Pharmacy

Greetings to all the pharmacists around the world

PHARMACY MANAGEMENT SYSTEM

TEAM MEMBERS

1. MUHAMMAD HUZAIFA
2. HASSAN RAZA

Table of Contents

Abstract:

The Pharmacy Management System addresses critical issues faced by pharmacies, including security concerns with record-keeping and the challenges of manual drug information handling. This system provides a systematic approach for pharmacists to manage their operations efficiently.

With the Pharmacy Management System, pharmacists can input a medicine's name and access comprehensive details instantly. This includes dosage information, expiration dates, and other essential details crucial for medication management. The system's ability to display real-time information greatly enhances decision-making processes.

In larger medical stores, managing numerous drug specifics manually becomes cumbersome. This system improves this challenge by maintaining a comprehensive database of medicines, regularly updated with new information as new products are introduced. It also includes features such as expiration date tracking and a robust search function, simplifying the process of retrieving vital drug information.

The Pharmacy Management System was developed using MySQL Workbench for database management. This choice ensured data integrity and efficient handling capabilities. The system empowers pharmacies to streamline their operations, improve accuracy, and enhance overall productivity through its modern technology integration with MySQL.

Introduction:

The Pharmacy Management System is an essential database application designed to fulfill the comprehensive requirements of storing, managing, querying, and retrieving pharmacy-related data. It involves the management of staff, customers, transactions, products, and suppliers within a pharmacy setting.

Description:

The decision to develop the Pharmacy Management System stemmed from the inherent complexities and large data management needs of a modern pharmacy. While manual systems using files or spreadsheets could suffice, they often fall short of providing the ideal practice for efficient data handling. Recognizing this, the system was conceptualized as a robust entity with distinct components, each with its attributes and operations, as delineated in the schema.

This schema aims to automate and streamline various operations, including adding, updating, and deleting records, generating reports, and performing calculations.

The schema comprises normalized tables representing crucial entities such as Staff, Customers, Suppliers, Products, Transactions, and Categories. Each table is designed carefully to maintain data integrity, ensure referential integrity through foreign keys, and enforce necessary constraints for data accuracy.

For instance, the Staff table records essential details like Staff ID, Name, Job, Salary, Hire Date, and Contact Information. Similarly, the Product table includes details like Product ID, Name, Supplier ID, Category ID, Quantity, and Unit Price, ensuring precise product management and inventory control.

The system's functionality extends to encompassing staff addresses, customer information, supplier details, transaction records, and transaction details to provide a comprehensive solution for pharmacy management. Data insertion, modification, and retrieval are streamlined through structured queries and intuitive interfaces, enhancing operational efficiency and decision-making processes within a pharmacy setting.

Problem Statement:

Many pharmacies encounter challenges such as insufficient service promotions, lack of coherence in pharmacy services within hospitals, poor drug information systems, and inconsistencies in pharmacy information management due to manual processes. These issues delay efficient operations, leading to potential errors, delays, and inefficiencies in service delivery.

Solution:

The Pharmacy Management System Project proposes to address pharmacy challenges by leveraging MySQL Workbench for a robust backend database. This solution brings streamlined data management, efficient operations, enhanced security, improved decision-making, and seamless integration into existing workflows. The system aims to revolutionize pharmacy operations, leading to improved efficiency, accuracy, and productivity, benefiting both the pharmacy and its customers.

Proposal:

The Pharmacy Management System Project aims to address these challenges by implementing a comprehensive solution for storing, managing, and organizing medication-related data within pharmacies. The system will leverage modern technology, specifically MySQL Workbench, to establish a robust backend database management system. This database will ensure data integrity, efficient data handling capabilities, and enhanced security measures.

