Pharmacy Database System

Professional Database Design & Implementation

Presented by:

- **Zaid Alshobaki**
- **Mohammad Alkharabsheh**
- **Reem Albanna**

Project Overview

Problem & Objective

- **Challenge:** Managing complex pharmacy data manually leads to errors and inefficiency.
- **Solution:** A robust SQL Server Database System to manage Patients, Doctors, Medicines, and Prescriptions.
- **Goal:** Ensure data integrity, security, and streamlined operations.

System Architecture

Pharmacist

Doctor

Manager

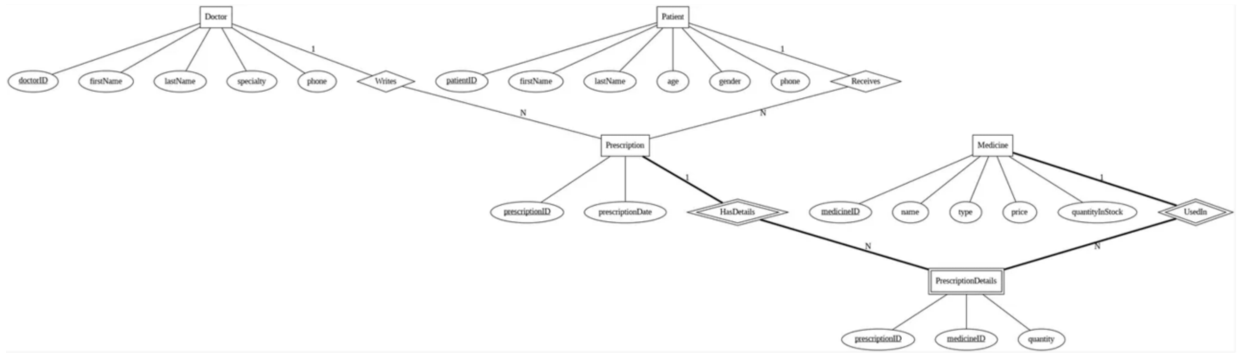
Core Functions

→ **Pharmacist:** Manage prescriptions and inventory.
→ **Doctor:** Write prescriptions and view patient history.
→ **Manager:** Analyze sales and stock levels.
→ **System:** CRUD operations, stock tracking, and reporting.

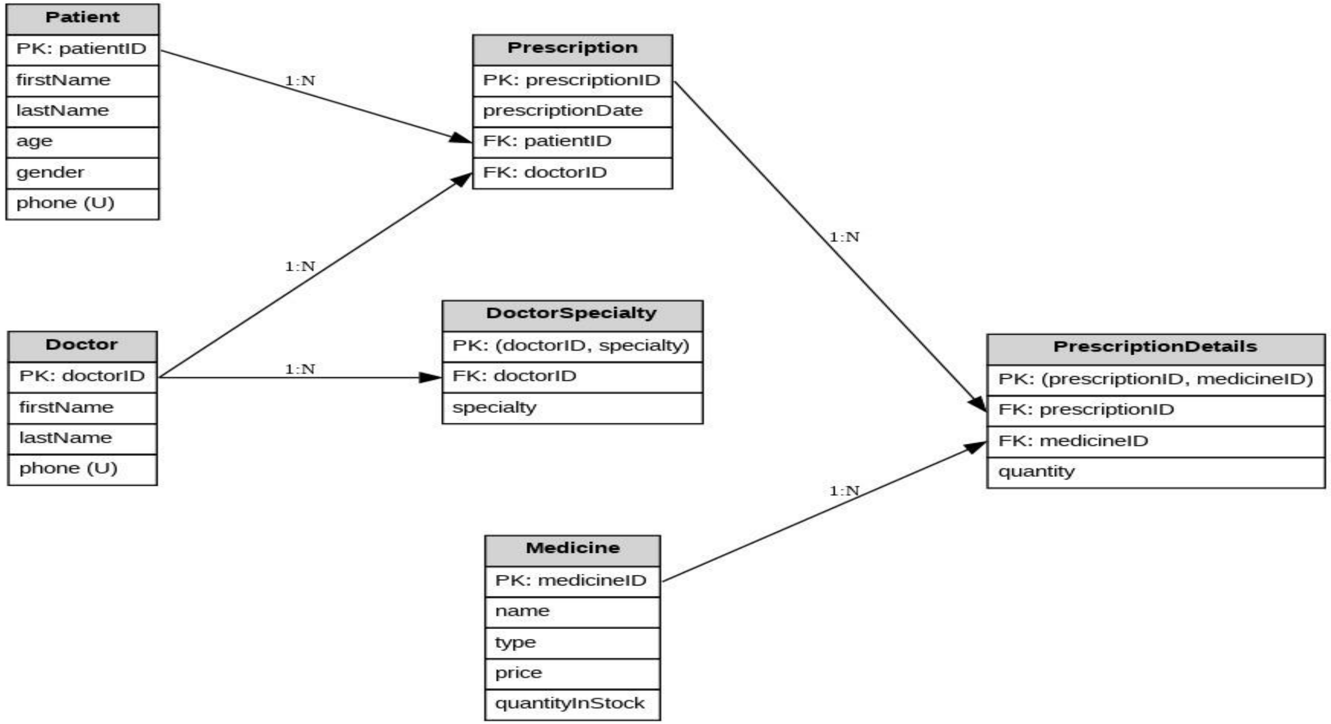
Entity Relationship Diagram

Conceptual Design

- **Chen Notation:** Standard academic representation**.**
- **Multi-valued:** *Specialty shown as double oval.*
- **Weak Entity:** *PrescriptionDetails with double rectangle.*
- **Identifying:** Double diamonds for dependent relationships.



