Supermarket Management System(SMS)

-By
Chetan Ingle (cmi8525)
Viswanath Nagarajan (vn2065)
Assigned Port:8620(http://localhost:8620/)

1. High-Level Application Description:

The SMS is a simple, easy to use database management system that allows small to medium scale supermarkets to run their business locally and through the internet.
It helps in tracking and maintaining the employees, inventory of the store, allows customers to place orders on the internet which the users can pick up within a couple of hours after placing the order, and enable invoice generation.

There are 2 main types of users:

- Customers: That can place orders online
- Admins: That can get the store insights about the inventory and most selling products and can also access information about the employees working at the store.

2. Entities, Relationships, and Business Rules:

- Entities:
    1. Login (user_id, user_type, login_username, login_password)
    2. Customers (customer_id, customer_name, customer_mobile, customer_address, customer_email, customer_dob , customer_gender, user_id)
    3. Employees(employee_id, employee_name, employee_mail, employee_dob, employee_mobile, employee_address, employee_gender)
    4. Stores(store_id,store_type,store_address)
    5. Departments(dept_id, dept_name, manager_id)
    6. Subdepartments(dept_id, sub_dept_id, sub_dept_name)
    7. Suppliers(supplier_id, supplier_name, supplier_category, supplier_address)
    8. Products(product_id, department_id, sub_department_id, supplier_id integer, product_name, price decimal, product_description)
    9. Cart(cart_id, customer_id)
    10. Cart_details(cart_id, store_id, product_id, quantity)
    11. Payment(payment_id, payment_datetime, payment_amount, payment_type, cart_id, customer_id)
    12. Order_History(customer_id, order_id)

- Relationship Set:

Has()…………………………………… Relationship between Customer & Login & Employees
Manage()……………………………. Relationship between Employees & Department
works-in()…………………………… Relationship between Employees & Department
contains()…………………………… Relationship between Departments & Subdepartments
has()…………………………………… Relationship between Departments & Store
contains()…………………………… Relationship between Stores & Products
supplied_by()……………………… Relationship between Products & Supplier
buys()…………………………………. Relationship between Customer & Products
added_to()…………………………. Relationship between Products & Carts
contains()…………………………… Relationship between Cart & Cart_details
creates()……………………………. Relationship between Customer & Cart
checkout()………………………….Relationship between Cart & Payment
makes()……………………………… Relationship between Customer & Payment
saved_to()…………………………. Relationship between Payment & Order_history
has()………………………………….. Relationship between Customer & Order History

- Business Rules:
    1. Each login credential belongs to either a Customer credential or an Employee credential. Every employee has exactly one login credential. Each customer must have a login credential in order to place orders to get it delivered or pick -up.
    2. Employees manage departments. Each department is managed by exactly one employee. Each employee works in some department. Each department has some employees working for it.
    3. Each department contains sub-departments. A department contains 0,1,2 or multiple sub-departments. Each sub-department belongs to exactly one department.
    4. A store can be an online store or a walk-in store. Each store has some departments. Each department can be present in multiple stores.
    5. Each store contains some products. Products can be available in 0,1 or multiple stores.
    6. Every product is supplied by exactly one supplier. A supplier can supply 0,1 or multiple products.
    7. A customer either explores the store or adds a product to the cart. A cart can have 0,1 or multiple products in it.
    8. Customer creates Cart. Each cart is created by exactly one customer.
    9. The cart is then checked out and the customer makes payment. Each cart has exactly one payment and each payment uniquely identifies the cart.
    10. Cart contains cart details. Each cart detail belongs to exactly one cart. Each cart contains 0 or 1 cart detail in it.
    11. Customers make payments, and each payment is made by exactly one customer.
    12. Customers have OrderHistory and OrderHistory will not occur independently of Customers. If the Customer is deleted, then OrderHistory is also deleted.
    13. After checking out every cart and payment detail is saved to OrderHistory. Each orderhistory corresponds to a unique payment detail.

3. Data Gathering:

For this application, we have created our own realistic data by referring to products listed on Amazon. The data we created contains 90 products, 7 departments, 23 sub-departments, 21 employees, 7 suppliers, and a few users and the orders placed by them.

4. Users Interacting with the Database:

We have 2 types of users in our application which are: Customers and Admin.

