Tutorium 5
Deanonymisierung, Anfrageoptimierung
Big Data Engineering

Prof. Dr. Jens Dittrich

bigdata.uni-saarland.de

7. Juni 2022

Verbesserung Übungsblätter - Häufige Fehler

Aufgabe 1:

- Annahme, dass der Lieblingsfilm auch bewertet werden muss bzw. automatisch der am besten bewertete Film sein muss.

Aufgabe 2:

- Keine Verbindung zum ursprünglichen Datensatz der Krankenkasse bzgl. der Behandlungen hergestellt.
- Informationen genutzt, die nicht vorhanden sind (z.B. Restaurantbesuche)

Aufgabe 3:

- c: Keine oder fehlende Projektionen.
- c: Kartesische Produkte (und zugehörige Selektionen) nicht durch Joins ersetzt.
- c: Projektion bei Personen über PID, Name, Geburtsjahr, Wohnort. Dies ist in diesem Fall überflüssig.

Aufgabe 4:

- Keine Überprüfung, ob die Personen auch tatsächlich zur Tatzeit in einem der Häuser gelebt haben.
- Überprüfung, ob genau so viele Lebensmittel gekauft wurden, wie das Rezept des Giftes spezifiziert hat.

Wiederholung - Frage 1

Frage

In welche Schritte kann Anfrageoptimierung unterteilt werden?

Wiederholung - Frage 1

Frage

In welche Schritte kann Anfrageoptimierung unterteilt werden?

Lösung

1. SQL
   ↓ kanonische Übersetzung
2. annotierte relationale Algebra/logischer Plan
   ↓ heuristische/regelbasierte Optimierung
3. transformierter logischer Plan
   ↓ kostenbasierte Optimierung
4. physischer Plan
   ↓ Code-Erzeugung
5. ausführbarer Code

Streng genommen ist die Code-Erzeugung **kein** Teil der Anfrageoptimierung, sondern ein eigenständiges Problem.

Wiederholung - Frage 2

Frage

Hat ein Prädikat p einer Selektion σ_p hohe Selektivität, dann ...

(A): erfüllen viele Tupel p

(B): gilt $|\text{Ausgaberelation}| < |\text{Eingaberelation}|$

(C): erfüllen wenige Tupel p

(D): gilt $|\text{Ausgaberelation}| > |\text{Eingaberelation}|$

Wiederholung - Frage 2

Frage

Hat ein Prädikat p einer Selektion σ_p hohe Selektivität, dann ...

Lösung

Die richtigen Antworten lauten (B) und (C):

Hohe Selektivität bedeutet, dass die Ausgaberelation im Vergleich zur Eingaberelation sehr klein ist. Zu beachten ist jedoch, dass wir bei **kleinen** numerischen Werten von einer **hohen** Selektivität sprechen!

(A): ist falsch, da Eingabe- und Ausgaberelation sonst ähnlich groß wären.

(B): gilt für jedes selektive Prädikat.

(C): ist richtig.

(D): gilt nie, weder bei hoher noch bei niedriger Selektivität.

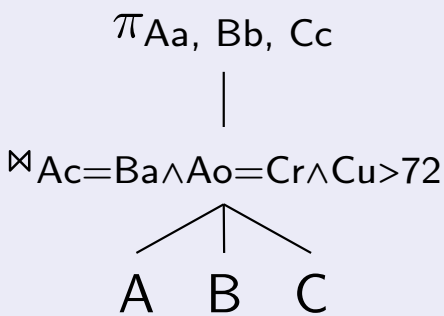
Wiederholung - Frage 3

Frage

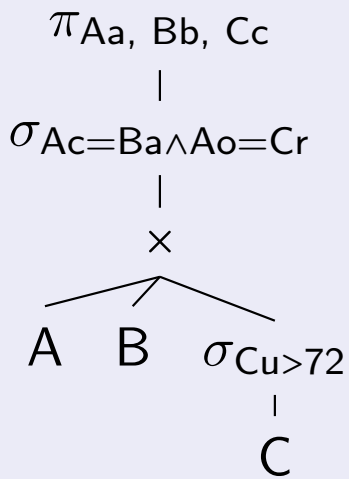
Was ist der logische Plan folgender SQL Anfrage nach der kanonischen Übersetzung?

```
SELECT Aa, Bb, Cc
FROM   A, B, C
WHERE  Ac=Ba AND Ao=Cr AND Cu > 72;
```

