DBMS Project Report

PES University

Database Management Systems

UE18CS252

Submitted By

PES2201800303	NIKHIL J K

The Bank Account Management System is an application for maintaining a person's account in a bank. In this project I tried to show the working of a banking account system and cover the basic functionality of a Bank Account Management System. To develop a project for solving financial applications of a customer in banking environment in order to nurture the needs of an end banking user by providing various ways to perform banking tasks. We aim to demonstrate the use of create, insert, update delete MySQL operations.

This data model consists of four tables employee, customer, transactions and accounts, each table has its own primary key which can be used to refer other values of the tables. For example, the customer identification number can be used to identify an individual and all his details. All the transactions and details of the transaction performed by the customer can be traced using his transaction id. All the details regarding a customer his/her account, employee InCharge, transaction details can be found.

Normalization is done to our database to minimize the data redundancy. Our database is in the second normalized form. Two triggers have been applied to the database in order to keep the track of time and the activity while inserting or updating the transactions table. Few numbers of SQL queries using aggregate functions, joins are also performed on the database which is used to help reduce the need for multiple OR conditions in a SELECT, INSERT, UPDATE, or DELETE statement.

The Traditional way of maintaining details of a user in a bank was to enter the details and record them. Every time the user needs to perform some transactions he has to go to bank and perform the necessary actions, which may not be so feasible all the time. It may be a hard-hitting task for the users and the bankers too. The project gives real life understanding

of Online Banking System and activities performed by various roles in the supply chain. Here, we provide automation for banking system through Internet. Online Banking System project captures activities performed by different roles in real life banking which provides enhanced techniques for maintaining the required information up-to-date, which results in efficiency. The project gives real life understanding of Online Banking System and activities performed by various roles in the supply chain.

Introduction

The **Bank Account Management System** is an application for maintaining a person's account in a bank. In this project I tried to show the working of a banking account system and cover the basic functionality of a Bank Account Management System. To develop a project for solving financial applications of a customer in banking environment in order to nurture the needs of an end banking user by providing various ways to perform banking tasks. We aim to demonstrate the use of create, insert, update delete MySQL operations.

Our mini world consists of.

• Customer: It contains attributes pertaining to general information about the customer, such as name, address, contact no ,CIF , where CIF is the primary key. CIF stands for 'Customer Identification File' which is a special ID given to a customer for the first time as he comes to a bank.

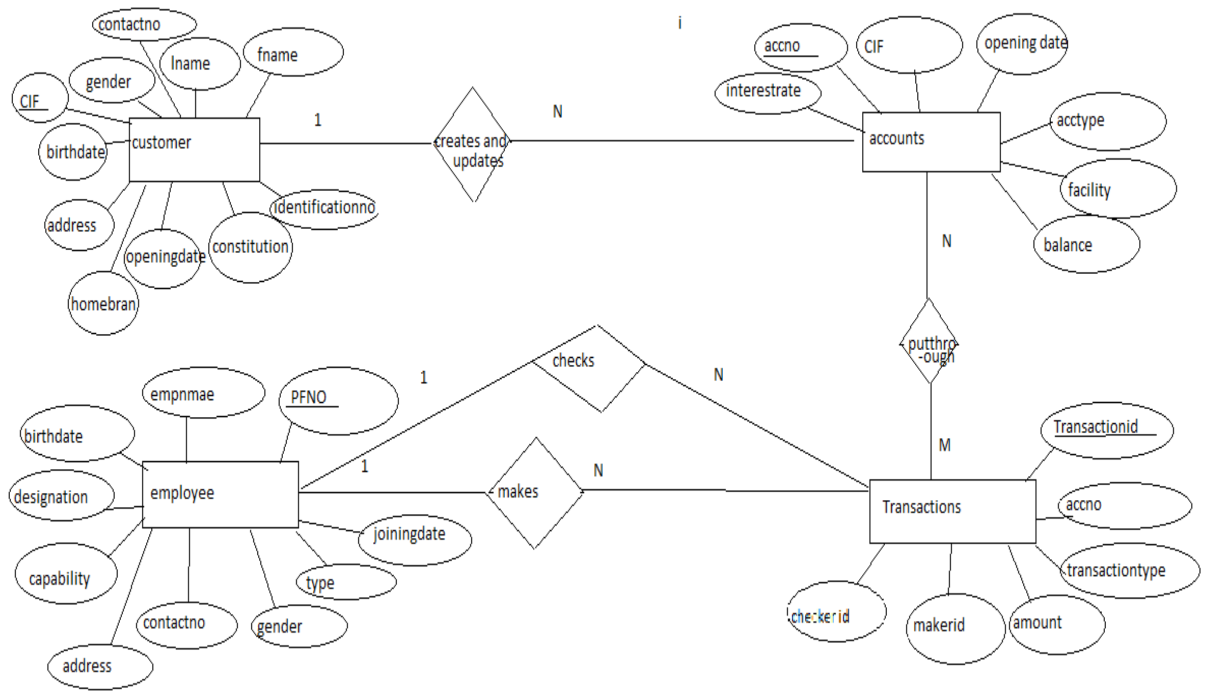
• Employee: This also contains many different attributes related to information about an employee, such as name, address, capability, pfno , contact where contact is the primary key. PF stands for 'Provident Fund' which is an ID given to every employee.

