11/19/2023

LAB MID (Project Report)

Title: Pharmacy Store management

Submitted to: Sir Haris

Submitted by: Ayesha Siddiqa (SP22-BSE-009)

Um-e-Rubab (SP22-BSE-050)

COMSATS UNIVERSITY ISLAMABAD

Task 1: Identify the topic of your final project and explain various modules which will be present in the DBMS.

Topic: **Pharmacy Store Management System**

System Introduction:

The Pharmacy Store Management System is a comprehensive platform designed to facilitate seamless operations for both customers and administrators of the store. Customers can browse, order medicines, view medicine details, and manage orders, with the added convenience of getting detailed receipts upon completing their purchases/orders. The system ensures secure authentication and user-friendly interfaces for a smooth experience.

Administrators benefit from efficient inventory management tools, enabling actions like adding, updating, and deleting medicines, alongside overseeing stock levels and supply chain interactions. This system not only streamlines medication procurement but also offers transparency and record-keeping in the transaction process.

Problem Statement:

The manual system for managing pharmacy operations faces challenges such as limited accessibility for customers, inefficient inventory management prone to errors, lack of transparent transaction records, and inconvenient order management. The Pharmacy Store Management System aims to resolve these issues by offering remote accessibility, automated inventory tracking, transparent transaction records, convenient order management, thereby enhancing operational efficiency and customer satisfaction.

System Modules (Solution Overview):

1. **Customer profile Management:**

   Entity: StoreCustomer

   - Features: Enables customers to view their profile, and manage personal details.

2. **Order Management:**

   Entity: Medicine_Order, OrderItem

   - Features: Allows customers to browse available medicines, place orders, and view order history.

3. **Medicine Information:**

   Entity: Medicine

   - Features: Manages information about available medicines and displays medicine details, such as name, description, dosage, price, and availability, for customers to browse.

4. **Admin Dashboard:**

Entity: Admin

- Features: Offers tools for adding, deleting, updating medicines, restocking inventory, and managing overall system functionality.

5. **Authentication and Authorization:**

Entity: User_Credential

- Features: Manages user authentication through login/signup mechanisms and provides role-based access control for customers and admins.

6. Purchase Receipt Generation:

Entity: Order_Receipt

- Features: Automatically triggers the creation of a detailed purchase receipt upon successful order completion. It includes order details such as purchased items, quantities, prices, total cost, customer information, and a transaction timestamp.

Task 2: Creating DB for your project

Q1:

- Create tables w.r.t identified entities.
- Assign them domains.
- Apply integrity constraints along with null and not null constraints on all tables to ensure that queries can be performed on them.

Worksheet | Query Builder

```
-- Creating Customer Table
CREATE TABLE StoreCustomer (
    CustomerID NUMBER(10) PRIMARY KEY,
    CustmFName VARCHAR2(30) NOT NULL,
    CONSTRAINT CHK_CustmFName CHECK (REGEXP_LIKE(CustmFName, '^[a-zA-Z]+$')),
    CustmLName VARCHAR2(30) ,
    CONSTRAINT CHK_CustmLName CHECK (REGEXP_LIKE(CustmLName, '^[a-zA-Z]+$')),
    Email VARCHAR2(100),
```

Script Output

Task completed in 0.152 seconds

```
Table STORECUSTOMER created.

Table MEDICINE_ORDER created.

Table MEDICINE created.

Table ORDERITEM created.

Table ADMIN created.

Table USER_CREDENTIAL created.

Table ORDER_RECEIPT created.
```

Q2: Create meaningful queries in English and then apply those using Database concepts in Oracle:

a) Display all attributes from the StoreCustomer table:

```
SELECT * FROM StoreCustomer;
```

Script Output | Task completed in 0.03 seconds

```
CUSTOMERID CUSTMFNAME                    CUSTMLNAME                    EMAIL
---------- ------------------------------ ------------------------------ ------------------------------------
         1 John                           Doe                            john@example.com
         2 Alice                          Smith                          alice@example.com
         3 Bob                            Johnson                        bob@example.com
```

b) Display concatenated column based on first and last name in StoreCustomer table:

```
SELECT CustmFName || ' ' || CustmLName AS FullName FROM StoreCustomer;
```

Script Output | Query Result | SQL | All Rows Fetched: 3 in 0.006 seconds

	FULLNAME
1	John Doe
2	Alice Smith
3	Bob Johnson

c) Apply concepts of AND, OR, and BETWEEN on Medicine table:

"Retrieve all medicines priced between $10 and $50."

