**PRODUCT RETURN SYSTEM**

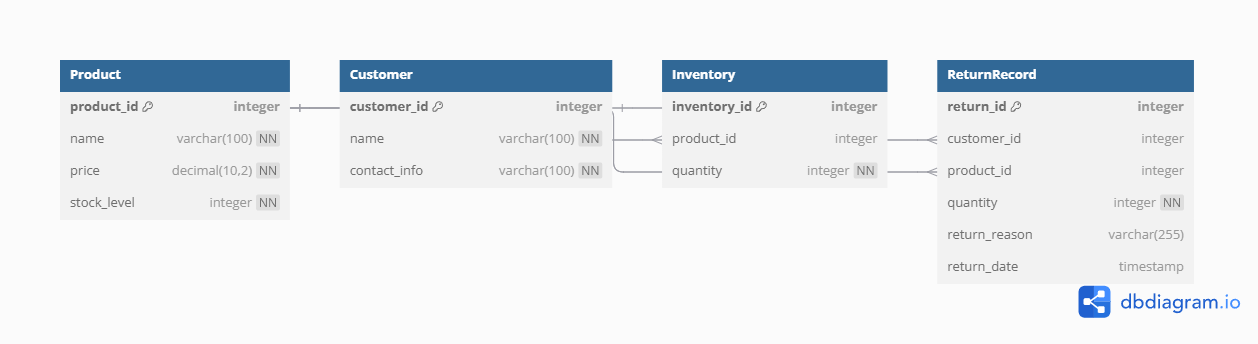
**Abstract:**

The \*Product Return and Restocking Management System\* is a console-based Java application designed to streamline the process of managing product returns, inspecting returned items, and restocking inventory within a retail environment. Leveraging a structured object-oriented approach, this system uses key classes, such as `Return`, `Product`, `Customer`, and `Inventory`, to model entities and operations involved in product returns and stock adjustments. The system integrates a MySQL database for persistent storage of product details, customer information, and transaction history, ensuring data integrity and scalability.

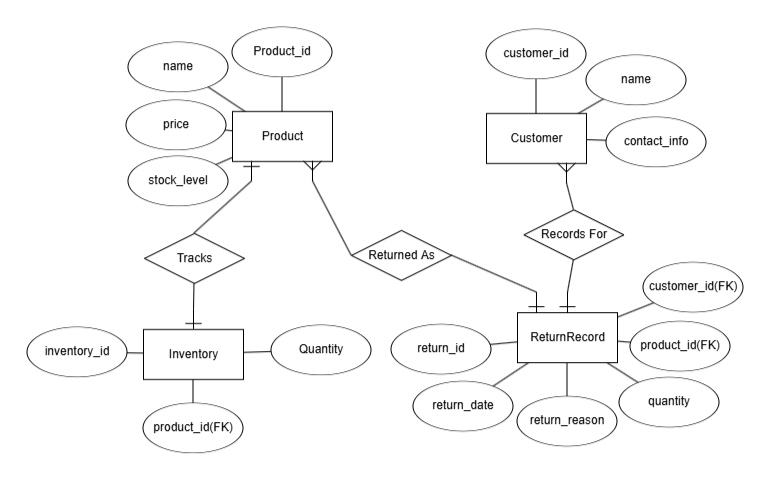
The system supports both `CustomerReturn` and `StoreReturn` types, encapsulating unique requirements for each return type through inheritance. An interface, `Returnable`, defines essential methods such as `processReturn()`, `updateStock()`, and `generateReturnReport()`, ensuring flexibility for further enhancements. Custom exceptions like `InvalidReturnException` and `StockUpdateFailureException` are implemented to handle error conditions robustly, enhancing reliability in various transaction scenarios.In addition to core return and restocking functionality, the system supports concurrent return processing using multithreading, enabling efficient handling of multiple returns simultaneously. File handling is also employed to log return activities, providing an audit trail for analysis. Comprehensive reporting capabilities are included, allowing the generation of return reports that summarize returned items, customer details, reasons for return, and dates. These reports facilitate management insights into return trends, supporting operational and inventory decisions.

This project demonstrates the integration of Java, MySQL, and multithreading in a practical application for inventory management, offering a scalable, efficient, and secure solution for return and restocking operations.

