CATERING SERVICE

DATABASE DESIGN PROJECT REPORT

MJ Padma Sri (jxm166230)

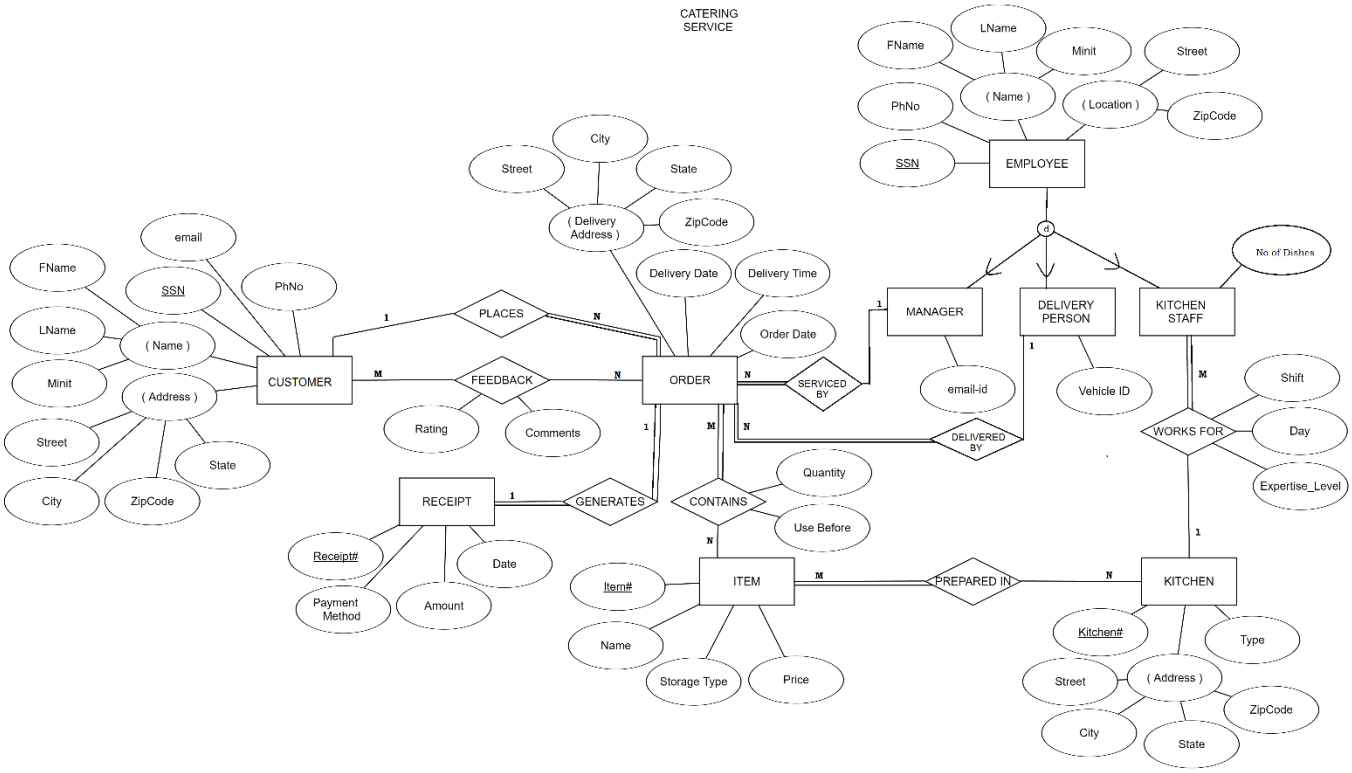
Vincy Shrine (vxc152130)

