Café DBMS Project Report

PES University

Database Management Systems

UE18CS252

Submitted By

PES2201800675 Afzal Mukhtar

Café Database Management System is designed for a small café for running its business in a more orderly fashion. The DBMS will keep track of its employees, their pay and some personal detail. It will keep a track of its customers, so it becomes easier to link orders to each customer and also to send them any promotional messages and offers. The Food Menu is set by the café, which can be changed whenever the café decides to. Having this in the Database can reduce the effort required by the employees to log in the order placed by the customer. The Order relation is used to maintain all the orders from the start till the present day.

The Data Model is designed with four entities and related by three relationships. One of the relationship creates a separate relation table called Items, which keeps track of all the items placed by the customer. All the entities are strong entities, and one relationship(Created From) has its own attributes(Name and Quantity).

The Schema has been normalised to reduce redundancy and increase the space and time performance efficiency. Normalisation also helps in understanding the database easily, for it removed any unrelated data and keeps only related data, this makes a database clean and efficient. There are many functionally dependent characteristics in the schema, which help in understanding how each data is related to the other, like an order cannot be placed unless there's a customer and an employee taking the order, a customer cannot place an order for items that aren't available in the menu. Functional dependencies makes the database design clear to other users of the database.

The DDL is the Data Definition Language that is used to define the SQL Statements. The table creation statements have been shown with the constraints to know the exact details behind the creation of each table.

Triggers have been created to make sure constraints aren't violated. Like removing the item from menu will cause a foreign key violation in the list of items that have been ordered. So on deleting the item from the menu, the item is deleted from the orders list.

SQL queries have been stated to perform simple insertion, deletion and update operations. Some complex queries like creating a statistical graphable data from all the data to visualise the sale statistics. Finally the database can perform various complex task and simple task without an issue, and can further be improved by keeping a track of historical data.

Contents

Introduction

The mini world is a café which includes employee, customers, the orders a customer places and the items on the menu.

- ## Customer (customer):

  The café keeps track of its customers by giving them a customer ID and storing their name and phone number. The customer details is needed for keeping a track of their orders. The following are the details of their attributes:

  - ### Customer ID(c_id):

    The café gives each new customer a new customer ID which is stored in the database, so as to keep track of the person who placed the order and to help the system easily access his/her detail if they visit again.

Domain/Type	Default	Null Value	Unique	Attribute Type
Int Unsigned	none	Not allowed	Yes	Key Attribute

  - ### First Name (fname):

    The café enters the first name of the user to identify them by their name.

Domain/Type	Default	Null Value	Unique	Attribute Type
Character String	‘ ’	Not allowed	No	Simple

  - ### Last Name (lname):

    The café uses this to store the last name of the customer

Domain/Type	Default	Null Value	Unique	Attribute Type
Character String	none	Allowed	No	Simple

  - ### Phone (phone):

    The café takes the phone number of the customer sending offer SMSs or for mobile payment.

Domain/Type	Default	Null Value	Unique	Attribute Type
Character String	none	Allowed	No	Simple

- Employee (employee):

  The café keeps track of its employees by giving each new employee an employee ID, and recording their name and phone numbers, along with their resident address and enters the decided pay. The following are the details of their attributes:

  - Employee ID (e_id):

    The café records an Employee ID for every employee so as to differentiate them from the other employees.

Domain/Type	Default	Null Value	Unique	Attribute Type
Int Unsigned	none	Not allowed	Yes	Key Atrribute

  - First Name (fname):

    The café records the first name of the employee to identify them by their name.

Domain/Type	Default	Null Value	Unique	Attribute Type
Character String	‘ ’	Not allowed	No	Simple

  - Last Name (lname):

    The café records the last name of the employee

Domain/Type	Default	Null Value	Unique	Attribute Type
Character String	‘ ’	Allowed	No	Simple

  - Phone (phone):

    The café takes the phone number of the employee to pass on any important information.

