Name:________________________

Student ID:_____________________

CS 411a/433a/538a — Databases II
Midterm, Nov. 15, 2006

50 Minutes

Answer all questions on the exam page

No aids; no electronic devices.

Question	Maximum	Your Mark
1	24	
2	16	
3	6	
4	9	
5	5	
Total	60	

1. (24 marks) For each of the following, circle **T** if the statement is always true, or **F** if the statement is sometimes or always false.

   (a) **F** XML is a programming language.

   (b) **T** All opening tags in XML documents must have a matching closing tag.

   (c) **F** In an XML document, subelements of an element are only allowed to appear once.

   (d) **T** XPath allows us to search for substrings at any depth in an XML document.

   (e) **F** XPath allows us to rearrange a document for output.

   (f) **F** Algebraic query optimization in a relational database considers only binary relational algebra operations.

   (g) **T** For distributed relational databases, algebraic query optimization helps reduce the amount of data sent over the network.

   (h) **T** For distributed relational databases, algebraic query optimization needs to consider both the whole, global query and the local queries.

   (i) **F** We can consider executing joins in any order because of the algebraic rewrite rule which tells us that select distributes over join.

   (j) **F** Algebraic query optimization is impossible for object-oriented databases.

   (k) **T** In relational databases, the attributes of a tuple are usually stored in contiguous bytes on the disc.

   (l) **F** Clustering means that the result of one query can be used as input to another query.

   (m) **F** The number of disc accesses required to search a B^+-tree index for a single primary key value is directly determined by the number of characters in the primary key.

   (n) **T** To join R(A, B) and S(B, C) using a hash join, R.B and S.B have to be defined over the same domain.

   (o) **F** In a relational database, projection can be done with a single pass through a relation.

   (p) **F** The coloured balls formula is used to estimate the number of bytes per tuple after a projection.

   (q) **F** There is no formula to estimate the number of tuples which will satisfy a query of the form $\sigma_{A>value}$(R), where A is one of the columns in relation R.

   (r) **F** The number of bytes per tuple in R $\ltimes$ S is the number of bytes in R, plus the number of bytes per tuple in S, minus the number of bytes in the overlapping attributes.

   (s) **F** It is never cheaper to execute a distributed join using the semijoin algorithm, versus the "ship whole" algorithm.

   (t) **F** OIDs in an object-oriented database are always the actual disc address of the object.

(u) **F** Shredding is necessary to convert an XML document from its disc version to its main memory version.

(v) **F** Node IDs in an XML database implementation are always the actual disc address of the XML node.

(w) **F** XML documents are always stored in contiguous bytes on the disc, in an XML database.

(x) **T** Clever encoding of the Node ID in an XML database implementation can make the checking of the "//" axis much more efficient.

2. (16 marks) Consider the following O_2 schema:

```
class User inherit Object public type
    tuple(name: string,
          domain: string,
          inbox: list(Email))
end;
class Email inherit Object public type
    tuple(from: User,
          to: User,
          subject: string,
          message: string,
          cc: unique set(User))
end;
name UserSet : unique set(User);
```

(a) Give an OQL query which creates a unique set of email messages found in the inboxes of the users in UserSet. Give this answer a name in case you want to use it in the following queries.

Define PartA as

Select distinct e
from u in UserSet, e in u.inbox

(b) Give an OQL query which returns the sender's name, sender's domain, and the recipient's name and domain where the subject starts with the word "Hi".

Select a.from.name, a.from.domain, a.to.name, a.to.domain

from a in PartA
where a.subject like ("Hi *")

(c) Give an OQL query which returns the sender's name and domain for email messages where the sender has included themself (as name, domain pair) in the cc set.

Select e.from.name, e.from.domain

from e in PartA

where exists cc in e.cc: e = c

or can say

where exists cc in e.cc: e.from.name = cc.name and e.from.domain = cc.domain

(d) Give an OQL query which returns the sender's name and domain where this sender has cc'd themself on ALL messages they have sent.

select u.name, u.domain

from u in UserSet

where for all i in u.inbox:
(exists u2 in i.cc: u2.name = u.name and u2.domain = u2.domain)

3. (6 marks) Consider the following relations for a relational database:

```
Messages(FromName, FromDomain, ToName, ToDomain, MessageID, Subject, Content)
       primary key: {MessageID}
EmailsCCd(CCName, CCDomain, MessageID}
       primary key: all attributes
```

For each of the following, state whether or not the two versions are guaranteed to produce the same answer, and if so, what rule(s) are being used. If not, say why not.

