

07.11.2019

## Übungsblatt 5 – Lösungsvorschlag

### Diskussionsteil (im PS zu lösen; keine Abgabe nötig)

- a) ☐ ★ 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.

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

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

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

#### Lösung



- Das Ergebnis des natural join  $R \bowtie S$  ist leer
- Das Ergebnis des equijoin ist:

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

- c) ☐ ★ Die Abfragen  $\sigma_{R.id=S.id}(R \times S)$  und  $R \bowtie_{R.id=S.id} S$  führen beide zum selben Ergebnis. Diskutieren und begründen Sie, welche die bessere Variante ist oder ob es überhaupt einen Unterschied macht.

#### Lösung



Die Abfrage  $R \bowtie_{R.id=S.id} S$  ist die bessere und effizientere Variante da hier bereits beim joinen die Bedingung  $R.id = S.id$  geprüft wird. Es werden keine unnötig große Relationen zwischengespeichert, so wie das bei  $\sigma_{R.id=S.id}(R \times S)$  der Fall ist.

- d) ☐ ★★ Prüfen ob die folgenden Aussagen gültig sind oder nicht.  $R$  ist eine Relation,  $A$  und  $B$  zwei unterschiedliche aber beliebige Bedingungen,  $C$  und  $D$  Attribute in  $R$ .

- $\sigma_A(\sigma_B(R)) = \sigma_B(\sigma_A(R))$
- $\sigma_A(\sigma_B(R)) = \sigma_{A \wedge B}(R)$
- $\pi_C(\pi_D(R)) = \pi_D(\pi_C(R))$
- $\pi_C(\pi_D(R)) = \pi_D(R)$
- $\rho_R(R) = R$
- $\gamma_{C;count(D) \rightarrow D}(R) = \gamma_{D;count(C) \rightarrow C}(R)$

#### Lösung



- $\sigma_A(\sigma_B(R)) = \sigma_B(\sigma_A(R))$  ✓
- $\sigma_A(\sigma_B(R)) = \sigma_{A \wedge B}(R)$  ✓
- $\pi_C(\pi_D(R)) = \pi_D(\pi_C(R))$  ✗
- $\pi_C(\pi_D(R)) = \pi_D(R)$  ✗
- $\rho_R(R) = R$  ✓
- $\gamma_{C;count(D) \rightarrow D}(R) = \gamma_{D;count(C) \rightarrow C}(R)$  ✗

- e) ☐ ★★ Gegeben sei eine Relation  $R$  und die Abfrage:

$$\gamma_{column1, column2; f(column3) \rightarrow column4}(R)$$

- Welche Bedeutung haben  $column1$  und  $column2$ ?
- Welche Spalten gibt diese Abfrage zurück?
- Welche Funktionen können Sie für  $f$  einsetzen?
- Welche Werte sollten  $column3$  und  $column4$  haben, welche machen keinen Sinn?
- Sollten Duplikate bei Aggregationen entfernt werden?

#### Lösung



- Die Einträge der Relation werden auf diese beiden Spalten gruppiert. Für jede Gruppe von Tupeln, in denen die Werte dieser beiden Spalten identisch sind, ist ein Tupel in der Ergebnisrelation vorhanden.

- Diese Abfrage gibt (column1, column2, column4) als Ergebnis zurück.
- Eine Aggregierungsfunktion wie z.B. MAX, MIN, AVG, COUNT, SUM.
- column3 ist eine Spalte in R, die der angegebenen Aggregierungsfunktion übergeben wird. column4 ist der neue Name der aggregierten Spalte. Um Missverständnisse zu vermeiden, sollte sie nicht schon zuvor in der Relation auftreten (es ist aber möglich).
- Aggregationen sollten im Allgemeinen auf Mengen mit Duplikaten (Multisets) angewendet werden, sonst entstehen falsche Ergebnisse (siehe Folien, Kapitel 3).

f)  Gegeben sei folgende Relation Drivers<sup>1</sup>:

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

- 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
Ferrari	221	2
Mercedes	338	2
Renault	42	2
Red Bull Racing	212	2

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

avg
174.75

g)  Betrachten Sie in RelaX nochmals die Beispieldatenbank<sup>2</sup> der Music Streaming Plattform von letzter Woche.

- Wo können Sie hier sinnvolle Aggregationen bzw. Gruppierungen berechnen? Nennen Sie zwei Beispiele.
- Berechnen Sie bitte die durchschnittliche Anzahl an gekauften Tracks pro Einkauf. Sie können dazu gerne RelaX verwenden.

