Reasoning Document: Multi-Exchange Derivatives Database Design

1. Overview

This project implements a PostgreSQL database to store and analyze derivatives trading data across multiple exchanges such as NSE and MCX. The system is designed to efficiently handle large-scale time-series financial data including futures and options contracts. The database supports analytical queries such as open interest tracking, volatility analysis, option chain summaries, and cross-exchange comparisons.

The primary goal of this design is to ensure scalability, query performance, and analytical flexibility while maintaining data integrity.

---

2. Design Choices

The central table, fo_data, was designed to store all contract-level trading information. Each row represents a specific derivative contract instance with attributes such as symbol, expiry date, strike price, option type, and trading metrics.

Key design decisions include:

• The symbol column identifies the underlying asset (e.g., NIFTY, BANKNIFTY, GOLD).
• The expiry_dt column allows grouping and filtering of contracts by expiration date.
• The strike_pr and option_typ columns allow representation of option contracts and enable option chain analysis.
• The exchange_name column was added to support a multi-exchange architecture, enabling comparison between exchanges such as NSE and MCX.

This structure ensures flexibility and supports advanced derivatives analytics without requiring frequent schema changes.

---

3. Table Structure and Column Types

Appropriate data types were chosen to ensure storage efficiency, query performance, and numerical accuracy:

• SERIAL PRIMARY KEY was used for the id column to uniquely identify each record.
• VARCHAR was used for instrument, symbol, option_typ, and exchange_name to support variable-length identifiers.
• DATE was used for expiry_dt and timestamp to enable efficient time-based filtering and partitioning.
• NUMERIC was used for price fields such as open, high, low, close, and settle_pr to preserve financial precision.
• BIGINT was used for contracts, open_int, and chg_in_oi to handle large trading volumes.

These choices ensure both accuracy and efficient storage for large financial datasets.

---

4. Indexing and Performance Optimization

Indexes were created to improve query performance on frequently accessed columns.

The following indexes were implemented:

• B-tree index on symbol to optimize symbol-based queries.
• B-tree index on timestamp to improve time-series filtering.
• B-tree index on exchange_name to support cross-exchange analysis.

Additionally, BRIN (Block Range Index) indexing was considered for the timestamp column. BRIN indexes are particularly efficient for large, time-ordered datasets and significantly reduce storage overhead while maintaining fast query performance.

---

5. Query Optimization Techniques

Query performance was evaluated using the PostgreSQL EXPLAIN ANALYZE command. This allowed analysis of execution plans and identification of performance bottlenecks.

Performance improvements were achieved through:

• Proper indexing of frequently filtered columns
• Use of window functions for efficient rolling calculations
• Avoidance of unnecessary full table scans
• Efficient aggregation using GROUP BY and partitioned ranking

These optimizations significantly improved query execution speed, especially for time-based and aggregation queries.

---

6. Analytical Capabilities

The database supports several advanced analytical use cases, including:

• Open interest trend analysis
• Rolling volatility calculations using window functions
• Option chain summaries grouped by strike price and expiry
• Cross-exchange price comparisons
• Volume and liquidity analysis

These capabilities demonstrate the system's ability to support real-world financial analytics workloads.

---

7. Scalability Considerations

The database was designed with scalability in mind. The schema supports partitioning by date or expiry to efficiently manage large datasets. Indexing strategies ensure query performance remains optimal even as data volume increases.

The design can easily be extended to include additional exchanges, instruments, or analytics features.

---

8. Conclusion

This database design provides a scalable and efficient foundation for storing and analyzing derivatives trading data. Through proper schema design, indexing strategies, and query optimization, the system supports advanced financial analytics while maintaining performance and data integrity.

The implementation demonstrates strong understanding of database design principles, performance optimization, and time-series financial data management.