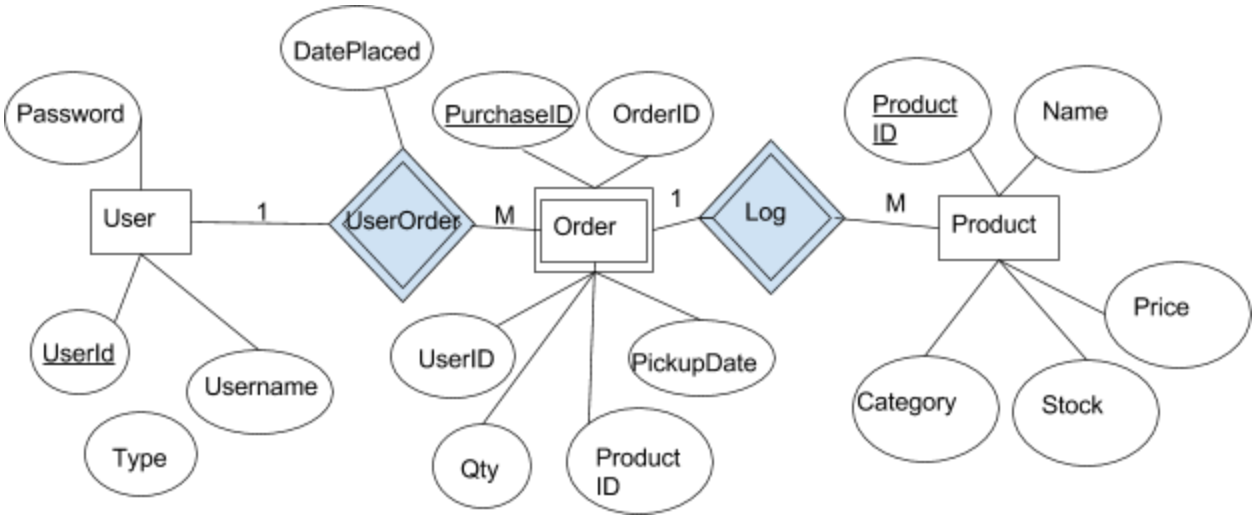
CSC40_Group:
Monica Sproul
Colin Widner
Bijan Anjavi
Alex Yee

Conceptual Database Design



Conceptual database design: Final ER diagram along with your design rationale and any necessary high-level text description of the data model (e.g., constraints or anything not able to show in the ER diagram but is necessary to help people understand your database design).

*Constraints: PK - FK connection between Orders and Users (on user_id) and between Orders and Products (on product_id)
*PKs are auto-incremented via the creation of triggers and sequences in setup.sql

Logical Database Design and Normalization Analysis

Schema:

User

UserID	Username	Password	Type

PK: UserID `
FK:
CK(s): UserId, Username
Functional Dependencies: UserID → Username, UserID → Password, UserID → Type
1NF: This relation has no set-valued attributes, thus it is in 1st normal form.

2NF: Password and Type are FFD on {UserId, Username}
3NF: UserID is a superkey in each functional dependency.
*Type could be either ‘Manager’, ‘Employee’, ‘Customer’

Product

ProductID	Name	Stock	Price	Category

PK: ProductID
FK:
CK(s): ProductID
Functional Dependencies: ProductID → Name, ProductID → Stock, ProductID → Price, ProductID → Category
1NF: This relation has no set-valued attributes, thus it is in 1st normal form.
2NF: Name, Stock, Price, and Category are FFD on {ProductID}.
3NF: ProductID is a superkey in each functional dependency.

Order

PurchaseID	OrderID	UserID	DatePlaced	PickUpDate	ProductID	Quantity

PK: PurchaseID
FK: ProductID, UserID
CK(s): PurchaseID
Functional Dependencies: PurchaseID → OrderID, PurchaseID → UserID, PurchaseID → DatePlaced, PurchaseID → PickUpDate, PurchaseID → ProductID, PurchaseID → Quantity
1NF: This relation has no set-valued attributes, thus it is in 1st normal form.
2NF: OrderID, UserId, DatePlaced, PickedUpDate, ProductId, and Quantity, Stock, Price, are FFD on {PurchaseID}.
3NF: PurchaseID is a superkey in each functional dependency.

Relationships:
UsrOrder(User-Order)
    Cardinality: 1:M
    Attributes: DatePlaced

Log (Order-Product)
    Cardinality: 1:M

Query Description

Query #1: What products are running low in stock?

- Method in DatabaseController.java: public Vector<String> ProductsToOrder(int number){
- Website page(s): managerLowStockProducts.jsp
- Description: Returns the list of all products with stock less than the **user-inputted** number.
- Example: Show all products with less than 2 in stock.

Query #2: What orders contain this item?

- Method in DatabaseController.java: public Vector<Integer> OrdersContainingProduct(int productID){
- Website page(s): managerOrdersContainingProduct.jsp, employeeOrdersContainingProduct.jsp
- Description: Returns a list of all orderIds containing the user-inputted product id.
- Example: Shows all ordersIds of orders that contain banana.

Query #3: What products are in a specific category?

- Method in DatabaseController.java: public Vector<Vector<String>> FindAllProducts(String filterCategory){
- Website page(s): managerGetStockInfo.jsp, employeeGetStockInfo.jsp
- Description: Returns a list of all products under a user-selected certain category filter.
- Example: Lists all foods under category dairy.

Query #4: Which users are employee, customers, or managers?

- Method in DatabaseController.java: public Vector<String> FindAllUsers(String filterType) {
- Website page(s): managerUserSettings.jsp
- Description: Returns a list of all users under a user-selected certain type filter.
- Example: Display the information of only users that are employees.

Query #5: What is the total cost of particular order?

- Method in DatabaseController.java: public double TotalCost(int orderID){
- Website page(s): customerViewOrders.jsp
- Description: Returns a total cost of a given orderID, used in displaying the total cost for each order for a customer. Calculate cost **using records from two relations (Order and Products)** and display to the user in their order statement.
- Example: What is the total cost of a particular order #28?