Quantum
UNIVERSITY
ROORKEE, INDIA
Growth
Unbound!

The Future is exciting..

Quantum UNIVERSITY

The Future of Education is here

PHARMACY MANAGEMENT SYSTEM

PRESENTED BY : ABHISHEK RAI
HEMANT KUMAR
YASHWANT SAINI
AYUSHMAN SINGH
NIKHIL GAUTAM

B.Tech CSE AIML

INTRODUCTION

- Pharmacies are essential component of healthcare and handle the function of selling medical drugs.
- Even though the pharmacies do not seem different than any other shop, their functioning is very different due to various laws regarding drugs.
- For example, most of the drugs available in a pharmacy cannot be purchased without a prescription
- In addition, there are other laws on the operations of pharmacy like requirement for safe disposal of expired medicine and requirement of license for employees that mix/prepare the drugs.
- Thus, preparing a Database Management System for a pharmacy not only requires study of how things are handled from a customer or employee point of view but also the relevant laws.

Requirements

During research phase, we arrived at following requirements based on the pharmacy flow:

- **Customer:**When a customer arrives in the pharmacy, we identify them based on their SSN. If they are a new customer, they are asked for their name, date of birth, phone number, gender and address.
- **Employee :**An employee has same details as a customer but they are also given a company ID, that is unique for them. An employee has to have one of the following roles:
    - 1. Pharmacist
    - 2. CPhT (Certified Pharmacy Technician)
    - 3. Intern (can work in the pharmacy part time)
    - 4. Cashier

Quantum UNIVERSITY

- **Prescription:**Most of the drugs in the pharmacy can only be sold with a prescription. A prescription contains customer's SSN, the prescribing Doctor's ID (required by law) and when the prescription was prescribed.

- **Order:**An order is created from the prescription. This data has to be stored separately because customer may: 1. Buy less medicine than prescription specifies 2. Come back for refills based on same prescription

- **Bill:**Once an order has been completed, a bill is generated by the system. This bill is handed over to the customer.

- **Medicine(Inventory):**Drugs are divided into "over the counter", "restricted" and "prescription only". Federal Law only divides restricted drugs into 5 schedules and require "readily accessible" inventory for schedule 2 drugs.

Notifications: The system should be able to generate notifications based on the following four events:

1. Stock for a medicine is low (less than 100 tablets)
2. 2. Some medicine will expire in next 60 days
3. 3. Drugs are marked for disposal
4. 4. Drugs are successfully disposed

Laws Affecting Pharmacies

- Food, Drug and Cosmetic (FDC) Act of 1938
- Comprehensive Drug Abuse Prevention and Control Act(CASA) of 1970
- Poison Prevention Packaging Act(PPPA) of 1970
- Omnibus Budget Reconciliation Act (OBRA) of 1990Omnibus Budget Reconciliation Act (OBRA) of 1990
- Health Insurance Portability and Accountability Act(HIPPA) of 1996

