CS 582 — Database Management System II — Spring 2007

Midterm

Name:

Signature:

Row/Seat:

Four questions on 5 pages (2–6). 100 points for 30%. Please read the question!

1. (25 points) Given the following database schema about computer manufacturers, their products, and how many items they have sold.

- `Product(maker, model)`
- `PC(model, speed, ram, hd, cd, unitprice, qtysold)`
- `Laptop(model, speed, ram, hd, screen, unitprice, qtysold)`

We assume that model number for each type of computer/laptiop is unique and hence can be used as a key for every relations. `unitprice` is the price sold for each unit and `qtysold` is the number of units sold. The unit of the hard disk and ram is in Kbytes (e.g., 1000 stands for 1000Kbytes). In what follows, a computer is understood as either a laptop or a PC. Write relational algebra expression for the following queries:

- List all computer models whose hard disk size is at least 4 Megabytes.

$$\pi_{model}(\sigma_{hd \geq 4000}(PC)) \cup \pi_{model}(\sigma_{hd \geq 4000}(Laptop))$$

- List all makers (or manufactures) who produce at least two laptop models.

$$R_1 = \pi_{maker,model}(Product \bowtie Laptop)$$

$$\pi_{maker}(\sigma_{model \neq model1}(R_1 \bowtie R_1[maker, model1]))$$

Write SQL statement for the following queries:

- List the total revenue that company (or maker) 'PH-M' made by selling its Laptops.

```
select sum(unitprice*qtysold)
from   laptop L, product P
where  L.model=P.model and P.maker='PH-M'
```

- List the companies with the worst performance in sale (i.e., companies with the lowest total revenue).

```
create view sale as
select maker, model, unitprice * qtysold as itemsale
from   laptop L, product P
where  L.model=P.model
uniton
select maker, model, unitprice * qtysold as itemsale
from   PC C, product P
where  C.model=P.model

create view totalsale as
select maker, sum(itemsale) as total
from   sale
group  by maker

select maker from totalsale
where  total <= ALL (select total from totalsale)
```

2. (25 points) Consider the `Student` relation

`Student(SID, Name, Street, Status, DOB)`

We know that

- it has 15,000 tuples and is stored on 5 tuples per page
- there are about 1000 different names and the students are all living on the 500 streets around the univerity
- it has a clustered 3 levels B+ index on `Name, Street`
- the index file is much smaller than the original file; it occupies only 1000 pages
- we have a 5-page buffer

We would like to know whether there is a student living on the 'Magnolia' street or not.

Please do the following:

- Write a SQL command for answering the question.

  Your answer:

  ```
  select * from student where street='Magnolia'
  ```

- Estimate the minimal cost for answering this question, explain your computation.

  Your answer: There is not index that can be used to quickly retrive the answer. One might scan the 3000 pages and get the result. An alternative is to look for 'street="Magnolia"' in the index file. This costs 1000 pages, clearly better.

- Suppose that you are allowed to create new indices so that you can answer this question in the fastest possible way. What will you do?

  Your answer: A clear winner is a hash index on `street` since to find out the answer, one needs (assuming everything is alright) only about 1.2 page IOs to locate a value given a hash index.

3. (25 points) Consider a relation schema `S(A,B,C)` with the following characteristics:

- Total number of tuples: 16,000
- 10 tuples per page
- `A` is the key, value range from 1 to 16,000
- There are about 4,000 values for pairs of `(B,C)` (A pair (X,Y) is different than a pair (Z,U) if either $X \neq Z$ or $Y \neq U$)
- Clustered 3 levels B+ index on `A`
- Unclustered hash index on `(B,C)`
- we have a 5-page buffer

Suppose that we would like to answer the question:

```
   Select DISTICNT B,C from S WHERE A > 300 AND A < 400
```

Find the best possible way to evaluate this query, explain your answer.

Your answer: The steps are as follows:

- Use the B+ index on A to locate the rows that contain the answer ($A > 300$ AND $A < 400$). This takes 3 IOs.
- There are about 100 rows that need to be considered. This occupies 10 pages, larger than the buffer; so we need to output 10 pages. The cost will be 20 pages.
- We need to eliminate duplications. So, before we output, we sort the 10 pages (5 pages each). So, in this step, we only need to merge the two runs. This costs 10 pages (to read the 10 pages) and output cost is 1/4 of the 10 pages (is around 3 pages).
- Total cost will be 3 + 10 + 10 + 10 + 3 = 36

In fact, one can even do better than the above as follows: during the sort, we can eliminate the duplicate. So, the 5 pages would reduce to 4 pages. So, merging will cost only 8 pages instead of 10.

