Large Scale Time Series Order Book Data Warehouse

Generated by Doxygen 1.13.2

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 BookProcessor Class Reference

The BookProcessor class.

`#include <BookProcessor.h>`

Public Member Functions

- BookProcessor (const std::vector< std::string > &filePaths)

  Construct a new BookProcessor object.
- void process ()

  Processes all provided files concurrently.

3.1.1 Detailed Description

The BookProcessor class.

Processes raw order book update files and builds the order book. Generates time-series snapshots and stores them in a persistent binary format.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 BookProcessor()

```
BookProcessor::BookProcessor (
            const std::vector< std::string > & filePaths)  [explicit]
```

Construct a new BookProcessor object.

Parameters

filePaths	A vector of file paths containing raw order data.

3.1.3 Member Function Documentation

3.1.3.1 process()

```
void BookProcessor::process ()
```

Processes all provided files concurrently.

The documentation for this class was generated from the following files:

- Include/BookProcessor.h
- Src/BookProcessor.cpp

3.2 DescendingComparator Struct Reference

Comparator for sorting prices in descending order.

```
#include <OrderBook.h>
```

Public Member Functions

- bool operator() (double lhs, double rhs) const

3.2.1 Detailed Description

Comparator for sorting prices in descending order.

Used for the buy side to ensure the highest price appears first.

3.2.2 Member Function Documentation

3.2.2.1 operator()()

```
bool DescendingComparator::operator() (
            double lhs,
            double rhs) const  [inline]
```

The documentation for this struct was generated from the following file:

- Include/OrderBook.h

3.3 IndexEntry Struct Reference

Public Attributes

- int64_t epoch
- int64_t offset

3.3.1 Member Data Documentation

3.3.1.1 epoch

```
int64_t IndexEntry::epoch
```

3.3.1.2 offset

```
int64_t IndexEntry::offset
```

The documentation for this struct was generated from the following files:

- Src/BookProcessor.cpp
- Src/QueryEngine.cpp

3.4 Order Struct Reference

Represents a single order from the log file.

```
#include <Order.h>
```

Public Attributes

- int64_t epoch
- std::string orderId
- std::string symbol
- OrderSide side
- OrderCategory category
- double price
- int quantity

3.4.1 Detailed Description

Represents a single order from the log file.

Contains the order's timestamp (epoch in nanoseconds), unique order ID, associated symbol, side (BUY/SELL), category (NEW/CANCEL/TRADE), price, and quantity.

3.4.2 Member Data Documentation

3.4.2.1 category

```
OrderCategory Order::category
```

3.4.2.2 epoch

```
int64_t Order::epoch
```

3.4.2.3 orderId

```
std::string Order::orderId
```

3.4.2.4 price

```
double Order::price
```

3.4.2.5 quantity

```
int Order::quantity
```

3.4.2.6 side

```
OrderSide Order::side
```

3.4.2.7 symbol

```
std::string Order::symbol
```

The documentation for this struct was generated from the following file:

- Include/Order.h

3.5 OrderBook Class Reference

The OrderBook class.

```
#include <OrderBook.h>
```

Public Member Functions

- OrderBook (const std::string &symbol)

  Construct a new OrderBook object for a given symbol.
- void processOrder (const Order &order)

  Process an order update.
- Snapshot getSnapshot (int64_t epoch) const

  Get a snapshot of the current order book state.

3.5.1 Detailed Description

The OrderBook class.

Maintains the current order book for a specific symbol by processing orders (NEW, CANCEL, and TRADE) and aggregates orders into bid and ask levels. Provides snapshots representing the order book state at a given epoch.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 OrderBook()

```
OrderBook::OrderBook (
            const std::string & symbol)  [explicit]
```

Construct a new OrderBook object for a given symbol.

Parameters

symbol	The symbol associated with this order book.

3.5.3 Member Function Documentation

3.5.3.1 getSnapshot()

```
Snapshot OrderBook::getSnapshot (
            int64_t epoch) const
```

Get a snapshot of the current order book state.

The snapshot reflects the order book status after processing all orders from time 0 up to the provided epoch.

