


15.11.2022

Übungsblatt 4 – Lösungsvorschlag

Diskussionsteil (im PS zu lösen; keine Abgabe nötig)

- a)  Gegeben seien folgende Relationen, die einen Blogpost (Post) und dazugehörige Tags (Tag) beinhalten, wobei Tags dazu verwendet werden, um Blogposts verschiedenen Themen zuzuordnen:

Post(PostID, Headline, Author)			Tag(PostID, Tag)	
PostID	Headline	Author	PostID	Tag
1	My favourite recipes	Mary Potter	1	cooking
2	Setting up a Linux VM	Jane Doe	1	recipe
3	First Steps in Python	Bob Smith	1	diy
4	Travels 2019	John Doe	3	coding
5	Introduction to SEO	Anne Johnson	3	hacking
6	Knitting a scarf	William Gold	4	usa
7	Getting ready for my first marathon	Alicia Silverstone	4	roadtrip
			4	travels
			6	wool
			6	diy

Lösung



RelaX zu dieser Datenbank: <http://dbis-uibk.github.io/relax/calc/gist/65141c587bdf3dd124e5ae27a93c85b8>

Berechnen Sie das Ergebnis folgender Abfragen:

- a) $\sigma_{\text{Headline}=\text{"Travels 2019"}} \text{Post}$

Lösung



ID	Headline	Author
4	Travels 2019	John Doe

RelaX-Abfrage:

$\sigma_{\text{Headline}=\text{'Travels 2019'}} (\text{Post})$

b) $\sigma_{Post.PostID < 4}(Post \bowtie_{Post.PostID=Tag.PostID} Tag)$

Lösung



Post.PostID	Headline	Author	Tag.PostID	Tag
1	My favourite recipes	Mary Potter	1	cooking
1	My favourite recipes	Mary Potter	1	recipe
1	My favourite recipes	Mary Potter	1	diy
2	Setting up a Linux VM	Jane Doe	null	null
3	First Steps in Python	Bob Smith	3	coding
3	First Steps in Python	Bob Smith	3	hacking

RelaX-Abfrage:

`sigma Post.PostID < 4 (Post left outer join Post.PostID=Tag.PostID Tag)`

- b) ☐ ★ Für die Ausführung der Operationen \cup , $-$, \cap müssen die Schemata der beiden Relationen ident sein - wieso gilt dies nicht für z.B. Joins?

Lösung



Für die Mengenoperationen *Vereinigung* \cup , *Differenz* $-$ und *Durchschnitt* \cap müssen die Schemata beider teilnehmenden Relationen miteinander kompatibel sein (Typkompatibel, Vereinigungskompatibel). Dies bedeutet, dass beide Schemata (1) gleiche viele Attribute haben müssen und (2), die Typen dieser Attribute müssen paarweise identisch sein. Als Beispiel die Relationen $R(a_1, a_2, \dots, a_n)$ und $S(b_1, b_2, \dots, b_m)$ sind Typkompatibel, wenn gilt $n = m$ und der Typ bzw. die Domäne jeweils paarweise gleich sind. D.h. $dom(a_1) = dom(b_1), dom(a_2) = dom(b_2)$, usw. Gegebenenfalls kann mittels Projektion nachgeholfen werden. Bei Joins hingegen wird der Join nur auf die angegebenen Spalten bzw. auf die Spalten mit gleichem Namen berechnet. Die restlichen Attribute im Schema werden bei der Join-Operation für die Berechnung nicht betrachtet.

- c) ☐ ★ Was ist der Unterschied zwischen einem NATURAL JOIN und einem EQUIJOIN?

Lösung



Der EQUIJOIN enthält nur Join-Bedingungen mit Gleichheitszeichen. Ein NATURAL JOIN ist ein EQUIJOIN der die Join-Bedingung aus den gegebenen Relationen automatisch aufstellt (gleiche Spaltennamen = gleiche Werte). Beim NATURAL JOIN ist jedes Join-Attribut in der resultierenden Relation nur einmal enthalten.

- d) ☐ ★ Berechnen und vergleichen Sie das Ergebnis folgender Abfragen für die gegebenen Relationen R und S .