Domain/Type	Default	Null Value	Unique	Attribute Type
Character String	none	Not allowed	No	Simple

  - Address (addr):

    The café records the employees residential address.

Domain/Type	Default	Null Value	Unique	Attribute Type
Text	none	Allowed	No	Simple

  - Pay (Pay):

    The decided pay of the employee is stored.

Domain/Type	Default	Null Value	Unique	Attribute Type
Float	none	Not allowed	No	Simple

- Menu (menu)

  The Menu keeps track of all items that are available for the customers to order from. It has Food ID and a name to identify the dish and its price.

  - Food ID (food_id):

    The food ID uniquely identifies the dish for the café.

Domain/Type	Default	Null Value	Unique	Attribute Type
Int Unsigned	none	Not allowed	Yes	Key Attribute

  - Name (food_name):

    This keeps track of the dish name, which the customers will use to place orders.

Domain/Type	Default	Null Value	Unique	Attribute Type
Text	none	Not allowed	No	Simple

  - Price (price):

    The price helps to keep track of the change in prices for the particular Food Item.

Domain/Type	Default	Null Value	Unique	Attribute Type
Float	none	Not allowed	No	Simple

- Order (orders)

  The order keeps track of all the orders placed by the customer and tracks which customer placed the order and by which employee the order was taken. It also keeps track of the date of the order.

  - Order ID (o_id):

    The order ID uniquely identifies one order from the other(Even from same customer).

Domain/Type	Default	Null Value	Unique	Attribute Type
Int Unsigned	none	Not allowed	Yes	Key Attribute

  - Date (order_date):

    It keeps track of the date and time the order was placed.

Domain/Type	Default	Null Value	Unique	Attribute Type
Date/Time	Current Date Time	Not allowed	No	Simple

- Customer ID (cus_id):

  It keeps track of which customer placed the order.

Domain/Type	Default	Null Value	Unique	Attribute Type
Int Unsigned	none	Not allowed	No	Key Attribute

- Employee ID (emp_id):

  Keeps track of which employee took the particular Order.

Domain/Type	Default	Null Value	Unique	Attribute Type
Int Unsigned	none	Not allowed	No	Key Attribute

- Items

Items is a relationship table that keeps track of the list of ordered items made by a customer.

- Item Name (item_name):

  The name of the dish the customer orders.

Domain/Type	Default	Null Value	Unique	Attribute Type
Text	none	Not allowed	No	Simple

- Quantity (quantity):

  The amount of each item the customer orders.

Domain/Type	Default	Null Value	Unique	Attribute Type
Int Unsigned	1	Not allowed	No	Simple

- Order ID (order_id):

  Keeps track of which order the item belongs to.

Domain/Type	Default	Null Value	Unique	Attribute Type
Int Unsinged	none	Not allowed	No	Simple

