$
$[R_1 \times R_1] = \{[R_1^0.\text{id:int}, R_1^1.\text{id:int}]\}$

Abgeleitete Operatoren: Intersection ∩

Alle nachfolgenden Operatoren können auch durch die Basisoperatoren ausgedrückt werden und sind folglich nur „syntactic sugar“.

Schnitt (intersection) ∩

Der Schnitt bildet die Schnittmenge der Tupel aus zwei Relationen R_1 und R_2. Es muss gelten: $[R_1] = [R_2]$. Dann ist das Ergebnis des Schnitts $R' := R_1 \cap R_2 = R_1 - (R_1 - R_2)$.

Beispiel:

nur neue gute Filme in einer Relation:

$\sigma_{year>2000}(movies) \cap \sigma_{rank\geq 7.5}(movies)$

id	name	year	rank
176711	Kill Bill: Vol. 1	2003	8.4
105938	Expedition: Bismarck	2002	7.5
176712	Kill Bill: Vol. 2	2004	8.2
159665	Inglorious Bastards	2006	8.3

Theta Join $\bowtie_\theta$

Theta-Verbund (theta join) $\bowtie_\theta$

Der Theta-Verbund bildet den Verbund aus zwei Relationen R_1 und R_2 unter dem allgemeinen Join-Prädikat θ. Hieraus wird eine neue Relation TJ erzeugt. Es gilt: $[TJ] = [R_1] \cup [R_2]$. $P : [R_1] \cup [R_2] \to \text{bool}$.
$TJ := R_1 \bowtie_\theta R_2 = \sigma_P(R_1 \times R_2) \subseteq R_1 \times R_2$. Die Anzahl der Tupel im Joinergebnis ist $0 \leq |TJ| \leq |R_1 \times R_2| = |R_1| \cdot |R_2|$.
Falls $[R_1] \cap [R_2] \neq \emptyset$, gelten dieselben Regeln wie für die Eindeutigkeit von Attributnamen beim Kreuzprodukt.

Beispiel:
[R] : {[a:int, b:int]}, R = {(3,1), (4,2), (7,2), (3,3), (7,6)}
[S] : {[c:int, d:int]}, S ={(2,3), (1,4), (5,4), (3,8), (2,5)}

$[TJ] = [R] \cup [S] = \{[a:int, b:int, c:int, d:int]\}$

$\bowtie_\theta = \bowtie_{R.b = S.c} = \{ (3,\underline{1},\underline{1},4), (4,\underline{2},\underline{2},3), (4,\underline{2},\underline{2},5), (7,\underline{2},\underline{2},3), (7,\underline{2},\underline{2},5), (3,\underline{3},\underline{3},8) \}$

$\bowtie_\theta = \bowtie_{R.a = S.d} = \{ (\underline{3},1,2,\underline{3}), (\underline{4},2,1,\underline{4}), (\underline{4},2,5,\underline{4}), (\underline{3},3,2,\underline{3}) \}$

Theta Join-Beispiel für IMDb

Beispiel:

$directors \bowtie_{id=director\_id} movies\_directors$

id	first_name	last_name	director_id	movie_id
43095	Stanley	Kubrick	43095	176891
43095	Stanley	Kubrick	43095	177019
43095	Stanley	Kubrick	43095	250612
11652	James (I)	Cameron	11652	328277
43095	Stanley	Kubrick	43095	30431
43095	Stanley	Kubrick	43095	106666
78273	Quentin	Tarantino	78273	267038
43095	Stanley	Kubrick	43095	65764
78273	Quentin	Tarantino	78273	118367
..	..	..	..	..

Equi Join

Equi-Verbund (equi join)

Der Equi-Verbund bildet den Verbund aus zwei Relationen R_1 und R_2 unter einem Equi-Join-Prädikat. D.h. anstatt das Join-Prädikat direkt zu spezifizieren, wie beim Theta-Join, werden Teilmengen der Schemata von R_1 und R_2 festgelegt, deren Attributwerte gleich sein müssen.

