

Task 1: Relational Database Normalization and Implementation using MySQL

1. Data Normalization Process

1.1 Unnormalized Form (UNF)

The given dataset consists of repetitive and redundant data, stored in a single table without clear separation of entities. The data structure is as follows:

Name	Email	DOB	Street	City	Country	Zip Code	Favorite Book	Favorite Drink	Favorite Activity	Neighbour 1	Neighbour1Email	Neighbour 2	Neighbour2Email
Person 1	person1@email.co.uk	3/15/1995	12 Maple St	London	England	E1 6AN	A New Beginning	Lemonade	Outdoor Running	Neighbor A	neighborA@email.com	Neighbor B	neighborB@email.com
Person 2	person2@email.co.uk	6/22/1993	45 Oak Ave	Manchester	England	M1 2WD	The Road to Success	Coffee	Hiking	Neighbor C	neighborC@email.com	Neighbor D	neighborD@email.com
Person 3	person3@email.co.uk	9/10/1991	89 Pine Rd	Birmingham	England	B1 1AB	Endless Possibilities	Smoothie	Swimming	Neighbor E	neighborE@email.com	Neighbor F	neighborF@email.com
Person 4	person4@email.co.uk	12/5/1998	23 Birch St	Edinburgh	Scotland	EH1 1YZ	Journey of Life	Iced Tea	Traveling	Neighbor G	neighborG@email.com	Neighbor H	neighborH@email.com
Person 5	person5@email.co.uk	11/30/1983	67 Cedar Ln	Bristol	England	BS1 3XE	The Adventure	Green Tea	Gardening	Neighbor I	neighborI@email.com	Neighbor J	neighborJ@email.com
Person 6	person6@email.co.uk	7/18/1989	56 Elm St	Liverpool	England	L1 1AA	Finding Inner Peace	Coconut Water	Reading	Neighbor K	neighborK@email.com	Neighbor L	neighborL@email.com
Person 7	person7@email.co.uk	4/25/1996	12 Maple St	Glasgow	Scotland	G1 2TF	Exploring New Horizons	Fruit Juice	Cycling	Neighbor M	neighborM@email.com	Neighbor N	neighborN@email.com
Person 8	person8@email.co.uk	1/9/1990	89 Oak Dr	Leeds	England	LS1 3AB	The Great Journey	Water	Hiking	Neighbor O	neighborO@email.com	Neighbor P	neighborP@email.com
Person 9	person9@email.co.uk	8/17/1993	123 Pine Rd	Newcastle	England	NE1 2AB	The Power of Clean	Hot Chocolate	Skiing	Neighbor Q	neighborQ@email.com	Neighbor R	neighborR@email.com
Person 10	person10@email.co.uk	10/22/1997	15 Elm St	Cardiff	Wales	CF10 3AF	New Beginning	Fruit Smoothie	Jogging	Neighbor S	neighborS@email.com	Neighbor T	neighborT@email.com
Person 11	person11@email.co.uk	5/13/1992	78 Oak Ln	Sheffield	England	S1 4GT	Wandering Souls	Sparkling Water	Rock Climbing	Neighbor U	neighborU@email.com	Neighbor V	neighborV@email.com
Person 12	person12@email.co.uk	2/27/1986	56 Birch Rd	Nottingham	England	NG1 2PB	Freedom and Choice	Herbal Tea	Yoga	Neighbor W	neighborW@email.com	Neighbor X	neighborX@email.com
Person 13	person13@email.co.uk	11/25/1991	10 Holy St	Cardiff	Wales	CF10 2NF	New Beginning	Smoothie	Hiking	Neighbor Y	neighborY@email.com	Neighbor Z	neighborZ@email.com
Person 14	person14@email.co.uk	2/1/1987	34 Willow Rd	Edinburgh	Scotland	EH1 1AB	Chasing Dreams	Lemonade	Running	Neighbor AA	neighborAA@email.com	Neighbor AB	neighborAB@email.com
Person 15	person15@email.co.uk	8/12/1984	78 Cedar Ave	Cambridge	England	CB1 2SE	The Endless Journey	Iced Coffee	Cycling	Neighbor AC	neighborAC@email.com	Neighbor AD	neighborAD@email.com
Person 16	person16@email.co.uk	3/9/1990	45 Maple Rd	Oxford	England	OX2 6TP	The Future Awaits	Fruit Juice	Yoga	Neighbor AE	neighborAE@email.com	Neighbor AF	neighborAF@email.com
Person 17	person17@email.co.uk	11/17/1995	23 Birch Ave	Southampton	England	SO14 3HL	The Path to Global	Green Tea	Gardening	Neighbor AG	neighborAG@email.com	Neighbor AH	neighborAH@email.com
Person 18	person18@email.co.uk	6/20/1994	12 Elm Blvd	Leicester	England	LE1 3PL	Life's Adventure	Coconut Water	Hiking	Neighbor AI	neighborAI@email.com	Neighbor AJ	neighborAJ@email.com
Person 19	person19@email.co.uk	12/11/1992	56 Oak Rd	Norwich	England	NR1 4BE	Into the Wild	Herbal Tea	Swimming	Neighbor AK	neighborAK@email.com	Neighbor AL	neighborAL@email.com
Person 20	person20@email.co.uk	9/25/1988	89 Pine Ave	Cardiff	Wales	CF10 3BC	The Adventure	Water	Hiking	Neighbor AM	neighborAM@email.com	Neighbor AN	neighborAN@email.com

