**Furniture Store**

By,

Manoj Kumar Natha(mxn151930)

Venkata Sai Chaitanya Vaddella(vxv150530)

Abhitej Boorla(axb154830)

**Description:**

This document contains the information about the Data Base design of the Furniture Store, PL/SQL are also implemented on this Data Base. This Data Base mainly stores the information required by the Furniture Store like that of the Customers, Bills, Orders, Employee, Employee Salary, and Products etc. We store the customer details such as his name, address, phone number, date of birth and email. Each customer is assigned a unique customer id. We have information of the all the products that are available in store as a part of Product catalog, it stores information such as name, category, price which is 10% of the purchase price, quantity left, and also a unique pid for each product. Employee information, who work in the store such as name, phone number, ssn, address, email, date of birth, pay\_scale. Employee is then specialized into three categories like Manager, Sales\_man, and Billing person. Manager’s department is stored in dept. Billing person’s counter number is stored in counter\_no. Sales\_man a score based on his performance which is stored in the sale\_score. When a customer purchases a bill will be generated which stores the information like bill\_id, total\_amt, bill\_date. This bill is related to the product to know the purchased items and the quantity of each item. When an item is purchased by customer its quantity in the product catalog will be reduced accordingly. Store purchases the stock through order. Order stores the information like oid, amount, ord\_date. This order is related to the product and the quantity of the each product in the order is incremented accordingly in the product catalog. Employee has the salary based on the hours he worked, salary details such as hours, salary, sal\_date, which are stored in the employee\_salary and is related to the employee. Using PL/SQL we have implemented triggers and procedure on the Data base. Triggers are made on the bill, order, employee salary.

**Data Requirements:**

The Data requirements of each entity is described below:

**Customer:**

**Cid**: It is the primary key, which is unique for each customer.

**Name**: Name of the customer, it is compound attribute. Which in turn has middle name(mname), last name(lname), first name(fname).

**Pno**: Phone number of the customer.

**Address**: Address of the customer.

**Dob**: Date of birth of the customer.

**Email**: Email address of the employee.

**Employee:**

**Ssn**: It is the primary key, which is unique for each employee.

**Name**: Name of the employee.

**Address**: Address of the employee.

**Pno**: Phone number of the employee.

**Pay\_scale**: Used to calculate the salary of the employee.

**Dob**: Date of birth of the employee.

**Email**: Email address of the employee.

Employee is specialized into three categories.

