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

Übung zur Vorlesung *Einsatz und Realisierung von Datenbanksystemen* im SoSe17

Maximilian E. Schüle (schuele@in.tum.de)
http://db.in.tum.de/teaching/ss17/impldb/

Blatt Nr. 10

Hausaufgabe 1

Schätzen sie die Anzahl der Cache Misses die entstehen, wenn man 1001 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.

Hausaufgabe 2

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

Hausaufgabe 3

Die Bayernwaldklinik

Name	verfügbar	Versionsvektor
House	ja	-
Green	ja	-
Brinkmann	ja	-

Abbildung 1: Hauptspeicher Column-Store

TID	Startzeit	Commitzeit	Aktion
Ta	$T0$	-	$\sum$
Tb	$T2$	-	$(Green)--$
Tc	$T3$	-	$\sum$
Td	$T5$	-	$\sum$

Abbildung 2: Transaktionen (bereits committete gekennzeichnet durch eine Commitzeit)

Beschäftigen wir uns mit *Multi-Version Concurrency Control* am Beispiel unserer verfügbaren Ärzte („Doctors on call/duty“), in dem wir sicherstellen wollen, dass immer mindestens ein Arzt verfügbar ist.

Uns stehen drei Operationen zur Verfügung, $\sum$ zählt alle verfügbaren Ärzte, $(X)++$ ändert Xs Status in verfügbar, $(X)--$ zählt alle verfügbaren Ärzte und ändert Xs Status auf nicht verfügbar, wenn mindestens ein Arzt noch anwesend ist.

1. Welche Bedingungen gelten für die Zeitstempel?
2. Green möchte zum Zeitpunkt $T2$ seinen Feierabend antreten. Vervollständigen Sie Tabelle 2 und legen Sie einen geeigneten Undo-Puffer (Zeitstempel, Attribut, Undo-Image) an. Wann muss Tb committen, damit Ta und Td erfolgreich committen? Was lesen Ta und Tb?
3. Brinkmann und House wollen zeitgleich den Feierabend antreten. House startet bei $T8$, Brinkmann bei $T9$. Wer darf gehen? Wie sorgt *Precision Locking* dafür, dass nur ein Arzt das Krankenhaus verlässt? Vervollständigen Sie die Einträge.