Aufgabenblatt 4
Anfrageoptimierung

- Abgabetermin: **Sonntag, 10.01.2021 (23:59 Uhr)**
- Zur Prüfungszulassung muss ein Aufgabenblatt mit mind. 25% der Punkte bewertet werden und alle weiteren Aufgabenblätter mit mindestens 50% der Punkte.
- Die Aufgaben sollen in Zweiergruppen bearbeitet werden.
- Abgabesystem unter `http://www.dcl.hpi.uni-potsdam.de/submit`
  - Für die nicht Programmieraufgaben:
    - ausschließlich pdf-Dateien
    - eine Datei pro Aufgabe namens Aufgabe-<aufgabenNr>.pdf
    - *jedes* Blatt beschriftet mit Namen
  - Für die Programmieraufgabe lediglich die Klasse *HashEquiJoin.java*, bzw. ein Zip-Archiv aller Klassen, falls ihr zusätzliche Hilfsklassen verwendet habt (das sollte aber eigentlich nicht nötig sein)

Aufgabe 1: Kardinalitätsschätzung

Gegeben seien die folgenden Relationen und deren Statistiken:

$W(a,b)$	$X(b,c)$	$Y(c,d)$	$Z(d,a)$
$T(W) = 300$	$T(X) = 600$	$T(Y) = 900$	$T(Z) = 1200$
$V(W,a) = 30$			$V(Z,a) = 300$
$V(W,b) = 60$	$V(X,b) = 50$		
	$V(X,c) = 100$	$V(Y,c) = 50$	
		$V(Y,d) = 60$	$V(Z,d) = 40$

Schätzen Sie die Kardinalität der Ergebnisrelationen der folgenden Ausdrücke: **9 P**

a) $W \bowtie X \bowtie Y \bowtie Z$

b) $\sigma_{a=10}(W)$

c) $\sigma_{c=20}(Y)$

d) $\sigma_{c=20}(Y) \bowtie Z$

e) $W \times Y$

f) $\sigma_{d>10}(Z)$

g) $\sigma_{a=1 \wedge d=2}(Z)$

h) $\sigma_{c>1 \wedge d=2}(Y)$

i) $X \overset{\bowtie}{}_{X.b=Z.d} Z$

Aufgabe 2: Join-Kardinalität

Gegeben sind zwei Relationen $R(A, B)$ und $S(B, C)$. Beide Relationen enthalten 20 unterschiedliche Werte im Attribut B, wobei die Werte in R den Werten in S entsprechen. Es gilt also $V(R, B) = V(S, B) = 20$. Die Werteverteilungen in $R(B)$ und $S(B)$ sind durch folgendes Histogramm beschrieben, welches die Häufigkeit der 4 häufigsten Werte angibt:

	0	1	2	3	4	andere Werte
$R.B$	5	6	4	5	*	32
$S.B$	10	8	5	*	7	48

Die mit * gekennzeichneten Werte gehören nicht zu den vier jeweils häufigsten Werten, sondern zu den "anderen Werten". Schätzen Sie nun unter Verwendung des Histogramms die Kardinalität des Joins über $R(A, B) \bowtie S(B, C)$. Schätzen Sie anschließend die Kardinalität, ohne das Histogram zu verwenden (wie üblich Annahme der Gleichverteilung aller 20 Attributwerte). Vergleichen Sie die beiden Ergebnisse. **5 P**

Aufgabe 3: Join-Implementierung

In dieser Aufgabe geht es um die Implementierung von Join-Algorithmen in einem dafür zur Verfügung gestellten Java-Framework[1]. Sie können sich bei Ihrer Implementierung an der Nested Loop Join-Implementierung in der Klasse `NestedLoopEquiJoin` orientieren. Um uns auf den Algorithmus konzentrieren zu können, machen wir folgende Vereinfachungen:

- Es gibt nur den `String` Datentyp.
- Wir implementieren nur den Equi-join mit einer einzigen Bedingung, nämlich der Äquivalenz zweier Spalten.

