DBMS Project Report

PES University

Database Management Systems

UE18CS252

Submitted By

|  |  |
| --- | --- |
| PES2201800165 | Rakshith.R.Acharya |

|  |
| --- |
| A carStore Management system may be necessary for any automobile organisation to keep track of their customers, the offices they have and the products that are sold.The DataBase is about a car store(Showroom) that stores data about their customer, their employees, their products, orders and order details and the payments made by the customer.Here I have done an update before trigger and a few nested subqueries. |

[**Introduction 2**](https://docs.google.com/document/d/1mnFQvdeQlHONiHlAEFlqMJaPJE3zsW7hlkFOARP7PFs/edit#heading=h.gwqiqapyxr0x)

[**Data Model 2**](https://docs.google.com/document/d/1mnFQvdeQlHONiHlAEFlqMJaPJE3zsW7hlkFOARP7PFs/edit#heading=h.p451stoz4u81)

[**FD and Normalization 2**](https://docs.google.com/document/d/1mnFQvdeQlHONiHlAEFlqMJaPJE3zsW7hlkFOARP7PFs/edit#heading=h.wc0v2611q54c)

[**DDL 3**](https://docs.google.com/document/d/1mnFQvdeQlHONiHlAEFlqMJaPJE3zsW7hlkFOARP7PFs/edit#heading=h.pn441yprtk2f)

[**Triggers 3**](https://docs.google.com/document/d/1mnFQvdeQlHONiHlAEFlqMJaPJE3zsW7hlkFOARP7PFs/edit#heading=h.a9bswny9ffj)

[**SQL Queries 3**](https://docs.google.com/document/d/1mnFQvdeQlHONiHlAEFlqMJaPJE3zsW7hlkFOARP7PFs/edit#heading=h.exvna2br67rz)

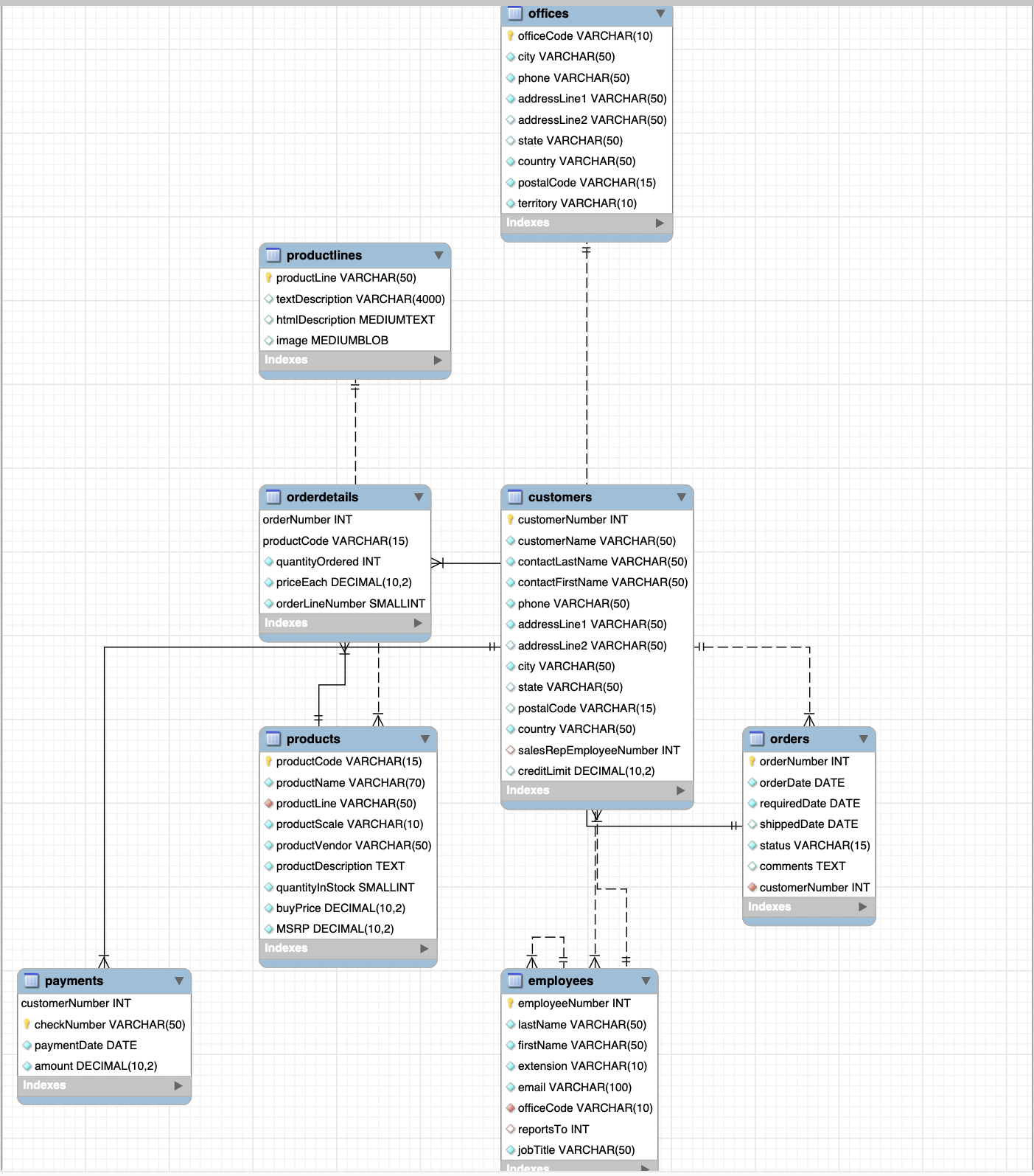
[**Conclusion 3**](https://docs.google.com/document/d/1mnFQvdeQlHONiHlAEFlqMJaPJE3zsW7hlkFOARP7PFs/edit#heading=h.k6bbxckcf51j)

