**A**
**Project Report**
**Submitted for**

# DATABASE MANAGEMENT SYSTEM (UCS310)


**Submitted by:**

**(101703329) Manmeet Kaur Chawla**

**(101883059) Priyanshu Tuli**

**(101883072) Yashwika**

**BE Second Year**


Submitted to-
Mr. Anil Vashisht



**Computer Science and Engineering Department**
**TIET, Patiala**

**Jan-May 2019**

# TABLE OF CONTENTS


**TABLE OF CONTENTS** i

**ABSTRACT** ii

**BACKGROUND OF THE AREA** iii

**NEED OF THE PROJECT** iv

**OBJECTIVE OF THE PROJECT** v

**INDEX** vi

# ABSTRACT

The following report is a detailed study of how we can use RDBMS for managing a stationary business and how with the use of PL/SQL we can calculate loss or profit in the business.

The sole purpose of creating this report is to understand how with the help of SQL, E-R diagrams and PL/SQL we are able to create a database according to our needs and how such type of system is used to provide the automation on any type of the stationary shop. That means a shop which has the type system which provides the facility to the customers of the shop to purchase the products from the shop without any complexity.

# BACKGROUND OF THE AREA

The business management system, discussed here forth, is the process of buying the stationary items from the wholesalers and then selling it to the customers. It acts as a middle person between the wholesalers and the sellers. The owner has to buy the products from the wholesalers. Managing the details that is regarding the whole sale is not efficient to be kept in track through the pen paper mode. It requires the database system where we can store the information of the people involved in the wholesale system management. It can be stored clearly in the database with more precision and can avoid the redundant data.

# NEED OF THE PROJECT

At present, the Wholesale and Retail outlets are working under manual management. The client uses MS Excel, all records related to Products, Sales, Suppliers, Orders, Payment are stored in excel files. There is lot of duplicate work, and chance of mistake. When the records are changed, they need to update each and every excel file. To manage the whole data, the person maintaining records has to take great pain. Various excel files has to be maintained for each separate process. There is no option to find previous saved records. There is no security; anybody can access any report and sensitive data.

