# Introduction

The aim of this project is to maintain the customer information in a banking system. Banking is one of the most synchronized systems in the world. In this project, a customer can create an account in the bank and keep track of personal information such as balance. Both bank and customer can view bank details and customer details.

A database named [BankingDatabaseSystem] has been created.

CREATE DATABASE [BankingDatabaseSystem]

GO

# Diagrammatic Representation – Database Diagram

Diagram

Description automatically generated

Entity**:** An entity is a thing that is either physical or logical.

Relationship: It describes how entities are correlated with each other.

The above entity relationship diagram describes the relationship between 4 entities.

|  |  |
| --- | --- |
| **Entity** | **Attributes** |
| Customer | CustomerID, CustomerName, Contact, BankID |
| Account | AccountID, CustomerID, AccountTypeID, AccountNo, Balance |
| Bank | BankID, BankName, Address |
| AccountType | AccountTypeID, AccountTypeName |

## Relationship

* Bank – Customer – One to Many (One bank can have many customers)
* Customer – Account – One to Many (One customer can have many accounts)
* Account – AccountType – One to One (One account can be of one account type)

# Normalization

The process of eliminating duplicate data from the tables and maintain integrity is known as ‘normalization.’

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **BankName** | **BankAddress** | **AccountTypeName** | **Account\_No** | **Customer\_Name** | **Contact** | **Balance** |
| Bank of America | California | Saving | 567 | Manjusha | 789456123 | 1000 |
| CITI | Illinois | Current | 1566 | Ganesh | 456121378 | 1240 |
| Saving | 2565 | Nigam | 123456789 | 1290 |
| Chase | Washington | Current | 3123 | Nikhil | 147852369 | 5252 |

## First Normal Form

The data in each column should be atomic and should not contain any duplicate values.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Customer\_Name** | **Contact** | **BankName** | **BankAddress** | **AccountTypeName** | **Account\_No** | **Balance** |
| Manjusha | 789456123 | Bank of America | California | Saving | 567 | 1000 |
| Ganesh | 456121378 | CITI | Illinois | Current | 1566 | 1240 |
| Nigam | 123456789 | CITI | Illinois | Saving | 2565 | 1290 |
| Nikhil | 147852369 | Chase | Washington | Current | 3123 | 5252 |

**Table 3‑I First Normal Form**

## Second Normal Form

The table should be in the first normal form (1NF) and all attributes of entities are dependent on the primary key.

|  |  |  |
| --- | --- | --- |
| **Customer\_ID(PK)** | **Customer\_Name** | **Contact** |
| 123 | Manjusha | 789456123 |
| 456 | Ganesh | 456121378 |
| 789 | Nigam | 123456789 |
| 1011 | Nikhil | 147852369 |

**Table 3‑II Customer**

Customer\_ID is the primary key

|  |  |  |
| --- | --- | --- |
| **Account\_ID(PK)** | **AccountNo** | **Balance** |
| 321 | 567 | 1000 |
| 654 | 1566 | 1240 |
| 987 | 2565 | 1290 |
| 1101 | 3123 | 5252 |

**Table 3‑III Account**

Account\_ID is the primary key

|  |  |  |
| --- | --- | --- |
| **BankID(PK)** | **BankName** | **BankAddress** |
| 101 | Bank of America | California |
| 102 | CITI | Illinois |
| 103 | Chase | Washington |

**Table 3‑IV Bank**

Bank\_ID is the primary key

## Third Normal Form

The table should be in the second normal form (2NF) and no Non-Key attribute should be dependent on another non key attribute.

### [dbo].[Customer]

CREATE TABLE [dbo].[Customer](

[Customer\_ID] [int] NOT NULL,

[Customer\_Name] [nvarchar](50) NOT NULL,

[Contact] [int] NOT NULL,

[Bank\_ID] [int] NOT NULL,

CONSTRAINT [PK\_Customer] PRIMARY KEY CLUSTERED

(

[Customer\_ID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

|  |  |  |  |
| --- | --- | --- | --- |
| **Customer\_ID(PK)** | **Customer\_Name** | **Contact** | **Bank\_ID(FK)** |
| 123 | Manjusha | 789456123 | 101 |
| 456 | Ganesh | 456121378 | 102 |
| 789 | Nigam | 123456789 | 102 |
| 1011 | Nikhil | 147852369 | 103 |

**Table 3‑V Customer**

Customer\_ID is the primary key.

Bank\_ID is the foreign key.

