Transaction

Q1: Specify which of the following schedules could be generated by
a) Two phase locking scheduler
b) Rigorous two-phase locking scheduler

If the schedule could be generated by 2PL, specify when transactions would acquire locks and release locks. If not, explain why the lock request will fail and hence the schedule would not be generated.

1. w_1 (x) w_2(x) w_1(y) w_2(y) c_1 c_2
2. w_1(x) w_2(x) w_2(y) w_1(y) c_1 c_2
3. w_1(x) r_2(x) w_3(y) w_2(y) c_1 c_2 c_3

In the schedules above, the subscript refers to the id of the transactions, that is, an operation w_1(x) is a write operation by transaction T1 on data item x.

Indexing

Q2: Consider a relation Employee(
ssn INT PRIMARY KEY,
name char(30),
salary REAL CHECK (salary BETWEEN 50K AND 250K)
department CHAR(50))

Let us consider the following indices:

I1: CREATE INDEX salary_index ON Employee (salary)

I2: CREATE INDEX composite_index ON Employee(name, department)

I3: CREATE INDEX ssn_index ON Employee (ssn)

Now consider the following queries:

Q1: SELECT * from Employee where ssn = 123 AND name LIKE "Mike%"

Q2: SELECT * from Employee where name = "Jim"

Q3: SELECT * from Employee where department = "CS"

Q4: SELECT * from Employee where salary > 51000 AND salary < 249000

For each of the queries above, explain which index, if any, would be useful in reducing the execution time of the query.

Object-Relational Database System

Q3: Consider the following sequence of SQL statements

```
create type Person
    (ID varchar(20) primary key,
     name varchar(20),
     address varchar(20))
     ref from(ID);
create table people of Person;


create type Department (
     dept_name varchar(20),
     head ref(Person) scope people);
create table departments of Department

insert into people values ('12345', 'Alice', '123 Wonderland Ave');
insert into people values ('23456', 'Bob', '456 Nowhere Blvd');
insert into people values ('34567', 'Charlie', '789 Imaginary St');

insert into departments values ('CS', '12345');
insert into departments values ('Math', '23456');
insert into departments values ('Physics', '34567');

create type Project (
    proj_name varchar(20),
    lead ref(Person) scope people,
    department ref(Department) scope departments
);
create table projects of Project;
```

```
insert into projects values ('AI Research', '23456', 'CS');
insert into projects values ('Quantum Computing', '34567', 'Physics');
```

Q3. What will be the output if we now execute the following SQL query

```
select proj.proj_name as ProjectName,
       proj.lead->name as LeadName,
       proj.department->dept_name as DepartmentName,
       proj.department->head->address as HeadAddress
from projects proj;
```

Q4: Transform the last query above into a relational query considering the reference of tables will become ID when creating the table and you need to do the join between tables rather than simply using "->".

Example of "reference of tables will become ID when creating the table":

```
create type Department (
    dept_name varchar(20),
    head ref(Person) scope people);
```

Will become

```
create type Department (
    dept_name varchar(20),
    people_id varchar(20) );
```