**Schema Diagram:**

****

**ER Diagram:**

****

**Query:**

**-- Create Database**

**CREATE DATABASE ProductReturnSystem;**

**USE ProductReturnSystem;**

**-- Create Product Table**

**CREATE TABLE Product (**

**product\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**name VARCHAR(100) NOT NULL,**

**price DECIMAL(10, 2) NOT NULL,**

**stock\_level INT NOT NULL,**

**UNIQUE (name)**

**);**

**-- Create Customer Table**

**CREATE TABLE Customer (**

**customer\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**name VARCHAR(100) NOT NULL,**

**contact\_info VARCHAR(100) NOT NULL,**

**UNIQUE (contact\_info)**

**);**

**-- Create Inventory Table**

**CREATE TABLE Inventory (**

**inventory\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**product\_id INT NOT NULL,**

**quantity INT NOT NULL,**

**FOREIGN KEY (product\_id) REFERENCES Product(product\_id)**

**);**

**-- Create ReturnRecord Table**

**CREATE TABLE ReturnRecord (**

**return\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**customer\_id INT NOT NULL,**

**product\_id INT NOT NULL,**

**quantity INT NOT NULL,**

**return\_reason VARCHAR(255),**

**return\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,**

**FOREIGN KEY (customer\_id) REFERENCES Customer(customer\_id),**

**FOREIGN KEY (product\_id) REFERENCES Product(product\_id)**

**);**

**-- Create Stored Procedure for Return Processing**

**DELIMITER //**

**CREATE PROCEDURE ProcessReturn(**

**IN cust\_id INT,**

**IN prod\_id INT,**

**IN qty INT,**

**IN reason VARCHAR(255)**

**)**

**BEGIN**

**DECLARE current\_stock INT;**

**SELECT stock\_level INTO current\_stock FROM Product WHERE product\_id = prod\_id;**

**IF current\_stock IS NULL THEN**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Product not found';**

**ELSEIF current\_stock < qty THEN**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Insufficient stock for return';**

**ELSE**

**-- Update Product stock**

**UPDATE Product SET stock\_level = stock\_level + qty WHERE product\_id = prod\_id;**

**-- Insert return record**

**INSERT INTO ReturnRecord (customer\_id, product\_id, quantity, return\_reason)**

**VALUES (cust\_id, prod\_id, qty, reason);**

**END IF;**

**END;**

**//**

**DELIMITER ;**

**-- Create Trigger to Update Inventory After Return**

**DELIMITER //**

**CREATE TRIGGER UpdateInventoryAfterReturn**

**AFTER INSERT ON ReturnRecord**

**FOR EACH ROW**

**BEGIN**

**UPDATE Inventory**

**SET quantity = quantity + NEW.quantity**

**WHERE product\_id = NEW.product\_id;**

**END;**

**//**

**DELIMITER ;**

**-- Create View for Return History**

**CREATE VIEW ReturnHistory AS**

**SELECT r.return\_id, c.name AS customer\_name, p.name AS product\_name, r.quantity, r.return\_reason, r.return\_date**

**FROM ReturnRecord r**

**JOIN Customer c ON r.customer\_id = c.customer\_id**

**JOIN Product p ON r.product\_id = p.product\_id;**

**-- Create Indexes for Performance**

**CREATE INDEX idx\_product\_id ON Product(product\_id);**

**CREATE INDEX idx\_customer\_id ON Customer(customer\_id);**

**CREATE INDEX idx\_return\_id ON ReturnRecord(return\_id);**

**UPDATE Inventory SET quantity = ? WHERE product\_id = ?;**

**SELECT \* FROM Customer;**

**CREATE TABLE AuditLog (**

**log\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**customer\_id INT NOT NULL,**

**product\_id INT NOT NULL,**

**quantity INT NOT NULL,**

**reason VARCHAR(255),**

**log\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,**

**FOREIGN KEY (customer\_id) REFERENCES Customer(customer\_id),**

**FOREIGN KEY (product\_id) REFERENCES Product(product\_id)**

**);**

**INSERT INTO Product (name, price, stock\_level) VALUES**

**('Product A', 10.99, 50),**

**('Product B', 25.50, 30),**

**('Product C', 15.75, 20);**

**INSERT INTO Customer (name, contact\_info) VALUES**

**('John Doe', 'john@example.com'),**

**('Jane Smith', 'jane@example.com');**

**INSERT INTO Inventory (product\_id, quantity) VALUES**

**(1, 50),**

**(2, 30),**

**(3, 20);**