4. (25 points) Consider the following schema, where the keys are underlined (different keys are underlined differently):

```
Student(Id, Name, Address, Status)
Course(CrsCode, Department, CrsName)
Transcript(StudId, CrsCode, Semester, Grade)
```

Consider the following query:

```
SELECT S.Name, S.Address, S.Id, C.CrsName, T.CrsCod, T.Grade, T.Semester
FROM Student S, Transcript T, Course C
WHERE S.Id = T.StudId AND T.CrsCode=C.CrsCode AND
    T.StudId IN
        (SELECT DISTINCT StudId
        FROM   Transcript T1
        WHERE  T1.Semester='S2006' AND T1.CrsCode='CS582')
```

Assume the following statistical information:

- 15,000 tuples with 10 tuples per page in `Student` relation
- 450,000 tuples with 10 tuples per page in `Transcript` relation
- **added:** there are about 300 classes per semester and 50 semesters
- 2000 tuples, 5 tuples per page, in Course
- 5-page buffer
- Each student has on average 30 courses
- Indices
  - `Student` relation
    - On Id: Unclustered, hash
  - `Course` relation
    - On CrsCode: Sorted (no index)
    - On CrsName: Hash, unclustered
  - `Transcript` relation
    - On StudId: Clustered, 2-level B+-tree
    - On Semester, CrsCode: Unclustered, 2-level B+ tree

Answer the following:

- What does the query compute?

  Your Answer: It computes the transcripts of student who takes CS582 in S2006.

- We are supposed to compute the result of the main query in two steps. It is said that in the first step, we should compute the result of the subquery. After that, in the second step, we continue with the computation of the main query assuming that the result of the first step is available for use. In this regards, some rewritting of the query is necessary.

  Assuming that the result of the subquery is stored in the table named `TEMP`.

  – Write the query that is processed in the second step.
  Your Answer:

  ```
  SELECT S.Name, S.Address, S.Id, C.CrsName, T.CrsCode, T.Grade, T.Semester
  FROM Student S, Transcript T, Course C, Temp TE
  WHERE S.Id = T.StudId AND T.CrsCode=C.CrsCode AND T.StudId=TE.StudId
  ```

  – Show the unoptimized relational algebra expression that corresponds to the SQL query that gets processed in the second step.
  Your Answer:

  $$\pi_{Name,Address,Id,CrsName,CrsCode,Grade,Semester}(S \bowtie T \bowtie C \bowtie TE)$$

  – Draw the corresponding fully pushed query tree.
  Your Answer: One could have pushed the projection down the tree but that will result in losing of indices

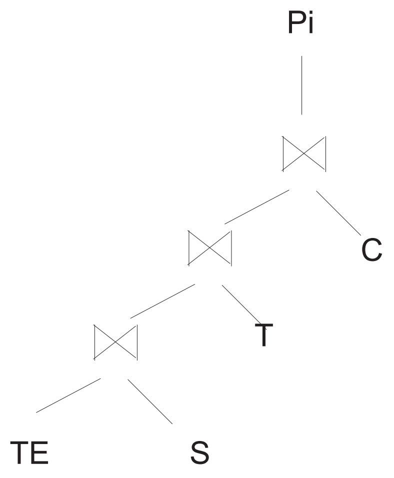

Figure 1: Query Tree

  and thus increasing the cost.

  – Estimate the cost of the execution plan based on the fully pushed query tree. Explain your answer.
  Your Answer:

  * To compute the cost of the join between TE and S, we note that there are about 450000/(300*50) = 30 students per class. Thus, TE has 30 tuples. Index nested join can be used (index on Id from Student). This costs 3 + 30(1.2 + 2) = 69 IOs plus output cost of 0 since the information fits 3 pages.
  * The join between TE, S, and T: we use index on StudID as well. For each student, there are 30 courses which are clustered in three pages. So, each student needs 2 (get to index) + 3 (read 3 pages) + 3 output. Total: 30 * 8 = 240 IOs. The result of this operation is a relation with 900 tuples. Let us call it R.

- $*$ The join between R and C: Since R has 900 tuples, it occupies 90 pages. For each student, all we need to do is to get the course name of his/her courses and project it out. So, the projection can be pipelined with this join. The 90 pages need to be read in and the fact that the file C is sorted can be used. So, for each tuple, the cost is $log_2 400$. Total cost for this operation is: $90 + 900 * log_2 400$.

The total cost: $69 + 240 + 90 + 900 * log_2 400$, approximately 8500.