

MULTIMODAL BIOMETRIC TRANSACTION SYSTEMS

A Project Report

by

Achal Sancheti, Pallak Jain, Shabbir Hussain, Shilpi Bundela, Tanisha Jain

*In partial fulfillment for the course
of*

INFO 6210

Database Management and Database Design Report



Date: Dec 17th 2016

**Information Systems Engineering
College of Engineering, Northeastern University
Boston, MA 02115**

ABSTRACT

Multimodal Biometric Transactional Systems aims to develop robust and extremely secured transactional systems worldwide. In the present scenario, there is a need to enhance the security and safety of existing transactional systems. The level of security is raised by introducing composite multimodal biometric techniques for user's authentication. Highly transactional databases are also prone to attacks by unauthorized users. An encryption technique SHA-2 is utilized to perform database encryption to make the user's private data templates secured. Template comparison is presented by a stored procedure named sp_verifier which compares the biometric data with customer's input and displays results if users are verified.

Table of Contents

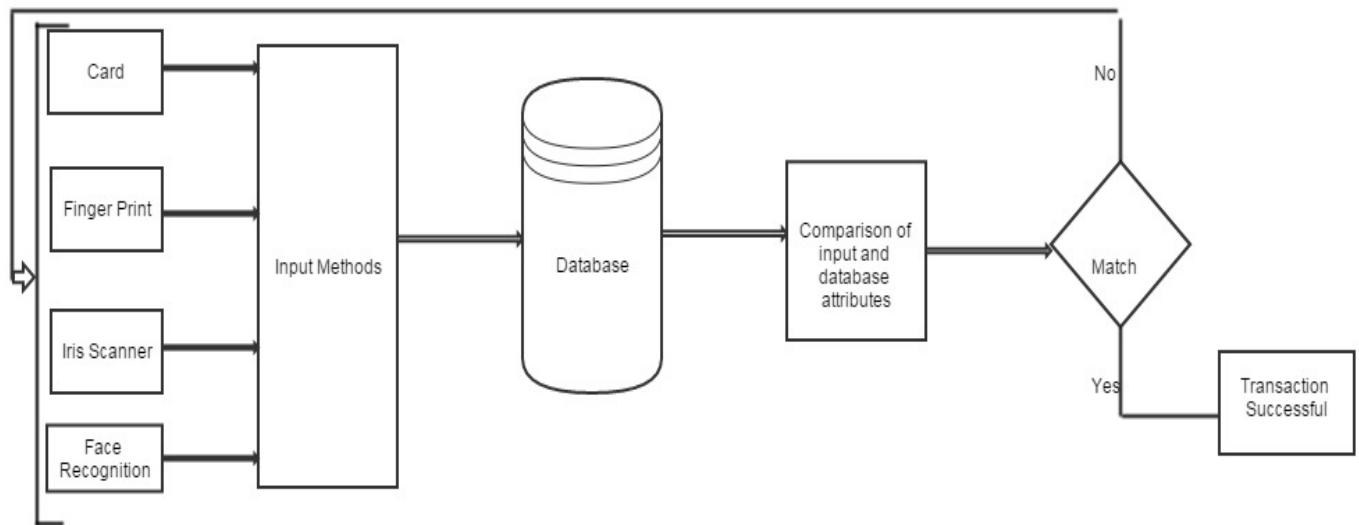
CHAPTER – 1	
1.1 Introduction	4
1.2 Problem Definition	5
1.3 Proposed Solution	6
CHAPTER – 2	
2.1 Why Multimodal Biometric	7
2.2 Key Function	8
CHAPTER – 3	
3.1 Use-Case Diagram	9-10
3.2 Entity-Relationship Diagram	11-12
3.3 Enhanced Entity-Relationship Diagram	13
CHAPTER – 4	
4.1 Encryption	14
4.2 Encryption Technique	15
4.3 Database Encryption	16-17
CHAPTER – 5	
5.1 Tables and Attributes	18-20
5.2 Relationship and Cardinalities	21
5.3 Queries	
5.3.1 Index	22-24
5.3.2 View	24-25
5.3.3 User and Privileges	25-26
5.3.4 Transaction	27
5.3.5 Sub-query	27
5.3.6 Trigger	28
5.3.7 Stored Procedure	29-31
CHAPTER – 6	
6.1 Backup Strategy	32
CHAPTER – 7	
7.1 Key Benefits	33
CHAPTER -8	
8.1 Assumption	34
8.2 Future Scope	34
CHAPTER – 9	
9.1 References	35

CHAPTER – 1

1.1 Introduction:

The purpose of our project is to introduce a multimodal biometric system which will enhance the security of all cashless transactions. It can be widely implemented in card reading and user authentication machines.

The authentication will be performed on the basis of the stored data of the customer in the database. If combination is validated, transaction can be done, else a case of fraud.



- Biometric template of user's fingerprints, facial schema and retina scan are stored in database at the time of account creation in bank.
- When a user performs any transaction, his biometric information is verified.
- Verification is successful if the transactions are validated on comparison with the stored templates.
- Encryption techniques used to secure private information in database.
- Multimodal authentication system can be widely implemented in ATMs, automatic cars and all existing card reading machines.

1.2 Problem Definition:

Since 2012, Identity theft victims suffered direct or indirect losses of more than \$24.7 billion dollars every year. On average, victims whose personal information was misused experienced a great loss. Victims whose personal information is misused are more likely to experience financial problems such as dealing with debt collectors, repeatedly correcting credit reports because of identity theft than those whose existing account information is stolen, but victims of all categories may experience these issues. Therefore, there is a need to improve the security of existing transactional systems, which uses person's identity as a parameter of authentication.

1.3 Proposed Solution:

The solution provided to this problem is that a composite biometric authentication is provided in all card-reading machines. The biometric techniques utilized in the project are-

- Finger print Scan
- Facial Recognition
- Retina Scan

Customer's biometric templates are stored in the bank at the time of account creation. User selects a combination out of the given authentication methods:

- Facial Recognition + Iris Scan
- Iris Scan + Finger Print
- Finger Print + Facial Recognition
- Card + Any of the Biometric method

The authentication is performed on comparison with the stored template of the customer in the database.

If combination is validated, transaction can be done, else a case of fraud.

SHA-2 algorithm converts the customer's sensitive data i.e. biometrics and card details into encrypted text and stores encrypted value in database.

CHAPTER – 2

2.1 Why Multimodal Biometric:

Unimodal systems perform person recognition based on single source of evidence which may sometimes be affected by sensory noise.

It can be useful for people with hand-related disabilities.

A small portion of population can have nearly identical facial appearance due to genetic factors.

2.2 Key Functions:

Biometric template of user's fingerprints, facial schema and retina scan are stored in database at the time of account creation in bank.

When a user performs any transaction, his biometric information is verified.

Verification is successful if the biometrics at the time of transactions are verified in comparison with the stored templates.

Encryption techniques used to secure private information in database.

Multimodal authentication system can be widely implemented in ATMs, automatic cars and all existing card reading machines.

CHAPTER – 3

3.1 Use-Case Diagram:

(a) CUSTOMER ROLE

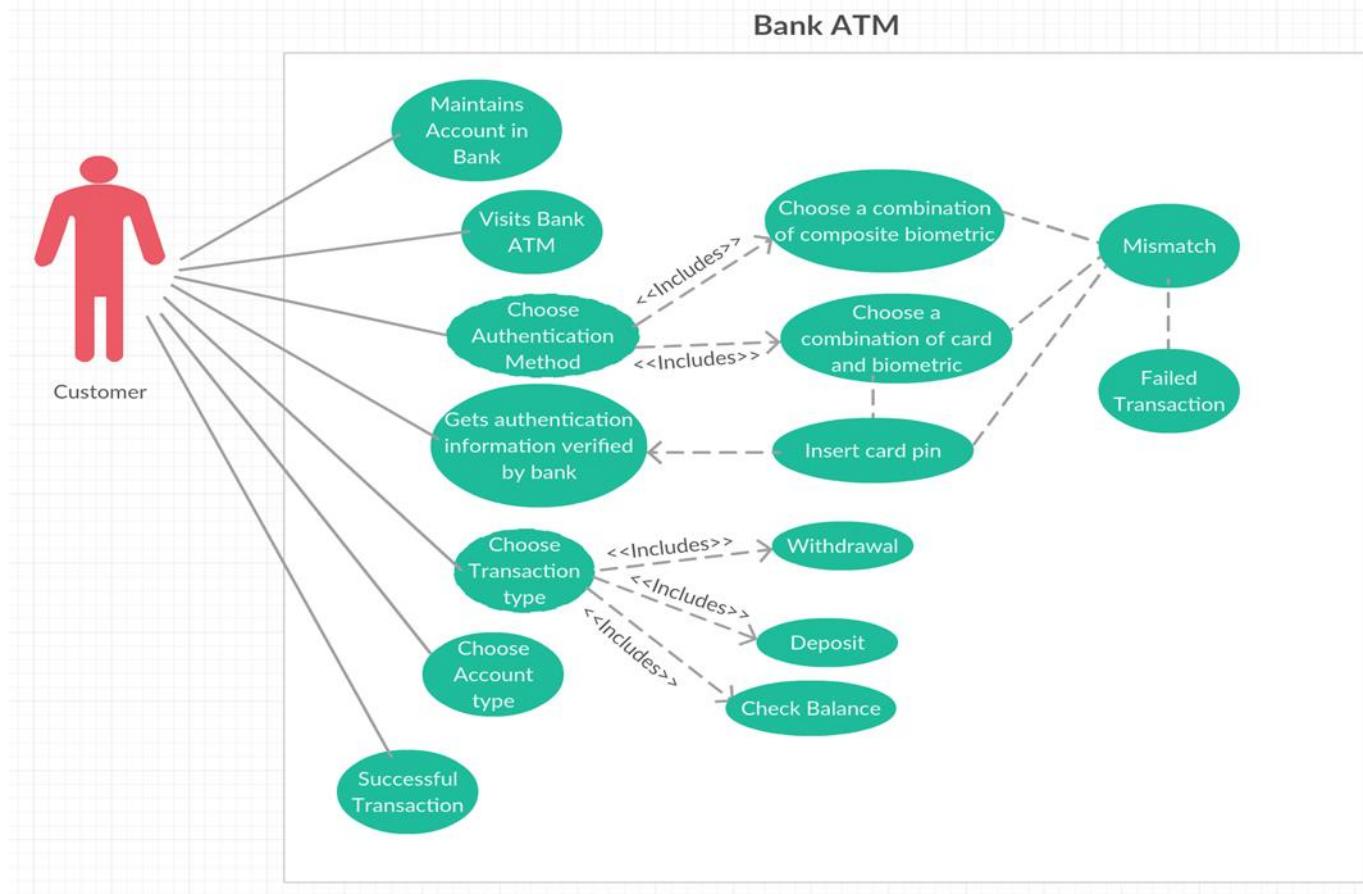


Figure (a)

- Customer, first, registers and opens an account in the bank. For direct transactions, he visits the bank ATM and chooses one set of authentication methods.
- Either he can choose a combination of a composite biometrics or he can select one biometric along with providing card details.
- Bank authenticates the information and as soon as there is a mismatch, the customer is prompted to start over and transactions fails. But in case of matching of the results with the one stored in the bank, the customer is then given liberty to choose the type of transaction.
- There are three types of transactions – withdrawal, deposit and checking the balance. The customer then chooses his account (if he has more than one account in that bank) and the transaction status is said to be completed or successful.

