LONDON METROPOLITAN UNIVERSITY

Islington college
(इस्लिङ्टन कलेज)

CC4057NI Introduction to Information Systems

30% Individual Coursework

2020-21 Spring

Student Name: Bisham Kunwor

Group: N1

London Met ID:

College ID: NP01NT4S210034

Assignment Due Date: April 30, 2021

Assignment Submission Date: April 27, 2021

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

Table of Contents

List of Figures

List of Tables

Introduction

Before learning about database let us know what data and information are. Any fact, figure, number, string, picture, video etc. are data. Data does not have meaning on its own. Example of data can be “Ramesh” and “Supervisor”. The two words does not make any sense on their own. Building upon the analogy of data if we relate “Ramesh” and “Supervisor” then we can get information that “Ramesh is supervisor of some work”. When two or more data are related then we can produce information. Information can be developed by using data processing models.

Database

It is the organized collection of data that produces information. Database is structured collection of data that together represent some information. Databases are used to easily manage, modify, store and update current data. Database can contain various data type as per need of an organization. (Morley & Parker, 2015)

Features of Database

- Easy to manage and manipulate data
- Data redundancy gets hugely reduced
- Modifying one data will automatically update others
- Easy to organize similar data
- Each data in database is unique
- Cost effective to manage
- Provides basic security to data

Description of organization

Our organization is a digital wallet payment system (like Esewa, IMEpay, Khalti) with integrated services like utility bill, electricity bill, movie booking etc. Through our database user can access services via using mobile app or computer browser as our organization is independent of platform (OS). Each user creates an account from our organization. The account is of two types; (verified, non-authenticated). These have their own set of restriction in case of transaction and the access to the services. The account gets verified by the organization by validating the id provided by the user while creating the account. The account provides access to all the service provided by our organization. Services have offers (if provided by organization). A single account can access multiple services. After the transaction (payment) the chosen services are available for the user. Our organization lets multiple users (Corporation) use a single account with unrestricted access. Also, a single user can use multiple accounts (e.g., one for personal use, one for business use). Each account can perform unlimited number of transactions. The account can hold certain amount of money provided by the organization rule and not more.

Description of project

This project is structured to create basic database of a digital wallet system. This project has 6 entities with constrain that provide basic security to our database. Constrain like NOT NULL, DEFAULT, UNIQUE etc. helps the data entry in our database strict which accepts certain data types providing basic security functionalities. The database begins with user entity who creates an account in our organization that provides digital services like utility payment, bus/aeroplane ticket reserve, movies booking etc. All the services are accessed via the organization account. When the user wants to use any service, the transaction is processed through the account and the service is booked to the account for the user. The user can have multiple account in our organization and an account can have multiple users as per need. A single account can access multiple services at a time and also a single service is accessible to multiple accounts. The account type (Verified, Non-authenticated) helps us define transaction behaviour of the account.

Goals and objectives of project

- To create database for digital payment system
- To create database that allows multiple users to access single account
- To create database that allows single user use multiple accounts
- Creating database that allows a single service accessed via multiple accounts
- Creating database that allows a single account access multiple service
- To create service of making unlimited transaction for verified account
- To create service of making limited (Rs. 5,000) transaction for non-authenticated accounts
- To explain the functionality of bridge entity in many to many relation
- Providing basic security to databases using constrains

Features present in database

- A single user can create multiple accounts
- A single account (Corporate Account) can belong to multiple users
- A single service can be used by multiple users
- A single account can access multiple services
- Transactions are unlimited for verified account
- Transaction is limited to Rs. 5,000 for non-authenticated accounts
- Transaction charge is 0 for amount up to Rs. 9,999
- Basic security via the use of constrains
- Avoid null values

Database Model

Business rules

- If the transaction is less than Rs. 10,000 than there is no transaction charge. But if it's more than Rs. 10,000 than Rs. 100 is transaction charge.
- Note, transaction charge only applies while loading or withdrawing to or from the account but not on the services the account access.
- Verified accounts can perform unlimited number of transactions with no money limits.
- Non-authenticated accounts cannot perform unlimited number of transactions, they can only make transaction up to Rs. 5,000 per day.
- For transaction of more than Rs. 10,000 to another account, the account needs to be verified.

