Nahid Sekander
HTTP5126 Final Project Proposal
Nov 24, 2024

MTG Card Store Database : Database Proposal

Real World Scenario

As a database administrator, the real-world scenario involves managing an online card store that sells unique Magic: The Gathering cards to customers.

Problem 1: Managing Inventory for Customer Checkout

Description: Customers request to purchase Magic: The Gathering (MTG) cards, generating a checkout list. The system must verify that the requested cards are available in the inventory before completing the checkout process.

Solution:

- **Trigger:** Automatically checks inventory levels before processing a checkout. If any card is out of stock, the checkout is blocked, and the customer is notified.
- **View:** Displays available stock for requested cards to simplify the inventory check process.
- **Procedure/Function:** Ensures the transaction proceeds only if all requested cards are in stock.

Required Tables:

1. **Customer Information Table:**
    - **Columns:**
        - `Customer_ID` (Primary Key)
        - `Name` (Not Null)
        - `Email` (Unique/ Not Null)
        - `Address` (Not Null)
    - **Data Types:**
        - `Customer_ID`: Integer
        - `Name`: Varchar(255)
        - `Email`: Varchar(255)

        - `Address`: Varchar(255)
2. **Customer Checkout Table:**
    - **Columns:**
        - `Checkout_ID` (Primary Key)
        - `Customer_ID` (Foreign Key to Customer Information Table)
        - `Card_ID` (Foreign Key to Full MTG Card Database)
        - `Quantity` (Not Null)
        - `Status` (Not Null)
    - **Data Types:**
        - `Checkout_ID`: Integer
        - `Customer_ID`: Integer
        - `Card_ID`: Integer
        - `Quantity`: Integer
        - `Status`: Varchar(50)
3. **Full MTG Card Database:**
    - **Columns:**
        - `Card_ID` (Primary Key)
        - `Name` (Not Null / Unique)
        - `Description` (Not Null / Unique)
    - **Data Types:**
        - `Card_ID`: Integer
        - `Name`: Varchar(255)
        - `Description`: TEXT
4. **Data Store Available Cards Table:**
    - **Columns:**
        - `Card_ID` (Primary Key, Foreign Key to Full MTG Card Database)
        - `Quantity` (Not Null)
        - `Price` (Not Null)
    - **Data Types:**
        - `Card_ID`: Integer
        - `Quantity`: Integer
        - `Price`: Decimal(10, 2)

Relationships:

