Design Reasoning – Futures & Options Database

---

1. Design Choices

The Futures & Options dataset used in this project contains millions of rows with repeated information such as exchange names, symbols, expiry dates, strike prices, and option types. Storing all this data in a single table would lead to redundancy and make the system difficult to scale.

To address this, the database was designed using normalization principles (Third Normal Form). Reference data such as exchanges and instruments was separated into dedicated tables so that repeated values are stored only once. This improves data consistency and reduces storage overhead.

The Expiry table was designed separately to handle contract-level attributes such as expiry date, strike price, and option type. Since the same contract appears across multiple trading records, separating this information avoids duplication and simplifies option chain analysis.

The Trades table was designed as the central fact table to store time-based trading data such as prices, volume, and open interest. This table is expected to grow very large and is optimized for analytical queries.

A star schema was intentionally avoided because the dataset is update-heavy and contains high-cardinality attributes. A normalized schema provides better flexibility for continuous data ingestion and reduces duplication for large Futures & Options datasets.

---

2. Table Structure Decisions

Each table was assigned a primary key to uniquely identify records. Foreign key relationships were used to maintain data integrity between exchanges, instruments, expiries, and trades.

Appropriate data types were chosen based on the nature of the data. Numeric data types were used for prices and volumes to support aggregation and analysis. Date and timestamp data types were used for expiry dates and trading timestamps to enable time-series queries.

This structure ensures data integrity while keeping the schema extensible for additional exchanges and instruments in the future.

---

3. Performance Optimizations

Since the Trades table contains high-volume time-series data, performance optimization was an important consideration. Indexes were added on frequently queried columns such as trade timestamp, exchange identifier, and instrument identifier to reduce query execution time.

The design also supports partitioning the Trades table by expiry date or exchange, which can significantly improve performance when the dataset grows beyond 10 million rows. Partitioning allows queries to scan only relevant subsets of data instead of the entire table.

These optimizations make the database suitable for analytical workloads and scalable for high-frequency trading (HFT)-like data ingestion.

---

4. Scalability Considerations

The database design separates high-cardinality and time-based data from reference data, allowing the system to scale efficiently as data volume increases. By isolating the largest dataset in the Trades table and applying indexing and partitioning strategies, the design can handle large-scale analytical queries with consistent performance.