Entity Relationship Model

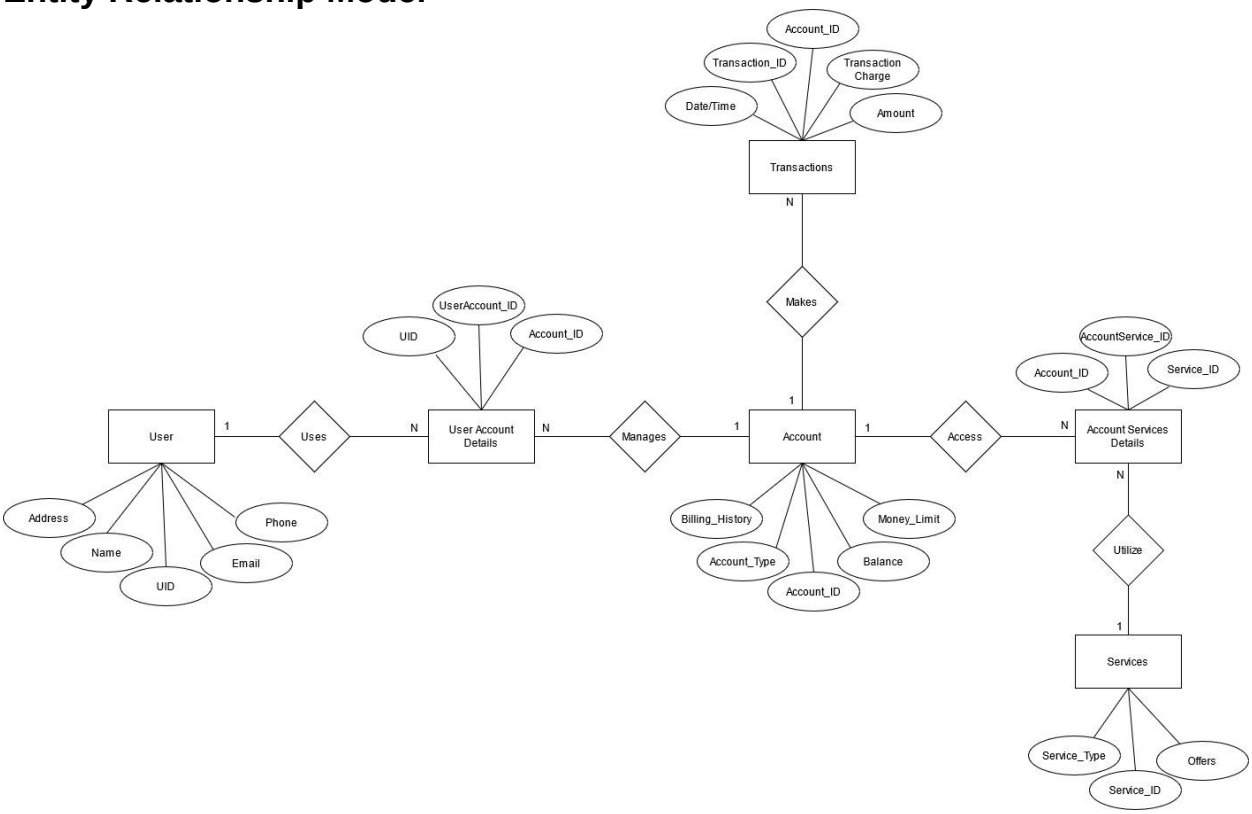

Figure 1: Digital Wallet Entity Relation Model

Relational diagram

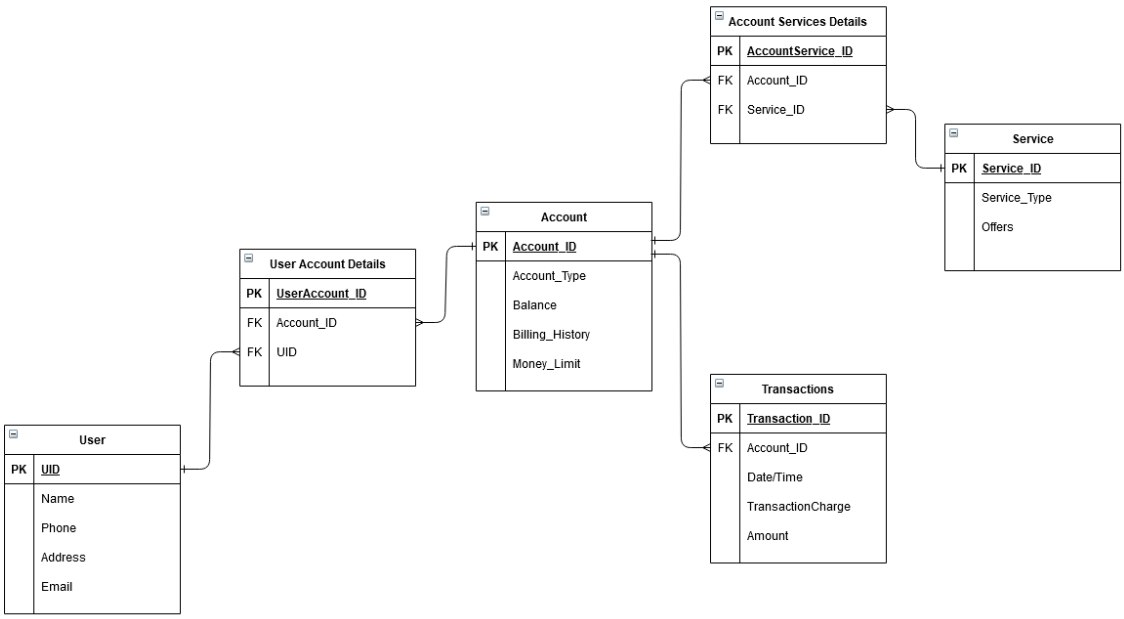

Figure 2: Digital Wallet Relational Diagram

Entities

1. User

The user entity is created as our organization needs the credentials of the beneficiary person or the corporation in order to control or manage transaction and use of different account associated to the particular person. It contains all the information about a user if needed by the account to validate if the user is associated to the account. If yes, the rights that this particular user have on the account and on the services provided. Different attributes present in this table are:

a. UID

As each user needs to be unique in a database, UID is set as primary key for user entity. It holds value of type integer representing more than 4 billion users which is sufficient for our organization. It is helpful as it can represent other attributes (name, phone, address) having duplicate values as different records. AUTO_INCREMENT keyword generates UID for each user.

b. Name

This attribute store full name of the user. It is of VARCHAR datatype which stores string up to 255 character which is sufficient for this attribute. This attribute cannot be null as the Name is associated with the account entity so NOT NULL constrain is applied.

c. Phone

This attribute stores phone number. It is of type VARCHAR which holds string values. Phone number is stored in string as phone number may be 9-10 numbers long which if stored in integer datatype then huge storage is required. String efficiently stores the phone in lot less storage. UNIQUE constrain is used as phone number is unique for each user.

d. Address

This attribute stores the location address of user. It is of datatype VARCHAR holing 255 string values. If user fails to specify their address, then DEFAULT keyword enables the database to set the user's address to 'Kathmandu'.

e. Email

This attribute stores email in datatype VARCHAR. Since, each account needs to be recovered (if password is forgetten) via email so this field should not be null so NOT NULL constrain is set on this attribute.

```
MariaDB [digital_wallet]> CREATE TABLE User (
    -> uid INT PRIMARY KEY AUTO_INCREMENT,
    -> name VARCHAR(255) NOT NULL,
    -> phone VARCHAR(255) UNIQUE,
    -> address VARCHAR(255) DEFAULT "Kathmandu",
    -> email VARCHAR(255) NOT NULL);
Query OK, 0 rows affected (0.018 sec)
```

