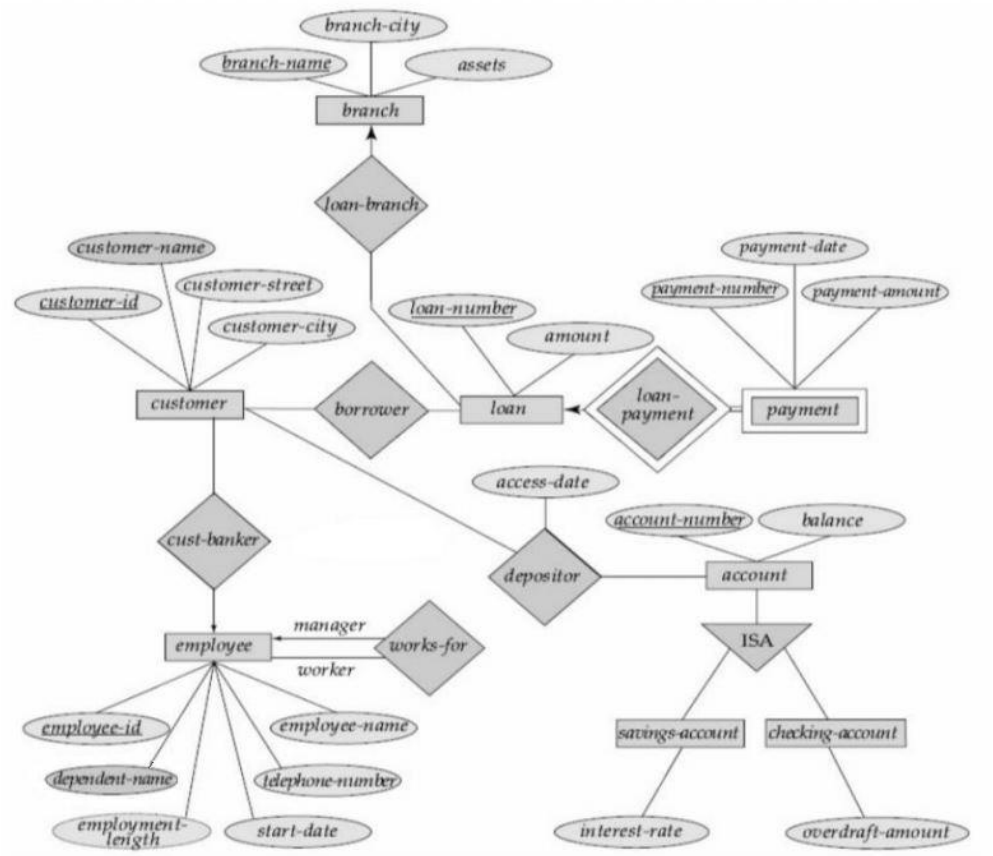
Amirkabir University of Technology (AUT)
Tehran, Iran,
Lab Database, Computer Engineering Department
Tina Gholami (9531307) Report 6

Amirkabir University of Technology
(Tehran Polytechnic)

Question 1)

۱- جداول نمودار ERD زیر را ایجاد کنید، تبدیل هر entity به جدول را توضیح دهید.



The above ERD diagram contains 8 tables. The many-to-one, one-to-many, and one-to-one relations are integrated into the many-table (for instance, the “cust-banker” relation will be an attribute in the “Customer” table). On the other hand, the many-to-many relations (Depositor and Borrower) cannot be embedded into any of their connecting table, but should be considered separate tables themselves. Here are the tables that were created in the database:

```sql
create table Branch(
    branch_name varchar(50) primary key,
    branch_city varchar(50),
    assets int
);

create table Loan(
    loan_number int primary key,
    amount int,
    branch_name varchar(50),
    foreign key (branch_name) references Branch(branch_name)
);

create table Payment(
    payment_number int primary key,
    payment_date date,
    payment_amount int,
    loan_number int,
    foreign key (loan_number) references Loan(loan_number)
);

create table Employee(
    employee_id int primary key,
    manager_id int,
    employee_name varchar(50),
    dependent_name varchar(50),
    employment_length int,
    starting_date date,
    telephone_number int,
    foreign key (manager_id) references Employee(employee_id)
);

create table Customer(
    customer_id int primary key,
    employee_id int,
    customer_name varchar(50),
    customer_street varchar(50),
    customer_city varchar(50),
    foreign key (employee_id) references Employee(employee_id)
);

create table Borrower(
    customer_id int,
    loan_number int,
    foreign key (customer_id) references Customer(customer_id),
    foreign key (loan_number) references Loan(loan_number)
);

create table Account(
    account_number int primary key,
    balance int,
    interest_rate int,
    overdraft_amount int
);

create table Depositor(
    customer_id int,
    account_number int,
    access_date date,
    foreign key (customer_id) references Customer(customer_id),
    foreign key (account_number) references Account(account_number)
);
```

1- پرس و جوهای زیر را با استفاده از T_SQL نوشته و عکس پرس‌وجو را به همراه نتیجه در گزارشکار بیاورید.