# OBJECTIVE OF THE PROJECT

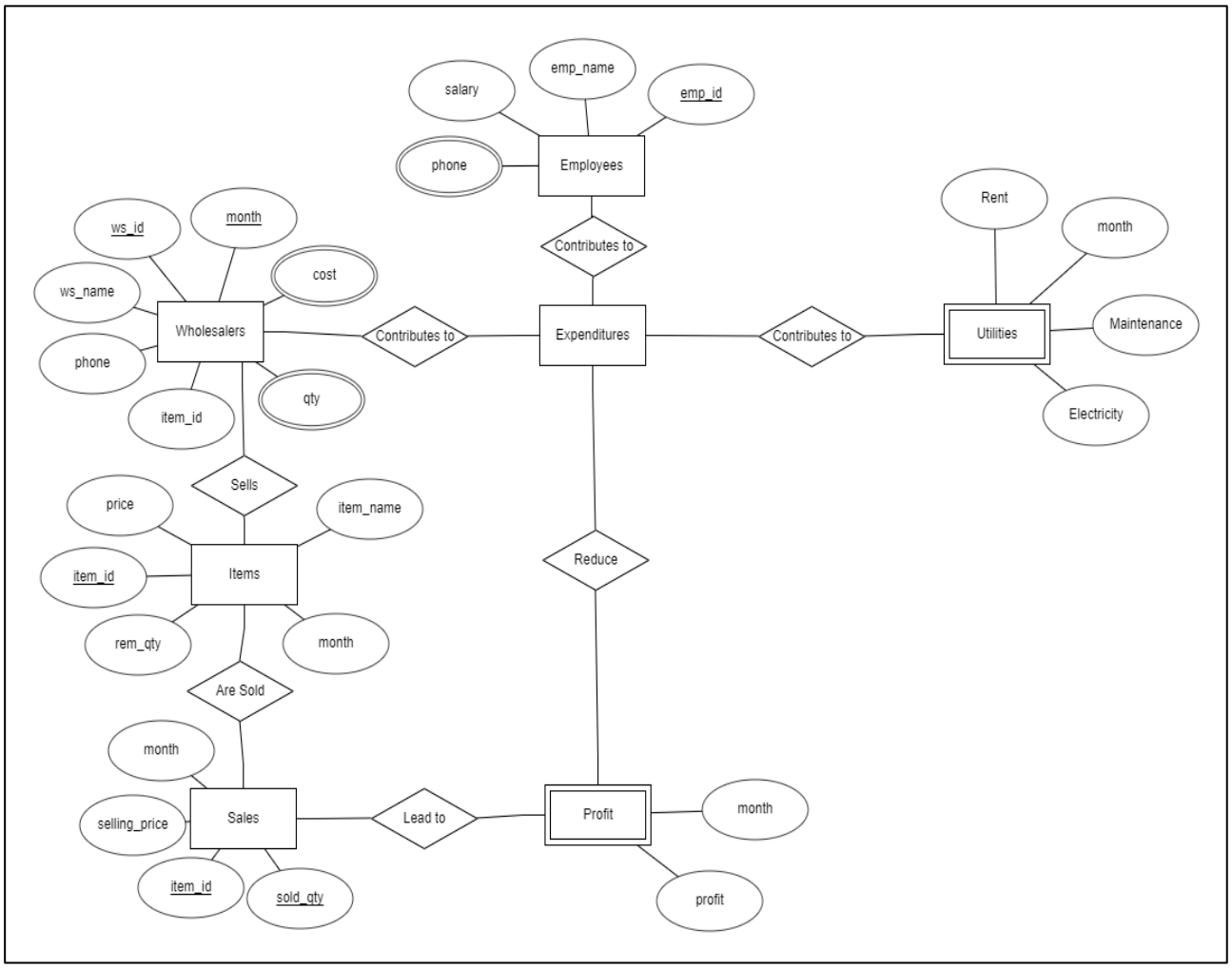
This Stationary Shop Management System is used to overcome the entire problem which they are facing currently, and making manual system to computerized system. Purposed stationary shop management system should help the customers query whether a product is in stock the user can query the availability of a book either by using the item id or by using the name of item. The objective and scope of our Project Stationary Shop Management System is to record the details various activities of user. It simplifies the task and reduce the paperwork. Stationary shop management system should generate sales statistics (item name, wholesaler, number of copies sold and the sales revenue) for any period.

If customer request for a product and the product is not currently sold by the shop, then the customer is asked to enter the full detail of the product for procurement of the book by the shop. If the requested product is in stock, the exact number of items are available and the price of the product should be displayed. In proposed system, as soon as customer selects his product for purchase, the sale clerk would enter the item id of the product. Stationary shop management system should update the stock and generate the sales receipt for the product. The purpose of this project is to manage the products in the shop. Generally, it includes the Order Processing, Stock Management and Accounts Management. We developed this project to maintain records of sales, purchase records. Here we try to develop such type system which is provide the automation on any type of the stationary shop. That means a shop which has the type system which provides the facility to the customers of the shop to purchase the products from the shop without any complexity.

# INDEX


| Topic | Page No. |
|---|---|
| ER Diagram | 1 |
| Table Creation | 2-3 |
| Table Description | 4-7 |
| Normalization and PL/SQL Codes | 8-9 |
| References | 10 |

# E-R DIAGRAM


salary
emp_name
emp_id
phone
Employees
Contributes to
Rent
month
Utilities
Maintenance
Electricity
ws_id
month
cost
ws_name
Wholesalers
phone
item_id
qty
Contributes to
Expenditures
Contributes to
Sells
price
item_name
item_id
Items
rem_qty
month
Are Sold
Reduce
month
selling_price
Sales
item_id
sold_qty
Lead to
Profit
month
profit

# TABLE CREATION

## WHOLESALERS

```
create table wholesalers(
ws_id number(10),
item_id number(10),
cost number(10,2),
name varchar2(20),
phone number(10),
qty number(10),
month date,
constraint wholesalers_wsid_itemid_pk primary key(item_id),
constraint wholesalers_wsid_fk foreign key(ws_id) references wholesalers_info(ws_id) on delete cascade)
```

Table created.

