Principles Of Database Systems Project Report - 3

By:
Jasmeet Singh Wadhwa (jw9180)
Aayush Anand (aa13015)

Languages and Frameworks Used

- **Languages**: Python, SQL
- **Frameworks**: Flask (Python web framework)
- **Libraries**:
- pymysql for database connectivity
- bcrypt for password hashing
- **Front-End**: HTML, CSS
- **Database**: MySQL

Schema Changes and Their Purpose

- Unified Login Schema: Replaced the Person table with a user table to centralize login credentials and personal details.Purpose: Simplified user authentication and allowed for a more intuitive and unified registration and login process.
- Role-Specific Tables: Created three additional tables: staff, client, and volunteer, each linked to the user table through a foreign key. Purpose: Separated role-specific details into their respective tables to make the schema modular and adaptable. This design allows each role to have unique attributes while sharing common login credentials from the user table.
- Improved Schema Relationships: Updated relationships across tables for better data integrity. For example, the donorUsername in the Donation table now references the user table instead of being standalone. Purpose: Ensured foreign key constraints align with the centralized user table for consistency.
- These changes collectively improve the schema's clarity, usability, and scalability, ensuring it aligns with the project's objectives while simplifying the application logic.

Additional Constraints, Triggers, Stored Procedures

- Foreign Key Constraints: Enforced foreign key relationships between tables to maintain data integrity, such as:

- donorUsername in the Donation table referencing the user table.
- roomNum and shelfNum in the Piece table referencing the Location table.
- mainCategory and subCategory in the Item table referencing the Category table.
- Purpose: Ensures data consistency and prevents orphan records.

Main Queries Executed

- Managing Orders - Query to retrieve user-specific orders:

```
SELECT

 o.orderID, o.orderDate, o.orderNotes, d.status, d.deliveryDate, d.username AS volunteer,

 o.client, o.supervisor,

 client_user.fname AS client_fname, client_user.lname AS client_lname,

 supervisor_user.fname AS supervisor_fname, supervisor_user.lname AS
supervisor_lname

FROM Ordered o

LEFT JOIN Delivered d ON o.orderID = d.orderID

LEFT JOIN user client_user ON o.client = client_user.username

LEFT JOIN user supervisor_user ON o.supervisor = supervisor_user.username

WHERE d.username = %s OR o.client = %s OR o.supervisor = %s;
```

- Popular Categories - Query to retrieve popular categories based on date range:

```
SELECT

c.mainCategory,

c.subCategory,

COUNT(DISTINCT ii.orderID) AS num_orders

FROM

Category c

JOIN

Item i ON c.mainCategory = i.mainCategory AND c.subCategory = i.subCategory
```

JOIN

ItemIn ii ON i.itemID = ii.itemID

JOIN

Ordered o ON ii.orderID = o.orderID

WHERE

 o.orderDate BETWEEN %s AND %s

GROUP BY

c.mainCategory, c.subCategory

ORDER BY

num_orders DESC

LIMIT 10;

- Query for items donated by category:

  SELECT

  c.mainCategory,

  c.subCategory,

  COUNT(i.itemID) AS total_items

  FROM

  Category c

  LEFT JOIN

  Item i ON c.mainCategory = i.mainCategory AND c.subCategory = i.subCategory

  GROUP BY

  c.mainCategory, c.subCategory;

- Query for summary of how clients were helped:

  SELECT

  o.client AS client_username,

COUNT(DISTINCT o.orderID) AS total_orders,

 COUNT(ii.itemID) AS total_items_received

FROM

Ordered o

LEFT JOIN

 ItemIn ii ON o.orderID = ii.orderID

GROUP BY

 O.client;

- Find Order Item - Role-specific query:
- **For Client:**

SELECT i.itemID, i.iDescription, p.pieceNum, l.roomNum, l.shelfNum, l.shelfDescription

FROM ItemIn ii

JOIN Item i ON ii.itemID = i.itemID

JOIN Piece p ON i.itemID = p.itemID

JOIN Location l ON p.roomNum = l.roomNum AND p.shelfNum = l.shelfNum

JOIN Ordered o ON ii.orderID = o.orderID

WHERE ii.orderID = %s AND o.client = %s

- **For Staff:**

SELECT i.itemID, i.iDescription, p.pieceNum, l.roomNum, l.shelfNum, l.shelfDescription

