

Btree

1

Generated by Doxygen 1.8.17

Fri May 15 2020 23:54:42

1 Source Code	1
2 Namespace Index	3
2.1 Namespace List	3
3 Hierarchical Index	5
3.1 Class Hierarchy	5
4 Class Index	7
4.1 Class List	7
5 File Index	9
5.1 File List	9
6 Namespace Documentation	11
6.1 bd2 Namespace Reference	11
7 Class Documentation	13
7.1 bd2::BPlusTree< T, ORDER > Class Template Reference	13
7.1.1 Detailed Description	14
7.1.2 Member Typedef Documentation	15
7.1.2.1 diskManager	15
7.1.2.2 iterator	15
7.1.2.3 node	15
7.1.3 Member Enumeration Documentation	15
7.1.3.1 state	15
7.1.4 Constructor & Destructor Documentation	15
7.1.4.1 BPlusTree() [1/2]	15
7.1.4.2 BPlusTree() [2/2]	15
7.1.4.3 ~BPlusTree()	16
7.1.5 Member Function Documentation	16
7.1.5.1 begin()	16
7.1.5.2 createNode() [1/2]	16
7.1.5.3 createNode() [2/2]	16
7.1.5.4 end()	17
7.1.5.5 find()	17
7.1.5.6 findKey() [1/2]	17
7.1.5.7 findKey() [2/2]	17
7.1.5.8 getRecordIdByKeyValue()	18
7.1.5.9 insert() [1/2]	18
7.1.5.10 insert() [2/2]	18
7.1.5.11 isKeyPresent()	19
7.1.5.12 null()	19
7.1.5.13 print() [1/2]	19

7.1.5.14 print() [2/2]	20
7.1.5.15 range_search() [1/2]	20
7.1.5.16 range_search() [2/2]	20
7.1.5.17 readNode()	20
7.1.5.18 search() [1/2]	21
7.1.5.19 search() [2/2]	21
7.1.5.20 showTree() [1/2]	21
7.1.5.21 showTree() [2/2]	21
7.1.5.22 splitNode()	22
7.1.5.23 splitRoot()	22
7.1.5.24 writeNode()	22
7.1.6 Member Data Documentation	22
7.1.6.1 disk_manager	22
7.1.6.2 header	22
7.2 bd2::BPlusTreeIterator< T, ORDER > Class Template Reference	23
7.2.1 Detailed Description	24
7.2.2 Member Typedef Documentation	24
7.2.2.1 diskManager	24
7.2.2.2 node	24
7.2.3 Constructor & Destructor Documentation	24
7.2.3.1 BPlusTreeIterator() [1/3]	24
7.2.3.2 BPlusTreeIterator() [2/3]	24
7.2.3.3 BPlusTreeIterator() [3/3]	25
7.2.4 Member Function Documentation	25
7.2.4.1 getRecordId()	25
7.2.4.2 operator!=(())	25
7.2.4.3 operator*()	25
7.2.4.4 operator++() [1/2]	26
7.2.4.5 operator++() [2/2]	26
7.2.4.6 operator--() [1/2]	26
7.2.4.7 operator--() [2/2]	26
7.2.4.8 operator=()	26
7.2.4.9 operator==(())	27
7.2.4.10 readNode()	27
7.2.5 Friends And Related Function Documentation	27
7.2.5.1 BPlusTree< T, ORDER >	27
7.2.6 Member Data Documentation	27
7.2.6.1 disk_manager	27
7.2.6.2 keys_pos	28
7.2.6.3 node_disk_id	28
7.3 bd2::Bucket_S< T, fd > Class Template Reference	28
7.3.1 Detailed Description	28

7.3.2 Member Typedef Documentation	28
7.3.2.1 value_key	28
7.3.3 Constructor & Destructor Documentation	29
7.3.3.1 Bucket_S()	29
7.3.4 Member Data Documentation	29
7.3.4.1 address	29
7.3.4.2 keys	29
7.3.4.3 NextBucket	29
7.3.4.4 size	29
7.4 bd2::DataBase< Record, Key, gd, fd > Class Template Reference	29
7.4.1 Detailed Description	30
7.4.2 Member Typedef Documentation	30
7.4.2.1 btree	30
7.4.2.2 diskManager	31
7.4.2.3 staticHashing	31
7.4.3 Constructor & Destructor Documentation	31
7.4.3.1 DataBase() [1/2]	31
7.4.3.2 DataBase() [2/2]	31
7.4.4 Member Function Documentation	31
7.4.4.1 findWithoutIndex()	31
7.4.4.2 insertThread()	32
7.4.4.3 insertWithBPlusTreeIndex()	32
7.4.4.4 insertWithoutIndex()	32
7.4.4.5 insertWithStaticHashing()	33
7.4.4.6 insertWithThreads()	33
7.4.4.7 loadFromExternalFile()	33
7.4.4.8 readRecord()	33
7.4.4.9 readRecord_SH()	33
7.4.4.10 readRecordRange()	34
7.4.4.11 showStaticHashingIndex()	34
7.4.4.12 showTreeIndex()	34
7.4.5 Member Data Documentation	34
7.4.5.1 bucketManager	34
7.4.5.2 index	35
7.4.5.3 indexManager	35
7.4.5.4 indexSH	35
7.4.5.5 kind_of_index	35
7.4.5.6 n_records	35
7.4.5.7 recordManager	35
7.5 bd2::DiskManager Class Reference	35
7.5.1 Constructor & Destructor Documentation	36
7.5.1.1 DiskManager() [1/2]	36

7.5.1.2 DiskManager() [2/2]	36
7.5.1.3 ~DiskManager()	36
7.5.2 Member Function Documentation	36
7.5.2.1 is_empty()	36
7.5.2.2 retrieve_record()	36
7.5.2.3 write_record()	37
7.5.2.4 write_record_to_ending()	37
7.5.3 Member Data Documentation	38
7.5.3.1 empty	38
7.5.3.2 filePath	38
7.6 bd2::BPlusTree< T, ORDER >:Header Struct Reference	38
7.6.1 Member Data Documentation	38
7.6.1.1 disk_id	38
7.6.1.2 n_nodes	38
7.7 bd2::Node< T, ORDER > Class Template Reference	38
7.7.1 Constructor & Destructor Documentation	39
7.7.1.1 Node() [1/2]	39
7.7.1.2 Node() [2/2]	39
7.7.2 Member Function Documentation	39
7.7.2.1 initChildrensWithZeros()	40
7.7.2.2 insertKeyInPosition()	40
7.7.2.3 isOverflow()	40
7.7.3 Friends And Related Function Documentation	40
7.7.3.1 BPlusTree< T, ORDER >	40
7.7.3.2 BPlusTreeIterator< T, ORDER >	40
7.7.4 Member Data Documentation	40
7.7.4.1 children	40
7.7.4.2 disk_id	41
7.7.4.3 is_leaf	41
7.7.4.4 keys	41
7.7.4.5 n_keys	41
7.7.4.6 next_node	41
7.7.4.7 prev_node	41
7.7.4.8 records_id	41
7.8 bd2::StaticHashing< T, gd, fd > Class Template Reference	41
7.8.1 Detailed Description	42
7.8.2 Member Typedef Documentation	42
7.8.2.1 Bucket	42
7.8.2.2 page	42
7.8.2.3 value_key	42
7.8.3 Constructor & Destructor Documentation	43
7.8.3.1 StaticHashing() [1/2]	43