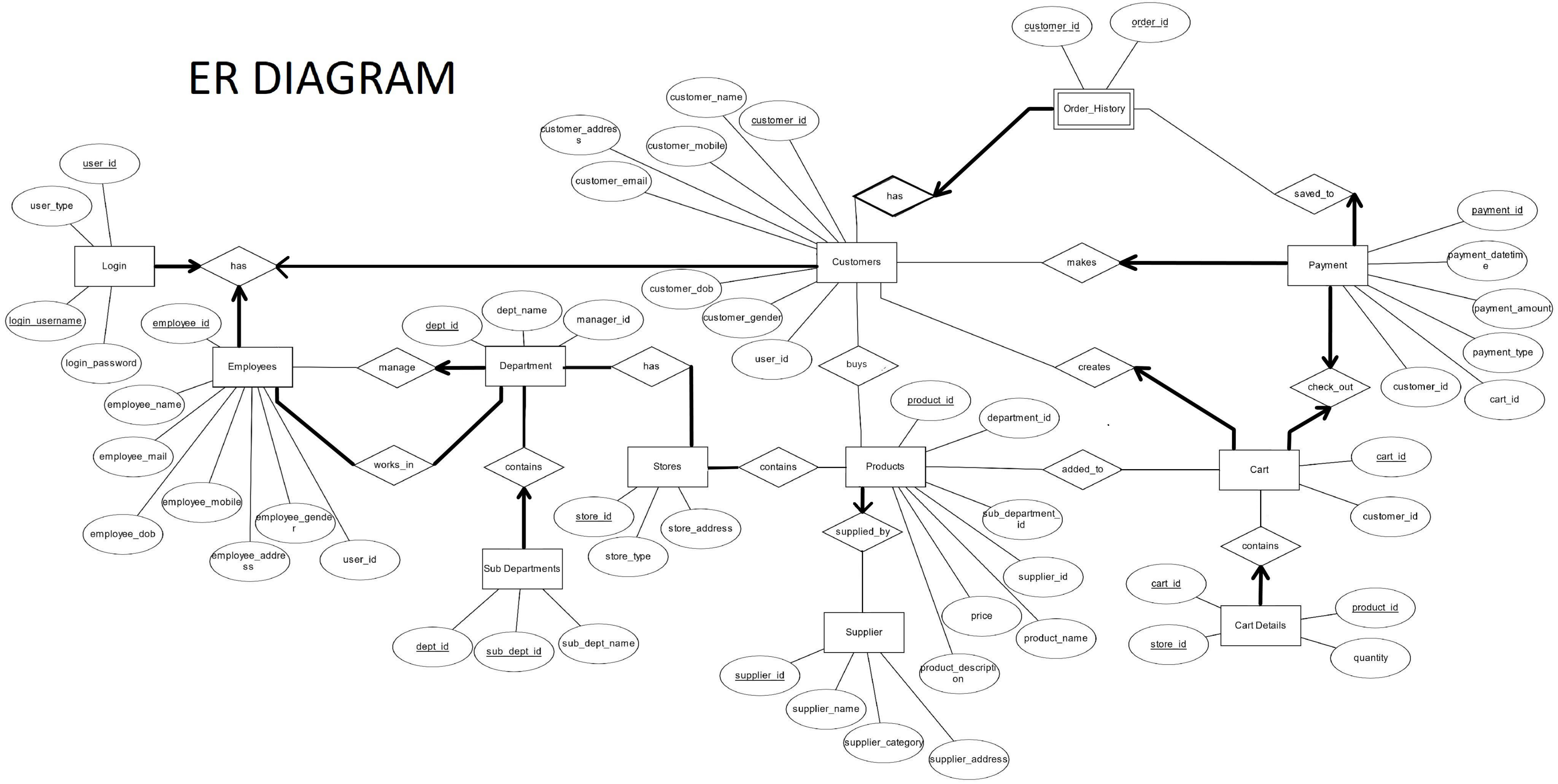
Admin:

1. The Admin is responsible for maintaining the inventory of the store.
2. Admin can get the information of which product is supplied by which supplier.
3. Admin can get a list of products and how much quantity they have for that product in each store. If a product is low in quantity or out of stock then the admin can contact the supplier and request more products
4. Admin can add/update/delete products from the store.
5. Admin can add/delete new departments.

Customers:

1. The Customer can buy products from the store by placing orders online
2. Since the customer is placing an online order, they need to have an account created.
3. A customer can get the list of all products that are available to them. They can also get a list of products filtered on specific categories and price ranges.
4. They can add/remove items to/from the shopping cart.
5. Once the customer adds all their products to the cart, they can proceed to checkout and make the payment.
6. The Customer can view their past orders and history along with the order details.
7. The Customer can also repeat one of their past orders.

