# PCSD Assignment 3

Ronnie Elken Lindsgaard, Jens Fredsskov, Henrik Bendt

December 12, 2013

## Question 1: Recovery Concepts

### 1

There is no need to implement a scheme for redo nor undo. As Force forces writes to disk for every write to memory, there is no need to ever redo an operation, as it is always commited. With No Steal means that dirty (uncommited) operations are never taken over by other operations, we never have to undo the opereration already written to disk.

### 2

Non-volatile storage simply means, that when we cut off power, stored data is not lost (as opposed to volatile memory). This data can however be lost due to damages to disk like water damage, electrical overcharging (due to lightning strike), magnetic damage, mechanical damage (e.g. to write/read arm), error due to worn out hardware, etc.

Stable storage means that data stored is *never* lost. This is a somewhat lose definition, as what does it mean to never lose stored data. If we assume that armageddon or a nuclear war is not around the corner, stable storage means that you do not lose data stored because of the above mentioned types of disk loss. This could be ensured by distributing data over a range of disks placed around the world (like clouding) in secured places and with lots of redundancy.

### 3

1. The log record for an update must be forced to disk before a data page for it gets written to disk
2. all log records for a transaction must be written to disk before a commit.

Why do we need the above? Lets assume data for an update/commit is written to disk, but the log is not forced to the stable storage.

1. Then a crash occures and the log and main memory is lost. As the log was not written, we have no way of knowing that the write we just did to disk is not rightfully part of the comitted data and we cannot undo the incomplete actions. Thus we must ensure log is written before an update.
2. Then a crash occures in the middle of the commit and the log and main memory is lost. As the log was not written, we have no way of knowing what data on disk is dirty and also we must never forget comitted data. Thus we must ensure the log is written before a commit.

These two protocols are sufficient, as the log explains all commits to the storage, so in principle we could do the log all over and reach the correct state of storage.

## Question 2: ARIES

| XID | Status | lastLSN |
|---|---|---|
| T1 | Running | 4 |
| T2 | Running | 9 |

Table 1: Xact table

| PID | recLSN |
|---|---|
| P2 | 3 |
| P1 | 4 |
| P5 | 8 |
| P3 | 9 |

Table 2: Dirty Page Table

1. The transaction table and dirty page table can be seen in tables 1 and 2 respectively. `T3` is removed from the transaction table as an `end` was found.
2. The winner transactions are the ones completed, and the loser transactions are the ones still in the transaction table. Hence $winners = \{T3\}$ and $losers = \{T1, T2\}$.
3. The redo phase starts at the log record where $LSN = 3$ as this is the record following the last checkpoint. The undo phase ends at the same log record, as `TRAN_ID` is one of the loser transactions.
4. The log records that may cause page writes are records with $LSN \in \{3, 4, 8, 9\}$ as all pages in the table is written to by the loser transactions.
5. The log records undone are the records with $LSN \in \{9, 8, 5, 4, 3\}$. The ToUndo set is initialised with `LSN`s $\{9, 4\}$ from the transaction table and is expanded by checking the `LAST_LSN` value of the record.
6. The contents of the log is added with the lines visible in figure 1. The log semantics are changed such that for CLR records, the column `LAST_LSN` points to the `LSN` of the record that was undone.

```
LSN   LAST_LSN  TRAN_ID   TYPE    PAGE_ID
---   --------  -------   ----    -------
11    9         T2        CLR     P3
12    8         T2        CLR     P5
13    5         T2        CLR     P5
14    4         T1        CLR     P1
15    3         T1        CLR     P2
```

Figure 1: Continuation of the log after crash recovery