BLINK DB 24CS60R02 - PART A

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 documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 BloomFilter Class Reference

Bloom Filter implementation for probabilistic membership testing.

`#include <bloomfilter.h>`

Public Member Functions

- void add (const std::string &key)
  Adds a key to the Bloom filter.
- bool mightContain (const std::string &key) const
  Checks if a key might be present in the Bloom filter.

Private Attributes

- std::bitset< BLOOM_FILTER_SIZE > **bitArray**
  Bit array used to store hashed elements.

3.1.1 Detailed Description

Bloom Filter implementation for probabilistic membership testing.

The Bloom Filter uses multiple hash functions to store elements in a bit array. It allows fast membership queries with a possibility of false positives but no false negatives.

3.1.2 Member Function Documentation

3.1.2.1 add()

```
void BloomFilter::add (
            const std::string & key)
```

Adds a key to the Bloom filter.

This function hashes the given key multiple times and sets the corresponding bits in the bit array.

Parameters

key	The string key to be added.

This function computes multiple hash values for the given key and sets the corresponding bits in the bit array.

Parameters

key	The string key to be added to the filter.

3.1.2.2 mightContain()

```
bool BloomFilter::mightContain (
            const std::string & key) const
```

Checks if a key might be present in the Bloom filter.

Checks if a key might be in the Bloom filter.

This function hashes the key and checks whether all the corresponding bits are set in the bit array. If all bits are set, the key might be present; otherwise, it is definitely not present.

Parameters

key	The string key to be checked.

Returns

True if the key might be present, false if it is definitely not present.

This function computes multiple hash values for the given key and checks whether all the corresponding bits are set in the bit array. If all bits are set, the key might be present; otherwise, it is definitely not present.

Parameters

key	The string key to be checked.

Returns

True if the key might be present, false if it is definitely not present.

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

- StorageEngine/bloomfilter.h
- StorageEngine/bloomfilter.cpp

3.2 LSMTree Class Reference

Log-Structured Merge (LSM) Tree implementation.

`#include <lsmtree.h>`

Public Member Functions

- LSMTree (const std::string &directory)
  Constructs an LSMTree instance with a specified SSTable directory.
- void set (const std::string &key, const std::string &value)
  Inserts a key-value pair into the LSM Tree.
- std::string get (const std::string &key)
  Retrieves the value associated with a given key.
- void remove (const std::string &key)
  Marks a key as deleted by inserting a tombstone marker.

Private Member Functions

- bool createSStableDirectory ()
  Creates the directory for storing SSTables if it does not exist.
- void flushMemtableToSSTable ()
  Flushes the memtable to SSTables when it reaches its maximum size.
- void writeSSTableToDisk (SSTable &sstable)
  Writes an SSTable to disk and adds it to the list of SSTables.

Private Attributes

- std::map< std::string, std::string > **memtable**
  In-memory key-value store (memtable)
- std::vector< SSTable > **sstables**
  List of SSTables stored on disk.
- int **sstableCounter** = 0
  Counter for naming SSTable files.
- std::string **sstableDirectory**
  Directory where SSTables are stored.

3.2.1 Detailed Description

Log-Structured Merge (LSM) Tree implementation.

The LSM Tree maintains an in-memory memtable and persistent SSTables for efficient key-value storage. It supports fast writes and range queries while leveraging Bloom filters for efficient lookups.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 LSMTree()

```
LSMTree::LSMTree (
            const std::string & directory)
```

Constructs an LSMTree instance with a specified SSTable directory.

Parameters

directory	The directory where SSTables will be stored.

Ensures the directory path ends with a separator for consistency.

Parameters

directory	The directory where SSTables will be stored.

3.2.3 Member Function Documentation

3.2.3.1 createSStableDirectory()

```
bool LSMTree::createSStableDirectory ()  [private]
```

Creates the directory for storing SSTables if it does not exist.

Returns

True if the directory is successfully created or already exists, false otherwise.

True if the directory was created or already exists, false on failure.

3.2.3.2 flushMemtableToSSTable()

```
void LSMTree::flushMemtableToSSTable ()  [private]
```

Flushes the memtable to SSTables when it reaches its maximum size.

Flushes the memtable to SSTables on disk.

The memtable entries are sorted and split into multiple SSTables if the data size exceeds the maximum SSTable limit.

3.2.3.3 get()

```
std::string LSMTree::get (
            const std::string & key)
```

Retrieves the value associated with a given key.

Parameters

key	The key to look up.

Returns

The associated value if found, otherwise "NOT_FOUND".

The lookup first checks the memtable, then searches SSTables if necessary.

Parameters

key	The key to look up.

Returns

The associated value if found, otherwise "NOT_FOUND".

3.2.3.4 remove()

```
void LSMTree::remove (
            const std::string & key)
```

Marks a key as deleted by inserting a tombstone marker.

Parameters

key	The key to remove.

If the memtable reaches its maximum size, it is flushed to an SSTable.

Parameters

key	The key to remove.

3.2.3.5 set()

```
void LSMTree::set (
            const std::string & key,
            const std::string & value)
```

Inserts a key-value pair into the LSM Tree.

Parameters

key	The key to insert.
value	The corresponding value.

If the memtable reaches its maximum size, it is flushed to an SSTable.

Parameters

key	The key to insert.
value	The corresponding value.

3.2.3.6 writeSSTableToDisk()

```
void LSMTree::writeSSTableToDisk (
            SSTable & sstable)  [private]
```

Writes an SSTable to disk and adds it to the list of SSTables.

Parameters

sstable	The SSTable to be written to disk.
sstable	The SSTable to be persisted.

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

- StorageEngine/lsmtree.h
- StorageEngine/lsmtree.cpp

3.3 SSTable Class Reference

Represents an SSTable (Sorted String Table) in the LSM tree.