## WHOLESALERS_INFO

```
create table wholesalers_info(
ws_id number(10),
ws_name varchar2(20),
phone number(10),
month date,
constraints wholesalers_info_month_pk primary key(month),
constraints wholesalers_info_wsid_unique unique(ws_id))
```

Table created.

## UTILITIES

```
create table utilities(
month date,
elec_bill number(10,2),
rent number(10),
maintenance number,
constraints utilities_month_fk foreign key(month) references wholesalers_info(month) on delete cascade)
```

Table created.

## EMPLOYEES

```
create table employees(
emp_id number(10),
emp_name varchar2(20),
phone number(10),
salary number(10,2),
constraints employees_empid_pk primary key(emp_id))
```

Table created.

**EXPENDITURE**

```
create table expenditure(
wholesalers number(10),
month date,
utilities number(10),
salary number(10,2),
total number(10),
constraints expenditure_month_fk foreign key(month) references wholesalers_info(month) on delete cascade)
```

Table created.

**ITEM**

```
create table item(
item_id number(10),
item_name varchar2(20),
rem_qty number(10),
month date,
price number(10,2),
constraints item_itemid_pk primary key(item_id),
constraints item_month_fk foreign key(month) references wholesalers_info(month) on delete cascade)
```

Table created.

**SALES**

```
create table sales(
month date,
item_id number(10),
sold_qty number(10),
selling_price number(10,2),
constraints sales_itemid_sold_qty primary key(item_id,sold_qty),
constraints sales_month_fk foreign key(month) references wholesalers_info(month) on delete cascade)
```

Table created.

**PROFIT**

```
create table profit(
month date,
profit number(10,2),
constraints profit_month_fk foreign key(month) references wholesalers_info(month) on delete cascade)
```

Table created.

# TABLE DESCRIPTION

## WHOLESALERS

Columns

| # | Column | Type | Length | Precision | Scale | Nullable | Semantics |
|---|---|---|---|---|---|---|---|
| 1 | WS_ID | NUMBER | 22 | 10 | 0 | Yes | |
| 2 | ITEM_ID | NUMBER | 22 | 10 | 0 | No | |
| 3 | COST | NUMBER | 22 | 10 | 2 | Yes | |
| 4 | NAME | VARCHAR2 | 20 | | | Yes | Byte |
| 5 | PHONE | NUMBER | 22 | 10 | 0 | Yes | |
| 6 | QTY | NUMBER | 22 | 10 | 0 | Yes | |
| 7 | MONTH | DATE | 7 | | | Yes | |

Indexes

| Index Name | Index Type | Uniqueness | Status | Columns |
|---|---|---|---|---|
| WHOLESALERS_WSID_ITEMID_PK | NORMAL | UNIQUE | VALID | ITEM_ID |

Constraints

| Constraint | Type | Condition | On Delete | Status |
|---|---|---|---|---|
| WHOLESALERS_WSID_FK | Foreign Key | - | CASCADE | ENABLED |
| WHOLESALERS_WSID_ITEMID_PK | Primary Key | - | - | ENABLED |

## WHOLESALERS_INFO

Columns

| # | Column | Type | Length | Precision | Scale | Nullable | Semantics |
|---|---|---|---|---|---|---|---|
| 1 | WS_ID | NUMBER | 22 | 10 | 0 | Yes | |
| 2 | WS_NAME | VARCHAR2 | 20 | | | Yes | Byte |
| 3 | PHONE | NUMBER | 22 | 10 | 0 | Yes | |
| 4 | MONTH | DATE | 7 | | | No | |