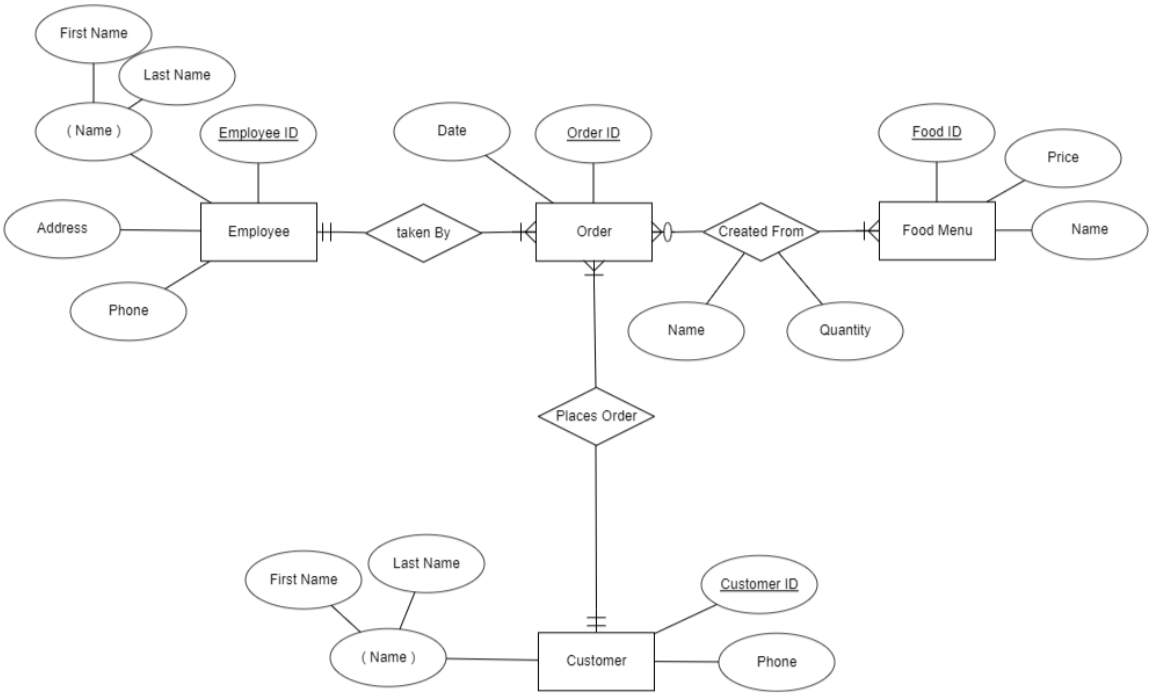
- Food ID (food_id):

  To map the item to the menu.

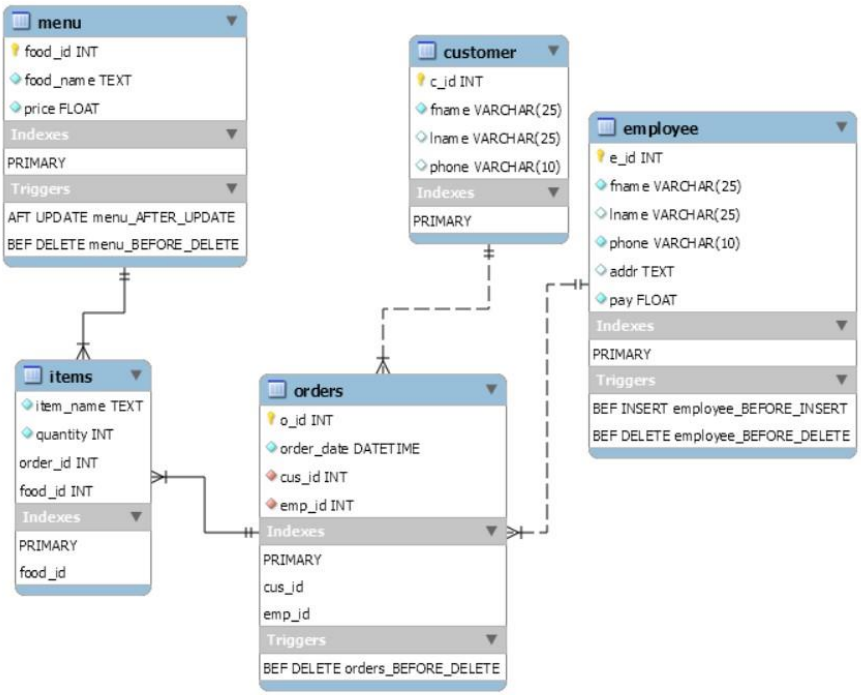
Domain/Type	Default	Null Value	Unique	Attribute Type
Int Unsigned	none	Not allowed	No	Simple

Data Model

- Entity Relationship Diagram:



- Schema Design:



- Keys in the Schema:
  - Customer

Candidate Keys	Primary Keys	Foreign Keys	Strong/Weak Entity
Customer ID	Customer ID	none	Strong

  - Employee

Candidate Keys	Primary Keys	Foreign Keys	Strong/Weak Entity
Employee ID	Employee ID	none	Strong

