TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

Übung zur Vorlesung
Einsatz und Realisierung von Datenbanksystemen im SoSe16

Moritz Kaufmann (moritz.kaufmann@tum.de)
http://db.in.tum.de/teaching/ss16/impldb/

Blatt Nr. 07

Hausaufgabe 1

Geben die Relation Klausur:

MatrNr	Vorbereitungszeit	Note
1	150	1.7
2	70	2.7
3	450	2.0
4	180	1.7
5	2500	1.3

a) Formulieren Sie die Anfrage, die die MatrNr in der Skyline für die Attribute Vorbereitungszeit und Note erzeugt (kleiner ist jeweils besser) in SQL mit Hilfe des Skyline Operators.

b) Formulieren Sie die Anfrage in SQL ohne Skyline Operator.

c) Bestimmen Sie das Ergebnis der Anfrage.

SQL mit Skyline:
select MatrNr from Klausur k skyline of k.Vorbereitungszeit min, k.Note min
SQL ohne Skyline:
select MatrNr from Klausur k
where not exists (
select * from klausur dom
where
dom.Vorbereitungszeit <= k.Vorbereitungszeit and
dom.Note <= k.Note and (
dom.Vorbereitungszeit < k.Vorbereitungszeit or
dom.Note < k.Note)
)

Ergebnis:

1) Ist in Skyline (Kann in Vorbereitungszeit nur von MatrNr 2 dominiert werden, dort ist aber Note schlechter)

2) Ist in Skyline (Minimum für Vorbereitungszeit)

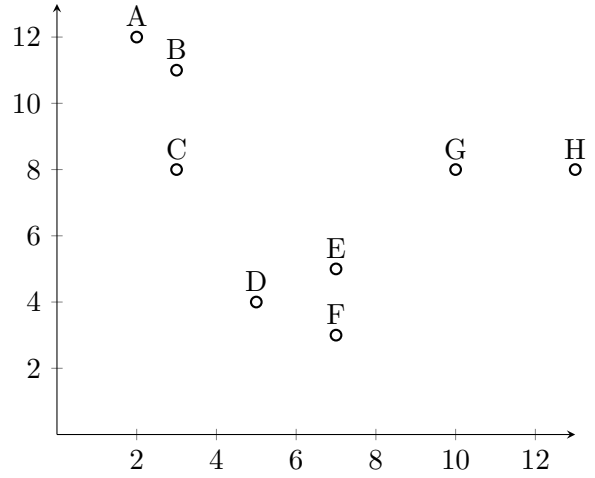
3) Ist nicht in Skyline, dominiert von MatrNr 1

4) Ist nicht in Skyline, dominiert von MatrNr 1

5) Ist in Skyline (Minimum für Note)

Hausaufgabe 2

Folgende Datenpunkte im euklidischen Raum seien gegeben:



Punkt	X	Y
A	2	12
B	3	11
C	3	8
D	5	4
E	7	5
F	7	3
G	10	8
H	13	8

Clustern Sie die Punkte mithilfe des *k-means*-Verfahren in 3 Cluster. Nutzen Sie als initiale Clusterzentren die Werte A, B und C. Wenn ein Punkt zu mehreren Clustern die gleiche Distanz hat, wird er dem Cluster der näher am Nullpunkt liegt zugeordnet. Geben Sie für jede Iteration jeweils die Zuordnung und die Mittelpunkte der Cluster an.
Eine Iteration des K-Mean Algorithmuss kann wie folgt ausgewertet werden:

```
with points(id,x,y) as (
        VALUES ('A', 2, 12), ('B', 3, 11), ('C', 3,8), ('D', 5,4),
        ('E',7,5),('F',7,3),('G',10,8),('H',13,8)
),
clusters_0(cid,x,y) as (
  VALUES ('1', 2, 12), ('2', 3, 11), ('3', 3,8)
),
clusters_1(cid, x,y, count) as (
  select cid, avg(px), avg(py), count(*) from (
    select cid, p.x as px, p.y as py, rank() OVER (
      partition by p.id
      order by (p.x-c.x)*(p.x-c.x)+(p.y-c.y)*(p.y-c.y) asc,
        (c.x*c.x+c.y*c.y) asc)
    from points p, clusters_0 c
  ) x
  where x.rank=1
  group by cid
)
```

Die Clusterzentren können mit folgender Abfrage ausgegeben werden

```
select * from clusters_1
```

