2020-2021 学年第 2 学期

（2021 春季）

《数据管理技术》

期末考试卷

班级_____学号_____

姓名_____成绩_____

2021 年 6 月 23 日

Problem 1: Single-Choice Questions (20 points).

1. Which is **NOT** an advantage of a DBMS over file-based systems: 【 E 】

A. Control of data redundancy
B. Improved data consistency
C. Easier data sharing.
D. Improved security.
E. Improved performance

2. The database system must take special actions to ensure that transactions operate properly without interference from concurrently executing database statements. This property is referred to as: C

A. Atomicity
B. Consistency
C. Isolation
D. Durability
E. All of the above

3. Select the relational expression that could possibly return the following result: A

a	c
1	2
2	3

A. $\sigma_{a<c}(\pi_{a,c}R)$
B. $\pi_{a<c}(\sigma_{a,c}R)$
C. $\sigma_{a<2}R$
D. $\sigma_{a,c}R$

4. Of the following fields, which would be the most appropriate for the primary key in a customer information table? B

A. Customer name
B. Customer ID number
C. Phone number
D. Customer address.

5. If your SELECT clause contains both aggregate and non-aggregate functions, _. B

A. all non-aggregate columns must be included in a WHERE clause
B. all non-aggregate columns must be included in a GROUP BY clause
C. all aggregate and non-aggregate columns must be included in a GROUP BY clause
D. all aggregate columns must be included in a WHERE clause

6. In condition that x > ALL(5, 9), the expression below will be TRUE for what values of x? (All is a key word of SQL) A

A. above 9
B. 5 to 9
C. 5 and above

D. 9 and above

7. Which of the following normal forms have atomic values? 【 d 】

A. 1NF B. 2NF
C. 3NF D. (A, B, and C)

8. Which of the following is **not** valid for NoSQL? c

A. High performance
B. Document oriented
C. Improved ability to keep data consistent
D. More easily allows for data to be held across multiple servers

9. Which statement about view is **not** correct? 【 c 】

A. View is external schema
B. View is virtual table
C. To query on view can accelerate the query speed
D. We can use view to simplify the query statement

10. Consider following transactions

T1	T2	T3
Read (X)		
	Read (Y)	
		Read (Y)
	Write (Y)	
Write (X)		
		Write (X)
	Read (X)	
	Write (X)	

Which of the following schedules below is the correct serialization (i.e., an equivalent non-serial schedule) of the above? C

A. T1 → T2 → T3
B. T2 → T3 → T1
C. T1 → T3 → T2
D. T3 → T2 → T1
E. This is a non-serializable schedule.

Problem 2: Answer the questions (15 points)

1. What's index? Should we build index on each field of the table? Why?

2. Please make a comparison study for the two consistency models --BASE and ACID.

3. Briefly explain the meaning of the following trigger.

```
CREATE TRIGGER FooTrigger
  AFTER UPDATE OF salary ON Emp
  REFERENCING
```

```
    OLD AS OldTuple, NEW AS NewTuple
  WHEN ((OldTuple.salary > NewTuple.salary) AND
    (OldTuple.name = 'Tom Smith'))
    UPDATE EmpSET salary = OldTuple.salary
    WHERE   SSN = NewTuple.SSN
  FOR EACH ROW;
```

Problem 3: Relational algebra and SQL (20 Points)

Consider a social network database, about people and their relationships. The database has two relations:

Person(pid, name)

Relationship(pid1, rel, pid2)

Here Person.pid is a key, and Relationship.pid1 and Relationship.pid2 are foreign keys; rel is a string representing the relation type, and can be "friend" or "enemy". For example, a tuple from **Relationship**: (p1, 'friend', p2) means that person p1 is friend with person p2. Note that the relationship is not necessarily symmetric: if Alice is friend with Bob, this does not imply that Bob is friend with Alice.

