The University of British Columbia
Department of Computer Science

CPSC 404: Assignment #1
Topics: Disks, Records, Tables, Relational Algebra, SQL, Joins, Buffer Pools

Last Updated on Jan. 18, 2013 @ 17:30

History of Changes (all minor/clarifications):
Jan. 18 @ 17:30: Part 1, Q7: Your choice of contiguous or worst-case placement (just state your assumption). Part 2, Q1: yyyymmdd, not yyymmdd. Part 3, Q3: Re-write the algorithm shown in class.
Jan. 18 @ 13:00: Part 1, Q1: 7.5 ms includes set-up time, but it's OK if you added 1 ms (we'll mark both correct); Part 1, Q8: Assume 25,000 contiguous pages.
Jan. 16 @ 14:00: Note that the due date is *not* on a class day. The hand-in box is box #10 in room ICCS X235.
Jan. 15 @ 21:00: Created and posted

Due Date: Thursday, January 24, 2013 at 11:00 AM

Where to Hand it in: In the metal hand-in box #10 in room ICCS X235 where the other new, metal hand-in boxes are located. This is also the place to hand in late assignments. Note that we are *no longer using* the old wooden hand-in boxes in the ICCS basement.

Late Penalty: 20% (of the assignment's maximum value) per school day, where 11:00 AM to (next day's) 11:00 AM is the extent of one school day for this assignment. For example, if you submit your late assignment to the hand-in box by Friday, January 25 @ 11:00, then that's a 20% deduction (so, if you would have gotten 90% if you were on-time, you'll now get 90-20=70%). Weekends count as one day.

The assignment you submit should be done on your own, not in pairs or groups, and not copied from any other source. You can discuss questions with each other as long as you don't copy answers. Please refer to the UBC CPSC rules on plagiarism if you have any questions.

NOTE: **For all of the calculation-based questions in this assignment, show your work**; otherwise, you may get no marks. In case of ambiguity, write down any (reasonable) assumptions that you make.

Part 1: Disk Geometry, Records, Pages, and Tables

Consider the following specifications for a disk drive: sector size of 512 bytes, 18,000 tracks per surface, 8,192 sectors per track, 5 platters (each with 2 usable surfaces) per cylinder, spinning at 16,000 rpm. We'll use a 4K block/page size (i.e., 4,096 bytes/page). The average seek time is 7.5 ms for each seek, and this includes the setup time. (For this assignment, we'll use the simplifying assumption that all seeks take 7.5 ms, regardless of the number of cylinders moved.)

Suppose we are trying to store **Personnel** records containing these attributes/columns: id (4 bytes), name (30 bytes), address (50 bytes), phoneNumber (string of 12 bytes), birthDate (integer, 4 bytes), and startDate (integer, 4 bytes). Let us assume that no record can span 2 pages (i.e., you

can't have a part of a record on each of 2 pages). Assume the buffer pool currently has no **Personnel** pages in it.

1. How long does it take, in the worst-case, to read a specific page X on this disk drive, when you're currently at some random point (elsewhere) on the disk drive?

2. Repeat Question 1, but for the best-case time.

3. How many **Personnel** records can be stored on one track?

4. If we reserve table space for 30 cylinders' worth of **Personnel** data, how many records can we store?

5. If we expect 10 cylinders' worth of **Personnel** records today, and expect a 20% (compounded) annual increase per year, how many cylinders should we reserve in advance (today) to account for the next 5 years (i.e., this year, plus the next 4)?

6. What is the transfer rate (in megabytes per second) for transferring (reading) one entire track? Let us assume that we can start reading immediately, rather than waiting for the "start" of the track to come around (because we're going to read all of those pages anyway, and we can rearrange the ordering in memory). Assume that you're on the correct cylinder, and there is no rotational delay.

7. Suppose we had to read 1,000 **Personnel** records, each of which appeared on a different page (your choice of contiguous pages or worst-case page placement—just specify your assumption), and we were reading the records, one page at a time. How long would it take to read these 1,000 Personnel records from disk?

8. If there is no index on the address field, how long (in milliseconds or seconds) would it take to bring the requested data pages into the buffer pool to allow us to answer the following SQL query, assuming the head is on a non-**Personnel** cylinder (i.e., somewhere far away)? Assume that there is no order to the records, and that there are 25,000 pages stored contiguously.

```
SELECT     count(*)
FROM       Personnel
WHERE      address contains "Richmond";
```

Question 2: Relational Algebra (RA) and Joins

This question is meant to refresh your memory on relational algebra, and get practice with joins. Consider the **Sailors-Boats-Reserves** DB from the textbook, with one additional relation (**Costs**) added. Relevant information from the schemas of these 4 tables is shown below. The primary key for each relation is underlined.

```
Sailors    (SID, SName, SRating, SAge)
Boats      (BID, BName, Color)
Reserves   (SID, BID, date)
Costs      (BID, Year, DailyRate, HourlyRate)
```

Your answer does not have to be one big expression. Use parentheses to make your evaluation order clear. Try to make your answer reasonably efficient (e.g., avoid unnecessary joins). Check out the textbook for a bunch of relational algebra examples. Remember that you can use the Greek *rho* symbol (rename operator) to rename expressions, and therefore break a query into parts.

1. List the names, ages, and ratings of all sailors who have reserved a boat called "Titanic". Assume that dates are integers in the format yyyymmdd.

2. List the names of the sailors who reserved a blue boat in July or August of 2012, and whose rental rate (for that same boat) was more than $50 per hour or more than $300 per day. In addition to the names of the sailors, include the name of the boat.

Question 3: Buffer Pools—Extended Clock, Clock, and LRU Algorithms

1. Apply the following reference string to an initially empty buffer pool, using the Extended Clock Algorithm (the one with both a reference bit (0,1) and a dirty bit (0, 0*,1)).

   `100, 200w, 200, 300w, 25, 50w, 100, 25w`
   (where "`w`" means we are updating the page in memory)

   Assume that there are 3 page frames available in memory. Construct a table to show which pages are present in the buffer pool as each request is serviced (just like in class), and determine the number of page faults that occur, and when the writes occur. Remember that all frames are initially empty, so your first reference to a given page will always result in a page fault.

2. Apply the same reference string as in Question 1 to an initially empty buffer pool, using the LRU Algorithm (for which we don't pay attention to dirty bits).

   Assume that there are 3 page frames available in memory. Construct a table to show which pages are present in the buffer pool as each request is serviced, and determine the number of page faults that occur. Remember that all frames are initially empty, so your first reference to a given page will always result in a page fault.

3. Re-write the Clock Algorithm (the one without a dirty bit, from class) so that it uses a reference bit/field (RB) that takes on values of more than just 0 or 1. In other words, assume that the RB can take on values of 0, 1, 2, …, *k* to account for *k* active, but different transactions. If we have trouble finding a victim on the first pass, note that we'd cycle through, as needed—and eventually get to 0 to find a valid victim. Make reasonable assumptions.