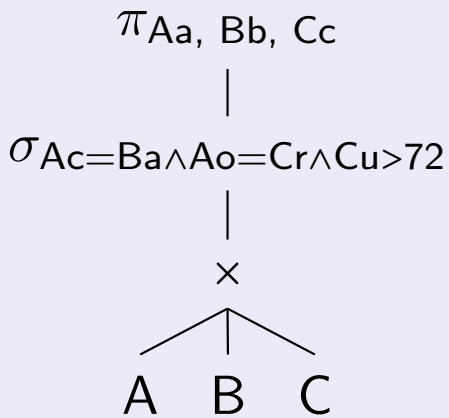
(A):



(B):



(C):



Wiederholung - Frage 3

Frage

Was ist der logische Plan folgender SQL Anfrage nach der kanonischen Übersetzung?

```
SELECT Aa, Bb, Cc
FROM   A, B, C
WHERE  Ac=Ba AND Ao=Cr AND Cu>72;
```

Lösung

Die richtige Antwort lautet (C):

In der kanonischen Übersetzung werden noch keine Optimierungen (wie z.B. Predicate Pushdown) durchgeführt.

Wiederholung - Frage 4

Frage

Wie sieht der folgende Ausdruck nach Anwendung der Predicate Pushdown Regel aus?

$\sigma_{Si=Tv \wedge Sb=30 \wedge Tx \geq 19}\ (S \bowtie_{Sv=Ti}\ T)$

(A): $\sigma_{Si=Tv}\ (S \bowtie_{Sv=Ti \wedge Sb=30 \wedge Tx \geq 19}\ T)$

(B): $\sigma_{Si=Tv}\ ((\sigma_{Sb=30}\ S) \bowtie_{Sv=Ti}\ (\sigma_{Tx \geq 19}\ T))$

(C): $S \bowtie_{Si=Tv \wedge Sv=Ti \wedge Sb=30 \wedge Tx \geq 19}\ T$

(D): $(\sigma_{Sb=30}\ S) \bowtie_{Si=Tv \wedge Sv=Ti}\ (\sigma_{Tx \geq 19}\ T)$

Wiederholung - Frage 4

Frage

Wie sieht der folgende Ausdruck nach Anwendung der Predicate Pushdown Regel aus?

$\sigma_{Si=Tv \wedge Sb=30 \wedge Tx \geq 19}\ (S \bowtie_{Sv=Ti} T)$

Lösung

Die richtige Antwort lautet (B):

$\sigma_{Si=Tv}\ ((\sigma_{Sb=30}\ S) \bowtie_{Sv=Ti} (\sigma_{Tx \geq 19}\ T))$

Wiederholung - Frage 5

Frage

Wie sieht der folgende Ausdruck nach Anwendung der Projection Pushdown Regel aus?

$\pi_{Ay,Bx}\ (A \bowtie_{Ax=Bx} B)$

(A): $\pi_{Ay,Bx}\ ((\pi_{Ax,Ay}\ A) \bowtie_{Ax=Bx} (\pi_{Bx,By}\ B))$

(B): $\pi_{Ay,Bx}\ ((\pi_{Ay}\ A) \bowtie_{Ax=Bx} (\pi_{Bx}\ B))$

(C): $(\pi_{Ax,Ay}\ A) \bowtie_{Ax=Bx} (\pi_{Bx}\ B)$

(D): $\pi_{Ay,Bx}\ ((\pi_{Ax,Ay}\ A) \bowtie_{Ax=Bx} (\pi_{Bx}\ B))$

Wiederholung - Frage 5

Frage

Wie sieht der folgende Ausdruck nach Anwendung der Projection Pushdown Regel aus?

