Documentation: ER Diagram Explanation

This documentation explains the Entity-Relationship (ER) diagram representing a database design for an application that involves users, products, carts, and orders. The diagram shows the entities, their attributes, and the relationships among them, including the use of joint tables to manage many-to-many relationships.

Entities and Their Attributes

1. **USERS**
    - **Attributes:**
        - user_id (int, Primary Key): Unique identifier for each user.
        - username (String): The username of the user.
        - password (String): The user's password.
        - email (String): The user's email address.
        - name (String): The user's full name.
        - role (Role): The role of the user, which can be used to distinguish between different user types (e.g., admin, customer).
        - cart_cart_id (int, Foreign Key): Reference to the CART entity to establish a one-to-one relationship between a user and their cart.
2. **PRODUCT**
    - **Attributes:**
        - product_id (int, Primary Key): Unique identifier for each product.
        - name (String): The name of the product.
        - price (double): The price of the product.
        - quantity (int): The available quantity of the product.
3. **ORDERS**
    - **Attributes:**
        - order_id (int, Primary Key): Unique identifier for each order.
        - product_list (List<Order>): A list of products associated with the order.
        - users_user_id (int, Foreign Key): Reference to the USERS entity to establish a one-to-many relationship between users and orders.
        - orderTotalPrice (double): The total price of the order.
        - deliveryAddress (String): The delivery address for the order.
4. **CART**
    - **Attributes:**

        - cart_cart_id (int, Primary Key): Unique identifier for each cart.
        - users_user_id (int, Foreign Key): Reference to the USERS entity to establish a one-to-one relationship between a user and their cart.

Joint Tables for Many-to-Many Relationships

5. **CART_PRODUCT_LIST**
    - **Purpose:** This joint table represents a many-to-many relationship between the CART and PRODUCT entities. A cart can contain multiple products, and a product can belong to multiple carts.
    - **Attributes:**
        - cart_cart_id (int, Foreign Key): Reference to the CART entity.
        - product_list_product_id (int, Foreign Key): Reference to the PRODUCT entity.
6. **ORDERS_PRODUCT_LIST**
    - **Purpose:** This joint table represents a many-to-many relationship between the ORDERS and PRODUCT entities. An order can include multiple products, and a product can be part of multiple orders.
    - **Attributes:**
        - orders_order_id (int, Foreign Key): Reference to the ORDERS entity.
        - product_list_product_id (int, Foreign Key): Reference to the PRODUCT entity.
7. **USERS_ORDERS_LIST**
    - **Purpose:** This joint table represents a many-to-many relationship between the USERS and ORDERS entities. A user can place multiple orders, and each order is associated with one user.
    - **Attributes:**
        - users_user_id (int, Foreign Key): Reference to the USERS entity.
        - orders_list_order_id (int, Foreign Key): Reference to the ORDERS entity.

Relationships

8. **One-to-One Relationship between USERS and CART**
    - A user has one cart, and each cart is associated with exactly one user. This relationship is established using the foreign key users_user_id in the CART table.
9. **One-to-Many Relationship between USERS and ORDERS**

- A user can place multiple orders, but each order is associated with exactly one user. This relationship is established using the foreign key users_user_id in the ORDERS table.

10. **Many-to-Many Relationship between CART and PRODUCT**
    - A cart can contain multiple products, and a product can be part of multiple carts. This relationship is managed by the CART_PRODUCT_LIST joint table.
11. **Many-to-Many Relationship between ORDERS and PRODUCT**
    - An order can have multiple products, and a product can be included in multiple orders. This relationship is managed by the ORDERS_PRODUCT_LIST joint table.
12. **Many-to-Many Relationship between USERS and ORDERS**
    - A user can have multiple orders, and each order is associated with a user. This relationship is managed by the USERS_ORDERS_LIST joint table.

Conclusion

The ER diagram effectively represents the relationships between users, products, carts, and orders, using joint tables to handle the many-to-many relationships. This design is suitable for applications such as e-commerce platforms where users can manage their carts, place orders, and view products.