DATABASE MANAGEMENT PROJECT

Restaurant Management System

Submitted by:

Niketan Nath (2016UCO1670)

Geetanshu Gupta (2016UCO1671)

Aayush Mahajan (2016UCO1672)

DIVISION OF COMPUTER SCIENCE ENGINEERING

NETAJI SUBHAS INSTITUTE OF TECHNOLOGY, DWARKA

CERTIFICATE OF COMPLETION

This is to certify that Niketan Nath (2016UCO1670), Geetanshu Gupta (2016UCO1671) and Aayush Mahajan(2016UCO1672), students of COE-III batch of 2020 have successfully completed the Database Management System Project on Restaurant Management System, under the guidance of Mrs. Sushma Nagpal (Subject Teacher) during the 3rd semester of the year 2017.

(Signature)

ACKNOWLEDGEMENT

It is a genuine pleasure to express our deep sense of thanks and gratitude to our teacher and guide Mrs Sushma Nagpal. Her dedication and keen interest above all her overwhelming attitude to help her students has been the reason for the completion of this project. Her timely advice, meticulous scrutiny, scholarly advice and able guidance has helped us to a very great extent to accomplish this task.
We would also like to extend our thanks to various online forums which helped us knit our project together. Without their constant help this project would not have been what it is right now.

Niketan Nath (2016UCO1670)
Geetanshu Gupta (2016UCO1671)
Aayush Mahajan (2016UCO1672)

TABLE OF CONTENTS

DATABASE MANAGEMENT PROJECT 1

- Certificate of Completion 2
- Acknowledgment 3
- Technologies Used 5
- Problem Statement 6
- ER Model 8
- Integrity Constraints 9
- Triggers to Maintain Integrity 9
- Mapping ER to Relational Schema 13
- DDL Code 18

TECHNOLOGIES USED

We have used several technologies in this Restaurant Management System project. And these are listed as follows:

1. JAVA (JDBC) - We used JAVA for the front-end portion of the project. JDBC code was used to link JAVA with MYSQL.
2. MYSQL - The main back-end was developed using the intuitive graphical interface of MYSQL. The DDL and DML portions were also made with the help of this software.
3. Webstorm Text Editor - In order to work as a team, we used the Webstorm Text Editor and its host of features to connect together snippets of code so it was easy to compile the front-end with the back-end.
4. ERDPLUS – It was used to create the ER model.

PROBLEM STATEMENT

Introduction to the problem statement

XYZ is a restaurant. It is divided into multiple tables. Customers come in, sit and place their orders. Waiters take the orders, and chefs prepare it. Finally, a bill is generated for each customer.