`#include <sstable.h>`

Public Member Functions

- bool writeToDisk (const std::string &filename)

  Writes the SSTable data to a file on disk.
- void addEntry (const std::string &key, const std::string &value)

  Adds an entry to the SSTable and updates the Bloom filter.

Public Attributes

- BloomFilter **bloomFilter**

  Bloom filter for fast key existence checks.
- std::map< std::string, std::string > **data**

  Sorted key-value pairs stored in the SSTable.

3.3.1 Detailed Description

Represents an SSTable (Sorted String Table) in the LSM tree.

An SSTable is a persistent, immutable key-value store used in LSM trees. It maintains a sorted map of key-value pairs and a Bloom filter for fast lookups.

3.3.2 Member Function Documentation

3.3.2.1 addEntry()

```
void SSTable::addEntry (
            const std::string & key,
            const std::string & value)
```

Adds an entry to the SSTable and updates the Bloom filter.

Parameters

key	The key to insert.
value	The corresponding value.

3.3.2.2 writeToDisk()

```
bool SSTable::writeToDisk (
            const std::string & filename)
```

Writes the SSTable data to a file on disk.

Parameters

filename	The name of the file where the SSTable data will be stored.

Returns

True if the write operation was successful, false otherwise.

Ensures that the parent directory exists before writing. The file is opened in truncation mode, meaning any previous content is overwritten.

Parameters

filename	The name of the file where SSTable data will be stored.

Returns

True if the write operation was successful, false otherwise.

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

- StorageEngine/sstable.h
- StorageEngine/sstable.cpp

Chapter 4

File Documentation

4.1 bloomfilter.h

```
00001 #ifndef BLOOM_FILTER_H
00002 #define BLOOM_FILTER_H
00003
00004 #include <string>
00005 #include <bitset>
00006 #include "config.h"
00007
00018 inline size_t hash(const std::string &key, int seed);
00019
00026 class BloomFilter
00027 {
00028 private:
00029     std::bitset<BLOOM_FILTER_SIZE> bitArray;
00030
00031 public:
00040     void add(const std::string &key);
00041
00052     bool mightContain(const std::string &key) const;
00053 };
00054
00055 #endif // BLOOM_FILTER_H
```

4.2 StorageEngine/config.h File Reference

Configuration file for LSM Tree and Bloom Filter settings.

Macros

- #define **MAX_MEMTABLE_SIZE** 10000

  Maximum number of key-value pairs allowed in the memtable before flushing to disk.
- #define **MAX_SSTABLE_SIZE** 10000

  Maximum number of key-value pairs stored in each SSTable.
- #define **BLOOM_FILTER_SIZE** 200000

  Size of the Bloom filter's bit array.
- #define **BLOOM_HASH_COUNT** 9

  Number of hash functions used in the Bloom filter.

4.2.1 Detailed Description

Configuration file for LSM Tree and Bloom Filter settings.

4.3 config.h

Go to the documentation of this file.

```
00001 #ifndef CONFIG_H
00002 #define CONFIG_H
00003
00008
00012 #define MAX_MEMTABLE_SIZE 10000
00013
00017 #define MAX_SSTABLE_SIZE 10000
00018
00022 #define BLOOM_FILTER_SIZE 200000
00023
00027 #define BLOOM_HASH_COUNT 9
00028
00029 #endif // CONFIG_H
```

4.4 StorageEngine/lsmtree.h File Reference

Header file for the LSM Tree implementation.

```
#include "sstable.h"
#include <vector>
#include <string>
#include <map>
#include "config.h"
```

Classes

- class LSMTree

  Log-Structured Merge (LSM) Tree implementation.

4.4.1 Detailed Description

Header file for the LSM Tree implementation.

4.5 lsmtree.h

Go to the documentation of this file.

```
00001 #ifndef LSM_TREE_H
00002 #define LSM_TREE_H
00003
00004 #include "sstable.h"
00005 #include <vector>
00006 #include <string>
00007 #include <map>
00008 #include "config.h"
00009
00014
00022 class LSMTree
00023 {
00024 private:
00025     std::map<std::string, std::string> memtable;
00026     std::vector<SSTable> sstables;
00027     int sstableCounter = 0;
00028     std::string sstableDirectory;
00029
00034     bool createSStableDirectory();
```

```
00035
00039     void flushMemtableToSSTable();
00040
00045     void writeSSTableToDisk(SSTable &sstable);
00046
00047 public:
00052     LSMTree(const std::string &directory);
00053
00059     void set(const std::string &key, const std::string &value);
00060
00066     std::string get(const std::string &key);
00067
00072     void remove(const std::string &key);
00073 };
00074
00075 #endif // LSM_TREE_H
```

4.6 StorageEngine/sstable.h File Reference

Header file defining the SSTable class.

```
#include "bloomfilter.h"
#include <map>
#include <string>
#include "config.h"
```

Classes

- class SSTable

  Represents an SSTable (Sorted String Table) in the LSM tree.

4.6.1 Detailed Description

Header file defining the SSTable class.

4.7 sstable.h

Go to the documentation of this file.

```
00001 #ifndef SSTABLE_H
00002 #define SSTABLE_H
00003
00004 #include "bloomfilter.h"
00005 #include <map>
00006 #include <string>
00007 #include "config.h"
00008
00013
00020 class SSTable
00021 {
00022 public:
00023     BloomFilter bloomFilter;
00024     std::map<std::string, std::string> data;
00025
00032     bool writeToDisk(const std::string &filename);
00033
00040     void addEntry(const std::string &key, const std::string &value);
00041 };
00042
00043 #endif // SSTABLE_H
```

Index