**Week 3: Design: Database**

Wasan Yotchai

Maryville University

SWDV 691: Software Development Capstone 1W 25/SP2

Professor Joseph Gradecki

March 29, 2025

****

Database Design

Wasan Yotchai

March 29, 2025

COMMENT: Rename the table from plural to singular form.

**OVERVIEW**

Based on my MVP project, PostgreSQL is a better choice for my database.

**Justification for Choosing PostgreSQL**

1. **Highly Structured & Normalized Data:**
   * The application will store structured financial data such as transactions, budgets, user details, and reports, which fit well into a relational model.
   * Relational databases are ideal for maintaining data integrity across multiple related entities.
2. **Many-to-Many & One-to-Many Relationships:**
   * Users can have multiple bank accounts.
   * Each account can have multiple transactions.
   * Transactions belong to categories, and categories can be customized by users.
   * Budgeting data links user income and expenses, making relationships more complex.
3. **ACID Compliance & Data Consistency:**
   * Financial applications require **accurate**, **consistent**, and **secure** transaction handling. PostgreSQL supports **transactions**, **foreign keys**, and **constraints** to ensure integrity.
4. **Scalability & Security:**
   * PostgreSQL is **scalable** for large datasets.
   * Supports **encryption, role-based access control, and auditing** for financial data security.

MongoDB would be ideal if each user’s financial data was stored as independent, self-contained documents with minimal relationships. However, since the app requires complex relationships and structured financial tracking, PostgreSQL is the better fit.

**Data Specifications:**

**Database Design (PostgreSQL)**

Below is an **Entity-Relationship Diagram (ERD)** description and a relational schema.

**Tables & Relationships**

1. **Users Table**
   * Stores user details.
   * One user can have multiple **accounts** and **budgets**.
2. **Accounts Table**
   * A user can have multiple **bank accounts**.
   * Linked to transactions.
3. **Transactions Table**
   * Stores **income & expenses**.
   * Linked to categories and accounts.
   * Supports AI-powered categorization.
4. **Categories Table**
   * AI assigns **transactions** to categories (e.g., Rent, Groceries).
   * Users can modify categories.
5. **Budgets Table**
   * Links users, categories, and spending goals.
   * AI adjusts budgets dynamically.
6. **Reports Table**
   * Stores user-generated **financial reports**.
   * Provides historical insights.

**PostgreSQL Schema Design**

**Entity-Relationship (ER) Diagram**

The database consists of the following tables:

**1. User Table**

**Purpose:** Stores user information.

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| user\_id | SERIAL | PRIMARY KEY |
| name | VARCHAR(225) | NOT NULL |
| email | VARCHAR(225) | UNIQUE, NOT NULL |
| password (hash) | TEXT | NOT NULL |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |

**Relationships:**

* One-to-Many with **Accounts**

**2. Account Table**

**Purpose:** Represents different financial accounts of users (e.g., Checking, Savings, Credit Card).

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| account\_id | SERIAL | PRIMARY KEY |
| user\_id | INT | FOREIGN KEY REFERENCES Users(user\_id) ON DELETE CASCADE |
| account\_name | VARCHAR(255) | NOT NULL |
| balance | DECIMAL(10,2) | DEFAULT 0.00 |
| account\_type | VARCHAR(50) | CHECK (account\_type IN ('Checking', 'Savings', 'Credit Card')) |

**Relationships:**

* One-to-Many with **Transactions**
* Many-to-One with **Users**

**3. Transaction Table**

**Purpose:** Stores all financial transactions.

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| transaction\_id | SERIAL | PRIMARY KEY |
| account\_id | INT | FOREIGN KEY REFERENCES Accounts(account\_id) ON DELETE CASCADE |
| category\_id | INT | FOREIGN KEY REFERENCES Categories(category\_id) |
| amount | DECIMAL(10,2) | NOT NULL |
| transaction\_type | VARCHAR(50) | CHECK (transaction\_type IN ('Income', 'Expense')) |
| description | TEXT |  |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |

**Relationships:**

* Many-to-One with **Accounts**
* Many-to-One with **Categories**

**4. Categorie Table**

**Purpose:** Defines spending or income categories for transactions (e.g., Rent, Salary, Groceries).

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| category\_id | SERIAL | PRIMARY KEY |
| category\_name | VARCHAR(255) | UNIQUE, NOT NULL |
| category\_type | VARCHAR(50) | CHECK (category\_type IN ('Income', 'Expense')) |