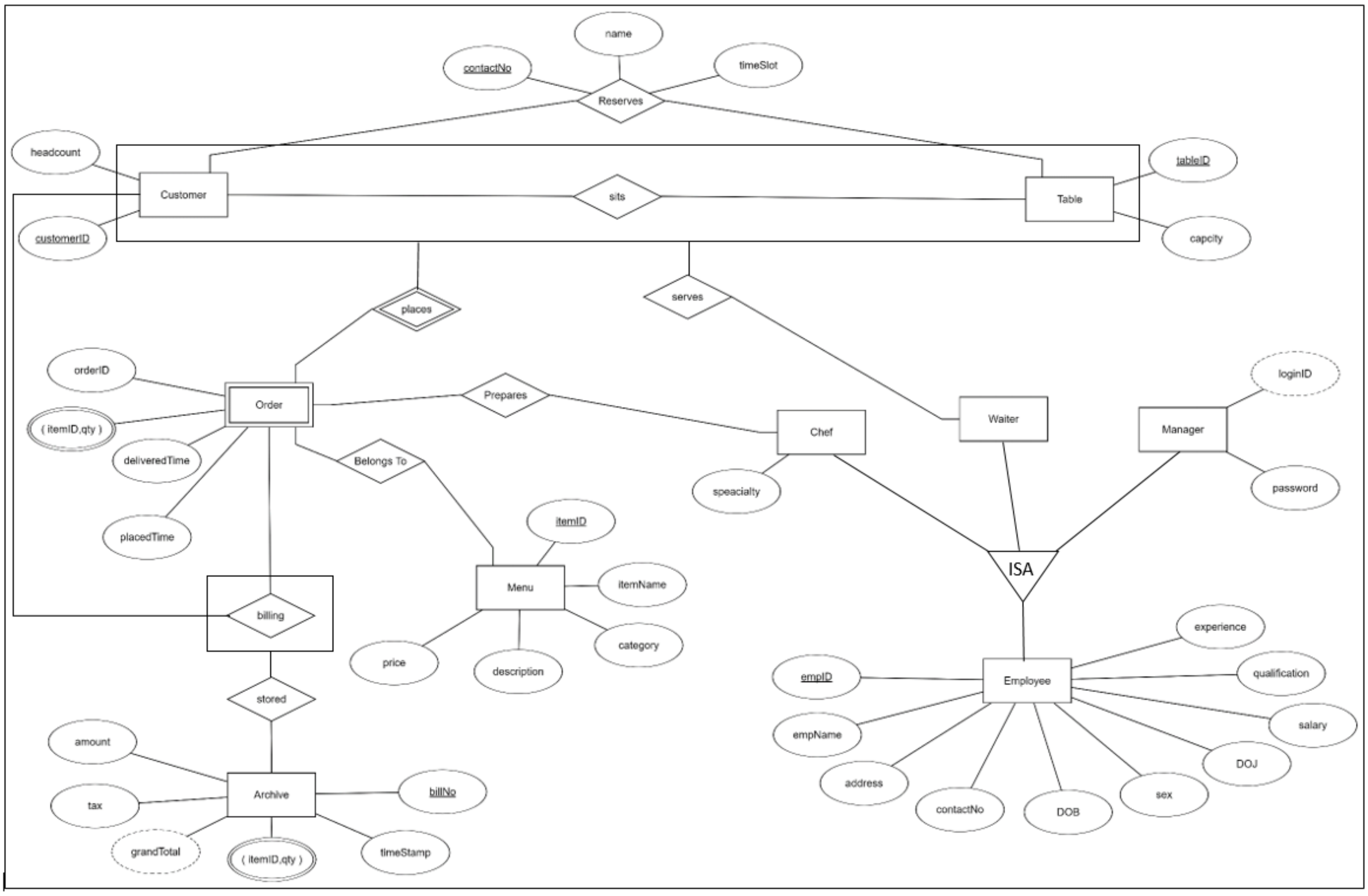
The aim of this case study is to design and develop a system for XYZ restaurant which streamlines its processes by providing a well-defined and easy to use communication channel between the kitchen and the seating area. The system keeps track of the orders placed by a table, the status of the order, etc. This helps in reducing the waiter effort, which minimizes errors, speeds up delivery times and reduces the man-hours needed for back office jobs.

Description

- When a customer arrives, a customerID (unique and serial in nature) is assigned to them. A party of people is considered a single customer (single billable entity).
- The system keeps a record of the capacity of each table.
- Managers are responsible for assigning table(s) to a customer and waiters to the occupied tables.
- Customers sit on table(s). The system keeps track of which tables are occupied by which customers. A table may be occupied by a maximum of one customer (party), but a customer may sit on multiple tables (depending upon the headcount). Additionally, a customer must be assigned to at least one table.
- Every occupied table is assigned exactly one waiter (assignment not a responsibility of the system). Waiter will take and deliver the orders.

- Order can be given from items in the menu. A waiter records the details of an order from a particular occupied table. Timestamp of the order is also recorded in the database.
- Multiple tables (allocated to a customer) can give orders any number of times and each such instance is recorded.
- The order is then sent to the kitchen, where it is prepared by chefs. An order is prepared by at least one chef. The system keeps track of the status of the order (if it is ready to be delivered or not). The system also keeps track of the time taken to prepare an order.
- Waiters take ready orders to their respective tables. Once the order is delivered to the occupied table, it can be marked as complete.
- After all orders are complete, if the customer wishes, they can ask any waiter assigned to them to generate the bill. All orders related to all tables occupied by that particular customer will be aggregated, and the relevant computations would be performed to form the bill.
- Once the bill is generated, it will be printed and stored in an archive. This archive would help in generation of relevant reports (monthly turnover, frequency of items ordered, tax filing, etc.).
- Furthermore, if a completed order is unsatisfactory, a manager may decide to replace or refund the order.
- After billing, manager takes feedback from the customer, which is optional.