7.8.3.2 StaticHashing() [2/2]	43
7.8.3.3 ~StaticHashing()	43
7.8.4 Member Function Documentation	43
7.8.4.1 getHash()	43
7.8.4.2 insert()	43
7.8.4.3 next_value()	43
7.8.4.4 print()	44
7.8.4.5 search()	44
7.8.4.6 search_by_range()	44
7.8.5 Member Data Documentation	44
7.8.5.1 control_bucket	44
7.8.5.2 control_data	45
8 File Documentation	47
8.1 b_plus_tree.h File Reference	47
8.2 b_plus_tree_iterator.h File Reference	47
8.2.1 Detailed Description	47
8.3 b_plus_tree_node.h File Reference	48
8.4 data_base_manager.h File Reference	48
8.4.1 Detailed Description	49
8.4.2 Macro Definition Documentation	49
8.4.2.1 B_ORDER	49
8.5 disk_manager.h File Reference	49
8.5.1 Detailed Description	49
8.6 README.md File Reference	50
8.7 statichashing.h File Reference	50
Index	51

Chapter 1

Source Code

Static Hashing and B+Tree on disk implementation using C++. The documentation of the following code can be generated using Doxygen just executing the following code:

```
doxygen Doxyfile
```

Then go to the latex folder created and execute `make` to get the PDF output.

Requirements

The basic requirements for this example is a conda enviroment:

Installation on LINUX/UNIX Systems

Download miniconda from <https://docs.conda.io/en/latest/miniconda.html>

```
chmod +x Miniconda3-latest-Linux-x86_64.sh
bash Miniconda3-latest-Linux-x86_64.sh
source activate base
```

Installation the following packages

```
conda install -c anaconda cmake
conda install -c conda-forge gtest
conda install -c conda-forge gmock
conda install -c hi2p-perim fmt
```

Note for osx: `brew install fmt`

Build process

```
./build.sh
```

run gtest:

```
./btree-gtest
```

or

```
cd /my_project_path/
mkdir build
cd build
cmake ..
make all
```


Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

bd2	11
-------------------------------	----

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

bd2::BPlusTree< T, ORDER >	13
bd2::BPlusTree< Key, B_ORDER >	13
bd2::BPlusTreeIterator< T, ORDER >	23
bd2::Bucket_S< T, fd >	28
bd2::DataBase< Record, Key, gd, fd >	29
fstream	
bd2::DiskManager	35
bd2::BPlusTree< T, ORDER >::Header	38
bd2::Node< T, ORDER >	38
bd2::StaticHashing< T, gd, fd >	41
bd2::StaticHashing< Key, 10000, 20 >	41

Chapter 4

Class Index

4.1 Class List

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

bd2::BPlusTree< T, ORDER >	
BPlusTree class	13
bd2::BPlusTreeIterator< T, ORDER >	
B Plus Tree Iterator Object	23
bd2::Bucket_S< T, fd >	
Bucket_S class	28
bd2::DataBase< Record, Key, gd, fd >	
Database Manager object	29
bd2::DiskManager	35
bd2::BPlusTree< T, ORDER >::Header	38
bd2::Node< T, ORDER >	38
bd2::StaticHashing< T, gd, fd >	
StaticHashing class	41

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

b_plus_tree.h	B+Tree Index Implementation based on the starting template for a B-Tree implementation by Alexander Ocsa in ADA 2019-2	47
b_plus_tree_iterator.h	B+Tree Iterators Implementation, operators ++, --, dereference * were implented	47
b_plus_tree_node.h	48
data_base_manager.h	Database manager, it permit to insert records using indexes like Static Hashing or B+Tree. It is possible to insert without indexes	48
disk_manager.h	Disk Manager Implementation, is used to read and write on file streams	49
statichashing.h	50

Chapter 6

Namespace Documentation

6.1 bd2 Namespace Reference

Classes

- class [BPlusTree](#)
BPlusTree class.
- class [BPlusTreeIterator](#)
B Plus Tree Iterator Object.
- class [Bucket_S](#)
Bucket_S class.
- class [DataBase](#)
Database Manager object.
- class [DiskManager](#)
- class [Node](#)
- class [StaticHashing](#)
StaticHashing class.

Chapter 7

Class Documentation

7.1 bd2::BPlusTree< T, ORDER > Class Template Reference

BPlusTree class.

```
#include <b_plus_tree.h>
```

Collaboration diagram for bd2::BPlusTree< T, ORDER >:

Classes

- struct [Header](#)

Public Member Functions

- [BPlusTree](#) ()
Default constructor.
- [BPlusTree](#) ([diskManager](#) d_manager)
Construct a new BPlusTree object by a disk manager object.
- void [insert](#) (const T value, const long record_id=-1)
Insert operation of a value, it calls to another insert function to store the value to a specific node and returns is a overflow occurs, if that true calls to split.
- void [showTree](#) ()
Print the tree values to the console.
- void [showTree](#) ([node](#) &ptr_node, int tree_level)
Print the tree values to the console by a tree level.
- void [print](#) (std::ostream &out)
Print just the leaf node values to the console, it is used just for testing.
- void [print](#) ([node](#) &ptr_node, int tree_level, std::ostream &out)
Print just the leaf node values to the console by a given tree level, it is used just for testing.
- [iterator begin](#) ()
Returns an iterator with the first leaf node.
- [iterator end](#) ()
Returns an iterator with the last leaf node.
- [iterator null](#) ()
Create a null iterator to check if we exceed the last or first node.
- [~BPlusTree](#) ()
- bool [isKeyPresent](#) (const T &val)
This function check is the value exist or not in the index.
- long [getRecordIdByKeyValue](#) (const T &val, int &disk_access)
Get the Record Id By Key Value object.
- void [find](#) (const T &val, long &record_id, int &key_pos)

Find a key by it value.

- long [findKey](#) ([node](#) &ptr, const T &val, int &key_pos, int &disk_access)

Function to find a in a node a key value, if not present it checks in his children until reach the leaf nodes.

