BlinkDB - Part 2

Generated by Doxygen 1.13.2

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Namespace Documentation

3.1 RESP Namespace Reference

Provides functions to encode various data types into RESP-2 format.

Functions

- string encodeSimpleString (const string &str)

  Encodes a simple string in RESP-2 format.
- string encodeBulkString (const string &str)

  Encodes a bulk string in RESP-2 format.
- string encodeError (const string &err)

  Encodes an error message in RESP-2 format.
- string encodeInteger (int value)

  Encodes an integer in RESP-2 format.

3.1.1 Detailed Description

Provides functions to encode various data types into RESP-2 format.

3.1.2 Function Documentation

3.1.2.1 encodeBulkString()

```
string RESP::encodeBulkString (
            const string & str)
```

Encodes a bulk string in RESP-2 format.

If the string is "null", it returns the RESP-2 null bulk string.

Parameters

str	The string to encode.

Returns

The encoded RESP bulk string.

3.1.2.2 encodeError()

```
string RESP::encodeError (
            const string & err)
```

Encodes an error message in RESP-2 format.

Parameters

err	The error message.

Returns

The encoded RESP error message.

3.1.2.3 encodeInteger()

```
string RESP::encodeInteger (
            int value)
```

Encodes an integer in RESP-2 format.

Parameters

value	The integer value.

Returns

The encoded RESP integer.

3.1.2.4 encodeSimpleString()

```
string RESP::encodeSimpleString (
            const string & str)
```

Encodes a simple string in RESP-2 format.

Parameters

str	The string to encode.

Returns

The encoded RESP simple string.

Chapter 4

Class Documentation

4.1 BlinkDBServer Class Reference

A high concurrency key-value store server using epoll and a thread pool.

Public Member Functions

- BlinkDBServer (int port, size_t threadPoolSize)

  Constructs a BlinkDBServer.
- ~**BlinkDBServer** ()

  Destructor. Closes the log file if open.
- void start ()

  Starts the TCP server.

4.1.1 Detailed Description

A high concurrency key-value store server using epoll and a thread pool.

This class sets up a TCP server that listens for incoming connections and processes RESP-2 formatted commands using a non-blocking socket and the epoll event mechanism. It utilizes a thread pool to handle client requests concurrently.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 BlinkDBServer()

```
BlinkDBServer::BlinkDBServer (
            int port,
            size_t threadPoolSize)  [inline]
```

Constructs a BlinkDBServer.

Parameters

port	The port number on which the server will listen.
threadPoolSize	The number of threads in the thread pool.

4.1.3 Member Function Documentation

4.1.3.1 start()

```
void BlinkDBServer::start ()  [inline]
```

Starts the TCP server.

This function sets up the server socket, binds to the specified port, listens for incoming connections, and uses epoll to handle multiple client connections concurrently.

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

- part2.cpp

4.2 StorageEngine Class Reference

A simple thread-safe in-memory key-value store.

Public Member Functions

- void set (const string &key, const string &value)
  Sets a key-value pair in the storage.
- string get (const string &key)
  Retrieves the value for a given key.
- void del (const string &key)
  Deletes a key from the storage.

4.2.1 Detailed Description

A simple thread-safe in-memory key-value store.

This class implements an in-memory database using an unordered_map with mutex protection for concurrent access.

4.2.2 Member Function Documentation

4.2.2.1 del()

```
void StorageEngine::del (
            const string & key)  [inline]
```

Deletes a key from the storage.

Parameters

key	The key to delete.

4.2.2.2 get()

```
string StorageEngine::get (
            const string & key)  [inline]
```

Retrieves the value for a given key.

Parameters

key	The key to retrieve.

Returns

The associated value if present; otherwise returns "null".

4.2.2.3 set()

```
void StorageEngine::set (
            const string & key,
            const string & value)  [inline]
```

Sets a key-value pair in the storage.

Parameters

key	The key to set.
value	The value to associate with the key.

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

- part2.cpp

4.3 ThreadPool Class Reference

A simple thread pool for executing tasks concurrently.

Public Member Functions

- ThreadPool (size_t threads)

  Constructs a ThreadPool with a specified number of threads.
- void enqueue (function< void()> task)

  Adds a new task to the thread pool.
- ~**ThreadPool** ()

  Destructor. Stops the thread pool and joins all worker threads.

4.3.1 Detailed Description

A simple thread pool for executing tasks concurrently.

This class creates a pool of worker threads that process tasks from a task queue. Tasks are submitted using the enqueue() method.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 ThreadPool()

```
ThreadPool::ThreadPool (
            size_t threads)  [inline]
```

Constructs a ThreadPool with a specified number of threads.

Parameters

threads	The number of threads to create in the pool.

4.3.3 Member Function Documentation

4.3.3.1 enqueue()

```
void ThreadPool::enqueue (
            function< void()> task)  [inline]
```

Adds a new task to the thread pool.

Parameters

task	A callable to be executed by one of the worker threads.

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

- part2.cpp

Index