$\pi_{Ay,Bx}\ (A \bowtie_{Ax=Bx} B)$

Lösung

Die richtige Antwort lautet (D):

$\pi_{Ay,Bx}\ ((\pi_{Ax,Ay}\ A) \bowtie_{Ax=Bx} (\pi_{Bx}\ B))$

Aufgabe 1

Frage (1/2)

Timmy (10) liebt das Spiel KlonemonGo, in welchem es darum geht, kleine Wesen zu fangen, die sich an realen Orten befinden. Dabei muss man sich an die entsprechenden Orte begeben, um das Tier zu fangen. Timmy ist nun jedoch am 19.07.2019 Opfer eines grausamen Verbrechens geworden. Er wollte gerade sein Lieblingsklonemon, Pikaschuh, um genau 14 Uhr in Saarbrücken fangen, als ihm jemand zuvor kam und es ihm vor seinen Augen weggeschnappt hat. Nicht nur das, es wurden ihm im gleichen Augenblick auch noch seine "6 Gum"-Kaugummis gestohlen, welche immerhin 3€ gekostet haben. Das will Timmy natürlich nicht auf sich sitzen lassen, weshalb er sich in die KlonemonGo-Datenbank hackt, die Informationen über die echten Namen der User*innen und gefangene Klonemon bereithält. Außerdem hat er sich auf dem örtlichen Schulhofkaugummischwarzmarkt über Kaugummiverkäufer*innen kundig gemacht.

Aufgabe 1

Frage (2/2)

Insgesamt hat Timmy also Zugriff auf folgendes Relationenschema:

[User*innen] : {[ID:int, Name:string]}

[fangen] : {[UID:(User*innen→ ID), Klonemon:(Klonemon→ Name),
Zeitpunkt: timestamp, Ort: string]}

[verkaufen] : {[Verkaeufer*in:(Personen→ Name), Kaugummi:(Kaugummis→ Name),
Preis: double]}

Timmy ist sich sicher, aufgrund dieser Daten eine*n Verdächtige*n identifizieren zu können. Leider ist Timmy erst 10 und während er bereits gelernt hat, sich in fremde Datenbanken zu hacken, hat er leider noch keine Erfahrung in SQL. Können Sie ihm helfen, den Kaugummidieb zu entlarven? Geben Sie eine SQL-Anfrage an, die die Namen von potentiellen Verdächtigen ausgibt. Gehen Sie davon, dass *timestamps* das Format `YYYY-MM-DD HH-MM-SS` haben.

Aufgabe 1

Lösung

```
SELECT U.Name
FROM   User*innen AS U
       JOIN fangen
            ON UID = U.ID
       JOIN verkaufen
            ON Verkaeufer*in = U.Name
WHERE  Klonemon = 'Pikaschuh'
       AND Kaugummi = '6 Gum'
       AND Zeitpunkt = '2019-07-19 14-00-00'
       AND Ort = 'Saarbruecken';
```