  - Order

Candidate Keys	Primary Keys	Foreign Keys	Strong/Weak Entity
Order ID	Order ID	Customer ID, Employee ID	Strong

  - Menu

Candidate Keys	Primary Keys	Foreign Keys	Strong/Weak Entity
Food ID	Food ID	none	Strong

  - Items

Candidate Keys	Primary Keys	Foreign Keys	Strong/Weak Entity
Order ID, Food ID	Order ID, Food ID (Composite PK)	Order ID, Food ID	Not an Entity, Reference Table

FD and Normalization

- Normal Forms:
  - First Normal Form:

    The schema needs to follow the following rules to be in 1NF:
    - ✓ Each column should have atomic values
    - ✓ A column should contain value of the same type
    - ✓ Each column name should be unique
    - ✓ The data can be stored in any order.
  - Second Normal Form:

    The schema needs to follow the following rules to be in 2NF:
    - ✓ It should be in 1NF
    - ✓ Should not have Partial Dependency

- Third Normal Form:

  The schema needs to follow the following rules to be in 3NF:
  - ✓ It should be in 2NF
  - ✓ Should not have Transitive Dependency

• Violation of the Normal Forms:

As all the relations are in Normal Forms, therefore they don't violate any rules of any of the normal forms. Anyway, these are few examples in which the normal forms may be violated.

- First Normal Form Violations:

  If the Customer or Employee has multiple phone numbers entered in the same column, then it will violate in 1NF.
  If a new column was inserted with the same name as Phone for additional phone numbers, then it can violate 1NF.
  In the Menu relation, if the Prices are filled with both float values and character values, then it will violate 1NF.
  In the Order relation, if the Date column is filled with numeric values instead of date-time values, then it will violate 1NF.
  In Items relation, if the quantity is filled with float values, instead of integer values, then it will violate the 1NF.

- Second Normal Form Violations:

  In Items relation if Customer ID was included, then the Customer ID will have a partial dependency with the Order ID, thereby violating 2NF.

- Third Normal Form Violations:

  In Employee, if Phone code was inserted, then it will have transitive dependency on the address. Thereby violating 3NF.

• Functional Dependencies:

The following are examples of a few functional dependencies:
- ✓ c_id → (fname, lname, phone)
- ✓ e_id → (fname, lname, phone, addr, pay)
- ✓ o_id → (cus_id, emp_id)
- ✓ (order_id, food_id) → (name, quantity)
- ✓ Food_id → (name, price)

DDL

- Create Customer Table

```
CREATE TABLE `customer` (
  `c_id` int unsigned NOT NULL AUTO_INCREMENT,
  `fname` varchar(25) NOT NULL DEFAULT '',
  `lname` varchar(25) DEFAULT NULL,
  `phone` varchar(10) DEFAULT NULL,
  PRIMARY KEY (`c_id`)
) ENGINE=InnoDB AUTO_INCREMENT=10001 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
```

- Create Employee Table

```
CREATE TABLE `employee` (
  `e_id` int unsigned NOT NULL AUTO_INCREMENT,
  `fname` varchar(25) NOT NULL DEFAULT '',
  `lname` varchar(25) DEFAULT '',
  `phone` varchar(10) NOT NULL,
  `addr` text,
  `pay` float NOT NULL,
  PRIMARY KEY (`e_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

- Create Items Table

```
CREATE TABLE `items` (
  `item_name` text NOT NULL,
  `quantity` int unsigned NOT NULL DEFAULT '1',
  `order_id` int unsigned NOT NULL,
  `food_id` int unsigned NOT NULL,
  PRIMARY KEY (`order_id`,`food_id`),
  KEY `food_id` (`food_id`),
  CONSTRAINT `items_ibfk_1` FOREIGN KEY (`order_id`) REFERENCES
`orders` (`o_id`),
  CONSTRAINT `items_ibfk_2` FOREIGN KEY (`food_id`) REFERENCES `menu`
(`food_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