Relational Mapping Schema



Precise PK-FK Mappings

Junction Tables for M:N

Clear Cardinality (1:N)

Database Implementation (DDL)

Defining the core structure and referential integrity.

```
CREATE TABLE Doctor (
   doctorID INT PRIMARY KEY,
   firstName VARCHAR(20) NOT NULL,
   lastName  VARCHAR(20) NOT NULL,
   phone   VARCHAR(10) NOT NULL UNIQUE
);

CREATE TABLE DoctorSpecialty (
   doctorID INT NOT NULL,
   specialty VARCHAR(20) NOT NULL,
   PRIMARY KEY (doctorID, specialty),
   FOREIGN KEY (doctorID) REFERENCES Doctor(doctorID)
);
```

Data Retrieval: Complex Joins

SQL Query: Prescription Summary

```
SELECT p.prescriptionID, p.prescriptionDate,
 pa.firstName + ' ' + pa.lastName AS Patient,
 d.firstName + ' ' + d.lastName AS Doctor
 FROM Prescription p
 JOIN Patient pa ON p.patientID = pa.patientID
 JOIN Doctor d ON p.doctorID = d.doctorID;
```

Query Output

ID	Date	Patient	Doctor
1	2026-01-05	Sarah Ahmad	Ahmad Saleh

Key Insight: This join combines data from three separate tables to provide a comprehensive view of medical prescriptions, ensuring all relevant entities are linked correctly.

Inventory Management

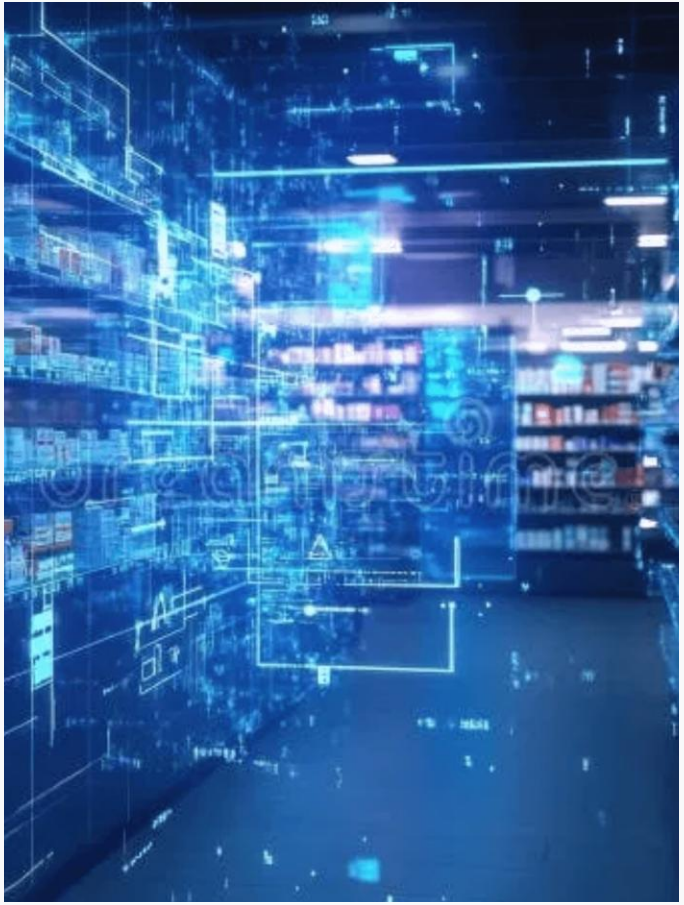
Low Stock Tracking

SQL Query

```
SELECT * FROM Medicine
 WHERE quantityInStock < 30;
```

Query Output

ID	Name	Type	Price	Stock
2	Amoxicillin	Capsule	8.00	20



Financial Reporting

Total Cost per Prescription

SQL Aggregation Query

```
SELECT pd.prescriptionID,
 SUM(pd.quantity * m.price) AS TotalCost
 FROM PrescriptionDetails pd
 JOIN Medicine m ON pd.medicineID = m.medicineID
 GROUP BY pd.prescriptionID;
```

Query Output

Prescription ID	Total Cost ($)
1	6.00

Business Value: Automated calculation of prescription totals ensures billing accuracy and provides immediate financial insights for pharmacy management.

Conclusion

- ✓ **Accuracy:** Eliminates manual data entry errors.
- ✓ **Scalability:** Designed to handle growing pharmacy data.
- ✓ **Insight:** Real-time reports for better decision making.

Thank You! Any Questions?