**Manager:**

**Dept**: Stores the department name of the manager.

**Sales\_man:**

**Scale\_score**: Based on the performance eachsales\_man is assigned a score.

**Billing\_person:**

**Counter\_no**: Stores the billing counter number of the billing person.

**Product:**

**Pid**: It is the primary key, assigned to each product.

**Price**: Price of product.

**Pname**: Name of the product.

**Category**: Stores the category of the product.

**Qty\_left**: Stores the quantity of the product left.

**Bill:**

**Bid**: It is the primary key, which is assigned to each bill generated.

**Bill\_date**: Date on which bill is generated.

**Total\_amt**: Amount purchased on that bill.

**Order:**

**Oid**: Primary key of the order, generated when an order is placed by the store.

**Amount**: Total amount made by the order.

**Ord\_date**: Date on which order is made.

**Employee\_salary:**

**Hours**: Number of hours employee worked.

**Sal\_date**: Date on which salary was issued to employee.

**Salary**: Calculated salary of the employee based on his worked hours.

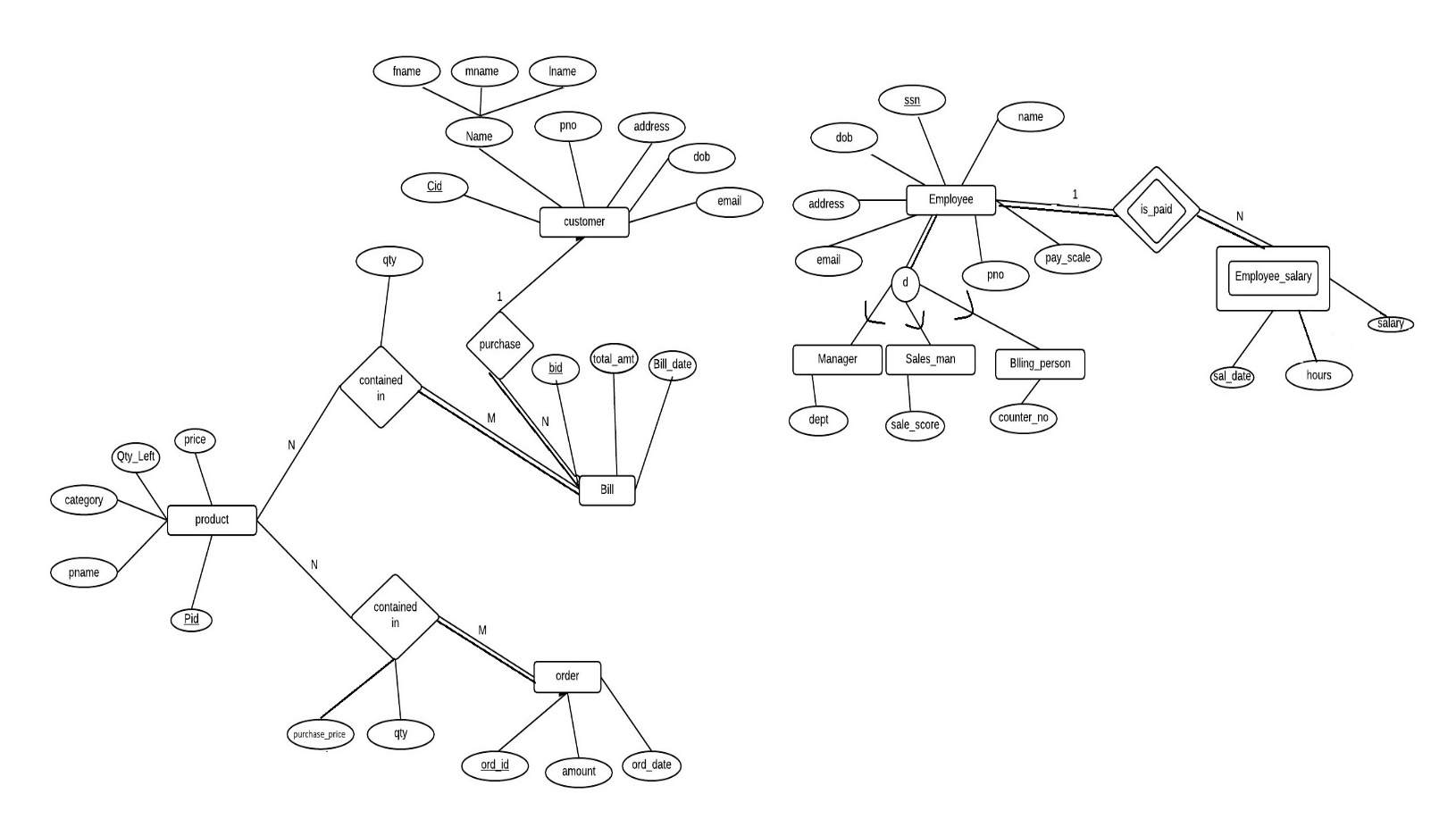
**PL/SQL:**

Once the bill is made we reduce the quantity in the ‘product’ according to the purchased product and the amount of purchase is calculated and updated in the amount in bill through trigger.

When the order is generated the quantity of the product is incremented in the ‘product’ and the amount is calculated and inserted in the order through the trigger.

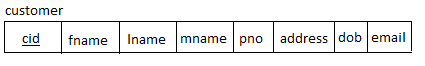
A procedure is made to calculate the profit/loss within the specified dates. We consider the bill, order amounts and employee salary to calculate the profit/loss made by the store during the given period.

**EER Diagram:**

****

* **The various entities present in the ER diagram are:**
* Customer
* Employee – Manager, Sales\_man, Billing\_person
* Product
* Bill
* Order
* Employee\_salary
* Customer information is store in the customer table.
* Employee are further specialized into 3 types – Manager, Sales\_man, billing\_person.
* Products gives the description of various products including the quantity left etc.
* The purchase made by the customer is stored in the Bill, and similarly is the purchase of stock is stored in orders.
* The information of hours worked of each employee and the salaries paid to him are stored in Employee\_salary.

**Mapping of the ER diagram into relational schema:**



Primary key: cid

Foreign Key: none



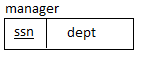
Primary key: pid

Foreign Key: none



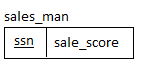
Primary key: ssn

Foreign Key: none



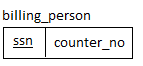
Primary key: ssn

Foreign Key: ssn -> employee(ssn)



Primary key: ssn

Foreign Key: ssn -> employee(ssn)



Primary key: ssn

Foreign Key: ssn -> employee(ssn)



Primary key: (ssn, sal\_date)

Foreign Key: ssn -> employee(ssn)



primary key: oid

Foreign Key: none



Primary key: (oid, pid)

Foreign key: o\_id -> orders(ord\_id)

Foreign\_key: pid -> product(pid)



Primary key: bid

Foreign key: cid -> customer(cid)