- Create Menu Table

```
CREATE TABLE `menu` (
  `food_id` int unsigned NOT NULL AUTO_INCREMENT,
  `food_name` text NOT NULL,
  `price` float NOT NULL,
```

```
  PRIMARY KEY (`food_id`)
) ENGINE=InnoDB AUTO_INCREMENT=201 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
```

- Create Orders Table

```
CREATE TABLE `orders` (
  `o_id` int unsigned NOT NULL AUTO_INCREMENT,
  `order_date` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `cus_id` int unsigned NOT NULL,
  `emp_id` int unsigned NOT NULL,
  PRIMARY KEY (`o_id`),
  KEY `cus_id` (`cus_id`),
  KEY `emp_id` (`emp_id`),
  CONSTRAINT `orders_ibfk_1` FOREIGN KEY (`cus_id`) REFERENCES
`customer` (`c_id`),
  CONSTRAINT `orders_ibfk_2` FOREIGN KEY (`emp_id`) REFERENCES
`employee` (`e_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Triggers

- Before Inserting Into Employee (PK and FK constraint)

```
DROP TRIGGER IF EXISTS `cafe`.`employee_BEFORE_INSERT`;
DELIMITER $$
USE `cafe`$$
CREATE DEFINER = CURRENT_USER TRIGGER `cafe`.`employee_BEFORE_INSERT`
BEFORE INSERT ON `employee` FOR EACH ROW
BEGIN
DECLARE MSG varchar(255);
if(exists(select * from employee where fname = new.fname and lname =
new.lname and phone = new.phone)) then
   set MSG = 'Employee Already Exists';
    SIGNAL SQLSTATE '42000' SET MESSAGE_TEXT = msg;
end if;
END$$
DELIMITER ;
```

- Before Deleting an Employee (PK and FK constraint)

```
DROP TRIGGER IF EXISTS `cafe`.`employee_BEFORE_DELETE`;
DELIMITER $$
USE `cafe`$$
CREATE DEFINER = CURRENT_USER TRIGGER `cafe`.`employee_BEFORE_DELETE`
BEFORE DELETE ON `employee` FOR EACH ROW
```

```
BEGIN
if(exists(select emp_id from orders where emp_id = old.e_id)) then
   delete from orders where emp_id = old.e_id;
end if;
END$$
DELIMITER ;
```

- After Update on Menu (To Rectify Unrelated Data)

```
DROP TRIGGER IF EXISTS `cafe`.`menu_AFTER_UPDATE`;
DELIMITER $$
USE `cafe`$$
CREATE DEFINER = CURRENT_USER TRIGGER `cafe`.`menu_AFTER_UPDATE` AFTER
UPDATE ON `menu` FOR EACH ROW
BEGIN
if(exists(select items.item_name from items where items.item_name =
old.food_name)) then
   update items set items.item_name = new.food_name where
items.item_name = old.food_name;
end if;
END$$
DELIMITER ;
```

- Before Deleting from Menu (PK and FK Constraint)

```
DROP TRIGGER IF EXISTS `cafe`.`menu_BEFORE_DELETE`;
DELIMITER $$
USE `cafe`$$
CREATE DEFINER=`root`@`localhost` TRIGGER `menu_BEFORE_DELETE` BEFORE
DELETE ON `menu` FOR EACH ROW BEGIN
if(exists(select items.food_id from items where items.food_id =
old.food_id)) then
        delete from items where items.food_id = old.food_id;
end if;
END$$
DELIMITER ;
```

- Before Deleting From Orders (PK and FK Constraint)

```
DROP TRIGGER IF EXISTS `cafe`.`orders_BEFORE_DELETE`;
DELIMITER $$
USE `cafe`$$
CREATE DEFINER = CURRENT_USER TRIGGER `cafe`.`orders_BEFORE_DELETE`
BEFORE DELETE ON `orders` FOR EACH ROW
BEGIN
if(exists(select o_id from orders where o_id = old.o_id)) then
```

