DATABASE EXAM

Konare Trade Bank(KTB)

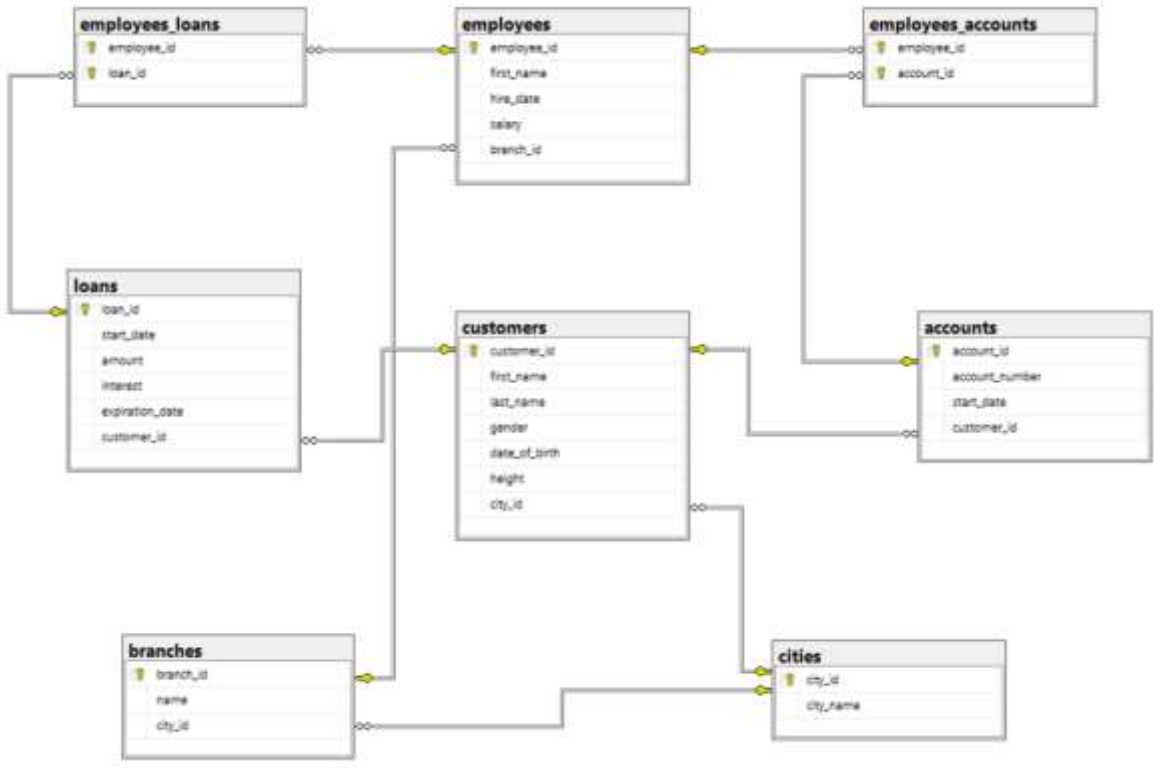
After months of hard work, you finally made it! You have started to work for Konare Trade Bank(KTB) as a database developer. It is so unique that you have the feeling that there is no bank like this. In order to get the complete amount of your precious salary at the end of the month you have to complete **4 simple sections**. But that is not all – there is a sweet bonus if you are good enough to finish section 5. Here are your tasks:

Section 1. DDL

25 pts

For this section put your queries in judge and use SQL Server run skeleton, run queries and check DB.

You have been given the E/R Diagram of the bank:



In KTB we have Customers who have 0, 1 or many Loans and Accounts. Moreover, we have employees. Employees are responsible for the maintenance of the loans and the accounts. Many

employees have responsibilities over many loans and accounts. Furthermore, Employees are located in different branches. Branches are located in cities just like our customers are.