Contents

Requirements for Catering Service

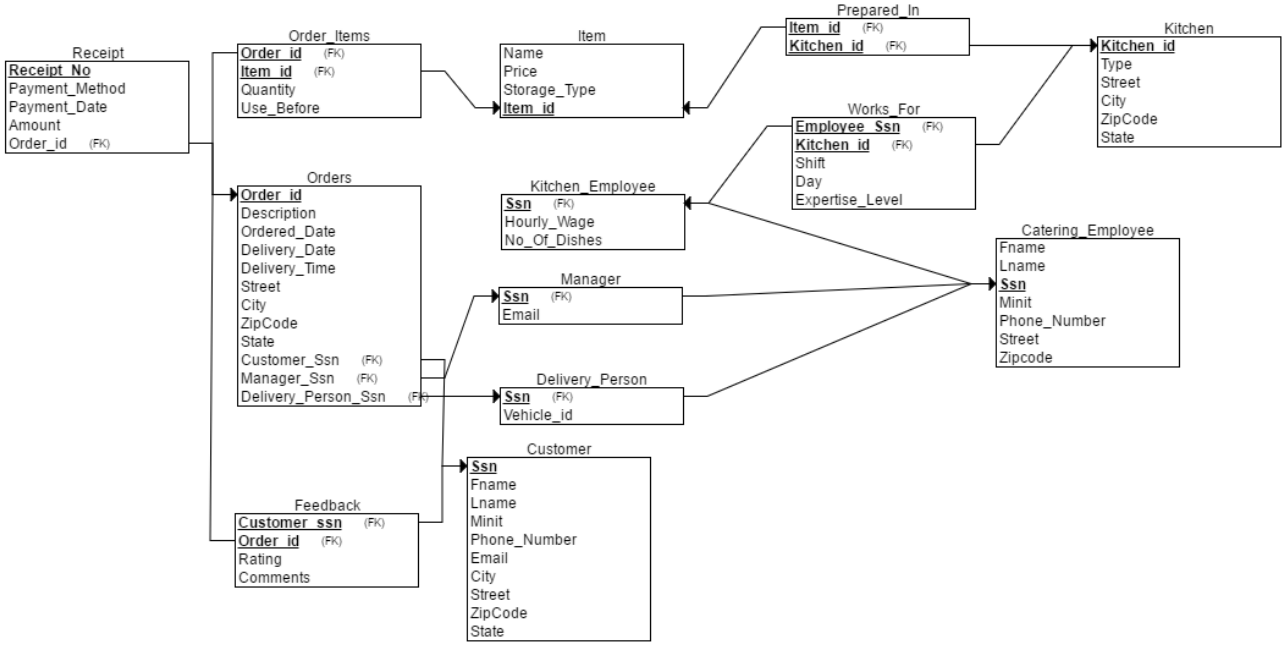
1) The system stores the details of every user who places a catering order. It should store SSN, name, email, phone number and address.
2) The system should keep track of all the orders placed by the users. Each order is identified uniquely by an order id. It should also contain information about order description, ordered date, requested delivery date, requested delivery time and delivery address.
3) The system should have track of all the items that have been placed as part of an order. Each item has a unique item id. An item should also have name, price, storage type and use before date.
4) The system should keep track of all its employees whose details are SSN, name, phone number and their location. Additionally, Manager should have an email id. Kitchen staff should have information about the number of dishes they can prepare, shift and day information for the kitchen they work for. Delivery Person should have vehicle id information.
5) The shifts that are assigned for a Kitchen staff is restricted to the values: morning, noon and night.
6) The system should keep track of all the kitchens where the items are prepared for the order. It should have details on the kitchen type and its address. Each kitchen is assigned a unique id.
7) The kitchens can be classified as a vegetarian or non-vegetarian type, based on the type of food they prepare.
8) The system should store information about the receipt generated for every order. It should contain details on amount, payment type, date of payment and a unique receipt number.
9) The system should store feedback from a user for a placed order, which should consist of rating and comments. The rating values are restricted from 1 to 5.
10) Each Order is assigned to a Manager who services it. Each Manager keeps track of all the items requested for an order and collects them from various kitchens and assigns a delivery person, who delivers to the user.

Enhanced Entity Relationship diagram



Mapping to Relational Model and Normalization

The mapping result from EER diagram to relational schema was already normalized. It was in 3NF. Below is the final relational schema.