Query Result | Query Result 1 | Query Result 2 | SQL | All Rows Fetched: 1 in 0.002 seconds

	MEDICINEID	MEDNAME	DESCRIPTION	DOSAGE	PRICE	INSTOCK	BATCHESINSTORE
1	201	Medicine A	Pain Reliever	1 pill per day	10.99	Yes	20

"Get medicines that are currently in stock and priced above $20."

Query Result | Query Result 1 | Query Result 2 | SQL | All Rows Fetched: 0 in 0.003 seconds

MEDICINEID	MEDNAME	DESCRIPT...	DOSAGE	PRICE	INSTOCK	BATCHESI...

"Retrieve medicines that are either in stock or have more than 5 batches in the store."

	MEDICINEID	MEDNAME	DESCRIPTION	DOSAGE	PRICE	INSTOCK	BATCHESINSTORE
1	201	Medicine A	Pain Reliever	1 pill per day	10.99	Yes	20
2	202	Medicine B	Cold and Flu	2 pills per day	8.5	Yes	15

d) Apply concepts of Group by to calculate total quantity of each medicine ordered:

"Calculate the total quantity of medicines ordered by customers for each medicineID.''

Worksheet | Query Builder

```
SELECT MedicineID, SUM(Quantity) AS TotalQuantityOrdered
FROM OrderItem
GROUP BY MedicineID;
```

Script Output

Task completed in 0.038 seconds

```
MEDICINEID TOTALQUANTITYORDERED
---------- --------------------
       201                    3
       202                    2
       203                    5
```

e) Apply concepts of & and &&

"Give order details of those customers whose id is given by the user at runtime."

```
SELECT * FROM Medicine_Order WHERE CustomerID = &cust_id AND OrderDate = '&order_date';
SELECT * FROM Medicine_Order WHERE CustomerID = &cust_id;
```

Script Output

Task completed in 17.899 seconds

```
old:SELECT * FROM Medicine_Order WHERE CustomerID = &cust_id AND OrderDate = '&order_date'
new:SELECT * FROM Medicine_Order WHERE CustomerID = 200 AND OrderDate = '2022/1/2'
no rows selected
old:SELECT * FROM Medicine_Order WHERE CustomerID = &cust_id
new:SELECT * FROM Medicine_Order WHERE CustomerID = 202
no rows selected
```

"Give order details of all those customers whose Id is 100 or they ordered on a specific date.

```
SELECT * FROM Medicine_Order WHERE CustomerID = 100 OR OrderDate = TO_DATE('&&order_date','YYYY-MM-DD');
SELECT * FROM Medicine_Order WHERE OrderDate = TO_DATE('&&order_date','YYYY-MM-DD');
```

Script Output

Task completed in 0.065 seconds

```
old:SELECT * FROM Medicine_Order WHERE CustomerID = 100 OR OrderDate = TO_DATE('&&order_date','YYYY-MM-DD')
new:SELECT * FROM Medicine_Order WHERE CustomerID = 100 OR OrderDate = TO_DATE('2020/1/3','YYYY-MM-DD')
no rows selected
old:SELECT * FROM Medicine_Order WHERE OrderDate = TO_DATE('&&order_date','YYYY-MM-DD')
new:SELECT * FROM Medicine_Order WHERE OrderDate = TO_DATE('2020/1/3','YYYY-MM-DD')
no rows selected
```

f) Apply concepts of UPDATE and ALTER TABLE:

"Update the 'Description' of the medicine with 'MedicineID' 201 to 'New Description' in the Medicine table."

"Add a new column named 'Age' of type NUMBER (2) to the StoreCustomer table."

```
UPDATE Medicine SET Description = 'New Description' WHERE MedicineID = 201;

ALTER TABLE StoreCustomer ADD Age NUMBER(2);
```

Script Output

Task completed in 0.087 seconds

```
1 row updated.


Error starting at line : 5 in command -
ALTER TABLE StoreCustomer ADD Age NUMBER(2)
Error report -
ORA-01430: column being added already exists in table
01430. 00000 -  "column being added already exists in table"
```

g) Apply various conditions using single table aggregate functions:

"Calculate the total count of medicines currently in stock, filtering from the Medicine table where the 'Instock' status is 'Yes'."

