Part B

Design Normalized Database

Bhanu Harsha Yanamadala

03/11/2025

1. Introduction

This document outlines a procedure for creating a **normalized relational database schema** to handle **customer restaurant visit data**, adhering to **Third Normal Form (3NF)** which helps us to **remove redundancy and maintain data consistency** of the restaurant database. The process involves identifying the initial functional dependencies for a customer restaurant visit relation before schema normalization that includes all relevant columns, breaking it down to **remove any transitive dependencies** through decomposition to achieve **3NF normalization**, and developing an **Entity-Relationship Diagram (ERD)** to visually represent the final **database structure** before creation of the actual database.

2. Functional Dependencies Before Schema Normalization

Functional Dependency
VisitID → {Restaurant, ServerEmpID, ServerName, StartDateHired, EndDateHired, HourlyRate, ServerBirthDate, ServerTIN, VisitDate, VisitTime, MealType, PartySize, Genders, WaitTime, CustomerName, CustomerPhone, CustomerEmail, LoyaltyMember, FoodBill, TipAmount, DiscountApplied, PaymentType, orderedAlcohol, AlcoholBill}

"Normalization Of The Schema"

3. Normalization Approach

Normalizing the **Restaurant Visits DB** according to **Third Normal Form(3NF)** will help us to **eliminate the duplicate data** which is piled up in the db, **increased data accuracy** while querying, and **simplifies handling of data** without any ambiguity. The below section tells us about the steps that are carried out.

The initial relational database contained several attributes in multiple tables which violated the atomicity principle, resulting in duplication and the increased risk of inconsistencies during insert, update, and delete actions. By breaking it down into smaller, specialized entities—like `CustomerDetails`, `RestaurantDetails`,`VisitDetails`, `MealType`, `ServerDetails`, `BillDetails` and `PaymentType`—we ensured that each table focuses on a single idea and eliminates redundancy.Hence there wont be any transitive dependencies.

Every table was designed to maintain the functional dependencies, ensuring that non-key attributes rely entirely on the primary key. This approach minimizes the risk of transitive dependencies, which could otherwise cause anomalies and data inconsistencies.

4. Entity Relationship Diagram(ERD) Of the Normalized Relational Database

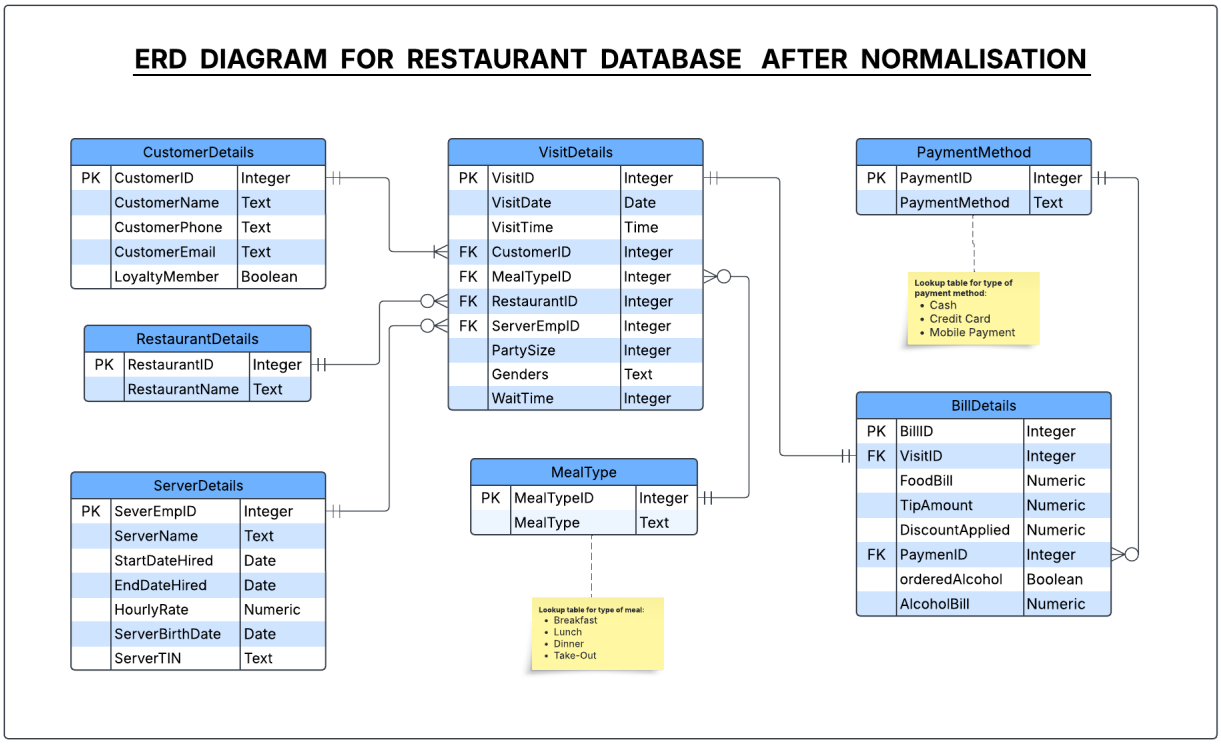

Figure 1: ERD Diagram

5.List of the Functional Dependencies after the Schema Normalisation

Functional Dependency
CustomerID → {CustomerName, CustomerPhone, CustomerEmail, LoyaltyMember}
RestaurantID → {RestaurantName}
MealTypeID → {MealType}
VisitID → {VisitDate, VisitTime, CustomerID, MealTypeID, RestaurantID, PartySize, Genders, WaitTime, ServerEmpID}
ServerEmpID → {ServerName, ServerBirthDate, ServerTINStartDateHired, EndDateHired, HourlyRate}
BillID → {VisitID, FoodBill, TipAmount, orderedAlcohol, AlcoholBill}, DiscountApplied, PaymentID}
PaymentID → {PaymentType}