Aufgabe 2

Frage

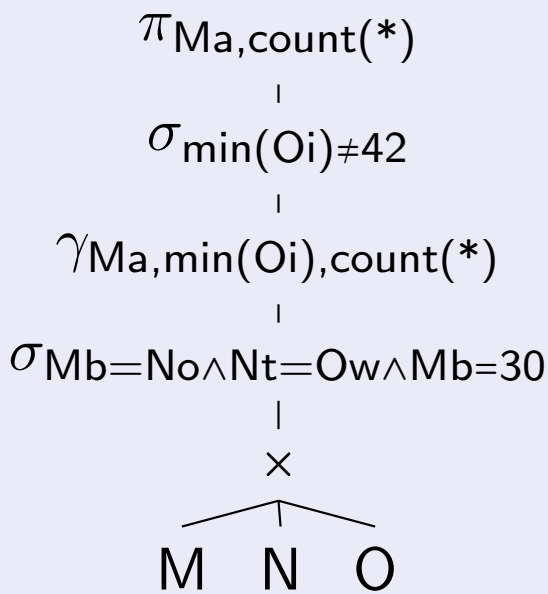
Betrachten Sie folgende SQL Anfrage.

```
SELECT    Ma, COUNT(*)
FROM      M, N, O
WHERE     Mb = No AND Nt = Ow AND Mb = 30
GROUP BY  Ma
HAVING    MIN(Oi) <> 42;
```

1. Überführen Sie die SQL Anfrage mithilfe der kanonischen Übersetzung in einen logischen Plan.
2. Brechen Sie nun im logischen Plan alle Konjunktionen in Prädikaten auf.
3. Drücken Sie nun die Prädikate innerhalb des Plans so weit nach unten wie möglich.
4. Fassen Sie nun Selektionen und kartesische Produkte zu Joins zusammen.
5. Der Plan birgt neben den in der Vorlesung eingeführten Regeln weiteres Optimierungspotential. Führen Sie die Optimierung durch und beschreiben Sie Ihre verwendete Regel.