• Accounts: This entity contains all details about the accounts maintained in the bank. One customer may have more than one account. The account is mapped to customer using the CIF. Here, Accno acts as the primary key.

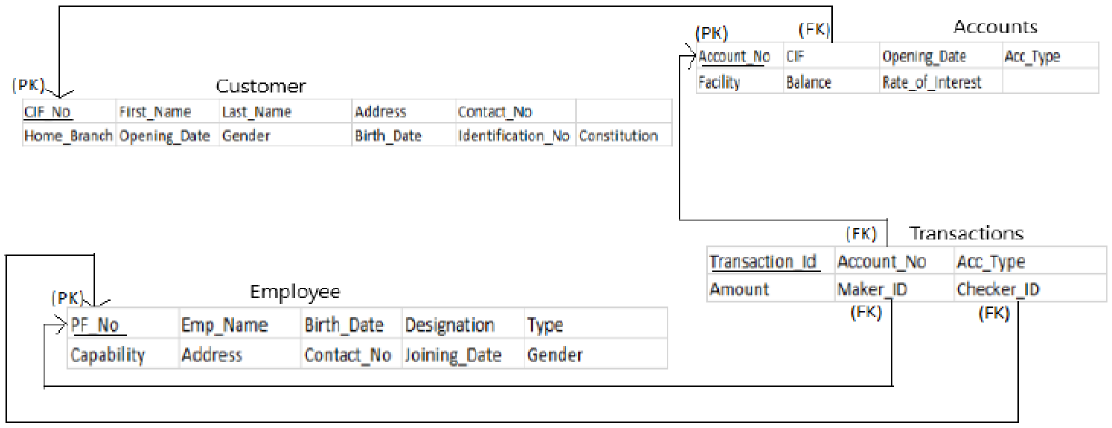
• Transactions: It contains all information related to a particular transaction. Its primary key is Transid. We can figure out information regarding a transaction like the account involved, type of transaction, customer details and even the employee details who approved of this transaction

Data Model

ER DIAGRAM



BANK SCHEMA



Primary keys

CIF –Customer identification file. (customer table)

CONTACTNO (employee table)

ACCOUNTNO (accounts table)

TRANSACTIONID (transactions table)

Foreign keys

CIF (accounts)

ACCOUNTNO (transactions)

MAKERID (transactions)

CHECKERID (transactions)

FD and Normalization

Functional Dependencies:

There are many function dependencies considering the schema that we have chosen since each table has a primary key. We can consider a primary key or any combination of primary keys on the left side to be mapped to one column or any combination of other columns in the table.

Here’s the functional dependency:

Customer table

IdentificationNo -->{Name, Address, OpeningDate, BirthDate}

Employee Table.

ContactNo -->{EmpName, BirthDate, Address, Designation}

We choose ContactNo in Employee table as candidate key but not in Customer table because a customer maybe minor and his/her contact number may match with their parents/guardians. But in the case of Employee, they are not minors and have their own unique contact number.

Accounts table

Account no-->{accounttype, facility, interest rate,balance}

Transactions table

TransactionID-->{transaction type,amount}

Normalization

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy(repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.

Normalization is used for mainly two purposes,

- Eliminating redundant(useless) data.
- Ensuring data dependencies make sense i.e data is logically stored.

Normalization check

• **1NF:**

According to 1NF,

* each column should have single value not multiple value.

*Column should contain value of same type.

*Each column should have unique name.

Our Customer table had ‘Name’ as composite attribute but now it has been split to ‘FName’ and ‘LName’. So, we do not have either composite or multi-values attributes. Also, all values belong to the domain and the column names are also unique. Hence, all our tables satisfy 1NF.

• **2NF**:

According to 2NF

*table must satisfy 1NF.

*Table must not have values that are partial dependent.

There are primary keys in all our tables from this we can find all other values in the tables using only one value of the primary key. Also, all our tables satisfy 1NF as mentioned above. Hence, the tables satisfy 2NF as well.

• **3NF:**

According to 3NF

*Table must satisfy 2NF

*Table should not have columns with transitive dependency.

For two tables, Employee and Accounts, we can observe that ‘Capability’ depends on ‘Type’ and ‘Interest’ depends on ‘AccType’ respectively. i.e. a non-prime attribute depends on another non-prime attribute which says that transitive dependency exists in both of these tables. So, we say that these two tables are of 2NF form only but on the other hand, the other two tables, Customer and Transaction do not satisfy transitive dependency and satisfy the requirements of 3NF. They both satisfy BCNF as well since the primary key is the only way we can get to all the other values of the tables and we cannot get other values using any other attribute.

Hence, we can conclude that all our tables satisfy **2NF**.

