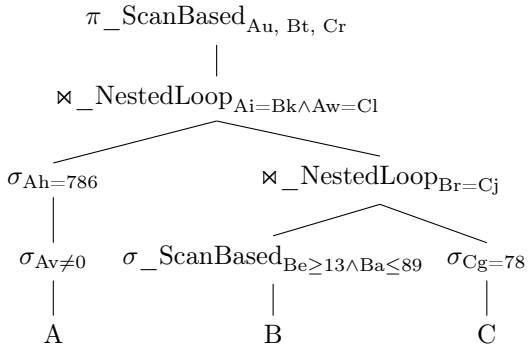
1 Kostenbasierte Optimierung (5 Punkte)

Gegeben sei der folgende "hybride" Anfragebaum, der bereits regelbasiert optimiert wurde (zusätzliche Projektionen wurden hier einfachheitshalber weggelassen). Dabei wurden bereits einige logische Operatoren durch physische ersetzt. Ihre Aufgabe ist es nun, die noch bestehenden logischen Operatoren ebenfalls in physische zu übersetzen.



Auf den Attributen Ah, Av und Cg existieren Indizes. Für ein Prädikat p mit Selektivität $sel(p)$ auf Tabelle T gelten die folgenden Kosten für einen vollständigen Scan bzw. Indexzugriff:

$$scan(T) = |\ T\ |$$
$$index(T,p) = log_2(|\ T\ |) + 37 \cdot sel(p) \cdot |\ T\ |$$

(a) Stellen Sie für die folgenden Tabellengrößen und Selektivitäten den kostengünstigsten physischen Plan auf und zeichnen Sie dessen Anfragebaum (mit physischen Operatoren).

1. $|\ \text{A}\ | = 70.000 \quad sel(\text{Ah=786}) = 0,2 \quad sel(\text{Av}\neq 0) = 0,05$
   $|\ \text{C}\ | = 230.000 \quad sel(\text{Cg=78}) = 0,02$
2. $|\ \text{A}\ | = 30.000 \quad sel(\text{Ah=786}) = 0,015 \quad sel(\text{Av}\neq 0) = 0,1$
   $|\ \text{C}\ | = 400.000 \quad sel(\text{Cg=78}) = 0,3$

Gehen Sie dabei wie folgt vor:

(i) Bestimmen Sie für jede der drei Selektionen, für die ein Index existiert, mithilfe der Kostenfunktionen, ob der Zugriff per Scan oder Index erfolgen soll.

(ii) Ersetzen Sie die verbliebenen logischen Operatoren durch physische Operatoren. Sie müssen hierbei nicht den ganzen Baum zeichnen, machen Sie allerdings kenntlich wie sich Ihre Übersetzungen in den Baum einfügen.

Beachten Sie, dass ein Index nur auf Tabellen existiert, nicht auf Zwischenergebnissen. Soll also beispielsweise ein Indexzugriff auf Ah=786 erfolgen, muss die Reihenfolge der beiden Prädikate auf T getauscht werden. Zudem kann daher pro Tabelle nur ein Indexzugriff erfolgen. Des Weiteren können bei einem Scan mehrere Prädikate ohne zusätzliche Kosten ausgewertet werden. Dies bedeutet insbesondere, dass Ah=786∧Av≠0 mit einem Scan ausgewertet werden kann. Da wir keine physischen Layouts besprochen haben, gehen wir davon aus, dass Scans auf einem Zwischenergebnis, das durch einen Indexzugriff entstanden ist, keine weiteren Kosten verursachen.

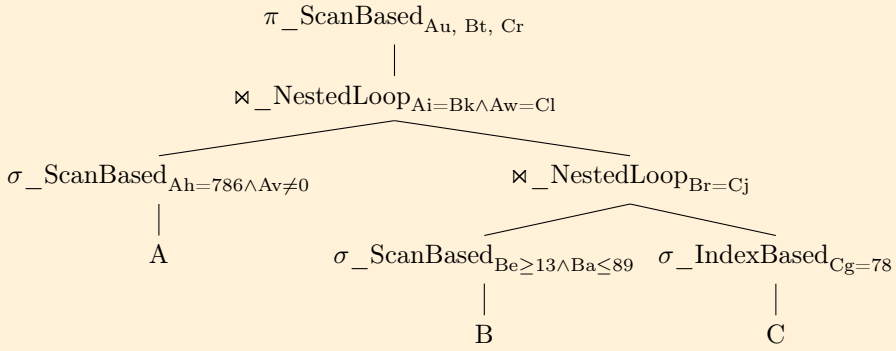
(b) Ab welcher Selektivität lohnt es sich den Index zu verwenden? Geben Sie die Selektivität in Abhängigkeit zur Tabellengröße an und interpretieren Sie Ihr Ergebnis, das heißt betrachten Sie, wie sich die Funktion für beliebig große Tabellengrößen verhält.