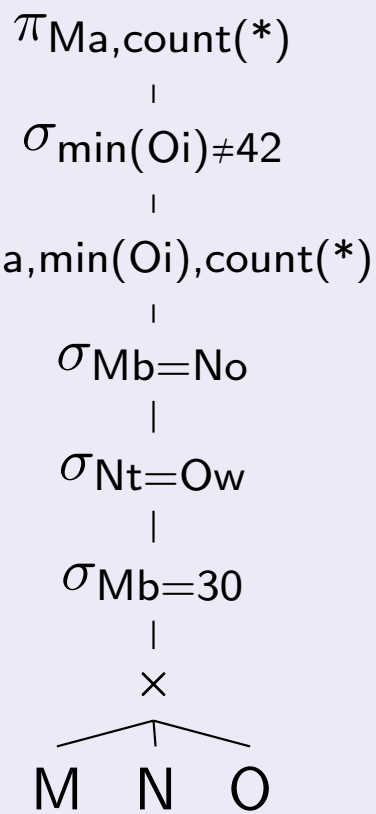
Aufgabe 2.1

Lösung



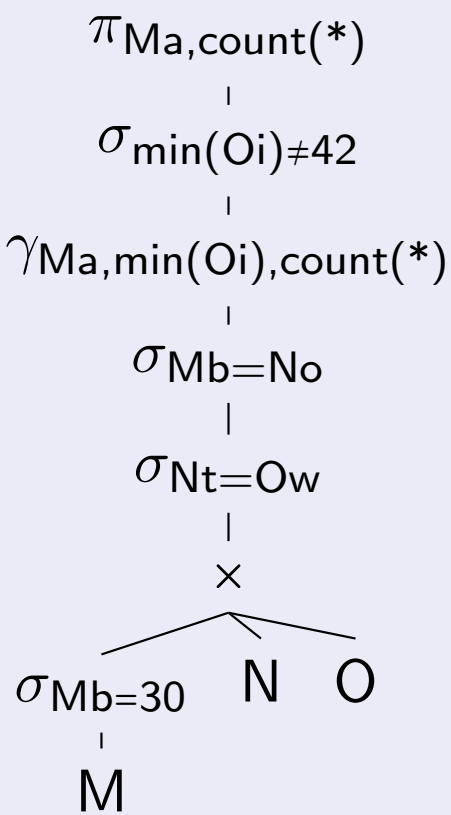
Aufgabe 2.2

Lösung



Aufgabe 2.3

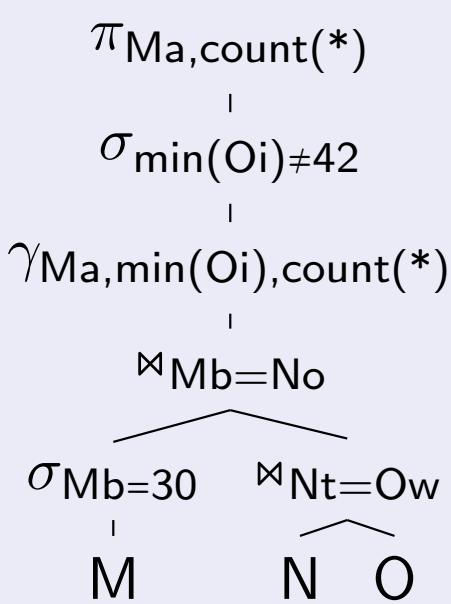
Lösung

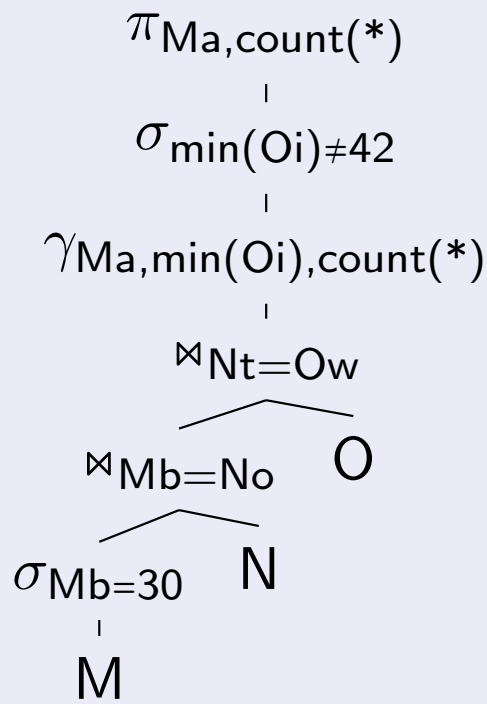


Aufgabe 2.4

Lösung

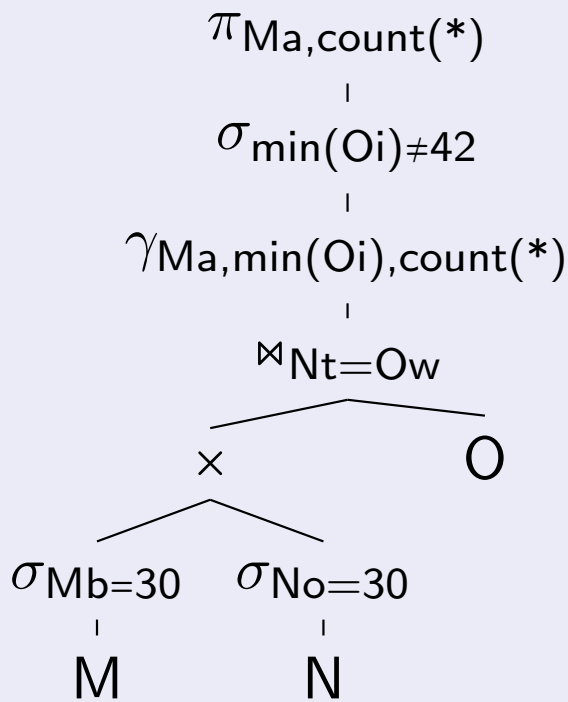
Das Festlegen der Joinreihenfolge ist Aufgabe der kostenbasierten Optimierung, folglich gibt es hier noch mehrere gültige Lösungen.