Questions:

1. Write a **relational algebra** expression to query all the enemies of Bob who didn't list Bob as their friend.

2. Write a **SQL** query that computes, for each person, the total number of their friends. Your query should return results containing the pid, the name, and the count. Note that your query must return exactly one answer for every person in Person.

```
select x.pid, x.name, count(*)
from Person x left outer join Relationship y
   on x.pid = y.pid1 and y.rel='friend'
group by x.pid, x.name;
```

3. "My enemies' enemies are my friends". Write a SQL statement to mark that all the enemies' enemies of each person as his friends in **Relationship**. NOTE: The Relationship table may already contains some records before your operation, so make sure that your SQL statement doesn't produce duplicate records.

For example:

Original Relationship table:

Pid1	Rel	Pid2
A	enemy	B
B	enemy	C
B	enemy	D

A	Friend	C

After the execution of your SQL statement:

Pid1	Rel	Pid2
A	enemy	B
B	enemy	C
B	enemy	D
A	friend	C
A	friend	D

```
insert into Relationship
   select x.pid1 as pid1, 'friend' as rel, y.pid2 as pid2
   from Relationship x, Relationship y
   where x.rel = 'enemy' and x.pid2=y.pid1 and y.rel='enemy'
    and not exists (select * from Relationship z
                        where x.pid1=z.pid1 and z.rel='friend'
                          and y.pid2=z.pid2);
```

4. "I shall make peace with all my enemies whose enemies are not my friends". Write a SQL statement to delete some records in **Relationship** to enforce above assertion. That is, for example, if you find in **Relationship** that A's enemy is B, B's enemy is C, and A is not friend with C, then delete the records of (A, 'enemy', B) in **Relationship**.

For example:

Original Relationship table:

Pid1	Rel	Pid2
A	enemy	B
B	enemy	C
A	enemy	C
A	enemy	E
E	enemy	F

After the execution of your SQL statement:

Pid1	Rel	Pid2
B	enemy	C
A	enemy	C
E	enemy	F

```
delete from Relationship
where rel='enemy' and
    exists (select *
              from Relationship y
              where pid2=y.pid1 and y.rel='enemy'
                and not exists(select * from Relationship z
                                   where pid1=z.pid1 and z.pid2=y.pid2
                                     and z.rel = 'friend'));
```

Problem 4: Normal Form (10 points)

Given relation R(A, B, C, D, E, F, G) and the set of functional dependencies F ={BCD → A, BC → E, A → F, F→G, C→D, A→G}, decompose R into 3NF. Show your steps. Is this decomposition also BCNF? Why or why not?

Note: This requires you to first determine the key(s) of R.

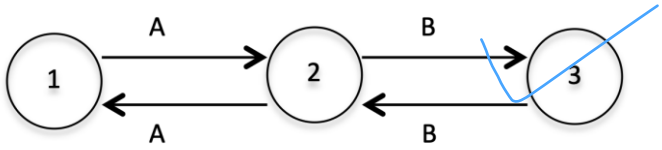
R1(B, C, A, E) R2(A, F) R3(F, G) R4(C, D)

Problem 5: Serializability (10 points)

For each of the following transaction schedules, draw the precedence graph and decide if the schedule is conflict-serializable. If the schedule is conflict-serializable, give an equivalent serial schedule (you just need to list the order of transactions). If the schedule is not conflict-serializable, explain why not.

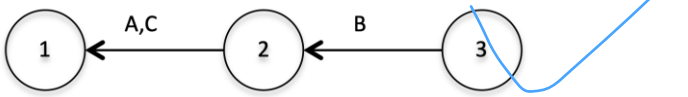
1. r1(A); r2(A); w2(A); r3(B); r2(B); w3(B); w2(B); w1(A)

This is not conflict-serializable. T1 and T2 have RW conflicts in both directions on A, and T2 and T3 similarly have RW conflicts on B. There are cycles in the graph so it is not conflict-serializable.



