PRODUCT RETURN SYSTEM

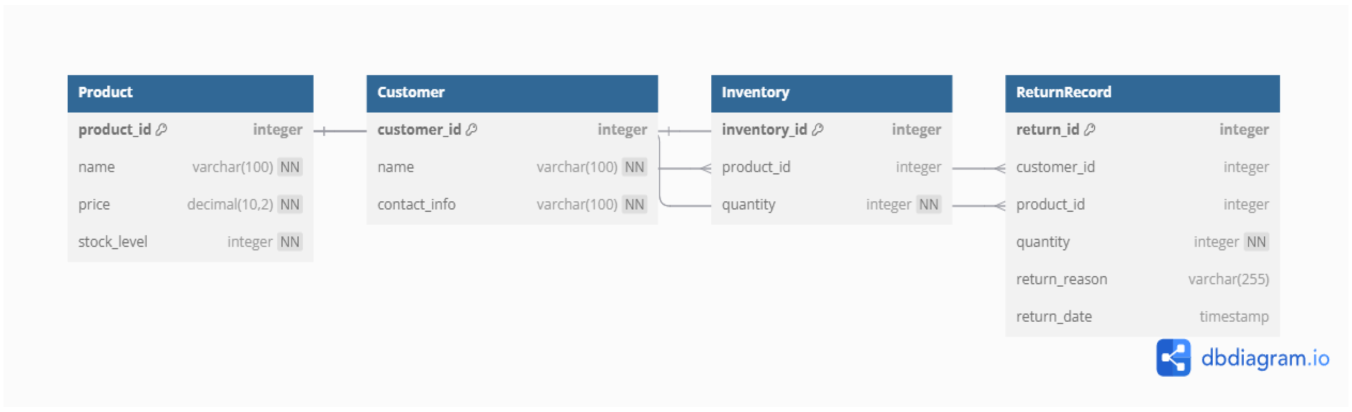
Abstract:

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.

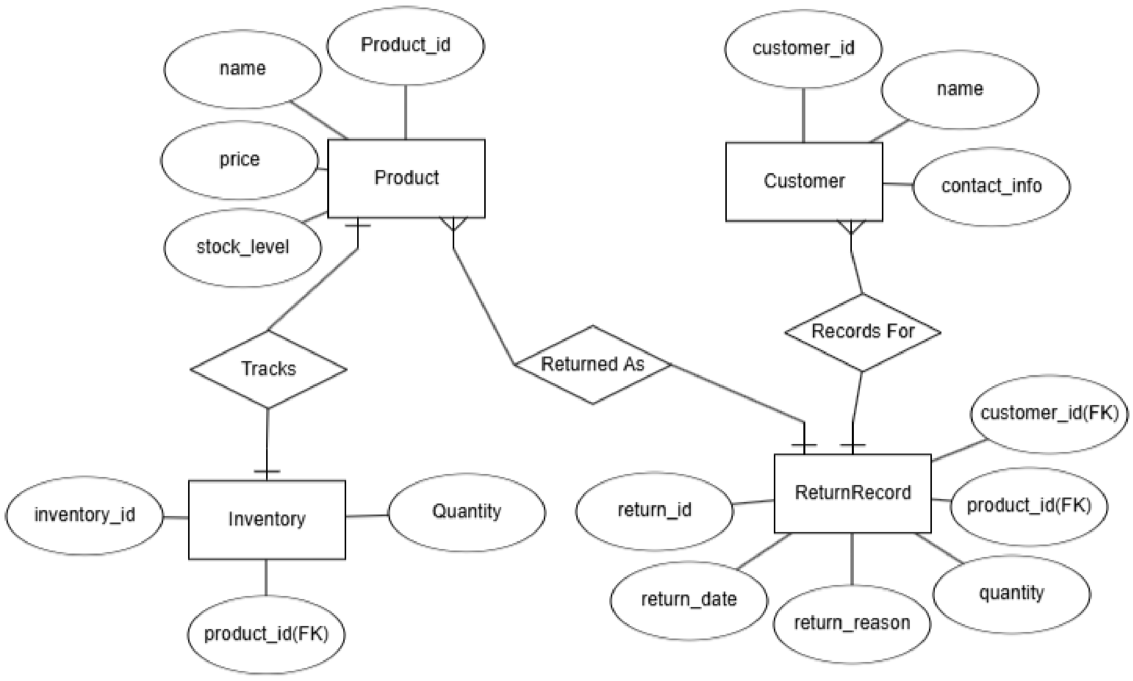
The system supports both `CustomerReturn` and `StoreReturn` types, encapsulating unique requirements for each return type through inheritance. An interface, `Returnable`, defines essential methods such as `processReturn()`, `updateStock()`, and `generateReturnReport()`, ensuring flexibility for further 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);
```