- $R \bowtie S$
- $R \bowtie_{R.id=S.id} S$

R(id,name,length)		
id	name	length
0	Nationalfeiertag	16
1	Allerheiligen	13
2	Mariä Empfängnis	18
3	Heiliger Abend	14
4	Weihnachten	11
5	Stefanitag	10
6	Silvester	9

S(id, from_date, to_date, length)			
id	from_date	to_date	length
1	2019-11-01	2019-11-03	3
4	2019-12-23	2020-01-06	15

Lösung



ReLaX zu dieser Datenbank: <https://dbis-uibk.github.io/relax/calc/gist/e0b27dcd0fa7e9657529be136fd09aff>

- Das Ergebnis des natural join $R \bowtie S$ ist leer ($R \text{ join } S$)
- Das Ergebnis des equijoin ($R \text{ join } R.id=S.id \text{ } S$) ist:

R.id	name	R.length	S.id	from_date	to_date	S.length
1	Allerheiligen	13	1	2019-11-01	2019-11-03	3
4	Weihnachten	11	4	2019-12-23	2020-01-06	15

e) Gegeben sei die folgende Relation Drivers¹:

driverID	firstname	lastname	team	points
44	Lewis	Hamilton	Mercedes	338
33	Max	Verstappen	Red Bull Racing	212
77	Valtteri	Bottas	Mercedes	274
5	Sebastian	Vettel	Ferrari	212
27	Nico	Hülkenberg	Renault	35
16	Charles	Leclerc	Ferrari	221
3	Daniel	Ricciardo	Renault	42
23	Alexander	Albon	Red Bull Racing	64

Sie können die Ergebnisse der folgenden Abfragen händisch oder mit dem ReLaX-Tool berechnen:

- Berechnen Sie das Ergebnis von $\gamma_{\text{team}; \text{MAX}(\text{points}) \rightarrow \text{top}, \text{COUNT}(\text{driverID}) \rightarrow \text{numDrivers}}(\text{Drivers})$.
- Was wird durch Abfrage $\gamma_{\text{AVG}(\text{points}) \rightarrow \text{avg}}(\text{Drivers})$ berechnet?

Lösung



- Resultat Aggregation der Tabelle Drivers:

team	top	numDrivers
Mercedes	338	2
RedBullRacing	212	2
Ferrari	221	2
Renault	42	2

¹Drivers <https://dbis-uibk.github.io/relax/calc/gist/bb4cec3e91daa1d18539ae56e9855774>

RelaX-Abfrage:

```
gamma team; MAX(points) → top, COUNT(driverID) → numDrivers (Drivers)
```

- Es wird über die gesamte Tabelle der Durchschnitt der points-Spalte berechnet. Resultat:

```
avg  
174.75
```

RelaX-Abfrage:

```
gamma AVG(points) → avg (Drivers)
```

Hausaufgabenteil (Zuhause zu lösen; Abgabe nötig)

Aufgabe 1 (Relationale Algebra 1)

[6 Punkte]

Gegeben sei das folgende Relationenschema:

```
Genre (GenreId, Name)  
Artist (ArtistId, Name)  
Album (AlbumId, Title, ArtistId)  
Track (TrackId, Name, AlbumId, GenreId, Miliseconds, Bytes, UnitPrice)  
Customer (CustomerId, FirstName, LastName, Address, Email)  
Invoice (InvoiceId, CustomerId, InvoiceDate, Total)  
InvoiceParts (InvoicePartId, InvoiceId, TrackId, UnitPrice, Quantity)  
Playlist (PlaylistId, Name)  
PlaylistContent (PlaylistId, TrackId)
```

Erstellen Sie auf Basis dieses Relationenschemas die folgenden Abfragen in relationaler Algebra. Sie können dazu RelaX (einen Rechner für relationale Algebra) verwenden. Überlegen Sie sich jedoch trotzdem, wie man die Operationen "händisch" berechnen würde. Mit folgendem Link ist RelaX inklusive des für diese Aufgabe benötigten Schemas und den enthaltenen Daten erreichbar: <https://dbis-uibk.github.io/relax/calculator/gist/e8628d74e467b945a564d27d4d74b83e>.