ER MODEL



The final ER Model is as shown above in the diagram. The following entities were used in the ER Model.

1. Customer
2. Table
3. Order
4. Archive
5. Menu
6. Employee
7. Chef
8. Waiter
9. Manager

INTEGRITY CONSTRAINTS

Our project deals with various domain and referential integrity constraints that allow for accurate and consistent information in the database. For e.g., the constraints on the age of the Chef, Manager, Waiter are essential to ensure that invalid entries do not make way into the database. Another e.g. would be dealing with the Monthly Salaries of the Employees which should lie between the specified range. Similarly, the quantity of any item ordered by the customer or entered in the database must be valid.
Foreign Keys such as empID, orderID, tableID etc are also maintained at different tables to ensure referential integrity.

Triggers to maintain Integrity

1. **Checks if customer headCount is positive**

```
create trigger customer_headCount_insert
before insert on customer
for each row
begin
        if(new.headCount <= 0)  then
        signal sqlstate '45000'
        set message_text = 'Head Count must be positive';
        end if;
end;
$$

create trigger customer_headCount_update
before update on customer
for each row
begin
        if(new.headCount <= 0)  then
        signal sqlstate '45000'
        set message_text = 'Head Count must be positive';
        end if;
```

```
end;
$$
```

2. Checks if employee age is atleast 18, DOJ of employee is after DOB of employee, and gender of employee belongs to 'male', 'female' or 'other'

```
create trigger employee_insert
before insert on employee
for each row
begin
        if(year(now()) - year(new.DOB) < 18
                OR new.DOJ<new.DOB
                OR new.gender NOT IN ('male','female','other')) then
        signal sqlstate '45000'
        set message_text = 'Error with value entered.';
        end if;
end;
$$

create trigger employee_update
before update on employee
for each row
begin
        if(year(now()) - year(new.DOB) < 18
                OR new.DOJ<new.DOB
                OR new.gender NOT IN ('male','female','other')) then
        signal sqlstate '45000'
        set message_text = 'Error with value entered.';
        end if;
end;
$$
```

3. Checks if quantity of item entered in the order process is positive

```
create trigger item_quantity_insert
before insert on itemsinorder
for each row
begin
        if(new.qty <= 0)  then
        signal sqlstate '45000'
        set message_text = 'Quantity must be positive.';
        end if;
end;
$$

create trigger item_quantity_update
before update on itemsinorder
for each row
begin
        if(new.qty <= 0)  then
        signal sqlstate '45000'
        set message_text = 'Quantity must be positive.';
        end if;
end;
$$
```

4. Checks if price of item is positive

```
create trigger menu_price_insert
before insert on menu
for each row
begin
        if(new.price <= 0)  then
        signal sqlstate '45000'
        set message_text = 'Price must be positive.';
        end if;
end;
$$

create trigger menu_price_update
before update on menu
for each row
begin
        if(new.price <= 0)  then
        signal sqlstate '45000'
        set message_text = 'Price must be positive.';
        end if;
end;
$$
```

5. Checks if table capacity is positive

```
create trigger table_capacity_insert
before insert on table
for each row
begin
        if(new.capacity <= 0)  then
        signal sqlstate '45000'
        set message_text = 'Capacity of a table cannot be non positive.';
        end if;
end;
$$

create trigger table_capacity_update
before update on table
for each row
begin
        if(new.capacity <= 0)  then
        signal sqlstate '45000'
        set message_text = 'Capacity of a table cannot be non positive.';
        end if;
end;
$$
```

Mapping ER Model to Relational Schema

1. Table Name: Customer
   Primary Key: customerID
   Purpose : To store the customer details.

Column Name	**Data Type**	**Null Constraint**	**Key Constraint**
customerID	int(11)	Not Null	Primary Key
headcount	int(11)	Not Null	