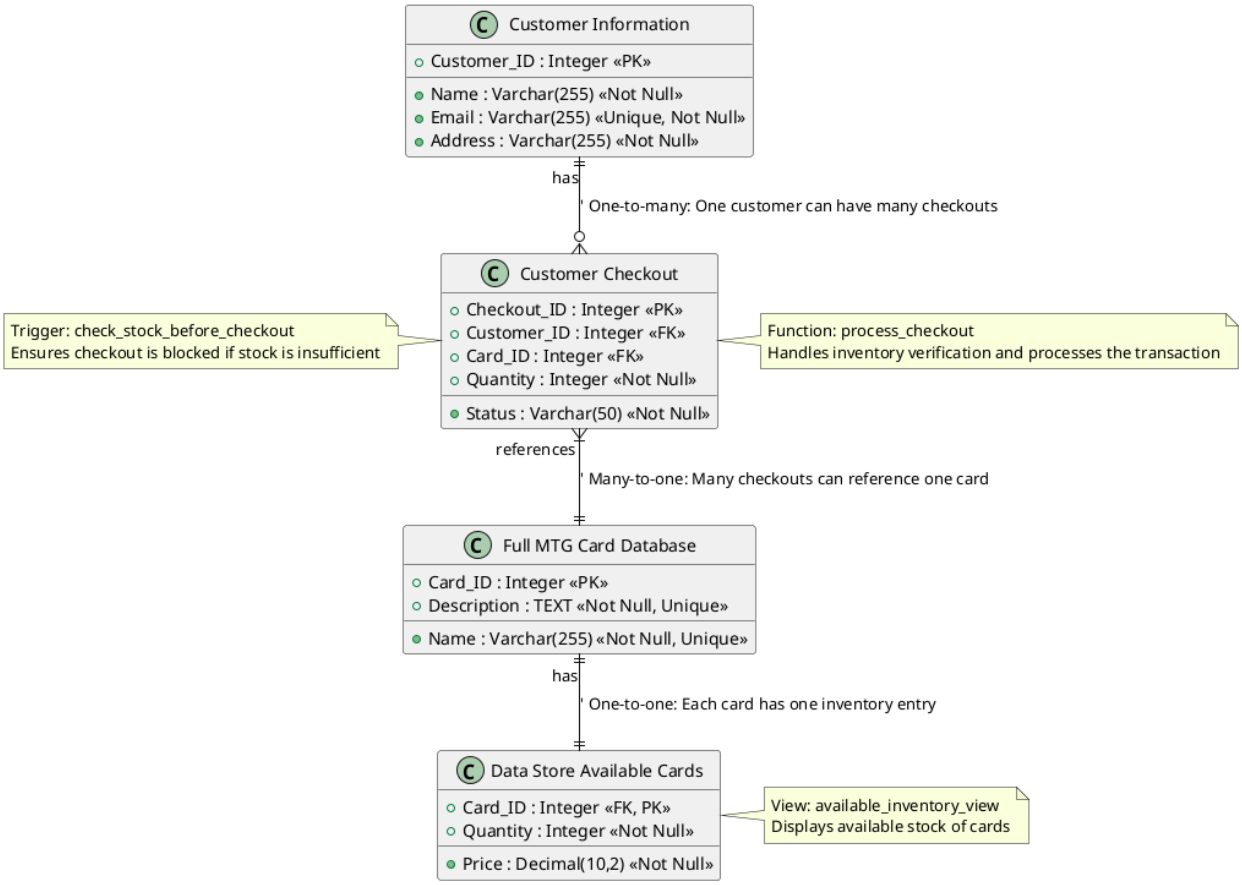
- **Customer Information → Customer Checkout:**
    - One customer can have multiple checkouts.
    - **Mapping:** One (Customer Information) → Many (Customer Checkout)
- **Customer Checkout → Full MTG Card Database:**

    - Many checkout entries can reference the same card.
    - **Mapping:** Many (Customer Checkout) → One (Full MTG Card Database)
- **Full MTG Card Database → Data Store Available Cards:**
    - Each card in the database has one corresponding stock entry in the data store.
    - **Mapping:** One (Full MTG Card Database) → One (Data Store Available Cards)

Architecture Design:

- **Trigger Name:** `check_stock_before_checkout`
    - Ensures the checkout is blocked if any card's stock is insufficient.
- **View Name:** `available_inventory_view`
    - Displays the available stock of cards to validate the customer's checkout request.
- **Function Name:** `process_checkout`
    - Handles inventory verification and processes the transaction.

ERD :



Nahid Sekander
HTTP5126 Final Project Proposal
Nov 24, 2024

Problem 2: Customer Requests for Out-of-Stock Cards

Description: If requested cards are unavailable in the current inventory, customers can submit a request for the store to order them. The system must estimate when these cards will become available based on the upcoming inventory table.

Solution:

- **Trigger:** Automatically updates the customer's request status when the requested cards are restocked.
- **View:** Combines current and upcoming inventory, providing a unified view to estimate restock dates.
- **Procedure/Function:** Manages requests for unavailable cards and tracks restocking status.

