CS150A Homework1 Solutions

I. MULTI-RELATION QUERIES [40 points]

Consider the following relations representing student information at UIUC:

Mentorship(mentee_sid, mentor_sid)

Study(sid, credits)

Enrollment(did, sid)

Student(sid, street , city)

- A tuple (M1, M2) inMentorship specifies that M2 is a mentor of another student M1.
- A tuple (S, C) in Study specifies that the student S has taken C credits.
- A tuple in Enrollment (D, S) specifies that student S is enrolled in department D.
- A (ST, S, C) in Student specifies that student ST lives on street S in city C.

1. Find all students who live in the same city and on the same street as their mentor.

Solution:

```
SELECT m1.mentee_sid
FROM Mentorship as m1, Student as s1 , Student as s2
WHERE m1.mentee_sid = s1 . sid
AND m1.mentor_sid = s2 . sid
AND s1 . s t r e e t = s2 . s t r e e t
AND s1 . c i t y = s2 . c i t y ;
```

2. Find all students(i.e., distinct sid) who have taken more credits than the average credits of all of the students of their department.

Solution:

```
SELECT s1 . sid
FROM Study as s1 , Enrollment as e1
WHERE e1 . sid = s1 . sid
AND s1 . credi t s > ( SELECT AVG( s2 . credi t s )
FROM Study as s2 , Enrollment as e2
WHERE s2 . sid = e2 . sid
AND e1 . did = e2 . did
) ;
```

II. **Storage: Disk, Files, Buffers** [30 points]

1. Write down the letters of true statements *in alphabetical order.* (If none are true, write ∅.)
   - A. When querying for a 16 byte record, exactly 16 bytes of data is read from disk. False, a page's worth of data is always read from disk.
   - B. Writing to an SSD drive is more costly than reading from an SSD drive. True, writes to an SSD can involve reorganization to avoid uneven wear and tear.
   - C. In a heap file, all pages must be filled to capacity except the last page. False, there is no such requirement. With clustered indices, it is common to keep heap file pages 2/3 full to accommodate inserts.
   - D. If the file size is smaller than the number of buffer frames, a sequential scan of the file using either MRU or LRU (starting with an empty buffer pool) will have the same hit rate. True, our eviction policy doesn't matter because the file fits in memory.
   - E. Assuming integers take 4 bytes and pointers take 4 bytes, a slot directory that is 256 bytes can address 64 records in a page. False, as shown in lecture, an entry in slot directory is 8 bytes, because a single entry consists of both a pointer and an integer (length).
   - F. In a page containing fixed-length records with no nullable fields, the size of the bitmap never changes. True, the size of the records is fixed, so the number we can fit on a page is fixed.

2. Assume we have 4 empty buffer frames and the following access pattern, in which pages are immediately unpinned.

T A M E T E A M M A T E M E A T L I D

Use the replacement policy listed, and list the four pages in the buffer pool at the end, *in alphabetical order.* Hint: you don't need to draw a big chart for every access — look for patterns.

1) MRU. ADEM
2) LRU. DILT
3) Clock. (*Assume the clock hand starts on the first buffer and does not move unless a page needs to be replaced.*). DEIL

III. **Indexes and B+ Trees** [30 points]

Note: for B+ tree page splits with an odd number of items, assume that the majority of the items is placed on the right-hand page after the split.

1. Alphabetically, write down the letters of statements that apply (or write ∅.)
   A. All internal keys in a B+ tree also appear in its leaf nodes. False, a leaf node which has been copied up can be deleted.
   B. The height of a B+ tree increases whenever any node splits. False, height increases when root node splits.
   C. An ISAM index is similar to a B+ tree, but does not allow for insertion of new values. False.
   D. The column(s) we select for our index key must have a unique value for every row in the table. False.

2. Assume we are trying to construct a B+ Tree of order 2 (max fanout of 5). Each leaf node can hold up to 4 entries. We insert a total of 16 unique keys via bulk loading, with a fill factor of ¾.
   1) How many leaf nodes will there be? 6
   2) How many internal (non-leaf) nodes will there be? 3
   3) How many internal (non-leaf) nodes do we traverse to do an equality search? 2