- long [findKey](#) ([node](#) &ptr, const T &val, int &key_pos)

Function to search a value in a given node.

- void [search](#) (const T &val)
- int [search](#) ([node](#) &ptr, const T &val)
- std::vector< long > [range_search](#) (const T &first, const T &end)
- void [range_search](#) ([node](#) &ptr, const T &first, const T &second, std::vector< long > &res)

Protected Member Functions

- [node createNode](#) (bool isLeaf)

Create a [Node](#) object by a isLeaf flag.

- [node createNode](#) (long disk_id, bool isLeaf)

Create a [Node](#) object by a disk_id and isLeaf flag, Its create a node with the same disk id of a previous one, it is used in the split function to overwrite an splitted node with just the left child values.

- [node readNode](#) (long disk_id)

Read a node from disk by a given disk id position.

- void [writeNode](#) (long disk_id, [node](#) n)

Write a node to disk by a given disk id position.

- int [insert](#) ([node](#) &ptr_node, const T value, const long record_id)

Insert a value to a given node, to do this, it search right position for the value, and insert on it if is a leaf node, else call insert of a value to a child.

- void [splitRoot](#) ()

split the root node in left and write child, by a left based split

- void [splitNode](#) ([node](#) &parent_node, int pos)

split a node in left and write child, by a left based split

Private Types

- enum [state](#) { [OVERFLOW](#), [NORMAL](#) }
- using [node](#) = [bd2::Node](#)< T, ORDER >
- using [iterator](#) = [bd2::BPlusTreeIterator](#)< T, ORDER >
- using [diskManager](#) = std::shared_ptr< [DiskManager](#) >

Private Attributes

- [diskManager](#) [disk_manager](#)
- struct [bd2::BPlusTree::Header](#) [header](#)

7.1.1 Detailed Description

```
template<typename T, int ORDER = 3>
```

```
class bd2::BPlusTree< T, ORDER >
```

[BPlusTree](#) class.

Template Parameters

<i>T</i>	type of the key value
<i>3</i>	default b+tree order

7.1.2 Member Typedef Documentation

7.1.2.1 diskManager

```
template<typename T , int ORDER = 3>
using bd2::BPlusTree< T, ORDER >::diskManager = std::shared_ptr<DiskManager> [private]
```

7.1.2.2 iterator

```
template<typename T , int ORDER = 3>
using bd2::BPlusTree< T, ORDER >::iterator = bd2::BPlusTreeIterator<T,ORDER> [private]
```

7.1.2.3 node

```
template<typename T , int ORDER = 3>
using bd2::BPlusTree< T, ORDER >::node = bd2::Node<T,ORDER> [private]
```

7.1.3 Member Enumeration Documentation

7.1.3.1 state

```
template<typename T , int ORDER = 3>
enum bd2::BPlusTree::state [private]
```

Enumerator

OVERFLOW	
NORMAL	

7.1.4 Constructor & Destructor Documentation

7.1.4.1 BPlusTree() [1/2]

```
template<typename T , int ORDER = 3>
bd2::BPlusTree< T, ORDER >::BPlusTree ( ) [inline]
Default constructor.
```

7.1.4.2 BPlusTree() [2/2]

```
template<typename T , int ORDER = 3>
bd2::BPlusTree< T, ORDER >::BPlusTree (
    diskManager d_manager ) [inline]
```

Construct a new **BPlusTree** object by a disk manager object.

Parameters

<i>d_manager</i>	disk manager to write and read access on file
------------------	---

7.1.4.3 ~BPlusTree()

```
template<typename T , int ORDER = 3>
bd2::BPlusTree< T, ORDER >::~~BPlusTree ( ) [inline]
```

7.1.5 Member Function Documentation

7.1.5.1 begin()

```
template<typename T , int ORDER = 3>
iterator bd2::BPlusTree< T, ORDER >::begin ( ) [inline]
```

Returns an iterator with the first leaf node.

Returns

iterator

7.1.5.2 createNode() [1/2]

```
template<typename T , int ORDER = 3>
node bd2::BPlusTree< T, ORDER >::createNode (
    bool isLeaf ) [inline], [protected]
```

Create a [Node](#) object by a isLeaf flag.

Parameters

<i>isLeaf</i>	if the node is a leaf node
---------------	----------------------------

Returns

node node created

7.1.5.3 createNode() [2/2]

```
template<typename T , int ORDER = 3>
node bd2::BPlusTree< T, ORDER >::createNode (
    long disk_id,
    bool isLeaf ) [inline], [protected]
```

Create a [Node](#) object by a disk_id and isLeaf flag. Its create a node with the same disk id of a previous one, it is used in the split function to overwrite an splitted node with just the left child values.

Parameters

<i>disk_id</i>	the disk id of the previous node
<i>isLeaf</i>	is the node leaf?

Returns

node node created

7.1.5.4 end()

```
template<typename T , int ORDER = 3>
iterator bd2::BPlusTree< T, ORDER >::end ( ) [inline]
```

Returns an iterator with the last leaf node.

Returns

iterator

7.1.5.5 find()

```
template<typename T , int ORDER = 3>
void bd2::BPlusTree< T, ORDER >::find (
    const T & val,
    long & record_id,
    int & key_pos ) [inline]
```

Find a key by it value.

Parameters

<i>val</i>	key to be finded
<i>record_id</i>	position of the record on disk
<i>key_pos</i>	position in the keys array

7.1.5.6 findKey() [1/2]

```
template<typename T , int ORDER = 3>
long bd2::BPlusTree< T, ORDER >::findKey (
    node & ptr,
    const T & val,
    int & key_pos ) [inline]
```

Function to search a value in a given node.

Parameters

<i>ptr</i>	node in which we are going to find the key
<i>val</i>	value to be finded
<i>key_pos</i>	position in the keys array of the key finded

Returns

long position on disk

7.1.5.7 findKey() [2/2]

```
template<typename T , int ORDER = 3>
long bd2::BPlusTree< T, ORDER >::findKey (
```

```

    node & ptr,
    const T & val,
    int & key_pos,
    int & disk_access ) [inline]

```

Function to find a in a node a key value, if not present it checks in his children until reach the leaf nodes.

Parameters

<i>ptr</i>	node in which search the key value
<i>val</i>	key value to be finded
<i>key_pos</i>	position of the key in the keys array
<i>disk_access</i>	quantity of disk accesses

Returns

long position of the record on disk

7.1.5.8 getRecordIdByKeyValue()

```

template<typename T , int ORDER = 3>
long bd2::BPlusTree< T, ORDER >::getRecordIdByKeyValue (
    const T & val,
    int & disk_access ) [inline]

```

Get the Record Id By Key Value object.

