DATABASE MANAGEMENT SYSTEM PROJECT REPORT

FARMER-CUSTOMER PRODUCT MANAGEMENT SYSTEM

SUBMITTED TO:
REAYA GREWAL

SUBMITTED BY:

KRISH - 102303688
SHUBHAM KANSAL - 102303689
MOHAK SETH - 102303690
DHANVEER SINGH - 102303685

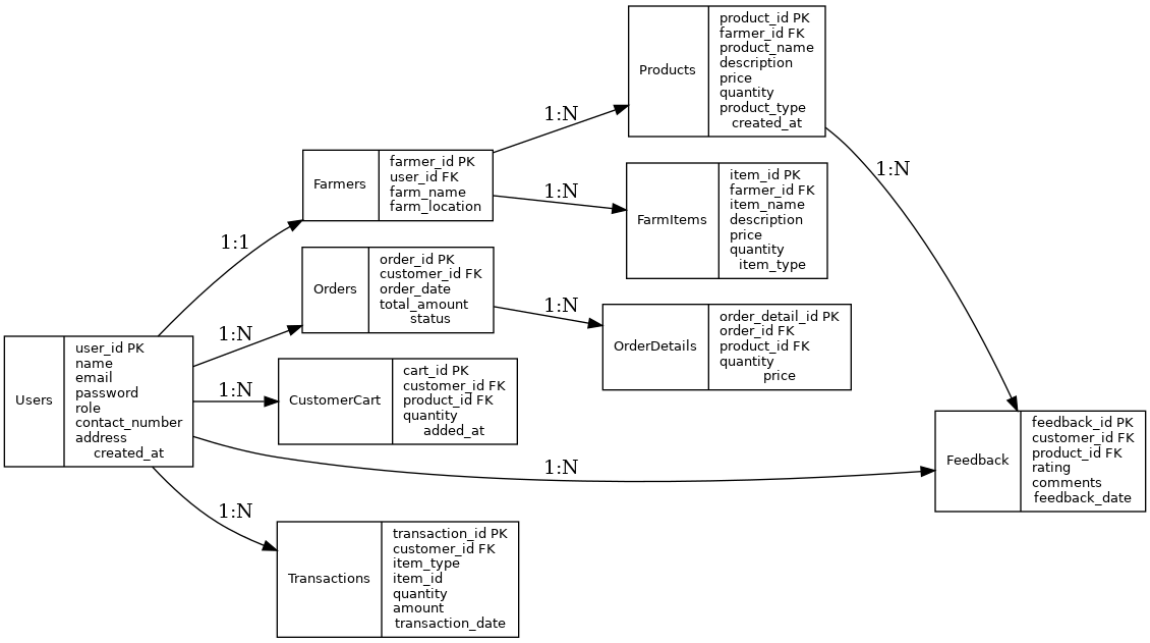
INDEX

INTRODUCTION

This project, Farmer-Customer Product Management System, is designed to enable a direct connection between farmers and customers through a well-structured database. It facilitates the sale of farm products by farmers, the placing of orders by customers, and tracks all transactions, feedback, and product listings efficiently.

The system reduces the gap between food producers and consumers by managing essential data using SQL-based relational tables. Users can be either farmers or customers, each interacting with the system differently. Farmers can list products, while customers can browse, order, give feedback, and track transactions.

ER-DIAGRAM

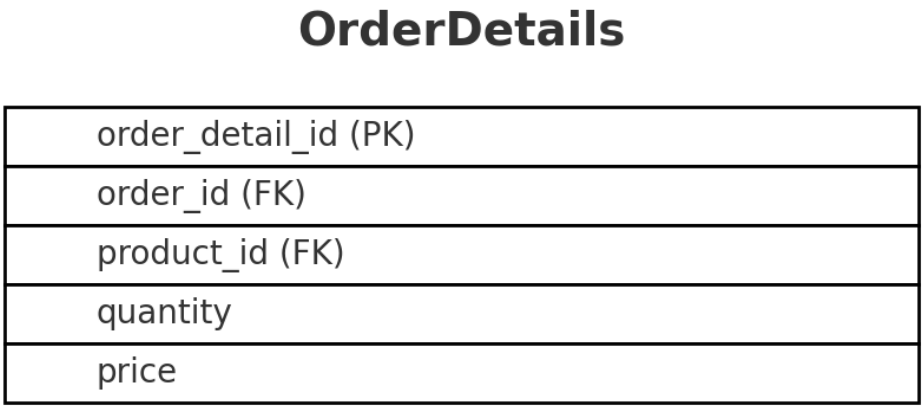


ER Diagram Representation:

The Entity Relationship Diagram for this system consists of the following entities:
1. Users – stores basic information for both farmers and customers.
2. Farmers – represents a subset of users who list products.
3. Products – items listed by farmers.
4. Orders and OrderDetails – represent purchases made by customers.
5. CustomerCart – stores pre-order selections.
6. Feedback – allows customers to review products.
7. FarmItems – fertilizers and machinery.
8. Transactions – tracks all purchases.

