BlinkDB - Part A

Generated by Doxygen 1.9.1

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 BlinkDB Class Reference

An in-memory key-value database with LRU caching and disk persistence.

`#include <blinkdb.h>`

Public Member Functions

- BlinkDB ()
  Constructor.
- ~BlinkDB ()
  Destructor.
- void set (const std::string &key, const std::string &value)
  Sets a key-value pair in the database.
- std::string get (const std::string &key)
  Retrieves a value by key.
- bool del (const std::string &key)
  Deletes a key-value pair from the database.
- void persistToFile ()
  Writes all in-memory data to disk.
- void clearPersistenceFile ()
  Deletes the persistence file.
- void flushToDiskPeriodically ()
  Background thread function that periodically flushes data to disk.
- void flushToDiskAsync ()
  Asynchronously flushes data to disk.

Private Member Functions

- void loadFromFile ()
  Loads data from persistence file into memory.
- void updateLRU (const std::string &key)
  Updates the LRU status of a key.
- void restoreFromDisk (const std::string &key)
  Restores an evicted key from disk.

Private Attributes

- std::unordered_map< std::string, std::string > store

  Main storage for key-value pairs.
- std::list< std::string > lru_keys

  List maintaining LRU order of keys.
- std::unordered_map< std::string, std::list< std::string >::iterator > lru_map

  Map for quick access to keys' positions in the LRU list.
- std::unordered_map< std::string, bool > evicted_keys

  Set of keys that have been evicted from memory but exist on disk.
- std::shared_mutex db_mutex

  Mutex for thread-safe access to the database.
- const size_t max_cache_size = MAX_CAPACITY

  Maximum number of items to keep in memory.
- const std::string persistence_file = FLUSH_FILE

  Path to the persistence file.
- bool dirty = false

  Flag indicating whether data has been modified since last flush.

3.1.1 Detailed Description

An in-memory key-value database with LRU caching and disk persistence.

BlinkDB implements a simple key-value store with an LRU (Least Recently Used) eviction policy. It provides persistence by periodically flushing data to disk and can restore evicted keys from disk when requested.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 BlinkDB()

```
BlinkDB::BlinkDB ( )
```

Constructor.

Constructor implementation.

Initializes the database and starts the background flush thread

Loads existing data from disk and starts the background flush thread

3.1.2.2 ~BlinkDB()

```
BlinkDB::~BlinkDB ( )
```

Destructor.

Destructor implementation.

Ensures any pending changes are persisted to disk

Ensures any unsaved changes are written to disk

3.1.3 Member Function Documentation

3.1.3.1 clearPersistenceFile()

```
void BlinkDB::clearPersistenceFile ( )
```

Deletes the persistence file.

Removes the persistence file from disk.

3.1.3.2 del()

```
bool BlinkDB::del (
            const std::string & key )
```

Deletes a key-value pair from the database.

Parameters

key	The key to delete

Returns

true if the key was found and deleted, false otherwise

3.1.3.3 flushToDiskAsync()

```
void BlinkDB::flushToDiskAsync ( )
```

Asynchronously flushes data to disk.

3.1.3.4 flushToDiskPeriodically()

```
void BlinkDB::flushToDiskPeriodically ( )
```

Background thread function that periodically flushes data to disk.

3.1.3.5 get()

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

Retrieves a value by key.

Parameters

key	The key to look up

Returns

The value associated with the key, or "NULL" if not found

3.1.3.6 loadFromFile()

```
void BlinkDB::loadFromFile ( )  [private]
```

Loads data from persistence file into memory.

3.1.3.7 persistToFile()

```
void BlinkDB::persistToFile ( )
```

Writes all in-memory data to disk.

3.1.3.8 restoreFromDisk()

```
void BlinkDB::restoreFromDisk (
            const std::string & key )  [private]
```

Restores an evicted key from disk.

Parameters

key	The key to restore

3.1.3.9 set()

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

Sets a key-value pair in the database.

Parameters

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

3.1.3.10 updateLRU()

```
void BlinkDB::updateLRU (
            const std::string & key )  [private]
```

Updates the LRU status of a key.