Parameters

<i>val</i>	value to be finded
<i>disk_access</i>	variable to measure the quantity of disk access

Returns

long id of the record finded by key value, if not exist return -1

7.1.5.9 insert() [1/2]

```

template<typename T , int ORDER = 3>
void bd2::BPlusTree< T, ORDER >::insert (
    const T value,
    const long record_id = -1 ) [inline]

```

Insert operation of a value, it calls to another insert function to store the value to a specific node and returns is a overflow occurs, if that true calls to split.

Parameters

<i>value</i>	
--------------	--

7.1.5.10 insert() [2/2]

```

template<typename T , int ORDER = 3>
int bd2::BPlusTree< T, ORDER >::insert (
    node & ptr_node,

```

```
const T value,
const long record_id ) [inline], [protected]
```

Insert a value to a given node, to do this, it search right position for the value, and insert on it if is a leaf node, else call insert of a value to a child.

Parameters

<i>ptr_node</i>	
<i>value</i>	

Returns

int

7.1.5.11 isKeyPresent()

```
template<typename T , int ORDER = 3>
bool bd2::BPlusTree< T, ORDER >::isKeyPresent (
    const T & val ) [inline]
```

This function check is the value exist or not in the index.

Parameters

<i>val</i>	value to be inserted
------------	----------------------

Returns

true the value already exist
false the value doesn't exist

7.1.5.12 null()

```
template<typename T , int ORDER = 3>
iterator bd2::BPlusTree< T, ORDER >::null ( ) [inline]
```

Create a null iterator to check if we exceed the last or first node.

Returns

iterator

7.1.5.13 print() [1/2]

```
template<typename T , int ORDER = 3>
void bd2::BPlusTree< T, ORDER >::print (
    node & ptr_node,
    int tree_level,
    std::ostream & out ) [inline]
```

Print just the leaf node values to the console by a given tree level, it is used just for testing.

Parameters

<i>out</i>	
------------	--

7.1.5.14 print() [2/2]

```
template<typename T , int ORDER = 3>
void bd2::BPlusTree< T, ORDER >::print (
    std::ostream & out ) [inline]
```

Print just the leaf node values to the console, it is used just for testing.

Parameters

<i>out</i>	
------------	--

7.1.5.15 range_search() [1/2]

```
template<typename T , int ORDER = 3>
std::vector<long> bd2::BPlusTree< T, ORDER >::range_search (
    const T & first,
    const T & end ) [inline]
```

Parameters

<i>first</i>	
<i>end</i>	

Returns

std::vector<long>

7.1.5.16 range_search() [2/2]

```
template<typename T , int ORDER = 3>
void bd2::BPlusTree< T, ORDER >::range_search (
    node & ptr,
    const T & first,
    const T & second,
    std::vector< long > & res ) [inline]
```

Parameters

<i>ptr</i>	
<i>first</i>	
<i>second</i>	
<i>res</i>	

7.1.5.17 readNode()

```
template<typename T , int ORDER = 3>
node bd2::BPlusTree< T, ORDER >::readNode (
    long disk_id ) [inline], [protected]
```

Read a node from disk by a given disk id position.

Parameters

<i>disk</i> ↔	
<i>_id</i>	

Returns

node

7.1.5.18 search() [1/2]

```
template<typename T , int ORDER = 3>
void bd2::BPlusTree< T, ORDER >::search (
    const T & val ) [inline]
```

Parameters

<i>val</i>	
------------	--

7.1.5.19 search() [2/2]

```
template<typename T , int ORDER = 3>
int bd2::BPlusTree< T, ORDER >::search (
    node & ptr,
    const T & val ) [inline]
```

Parameters

<i>ptr</i>	
<i>val</i>	

Returns

int

7.1.5.20 showTree() [1/2]

```
template<typename T , int ORDER = 3>
void bd2::BPlusTree< T, ORDER >::showTree ( ) [inline]
```

Print the tree values to the console.

7.1.5.21 showTree() [2/2]

```
template<typename T , int ORDER = 3>
void bd2::BPlusTree< T, ORDER >::showTree (
    node & ptr_node,
    int tree_level ) [inline]
```

Print the tree values to the console by a tree level.

Parameters

<i>ptr_node</i>	ndoe to be printed
<i>tree_level</i>	tree level

7.1.5.22 splitNode()

```
template<typename T , int ORDER = 3>
void bd2::BPlusTree< T, ORDER >::splitNode (
    node & parent_node,
    int pos ) [inline], [protected]
```

split a node in left and write child, by a left based split

Parameters

<i>parent_node</i>	parent of the node to be splitted
<i>pos</i>	position of the node to be splitted in the parent node

7.1.5.23 splitRoot()

```
template<typename T , int ORDER = 3>
void bd2::BPlusTree< T, ORDER >::splitRoot ( ) [inline], [protected]
```

split the root node in left and write child, by a left based split

7.1.5.24 writeNode()

```
template<typename T , int ORDER = 3>
void bd2::BPlusTree< T, ORDER >::writeNode (
    long disk_id,
    node n ) [inline], [protected]
```

Write a node to disk by a given disk id position.

Parameters

<i>disk↔ _id</i>	
<i>n</i>	

7.1.6 Member Data Documentation

7.1.6.1 disk_manager

```
template<typename T , int ORDER = 3>
diskManager bd2::BPlusTree< T, ORDER >::disk_manager [private]
```

7.1.6.2 header

```
template<typename T , int ORDER = 3>
struct bd2::BPlusTree::Header bd2::BPlusTree< T, ORDER >::header [private]
```

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

- [b_plus_tree.h](#)

7.2 bd2::BPlusTreeIterator< T, ORDER > Class Template Reference

B Plus Tree Iterator Object.

```
#include <b_plus_tree_iterator.h>
```

Public Member Functions

- [BPlusTreeIterator](#) ([diskManager](#) &manager, long ndi)
Construct a new [BPlusTreeIterator](#) object.
- [BPlusTreeIterator](#) ([diskManager](#) &manager, long ndi, int _keys_pos)
Construct a new [BPlusTreeIterator](#) object.
- [BPlusTreeIterator](#) (const [BPlusTreeIterator](#) &bpti)
Construct a new [BPlusTreeIterator](#) object by and other iterator.
- [BPlusTreeIterator](#) & [operator++](#) ()
Prefix ++ operator Increase in one the value of keys_pos, if we reach the end of a node, we go to the next node.
- [BPlusTreeIterator](#) [operator++](#) (int)
Postfix ++ operator Increase in one the value of keys_pos, if we reach the end of a node, we go to the next node.
- [BPlusTreeIterator](#) & [operator--](#) ()
Prefix -- operator Decrease in one the value of keys_pos, if we reach the start -1 position of a node, we go to the previous node.
- [BPlusTreeIterator](#) [operator--](#) (int)
Postfix -- operator Decrease in one the value of keys_pos, if we reach the start -1 position of a node, we go to the previous node.
- [BPlusTreeIterator](#) & [operator=](#) (const [BPlusTreeIterator](#) &bpti)
Assing the value of one iterator to another.
- bool [operator==](#) (const [BPlusTreeIterator](#) &bpti)
Check if the two iterators are equal.
- bool [operator!=](#) (const [BPlusTreeIterator](#) &bpti)
Check is the two iterators are different.
- T [operator*](#) ()
Dereference operator, return the value of the key in the current keys_pos position.
- long [getRecordId](#) ()