"Retrieve the highest price among all medicines available in the Medicine table and display it as 'MaxMedicinePrice'."

```
SELECT COUNT(*) AS TotalMedicinesInStock FROM Medicine WHERE Instock = 'Yes';
SELECT MAX(Price) AS MaxMedicinePrice FROM Medicine;
```

Script Output

Task completed in 0.041 seconds

```
TOTALMEDICINESINSTOCK
---------------------
                    2


MAXMEDICINEPRICE
----------------
           10.99
```

h) Update values into one particular row:

"Update the 'Address' of the customer with 'CustomerID' 200 to 'New Address'.

```
UPDATE StoreCustomer SET Address = 'New Address' WHERE CustomerID = 200;
```

Script Output

Task completed in 0.051 seconds

```
0 rows updated.
```

Task 3: lab mid-term.

Part1: Design and Develop ERD diagram:

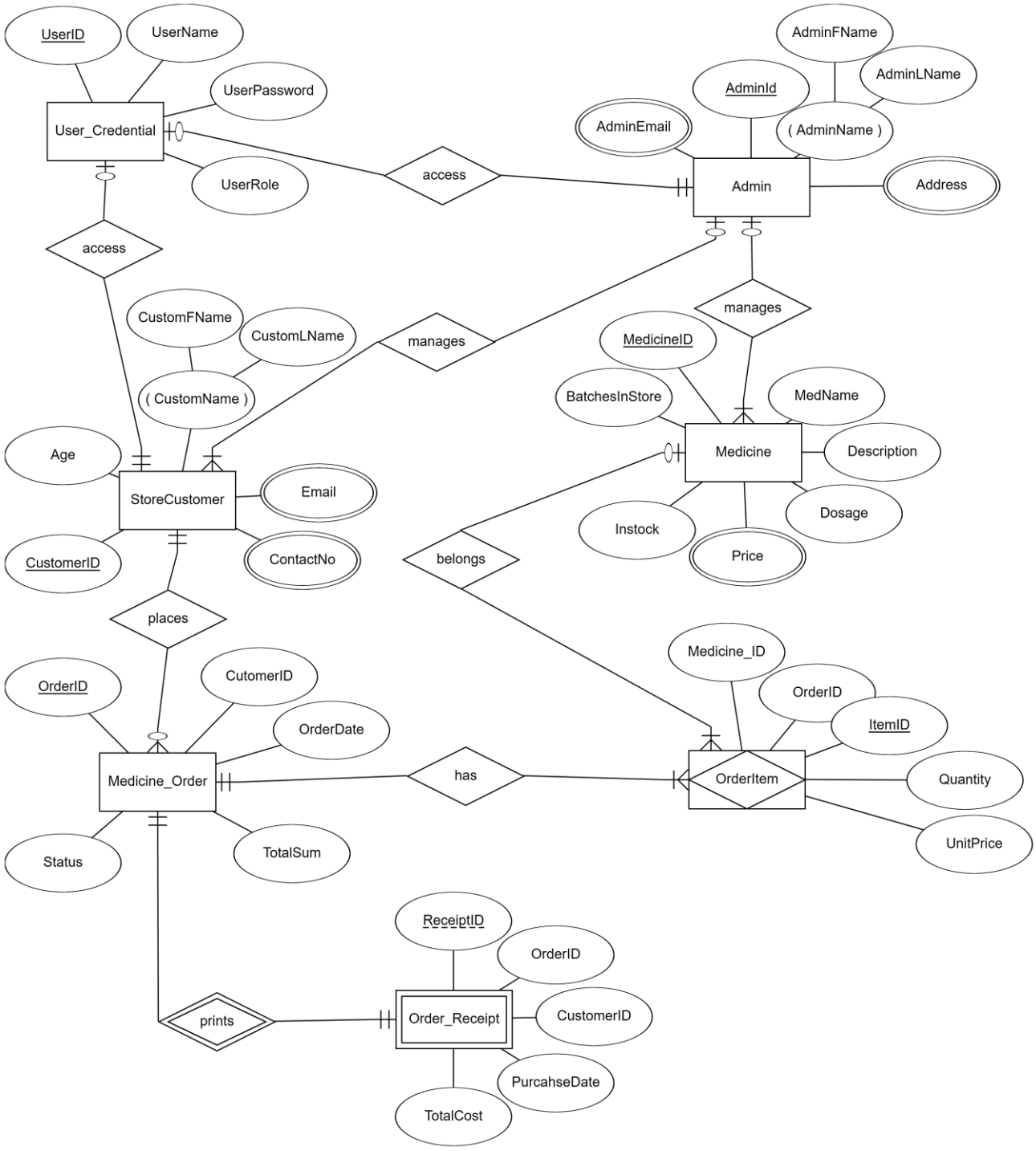


Figure 1. ERD of pharmacy store management system

Part2: Export the ddl script:

```
-- Creating Customer Table
CREATE TABLE StoreCustomer (
   CustomerID NUMBER(10) PRIMARY KEY,
   CustmFName VARCHAR2(30) NOT NULL,
   CONSTRAINT CHK_CustmFName CHECK (REGEXP_LIKE(CustmFName, '^[a-zA-Z]+$')),
   CustmLName VARCHAR2(30) ,
   CONSTRAINT CHK_CustmLName CHECK (REGEXP_LIKE(CustmLName, '^[a-zA-Z]+$')),
   Email VARCHAR2(100),
   Address VARCHAR2(255),
   ContactNo VARCHAR2(11) NOT NULL,
   CONSTRAINT CHK_ContactNo_Length CHECK (LENGTH(TRIM(ContactNo)) = 11)
);
-- Creating Order Table
CREATE TABLE Medicine_Order (
   OrderID NUMBER(10) PRIMARY KEY,
   CustomerID NUMBER(10) REFERENCES StoreCustomer(CustomerID),
   OrderDate DATE NOT NULL,
   TotalSum NUMBER(10, 2)NOT NULL,
   Status VARCHAR2(30)
);
-- Creating Medicine Table
CREATE TABLE Medicine (
