Extra exercises 6: Query operators I

Question 1: True or False: It is always possible to use materialization instead of pipelined execution for any single step.

Question 2: True or False: Some operators cannot be implemented with materialization.

Question 3: Consider the following table:

```
CREATE TABLE patients (
  id INTEGER PRIMARY KEY,
  name CHAR(80),
  birth_year INTEGER,
  priority INTEGER,
  hospital_id INTEGER REFERENCES hospitals(id),
  days_in_hospital INTEGER
);
```

With the following indexes:

- Primary key index on (id) using b+ tree, clustered
- Index on (name) using hash, unclustered
- Index on (priority, birth_year) using b+ tree, unclustered
- Index on (hospital_id) using hash, unclustered

Assume that there are 1 million records of random patients, where birth_year is uniformly distributed between 1971 and 2010, hospital_id are uniformly distributed within 1000 different values, and priority is uniformly distributed from 1 to 10. Names are almost unique.

Choose the most efficient selection strategy for the following queries:

1. **SELECT * FROM patients ORDER BY name, birth_year**
2. **SELECT min(id) FROM patients WHERE priority = 1 AND hospital_id < 3**
3. **SELECT * FROM patients WHERE hospital_id = 5 AND name = 'Adam Baker'**
4. **SELECT average(priority) FROM patients WHERE hospital_id = 1 AND birth_year > 1995**

Answers:

A. Use no indexes or just primary key index.
B. Use index on (name), then filter

C. Use index on (priority, birth_year), then filter
D. Use index on (hospital_id), then filter

Question 4: What is the reduction factor of the following predicates?

1) “WHERE hospital_id = 5”
2) “WHERE birth_year >= 1991”
3) “WHERE hospital_id = 1 AND priority > 5”
4) “WHERE birth_year > 2000 OR priority = 1”

Assuming that an INTEGER costs 4 bytes to store in a record. Note that the “patient” record is exactly 100 bytes large.
Consider that we store the table in 25000 blocks, each block storing 40 records.

Question 5: Suppose that we want to execute the following query:
SELECT DISTINCT days_in_hospital FROM patients WHERE priority = 1;

- How many records do we expect to get, *before* removing duplicates? 100_000
- How many blocks do we need to store the results: 100 = 100_000 / (40 * 100 / 4)
- Including the cost to read in all initial records, the total I/O cost of removing duplicates using *optimized external sort* with **3 total passes (1 initial + 2 extra passes)** is: 25800 = 25000 + 100 + (1+2) * 100 * 2 + 100
- What is the minimum number of buffer pages required to achieve the optimal I/O cost when removing duplicates using *projection with hashing*: 11 (> sqrt(100))

The table “hospitals” is as follows:

```
CREATE TABLE hospitals (
  id INTEGER PRIMARY KEY,
  priority INTEGER,
  opening_year INTEGER,
  number_of_beds INTEGER
);
```

With the same block size, each block can hold 250 “hospital” records.
Assume that there are 25000 hospitals, stored in 100 blocks.

1) Suppose we want to run the following query:

```
SELECT patients.id AS patient_id,
       hospitals.id AS hospital_id
FROM patients, hospital
WHERE patients.priority = hospitals.priority;
```

Calculate the I/O cost for the *joining* operation using each of the following algorithms:

1. Simple nested loop join (patients first, then hospitals):
2. Page-Oriented Nested Loops Join
3. Block nested loops, with 100 outer blocks
4. Suppose that we can store the hash table in memory.
   How many extra memory blocks do we need so that we can perform one-pass hash join?

Extra exercises 6: Query operators I
Solutions

Answer 1: true
Answer 2: false
Answer 3

1. A
2. C
3. B
4. D

Answer 4:

1. 0.001
2. 0.5
3. 0.0005
4. 0.325 (first one is 0.25, second one is 0.1)

Answer 5:

1) 100_025_000 = 25000 + 25000 * 40 * 100
2) patients first, then hospitals: 2_525_000 = 25000 + 25000 * 100
   hospitals first, then patients: 2_500_100 = 100 + 25000 * 100
3) patients first, then hospitals: 50000 = 25000 + 250 * 100
   hospitals first, then patients: 125500 = 100 + 1 * 25000
4) 100 (= # of blocks of hospitals). The I/O cost of the join is 25100 = 25000 + 100