1. (30 marks) For each of the following, state whether the statement is **true** or **false**. If it is false, correct the statement **without changing the underlined text**. If the statement is true, do not write anything more, just indicate true.

(a) A database being of **arbitrary size** means that the schema can have an arbitrary number of new types defined.
   **F** means the size of the database can exceed the size of main memory or virtual memory.

(b) The ANSI/SPARC architecture is an example of a client/server system.
   **F** is a data organization architecture for centralized databases.

(c) The global schema for a distributed relational database describes the relations that are stored locally on the site where the global schema resides.
   **F** describes the global relations, i.e. the schema for the whole database, and perhaps the fragments that have been defined and where they are stored.

(d) Fragments are used in a distributed database in order to satisfy functional dependencies.
   **F** to break the tables into smaller parts which can then be stored close to where they are most commonly used.

(e) For vertical fragmentation, the disjointness condition means that the fragments should have disjoint rows.
   **F** disjoint columns, plus replicating the primary key.

(f) Vertical fragmentation divides a relation into fragments using the relational algebra select ($\sigma$) operator.
   **F** uses projection ($\pi$).

(g) Aggregation is that process by which the system verifies that the user has permissions to execute a query.
   **F** new types are built by juxtaposing attributes of possibly different types.

(h) **Name conflicts arise with attribute names** when we have replication.
   **F** when we have multiple inheritance

(i) Distributed relational databases are good for companies which have data with complex structure or complex processing requirements.
   **F** which have application requirements and data at multiple locations

(j) Persistence means that the result of one query can be used as input to another query.
   **F** means that the data exists when the program/system that created it terminates, and is there tomorrow.

(k) Path expressions in XPath are used to mimic relational algebra projection ($\pi$).
   **F** drill down into nested subelements to express predicates or to extract pieces.

(l) In an object-oriented database query language, two objects which share no storage but have the same values in their attributes are said to be identical.
   **F** equal or deep equal.
(m) Having an XML Schema is compulsory for dealing with an XML document.
   F not compulsory

(n) The goal of rearranging relational algebra expressions during query optimization is to find an execution order that reduces the amount of disc I/O.
   T

(o) For distributed relational databases, algebraic query optimization needs to consider both the whole, global query and the local queries.
   T
2. (10 marks) Consider the following relations for a relational database:

Singer(SingerID, Name, Genre, Town) primary key: {SingerID}
Song(SongID, Title, Composer, Genre) primary key {SongID}
Performs(SingerID, SongID, Date, Position) primary key {SingerID, SongID}

(a) Give the Relational Algebra expression for the fragment of the Song relation for songs composed by Paul McCartney.

\[ \text{PartA} = \sigma_{\text{Composer}='\text{PaulMcCartney'}} (\text{Song}) \]

(b) Give the Relational Algebra expression for that fragment of the Performs relation where the song is composed by Paul McCartney. Express this in terms of your answer to part (a).

\[ \text{PartB} = \text{Performs} \bowtie \text{PartA} \]

(c) Looking at your answer to part (a), what other fragment must be present to guarantee reconstruction of the Song relation?

\[ \text{PartC} = \sigma_{\text{Composer} \neq '\text{PaulMcCartney'}} (\text{Song}) \]

(d) Give the Relational Algebra expression to perform the reconstruction of relation Song from the fragments in parts (a) and (c).

\[ \text{Song} = \text{PartA} \cup \text{PartB} \]

(e) For privacy reasons, information concerning the Towns of singers cannot be released to the public. Give relational algebra expressions for two fragments of Singer, one which can be publicly released, and one which should not. Make sure that your fragments follow the guidelines for fragmentation for distributed databases.

we will use vertical fragments here:

\[ \text{Public} = \pi_{\text{SingerID}, \text{Name}, \text{Genre}} (\text{Singer}) \]

\[ \text{Private} = \pi_{\text{SingerID}, \text{Town}} (\text{Singer}) \]
3. (8 marks) Which of the following pairs of relational algebra expressions give “the same” answer, i.e. are what we would call equivalent?