Figure 3: User Entity Creation

```
MariaDB [digital_wallet]> DESCRIBE user;
+---------+--------------+------+-----+-----------+----------------+
| Field   | Type         | Null | Key | Default   | Extra          |
+---------+--------------+------+-----+-----------+----------------+
| uid     | int(11)      | NO   | PRI | NULL      | auto_increment |
| name    | varchar(255) | NO   |     | NULL      |                |
| phone   | varchar(255) | YES  | UNI | NULL      |                |
| address | varchar(255) | YES  |     | Kathmandu |                |
| email   | varchar(255) | NO   |     | NULL      |                |
+---------+--------------+------+-----+-----------+----------------+
5 rows in set (0.022 sec)
```

Figure 4: Description of User Entity

```
MariaDB [digital_wallet]> INSERT INTO user VALUES
    -> ("", "Ramesh Kunwor", "9854125412", "Maitidevi", "rameshkunwor@gmail.com"),
    -> ("", "Rajesh Kunwor", "9854125566", "New Baneshwor", "rajeshkumarkunwor@gmail.com"),
    -> ("", "Vansh Gupta", "9854196512", "Gaaur", "guptavansh@gmail.com"),
    -> ("", "Parash Niraula", "9856125417", "Basundhara", "niraula212@gmail.com"),
    -> ("", "Jeevan Bhatta", "98541259418", "Pepsicola", "kapadi@gmail.com"),
    -> ("", "Dheeraz Shah", "9854457414", "Tinkune", "sahdzre@gmail.com"),
    -> ("", "Bishal Shah", "9854267412", "Sinamangal", "bishal82@gmail.com"),
    -> ("", "Bishal Dhungana", "9855795415", "New Baneshwor", "dhungana131@gmail.com"),
    -> ("", "Bibek Thapa", "9854125578", "Kamalpokhari", "thapabibek@gmail.com"),
    -> ("", "Gaurav Gandhi", "9854124573", "New Road", "gandhibhai@gmail.com");
Query OK, 10 rows affected, 10 warnings (0.004 sec)
Records: 10  Duplicates: 0  Warnings: 10
```

Figure 5: Value Insertion in User Entity

```
MariaDB [digital_wallet]> SELECT * FROM user;
+-----+----------------+------------+---------------+-----------------------------+
| uid | name           | phone      | address       | email                       |
+-----+----------------+------------+---------------+-----------------------------+
|   1 | Ramesh Kunwor  | 9854125412 | Maitidevi     | rameshkunwor@gmail.com      |
|   2 | Rajesh Kunwor  | 9854125566 | New Baneshwor | rajeshkumarkunwor@gmail.com |
|   3 | Vansh Gupta    | 9854196512 | Gaaur         | guptavansh@gmail.com        |
|   4 | Parash Niraula | 9856125417 | Basundhara    | niraula212@gmail.com        |
|   5 | Jeevan Bhatta  | 98541259418 | Pepsicola     | kapadi@gmail.com            |
|   6 | Dheeraz Shah   | 9854457414 | Tinkune       | sahdzre@gmail.com           |
|   7 | Bishal Shah    | 9854267412 | Sinamangal    | bishal82@gmail.com          |
|   8 | Bishal Dhungana | 9855795415 | New Baneshwor | dhungana131@gmail.com       |
|   9 | Bibek Thapa    | 9854125578 | Kamalpokhari  | thapabibek@gmail.com        |
|  10 | Gaurav Gandhi  | 9854124573 | New Road      | gandhibhai@gmail.com        |
+-----+----------------+------------+---------------+-----------------------------+
10 rows in set (0.000 sec)
```

Figure 6: Select Statement in User Entity

2. Account

Account is the major entity responsible for accessing all the services and transactions. This entity defines how a user or corporation uses each service provided by the organization. The services and users link to the account to associate access of service by the user of the account. Attribute account type defines how much transaction at a particular time does the account permit. Its attributes are:

a. Account ID

Account ID is set as primary key for Account entity as each account in a database must be unique. It is of INT datatype storing more than 4 billion account details. AUTO_INCREMENT keyword is applied to automatically generate ID for each account created.

b. Balance

This attribute stores the current balance. DOUBLE datatype is used which allows to store precise amount. While opening account it this field may be null so to avoid null value, DEFAULT constrain sets the balance to 0.

c. Money Limit

This attribute stores the information about how much amount this account is eligible to transfer at particular time. DOUBLE datatype is used to allow precise amount transaction.

d. Account Type

This attribute stores the account type (Verified, non-authenticated) which allows our organization to provide the service type and money limit this account is able to transact. VARCHAR stores this information in form of string. NOT NULL constrain is used as an account can either be verified or not.

e. Billing History

This attribute stores the transactions made by the account. VARCHAR is used to store all the transaction done via this account. Since, each bill needs to be unique so UNIQUE constrain is set.

```
MariaDB [digital_wallet]> CREATE TABLE Account (
    -> accountID INT PRIMARY KEY AUTO_INCREMENT,
    -> balance DOUBLE DEFAULT 0.0,
    -> money_limit DOUBLE,
    -> account_type VARCHAR(255) NOT NULL,
    -> billing_history VARCHAR(255) UNIQUE);
Query OK, 0 rows affected (0.023 sec)
```

Figure 7: Account Entity Creation

```
MariaDB [digital_wallet]> DESCRIBE account;
+-----------------+--------------+------+-----+---------+----------------+
| Field           | Type         | Null | Key | Default | Extra          |
+-----------------+--------------+------+-----+---------+----------------+
| accountID       | int(11)      | NO   | PRI | NULL    | auto_increment |
| balance         | double       | YES  |     | 0       |                |
| money_limit     | double       | YES  |     | NULL    |                |
| account_type    | varchar(255) | NO   |     | NULL    |                |
| billing_history | varchar(255) | YES  | UNI | NULL    |                |
+-----------------+--------------+------+-----+---------+----------------+
5 rows in set (0.023 sec)
```

Figure 8: Description of Account Entity

```
MariaDB [digital_wallet]> INSERT INTO account VALUES
    -> (1, 80000, 10000000, "Verified", "p001, pk024"),
    -> (2, 160000, 10000000, "Verified", "p051"),
    -> (3, 320000, 10000000, "Verified", "ps141, ps5412, pka44"),
    -> (4, 680000, 10000000, "Verified", "a041, as522"),
    -> (5, 80000, 10000000, "non-authenticated", "d295s, bs596"),
    -> (6, 9000, 10000000, "non-authenticated", NULL);
Query OK, 6 rows affected (0.004 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

Figure 9: Data Insertion in Account Entity

