Assignment-3 CS303

Shashank P
200010048

October 17, 2022

1 Problem 1

Design a database for an automobile company to provide to its dealers to assist them in maintaining customer records and dealer inventory and to assist sales staff in ordering cars. Each vehicle is identified by a vehicle identification number (VIN). Each individual vehicle is a particular model of a particular brand offered by the company (e.g., the XF is a model of the car brand Jaguar of Tata Motors). Each model can be offered with a variety of options, but an individual car may have only some (or none) of the available options. The database needs to store information about models, brands, and options, as well as information about individual dealers, customers, and cars. Your design should include an E-R diagram, a set of relational schemas, and a list of constraints, including primary-key and foreign-key constraints.

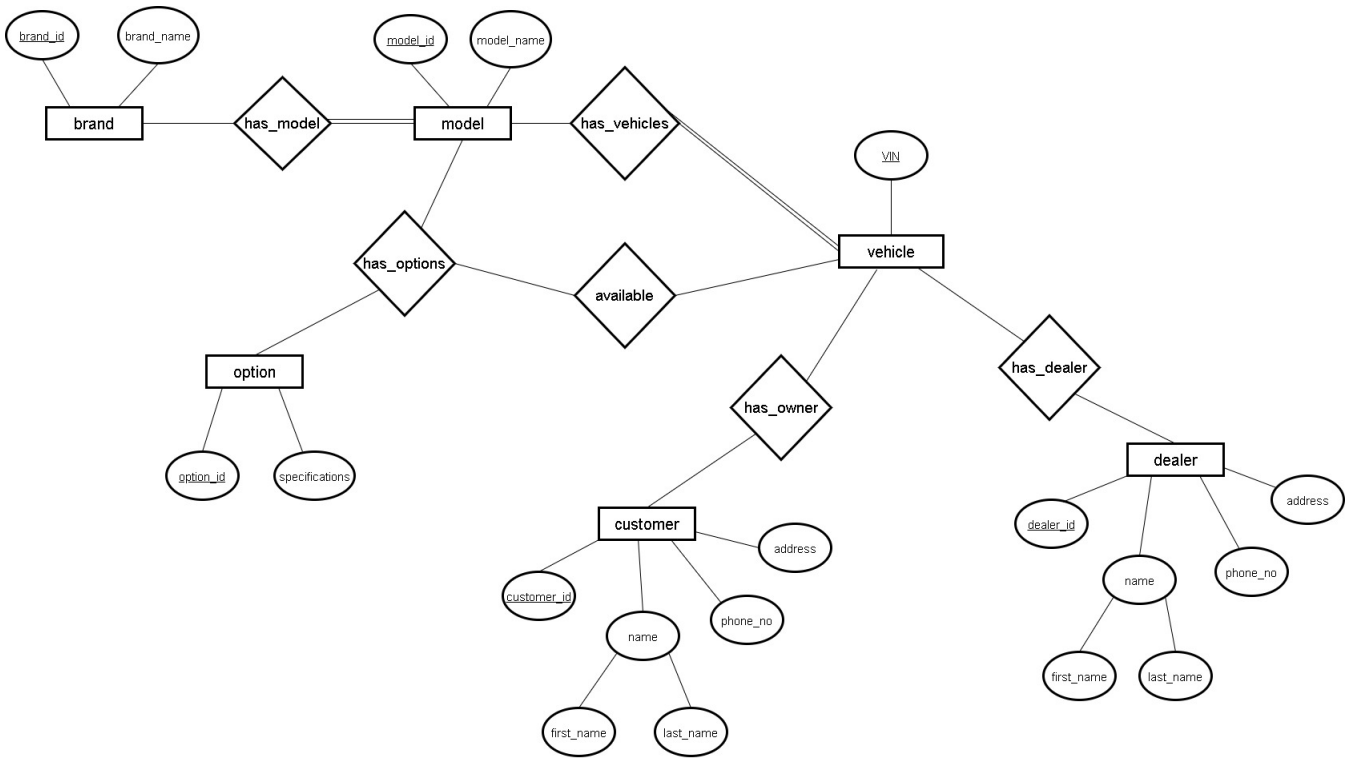

Figure 1: ER Model

1.1 Schema

brand (brand_id, brand_name),
model (model_id, mdoel_name),
option (option_id, specifications),
vehicle (VIN),
customer (customer_id, name, phone_no, address),
dealer (dealer_id, name, phone_no, address),
has_model (brand_id ref. **brand**, model_id ref. **model**),
has_options (model_id ref. **model**, option_id ref. **option**),
has_vehicles (model_id ref. **model**, VIN ref. **vehicle**),
available ((model_id, option_id) ref. **has_options**, VIN ref. **vehicle**),
has_owner (VIN ref. **vehicle**, customer_id ref. **customer**),
has_dealer (VIN ref. **vehicle**, dealer_id ref. **dealer**)

2 Problem 2

Design a database for a world-wide package delivery company (e.g., DHL or Fed EX). The database must be able to keep track of customers (who ship items) and customers (who receive items); some customers may do both. Each package must be identifiable and trackable, so the database must be able to store the location of the package and its history of locations. Locations include trucks, planes, airports, and warehouses. Your design should include an E-R diagram, a set of relational schemas, and a list of constraints, including primary-key and foreign-key constraints.