Geben Sie für die folgenden Aufgaben **sowohl die Abfrage als auch das Ergebnis und die Anzahl an Tupel** ab. Bei sehr großen Ergebnismengen geben Sie bitte anstatt des gesamten Ergebnisses nur die **ersten zehn Zeilen** an.

Verwenden Sie für die Operatoren die ausgeschriebene Form, nicht die Symbole (π statt π , join statt \bowtie , ...).

Hinweis

Sie können im ReläX-Tool Zwischenergebnisse einer Variable zuweisen und später auf diese zugreifen:

```
Result1 = Artist join (Artist.ArtistId = Album.AlbumId) Album
```

```
pi Name, Title Result1
```

- a) **0.5 Punkte** Geben Sie bitte die Id, die CustomerId, das InvoiceDate und die Gesamtsumme (Total) aller Rechnungen aus, die eine Gesamtsumme von kleiner als 5 Euro aufweisen.

Abgabe

1a_query.txt

1a_result.txt

Lösung

```
pi InvoiceId, CustomerId, InvoiceDate, Total (sigma Total < 5 (Invoice))
```

InvoiceId	CustomerId	InvoiceDate	Total
69	25	'2009-10-25'	0.99
70	26	'2009-11-07'	1.98
71	28	'2009-11-07'	1.98
72	30	'2009-11-08'	3.96
90	21	'2010-01-26'	0.99
91	22	'2010-02-08'	1.98
92	24	'2010-02-08'	1.98
93	26	'2010-02-09'	3.96
198	6	'2011-05-20'	3.96

9 Tupel gesamt

Beachten Sie bitte, dass bei den Musterlösungen die Spaltennamen der angeführten Lösungstabellen aus Gründen der Übersichtlichkeit abgekürzt sind (so wird z.B. aus Invoice.InvoiceId lediglich InvoiceId).

- b) **0.5 Punkte** Geben Sie bitte alle Rechnungen aus, die mit November 2009 datiert sind. Dabei sollten die Id und das Datum der Rechnung, die Gesamtsumme und den Nachnamen der Kunden ausgegeben werden.

Hinweis

Datumseinträge werden im Format YYYY-MM-DD gespeichert und angegeben und können mit <, >, etc. verglichen werden. (z.B. birthday > '1934-01-01')

Abgabe

1b_query.txt

1b_result.txt

Lösung

```
pi InvoiceId, InvoiceDate, Total, LastName
((sigma InvoiceDate > '2009-10-31' and InvoiceDate < '2009-12-01' (Invoice))
join Invoice.CustomerId = Customer.CustomerId Customer)
```

InvoiceId	InvoiceDate	Total	LastName
70	2009-11-07	1.98	Cunningham
71	2009-11-07	1.98	Barnett
72	2009-11-08	3.96	Francis
74	2009-11-12	8.91	Lefebvre

4 Tupel gesamt

- c) 0.5 Punkte Finden Sie alle Tracks des Genres Rock, die auch tatsächlich gekauft wurden. Geben Sie dazu den Namen des Tracks und dessen Id aus.

Abgabe

1c_query.txt

1c_result.txt

Lösung

```
pi Track.Name, Track.TrackId
((Track join Track.TrackId = InvoiceParts.TrackId InvoiceParts)
join Track.GenreId = Genre.GenreId (sigma Genre.Name = 'Rock' (Genre)))
```

Track.Name	Track.TrackId
In Your Honor	989
No Way Back	990
The Last Song	994
Free Me	995
Still	999
What If I Do?	1000
Friend Of A Friend	1003
...	

64 Tupel gesamt

- d) 0.5 Punkte Finden Sie für die vorhandenen Playlists die enthaltenen Tracks. Geben Sie dazu bitte den Namen des Tracks und auch den Namen der Playlist aus.