<sup>1</sup>Drivers <https://dbis-uibk.github.io/relax/calc.htm?data=gist:bb4cec3e91daa1d18539ae56e9855774>

<sup>2</sup>Music <https://dbis-uibk.github.io/relax/calc.htm?data=gist:e8628d74e467b945a564d27d4d74b83e>

- Wie können Sie die Anzahl der im Rahmen des größten Einkaufs (also den Einkauf, der die meisten Tracks umfasst) gekauften Tracks bestimmen?

#### Lösung



- Folgende Berechnungen könnte man z.B. durchführen: durchschnittliche Anzahl an gekauften Tracks pro Einkauf; minimale, maximale Anzahl an Tracks in Playlists.
- `gamma ; AVG(trackCount) -> avgTracks (gamma InvoiceId; count(TrackId) -> trackCount (InvoiceParts))`

$$\frac{\text{avgTracks}}{3.09375}$$

- `gamma ; MAX(trackCount) -> maxTracks ((gamma InvoiceId; count(TrackId) -> trackCount (InvoiceParts)))`

$$\frac{\text{maxTracks}}{8}$$

- h) ★★ Die Menge relationaler Algebra-Operationen  $\Gamma = \{\sigma, \pi, \cup, -, \times\}$  ist vollständig. Das bedeutet, dass jede relationale Algebra-Operation durch Operationen aus dieser Menge ausgedrückt werden kann. Schreiben Sie folgende Operationen um indem sie nur Operationen aus  $\Gamma$  verwenden.

- $R \bowtie_{R.id=S.id} S$
- $R \cap S$
- $R \div S$

#### Lösung



- $R \bowtie_{R.id=S.id} S = \sigma(R.id = S.id)(R \times S)$
- $R \cap S$  siehe Volesungsfolien, Kapitel 3 S. 87
- $R \div S$  siehe Volesungsfolien, Kapitel 3 S. 87

## Hausaufgabenteil (Zuhause zu lösen; Abgabe nötig)

### Hinweis



Wir werden in Zukunft, falls eine Abgabe in einer bestimmten Form bzw. in einem bestimmten Format notwendig ist, angeben, in welchem Dateiformat und mit welcher Dateibezeichnung die Lösungen abgegeben werden müssen. Halten Sie sich an die Vorgaben, wenn diese nicht eingehalten werden, gibt es Punktabzüge.

Im Folgenden sehen Sie ein Beispiel, bei dem als Abgabe eine PDF-Datei `sheet5_1_solution.pdf` und einmal eine TXT-Datei `sheet5_1_description.pdf` gefordert wird.

### Abgabe



`sheet5_1_solution.pdf`

`sheet5_1_description.txt`

## Aufgabe 1 (Relationale Algebra: Aggregation)

[6 Punkte]

Gegeben sei das relationale Modell, das wir bereits beim letzten Übungszettel verwendet haben:

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)

Erstellen Sie auf Basis dieses Relationenschemas die folgenden Abfragen in relationaler Algebra. Sie können dazu wie gehabt 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 erreichbar: <https://dbis-uibk.github.io/relax/calc.htm?data=gist:e8628d74e467b945a564d27d4d74b83e>.

Geben Sie für die folgenden Aufgaben **sowohl die Abfrage als auch das Ergebnis** mit den vorgegebenen Bezeichnungen und Dateiformaten ab. Verwenden Sie die ausgeschriebene Form der griechischen Symbole ( $\sigma$  anstatt  $\sigma$ ,  $\gamma$  anstatt  $\gamma$ ,  $\bowtie$  anstatt  $\times$ , ...). Stellen Sie sicher, dass die Abfragen mit dem RelaX Tool ausgeführt werden können.

### Hinweis



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

```
Result1 = Artist join (Artist.ArtistId = Album.AlbumId) Album
pi Name, Title Result1
```

- a) 1 Punkt Geben Sie den 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



📄 sheet5\_1a\_query\_1.txt  
📄 sheet5\_1a\_query\_2.txt  
📄 sheet5\_1a\_result.txt

### Lösung



```
tracksInPlaylist5 = pi TrackId (sigma PlaylistId = 5 (PlaylistContent))
```

```
pi TrackId, Name, UnitPrice  
  (((pi TrackId (Track)) - tracksInPlaylist5)  
   natural join Track)
```

```
-- 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<sup>a</sup>

<sup>a</sup>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).

- b) 1 Punkt Geben Sie für jeden Artist die Anzahl der veröffentlichten Alben aus. Geben Sie dabei den Namen und die Anzahl der Alben absteigend aus.

### Abgabe



📄 sheet5\_1b\_query.txt  
📄 sheet5\_1b\_result.txt

### Lösung