Lösung:

(a) 1. Vorgesehen sind 2 Punkte (je 0,5 Punkte Abzug für inkorrekte Berechnung - aber Folgefehler für physische Operatoren, inkorrekte physische Operatoren, nicht zusammengefasste Selektionen bei A oder B, usw.).

$$
\begin{aligned}
scan(\text{A}) &= 70.000 \\
index(\text{A,Ah=786}) &= log_2(70.000) + 37 \cdot 0,2 \cdot 70.000 \approx 518016 \\
index(\text{A,Av}\neq 0) &= log_2(70.000) + 37 \cdot 0,05 \cdot 70.000 \approx 129516 \\
scan(\text{C}) &= 230.000 \\
index(\text{C,Cg=78}) &= log_2(230.000) + 37 \cdot 0,02 \cdot 230.000 \approx 170218
\end{aligned}
$$

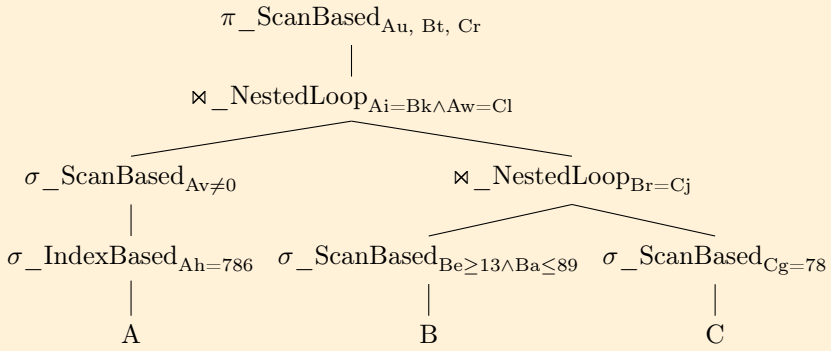
Folglich lohnt sich bei C ein Indexzugriff über Cg=78. Bei A hingegen lohnt sich kein Indexzugriff und wir verwenden einen scan-basierten Zugriff über Ah=786∧Av≠0.



2. Vorgesehen sind 2 Punkte (je 0,5 Punkte Abzug für inkorrekte Berechnung - aber Folgefehler für physische Operatoren, inkorrekte physische Operatoren, nicht zusammengefasste Selektionen bei B, falsche Selektionsreihenfolge bei A, usw.).

$$
\begin{aligned}
scan(\text{A}) &= 30.000 \\
index(\text{A,Ah=786}) &= log_2(30.000) + 37 \cdot 0,015 \cdot 30.000 \approx 16665 \\
index(\text{A,Av}\neq 0) &= log_2(30.000) + 37 \cdot 0,1 \cdot 30.000 \approx 111015 \\
scan(\text{C}) &= 400.000 \\
index(\text{C,Cg=78}) &= log_2(400.000) + 37 \cdot 0,3 \cdot 400.000 \approx 4440019
\end{aligned}
$$

Folglich lohnt sich bei C kein Indexzugriff und wir verwenden einen scan-basierten Zugriff über Cg=78. Bei A lohnt sich ein Indexzugriff über Ah=786, nicht aber über Av≠0. Somit müssen die Selektionen getauscht werden, um zuerst den Indexzugriff über Ah=786 und danach den scan-basierten Zugriff über Av≠0 zu verwenden.



(b) Vorgesehen ist 1 Punkt (0,5 Punkte für die Berechnung und 0,5 Punkte für die Interpretation).