Abgabe

1d_query.txt

1d_result.txt

Lösung



```
pi Playlist.Name, Track.Name
(Playlist
  join Playlist.PlaylistId = PlaylistContent.PlaylistId PlaylistContent
  join Track.TrackId = PlaylistContent.TrackId Track
)


Playlist.Name  Track.Name
Music          In Your Honor
Music          No Way Back
Music          Best Of You
Music          DOA
Music          Hell
Music          The Last Song
Music          Free Me
Music          Resolve
Music          The Deepest Blues Are Black
...


372 Tupel gesamt
```

- e) 1 Punkt Geben Sie für alle Kunden, deren Nachname mit 'A' oder 'B' beginnt, die von ihnen gekauften Lieder (Track Name, Artist Name, Album Name) aus und führen Sie auch den Vor- und Nachnamen des Kunden an.

Abgabe



 1e_query.txt

 1e_result.txt

Lösung



```
pi LastName, FirstName, Track.Name -> Trackname,
Artist.Name -> Artistname, Album.Title -> Albumtitle
(sigma LastName < 'C' Customer
join Customer.CustomerId = Invoice.CustomerId Invoice
join Invoice.InvoiceId = InvoiceParts.InvoiceId InvoiceParts
join InvoiceParts.TrackId = Track.TrackId Track
join Track.AlbumId = Album.AlbumId Album
join Album.ArtistId = Artist.ArtistId Artist)

LastName  FirstName  Trackname      Artistname      Albumtitle
Barnett   Julia          So Central Rain  R.E.M.          The Best Of R.E.M.: The IRS Years
Barnett   Julia          Pretty Persuasion R.E.M.          The Best Of R.E.M.: The IRS Years
Barnett   Julia          Gimmie Shelters Rolling Stones   No Security
Barnett   Julia          Thief In The Night Rolling Stones   No Security
Barnett   Julia          Out Of Tears    Rolling Stones   Voodoo Lounge
Barnett   Julia          In Your Honor   Foo Fighters     In Your Honor [Disc1]
```

Barnett	Julia	Free Me	Foo Fighters	In Your Honor [Disc1]
Brown	Robert	Californication	Red Hot Chili Peppers	Californication
Brown	Robert	Road Trippin'	Red Hot Chili Peppers	Californication
Bernard	Camille	Don't Go Back	R.E.M.	The Best Of R.E.M.: The IRS Years
Bernard	Camille	I Believe	R.E.M.	The Best Of R.E.M.: The IRS Years

11 Tupel gesamt

- f) **1 Punkt** Geben Sie die ID, den Titel und den UnitPrice aller Lieder aus, die nicht in Playlist 5 enthalten sind. Finden Sie hierfür zwei Lösungswege: einmal indem Sie den Mengendifferenz-Operator verwenden und einmal über einen Outer Join.

Abgabe



1f_query_1.txt

1f_query_2.txt

1f_result.txt

Lösung



```
-- über Mengendifferenz
pi TrackId, Name, UnitPrice
  (Track - (Track left semi join (sigma PlaylistId = 5 PlaylistContent)))

-- alternativ über left join
pi Track.TrackId, Name, UnitPrice (
  sigma PlaylistId = null (
    Track left join (sigma PlaylistId = 5 (PlaylistContent))))
```

TrackId	Name	UnitPrice
989	In Your Honor	0.99
990	No Way Back	0.99
991	Best Of You	0.99
992	DOA	0.99
993	Hell	0.99
994	The Last Song	0.99
995	Free Me	0.99
...		

62 Tupel gesamt

Beachten Sie bitte, dass bei den Musterlösungen die Spaltennamen der angeführten Lösungstabellen aus Gründen der Übersichtlichkeit abgekürzt sind (so wird z.B. aus Track.TrackId lediglich TrackId).

- g) **1 Punkt** Geben Sie bitte die ID und den Namen aller Künstler aus,
- deren durchschnittliche Songlänge über 4 Minuten und 10 Sekunden liegt *und*
 - deren durchschnittliche Dateigröße der Songs unter 8,5 MB liegt.