Key Features and Scope:

1. Staff Management: Manage staff details such as ID, name, salary, job role, and contact information.
2. Customer Management: Maintain customer records including ID, name, and contact details for effective communication and service delivery.
3. Supplier Management: Store supplier information including ID, company name, address, and contact details to facilitate seamless supply chain operations.
4. Product Management: Manage product details such as ID, name, supplier ID, category ID, quantity, and unit price for accurate inventory management.
5. Transaction Management: Track transactions including transaction ID, customer ID, staff ID, transaction date, and total amount for financial tracking and reporting.

Schema

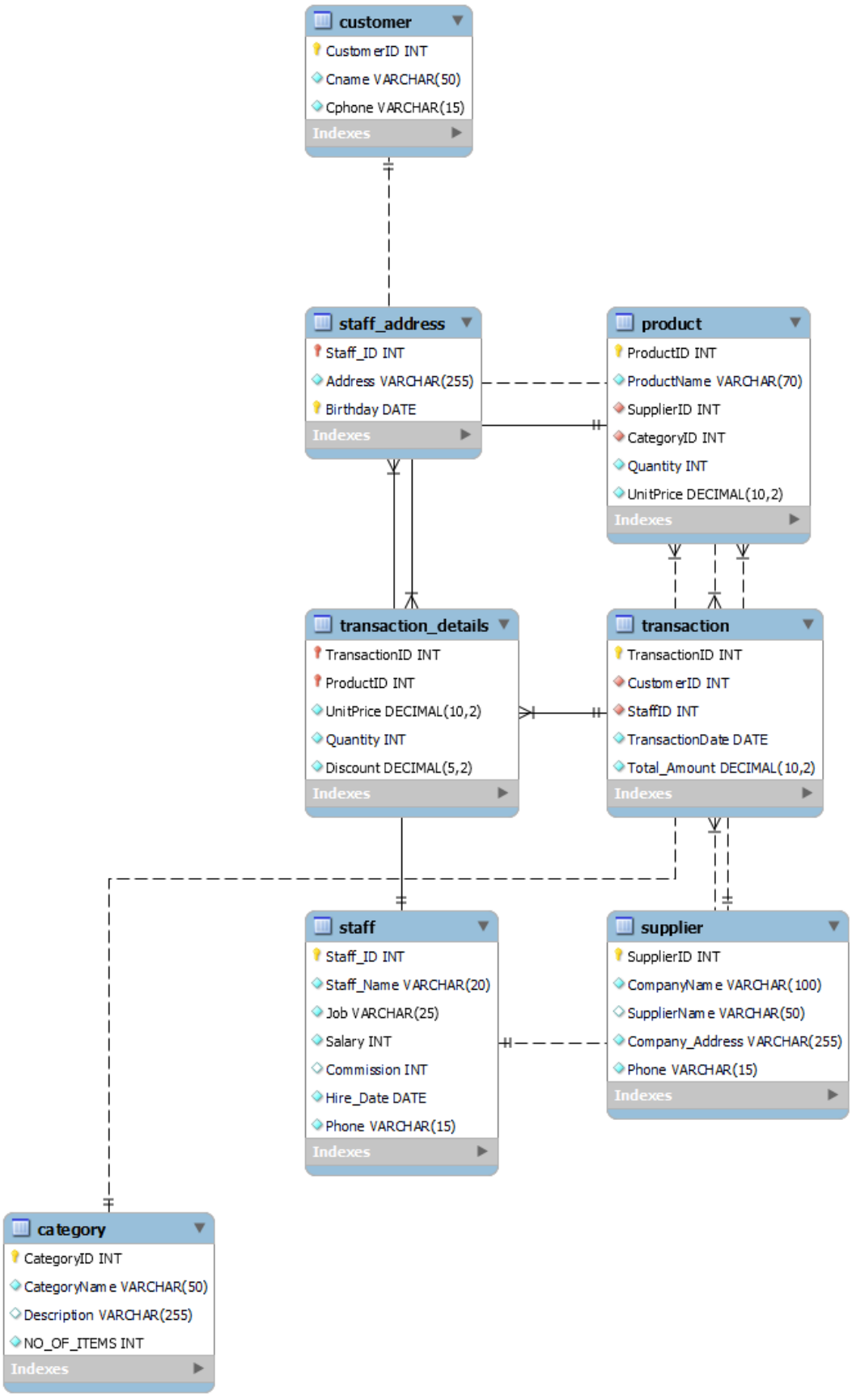
```
CREATE SCHEMA pharmacy_management_system ;

CREATE TABLE staff(
  Staff_ID INT NOT NULL,
  Staff_Name VARCHAR(20) NOT NULL,
  Job VARCHAR(25) NOT NULL,
  Salary INT NOT NULL,
  Commission INT NULL,
  Hire_Date DATE NOT NULL,
  Phone VARCHAR(15) NOT NULL,
  CONSTRAINT PK_Staff
  PRIMARY KEY (Staff_ID)
);


CREATE TABLE staff_address(
  Staff_ID INT NOT NULL,
  Address VARCHAR(255) NOT NULL,
  Birthday DATE NOT NULL,
  PRIMARY KEY (Staff_ID, Birthday),
  FOREIGN KEY (Staff_ID) REFERENCES staff(Staff_ID)
);
```

```
CREATE TABLE category(
  CategoryID INT NOT NULL,
  CategoryName VARCHAR(50) NOT NULL,
  Description VARCHAR(255) NULL,
  NO_OF_ITEMS INT NOT NULL DEFAULT 0,
  PRIMARY KEY (CategoryID)
);

CREATE TABLE customer(
  CustomerID INT NOT NULL,
  Cname VARCHAR(50) NOT NULL,
  Cphone VARCHAR(15) NOT NULL,
  PRIMARY KEY (CustomerID)
);

CREATE TABLE Supplier(
