BD2 – 23/7/2015 – first part

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

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

```
SELECT   R.A, S.C
FROM     R, S, T
WHERE    R.IdT = T.IdT and S.IdT = T.IdT and 0 < T.D < 100
```

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.A, T.D, and on all the primary keys (R.IdR, S.IdS, T.IdT) and foreign keys (R.IdT, S.IdT).
Assume the following table for the optimization parameters of tables R and S, and of indexes on R.A and T.D.
The size of indexes R.IdR, S.IdS, T.IdT, R.IdT and S.IdT can be computed by assuming that each leaf may contain 500 RID's and ignoring the space needed to contain the value of IdR, IdS, IdT.
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	50.000	400				
Idx.R.A			60	10.000	0	10.000.000
Idx.T.D			1002	100	0	1.000.000

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

2) Define the deferred commit and the immediate commit 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?

3) 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,1,20) (begin-ckp,{T1,T2}) (W,T2,C,1,30) (end-ckp) (begin,T3) (commit,T1) (begin,T4) (W,T3,B,10,40) (W,T4,C,30,50) (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 – 23/7/2015 – second part

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

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

   a) Is this history serializable?
   b) 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
   c) Would a strict 2PL protocol with wound-wait deadlock prevention produce the same schedule as the one in point (b)?

2) Consider the following XML document that describe which languages a set of people is able to speak, reporting the proficiency in each language. Proficiency is reported in a scale 1-10, the bigger the better. For each language, the document describes the name and a set of countries where the language is spoken.

```
speaker*
    @id
    name
    language*
        @languageId
        proficiency
language*
    @id
    name
    country*
```

   a) For each language, return the name and then, for each proficiency value, the number of speakers with that level (unless is zero) and their names, with the following structure

```
language*
    name
    speakersByProficiency*
            proficiency
            speakerName*
```

2) Consider an ontology that speakers and languages, with classes Speaker, Language, Country, and with the following predicates

Speaks:	Speaker × Language
MotherLanguage:	Speaker × Language
IsSpoken:	Language × Country
HasCountry:	Speaker × Country

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) If a speaker S has country C and L is spoken in country C, then S has L as motherlanguage
b) If a speaker S has a country C where Italian is the only spoken language, then S has Italian as motherlanguage

Assume to have an RDF graph that contains no MotherLanguage triples. 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 a speaker has Italian as a mother language
b) That a speaker has not Italian among its a mother languages

Assume that one would also record the proficiency level of any speaker for any language.

c) Explain why the simplest solution of adding a predicate 'HasProficiencyLevel' whose range is Speaker and whose domain is the integer type does not work
d) Define a possible way of encoding this information, declaring as many classes and properties as needed.