2. Table Name: Archive
   Primary Key: billNo
   Purpose : To store the bill details.

Column Name	**Data Type**	**Null Constraint**	**Key Constraint**
billNo	int(11)	Not Null	Primary Key
timeStamp	int(11)	Not Null	
amount	decimal(10,2)	Not Null	
tax	decimal(10,2)	Not Null	
grandTotal	decimal(10,2)	Not Null	

3. Table Name: Menu
   Primary Key: itemID
   Purpose : To store the details of different dishes and beverages.

Column Name	**Data Type**	**Null Constraint**	**Key Constraint**
itemID	varchar(20)	Not Null	Primary Key
itemName	varchar(100)	Not Null	
category	varchar(30)	Default Null	
description	varchar(200)	Default Null	
price	int(11)	Not Null	

4. Table Name: Serves
   Primary Key: empID, tableID
   Purpose : To store which table is served by which employee.

Column Name	**Data Type**	**Null Constraint**	**Key Constraint**
empID	varchar(20)	Not Null	Primary Key
tableID	int(11)	Not Null	Primary Key

5. Table Name: Chef
   Primary Key: empID
   Purpose : To store specialization of each chef working.

Column Name	**Data Type**	**Null Constraint**	**Key Constraint**
empID	varchar(20)	Not Null	Primary Key
specialization	varchar(45)	Default Null	

6. Table Name: Manager
   Primary Key: empID
   Purpose : To store login credentials of each manager.

Column Name	**Data Type**	**Null Constraint**	**Key Constraint**
empID	varchar(20)	Not Null	Primary Key
password	varchar(45)	Default Null	

7. Table Name: ItemsInOrder
   Primary Key: orderID, itemID
   Purpose : To store items ordered by a customer.

Column Name	**Data Type**	**Null Constraint**	**Key Constraint**
orderID	int(11)	Not Null	Primary Key
itemID	varchar(20)	Not Null	Primary Key
qty	int(11)	Not Null	

8. Table Name: ItemsInBill
   Primary Key: billNo, itemID
   Purpose : To store items to be included in the bill.

Column Name	**Data Type**	**Null Constraint**	**Key Constraint**
billNo	int(11)	Not Null	Primary Key
itemID	varchar(20)	Not Null	Primary Key
qty	int(11)	Not Null	

9. Table Name: Employee

Primary Key: empID

Purpose : To store the details of all employees.

Column Name	Data Type	Null Constraint	Key Constraint
empID	varchar(20)	Not Null	Primary Key
empName	varchar(45)	Not Null	
address	varchar(200)	Default Null	
contactNo	varchar(20)	Not Null	
DOB	date	Default Null	
DOJ	date	Default Null	
gender	varchar(10)	Default Null	
salary	int(11)	Not Null	
qualification	varchar(45)	Default Null	
experience	varchar(45)	Default Null	

10. Table Name: Table

Primary Key: tableID

Purpose : To store the details of all tables.

Column Name	Data Type	Null Constraint	Key Constraint
tableID	int(11)	Not Null	Primary Key
capacity	int(11)	Not Null	
customerID	int(11)	Default Null	

11. Table Name: Order

Primary Key: orderID

Purpose : To store the details of all orders.

Column Name	**Data Type**	**Null Constraint**	**Key Constraint**
orderID	int(11)	Not Null	Primary Key
tableID	int(11)	Not Null	
placedTime	int(11)	Not Null	
deliveredTime	int(11)	Default Null	