Aufgabe 2.5

Lösung

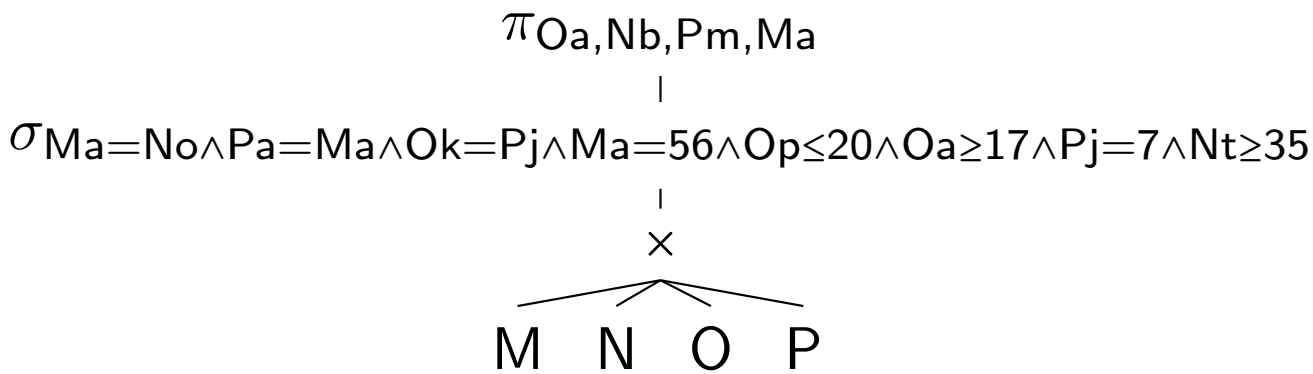


Regel: Transitivität
Seien A und B Attribute beliebiger Tabellen, c eine Konstante und op ein Vergleichsoperator. Gilt $A = B$ und $A\ op\ c$, so gilt auch $B\ op\ c$. Somit kann eine weitere Selektion eingefügt und die Datenmenge, die beim Prüfen von $A = B$ verarbeitet werden muss, verringert werden. Gilt zudem $A = c$ (und daher nach obiger Anwendung $B = c$), so kann die Bedingung $A = B$ beim Join entfernt und daher gegebenfalls durch ein Kreuzprodukt ersetzt werden.