DDL

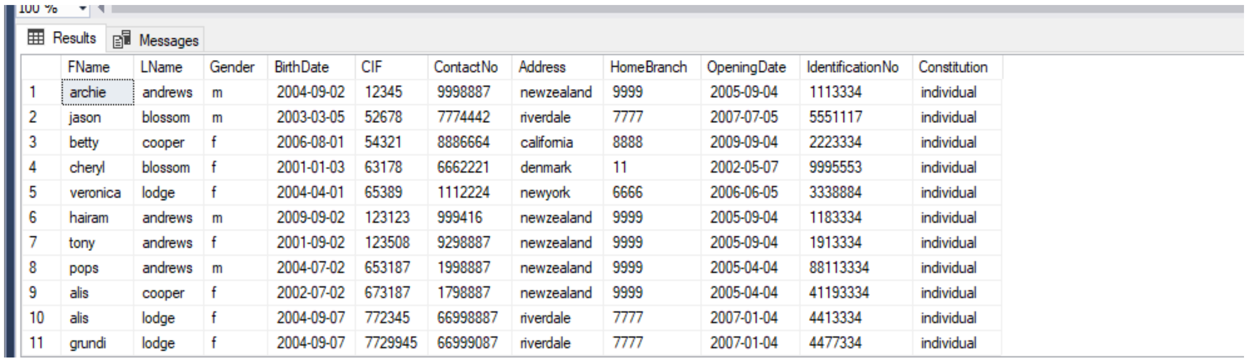
A data definition language (DDL) is a computer language used to create and modify the structure of database objects in a database. These database objects include views, schemas, tables, indexes, etc.

This term is also known as data description language in some contexts, as it describes the fields and records in a database table.

DDL of bankmanagement database

```
create database bankmanagement

use bankmanagement


Create Table Customer(

FName varchar(30) not null,LName varchar(30) not null ,Gender char(1) not null,BirthDate date
not null,CIF bigint not null primary key

,ContactNo bigint not null, Address varchar(50) not null,

HomeBranch int not null ,OpeningDate date not null,  IdentificationNo varchar(15) not null,

Constitution varchar(10) not null , unique (ContactNo),

/*check constraint*/

check (CIF <= 999999999999),check (ContactNo <= 99999999999), check (HomeBranch <=
999999));


insert into Customer(FName,LName,Gender,BirthDate,CIF,ContactNo,
Address,HomeBranch,OpeningDate,IdentificationNo,Constitution)

values('archie','andrews','m','2004-09-02','12345','9998887','newzealand','9999','2005-09-
04','1113334','individual'),

(‘betty','cooper','f','2006-08-01','54321','8886664','california','8888','2009-09-
04','2223334','individual'),

(‘jason','blossom','m','2003-03-05','52678','7774442','riverdale','7777','2007-07-
05','5551117','individual'),

(‘veronica','lodge','f','2004-04-01','65389','1112224','newyork','6666','2006-06-
05','3338884','individual'),

(‘cheryl','blossom','f','2001-01-03','63178','6662221','denmark','0011','2002-05-
07','9995553','individual'),

(‘hairam','andrews','m','2009-09-02','123123','999416','newzealand','9999','2005-09-
04','1183334','individual'),
('tony','andrews','f','2001-09-02','123508','9298887','newzealand','9999','2005-09-04','1913334','individual'),

('pops','andrews','m','2004-07-02','653187','1998887','newzealand','9999','2005-04-04','88113334','individual'),

('alis','lodge','f','2004-09-07','772345','66998887','riverdale','7777','2007-01-04','4413334','individual'),

('alis','cooper','f','2002-07-02','673187','1798887','newzealand','9999','2005-04-04','41193334','individual'),

('grundi','lodge','f','2004-09-07','7729945','66999087','riverdale','7777','2007-01-04','4477334','individual');
```

```
select * from Customer
```



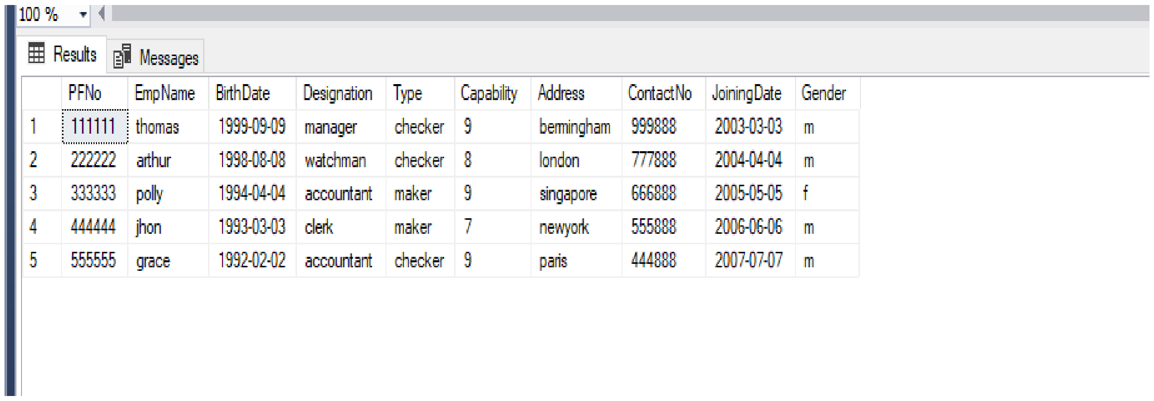
100 %

Results | Messages

	FName	LName	Gender	BirthDate	CIF	ContactNo	Address	HomeBranch	OpeningDate	IdentificationNo	Constitution
1	archie	andrews	m	2004-09-02	12345	9998887	newzealand	9999	2005-09-04	1113334	individual
2	jason	blossom	m	2003-03-05	52678	7774442	riverdale	7777	2007-07-05	5551117	individual
3	betty	cooper	f	2006-08-01	54321	8886664	califomia	8888	2009-09-04	2223334	individual
4	cheryl	blossom	f	2001-01-03	63178	6662221	denmark	11	2002-05-07	9995553	individual
5	veronica	lodge	f	2004-04-01	65389	1112224	newyork	6666	2006-06-05	3338884	individual
6	hairam	andrews	m	2009-09-02	123123	999416	newzealand	9999	2005-09-04	1183334	individual
7	tony	andrews	f	2001-09-02	123508	9298887	newzealand	9999	2005-09-04	1913334	individual
8	pops	andrews	m	2004-07-02	653187	1998887	newzealand	9999	2005-04-04	88113334	individual
9	alis	cooper	f	2002-07-02	673187	1798887	newzealand	9999	2005-04-04	41193334	individual
10	alis	lodge	f	2004-09-07	772345	66998887	riverdale	7777	2007-01-04	4413334	individual
11	grundi	lodge	f	2004-09-07	7729945	66999087	riverdale	7777	2007-01-04	4477334	individual