Introduction

A carStore Management system may be necessary for any automobile organisation to keep track of their customers, the offices they have and the products that are sold.It provides a framework that helps you manage your resources efficiently and conveniently with controlled access to the inventory.The Sales of a product is meticulously done through the system which keeps track of sales in terms of the customer, the office that sold the product and the employee who sold the product.It also gives a detailed report of the products that are sold and the feedbacks of the previous customers.

Data Model

ER and Schema Diagram:



Relational Schema

offices(officeCode(PK), city, phone, addressLine1, addressLine2, state, country, postalCode, territory)

payments(customerNumber, checkNumber(PK), paymentDate, amount)

employees\_audit(id(PK), employeeNumber, lastname, changedat, action)

orderdetails(orderNumber, productCode(PK), quantityOrdered, priceEach, orderLineNumber)

productlines(productline(PK), textDescription, htmlDescription, image)

customers(customerNumber(PK), customerName, customerLastName, customerFirstName, phone, addressLine1, addressLine2, city, state, postalCode, country, salesRepEmployeeNumber(FK), creditLimit )

orders(orderNumber(PK), orderDate, requiredDate, shippedDate, status, comments, customerNumber(FK))

employees(employeeNumber(PK), lastName, firstName, extension, email, officeCode(FK), reportsTo, jobTitle)

products(productCode(PK), productName, productLine(FK), productScale, productVendor, productDescription, quantityInStock, buyPrice, MSRP)

FD and Normalization

Normal Form

1. **1NF**

There are 4 basics rules that a relation should follow to be in the 1st NF

* Each column should contain atomic values
* A column should contain values that are of the same data type
* Each column should have a unique name
* The order in which the data is saved does not matter

1. **2NF**

There are 2 basic rules that a relation should follow to be in the 2nd NF

* It should be in 1st NF
* It should not have a partial dependency

1. **3 NF**

There are 2 basic rules that a relation should follow to be in 3rd NF

* It should be in 2nd NF
* It should not have transitive dependency

Violation of the Normal Forms

* First Normal Form Violations

If orderNumber has two or more different order numbers entered in the same column, then it will violate 1NF.

In employeeNumber, if one of the values entered is a char, then it will violate 1NF.

If there are two columns with the same name, then it will violate 1NF.