SQL Statements

```
create table Catering_Employee(Ssn VARCHAR2(9),
Fname VARCHAR2(20),
Lname VARCHAR2(20),
Minit VARCHAR2(2),
Phone_Number VARCHAR2(10),
Street VARCHAR2(20),
ZipCode VARCHAR(10),
PRIMARY KEY(Ssn)
);

create table Manager(Ssn VARCHAR2(9),
Email VARCHAR2(20),
PRIMARY KEY(Ssn),
CONSTRAINT FK_Mgr_Ssn FOREIGN KEY (Ssn) references Catering_Employee(Ssn)on delete
cascade
);

create table Delivery_Person(Ssn VARCHAR2(9),
Vehicle_id VARCHAR2(10),
PRIMARY KEY(Ssn),
CONSTRAINT FK_DP_Ssn FOREIGN KEY (Ssn) references Catering_Employee(Ssn) on delete cascade
);

create table Customer(Ssn VARCHAR2(9),
Fname VARCHAR2(20),
Lname VARCHAR2(20),
Minit VARCHAR2(2),
Phone_Number VARCHAR2(10),
Email VARCHAR2(20),
City VARCHAR2(10),
Street VARCHAR2(20),
ZipCode VARCHAR2(10),
State VARCHAR2(10),
PRIMARY KEY(Ssn)
);

create table Item(Item_id VARCHAR2(9),
Name VARCHAR2(20),
Price FLOAT,
Storage_Type VARCHAR2(20),
PRIMARY KEY(Item_id)
);

create table Kitchen(Kitchen_id VARCHAR2(9),
Type VARCHAR2(20),
Street VARCHAR2(20),
City VARCHAR2(20),
ZipCode VARCHAR2(10),
```

```
State VARCHAR2(10),
PRIMARY KEY(Kitchen_id)
);

create table Orders(Order_id VARCHAR2(9),
DESCRIPTION VARCHAR2(30),
Ordered_Date DATE,
Delivery_Date DATE,
Delivery_Time VARCHAR2(10),
Street VARCHAR2(20),
City VARCHAR2(10),
ZipCode  VARCHAR2(10),
State VARCHAR2(10),
Delivery_Person_Ssn VARCHAR2(9),
Customer_Ssn VARCHAR2(9),
Manager_Ssn VARCHAR2(9),
PRIMARY KEY(Order_id),
CONSTRAINT FK_ORD_DP_SSN FOREIGN KEY (Delivery_Person_Ssn) references
Delivery_Person(Ssn) on delete set null,
CONSTRAINT FK_ORD_CUST_SSN FOREIGN KEY (Customer_Ssn) references Customer(Ssn) on
delete set null,
CONSTRAINT FK_ORD_MGR_SSN FOREIGN KEY (Manager_Ssn) references Manager(Ssn) on delete
set null
);

create table Receipt(Receipt_No VARCHAR2(9),
PAYMENT_METHOD VARCHAR2(20),
PAYMENT_DATE DATE,
AMOUNT FLOAT,
Order_id VARCHAR2(9),
PRIMARY KEY(Receipt_No),
CONSTRAINT FK_Order FOREIGN KEY (Order_id) references ORDERS(Order_id) on delete set null
);

create table FeedBack(Customer_Ssn VARCHAR2(9),
Order_id VARCHAR2(9),
Rating NUMBER(1),
Comments VARCHAR2(100),
PRIMARY KEY(Order_id,Customer_Ssn),
CONSTRAINT FK_FDBK_ORD_ID FOREIGN KEY (Order_id) references ORDERS(Order_id),
CONSTRAINT FK_FDBK_SSN FOREIGN KEY (Customer_Ssn) references Customer(Ssn)
);

create table Order_Items(Order_id VARCHAR2(9),
Item_id VARCHAR2(9),
Quantity NUMBER,
Use_Before DATE,
PRIMARY KEY(Order_id,Item_id),
CONSTRAINT FK_OI_ORD_ID FOREIGN KEY (Order_id) references ORDERS(Order_id),
CONSTRAINT FK_OI_ITM_ID FOREIGN KEY (Item_id) references Item(Item_id)
```

```
);

create table Prepared_In(Item_id VARCHAR2(9),
Kitchen_id VARCHAR2(9),
PRIMARY KEY(Item_id,Kitchen_id),
CONSTRAINT FK_PREP_IN_KTCH_ID FOREIGN KEY (Kitchen_id) references Kitchen(Kitchen_id) on
delete cascade,
CONSTRAINT FK_PREP_IN_ITM_ID FOREIGN KEY (Item_id) references Item(Item_id) on delete
cascade
);

create table Kitchen_Employee(
Ssn VARCHAR2(9),
Hourly_Wage NUMBER(5,2),
No_of_Dishes NUMBER(3,0),
PRIMARY KEY(Ssn),
CONSTRAINT FK_KE_Ssn FOREIGN KEY (Ssn) references Catering_Employee(Ssn) on delete cascade
);

create table Works_For(
Kitchen_id VARCHAR2(9),
Employee_Ssn VARCHAR2(9),
Shift VARCHAR2(8),
DAY VARCHAR2(10),
EXPERTISE_LEVEL VARCHAR2(20),
CHECK(Shift IN ('MORNING','NOON','NIGHT')) );

alter table Feedback
ADD check(Rating  BETWEEN 1 AND 5);

alter table Kitchen
ADD check(TYPE IN('VEG','NON-VEG'));
```

PL/SQL

Procedures

- GetTotalPrice - Given the orderId, computes the total cost of complete order, taking into account every item's price and quantity. This procedure is used for generating receipt for an order.

```
set SERVEROUTPUT ON;
DECLARE
totalPrice FLOAT;

PROCEDURE GetTotalPrice(
 orderId IN VARCHAR2,
 totalPrice OUT FLOAT)AS
thisOrderId ORDER_ITEMS.ORDER_ID%TYPE;
BEGIN
 SELECT oi.order_id , SUM(oi.quantity * it.price) INTO thisOrderId, totalPrice
 FROM ORDER_ITEMS oi, ITEM it
 WHERE it.item_id = oi.item_id
 GROUP BY oi.order_id;
END;

BEGIN
 GetTotalPrice(101, totalPrice);
 DBMS_OUTPUT.PUT_LINE('Order#'||101 ||' has total price: '|| totalPrice ||'$'
); END;
```