Parameters

epoch	The snapshot time in nanoseconds.

Returns

Snapshot The order book snapshot in fixed format.

3.5.3.2 processOrder()

```
void OrderBook::processOrder (
            const Order & order)
```

Process an order update.

Updates the order book state based on the order type (NEW, CANCEL, or TRADE).

Parameters

order	The order to process.

The documentation for this class was generated from the following files:

- Include/OrderBook.h
- Src/OrderBook.cpp

3.6 QueryCriteria Struct Reference

Structure representing query criteria.

`#include <QueryEngine.h>`

Public Attributes

- int64_t startEpoch
  Start of the epoch range (inclusive).
- int64_t endEpoch
  End of the epoch range (inclusive).
- std::vector< std::string > symbols
  List of symbols to query. If empty, use all known symbols.
- std::unordered_set< std::string > selectedFields
  Set of fields to output. If empty, output default grouped view.

3.6.1 Detailed Description

Structure representing query criteria.

Specifies the time range, symbols to query, and (optionally) a set of selected fields.

3.6.2 Member Data Documentation

3.6.2.1 endEpoch

`int64_t QueryCriteria::endEpoch`

End of the epoch range (inclusive).

3.6.2.2 selectedFields

`std::unordered_set<std::string> QueryCriteria::selectedFields`

Set of fields to output. If empty, output default grouped view.

3.6.2.3 startEpoch

```
int64_t QueryCriteria::startEpoch
```

Start of the epoch range (inclusive).

3.6.2.4 symbols

```
std::vector<std::string> QueryCriteria::symbols
```

List of symbols to query. If empty, use all known symbols.

The documentation for this struct was generated from the following file:

- Include/QueryEngine.h

3.7 QueryEngine Class Reference

The QueryEngine class.

```
#include <QueryEngine.h>
```

Public Member Functions

- QueryEngine (const std::vector< std::string > &symbolList)
  Constructs a QueryEngine with a list of symbols.
- std::vector< Snapshot > query (const QueryCriteria &criteria)
  Queries snapshots based on the given criteria.
- void printSnapshots (const std::vector< Snapshot > &snapshots, const QueryCriteria &criteria) const
  Prints the snapshots based on the query criteria.

3.7.1 Detailed Description

The QueryEngine class.

Provides an optimized query processing engine to access time-series order book snapshots. Supports querying by time range, symbol(s), and outputting all fields or a selective set.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 QueryEngine()

```
QueryEngine::QueryEngine (
            const std::vector< std::string > & symbolList)
```

Constructs a QueryEngine with a list of symbols.

Parameters

symbolList	List of symbols for which snapshot files exist.

3.7.3 Member Function Documentation

3.7.3.1 printSnapshots()

```
void QueryEngine::printSnapshots (
            const std::vector< Snapshot > & snapshots,
            const QueryCriteria & criteria) const
```

Prints the snapshots based on the query criteria.

If no selective fields are provided, prints the default grouped view. Otherwise, prints only the exact fields requested (with prices rounded to two decimals).

Parameters

snapshots	The snapshots to print.
criteria	The query criteria (including any selected fields).

3.7.3.2 query()

```
std::vector< Snapshot > QueryEngine::query (
            const QueryCriteria & criteria)
```

Queries snapshots based on the given criteria.

Uses an index file for each symbol to perform a binary search for fast retrieval.

Parameters

criteria	Query criteria.

Returns

std::vector<Snapshot> Filtered and sorted snapshots.