Relationships:
- One-to-One between Users and Farmers
- One-to-Many between Farmers and Products
- Many-to-Many between Orders and Products via OrderDetails
- One-to-Many between Users and Orders, Feedback, Cart, Transactions

ER TO TABLE CONVERSION



OrderDetails Table Schema:

Orders Table Schema:

Products

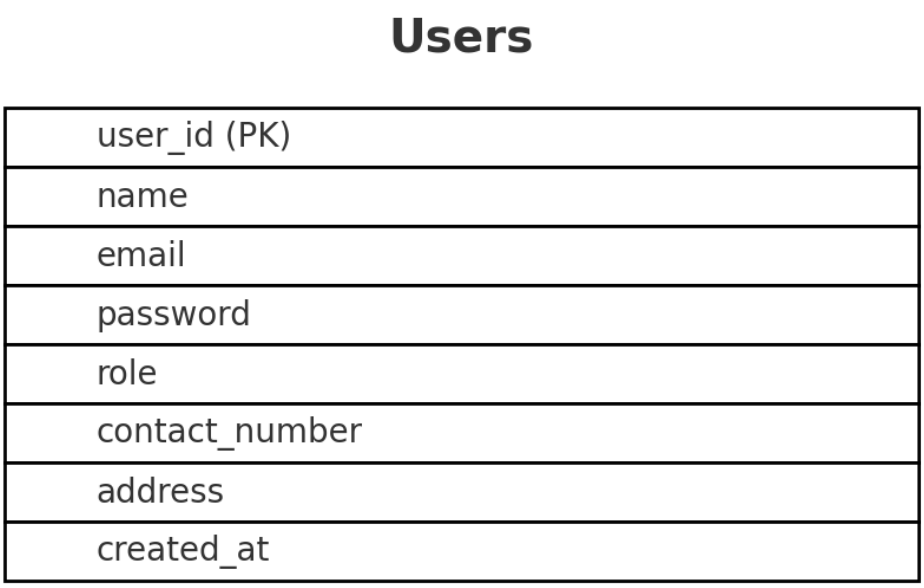
product_id (PK)
farmer_id (FK)
product_name
description
price
quantity
product_type
created_at

Products Table Schema:

Farmers

farmer_id (PK)
user_id (FK)
farm_name
farm_location

Farmers Table Schema:



Users Table Schema:

Each entity has been mapped to a table. Below are the tables derived from the ER model:

- - Users(user_id, name, email, password, role, contact_number, address, created_at)
- - Farmers(farmer_id, user_id, farm_name, farm_location)
- - Products(product_id, farmer_id, product_name, description, price, quantity, product_type, created_at)
- - Orders(order_id, customer_id, order_date, total_amount, status)
- - OrderDetails(order_detail_id, order_id, product_id, quantity, price)
- - CustomerCart(cart_id, customer_id, product_id, quantity, added_at)
- - Feedback(feedback_id, customer_id, product_id, rating, comments, feedback_date)
- - FarmItems(item_id, farmer_id, item_name, description, price, quantity, item_type)
- - Transactions(transaction_id, customer_id, item_type, item_id, quantity, amount, transaction_date)

NORMALIZATION

Normalization helps reduce data redundancy and improve data integrity. The Farmer-Customer system adheres to 1NF, 2NF, and 3NF:

1NF: Each table contains atomic values with unique entries.
2NF: All non-key attributes are fully functionally dependent on the primary key.
3NF: There is no transitive dependency among non-key attributes.

Example:
- Users Table: All columns depend on user_id.
- Products Table: All attributes are fully dependent on product_id.
- Orders and OrderDetails eliminate redundancy by separating order metadata and items.

SQL CODE

Following are the SQL statements used for database creation:

Users Table

```
CREATE TABLE Users (
   user_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
   name VARCHAR2(100) NOT NULL,
   email VARCHAR2(100) NOT NULL UNIQUE,
   password VARCHAR2(255) NOT NULL,
   role VARCHAR2(10) CHECK (role IN ('farmer', 'customer')) NOT NULL,
   contact_number VARCHAR2(15),
   address VARCHAR2(255),
   created_at TIMESTAMP
);
```

Farmers Table

```
CREATE TABLE Farmers (
   farmer_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
   user_id NUMBER NOT NULL,
   farm_name VARCHAR2(100),
   farm_location VARCHAR2(255),
   CONSTRAINT fk_farmer_user FOREIGN KEY (user_id) REFERENCES Users(user_id)
);
```

