

## BlinkDB - Part A

Generated by Doxygen 1.9.1



<b>1 Class Index</b>	<b>1</b>
1.1 Class List	1
<b>2 File Index</b>	<b>3</b>
2.1 File List	3
<b>3 Class Documentation</b>	<b>5</b>
3.1 BlinkDB Class Reference	5
3.1.1 Detailed Description	6
3.1.2 Constructor & Destructor Documentation	6
3.1.2.1 BlinkDB()	6
3.1.2.2 ~BlinkDB()	6
3.1.3 Member Function Documentation	7
3.1.3.1 clearPersistenceFile()	7
3.1.3.2 del()	7
3.1.3.3 flushToDiskAsync()	7
3.1.3.4 flushToDiskPeriodically()	7
3.1.3.5 get()	7
3.1.3.6 loadFromFile()	8
3.1.3.7 persistToFile()	8
3.1.3.8 restoreFromDisk()	8
3.1.3.9 set()	8
3.1.3.10 updateLRU()	9
3.1.4 Member Data Documentation	9
3.1.4.1 db_mutex	9
3.1.4.2 dirty	9
3.1.4.3 evicted_keys	9
3.1.4.4 lru_keys	10
3.1.4.5 lru_map	10
3.1.4.6 max_cache_size	10
3.1.4.7 persistence_file	10
3.1.4.8 store	10
<b>4 File Documentation</b>	<b>11</b>
4.1 src/benchmark.cpp File Reference	11
4.2 src/blinkdb.cpp File Reference	11
4.2.1 Detailed Description	11
4.3 src/blinkdb.h File Reference	12
4.3.1 Detailed Description	12
4.3.2 Macro Definition Documentation	12
4.3.2.1 COMPACTION_THRESHOLD	13
4.3.2.2 FLUSH_FILE	13
4.3.2.3 MAX_CAPACITY	13

4.3.2.4 VALUE_SIZE . . . . .	13
4.4 src/main.cpp File Reference . . . . .	13
4.4.1 Detailed Description . . . . .	13
4.4.2 Function Documentation . . . . .	14
4.4.2.1 main() . . . . .	14
<b>Index</b>	<b>15</b>

# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">BlinkDB</a>	An in-memory key-value database with LRU caching and disk persistence . . . . .	<a href="#">5</a>
-------------------------	---	-------------------



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

src/ <a href="#">benchmark.cpp</a>	
Performance benchmarking for <a href="#">BlinkDB</a> . . . . .	11
src/ <a href="#">blinkdb.cpp</a>	
Implementation of the <a href="#">BlinkDB</a> class . . . . .	11
src/ <a href="#">blinkdb.h</a>	
Header file for the <a href="#">BlinkDB</a> in-memory database with LRU caching . . . . .	12
src/ <a href="#">main.cpp</a>	
REPL interface for <a href="#">BlinkDB</a> . . . . .	13





## 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

<i>key</i>	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**

<i>key</i>	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**

<i>key</i>	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

<i>key</i>	The key to set
<i>value</i>	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

<i>key</i>	The key to update in the LRU cache
<i>key</i>	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

- ~BlinkDB
  - BlinkDB, [6](#)
- BlinkDB, [5](#)
  - ~BlinkDB, [6](#)
  - BlinkDB, [6](#)
  - clearPersistenceFile, [7](#)
  - db\_mutex, [9](#)
  - del, [7](#)
  - dirty, [9](#)
  - evicted\_keys, [9](#)
  - flushToDiskAsync, [7](#)
  - flushToDiskPeriodically, [7](#)
  - get, [7](#)
  - loadFromFile, [8](#)
  - lru\_keys, [9](#)
  - lru\_map, [10](#)
  - max\_cache\_size, [10](#)
  - persistence\_file, [10](#)
  - persistToFile, [8](#)
  - restoreFromDisk, [8](#)
  - set, [8](#)
  - store, [10](#)
  - updateLRU, [9](#)
- blinkdb.h
  - COMPACTION\_THRESHOLD, [12](#)
  - FLUSH\_FILE, [13](#)
  - MAX\_CAPACITY, [13](#)
  - VALUE\_SIZE, [13](#)
- clearPersistenceFile
  - BlinkDB, [7](#)
- COMPACTION\_THRESHOLD
  - blinkdb.h, [12](#)
- db\_mutex
  - BlinkDB, [9](#)
- del
  - BlinkDB, [7](#)
- dirty
  - BlinkDB, [9](#)
- evicted\_keys
  - BlinkDB, [9](#)
- FLUSH\_FILE
  - blinkdb.h, [13](#)
- flushToDiskAsync
  - BlinkDB, [7](#)
- flushToDiskPeriodically
  - BlinkDB, [7](#)
- get
  - BlinkDB, [7](#)
- loadFromFile
  - BlinkDB, [8](#)
- lru\_keys
  - BlinkDB, [9](#)
- lru\_map
  - BlinkDB, [10](#)
- main
  - main.cpp, [14](#)
- main.cpp
  - main, [14](#)
- max\_cache\_size
  - BlinkDB, [10](#)
- MAX\_CAPACITY
  - blinkdb.h, [13](#)
- persistence\_file
  - BlinkDB, [10](#)
- persistToFile
  - BlinkDB, [8](#)
- restoreFromDisk
  - BlinkDB, [8](#)
- set
  - BlinkDB, [8](#)
- src/benchmark.cpp, [11](#)
- src/blinkdb.cpp, [11](#)
- src/blinkdb.h, [12](#)
- src/main.cpp, [13](#)
- store
  - BlinkDB, [10](#)
- updateLRU
  - BlinkDB, [9](#)
- VALUE\_SIZE
  - blinkdb.h, [13](#)