* Second Normal Form Violations

In the order details table if productName was included, then the productName will have a partial dependency on productNumber therefore violating 2NF. (because the key in this table is orderNumber, productCode)

* Third Normal Form Violations

DDL

**CUSTOMER TABLE:**

CREATE TABLE `customers` ( `customerNumber` int(11) NOT NULL, `customerName` varchar(50) NOT NULL, `contactLastName` varchar(50) NOT NULL, `contactFirstName` varchar(50) NOT NULL, `phone` varchar(50) NOT NULL, `addressLine1` varchar(50) NOT NULL, `addressLine2` varchar(50) DEFAULT NULL, `city` varchar(50) NOT NULL, `state` varchar(50) DEFAULT NULL, `postalCode` varchar(15) DEFAULT NULL, `country` varchar(50) NOT NULL, `salesRepEmployeeNumber` int(11) DEFAULT NULL, `creditLimit` decimal(10,2) DEFAULT NULL, PRIMARY KEY (`customerNumber`), KEY `salesRepEmployeeNumber` (`salesRepEmployeeNumber`), CONSTRAINT `customers\_ibfk\_1` FOREIGN KEY (`salesRepEmployeeNumber`) REFERENCES `employees` (`employeeNumber`));

**EMPLOYEE TABLE:**

CREATE TABLE `employees` ( `employeeNumber` int(11) NOT NULL, `lastName` varchar(50) NOT NULL, `firstName` varchar(50) NOT NULL, `extension` varchar(10) NOT NULL, `email` varchar(100) NOT NULL, `officeCode` varchar(10) NOT NULL, `reportsTo` int(11) DEFAULT NULL, `jobTitle` varchar(50) NOT NULL, PRIMARY KEY (`employeeNumber`), KEY `reportsTo` (`reportsTo`), KEY `officeCode` (`officeCode`), CONSTRAINT `employees\_ibfk\_1` FOREIGN KEY (`reportsTo`) REFERENCES `employees` (`employeeNumber`), CONSTRAINT `employees\_ibfk\_2` FOREIGN KEY (`officeCode`) REFERENCES `offices` (`officeCode`));

**OFFICE TABLE:**

CREATE TABLE `offices` ( `officeCode` varchar(10) NOT NULL, `city` varchar(50) NOT NULL, `phone` varchar(50) NOT NULL, `addressLine1` varchar(50) NOT NULL, `addressLine2` varchar(50) DEFAULT NULL, `state` varchar(50) DEFAULT NULL, `country` varchar(50) NOT NULL, `postalCode` varchar(15) NOT NULL, `territory` varchar(10) NOT NULL, PRIMARY KEY (`officeCode`));

**ORDER DETAILS TABLE:**

CREATE TABLE `orderdetails` ( `orderNumber` int(11) NOT NULL, `productCode` varchar(15) NOT NULL, `quantityOrdered` int(11) NOT NULL, `priceEach` decimal(10,2) NOT NULL, `orderLineNumber` smallint(6) NOT NULL, PRIMARY KEY (`orderNumber`,`productCode`), KEY `productCode` (`productCode`), CONSTRAINT `orderdetails\_ibfk\_1` FOREIGN KEY (`orderNumber`) REFERENCES `orders` (`orderNumber`), CONSTRAINT `orderdetails\_ibfk\_2` FOREIGN KEY (`productCode`) REFERENCES `products` (`productCode`));

**ORDERS TABLE:**

CREATE TABLE `orders` ( `orderNumber` int(11) NOT NULL, `orderDate` date NOT NULL, `requiredDate` date NOT NULL, `shippedDate` date DEFAULT NULL, `status` varchar(15) NOT NULL, `comments` text, `customerNumber` int(11) NOT NULL, PRIMARY KEY (`orderNumber`), KEY `customerNumber` (`customerNumber`), CONSTRAINT `orders\_ibfk\_1` FOREIGN KEY (`customerNumber`) REFERENCES `customers` (`customerNumber`));