الف)view بنویسید که نام و id مشتری را به همراه میزان وام دریافتی و شعبه‌ی دریافتی او نشان دهد.

ب) شماره حساب و میزان سود این حساب‌ها در صورتی که زمان واریز بعد از سال ۲۰۰۹ باشد.

ج) شماره‌ی بازپرداخت وام‌هایی که شعبه‌ی آن‌ها تهران است.

Part (a):

First, data is inserted into some of the tables in order to have the output for the queries:

```
insert into branch values('Branch Finance', 'London', 5000000);
insert into branch values('Branch House', 'Berlin', 10000000);
insert into branch values('Branch Car', 'Toronto', 1000000);

insert into Employee values(1, null, 'Kim Mng', 'Ryan', 30, '1990-03-03', 1111111111);
insert into Employee values(2, 1, 'Sam Emp', 'Ryan', 3, '2018-03-03', 1111111111);
insert into Employee values(3, 1, 'David Emp', 'Ryan', 5, '2016-03-03', 1111111111);

insert into Customer values(1, 2, 'Martin Lam', '1st Rose Avenue', 'London');
insert into Customer values(2, 3, 'Amy Clam', 'Richmond Street', 'Berlin');

insert into Loan values(101, 230000, 'Branch Finance');
insert into Loan values(102, 510000, 'Branch House');

insert into Borrower values(1, 101);
insert into Borrower values(2, 102);
```

Here is the query for creating the view in part (a):

```
----part (a):
create view v1 as
select Customer.customer_name, Customer.customer_id, Loan.amount, Loan.branch_name
from Customer, Loan, Borrower
where Borrower.customer_id = Customer.customer_id and Borrower.loan_number = Loan.loan_number;
```

And here is the result as expected: (costumer name + customer id + loan amount + branch name)

```
select * from v1
```

	customer_name	customer_id	amount	branch_name
1	Martin Lam	1	230000	Branch Finance
2	Amy Clam	2	510000	Branch House

Part (b):

First, data is inserted into some of the tables in order to have the output for the queries:

```
insert into Account values(1001, 20000, 10, 100);
insert into Account values(1002, 30000, 12, 200);

insert into Depositor values(1, 1001, '2005-09-21');
insert into Depositor values(2, 1002, '2012-04-04');
```

Here is the query to get (account number + interest rate) of the deposits that were made after 2009:

```
select Account.account_number, Account.interest_rate
from Account, Depositor
where Account.account_number = Depositor.account_number and YEAR(access_date) > 2009;
```

As seen above, in the output, we should only have the Depositor(2, 1002, '2012-04-04') for the Account(1002, 30000, 12, 200):

	account_number	interest_rate
1	1002	12

Part (c):

First, data is inserted into some of the tables in order to have the output for the queries:

```
insert into Branch values('Branch Health', 'Tehran', 9000000);

insert into Loan values(103, 54000, 'Branch Health');

insert into Payment values(1, '2020-11-08', 4000, 103);
insert into Payment values(2, '2020-12-08', 8000, 103);
insert into Payment values(3, '2020-12-10', 5000, 102);
```

And here is how the table Loan looks like after inserting some new data:

	loan_number	amount	branch_name
1	101	230000	Branch Finance
2	102	510000	Branch House
3	103	54000	Branch Health

And here is how the table Branch looks like after inserting some new data:

	branch_name	branch_city	assets
1	Branch Car	Toronto	1000000
2	Branch Finance	London	5000000
3	Branch Health	Tehran	9000000
4	Branch House	Berlin	10000000

And here is how the table Payment looks like after inserting some new data:

	payment_number	payment_date	payment_amount	loan_number
1	1	2020-11-08	4000	103
2	2	2020-12-08	8000	103
3	3	2020-12-10	5000	102

The query is as follows:

```
select payment_number
from Loan, Branch, Payment
where Payment.loan_number = Loan.loan_number and Loan.branch_name = Branch.branch_name and branch_city = 'Tehran';
```

And here is the output (payment number) for an account based in branch ‘Tehran’ (which according to the above tables and inserts, should be 1 and 2):

	payment_number
1	1
2	2

Question 2)

۲- رویه‌های زیر را ایجاد کنید:

الف) روالی بنویسید که نام مراجعه‌کننده به بانک را به عنوان ورودی بگیرد و balance و شماره حساب را به عنوان خروجی برگرداند.

