AD Applied Database Systems

Assignment 3. Due Thursday, 2 December, 2010

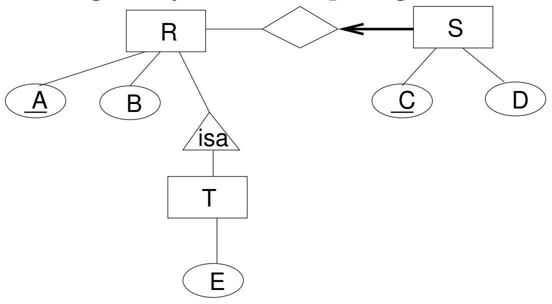
Please staple your answers and put your NAME, Student ID and COURSE (e.g. MS.) on the front.

Unless the web site says otherwise (please check it) this assignment is due on **16:00 Thursday, 2 December 2010.** Please consult the lab web page for instructions on how to turn in your project work.

Marking. This assignment counts for 11% of the course (coursework + exam) total. Of this, the project part (your answer to question 2) will account for about two-thirds, i.e., at least 7% of the course total. Do not neglect the written questions as there will be questions like these on the exam.

Written Part - Assignment 3 [50pt]

1. [20 points] Consider the following entity relationship diagram:



   (a) Write down a DTD that would best represent this schema if the database were to be exported in XML.

   Answer: The diagram indicates an "isa" relationship between T and R and a many-one "weak entity" relationship between S and R. What this means is that S and T tuples cannot exist independently of an R tuple. For XML, this means that we can place S and T elements *inside* the corresponding R element. Here goes:

   ⟨!ELEMENT DB (R*)⟩
   ⟨!ELEMENT R (A,B,T?,S*)⟩
   ⟨!ELEMENT T (E)⟩
   ⟨!ELEMENT S (C,D)⟩
   . . . everything else is PCDATA

   It is also acceptable, maybe even better, to have something like ⟨!ELEMENT R (A,B,TT,SS)⟩, ⟨!ELEMENT SS (S*)⟩, ⟨!ELEMENT SS (S*)⟩

   We have left open the issue of keys. Making C and A ID attributes is a partial answer. ID attributes are unique within a document, so we would have to make all the C attribute values different from the A attribute values in any XML. It would be equally valid to say that we cannot properly represent keys and use this as an answer to the next question. However, in this case, we are able to represent the "foreign key" relationships (that one would see in the SQL DDL) in the DTD.

   (b) Give one example of a constraint that can be expressed in SQL's DDL but not in a DTD.

   Answer: There are numerous possibilities. As we have just seen, we cannot properly represent keys. Nor can we in general represent foreign key (inclusion) relationships. At a more basic level, we cannot even constrain values to be of various types, as we have only one type PCDATA, which is some sort of TEXT field.

2. [20 points] Given the SQL

```
SELECT R.A, S.C
FROM R, S
WHERE R.B = S.B
```

Give pseudocode for (a) the naive implementation and (b) a block nested loop implementation.

Answer:

Naive:

```
for r in R:
    for s in S:
        if r.B = s.B:
            print r.A, s.C
```

BNL:

```
for br in buffers(R):
    for s in S:
        for r in br:
                if r.B = s.B:
                    print r.A, s.C
```

In BNL it is only the outer two loops that require i/o, which is what we are measuring. The inner loop can be replaced by a hash-table look-up if needed.

Suppose that

- the blocks read from disk are 50kB,
- R has 10^6 tuples and occupies 100mB,
- S has 10^5 tuples and occupies 200mB, and
- the cache holds *six* blocks, a 1-block input buffer for one table, a 4-block buffer for the other, and a 1-block output buffer.

estimate the i/o (in blocks) required for a naive join and a block nested loop join. Which outer loop – on R or S – is most efficient?

Answer: R requires $100mB/50kB = 2000$ disk pages or $2000/4 = 500$ 4-block buffers. S requires $200mB/50kB = 4000$ disk pages or $4000/4 = 1000$ 4-block buffers.

- Naive, R outermost: 2000 (for R) $+10^6 \times 4000$ (for passes through S) $= 4,000,002,000$ reads.
- Naive, S outermost: 4000 for (for S) $+10^5 \times 2000$ (for passes through R) $= 200,004,000$ reads
- BLN, R outermost: 2000 (for R) $+500 \times 4000 = 2,002,000$ reads.
- BLN, S outermost: 4000 (for S) $+ 1000 \times 2000 = 2,004,000$ reads.

in each case, R outermost wins. for BNL only by a small margin, but hugely for the naive nethod. Of course, still not nearly good enough for one to want to use the naive method in practice.