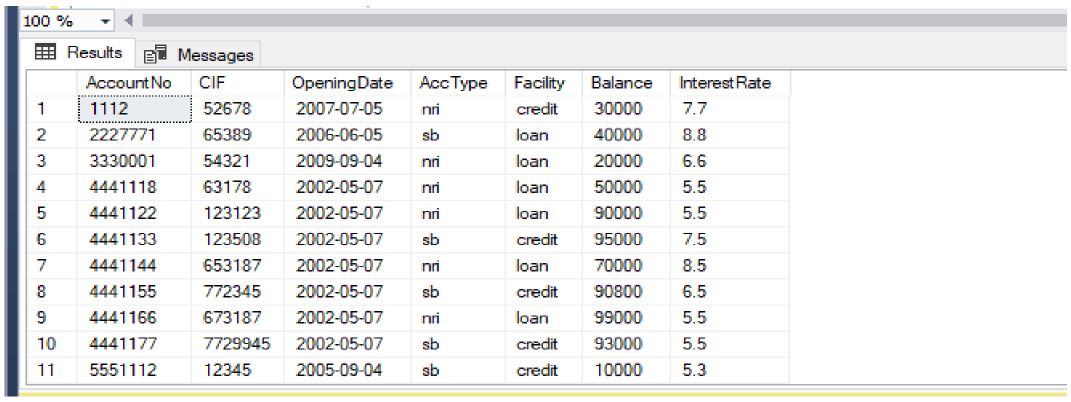
```
Create Table Employee(PFNo bigint not null primary key, EmpName varchar(30) not null,
BirthDate date not null, Designation varchar(25),

Type varchar(7) not null, Capability int not null,

Address varchar(50) not null, ContactNo bigint not null,

JoiningDate date not null, Gender char(1) not null,

/*check constraint*/

check (PFNo <= 99999999), check (Capability<10),

check (ContactNo <= 99999999999));
```

```
insert into
Employee(PFNo,EmpName,BirthDate,Designation,Type,Capability,Address,ContactNo,JoiningDate,Gender)

values('111111','thomas','1999-09-09','manager','checker','9','bermingham','999888','2003-03-03','m'),
('222222','arthur','1998-08-08','watchman','checker','8','london','777888','2004-04-04','m'),
('333333','polly','1994-04-04','accountant','maker','9','singapore','666888','2005-05-05','f'),
('444444','jhon','1993-03-03','clerk','maker','7','newyork','555888','2006-06-06','m'),
('555555','grace','1992-02-02','accountant','checker','9','paris','444888','2007-07-07','m');
```



	PFNo	EmpName	BirthDate	Designation	Type	Capability	Address	ContactNo	JoiningDate	Gender
1	111111	thomas	1999-09-09	manager	checker	9	bermingham	999888	2003-03-03	m
2	222222	arthur	1998-08-08	watchman	checker	8	london	777888	2004-04-04	m
3	333333	polly	1994-04-04	accountant	maker	9	singapore	666888	2005-05-05	f
4	444444	jhon	1993-03-03	clerk	maker	7	newyork	555888	2006-06-06	m
5	555555	grace	1992-02-02	accountant	checker	9	paris	444888	2007-07-07	m