ب) روالی بنویسید که Payment_number را به عنوان ورودی بگیرد و نام شعبه‌ای را که از آن پرداخت صورت گرفته برگرداند.

ج)روالی بنویسید که شماره‌ی مراجعه‌کننده را به عنوان ورودی گرفته و پس از ۱۰ ثانیه اطلاعات مراجعه‌کننده را نمایش دهد.

Part (a):

Here is the query for the procedure:

```
--part (a):
create procedure p1
@name varchar(50), @balance int output, @acc_num int output
as
select @balance = balance
from Account, Depositor, Customer
where Account.account_number = Depositor.account_number and Depositor.customer_id = Customer.customer_id and @name = customer_name;

select @acc_num = Account.account_number
from Account, Depositor, Customer
where Account.account_number = Depositor.account_number and Depositor.customer_id = Customer.customer_id and @name = customer_name;
```

And here is how we produce the output:

```
declare @balance_temp int, @acc_num_temp int;
exec p1
@name = 'Martin Lam',
@balance = @balance_temp output,
@acc_num = @acc_num_temp output;
print @balance_temp;
print @acc_num_temp;
```

This is the output:

```
20000
1001

Completion time: 2021-05-02T22:02:02.2187874+04:30
```

P.S: To make sure this is correct, I did the following: (And as can be seen, ‘Martin Lam’ has balance 20000 for the account number = 1001)

```
select * from Depositor;
select * from Customer;
select * from Account;
```

	customer_id	account_number	access_date
1	1	1001	2005-09-21
2	2	1002	2012-04-04

	customer_id	employee_id	customer_name	customer_street	customer_city
1	1	2	Martin Lam	1st Rose Avenue	London
2	2	3	Amy Clam	Richmond Street	Berlin

	account_number	balance	interest_rate	overdraft_amount
1	1001	20000	10	100
2	1002	30000	12	200

So, the former query has given the correct output.

Part (b):

Here is the query for the procedure:

```
----part (b):
create procedure p2
@pay_num int, @b_name varchar(50) output
as
select @b_name = branch_name
from Payment, Loan
where Payment.loan_number = Loan.loan_number;
```

And here is how we produce the output:

```
declare @b_name_temp varchar(50);
exec p2
@pay_num = 3,
@b_name = @b_name_temp output;
print  @b_name_temp;
```

This is the output:

```
Branch House

Completion time: 2021-05-03T00:50:47.8441732+04:30
```

P.S: To make sure this is correct, I did the following: (And as can be seen, payment number = 3 is related to branch 'Branch House')

```
select * from Payment;
select * from Loan;
```

	payment_number	payment_date	payment_amount	loan_number
1	1	2020-11-08	4000	103
2	2	2020-12-08	8000	103
3	3	2020-12-10	5000	102

	loan_number	amount	branch_name
1	101	230000	Branch Finance
2	102	510000	Branch House
3	103	54000	Branch Health

Part (c):

Here is the query for the procedure:

```
create procedure p3
@cust_num int
as
WAITFOR delay '000:00:10';
select * from Customer
where customer_id = @cust_num;
```

And here is how we produce the output:

```
exec p3
@cust_num = 2;
```

This is the output after 10 seconds:

	customer_id	employee_id	customer_name	customer_street	customer_city
1	2	3	Amy Clam	Richmond Street	Berlin

P.S: To make sure this is correct, I did the following: (printing the ‘Customer’ table to make sure the info is correct)

```
select * from Customer;
```

	customer_id	employee_id	customer_name	customer_street	customer_city
1	1	2	Martin Lam	1st Rose Avenue	London
2	2	3	Amy Clam	Richmond Street	Berlin

As can be seen above, the output that was previously yielded, is legit.

Question 3)

۳- توابع زیر را ایجاد کنید:

الف)تابعی بنویسید که شماره‌حساب، حسابی را که بیش‌ترین سود را داشته برگرداند.