Es seien $[L] \subseteq [R_1]$ sowie $[R] \subseteq [R_2]$, $|[L]| = |[R]|$.

Das Joinprädikat $\theta : R_1.[L] = R_2.[R]$ testet die einzelnen Komponenten **paarweise** auf Gleichheit.
Notation: $R_1 \bowtie_{[L],[R]} R_2$.

Beispiel:
folgende Teilschemata wurden festgelegt: $[L] = \{[ID, Gehalt]\}$ und $[R] = \{[SID, Bonus]\}$

$\Rightarrow \theta : ID = SID \wedge Gehalt = Bonus.$

$$R_1 \bowtie_{[L],[R]} R_2 = R_1 \bowtie_{\{[ID,Gehalt]\},\{[SID,Bonus]\}} R_2 = R_1 \bowtie_{\theta} R_2$$

Erweiterungen: Gruppierung γ

Gruppierung (group by) γ

Die Gruppierung gruppiert eine Eingaberelation anhand einer (möglicherweise leeren) Teilmenge von Attributen und erzeugt ein Tupel für jede so entstandene Gruppe (durch **Aggregation**). Seien $[G] \subseteq [R]$ und $[G_1] \subseteq [R], \ldots, [G_n] \subseteq [R]$ beliebige (auch leere) Teilmengen der Attribute der Relation R. Ferner seien Aggregatfunktionen definiert mit $f_1 : [G_1] \to [D_1], \ldots, f_n : [G_n] \to [D_n]$.
Dann ist das Ergebnis der Gruppierung $R' := \gamma_{[G],f_1([G_1]),\ldots,f_n([G_n])}(R)$.

Beispiele:

$\gamma_{\text{gender, count(*)}}(actors)$

gender	count(*)
F	289
M	802

$\gamma_{\text{first\_name, last\_name, count(*)}}(directors \bowtie_{id=director\_id} movies\_directors)$

last_name	first_name	count(*)
Tarantino	Quentin	10
Kubrick	Stanley	16
Cameron	James (I)	14

Die wichtigsten Aggregatfunktionen

sum(A)

Berechnet die Summe der Werte von Attribut A.

avg(A)

Berechnet den Durchschnitt (arithmetisches Mittel) der Werte von Attribut A.

min(A)

Berechnet das Minimum der Werte von Attribut A.

max(A)

Berechnet das Maximum der Werte von Attribut A.

count(*)

Berechnet die Anzahl der Tupel (unabhängig von einem Attribut).

Gruppierung

Der Name „Gruppierung“ ist für diesen Operator eigentlich irreführend. Denn bei der „Gruppierung“ werden drei verschiedene Operationen durchführt:

1. **Gruppierung:** es werden alle Tupel der Eingaberelation nach den Attributen in [G] gruppiert (mit anderen Worten: **partitioniert**). Alle Tupel, die bezüglich [G] die gleichen Werte haben, kommen in dieselbe Gruppe/Partition.
2. **Aggregation:** für jede in (1.) entstandene Gruppe/Partition werden Aggregatfunktionen $f_1([G_1]), \ldots, f_n([G_n])$ berechnet. D.h. **für jede Gruppe/Partition wird jede dieser Funktionen unabhängig** berechnet.
3. **Projektion:** für jede in (1.) entstandene Gruppe/Partition wird ein Tupel mit den zugehörigen Aggregaten aus (2.) erzeugt. Das Schema dieses Tupels ist $[G, D_1, \ldots, D_n]$.

IMDb (Teil 2)

und damit zurück zu:

2. Was sind die Datenmanagement und -analyseprobleme dahinter?

Frage 3

Wie stellen wir Anfragen an diese Daten?