6. Proof Of 3NF Compliance

VisitDetails table:

1NF: It is in 1NF because all attributes (VisitID, VisitDate, VisitTime, CustomerID, MealTypeID, RestaurantID, PartySize, Genders, WaitTime, ServerEmpID) contain atomic values and there are no repeating groups of attributes.

2NF: It is in 2NF because it has 'VisitID' as the primary key, and all non-key attributes (VisitDate, VisitTime, CustomerID, MealTypeID, RestaurantID, PartySize, Genders, WaitTime, ServerEmpID) are fully functionally dependent on the primary key (VisitID).

3NF: It is in 3NF since transitive dependencies do not exist. All non-key attributes depend directly on VisitID and not on other non-key attributes.

CustomerDetails table:

1NF: It is in 1NF because all attributes (CustomerID, CustomerName, CustomerPhone, CustomerEmail, LoyaltyMember) contain atomic values and there are no repeating groups of attributes.

2NF: It is in 2NF because it has 'CustomerID' as the primary key, and all non-key attributes (CustomerName, CustomerPhone, CustomerEmail, LoyaltyMember) are fully functionally dependent on the primary key (CustomerID).

3NF: It is in 3NF since transitive dependencies do not exist. All non-key attributes depend directly on CustomerID and not on other non-key attributes.

RestaurantDetails table:

1NF: It is in 1NF because all attributes (RestaurantID, RestaurantName) contain atomic values and there are no repeating groups of attributes.

2NF: It is in 2NF because it has 'RestaurantID' as the primary key, and the non-key attribute (RestaurantName) is fully functionally dependent on the primary key (RestaurantID).

3NF: It is in 3NF since transitive dependencies do not exist. The non-key attribute (RestaurantName) depends directly on RestaurantID and not on other non-key attributes.

MealTypeDetails table:

1NF: It is in 1NF because all attributes (MealTypeID, MealType) contain atomic values and there are no repeating groups of attributes.

2NF: It is in 2NF because it has 'MealTypeID' as the primary key, and the non-key attribute (MealType) is fully functionally dependent on the primary key (MealTypeID).

3NF: It is in 3NF since transitive dependencies do not exist. The non-key attribute (MealType) depends directly on MealTypeID and not on other non-key attributes.

ServerDetails table:

1NF: It is in 1NF because all attributes (ServerEmpID, ServerName, ServerBirthDate, ServerTIN) contain atomic values and there are no repeating groups of attributes.

2NF: It is in 2NF because it has 'ServerEmpID' as the primary key, and all non-key attributes (ServerName, ServerBirthDate, ServerTIN, StartDateHired, EndDateHired, HourlyRate) are fully functionally dependent on the primary key (ServerEmpID).

3NF: It is in 3NF since transitive dependencies do not exist. All non-key attributes depend directly on ServerEmpID and not on other non-key attributes.

BillDetails table:

1NF: It is in 1NF because all attributes (BillID, VisitID, FoodBill, TipAmount, DiscountApplied, PaymentID, OrderedAlcohol,AlcoholBill) contain atomic values and there are no repeating groups of attributes.

2NF: It is in 2NF because it has 'BillID' as the primary key, and all non-key attributes (VisitID, FoodBill, TipAmount, DiscountApplied, PaymentID, OrderedAlcohol,AlcoholBill) are fully functionally dependent on the primary key (BillID).

3NF: It is in 3NF since transitive dependencies do not exist. All non-key attributes depend directly on BillID and not on other non-key attributes.

PaymentType table:

1NF: It is in 1NF because all attributes (PaymentID, PaymentType) contain atomic values and there are no repeating groups of attributes.

2NF: It is in 2NF because it has 'PaymentID' as the primary key, and the non-key attribute (PaymentType) is fully functionally dependent on the primary key (PaymentID).

3NF: It is in 3NF since transitive dependencies do not exist. The non-key attribute (PaymentType) depends directly on PaymentID and not on other non-key attributes.

7.Conclusion

The normalization of the restaurant visit database to Third Normal Form (3NF) successfully eliminates redundancy, enhances data integrity, and streamlines data management by adhering to the principles of atomicity and functional dependency. Through the decomposition of the initial relation into specialized entities—such as `CustomerDetails`, `RestaurantDetails`, `VisitDetails`, `MealType`, `ServerDetails`, `BillDetails` and `PaymentType`—the schema ensures that each table represents a single concept, free from transitive dependencies and anomalies during insert, update, and delete operations. The final structure, validated through the proof of 3NF compliance and illustrated by the Entity-Relationship Diagram (ERD).