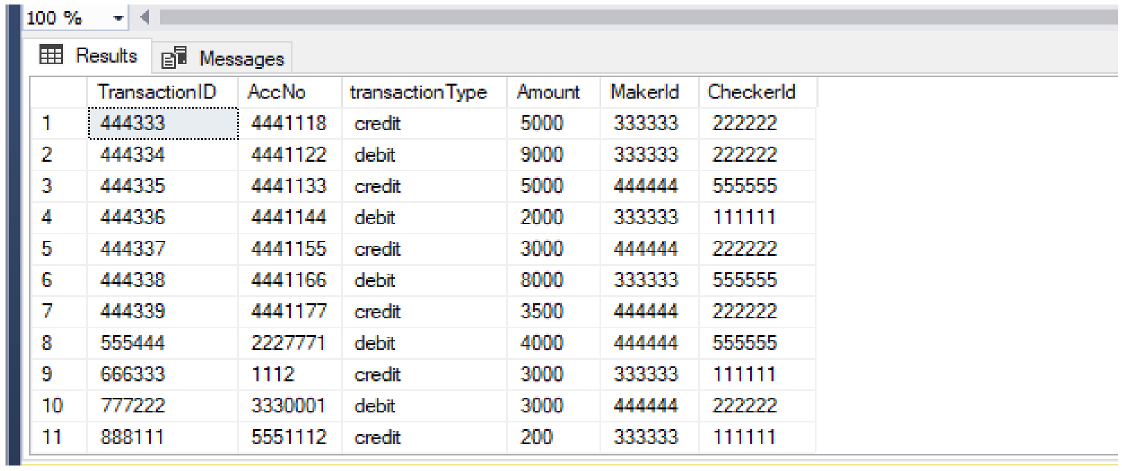
```
Create Table Accounts(
AccountNo bigint not null primary key, CIF bigint not null,
OpeningDate date not null, AccType varchar(7) not null,
Facility varchar(11) not null, Balance int not null,InterestRate float not null,
/*check constraint*/
check (AccountNo<=99999999999),
check (CIF<=99999999999), check(InterestRate<=10.00));


 ALTER TABLE Accounts ADD CONSTRAINT Foreign Key (CIF) References Customer (CIF)
on delete cascade on update cascade.


insert into Accounts(AccountNo,CIF,OpeningDate, AccType,Facility,Balance,InterestRate)
values('5551112','12345','2005-09-04','sb','credit','10000','5.30'),
('3330001','54321','2009-09-04','nri','loan','20000','6.60'),
('0001112','52678','2007-07-05','nri','credit','30000','7.70'),
('2227771','65389','2006-06-05','sb','loan','40000','8.80'),
('4441118','63178','2002-05-07','nri','loan','50000','5.50'),
('4441122','123123','2002-05-07','nri','loan','90000','5.50'),
('4441133','123508','2002-05-07','sb','credit','95000','7.50'),
('4441144','653187','2002-05-07','nri','loan','70000','8.50'),

('4441155','772345','2002-05-07','sb','credit','90800','6.50'),

('4441166','673187','2002-05-07','nri','loan','99000','5.50'),

('4441177','7729945','2002-05-07','sb','credit','93000','5.50'),

('44434217','7729945','2002-05-07','nri','loan','88000','5.60');
```



	AccountNo	CIF	OpeningDate	AccType	Facility	Balance	InterestRate
1	1112	52678	2007-07-05	nri	credit	30000	7.7
2	2227771	65389	2006-06-05	sb	loan	40000	8.8
3	3330001	54321	2009-09-04	nri	loan	20000	6.6
4	4441118	63178	2002-05-07	nri	loan	50000	5.5
5	4441122	123123	2002-05-07	nri	loan	90000	5.5
6	4441133	123508	2002-05-07	sb	credit	95000	7.5
7	4441144	653187	2002-05-07	nri	loan	70000	8.5
8	4441155	772345	2002-05-07	sb	credit	90800	6.5
9	4441166	673187	2002-05-07	nri	loan	99000	5.5
10	4441177	7729945	2002-05-07	sb	credit	93000	5.5
11	5551112	12345	2005-09-04	sb	credit	10000	5.3

```
Create Table Transactions(

TransactionID varchar(10) not null primary key,

AccNo bigint not null, transactionType varchar(9) not null,

Amount int not null, MakerId bigint not null,

CheckerId bigint not null,

/*check constraint*/

check (AccNo <= 99999999999),

check (MakerId <= 99999999), check (CheckerId <= 99999999));


ALTER TABLE Transactions ADD CONSTRAINT AccNo Foreign key(AccNo)references
Accounts(AccountNo)on delete cascade on update cascade

ALTER TABLE Transactions ADD CONSTRAINT Foreign Key (MakerId) References
Employee(PFNo) on delete cascade on update cascade

ALTER TABLE Transactions ADD CONSTRAINT Foreign Key (checkerId) References
Employee(PFNo) on delete cascade on update cascade


insert into Transactions(TransactionID,AccNo, transactionType,Amount, MakerId,CheckerId)

values('888111','5551112','credit','200','333333','111111'),

('777222','3330001','debit','3000','444444','222222'),
('666333','0001112','credit','3000','333333','111111'),
('555444','2227771','debit','4000','444444','555555'),
('444333','4441118','credit','5000','333333','222222'),
('444334','4441122','debit','9000','333333','222222'),
('444335','4441133','credit','5000','444444','555555'),
('444336','4441144','debit','2000','333333','111111'),
('444337','4441155','credit','3000','444444','222222'),
('444338','4441166','debit','8000','333333','555555'),
('444339','4441177','credit','3500','444444','222222');
```

```
select * from Transactions
```



	TransactionID	AccNo	transactionType	Amount	MakerId	CheckerId
1	444333	4441118	credit	5000	333333	222222
2	444334	4441122	debit	9000	333333	222222
3	444335	4441133	credit	5000	444444	555555
4	444336	4441144	debit	2000	333333	111111
5	444337	4441155	credit	3000	444444	222222
6	444338	4441166	debit	8000	333333	555555
7	444339	4441177	credit	3500	444444	222222
8	555444	2227771	debit	4000	444444	555555
9	666333	1112	credit	3000	333333	111111
10	777222	3330001	debit	3000	444444	222222
11	888111	5551112	credit	200	333333	111111

Triggers

A trigger is a special kind of a store procedure that executes in response to certain action on the table like insertion, deletion or updation of data. It is a database object which is bound to a table and is executed automatically.