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

(a) $R \bowtie S$ and $
\sigma_{R.C=S.C}(R \times S)$

(b) $\pi_C(R \bowtie S)$ and $
\pi_C(R) \bowtie \pi_C(S)$

(c) $\pi_A (R \cup S)$ and $
\pi_A(R) \cup \pi_A(S)$

(d) $\sigma_{A=5 \text{ and } D=\text{"any"}} (R \bowtie S)$ and $
\sigma_{A=5}(R) \bowtie \sigma_{D=\text{"any"}} (S)$

(e) $\sigma_{A=5 \text{ and } E>5} (R \bowtie S)$ and $
\sigma_{E>5}(S) \bowtie \sigma_{A=5} (R)$

(f) $\pi_{C,D}(\sigma_{A=5} (R)) \cap \pi_{C,D} (S)$ and $
\sigma_{A=5} (\pi_{C,D} (R) \cap \pi_{C,D} (S))$

(g) $R \bowtie S$ and $
\pi_{A,B,C,D} (R \bowtie S)$

(h) $R \bowtie \sigma_{E>5} (S)$ and $
\sigma_{E>5} (R \bowtie S)$
4. (10 marks) Consider the following relations for data for a relational database:

- \text{Employee}(\text{SIN, name, address, jobtitle, DeptID}) ~~~ \text{primary key is \{SIN\}}
- \text{Project}(\text{ProjNo, projectName, projectLocation}) ~~~ \text{primary key is \{ProjNo\}}
- \text{WorksOn}(\text{SIN, ProjNo, hoursPerWeek}) ~~~ \text{primary key is \{SIN, ProjNo\}}

Suppose the following fragments have been created (we intend to store the first and third fragments at one site, and the second and fourth fragments at another):

- \text{Programmers} = \sigma_{\text{jobtitle} = \text{"programmer"}} (\text{Employee})
- \text{Non-P} = \sigma_{\text{jobtitle} \neq \text{"programmer"}} (\text{Employee})
- \text{WorksOnP} = \text{WorksOn} \bowtie \text{Programmers}
- \text{WorksOnNonP} = \text{WorksOn} \bowtie \text{Non-P}

(a) (2 marks) Suppose we have the SQL query:

\text{Select name, ProjNo, hoursPerWeek}
\text{From Employee, WorksOn}
\text{where hoursPerWeek > 20}
\text{AND Employee.SIN = WorksOn.SIN}

(last line added during exam)

Translate this query to relational algebra.

\begin{align*}
\pi_{\text{name, ProjNo, HoursPerWeek}} \\
(\sigma_{\text{hoursPerWeek} > 30 \ \text{AND Employee.SIN} = \text{WorksOn.SIN}} (\text{Employee} \times \text{WorksOn}))
\end{align*}

(b) (3 marks) Show the initial query tree corresponding to your algebra query just above.
(c) (5 marks) Replace any relations in your tree with the fragments defined above, expressed as qualified relations of the form \([R:F]\) where \(F\) contains any predicates you know to be true about the fragment. After doing this, perform any further optimizations possible on your algebra tree.

Employee gets replaced by the subtree formed from, \([\text{Programmers: jobtitle} = \text{“programmer”}]\) \(\cup\) \([\text{Non-P: jobtitle} \neq \text{“programmer”}]\)

WorksOn gets replaced by the subtree formed from, \([\text{WorksOnP: jobtitle} = \text{“programmer”} \text{ and E.SIN = W.SIN}]\) \(\cup\) \([\text{WorksOnNonP: jobtitle} \neq \text{“programmer”} \text{ and E.SIN = W.SIN}]\)

This gives the join of 2 unions, which is replaced by the union of 4 joins. 2 of them are empty and get pruned. \(\pi’s\) and \(\sigma’s\) get pushed down. Extra \(\pi’s\) are inserted to “just keep what you need”. The ones on the WorksOn branches are deleted because they project onto all attributes.
5. (12 marks) Answer any three of the following (only the first three answers will be marked):

(a) When looking at distributed DBMS alternatives, we said that choices of autonomy, distribution and heterogeneity are orthogonal. What did we mean by orthogonal in this case?

(b) When we design fragments for a distributed database, we require the reconstruction condition to be satisfied. What does the reconstruction condition guarantee?

(c) Why is the ER model not always adequate to model data for a complex database application?

(d) What does closure mean for a query language and why is it a good property to have?

(e) Why might we need to declare new classes to hold answer objects in an object-oriented query environment?

(f) Why is an XML database not ideal for handling many-to-many relationships?

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

(h) Show by a counterexample that projection does not distribute over set union ** this should be set difference**.
For rough work or answers to question 5.