1. (20 marks) Terminology: For each of the following statements, state whether it is **true** or **false**. If it is false, **correct the statement** without changing the underlined text. (Note: there might be more than one correction to make!)

   (a) SQL is an example of a non-procedural query language. **T**

   (b) Data in a database is partitioned if the data exists after the program that created it finishes its execution. **F**

   if the tables are broken into fragments and stored possibly on several computers.

   (c) A query language is said to be closed if we we need a key to use it. **F**

   if the answer to one query can be used as input to another query.

   (d) The purpose of using fragmentation in an object-oriented database is to break the objects down to make them easier to store. **F**

   two things to correct - in a distributed database is to create fragments which can be stored on the site(s) where they are most frequently used.

   (e) Replication transparency means that the user is aware that the data is fragmented. **F**

   two things again - the user is not aware that the data is replicated.

   (f) The Global Conceptual Schema in a distributed database contains information about global relations . **T**

   (g) Tuples in a relational database commonly have nested sets of subtuples. **F**

   have flat tuples

   (h) The concept of shallow equal in OODB querying is used to talk about whether or not the answers to two queries can be considered "the same". **T**

   (i) All XML document opening tags must contain attributes. **F**

   attributes are optional

   (j) XML documents must conform to a given XML schema. **F**

   schemas are optional

   (k) XML Schemas can include enumerated types. **T**

   (l) XQuery FLWOR expressions can be used to change the "shape" of an XML document. **T**

2. (10 marks) For each of the following, say whether or not the two relational algebra expressions are always equivalent. If they are equivalent, say by what type of rule(s) this is so (if you don't remember the name of the rule(s), just explain why in words); if not, say why they are not equivalent.

   Assume the relations being used are R(A, B, D) and S(C, D, E, F)

   (a) $\sigma_{A=5}(\pi_{A,D}(\mathrm{R}))$
   and
   $\pi_{A,D}\ (\sigma_{A=5}(\mathrm{R}))$
   These are the same. commutativity of select and project providing the attributes for select are still there in the case of the first expression above.

   (b) $\pi_{A,D}(\mathrm{R} \bowtie \mathrm{S})$
   and
   $\pi_{A,D}(\mathrm{R}) \bowtie \pi_{A,D}(\mathrm{S})$
   Not always equivalent, in fact the second expression would not compile as there is no attribute A in relation S.

   (c) $\sigma_{A=5}\ (\mathrm{R} \cup \mathrm{S})$
   and
   $\sigma_{A=5}\ (\mathrm{R}) \cup \sigma_{A=5}\ (\mathrm{S})$
   again not the same, in fact the two tables are not union compatible, so neither expression would compile.

   (d) $\sigma_{A=5\ and\ F=\text{"}any\text{"}}\ (\mathrm{R} \bowtie \mathrm{S})$
   and
   $\sigma_{F=\text{"}any\text{"}}\ (\mathrm{S}) \bowtie \sigma_{A=5}\ (\mathrm{R})$
   These are the same. Commutatitivy of natural join lets us reorder R and S, and select distributes over join.

   (e) $\sigma_{A=5}\ (\mathrm{R}) \ltimes\ (\mathrm{S})$
   and
   $(\mathrm{R}) \ltimes\ (\sigma_{A=5}(\mathrm{S}))$
   these are not the same, The second version won't compile because there is no attribute A in S, and in any case, the result only gives a subset of relation R, so even if S had an attribute A, it would not be valid unless A was the join attribute.

3. (17 marks) Consider these relations:

```
Papers(DocId, Title, WherePub, Year)   primary key is {DocID}
Authors(FName, LName, Location)        primary key is {FName, LName}
PaperAuths(FName, LName, DocId, position)  primary key is {FName, LName, DocID}
```

Furthermore, assume we have created fragments so that we can store information about old papers on one site, and information about newer papers on another site in a distributed database.