Since the business is blooming you have to extend the following diagram with 6 tables and to modify one of the existing tables. Make sure you have done appropriately the following specifics:

- Primary Keys
- Foreign Keys
- Data Types
- 6 Tables
- Create the constraint within the CREATE TABLE Statement
- 1 Table Modification

Here are the tables that you need to create:

The bank is about to start offering deposits. That is why we need few tables. Every customer has 1, 0 or many deposits. Every deposit has 0 or 1 one deposit types. Many employees will be responsible for the maintenance of the deposits. That's why we would need a mapping table. Customers now have credit history. Every customer can have many different credit marks. Our customers have started to pay their loans. Therefore, we need a table Payments. Every loan can have 0, 1 or many payments. KTB has a new web site and we would need another table for users. Every customer has exactly one user. Here are some details about the tables we need.

deposits

Column Name	Data Type	Constraints
deposit_id	Integer from –2,147,483,648 to 2,147,483,647	Unique table identificator, Identity
amount	Decimal with length of 10, 2 digits after the decimal point	
start_date	Date without time	
end_date	Date without time	
deposit_type_id	Integer from –2,147,483,648 to 2,147,483,647	Relationship with deposit_types
customer_id	Integer from–2,147,483,648 to 2,147,483,647	Relationship with table Customers

deposit_types

Column Name	Data Type	Constraints
deposit_type_id	Integer from –2,147,483,648 to 2,147,483,647	Unique table identificator
name	String up to 20 symbols	

employees_deposits

Column Name	Data Type	Constraints
employee_id	Integer from –2,147,483,648 to 2,147,483,647	Unique table identificator; Relationship with table Employees
deposit_id	Integer from –2,147,483,648 to 2,147,483,647	Unique table identificator; Relationship with table Deposits

credit_history

Column Name	Data Type	Constraints
credit_history_id	Integer from –2,147,483,648 to 2,147,483,647	Unique table identificator
mark	Character with 1 symbol	
start_date	Date without time	
end_date	Date without time	
customer_id	Integer from –2,147,483,648 to 2,147,483,647	Relationship with table Customers

payments

Column Name	Data Type	Constraints
payement_id	Integer from –2,147,483,648 to 2,147,483,647	Unique table identificator
date	Date without time	
amount	Decimal with length of 10, 2 digits after the decimal point	
loan_id	Integer from –2,147,483,648 to 2,147,483,647	Relationship with table Loans

users

Column Name	Data Type	Constraints
user_id	Integer from –2,147,483,648 to 2,147,483,647	Unique table identificator
user_name	String up to 20 symbols	
password	String up to 20 symbols	
customer_id	Integer from –2,147,483,648 to 2,147,483,647	Relationship with table Customers, Unique Values

Modify table **employees.** Add column manager_id

employees

Column Name	Data Type	Constraints
….		
manager_id	Integer from –2,147,483,648 to 2,147,483,647	Relationship with table employees

Put all of your queries that solve Section 1. In judge use SQL Server run skeleton, run queries and check DB.

Section 2. DML

20 pts

For this section put your queries in judge and use SQL Server run skeleton, run queries and check DB.

In this section you have to do couple of data manipulations:

1. Insert data into the following tables

employees_deposits

employee_id	deposit_id
15	4
20	15
8	7
4	8
3	13
3	8
4	10
10	1
13	4
14	9

deposit_types

deposit_type_id	name
1	Time Deposit
2	Call Deposit
3	Free Deposit

deposits

Add all customers with ID lower than 20 to table Deposits.

- The id should be auto generated from table deposits
- The deposit amount is based on the following logic
    - If you are born after 01-01-1980 then it is 1000 BGN else, it is 1500 BGN
    - If you are Male add 100 BGN else add 200 BGN
- The start date should be the current date
- The end should be empty

- The deposit type id should 1 if the customer id is odd and 2 if the customer id is even. If the customer id is larger than 15 then the deposit type should be 3.
- The customer id should come from table Customers

2. Update Employees

Update table Employees. The manager id should have the following values:

- If EmployeeID is in the range [2;10] then the value is 1
- If EmployeeID is in the range [12;20] then the value is 11
- If EmployeeID is in the range [22;30] then the value is 21
- If EmployeeID is in 11 or 21 then 1

3. Delete Records

Delete all records from EmployeeDeposits if the DepositID is 9 or the EmployeeID is 3;

Section 3. Querying

40 pts

For this section put your queries in judge and use SQL Server prepare DB and run queries.

1. Employees' Salary

Write a query that returns

- employee_id
- hire_date
- salary
- branch_id

From table Employees. Filter employees which salaries are higher than 2000 and their hire date is after 15/06/2009.

Example

employee_id	hire_date	salary	branch_id
5	2009-12-23	2295.88	12

2. Customer Age

Write a query that returns

- `first_name`
- `date_of_birth`
- `age`

of all customers who are between 40 and 50 years old. The range is inclusive. Consider that today is 01-10-2016. Assume that each year has 360 days.