ER Modelling

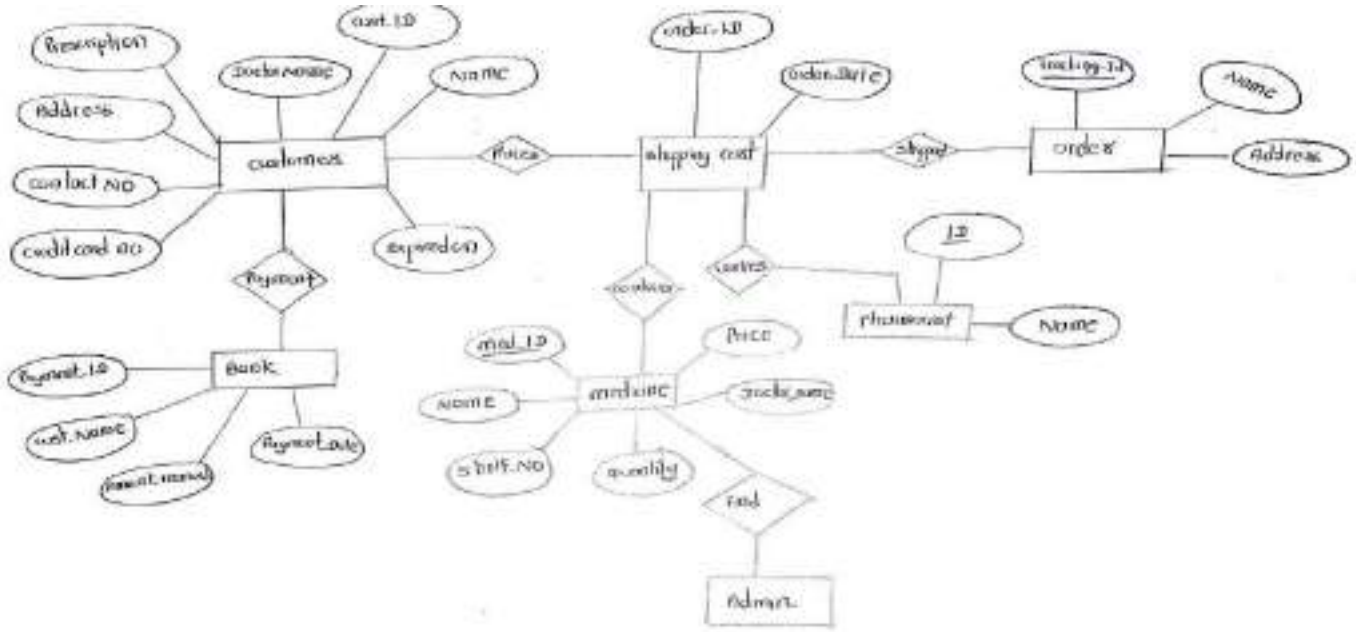

fig. ER Diagram of Pharmacy Management System

Quantum UNIVERSITY

Data Dictionary :-

A data dictionary in a Relational Database Management System (RDBMS) is essentially a collection of read-only tables and views that store information about the database itself. This information, also known as metadata, describes the structure and relationships of data with in the database.

It contains information about the database like what is in the database owners, storage information. It is handeled by the database administrators. In the context of a pharmacy Management System (PMS) the data dictionary acts like a detailed blueprint for all the information stored within the system.

Data Dictionary contains the following information

1. Name of all the database tables and their schemas.
2. Detail about all the tables of the Database such as their owners, security constraints, Date of creation, storage etc.
3. Table constraints like primary key, foreign key etc information about views.

Types of Data Dictionary :-

Active Data Dictionary :- It is in built with the DBMS. Automatically updated according to the change in database.

Passive Data Dictionary :- It is maintained seperately to the Database. It is not updated automatically.

Quantum UNIVERSITY

Data Dictionary of Entity Relationship Diagram for Pharmacy Database System

Entity	Attributes	Data type	Primary key	Foreign key	constraints	Length
Patient	Patient ID	Integer	Patient ID	Physician ID	NOT NULL, a	20
	Physician Name	CHAR			Not Null	255
	physician street	CHAR			Null	50
	Physician ID	Integer			Not Null	20
	Physician city	CHAR			Not Null	25
	Physician zipcode	Integer			Not Null	50
Medication	Medication_ID	Integer	Medication_ID	No	NOT NULL	Auto_increment
	Medication name	string			NOT NULL	255
	Dosage_form	string			NOT NULL	50
	strength	string			NOT NULL	25
~~Dosage_Form~~ Physician	Physician_ID	Integer	Physician_ID	Patient_ID	NOT NULL	Auto_increment
	First_Name	CHAR			NOT NULL	string 50
	Last_Name	CHAR			NOT NULL	50
	Patient_ID	Integer			NOT NULL	20
	Physician state	CHAR			NOT NULL	50
	Physician Zipcode	Integer			NOT NULL	10
	~~Patient_ID~~					
Pharmacy	Pharmacy_ID	Integer	Pharmacy_ID	No	NOT NULL	Auto_increment
	Name	string			NOT NULL	250
	Address_1	string			NOT NULL	255
	Address_2	string			Allow NULL	250
	city	string			NOT NULL	150
	Phone_no	Integer			NOT NULL	50
Drug	Drug_ID	Integer	Drug_ID	No	NOT NULL	60
	Drug name	string			NOT NULL	100
	unit_of_measure	string			NOT NULL	150
	strength	string			NOT NULL	255