3. [10 points] Briefly describe a real-world indexing problem for which a hash index is most appropriate and one for which a B$^+$-tree is appropriate. Give reasons.

Answer: Hash table when range searches are not needed. E.g., when student id's are randomly assigned, find a student by student id. B$^+$-tree when range searches are important. E.g., lexicographic searches in an on-line dictionary or telephone book.

Project Part - Assignment 3[50pt]

For this assignment you will no longer need the data that was loaded during the first assignment. Thus, please follow the instructions on the lab web page.

in order to help save disk space before you start with the following project part. Note that the size of your database should not exceed a total of 500 MB after you are finished with this assignment!

Disease Gene Ranking

The identification of genes that are involved in certain diseases remains a challenge. A common method to identify the role of a gene in a disease is to prioritize candidate genes based on their similarity to genes that are known to be involved in that disease. That is, we want to rank genes that are not yet known to be involved in a disease according to their similarity with genes that are involved in a disease. Note that we are only interested in human diseases, and therefore limit the set of candidate genes to those from species Human.

The similarity of two genes will be derived based on the annotations they have in common. We start by assigning the following weights to the different annotations:

Annotation Type	**Weight**
Sequence Feature (with LENGTH(seq) ≤ 5)	1
Sequence Feature (with LENGTH(seq)> 5)	3
Protein Family	2
GO Process	5
GO Function	5
GO Component	1

The equality of two annotations $a1$, $a2$ is defined as follows:

Annotation Type	**Equal if ...**
Sequence Feature	a1.type = a2.type AND a1.seq = a2.seq
Protein Family	a1.dbname = a2.dbname AND a1.dbacc = a2.dbacc
GO Term	a1.goacc = a2.goacc

The similarity of two genes is then defined as the sum of the weights of annotations they have in common.

Example The following example lists the annotations of genes Q9HAT0 and Q96C74 together with their respective weight. The two genes have two annotations in common (highlighted in gray) and the similarity between them is 10.

Q9HAT0

Annotation Type	**Value(s)**	**Weight**
Sequence Feature	type: DOMAIN seq: PELPKMLKE...	3
Protein Family	dbname: InterPro dbacc: IPR003117	2
Protein Family	dbname: Pfam dbacc: PF02197	2
GO Function	goacc: GO:0008603	5
GO Process	goacc: GO:0007165	5

Q96C74

Annotation Type	Value(s)	Weight
Sequence Feature	type: DOMAIN seq: PELPDILKQ...	3
GO Function	goacc: GO:0008603	5
GO Process	goacc: GO:0007165	5

Questions: In this assignment you are asked to deliver the following: For each disease, list the best candidate genes based on their similarity to genes that are known to be involved in that disease. That is:

1. Compute the similarity between all human genes that are involved in some disease and those human genes that are not yet known to be involved in any disease.
2. Create the following table:

```
CREATE TABLE disease_candidate_gene (
disease_db          char(8) NOT NULL,
disease_acc         char(6) NOT NULL,
cand_primacc        char(6) NOT NULL,
dg_primacc          char(6) NOT NULL,
similarity          int NOT NULL
);
```

   Attributes disease_db and disease_acc correspond to disease.dbname and disease.dbacc. Attribute cand_primacc is the primary accession number of the candidate gene, dg_primacc is the primary accession number of the gene that is known to be involved in the disease identified by disease_db, disease_acc, and similarity is the similarity between cand_primacc and dg_primacc.

3. Populate disease_candidate_gene with those candidates having the highest similarity for each disease (note that there may exist multiple candidates with the same similarity for each disease).

Try to exploit indexes as much as possible in order to reduce the execution time of your queries. Materializing large intermediate result is preferable over using views. Make sure, however, that you delete all temporary tables that you create when they are no longer needed.

You are asked to hand in the following:

- A file containing all the SQL-commands that you used to compute the similarity between genes and populate disease_candidate_gene.
- A file that contains the content of disease_candidate_gene. Note that you can use \copy to export the content of a table to a file.

You should use your result from the previous assignment for this assignment. However, we provide an example schema, load files, and a load script on the lab web page.

that you may use if you are not satisfied with your own solution.