Types of Triggers

1. After Triggers (For Triggers): After Insert, After Update, After

2. Instead of Trigger

EXAMPLE FOR TRIGGER

```
  create table rleviolation(message varchar (20),ACC1 bigint,log_action
varchar(20),log_timestamp date)
```

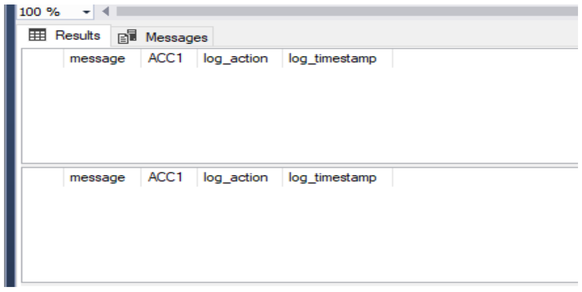
create table violationafterupdate(message varchar (20),ACC1 bigint,log_action varchar(20),

log_timestamp date)

The main purpose of this rleviolation and violationafterupdate table is to record the changes in the main table.

select * from rleviolation

select * from violationafterupdate



```
message  ACC1  log_action  log_timestamp

message  ACC1  log_action  log_timestamp
```

1.Trigger after insert

The CREATE TRIGGER statement is used to create the trigger. THE ON clause specifies the table name on which the trigger is to be attached. The after INSERT specifies that this is an AFTER INSERT trigger. In the trigger body, table named inserted has been used. This table is a logical table and contains the row that has been inserted. I have selected the fields from the logical inserted table from the row that has been inserted into different variables, and finally inserted those values into the rleviolation table.

This trigger is fired after an INSERT on the table.

GO

CREATE TRIGGER violationafterinsert on Transactions AFTER INSERT

AS

DECLARE @Amount bigint;

DECLARE @Type varchar(20);

DECLARE @ACC1 bigint;

DECLARE @log_action varchar(20);

select

HYPERLINK "mailto:@Amount=i.Amount,@ACC1=i.AccNo,@Type=i.Transactiontype"@Amount=i.Amount, @ACC1=i.AccNo,@Type=i.Transactiontype from inserted i;

set @log_action='inserted record';

if(@Amount<1000 or @Amount>1000000)

insert into rleviolation(message,ACC1 ,log_action,log_timestamp)values('rule violated',@ACC1,@log_action,getdate());

else

insert into rleviolation(message,ACC1,log_action,log_timestamp)values('',@ACC1,@log_action,getdate());

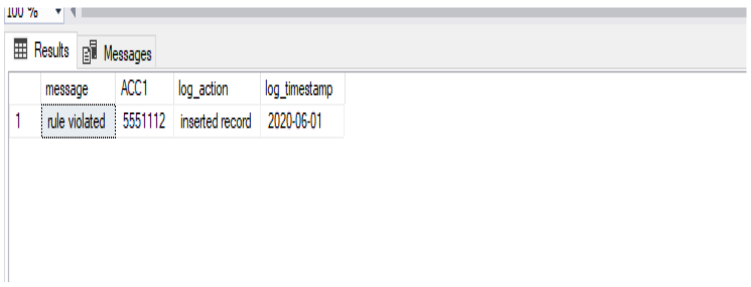
print'after inserted triggered fired'

To see the newly created trigger in action, lets insert a row into the main table as:

insert into Transactions values('188111','5551112','credit','200','333333','111111');

select * from rleviolation

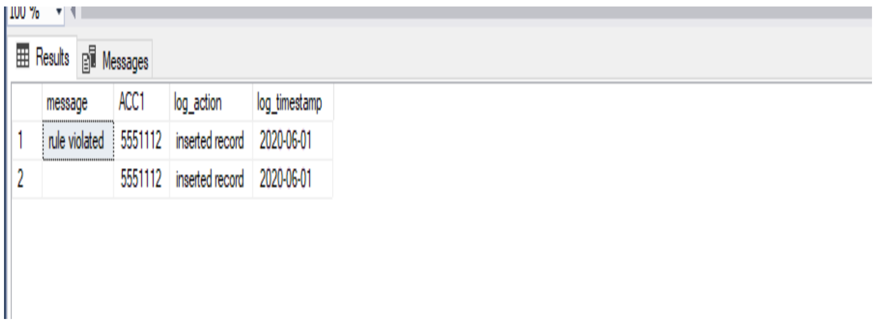
Now, a record has been inserted into the transactions table. The AFTER INSERT trigger attached to this table has inserted the record into the rleviolation as:



	message	ACC1	log_action	log_timestamp
1	rule violated	5551112	inserted record	2020-06-01

insert into Transactions values('988111','5551112','credit','2000','333333','111111');

Now, a record has been inserted into the transactions table. The AFTER INSERT trigger attached to this table has inserted the record into the rleviolation as:



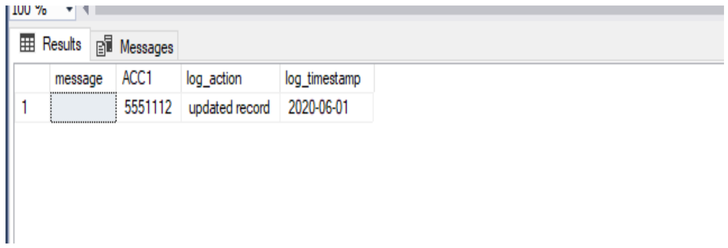
	message	ACC1	log_action	log_timestamp
1	rule violated	5551112	inserted record	2020-06-01
2		5551112	inserted record	2020-06-01

2.Trigger after update

The AFTER-UPDATE Trigger is created in which the updated record is inserted into the violationafterupdate table.