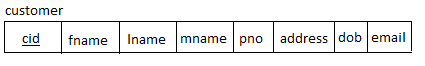


Primary key: (bid, pid)

Foreign key: bid -> bills(bid)

Foreign key: pid -> product(pid)

**Functional Dependencies:**



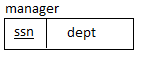
cid {fname, lname, mname, pno, address, dob, email}



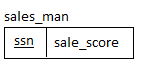
pid {pname, price, qty\_left, categories}



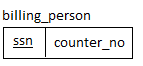
ssn {ename, pno, address, dob, email, pay\_scale}



ssn {dept}



ssn {sale\_score}



ssn {counter\_no}



{ssn, sal\_date} {hours, salary}



ord\_id {amount, ord\_date}



{oid, pid} {cprice, qty,purchase\_price}



bid {cid, total\_amount, bill\_date}

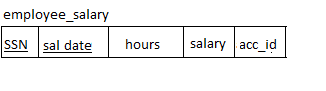


{bid, pid} {qty}

**Normalization Rules:**

We designed the tables directly in 3NF by keeping the normalization rules in the mind. So we don’t have any table to normalize step by step to 3NF.

The tabl



We separated the tables above into 2 as the employee may have many salaries with many different dates, it might not even been in 1NF if we use a multi valued table.

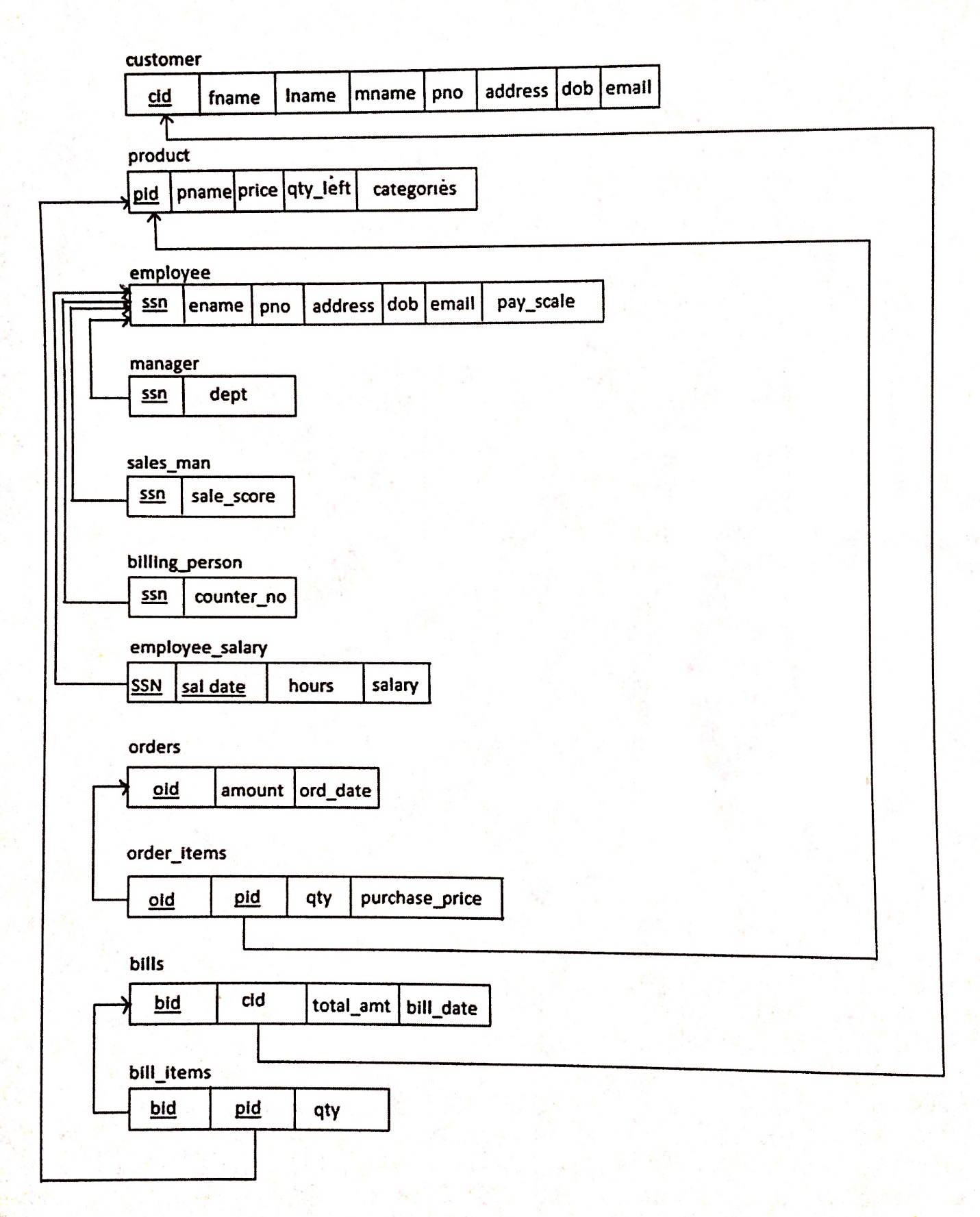
**For All the remaining tables:**

All the remaining tables that are created does not have multi values, so those are in 1NF.