ER DIAGRAM



customer_id
order_id
Order_History
customer_name
customer_id
customer_address
customer_mobile
customer_email
has
saved_to
user_id
user_type
Login
has
Customers
makes
Payment
payment_id
payment_datetime
payment_amount
payment_type
customer_id
cart_id
check_out
login_username
login_password
employee_id
customer_dob
customer_gender
user_id
buys
creates
dept_id
dept_name
manager_id
Employees
manage
Department
has
employee_name
employee_mail
employee_mobile
employee_dob
employee_gender
employee_address
user_id
works_in
contains
Stores
contains
Products
product_id
department_id
added_to
Cart
cart_id
customer_id
store_id
store_type
store_address
supplied_by
sub_department_id
supplier_id
price
product_name
product_description
contains
Sub Departments
dept_id
sub_dept_id
sub_dept_name
Supplier
supplier_id
supplier_name
supplier_category
supplier_address
cart_id
store_id
Cart Details
product_id
quantity

Schema: Supermarket Management System(SMS):

```
DROP TABLE IF EXISTS cart CASCADE;

DROP TABLE IF EXISTS cart_check_out_payment CASCADE;

DROP TABLE IF EXISTS cart_details CASCADE;

DROP TABLE IF EXISTS Order_History CASCADE;

DROP TABLE IF EXISTS customers CASCADE;

DROP TABLE IF EXISTS employees CASCADE;

DROP TABLE IF EXISTS login CASCADE;

DROP TABLE IF EXISTS manage_departments CASCADE;

DROP TABLE IF EXISTS products_suppliedby CASCADE;

DROP TABLE IF EXISTS stores CASCADE;

DROP TABLE IF EXISTS store_products CASCADE;

DROP TABLE IF EXISTS store_departments CASCADE;

DROP TABLE IF EXISTS suppliers CASCADE;

DROP TABLE IF EXISTS works_in CASCADE;

DROP TABLE IF EXISTS subdepartments CASCADE;

create table Login(
   user_id serial primary key,
   user_type integer not null,
   login_username varchar(128) not null unique,
   login_password varchar(128) not null
);

create table Customers(
   customer_id serial primary key,
   customer_name varchar(128),
   customer_mobile varchar(128),
   customer_address varchar(128),
   customer_email varchar(128),
   customer_dob date,
   customer_gender varchar(128),
   user_id integer unique not null,
   foreign key(user_id) references Login(user_id)
);

create table Employees(
   employee_id serial primary key,
   employee_name varchar(128) not null,
   employee_mail varchar(128) not null,
   employee_dob date,
   employee_mobile varchar(128),
   employee_address varchar(128),
   employee_gender varchar(20)
);

create table manage_Departments(
   dept_id integer primary key,
   dept_name varchar(128),
   manager_id integer not null,
   foreign key(manager_id) references Employees(employee_id)
);

create table works_in(
   employee_id integer primary key,
   department_id integer not null,
   foreign key(employee_id) references Employees(employee_id),
   foreign key(department_id) references manage_Departments(dept_id)
);

create table Stores(
   store_id integer primary key,
   store_type varchar(128) not null,
   store_address varchar(128)
);

create table store_departments(
   dept_id integer,
   store_id integer,
   primary key(dept_id, store_id),
   foreign key (dept_id) references manage_Departments(dept_id),
   foreign key(store_id) references Stores(store_id)
);

create table subdepartments(
   dept_id integer,
   sub_dept_id integer,
   sub_dept_name varchar(128),
   foreign key (dept_id) references manage_Departments(dept_id),
   primary key (dept_id, sub_dept_id)
```

```
);

create table Suppliers(
   supplier_id integer primary key,
   supplier_name varchar(128),
   supplier_category integer not null,
   supplier_address varchar(128),
   foreign key(supplier_category) references manage_Departments(dept_id)
);

create table Products_suppliedBy(
   product_id integer primary key,
   department_id integer not null,
   sub_department_id integer not null,
   supplier_id integer not null,
   product_name varchar,
   price decimal not null,
   product_description varchar,
   foreign key(supplier_id) references Suppliers(supplier_id)
);

create table store_products(
   store_id integer,
   product_id integer,
   quantity integer CHECK(quantity >= 0),
   primary key(store_id, product_id),
   foreign key(store_id) references Stores(store_id),
   foreign key(product_id) references Products_suppliedBy(product_id)
);

create table cart (
   cart_id serial primary key,
   customer_id integer
);

create table cart_details(
   cart_id integer,
   store_id integer,
   product_id integer not null,
   quantity integer not null,
   primary key (cart_id, store_id, product_id),
   foreign key(cart_id) references cart(cart_id),
   foreign key(store_id) references stores(store_id),
   foreign key(product_id) references products_suppliedby(product_id)
);
```

```
create table Cart_check_out_Payment(
   payment_id serial primary key,
   payment_datetime timestamp,
   payment_amount decimal,
   payment_type varchar(128),
   cart_id integer unique not null,
   customer_id integer not null,
   foreign key (cart_id) references cart(cart_id)
);

create table Order_History(
   customer_id integer,
   order_id integer,
   primary key(customer_id, order_id),
   foreign key (customer_id) references Customers(customer_id) on delete cascade,
   foreign key (order_id) references Cart_check_out_Payment(payment_id)
);
```