Die Zuordnung kann mit folgender Abfrage ausgewertet werden

```
select cid,pid from (
    select cid, p.id as pid, rank() OVER (
        partition by p.id
        order by (p.x-c.x)*(p.x-c.x)+(p.y-c.y)*(p.y-c.y) asc,
            (c.x*c.x+c.y*c.y) asc)
    from points p, clusters_1 c
) x
where x.rank=1
```

Hausaufgabe 3

In Hauptspeicherdatenbanken ist die Geschwindigkeit oft durch Limitierungen des Speichersystems begrenzt. Analysieren sie dazu folgende Fragestellungen:

1. Was versteht man unter NUMA und welche Schichten gibt es in der Speicherhierarchie? Geben Sie zu jeder Schicht auch die Zugriffszeiten und Bandbreite an.
2. Was bedeuten die Begriffe *Cacheline* und *Seite*. Auf welcher Schicht sind diese jeweils relevant?

0.0.1 NUMA

Non-Uniform Memory Access oder kurz NUMA ist eine Computer-Speicher-Architektur für Multiprozessorsysteme, bei denen jeder Prozessor einen eigenen, lokalen Speicher hat, aber anderen Prozessoren über einen gemeinsamen Adressraum direkten Zugriff darauf gewährt (Distributed Shared Memory). Die Speicherzugriffszeiten in einem solchen Verbund hängen daher davon ab, ob sich eine Speicheradresse im lokalen oder im fremden Speicher befindet (Wikipedia).
Bei der Programmierung von Programmen ist diese Trennung nicht sichtbar, bei der Ausführung kann diese aber zu großen Geschwindigkeitsschwankungen führen.

0.0.2 Speicherhierarchie

Register <1ns Zugriffszeit

L1 2ns Zugriffszeit, Bandbreite: 16 byte/Takt, 44 GBytes/s *pro CPU Kern*

L2 20ns Zugriffszeit, Bandbreite: 16 byte/Takt, 44 GBytes/s *pro CPU Kern*

Hauptspeicher 200ns Zugriffszeit, Bandbreite: 50 GByte/s to local socket, 16 GByte/s to remote socket

Festplatte 5ms Zugriffszeit, Bandbreite: 1GByte/s

Wichtig ist vor allem ein Gefühl für die Größenordnungen.

0.0.3 Zugriffsgranularität

Wenn der Prozessor auf einen Datenwert zugreift, müssen die entsprechenden Daten aus dem Hauptspeicher bzw Cache geladen werden. Eine Cache-Line ist die kleinste Verwaltungseinheit innerhalb des Caches von Prozessoren. Die Zugriffe vom Cache-Speicher zur CPU oder zum Hauptspeicher erfolgen somit in einem einzigen, blockweisen Transfer. Falls z.B. auf eine Zahl in einem der Caches zugegriffen wird muss immer die gesamte Cacheline, ein Block von 64-Byte Größe, geladen werden. Genauso beim Schreiben: Selbst wenn nur ein einzelnes Byte verändert wurde muss immer die ganze Cacheline aktualisiert werden. Die Verwaltung des Hauptspeichers durchs Betriebssystem erfolgt in noch gröberen Blöcken, sogenannten Seiten. Eine Seite ist dabei typischerweiße 4 kilobyte groß. Dies ist typischerweiße auch die minimale Granularität mit der Daten auf die Festplatte geschrieben werden können.

Hausaufgabe 4

Gegeben eine Tabelle *Produkte* mit folgendem Schema und 10000 Einträgen:

Id (8 Byte) | Name (32 Byte) | Preis (8 Byte) | Anzahl (8 Byte)

Wieviele Daten werden für folgende Queries in die CPU-Caches geladen? Unterscheiden sie jeweils zwischen Row und Column Store.

1. *select * from Produkte*

2. *select Anzahl from Produkte*

Daten können maximal mit Granularität (64 Byte) in den Cache geladen werden. Das heißt, selbst wenn nur auf einen 64 bit Integer Wert zugegriffen wird, muss ein kompletter 64-Byte Block geladen werden. Mit diesem Hintegrund ergeben sich folgende Ergebnisse:

1. *select * from Produkte*
   a) Row: $10000 * 56 = 560000$ Byte
   b) Column: $10000 * 8 + 10000 * 32 + 10000 * 8 + 10000 * 8 = 560000$ Byte
2. *select Anzahl from Produkte*
   a) Row: $10000 * 56 = 560000$ Byte
   b) Column: $10000 * 8 = 80000$ Byte