Abgabe

1g_query.txt
 1g_result.txt

Lösung

```
pi ArtistId, Name (  
  sigma avg_duration > 250000  
(gamma Artist.ArtistId, Artist.Name; avg(Milliseconds) -> avg_duration  
  (Track  
    natural join Album  
    join Album.ArtistId = Artist.ArtistId Artist )))  
  
intersect  
  
pi ArtistId, Name (  
  sigma avg_size < 8500000  
(gamma Artist.ArtistId, Artist.Name; avg(Bytes) -> avg_size  
  (Track  
    natural join Album  
    join Album.ArtistId = Artist.ArtistId Artist )))  
  
ArtistId  Name  
84        Foo Fighters  
  
1 Tupel gesamt
```

- h) 1 Punkt Finden Sie jene Käufer, die nach dem 1. Januar 2010 mindestens drei Lieder eines Albums gekauft haben und geben Sie den Nachnamen und die ID des Käufers, die ID und den Namen des Albums und die Anzahl der gekauften Tracks aus. Benennen Sie die Spalte Title des Albums in Name um.

Abgabe

1h_query.txt
 1h_result.txt

Lösung

```
rho Name <- Title (  
  sigma albumCnt > 2  
  (gamma CustomerId, LastName, AlbumId, Title;
```

```

count(AlbumId) -> albumCnt
(Customer
  natural join (
    sigma InvoiceDate > '2010-01-01' Invoice
  )
  natural join InvoiceParts
  natural join Track
  natural join Album)))

```

CustomerId	LastName	AlbumId	Name	albumCnt
10	Martins	238	The Best Of 1980-1990	3
14	Philips	190	The Best Of R.E.M.: The IRS Years	4
20	Miller	194	By The Way	3
26	Cunningham	239	War	3

4 Tupel gesamt

Aufgabe 2 (Query-Abarbeitung und Optimierung)

[4 Punkte]

Betrachten wir wieder das relationale Modell aus Aufgabe 1:

Customer (CustomerId, FirstName, LastName, Address, Email)
 InvoiceParts (InvoicePartId, InvoiceId, TrackId, UnitPrice, Quantity)
 Invoice (InvoiceId, CustomerId, InvoiceDate)
 Genre (GenreId, Name)
 Playlist (PlaylistId, Name)
 PlaylistContent (PlaylistId, TrackId)
 Artist (ArtistId, Name)
 Album (AlbumId, Title, ArtistId)
 Track (TrackId, Name, AlbumId, GenreId, Miliseconds, Bytes, UnitPrice)

Das Ziel dieser Aufgabe ist es, eine Abfrage auf drei verschiedene Weisen zu implementieren und deren Abarbeitung zu analysieren. Da die naive Implementierung dieser Abfrage sehr Ressourcenintensiv ist, können Sie ihre Abfragen auf einer leeren Datenbank testen. Die leere Datenbank ist unter folgendem Link verfügbar:

<https://dbis-uibk.github.io/relax/calc/gist/1250111065ab1e60e019928fd51dd72b>.

Die mit Daten befüllte Datenbank ist unter folgendem Link verfügbar:

<https://dbis-uibk.github.io/relax/calc/gist/e8628d74e467b945a564d27d4d74b83e>

Für die folgenden Aufgaben soll folgende Anfrage in relationaler Algebra beantwortet werden:

Finden Sie Vorname, Nachname, Songname und Kaufdatum aller Kunden, die nach dem 01.01.2010 einen Rock-Song gekauft haben.

- a) 0.5 Punkte Damit man die Effizienz einer Abfrage abschätzen kann, muss man die Größe der enthaltenen Relationen kennen. Finden Sie mittels Relax-Abfragen heraus, wie viele Tupel sich in den folgenden Relationen in der mit Daten befüllten Datenbank befinden und ergänzen Sie die fehlenden Werte. Geben Sie die ausgefüllte Tabelle als Textdatei ab.