```
    delete from items where items.order_id = old.o_id;
end if;
END$$
DELIMITER ;
```

SQL Queries

- Menu Related Queries
  - Add to Menu

    ```
    insert into menu(food_name, price) values('Popcorn Chicken', 100);
    ```

  - Delete From Menu

    ```
    delete from menu where food_id = 201;
    ```

  - Update Menu

    ```
    update menu set food_name = "Garlic Cheese Toast" where food_id = 221;
    ```

- Customer Related Queries
  - Add to Customer

    ```
    insert into customer(fname, lname, phone) values('Maksymilian', 'Rhea',
    '9729346269');
    ```

  - Update Customer

    ```
    update customer set fname = 'Sherlock', lname = 'Holmes', phone =
    '9822120002' where c_id = 10001;
    ```

- Employee Related Queries
  - Add to Employee

    ```
    insert into employee(fname, lname, phone, addr, pay) values('Rhea', 'Ware',
    '8200353144', '174 Woodland St. North Olmsted, OH 44070', '4800');
    ```

  - Update Employee

    ```
    update employee set fname = 'Spencer', lname = 'Hastings', phone =
    '9029102123', addr = ''174 Woodland St. North Olmsted ', pay = 4000;
    ```

  - Delete Employee

    ```
    delete from employee where e_id = 52;
    ```

- Other Queries
  - Add Orders and Items Ordered

```
insert into orders(cus_id, emp_id, order_date) values(10021, 41, '2020-03-17 03:39:48.993222');
insert into items(item_name, quantity, order_id, food_id) values("Popcorn Chicken", 4, 3, 213);
```

  - Get Sales Detail for a Particular Item (For Specific Item Sale Statistics)

```
SELECT
    quantity, order_date
FROM
    (SELECT
        COALESCE(SUM(quantity), 0) AS quantity, order_date, food_id
    FROM
        (SELECT
        *
    FROM
        (SELECT
        quantity, DATE(order_date) AS order_date, food_id
    FROM
        items
    LEFT OUTER JOIN orders ON orders.o_id = items.order_id) AS temp
    WHERE
        DATEDIFF(temp.order_date, CURRENT_TIMESTAMP) <= 30) AS temp1
    GROUP BY food_id , order_date) AS temp2
WHERE
    food_id = 221;
```

  - Get Sales Detail for all the Items (For Item Sale Statistics)

```
SELECT
    COALESCE(SUM(quantity), 0),
    menu.food_name,
    menu.food_id,
    price
FROM
    items
        RIGHT JOIN
    menu ON menu.food_id = items.food_id
GROUP BY menu.food_id;
```

  - Get Monthly Sale Detail (For Overall Sale Statistics)

```
SELECT
    quantity, food_id
FROM
    (SELECT
        COALESCE(SUM(quantity), 0) AS quantity, order_date, food_id
    FROM
        (SELECT
        *
    FROM
        (SELECT
        quantity, DATE(order_date) AS order_date, food_id
```

```
        FROM
            items
        LEFT OUTER JOIN orders ON orders.o_id = items.order_id) AS temp
        WHERE
            DATEDIFF(temp.order_date, CURRENT_TIMESTAMP) <= 30) AS temp1
        GROUP BY food_id) AS temp2
    ORDER BY food_id;
```

Conclusion

The System is designed for a small business which can handle all the data related to the café. It provides various important features, like:

- ✓ Adding or Updating a Customer
- ✓ Modifying the menu to include new dishes or modify pricings or change names of the food.
- ✓ Adding, removing or updating Employee Information, like their address, name, phone number or pay.
- ✓ This Database also performs some of the complex tasks such as graphing the Sales Statistics for the various necessary information.

These functionalities is provided keeping in mind the data that will be required to place orders and to keep track of the sales, which can help the owner decide which item is successful or what the sales during certain periods of time.

Further this system can be improved by having an increased usage of historical data, where any data removed from the database will be saved in another table/database in order to keep track of the previously existing data. This can also help in having a better statistical view of the entire yearly sales based on each month.