(b) BANK ADMIN ROLE

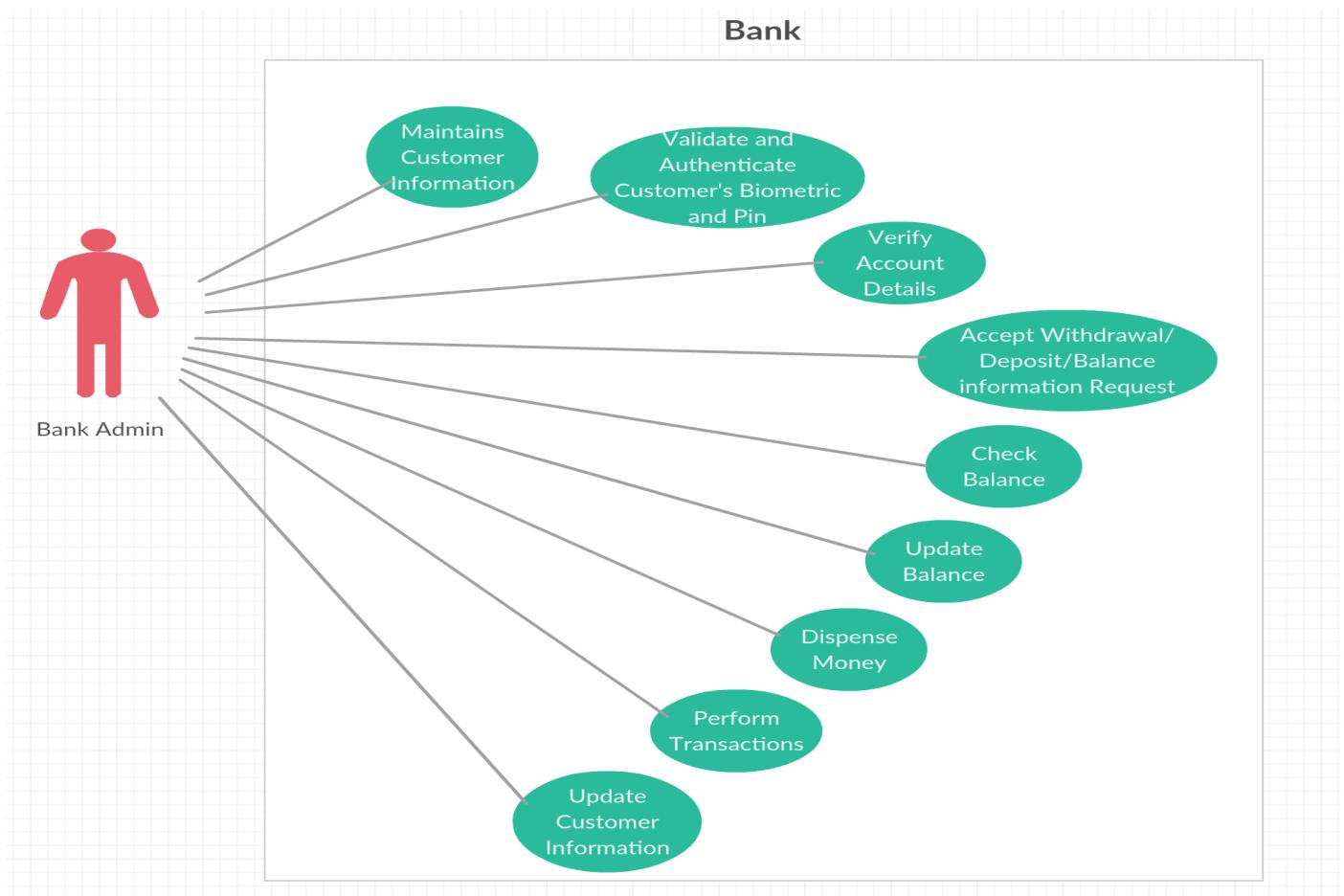


Figure (b)

- The bank admin maintains the customer information - validate and authenticate Customer's biometric and pin whenever customer is expected to make a transaction.
- The bank admin is responsible for verifying account details of the customer.
- Admin has the authority to accept or reject withdrawal, deposit or balance information request to be provided to the customer.
- The admin can check or update the balance and it can also dispense the money. The transactions are performed by the customer and if it is a successful transaction, the customer information is updated with the particular transaction.

3.2 Entity Relationship Diagram:

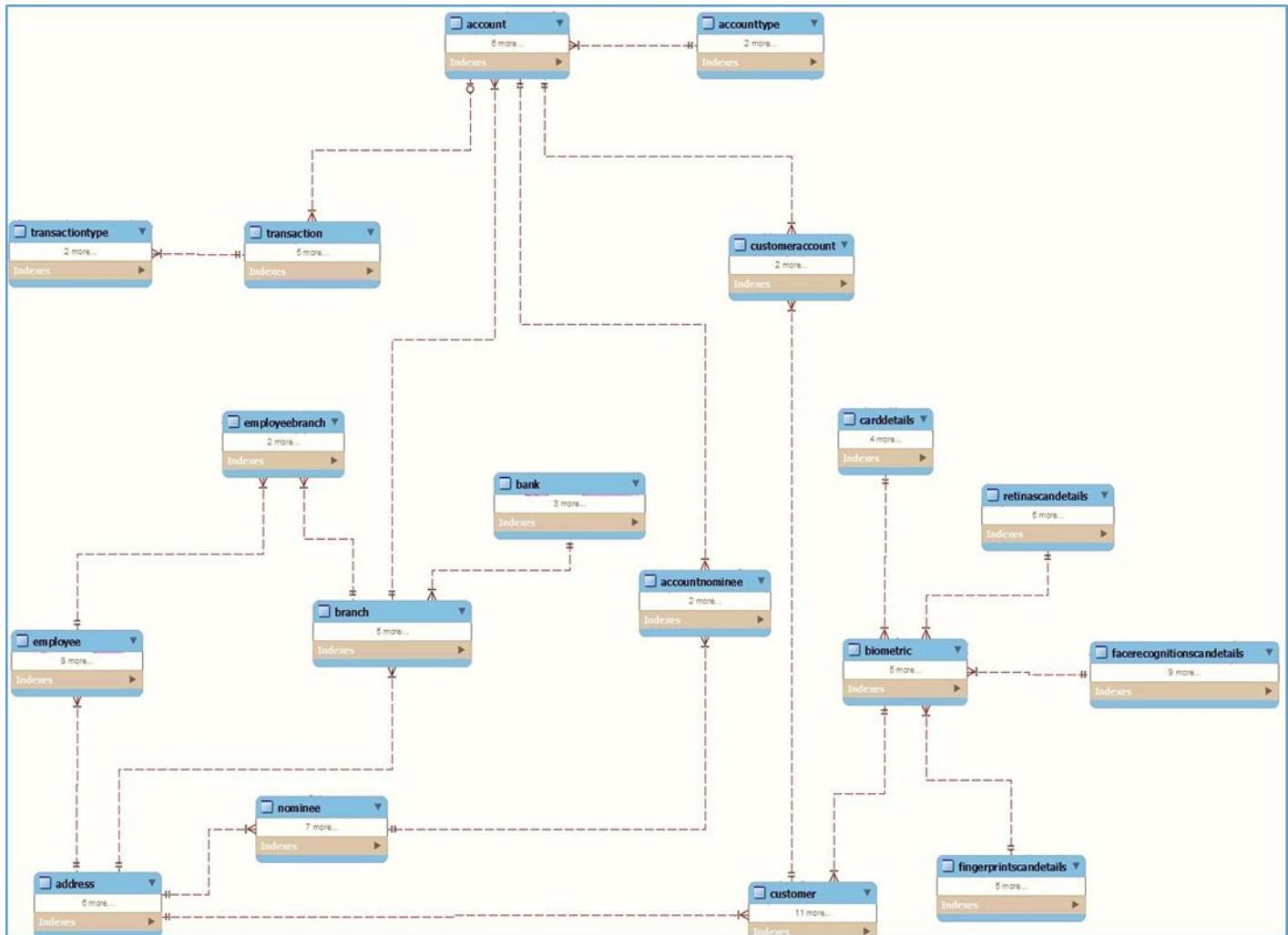


Figure 3.2 Entity Relationship Diagram

The main entities in the project are

1. Bank
2. Branch
3. Customer
4. Employee
5. Accounts
6. Biometric
7. Nominee
8. Transaction

These entities are connected to each other with the following relationships

1. Bank and Branch has one to many one relationship.
2. Branch and customer has many to many relationship
3. Branch and employee has one to many relationship
4. Branch and account has one to many relationship

5. Customer has one to many relationship with biometric
6. Customer has one to one relationship with account
7. Account has many to many relationship with nominee
8. Account has one to many relationship with the transaction

As seen above there are many entities which have many to many relationship with each other, thus to link them we have made following linking tables

1. Employee Branch Table
2. Customer Account Table
3. Account Nominee Table

For the purpose of normalization, the following has been done

1. An Address table has been added which contains detailed information about addresses of all customers and employees
2. Transaction type table has been added giving more details about transaction
3. Account type table has been added giving info of the type of account
4. Biometric table has further been broken down into Finger Print Details, Retina Scan details, Facial Recognition details and Card Details giving detailed information about each biometric, for the purpose of normalization.

3.3 Enhanced Entity Relationship Diagram:

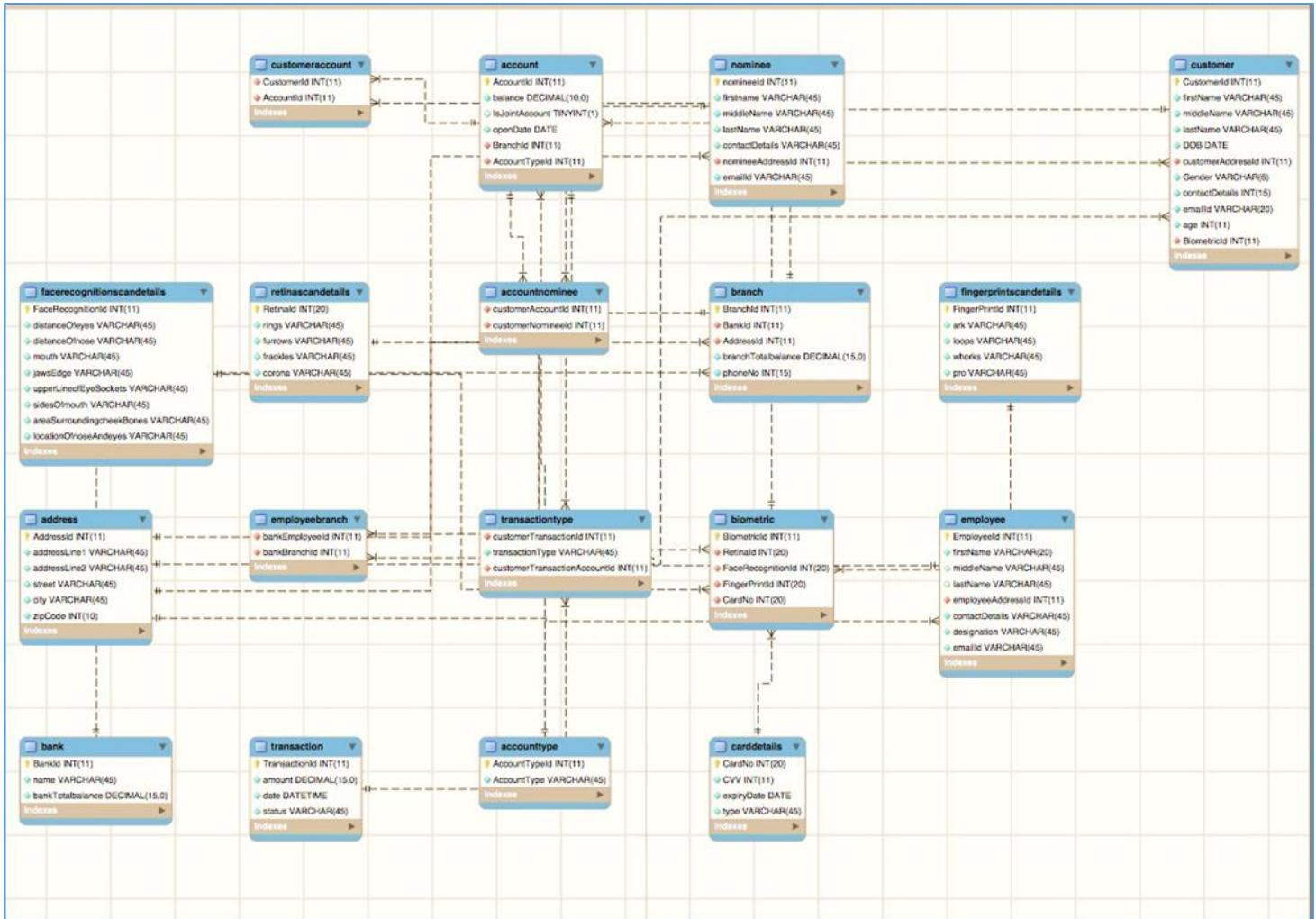


Figure 3.3 Enhanced Entity Relationship Diagram

CHAPTER – 4

4.1 Encryption:

Highly transactional databases are prone to database attacks. SQL injection is a [code injection](#) technique, used to [attack](#) data-driven applications, in which malicious SQL statements are inserted into an entry field for execution. SQL injection is mostly known as an attack [vector](#) for websites but can be used to attack any type of SQL database.

SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, and become administrators of the database server.

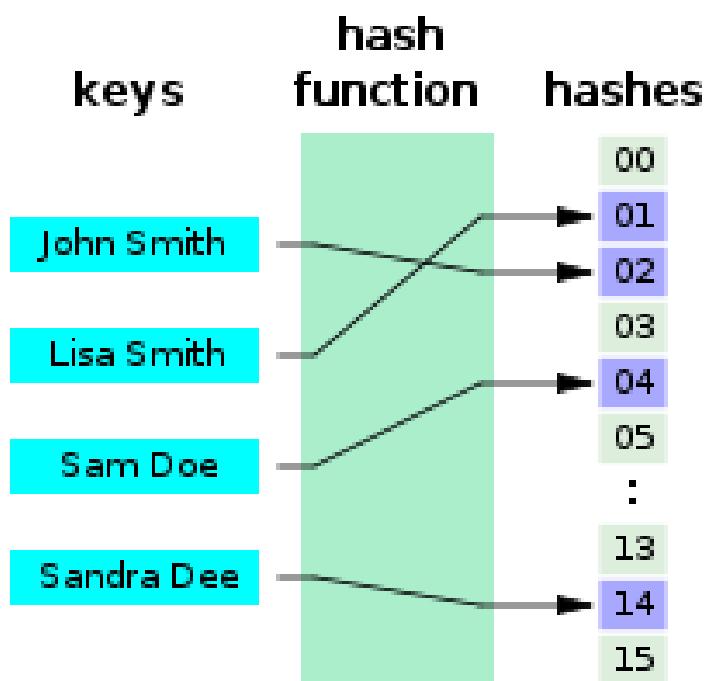
The solution to secure the sensitive data in database is Encryption.

4.2 Encryption Technique:

The technique used in the project is SHA-2.

SHA -2 Algorithm: Secure Hash Algorithm 2 uses a set of cryptographic hash functions which are collision resistant. SHA-256 and SHA-512 are novel hash functions computed with 32-bit and 64-bit words, respectively.

This technique improves the robustness of secured transactions. The SHA-2 family consists of six hash functions with digests.