FROM ItemIn ii

 JOIN Item i ON ii.itemID = i.itemID

  JOIN Piece p ON i.itemID = p.itemID

  JOIN Location l ON p.roomNum = l.roomNum AND p.shelfNum = l.shelfNum

  WHERE ii.orderID = %s

- **For Voluteer:**

  ```
  SELECT i.itemID, i.iDescription, p.pieceNum, l.roomNum, l.shelfNum,
  l.shelfDescription

  FROM ItemIn ii

  JOIN Item i ON ii.itemID = i.itemID

  JOIN Piece p ON i.itemID = p.itemID

  JOIN Location l ON p.roomNum = l.roomNum AND p.shelfNum = l.shelfNum

  JOIN Delivered d ON d.orderID = ii.orderID

  WHERE ii.orderID = %s AND d.username = %s
  ```

Difficulties Encountered and Lessons Learned

1. One of the biggest challenges was managing foreign key constraints across the tables. Specifically:
   - Ensuring data integrity while inserting or updating records with dependencies.
   - Resolving errors like "Cannot add or update a child row" due to missing parent records in referenced tables.
   - Adjusting foreign key relationships to balance flexibility and data integrity. For example, removing or modifying constraints when it became evident that certain tables (e.g., Donor) should operate independently.
   - This required frequent schema updates and careful ordering of data insertion.
2. Complex Query Design:
   - Designing queries for features like "Year-End Report" and "Popular Categories" was challenging due to the need for multiple table joins, groupings, and aggregations.
   - Queries had to be optimized to ensure performance with large datasets while providing meaningful insights.
3. Handling User Roles:
   - Managing user roles (staff, client, and volunteer) via separate tables introduced complexity in login and feature-specific functionalities.
   - Each role had unique permissions, requiring robust checks in both queries and application logic.
4. Input Validation and Error Handling:

    - Ensuring that user inputs (e.g., locations, categories, item descriptions) were validated properly to prevent invalid or incomplete data from being inserted into the database.
    - Handling database errors gracefully and providing user-friendly feedback was another challenge.
5. Testing and Debugging:
    - Debugging issues like missing records, incorrect joins, and improper query results required iterative testing and logging.
    - Ensuring that all queries worked as expected for edge cases (e.g., orders with no items, donors with no donations).

Lesson Learned

- The Importance of Schema Design:A well-thought-out database schema is critical. Poorly designed relationships can lead to cascading issues. Planning and reviewing the schema upfront saves time in the long run.
- Foreign Key Constraints Require Careful Management: While foreign keys are invaluable for maintaining data integrity, they can complicate data insertion and updates. In some cases, loosening constraints or introducing nullable columns can enhance flexibility.
- Error Logs and Debugging Tools Are Lifesavers: Logging SQL errors and application flow made debugging much more manageable. These logs helped identify missing records or mismatched inputs quickly.

Team Contributions

Jasmeet's Contributions

Feature 1: Login & User Session Handling

- Implemented secure login and registration functionality.
- Ensured passwords are stored using cryptographic hashing (bcrypt with salt).
- Managed user roles (staff, client, volunteer) and session variables to control access.
- Redirected users to their respective dashboards based on roles.

Feature 4: Accept Donation

- Developed the functionality for staff members to accept donations.
- Ensured donor verification and registration process.
- Designed and implemented donation flow to record item, donor, and storage details.
- Handled database updates for donated items and their storage locations.

Feature 10: Update Order Status

- Created a system to allow users (volunteers and staff) to update the status of orders they are managing or delivering.
- Implemented dynamic status updates with proper permission checks.
- Added error handling to prevent unauthorized updates

.

Feature 11: Year-End Report

- Developed a comprehensive year-end report feature.
- Included data on the number of clients served, items donated by category, and a summary of how clients were helped.
- Designed queries to fetch and aggregate relevant data efficiently.

Aayush's Contributions

Feature 2: Find Single Item

- Implemented functionality to locate pieces of an item based on its item ID.
- Designed queries to retrieve the locations of all pieces associated with a given item.

Feature 3: Find Order Items

- Developed a feature to retrieve all items in an order, along with the locations of each item's pieces.
- Included user prompts for order ID and returned detailed item location data.

Feature 8: User’s Tasks

- Created functionality to display all orders associated with the currently logged-in user.
- Included relevant details for each order based on the user's role (client, volunteer, or staff).

Feature 9: Rank System

- Implemented the rank system for identifying the most popular category/subcategory based on the number of orders in a given time period.
- Designed queries to rank categories and subcategories effectively.