

CS 386/586 Winter 2009 Exercise 7

Work through the exercises

Undergraduate students, please turn in your paper (with papers from all other exercises) at the end of the term.

(This is exercise 16.3 from the textbook by Ramakrishnan and Gehrke.)

1. Consider a database with objects X and Y. Assume that there are two transactions T1 and T2. T1 reads objects X and Y and then writes object X. T2 reads objects X and Y and then writes objects X and Y.

There are several correct answers to all three of these questions. We provide one answer for each.

- a. Write a schedule for T1 and T2 that results in a Write-Read conflict.

Show how Strict 2PL would prevent this schedule from happening. That is, show the point in the schedule where Strict 2PL would NOT allow a lock to be acquired and force a transaction to wait.

The following schedule results in a write-read conflict:

T2: R(X), T2:R(Y), T2:W(X), T1:R(X) ...

T1: R(X) is a dirty read here because it follows T2:W(X).

Using Strict 2PL, T1 could not get a shared lock on X because T2 would be holding an exclusive lock on X. Thus, T1 would have to wait until T2 was finished.

- b. Write a schedule for T1 and T2 that results in a Read-Write conflict.

Show how Strict 2PL would prevent this schedule from happening. That is, show the point in the schedule where Strict 2PL would NOT allow a lock to be acquired and force a transaction to wait.

The following schedule results in a read-write conflict:

T2:R(X), T2:R(Y), T1:R(X), T1:R(Y), T1:W(X) ...

Now, T2 will get an unrepeatable read on X because T1 has written X.

Using Strict 2PL, T1 could not get an exclusive lock on X because T2 would already be holding a shared or exclusive lock on X (depending on whether or not the programmer for T2 requested an exclusive lock at the time X was read – knowing that they were going to update it a bit later). So T1 would have to wait until T2 was finished.

- c. Write a schedule for T1 and T2 that results in a Write-Write conflict.

Show how Strict 2PL would prevent this schedule from happening. That is, show the point in the schedule where Strict 2PL would NOT allow a lock to be acquired and force a transaction to wait.

The following schedule results in a write-write conflict:

T2:R(X), T2:R(Y), T1:R(X), T1:R(Y), T1:W(X), T2:W(X) ...

Now, T2 has overwritten uncommitted data when it issues T2:W(X).

Strict 2PL would prevent T1 from getting an exclusive lock on X because T2 would already have a shared lock (or an exclusive lock, if the programmer has asked for it at the time of the read) on X. So T1 would have to wait until T2 was finished.

2. Given that a log record contains the following information: *Transaction-id*, *Page-id*, *offset*, and *length* of record that was involved in this update old-data – this is what we call the “before image” of the record new-data – this is what we call the “after image” of the record

Answer the following questions.

- a. What does a log record look like if a transaction has inserted a new record? That is, what is the content of each field?

The “before image” of the record is <null>. This means that insert operations can be identified simply by the before image being null.

- b. What does a log record look like if a transaction has deleted a record? That is, what is the content of each field?

The “after image” of the record is <null>. In a similar way, this means that delete operations can be identified simply by the after image being null.

- c. What does a log record look like if a transaction has updated a record? That is what is the content of each field?

Both the “before image” and the “after image” has a copy of the relevant data. (There is no null field in a log record that represents an update.)

3. During the UNDO phase, suppose that transaction with id of 128 is being redone. What exactly needs to happen when a log record for transaction 128 is encountered in the log file:

For an insert operation?

Use the page-id to read in the appropriate page. Use the offset and length to locate the appropriate record. Delete that record. Rewrite the page.

For a delete operation?

Use the page-id to read in the appropriate page. Use the offset and length to locate the appropriate record. Insert the record that had been deleted. (Or if deletes were done by simply marking the record as being deleted, reset that marker so that the record is no longer deleted.) Rewrite the page.

For an update operation?

Use the page-id to read in the appropriate page. Use the offset and length to locate the appropriate record. Replace the data on the page with the "before image" from the log record. Rewrite the page.