FACULTE INFORMATIQUE ET COMMUNICATIONS
SCHOOL OF COMPUTER AND COMMUNICATION SCIENCES

Laboratoire de Systèmes d'Information Repartis
Distributed Information Systems Laboratory

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Course: Distributed Information Systems

(Karl Aberer)

Final exam, winter semester 2010 / 2011
January 13th, 2011, 16:15 – 19:15

The following materials are allowed: lecture slides, exercise sheets and solutions, past exams with your own solution, and personally written notes. You can use a pocket calculator but any other electronic devices (including mobile phones, laptops, handheld devices, etc.) must be switched off. The exam consists of 34 sheets including the cover sheet. **Please write your answers on the question sheets.** If necessary you can use additional sheets. Do not separate the exam sheets (by unstapling etc.). Please number additional sheets and do not forget to put your name on them.

Seat number: …………

Name: ……………………...……………………...……………………...

SECTION: ……………………..

Please have your student card ready for control!

Each question receives a maximum of 25 points.

Question 1	**Points**	____________
Question 2	**Points**	____________
Question 3	**Points**	____________
Question 4	**Points**	____________
Total Points		____________

GOOD LUCK!

Question 1: XML Storage
Points: ___

1. Consider a DTD graph $DG = (V, E)$ where V represents the set of vertices and E the set of edges. When applying the hybrid inlining algorithm to generate a suitable relational schema for DG, what is the upper bound for the number of tables generated?
   - o $|E|$
   - o $|V|$
   - o $|E + V|$
   - o none of the above

2. Given the following DTD:

```
<!ELEMENT rockband (name, member+, album*)>
<!ELEMENT member(name,age)>
<!ELEMENT album(name,year,track+)>
<!ELEMENT name(#PCDATA)>
<!ELEMENT age(#PCDATA)>
<!ELEMENT track(#PCDATA)>
<!ELEMENT year(#PCDATA)>
```

   a) Create a relational schema for storing a document conforming to this DTD using the hybrid inlining algorithm.

   b) Create a valid XML document instance of this DTD and populate the relational tables.

   c) Translate the following XPath query: `/rockband/album/year` to SQL.

ANSWER (Q1):

ANSWER (Q1):

ANSWER (Q1):

ANSWER (Q1):

ANSWER (Q1):

Question 2: Data Broadcasting in Mobile Networks
Points: ___

Consider the following data pages in a mobile broadcast grouped by their access probabilities. All pages (A_i, B_i, C_i and E_i's) are of equal size and take 1-time unit for transmission.

Data Pages	**Access Probabilities**
A_1,A_2,A_3	$8/41$
B_1,B_2,B_3,B_4,B_5,B_6	$2/41$
$C_1,C_2,C_3,C_4,C_5,C_6,C_7,C_8,C_9,C_{10}$	$1/82$

1. Find the optimal broadcast schedule for these data pages using the **broadcast scheduling algorithm** with $f_{min} = 1$. If you need to make use of additional pages for filling up the broadcast disks, in order to properly divide the broadcast disks into chunks, make sure you use a minimal number of such pages and clearly indicate those pages as ***E_i*'s** in the schedule.

2. For the above broadcast schedule find the expected delay for receiving pages A_1, B_1 and C_1. Also find the overall expected delay.

3. Briefly discuss the impact on the broadcast schedule and expected delays when using $f_{min} = 2$.

ANSWER (Q2):

ANSWER (Q2):

ANSWER (Q2):

ANSWER (Q2):

ANSWER (Q2):

ANSWER (Q2):

ANSWER (Q2):

Question 3: Distributed Retrieval
Points: ___

Suppose we have two posting lists L_1 and L_2 of size n corresponding to terms t_1 and t_2. Let us assume that L_1 is stored on server S_1 and L_2 is stored on server S_2.

Now, suppose a two-term query composed of terms t_1 and t_2 is sent to server S_1. For executing the Fagin's algorithm, server S_1 requests server S_2 to send the first b documents in the posting list L_2. Let us call b the **request size** of a server. Then S_1 scans in parallel the first b documents from L_1 and L_2. If k documents that are common to both the posting lists are found**,** the execution continues to the random access phase of Fagin's algorithm. If less than k documents are found then server S_1 requests server S_2 to send the next b documents and continues the parallel scan. This procedure continues until, either all the documents of posting list L_2 are requested by server S_1 or k documents that are common to both the posting lists are found.

In the following we analyze the performance of this distributed retrieval algorithm. We find that the probability that server S_1 finds exactly **1 common document** among its own b top-ranked documents and the first b documents **received** from server S_2 is approximately:

$$P(one\ common\ document) = b\left(\frac{b}{n}\right)\left(\frac{n-b}{n}\right)^{b-1}$$

This is found by making the following assumptions:

- The both posting lists contain the same documents.
- The documents appear uniformly at random and they appear independently of each other in the posting lists.
- n is significantly larger than b, and therefore $\left(\frac{n-b}{n}\right)<1$ can be used as an approximation for the values of $\frac{n-b-1}{n-1}, \frac{n-b-2}{n-2}, \ldots\ldots, \frac{n-2b}{n-b}$.