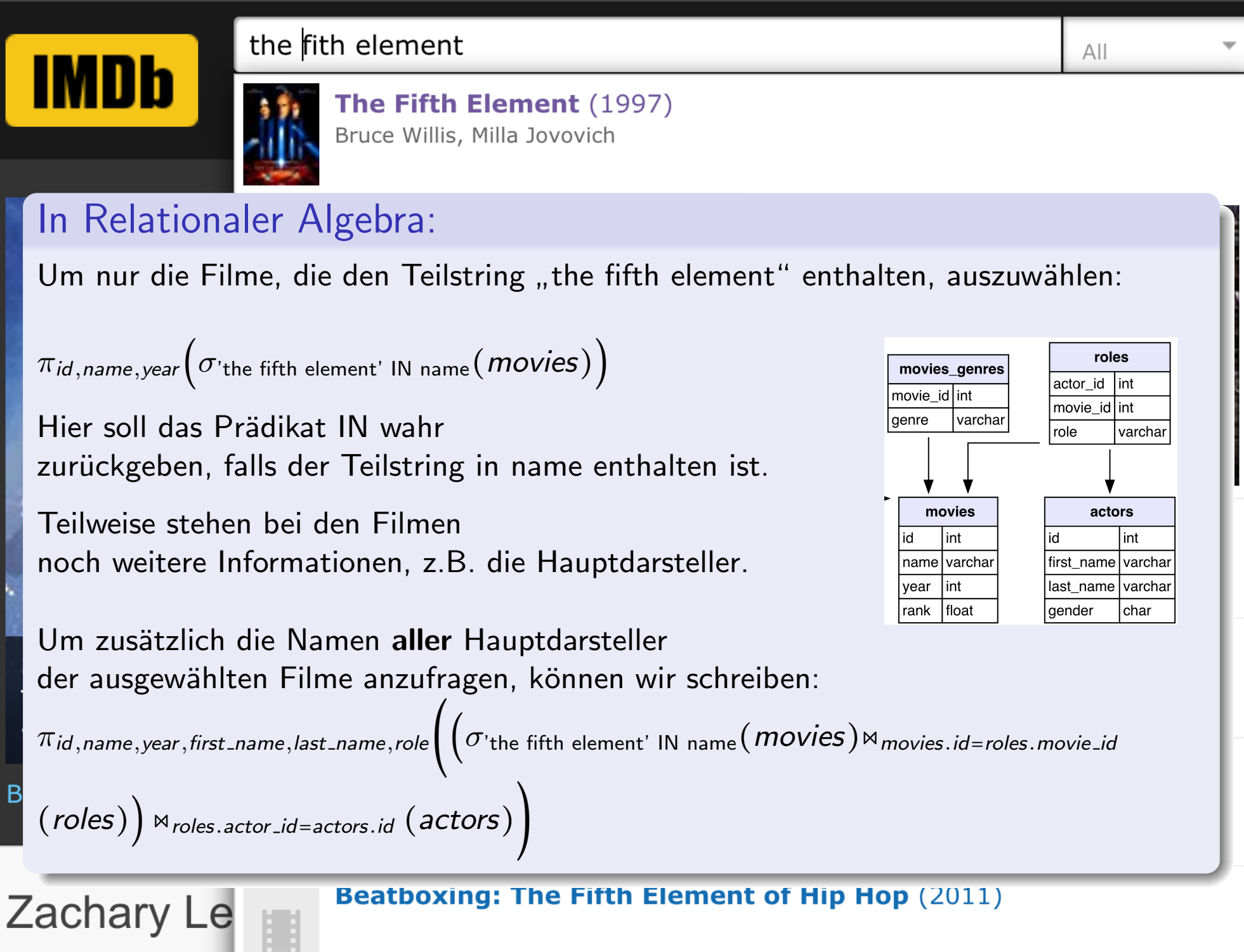
Mit der relationalen Algebra!

Transfer auf IMDb

4. Transfer der Grundlagen auf die konkrete Anwendung

Das haben wir diese und vergangene Woche für ER, das relationale Modell und die relationale Algebra bereits ‘inline’ gemacht in zahlreichen Beispielen.

Aber nochmal zurück zu unserem Anfangsbeispiel:



In Relationaler Algebra:

Um nur die Filme, die den Teilstring „the fifth element“ enthalten, auszuwählen:

$$\pi_{id,name,year}\left(\sigma_{\text{'the fifth element' IN name}}(movies)\right)$$

Hier soll das Prädikat IN wahr
zurückgeben, falls der Teilstring in name enthalten ist.