Issues with UNF:

1. Multivalued attributes (Favorite Book, Favorite Activity, Neighbors)
2. Repeating groups (Neighbor 1 & 2, Neighbor1Email & Neighbor2Email)
3. Lack of primary key

1.2 First Normal Form (1NF)

To achieve **1NF**, we eliminate multi-valued attributes:

A table is in **1NF** if:

1. All columns contain atomic values (no multi-valued attributes).
2. Each row has a unique identifier (Primary Key).

Converting to 1NF:

- Break multi-valued attributes into separate rows
- Create separate tables for Address, Favorites, and Neighbor

Updated Tables in 1NF:

Person Table

PersonID	Name	Email	DOB
1	Person 1	person1@email.com	3/15/1995
2	Person 2	person2@email.com	6/22/1993

Address Table

AddressID	PersonID	Street	City	Country	Zip Code
1	1	12 Maple St	London	England	E1 6AN
2	2	45 Oak Ave	Manchester	England	M1 2WD

Favorites Table

FavouriteID	PersonID	Favorite Book	Favorite Drink	Favorite Activity
1	1	A New Beginning	Lemonade	Outdoor Running
2	2	The Road to Success	Coffee	Hiking

Neighbor Table

NeighborID	PersonID	NeighborName	NeighborEmail
1	1	Neighbor A	neighborA@email.com
2	1	Neighbor B	neighborB@email.com
3	2	Neighbor C	neighborC@email.com
4	2	Neighbor D	neighborD@email.com

1.3 Second Normal Form (2NF)

To achieve 2NF, we must follow these rules:

A table is in 2NF if:

1. It is in 1NF.
2. No partial dependencies (Every non-key attribute depends on the whole primary key).

Converting to 2NF:

- **Favorite Book, Favorite Drink, and Favorite Activity** depend only on **PersonID** → No change needed
 - **NeighborName** and **NeighborEmail** depend only on **PersonID** → No change needed
 - **Street, City, Country, and Zip Code** depend only on **PersonID** → No change needed
-

1.4 Third Normal Form (3NF)

To achieve 3NF, we remove transitive dependencies. The final table is created:

A table is in 3NF if:

1. It is in 2NF.
2. No transitive dependencies (non-key attributes should not depend on another non-key attribute).

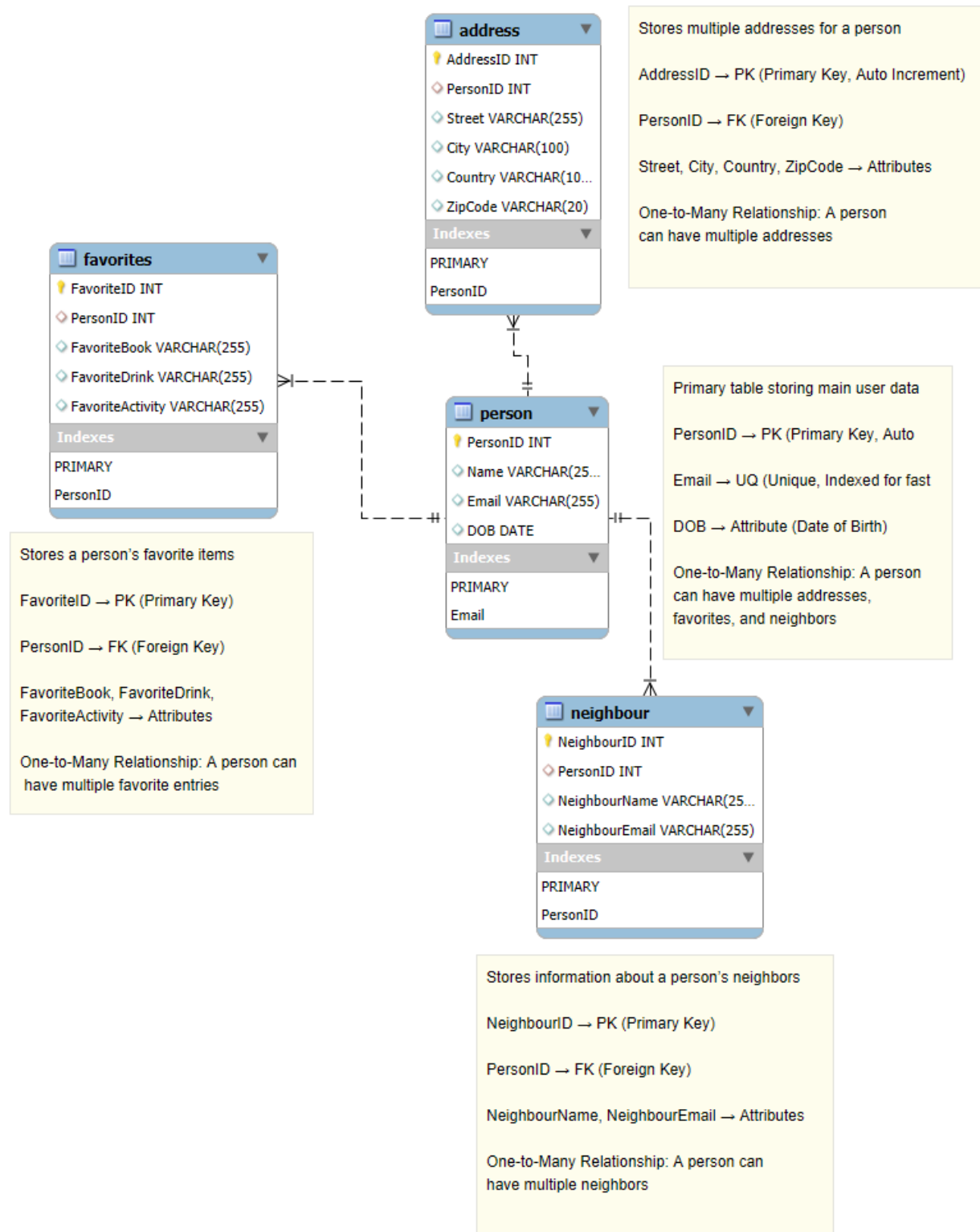
Converting to 3NF:

The current table structure has no transitive dependencies since all attributes depend directly on the primary key.

Final Normalized Schema:

1. Person (PersonID, Name, Email, DOB)
 2. Address (AddressID, PersonID, Street, City, Country, Zip Code)
 3. Favorites (FavouriteID, PersonID, FavoriteBook, FavoriteDrink, FavoriteActivity)
 4. Neighbor (NeighborID, PersonID, NeighborName, NeighborEmail)
-

2. ER Diagram



3. SQL Database Implementation

3.1 Creating the Database and Tables

```
CREATE DATABASE PeopleDB;
```

```
USE PeopleDB;
```

```
CREATE TABLE person (  
    PersonID INT PRIMARY KEY,  
    Name VARCHAR(255) NOT NULL,  
    Email VARCHAR(255) UNIQUE NOT NULL,  
    DOB DATE NOT NULL  
);
```

```
CREATE TABLE address (  
    AddressID INT PRIMARY KEY,  
    PersonID INT,  
    Street VARCHAR(255) NOT NULL,  
    City VARCHAR(100) NOT NULL,  
    Country VARCHAR(100) NOT NULL,  
    ZipCode VARCHAR(20) NOT NULL,  
    FOREIGN KEY (PersonID) REFERENCES person(PersonID) ON DELETE CASCADE  
);
```

```
CREATE TABLE favorites (  
    FavoriteID INT PRIMARY KEY,  
    PersonID INT,  
    FavoriteBook VARCHAR(255),  
    FavoriteDrink VARCHAR(255),  
    FavoriteActivity VARCHAR(255),  
    FOREIGN KEY (PersonID) REFERENCES person(PersonID) ON DELETE CASCADE  
);
```

```
CREATE TABLE neighbour (  
    NeighbourID INT PRIMARY KEY,  
    PersonID INT,  
    NeighbourName VARCHAR(255),  
    NeighbourEmail VARCHAR(255),  
    FOREIGN KEY (PersonID) REFERENCES person(PersonID) ON DELETE CASCADE  
);
```

3.2 Populating the Database

Data is populated & imported as a CSV format, so at first, we have changed file type from “xlsx” to “csv”, and then we used MySQL scripts to import the data to each table in 3NF format.

