CS411 Database Systems

Fall 2009

HW#5

Due: 3:15pm CST, December 1, 2009

Note: Print your name and NetID in the upper right corner of every page of your submission. Hand in your stapled homework to Donna Coleman in 2106 SC. In case Donna is not in office, slide your homework under the door.

To grade homeworks faster, the homework is partitioned into two parts. **Please, submit each part separately.** For each part, make sure to write down your name and NetID. Handwritten submissions will be graded but they will take longer to grade. For clarity, machine formatted text is preferable: Expect to lose points if your handwritten answer is unclear or misread by the grader.

This homework is partitioned into two parts as follows:

- Part 1: Problem 1 - Problem 3
- Part 2: Problem 4 - Problem 6

Part 1

Problem 1 Undo logging and redo logging (30 points, 6 points each)

1. Consider the following sequence of log records:

<START S>;
<S, A, 60>;
<COMMIT S>;
<START T>;
<T, A, 10>;
<START U>;
<U, B, 20>;
<T, C, 30>;
<START V>;
<U, D, 40>;
<V, F, 70>;
<COMMIT U>;
<T, E, 50>;
<COMMIT T>;
<V, B 80>;
<COMMIT V>;

Note that we assume undo logging for questions (i), (ii) and (iii), and redo logging for questions (iv) and (v).

(i) First, we consider undo logging. Suppose that we begin a nonquiescent checkpoint immediately after record <T, A, 10> has been written (in memory). Where a <END CKPT> record should be written?

<END CKPT> should be written immediately after <COMMIT T>.

(ii) In (i), suppose that a crash occurs just after record <V, B, 80> has been written. Give a sequence of records to undo. The sequence must specify in which order the system undo the records.

Given that the crash occurs after <END CKPT>, we know that any incomplete transactions started after the matching <START CKPT> unless they are committed. The transactions started after start-checkpoint record are U and V, but <COMMIT U> shows that transaction U is completed before the crash. This means only the transaction V needs to be undone, in order:

<V, B 80>;
<V, F, 70>;

(iii) In (i), suppose that a crash occurs just after record <V, F, 70> has been written. Give a sequence of records to undo.

The crash occurs before <END CKPT>. Therefore, if any transaction that is either in the list of active transactions in the <START CKPT> record or started after that reccord is incomplete, we must undo actions done by that transaction. Transaction T is the only active transaction when the <START CKPT> record is placed, and transactions U and V start after the checkpoint. Since none of the transactions commits before the crash, the records that must be undone are, in order:

<V, F, 70>;
<U, D, 40>;
<T, C, 30>;
<U, B, 20>;
<T, A, 10>;

(iv) Next, we consider redo logging. Suppose that we begin a nonquiescent checkpoint immediately after record <T, E, 50> has been written (in memory). At what points could a <END CKPT> record should be written? Give all possible points for the <END CKPT> record.

The checkpoint could complete when the changes made by transactions committed before the start checkpoint record are written to the log. Transactions S and U are complete before the <START CKPT> record. Therefore, after the system copies the blocks A, B, and D, which are modified by transactions S and U, to the disk, we can place a <END CKPT> record. Since those copy operations (e.g., OUTPUT(A)) could occur at any point after the <START CKPT> record, the <END CKPT> record could also appear anywhere after <START CKPT>.

(v) In (iv), suppose that a crash occurs at the end. Give a sequence of records to redo. The sequence must specify in which order the system should redo the records. (You answer should be same regardless of the position of the <END CKPT> record.)

Since transaction U and S are committed before <START CKPT>, it is not necessary to redo actions of those committed transactions. Transactions T and V have committed after the start checkpoint, and thus we need to redo all actions done by transactions T and V, in order:

<T, A, 10>;
<T, C, 30>;
<V, F, 70>;
<T, E, 50>;
<V, B 80>;

Problem 2 Undo/redo logging (16 points, 8 points each)

Consider the following sequence of log records for undo/redo logging:

<START S>;
<S, A, 60, 61>;
<COMMIT S>;
<START T>;
<T, A, 61, 62>;
<START U>;
<U, B, 20, 21>;
<T, C, 30, 31>;
<START V>;
<U, D, 40, 41>;
<V, F, 70, 71>;
<COMMIT U>;
<T, E, 50, 51>;
<COMMIT T>;
<V, B 21, 22>;
<COMMIT V>;

(i) Suppose that we begin a nonquiescent checkpoint immediately after record <T, E, 50, 51> has been written (in memory). At what points could the <END CKPT> record be written? Give all the possible points.

<END CKPT> chould be placed anywhere after the start checkpoint flag. In the case of undo/redo logging, we may place a <END CKPT> record only after all the changes prior to the <START CKPT> record are written to the disk. Thus, blocks A, B, C, D, E, F must be copied to the disk before the <END CKPT> record.