2. r1(A); r2(A); r3(B); w1(A); r2(C); r2(B); w2(B); w1(C)

This schedule is conflict-serializable. T3 must occur before T2 because of a RW conflict on B and T2 must occur before T1 because of RW conflicts on A and C. Since there are no cycles, T3, T2, T1 is an equivalent serial schedule



Problem 6: Locking (10 points)

Consider the following schedule:

r1(X) w1(X) r2(X) r1(Y) w1(Y) r2(Y) w2(Y) w2(X)

1. Is this schedule a two-phase locking protocol (2PL) schedule? Explain briefly why or why not. (If it is 2PL schedule, an easy way to demonstrate that is to rewrite the schedule with lock and unlock operations inserted in the appropriate places.)

Yes. One equivalent schedule with 2PL inserted is

L1(X) L1(Y) r1(X) w1(X) U1(X) L2(X) r2(X) r1(Y) w1(Y) U1(Y) L2(Y) r2(Y) w2(Y) w2(X)

2. Is this schedule the strict two-phase locking protocol (strict 2PL) schedule? Explain briefly why or why not. (If it is strict 2PL, an easy way to demonstrate that is to rewrite the schedule with lock and unlock operations inserted in the appropriate places.)

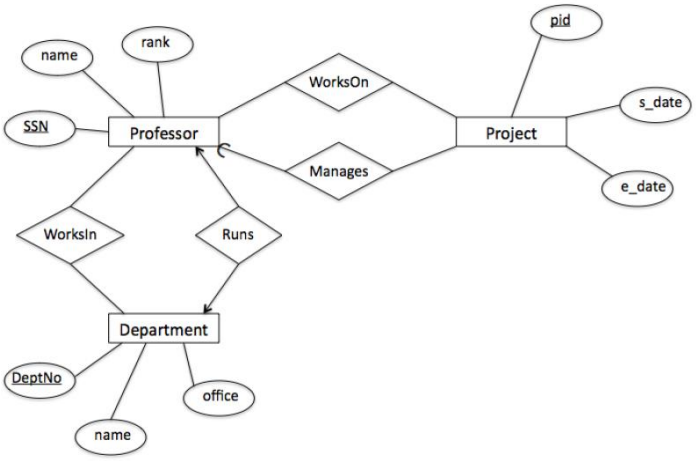
No. With strict 2PL, T1 cannot release either of its locks until it is finished. In the original schedule, T2 reads X before T1 finishes, but that cannot happen since T2 would not be able to acquire the lock on X while T1 is still active.

Problem 7: Database modelling (10 points)

You have been asked to design a database about a university:

1. Professors have an ID, a name, and a rank (e.g., Professor, Associate Professor (副教授))
2. Research Projects are identified by a project number. Each project also has a starting date and an ending date
3. Each project is always managed by one professor
4. Each project is worked on by one or more professors
5. Professors can work on multiple projects. Professors also can manage multiple projects.
6. Departments have a unique department number, a department name, and a main office
7. Departments can have a professor (department chair) who runs the department. A department can have no more than one department chair, but may temporarily have no chair.
8. Professors can work in one or more departments, but a professor can't be chair of more than one department.

(a) Draw an ER diagram for this application. Be sure to mark the multiplicity(基数比约束) of each relationship of the diagram. Decide the key attributes and identify them on the diagram. Please state all assumptions you make in your answers. (5 points)



(b) Translate your ER diagram into a relational schema. Specify the key of each relation in your schema. (5 points)

Solution: If we do not merge relations at all, we get:

Professor(SSN, name, rank)
Project(pid, s date, e date)
Department(DeptNO, name, office)
WorkOn(SSN, pid)
Manages(SSN, pid)
WorksIn(SSN, DeptNO)
Runs(SSN, DeptNO)

But, we could combine the Project relation with the Manages relation, and the Department relation with the Runs relation:

Professor(SSN, name, rank)
Project(pid, SSN, s date, e date)
Department(DeptNO, SSN, name, office)
WorkOn(SSN, pid)
WorksIn(SSN, DeptNO)