In all the tables non prime attributes depends on the prime attributes, hence those are in 2NF.

In all the tables non prime depends on prime only, hence all the attributes are in 3NF

**Final Relational Schema:**



**PL/SQL:**

1) **Bill\_items\_insert Trigger:**

This trigger is used to update the price in bills whenever there is a new entry is made into bill\_item table and reduces the stock by the quantity of the product in the bill item.

Trigger code:

create or replace trigger bill\_items\_insert

AFTER insert on bill\_items

for each row

DECLARE

x bills.total\_amt%type;

y product.qty\_left%type;

z product.price%type;

begin

Select total\_amt into x from bills where bid=:new.bid;

Select price into z from product where pid=:new.pid;

x:=x+(:new.qty\*z);

Select qty\_left into y from product where pid=:new.pid;

y:=y-:new.qty;

update bills set total\_amt=x where bid=:new.bid;

update product set qty\_left=y where pid=:new.pid;

end;

2) **Order\_items\_insert Trigger:**

This trigger is used to update the price in orders whenever there is a new entry is made in order\_items table and increases the stock by the quantity of the product in the order item.

**Trigger code:**

create or replace trigger order\_items\_insert

AFTER insert on order\_items

for each row

DECLARE

x orders.amount%type;

y PRODUCT.QTY\_LEFT%type;

z product.price%type;

begin

Select amount into x from orders where oid=:new.oid;

Select price into z from product where pid=:new.pid;

x:=x+(z\*:new.qty);

z:=:new.purchase\_price\*(1.1);

Select qty\_left into y from product where pid=:new.pid;

y:=y+:new.qty;

update orders set amount=x where oid=:new.oid;

update product set qty\_left=y where pid=:new.pid;

update product set price=z where pid=:new.pid;

end;

3) **PL/SQL to find profit or loss in a given period of time:**

This procedure is used to check if there is profit or loss for some certain given dates and it tells how much it is.

**Procedure code:**

create or replace procedure checkProfitorLoss(date1 IN date,date2 IN date) AS

thisord orders%ROWTYPE;

ordsum orders.amount%type;

thisbill bills%ROWTYPE;

billsum bills.total\_amt%type;

thissal employee\_salary%ROWTYPE;

salsum employee\_salary.salary%type;

finalsum employee\_salary.salary%type;

CURSOR Allord IS

SELECT \* FROM orders where ord\_date between (date1-1) and (date2+1);

CURSOR Allbill IS

SELECT \* FROM bills where bill\_date between (date1-1) and (date2+1);

CURSOR Allsal IS

SELECT \* FROM employee\_salary where sal\_date between (date1-1) and (date2+1);

BEGIN

ordsum:=0;

billsum:=0;

salsum:=0;

OPEN Allord;

Open Allbill;

Open allsal;

LOOP

FETCH Allord INTO thisord;

EXIT WHEN (Allord%NOTFOUND);

ordsum:=ordsum+thisord.amount;

END LOOP;

LOOP

FETCH Allbill INTO thisbill;

EXIT WHEN (Allbill%NOTFOUND);

billsum:=billsum+thisbill.total\_amt;

END LOOP;

LOOP

FETCH Allsal INTO thissal;

EXIT WHEN (Allsal%NOTFOUND);

salsum:=salsum+thissal.salary;

END LOOP;

salsum:=salsum\*(-1);

ordsum:=ordsum\*(-1);

finalsum:=salsum+ordsum+billsum;

IF finalsum < 0 THEN

dbms\_output.put\_line('There was a loss in this period of amount:'||finalsum);

ElSIF finalsum > 0 THEN

dbms\_output.put\_line('There was a profit in this period of amount:'||finalsum);

ELSE

dbms\_output.put\_line('There was no profit or loss in this period');

END IF;

CLOSE Allord;

CLOSE Allsal;

CLOSE Allbill;

END;

**PL/SQL Code to execute the procedure:**

set serveroutput on;

BEGIN

dbms\_output.put\_line('The details of profit or loss :');

CHECKPROFITORLOSS('01-dec-15','10-dec-15');

END;

.

RUN;