The documentation for this class was generated from the following files:

- Include/QueryEngine.h
- Src/QueryEngine.cpp

3.8 Snapshot Struct Reference

```
#include <Snapshot.h>
```

Public Attributes

- char symbol [8]
- int64_t epoch
- double bidPrices [5]
- int32_t bidQuantities [5]
- double askPrices [5]
- int32_t askQuantities [5]
- double lastTradePrice
- int32_t lastTradeQuantity

3.8.1 Member Data Documentation

3.8.1.1 askPrices

```
double Snapshot::askPrices[5]
```

3.8.1.2 askQuantities

```
int32_t Snapshot::askQuantities[5]
```

3.8.1.3 bidPrices

```
double Snapshot::bidPrices[5]
```

3.8.1.4 bidQuantities

```
int32_t Snapshot::bidQuantities[5]
```

3.8.1.5 epoch

```
int64_t Snapshot::epoch
```

3.8.1.6 lastTradePrice

```
double Snapshot::lastTradePrice
```

3.8.1.7 lastTradeQuantity

```
int32_t Snapshot::lastTradeQuantity
```

3.8.1.8 symbol

```
char Snapshot::symbol[8]
```

The documentation for this struct was generated from the following file:

- Include/Snapshot.h

Chapter 4

File Documentation

4.1 Include/BookProcessor.h File Reference

```
#include "OrderBook.h"
#include "Snapshot.h"
#include "Order.h"
#include <string>
#include <vector>
```

Classes

- class BookProcessor

  The BookProcessor class.

4.2 BookProcessor.h

Go to the documentation of this file.

```
00001 #ifndef BOOKPROCESSOR_H
00002 #define BOOKPROCESSOR_H
00003
00004 #include "OrderBook.h"
00005 #include "Snapshot.h"
00006 #include "Order.h"
00007 #include <string>
00008 #include <vector>
00009
00016 class BookProcessor {
00017 public:
00023     explicit BookProcessor(const std::vector<std::string>& filePaths);
00024
00028     void process();
00029
00030 private:
00031     std::vector<std::string> filePaths_;  // List of raw data file paths.
00032
00041     void processFile(const std::string &filePath);
00042
00052     bool parseLine(const std::string &line, Order &order);
00053
00063     void writeSnapshotBinary(const Snapshot &snapshot, const std::string &symbol);
00064 };
00065
00066 #endif
```

4.3 Include/Order.h File Reference

```
#include <string>
#include <cstdint>
```

Classes

- struct Order

  Represents a single order from the log file.

Enumerations

- enum class OrderSide { BUY , SELL }

  Enum for order side.
- enum class OrderCategory { NEW , CANCEL , TRADE }

  Enum for order category.

4.3.1 Enumeration Type Documentation

4.3.1.1 OrderCategory

```
enum class OrderCategory  [strong]
```

Enum for order category.

Enumerator

NEW	
CANCEL	
TRADE	

4.3.1.2 OrderSide

```
enum class OrderSide  [strong]
```

Enum for order side.

Enumerator

BUY	
SELL	

4.4 Order.h

Go to the documentation of this file.

```
00001 #ifndef ORDER_H
00002 #define ORDER_H
00003
00004 #include <string>
00005 #include <cstdint>
00006
00010 enum class OrderSide {
00011     BUY,
00012     SELL
00013 };
00014
00018 enum class OrderCategory {
00019     NEW,
00020     CANCEL,
00021     TRADE
00022 };
00023
00030 struct Order {
00031     int64_t epoch;
00032     std::string orderId;
00033     std::string symbol;
00034     OrderSide side;
00035     OrderCategory category;
00036     double price;
00037     int quantity;
00038 };
00039
00040 #endif
```

4.5 Include/OrderBook.h File Reference

```
#include "Order.h"
#include "Snapshot.h"
#include <map>
#include <unordered_map>
#include <string>
```

Classes

- struct DescendingComparator

  Comparator for sorting prices in descending order.
- class OrderBook

  The OrderBook class.