Quantum UNIVERSITY

Entity	Attributes	Data type	Primary key	Foreign key	Constraints	Length
Payment	Payment_ID Prescription_ID Payment method Payment amount Payment Date	Integer Integer String Decimal Date	Payment_ID	Prescription_ID	NOT NULL NOT NULL NOT NULL NOT NULL NOT NULL	Auto increment Integer string 50 (10,2) Date
Prescription	prescription_ID patient_ID physician_ID Drug_ID Date_written Quantity_Dispensed	Integer Integer Integer Integer Date Integer	prescription_ID	Patient_ID Physician_ID Drug_ID	NOT NULL NOT NULL NOT NULL NOT NULL NOT NULL Allow NULL	Auto increment Auto increment Auto increment Auto increment Date 50
Order	order_ID order Date order_status payment_ID shipping method	Integer Date String Integer String	order_ID	payment_ID	NOT NULL Allow NOT NULL NOT NULL Allow NULL Allow NULL	Auto increment 50 50 Auto increment 255
Shipping cost	Cart_ID patient_ID Quantity Date_added	Integer	Cart_ID	patient_ID	NOT NULL Allow NULL NOT NULL NOT NULL	Auto increment Auto increment 50 50
Supplier	Supplier_ID Supplier_name Address_1 Address_2	Integer String String String	Supplier_ID	NO	NOT NULL NOT NULL NOT NULL Allow NULL	Auto increment 255 255 255

Generalization and Specialization

Generalization and Specialization are the processes used to create a hierarchial relationship between a higher-level entity type (Supertype) and lower-level entity type (subtypes).

- Generalization :-

Generalization is the process of extracting shared characteristics from multiple entities and combining them into generalized supertype. This a bottom-up approach.

In a pharmacy management system, we might have different types of users such as 'Pharmacists', 'Doctors' and 'Patients'. These entities share common attributes like 'user_id', 'name', 'address' and 'phone_numbers'. By using generalization, we can combine these common attributes into a 'Users' Supertype.

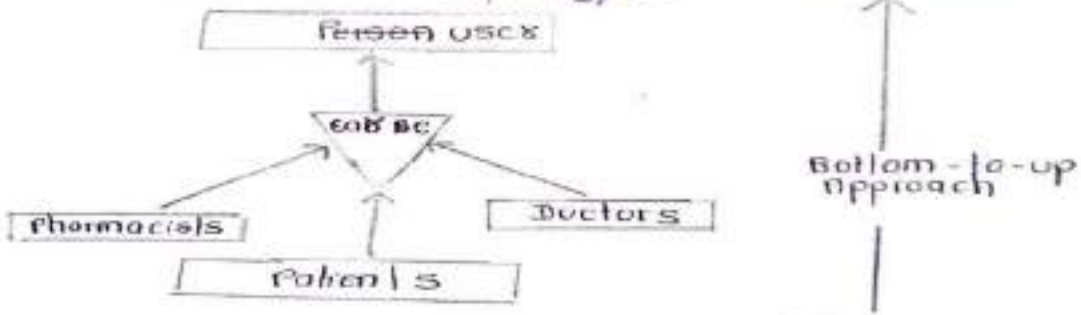

Fig: Generalization example.

Specialization

Specialization in general terms, is a process of mastering a specific domain. But, with respect to ER Model, specialization is the procedure to split up the entities into further sub entities on the basis of their functionalities, specialities and features - This process is a Top-to-Down approach

'Doctors' and 'patients'.

create specialized subtypes for 'pharmacists',

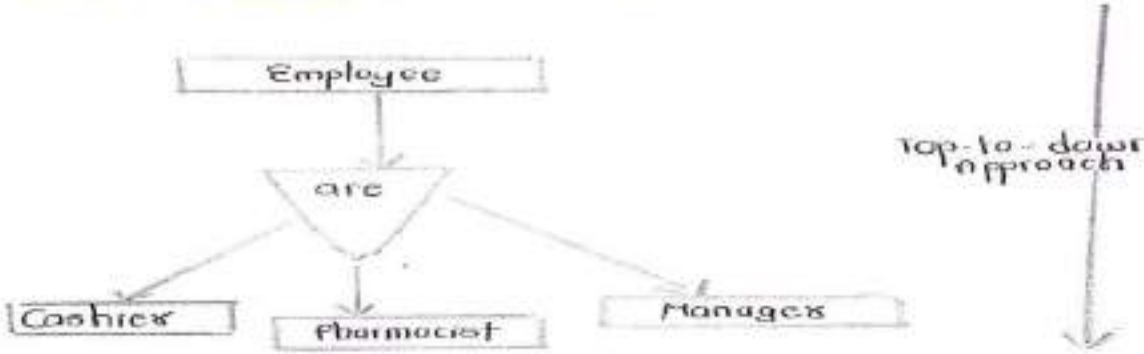

Fig: Specialization example

Super type :- A supertype is a generic entity type that has a relationship with one or more subtypes. It represents a higher level abstraction in the data model.

example :- In pharmacy management system Employee is a supertype that includes common to all its subtypes. that includes common attributes for all employees in the pharmacy, such as 'employee_id', 'name', 'address', 'phone_numbes', and salary.

