Database Architecture Design for E-commerce Platform

Database System

Choice: Relational Database Management System (RDBMS)

Reasoning: A NoSQL database is an ideal choice for scenarios with rapidly evolving or flexible data structures, diverse data types (unstructured or semi-structured), or high write throughput. However, for our e-commerce platform, where maintaining ACID (Atomicity, Consistency, Isolation, Durability) properties is crucial, especially in financial transactions, and where structured data with clear relationships between entities is prevalent, an RDBMS (Relational Database Management System) is preferred. RDBMS is specifically designed for structured data, complex queries, ACID transactions, and employs concepts like primary and foreign keys to manage relationships between different tables. Therefore, for our e-commerce architecture, an RDBMS is deemed more suitable than a NoSQL database.

Table Structure

1. Users Table
    a. ‘user_id’ (Primary Key)
    b. ‘username’
    c. ‘email’ (Candidate Key)
    d. ‘hashed_password’
    e. ‘address’
    f. ‘mobile_number’ (Candidate Key)
    g. ‘profile_image’
    h. ‘signup_DateTime’
    i. ‘status’

2. Products Table
    a. ‘product_id’ (Primary Key)
    b. ‘category_id’ (Foreign Key to Categories table)
    c. ‘product_name’
    d. ‘product_description’
    e. ‘product_price’

f. ‘status’

3. Categories Table
    a. ‘category_id’ (Primary Key)
    b. ‘category_name’
    c. ‘status’

4. Orders Table
    a. ‘order_id’ (Primary Key)
    b. ‘user_id’ (Foreign Key to Users table)
    c. ‘order_DateTime’
    d. ‘order_status’

5. OrderDetails Table
    a. ‘order_detail_id’ (Primary Key)
    b. ‘order_id’ (Foreign Key to Orders table)
    c. ‘product_id’ (Foreign Key to Products table)
    d. ‘quantity’
    e. ‘price’

6. Transactions Table
    a. ‘transaction_id’ (Primary Key)
    b. ‘user_id’ (Foreign Key to Users table)
    c. ‘amount’
    d. ‘transaction_DateTime’
    e. ‘payment_status’

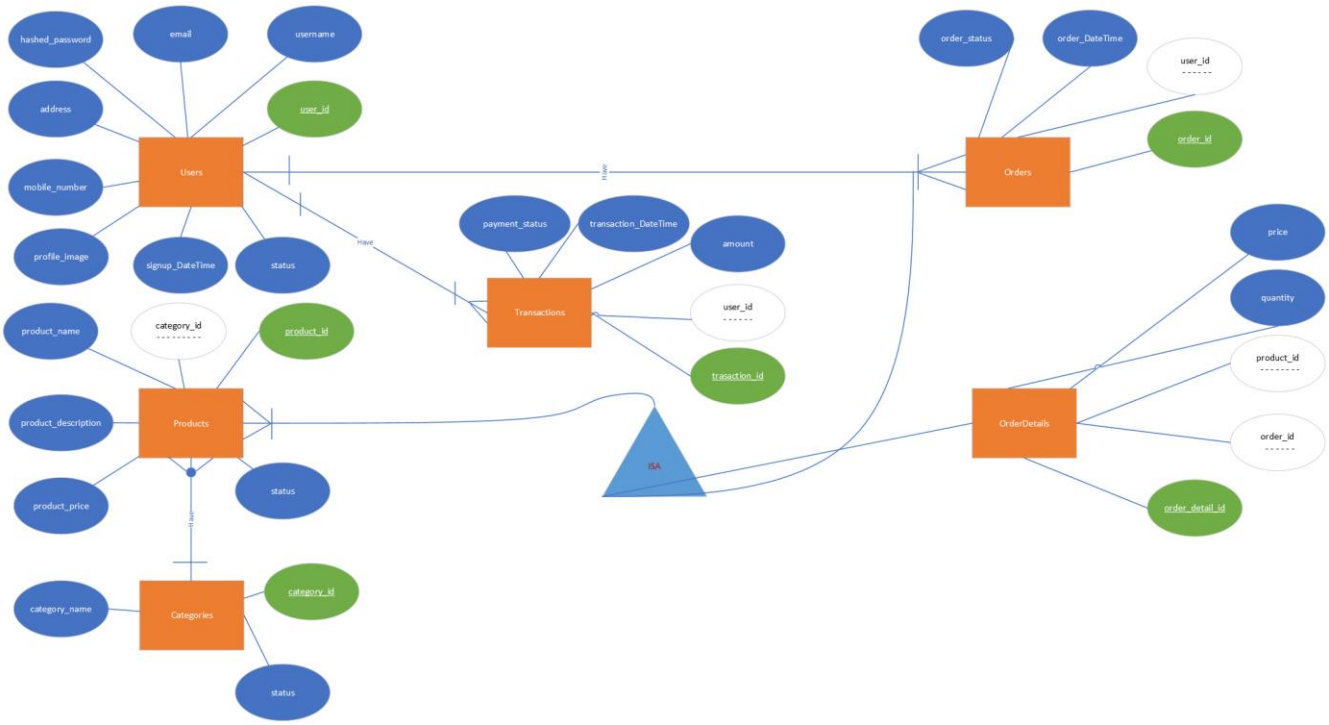

Fig: ER Diagram for e-commerce platform

Relationship

- Users have a one-to-many relationship with both Orders and Transactions.
- Products have a many-to-one relationship with Categories.
- Orders and Products have a many-to-many relationship via the OrderDetails table.

Scalability and Performance

- Apply indexing on commonly queried columns (e.g: user_id, product_id) for faster retrieval.
- Implement normalization to reduce data redundancy and maintain data integrity.
- Consider partitioning large tables like Orders or Transactions based on date ranges to optimize query performance.
- Configure caching technique such as in-memory caching (Redis\Memcached) for frequently accessed data to reduce database load.
- Configuring load balancing to distribute incoming traffic across multiple database servers preventing overload on a single server and improve overall system performance.

Security Measures

- Use encryption mechanism to enhance sensitive data security both at rest and in transit..
- Hashing and salting passwords to enhance security.
- Use parameterized queries or prepared statements/reliable framework's active database query record to prevent SQL injection attacks.
- Apply the principle of least privilege to database users, granting only necessary permissions.
- Regularly updating and patching the database system to address security vulnerabilities by conducting regular security audits and vulnerability assessments.
- Practicing audit trails to log and track changes to sensitive data through monitoring user activities which can be valuable for security and compliance purposes.

Backup and Recovery

- Setup regular backups/point in time data snapshot to prevent data loss.
- Considering database replication for high availability and fault tolerance. This involves maintaining duplicate copies of the database on separate servers, allowing for failover in case of hardware failover in case of hardware failure or other issues.
- Establishing a robust disaster recovery plan to ensure business continuity.

By adopting these strategies, we can establish a database architecture that not only supports the current requirements of the ecommerce platform but also provides a strong base for future scalability and security. Regular monitoring, updates and continuous improvement will be essential to ensure the ongoing success of the database system.