```
MariaDB [digital_wallet]> SELECT * FROM account;
+-----------+---------+-------------+------------------+----------------------+
| accountID | balance | money_limit | account_type     | billing_history      |
+-----------+---------+-------------+------------------+----------------------+
|         1 |   80000 |    10000000 | Verified         | p001, pk024          |
|         2 |  160000 |    10000000 | Verified         | p051                 |
|         3 |  320000 |    10000000 | Verified         | ps141, ps5412, pka44 |
|         4 |  680000 |    10000000 | Verified         | a041, as522          |
|         5 |   80000 |    10000000 | non-authenticated | d295s, bs596         |
|         6 |    9000 |    10000000 | non-authenticated | NULL                 |
+-----------+---------+-------------+------------------+----------------------+
6 rows in set (0.000 sec)
```

Figure 10: Select Statement in Account Entity

Transactions

This entity defines the transaction performed by the account. It contains foreign key Account ID so it's parent table is Account entity. This attribute helps transaction entity to know which account performed certain transaction. When accessing any service provided by the organization, transaction needs to be performed so this table defines how a service can be accessed. Different attribute that defines this entity are:

a. Transaction ID

This attribute is primary key for transactions entity. AUTO_INCREMENT keyword is used so for each transaction is auto-generated of type integer. Integer Datatype is used as it can represent more than 4 billion transaction for each account.

b. Amount

This attribute stores the amount that takes place in particular transaction. It is of type DOUBLE as amount need precision given by floating point. NOT NULL constrain is used as for a transaction to occur some amount is required.

c. Account ID

This attribute stores account ID of the account which is performing the transaction. This attribute is foreign key representing a particular account in account entity. This helps in verifying which account proceeded the transaction.

d. Date Time

This attribute stores date at transaction time. It is type DATE which stores date of the transaction time.

e. Transaction Charge

This attribute stores the transaction charge for particular transaction. It is of type DOUBLE as it is amount and needs precision provided by floating point datatype. If no transaction charge is supplied then instead of the field having null value, the DEFAULT constrain makes transaction charge 0.

```
MariaDB [digital_wallet]> CREATE TABLE Transactions (
    -> transactionID INT PRIMARY KEY AUTO_INCREMENT,
    -> amount DOUBLE NOT NULL,
    -> accountID INT,
    -> date_time DATE,
    -> transaction_charge DOUBLE DEFAULT 0.0,
    -> FOREIGN KEY (accountID) REFERENCES account(accountID));
Query OK, 0 rows affected (0.023 sec)
```

Figure 11: Transaction Entity Creation

```
MariaDB [digital_wallet]> DESCRIBE transactions;
+--------------------+---------+------+-----+---------+----------------+
| Field              | Type    | Null | Key | Default | Extra          |
+--------------------+---------+------+-----+---------+----------------+
| transactionID      | int(11) | NO   | PRI | NULL    | auto_increment |
| amount             | double  | NO   |     | NULL    |                |
| accountID          | int(11) | YES  | MUL | NULL    |                |
| date_time          | date    | YES  |     | NULL    |                |
| transaction_charge | double  | YES  |     | 0       |                |
+--------------------+---------+------+-----+---------+----------------+
5 rows in set (0.025 sec)
```

Figure 12: Description of Transaction Entity

```
MariaDB [digital_wallet]> INSERT INTO transactions (transactionID, amount, accountID, date_time) VALUES
    -> (1, 5000, 4, "2021-4-22"),
    -> (2, 5000, 1, "2021-4-22");
Query OK, 2 rows affected (0.003 sec)
Records: 2  Duplicates: 0  Warnings: 0

MariaDB [digital_wallet]> INSERT INTO transactions (transactionID, amount, accountID, date_time, transaction_charge) VALUES
    ->
    -> (3, 10000, 1, "2021-4-22", 100),
    -> (4, 50000, 2, "2021-4-21", 100),
    -> (5, 10000, 3, "2021-4-21", 100),
    -> (6, 65000, 3, "2021-4-22", 100),
    -> (7, 25000, 3, "2021-4-23", 100),
    -> (8, 15000, 4, "2021-4-21", 100),
    -> (9, 35000, 5, "2021-4-22", 100),
    -> (10, 40000, 5, "2021-4-23", 100);
Query OK, 8 rows affected (0.003 sec)
Records: 8  Duplicates: 0  Warnings: 0
```

Figure 13: Data Insertion (default and all) in Transaction Entity

```
MariaDB [digital_wallet]> SELECT * FROM transactions;
+---------------+--------+-----------+------------+--------------------+
| transactionID | amount | accountID | date_time  | transaction_charge |
+---------------+--------+-----------+------------+--------------------+
|             1 |   5000 |         4 | 2021-04-22 |                  0 |
|             2 |   5000 |         1 | 2021-04-22 |                  0 |
|             3 |  10000 |         1 | 2021-04-22 |                100 |
|             4 |  50000 |         2 | 2021-04-21 |                100 |
|             5 |  10000 |         3 | 2021-04-21 |                100 |
|             6 |  65000 |         3 | 2021-04-22 |                100 |
|             7 |  25000 |         3 | 2021-04-23 |                100 |
|             8 |  15000 |         4 | 2021-04-21 |                100 |
|             9 |  35000 |         5 | 2021-04-22 |                100 |
|            10 |  40000 |         5 | 2021-04-23 |                100 |
+---------------+--------+-----------+------------+--------------------+
10 rows in set (0.000 sec)
```

Figure 14: Select Statement in Transaction Entity

4. Services

This entity is essential as it contains all the services that the account can access. Since, our organization main objective is to sell services to user, this entity plays significant role. All the services belong in this table. If any service is required by the user (account) then this table is searched. The attribute representing the services are:

a. Service ID

This attribute is primary key for service entity. It is of INT datatype which enables us for providing unique value for each service provided. AUTO_INCREMENT keyword is used to auto-generate service id for each service available.

b. Service Type

This attribute stores the type of service (Utility, Movies, Food etc.). It is VARCHAR datatype storing string. NOT NULL constrain is set as each service must be categorised.

c. Offers

This attribute stores the offers available for particular service. VARCHAR datatype is used to store the data in string format. This attribute offers if any discounts and prices (if any available). DEFAULT constrain is set to “No Offers” if no offer is available.