(ii) Suppose that we have <START CKPT (T,U,V)> record immediately after record <U, D, 40, 41> and <END CKPT> record immediately after <T, E, 50, 51> respectively in the log. When a crash occurs at the end, how should the system undo and redo records in the log file? Give both a sequence of records to undo and a sequence of records to redo.

In this case, we see the <END CKPT> record on the log. With undo/redo logging, we only need to look back until the matching <START CKPT> record, and redo the transactions identified in it - T, U and V in this case. Note however we do not need to look prior to <START CKPT> record for all three transactions (i.e. all three transactions start before <START CKPT> record) because we know that any changes made by T, U and V before the start of the checkpoint were flushed to disk during the checkpoint. Records that need to be redone are, in order:

<V, F, 70, 71>;
<T, E, 50, 51>;
<V, B 21, 22>;

There is no record to undo because all transactions were completed (i.e. committed) before the crash occurred.

Problem 3 Conflict Serializability (24 points, 8 points each)

For each of the following schedule:

(a) $w_3(A)$; $r_1(A)$; $w_1(B)$; $r_2(B)$; $w_2(C)$; $r_3(C)$;

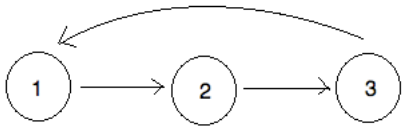
(b) $r_1(A)$; $r_2(A)$; $w_1(B)$; $w_2(B)$; $r_1(B)$; $r_2(B)$; $w_2(C)$; $w_1(D)$;

(c) $r_1(A)$; $r_2(A)$; $r_1(B)$; $r_2(B)$; $r_3(A)$; $r_4(B)$; $w_1(A)$; $w_2(B)$;

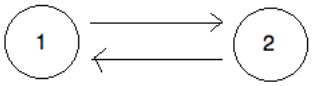
Answer the following questions:

(i) What is the precedence graph for the schedule?

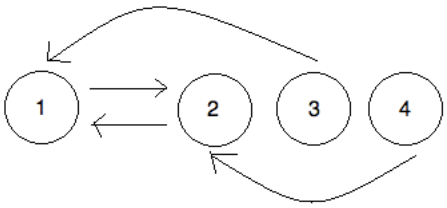
(a)



(b)



(c)



(ii) Is the schedule conflict-serializable? If so, what are all the equivalent serial schedules?
(a) Not conflict-serializable
(b) Not conflict-serializable
(c) Not conflict-serializable

(iii) Are there any serial schedules that must be equivalent (regardless of what the transactions do to the data), but are not conflict-equivalent?
(a) Serial schedule (3, 1, 2) is equivalent. We can move $r_3(C)$ in the beginning because this read action does not change the state of main memory buffer.
(b) None
(c) None

Part 2

Problem 4 Information Retrieval (15 points)

(1) According to TF-IDF weighting, what kind of terms would have high weights?

those terms with high frequency in documents (high TF) and low frequency in the collection (high IDF).

(2) Would removing a few words with low IDF values from an inverted index help reduce the size of the index substantially?

Low IDF means the word is frequent in the collection, so removing them would help.

(3) Suppose a query has 4 relevant documents and an IR system retrieves 20 documents; out of which 3 are relevant. What is the Precision and Recall of this system for this query?

precision $= \frac{3}{20}$
Recall: $\frac{3}{4}$

(4) What would be the PageRank results if there is a directional link between every two documents?

it would be the same for all documents.

Problem 5 Data Integration (15 points, 5 points each)

(1) In the lecture, we discussed data integration is difficult because of the heterogeneity problem. Explain the schema issue in this heterogeneity problem, using Event Search as an example.

Assuming all event sources are relational databases, their schemas are different. One source may have Events(title, where, description, start, end). Another source may have Events(title, location, cost, start, end). In data integration, we need to know how to integrate these two events schemas.

(2) In the lecture, we discussed that wrappers have a scalability problem. Explain this scalability problem.

Wrappers are data source specific. As the number of data sources increases, the number of wrappers needed to build and maintain increases.

(3) Suppose we want to build a news mashup application (e.g. mashup in igoogle). Which data integration approach should we use and why?

We should use the virtual integration approach because news data are time sensitive.

Problem 6 Survey for Newsletters Application (5 extra points) If you attended the guest lecture by Maryam Karimzadehgan on November 10th, please get and fill in a survey form at

http://www.cs.uiuc.edu/class/fa09/cs411/assignments.html

and email it to her at mkarimz2@uiuc.edu with the title "CS411-survey".