Teilweise stehen bei den Filmen
noch weitere Informationen, z.B. die Hauptdarsteller.

Um zusätzlich die Namen **aller** Hauptdarsteller
der ausgewählten Filme anzufragen, können wir schreiben:

$$\pi_{id,name,year,first\_name,last\_name,role}\left(\left(\sigma_{\text{'the fifth element' IN name}}(movies) \bowtie_{movies.id=roles.movie\_id} (roles)\right) \bowtie_{roles.actor\_id=actors.id} (actors)\right)$$

Cast

Edit

Cast overview, first billed only:

Bruce Willis ... Korben Dallas

In Relationaler Algebra:

Der Click im Webbrowser auf "The Fifth Element" wählt aus der oben erzeugten Liste den Film mit der id=112205 aus.

Damit können wir die Anfrage so formulieren:

$$\pi_{first\_name,last\_name,role}\left(\sigma_{movies.id=112205}(roles) \bowtie_{roles.actor\_id=actors.id} (actors)\right)$$

Problem:

Was wir **nicht** ausdrücken konnten in relationaler Algebra:

1. nicht alle Hauptdarsteller sondern nur k-viele
2. nur die tollsten, wichtigsten Schauspieler (first billed only)
3. die Reihenfolge der tollsten, wichtigsten Schauspieler
4. ob überhaupt Schauspieler angezeigt werden

Lee Evans ... Fog

Ausblick auf nächste Woche

Leider sind Ausdrücke in relationaler Algebra manchmal etwas unübersichtlich. Deswegen ist es eine andere Option, die relationale Algebra unter einer anderen Sprache zu verstecken, d.h. eine andere Anfragesprache zu konzipieren und diese dann jeweils automatisch in die relationale Algebra zu übersetzen.

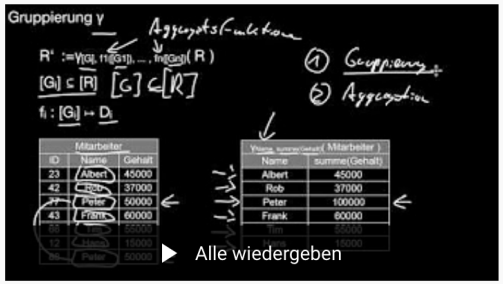
Fundamental Theorem of Software Engineering (FTSE):

„We can solve any problem by introducing an extra level of indirection . . . except for the problem of too many levels of indirection."

[David Wheeler]

Und genau das machen wir nächste Woche...

Weiterführendes Material



Relationale Algebra

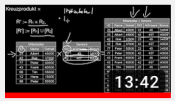
5 Videos • 33.225 Aufrufe • Zuletzt am 27.01.2014 aktualisiert

Öffentlich

Grundlagen der Relationalen Algebra

Prof. Dr. Jens Dittrich

SORTIEREN NACH

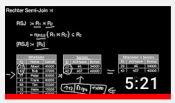


13.17 Relationale Algebra: Selektion, Projektion, Vereinigung, Differenz, Kreuzprodukt, Umbenennung

Prof. Dr. Jens Dittrich

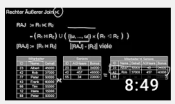
13.18a Relationale Algebra: Schnitt, Theta Join, Equi Join, Natural Join

Prof. Dr. Jens Dittrich



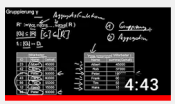
13.18b Relationale Algebra: Semi Joins, Anti Semi Joins

Prof. Dr. Jens Dittrich



13.18c Relationale Algebra: Äussere Joins

Prof. Dr. Jens Dittrich



13.18d Relationale Algebra: Gruppierung und Aggregation

Prof. Dr. Jens Dittrich

Youtube Videos von Prof. Dittrich zu Relationaler Algebra

sowie:

- Kapitel 3.4 in Kemper&Eickler (verfügbar in der Bibliothek)
- RelaX - relational algebra calculator
  - gist für IMDb_sample
  - gist für fotodb