### [dbo].[Account]

CREATE TABLE [dbo].[Account](

[Account\_ID] [int] NOT NULL,

[Customer\_ID] [int] NOT NULL,

[AccountType\_ID] [int] NOT NULL,

[Account\_No] [int] NOT NULL,

[Balance] [int] NOT NULL,

CONSTRAINT [PK\_Account] PRIMARY KEY CLUSTERED

(

[Account\_ID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Account\_ID(PK)** | **Customer\_ID(FK)** | **AccountType\_ID(FK)** | **AccountNo** | **Balance** |
| 321 | 123 | 1 | 567 | 1000 |
| 654 | 456 | 2 | 1566 | 1240 |
| 987 | 789 | 1 | 2565 | 1290 |
| 1101 | 1011 | 2 | 3123 | 5252 |

**Table 3-VI Account**

Account\_ID is the primary key.

Customer\_ID is the foreign key.

AccountType\_ID is the foreign key.

### [dbo].[Bank]

CREATE TABLE [dbo].[Bank](

[Bank\_ID] [int] NOT NULL,

[BankName] [nvarchar](50) NOT NULL,

[BankAddress] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_Bank] PRIMARY KEY CLUSTERED

(

[Bank\_ID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

|  |  |  |
| --- | --- | --- |
| **Bank\_ID(PK)** | **BankName** | **BankAddress** |
| 101 | Bank of America | California |
| 102 | CITI | Illinois |
| 103 | Chase | Washington |

**Table 3‑VI Bank**

Bank\_ID is the primary key.

### [dbo].[AccountType]

CREATE TABLE [dbo].[AccountType](

[AccountType\_ID] [int] NOT NULL,

[AccountTypeName] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_AccountType] PRIMARY KEY CLUSTERED

(

[AccountType\_ID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

|  |  |
| --- | --- |
| **AccountTypeID(PK)** | **AccountTypeName** |
| 1 | Saving |
| 2 | Current |

GO

**Table 3‑VII AccountType**

AccountType\_ID is the primary key.

# Constraints

Constraints limit the type of data that goes into a table. They also provide accuracy and reliability of the data.

## [dbo].[Account]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Account\_ID(PK)** | **Customer\_ID(FK)** | **AccountType\_ID(FK)** | **AccountNo** | **Balance** |
| 321 | 123 | 1 | 567 | 1000 |
| 654 | 456 | 2 | 1566 | 1240 |
| 987 | 789 | 1 | 2565 | 1290 |
| 1101 | 1011 | 2 | 3123 | 5252 |

[dbo].[Account] has 3 constraints. They are:

**Key Constraint:** [Account\_ID] is the primary key and it cannot be null.

**Entity – Integrity Constraint**: Account\_ID is the primary key and it cannot be null.

**Referential Integrity Constraint**: Customer\_ID and AccountType\_ID are foreign keys this table and they refer to the primary key of Customer table and AccountType table respectively.

**Default Constraint**: Added default constraint df\_balance which by default contains balance as 0 when no balance is entered.

ALTER TABLE ACCOUNT

ADD CONSTRAINT df\_balance

DEFAULT 0 FOR [Balance]

INSERT INTO Account (Account\_ID,Customer\_ID,AccountType\_ID,Account\_No) VALUES(100,566,1,987)

Graphical user interface

Description automatically generated with low confidence

## [dbo].[AccountType]:

|  |  |
| --- | --- |
| **AccountTypeID(PK)** | **AccountTypeName** |
| 1 | Saving |
| 2 | Current |

**Key Constraint:** AccountTypeID is the primary key and it cannot be null.

**Entity-Integrity Constraint:** AccountTypeID is the primary key and it cannot be null.

## [dbo].[Bank]:

|  |  |  |
| --- | --- | --- |
| **Bank\_ID(PK)** | **BankName** | **BankAddress** |
| 101 | Bank of America | California |
| 102 | CITI | Illinois |
| 103 | Chase | Washington |

**Key Constraint:** Bank\_ID is the primary key and it cannot be null.

**Entity-Integrity Constraint:** Bank\_ID is the primary key and it cannot be null.

## [dbo].[Customer]:

|  |  |  |  |
| --- | --- | --- | --- |
| **Customer\_ID(PK)** | **Customer\_Name** | **Contact** | **Bank\_ID(FK)** |
| 123 | Manjusha | 789456123 | 101 |
| 456 | Ganesh | 456121378 | 102 |
| 789 | Nigam | 123456789 | 102 |
| 1011 | Nikhil | 147852369 | 103 |

**Key Constraint:** Customer\_ID is the primary key and it cannot be null**.**

**Entity Integrity Constraint:** Customer\_ID is the primary key and it cannot be null.

**Referential Integrity Constraint:** Bank\_ID is the foreign key that refers to the primary key in the Bank table.

**Check Constraint**: A check constraint named Ck\_PhValidation has been created to check the contact range from 100000000 to 999999999

ALTER TABLE [dbo].[Customer]

ADD CONSTRAINT Ck\_PhValidation

CHECK ([Contact] >= 100000000 AND [Contact] <= 999999999)

INSERT INTO [dbo].[Customer] VALUES(401,'Virat',10000000000,101)

Text

Description automatically generated

# Views

View is a logical structure of the table which can obtain the data from another table.

customer\_balance\_view has been created which results the customer name, balance, the type of account.