Private Types

- using [node](#) = [bd2::Node](#)< T, ORDER >
- using [diskManager](#) = std::shared_ptr< [DiskManager](#) >

Private Member Functions

- [node](#) [readNode](#) (long disk_id)
Read a node from disk by a given disk id position.

Private Attributes

- long [node_disk_id](#)
- int [keys_pos](#)
- [diskManager](#) [disk_manager](#)

Friends

- class [BPlusTree](#)< T, ORDER >

7.2.1 Detailed Description

```
template<class T, int ORDER>
class bd2::BPlusTreeIterator< T, ORDER >
```

B Plus Tree Iterator Object.

Template Parameters

<i>T</i>	type of the index
<i>ORDER</i>	order of the btree

7.2.2 Member Typedef Documentation

7.2.2.1 diskManager

```
template<class T , int ORDER>
using bd2::BPlusTreeIterator< T, ORDER >::diskManager = std::shared_ptr<DiskManager> [private]
```

7.2.2.2 node

```
template<class T , int ORDER>
using bd2::BPlusTreeIterator< T, ORDER >::node = bd2::Node<T, ORDER> [private]
```

7.2.3 Constructor & Destructor Documentation

7.2.3.1 BPlusTreeIterator() [1/3]

```
template<class T , int ORDER>
bd2::BPlusTreeIterator< T, ORDER >::BPlusTreeIterator (
    diskManager & manager,
    long ndi ) [inline]
```

Construct a new [BPlusTreeIterator](#) object.

Parameters

<i>manager</i>	disk manager of the btree
<i>ndi</i>	disk id of the node

7.2.3.2 BPlusTreeIterator() [2/3]

```
template<class T , int ORDER>
bd2::BPlusTreeIterator< T, ORDER >::BPlusTreeIterator (
    diskManager & manager,
    long ndi,
    int _keys_pos ) [inline]
```

Construct a new [BPlusTreeIterator](#) object.

Parameters

<i>manager</i>	disk manager of the btree
----------------	---------------------------

Parameters

<i>ndi</i>	disk id of the node
<i>_keys_pos</i>	position on key to start the iterator

7.2.3.3 BPlusTreeIterator() [3/3]

```
template<class T , int ORDER>
bd2::BPlusTreeIterator< T, ORDER >::BPlusTreeIterator (
    const BPlusTreeIterator< T, ORDER > & bpti ) [inline]
```

Construct a new **BPlusTreeIterator** object by and other iterator.

Parameters

<i>bpti</i>	other B+Tree iterator
-------------	-----------------------

7.2.4 Member Function Documentation

7.2.4.1 getRecordId()

```
template<class T , int ORDER>
long bd2::BPlusTreeIterator< T, ORDER >::getRecordId ( ) [inline]
```

7.2.4.2 operator"!="()

```
template<class T , int ORDER>
bool bd2::BPlusTreeIterator< T, ORDER >::operator!= (
    const BPlusTreeIterator< T, ORDER > & bpti ) [inline]
```

Check is the two iterators are different.

Parameters

<i>bpti</i>	another iterator
-------------	------------------

Returns

true are different
false are equal

7.2.4.3 operator*()

```
template<class T , int ORDER>
T bd2::BPlusTreeIterator< T, ORDER >::operator* ( ) [inline]
```

Dereference operator, return the value of the key in the current *keys_pos* position.

Returns

T key value

7.2.4.4 operator++() [1/2]

```
template<class T , int ORDER>
```

```
BPlusTreeIterator& bd2::BPlusTreeIterator< T, ORDER >::operator++ ( ) [inline]
```

Prefix ++ operator Increase in one the value of keys_pos, if we reach the end of a node, we go to the next node.

Returns

[BPlusTreeIterator&](#)

7.2.4.5 operator++() [2/2]

```
template<class T , int ORDER>
```

```
BPlusTreeIterator bd2::BPlusTreeIterator< T, ORDER >::operator++ (
    int ) [inline]
```

Postfix ++ operator Increase in one the value of keys_pos, if we reach the end of a node, we go to the next node.

Returns

[BPlusTreeIterator&](#)

7.2.4.6 operator--() [1/2]

```
template<class T , int ORDER>
```

```
BPlusTreeIterator& bd2::BPlusTreeIterator< T, ORDER >::operator-- ( ) [inline]
```

Prefix – operator Decrease in one the value of keys_pos, if we reach the start -1 position of a node, we go to the previous node.

Returns

[BPlusTreeIterator&](#)

7.2.4.7 operator--() [2/2]

```
template<class T , int ORDER>
```

```
BPlusTreeIterator bd2::BPlusTreeIterator< T, ORDER >::operator-- (
    int ) [inline]
```

Postfix – operator Decrease in one the value of keys_pos, if we reach the start -1 position of a node, we go to the previous node.

Returns

[BPlusTreeIterator&](#)

7.2.4.8 operator=()

```
template<class T , int ORDER>
```

```
BPlusTreeIterator& bd2::BPlusTreeIterator< T, ORDER >::operator= (
    const BPlusTreeIterator< T, ORDER > & bpti ) [inline]
```

Assing the value of one iterator to another.

Parameters

<i>bpti</i>	
-------------	--

Returns[BPlusTreeIterator](#)&**7.2.4.9 operator==()**

```
template<class T , int ORDER>
bool bd2::BPlusTreeIterator< T, ORDER >::operator== (
    const BPlusTreeIterator< T, ORDER > & bpti ) [inline]
```

Check if the two iterators are equal.

Parameters

<i>bpti</i>	another iterator
-------------	------------------

Returns

true are equal

false are different

7.2.4.10 readNode()

```
template<class T , int ORDER>
node bd2::BPlusTreeIterator< T, ORDER >::readNode (
    long disk_id ) [inline], [private]
```

Read a node from disk by a given disk id position.

