CSE-4411M

Quiz #1

1. (15 points) **Join Enumeration.** *Down for the count.* [Short Answer]

   a. (3 points) Name two advantages of left-linear join trees compared with join trees generally.

   b. (3 points) Name a disadvantage of left-linear join trees when many joins are involved.

Consider a query with five tables, **A**, **B**, **C**, **D**, and **E**, such that there is a join condition between each pair.

c. (3 points) How many left-linear join trees are possible?

d. (3 points) At least how many plans involving three tables are carried forward from stage 3 to stage 4 of System R's join enumeration algorithm?

e. (3 points) At least how many plans will System R have enumerated before choosing the final one?

2. (10 points) **Query Planning I.** *Sign up!* [Exercise]

Schema:

Student(id, name, major)
Enrol(id, course#, section, term, grade)
FK (id) refs **Student**
FK (course#, section, term) refs **Class**
Class(course#, section, term, instructor, room, time)

Statistics:

- **Student**: 100,000 records on 2,000 pages
  - major: 100 distinct values
- **Enrol**: 4,000,000 records on 40,000 pages
  - course#: 1000, ..., 4999 (so 4000 values)
- **Class**: 200,000 records on 6,000 pages
  - instructor: 8,000 distinct values

Indexes:

- **Student**:
  - hash index on id (linear hash, 200 data entries per page)
- **Enrol**:
  - clustered tree index on id, course#, section, term (50 data entries per page)
  - unclustered tree index on course#, section, term, id (50 data entries per page)
- **Class**:
  - clustered tree index on course#, section, term (60 data entries per page)
  - unclustered tree index on instructor# (200 data entries per page)

All indexes are of alternative #2. For each tree index, the index pages are 3 deep.

Query:

```
select name, instructor, C.term
    from Student S, Enrol E, Class C
    where S.id = E.id
      and E.course# = C.course# and E.section = C.section
      and E.term = C.term
      and instructor = 'Dogfurry';
```

a. (3 points) How many records should the query produce?

b. (12 points) Devise a good query plan for the query. Show the query tree, *fully* annotated with the chosen algorithms and access paths.

You have an allocation of 20 buffer-pool frames.

Estimate the cost of your plan. For full credit, you should have a plan that costs less than 2,000 I/O's.

3. (15 points) **Query Planning II.** *Of course a course is par for the course.* [EXERCISE]

Schema:

Student(sid, sname, startdate, major, advisor)
FK (advisor) refs **Prof** (pid)
Class(cid, dept, number, section, term, year, room, time, pid, ta)
FK (pid) refs **Prof**
FK (ta) refs **Student** (sid)
Enrol(sid, cid, date, grade)
FK (sid) refs **Student**
FK (cid) refs **Class**
Prof(pid, pname, pdept, office)

Assume no attribute is nullable. The attribute pid in **Class** refers to the the professor / instructor for the class. The attribute ta in **Class** refers to the teaching assistant for the class. The attribute advisor in **Student** refers to the student's academic advisor.

Statistics:

- **Student**: 50,000 records on 1,000 pages
  - advisor: 2,500 distinct values
- **Enrol**: 2,000,000 records on 20,000 pages
  - sid: 50,000 distinct values
  - cid: 80,000 distinct values
- **Class**: 80,000 records on 1,600 pages
  - pid: 4,000 distinct values
  - ta: 5,000 distinct values
- **Prof**: 4,000 records on 40 pages

Indexes:

- **Student**:
  - clustered tree index on sid (200 data entries per page)
- **Enrol**:
  - clustered tree index on cid, sid (167 data entries per page)
  - unclustered tree index on sid, cid (167 data entries per page)
- **Class**:
  - clustered tree index on cid (200 data entries per page)
- **Prof**:
  - clustered tree index on pid (200 data entries per page)

All indexes are of alternative #2. For each tree index, the index pages are 3 deep, except for the index on **Prof**.pid which is 2 deep.

```
select sid, sname, dept, number, section, term, year, pid
    from Student S, Enrol E, Class C
    where S.sid = E.sid and E.cid = C.cid
      and S.advisor = C.pid;
```

a. (2 points) Estimate the number of rows the query returns.

b. (8 points) Devise the best query plan for the query. Show the query tree, *fully* annotated with the chosen algorithms and access paths.

Assume you have an allocation of 50 buffer-pool frames.

Estimate the cost of your plan.

c. (5 points) Name an additional index that would allow a less expensive query plan than in 3b, and sketch briefly that query plan using the index.

4. (5 points) **Index Usage.** *The DBA playoffs.* [analysis]

   Consider the following schema.

   Employee(e#, name, salary, d#)
   FK (d#) refs **Department**
   Department(d#, name, location, budget)

   It is important that the following queries be fast to be evaluated.

   A. Find the location where a user-specified employee works.

   B. List all the departments such that the sum of the employees' salaries who belong to the department exceeds the department's budget.

What indexes would you make to benefit queries **A** and **B**? Indicate which are clustered and their types (B+ tree, hash).

Briefy explain how they would be useful.