COSC 641

Spring 2021

FSU

# Blockchain Finance

# Database Final Project COSC 641 Part 1

You are employed to work on an existing project. This new financial organization (bank) is based on Bitcoin. The base tables and some data are already provided for you. The ER diagram and structure of tables are in a supplemental document.

Please number your chapters correctly. I grade one chapter at a time.

The data and based tables are under PROJECT2020 account. You can access each table by using PROJECT2020.table\_name. You can copy each table into your Oracle account like:

CREATE TABLE branch

AS

SELECT \*

FROM PROJECT2020.branch;

Check your data:

SELECT \*

FROM branch;

Check the attributes:

DESC branch;

Note 1: Frequently it is required to make changes to the base tables (add or delete fields). You can use ALTER TABLE. To add/delete fields to your tables. The added fields will have value of NULL for now.

Note 2: All of the data must be in your based tables. We will not create FORCED view.

Note 3: For complex queries you can create intermediate views and create view from view.

Note 4: We will use the views and sequences later on in the project. For now, just create them.

Note 4: The data in the tables are test data. By no means they are complete. You welcome to add additional records to the tables.

Please try to keep the names tables as close to the original names as you can. If there is any missing data, feel free to add it to your tables.

List of tables are:

1. BRANCH
2. DEPARTMENT
3. CAR
4. DRIVER (emp drivers’ car)
5. CUSTOMER
6. BANK\_EMPLOYEE
7. JOB
8. EMP\_ANNUAL\_DATA
9. BRANCH\_EMPLOYEE (emp work at branch)
10. ATM
11. BRANCH\_ACCESS\_POINTS (branch has access points)
12. BRANCH\_MANAGER (emp manager branch)
13. LOAN
14. LOAN\_PROJECT
15. DEPOSIT\_ACC
16. DEPOSIT\_ACC\_PRODUCT
17. CD\_ACCOUNT
18. CD\_PRODUCT
19. CREDIT\_ACCOUNT
20. CREDIT\_PRODUCT
21. LOAN\_PAYMENT
22. DEPOSIT\_ACCT\_TRANSACTION
23. CREDIT\_ACCT\_TRASACTION

**Chapter 1**-Create the following views: Please show you **code** (query) and **result** of running your code. Such as:

1a:

SELECT \*

FROM EMPLOYEE\_DATA;

a. Employee\_data with the following attributes:

|  |
| --- |
| Name of Employee (first, middle, last) |
| Address |
| Zip code of Employee Address |
| SSN |
| Title |
| Current Year |
| Current Yearly Salary |
| Current Tax Deduction Rate |
| The date s/he was employed at the Current Branch |
| Birth Date |
| Age of Employee |
| Employee Branch Phone Extension s/he Works at |
| Branch Phone Number |
| Branch Name (Employee Works at) |
| Highest Degree |
| Highest Degree date |

b. Employee\_salary with the following attributes:

|  |
| --- |
| Name |
| Current Year |
| SSN |
| Current Salary |
| Branch Employee Works at |
| Total Cost of Employee Salaries at the branch s/he works |
| Highest salary at his/her branch |
| Average salary at his/her branch |

c. Branch\_data with the following attributes:

|  |
| --- |
| Branch ID |
| Branch Name |
| Address |
| Phone Number |
| Fax Number 🡨 to be inputted later |
| Number of Employee at this Branch |
| Category |
| Manager Name |
| Total Transactions Done at this Branch for year 2020 |

d. Valued\_Customers with the following attributes:

Describe how you select the important customers in this bank.

|  |
| --- |
| SSN |
| Name |
| Age |
| Home Phone |
| Work Phone |
| Address |
| Zip Code |
| Email 🡨 to be inputted later |
| State they live in |
| Total number of tractions the customer has done in a given year |
| Total amount (in bitcoin) of tractions the customer has done in a given year (you choose the year) |

e. Statistics\_by\_Branch with the following attributes (Read only view):

|  |
| --- |
| Branch Id |
| Branch name |
| Year |
| Total deposit in that year for this location |
| Total number of transactions |
| Total number of employees at this branch |

f, g-Create two more views that can be used by **customers**. (Make sure it is useful to the customers. You will be graded based on the usefulness of the views)

h, i-Create two more views that can be used by **management**. (Make sure it is useful for managerial decisions)

**Chapter 2**

a- Create a sequence called ID\_generator to be used for Account ID.