Die Schnittstelle für den Join sowie Hilfskassen, Build-Prozess, Blockzustände etc. sind in der Markdown-Datei `README.md` im Root-Verzeichnis beschrieben. Beim Bearbeiten der Aufgabe gelten folgende Regeln:

- Der an den Join-Algorithmus übergebene `BlockManager` dient als einzige Schnittstelle, um Blöcke einzulesen bzw. neue Blöcke zu erzeugen bzw. deren Speicher wieder freizugeben. Der verfügbare Speicher ist begrenzt.
  Um diese Beschränkung einzusehen, gibt es die `getFreeBlockCount`-Methode.
- Sämtlicher weiterer Speicher, den Sie für den Algorithmus benötigen (z.B. für Datenstrukturen, die Referenzen auf die bereits reservierten Blöcke speichern), sollte so gering wie möglich gehalten werden.

Lösen Sie die folgenden Aufgaben:

a) Implementieren Sie in der schon angelegten Datei `HashEquiJoin.java` einen einfachen Hash-Join (keinen Hybrid-Hash-Join). Stellen Sie sicher, dass Ihre Implementierung ohne Änderungen in anderen Dateien auskommt und geben Sie auch nur diese Datei ab. **8 P**

b) Beantworten Sie außerdem folgende Fragen: **2 P**

   - Warum weicht die Schätzung beim Hash-Join oft von den gemessenen IO-Kosten ab?
   - Wie lässt sich diese Abweichung erklären? Auf welche Annahme aus der Vorlesung geht diese Abweichung zurück?
   - Bei welchen Parameter-Einstellungen wird die Abweichung größer? Auf welche Annahme(n) aus der Vorlesung geht diese zusätzliche Abweichung zurück?

Hinweise:

- Die Klasse `Main` kann verwendet werden, um die Join-Implementierungen zu testen. Drei Testdateien sind ebenfalls in den Materialien[2] zu finden. Die drei Relationen haben die Film-ID als erstes Attribut, welches sich gut als Join-Attribut eignet. Das Join-Ergebnis für einen solchen Join von `title.basics.sample` and `title.principals.sample` umfasst 3563 Tupel.
- Nutzen Sie gerne die Mailingliste, z.B. für Probleme beim build bzw. allgemeine Fragen.

[1] `https://github.com/HPI-Information-Systems/dbs2-join-exercise`

[2] `https://hpi.de/internalfiles/materialien/FG\%20Informationssysteme/VL\%20DBS\%20II/Programmier\%C3\%BCbung/imdb.sample.zip`

Aufgabe 4: Freiwillig: Join-Reihenfolge

Hinweis: Diese Aufgabe ist für das Bestehen des Übungszettels freiwillig und dient nur der Übenung. Diese Aufgabe bringt also keine Punkte. Wir korrigieren die Aufgabe jedoch. Falls Sie also unsicher sind, ob sie die Bestimmung der Join-Reihenfolge verstanden haben, ist es sicher hilfreich diese Aufgabe zu lösen.

Gegeben seien die folgenden Relationen und deren Statistiken:

$E(a,b,c)$	$F(a,b,d)$	$G(a,c,d)$	$H(b,c,d)$
$T(E) = 1000$	$T(F) = 2000$	$T(G) = 3000$	$T(H) = 4000$
$V(E,a) = 1000$	$V(F,a) = 50$	$V(G,a) = 50$	
$V(E,b) = 50$	$V(F,b) = 100$		$V(H,b) = 40$
$V(E,c) = 20$		$V(G,c) = 300$	$V(H,c) = 100$
	$V(F,d) = 200$	$V(G,d) = 500$	$V(H,d) = 400$

Bestimmen Sie die Join-Reihenfolge als Left Deep Tree. Geben Sie dazu alle Tabellen des Algorithmus der Dynamischen Programmierung an. Was ist die optimale Join-Reihenfolge, wie hoch sind deren Kosten und welche Kardinalität hat der Join am Ende?

Hinweis: Bei dieser Aufgabe können bei der Berechung für Kardinalität und Kosten Bruchzahlen herauskommen. Auch wenn man in der Realität keine Kardinalität von z.B. 1/10 Tupeln erwartet, können Sie in dieser Aufgabe damit ruhig weiterrechnen. 0 P