Below is the query for the view:

## View: customer\_balance\_view

create view customer\_balance\_view as

select [Customer\_Name], [Balance], [AccountTypeName]

from [dbo].[Customer] c

join

[dbo].[Account] a

on

c.[Customer\_ID] = a.Customer\_ID

join

[dbo].[AccountType] ac

on

a.AccountType\_ID = ac.AccountType\_ID

go

Query to view the result:

select \* from [dbo].[customer\_balance\_view]

**Result:**

Table

Description automatically generated

Here we are able to view customer name, balance, account type in a single table with the help of views concept.

## View: VSAVINGACCOUNT

vSavingAccount view gives the customer details and account details of savings account.

CREATE VIEW vSavingAccount

AS

SELECT C.Customer\_ID,C.Customer\_Name, C.Contact,A.Account\_No,A.Balance,B.BankName,ATY.AccountTypeName

FROM

Customer C

JOIN

Account A

ON A.Customer\_ID = C.Customer\_ID

JOIN

Bank B

ON

B.Bank\_ID = C.Bank\_ID

JOIN

AccountType ATY

ON

A.AccountType\_ID = ATY.AccountType\_ID

WHERE ATY.AccountTypeName = 'SAVING'

Query to view the result:

SELECT \* FROM vSavingAccount

Result:

Table

Description automatically generated

# Stored Procedures

Stored procedure is a collection of statements and logics that can be stored on the database. They can be reused and they hide the queries to improve performance.

A stored procedure named SP\_LOWBALANCE has been created which fetches customer name and balance when the balance is below 40.

## [dbo].[SP\_LOWBALANCE]

 CREATE PROCEDURE [dbo].[SP\_LOWBALANCE]

-- Add the parameters for the stored procedure here

@LOWBAL INT

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for procedure here

SELECT Customer\_Name, BALANCE LOWBALANCE

FROM [dbo].[Customer] C

JOIN

[dbo].[Account] AC

ON

C.Customer\_ID = AC.Customer\_ID

WHERE AC.Balance <= @LOWBAL

END

GO

**Query to view the result**:

EXEC [dbo].[SP\_LOWBALANCE] @LOWBAL =40

**Result:**

**Table

Description automatically generated with medium confidence**

# TRIGGERS

The SQL statements that are executed in response when an event is occurred in the database. The maintain the data integrity.

A trigger name Trg\_UpdateRestricted has been created which doesnot let the update operation.

## Trg\_UpdateRestricted:

CREATE TRIGGER Trg\_UpdateRestricted

ON [dbo].[AccountType]

AFTER UPDATE

AS

BEGIN

PRINT 'Update Operation - Restricted!!'

ROLLBACK TRANSACTION

END

Trying To Update:

update [dbo].[AccountType]

set AccountTypeName = 'savings'

where [AccountType\_ID] = 1

Result:

Text

Description automatically generated

# Functions

A function is a set of SQL statements that perform a specific task. It takes inputs as parameters and returns a value.

Here a function named [dbo].[fun\_avgbal]() has been created. It gives the average balance in a banking database as output.

## [dbo].[fun\_avgbal]()

create function [dbo].[fun\_avgbal]()

returns table

as

return (select AVG(balance) Average\_Balance from Account

GO

**Query to view the result:**

select \* from [dbo].[fun\_avgbal]()

**Result:**

A picture containing table

Description automatically generated

## fn\_GroupbyAccount():

A function named fn\_GroupbyAccount() has been created. Where it takes either ‘savings’ or ‘current’ as a parameter and displays customer details of those accounts.

CREATE FUNCTION fn\_GroupbyAccount(

@acctype NVARCHAR(100) NULL

)

RETURNS TABLE

AS

RETURN

(SELECT C.Customer\_ID,C.Customer\_Name, C.Contact,A.Account\_No,A.Balance,B.BankName,ATY.AccountTypeName

FROM

Customer C

JOIN

Account A

ON A.Customer\_ID = C.Customer\_ID

JOIN

Bank B

ON

B.Bank\_ID = C.Bank\_ID

JOIN

AccountType ATY

ON

A.AccountType\_ID = ATY.AccountType\_ID

WHERE ATY.AccountTypeName = @acctype)

**Query to view the result:**

SELECT \* FROM fn\_GroupbyAccount('CURRENT')

**Result:**

Table

Description automatically generated