Parameters

<i>disk_id</i>	position on disk to be read
----------------	-----------------------------

Returns

node node with the read values

7.2.5 Friends And Related Function Documentation**7.2.5.1 BPlusTree< T, ORDER >**

```
template<class T , int ORDER>
friend class BPlusTree< T, ORDER > [friend]
```

7.2.6 Member Data Documentation**7.2.6.1 disk_manager**

```
template<class T , int ORDER>
diskManager bd2::BPlusTreeIterator< T, ORDER >::disk_manager [private]
```

7.2.6.2 keys_pos

```
template<class T , int ORDER>
int bd2::BPlusTreeIterator< T, ORDER >::keys_pos [private]
```

7.2.6.3 node_disk_id

```
template<class T , int ORDER>
long bd2::BPlusTreeIterator< T, ORDER >::node_disk_id [private]
```

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

- [b_plus_tree_iterator.h](#)

7.3 bd2::Bucket_S< T, fd > Class Template Reference

[Bucket_S](#) class.

```
#include <statichashing.h>
```

Public Member Functions

- [Bucket_S](#) ()

Public Attributes

- int [size](#)
- long [address](#) [fd]
- [value_key](#) keys [fd]
- long [NextBucket](#)

Private Types

- using [value_key](#) = T

7.3.1 Detailed Description

```
template<typename T, int fd>
class bd2::Bucket_S< T, fd >
```

[Bucket_S](#) class.

Template Parameters

<i>T</i>	type of the key value
<i>fd</i>	max size in each Bucket's object

7.3.2 Member Typedef Documentation

7.3.2.1 value_key

```
template<typename T , int fd>
using bd2::Bucket_S< T, fd >::value_key = T [private]
```

7.3.3 Constructor & Destructor Documentation

7.3.3.1 Bucket_S()

```
template<typename T , int fd>
bd2::Bucket_S< T, fd >::Bucket_S ( ) [inline]
```

7.3.4 Member Data Documentation

7.3.4.1 address

```
template<typename T , int fd>
long bd2::Bucket_S< T, fd >::address[fd]
```

7.3.4.2 keys

```
template<typename T , int fd>
value_key bd2::Bucket_S< T, fd >::keys[fd]
```

7.3.4.3 NextBucket

```
template<typename T , int fd>
long bd2::Bucket_S< T, fd >::NextBucket
```

7.3.4.4 size

```
template<typename T , int fd>
int bd2::Bucket_S< T, fd >::size
```

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

- [statichashing.h](#)

7.4 bd2::DataBase< Record, Key, gd, fd > Class Template Reference

Database Manager object.

```
#include <data_base_manager.h>
```

Collaboration diagram for bd2::DataBase< Record, Key, gd, fd >:

Public Member Functions

- [DataBase](#) (int k_index=0)
Construct a new Data Base object.
- [DataBase](#) ([diskManager](#) idxMan, [diskManager](#) recMan, int _n_records, int k_index=0)
Construct a new Data Base object.
- void [insertWithoutIndex](#) (Record &record)
Insert without index.
- void [findWithoutIndex](#) (Record &record, Key key_value, int &disk_access)
Sequential search without indexes.
- void [loadFromExternalFile](#) (const std::string &filename)
Load data to the Database from an external file.

- bool [insertWithBPlusTreeIndex](#) (Record &record, Key &key_value, bool checkIsTheKeyExist)
Insert with B+Tree index.
- bool [readRecord](#) (Record &record, Key key_value)
Read a record with B+Tree index.
- bool [readRecordRange](#) (std::vector< Record > &vector_record, Key first, Key last)
Make a Range Search using B+Tree.
- void [showTreeIndex](#) ()
Show the B+Tree Index to the console.
- void [insertWithStaticHashing](#) (Record &record)
Insetion with Static Hashing.
- bool [readRecord_SH](#) (Record &record, Key key_value)
Read a record with Static Hashing.
- void [showStaticHashingIndex](#) ()
- void [insertWithThreads](#) (int size, int n_threads)
- void [insertThread](#) (long begin, long end)

Private Types

- using [diskManager](#) = std::shared_ptr< [bd2::DiskManager](#) >
- using [btree](#) = [bd2::BPlusTree](#)< Key, [B_ORDER](#) >
- using [staticHashing](#) = [bd2::StaticHashing](#)< Key, gd, fd >

Private Attributes

- long [n_records](#)
- [diskManager](#) [indexManager](#)
- [diskManager](#) [recordManager](#)
- [diskManager](#) [bucketManager](#)
- [btree](#) [index](#)
- [staticHashing](#) [indexSH](#)
- int [kind_of_index](#)

7.4.1 Detailed Description

```
template<typename Record, typename Key, int gd = 10000, int fd = 20>
class bd2::DataBase< Record, Key, gd, fd >
```

Database Manager object.

Template Parameters

<i>Record</i>	structure of the record to be inserted
<i>Key</i>	the type of the record key
<i>10000</i>	global depth of the static hashing
<i>20</i>	

7.4.2 Member Typedef Documentation

7.4.2.1 btree

```
template<typename Record , typename Key , int gd = 10000, int fd = 20>
using bd2::DataBase< Record, Key, gd, fd >::btree = bd2::BPlusTree<Key, B\_ORDER> [private]
```

7.4.2.2 diskManager

```
template<typename Record , typename Key , int gd = 10000, int fd = 20>
using bd2::DataBase< Record, Key, gd, fd >::diskManager = std::shared_ptr<bd2::DiskManager>
[private]
```

7.4.2.3 staticHashing

```
template<typename Record , typename Key , int gd = 10000, int fd = 20>
using bd2::DataBase< Record, Key, gd, fd >::staticHashing = bd2::StaticHashing<Key, gd, fd>
[private]
```

7.4.3 Constructor & Destructor Documentation

7.4.3.1 DataBase() [1/2]

```
template<typename Record , typename Key , int gd = 10000, int fd = 20>
bd2::DataBase< Record, Key, gd, fd >::DataBase (
    int k_index = 0 ) [inline]
```

Construct a new Data Base object.

Parameters

<i>k_index</i>	type of index to be selected, (0) B+Tree (1)Static Hashing (else) Without Index
----------------	---

7.4.3.2 DataBase() [2/2]

```
template<typename Record , typename Key , int gd = 10000, int fd = 20>
bd2::DataBase< Record, Key, gd, fd >::DataBase (
    diskManager idxMan,
    diskManager recMan,
    int _n_records,
    int k_index = 0 ) [inline]
```

Construct a new Data Base object.