Example

first_name	date_of_birth	age
Bruce	1970-09-17	46

3. Customer City

Write a query that returns

- customer_id
- first_name
- last_name
- gender
- city_name

for all customers whose last name starts with 'Bu' or first name ends with 'a'. Moreover, for those customers the length of the city name should at least 8 letters.

Example

customer_id	first_name	Last_name	gender	city_name
25	Debra	Crawford	F	Povorino

4. Employee Accounts

Write a query that returns top 5

- `employee_id`
- `first_name`
- `account_number`

of all employees who are responsible for maintaining accounts which has started after the year of 2012. Sort the results by the first name of the employees in descending order.

Example

employee_id	first_name	account_number
26	William	501012430845

5. Employee Cities

Write a query that returns

- `city_name`
- `name` (of the branch)
- employees_count

The count of all employees grouped by city name and branch name. Exclude cities with id 4 and 5. Don't show groups with less than 3 employees.

Example

city_name	name	employees_count
Budapest	Angela	3

6. Loan Statistics

Write a query that returns

- `The total amount of loans`
- Max interest of loans
- Min salary of employees

Of all employees that are responsible for maintaining the loans.

Example

total_loan_amount	max_interest	min_employee_salary
1469561.30	0.91	1639.97

7. Unite People

Write a query that returns top 3 employees' first names and the city name of their branch followed by top 3 customer's first names and the name of the city they live in.

Example

first_name	city_name
Sarah	Uppsala

8. Customers without Accounts

Write a query that returns

- `customer_id`
- height

of all customers who doesn't have accounts. Filter only those who are tall between 1.74 and 2.04.

Example

customer_id	height
11	1.78

9. Customers without Accounts

Write a query that returns top 5 rows

- `customer_id`
- amount

of all customers who have loans higher than the average loan amount of the male customers. Sort the data by customer last name in ascending order.

Example

customer_id	amount
4	88067.24

10. Oldest Account

Write a query that returns

- `customer_id`
- `first_name`
- start_date

for the customer with the oldest account.

Example

customer_id	first_name	start_date
27	Howard	2010-10-13

Section 4. Programmability

20 pts

For this section put your queries in judge and use SQL Server run skeleton, run queries and check DB.

1. String Joiner Function

Write a function with name **udf_concat_string** that reverses two strings, joins them and returns the concatenation. The function should have two input parameters of type VARCHAR.

2. Unexpired Loans Procedure

Write a procedure that returns a customer if it has unexpired loan. The following result set should be returned:

- customer_id
- first_name
- loan_id

The function should have one parameter for customer_id of type integer. Name the function **usp_customers_with_unexpired_loans.** If the id of the customer doesn't have unexpired loans return an empty result set.

Example

```
CALL usp_customers_with_unexpired_loans(9)
```

CustimerID	FirstName	LoanID
9	Bobby	23

3. Take Loan Procedure

Write a procedure that adds a loan to an existing customer. The procedure should have the following input parameters:

- customer_id
- loan_amount
- interest
- start_date

If the loan amount is not between 0.01 AND 100000 raise an error 'Invalid Loan Amount.' And rollback the transaction.

If no error is raised insert the loan into table Loans. The column loan_id has an auto_increment property so there is no need to specify a value for it.

Name the procedure **usp_take_loan.**

Example

```
CALL usp_take_loan (1, 500, 1,'20160915')
```

```
One row added.
```

4. Trigger Hire Employee

Write a trigger on table Employees. After an insert of a new employee the new employee takes the loan maintenance of the previous employee.

Hint

Your trigger should update table employees_loans

Example

Before Insert

employee_id	first_name	hire_date	salary	branch_id
30	Diane	2006-03-18	2574.01	6

employee_id	loan_id
30	7

```
INSERT INTO employees values (31,' Jake ','20161212',500,2)
```

After Insert

employee_id	first_name	hire_date	salary	branch_id
30	Diane	2006-03-18	2574.01	6
31	Jake	2016-12-12	500	2

employee_id	loan_id
31	7

Section 5. Bonus

10 pts

For this section put your queries in judge and use SQL Server run skeleton, run queries and check DB.

1. Delete Trigger

Create a table with the same structure as table accounts and name it **account_logs**. Then create a trigger that logs the deleted records from table accounts into table **account_logs**. Post in judge only the create trigger statement.

Example

```
DELETE FROM accounts
WHERE customer_id = 4
```

account_logs

account_id	account_number	start_date	customer_id
31	352806149112	2016-08-05	4