  SupplierID INT NOT NULL,
  CompanyName VARCHAR(100) NOT NULL,
  SupplierName VARCHAR(50) NULL,
  Company_Address VARCHAR(255) NOT NULL,
  Phone VARCHAR(15) NOT NULL,
  PRIMARY KEY (SupplierID)
);
```

```
CREATE TABLE Product(
  ProductID INT NOT NULL,
  ProductName VARCHAR(70) NOT NULL,
  SupplierID INT NOT NULL,
  CategoryID INT NOT NULL,
  Quantity INT NOT NULL DEFAULT 0,
  UnitPrice DECIMAL(10,2) NOT NULL,
  PRIMARY KEY (ProductID),
  CONSTRAINT CHK_Product_Price CHECK ((UnitPrice >= 0)),
  FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID),
  FOREIGN KEY (CategoryID) REFERENCES category(CategoryID)
);

CREATE TABLE transaction(
  TransactionID INT NOT NULL,
  CustomerID INT NOT NULL,
  StaffID INT NOT NULL,
  TransactionDate DATE NOT NULL,
  Total_Amount DECIMAL(10,2) NOT NULL,
  PRIMARY KEY (TransactionID),
  FOREIGN KEY (CustomerID) REFERENCES customer(CustomerID),
  FOREIGN KEY (StaffID) REFERENCES staff(Staff_ID)
);
```

```
CREATE TABLE transaction_details(
  TransactionID INT NOT NULL,
  ProductID INT NOT NULL,
  UnitPrice DECIMAL(10,2) NOT NULL,
  Quantity INT NOT NULL,
  Discount DECIMAL(5,2) NOT NULL DEFAULT 0.00,
  PRIMARY KEY (TransactionID,ProductID),
  CONSTRAINT CHK_Discount   CHECK ((Discount >= 0 and Discount <= 1)),
  CONSTRAINT CHK_Quantity   CHECK ((Quantity > 0)),
  CONSTRAINT CHK_UnitPrice   CHECK ((UnitPrice >= 0)),
  FOREIGN KEY (TransactionID) REFERENCES transaction(TransactionID),
  FOREIGN KEY (ProductID) REFERENCES product(ProductID)
);
```

Entity Relationship Diagram:



Normalization

Functional Dependencies:

In start we identify these tables with these attributes and then find functional dependencies from each table.