Parameters

<i>idxMan</i>	disk manager for the index
<i>recMan</i>	disk manager for the records
<i>_n_records</i>	number of records
<i>k_index</i>	type of index

7.4.4 Member Function Documentation

7.4.4.1 findWithoutIndex()

```
template<typename Record , typename Key , int gd = 10000, int fd = 20>
void bd2::DataBase< Record, Key, gd, fd >::findWithoutIndex (
```

```
Record & record,
Key key_value,
int & disk_access ) [inline]
```

Sequential search without indexes.

Parameters

<i>record</i>	record in which we are going to store the result
<i>key_value</i>	value to be finded
<i>disk_access</i>	quantity of disk access

7.4.4.2 insertThread()

```
template<typename Record , typename Key , int gd = 10000, int fd = 20>
void bd2::DataBase< Record, Key, gd, fd >::insertThread (
    long begin,
    long end ) [inline]
```

7.4.4.3 insertWithBPlusTreeIndex()

```
template<typename Record , typename Key , int gd = 10000, int fd = 20>
bool bd2::DataBase< Record, Key, gd, fd >::insertWithBPlusTreeIndex (
    Record & record,
    Key & key_value,
    bool checkIsTheKeyExist ) [inline]
```

Insert with B+Tree index.

Parameters

<i>record</i>	record to be inserted
<i>key_value</i>	key value
<i>checkIsTheKeyExist</i>	bool to check if the key already exist

Returns

true insert successfull
false insertion wrong

7.4.4.4 insertWithoutIndex()

```
template<typename Record , typename Key , int gd = 10000, int fd = 20>
void bd2::DataBase< Record, Key, gd, fd >::insertWithoutIndex (
    Record & record ) [inline]
```

Insert without index.

Parameters

<i>record</i>	record to be inserted
---------------	-----------------------

7.4.4.5 insertWithStaticHashing()

```
template<typename Record , typename Key , int gd = 10000, int fd = 20>
void bd2::DataBase< Record, Key, gd, fd >::insertWithStaticHashing (
    Record & record ) [inline]
```

Insetion with Static Hashing.

Parameters

<i>record</i>	record to be inserted
---------------	-----------------------

7.4.4.6 insertWithThreads()

```
template<typename Record , typename Key , int gd = 10000, int fd = 20>
void bd2::DataBase< Record, Key, gd, fd >::insertWithThreads (
    int size,
    int n_threads ) [inline]
```

7.4.4.7 loadFromExternalFile()

```
template<typename Record , typename Key , int gd = 10000, int fd = 20>
void bd2::DataBase< Record, Key, gd, fd >::loadFromExternalFile (
    const std::string & filename ) [inline]
```

Load data to the Database from an external file.

Parameters

<i>filename</i>	filename of the data
-----------------	----------------------

7.4.4.8 readRecord()

```
template<typename Record , typename Key , int gd = 10000, int fd = 20>
bool bd2::DataBase< Record, Key, gd, fd >::readRecord (
    Record & record,
    Key key_value ) [inline]
```

Read a record with B+Tree index.

Parameters

<i>record</i>	record in which we are going to store the result
<i>key_value</i>	key of the record finded

Returns

true the key exist

false the key doesn't exist

7.4.4.9 readRecord_SH()

```
template<typename Record , typename Key , int gd = 10000, int fd = 20>
bool bd2::DataBase< Record, Key, gd, fd >::readRecord_SH (
```

```
Record & record,
Key key_value ) [inline]
```

Read a record with Static Hashing.

Parameters

<i>record</i>	
<i>key_value</i>	

Returns

true

false

7.4.4.10 readRecordRange()

```
template<typename Record , typename Key , int gd = 10000, int fd = 20>
bool bd2::DataBase< Record, Key, gd, fd >::readRecordRange (
    std::vector< Record > & vector_record,
    Key first,
    Key last ) [inline]
```

Make a Range Search using B+Tree.

Parameters

<i>vector_record</i>	Vector in which we are going to store the result
<i>first</i>	first key value
<i>last</i>	last key value

Returns

true successfull

false wrong

7.4.4.11 showStaticHashingIndex()

```
template<typename Record , typename Key , int gd = 10000, int fd = 20>
void bd2::DataBase< Record, Key, gd, fd >::showStaticHashingIndex ( ) [inline]
```

7.4.4.12 showTreeIndex()

```
template<typename Record , typename Key , int gd = 10000, int fd = 20>
void bd2::DataBase< Record, Key, gd, fd >::showTreeIndex ( ) [inline]
```

Show the B+Tree Index to the console.

7.4.5 Member Data Documentation

7.4.5.1 bucketManager

```
template<typename Record , typename Key , int gd = 10000, int fd = 20>
diskManager bd2::DataBase< Record, Key, gd, fd >::bucketManager [private]
```

7.4.5.2 index

```
template<typename Record , typename Key , int gd = 10000, int fd = 20>
btrees bd2::DataBase< Record, Key, gd, fd >::index [private]
```

7.4.5.3 indexManager

```
template<typename Record , typename Key , int gd = 10000, int fd = 20>
diskManager bd2::DataBase< Record, Key, gd, fd >::indexManager [private]
```

7.4.5.4 indexSH

```
template<typename Record , typename Key , int gd = 10000, int fd = 20>
staticHashing bd2::DataBase< Record, Key, gd, fd >::indexSH [private]
```

7.4.5.5 kind_of_index

```
template<typename Record , typename Key , int gd = 10000, int fd = 20>
int bd2::DataBase< Record, Key, gd, fd >::kind_of_index [private]
```

7.4.5.6 n_records

```
template<typename Record , typename Key , int gd = 10000, int fd = 20>
long bd2::DataBase< Record, Key, gd, fd >::n_records [private]
```

7.4.5.7 recordManager

```
template<typename Record , typename Key , int gd = 10000, int fd = 20>
diskManager bd2::DataBase< Record, Key, gd, fd >::recordManager [private]
```

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

- [data_base_manager.h](#)

7.5 bd2::DiskManager Class Reference

```
#include <disk_manager.h>
```

Inheritance diagram for bd2::DiskManager:

Collaboration diagram for bd2::DiskManager:

Public Member Functions

- [DiskManager](#) ()
Default constructor.
- [DiskManager](#) (std::string fp, bool reset=false)
Construct a new Disk Manager object to read and write nodes from disk.
- [~DiskManager](#) ()
- template<typename Record >
void [write_record](#) (const long &n, Record ®)
Write a record to a disk file.
- template<typename Record >
long [write_record_to_ending](#) (Record ®)
Write a record to the file's end.

- `template<typename Record >`
`bool retrieve_record (const long &n, Record ®)`
Read a record from a disk file and returns if was successfully.
- `bool is_empty ()`
Function to check is the file is empty or not.

Private Attributes

- `std::string filePath`
- `bool empty`

7.5.1 Constructor & Destructor Documentation

7.5.1.1 DiskManager() [1/2]

`bd2::DiskManager::DiskManager () [inline]`
 Default constructor.