4.3 Database Encryption:

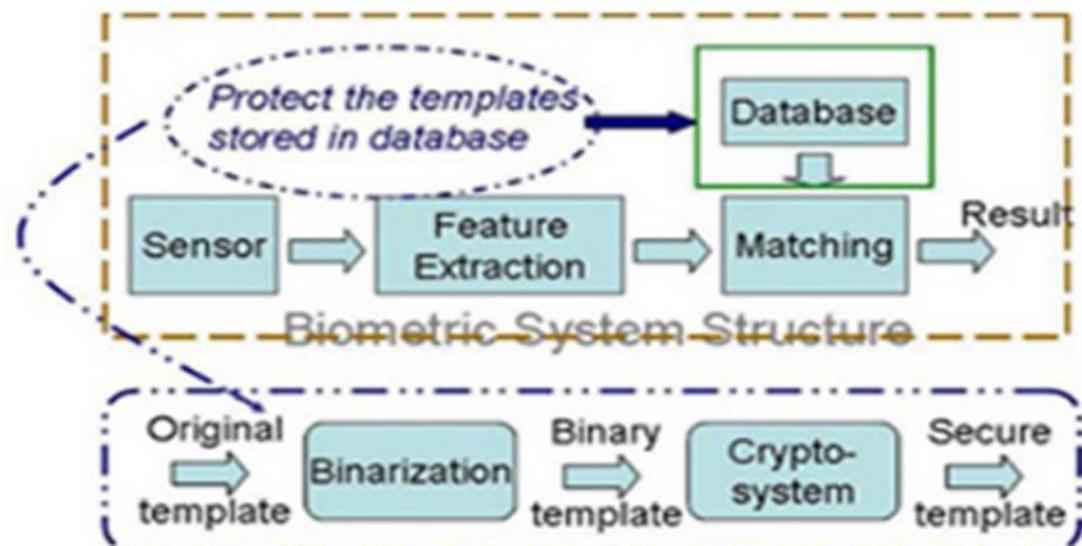


Figure 4.3 (a) Process of encryption

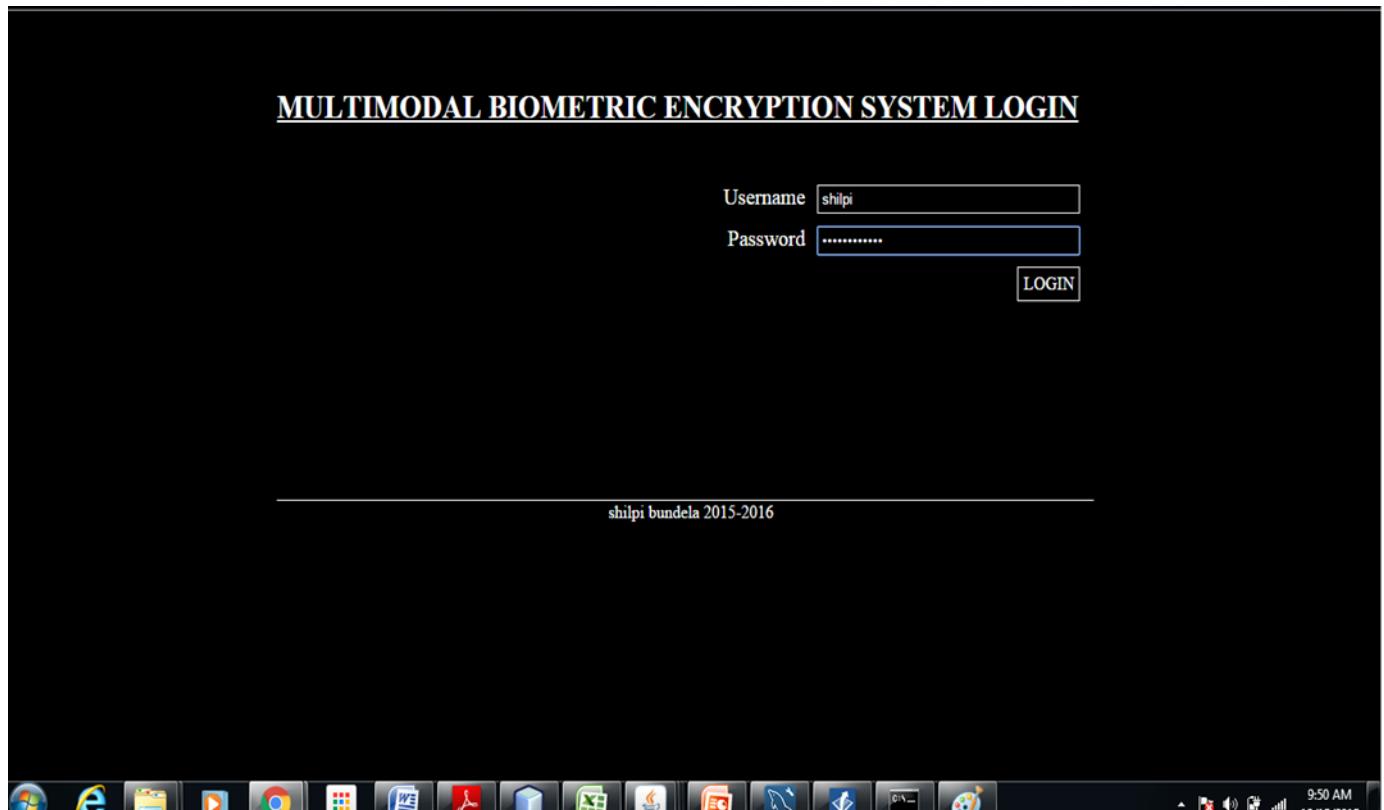


Figure 1.6 (b) : GUI for the encryption system login

MULTIMODAL BIOMETRIC ENCRYPTION SYSTEM

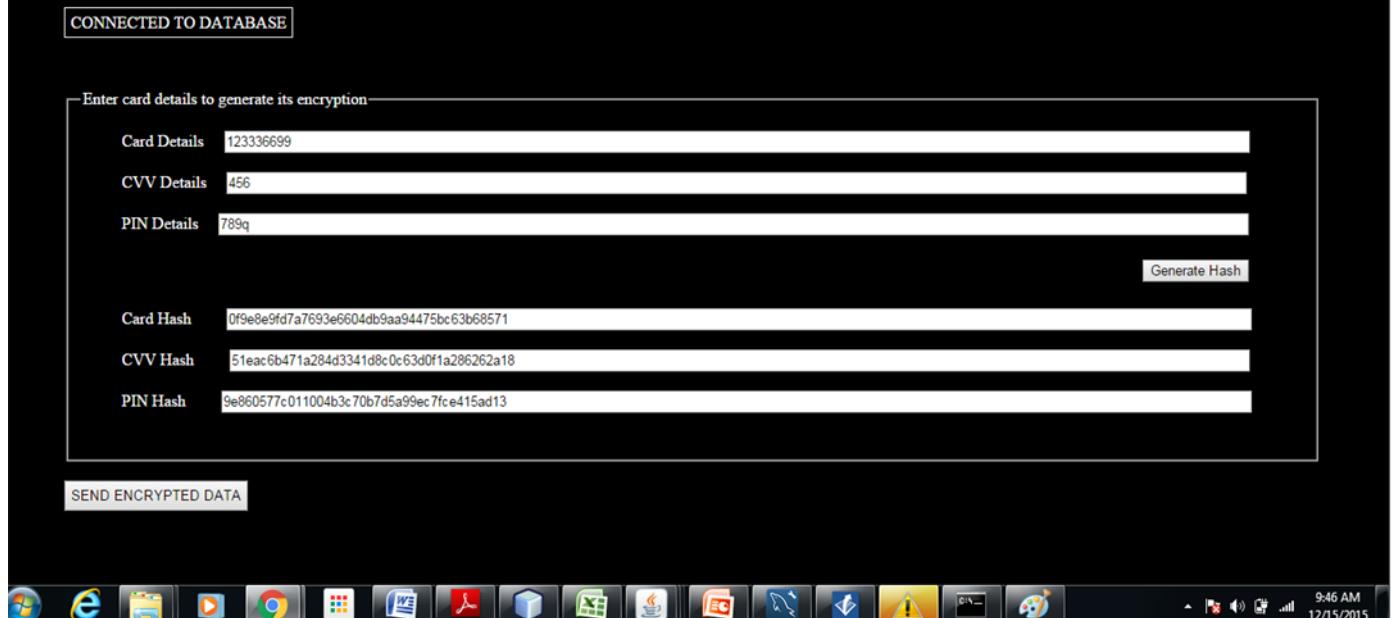


Figure 4.3 (c) : Enter the card details to encrypt

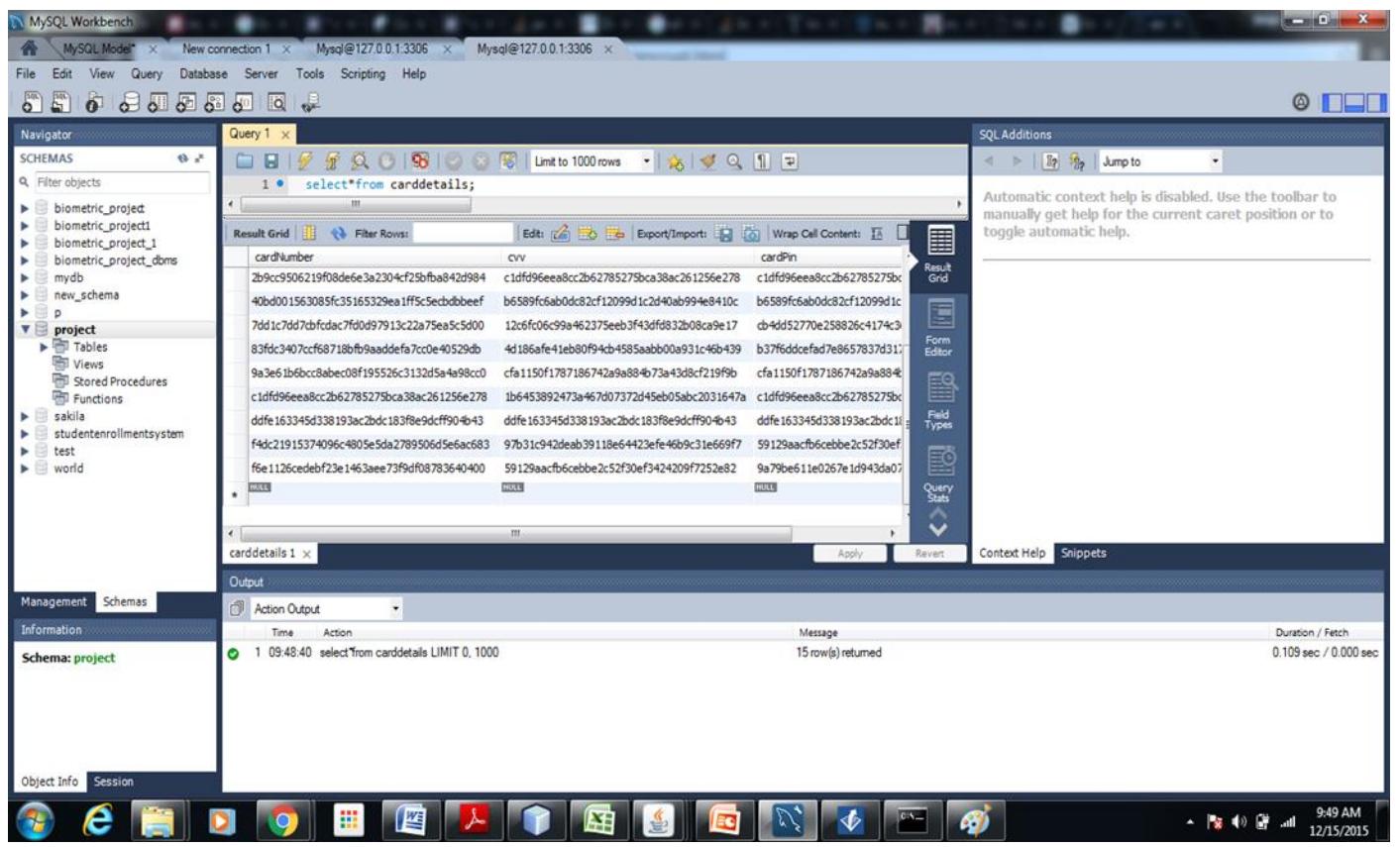


Figure 4.3 (d) : Database that will store the encrypted information in hash codes

CHAPTER – 5

5.1 Tables and Attributes:

Attributes / Properties

Attributes (also called properties) describe the characteristics of the entity, thus giving it definition. The table properties include the physical name and conceptual name (which in this case are the same), columns, primary key, and indexes. Properties for columns include the physical name, display name, data type and length, and whether or not it is a primary key and or required. The attributes of our ingredients are self-describing, and therefore make transaction easier to understand. A transactional database has to be highly normalized. The project schema tables are normalized to 3NF. All attributes in the table are defined by the candidate keys.

1) Account:

<u>AccountID</u>	Balance	IsJointAccount	OpenDate	BranchID	AccountTypeID

2) Account Type:

AccountTypeID	AccountType
---------------	-------------

3) Account Nominee:

CustomerAccountID	CustomerNomineeID
-------------------	-------------------

4) Address:

AddressID	AddressLine1	AddressLine2	Street	City	ZipCode

5) Bank:

Bank ID	Name	BankTotalBalance
---------	------	------------------

6) Biometric:

BiometricID	RetinaID	FaceRecognitionID	FingerprintID	CardNo.
-------------	----------	-------------------	---------------	---------

7) Branch:

BranchID	BankID	AddressID	BranchTotalBalance	PhoneNo.
----------	--------	-----------	--------------------	----------

8) Card Details:

CardNo.	CVV	ExpiryDate	Type
---------	-----	------------	------

9) Customer:

CustomerID	FirstName	MiddleName	LastName	DOB	CustomerAddressID
------------	-----------	------------	----------	-----	-------------------

Gender	ContactDetails	EmailID	Age	BiometricId
--------	----------------	---------	-----	-------------

10) Customer Account:

CustomerAccountID	AccountID
-------------------	-----------

11) Employee:

EmployeeID	FirstName	MiddleName	LastName	EmployeeAddressID	ContactDetails
------------	-----------	------------	----------	-------------------	----------------

Designation	EmailID
-------------	---------

12) Employee Branch:

BankEmployeeID	BankBranchID
----------------	--------------

13) Face Recognition Scan Details

FaceRecognitionID	DistanceofEyes	DistanceofNose	Mouth	JawsEdge
-------------------	----------------	----------------	-------	----------

UpperLineofEyeSocket	SidesofMouth	AreaSurroundingCheek	LocationofNoseAndEyes
----------------------	--------------	----------------------	-----------------------

14) Fingerprint Scan Details:

FingerprintID	Ark	Loops	Whorks	Pro
---------------	-----	-------	--------	-----

15) Nominee:

NomineeID	FirstName	MissdleName	Last Name	Contact Details	NomineeAddressID	Email ID
-----------	-----------	-------------	-----------	-----------------	------------------	----------

16) Retina Scan Details:

RetinaID	Rings	Furrows	Frackles	Corona
----------	-------	---------	----------	--------

17) Transaction:

TransactionID	Amount	Date	Status
---------------	--------	------	--------

18) Transaction Type:

CustomerTransactionID	TransactionType	CustomerTransactionAccountID
-----------------------	-----------------	------------------------------

5.2 Relationships and Cardinalities:

- Relationships describe how entities are associated with each other. There are three types of relationships, one to one, one to many, and many to many. These relationships define how entity instances relate.
- Relationships also provide us with critical information about entity interactions.
- In general, each relationship can also have a minimum and maximum cardinality associated with it to show which relationships are optional, which are mandatory and any numerical limitations on these relationships.

Following list shows the important relationships with their cardinalities in Multimodal Biometric Transaction System database design.

- 1) Account and AccountType relationship (One to One)
- 2) Account and CustomerAccount (One to One)
- 3) Account and AccountNominee (One to Many)
- 4) Branch and Account (One to Many)
- 5) Account & Transaction (One to Many)
- 6) EmployeeBranch and Employee (One to Many)
- 7) Bank and Branch (One to Many)
- 8) Customer & Biometrics (One to One)
- 9) AccountNominee and Account (One to Many)
- 10) Employee and Address (One to One)
- 11) Transaction and TransactionType (One to One)

5.3 Queries:

5.3.1 Index:

(a) Index on nominee table

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database structure, including the **biometric_project1** schema which contains **Tables**, **Views**, and **Stored Procedures**.
- Query Editor:** Displays the SQL code for creating an index and selecting data from the **nominee** table.

```
16
17
18 • create index index_nominee on nominee
19   (nomineeId, firstname);
20
21 • select nomineeId as NomineeID,firstname as FirstName from nominee;
22
23
24
```

- Result Grid:** Shows the results of the query, listing 16 rows of data with columns **NomineeID** and **FirstName**.

	NomineeID	FirstName
1	Janet	
2	Thomas	
3	Jack	
4	Timothy	
5	Jeffrey	
6	Victor	
7	Ruby	
8	Deborah	
9	Kathleen	
10	Eugene	
11	Alan	
12	Daniel	
13	Christine	
14	Juan	
15	Louise	
16	Elizabeth	

- SQL Additions:** A panel on the right containing links for **Result Grid**, **Form Editor**, **Field Types**, **Query Stats**, and **Execution Plan**.

(b) Index on Account table

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database structure, including the **biometric_project1** schema which contains **Tables**, **Views**, and **Stored Procedures**.
- Query Editor:** Displays the SQL code for creating an index and selecting data from the **account** table.