Indexes

| Index Name | Index Type | Uniqueness | Status | Columns |
|---|---|---|---|---|
| WHOLESALERS_INFO_MONTH_PK | NORMAL | UNIQUE | VALID | MONTH |
| WHOLESALERS_INFO_WSID_UNIQUE | NORMAL | UNIQUE | VALID | WS_ID |

Constraints

| Constraint | Type | Condition | On Delete | Status |
|---|---|---|---|---|
| WHOLESALERS_INFO_MONTH_PK | Primary Key | - | - | ENABLED |
| WHOLESALERS_INFO_WSID_UNIQUE | Unique Key | - | - | ENABLED |

## EMPLOYEES

Columns

| # | Column | Type | Length | Precision | Scale | Nullable | Semantics |
|---|---|---|---|---|---|---|---|
| 1 | EMP_ID | NUMBER | 22 | 10 | 0 | No | |
| 2 | EMP_NAME | VARCHAR2 | 20 | | | Yes | Byte |
| 3 | PHONE | NUMBER | 22 | 10 | 0 | Yes | |
| 4 | SALARY | NUMBER | 22 | 10 | 2 | Yes | |

Indexes

| Index Name | Index Type | Uniqueness | Status | Columns |
|---|---|---|---|---|
| EMPLOYEES_EMPID_PK | NORMAL | UNIQUE | VALID | EMP_ID |

Constraints

| Constraint | Type | Condition | On Delete | Status |
|---|---|---|---|---|
| EMPLOYEES_EMPID_PK | Primary Key | - | - | ENABLED |

## UTILITIES

Columns

| # | Column | Type | Length | Precision | Scale | Nullable | Semantics |
|---|---|---|---|---|---|---|---|
| 1 | MONTH | DATE | 7 | | | Yes | |
| 2 | ELEC_BILL | NUMBER | 22 | 10 | 2 | Yes | |
| 3 | RENT | NUMBER | 22 | 10 | 0 | Yes | |
| 4 | MAINTENANCE | NUMBER | 22 | | | Yes | |

Constraints

| Constraint | Type | Condition | On Delete | Status |
|---|---|---|---|---|
| UTILITIES_MONTH_FK | Foreign Key | - | CASCADE | ENABLED |

## EXPENDITURES

Indexes

| Index Name | Index Type | Uniqueness | Status | Columns |
|---|---|---|---|---|
| EMPLOYEES_EMPID_PK | NORMAL | UNIQUE | VALID | EMP_ID |

Constraints

| Constraint | Type | Condition | On Delete | Status |
|---|---|---|---|---|
| EMPLOYEES_EMPID_PK | Primary Key | - | - | ENABLED |

## ITEMS

Columns

| # | Column | Type | Length | Precision | Scale | Nullable | Semantics |
|---|---|---|---|---|---|---|---|
| 1 | ITEM_ID | NUMBER | 22 | 10 | 0 | No | |
| 2 | ITEM_NAME | VARCHAR2 | 20 | | | Yes | Byte |
| 3 | REM_QTY | NUMBER | 22 | 10 | 0 | Yes | |
| 4 | MONTH | DATE | 7 | | | Yes | |
| 5 | PRICE | NUMBER | 22 | 10 | 2 | Yes | |

Indexes

| Index Name | Index Type | Uniqueness | Status | Columns |
|---|---|---|---|---|
| ITEM_ITEMID_PK | NORMAL | UNIQUE | VALID | ITEM_ID |

Constraints

| Constraint | Type | Condition | On Delete | Status |
|---|---|---|---|---|
| ITEM_MONTH_FK | Foreign Key | - | CASCADE | ENABLED |
| ITEM_ITEMID_PK | Primary Key | - | - | ENABLED |

