**Fintech Platform Documentation**

**XNL Innovation**

**Task submitted by Aishani Das – 21BRS1002**

**Introduction**

The fintech platform is designed to handle financial transactions efficiently while ensuring scalability, security, and performance optimization. This document provides details about the database schema, implementation decisions, and performance testing results.

**Schema Design**

**Overview**

The database schema is designed to support users, vendors, accounts, transactions, market data, and audit logs. The primary goal is to ensure data integrity, fast query performance, and scalability.

**Entities**

1. **Users**: Stores user information with constraints to maintain unique identifiers.
2. **Vendors**: Maintains vendor details, including category and contact information.
3. **Accounts**: Tracks financial accounts linked to users and vendors.
4. **Transactions**: Logs financial transactions with status tracking.
5. **Market Data**: Stores asset pricing data for financial market analysis.
6. **Audits**: Logs system events for security and compliance.

**Relationships**

* Users are linked to accounts.
* Vendors are associated with transactions.
* Transactions reference accounts and vendors.
* Audits track user activity and transactions.

**Implementation Decisions**

**Storage Engine Selection**

* **InnoDB** was chosen for its transactional integrity, ACID compliance, and row-level locking.
* **Compressed tables** were implemented to optimize disk usage for large datasets.

**Indexing Strategy**

* Indexed frequently used columns such as email, phone numbers, vendor IDs, and transaction statuses.
* Composite indexes were created to improve query performance on multi-column searches.

**Query Optimization**

* Used EXPLAIN to analyze query execution plans.
* Replaced subqueries with joins to minimize redundant computations.
* Indexed timestamp-based searches for time-series data.
* Applied partitioning to large tables (e.g., Transactions) for better query execution.

**Partitioning Strategy**

* Implemented **range partitioning** for transactions based on the created\_at timestamp.
* This ensures that older transactions are efficiently queried and archived.

**Performance Test Results**

**Testing Approach**

* Simulated large datasets with millions of records.
* Measured execution times, memory consumption, and CPU usage.
* Compared performance before and after optimizations.

**Benchmark Results**

|  |  |  |
| --- | --- | --- |
| Query Type | Before Optimization | After Optimization |
| Vendor Revenue Aggregation | 4.5s | 1.2s |
| User Account Balance Query | 3.2s | 0.9s |
| Transaction Search (Last 30 Days) | 6.7s | 1.8s |

**Key Improvements**

1. **Indexing** reduced query execution time by over 70%.
2. **Partitioning** improved efficiency in retrieving recent transactions.
3. **Query Refactoring** using optimized joins reduced memory footprint.

**Conclusion**

This fintech platform database is optimized for scalability, high-performance queries, and data integrity. Through indexing, partitioning, and query tuning, the system is capable of handling large-scale financial transactions efficiently. The implementation decisions were driven by real-world performance considerations and best practices in database optimization.