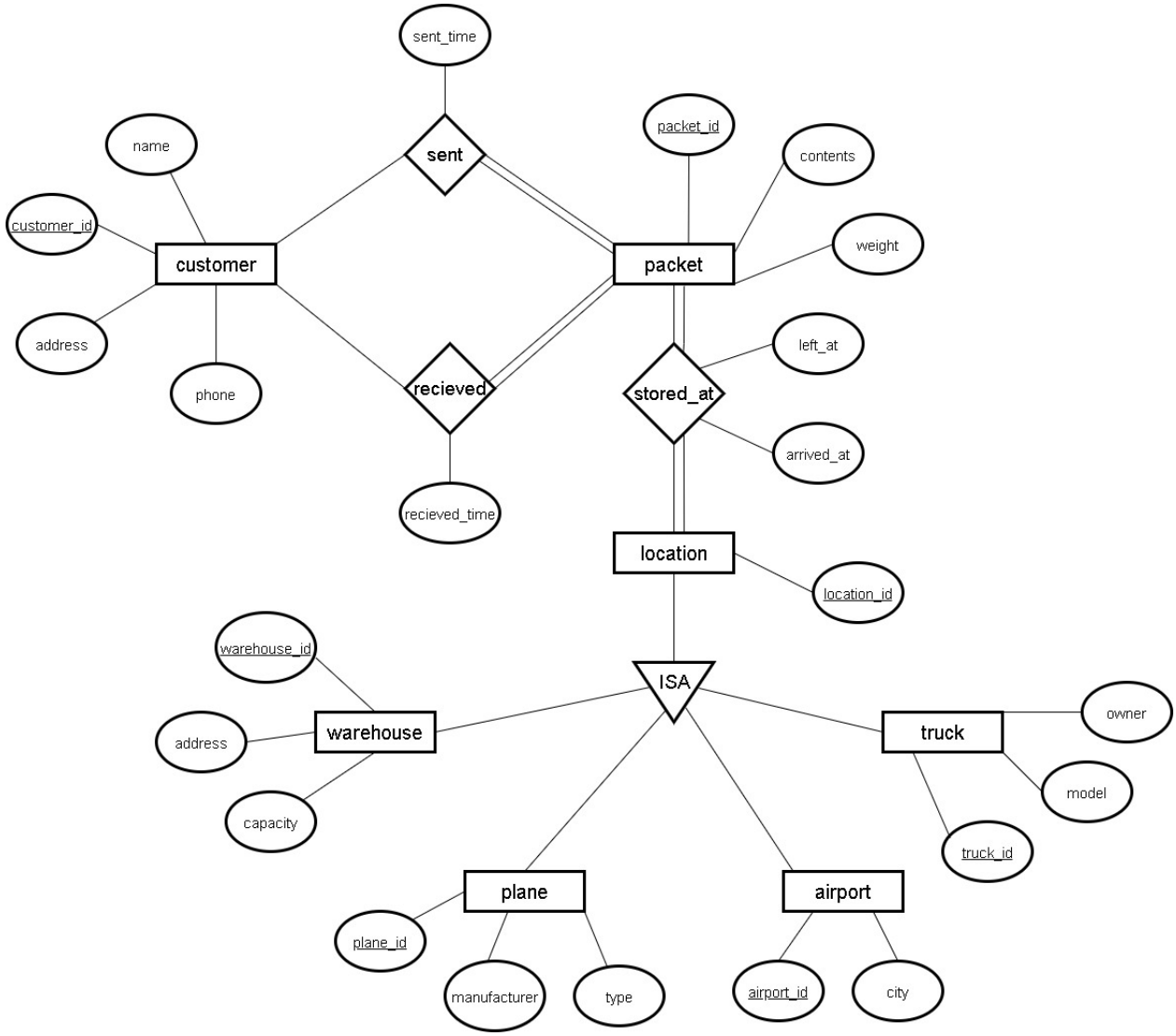

Figure 2: ER Model

2.1 Schema

customer (customer_id, name, phone, address),
packet (packet_id, contacts, weight),
location (location_id),
warehouse (warehouse_id ref. **location(location_id)**, address, capacity),
plane (plane_id ref. **location(location_id)**, manufacturer, type),
airport (airport_id ref. **location(location_id)**, city),
truck (truck_id ref. **location(location_id)**, model, owner),
sent (customer_id ref. **customer**, packet_id ref. **packet**, sent_time),
recived (customer_id ref. **customer**, packet_id ref. **packet**, recieved_time),
stored_at (packet_id ref. **packet**, location_id ref. **location**, arrived_at, left_at),

3 Problem 3

Since the HTML page will be publicly available to everyone as it is client-side, a user can edit the HTML page's source code to replace the old price value with the new one. The server would not know whether the user modified the price value and would place the order using the modified price.