Hierzu müssen wir die Ungleichung $index(T,p) \leq scan(T)$ nach $sel(p)$ auflösen.

$$\begin{aligned} index(T,p) &\leq scan(T) \\ log_2(\mid T \mid) + 37 \cdot sel(p) \cdot \mid T \mid &\leq \mid T \mid \\ 37 \cdot sel(p) \cdot \mid T \mid &\leq \mid T \mid - log_2(\mid T \mid) \\ sel(p) &\leq \frac{\mid T \mid - log_2(\mid T \mid)}{37 \cdot \mid T \mid} \\ sel(p) &\leq \frac{1}{37} - \frac{log_2(\mid T \mid)}{37 \cdot \mid T \mid} \end{aligned}$$

Die entscheidende Grenze liegt knapp unter $\frac{1}{37}$. Je größer die Tabelle, desto verhältnismäßig geringer fällt der zweite Bruch in der Kostenfunktion ins Gewicht und desto näher liegt die Grenze tatsächlich an $\frac{1}{37}$.

2 Kanonische Übersetzung (4 Punkte)

In der Vorlesung haben wir SQL zunächst kanonisch in annotierte relationale Algebra übersetzt. Alternativ könnten wir SQL direkt in einen Query-Graphen übersetzen, ohne Umweg über die relationale Algebra. Diskutieren Sie Vor- und Nachteile beider Ansätze.

Lösung: 4 Punkte, 1 Punkt pro Stichpunkt. Andere korrekte Aussagen werden auch akzeptiert.

Vorteile:

- Der Weg über relationale Algebra wird überflüssig und man vermeidet somit auch inhärente Probleme der relationen Algebra, siehe Umbenennung.
- Teile der heuristischen Optimierung sind automatisch im Query-Graphen verankert, siehe Predicate Pushdown und Zusammenfassen von Selektion mit kartesischem Produkt zu einem Join (das kartesische Produkt ist ein Artefakt der kanonischen Übersetzung).

Nachteile:

- Teilweise können heuristische Optimierungen nicht ohne Weiteres auf dem Query-Graphen ausgeführt werden, z.B. Projection Pushdown. Aufsammeln aller benötigten Attribute einer Relation ist dabei vergleichsweise einfach; Projektionen, die in der Baumdarstellung zwischen Operatoren stehen, sind nicht ganz so einfach, da hier die Reihenfolge der Operatoren eine Rolle spielt.
- Der Query-Graph, so wie in der Vorlesung definiert, enthält nur Joins und Filter. Andere Operatoren wie zum Beispiel Gruppierung können nicht dargestellt werden. Dafür muss der Query-Graph erweitert werden. (Generell kann der Query-Graph auch um andere Aspekte wie physische Operatoren erweitert werden.) Weiterhin sind Projektionen nicht definiert, insbesondere nicht die finale Projektion für das Ergebnis der Anfrage (was in SQL im SELECT steht).

3 Plan Enumeration (5 Punkte)

Betrachten Sie die folgende SQL Anfrage.

```
SELECT A.c, B.g, C.r, D.v
FROM   A, B, C, D
WHERE  A.d = B.k AND B.r = C.v AND C.q = D.o AND D.t = B.u ;
```

Nehmen Sie dabei die folgenden Ergebnisgrößen an:

Teilproblem	Ergebnisgröße
{A}	600
{B}	300
{C}	150
{D}	400
{A,B}	9.000
{A,C}	90.000
{A,D}	240.000
{B,C}	2.700
{B,D}	2.400
{C,D}	2.400
{A,B,C}	81.000
{A,B,D}	72.000
{A,C,D}	1.440.000
{B,C,D}	864
{A,B,C,D}	25.920

In obiger Tabelle beschreibt beispielsweise Teilproblem {A,B,C} alle Joinreihenfolgen, um die drei Tabellen A, B und C miteinander zu verknüpfen, unabhängig davon, ob zwischen den einzelnen Tabellen ein Joinprädikat existiert.
Ihnen steht ausschließlich ein einfacher hashbasierter Join mit der folgenden Kostenfunktion zur Verfügung:

$$C_{\text{HashJoin}}(R \bowtie S) = |R| + |S|$$

Beachten Sie, dass diese Kostenfunktion nur bei Joins benutzt werden kann. Im Fall eines kartesischen Produktes ergeben sich die Kosten folgendermaßen:

$$C(R \times S) = |R| \cdot |S|$$

(a) Bestimmen Sie die optimale Joinreihenfolge ausschließlich unter Berücksichtigung von “left-deep” Plänen. Ergänzen und erweitern Sie hierzu die folgende Tabelle um Teilpläne der Größen zwei, drei und vier:

Teilplan	Kosten	Ergebnisgröße
A	0	600
B	0	300
C	0	150
D	0	400
A $\bowtie$ B		
A $\times$ C		
A $\times$ D		
B $\bowtie$ C		
B $\bowtie$ D		
C $\bowtie$ D		

Achten Sie darauf, dass sich die Kosten eines Joins aus den Kosten der Berechnung der Inputs sowie der Ausführung des Joins zusammensetzen. Da die Kostenfunktion symmetrisch ist, muss bei Teilplänen der Größe zwei die Reihenfolge nicht beachtet werden, beispielsweise müssen also A $\bowtie$ C und C $\bowtie$ A oben nicht separat aufgelistet werden.

(b) Zeichnen Sie den Joingraphen für die gegebene Anfrage. Beschriften Sie die Kanten mit den jeweiligen Joinprädikaten.

(c) Wie hätten Sie mithilfe des Joingraphen gewisse Einträge der Tabelle aus Aufgabenteil (a) von vorneherein ausschließen können?

Lösung:

(a) Vorgesehen sind 3 Punkte (je 0,5 Punkte Abzug für falsche Berechnungen unter Berücksichtigung von Folgefehlern, usw.).

Es gilt

$$\begin{aligned} C(R \bowtie S) &= C(R) + C(S) + C_{HashJoin}(R \bowtie S) \\ &= C(R) + C(S) + |R| + |S| \end{aligned}$$

oder:

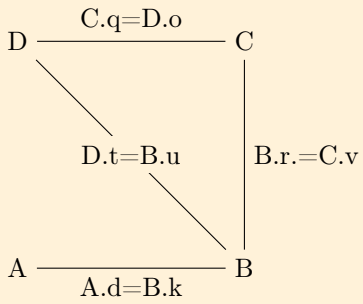
$$\begin{aligned} C(R \times S) &= C(R) + C(S) + C(R \times S) \\ &= C(R) + C(S) + |R| \cdot |S| \end{aligned}$$

Mit diesen Formeln können wir die komplette Tabelle ausfüllen.

Teilplan	Kosten	Ergebnisgröße
A	0	600
B	0	300
C	0	150
D	0	400
A $\bowtie$ B	900	9.000
A $\times$ C	90.000	90.000
A $\times$ D	240.000	240.000
B $\bowtie$ C	450	2700
B $\bowtie$ D	700	2400
C $\bowtie$ D	550	2400
(A $\bowtie$ B) $\bowtie$ C	10.050	81.000
(A $\times$ C) $\bowtie$ B	180.300	81.000
(B $\bowtie$ C) $\bowtie$ A	3.750	81.000
(A $\bowtie$ B) $\bowtie$ D	10.300	72.000
(A $\times$ D) $\bowtie$ B	480.300	72.000
(B $\bowtie$ D) $\bowtie$ A	3.700	72.000
(A $\times$ C) $\bowtie$ D	180.400	1.440.000
(A $\times$ D) $\bowtie$ C	480.150	1.440.000
(C $\bowtie$ D) $\times$ A	1.440.550	1.440.000
(B $\bowtie$ C) $\bowtie$ D	3.550	864
(B $\bowtie$ D) $\bowtie$ C	3.250	864
(C $\bowtie$ D) $\bowtie$ B	3.250	864
((B $\bowtie$ C) $\bowtie$ A) $\bowtie$ D	85.150	25.920
((B $\bowtie$ D) $\bowtie$ A) $\bowtie$ C	75.850	25.920
((A $\times$ C) $\bowtie$ D) $\bowtie$ B	1.620.700	25.920
((B $\bowtie$ D) $\bowtie$ C) $\bowtie$ A	4.714	25.920
((C $\bowtie$ D) $\bowtie$ B) $\bowtie$ A	4.714	25.920

Die optimale Joinreihenfolge ist ((C $\bowtie$ D) $\bowtie$ B) $\bowtie$ A oder ((B $\bowtie$ D) $\bowtie$ C) $\bowtie$ A.