Aufgabe 3.1

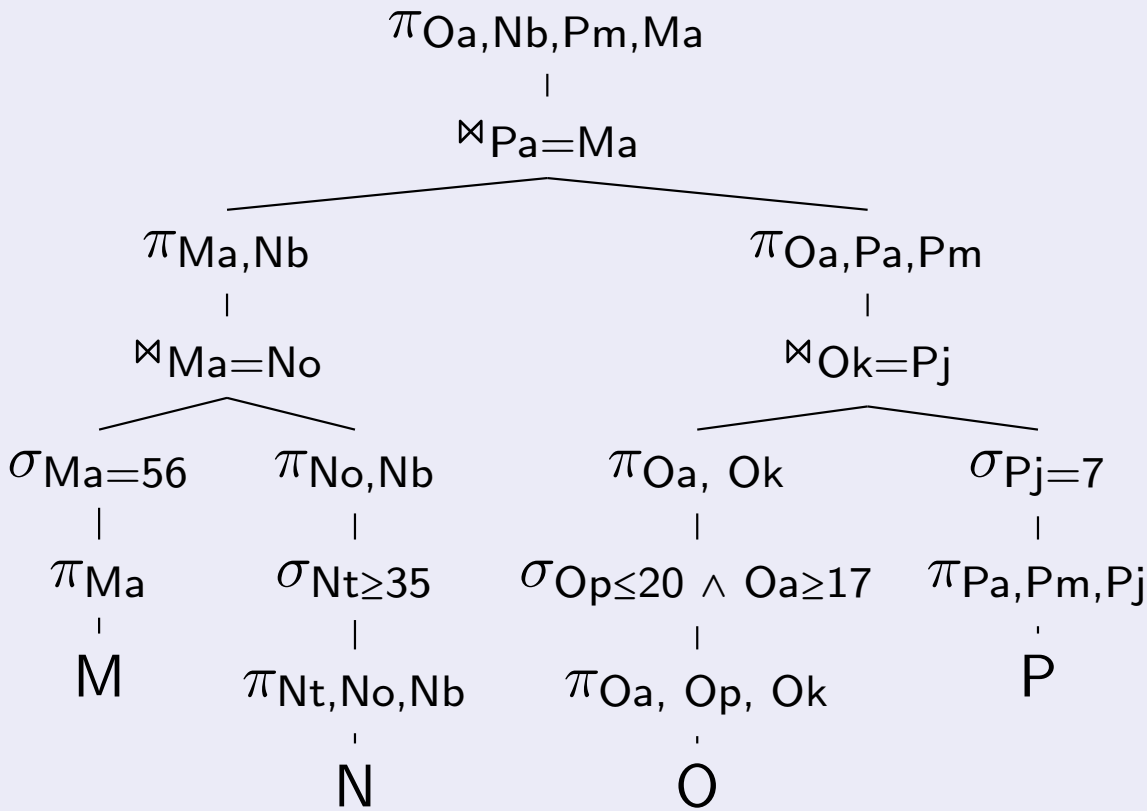
Frage

Optimieren Sie den unten stehenden logischen Plan anhand der in der Vorlesung vorgestellten Regeln so weit wie möglich. Benutzen Sie die Joinreihenfolge $(M \bowtie N) \bowtie (O \bowtie P)$.



