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

Übung zur Vorlesung
***Einsatz und Realisierung von Datenbanksystemen* im SoSe14**
Moritz Kaufmann (moritz.kaufmann@tum.de)
http://www-db.in.tum.de/teaching/ss14/impldb/

Blatt Nr. 9

Aufgabe 1

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 den CPU-Cache geladen? Unterscheiden sie jeweils zwischen Row und Column Store.

1. *select * from Produkte*
2. *select Anzahl from Produkte*

Daten können maximal mit Cacheline Granularität (64 Byte) in den Cache geladen werden. Das heißt, selbst wenn nur auf einen 64 bit Integer Wert zugegriffen wird, muss die komplette Cacheline 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

Aufgabe 2

Schätzen sie die Anzahl der Cache Misses die entstehen, wenn man 1000 32-bit Integer Werte (0-1000) in aufeinanderfolgender Reihenfolge in einen ART Baum einfügt. Wäre ein B+ Baum besser oder schlechter?

Größe der einzelnen ART Knoten (mit 64bit Pointern und ohne Header):

Node4 $4 + 4 * 8 = 36$ Byte

Node48 $256 + 48 * 8 = 640$ Byte

Node256 $256 * 8 = 2048$ Byte

Die Höhe eines ART-Baums ist durch die Schlüssellänge beschränkt (in unserem Fall maximal Höhe 4), da in jedem Knoten ein Byte des Schlüssels gespeichert wird. Da die Integerzahlen aufeinanderfolgend sind, unterscheiden sie sich maximal in den letzten zwei Bytes. Für die ersten zwei Bytes reicht es einen Node4 zu nehmen, da hier alle Einträge den selben Wert besitzen. Auf dem letzten Level reichen 4 Node256 um die letzten Bytes der 1000 Integer Werte einzufügen. Da es nur vier Kindnoten gibt reicht auf Level drei auch ein Node4. Die Gesamtgröße des Baums ist somit $4 * 2048 + 3 * 36 = 8300$ Byte. Dies passt locker in den L1 Cache heutiger CPUs der typischerweiße 64 KB groß ist. Somit gibt es keine Cache Misses.
Ein B+ Baum ist schlechter, da bei sequentiellen Einfügen die Knoten nur halb gefüllt sind. Außerdem werden in den Knoten jeweils pro Pointer auch noch ein ein kompletter 32 bit Rangeschlüssel gespeichert was den Speicherbedarf zusätzlich erhöht.

Aufgabe 3

Formulieren Sie die zuvor in SQL bearbeiteten Anfragen zur Universitätsdatenbank in XQuery. Erstellen Sie insbesondere XQuery-Anfragen, um folgende Fragestellungen zu beantworten:

a) Suchen Sie die Professoren, die Vorlesungen halten.

b) Finden Sie die Studenten, die alle Vorlesungen gehört haben.

c) Finden Sie die Studenten, die alle vierstündigen Vorlesungen gehört haben.

d) Finden Sie die Studenten mit der größten Semesterzahl unter Verwendung von Aggregatfunktionen.

e) Berechnen Sie die Gesamtzahl der Semesterwochenstunden, die die einzelnen Professoren erbringen. Dabei sollen auch die Professoren berücksichtigt werden, die keine Vorlesungen halten.

f) Finden Sie die Namen der Studenten, die in keiner Prüfung eine bessere Note als 3.0 hatten.

g) Berechnen Sie den Umfang des Prüfungsstoffes jedes Studenten. Es sollen der Name des Studenten und die Summe der Semesterwochenstunden der Prüfungsvorlesungen ausgegeben werden.

h) Finden Sie Studenten, deren Namen den eines Professors enthalten.