## SALES

Columns

| # | Column | Type | Length | Precision | Scale | Nullable | Semantics |
|---|---|---|---|---|---|---|---|
| 1 | MONTH | DATE | 7 | | | Yes | |
| 2 | ITEM_ID | NUMBER | 22 | 10 | 0 | No | |
| 3 | SOLD_QTY | NUMBER | 22 | 10 | 0 | No | |
| 4 | SELLING_PRICE | NUMBER | 22 | 10 | 2 | Yes | |

Indexes

| Index Name | Index Type | Uniqueness | Status | Columns |
|---|---|---|---|---|
| SALES_ITEMID_SOLD_QTY | NORMAL | UNIQUE | VALID | ITEM_ID, SOLD_QTY |

Constraints

| Constraint | Type | Condition | On Delete | Status |
|---|---|---|---|---|
| SALES_MONTH_FK | Foreign Key | - | CASCADE | ENABLED |
| SALES_ITEMID_SOLD_QTY | Primary Key | - | - | ENABLED |

## PROFIT

Columns

| # | Column | Type | Length | Precision | Scale | Nullable | Semantics |
|---|---|---|---|---|---|---|---|
| 1 | MONTH | DATE | 7 | | | Yes | |
| 2 | PROFIT | NUMBER | 22 | 10 | 2 | Yes | |

Constraints

| Constraint | Type | Condition | On Delete | Status |
|---|---|---|---|---|
| PROFIT_MONTH_FK | Foreign Key | - | CASCADE | ENABLED |

## NORMALIZATION AND PL/SQL CODES

Normalization is used for mainly two purposes,

- Eliminating redundant (useless) data.
- Ensuring data dependencies make sense i.e data is logically stored.

**All the above created tables are already in 3rd normal form.**

TRIGGER TO CALCULATE REMAINING ITEMS IN ITEM TABLE

```
create or replace trigger update_rem_qty
after update of rem_qty on item
for each row
declare
diff number:=:old.rem_qty-:new.rem_qty;
begin
if diff>0 then
update sales set sold_qty=sold_qty+diff;
end if;
end;
```

Trigger created.

Triggers

| Trigger | Body |
|---|---|
| UPDATE_REM_QTY<br>ENABLED<br>UPDATE | declare<br>diff number:=:old.rem_qty-:new.rem_qty;<br>begin<br>if diff>0 then<br>update sales set sold_qty=sold_qty+diff;<br>end if;<br>end; |

CURSOR TO CALCULATE PROFIT EARNED

```
declare
cursor c1 is select total from expenditure;
cursor c2 is select selling_price from sales;
x1 expenditure.total%type;
x2 sales.selling_price%type;
begin
open c1;
open c2;
loop
fetch c1 into x1;
fetch c2 into x2;
exit when c1%notfound;
exit when c2%notfound;
insert into profit(profit) values(x1+x2);
end loop;
close c1;
close c2;
end;
```

Statement processed.

CURSOR TO FIND TOTAL EXPENDITURE

```
declare
cursor c1 is select * from wholesalers;
cursor c2 is select * from utilities;
cursor c3 is select salary from employees;
cursor c4 is select * from expenditure;
x1 wholesalers%rowtype;
x2 utilities%rowtype;
x3 employees.salary%type;
x4 expenditure%rowtype;
begin
open c1;
loop
fetch c1 into x1;
exit when c1%notfound;
insert into expenditure(wholesalers) values(x1.cost*x1.qty);
end loop;
close c1;
open c2;
loop
fetch c2 into x2;
exit when c2%notfound;
insert into expenditure(utilities) values(x2.rent+x2.maintenance+x2.elec_bill);
end loop;
close c2;
open c3;
loop
fetch c3 into x3;
exit when c3%notfound;
insert into expenditure(salary) values(x3);
end loop;
close c3;
open c4;
loop
fetch c4 into x4;
exit when c4%notfound;
insert into expenditure(total) values(x4.wholesalers+x4.utilities+x4.salary);
end loop;
close c4;
end;
```

```
Statement processed.
```

# REFERENCES


- GEEKS FOR GEEKS
- ORACLE 9I SQL+ BOOK
- WIKIPEDIA