```
MariaDB [digital_wallet]> CREATE TABLE Services (
    -> serviceID INT PRIMARY KEY AUTO_INCREMENT,
    -> service_type VARCHAR(255) NOT NULL,
    -> offers VARCHAR(255) DEFAULT "No Offers");
Query OK, 0 rows affected (0.016 sec)
```

Figure 15: Service Entity Creation

```
MariaDB [digital_wallet]> DESCRIBE services;
+--------------+--------------+------+-----+-----------+----------------+
| Field        | Type         | Null | Key | Default   | Extra          |
+--------------+--------------+------+-----+-----------+----------------+
| serviceID    | int(11)      | NO   | PRI | NULL      | auto_increment |
| service_type | varchar(255) | NO   |     | NULL      |                |
| offers       | varchar(255) | YES  |     | No Offers |                |
+--------------+--------------+------+-----+-----------+----------------+
3 rows in set (0.022 sec)
```

Figure 16: Description of Services Entity

```
MariaDB [digital_wallet]> INSERT INTO services (serviceID, service_type, offers) VALUES
    -> ("", "Top Up", "5% cashback"),
    -> ("", "Antivirus", "10% cashback"),
    -> ("", "Movies", "2% cashback"),
    -> ("", "Food and Hospitality", "15% cashback");
Query OK, 4 rows affected, 4 warnings (0.004 sec)
Records: 4  Duplicates: 0  Warnings: 4

MariaDB [digital_wallet]> INSERT INTO services (serviceID, service_type) VALUES
    -> ("", "Electricity Bill"),
    -> ("", "Internet Bill"),
    -> ("", "Khanepani Bill"),
    -> ("", "Online Shopping"),
    -> ("", "Scan and Pay"),
    -> ("", "Stock Payment"),
    -> ("", "Insurance Payment");
Query OK, 7 rows affected, 7 warnings (0.003 sec)
Records: 7  Duplicates: 0  Warnings: 7
```

Figure 17: Data Insertion (default and all) in Services Entity

```
MariaDB [digital_wallet]> SELECT * FROM services;
+-----------+---------------------+--------------+
| serviceID | service_type        | offers       |
+-----------+---------------------+--------------+
|         1 | Top Up              | 5% cashback  |
|         2 | Antivirus           | 10% cashback |
|         3 | Movies              | 2% cashback  |
|         4 | Food and Hospitality | 15% cashback |
|         5 | Electricity Bill    | No Offers    |
|         6 | Internet Bill       | No Offers    |
|         7 | Khanepani Bill      | No Offers    |
|         8 | Online Shopping     | No Offers    |
|         9 | Scan and Pay        | No Offers    |
|        10 | Stock Payment       | No Offers    |
|        11 | Insurance Payment   | No Offers    |
+-----------+---------------------+--------------+
11 rows in set (0.000 sec)
```

Figure 18: Select Statement in Services Entity

5. User Account Details

There exist many (N) to many (N) relation between User and Account entity. In a database N to N relation must be avoided. When there is N to N relation between two entity then each entity (table) assumes the other is parent table, this cause error in RDMBS so to avoid this; User Account Details bridge entity is introduced. Creating user account details entity enables our organization to associate multiple users to a single account, or associate single user to multiple account as per need by the organization. The attributes that avoid N to N relation are:

a. User Account ID

Primary key for User Account Details entity is User Account ID. INT datatype is used as it can represent more than billion id uniquely. AUTO_INCREMENT keyword is used in order to auto-generate user account id.

b. UID

This attribute is foreign key taking reference from uid of user entity. This attribute has same datatype as its original in user entity. It enables our database to use relation in order to associate a user with multiple account.

c. Account ID

This attribute is foreign key taking reference from account id of account entity. This attribute has same datatype as its original. It enables our database to use relation in order to associate an account with multiple users.

```
MariaDB [digital_wallet]> CREATE TABLE UserAccountDetails (
    -> useraccountID INT PRIMARY KEY AUTO_INCREMENT,
    -> uid int,
    -> accountID int,
    -> FOREIGN KEY (uid) REFERENCES user(uid),
    -> FOREIGN KEY (accountID) REFERENCES account(accountID));
Query OK, 0 rows affected (0.030 sec)
```

Figure 19: User Account Details Entity Creation

```
MariaDB [digital_wallet]> DESCRIBE useraccountdetails;
+---------------+---------+------+-----+---------+----------------+
| Field         | Type    | Null | Key | Default | Extra          |
+---------------+---------+------+-----+---------+----------------+
| useraccountID | int(11) | NO   | PRI | NULL    | auto_increment |
| uid           | int(11) | YES  | MUL | NULL    |                |
| accountID     | int(11) | YES  | MUL | NULL    |                |
+---------------+---------+------+-----+---------+----------------+
3 rows in set (0.025 sec)
```

Figure 20: Description of User Account Details Entity

```
MariaDB [digital_wallet]> INSERT INTO useraccountdetails VALUES
    -> (1, 1, 4),
    -> (2, 2, 4),
    -> (3, 3, 6),
    -> (4, 4, 3),
    -> (5, 5, 3),
    -> (6, 6, 1),
    -> (7, 7, 2),
    -> (8, 8, 2),
    -> (9, 9, 5),
    -> (10, 10, 5);
Query OK, 10 rows affected (0.004 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

Figure 21: Data Insertion in User Account Details Entity

```
MariaDB [digital_wallet]> SELECT * FROM useraccountdetails;
+---------------+------+-----------+
| useraccountID | uid  | accountID |
+---------------+------+-----------+
|             1 |    1 |         4 |
|             2 |    2 |         4 |
|             3 |    3 |         6 |
|             4 |    4 |         3 |
|             5 |    5 |         3 |
|             6 |    6 |         1 |
|             7 |    7 |         2 |
|             8 |    8 |         2 |
|             9 |    9 |         5 |
|            10 |   10 |         5 |
+---------------+------+-----------+
10 rows in set (0.000 sec)
```