(a) $\sigma_{FromName=\text{"}sylvia\text{"}}(\text{Messages} \bowtie \text{EmailsCCd})$

and

$\sigma_{FromName=\text{"}sylvia\text{"}}(\text{Messages}) \bowtie \sigma_{CCName=\text{"}sylvia\text{"}}(\text{EmailsCCd})$

These are not the same answer - the first will join over MessageID, whereas the second will probably do a Cartesian product because there are no attribute names in common. Even if the second does a join, it gets a different answer.

(b) $\pi_{FromDomain,ToDomain}(\sigma_{Subject=\text{"}DB\text{"}}(\text{Messages}))$

and

$\pi_{FromDomain,ToDomain}(\sigma_{Subject=\text{"}DB\text{"}}(\pi_{FromDomain,ToDomain,Subject}(\text{Messages})))$

These are the same. The rule is the extended version of rule 5, which encapsulates the ”just keep what you need” heuristic.

(c) $\pi_{FromName,FromDomain}(\text{Messages}) \cup \pi_{ToName,ToDomain}(\text{Messages})$

and

$\pi_{FromDomain,ToDomain}(\text{Messages} \cup \pi_{ToName,ToDomain}(\text{Messages}))$

These are not the same. The second version will not compile because the union is over two relations with different ”shape”.

4. (9 marks) Assume the following statistics are available for the above relations:

For the EmailsCCd relation:	
number of tuples	1000
bytes per tuple in CCName	20
bytes per tuple in CCDomain	20
bytes per tuple in MessageID	10
distinct values in CCName	300
distinct values in CCDomain	10
distinct values in MessageID	200

For Messages relation:	
number of tuples	500
bytes per tuple in FromName	20
bytes per tuple in FromDomain	20
bytes per tuple in ToDomain	20
bytes per tuple in ToDomain	20
bytes per tuple of MessageID	10
bytes per tuple in Subject	100
bytes per tuple in Content	500
d.v. in FromName and ToName	300
d.v. in FromDomain and ToDomain	10
d.v. in Subject	400
d.v. in Content	450

(a) What is the number of distinct values of MessageID in relation Messages?
500 (MessageID is the primary key).

(b) How many bytes per tuple are in $\sigma_{CCName=\text{"}sylvia\text{"}}$(EmailsCCd)?
add up 20 + 20 + 10 = 50

(c) How many distinct values of CCName are in $\sigma_{CCName=\text{"}sylvia\text{"}}$(EmailsCCd)?
1 (or 0), "sylvia" is the only value left in the CCName column.

(d) How many tuples are in $\sigma_{CCName=\text{"}sylvia\text{"}}$(EmailsCCd)?
here you have to estimate. The formula is (number of tuples)/(distinct values) or 1000/300.

(e) How many bytes per tuple are in EmailsCCd $\ltimes$ Messages?
The semijoin gives an answer which is the "shape" of the left hand argument. So this is the same as part (b), or 50.

(f) (2 marks) How many distinct values of MessageID are in Messages $\bowtie$ EmailsCCd?
If we assume that every message ID in EmailsCCd has a match in Messages, then the 200 distinct values in EmailsCCd will find a match. The rest of the d.v. in Messages will not. So in the join, there will be 200 d.v. of MessageID.

(g) (2 marks) How many tuples are in $\pi_{CCName,CCDomain}$(EmailsCCd)?
π keeps the three columns. The maximum number of values possible is
(d.v. of CCName) * (d.v. of CCDomain) which is 300 * 10, which is > 1000 (the total number of tuples in the EmailsCCd relation). So the upper bound on this is 1000. (this is the answer I was looking for).
If each CCName always occurs with the same CCDomain, then the lower bound is 300.

5. (5 marks) Answer **ONE** of the following (only the first one will be ma rked):

   (a) Explain why the union operator ($\cup$) plays a bigger role in processing distributed relational database queries than it does for centralized relational database queries.

   (b) Describe an algorithm for executing intersection of two relations, i.e. the intersection of two relations which have no duplicates, giving an answer with no duplicates.

   (c) Describe the top-down query execution of your OQL query in question 2(b).

   (d) If you had an application with XML documents which requires a lot of transaction processing, which type of XML database software would you choose?

   (a) in distributed databases we have fragments at different sites. Horizontal fragments are put back together by union.

   (b) sort both and then do one merge pass.

   (c) query tree has one node, and one predicate. So basically say if there is an index, use it; else use brute force to filter out the objects that do not satisfy the predicate. Then assemble the answer.

   (d) I would choose a system within an relational database package since all commercial relational packages have very mature transaction processing capabilities.