i) Ermitteln Sie den Bekanntheitsgrad der Professoren unter den Studenten, wobei wir annehmen, dass Studenten die Professoren nur durch Vorlesungen oder Prüfungen kennen lernen.

```
(: a) Suchen Sie die Professoren, die Vorlesungen halten. :)
doc('uni2')//ProfessorIn[.//Vorlesung]/Name

(: b) Finden Sie die Studenten, die alle Vorlesungen gehört haben. :)
doc('uni2')//Student[count(tokenize(hoert/@Vorlesungen," "))=count(//Vorlesung)]/Name

(: c) Finden Sie die Studenten, die alle vierstundigen Vorlesungen
   gehört haben. :)
let $fourcount:=count(doc('uni2')//Vorlesung[SWS=4])
for $s in doc('uni2')//Student
where count(
  for $h in tokenize($s/hoert/@Vorlesungen," ")
  where doc('uni2')//Vorlesung[@VorlNr=$h and SWS=4]
  return $h
) = $fourcount
return $s/Name

(: d) Finden Sie die Studenten mit der größten Semesterzahl unter
  Verwendung von Aggregatfunktionen. :)
let $maxsws:=max(data(doc('uni2')//Student/Semester))
return doc('uni2')//Student[Semester=$maxsws]


(: e) Berechnen Sie die Gesamtzahl der Semesterwochenstunden, die die
  einzelnen Professoren erbringen. Dabei sollen auch die Professoren
  berücksichtigt werden, die keine Vorlesungen halten. :)
for $p in doc('uni2')//ProfessorIn
return <Prof>{$p/Name}<Summe>{sum(data($p//SWS))}</Summe></Prof>


(: f) Finden Sie die Namen der Studenten, die in keiner Prufung
   eine bessere Note als 3.0 hatten. :)
for $s in doc('uni')//Student
where count(
  for $p in $s//Pruefung
  where $p/@Note < 3
  return $p
) = 0
return $s

(: g) Berechnen Sie den Umfang des Prüfungsstoffes jedes Studenten.
   Es sollen der Name des Studenten und die Summe der Semesterwochenstunden
   der Prüfungsvorlesungen ausgegeben werden. :)
for $s in doc('uni')//Student
return <Student>{$s/Name}<sum>{
  sum(for $p in $s//Pruefung
      return doc('uni')//Vorlesung[@VorlNr=$p/@Vorlesung]/SWS)}
</sum></Student>
```

```
(: h) Finden Sie Studenten, deren Namen den eines Professors enthalten. :)
for $s in doc('uni')//Student
where doc('uni')//ProfessorIn[contains($s/Name,Name)]
return $s/Name


(: i) Ermitteln Sie den Bekanntheitsgrad der Professoren unter den Studenten,
   wobei wir annehmen, dass Studenten die Professoren nur durch Vorlesungen
   oder Prüfungen kennen lernen. :)
for $p in doc('uni')//ProfessorIn
return
  <Professor>
    {$p/Name}
    <Bekanntheit>
    {
      count(
        doc('uni')//Student[.//Pruefung[@Pruefer=$p/@PersNr]]/Name
        union
        ( for $v in $p//Vorlesung/@VorlNr
          return doc('uni')//Student[contains(hoert/@Vorlesungen,$v)]/Name)
      )
    }
    </Bekanntheit>
  </Professor>
```

Aufgabe 4

Geben Sie ein Vorlesungsverzeichnis aus, welches nach dem Umfang der Vorlesungen in SWS gruppiert ist.

Die Ausgabe Ihrer Anfrage soll wie folgt aufgebaut sein:

```
<Vorlesungsverzeichnis>
  <Vorlesungen SWS="2">
    <Vorlesung VorlNr="V5216" Titel="Bioethik"/>
    <Vorlesung VorlNr="V5259" Titel="Der Wiener Kreis"/>
    <Vorlesung VorlNr="V5022" Titel="Glaube und Wissen"/>
    <Vorlesung VorlNr="V5049" Titel="Maeeutik"/>
  </Vorlesungen>
  <Vorlesungen SWS="3">
    <Vorlesung VorlNr="V5043" Titel="Erkenntnistheorie"/>
    <Vorlesung VorlNr="V5052" Titel="Wissenschaftstheorie"/>
  </Vorlesungen>
  <Vorlesungen SWS="4">
    <Vorlesung VorlNr="V4630" Titel="Die 3 Kritiken"/>
    <Vorlesung VorlNr="V5041" Titel="Ethik"/>
    <Vorlesung VorlNr="V5001" Titel="Grundzuege"/>
    <Vorlesung VorlNr="V4052" Titel="Logik"/>
  </Vorlesungen>
</Vorlesungsverzeichnis>
```

```
<Vorlesungsverzeichnis>
{
  for $sws in distinct-values(doc('uni2')//SWS)
  order by $sws
  return
  <Vorlesungen SWS="{$sws}">
  {
    for $vl in doc('uni2')//Vorlesung[SWS=$sws]
    order by $vl/Titel
    return <Vorlesung VorlNr="{$vl/@VorlNr}" Titel="{$vl/Titel}" />
  }
  </Vorlesungen>
}
</Vorlesungsverzeichnis>
```