This is a sample for (person, neighbor) tables import, you could check other codes from GitLab.

```
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/Assessment2425DataCaseScenario.csv'
INTO TABLE Person
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\r\n'
IGNORE 1 ROWS
(@Name, Email, @DOB, @dummy1, @dummy2, @dummy3, @dummy4,
@dummy5, @dummy6, @dummy7, @dummy8, @dummy9, @dummy10, @dummy11)
SET
    PersonID = CAST(SUBSTRING_INDEX(@Name, ' ', -1) AS UNSIGNED), -- Extracts numeric ID
    DOB = STR_TO_DATE(@DOB, '%m/%d/%Y');
```

```
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/Assessment2425DataCaseScenario.csv'
INTO TABLE Neighbour
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\r\n'
IGNORE 1 ROWS
(@dummy1, @dummy2, @dummy3, @dummy4, @dummy5, @dummy6, @dummy7, @dummy8, @dummy9, @dummy10,
@Neighbour1, @Neighbour1Email, @Neighbour2, @Neighbour2Email)
SET
    PersonID = (SELECT PersonID FROM Person WHERE Email = @dummy2),
    NeighbourName = @Neighbour1,
    NeighbourEmail = @Neighbour1Email;
```

4. SQL Queries

4.1 Display Person's Name and Their Age in Years

```
USE PeopleDB;

SELECT Name, TIMESTAMPDIFF(YEAR, DOB, CURDATE()) AS Age FROM Person;
```

Name	Age
Person 1	30
Person 2	31
Person 3	33
Person 4	26
Person 5	41
Person 6	35
Person 7	28
Person 8	35
Person 9	31
Person 10	27
Person 11	32
Person 12	39
Person 13	33
Person 14	38
Person 15	40
Person 16	35
Person 17	29
Person 18	30

4.2 Group Persons by Favourite Drink and Return Average Age

```
USE PeopleDB;
```

```
SELECT f.FavoriteDrink, AVG(TIMESTAMPDIFF(YEAR, p.DOB, CURDATE())) AS AvgAge
FROM Person p
JOIN favorites f ON p.PersonID = f.PersonID
GROUP BY f.FavoriteDrink;
```

FavoriteDrink	AvgAge
Lemonade	34.0000
Coffee	31.0000
Smoothie	33.0000
Iced Tea	26.0000
Green Tea	35.0000
Coconut Water	32.5000
Fruit Juice	31.5000
Water	35.5000
Hot Chocolate	31.0000
Fruit Smoothie	27.0000
Sparkling Water	32.0000
Herbal Tea	35.5000
Iced Coffee	40.0000

4.3 Display Average Age of People Who Like Hiking

```
SELECT AVG(TIMESTAMPDIFF(YEAR, p.DOB, CURDATE())) AS AvgAge
FROM Person p
JOIN favorites f ON p.PersonID = f.PersonID
WHERE f.FavoriteActivity = 'Hiking';
```

Result Grid	
	AvgAge
▶	33.0000

4.4 Display Total People per City in Ascending Order

```
SELECT City, COUNT(*) AS TotalPeople
FROM Address
GROUP BY City
ORDER BY TotalPeople ASC;
```

City	TotalPeople
London	1
Manchester	1
Birmingham	1
Bristol	1
Liverpool	1
Glasgow	1
Leeds	1
Newcastle	1
Sheffield	1
Nottingham	1
Cambridge	1
Oxford	1
Southampton	1
Leicester	1
Norwich	1
Edinburgh	2
Cardiff	3

4.5 Display Name of Person(s) Whose Neighbour is 'Neighbour C'

```
SELECT p.Name
FROM Person p
JOIN Neighbour n ON p.PersonID = n.PersonID
WHERE n.NeighbourName = 'Neighbor C';
```

Result Grid	
	Name
▶	Person 2

5. Conclusion

This task demonstrated the process of database normalization to 3NF, ER modeling, SQL-based database, population, and query execution. The relational model ensures data integrity.

Note: All of these implementations are found here: https://gitlab.uwe.ac.uk/y2-youssef/advanced-db/-/tree/main/Task%201?ref_type=heads