Figure 22: Select Statement in User Account Details Entity

6. Account Services Details

There exist many (N) to many (N) relation between Account and Services entity. In a database N to N relation must be avoided. When there is N to N relation between two entity then each entity (table) assumes the other is parent table, this cause error in RDMBS so to avoid this; Account Services Details bridge entity is introduced. Creating account services details entity enables our organization to associate multiple services to a single account, or associate single service to multiple accounts as per need by the user or organization. The attributes that avoid N to N relation are:

a. Account Service ID

Account Service ID is primary key for Account Service Details entity. It uses INT datatype to represent each id uniquely. AUTO_INCREMENT keyword is used to auto-generate id

b. Service ID

This attribute is foreign key taking reference from service id of services entity. This attribute has same datatype as its original. It enables our database to use relation in order to associate a service with multiple accounts.

c. Account ID

This attribute is foreign key taking reference from account id of account entity. This attribute has same datatype as its original. It enables our database to use relation in order to associate an account with multiple services.

```
MariaDB [digital_wallet]> CREATE TABLE AccountServicesDetails (
    -> accountserviceID INT PRIMARY KEY AUTO_INCREMENT,
    -> serviceID int,
    -> accountID int,
    -> FOREIGN KEY (serviceID) REFERENCES services(serviceID),
    -> FOREIGN KEY (accountID) REFERENCES account(accountID));
Query OK, 0 rows affected (0.019 sec)
```

Figure 23: Account Services Details Entity Creation

```
MariaDB [digital_wallet]> DESCRIBE accountservicesdetails;
+-----------------+---------+------+-----+---------+----------------+
| Field           | Type    | Null | Key | Default | Extra          |
+-----------------+---------+------+-----+---------+----------------+
| accountserviceID | int(11) | NO   | PRI | NULL    | auto_increment |
| serviceID       | int(11) | YES  | MUL | NULL    |                |
| accountID       | int(11) | YES  | MUL | NULL    |                |
+-----------------+---------+------+-----+---------+----------------+
3 rows in set (0.024 sec)
```

Figure 24: Description of Account Services Details Entity

```
MariaDB [digital_wallet]> INSERT INTO accountservicesdetails VALUES
    -> (1, 11, 1),
    -> (2, 10, 2),
    -> (3, 6, 3),
    -> (4, 9, 4),
    -> (5, 4, 5);
Query OK, 5 rows affected (0.004 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

Figure 25: Data Insertion in Account Service Details Entity

```
MariaDB [digital_wallet]> SELECT * FROM accountservicesdetails;
+-----------------+-----------+-----------+
| accountserviceID | serviceID | accountID |
+-----------------+-----------+-----------+
|               1 |        11 |         1 |
|               2 |        10 |         2 |
|               3 |         6 |         3 |
|               4 |         9 |         4 |
|               5 |         4 |         5 |
+-----------------+-----------+-----------+
5 rows in set (0.001 sec)
```

Figure 26: Select Statement in Account Services Details Entity

Data Dictionary

Data Dictionary is metadata of the data. This means that the information of the datatype, constrains used in the database, if the database accept null values lies within data dictionary. Data dictionary provides the blueprint for the data that is being or going to be stored in the database.

Some Notation used in the data dictionary:

- PK – Primary Key
- FK – Foreign Key

Entity name	Entity description	Column name	Column description	Data type	Length	Primary key	Foreign key	Nullable	Unique	Notes
User	User is someone who uses the organization's account to access the services provided by the same organization	UID	For uniquely identifying individual users uid is used	INT	11	TRUE	FALSE	NO	TRUE	PK, AUTO-INCREMENT
		Name	Name of the user	VARCHAR	255	FALSE	FALSE	NO	FALSE	
		Phone	Phone number of the user	VARCHAR	255	FALSE	FALSE	YES	FALSE	
		Address	Address where the user lives	VARCHAR	255	FALSE	FALSE	YES	FALSE	DEFAULT: "Kathmandu"
		Email	Email address of the user	VARCHAR	255	FALSE	FALSE	NO	TRUE	

Table 1: User Entity Data Dictionary

Entity name	Entity description	Column name	Column description	Data type	Length	Primary key	Foreign key	Nullable	Unique	Notes
Account	This is account provided by the organization which is responsible for all the transaction and access to services	Account ID	For uniquely identifying each account in database account id is provided	INT	11	TRUE	FALSE	NO	TRUE	PK, AUTO-INCREMENT
		Balance	Total amount present in the account	DOUBLE		FALSE	FALSE	YES	FALSE	DEFAULT: 0.0
		Money limit	Total amount the account can hold	DOUBLE		FALSE	FALSE	YES	FALSE	
		Account type	Limits the number of the transaction	VARCHAR	255	FALSE	FALSE	NO	FALSE	
		Billing history	Stores all the transaction history	VARCHAR	255	FALSE	FALSE	YES	TRUE	

Table 2: Account Entity Data Dictionary

Entity name	Entity description	Column name	Column description	Data type	Length	Primary key	Foreign key	Nullable	Unique	Notes
Transactions	All the transaction is done is stored in transaction entity with the information of which account proceeded the transaction	Transaction ID	For uniquely identifying each transaction in database transaction id is provided	INT	11	TRUE	FALSE	NO	TRUE	PK, AUTO-INCREMENT
		Amount	Total amount that is processed in the transaction	DOUBLE		FALSE	FALSE	NO	FALSE	
		Account ID	The account who proceeded the transaction	INT	11	FALSE	TRUE	YES	FALSE	Reference to accountID column of account table
		Date Time	Date of the transaction	DATE		FALSE	FALSE	YES	FALSE	
		Transaction Charge	Transaction charge on the amount	DOUBLE		FALSE	FALSE	YES	FALSE	DEFAULT: 0.0

Table 3: Transaction Entity Data Dictionary

Entity name	Entity description	Column name	Column description	Data type	Length	Primary key	Foreign key	Nullable	Unique	Notes
Services	All the services that an account access lies in services entity	Service ID	For uniquely identifying each service in database service id is provided	INT	11	TRUE	FALSE	FALSE	TRUE	PK, AUTO-INCREMENT
		Service Type	Services are categorised into types. So, each service belongs to a type (e.g., utility, top-up)	VARCHAR	255	FALSE	FALSE	FALSE	FALSE	
		Offers	Offers like discount or cashback are represented	VARCHAR	255	FALSE	FALSE	TRUE	FALSE	DEFAULT: "No Offers"

Table 4: Services Entity Data Dictionary

Entity name	Entity description	Column name	Column description	Data type	Length	Primary key	Foreign key	Nullable	Unique	Notes
User Account Details	This works as bridge entity connecting single user to multiple accounts or a single account to multiple users	User Account ID	For uniquely identifying each user account details in database user account id is provided	INT	11	TRUE	FALSE	FALSE	TRUE	PK, AUTO-INCREMENT
		UID	The user id of the user who is associated with the account or trying to get access to the account	INT	11	FALSE	TRUE	TRUE	FALSE	Reference to UID column of user table
		Account ID	The account id of the account who is associated with the user	INT	11	FALSE	TRUE	TRUE	FALSE	Reference to accountID column of account table

Table 5: User Account Details Entity Data Dictionary

Entity name	Entity description	Column name	Column description	Data type	Length	Primary key	Foreign key	Nullable	Unique	Notes
Account Services Details	This works as bridge entity connecting single account to multiple services or a single service to multiple accounts	Account Services ID	For uniquely identifying each account services details in database account services id is provided	INT	11	TRUE	FALSE	FALSE	TRUE	PK, AUTO-INCREMENT
		Service ID	Service Id accesses the services in services entity	INT	11	FALSE	TRUE	TRUE	FALSE	Reference to serviceID column of services table
		Account ID	Account id accesses the account in account entity	INT	11	FALSE	TRUE	TRUE	FALSE	Reference to accountID column of account table

Table 6: Account Services Details Entity Data Dictionary

Queries

Query Number	Query 1
Query	SELECT * FROM user WHERE uid BETWEEN 4 AND 8;
Keywords Used	SELECT, FROM, WHERE, BETWEEN, AND
Purpose/Result	Selects and displays all the records of user table having uid from 4 to 8

Table 7: Between Query