**Relationships:**

* One-to-Many with **Transactions**

**5. Budget Table**

**Purpose:** Stores user-defined budget constraints for specific categories.

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| budget\_id | SERIAL | PRIMARY KEY |
| user\_id | INT | FOREIGN KEY REFERENCES Users(user\_id) ON DELETE CASCADE |
| category\_id | INT | FOREIGN KEY REFERENCES Categories(category\_id) |
| amount | DECIMAL(10,2) | NOT NULL |
| start\_date | DATE | NOT NULL |
| end\_date | DATE | NOT NULL |

**Relationships:**

* Many-to-One with **Users**
* Many-to-One with **Categories**

**6. Report Table**

**Purpose:** Stores summarized reports for a user’s financial activity.

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| report\_id | SERIAL | PRIMARY KEY |
| user\_id | INT | FOREIGN KEY REFERENCES Users(user\_id) ON DELETE CASCADE |
| report\_type | VARCHAR(50) | CHECK (report\_type IN ('Monthly', 'Yearly')) |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |

**Relationships:**

* Many-to-One with **Users**

**Explanation of Data Usage**

* **Users** register and log in to manage their finances.
* **Accounts** are linked to users and store balances.
* **Transactions** log deposits and withdrawals.
* **Categories** classify transactions.
* **Budgets** help users track spending.
* **Reports** summarize financial activities.

**Purpose, Implementation, and Interactions**

**Users**

**Purpose:**  
The **Users** collection is critical for authentication and authorization within the application. It ensures that users can securely log in, manage their financial data, and interact with other features of the application. Without this collection, the system would have no way to associate financial transactions, budget plans, or reports with specific individuals.

**Implementation:**

* Users will sign up with an email and password, which will be securely stored using encryption.
* Upon successful authentication, a session token or JWT (JSON Web Token) will be issued for access control.
* Each user will have personal financial data linked to their account, ensuring data privacy and security.

**Interaction:**

* A user accesses the application and is presented with the login page.
* If the login credentials are correct, they are redirected to the dashboard, where they can manage their finances.
* If the credentials are incorrect, they receive an error message prompting them to try again.
* Users can also reset their password via email verification.

**Transactions**

**Purpose:**  
The **Transactions** collection is at the core of the application. It stores all financial transactions, allowing users to track income and expenses, categorize spending, and generate reports.

**Implementation:**

* Each transaction will be associated with a **user ID**, ensuring that users only access their own financial data.
* Transactions will include details such as date, amount, category (e.g., rent, groceries, salary), and payment method.
* CRUD (Create, Read, Update, Delete) operations will be supported for users to manage their financial records.

**Interaction:**

* Users can add a new transaction by selecting a category, entering an amount, and specifying the date.
* They can view a list of all past transactions on the **Transactions Page**, with sorting and filtering options.
* Users can edit or delete transactions if needed.
* If an invalid transaction (e.g., negative income, missing required fields) is submitted, an error message will be displayed.

**Budget**

**Purpose:**  
The **Budget** collection helps users set financial goals by allowing them to define spending limits for different categories. It provides an overview of whether users are staying within their budgets or exceeding them.

**Implementation:**

* Users can create budgets by defining a maximum spending limit for categories like "Food," "Entertainment," or "Bills."
* The system will track actual spending in each category and notify users when they approach or exceed their limits.

**Interaction:**

* Users access the **Budget Page** to set monthly spending limits.
* When they add new transactions, the system updates the budget and provides a summary.
* If they exceed their budget, they receive a notification or warning message.
* Users can update or delete their budget plans if their financial goals change.

**Reports**

**Purpose:**  
The **Reports** collection helps users analyze their financial habits by generating visual reports and insights based on their transaction history.

**Implementation:**

* The application will generate **monthly spending reports** and **income vs. expense breakdowns** based on stored transactions.
* Users can select a time range and view reports in different formats (e.g., bar charts, pie charts).
* The backend will aggregate transaction data and return structured reports.

**Interaction:**

* Users navigate to the **Reports Page** to generate and view financial insights.
* They can filter reports by date, category, or payment method.
* If there are insufficient data points, the system will inform the user that a report cannot be generated.

**Goals**

**Purpose:**  
The **Goals** collection allows users to define long-term savings goals, track their progress, and stay motivated to save money for specific purposes (e.g., an emergency fund, a vacation, or a car purchase).

**Implementation:**

* Users can set a **target savings amount** and a **deadline** for each financial goal.
* The system will track their contributions and display their progress over time.
* Notifications can remind users if they are falling behind on their savings plan.

**Interaction:**

* Users visit the **Goals Page** to set up new savings goals.
* When they add savings transactions, their goal progress updates automatically.
* The dashboard displays progress bars to visually represent their savings progress.
* If a goal is reached, users receive a success notification.