- A single customer can have multiple prescriptions. Thus, the relation between them is one to many.
- A prescription consists of multiple drugs, so the relation is one to many. In case of refills, a prescription can generate multiple orders. So, this relation is one to many as well.
- A single order can contain multiple drugs, thus relationship is one to many. One order, however, can generate only one bill. Thus, the relation between bill and order is one to one.
- A customer can make multiple purchases and hence, the relation between customer and bill is one to many. This is due to the fact that every bill has only one customer.
- In medicine table (stock), drug name and batch number can uniquely identify every drug we have in inventory. Batch number is assumed to be unique among manufacturers.
- Disposed drugs are weak entity and use foreign key Drug Name and Batch Number as their primary key.
- One employee can receive multiple notifications and one notification can be sent to multiple employees, thus relationship is many to many.

- Multiple employees can dispose same drug. Similarly, one employee can dispose multiple drugs. Hence, relationship is many to many.

- One employee can prepare multiple orders. However, a specific order can only be prepared by one employee. Thus, relationship is one to many.

Relations and Normalization

Relations

The final relations are listed below:

Customer

SSN	First Name	Last Name	Phone	Gender	Address	Date of Birth	Insurance ID

Primary Key: SSN

Foreign Key: Customer(Insurance ID) → Insurance(Insurance ID)

Insurance

Insurance ID	Company Name	Start Date	End Date	Co-Insurance

Primary Key: Insurance ID

Employee

ID	SSN	License	First Name	Last Name	Start Date	End Date	Role
Salary	Phone Number	Date of Birth					

Primary Key: ID

Prescription

Prescription ID	SSN	Doctor ID	Prescription Date

Primary Key: Prescription ID

Foreign Key: Prescription(SSN) → Customer(SSN)

Prescribed Drugs

Prescription ID	Drug Name	Prescribed Quantity	Refill Limit

Primary Key: Prescription ID, Drug Name

Foreign Key: Prescribed Drugs(Prescription ID) → Prescription(Prescription ID)

Order

Order ID	Prescription ID	EmployeeID	Order Date

Primary Key: Order ID

Foreign Key: Order(Prescription ID) → Prescription(Prescription ID), Order(Employee ID) → Employee(ID)

Notification

ID	Message	Type

Primary Key: ID

Employee_Disposed Drugs

Employee ID	Drug Name	Batch Number	Disposal Date

Primary Key: Employee ID, Drug Name, Batch Number, Disposal Date

Foreign Key: Employee_Disposed Drugs(Employee ID) → Employee (Employee ID), Employee_Disposed Drugs(Drug Name, Batch Number) → Disposed Drugs(Drug Name, Batch Number)

Employee Notification

Employee ID	Notification ID

Quantum UNIVERSITY

Table Creation

SQL commands for creating the tables in our database:

```
CREATE TABLE CUSTOMER (
  SSN           NUMBER(10) NOT NULL,
  first_name    CHAR(255)  NOT NULL,
  last_name     CHAR(255)  NOT NULL,
  phone         NUMBER(10) NOT NULL UNIQUE,
  gender        CHAR(1)    NOT NULL,
  address       CHAR(1000) NOT NULL,
  date_of_birth DATE       NOT NULL,
  insurance_id  NUMBER(10) NOT NULL UNIQUE,
  PRIMARY KEY (SSN)
);
ALTER TABLE Customer
  ADD CONSTRAINT insures FOREIGN KEY (insurance_id) REFERENCES Insurance (insurance_id) ON DELETE
SET NULL;
```

```
CREATE TABLE Prescription (
  prescription_id NUMBER(10) NOT NULL,
  SSN             NUMBER(10) NOT NULL,
  doctor_id       NUMBER(10) NOT NULL,
  prescribed_date DATE       NOT NULL,
  PRIMARY KEY (prescription_id)
);
ALTER TABLE Prescription
  ADD CONSTRAINT holds FOREIGN KEY (SSN) REFERENCES Customer (SSN);

CREATE TABLE "PRESCRIBED_DRUGS" (
  prescription_id     NUMBER(10) NOT NULL,
  drug_name           CHAR(255)  NOT NULL,
  prescribed_quantity NUMBER(10) NOT NULL,
  refill_limit        NUMBER(10) NOT NULL,
  PRIMARY KEY (prescription_id,
               drug_name)
);
ALTER TABLE "PRESCRIBED_DRUGS"
```

```
 ADD CONSTRAINT "consists of" FOREIGN KEY (prescription_id) REFERENCES Prescription (prescription_id) ON
DELETE CASCADE;

CREATE TABLE "Order" (
 order_id       NUMBER(10) NOT NULL,
 prescription_id NUMBER(10) NOT NULL,
 EmployeeID     NUMBER(5)  NOT NULL,
 order_date     DATE       NOT NULL,
 PRIMARY KEY (order_id)
);
ALTER TABLE "Order"
 ADD CONSTRAINT prepares FOREIGN KEY (EmployeeID) REFERENCES Employee (ID);
ALTER TABLE "Order"
 ADD CONSTRAINT uses FOREIGN KEY (prescription_id) REFERENCES Prescription (prescription_id);
```

