BlinkDB - Storage Engine

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

Class Documentation

2.1 BlinkDB Class Reference

A lightweight key-value store with LRU eviction, AOF persistence, and parallel processing.

Public Member Functions

- BlinkDB (size_t cap=100)

  Constructs a BlinkDB instance with a given capacity.
- ~BlinkDB ()

  Destroys the BlinkDB instance and cleans up resources.
- void set (const string &key, const string &value)

  Sets a key-value pair in the database.
- string get (const string &key)

  Retrieves the value associated with a key.
- void del (const string &key)

  Deletes a key-value pair from the database.
- void runREPL ()

  Runs a simple Read-Eval-Print Loop (REPL) for interacting with BlinkDB.

2.1.1 Detailed Description

A lightweight key-value store with LRU eviction, AOF persistence, and parallel processing.

BlinkDB stores key-value pairs in memory with a fixed capacity. When the cache is full, the least recently used (LRU) key is evicted. All operations are logged to an append-only file (AOF) for persistence. Commands are processed concurrently using a fixed-size thread pool.

2.1.2 Constructor & Destructor Documentation

2.1.2.1 BlinkDB()

```
BlinkDB::BlinkDB (
            size_t cap = 100)  [inline]
```

Constructs a BlinkDB instance with a given capacity.

Parameters

cap	The maximum number of entries allowed in memory (default is 100).

The constructor removes any existing AOF file and launches the worker threads that will process commands concurrently.

2.1.2.2 ~BlinkDB()

```
BlinkDB::~BlinkDB ()  [inline]
```

Destroys the BlinkDB instance and cleans up resources.

Signals worker threads to stop, waits for them to join, and removes the AOF file.

2.1.3 Member Function Documentation

2.1.3.1 del()

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

Deletes a key-value pair from the database.

Parameters

key	The key to delete.

If the key exists in memory, it is removed from both the key-value store and the LRU list. The delete operation is logged in the AOF file.

2.1.3.2 get()

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

Retrieves the value associated with a key.

Parameters

key	The key to look up.

Returns

The value associated with the key, or "NULL" if the key does not exist.

The function first checks the in-memory cache. If the key is not found, it attempts to load the value from disk.

2.1.3.3 runREPL()

```
void BlinkDB::runREPL ()  [inline]
```

Runs a simple Read-Eval-Print Loop (REPL) for interacting with BlinkDB.

The REPL accepts commands (SET, GET, DEL, EXIT) from the user, enqueues them for parallel processing by worker threads, and outputs the results of GET commands.

2.1.3.4 set()

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

Sets a key-value pair in the database.

Parameters

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

If the key already exists, its value is updated and its position in the LRU list is refreshed. If the database reaches its capacity, the least recently used key is evicted and flushed to disk. The command is appended to the AOF file for persistence.

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

- main.cpp

2.2 Command Struct Reference

Structure to represent a database command for parallel processing.

Public Attributes

- string **type**

  The type of command (e.g., "SET", "GET", "DEL").
- string **key**

  The key involved in the command.
- string **value**

  The value associated with the key (for SET commands).
- promise< string > * **resultPromise**

  Pointer to a promise for returning the result of GET commands. For SET and DEL commands, this is nullptr.

2.2.1 Detailed Description

Structure to represent a database command for parallel processing.

This structure holds the details of a command that will be processed by one of the worker threads. For GET commands, a promise is used to return the result asynchronously.

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

- main.cpp

Index