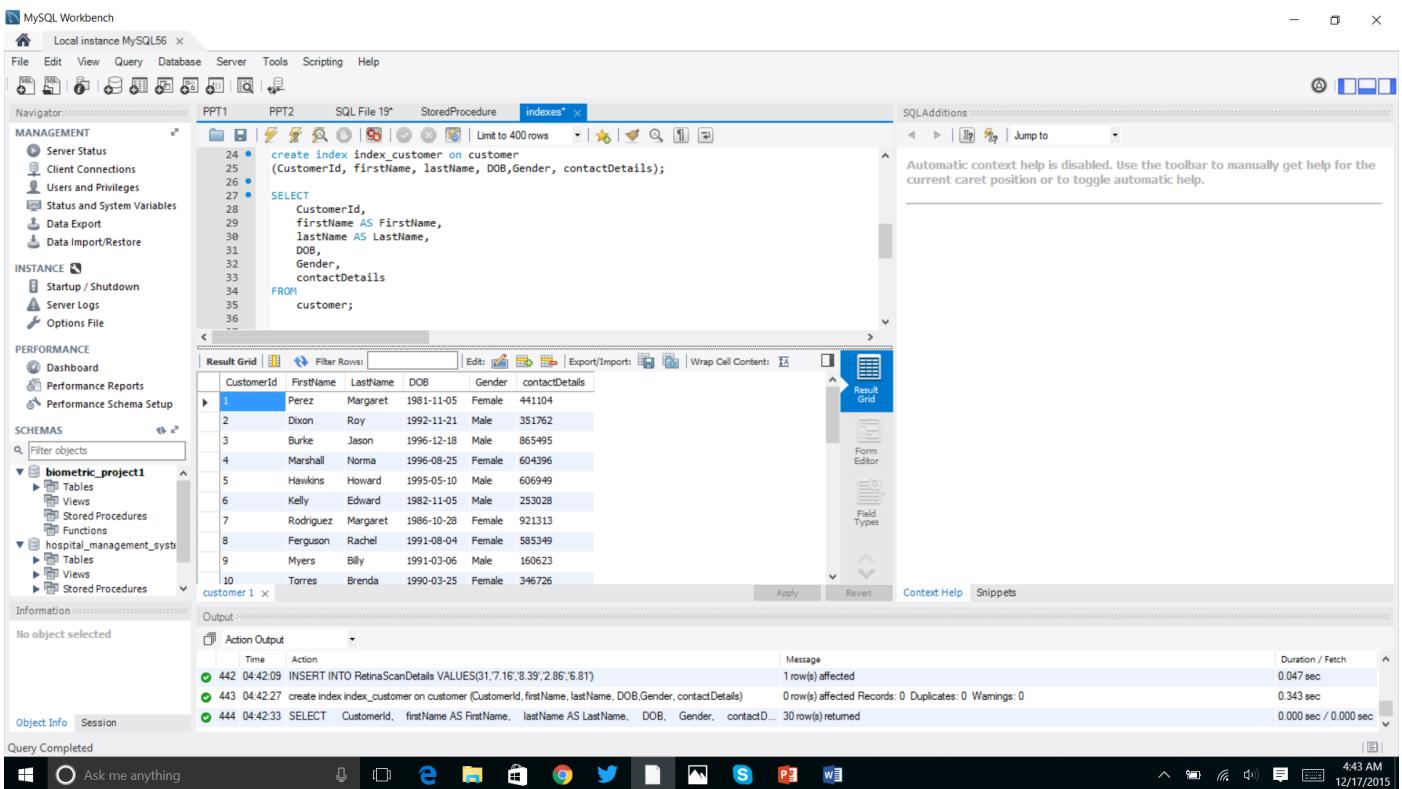
```
24
25 • create index index_account on account
26   (AccountId, balance, openDate, BranchId);
27
28 • SELECT
29   AccountId AS AccountID,
30   balance AS Balance,
31   openDate AS OpenDate,
32   BranchId
33   FROM
34     account;
35
36
```

- Result Grid:** Shows the results of the query, listing 13 rows of data with columns **AccountId**, **Balance**, **OpenDate**, and **BranchId**.

	AccountId	Balance	OpenDate	BranchId
1	10607	2007-01-01	1	
2	6445	2001-10-01	2	
3	7062	2005-02-01	3	
4	6714	2010-11-01	4	
5	3437	2015-01-11	5	
6	5270	2013-05-01	1	
7	5866	2011-08-15	2	
8	2457	2010-04-10	5	
9	8973	2005-02-01	3	
10	8398	2013-05-01	1	
11	1492	2010-11-01	4	
12	2297	2015-01-11	5	
13	3015	2013-05-01	1	

- SQL Additions:** A panel on the right containing links for **Result Grid**, **Form Editor**, **Field Types**, **Query Stats**, and **Execution Plan**.

(c) Index on Customer table



The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Standard MySQL icons for connection, schema, table, view, stored procedure, and help.
- Navigator:** MANAGEMENT, INSTANCE, SCHEMAS (biometric_project1, hospital_management_system), and Information sections.
- SQL Editor:** Tab titled "indexes" containing the following SQL code:

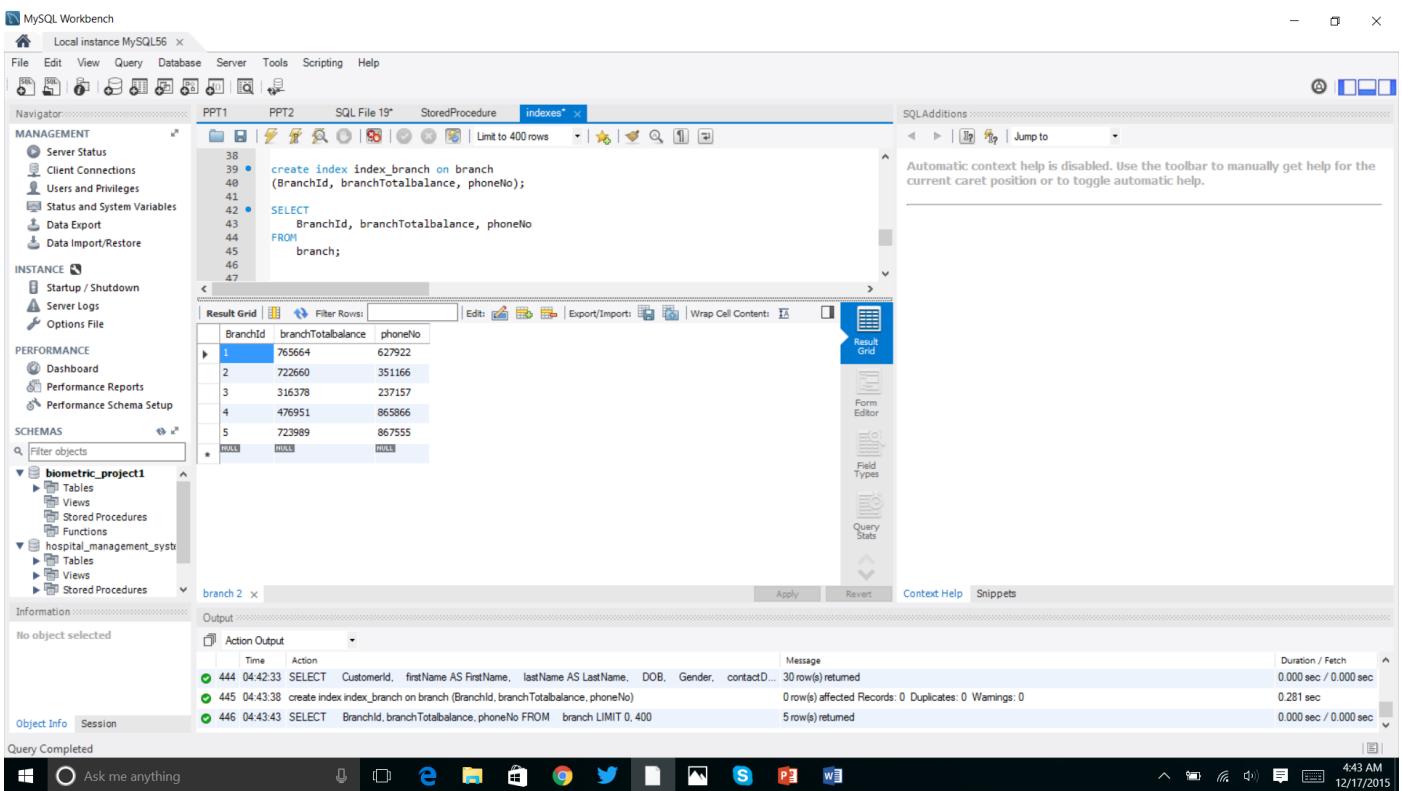

```

24 • create index index_customer on customer
25   (CustomerId, firstName, lastName, DOB, Gender, contactDetails);
26 •
27 •
28   SELECT
29     CustomerId,
30     firstName AS FirstName,
31     lastName AS LastName,
32     DOB,
33     Gender,
34     contactDetails
35
36   FROM
37     customer;
      
```
- Result Grid:** Shows 10 rows of data from the customer table:

CustomerId	FirstName	LastName	DOB	Gender	contactDetails
1	Perez	Margaret	1981-11-05	Female	44104
2	Dixon	Roy	1992-11-21	Male	351762
3	Burke	Jason	1996-12-18	Male	865495
4	Marshall	Norm	1996-08-25	Female	604396
5	Hawkins	Howard	1995-05-10	Male	606949
6	Kelly	Edward	1982-11-05	Male	253028
7	Rodriguez	Margaret	1986-10-28	Female	921313
8	Ferguson	Rachel	1991-08-04	Female	585349
9	Myers	Billy	1991-03-06	Male	160623
10	Torres	Brenda	1990-03-25	Female	346726
- Output:** Shows the history of actions:

Action	Time	Message	Duration / Fetch
442	04:42:09	INSERT INTO RetinaScanDetails VALUES(31,7.16,'8.39','2.86','6.81)	0.047 sec
443	04:42:27	create index index_customer on customer (CustomerId,firstName,lastName,DOB,Gender,contactDetails)	0.343 sec
444	04:42:33	SELECT CustomerId, firstName AS FirstName, lastName AS LastName, DOB, Gender, contactD... 30 row(s) returned	0.000 sec / 0.000 sec

(d) Index on Branch table



The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Standard MySQL icons for connection, schema, table, view, stored procedure, and help.
- Navigator:** MANAGEMENT, INSTANCE, SCHEMAS (biometric_project1, hospital_management_system), and Information sections.
- SQL Editor:** Tab titled "indexes" containing the following SQL code:


```

38 •
39 • create index index_branch on branch
40   (BranchId, branchTotalBalance, phoneNo);
41 •
42 •
43   SELECT
44     BranchId, branchTotalBalance, phoneNo
45
46   FROM
47     branch;
      
```
- Result Grid:** Shows 5 rows of data from the branch table:

BranchId	branchTotalBalance	phoneNo
1	765664	627922
2	722660	351166
3	316378	237157
4	476951	865866
5	723989	867555
- Output:** Shows the history of actions:

Action	Time	Message	Duration / Fetch
444	04:42:33	SELECT CustomerId, firstName AS FirstName, lastName AS LastName, DOB, Gender, contactD... 30 row(s) returned	0.000 sec / 0.000 sec
445	04:43:38	create index index_branch on branch (BranchId,branchTotalBalance,phoneNo)	0.281 sec
446	04:43:43	SELECT BranchId,branchTotalBalance,phoneNo FROM branch LIMIT 0,400	0.000 sec / 0.000 sec

(e) Index on Employee table

The screenshot shows the MySQL Workbench interface. In the top-left pane, the Navigator displays the database schema with tables like biometric_project1 and hospital_management_system. The central pane shows a SQL editor with the following code:

```

48 • create index index_employee on employee
49   (EmployeeId, firstName, lastName, contactDetails, designation);
50
51 • SELECT
52   EmployeeId, firstName, lastName, contactDetails, designation
53   FROM
54     employee;
55
56

```

Below the code is a Result Grid showing ten rows of employee data:

EmployeeId	firstName	lastName	contactDetails	designation
1	Donna	Kelly	75278559754	Desktop Support Technician
2	Ryan	Barnes	626641912431	Accounting Assistant II
3	George	Romero	380665438418	Financial Analyst
4	George	Little	867205584934	Project Manager
5	Judy	Meyer	637167196644	Geologist II
6	Shirley	Cooper	636707242298	Environmental Tech
7	Lori	Hernandez	869961354734	Paralegal
8	Emily	Lawrence	556551009545	Account Executive
9	Joan	Mason	337537253218	Quality Control Specialist
10	Nicole	Tucker	586103988664	Business Systems Development Analyst

The bottom pane shows the Output window with three log entries:

- 446 04:43:43 SELECT BranchId, branchTotalBalance, phoneNo FROM branch LIMIT 0,400
- 447 04:44:15 create index index_employee on employee (EmployeeId,firstName,lastName,contactDetails,designation)
- 448 04:44:20 SELECT EmployeeId,firstName,lastName,contactDetails,designation FROM employee LIMIT 0,400

5.3.2 View:

(a) View to show customer details

The screenshot shows the MySQL Workbench interface. In the top-left pane, the Navigator displays the database schema with tables like biometric and biometric_project1. The central pane shows a SQL editor with the following code:

```

1 • CREATE VIEW view_customerdetails AS
2   SELECT
3     c1.customerId,
4     c1.firstName AS FirstName,
5     c1.lastName AS LastName,
6     a1.balance AS Balance,
7     b1.BranchId,
8     a1.addressLine1 AS AddressLine1,
9     a1.addressLine2 AS AddressLine2,
10    a1.zipCode AS ZipCode
11
12   FROM
13     (((customer c1
14       INNER JOIN customeraccount ac1 ON c1.CustomerId = ac1.customerId)
15       INNER JOIN account a1 ON ac1.accountId = a1.accountId)
16       INNER JOIN branch b1 ON a1.branchId = b1.BranchId)
17       INNER JOIN address ad1 ON c1.customerAddressId = ad1.AddressId)
18
19   WHERE
20     b1.BranchId = 1
21   ORDER BY balance DESC;
22
23 • SELECT
24   *
25   FROM
26     view_customerdetails;

```

Below the code is a Result Grid showing four rows of customer data:

customerId	FirstName	LastName	Balance	BranchId	AddressLine1	AddressLine2	ZipCode
30	Garrett	Antonio	9607	1	9 Walton Park	Lane	1850933
9	Myers	Billy	8850	1	97 Village Terrace	Drive	10956791
21	Long	Peter	8398	1	41 Jenifer Circle	Point	489513083
17	Franklin	George	8187	1	16 Golf Course Road	Road	597304203

The bottom pane shows the Output window with one log entry:

- Formatted 1 statements.