1. **Staff:** StaffID (PK), SName, job, Salary, Commission, Hiredate, sAddress, Phone,BIRTHDAY

Functional Dependencies:

Staffid -> SName, job, Salary, Commission, Hiredate, Phone

Staffid,sName - > saddress,birthday

2. **Category**: CategoryID (PK), CategoryName, Description, NO_OF_ITEMS

Functional Dependencies:

CategoryID -> CategoryName, NO_OF_ITEMS, Description

CategoryName -> Description

3. **Customer:** custumerID (PK), Cname, Cphone

Functional Dependencies:

custumerID -> Cname, Cphone

4. **Supplier:** supplierID (PK), CompanyName, supplierName, company_address, phone

Functional Dependencies:

supplierID -> CompanyName, supplierName, phone

CompanyName -> company_address

5. **Product:** ProductID (PK), ProductName, SupplierID (FK), CategoryID (FK), Quantity , UnitPrice (>0)**,** EXP_DATE

Functional Dependencies:

productid -> productName,Quantity,unitprice

productid, productName -> exp_date

6. **Transaction:** TransactionID (PK), CustomerID (FK), StaffID (FK), TransactionDate , total_amount.

Functional Dependencies:

transactionID -> transactionDate, total amount, customerID, StaffID

7. **TransactionDetails: (TransactionID (FK), ProductID (FK))(**PK), UnitPrice (>=0), Quantity (>0), Discount (>=0 AND<=1)

Functional Dependencies:

TransactionID , ProductID -> unitprice, quantity, Discount

2NF:

Now removing the partial dependencies to make the tables in 2NF.

Pharmacy Management:

1. Staff:{ StaffID (PK), SName, job, Salary, Commission, Hiredate, Phone }
2. **Staff_adress: {**Staffid, sName, Saddress, birthday}

Fd:

Staffid -> SName, job, Salary, Commission, Hiredate, Phone

Staffid,sName - > saddress,birthday

3. **Category:** CategoryID (PK), CategoryName, Description, **NO_OF_ITEMS**

FD:

CategoryID -> CategoryName, NO_OF_ITEMS, Description

4. **Customer:** custumerID (PK), Cname, Cphone

custumerID -> Cname, Cphone

5. **Supplier:** supplierID (PK), CompanyName, supplierName, company_address, phone

supplierID -> CompanyName, supplierName, phone

CompanyName -> company_address

6. **Product:** ProductID (PK), ProductName, SupplierID (FK), CategoryID (FK), Quantity , UnitPrice (>0)
   productid -> productName,Quantity,unitpric
7. **Transaction:** TransactionID (PK), CustomerID (FK), StaffID (FK), TransactionDate , total_amount.
   transactionID -> transactionDate, total amount, customerID, StaffID
8. **TransactionDetails: (TransactionID (FK), ProductID (FK))(**PK), UnitPrice (>=0), Quantity (>0), Discount (>=0 AND<=1)

TransactionID , ProductID -> unitprice, quantity, Discount

Final Normalized Tables:

1. **Staff:** StaffID (PK), SName, job, Salary, Commission, Hiredate, Phone
2. **StaffAddress:** StaffID (FK - references Staff.StaffID), Address, Birthday (PK)
3. **Category:** CategoryID (PK), CategoryName, Description, NO_OF_ITEMS
4. **Customer:** customerID (PK), Cname, Cphone
5. **Supplier:** supplierID (PK), CompanyName, supplierName, company_address, phone
6. **Product:** ProductID (PK), ProductName, SupplierID (FK), CategoryID (FK), Quantity , UnitPrice (>0)
7. **Transaction:** TransactionID (PK), CustomerID (FK), StaffID (FK), TransactionDate , total_amount
8. **TransactionDetails:** (TransactionID (FK), ProductID (FK)) (PK), UnitPrice (>=0), Quantity (>0), Discount (>=0 AND<=1)