```
MariaDB [digital_wallet]> SELECT * FROM user WHERE uid BETWEEN 4 AND 8;
+-----+-----------------+-------------+---------------+-----------------------+
| uid | name            | phone       | address       | email                 |
+-----+-----------------+-------------+---------------+-----------------------+
|   4 | Parash Niraula  | 9856125417  | Basundhara    | niraula212@gmail.com  |
|   5 | Jeevan Bhatta   | 98541259418 | Pepsicola     | kapadi@gmail.com      |
|   6 | Dheeraz Shah    | 9854457414  | Tinkune       | sahdzre@gmail.com     |
|   7 | Bishal Shah     | 9854267412  | Sinamangal    | bishal82@gmail.com    |
|   8 | Bishal Dhungana | 9855795415  | New Baneshwor | dhungana131@gmail.com |
+-----+-----------------+-------------+---------------+-----------------------+
5 rows in set (0.000 sec)
```

Figure 27: Between Query In SQL

Query Number	Query 2
Query	SELECT * FROM user ORDER BY name;
Keywords Used	SELECT, FROM, ORDER BY
Purpose/Result	Selects and displays all the records of user table in ascending order by name

Table 8: Order By Query

```
MariaDB [digital_wallet]> SELECT * FROM user ORDER BY name;
+-----+-----------------+-------------+---------------+------------------------------+
| uid | name            | phone       | address       | email                        |
+-----+-----------------+-------------+---------------+------------------------------+
|   9 | Bibek Thapa     | 9854125578  | Kamalpokhari  | thapabibek@gmail.com         |
|   8 | Bishal Dhungana | 9855795415  | New Baneshwor | dhungana131@gmail.com        |
|   7 | Bishal Shah     | 9854267412  | Sinamangal    | bishal82@gmail.com           |
|   6 | Dheeraz Shah    | 9854457414  | Tinkune       | sahdzre@gmail.com            |
|  10 | Gaurav Gandhi   | 9854124573  | New Road      | gandhibhai@gmail.com         |
|   5 | Jeevan Bhatta   | 98541259418 | Pepsicola     | kapadi@gmail.com             |
|   4 | Parash Niraula  | 9856125417  | Basundhara    | niraula212@gmail.com         |
|   2 | Rajesh Kunwor   | 9854125566  | New Baneshwor | rajeshkumarkunwor@gmail.com  |
|   1 | Ramesh Kunwor   | 9854125412  | Maitidevi     | rameshkunwor@gmail.com       |
|   3 | Vansh Gupta     | 9854196512  | Gaaur         | guptavansh@gmail.com         |
+-----+-----------------+-------------+---------------+------------------------------+
10 rows in set (0.004 sec)
```

Figure 28: Order By Query In SQL

Query Number	Query 3
Query	SELECT * FROM services WHERE offers IN ("5% cashback", "2% cashback", "10% cashback");
Keywords Used	SELECT, FROM, WHERE, IN
Purpose/Result	Selects and displays all the records of services having value 5% cashback, 10% cashback and 2% cashback

Table 9: IN Query

```
MariaDB [digital_wallet]> SELECT * FROM services WHERE offers IN ("5% cashback", "2% cashback", "10% cashback");
+-----------+--------------+-------------+
| serviceID | service_type | offers      |
+-----------+--------------+-------------+
|         1 | Top Up       | 5% cashback |
|         2 | Antivirus    | 10% cashback |
|         3 | Movies       | 2% cashback |
+-----------+--------------+-------------+
3 rows in set (0.003 sec)
```

Figure 29: IN Query In SQL

Query Number	Query 4
Query	SELECT * FROM user LIMIT 4, 3;
Keywords Used	SELECT, FROM, LIMIT
Purpose/Result	Selects and displays all the records of user table skipping first 4 records and displaying 3 records afterwards represented by (4, 3)

Table 10: Limit Query