(b) View to show account nominee

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** MANAGEMENT (Server Status, Client Connections, Users and Privileges, Status and System Variables, Data Export, Data Import/Restore); INSTANCE (Startup / Shutdown, Server Logs, Options File); PERFORMANCE (Dashboard, Performance Reports, Performance Schema Setup).
- Schemas:** biometric, biometric_project1 (Tables, Views, Stored Procedures, Functions).
- Current Query:** customerQuery*
- Result Grid:** Displays the results of the query "SELECT * FROM view_Customer;". The data is as follows:

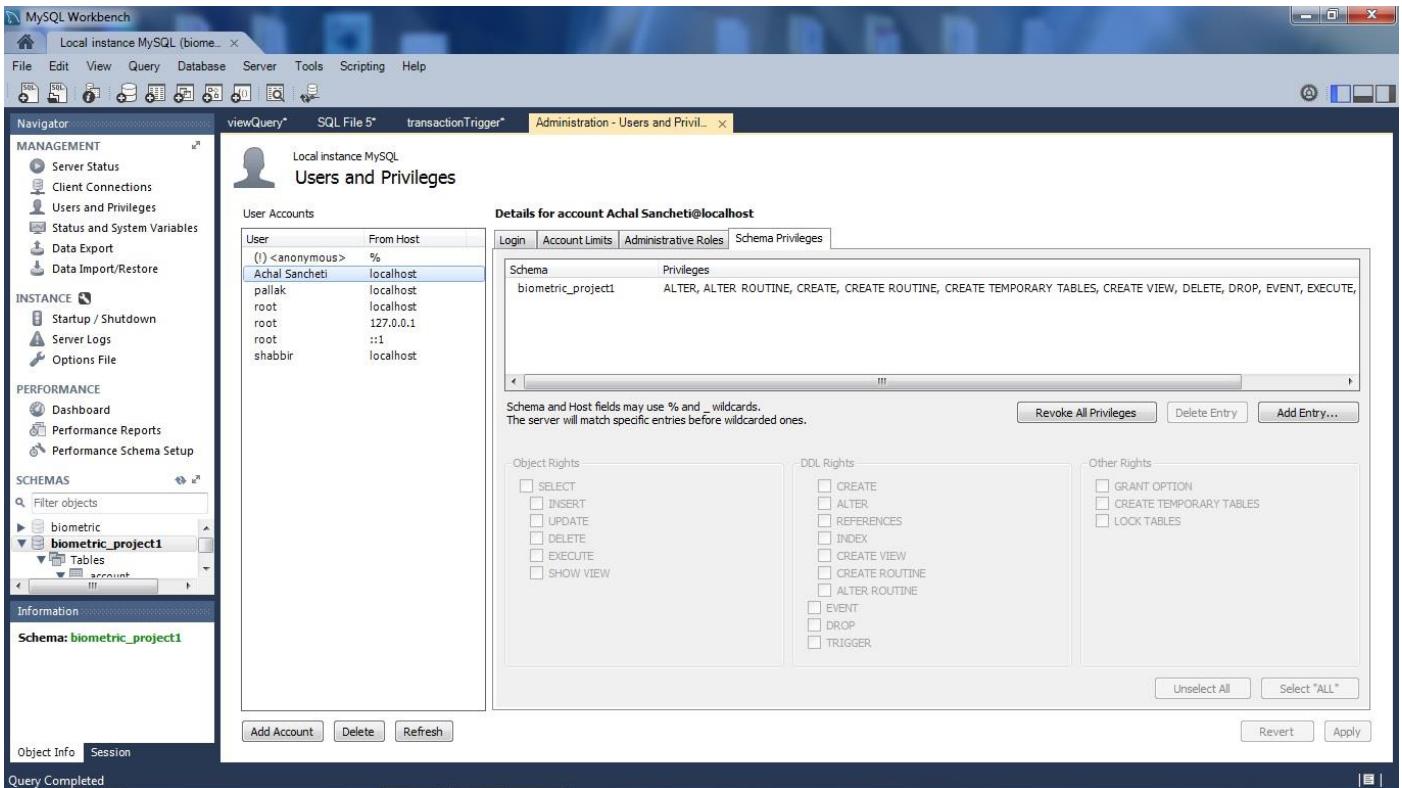
customerId	CustomerName	NomineeName
1	Perez Margaret	Janet Harvey
2	Dixon Roy	Thomas Bennett
3	Burke Jason	Jack Jacobs
4	Marshall Norma	Timothy Spencer
5	Hawkins Howard	Jeffrey Burton
6	Kelly Edward	Victor Mason
7	Rodriguez Margaret	Ruby Gonzalez

- Output:** Action Output table showing the history of actions taken on the view.

Time	Action	Message	Duration / Fetch
32 17:50:55	SELECT * FROM view_customerdetails LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec
33 17:51:57	SELECT * FROM view_Customer LIMIT 0, 1000	30 row(s) returned	0.000 sec / 0.000 sec

5.3.3 User and Privileges:

(a) Admin Privilege



(b) Customer Privilege

The screenshot shows the MySQL Workbench interface under the 'Administration - Users and Privileges' tab. The left sidebar displays various management and performance metrics. The main panel shows the 'Users Accounts' table with the following data:

User	From Host
(!) <anonymous>	%
Achal Sancteti	localhost
pallak	localhost
root	localhost
root	127.0.0.1
root	::1
shabbir	localhost

In the 'Schema Privileges' tab, the 'biometric_project1' schema has the 'SHOW VIEW' privilege granted to the 'pallak' user. The 'Object Rights' section includes checkboxes for SELECT, INSERT, UPDATE, DELETE, EXECUTE, and SHOW VIEW. The 'DDL Rights' section includes checkboxes for CREATE, ALTER, REFERENCES, INDEX, CREATE VIEW, CREATE ROUTINE, ALTER ROUTINE, EVENT, DROP, and TRIGGER. The 'Other Rights' section includes checkboxes for GRANT OPTION, CREATE TEMPORARY TABLES, and LOCK TABLES.

(c) Employee Privilege

The screenshot shows the MySQL Workbench interface under the 'Administration - Users and Privileges' tab. The left sidebar displays various management and performance metrics. The main panel shows the 'Users Accounts' table with the following data:

User	From Host
(!) <anonymous>	%
Achal Sancteti	localhost
pallak	localhost
root	localhost
root	127.0.0.1
root	::1
shabbir	localhost

In the 'Schema Privileges' tab, the 'biometric_project1' schema has the 'CREATE, INDEX, INSERT, SELECT' privilege granted to the 'shabbir' user. The 'Object Rights' section includes checkboxes for SELECT, INSERT, UPDATE, DELETE, EXECUTE, and SHOW VIEW. The 'DDL Rights' section includes checkboxes for CREATE, ALTER, REFERENCES, INDEX, CREATE VIEW, CREATE ROUTINE, ALTER ROUTINE, EVENT, DROP, and TRIGGER. The 'Other Rights' section includes checkboxes for GRANT OPTION, CREATE TEMPORARY TABLES, and LOCK TABLES.

5.3.4 Transaction:

The screenshot shows the MySQL Workbench interface. In the central query editor, a transaction script is displayed:

```

1 • START TRANSACTION;
2 • savepoint savepoint1;
3 • insert into transaction values(27,2,1000,'2015/12/02 2:20:45');
4 • insert into transaction values(28,3,1000,'2015/12/02 2:20:45');
5 • insert into transaction values(29,4,1000,'2015/12/02 2:20:45');
6 • insert into transaction values(30,2,1000,'2015/12/02 2:20:45');
7 • insert into transaction values(31,6,1000,'2015/12/02 2:20:45');
8 • insert into transaction values(32,4,1000,'2015/12/02 2:20:45');
9 • insert into transaction values(33,1,1000,'2015/12/02 2:20:45');
10 • SELECT * from transaction;
11 • rollback to SAVEPOINT savepoint1;
12 • insert into transaction values(26,1,1000,'2015/12/02 2:20:45');
13 • commit;

```

The Result Grid shows the data inserted into the transaction table:

transactionId	transactionAccountId	amount	date
27	2	1000	2015-12-02 02:20:45
28	3	1000	2015-12-02 02:20:45
29	4	1000	2015-12-02 02:20:45
30	2	1000	2015-12-02 02:20:45
31	6	1000	2015-12-02 02:20:45
32	4	1000	2015-12-02 02:20:45
33	1	1000	2015-12-02 02:20:45
*	NULL	NULL	NULL

The Output pane shows the execution log:

Action	Time	Message	Duration / Fetch
insert into transaction values(33,1,1000,'2015/12/02 2:20:45')	21:00:03	1 row(s) affected	0.000 sec
SELECT *from transaction LIMIT 0, 1000	21:00:03	32 row(s) returned	0.000 sec

5.3.5 Sub-query:

Sub-query to view customer information

The screenshot shows the MySQL Workbench interface. In the central query editor, a subquery is used to select customers with balances above the average:

```

1 • SELECT
2     c1.customerId,
3     c1.firstName AS FirstName,
4     c1.lastName AS LastName,
5     a1.balance AS Balance,
6     b1.BranchId
7
8     FROM
9     (((customer c1
10    INNER JOIN customeraccount ac1 ON c1.CustomerId = ac1.customerId)
11    INNER JOIN account a1 ON ac1.accountId = a1.accountId)
12    INNER JOIN branch b1 ON a1.branchId = b1.BranchId)
13 WHERE
14     a1.balance > ( SELECT avg(balance) from account )
15 ORDER BY balance DESC;

```

The Result Grid shows the selected customer data:

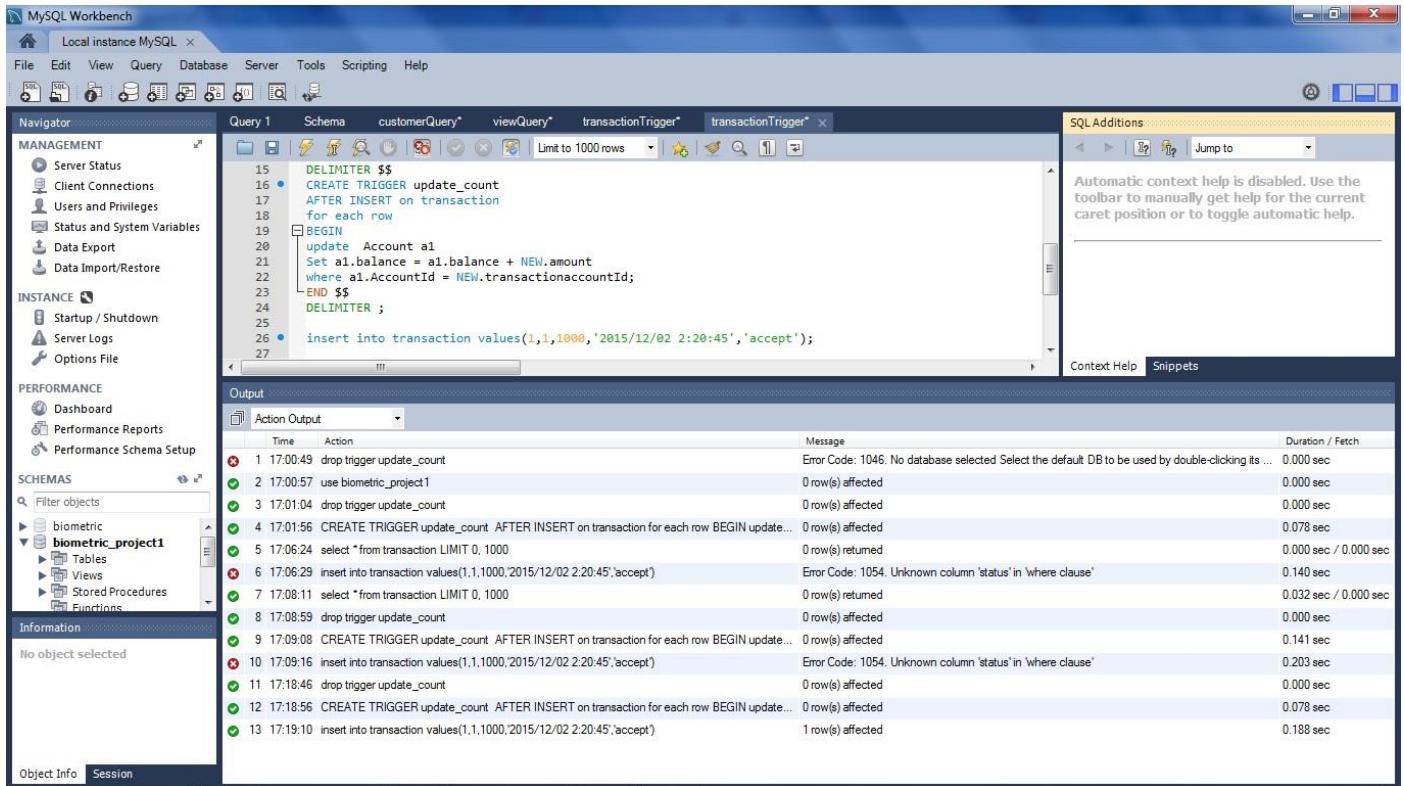
CustomerId	FirstName	LastName	Balance	BranchId
2	Dixon	Roy	9513	4
15	Bishop	Roy	9228	5
22	Turner	Anna	8973	3
9	Myers	Billy	8850	1
5	Hawkins	Howard	8654	5
30	Garrett	Antonio	8407	1
21	Long	Peter	8398	1
17	Franklin	George	8187	1
10	Torres	Brenda	7844	3

The Output pane shows the execution log:

Action	Time	Message	Duration / Fetch
create index_index_employee on employee (EmployeeId,firstName,lastName,contactDetails,designation)	04:44:15	0 rows affected Records: 0 Duplicates: 0 Warnings: 0	0.234 sec
SELECT EmployeeId,firstName,lastName,contactDetails,designation FROM employee LIMIT 0, 400	04:44:20	10 row(s) returned	0.000 sec / 0.000 sec
SELECT c1.customerId, c1.firstName AS FirstName, c1.lastName AS LastName, a1.balanc...	04:50:35	17 row(s) returned	0.000 sec / 0.000 sec

5.3.6 Trigger:

(a) Trigger to update balance for any transaction



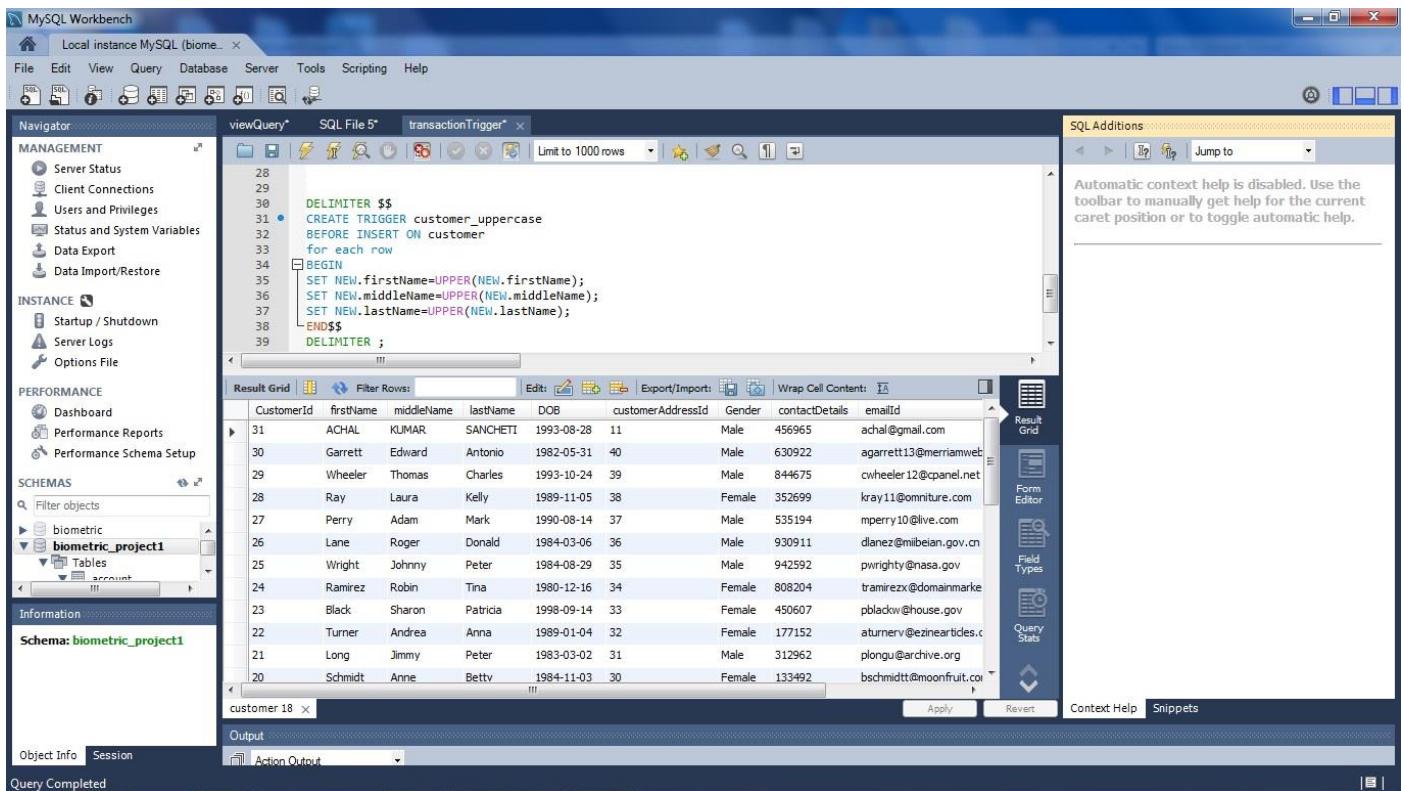
The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database structure, including the schema `biometric_project1` which contains tables like `account`, `transaction`, and `customer`.
- Query Editor:** Contains the SQL code for creating a trigger named `update_count`:


```

15  DELIMITER $$ 
16  CREATE TRIGGER update_count
17  AFTER INSERT on transaction
18  for each row
19  BEGIN
20    update Account a1
21    Set a1.balance = a1.balance + NEW.amount
22    where a1.AccountId = NEW.transactionaccountId;
23  END $$ 
24  DELIMITER ;
25
26  insert into transaction values(1,1,1000,'2015/12/02 2:20:45','accept');
27
      
```
- Output:** Displays the execution log with various actions and their results.

(b) Trigger to change the case of letter on insertion



The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database structure, including the schema `biometric_project1` which contains tables like `customer`.
- Query Editor:** Contains the SQL code for creating a trigger named `customer_uppercase`:


```

28
29
30  DELIMITER $$ 
31  CREATE TRIGGER customer_uppercase
32  BEFORE INSERT ON customer
33  for each row
34  BEGIN
35    SET NEW.firstName=UPPER(NEW.firstName);
36    SET NEW.middleName=UPPER(NEW.middleName);
37    SET NEW.lastName=UPPER(NEW.lastName);
38  END$$ 
39  DELIMITER ;
      
```
- Result Grid:** Displays the data from the `customer` table.

CustomerId	firstName	middleName	lastName	DOB	customerAddressId	Gender	contactDetails	emailId
31	ACHAL	KUMAR	SANCHETI	1993-08-28	11	Male	456965	achal@gmail.com
30	Garrett	Edward	Antonio	1982-05-31	40	Male	630922	agarrett13@merriamweb...
29	Wheeler	Thomas	Charles	1993-10-24	39	Male	844675	cwheeler12@cppanel.net
28	Ray	Laura	Kelly	1989-11-05	38	Female	352699	kray11@omniture.com
27	Perry	Adam	Mark	1990-08-14	37	Male	535194	mperry10@live.com
26	Lane	Roger	Donald	1984-03-06	36	Male	930911	dlanez@milbeian.gov.cn
25	Wright	Johnny	Peter	1984-08-29	35	Male	942592	pwrighty@nasa.gov
24	Ramirez	Robin	Tina	1980-12-16	34	Female	808204	tramirezx@domainmarke...
23	Black	Sharon	Patricia	1998-09-14	33	Female	450607	pblackw@house.gov
22	Turner	Andrea	Anna	1989-01-04	32	Female	177152	aturnerv@ezinearticles.co...
21	Long	Jimmy	Peter	1983-03-02	31	Male	312962	plongu@archve.org
20	Schmidt	Anne	Betty	1984-11-03	30	Female	133492	bschmidtt@moonfruit.co...