Products Table

```
CREATE TABLE Products (
   product_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
   farmer_id NUMBER NOT NULL,
   product_name VARCHAR2(100) NOT NULL,
   description VARCHAR2(500),
   price NUMBER(10,2) NOT NULL,
   quantity NUMBER NOT NULL,
   product_type VARCHAR2(20) CHECK (product_type IN ('vegetable', 'fruit', 'other')) NOT
NULL,
   created_at TIMESTAMP,
   CONSTRAINT fk_product_farmer FOREIGN KEY (farmer_id) REFERENCES
Farmers(farmer_id)
);
```

Orders&OrderDetails

```
CREATE TABLE Orders (
   order_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
```

```
  customer_id NUMBER NOT NULL,
  order_date TIMESTAMP,
  total_amount NUMBER(10,2) NOT NULL,
  status VARCHAR2(10) DEFAULT 'pending' CHECK (status IN ('pending', 'completed',
'cancelled')),
  CONSTRAINT fk_orders_customer FOREIGN KEY (customer_id) REFERENCES
Users(user_id)
);

CREATE TABLE OrderDetails (
  order_detail_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  order_id NUMBER NOT NULL,
  product_id NUMBER NOT NULL,
  quantity NUMBER NOT NULL,
  price NUMBER(10,2) NOT NULL,
  CONSTRAINT fk_orderdetails_order FOREIGN KEY (order_id) REFERENCES
Orders(order_id),
  CONSTRAINT fk_orderdetails_product FOREIGN KEY (product_id) REFERENCES
Products(product_id)
);
```

CustomerCart&Feedback

```
CREATE TABLE CustomerCart (
  cart_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  customer_id NUMBER NOT NULL,
  product_id NUMBER NOT NULL,
  quantity NUMBER NOT NULL,
  added_at TIMESTAMP,
  CONSTRAINT fk_cart_customer FOREIGN KEY (customer_id) REFERENCES
Users(user_id),
  CONSTRAINT fk_cart_product FOREIGN KEY (product_id) REFERENCES
Products(product_id)
);

CREATE TABLE Feedback (
  feedback_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  customer_id NUMBER NOT NULL,
  product_id NUMBER NOT NULL,
  rating NUMBER CHECK (rating BETWEEN 1 AND 5) NOT NULL,
  comments VARCHAR2(500),
  feedback_date TIMESTAMP,
  CONSTRAINT fk_feedback_customer FOREIGN KEY (customer_id) REFERENCES
Users(user_id),
```

```
  CONSTRAINT fk_feedback_product FOREIGN KEY (product_id) REFERENCES
Products(product_id)
);
```

FarmItems&Transactions

```
CREATE TABLE FarmItems (
  item_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  farmer_id NUMBER NOT NULL,
  item_name VARCHAR2(100) NOT NULL,
  description VARCHAR2(500),
  price NUMBER(10,2) NOT NULL,
  quantity NUMBER NOT NULL,
  item_type VARCHAR2(20) CHECK (item_type IN ('fertilizer', 'machinery')) NOT NULL,
  CONSTRAINT fk_farmitem_farmer FOREIGN KEY (farmer_id) REFERENCES
Farmers(farmer_id)
);

CREATE TABLE Transactions (
  transaction_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  customer_id NUMBER NOT NULL,
  item_type VARCHAR2(20) CHECK (item_type IN ('product', 'fertilizer', 'machinery')) NOT
NULL,
  item_id NUMBER NOT NULL,
  quantity NUMBER NOT NULL,
  amount NUMBER(10,2) NOT NULL,
  transaction_date TIMESTAMP,
  CONSTRAINT fk_transaction_customer FOREIGN KEY (customer_id) REFERENCES
Users(user_id)
);
```

TRIGGERS&SAMPLE DATA

Sample Trigger: Logging New Users

```
CREATE OR REPLACE TRIGGER trg_after_user_insert
AFTER INSERT ON Users
FOR EACH ROW
BEGIN
   INSERT INTO LogTable (action_type, description, action_date)
   VALUES ('INSERT', 'New user added with ID ' || :NEW.user_id, SYSTIMESTAMP);
END;
/
```

Sample Insertions

```
INSERT INTO Users (name, email, password, role, contact_number, address)
VALUES ('John Doe', 'john@example.com', 'pass123', 'farmer', '9876543210', 'Village
Road');

INSERT INTO Farmers (user_id, farm_name, farm_location)
VALUES (1, 'Green Fields', 'Punjab');

INSERT INTO Products (farmer_id, product_name, description, price, quantity,
product_type)
VALUES (1, 'Tomatoes', 'Fresh organic tomatoes', 25.50, 100, 'vegetable');
```

CONCLUSION

This project aims to streamline the connection between farmers and customers using a robust and well-structured database system. It allows for effective management of product listings, order tracking, transactions, and customer feedback. By implementing normalization and relational design, the database ensures data integrity, scalability, and usability. This system is not only practical but promotes direct agriculture-market linkage.

REFERENCES

- Oracle SQL Documentation
- https://www.w3schools.com/sql/
- Class Notes and Lab Work