GDB Tutorium

Woche 11

Jigao Luo

TUM

15. Januar 2020

Wiederholung [1]–[3]

- die kanonische Übersetzung
- Join Typen
- Iteratorkonzept

Wiederholung - die kanonische Übersetzung

- SQL ist deklarativ, aber muss Anfrage in etwas prozedurales übersetzt werden.
- Die Kanonische Übersetzung ist eine Standardübersetzung von SQL in relationale Algebra. (Fast 1:1 Abbildung)

- `Select` $\equiv$ Projektion π
- `From R1, R2, ...` $\equiv$ Kreuzprodukt $\times$ von R1, R2, ...
- `where` $\equiv$ Selektion σ

Wiederholung - Nested Loop Join

```
for each r ∈ R
    for each s ∈ S
        if r.A = s.B then
            res := res ∪ (r × s)
```

- +: sehr einfach, Theta-Join und Semi-Join berechnen kann
- -: teuer, $O(|R| * |S|)$

Wiederholung - Sort Merge Join

Sort R and S
while r in [0, $|R|$] and s in [0, $|S|$]
    if R[r].A = S[s].B then
        res := res $\cup$ (R[r] $\times$ S[s]), s++
    if R[r].A $>$ S[s].B then
        s++
    if R[r].A $<$ S[s].B then
        r++

- +: Range-Query (z.b. $r < s, r > s$) erledigen kann (muss angepasst werden)
- +: normalerweise $O(|R| + |S|)$ falls sortiert
- (Worst case: R, S haben den gleichen Wert: $O(|R| * |S|)$)
- -: Sort ist auch teuer (Big Data: external Sort möglich)

Wiederholung - Index Join (angenommen R hat Index)

r: pointer to R's Index
for each s $\in$ S
    if r.exist(s) then
        res := res $\cup$ (r.search(s) $\times$ s)

- +: Parallel Look Up (Nachschlagen) in Index (z.B. B+ Tree)
- +: falls geeignete Index (z.B. B+ Tree) vorhanden, Range-Query erledigen kann
- -: falls Index vorher nicht vorhanden, aufwendig zu builden
- -: $|S|$ mal Look Up im Index kann auch aufwendig sein.

Wiederholung - Hash Join

Hashtable ht
for each r $\in$ R
    ht.insert(r)
for each s $\in$ S
        if ht.exist(s) then res := res $\cup$ (r.search(s) $\times$ s)

- +: Parallel Look Up (Nachschlagen) möglich, falls die Hashtable ge-built ist
- +: Sequential I/O oder Sequential memory read
- -: Hashfunktion muss geeignete gesucht werden.
- -: nur Gleichheit checken kann (Point Query, kein Range Query)

Wiederholung - Iteratorkonzept

- Iteratorkonzept (auf Englisch: Iterator Model) war Standard Implementation and angewendet.
- Scan (auch als Table Scan genannt), Select, Join (Typen) sind Unterklassen.
- Die Oberklasse hat Interface implementiert: open(), next(), close()
- open(): Operator(z.B. Tuple) vorbereiten
- next(): nächsten Operator(z.B. Tuple) erzeugen
- close(): Operator fertig.(z.B. alle Tuple ausgearbeitet)

Wiederholung - Iteratorkonzept
Table Scan Relation R

next():
for each r $\in$ R
    emit(r)

- Hole nächstes Tupel
- Gib dieses Tupel zurück

Wiederholung - Iteratorkonzept
Nested Loop Join

```
next():
for each t1 ∈ left.next()
    for each t2 in right.next()
        emit(t1 join t2)
```

Wiederholung - Iteratorkonzept
Hash Join

```
next():
for each t1 ∈ left.next()
      buildHashtable(t1)
for each t2 in right.next()
          if (probe(t2)): emit(t1 join t2)
```

Hilfreiche Sachen

- Datenbanksysteme: Eine Einführung von Prof. Kemper + seine Vorlesung Slides
- Prof. Neumann's Vorlesung: Database Systems on Modern CPU Architectures, Query Processing Chap: Algebraic Operators
- 10Min-Level Youtube Crash Course: Prof. Dr. Jens Dittrich
- CMU Database Group Vorlesung Aufnahme: Intro to Databases Systems - Query Execution
- State-of-art Datenbanken Papers
- Der Konzept ist bei allen Vorlesungen gleich, aber Detail nicht unbedingt gleich.

Hausaufgabe 1

- Geben Sie die kanonische Übersetzung der folgenden Anfrage in die relationale Algebra an. Verwenden Sie zur Darstellung des relationalen Algebraausdrucks die Baumdarstellung.

```
select v.VorlNr, v.Titel, p.Name, count(h.MatrNr) as hoerer
from
      Vorlesungen v left outer join
      hoeren h on (v.VorlNr = h.VorlNr),
      Professoren p
where       v.elesenVon = p.PersNr
group by v.VorlNr, v.Titel, p.Name
having count (h.MatrNr) > 3
```

Hausaufgabe 2

Sie verwenden ein Datenbanksystem, welches die folgenden Joinalgorithmen unterstützt:

- (Blockwise-)Nested-Loop-Join
- Index-Join
- Sort-Merge-Join
- Hash-Join

Geben Sie für jeden der vier Algorithmen ein Beispiel an, bei dem dieses Datenbanksystem diesen Algorithmus verwenden würde. Geben Sie dazu für jedes Beispiel ein Schema und eine SQL-Anfrage an. Begründen Sie, warum der gewählte Joinalgorithmus den anderen vorgezogen wird.

Hausaufgabe 3

Siehe Blatt

Hausaufgabe 4

Siehe Blatt


T. Neumann, “Grundlagen: Datenbanken, Kapitel 06”,,


A. Kemper, “Grundlagen: Datenbanken, Kapitel 07”,,


A. Pavlo, “CMU 15-445/645 (FALL 2019) Intro to Datbase Systems, 12 - Query Execution I”,, Adresse: https://15445.courses.cs.cmu.edu/fall2019/slides/12-queryexecution1.pdf/.