BD2 – 29/6/2015 – first part

Please feel free to answer your test in English, Italian, or any mixture

1) Consider a schema R(IdR, A, B, IdT*), S(IdS, C, IdT*), T(IdT,D) and the following query

```
SELECT   R.IdT, R.A, count(*)
FROM     R, S
WHERE    R.IdT = S.IdT and 0 < S.C < 1000
GROUP BY R.IdT, R.A
```

Assume that R and S are stored as heap files. Assume that T.IdT is a primary key while R.IdT and S.IdT are foreign keys that refer to T.
Assume that unclustered RID-sorted indexes are defined on attributes R.IdT, R.A, S.IdT, S.C.
Assume the following table for the optimization parameters of tables R and S, and of indexes on R.A and S.C.
The size of indexes R.IdS and S.IdS can be computed by assuming that each leaf may contain 500 RID's and ignoring the space needed to contain the value of IdS.
If you need Cardenas formula Φ(n,k), approximate it with min(n,k).

	NReg	NPag	NLeaf	NKey	Min	Max
R	20.000	200				
S	100.000	10.000				
T	40.000	400				
Idx.R.A			60	10.000	0	10.000.000
Idx.S.C			205	100	0	10.000.000

a) Draw a logical access plan for the query
b) Compute NLeaf for Idx.R.IdT and Idx.S.IdT
c) Draw an **efficient** access plan for the query that uses no indexes and compute its cost
d) Compute the cost of an efficient access plan which is based on an IndexNestedLoop where R is the external relation, using all the indexes that are useful for this plan
e) Do you think that the plan in (c) is the most efficient plan for this query? Why?

2) Consider the following query the same tables as exercise 1 and assume now that the indexes on R.A and S.C are **clustered**

```
SELECT   R.IdR, S.IdS
FROM     R, S
WHERE    R.A < S.C
```

a) Consider the following operators for equijoin: IndexNestedLoop. MergeJoin and HashJoin. Which ones may be reasonably extended in order to execute this join and which ones may not? By 'reasonably' we mean that the corresponding cost may be, at least in some situations, less than the

cost of a NestedLoop – it is not reasonable to define a join algorithm that is never faster than NestedLoop

b) If you believe that one or more of the above algorithm may be reasonably extended for this, please compute the corresponding cost in this case.

3) Define the deferred update and the immediate update policies. Which one is a constraint (*requires* something) and which one is not (*allows* something)? Is this policy choice related to the Redo/NoRedo choice or to the Undo/NoUndo choice?

4) Consider the following log content. Assume that the DB was identical to the buffer before the beginning of this log, and consider a undo-redo protocol

(begin,T1) (W,T1,B,1,10) (begin,T2) (W,T2,A,1,10) (begin-ckp,{T1,T2}) (W,T2,C,1,30) (end-ckp) (begin,T3) (commit,T2) (begin,T4) (W,T4,A,10,20) (W,T3,C,30,20) (commit,T3)

a. Before starting this log, what was the content of A, B and C in the PS (Persistent Store)?
b. Assume there was a crash at the end of the logging period. At crash time, what was the content of A, B and C in the buffer? What can be said about the content of A, B and C in the PS?
c. At restart time, which transactions are put in the undo-list? Which in the redo-list?
d. List the operations that are redone, in the order in which are redone
e. After restart is finished, what is the content of A, B and C in the buffer?
f. Undo and Redo are executed in the buffer or on the PS?
g. After restart is finished, what is the content of A, B and C in the PS?
h. Assume now a NoUndo-Redo protocol. In this case, what can be said about the content of A, B and C in the PS at crash time, that is, after the completion of (commit,T3)?
i. Assume now an Undo-NoRedo protocol. In this case, what can be said about the content of A, B and C in the PS at crash time?

BD2 – 29/6/2015 – second part

1) Assume that a system with no scheduler produces the following history, where we omit the commits:

$r_2[B], r_1[A], w_3[C], r_3[B], w_2[C], r_1[C], w_1[B], r_2[A]$

a) Is this history serializable?
b) Is it possible to insert commits so that this history may be produced by a strict 2PL scheduler? If it is not, exhibit a history that may be produced by a strict 2PL scheduler if presented with the above operations in that order, assuming that each transaction commits immediately after its last operation

$r_2[B], r_1[A], w_3[C], r_3[B], c_3, w_2[C], r_2[A], c_2, r_1[C], w_1[B], c_1$

2) Consider a cooperative encyclopedia where each entry may cite other entries and is attributed to a set of authors, describe by the following XML document

```
entry*
    @id
    date
        year
        month
        day
    title
    text
    cites*
        @entryId
    author*
        @authorId
author*
    @id
    name
```

a) For each author return the name and, for each year in which there is an entry written by that author, return the year, the number of entries of that year written by that author only, the title of each such entry, and the number of entries of that year written by that author with or without any coauthor, according to the following structure

```
author*
    name
    yearByYear*
            year
            numberOfEntriesByThisAuthorOnlyThisYear
            titleOfEntryByThisAuthorOnlyThisYear*
            numberOfEntriesByThisAuthorThisYear
```

b) List the name of all pairs of countries such that the first is among the 'borderingCountry' of the second, but the second is NOT among the 'borderingCountry' of the first (of course, we are looking for inconsistent data)

3) Consider an ontology that describes an encyclopedia, with classes Entry, Category. Author, with the following predicates – in general any Entry may belong to many categories and have many authors

HasCategory: Entry × Category
HasAuthor : Entry × Author
IsAuthorOfCategory: Author × Category

Formalize the following assertions, trying not to confuse implications with double implications – when we say that an entry is of category C we mean that C is ***one*** of the (possibly many) categories of C:

a) Every entry has one category minimum and four categories at most
b) If an author A has written an entry E of a category C than A IsAuthorOfCategory C
c) If an author A has only written entries of category Math than A is in class OnlyMathOne
d) If an author A is in class OnlyMathTwo than A has only written entries whose category is Math
e) If an author A has written at least ten entries of category Math and at most five entries which do not have Math among their categories, we say the A belongs to MostlyMath

Assume to have an RDF graph plus the four statements above. Keeping into account the fact that OWL is interpreted according to the Open World approach, is there a possibility that one may prove:

a) That an author belongs to OnlyMathOne
b) That an author does not belong to OnlyMathOne
c) That an author does not belong to OnlyMathTwo
d) That an author does not belong to MostlyMath

Assume that a graph is given where England and Germany have no common language, and assume to add the following statement to the database:

4) (Optional) Assume you have an OWL reasoner with the following functionalities:
   - You can enter any OWL statement, hence you can define new classes as you like
   - You can ask: is A a subclass of B? and the reasoner will answer either Yes (if it can be derived by the current ontology) or DontKnow (if it cannot be derived)

Assume you have two classes X and Y and you want to know whether, in the current ontology, they are disjoint or are not. More precisely, you want to know whether, from the given ontology, one can derive that X and Y are disjoint (yes), or whether one can derive that they are not disjoint (no), or whether you cannot derive any of the two answers (dontknow). To solve your problem you will define some auxiliary classes and ask two separate questions to your reasoned. Show the classes you would define and the questions that you would ask