```

```
    MedicineID NUMBER(10) PRIMARY KEY,
    MedName VARCHAR2(100) NOT NULL,
    Description VARCHAR2(255),
    Dosage VARCHAR2(50),
    Price NUMBER(10, 2) NOT NULL,
    Instock VARCHAR2(3),
    BatchesInStore VARCHAR2(3) NOT NULL
);
-- Creating OrderItem Table
CREATE TABLE OrderItem (
    ItemID NUMBER(10) PRIMARY KEY,
    OrderID NUMBER(10) REFERENCES Medicine_Order(OrderID),
    MedicineID NUMBER(10) REFERENCES Medicine(MedicineID),
    Quantity NUMBER(5) NOT NULL,
    UnitPrice NUMBER(10, 2) NOT NULL
);
-- Creating Admin Table
CREATE TABLE Admin (
    AdminID NUMBER(10) PRIMARY KEY,
    AdminFName VARCHAR2(30) NOT NULL,
    CONSTRAINT CHK_AdminFName CHECK (REGEXP_LIKE(AdminFName, '^[a-zA-Z]+$')),
    AdminLName VARCHAR2(30) ,
    CONSTRAINT CHK_AdminLName CHECK (REGEXP_LIKE(AdminLName, '^[a-zA-Z]+$')),
```

```
    AdminEmail VARCHAR2(100)
);
-- Creating UserCredential Table
CREATE TABLE User_Credential (
    UserID NUMBER(10) PRIMARY KEY,
    Username VARCHAR2(50) NOT NULL,
    UserPassword VARCHAR2(15) NOT NULL,
    CONSTRAINT CHK_UserPassword CHECK (REGEXP_LIKE(UserPassword, '^[a-zA-Z0-9]{8,}$')),
    UserRole VARCHAR2(30) NOT NULL
);
-- Creating PurchaseReceipt Table
CREATE TABLE Order_Receipt (
    ReceiptID NUMBER(10) NOT NULL,
    OrderID NUMBER(10) REFERENCES Medicine_Order(OrderID),
    CustomerID NUMBER(10) REFERENCES StoreCustomer(CustomerID),
    PurchaseDate DATE NOT NULL,
    TotalCost NUMBER(10, 2) NOT NULL
);
```

Part3: Implement the EER constraints 1-1, 1-many, many to many, “U”, “O”, “d”, “Partial participation”, “Total participation” , multi-attribute, super-class, sub-class, multiple-inheritance.