Required Tables:

1. **Customer Information Table:**
    - **Columns:**
        - `Customer_ID` (Primary Key)
        - `Name` (Not Null)
        - `Email` (Unique / Not Null)
        - `Address` (Not Null)
    - **Data Types:**
        - `Customer_ID`: Integer
        - `Name`: Varchar(255)
        - `Email`: Varchar(255)
        - `Address`: Varchar(255)
2. **Customer Requested Cards Table:**
    - **Columns:**
        - `Request_ID` (Primary Key)
        - `Customer_ID` (Foreign Key to Customer Information Table)
        - `Card_ID` (Foreign Key to Full MTG Card Database)
        - `Request_Date` (Date) (Not Null)
    - **Data Types:**
        - `Request_ID`: Integer

        - `Customer_ID`: Integer
        - `Card_ID`: Integer
        - `Request_Date`: Date
3. **Full MTG Card Database:**
    - **Columns:**
        - `Card_ID` (Primary Key)
        - `Name` (Not Null / Unique)
        - `Description` (Not Null / Unique)
    - **Data Types:**
        - `Card_ID`: Integer
        - `Name`: Varchar(255)
        - `Description`: TEXT
4. **Data Store Available Cards Table:**
    - **Columns:**
        - `Card_ID` (Primary Key, Foreign Key to Full MTG Card Database)
        - `Quantity` (Not Null)
        - `Price` (Not Null)
    - **Data Types:**
        - `Card_ID`: Integer
        - `Quantity`: Integer
        - `Price`: Decimal(10, 2)
5. **Upcoming Inventory View:**
    - **Columns:**
        - `Card_ID` (Primary Key, Foreign Key to Full MTG Card Database)
        - `Expected_Arrival` (Date)
        - `Quantity` (Not Null)
    - **Data Types:**
        - `Card_ID`: Integer
        - `Expected_Arrival`: Date
        - `Quantity`: Integer

Relationships:

- **Customer Information → Customer Requested Cards:**
    - One customer can make multiple requests.
    - **Mapping:** One (Customer Information) → Many (Customer Requested Cards)
- **Customer Requested Cards → Full MTG Card Database:**
    - Many requests can reference the same card.
    - **Mapping:** Many (Customer Requested Cards) → One (Full MTG Card Database)

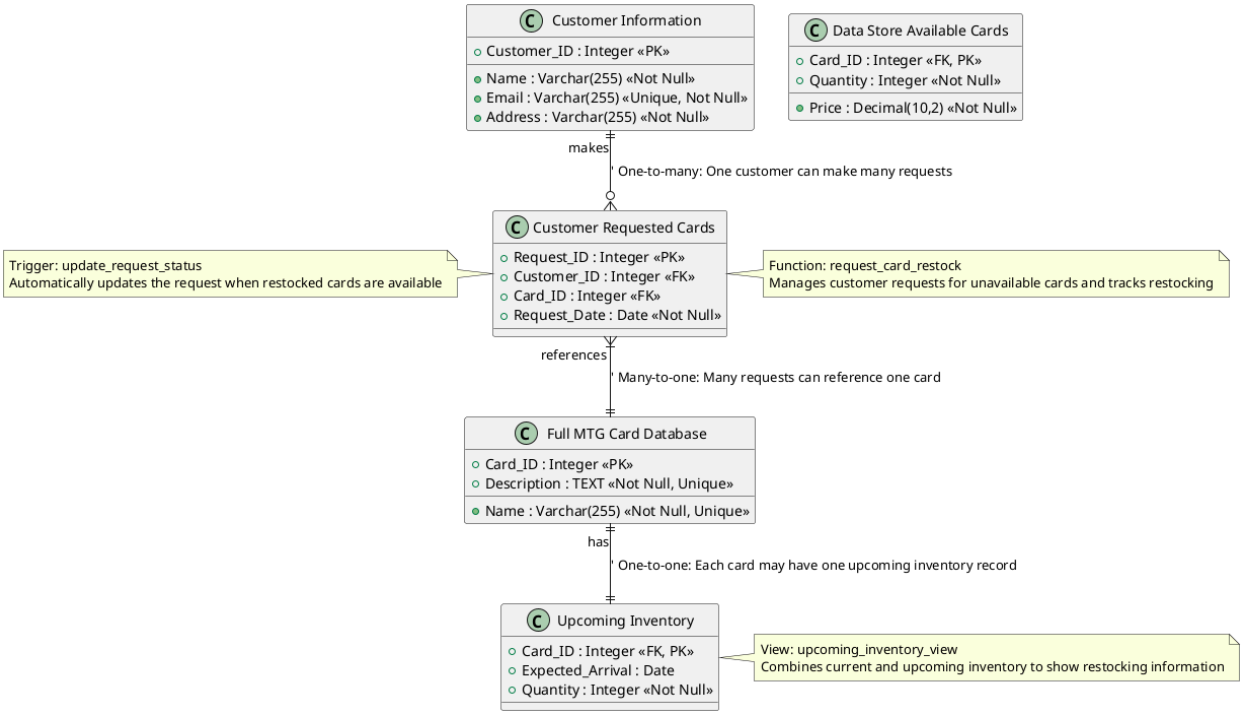
Nahid Sekander
HTTP5126 Final Project Proposal
Nov 24, 2024