This trigger is fired after an update on the table.

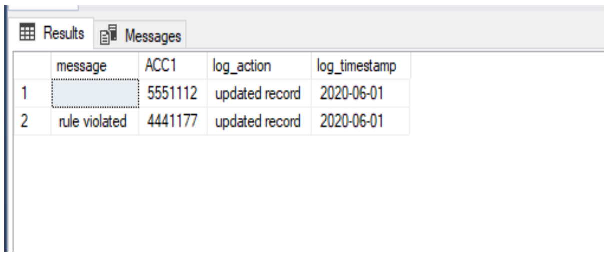
```
GO

CREATE TRIGGER violationafterupdation on Transactions AFTER update

AS


DECLARE @Amount bigint;

DECLARE @Type varchar(20);

DECLARE @ACC1 bigint;

DECLARE @log_action varchar(20);

select

HYPERLINK
"mailto:@Amount=i.Amount,@ACC1=i.AccNo,@Type=i.Transactiontype"@Amount=i.Amount,
@ACC1=i.AccNo,@Type=i.Transactiontype from inserted i;

set @log_action='updated record';

if(@Amount<1000 or @Amount>1000000)

insert into violationafterupdate(message,ACC1  ,log_action,log_timestamp)values('rule
violated',@ACC1,@log_action,getdate());

else

insert into violationafterupdate(message,ACC1
,log_action,log_timestamp)values('',@ACC1,@log_action,getdate());

print'after updated triggered fired'


update Transactions set Amount=20000 where AccNo=5551112;


select* from violationafterupdate;
```


100 %

Results Messages

```
   message  ACC1     log_action      log_timestamp
1           5551112  updated record  2020-06-01
```

update Transactions set Amount=200 where AccNo=4441177


Results Messages

```
   message        ACC1     log_action      log_timestamp
1                 5551112  updated record  2020-06-01
2  rule violated  4441177  updated record  2020-06-01
```

SQL Queries

A query is a question, often expressed in a formal way. A database query can be either a select query or an action query. A select query is a data retrieval query, while an action query asks for additional operations on the data, such as insertion, updating or deletion.

Queries can accomplish a few different tasks. Primarily, queries are used to find specific data by filtering specific criteria. Queries can also calculate or summarize data, as well as automate data management tasks. Other queries include parameter, totals, crosstab, make table, append, update and delete. For example, a parameter query runs variations of a particular query, which prompts a user to insert a field value, and then it uses that value to create the criteria, while totals queries allow users to group and summarize data.

1. Return Transaction_ID's of the transactions performed by cheryl blossom

Select TransactionID

From Transactions t

Where AccNo IN(

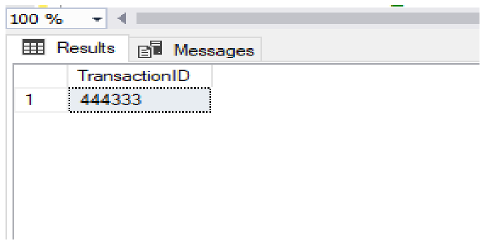
select AccountNo

from Accounts a

where CIF IN(

select CIF

from Customer where Fname='cheryl'));



	TransactionID
1	444333

2. Return CIF and Account_No of a customer who inputs her Identifcation_No as '3338884'

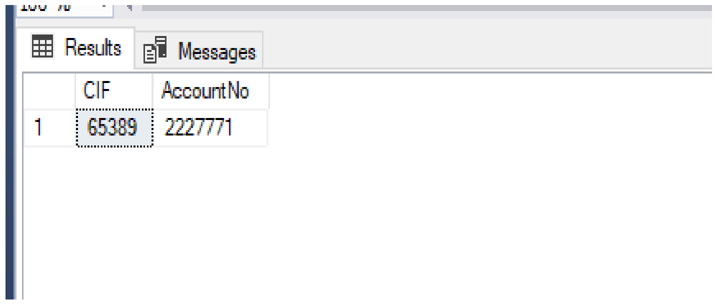
select CIF,AccountNo

From Accounts a

where CIF IN(

select CIF from Customer c

where IdentificationNo='3338884')



	CIF	AccountNo
1	65389	2227771

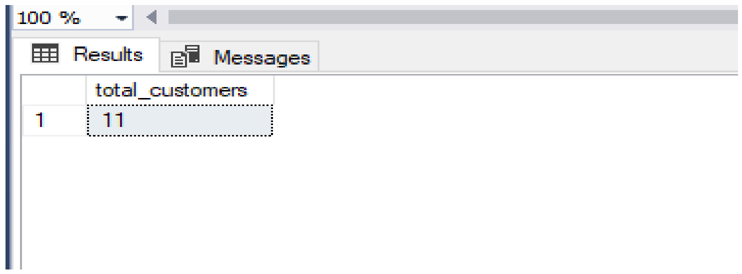
Aggregate Functions

An aggregate function performs a calculation on a set of values, and returns a single value. Except for COUNT , aggregate functions ignore null values. Aggregate functions are often used with the GROUP BY clause of the SELECT statement. All aggregate functions are deterministic.