5.3.7 Stored Procedure:

(a) Stored Procedure to find highest balance of customer

The screenshot shows the MySQL Workbench interface with the 'storedProcedure1*' tab selected. The code in the query editor is:

```
3 use biometric_project1;
4 create procedure highestCustomerBalance()
5 SELECT c1.firstName as Name, c1.lastName, a1.balance
6 FROM (( customer c1
7 INNER JOIN customeraccount ac1
8 ON c1.CustomerId =ac1.customerId)
9 INNER JOIN account a1
10 ON a1.accountId = ac1.accountId)
11 order by balance desc;
12
13 call highestCustomerBalance()
```

The result grid displays the following data:

Name	lastName	balance
Garrett	Antonio	9607
Dixon	Roy	9513
Bishop	Roy	9228
Turner	Anna	8973
Myers	Billy	8850
Hawkins	Howard	8654
Long	Peter	8398
Franklin	George	8187
Torres	Brenda	7844
Kelly	Edward	7532
Marshall	Norma	7395
Ray	Kelly	7062
Perry	Mark	6714
	Total	6661

(b) Stored Procedure to find second highest balance of customer

The screenshot shows the MySQL Workbench interface with the 'storedProcedure1*' tab selected. The code in the query editor is:

```
19 create procedure secondhighestCustomerBalance()
20 SELECT c1.firstName as Name, c1.lastName, MAX(a1.balance)
21 FROM (( customer c1
22 INNER JOIN customeraccount ac1
23 ON c1.CustomerId =ac1.customerId)
24 INNER JOIN account a1
25 ON a1.accountId = ac1.accountId)
26 where a1.balance < (SELECT MAX(a1.balance))
27 FROM (( customer c1
28 INNER JOIN customeraccount ac1
29 ON c1.CustomerId =ac1.customerId)
30 INNER JOIN account a1
31 ON a1.accountId = ac1.accountId)
32
33 call secondhighestCustomerBalance()
```

The result grid displays the following data:

Name	lastName	MAX(a1.balance)
Perez	Margaret	9513

(c) Stored Procedure to insert data in fingerprintsanddetails table

```

39  DELIMITER $$;
40  USE biometric_project1$$
41  DROP PROCEDURE IF EXISTS sp_fingerprintsanddetails$$
42  CREATE PROCEDURE sp_fingerprintsanddetails(IN FingerPrintIdparam int(11),
43  , IN arkparam VARCHAR(45)
44  , IN loopsparam VARCHAR(45)
45  , IN worksparam VARCHAR(45)
46  , IN proparam VARCHAR(45))
47  BEGIN
48  INSERT INTO fingerprintsanddetails (
49  FingerPrintId
50  , ark
51  , loops
52  , worksp
53  , pro)
54  VALUES (FingerPrintIdparam
55  , arkparam
56  , loopsparam
57  , worksparam
58  , proparam);
59
60  END$$
61  DELIMITER ;
62
63  call sp_fingerprintsanddetails(32, "0.47jhcas6s5","1.46a6g6","8.95a4fh","7.64adr9e7")

```

The Output pane shows the execution of the stored procedure:

Action	Time	Message	Duration / Fetch
EXPLAIN SELECT * FROM view_customerdetails	20 16:13:17	OK	0.000 sec
EXPLAIN FORMAT=JSON SELECT * FROM view_customerdetails	21 16:13:17	OK	0.000 sec
call highestCustomerBalance()	22 16:25:09	30 row(s) returned	0.234 sec / 0.000 sec
call sp_verifier()	23 16:27:11	31 row(s) returned	0.296 sec / 0.000 sec
call sp_fingerprintsanddetails(32, "0.47jhcas6s5","1.46a6g6","8.95a4fh","7.64adr9e7")	24 16:29:27	1 row(s) affected	0.281 sec

(d) Stored Procedure to delete a row from fingerprintsanddetails table

```

68
69  DELIMITER $$;
70  DROP PROCEDURE IF EXISTS delete_sp_fingerprintsanddetails_by_id$$
71  CREATE PROCEDURE delete_sp_fingerprintsanddetails_by_id(IN FingerPrintIdparam int(11))
72  BEGIN
73  delete from fingerprintsanddetails
74  where FingerPrintId=FingerPrintIdparam;
75  END$$
76  DELIMITER ;
77
78  call delete_sp_fingerprintsanddetails_by_id(32)

```

The Output pane shows the execution of the stored procedure:

Action	Time	Message	Duration / Fetch
drop trigger update_count	14 16:09:08	0 row(s) affected	0.000 sec
CREATE TRIGGER update_count AFTER INSERT on transaction for each row BEGIN upda...	15 16:09:19	0 row(s) affected	0.093 sec
select * from update_count LIMIT 0, 1000	16 16:09:29	Error Code: 1146. Table 'biometric_project1.update_count' doesn't exist	0.000 sec
SELECT * FROM view_customerdetails LIMIT 0, 1000	17 16:12:10	8 row(s) returned	0.000 sec / 0.000 sec
SELECT * FROM view_customerdetails LIMIT 0, 1000	18 16:12:37	8 row(s) returned	0.000 sec / 0.000 sec
SELECT * FROM view_TRANSACTION LIMIT 0, 1000	19 16:12:37	1 row(s) returned	0.000 sec / 0.000 sec
EXPLAIN SELECT * FROM view_customerdetails	20 16:13:17	OK	0.000 sec
EXPLAIN FORMAT=JSON SELECT * FROM view_customerdetails	21 16:13:17	OK	0.000 sec
call highestCustomerBalance()	22 16:25:09	30 row(s) returned	0.234 sec / 0.000 sec
call sp_verifier()	23 16:27:11	31 row(s) returned	0.296 sec / 0.000 sec
call sp_fingerprintsanddetails(32, "0.47jhcas6s5","1.46a6g6","8.95a4fh","7.64adr9e7")	24 16:29:27	1 row(s) affected	0.281 sec
call highestCustomerBalance()	25 16:33:49	30 row(s) returned	0.000 sec / 0.000 sec
call secondhighestCustomerBalance()	26 16:36:11	1 row(s) returned	0.203 sec / 0.000 sec
call delete_sp_fingerprintsanddetails_by_id(32)	27 16:37:45	1 row(s) affected	0.234 sec

(e) Stored Procedure for update data in fingerprintsanddetails table

```

DELIMITER $$
DROP PROCEDURE IF EXISTS update_sp_fingerprintsanddetails_by_id$$
CREATE PROCEDURE update_sp_fingerprintsanddetails_by_id(IN FingerPrintIdparam int(11),
    IN arkparam VARCHAR(45))
BEGIN
    update fingerprintsanddetails
    set ark=arkparam
    where FingerPrintId=FingerPrintIdparam;
END$$
DELIMITER ;
call update_sp_fingerprintsanddetails_by_id(29,"1.87kal75")

```

The screenshot shows the MySQL Workbench interface with the SQL editor tab active. The code above creates a stored procedure named `update_sp_fingerprintsanddetails_by_id` that takes two parameters: `FingerPrintIdparam` (int(11)) and `arkparam` (VARCHAR(45)). It updates the `fingerprintsanddetails` table by setting the `ark` column to `arkparam` where the `FingerPrintId` matches `FingerPrintIdparam`. Finally, it calls the stored procedure with the arguments `(29, "1.87kal75")`.

The Output pane shows the execution history of the session, including the creation of the stored procedure and its execution:

Action	Time	Message	Duration / Fetch
16:09:29 select * from update_count LIMIT 0, 1000		Error Code: 1146. Table biometric_project1.update_count' doesn't exist	0.000 sec
16:12:10 SELECT * FROM view_customerdetails LIMIT 0, 1000		8 row(s) returned	0.000 sec / 0.000 sec
16:12:37 SELECT * FROM view_customerdetails LIMIT 0, 1000		8 row(s) returned	0.000 sec / 0.000 sec
16:12:37 SELECT * FROM view_TRANSACTION LIMIT 0, 1000		1 row(s) returned	0.000 sec / 0.000 sec
16:13:17 EXPLAIN SELECT * FROM view_customerdetails		OK	0.000 sec
16:13:17 EXPLAIN FORMAT=JSON SELECT * FROM view_customerdetails		OK	0.000 sec
16:25:09 call highestCustomerBalance()		30 row(s) returned	0.234 sec / 0.000 sec
16:27:11 call sp_verifier()		31 row(s) returned	0.296 sec / 0.000 sec
16:29:27 call sp_fingerprintsanddetails(32, "0.47jcas65","1.46a6g6","8.95a4h","7.64adr9e7")		1 row(s) affected	0.281 sec
16:33:49 call highestCustomerBalance()		30 row(s) returned	0.000 sec / 0.000 sec
16:36:11 call secondhighestCustomerBalance()		1 row(s) returned	0.203 sec / 0.000 sec
16:37:45 call delete_sp_fingerprintsanddetails_by_id(32)		1 row(s) affected	0.234 sec
16:39:46 call update_sp_fingerprintsanddetails_by_id(29,"1.87kal75")		0 row(s) affected	0.156 sec

(f) Stored Procedure to verify customer using two biometrics

```

CREATE PROCEDURE sp_verifier()
BEGIN
    Select b1.BiometricId, b1.FingerPrintId, b1.retinaId, c1.customerId, a1.AccountId
    from (((biometric b1
    INNER JOIN fingerprintsanddetails f1
    ON b1.FingerPrintId = f1.FingerPrintId
    Left Outer JOIN retinasanddetails r1
    ON b1.retinaId = r1.retinaId)
    Left Outer JOIN customer c1
    ON b1.BiometricId = c1.BiometricId)
    Left Outer join CustomerAccount a1
    On c1.CustomerId = a1.CustomerId)
    order by c1.customerId;
END$$
DELIMITER ;
call sp_verifier()

```

The screenshot shows the MySQL Workbench interface with the SQL editor tab active. The code above creates a stored procedure named `sp_verifier()`. It selects `BiometricId`, `FingerPrintId`, `retinaId`, `customerId`, and `AccountId` from a complex query involving four tables: `biometric` (alias `b1`), `fingerprintsanddetails` (alias `f1`), `retinasanddetails` (alias `r1`), and `customer` (alias `c1`). The query uses multiple joins (INNER JOIN, Left Outer JOIN) and conditions (ON clauses). The results are ordered by `customerId`. Finally, the stored procedure is called.

The Result Grid pane displays the results of the query:

BiometricId	FingerPrintId	retinaId	customerId	AccountId
31	31	31	NULL	NULL
11	11	20	1	30
12	12	19	2	29
13	13	18	3	28
14	14	17	4	27
15	15	16	5	26
16	16	15	6	25
17	17	14	7	24
18	18	13	8	23

CHAPTER – 6

6.1 Backup Strategy:

- Backups can guard against data loss, support compliance and retention needs and protect the business from disaster.
- Back up strategy used for project is RMAN (Recovery Manager/Oracle Database Manager).
- Fast Recovery Area (FRA) – A disk location in which database can store and manage back up & recovery files.
- Strategy: Back up of FRA sized for 7-day retention on disk using a weekly full, daily incremental back-up and archived logs.
- Weekly tape back -up maintained for 90 days.
- Monthly tape back -up for 7 years.
- RMAN back up optimizes disk utilization space.

CHAPTER – 7

7.1 Key Benefits:

- Uniqueness
- Biometrics is the only means of determining “who” is actually using the system.
- High level of security.
- Very high accuracy.
- Minimizing frauds.
- Card-less feature in ATMs.
- Spoof reduction in cash less transactions.
- Ease of access and convince.
- Easier to use verification solutions.
- Most definitive, real time tool today.
- Universality
- Permanence
- Measurability
- User friendliness
- Comfort

CHAPTER – 8

8.1 Assumptions:

- We have assumed that all the customers maintain their account in a single bank.
- Customer's biometrics are stored in database at the time of account creation.
- Biometric data is stored as random strings in database.

8.2 Future Scope:

- Online transaction scenario can be changed using various smart devices in addition to our database.
- Key Stroke Patterning, Fingerprint palettes and other more behavioral biometrics can be implemented.
- Cars, automobiles and other automatic devices can be operated using biometrics.

CHAPTER – 9

9.1 References:

- <https://dev.mysql.com/doc/refman/5.7/en/innodb-introduction.html>
- [https://technet.microsoft.com/en-us/library/aa933055\(v=sql.80\).aspx](https://technet.microsoft.com/en-us/library/aa933055(v=sql.80).aspx)
- <https://www.daniweb.com/programming/databases/threads/439533/how-to-store-an-image-in-mysql-database>
- www.mysqltutorials.org
- www.w3schools.com
- <http://dev.mysql.com/doc/refman/5.7/en/commit.html>
- <http://www.tutorialspoint.com/mysql/mysql-transactions.htm>