- **Full MTG Card Database → Upcoming Inventory:**
    - Each card in the database may have one upcoming inventory record.
    - **Mapping:** One (Full MTG Card Database) → One (Upcoming Inventory)

Architecture Design:

- **Trigger Name:** `update_request_status`
    - Automatically updates the customer request when restocked cards become available.
- **View Name:** `upcoming_inventory_view`
    - Combines current and upcoming inventory to provide restock information.
- **Function Name:** `request_card_restock`
    - Manages customer requests for unavailable cards and tracks restocking.

ERD:



Nahid Sekander
HTTP5126 Final Project Proposal
Nov 24, 2024

Retrospective

Changes Made to the Proposal

From the original proposal, additional triggers and logic refinements were incorporated to address gaps in functionality :

1. A new trigger was added to Problem 1 to automatically update the status of pending checkout requests when new stock is added. This ensures customers are promptly notified when their requested items become available for purchase.

2. For Problem 2, two triggers were added:

after_insert_upcoming_inventory: Updates the customer request table when new inventory is added, and the arrival date is in the present or past.

after_update_upcoming_inventory: Checks if updates to the upcoming inventory table include a date change. If the new date is later than the current date, the customer request status is updated to "restocked."

Challenges Encountered

During the design and implementation phases, several challenges arose:

1. Trigger and Procedure Logic: Developing the triggers and procedures required multiple iterations to refine the logic. In some cases, the initial implementation failed, requiring the triggers and procedures to be dropped and restructured.

Resolution: Logic was saved separately in a dedicated file for offline editing and testing. This allowed for a more systematic approach to debugging and refining the logic before applying it to the database.

2. Data Consistency: Ensuring data remained consistent across related tables (e.g., customer requests, inventory, and checkout) proved complex due to cascading updates triggered by multiple processes.

Resolution: Rigorous testing was conducted to validate the interdependencies between tables, ensuring triggers and procedures behaved as intended.

Nahid Sekander
HTTP5126 Final Project Proposal
Nov 24, 2024

Real-World Problem Resolution

The implemented database successfully addresses the identified problems:

1. Managing Inventory for Customer Checkout: The system blocks purchases if requested quantities exceed available stock. This ensures accurate inventory management and avoids overselling.

2. Handling Customer Requests for Out-of-Stock Cards: Customers can request unavailable cards, and the status of their requests is automatically updated based on inventory restocking or upcoming inventory changes.

Future Plans

To enhance the database further, several features and improvements are planned:

1. Deck Creation Feature: Customers will be able to create a "deck" table where they can compile a list of desired cards. This feature will:

Check inventory availability for all cards in the deck.

Return the total cost of the requested cards, providing customers with a comprehensive purchase estimate.

2. User Interface Integration: Building a user-friendly interface for customers to manage their requests, view inventory statuses, and receive notifications when their requested cards are restocked.

These future steps aim to expand the functionality and user experience of the MTG card store database while maintaining its scalability and efficiency.