4.6 OrderBook.h

Go to the documentation of this file.

```
00001 #ifndef ORDERBOOK_H
00002 #define ORDERBOOK_H
00003
00004 #include "Order.h"
00005 #include "Snapshot.h"
00006 #include <map>
00007 #include <unordered_map>
00008 #include <string>
00009
00015 struct DescendingComparator {
00016     bool operator()(double lhs, double rhs) const {
00017         return lhs > rhs;
```

```
00018     }
00019 };
00020
00028 class OrderBook {
00029 public:
00035     explicit OrderBook(const std::string& symbol);
00036
00044     void processOrder(const Order& order);
00045
00055     Snapshot getSnapshot(int64_t epoch) const;
00056
00057 private:
00058     std::string symbol_;
00059
00060     // Maps to track individual orders.
00061     std::unordered_map<std::string, Order> buyOrders_;
00062     std::unordered_map<std::string, Order> sellOrders_;
00063
00064     // Aggregated bid levels (sorted in descending order) and ask levels (sorted in ascending order).
00065     std::map<double, int, DescendingComparator> buyLevels_;
00066     std::map<double, int> sellLevels_;
00067
00068     double lastTradePrice_;
00069     int lastTradeQuantity_;
00070
00078     void addOrderToBook(const Order& order);
00079
00088     void removeOrderFromBook(const Order& order, int quantityToRemove);
00089 };
00090
00091 #endif
```

4.7 Include/Progress.h File Reference

```
#include <atomic>
#include <cstdint>
```

Variables

- std::atomic< uint64_t > g_totalBytes

  Global atomic counter for the total number of bytes to be processed.
- std::atomic< uint64_t > g_bytesProcessed

  Global atomic counter for the number of bytes that have been processed so far.

4.7.1 Variable Documentation

4.7.1.1 g_bytesProcessed

```
std::atomic<uint64_t> g_bytesProcessed  [extern]
```

Global atomic counter for the number of bytes that have been processed so far.

4.7.1.2 g_totalBytes

```
std::atomic<uint64_t> g_totalBytes  [extern]
```

Global atomic counter for the total number of bytes to be processed.

4.8 Progress.h

Go to the documentation of this file.

```
00001 #ifndef PROGRESS_H
00002 #define PROGRESS_H
00003
00004 #include <atomic>
00005 #include <cstdint>
00006
00010 extern std::atomic<uint64_t> g_totalBytes;
00011
00015 extern std::atomic<uint64_t> g_bytesProcessed;
00016
00017 #endif
```

4.9 Include/QueryEngine.h File Reference

```
#include "Snapshot.h"
#include <string>
#include <vector>
#include <unordered_set>
```

Classes

- struct QueryCriteria

  Structure representing query criteria.
- class QueryEngine

  The QueryEngine class.

4.10 QueryEngine.h

Go to the documentation of this file.

```
00001 #ifndef QUERYENGINE_H
00002 #define QUERYENGINE_H
00003
00004 #include "Snapshot.h"
00005 #include <string>
00006 #include <vector>
00007 #include <unordered_set>
00008
00014 struct QueryCriteria {
00015     int64_t startEpoch;
00016     int64_t endEpoch;
00017     std::vector<std::string> symbols;
00018     std::unordered_set<std::string> selectedFields;
00019 };
00020
00027 class QueryEngine {
00028 public:
00034     QueryEngine(const std::vector<std::string>& symbolList);
00035
00044     std::vector<Snapshot> query(const QueryCriteria &criteria);
00045
00055     void printSnapshots(const std::vector<Snapshot>& snapshots, const QueryCriteria &criteria) const;
00056
00057 private:
00058     std::vector<std::string> symbolList_;
00059
00068     std::vector<Snapshot> readSnapshotsForSymbol(const std::string& symbol, int64_t startEpoch,
      int64_t endEpoch);
00069 };
00070
00071 #endif
```

4.11 Include/Snapshot.h File Reference

```
#include <cstdint>
#include <fstream>
#include <cstring>
```

Classes

- struct Snapshot

Functions

- bool writeBinarySnapshot (std::ofstream &ofs, const Snapshot &snap)
- bool readBinarySnapshot (std::ifstream &ifs, Snapshot &snap)

4.11.1 Function Documentation

4.11.1.1 readBinarySnapshot()

```
bool readBinarySnapshot (
            std::ifstream & ifs,
            Snapshot & snap)  [inline]
```

4.11.1.2 writeBinarySnapshot()

```
bool writeBinarySnapshot (
            std::ofstream & ofs,
            const Snapshot & snap)  [inline]
```