```
MariaDB [digital_wallet]> SELECT * FROM user LIMIT 4, 3;
+-----+---------------+-------------+------------+--------------------+
| uid | name          | phone       | address    | email              |
+-----+---------------+-------------+------------+--------------------+
|   5 | Jeevan Bhatta | 98541259418 | Pepsicola  | kapadi@gmail.com   |
|   6 | Dheeraz Shah  | 9854457414  | Tinkune    | sahdzre@gmail.com  |
|   7 | Bishal Shah   | 9854267412  | Sinamangal | bishal82@gmail.com |
+-----+---------------+-------------+------------+--------------------+
3 rows in set (0.000 sec)
```

Figure 30: Limit Query In SQL

Query Number	Query 5
Query	SELECT * FROM services WHERE service_type LIKE "I%";
Keywords Used	SELECT, FROM, WHERE, LIKE
Purpose/Result	Selects and displays all the records of services table starting with “I” letter in service_type column

Table 11: Like Query

```
MariaDB [digital_wallet]> SELECT * FROM services WHERE service_type LIKE "I%";
+-----------+------------------+-----------+
| serviceID | service_type     | offers    |
+-----------+------------------+-----------+
|         6 | Internet Bill    | No Offers |
|        11 | Insurance Payment | No Offers |
+-----------+------------------+-----------+
2 rows in set (0.001 sec)
```

Figure 31: Like Query In SQL

Query Number	Query 6
Query	SELECT DISTINCT account_type AS type FROM account;
Keywords Used	SELECT, DISTINCT, FROM
Purpose/Result	Selects and displays all the records of account table with non-repeated value of column account_type

Table 12: Distinct Query

```
MariaDB [digital_wallet]> SELECT DISTINCT account_type AS type FROM account;
+--------------------+
| type               |
+--------------------+
| Verified           |
| non-authenticated  |
+--------------------+
2 rows in set (0.003 sec)
```

Figure 32: Distinct Query In SQL

Query Number	Query 7
Query	SELECT COUNT(uid) FROM user;
Keywords Used	SELECT, COUNT, FROM
Purpose/Result	Selects and returns total records present in user table

Table 13: Count Query

```
MariaDB [digital_wallet]> SELECT COUNT(uid) FROM user;
+-------------+
| COUNT(uid) |
+-------------+
|          10 |
+-------------+
1 row in set (0.003 sec)
```

Figure 33: Count Query In SQL

Query Number	Query 8
Query	SELECT account_type, COUNT(*) AS total FROM account GROUP BY account_type;
Keywords Used	SELECT, COUNT, AS, FROM, GROUP BY
Purpose/Result	Counts and returns total account type of each account

Table 14: Group By Query

```
MariaDB [digital_wallet]> SELECT account_type, COUNT(*) AS total FROM account GROUP BY account_type;
+-------------------+-------+
| account_type      | total |
+-------------------+-------+
| non-authenticated |     2 |
| Verified          |     4 |
+-------------------+-------+
2 rows in set (0.001 sec)
```

Figure 34: Group By Query In SQL

Query Number	Query 9
Query	SELECT uid, count(*) AS "Total Associated Accounts" FROM useraccountdetails GROUP BY accountID HAVING COUNT(*) > 1 ORDER BY uid;
Keywords Used	SELECT, COUNT, AS, FROM, GROUP BY, HAVING, ORDER BY
Purpose/Result	Returns total count of all uid associated with more than one account

Table 15: Having Query

```
MariaDB [digital_wallet]> SELECT uid, count(*) AS "Total Associated Accounts" FROM useraccountdetails GROUP BY accountID HAVING COUNT(*) > 1 ORDER BY uid;
+------+---------------------------+
| uid  | Total Associated Accounts |
+------+---------------------------+
|    1 |                         2 |
|    4 |                         2 |
|    7 |                         2 |
|    9 |                         2 |
+------+---------------------------+
4 rows in set (0.001 sec)
```

Figure 35: Having Query In SQL

Query Number	Query 10
Query	SELECT uid, serviced FROM useraccountdetails JOIN accountservicesdetails ON useraccountdetails.accountID = accountservicesdetails.accountID ORDER BY uid;
Keywords Used	SELECT, FROM, JOIN, ON, ORDER BY
Purpose/Result	Returns uid, servicedID from useraccountdetails and accountservicesdetails whose accountID is same on both entity

Table 16: Join Query

```
MariaDB [digital_wallet]> SELECT uid, serviceID FROM useraccountdetails
    -> JOIN accountservicesdetails ON useraccountdetails.accountID = accountservicesdetails.accountID
    -> ORDER BY uid;
+------+-----------+
| uid  | serviceID |
+------+-----------+
|    1 |         9 |
|    2 |         9 |
|    4 |         6 |
|    5 |         6 |
|    6 |        11 |
|    7 |        10 |
|    8 |        10 |
|    9 |         4 |
|   10 |         4 |
+------+-----------+
9 rows in set (0.001 sec)
```

Figure 36: Join Query In SQL

Conclusion

This project was an in-depth guide to use database. Thought the project questions like why we needed bridge entity was solved. The answer was simple. It is because when we create two table with many to many relations (in our case User and Account table) then we need to assign primary key of each table to another as foreign key. While assigning the foreign key to either table one becomes the child table of another and both tables search for its parent table, but both the parent table is not created so the child table shows an error that it cannot find parent table and how it is unable the child table. While introducing bridge entity (User Account Details), we create one to many relations so we can create parent table (User and Account). Afterwards, we create bridge entity (User Account Details) and assign the primary key of parent table as foreign key in bridge entity. This solves the search for parent table (among User and Account) when bridge entity is not introduced. (Tung, et al., 2005)

While creating the database, we learned that we can provide basic security to our database by using constrains. MySQL has different constrains that together provides basic security. Constrains like NOT NULL does not allow empty record, DEFAULT provides a default value if the value is not assigned, UNIQUE will only accept unique values in the database.

References

Morley, D. & Parker, C. S., 2015. *Understanding Computers: Today and Tomorrow.* 15 ed. s.l.:Cengage Learning.

Tung, L. H., Kung, H. J. & Gardiner, A., 2005. Bridging entity-relationship and relational data models. *Proceedings of the 2005 International Conference on Informatics Education Research,* 1(1), pp. 303-311.