ب) تابعی بنویسید که employee_id را به عنوان ورودی بگیرد و نام دپارتمانی را که کارمند در آن کار می‌کند به عنوان خروجی برگرداند.

Part (a):

The query for the function is:

```
----part (a):
create function max_rate()
returns int
as
begin

    declare @num int;
    select @num = account_number
    from Account
    where interest_rate = (select MAX(interest_rate)
    from Account);
    return @num;

end
```

And here is how we call the function:

```
declare @acc_num_temp int;
exec @acc_num_temp = max_rate;
print(@acc_num_temp);
```

And the output is:

```
1002

Completion time: 2021-05-03T02:10:42.2036045+04:30
```

P.S: Here I have printed out the whole ‘Account’ table to see what is the account number for the account with the maximum interest rate: (As can be seen, the max interest rate is for account number = 1002. So the yielded output above is legit.)

```
select * from Account;
```

	account_number	balance	interest_rate	overdraft_amount
1	1001	20000	10	100
2	1002	30000	12	200

Part (b):

(In the description file, there was a typo and the department name was wrongly spelled as "depending_name" in page 74. Therefore, I am also going to use "depending_name" for the department name.)

So, the query for the function is:

```
----part (b):
create function give_department(@emp_id int)
returns varchar(50)
as
begin
    declare @dept_name varchar(50);
    select @dept_name = dependent_name
    from Employee
    where employee_id = @emp_id;
    return @dept_name;
end
```

And here is how to call the function to get the department name of an employee with id = 3:

```
declare @dept_name_temp varchar(50);
exec @dept_name_temp = give_department @emp_id = 3;
print @dept_name_temp;
```

And here is the output:

```
Ryan

Completion time: 2021-05-03T02:30:16.6738131+04:30
```

P.S: Just to make sure the above result is legit, I have printed out table "Employee" and here we can see the department name for an employee with id = 3 is the 'Ryan' department.

```
select * from Employee;
```

	employee_id	manager_id	employee_name	dependent_name	employment_length	starting_date	telephone_number
1	1	NULL	Kim Mng	Ryan	30	1990-03-03	1111111111
2	2	1	Sam Emp	Ryan	3	2018-03-03	1111111111
3	3	1	David Emp	Ryan	5	2016-03-03	1111111111

Question 4)

۴- آغازگرهای زیر را بنویسید:

الف) آغازگری بنویسید که از جدول payment، log ایجاد کند.

ب) آغاز گری بنویسید که مانع از update شدن فیلد branch_name در جدول Branch شود.

Part (a):

Here is the query for creating the log table and trigger t1:

```
--part (a):
create table payment_log(
    payment_number int primary key,
    payment_date date,
    payment_amount int,
    loan_number int,
    foreign key (loan_number) references Loan(loan_number)
);


create trigger t1 on Payment
after insert as
declare @p_num int, @p_date date, @p_amount int, @l_num int;
select @p_num = payment_number, @p_date = payment_date, @p_amount = payment_amount, @l_num = loan_number
from inserted;
insert into payment_log values(@p_num, @p_date, @p_amount, @l_num);
```

And then, new data is inserted to table “Payment”. Here is the result of the table “payment_log” after insertion:

```
insert into Payment values(4, '2021-01-01', 3000, 102);
select * from payment_log;
```

The output is:

	payment_number	payment_date	payment_amount	loan_number
1	4	2021-01-01	3000	102

Part (b):

Here is the query to avoid any updates on the “branch_name” in the table “Branch”:

```
--part (b):
create trigger t2 on Branch
after update as
if UPDATE(branch_name)
begin
    print('Hi! You are not allowed to upfdate the name of the branch :(');
    ROLLBACK TRANSACTION;
end;
```

Firstly, here is the “Payment” table before updating the name of the branch:

```
select * from Branch;
```

	branch_name	branch_city	assets
1	Branch Car	Toronto	1000000
2	Branch Finance	London	5000000
3	Branch Health	Tehran	9000000
4	Branch House	Berlin	10000000

Secondly, here is what will happen as the result of the query if we tend to update the branch name from the “Branch” table:

```
update Branch
set branch_name = 'Random!'
where branch_city = 'Toronto';
```

```
Hi! You are not allowed to upfdate the name of the branch :(
Msg 3609, Level 16, State 1, Line 24
The transaction ended in the trigger. The batch has been aborted.

Completion time: 2021-05-03T03:20:14.0349444+04:30
```

Hence, the query is doing right.