4.12 Snapshot.h

Go to the documentation of this file.

```
00001 #ifndef SNAPSHOT_H
00002 #define SNAPSHOT_H
00003
00004 #include <cstdint>
00005 #include <fstream>
00006 #include <cstring>
00007
00008 // A fixed order book snapshot structure for efficient storage and retrieval.
00009 // This structure uses fixed arrays for bid and ask levels (5 levels each).
00010 struct Snapshot {
00011     char symbol[8];            // Fixed-length symbol (zero-terminated if possible)
00012     int64_t epoch;             // Timestamp in nanoseconds
00013
00014     double bidPrices[5];       // 5 best bid prices (highest first)
00015     int32_t bidQuantities[5];  // Corresponding bid quantities
00016
00017     double askPrices[5];       // 5 best ask prices (lowest first)
00018     int32_t askQuantities[5];  // Corresponding ask quantities
00019
00020     double lastTradePrice;     // Last trade price (or -1 if none)
00021     int32_t lastTradeQuantity; // Last trade quantity (or 0 if none)
00022 };
```

```
00023
00024 // Write a Snapshot to a binary stream in fixed format.
00025 inline bool writeBinarySnapshot(std::ofstream &ofs, const Snapshot &snap) {
00026     ofs.write(reinterpret_cast<const char*>(&snap), sizeof(snap));
00027     return ofs.good();
00028 }
00029
00030 // Read a Snapshot from a binary stream in fixed format.
00031 inline bool readBinarySnapshot(std::ifstream &ifs, Snapshot &snap) {
00032     ifs.read(reinterpret_cast<char*>(&snap), sizeof(snap));
00033     return ifs.gcount() == sizeof(snap);
00034 }
00035
00036 #endif
```

4.13 Src/BookProcessor.cpp File Reference

```
#include "BookProcessor.h"
#include "Order.h"
#include "OrderBook.h"
#include "Snapshot.h"
#include "Progress.h"
#include <fstream>
#include <iostream>
#include <sstream>
#include <mutex>
#include <string>
#include <thread>
#include <vector>
```

Classes

- struct IndexEntry

Variables

- std::mutex coutMutex
- std::mutex fileWriteMutex

4.13.1 Variable Documentation

4.13.1.1 coutMutex

```
std::mutex coutMutex
```

4.13.1.2 fileWriteMutex

```
std::mutex fileWriteMutex
```

4.14 Src/main.cpp File Reference

```
#include "BookProcessor.h"
#include "QueryEngine.h"
#include "Progress.h"
#include <chrono>
#include <iostream>
#include <thread>
#include <atomic>
#include <vector>
#include <string>
#include <cstdint>
#include <sstream>
#include <iomanip>
#include <fstream>
#include <stdexcept>
```

Functions

- vector< string > split (const string &s, char delimiter)
- uint64_t getFileSize (const string &filePath)
- int main (int argc, char *argv[])

4.14.1 Function Documentation

4.14.1.1 getFileSize()

```
uint64_t getFileSize (
            const string & filePath)
```

4.14.1.2 main()

```
int main (
            int argc,
            char * argv[])
```

4.14.1.3 split()

```
vector< string > split (
            const string & s,
            char delimiter)
```

4.15 Src/OrderBook.cpp File Reference

```
#include "OrderBook.h"
#include <algorithm>
#include <iostream>
#include <cstring>
```

4.16 Src/Progress.cpp File Reference

```
#include "Progress.h"
```

Variables

- std::atomic< uint64_t > g_totalBytes {0}

  Global atomic counter for the total number of bytes to be processed.
- std::atomic< uint64_t > g_bytesProcessed {0}

  Global atomic counter for the number of bytes that have been processed so far.

4.16.1 Variable Documentation

4.16.1.1 g_bytesProcessed

```
std::atomic<uint64_t> g_bytesProcessed {0}
```

Global atomic counter for the number of bytes that have been processed so far.

4.16.1.2 g_totalBytes

```
std::atomic<uint64_t> g_totalBytes {0}
```

Global atomic counter for the total number of bytes to be processed.

