

Databases & Web Services Project 2024

Illia Shust & Kinlo Ephriam Tangier, Constructor University

2024-Sep-11

1 Project Overview

The **Sustainable Shopping Assistant** is a web service aimed at promoting eco-friendly shopping habits by providing users with environmental ratings of products. The service helps users make informed decisions by showing the carbon footprint, eco-ratings, and sustainability certifications of products in various categories ranging from electronics, groceries to clothing.

Functional Overview

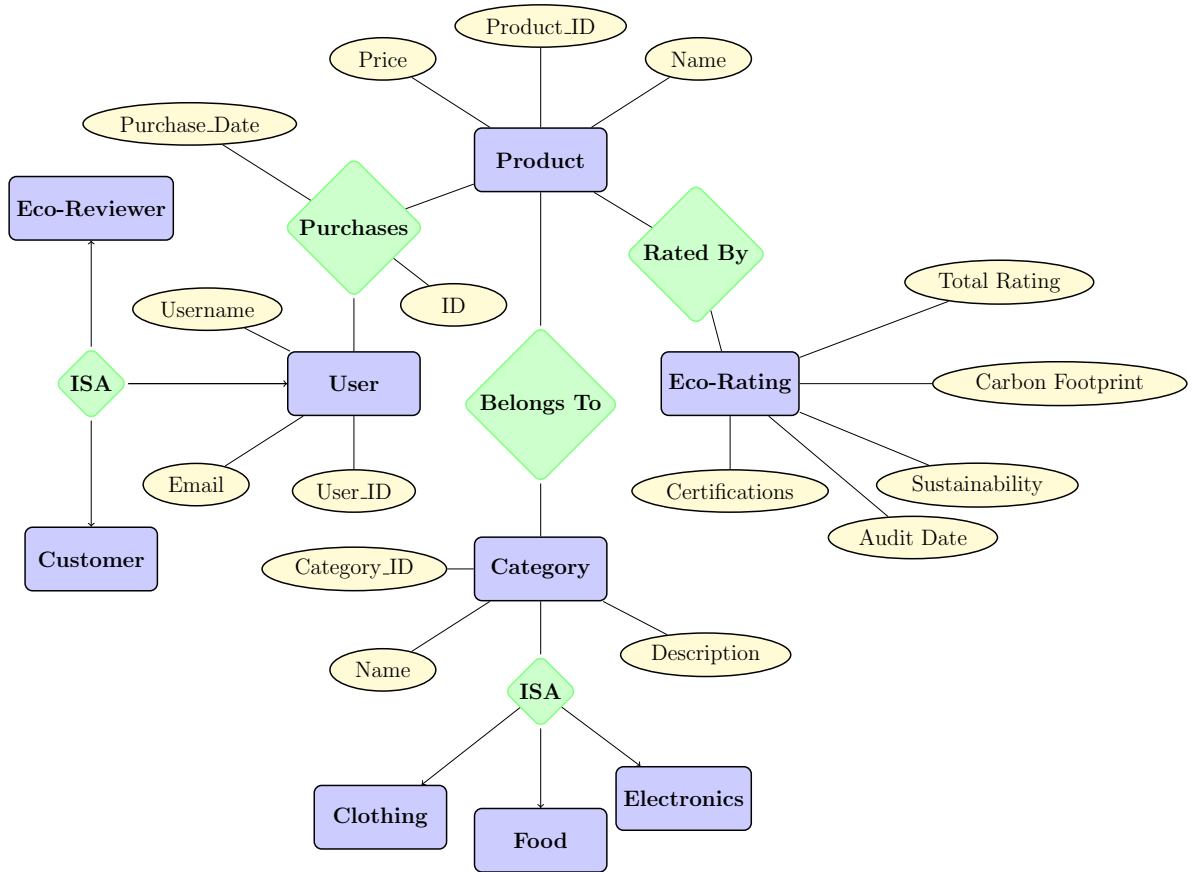
From the user's perspective, the system allows the following actions:

- **Search for products by category** (e.g., electronics, clothing, groceries).
- **View detailed eco-ratings**, including carbon footprint, environmental certifications, and sustainability scores.
- **Compare products** based on their environmental impact and find alternatives.
- **Set preferences** to receive suggestions for eco-friendly versions of desired products (e.g., plastic-free, organic).
- **Add products to a wishlist** and access seen eco-friendly alternatives.
- **Rate products** based on their perceived eco-friendliness.

The service aims to make the process of environmentally conscious shopping easier for the general public while encouraging sustainable consumer habits.

2 ER Diagram of the Miniworld

The ER diagram below includes the main entities involved in the system. For a team of two ($N = 2$), the system includes two "ISA" hierarchies: one for the products categories and another for users (customers and eco-reviewers).