```
tau NumAlbums desc  
(gamma Name; count(ArtistId) -> NumAlbums  
(Artist left join Album))
```

Name	NumAlbums
Red Hot Chili Peppers	3
U2	3
Foo Fighters	2
Rolling Stones	2
R.E.M.	1

5 Tupel gesamt

- c) 1 Punkt Geben Sie die Alben von U2 aus, von denen ein Track mehr als einmal verkauft wurde.

#### Abgabe



sheet5\_1c\_query.txt

sheet5\_1c\_result.txt

#### Lösung



```
pi Title (
  sigma NumBuys >= 2
  (gamma InvoiceParts.TrackId, Album.Title; count(TrackId) -> NumBuys
    (InvoiceParts
      natural join Track
      natural join Album
      join Album.ArtistId = Artist.ArtistId (
        sigma Name='U2' (Artist))))))
```

Title
War
Zooropa
The Best Of 1980-1990

3 Tupel gesamt

- d) 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



sheet5\_1d\_query.txt

sheet5\_1d\_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

- e) **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



sheet5\_1e\_query.txt

sheet5\_1e\_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



- f) **1 Punkt** Ermitteln Sie den Namen aller Songs mit TrackId < 1000 deren Dauer (Miliseconds) über der durchschnittlichen Dauer liegt.

#### Abgabe



sheet5\_1f\_query.txt  
sheet5\_1f\_result.txt

#### Lösung



```
X = gamma ; AVG(Miliseconds)->avg Track
pi Name (sigma Miliseconds > avg and TrackId < 1000 (Track natural join X))
```

```
-- Alternative Möglichkeit
pi Name (sigma (Miliseconds > avg and TrackId < 1000)
((gamma ;avg(Miliseconds) -> avg
(Track)) cross join (rho B (Track))))
```

Name
Free Me
Resolve
End Over End
Still

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, könne Sie ihre Abfragen auf einer leeren Datenbank testen. Die leere Datenbank ist unter folgendem Link verfügbar: <https://dbis-uibk.github.io/relax/calc.htm?data=gist:1250111065ab1e60e019928fd51dd72b>.

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 relationaler Algebra heraus, wie viele Tupel sich in den folgenden Relationen in der mit Daten befüllten Datenbank (<https://dbis-uibk.github.io/relax/calc.htm?data=gist:e8628d74e467b945a564d27d4d74b83e>) befinden und ergänzen Sie die fehlenden Werte.

### Abgabe



sheet5\_2a.txt

Relation	Anzahl Tupel
Artist	5
Album	
Track	
Rock Tracks	
Playlist	
PlaylistContent	
Genre	9
Invoice	
Invoice after 2010-01-01	
InvoiceParts	
Customer	27

### Lösung



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

$\gamma_{\text{count}(A) \rightarrow c}(R)$

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:

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

Zeichnen Sie den Operatorbaum (bzw. verwenden Sie den von RelaX zur Verfügung gestellten) und berechnen Sie für jeden Knoten im Operatorbaum, wieviele Tupel die Abfrage zu diesem Zeitpunkt enthält.

## Abgabe



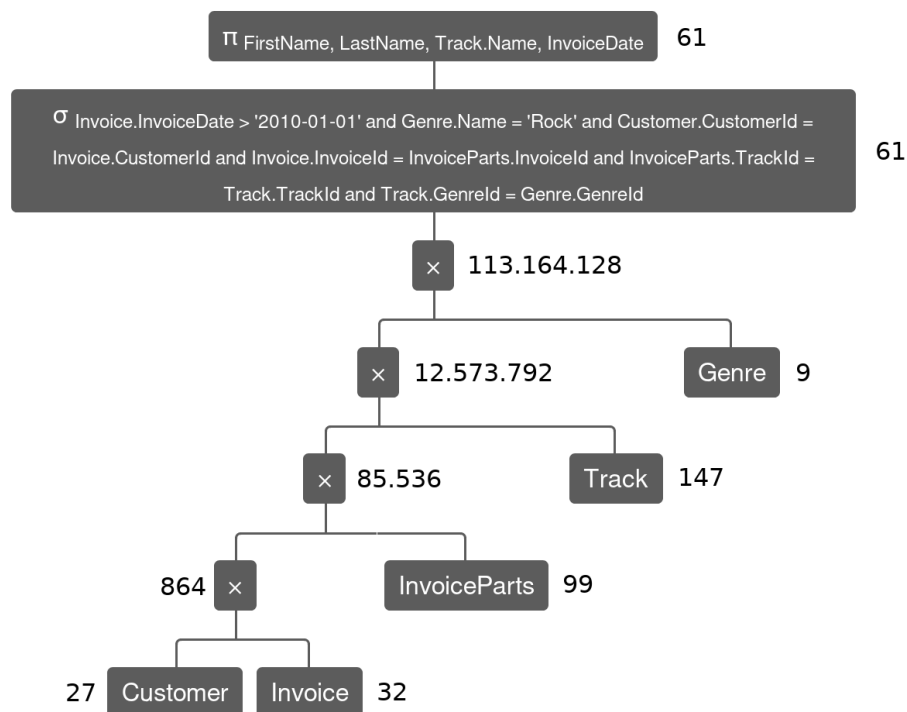
sheet5\_2b\_query.txt

sheet5\_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 keine expliziten Joins, sondern simulieren Sie diese nach der Regel

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

## Abgabe



sheet5\_2c\_query.txt

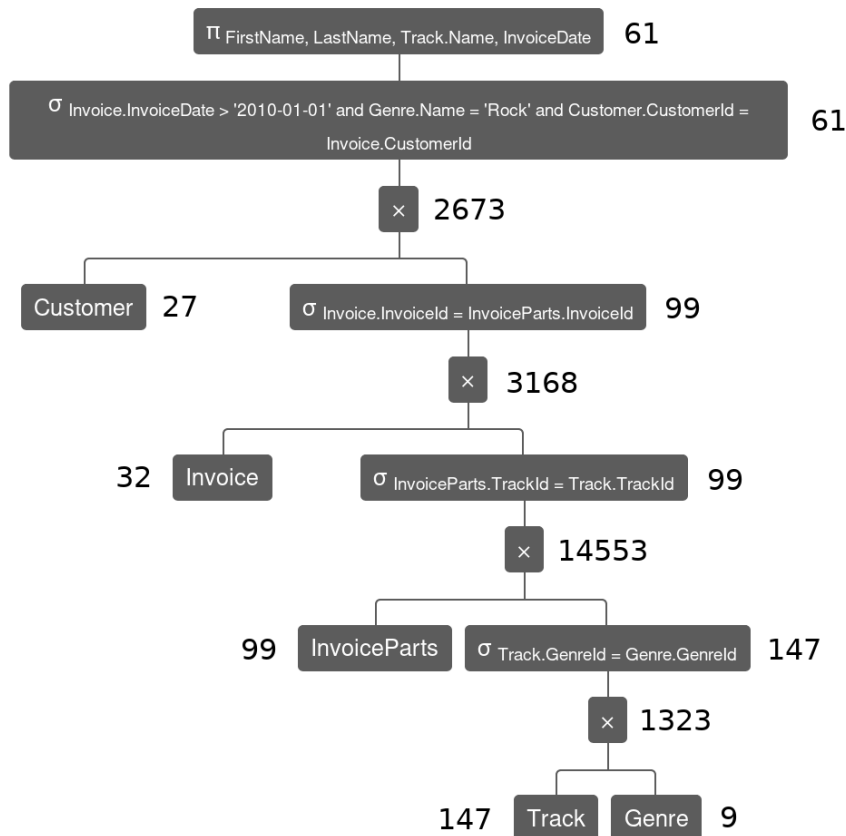
sheet5\_2c\_tree.pdf

## Lösung