4.17 Src/QueryEngine.cpp File Reference

```
#include "QueryEngine.h"
#include "Snapshot.h"
#include <fstream>
#include <iostream>
#include <algorithm>
#include <sstream>
#include <unordered_map>
#include <unordered_set>
#include <iomanip>
#include <vector>
#include <string>
```

Classes

- struct IndexEntry

Functions

- bool compareIndexEntry (const IndexEntry &a, const IndexEntry &b)
- std::string formatDouble (double value)
- std::string formatPriceLevel (int32_t quantity, double price)
- std::string formatPrice (double price)

4.17.1 Function Documentation

4.17.1.1 compareIndexEntry()

```
bool compareIndexEntry (
            const IndexEntry & a,
            const IndexEntry & b)
```

4.17.1.2 formatDouble()

```
std::string formatDouble (
            double value)
```

4.17.1.3 formatPrice()

```
std::string formatPrice (
            double price)
```

4.17.1.4 formatPriceLevel()

```
std::string formatPriceLevel (
            int32_t quantity,
            double price)
```

4.18 Tests/tests.cpp File Reference

```
#include <cassert>
#include <fstream>
#include <iostream>
#include <cstdio>
#include <vector>
#include <string>
#include <sstream>
#include <unordered_set>
#include <unordered_map>
#include <cstring>
#include "OrderBook.h"
#include "Order.h"
#include "Snapshot.h"
#include "QueryEngine.h"
#include "BookProcessor.h"
```

Functions

- bool compareBidLevel (const Snapshot &snap, int index, double expectedPrice, int expectedQuantity)
- bool compareAskLevel (const Snapshot &snap, int index, double expectedPrice, int expectedQuantity)
- void writeToFile (const string &filename, const vector< string > &lines)
- void createIndexFile (const string &symbol)
- void testOrderBook ()
- void testSnapshotSerialization ()
- void testQueryEngineDefaultOutput ()
- void testQueryEngineSelectiveOutput ()
- void testQueryEngineInvalidFields ()
- void testQueryEngineMultiSymbol ()
- void testQueryEngineNoResults ()
- void testIndexFileContent ()
- void testBookProcessorEmptyFile ()
- void testBookProcessorSingleOrder ()
- void testBookProcessorInvalidInput ()
- void testProcessAndQueryABB_CDD ()
- int main ()

4.18.1 Function Documentation

4.18.1.1 compareAskLevel()

```
bool compareAskLevel (
            const Snapshot & snap,
            int index,
            double expectedPrice,
            int expectedQuantity)
```

4.18.1.2 compareBidLevel()

```
bool compareBidLevel (
            const Snapshot & snap,
            int index,
            double expectedPrice,
            int expectedQuantity)
```

4.18.1.3 createIndexFile()

```
void createIndexFile (
            const string & symbol)
```

4.18.1.4 main()

```
int main ()
```

4.18.1.5 testBookProcessorEmptyFile()

```
void testBookProcessorEmptyFile ()
```

4.18.1.6 testBookProcessorInvalidInput()

```
void testBookProcessorInvalidInput ()
```

4.18.1.7 testBookProcessorSingleOrder()

```
void testBookProcessorSingleOrder ()
```

4.18.1.8 testIndexFileContent()

```
void testIndexFileContent ()
```

4.18.1.9 testOrderBook()

```
void testOrderBook ()
```

4.18.1.10 testProcessAndQueryABB_CDD()

```
void testProcessAndQueryABB_CDD ()
```

4.18.1.11 testQueryEngineDefaultOutput()

```
void testQueryEngineDefaultOutput ()
```

4.18.1.12 testQueryEngineInvalidFields()

```
void testQueryEngineInvalidFields ()
```

4.18.1.13 testQueryEngineMultiSymbol()

```
void testQueryEngineMultiSymbol ()
```

4.18.1.14 testQueryEngineNoResults()

```
void testQueryEngineNoResults ()
```

4.18.1.15 testQueryEngineSelectiveOutput()

```
void testQueryEngineSelectiveOutput ()
```

4.18.1.16 testSnapshotSerialization()

```
void testSnapshotSerialization ()
```

4.18.1.17 writeToFile()

```
void writeToFile (
            const string & filename,
            const vector< string > & lines)
```

Index