3 User Interactions

Below is an outline of how the user will interact with the system:

- **Search Products:** Users can search for products by entering a keyword or filtering by categories (e.g., electronics, food, clothing, etc.).
- **View Eco-Scores:** When users view a product, they are presented with an overall "Eco-Score," which includes details like carbon footprint, sustainability certifications (e.g., Fair Trade, Organic), and material composition.
- **Set Preferences:** Users can input eco-friendly preferences, such as seeking products that are plastic-free, organic, or have a minimal carbon footprint. The system will recommend relevant options accordingly.
- **Add to Wishlist:** Users can add eco-friendly products to their wishlist for future reference.
- **Rate a Product:** Eco-reviewers can rate products they have used on a scale from 1-10 based on sustainability factors. Users with sufficient credibility can add reviews and scores.
- **Error Handling (Invalid Actions):**
 - Searching for a non-existent product will result in a friendly message indicating no such product is found.

- Users attempting to unfairly rate a product multiple times will be shown a restriction message.
- Invalid eco-preference selections (e.g., choosing incompatible preferences) will prompt the user to adjust their input.

4 SQL Schema

4.1 Product Table

```
CREATE TABLE Product (  
    product_id INT PRIMARY KEY,  
    price DECIMAL(10, 2),  
    name VARCHAR(255)  
);
```

This table stores product details. Each product has a unique `product_id`, a `price`, and a `name`.

4.2 User Table

```
CREATE TABLE User (  
    user_id INT PRIMARY KEY,  
    username VARCHAR(255),  
    email VARCHAR(255),  
    role ENUM('Customer', 'EcoReviewer') NOT NULL  
);
```

The `User` table includes both customers and eco-reviewers. The `role` field determines the user type. This is part of our inheritance strategy (explained below).

4.3 Category Table

```
CREATE TABLE Category (  
    category_id INT PRIMARY KEY,  
    name VARCHAR(255),  
    description VARCHAR(255),  
    category_type ENUM('Clothing', 'Food', 'Electronics') NOT NULL  
);
```

This table defines categories for products, with `category_type` distinguishing between different product types.

4.4 EcoRating Table

```
CREATE TABLE EcoRating (  
    eco_rating_id INT PRIMARY KEY,  
    total_rating INT CHECK (total_rating BETWEEN 1 AND 100),  
    certification VARCHAR(255),  
    audit_date DATE,
```

```

        sustainability INT,
        carbon_footprint INT
    );

```

The EcoRating table stores sustainability-related metrics for products.

4.5 Purchases Table

```

CREATE TABLE Purchases (
    purchase_id INT PRIMARY KEY AUTO_INCREMENT,
    purchase_date DATE,
    user_id INT,
    product_id INT,
    FOREIGN KEY (user_id) REFERENCES User (user_id),
    FOREIGN KEY (product_id) REFERENCES Product (product_id)
);

```

This table records purchases made by users. The many-to-many relationship between users and products is captured by linking `user_id` and `product_id`.

4.6 Rated By Table

```

CREATE TABLE RatedBy(
    product_id INT UNIQUE,
    total_rating INT,
    PRIMARY KEY(product_id, total_rating),
    FOREIGN KEY (product_id) REFERENCES Product (product_id),
    FOREIGN KEY (total_rating) REFERENCES EcoRating (total_rating)
);

```

This table handles the many-to-one relationship between products and eco-ratings, where each product can have only one eco-rating.

4.7 Belongs To Table

```

CREATE TABLE BelongsTo(
    product_id INT UNIQUE,
    category_id INT,
    PRIMARY KEY (product_id, category_id),
    FOREIGN KEY (product_id) REFERENCES Product (product_id),
    FOREIGN KEY (category_id) REFERENCES Category (category_id)
);

```

This table defines the many-to-one relationship between products and categories. Each product belongs to exactly one category.

5 Inheritance Strategy: Alt 3 (One Big Relation)

In this design, we adopted **Alt 3: One Big Relation** for handling inheritance. Instead of creating separate tables for the `Customer` and `EcoReviewer`, we used a single `User`

table with an `ENUM` field, `role`, to distinguish between the two types. This approach was chosen for the following reasons:

1. All users, regardless of their type, are stored in a single table. This reduces the complexity of queries, as there is no need for joins between multiple user tables.
2. Since most operations (like purchases) involve both customers and eco-reviewers, this strategy eliminates the overhead of joins and simplifies data retrieval.
3. While some inheritance strategies may introduce many `NULL` fields in a large table, our design minimizes this by ensuring that all users share common fields, and the `role` column handles user type distinctions.

6 Mapping Approach

- **One-to-Many (1:n) Relationship:** Many products can belong to the same category, but each product belongs to exactly one category. This is captured by the `BelongsTo` table.
- **Many-to-One (n:1) Relationship:** Each product can have only one eco-rating, but multiple products can share the same eco-rating. This is represented by the `RatedBy` table.
- **Many-to-Many (n:n) Relationship:** Many users can purchase many products. The `Purchases` table captures this relationship between `User` and `Product`.