1 Short Answer

a) When talking about distributing our databases across multiple servers, we discuss data needs as being either transactional or analytical. For both of these cases, do we prefer partitioning or duplication and why?

Transactional - many writes
prefer partitioning since we only have to write to one location

Analytical - many reads
prefer duplication so we can distribute read commands over many servers

b) In datalog, what is "fixed point"? To what sorts of rules does this semantic apply?

In recursive Datalog, a "fixed point" is the base case (from our knowledge base) in our recursive definition of a rule.

Fixed point semantics are such that the definition will iteratively add new facts to the knowledge base until there is no change between iterations. This indicates the program can stop running

c) What is the standard order of operations for a SQL query?
Hint: Select, from, where, group by, having, order by

FWGHOS

d) Give an example of a query that can be expressed with datalog, but not with set-semantic Relational Algebra

any recursive query

ex: Find all members of a family that are in the same generation.

ex: Find all servers connected to server 'x' through any servers.

2 SQL

For this question, use a relational database storing student course records with the following schema. Use AS when aliasing. Primary keys are underlined.

`Student(sid,dept,gradyear,fname,lname)`

`Course(dept,coursenum,name,description)`

`Takes(sid,dept,coursenum,quarter)`

a) Write the `CREATE TABLE` statement for the table `Takes`. `Sid` is an `INT`, `dept` is a `VARCHAR(5)`, `coursenum` is an `INT` and `quarter` is a `VARCHAR(4)`. `Sid`, `dept` and `coursenum` are all foreign keys.

ex:

```
CREATE TABLE Takes (
    sid INT,
    dept VARCHAR(5),
    coursenum INT,
    quarter VARCHAR(4),
    FOREIGN KEY (sid) REFERENCES Student(sid),
    FOREIGN KEY (dept, coursenum)
        REFERENCES Course(dept, coursenum));
```

b) Write a query which finds the unique first and last names of all students that took UWBW 599 (Underwater Basket Weaving Dissertation)

ex:

```
SELECT DISTINCT S.fname, S.lname
  FROM Student AS S JOIN Takes AS T
       ON S.sid = T.sid
 WHERE T.dept = 'UWBW' AND
       T.coursenum = 599;
```

e) Explain the difference between OPEN and CLOSED types in semi-structured data

OPEN types allow other fields/keys than those defined in the Type

CLOSED types can only have the defined fields/keys in the Type

f) Provide two unique relational algebra expressions which provide the same final relation. Use `A,B,C...` as your starting relations as needed.

ex: $\Pi_{A.a}(\sigma_{A.a>100}(A))$

$\sigma_{A.a>100}(\Pi_{A.a}(A))$

ex: $(A \bowtie B) \bowtie C$

$A \bowtie (B \bowtie C)$

3 Datalog

For the following question, use a schema around published authors and scholarly articles.

`Author(autid,fname,lname)`

`Article(articleid,name,year)`

`Authored(autid,articleid)`

Write a datalog rule degreeSeparation(autid,num) which calculates the degree of separation the author has from `Author(_,Paul,Erdos)`. You may assume there is exactly one author named Paul Erdos.

For those unfamiliar with the problem, degrees of separation indicates how close two authors are. If you imagine authors as vertices in a graph and edges representing whether those two authors have co-authored a paper, the degree of separation for two authors is the length of the *minimum* path between them.

As an aside: because this problem is difficult to describe, it is too difficult for the actual exam, but I think it illustrates the type of datalog program you should be able to write. Use as many rules as necessary.

numAuthors(n) :- n = sum(1) : Author(_,_,_)

erdosNum(a,0) :- Author(a, 'Paul', 'Erdos')

erdosNum(a,n+1) :- erdosNum(x,n), Authored(x,id),
Authored(a,id), numAuthors(m),
n+1 ≤ m

degreeSeparation(a,n) :- erdosNum(a,_),
n = min(d) : erdosNum(a,d)

c) Write a query to find all the students who took only classes 400-level or lower.

ex:

```
SELECT DISTINCT S.fname, S.lname
  FROM Student AS S
 WHERE 500 > ALL(SELECT T.coursenum
                   FROM Takes AS T
                  WHERE T.sid = S.sid);
```

ex:

```
SELECT DISTINCT S.fname, S.lname
  FROM Student AS S
 WHERE NOT EXIST(SELECT *
                   FROM Takes AS T
                  WHERE T.coursenum ≥ 500
                    AND T.sid = S.sid);
```

d) Is this query monotone? Why or why not?

No. If a student is specified in the original query, we can eliminate that student by adding a tuple where in Takes, that student takes a 500+ course.

4 Relational Algebra

Use the following schema.

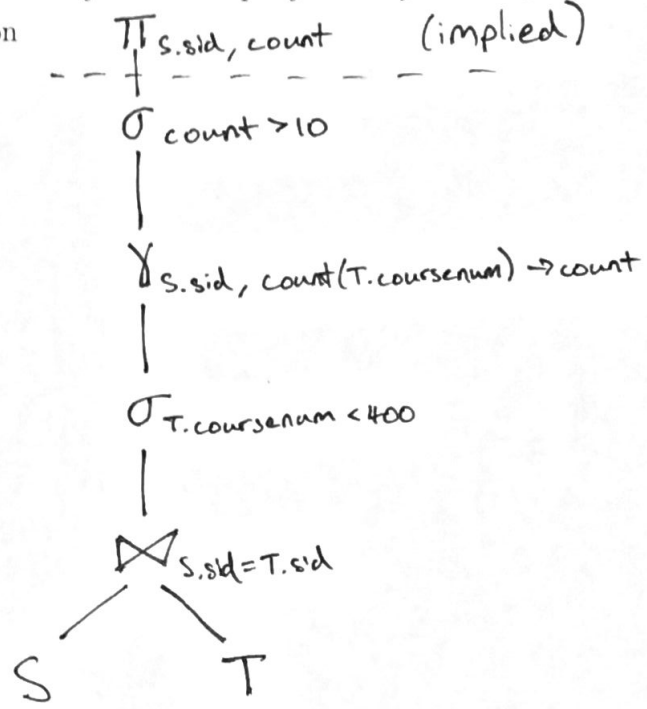
Student(sid,dept,gradyear,fname,lname)

Course(dept,coursenum,name,description)

Takes(sid,dept,coursenum,quarter)

Write the bag-semantic relational algebra expression for the following query. You may use S for Student and T for takes without renaming. You may draw the query tree if you'd like, but your final solution should be the RA expression

```
SELECT S.sid,
     COUNT(T.coursenum) as count
FROM Student AS S
     JOIN Takes AS T
          ON S.sid = T.sid
WHERE T.coursenum < 400
GROUP BY S.sid
HAVING COUNT(T.coursenum) > 10
```



$$\sigma_{\text{count}>10}\left(\gamma_{\text{S.sid, count(T.coursenum)}\rightarrow\text{count}}\left(\sigma_{\text{T.coursenum}<400}\left(S \bowtie_{S.sid=T.sid} T\right)\right)\right)$$