4 Problem 4

Even though a link is visible only to an authorized user, an unauthorized user may somehow know of the URL's existence (for example, via Web history logs). The user may then log in to the system and access the unauthorized page by entering its URL in the browser. If the check for user authorization were left out from that page, the user would be able to see the result of the page. The HTTP referer attribute can be used to block such loopholes by ensuring the referer value is from a valid page of the Web site. However, the browser sets the referrer attribute, so a malicious user can quickly work around the referer check.

5 Problem 5

HTTPS protocol is used to prevent multiple types of attacks, by ensuring that the HTTP request being sent is encrypted using public / private key encyption and also whether the request is going to the intended user using digital certificates.

1. HTTPS encrypts the request sent from the end user, which can only be decrypted by server. This prevents hackers from viewing or modifying the request on route.
2. Buying an SSL certificate from a verified company, allows the company to sign the public key. This SSL certificate can be used by the website to show users that it is actually the website that they are trying to access and not a malicious one.
3. If the website has an SSL certificate it will have **https** pad-lock.

6 Problem 6

The login HTML is shown below.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Login</title>
  </head>
<body>
  <form action="LoginServlet" method="post">
    User ID: <input type="text" name="userid"> <br>
    Password: <input type="password" name="password"> <br>
    <input type="submit" value="Login">
  </form>
</body>
</html>
```

The LoginServlet used to handle login, is shown below.

```
@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {
  private static final long serialVersionUID = 1L;

  protected void doPost(HttpServletRequest request,
      HttpServletResponse response) throws ServletException, IOException {
    try{
      // Getting the request parameters
      String userid = request.getParameter("userid");
      String password = request.getParameter("password");

      // Connecting to mysql databse
      Connection con = null;
      String url = "jdbc:mysql://localhost:3306/users"; //MySQL URL and followed by
  the database name
      String username = "root"; //MySQL username
      String password = "password"; //MySQL password
      Class.forName("com.mysql.jdbc.Driver");
      con = DriverManager.getConnection(url, username, password);

      // Create Prepaared Statement
      PreparedStatement check = con .prepareStatement("select userid, password from
  data where userid=? and password=?");
          check.setString(1, userid);
          check.setString(2, password);

      // Execute the query
      ResultSet rs = check.executeQuery();

      // Check output of query
```

```
29          if(rs.isBeforeFirst()){
30            while(rs.next()){
31              // Creating a session cookie
32              Cookie loginCookie = new Cookie("userid", rs.getString("userid"));
33
34              // Setting cookie to expiry
35              loginCookie.setMaxAge(30*60);
36
37              // Adding the cookie to the response
38              response.addCookie(loginCookie);
39              response.sendRedirect("LoginSuccess.jsp");
40            }
41          }else{
42
43            // Login Error if user or password do not match
44            RequestDispatcher rd = request.getRequestDispatcher("LoginError.jsp");
45            rd.forward(request, response);
46          }
47        }catch(Exception e) {
48
49          // General Exception Handling.
50          e.printStackTrace();
51          RequestDispatcher rd = request.getRequestDispatcher("Error.jsp");
52          rd.forward(request, response);
53        }
54
55      }
56
57    }
```

This will set a session cookie named **userid** after the user logs in successfully, but gives an error in all other cases.

7 Problem 7

XSS or Cross-Site Scripting is a client side code injection attack. In this type of attack, the malicious user tries to inject code into legitimate websites. These websites then become a carrier of malicious code, which effects other users who use the website.

Most common place of injections include message boards, forums, discussion websites, text input output based services. The attack can occur if the server does not filter the request for malicious code and tries to render the text given as input without checking for hidden code.

Ways of preventing an XSS attack are

1. Secuirty training and awareness needs do be given to the developers, ops team, etc., before they start building an application.
2. Do not trust any input that comes from the user, requests needs to go through a trusted filtering process.
3. Using text formatters such as prepareStatement, string formatter, etc., to execute user inputs.
4. Parse and clean the HTML page to be returned using trusted libraries.
5. Use HttpOnly cookies which prevent client side users from modifying cookies.
6. Content Security Policy allows you to declare where and how a dynamic code snippet can be executed.