DDL CODE

```
CREATE TABLE `customer`
 (
  `customerID` int(11) NOT NULL AUTO_INCREMENT,
  `headCount` int(11) NOT NULL,
  PRIMARY KEY (`customerID`)
)


CREATE TABLE `billarchive`
 (
  `billNo` int(11) NOT NULL,
  `timeStamp` int(11) NOT NULL,
  `amount` decimal(10,2) NOT NULL,
  `tax` decimal(10,2) NOT NULL,
  `grandTotal` decimal(10,2) NOT NULL,
  PRIMARY KEY (`billNo`),
  KEY `billArchiveToCustomer` (`billNo`),
  CONSTRAINT `billArchiveToCustomer` FOREIGN KEY (`billNo`)
REFERENCES `customer` (`customerID`)
)


CREATE TABLE `menu`
(
  `itemID` varchar(20) NOT NULL,
  `itemName` varchar(100) NOT NULL,
  `category` varchar(30) DEFAULT NULL,
  `description` varchar(200) DEFAULT NULL,
  `price` int(11) NOT NULL,
  PRIMARY KEY (`itemID`)
)


CREATE TABLE `serves`
(
  `empID` varchar(20) NOT NULL,
  `tableID` int(11) NOT NULL,
  PRIMARY KEY (`empID`,`tableID`),
  KEY `servesToEmployee` (`empID`),
  KEY `servesToTable` (`tableID`),
  CONSTRAINT `servesToEmployee` FOREIGN KEY (`empID`)
REFERENCES `employee` (`empID`)
  CONSTRAINT `servesToTable` FOREIGN KEY (`tableID`)
REFERENCES `table` (`tableID`)
)


CREATE TABLE `chef`
(
  `empID` varchar(20) NOT NULL,
  `specialization` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`empID`),
  KEY `empID` (`empID`),
  CONSTRAINT `chefFK` FOREIGN KEY (`empID`) REFERENCES
`employee` (`empID`)
)
```

```
CREATE TABLE `manager`
(
  `empID` varchar(20) NOT NULL,
  `password` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`empID`),
  KEY `empIDManager` (`empID`),
  CONSTRAINT `managerFK` FOREIGN KEY (`empID`) REFERENCES
`employee` (`empID`)
)


CREATE TABLE `itemsinorder`
 (
  `orderID` int(11) NOT NULL,
  `itemID` varchar(20) NOT NULL,
  `qty` int(11) NOT NULL,
  PRIMARY KEY (`orderID`,`itemID`),
  KEY `IIOToOrder` (`orderID`),
  KEY `IIOToItem` (`itemID`),
  CONSTRAINT `IIOToOrder` FOREIGN KEY (`orderID`) REFERENCES
`order` (`orderID`),
  CONSTRAINT `IIOToItem` FOREIGN KEY (`itemID`) REFERENCES
`menu` (`itemID`)
)


CREATE TABLE `itemsinbill`
```

```
(
  `billNo` int(11) NOT NULL,
  `itemID` varchar(20) NOT NULL,
  `qty` int(11) NOT NULL,
  PRIMARY KEY (`billNo`,`itemID`),
  KEY `IIBToBill` (`billNo`),
  KEY `IIBToItem` (`itemID`),
  CONSTRAINT `IIBToBill` FOREIGN KEY (`billNo`) REFERENCES
`billarchive` (`billNo`),
  CONSTRAINT `IIBToItem` FOREIGN KEY (`itemID`) REFERENCES
`menu` (`itemID`)
)


CREATE TABLE `employee`
(
  `empID` varchar(20) NOT NULL,
  `empName` varchar(45) NOT NULL,
  `address` varchar(200) DEFAULT NULL,
  `contactNo` varchar(20) NOT NULL,
  `DOB` date DEFAULT NULL,
  `DOJ` date DEFAULT NULL,
  `gender` varchar(10) DEFAULT NULL,
  `salary` int(11) NOT NULL,
  `qualification` varchar(45) DEFAULT NULL,
  `experience` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`empID`)
)
```

```
CREATE TABLE `table`
(
  `tableID` int(11) NOT NULL,
  `capacity` int(11) NOT NULL,
  `customerID` int(11) DEFAULT NULL,
  PRIMARY KEY (`tableID`),
  KEY `tableToCustomer` (`customerID`),
  CONSTRAINT `tableToCustomer` FOREIGN KEY (`customerID`)
REFERENCES `customer` (`customerID`)
)


CREATE TABLE `order`
 (
  `orderID` int(11) NOT NULL AUTO_INCREMENT,
  `tableID` int(11) NOT NULL,
  `placedTime` int(11) NOT NULL,
  `deliveredTime` int(11) DEFAULT NULL,
  PRIMARY KEY (`orderID`),
  KEY `orderToTable` (`tableID`),
  CONSTRAINT `orderToSits` FOREIGN KEY (`tableID`) REFERENCES
`table` (`tableID`)
)
```