Entity1	**Relationship**	**cardinality**	**Entity2**
Admin	manages	0-many	StoreCustomer
Admin	access	1-1	User_Credential

Admin	manages	0-many	Medicine
StoreCustomer	places	0-many	Medicine_Order
StoreCustomer	access	1-1	User_Credential
Medicine	belongs	1-many	OrderItem
Medicine_Order	has	1-many	OrderItem
Medicine_Order	prints	1-1	Order_Receipt

- AdminFName and AdminLName are part of composite attribute AdminName.
- CustomFName and CustomLName are part of composite attribute StoreCustomerName.
- Email, ContactNo, Address are multivalued attributes in StoreCustomer relation.
- AdminEmail is multivalued attribute in Admin relation.

Part4: Insert few meaningful tupules in the resultant relations.

Tupules are already inserted at the time of table creation in Task2.

Part 5: Now in this part, you are going to first create a meaningful query in English and then in SQL for each of the following situations:

- **Need Requiring Joining at Least 3 Tables:**

  English: Find the details of customers who have placed orders along with the medicines they ordered.

  Query:

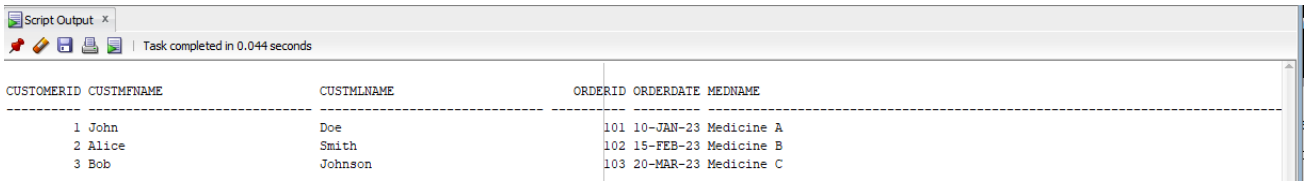
  SELECT StoreCustomer.CustomerID, StoreCustomer.CUSTMFNAME, StoreCustomer.CUSTMLNAME, Medicine_Order.OrderID, Medicine_Order.OrderDate, Medicine.MedName

  FROM StoreCustomer

  JOIN Medicine_Order ON StoreCustomer.CustomerID = Medicine_Order.CustomerID

  JOIN OrderItem ON Medicine_Order.OrderID = OrderItem.OrderID

  JOIN Medicine ON OrderItem.MedicineID = Medicine.MedicineID;



Script Output

Task completed in 0.044 seconds

CUSTOMERID	CUSTMFNAME	CUSTMLNAME	ORDERID	ORDERDATE	MEDNAME
1	John	Doe	101	10-JAN-23	Medicine A
2	Alice	Smith	102	15-FEB-23	Medicine B
3	Bob	Johnson	103	20-MAR-23	Medicine C

- **A need that will require the use of join with grouping and aggregation.**

English: Give a list of medicines and their total quantities sold in descending order based on sales quantity.

Query:

SELECT Medicine.MedName, COUNT (OrderItem.MedicineID) AS TotalSold

FROM Medicine

JOIN OrderItem ON Medicine.MedicineID = OrderItem.MedicineID

GROUP BY Medicine.MedName

ORDER BY TotalSold DESC;

Script Output

Task completed in 0.058 seconds

MEDNAME	TOTALSOLD
Medicine A	2
Medicine C	1

- **A need that involves the use of a sub-query**

English: Retrieve the orders made by customers with a specific email domain.

Query:

SELECT *

FROM Medicine_Order

WHERE CustomerID IN (SELECT CustomerID FROM StoreCustomer WHERE Email LIKE '%@example.com');

ORDERID	CUSTOMERID	ORDERDATE	TOTALSUM	STATUS
101	1	10-JAN-23	150.5	Pending
102	2	15-FEB-23	75.25	Delivered
103	3	20-MAR-23	200	Shipped

- **A need that involves the use of any set operators**

English: Give a consolidated list of all individuals involved with the store.

Query:

SELECT CustmFName, CustmLName FROM StoreCustomer

UNION

SELECT AdminFName, AdminLName FROM Admin;

```
CUSTMFNAME                     CUSTMLNAME
------------------------------ ------------------------------
Admin                          One
Admin                          Three
Admin                          Two
Alice                          Smith
Bob                            Johnson
John                           Doe

6 rows selected.
```