Aufgabe 3.1

Lösung



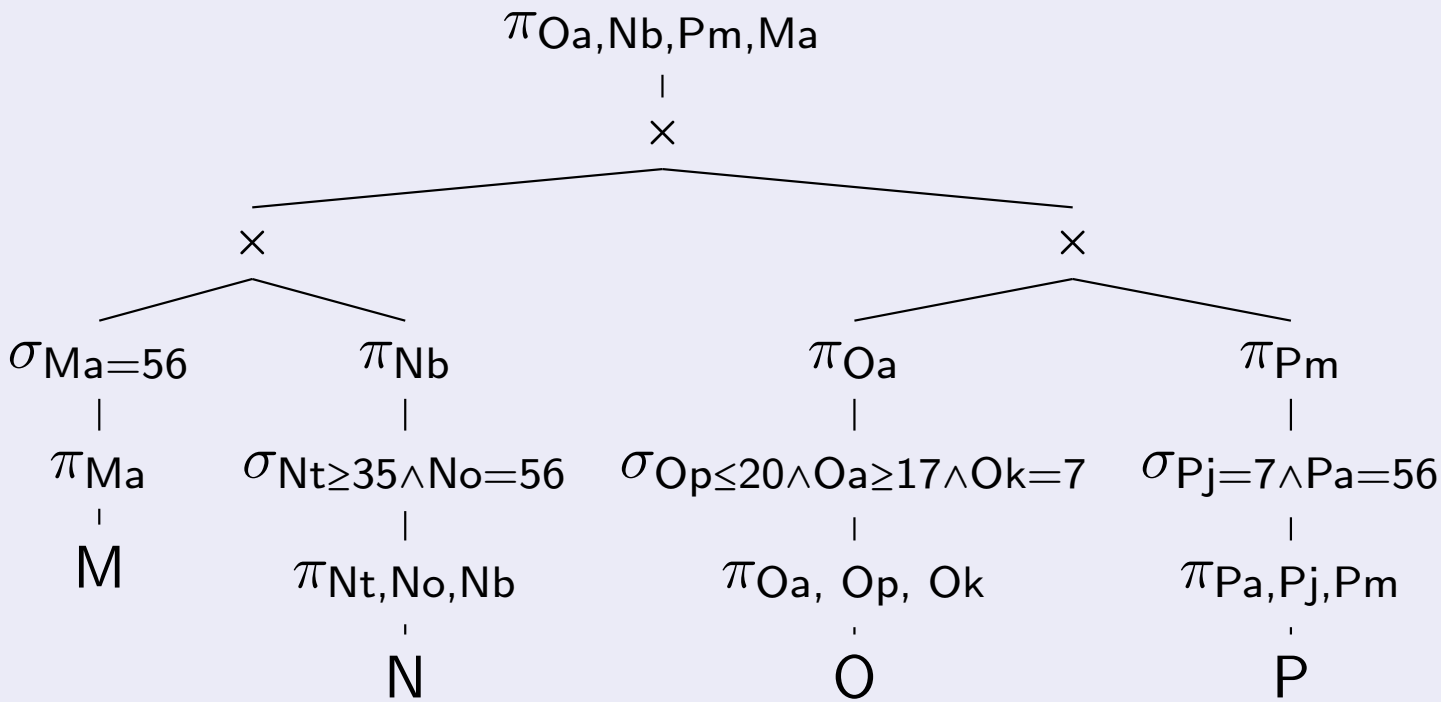
Aufgabe 3.2

Frage

Der Plan lässt sich anhand der in Aufgabe 2.5 etablierten Regel (Transitivität) noch weiter optimieren. Führen diese Optimierungen durch.

Aufgabe 3.2

Lösung



Aufgabe 3.3

Frage

Weshalb versuchen wir innerhalb eines Plans sowohl Selektionen als auch Projektionen so früh wie möglich auszuführen? Vergleichen Sie Projektion und Selektion bezüglich ihrer Gemeinsamkeiten und Unterschiede.

Aufgabe 3.3

Lösung

Wir versuchen Projektionen und Selektionen so früh wie möglich auszuführen, da beide Operationen die Datenmenge verkleinern (→ Gemeinsamkeit). Das bedeutet, dass Operationen, insbesondere Joins, die später im Plan ausgeführt werden, weniger Daten verarbeiten müssen und somit schneller ausgeführt werden können. Hierbei verringert die Selektion die Datenmenge horizontal durch Entfernen von Tupeln/Zeilen und die Projektion vertikal durch Entfernen von Attributen/Spalten (→ Unterschied).

Aufgabe 4.1

Frage

Angenommen Ihnen steht ein perfektes Kostenmodell zur Verfügung.
Wieso bleibt es schwer den optimalen Plan zu bestimmen?

Aufgabe 4.1

Lösung

Das Kostenmodell weist einem Plan lediglich seine Kosten zu. Um den günstigsten Plan zu finden, muss dieser Plan überhaupt erst einmal aufgezählt und an das Kostenmodell gegeben werden. Je größer eine Anfrage wird, desto mehr Pläne gibt es zum Berechnen dieser Anfrage. Insbesondere die Anzahl der möglichen Joinreihenfolgen wächst mit zunehmender Anzahl an Tabellen exponentiell. Daher werden in der Regel nicht alle Joinreihenfolgen aufgezählt (Join Enumeration) und so der beste Plan potentiell übersehen.

Aufgabe 4.2

Frage

Wieso gibt es unterschiedliche Kostenmodelle? Was bedeutet es, wenn wir einen Plan *optimal* nennen?

Aufgabe 4.2

Lösung

- Mit unterschiedlichen Kostenmodellen verfolgen wir unterschiedliche Ziele. Wird die Datenbank beispielsweise auf einem mobilen Gerät ausgeführt, könnte es beispielsweise wichtiger sein, dass die Verarbeitung einer Anfrage wenig Energie verbraucht. Haben wir ein großes Rechenzentrum zur Verfügung und stellen eine Anwendung in Echtzeit zur Verfügung, priorisieren wir vor allem geringe Anfragezeiten.
- Finden wir mithilfe eines geeigneten Kostenmodells für eine Anfrage den *optimalen* Plan, dann bedeutet das immer nur, dass der Plan optimal hinsichtlich des Kostenmodells ist, unter einem anderen Kostenmodell könnte der Plan suboptimal sein.