Relation	Anzahl Tupel
Artist	
Album	11
Track	
Rock Tracks	
Playlist	
PlaylistContent	
Genre	9
Invoice	32
Invoice after 2010-01-01	
InvoiceParts	
Customer	

Abgabe



2a.txt

Lösung



Die Anzahl der Tupel einer Relation können abgefragt werden, indem man die γ -Operation ohne Gruppierung ausführt wie im folgenden zu sehen. A ist eine beliebiges Attribut der Relation R.

```
gamma ;count(A)-> c (R)
```

Relation	Anzahl Tupel
Artist	5
Album	11
Track	147
Rock Tracks	114
Playlist	3
PlaylistContent	379
Genre	9
Invoice	32
Invoice after 2010-01-01	25
InvoiceParts	99
Customer	27

- b) 1 Punkt Formulieren Sie die Abfrage, indem Sie nur **eine** Projektion, **eine** Selektion und beliebig viele Kreuzprodukte (und ggf. Umbenennungen falls nötig/gewünscht) verwenden. Ihre Abfrage soll also dem folgenden Schema entsprechen:

$$\pi_{...}(\sigma_{...}(A \times \dots \times Z)).$$

Führen Sie Ihre Abfrage auf der leeren Datenbank aus und zeichnen Sie den Operatorbaum (bzw. verwenden Sie den von RelaX zur Verfügung gestellten) auf. Berechnen Sie außerdem für jeden Knoten im Operatorbaum, wieviele Tupel die Abfrage zu den entsprechenden Zeitpunkten enthält.

Abgabe



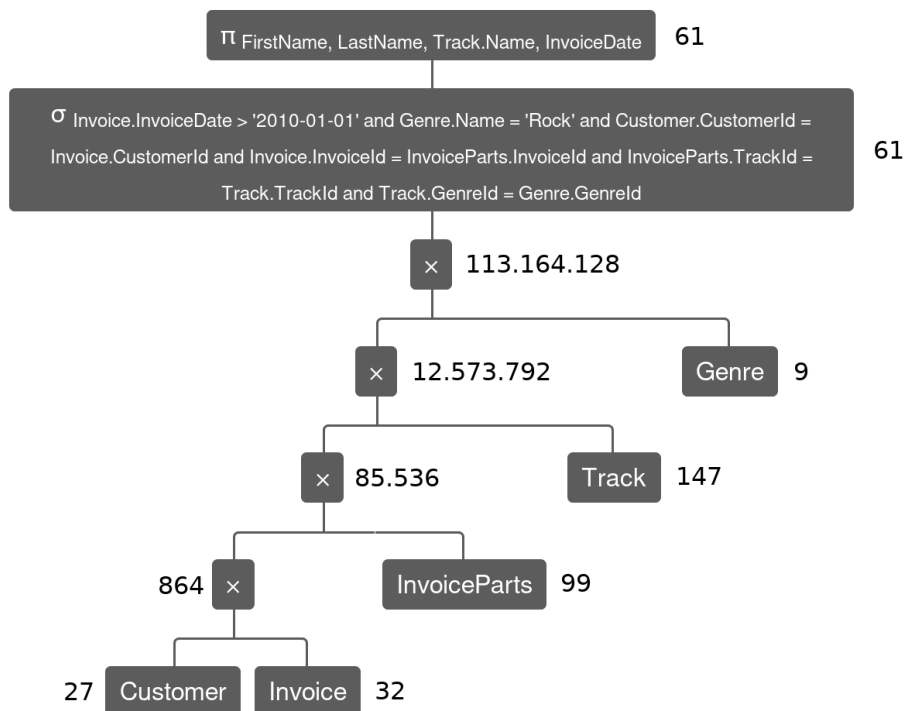
2b_query.txt

2b_tree.pdf

Lösung



```
pi FirstName, LastName, Track.Name, InvoiceDate
(sigma Invoice.InvoiceDate > '2010-01-01' and
Genre.Name = 'Rock' and
Customer.CustomerId = Invoice.CustomerId and
Invoice.InvoiceId = InvoiceParts.InvoiceId and
InvoiceParts.TrackId = Track.TrackId and
Track.GenreId = Genre.GenreId
(Customer x Invoice x InvoiceParts x Track x Genre))
```