1. Return the total number of customers

select count(CIF) total_customers

from Customer

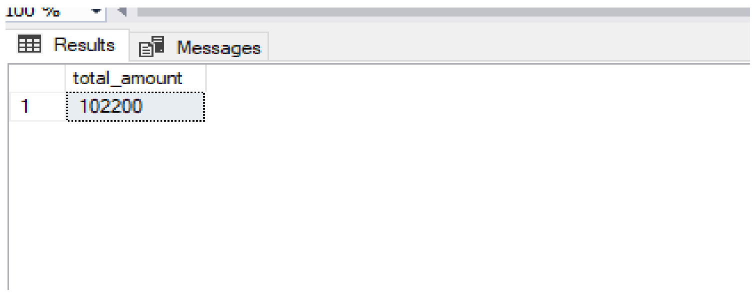


	total_customers
1	11

2. Return the total amount of money transferred during transactions

select sum(Amount) total_amount

from Transactions



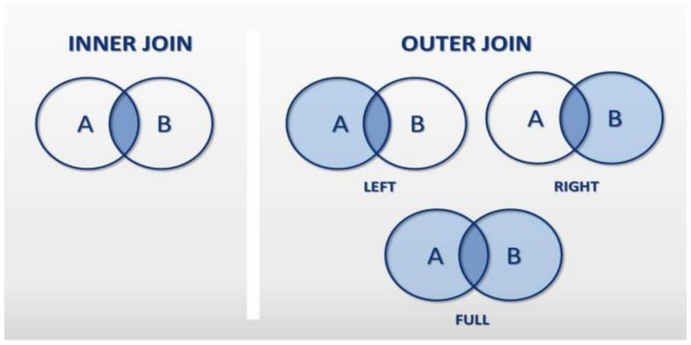
```
   total_amount
1  102200
```

Outer-Join Queries

The SQL OUTER JOIN returns all rows from both the participating tables which satisfy the join condition along with rows which do not satisfy the join condition. The SQL OUTER JOIN operator (+) is used only on one side of the join condition only.

The subtypes of SQL OUTER JOIN

- LEFT OUTER JOIN or LEFT JOIN
- RIGHT OUTER JOIN or RIGHT JOIN
- FULL OUTER JOIN



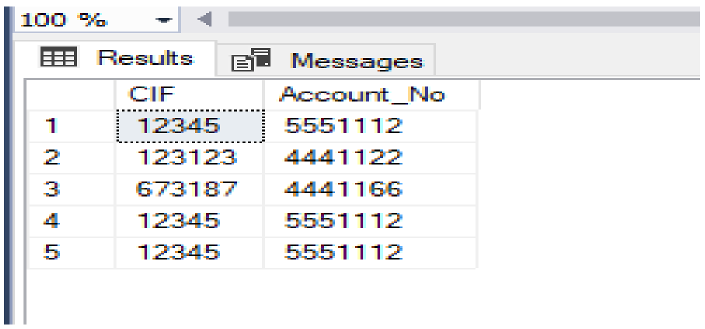
Q. Find CIF, AccountNo of customers who have done transactions of amount > 5000

select c.CIF as CIF ,a.AccountNo as Account_No

from (Customer c left outer join Accounts a on c.CIF=a.CIF) join Transactions as t

on AccountNo=t.AccNo

where Amount>5000



	CIF	Account_No
1	12345	5551112
2	123123	4441122
3	673187	4441166
4	12345	5551112
5	12345	5551112

Conclusion

This project is developed to nurture the needs of a user in a banking sector by embedding all the tasks of transactions taking place in a bank. Future version of this project will still be much enhanced than the current version. Writing and depositing checks are perhaps the most fundamental ways to move money in and out of a checking account, but advancements in technology have added ATM and debit card transactions. All banks have rules about how long it takes to access your deposits, how many debit card transactions you're allowed in a day, and how much cash you can withdraw from an ATM. Access to the balance in your checking account can also be limited by businesses that place holds on your funds.

Banks are providing internet banking services also so that the customers can be attracted. By asking the bank employs we came to know that maximum numbers of internet bank account holders are youth and business man. Online banking is an innovative tool that is fast becoming a necessity. It is a successful strategic weapon for banks to remain profitable in a volatile and competitive marketplace of today. If proper training should be given to customer by the bank employs to open an account will be beneficial secondly the website should be made friendlier from where the first-time customers can directly make and access their accounts.

Thus, the Bank Management System it is developed and executed successfully.

Most available benefits

1. Online banking with key bank is fast, secure, convenient and free.
2. Quick, simple, authenticated access to accounts via the web application.
3. Simply scalable to grow with changing system requirement.
4. Global enterprise wide access to information.
5. Improved data security, restricting unauthorized access.
6. Minimize Storage Space.

Limitations

Technology and Service Interruptions
Security and Identity Theft Concerns
Limitations on Deposits
Convenient but Not Always Faster
Lack of Personal Banker Relationship
A Limited Scope of Services

Future enhancements

To know what the future of online banking looks like, it’s probably worth looking at the present – online banking isn’t new. When you think of online banking, you probably think about a computer (either a desktop or laptop), a three or four step security process and then an interface that lets you view the balance of your various bank accounts and credit cards, whilst permitting you to transfer money and pay bills. And you’re not wrong either. The most valuable future looks are following below:

1- More branches of the bank, maybe it will be international, that means more ATM machines outside.

2- Customer issues development based on their needs, so the help desk will be aware of their needs and easy to use.

3- Developing a mobile App for banking system that help users to do the obtained his operations without go to the bank only he needs to sign in using his A/C NO. And password and then use your own PIN. Finally, the system will update automatically.