**PAYMENTS TABLE:**

CREATE TABLE `payments` ( `customerNumber` int(11) NOT NULL, `checkNumber` varchar(50) NOT NULL, `paymentDate` date NOT NULL, `amount` decimal(10,2) NOT NULL, PRIMARY KEY (`customerNumber`,`checkNumber`), CONSTRAINT `payments\_ibfk\_1` FOREIGN KEY (`customerNumber`) REFERENCES `customers` (`customerNumber`));

**PRODUCTLINES TABLE:**

CREATE TABLE `productlines` ( `productLine` varchar(50) NOT NULL, `textDescription` varchar(4000) DEFAULT NULL, `htmlDescription` mediumtext, `image` mediumblob, PRIMARY KEY (`productLine`));

**PRODUCTS TABLE:**

CREATE TABLE `products` ( `productCode` varchar(15) NOT NULL, `productName` varchar(70) NOT NULL, `productLine` varchar(50) NOT NULL, `productScale` varchar(10) NOT NULL, `productVendor` varchar(50) NOT NULL, `productDescription` text NOT NULL, `quantityInStock` smallint(6) NOT NULL, `buyPrice` decimal(10,2) NOT NULL, `MSRP` decimal(10,2) NOT NULL, PRIMARY KEY (`productCode`), KEY `productLine` (`productLine`), CONSTRAINT `products\_ibfk\_1` FOREIGN KEY (`productLine`) REFERENCES productlines` (`productLine`));

Triggers

Identify a constraint to implement as a trigger.

If you did not get any idea, check the notes and create a trigger for creating an audit-trail for insert/update/delete operations.

#sample table for showing triggers features

DROP TABLE IF EXISTS sales;

CREATE TABLE sales ( id INT AUTO\_INCREMENT, product VARCHAR(100) NOT NULL, quantity INT NOT NULL DEFAULT 0, fiscalYear SMALLINT NOT NULL, fiscalMonth TINYINT NOT NULL, CHECK(fiscalMonth >= 1 AND fiscalMonth <= 12), CHECK(fiscalYear BETWEEN 2000 and 2050),

CHECK (quantity >=0), UNIQUE(product, fiscalYear, fiscalMonth), PRIMARY KEY(id));

#Sample data for the sample table

INSERT INTO sales(product, quantity, fiscalYear, fiscalMonth)

VALUES ('2003 Harley-Davidson Eagle Drag Bike',120, 2020,1), ('1969 Corvair Monza', 150,2020,1),

('1970 Plymouth Hemi Cuda', 200,2020,1);

#creating a before update trigger

DELIMITER $$

CREATE TRIGGER before\_sales\_update

BEFORE UPDATE

ON sales FOR EACH ROW

BEGIN

DECLARE errorMessage VARCHAR(255);

SET errorMessage = CONCAT('The new quantity ',

NEW.quantity,

' cannot be 3 times greater than the current quantity ',

OLD.quantity);

IF new.quantity > old.quantity \* 3 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = errorMessage;

END IF;

END $$

DELIMITER ;

#Testing of the trigger

UPDATE sales SET quantity = 150 WHERE id = 1;

select \* from sales;

UPDATE sales SET quantity = 1540 WHERE id = 1;

SQL Queries

<Write a few english sentences and SQL queries for them. Ensure Advanced at least 2 correlated-nested and 2 aggregate queries. 1 or 2 outer join queries>

1)Correlated subquery to find customers who placed at least one sales order with the total value greater than 60000.

SELECT customerNumber, customerName

FROM customers

WHERE EXISTS( SELECT orderNumber, SUM(priceEach \* quantityOrdered)

FROM orderdetails INNER JOIN orders USING (orderNumber)

WHERE customerNumber = customers.customerNumber

GROUP BY orderNumber

HAVING SUM(priceEach \* quantityOrdered) > 60000);

2)Correlated subquery to select products whose buy prices are greater than the average buy price of all products in eachproduct line.

SELECT productname, buyprice

FROM products p1

WHERE buyprice > (SELECT AVG(buyprice)

FROM products

WHERE productline = p1.productline);

Conclusion

<

Write a few sentences about the capabilities of your system

Limitations and future enhancements

>