- c) **1 Punkt** Formulieren Sie die Abfrage nun so um, dass die Selektionen, die den jeweiligen Joins entsprechen, auch direkt nach dem Kreuzprodukt ausgeführt werden. Verwenden Sie also hier noch keine expliziten Joins, sondern simulieren Sie diese nach der Regel

$$A \bowtie_c B = \sigma_c(A \times B)$$

Abgabe



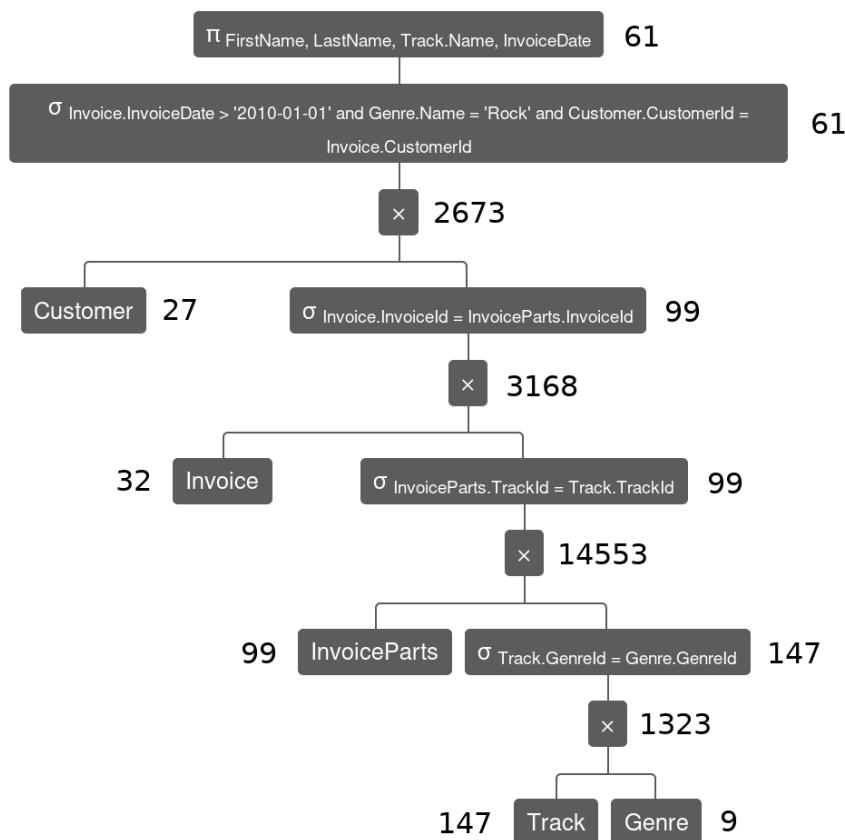
2c_query.txt

2c_tree.pdf

```

pi FirstName, LastName, Track.Name, InvoiceDate
(sigma Invoice.InvoiceDate > '2010-01-01' and
Genre.Name = 'Rock' and
Customer.CustomerId = Invoice.CustomerId
(Customer x
(sigma Invoice.InvoiceId = InvoiceParts.InvoiceId
(Invoice x
(sigma InvoiceParts.TrackId = Track.TrackId
(InvoiceParts x
(sigma Track.GenreId = Genre.GenreId
(Track x Genre))))))))))

```



d) **1.5 Punkte** Wie Sie aus der vorigen Aufgabe wahrscheinlich gesehen haben, lohnt es sich so früh wie möglich die Ergebnismenge durch Selektion zu verringern. Optimieren Sie Ihre Abfrage nun weiter, indem Sie

- 1) anstatt Kreuzprodukt und Selektion einen Join-Operator verwenden
- 2) die Selektionen bzgl. Datum und Genre nach unten schieben
- 3) die Join-Reihenfolge optimieren

Geben Sie für jeden dieser Punkte an, warum diese die Abfrage optimieren, und zeichnen Sie schlussendlich noch einmal einen Operatorbaum mit den optimierten Tupelzahlen pro Knoten.