Quantum UNIVERSITY

```
CREATE TABLE Employee (
 ID          NUMBER(5)  NOT NULL,
 SSN         NUMBER(10) NOT NULL UNIQUE,
 License     NUMBER(10) UNIQUE,
 first_name  CHAR(255)  NOT NULL,
 last_name   CHAR(255)  NOT NULL,
 start_date  DATE       NOT NULL,
 end_date    DATE,
 role        CHAR(255)  NOT NULL,
 salary      NUMBER(4)  NOT NULL,
 phone_number NUMBER(10) NOT NULL,
 date_of_birth DATE     NOT NULL,
 PRIMARY KEY (ID)
);

CREATE TABLE Medicine (
 drug_name      CHAR(255)  NOT NULL,
 batch_number   NUMBER(10) NOT NULL,
 MedicineType   CHAR(255)  NOT NULL,
 Manufacturer   CHAR(255)  NOT NULL,
 stock_quantity NUMBER(10) NOT NULL,
 expiry_date    DATE       NOT NULL,
 Price          NUMBER(4)  NOT NULL,
 PRIMARY KEY (drug_name,
              batch_number)
);
```

```
CREATE TABLE Bill (
 order_id        NUMBER(10) NOT NULL,
 CustomerSSN     NUMBER(10) NOT NULL,
 total_amount    NUMBER(4)  NOT NULL,
 customer_payment  NUMBER(4)  NOT NULL,
 insurance_payment NUMBER(4)  NOT NULL,
 PRIMARY KEY (order_id,
              CustomerSSN)
);
```

```
ALTER TABLE Bill
 ADD CONSTRAINT makes FOREIGN KEY (order_id) REFERENCES "Order" (order_id);
ALTER TABLE Bill
 ADD CONSTRAINT pays FOREIGN KEY (CustomerSSN) REFERENCES Customer (SSN);
```

Quantum UNIVERSITY

```
CREATE TABLE Notification (
 ID      NUMBER(10) NOT NULL,
 Message CHAR(255)  NOT NULL,
 Type    CHAR(255)  NOT NULL,
 PRIMARY KEY (ID)
);

CREATE TABLE Employee_Notification (
 EmployeeID     NUMBER(5)  NOT NULL,
 NotificationID NUMBER(10) NOT NULL,
 PRIMARY KEY (EmployeeID,
              NotificationID)
);
```

```
DELETE CASCADE;
ALTER TABLE Employee_Notification
 ADD CONSTRAINT FKEmployee_N665447 FOREIGN KEY (NotificationID) REFERENCES Notification (ID) ON
DELETE CASCADE;
```

```
ALTER TABLE Employee_Notification
 ADD CONSTRAINT FKEmployee_N849182 FOREIGN KEY (EmployeeID) REFERENCES Employee (ID) ON
```

Conclusion

The pharmacy project was a good learning experience for implementing a real world DBMS and helped us understand the nuances of a full implementation.

The most interesting part was the experience of starting from real world and then translating the concepts into the terms of a DBMS.

The final implementation is robust and can handle various edge cases and scenarios. Paired with a capable application front end, it can handle day to day operations for a pharmacy

Quantum UNIVERSITY

INNOVATE EMPOWER PERSEVERE SUCCEED

www.quantumuniversity.edu.in