Assignment 3

Qinglin Li, 5110309074

Problem 1

- **Atomicity**: Each transaction is "all or nothing"—if one part of the transaction fails, the entire transaction fails, and the database state is left unchanged.
  Example: Assume that a transaction attempts to subtract 10 from A and add 10 to B. If after removing 10 from A, the system crashes. Then when the system recovers, either A remains the old value or 10 is added to B.

- **Consistency**: Any transaction will bring the database from one valid state to another. Any data written to the database must be valid according to all defined rules.
  Example: Assume in a table there is a contraint requires the sum of A and B to be 100. If a transaction attempts to subtract 10 from A without modifying B, the transaction should not be commited.

- **Isolation**: The concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially.
  Example: Assume two transactions T1 and T2 both attempts to subtract 10 from A and add 10 to B. The order of two transactions should either be

  1. T1 subtracts 10 from A.
  2. T1 adds 10 to B.
  3. T2 subtracts 10 from B.
  4. T2 adds 10 to A.

  or

  1. T2 subtracts 10 from A.
  2. T2 adds 10 to B.
  3. T1 subtracts 10 from B.
  4. T1 adds 10 to A.

- **Durability**: Transactions that have committed will survive permanently.
  Example: A transaction is commited, but when the changes are still queued in the disk buffer waiting to be commited to the disk, the power fails. Then after the system recovers, the uncommited changes should be commited to the disk.

Problem 2

```
create table loan(
    loan_number char(10),
    branch_name char(15),
    amount integer,
    primary key (loan_number),
    foreign key (branch_name) references branch
)

create table borrower(
    customer_name char(20),
    loan_number char(10),
    primary key (customer_name, loan_number),
    foreign key (customer_name) references customer,
    foreign key (loan_number) references loan
)
```

Problem 3

a. **foreign key** (name) **references** salaried_workder or hourly_worker

b. use general assertions

```
create assertion name_ref
check(
    not exists (
        select * from address
        where name not in (
            select name from salaried_workder
            union
            select name from hourly_worker
        )
    )
)
```

Problem 4

a.
```
select company from works
group by company
having avg(salary) > (
    select avg(salary) from works
    where company = 'First Bank Corporation'
)
```

b. It should work with MySQL

```
delimiter //
create function avg_salary (company_name varchar(20))
returns double
begin
    declare ret double;
    select avg(salary) into ret from works where company = company_name;
    return ret;
end;//
delimiter ;
select name from works
where avg_salary(name) > avg_salary('First Bank Corporation');
```

Problem 5

a. $\{t|\exists p \in r(t[A] = p[A])\}$

b. $\{t|t \in r \wedge t[B] = 17\}$

c. $\{t|\exists p \in r, \exists q \in s(t[A] = p[A] \wedge t[B] = p[B] \wedge t[C] = p[C] \wedge t[D] = q[D] \wedge t[E] = q[E] \wedge t[F] = q[F])\}$

d. $\{t|\exists p \in r, \exists q \in s(t[A] = p[A] \wedge t[F] = q[F] \wedge p[C] = q[D]\}$

Problem 6

a. $\Pi_A(\sigma_{B=17}(r))$

b. $r \bowtie s$

c. $\Pi_A(r) \bigcup (s \div \Pi_C(s))$

d. $\Pi_A\big(\sigma_{B>D}(r \bowtie s \bowtie \rho_{t(C,D)}(r))\big)$

Problem 7

a. $\{t|\exists r \in R \exists s \in S(r[A] = s[A] \wedge t[A] = r[A] \wedge t[B] = r[B] \wedge t[C] = s[C]) \vee \exists s \in S(\neg \exists r \in R(r[A] = s[A]) \wedge t[A] = s[A] \wedge t[C] = s[C] \wedge t[B] = null)\}$

b. $\{t|\exists r \in R \exists s \in S(r[A] = s[A] \wedge t[A] = r[A] \wedge t[B] = r[B] \wedge t[C] = s[C]) \vee \exists r \in R(\neg \exists s \in S(r[A] = s[A]) \wedge t[A] = r[A] \wedge t[B] = r[B] \wedge t[C] = null) \vee \exists s \in S(\neg \exists r \in R(r[A] = s[A]) \wedge t[A] = s[A] \wedge t[C] = s[C] \wedge t[B] = null)\}$

c. $\{t|\exists r \in R \exists s \in S(r[A] = s[A] \wedge t[A] = r[A] \wedge t[B] = r[B] \wedge t[C] = s[C]) \vee \exists r \in R(\neg \exists s \in S(r[A] = s[A]) \wedge t[A] = r[A] \wedge t[B] = r[B] \wedge t[C] = null)\}$