Old = $\sigma_{Year<2000}$(Papers)
New = $\sigma_{Year>=2000}$(Papers)
OldPaperAuths = PaperAuths $\ltimes$ Old
NewPaperAuths = PaperAuths $\ltimes$ New

(a) (1 mark) What kind of fragmentation is represented by Old and New?
horizontal

(b) (1 mark) What kind of fragmentation is represented by OldPaperAuths and NewPaperAuths?
derived horizontal

(c) (2 marks) What attributes are included in the fragment NewPaperAuths?
the attributes of PaperAuths: Fname, LName, DocId and position

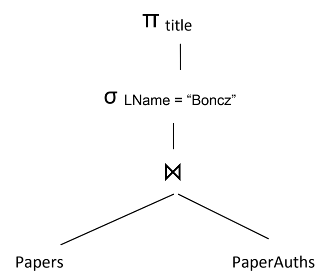
(d) (2 marks) Give the relational algebra expression to reconstruct PaperAuths from OldPaperAuths and NewPaperAuths.
PaperAuths = OldPaperAuths $\cup$ NewPaperAuths

(e) (3 marks) We want to execute the following SQL query over the distributed data above:

```
Select p.title
From Papers as p, PaperAuths as a
Where p.DocID = a.DocId and a.LName = "Boncz"
```

Translate the query to a relational algebra expression on global relations. Use a join operator if appropriate.
$\pi_{title}$ ( $\sigma_{LName=\text{"}Boncs\text{"}}$ (Papers $\bowtie$ PaperAuths ))

(f) (2 marks) Show the query tree corresponding to your algebra query just above.

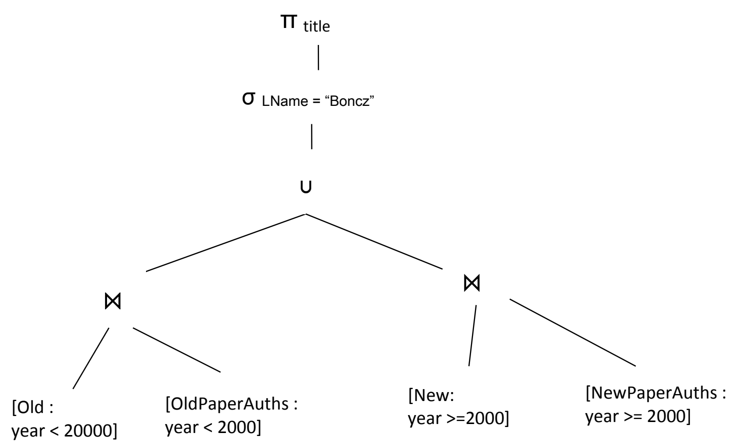



(g) (6 marks)
i) Replace any relations in your tree with the fragments defined above, if appropriate.
ii) Express the relations as qualified relations of the form

`[R:F]`

where F contains any predicates you know to be true about the fragment.
iii) After doing this, perform any further optimizations possible on your algebra tree (put your answer on the back of page 6).

first substitute Old ∪ New for Papers, and OldPaperAuths ∪ NewPaperAuths for PaperAuths. This gives the join of 2 unions. Replace that with the union of 4 joins, and delete the 2 joins where the :F's are a contradiction. This gives the next tree, which can be further optimized by pushing $\sigma$'s down and introcuding more $\pi$'s.


π title
σ LName = "Boncz"
∪
⋈
⋈
[Old :
year < 20000]
[OldPaperAuths :
year < 2000]
[New:
year >=2000]
[NewPaperAuths :
year >= 2000]

4. (5 marks) Answer **One** of the following (only the first answer will be marked).

   (a) Explain what types of queries in an object-oriented database might require new class definitions before the query can be executed.

   (b) Explain the completeness condition for fragmentation in a distributed database and why it is needed.

   (c) Explain what bag operations like bag union and bag intersection are and why SQL systems need to worry about them.

   (d) Give a counterexample to show that projection does not distribute over set difference.

(for rough work or answers to Question 10)