Abgabe



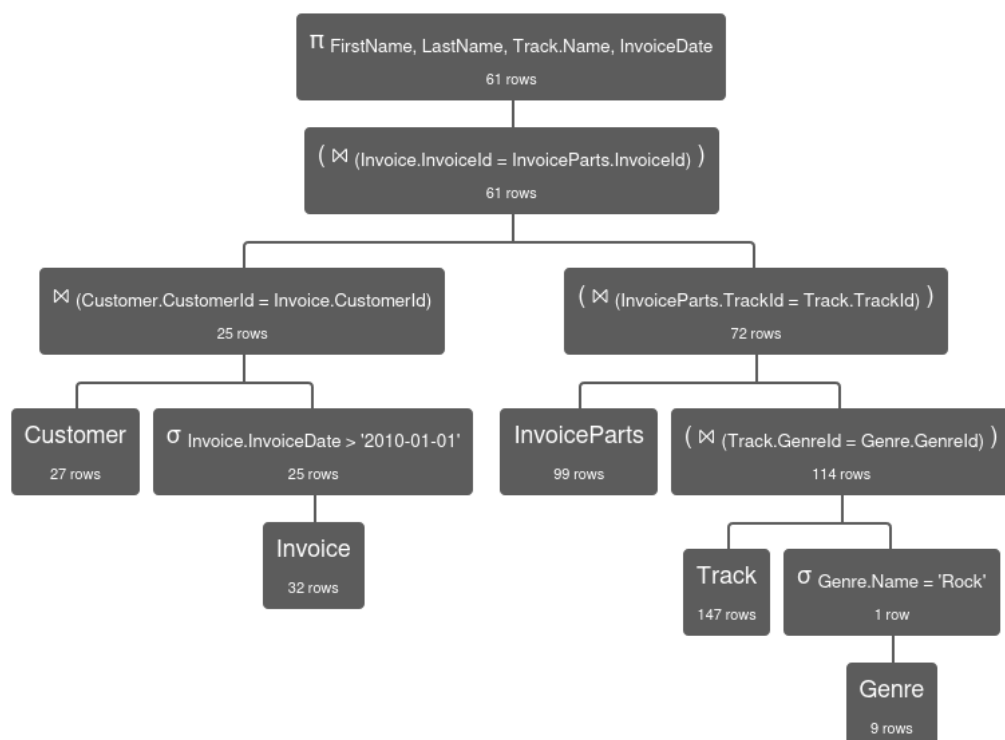
2d_query.txt
2d_tree.pdf
2d_explanation.txt

Wenn Sie diese Aufgabe gelöst haben, probieren Sie die naive Abfrage aus 2b) und die optimierte Abfrage auf der mit Daten befüllten Datenbank auszuführen. Merken Sie den Unterschied?

Lösung



```
pi FirstName, LastName, Track.Name, InvoiceDate
(Customer join (Customer.CustomerId = Invoice.CustomerId)
(sigma Invoice.InvoiceDate > '2010-01-01' Invoice)
join (Invoice.InvoiceId = InvoiceParts.InvoiceId)
(InvoiceParts join (InvoiceParts.TrackId = Track.TrackId)
(Track join (Track.GenreId = Genre.GenreId) (
sigma Genre.Name = 'Rock' Genre))))
```



Warum stellen die Änderungen Optimierungen dar?

- 1) Echter Join-Operator: diese können vorhandene Indexstrukturen verwenden bzw. ggf. sogar 'spontan' für die jeweilige Operation temporäre Indizes erstellen, wenn dies rentabel ist.

- 2) Selektionen nach unten drücken reduziert die Anzahl der Tupel in den Zwischenergebnissen, direkt vor Ort.
- 3) Die Join-Reihenfolge ist wichtig, aber komplex zu bestimmen. Ziel ist hier auch, die Zwischenergebnisse klein zu halten, jedoch kann das DBS oft nur (über Heuristiken) abschätzen, wie groß die Daten nach dem Join sein werden (wieviele Tupel entsprechen der Joinbedingung?).

Wichtig: Laden Sie bitte Ihre Lösung in OLAT hoch und geben Sie mittels der Ankreuzliste auch unbedingt an, welche Aufgaben Sie gelöst haben. Die Deadline dafür läuft am Vortag des Proseminars um 23:59 (Mitternacht) ab.