Parameters

key	The key to update in the LRU cache
key	The key to update in the LRU cache

Moves the key to the front of the LRU list and handles eviction if needed

3.1.4 Member Data Documentation

3.1.4.1 db_mutex

```
std::shared_mutex BlinkDB::db_mutex  [mutable], [private]
```

Mutex for thread-safe access to the database.

3.1.4.2 dirty

```
bool BlinkDB::dirty = false  [private]
```

Flag indicating whether data has been modified since last flush.

3.1.4.3 evicted_keys

```
std::unordered_map<std::string, bool> BlinkDB::evicted_keys  [private]
```

Set of keys that have been evicted from memory but exist on disk.

3.1.4.4 lru_keys

```
std::list<std::string> BlinkDB::lru_keys  [private]
```

List maintaining LRU order of keys.

3.1.4.5 lru_map

```
std::unordered_map<std::string, std::list<std::string>::iterator> BlinkDB::lru_map  [private]
```

Map for quick access to keys' positions in the LRU list.

3.1.4.6 max_cache_size

```
const size_t BlinkDB::max_cache_size = MAX_CAPACITY  [private]
```

Maximum number of items to keep in memory.

3.1.4.7 persistence_file

```
const std::string BlinkDB::persistence_file = FLUSH_FILE  [private]
```

Path to the persistence file.

3.1.4.8 store

```
std::unordered_map<std::string, std::string> BlinkDB::store  [private]
```

Main storage for key-value pairs.

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

- src/blinkdb.h
- src/blinkdb.cpp

Chapter 4

File Documentation

4.1 src/benchmark.cpp File Reference

Performance benchmarking for BlinkDB.

```
#include "blinkdb.h"
#include <chrono>
#include <iostream>
#include <string>
#include <sstream>
```

Include dependency graph for benchmark.cpp:

4.2 src/blinkdb.cpp File Reference

Implementation of the BlinkDB class.

```
#include "blinkdb.h"
```

Include dependency graph for blinkdb.cpp:

4.2.1 Detailed Description

Implementation of the BlinkDB class.

Author

Madhumita

Date

2025-03-31

4.3 src/blinkdb.h File Reference

Header file for the BlinkDB in-memory database with LRU caching.

```
#include <unordered_map>
#include <list>
#include <string>
#include <fstream>
#include <mutex>
#include <shared_mutex>
#include <thread>
#include <chrono>
#include <iostream>
#include <future>
#include <exception>
#include <cstdio>
```

Include dependency graph for blinkdb.h: This graph shows which files directly or indirectly include this file:

Classes

- class BlinkDB

  An in-memory key-value database with LRU caching and disk persistence.

Macros

- #define VALUE_SIZE 256
- #define MAX_CAPACITY 10000
- #define FLUSH_FILE "flush_data.txt"
- #define COMPACTION_THRESHOLD 1000

4.3.1 Detailed Description

Header file for the BlinkDB in-memory database with LRU caching.

Author

Madhumita

Date

2025-03-31

4.3.2 Macro Definition Documentation

4.3.2.1 COMPACTION_THRESHOLD

```
#define COMPACTION_THRESHOLD 1000
```

4.3.2.2 FLUSH_FILE

```
#define FLUSH_FILE "flush_data.txt"
```

4.3.2.3 MAX_CAPACITY

```
#define MAX_CAPACITY 10000
```

4.3.2.4 VALUE_SIZE

```
#define VALUE_SIZE 256
```

4.4 src/main.cpp File Reference

REPL interface for BlinkDB.

```
#include "blinkdb.h"
#include <iostream>
#include <string>
#include <sstream>
```

Include dependency graph for main.cpp:

Functions

- int main ()

  Main function implementing a REPL for BlinkDB.

4.4.1 Detailed Description

REPL interface for BlinkDB.

Author

Madhumita

Date

2025-03-31

Compilation: g++ -std=c++17 -lpthread blinkdb.cpp benchmark.cpp -o benchmark Execution: ./benchmark

4.4.2 Function Documentation

4.4.2.1 main()

```
int main ( )
```

Main function implementing a REPL for BlinkDB.

Returns

Exit code

Provides a command-line interface for interacting with BlinkDB. Supported commands:

- SET <key>

: Sets a key-value pair

- GET <key>: Retrieves a value by key
- DEL <key>: Deletes a key-value pair
- EXIT/QUIT: Exits the program

Index