CS186 Vitamin #3

Have a nice weekend!

File formats

Assume that each page in our system can hold 64 KB (1 KB = 1024 bytes), integers are 32-bits wide, and bytes are 8-bits wide.

Consider the following relation:

```
CREATE TABLE Submissions (
  record_id integer UNIQUE,
  assignment_id integer,
  student_id integer,
  time_submitted integer,
  grade_received byte,

  PRIMARY KEY(assignment_id, student_id)
);
```

Assume the column record_id corresponds to the row's actual record ID.

1. **Q1: How large (in bytes) is a record?**

2. **Q2: Suppose we begin each page with a 24-byte header plus a bitmap. At most, how many records can fit in an unpacked page?**

We add two variable-length fields to our table schema. Now our table looks like this:

```
CREATE TABLE Submissions (
  record_id integer UNIQUE,
  assignment_id integer,
  student_id integer,
  time_submitted integer,
  grade_received byte,

  comment text,
  regrade_request text,

  PRIMARY KEY(assignment_id, student_id)
);
```

We decide to use slotted pages to store the variable length records. Each page begins with a 24-byte header plus a slot directory. (Assume this header contains information such as the number of

valid records in the page.) Each pointer inside the slot directory consumes 20 bits/record, while the record header storing field offsets is 32 bits wide.

3. **Q3: What is the maximum number of records that can fit in our slotted pages?**

4. **Q4: We decide to squash the two text fields together into one field using a semicolon separator character (;), which allows us to shrink the record header from 32 bits to 16 bits at the cost of 8 bits (for the semicolon). For example, the columns ("Submitted late", "Dog ate my homework") get compressed into "Submitted late;Dog ate my homework". Which of the following are true with this new scheme?**
   Check all that apply.

   - [ ] Professor Gonzales cannot enter the comment "Fantastic work; good job!"
   - [ ] Fewer records will fit in a page
   - [ ] It is possible for the query "SELECT grade_received FROM Submissions" to finish faster

Indices and file layouts

Suppose we have an alternative 2 unclustered index on (assignment_id, student_id) with a depth of 3 (one must traverse 3 index pages to reach any leaf page). Here's the schema:

```
CREATE TABLE Submissions (
  record_id integer UNIQUE,
  assignment_id integer,
  student_id integer,
  time_submitted integer,
  grade_received byte,

  comment text,
  regrade_request text,

  PRIMARY KEY(assignment_id, student_id)
);

CREATE INDEX SubmissionLookupIndex
ON Submissions (assignment_id, student_id);
```

Assume the table takes up 12 MB on disk (1 MB = 1024 KB). (This includes extra space allocated for future insertions.)

5. **Q5: We want to scan all the records in Submissions. How many I/Os will this operation take?**

6. **Q6: UPDATE Students SET grade_received=85 WHERE assignment_id=20 AND student_id=12345: How many I/Os will this operation take?**

7. **Q7: In the best case, how many I/Os does it take to perform an equality search on grade_received?**

8. **Q8: We want to speed up the process of looking up students' grades by student_id, so we will add an index to our current schema. Which of the following indices will help us the most if each student submits many assignments?**
   Select the option which doesn't require any additional special assumptions about the distribution of our data.
   Mark only one oval.

   - ◯ Add an unclustered index on the key (record_id, student_id)
   - ◯ Add a clustered index on (student_id, time_submitted)
   - ◯ Add a clustered index on grade_received
   - ◯ Add an unclustered index on student_id