(b) Vorgesehen ist 1 Punkt (je 0,5 Punkte Abzug für falsche Knoten(-benennung), falsche oder fehlende Kanten, fehlende Joinprädikate, usw.).



(c) Vorgesehen ist 1 Punkt.

Aufgrund der impliziten Selektivität von 1 und den damit verbundenen großen Zwischenergebnissen, die in die Kosten der Joins einberechnet werden, müssen Pläne, die ein kartesisches Produkt enthalten nicht berücksichtig werden. Somit hätten wir uns 8 Zeilen sparen können.

4 Aufzählen aller Anfragepläne unter Ausschluss von suboptimalen Teilplänen (6 Punkte)

Im Notebook Plan Enumeration.ipynb haben wir bereits gesehen, wie wir für eine gegebene Kostenfunktion durch vollständiges Aufzählen aller Planalternativen für eine Anfrage den günstigsten Plan bestimmen können.

Implementieren Sie nun die Funktion `find_cheapest_plan_with_pruning()` im beigefügten Notebook, die im Gegensatz zu `find_cheapest_plan_exhaustive()` Pläne ignoriert, die suboptimale Teilpläne enthalten. Hiermit sind Teilpläne von Relationen gemeint, für die eine bessere Joinreihenfolge existiert. Beachten Sie hierbei folgende Hinweise:

- Sie sollten in Ihrer Implementierung dynamische Programmierung verwenden. Das heißt, sie müssen zunächst die optimalen Pläne (und die jeweiligen Kosten und Ergebnisgrößen) für Joins der Länge n berechnen. Diese müssen daraufhin gespeichert und genutzt werden um die optimalen Pläne (sowie Kosten und Ergebnisgrößen) für Joins der Länge $m > n$ zu berechnen.
- Ihre Funktion soll "bushy"-Pläne (und damit auch "left-deep"- und "right-deep"-Pläne) in Betracht ziehen. Sie können davon ausgehen, dass der übergebene Joingraph zusammenhängend ist.
- Sie dürfen annehmen, dass die Kostenfunktion symmetrisch ist.
- Verwenden Sie in Ihrer Implementierung die Funktion `compute_cost_and_size()`, die die Kosten, sowie die Ergebnisgröße eines Joins berechnet. Beachten Sie, dass die resultierenden Kosten bereits rekursiv die Kosten der Teilpläne beinhalten.
- Sowohl in der Ausgabe von `find_cheapest_plan_with_pruning()`, als auch der Eingabe von `compute_cost_and_size()` wird ein bestimmtes Format Ihrer Pläne erwartet. Joinpläne werden dabei rekursiv als Tupel definiert, so würde zum Beispiel der Plan "$\big((\mathrm{A} \bowtie \mathrm{B}) \bowtie (\mathrm{C} \bowtie \mathrm{D})\big) \bowtie \mathrm{E}$" die folgende Struktur haben:

$$\Big(\big((\mathrm{A}, \mathrm{B}), (\mathrm{C}, \mathrm{D})\big), \mathrm{E}\Big)$$

- Ihre Funktion soll ein Tupel zurückgeben, dass den endgültigen Joinplan, sowie die zugehörigen Kosten enthält.
- Unsere Referenzimplementierung benötigt für das komplexe Beispiel der IMDb-Datenbank weniger als eine Sekunde. Ihr Implementierung sollte sich an dieser Zeit orientieren.

Abgabe

Lösungen sind in Teams von 2 bis 3 Studierenden bis zum 9. Juni 2022, 10:15 Uhr über Ihre persönlichen Statusseite im CMS einzureichen. Nutzen Sie hierfür die Team Groupings Funktionalität im CMS.

Ihre Abgabe muss dem folgenden Format entsprechen:

```
abgabe.zip
├── abgabe.pdf
└── jupyter.txt
```

Hierbei enthält `abgabe.pdf` Ihre Lösungen zu Aufgabe 1, 2 und 3 und `jupyter.txt` Ihre Lösung zu Aufgabe 4. Achten Sie darauf, dass Sie nur die von Ihnen zu ergänzenden Jupyter Zellen so kopieren, dass Einrückung und Formatierung korrekt sind.

Abgaben, die nicht den oben angegeben Vorgaben entsprechen, führen zu Punktabzug. Einzelabgaben werden nicht mehr korrigiert und mit 0 Punkten bewertet.