Given the above description answer the following questions:

1. If the request size is b, then give the probability of finding **at least 2 common documents** after the first request for b documents by server S_1 is served?

2. How should server S_1 choose the request size b such that the probability of finding **at least 2 common documents** is greater than 95% after the first request for b documents by server S_1 is served? Assume $n=200$ and due to network layer constraints b must be greater equal than 8 and can only increase by the powers of 2, i.e., $b=\{8, 16, 32, 64, 128, \ldots\}$.

3. Discuss how you would efficiently implement the random access phase after k common documents have been found with the first request, such that the number of exchanged messages is minimized.

ANSWER (Q3):

ANSWER (Q3):

ANSWER (Q3):

ANSWER (Q3):

ANSWER (Q3):

ANSWER (Q3):

ANSWER (Q3):

Question 4: Vertical Fragmentation
Points: ___

The effectiveness of a vertical fragmentation of a relational table depends on how frequently a query finds multiple attributes it accesses in the same fragment. This observation gives rise to an approach for vertical fragmentation by determining frequently accessed fragments by using frequent itemset mining.

In this approach a set of attributes is considered as itemset I and frequency of this itemset *(Freq(I))* is the sum of the frequencies of all accesses by those queries that need ALL the attributes in itemset I , i.e.:

$$Freq(I) = \sum_{sites\, S_i} \left(\sum_{k\ such\ that\, Q_{kj}=1\ \forall\ A_j \in I} S_{ki} \right)$$

where,

S_{ki} = frequency of access of query Q_k from site S_i

Q_{kj} =1 iff attribute A_j is used in query Q_k

Note that with this model the apriori property holds: when a set of queries accesses a set of attributes frequently, then it accesses also each subset of the attributes frequently. Therefore the apriori algorithm can be applied to obtain frequently accessed attribute sets.

Assume now that a relational table with 7 attributes (A_1 -A_7) is accessed by four different queries (Q_1 -Q_4). The queries access the following attributes

Q_1	A_1, A_3, A_5, A_6, A_7
Q_2	A_1, A_2, A_3, A_4, A_5
Q_3	A_1, A_2, A_4
Q_4	A_5, A_6, A_7

Three sites S_1, S_2, and S_3 issue these queries with the following frequency.

	Q_1	Q_2	Q_3	Q_4
S_1	*10*	*5*	*0*	*5*
S_2	*0*	*10*	*5*	*0*
S_3	*0*	*0*	*0*	*10*

Answer the following questions:

1. Determine $Freq(\{A_1\})$, $Freq(\{A_1, A_2\})$ and $Freq(\{A_1, A_2, A_3\})$.

2. Use the apriori algorithm to compute the set FI of all frequent attribute sets using a minimal frequency threshold of 25.

3. Apply the following algorithm to the result FI of the apriori algorithm to construct a **partitioning** P of the attribute set.

```
1. $P = \emptyset$; $A = \{A_1, \dots, A_7\}$
2. Sort $FI$ by decreasing cardinality and then by decreasing frequency of
   its elements
3. FOR every element I $\in$ $FI$ DO
     a. Remove all $I' \in P$ where $I' \subset I$
     b. IF $I$ intersects with some $I' \in P$, THEN $P = P \cup (I \setminus I')$
        ELSE $P = P \cup I$
4. If $\cup_{I \in P} I \neq A$, add the missing attributes as singleton sets
```

   This algorithm produces a partitioning P of the attributes where all attribute sets are frequent, except possibly singleton sets. The result of the algorithm becomes the input to BEA, where each attribute set in the partition plays the role of a single attribute in the original BEA.

4. Reorder the attribute sets in P using the BEA algorithm. Affinity between any two elements P_m and P_n of the set P is defined as

$$aff\,(P_m, P_n) = \sum_{sites\ S_i} \left(\sum_{k\ such\ that\ R_{k\,m}=1,\ R_{k\,n}=1} S_{k\,i} \right)$$

Where, R_{km} =1 iff some attribute $A_j \in P_m$ is used in query Q_k

5. Consider the two cases where BEA is applied on affinity matrices constructed from the sets P and A. Now, discuss the difference in computational complexity involved in determining the vertical fragments from these affinity matrices. Assume the following:

   - The size of the set P is about half of the size of A.
   - The complexity to determine P is O($k^2\ n$), where k is the number of attributes and n is the number of queries.
   - All possible vertical fragmentations are considered.

ANSWER (Q4):

ANSWER (Q4):

ANSWER (Q4):

ANSWER (Q4):

ANSWER (Q4):

ANSWER (Q4):

ANSWER (Q4):

ANSWER (Q4):