Start with 1111

Generate only odd numbers for security

Cache 50 numbers at a time

b- Create a sequence to be used for the Transaction ID. (Make your own assumption).

COSC 641

Spring 2021

FSU

# Blockchain Finance

# Database Final Project COSC 641 Part 2

Continuation of our project. Please number each sub-section correctly. Like 3.A, 3.B … . **Please show your code.**

**Chapter 3**

**Write Subprograms with exception handling: (make sure your subprograms have appropriate exception handling).**

1. Procedure to transfer $x from one account to another account. You will pass the amount of transfer, and two account numbers. Also, write the FROM account number, to TO account number, the amount and the date of transaction into a Transaction\_Log file (Create this table). Show your work. You may use CREDIT\_ACCOUNT table for this procedure.
2. Create a subprogram called **Birthday\_sub** that accepts today’s date as default and writes the first name, last name, and address of a customer whose birth date (day and month only) is 15 days from today’s date. Write them into a file (create a table called B\_C\_File). You may add additional records to your customer table.
3. Procedure to write all daily (today) transactions (credits and debits) into a database table called **Today\_Transaction.** (all deposits and withdraw for that day). You may use CREDIT\_ACCT\_TRANSACTION table
   1. Use SYSDATE to get today’s date.
   2. The structure of Today\_Transaction is:
      1. (Date & Time, Account number, Account Type, Amount, deposit/withdraw)
4. Create a function called CustomerInfo to accept a customer account number and return the total deposit for the customer account.
5. Create a function with the same name CustomerInfo (overload) to accept a customer account number and a date; and return the total deposit of that customer account for that date.
6. Create a function with the same name CustomerInfo (overload) to accept a customer ID, a date and a co-owner account number, and return the total deposit of that join account for that date.
7. Procedure to list the last 10 transactions of a customer by passing Customer Id.
8. Create a subprogram to accept a customer ID and output name (first, mid, and last), loan no for this customer, and amount of loan. (a customer may have more than one loan.
9. Create a subprogram to write the following information for each employee into a database called Emp\_list only for employees that older than 30 years old. Name (first, mid, and last), address, date of birth, and their salary.

**Chapter 4:**

**Create the following packages with exception handling: Please show you code.**

1. Create a package with the following functions called **BankP** for each customer (customer id):
2. Function to return the current balance.
3. Function to return the last deposit from checking.
4. Function to return the last deposit from saving
5. Function to return the last withdraw from checking.
6. Function to return the last withdraw from saving.
7. Create a package with the following functions & procedures called **BranchP** by branch id.
8. Function to return the current branch address.
9. Function to return the current branch phone number.
10. Procedure to output the name of employees working at that branch.
11. Create a package called **Insert\_pkg** with subprograms to automate the insert, delete, and update of data in your database.
    * 1. Subprogram to insert a row in table customer (make sure you use the sequence for generating ID’s)
      2. Subprogram to insert a row in table bank employee
      3. Subprogram to insert a row in table CD
      4. Subprogram to insert a row in table driver.
      5. Subprogram to delete a row from table driver.
      6. Subprogram to update a row in table driver.
12. Create a useful package of your choice with functions, procedures, and datatype of your own.

**Chapter 5:**

**Create the following triggers:**

1. Any deletion from employee file, trigger to write OLD attributes into an **Employee\_History** file.
2. Any modification to the customer accounts, write who, date, and the nature of (OLD and NEW) modification into a **Cust\_Mod\_Log** file.
3. Any deposit larger than $5000 to any account will also be written in a **Large\_Dep\_Log** file.
4. Any withdraw larger than $10000 from any account also will be written in a **Large\_With\_Log** file.

**Chapter 6:**

All the errors generated by your run-time programs must be stored into an Error\_log file. We would like to know the name of table, trigger name, the date, trigger event, and the trigger type.

**Grading:**

Chapter 1: 9 views each 7 points= 63

Chapter 2: 2 sequences each 5 points= 10

Chapter 3: 9 subprograms each 10 points= 90

Chapter 4: part A 5\*3 15

part B 3\*3 9

part C 6\*3 18

part D 7

Chapter 5: 4 triggers each 7 points 28

Chapter 6: Error 10

250