- Use_Before_Notification - Retrieves items that are going to perish in the upcoming week, along with details on the number of remaining days. This is used to notify the customer in advance about the items going to be expired.

```
set SERVEROUTPUT ON;
DECLARE

PROCEDURE Use_Before_Notification AS
thisFName CUSTOMER.FNAME%TYPE;
thisLName CUSTOMER.LNAME%TYPE;
thisItemName ITEM.NAME%TYPE;
thisOrderId ORDER_ITEMS.ORDER_ID%TYPE;
thisItemId ORDER_ITEMS.ITEM_ID%TYPE;
remaining_days NUMBER;
CURSOR UseBeforeCursor IS
        SELECT CUSTOMER.FNAME, CUSTOMER.LNAME,
ITEM.NAME,ORDER_ITEMS.ORDER_ID, ORDER_ITEMS.ITEM_ID,
TRUNC(USE_BEFORE) -(TRUNC(SYSDATE)) AS days
        FROM ORDER_ITEMS, ORDERS, ITEM, CUSTOMER
        WHERE  USE_BEFORE BETWEEN TRUNC(SYSDATE) AND
((TRUNC(SYSDATE))+7)
 AND ORDER_ITEMS.ORDER_ID = ORDERS.ORDER_ID
    AND ORDERS.CUSTOMER_SSN = CUSTOMER.SSN
```

```
      AND ORDER_ITEMS.ITEM_ID = ITEM.ITEM_ID;
BEGIN
 OPEN UseBeforeCursor;
 LOOP
  FETCH UseBeforeCursor INTO thisFName, thisLName, thisItemName,
thisOrderId, thisItemId, remaining_days;
  EXIT WHEN (UseBeforeCursor%NOTFOUND);
  DBMS_OUTPUT.PUT_LINE('Notifying '||thisFName ||' '||thisLName ||' about
'|| thisItemName ||' which expires in '|| remaining_days || ' day(s)');
 END LOOP;
 CLOSE UseBeforeCursor;
END;
BEGIN
 Use_Before_Notification();
END;
```

Triggers

- assign_expertise - A trigger is created that updates the expertise level of the employee when the number of dishes that is cooked by the employee is updated.

```
create or replace TRIGGER assign_expertise
          AFTER
          UPDATE OF no_of_dishes
          ON KITCHEN_EMPLOYEE
 FOR EACH ROW
BEGIN
 IF (:NEW.no_of_dishes >= 20) THEN
  UPDATE WORKS_FOR SET EXPERTISE_LEVEL= 'EXPERT' WHERE
EMPLOYEE_SSN=:NEW.SSN;
 END IF;
 IF (:NEW.no_of_dishes > 10) AND(:NEW.no_of_dishes < 20) THEN
  UPDATE WORKS_FOR SET EXPERTISE_LEVEL= 'INTERMEDIATE' WHERE
EMPLOYEE_SSN=:NEW.SSN;
 END IF;
 IF (:NEW.no_of_dishes <= 10) THEN
  UPDATE WORKS_FOR SET EXPERTISE_LEVEL= 'BEGINNER' WHERE
EMPLOYEE_SSN=:NEW.SSN;
 END IF;
END;
```

- average_rating - A trigger is created that displays the average rating of the catering service after each feedback is inserted.

```
set SERVEROUTPUT ON
create or replace TRIGGER average_rating
        AFTER
        INSERT
        ON FEEDBACK
DECLARE thisAvg NUMBER;
CURSOR average_cursor IS
 SELECT AVG(RATING)FROM FEEDBACK;
BEGIN
   OPEN average_cursor;
 LOOP
   FETCH average_cursor INTO  thisAvg;
   EXIT WHEN (average_cursor%NOTFOUND);
   DBMS_OUTPUT.PUT_LINE(' average rating: '||thisAvg);
 END LOOP;
 CLOSE average_cursor;
END;
```

Business Rules

- User submits a feedback for an order. He gives a rating and optional comments. The rating submitted by user must lie in the range of 1 to 5.

```
alter table Feedback
ADD check(Rating  BETWEEN 1 AND 5);
```

- There must be only two types of kitchen. One which just cooks vegetarian food and the other that cooks only non vegetarian food.

```
alter table Kitchen
ADD check(TYPE IN('VEG','NON-VEG'));
```

- Shift for Kitchen_Employee is restricted to the below three values: MORNING, NOON, NIGHT.

```
CHECK(Shift IN ('MORNING','NOON','NIGHT'))
```

Revision History

Date	Version	Changes	Authors
12/06/2016	1.0	Report Draft	Padma Sri, Vincy Shrine