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. 09**

### Hausaufgabe 1

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? Bei den Baumknoten müssen die Header nicht berücksichtigt werden, Pointer habe eine Größe von 64 bit.

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 (die werte zwischen 0 und 1000 haben immer 0x00 0x00 als prefix). 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. Während der Baum gebaut wird sind auf der untersten Eben ursprünglich auch Node 4, die aber über Node 16, zu Node 48 zu Node 256 wachsen. 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.

### Hausaufgabe 2

Geben die folgenden Anfragen,

Anfrage 1: select Note from Noten where MatrNr=12345

Anfrage 2: select count(*) from Noten where Note<1.5

Anfrage 3: insert into Noten(MatrNr,Note) values (54321, 3.0)

Anfrage 4: update Noten set Note=1.4 where MatrNr=32154

Anfrage 5: insert into Noten(MatrNr,Note) values (54321, 1.3)

Anfrage 6: update Noten set Note=1.6 where MatrNr=12345

analysieren sie ob die folgenden Historien unter dem MVCC Modell wie in der Vorlesung vorgestellt auftreten können. Transaktion $X$ entspricht dabei Anfrage $X$, also z.B. Tx 1

ist das Ergebnis von Anfrage 1. Jede Historie steht für sich selbst und starten jeweils von einem ursprünglichen Datenzustand. Die Buchstaben innerhalb der Klammer entsprechen dabei jeweils den Tupeln auf die Zugegriffen wird. Wenn in Anfrage 2 z.B. drei Werte das Prädikat Note<1.5 erfüllen, gäbe es entsprechend drei $r(...)$ Einträge auf die jeweiligen Tupel.

Historie 1: $bot_1, r_1(A), bot_3, w_3(B), commit_1, commit_3$

Historie 2: $bot_2, r_2(A), bot_3, w_3(B), r_2(C), commit_2, commit_3$

Historie 3: $bot_2, r_2(A), r_2(B), bot_4, r_4(B), w_4(B), r_2(C), commit_2, commit_4$

Historie 4: $bot_1, r_1(B), bot_6, r_6(B), w_6(B), commit_1, commit_6$

Historie 5: $bot_2, r_2(A), bot_4, r_4(B), w_4(B), r_2(C), commit_4, commit_2$

Historie 6: $bot_1, r_1(B), bot_6, r_6(B), w_6(B), commit_6, commit_1$

Historie 7: $bot_2, r_2(A), bot_5, w_5(D), commit_5, r_2(D), commit_2$

Historie 8: $bot_2, r_2(A), bot_3, w_3(B), r_2(C), commit_3, commit_2$

Historie 9: $bot_2, r_2(A), bot_5, w_5(B), r_2(C), commit_5, commit_2$

Lösung:

Historie 1 Kann so auftreten.

Historie 2 Kann so auftreten.

Historie 3 Kann so auftreten. Bei 2PL wäre diese Historie nicht möglich, da B gesperrt wäre.

Historie 4 Kann so auftreten. Bei 2PL wäre diese Historie nicht möglich, da B gesperrt wäre.

Historie 5 Kann so nicht auftreten. Die Transaktion 2 würde abbrechen.

Historie 6 Kann so nicht auftreten. Die Transaktion 1 würde abbrechen.

Historie 7 Kann so nicht auftreten. Die Transaktion 2 kann nur Werte basierend auf ihrem eigenen Startzeitstempel lesen, daher wäre $r_2(D)$ nicht möglich.

Historie 8 Kann so auftreten. Der eingefügte Wert ist außerhalb des Prädikatsbereichs der Anfrage 2, daher gibt es keinen Konflikt.

Historie 9 Kann so nicht auftreten. Die Schnittmenge zwischen dem Prädikat Note<1.5 von Anfrage 2 und dem abgeleiteten Prädikat Note=1.3 von Anfrage 5 ist nicht leer, daher Konflikt.

### Hausaufgabe 3

Definieren Sie das Prädikat `sg(X,Y)` das für "same generation" steht. Zwei Personen gehören zur selben Generation, wenn Sie mindestens je ein Elternteil haben, das derselben Generation angehört.

Verwenden Sie beispielsweise die folgende Ausprägung einer ElternKind Relation. Das erste Element ist hier das Kind, das Zweite ein Elternteil.

```
parent(c,a).
parent(d,a).
parent(d,b).
parent(e,b).
parent(f,c).
parent(g,c).
parent(h,d).
parent(i,d).
parent(i,e).
parent(f,e).
parent(j,f).
parent(j,h).
parent(k,g).
parent(k,i).
```

a) Definieren Sie das Prädikat in Datalog.

b) Demonstrieren Sie die naive Ausführung des Prädikats.

c) Erläutern Sie das Vorgehen bei der seminaiven Auswertung.

Siehe Übungsbuch