```

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, Joins einzusetzen. Optimieren Sie Ihre Anfrage 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



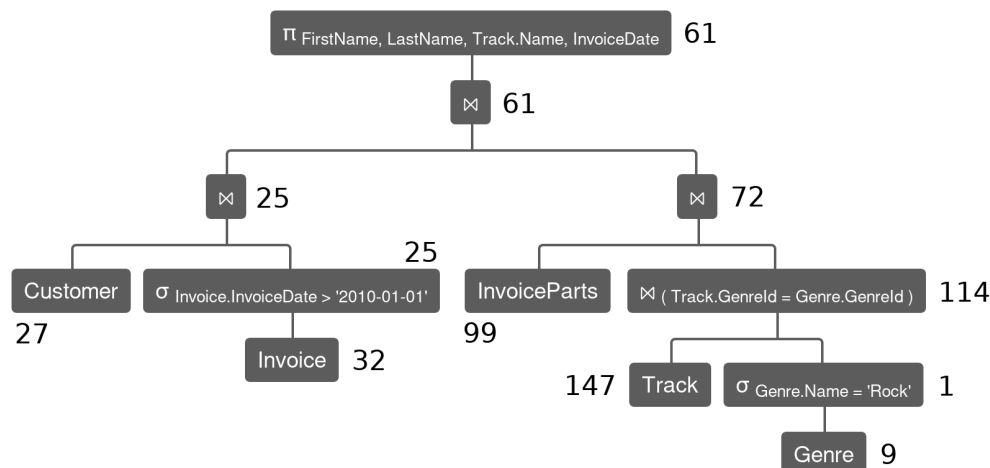
- sheet5\_2d\_query.txt
- sheet5\_2d\_tree.pdf
- sheet5\_2d\_explanation.txt

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
  (sigma Invoice.InvoiceDate > '2010-01-01' Invoice) join
  (InvoiceParts join
    (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.