**Use Case 1: Book Recommendation Website**

For a book recommendation website that allows users to rate and review books, a combination of a graph database and a document store provides an ideal solution to meet its needs for data flexibility, personalized recommendations, and efficient querying.

Graph Database for Recommendations

A graph database like Neo4j is an effective choice for managing the website’s recommendation system, which depends on relationships between users and books. Graph databases store data as nodes and edges where nodes represent entities—such as users and books— and edges represent relationships—such as a user reading and liking or disliking a book—. This structure enables efficient relationship-based queries essential for recommendations. For instance, to suggest books, the system can quickly find “similar” users who have rated or liked the same books as the current user. With a graph database, algorithms based on collaborative filtering are more manageable, enabling us to recommend books based on what similar users have enjoyed. Additionally, if the algorithm needs adjustments, a graph database can adapt with minimal structural changes.

Document Store for Book Metadata and User Data

A document store such as MongoDB complements the graph database by handling general book metadata and user data. Unlike relational databases, document stores are schema-less, allowing more flexibility for data that may change over time. Each book’s metadata—such as title, author, genre, and publisher—can be stored in individual JSON-like documents. Similarly, user data containing various details like username, demographics, and site activity can also be managed in this format. This flexibility is especially useful if new attributes are added to books or user data. For example, if the site begins tracking additional user metrics, such as average reading time, document stores allow these additions without restructuring. Document databases are also known for efficient read and write speeds, making them suitable for quickly displaying book and user information.

Sources

1. Sheng, Wayne. (2022). *Graph Database For Recommendation Systems* <https://www.nebula-graph.io/posts/use-cases-of-graph-databases-in-real-time-recommendation>
2. Neo4j. (2024). *Why Neo4j for Knowledge Graphs?*  
   <https://neo4j.com/use-cases/knowledge-graph/>
3. pheonixNAP. (2021). *What is a Document Database?* <https://phoenixnap.com/kb/document-database>

**Use Case 2: Money Exchange App**

For a money exchange app, a relational database is the best choice for ensuring data consistency, supporting complex queries, and meeting regulatory requirements. Relational databases like PostgreSQL or MySQL are well-suited for financial applications because they provide ACID (Atomicity, Consistency, Isolation, Durability) compliance, essential for maintaining data integrity in transaction processing.

A relational database can store all primary data types required by the app, such as account balances, transaction records, user profiles, and linked bank account information. Each piece of data—such as user accounts, transactions, and relationships between users—can be organized into separate tables, enabling a structured, reliable way to manage the app’s core data.

Data Integrity and Consistency: Relational databases support ACID transactions, which is critical for handling payments. When a user sends money, the relational database can ensure that the payor’s balance is debited and the payee’s balance is credited in a single, atomic transaction. If anything goes wrong, the entire transaction can be rolled back, preventing partial updates and ensuring accurate balances.

Efficient Querying for User History and Analytics: Relational databases are optimized for complex queries, which allows users to view their complete transaction histories and account balances. Additionally, the app can perform analytics on the relational data to track money transfers over various timeframes, identify trends in bank usage, and generate reports. This is especially helpful for producing financial summaries or generating “frequent partner” lists for users.

Regulatory Compliance and Security: Financial services must comply with regulatory standards that require detailed records of all user activity. A relational database’s structured, queryable format makes it easier to maintain and retrieve these records. For instance, detailed user session data, debit and credit records, and metadata for each transaction (e.g., timestamps, transaction IDs) can all be stored and easily queried for audit purposes.

Sources

1. Zeflikilis, George. (2024). *Building a Personal Finance Management App: Database Setup with PostgreSQL and Docker.*

<https://medium.com/towards-data-engineering/building-a-personal-finance-management-app-database-setup-with-postgresql-and-docker-5075e283303e>

1. EASA. (2022). *8 advantages of a Relational Database* <https://www.easasoftware.com/insights/8-advantages-of-a-relational-database/>
2. Ahmed, Aisha. (2024). *What are the Main Advantages of Relational Database Model?*

<https://hapy.co/journal/advantages-of-relational-database/>