7.5.1.2 DiskManager() [2/2]

`bd2::DiskManager::DiskManager (`
`std::string fp,`
`bool reset = false) [inline]`

Construct a new Disk Manager object to read and write nodes from disk.

Parameters

<i>fp</i>	filename of the index file
<i>reset</i>	flag to truncate or not the current filename

7.5.1.3 ~DiskManager()

`bd2::DiskManager::~~DiskManager () [inline]`

7.5.2 Member Function Documentation

7.5.2.1 is_empty()

`bool bd2::DiskManager::is_empty () [inline]`
 Function to check is the file is empty or not.

Returns

- true the file is empty
- false the file has elements

7.5.2.2 retrieve_record()

`template<typename Record >`
`bool bd2::DiskManager::retrieve_record (`

```
const long & n,
Record & reg ) [inline]
```

Read a record from a disk file and returns if was successfully.

Template Parameters

<i>Record</i>	class to be read
---------------	------------------

Parameters

<i>n</i>	position in which the record is going to be read
<i>reg</i>	record to save the read value

Returns

true was successfully read
false an error occurs

7.5.2.3 write_record()

```
template<typename Record >
void bd2::DiskManager::write_record (
    const long & n,
    Record & reg ) [inline]
```

Write a record to a disk file.

Template Parameters

<i>Record</i>	class to be stored
---------------	--------------------

Parameters

<i>n</i>	position in which the record is going to be stored
<i>reg</i>	record value

7.5.2.4 write_record_to_ending()

```
template<typename Record >
long bd2::DiskManager::write_record_to_ending (
    Record & reg ) [inline]
```

Write a record to the file's end.

Template Parameters

<i>Record</i>	
---------------	--

Parameters

<i>reg</i>	
------------	--

7.5.3 Member Data Documentation

7.5.3.1 empty

```
bool bd2::DiskManager::empty [private]
```

7.5.3.2 filePath

```
std::string bd2::DiskManager::filePath [private]
```

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

- [disk_manager.h](#)

7.6 bd2::BPlusTree< T, ORDER >::Header Struct Reference

Public Attributes

- long [disk_id](#) = 1
- long [n_nodes](#) = 0

7.6.1 Member Data Documentation

7.6.1.1 disk_id

```
template<typename T , int ORDER = 3>
long bd2::BPlusTree< T, ORDER >::Header::disk_id = 1
```

7.6.1.2 n_nodes

```
template<typename T , int ORDER = 3>
long bd2::BPlusTree< T, ORDER >::Header::n_nodes = 0
```

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

- [b_plus_tree.h](#)

7.7 bd2::Node< T, ORDER > Class Template Reference

```
#include <b_plus_tree_node.h>
```

Public Member Functions

- [Node](#) (long d_id)
Construct a new [Node](#) object.
- [Node](#) (long d_id, bool is_leaf_flag)
Construct a new [Node](#) object for leaf.
- void [insertKeyInPosition](#) (int pos, const T &key_value, const long record_id)
Function to insert a key_value value in a given position.
- bool [isOverflow](#) ()
Check is the node is in overflow.

Private Member Functions

- void [initChildrensWithZeros](#) ()

Private Attributes

- T [keys](#) [ORDER+1]
- long [children](#) [ORDER+2]
- long [records_id](#) [ORDER+1]
- long [n_keys](#) = 0
- bool [is_leaf](#) = false
- long [next_node](#) = -1
- long [prev_node](#) = -1
- long [disk_id](#) = -1

Friends

- class [BPlusTree](#)< T, ORDER >
- class [BPlusTreeliterator](#)< T, ORDER >

7.7.1 Constructor & Destructor Documentation

7.7.1.1 Node() [1/2]

```
template<class T , int ORDER>
bd2::Node< T, ORDER >::Node (
    long d_id ) [inline]
```

Construct a new [Node](#) object.

Parameters

$d \leftrightarrow$ _id	diskManager id of the node on disk
----------------------------	------------------------------------

7.7.1.2 Node() [2/2]

```
template<class T , int ORDER>
bd2::Node< T, ORDER >::Node (
    long d_id,
    bool is_leaf_flag ) [inline]
```

Construct a new [Node](#) object for leaf.

Parameters

d_id	
is_leaf	

7.7.2 Member Function Documentation

7.7.2.1 initChildrensWithZeros()

```
template<class T , int ORDER>
void bd2::Node< T, ORDER >::initChildrensWithZeros ( ) [inline], [private]
```

7.7.2.2 insertKeyInPosition()

```
template<class T , int ORDER>
void bd2::Node< T, ORDER >::insertKeyInPosition (
    int pos,
    const T & key_value,
    const long record_id ) [inline]
```

Function to insert a key_value value in a given position.

Parameters

<i>key_value</i>	
<i>pos</i>	

7.7.2.3 isOverflow()

```
template<class T , int ORDER>
bool bd2::Node< T, ORDER >::isOverflow ( ) [inline]
```

Check is the node is in overflow.

Returns

true overflow
false not overflow

7.7.3 Friends And Related Function Documentation

7.7.3.1 BPlusTree< T, ORDER >

```
template<class T , int ORDER>
friend class BPlusTree< T, ORDER > [friend]
```

7.7.3.2 BPlusTreeIterator< T, ORDER >

```
template<class T , int ORDER>
friend class BPlusTreeIterator< T, ORDER > [friend]
```

7.7.4 Member Data Documentation

7.7.4.1 children

```
template<class T , int ORDER>
long bd2::Node< T, ORDER >::children[ORDER+2] [private]
```


7.7.4.2 disk_id

```
template<class T , int ORDER>
long bd2::Node< T, ORDER >::disk_id = -1 [private]
```

7.7.4.3 is_leaf

```
template<class T , int ORDER>
bool bd2::Node< T, ORDER >::is_leaf = false [private]
```

7.7.4.4 keys

```
template<class T , int ORDER>
T bd2::Node< T, ORDER >::keys[ORDER+1] [private]
```

7.7.4.5 n_keys

```
template<class T , int ORDER>
long bd2::Node< T, ORDER >::n_keys = 0 [private]
```

7.7.4.6 next_node

```
template<class T , int ORDER>
long bd2::Node< T, ORDER >::next_node = -1 [private]
```

7.7.4.7 prev_node

```
template<class T , int ORDER>
long bd2::Node< T, ORDER >::prev_node = -1 [private]
```

7.7.4.8 records_id

```
template<class T , int ORDER>
long bd2::Node< T, ORDER >::records_id[ORDER+1] [private]
```

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

- [b_plus_tree_node.h](#)

7.8 bd2::StaticHashing< T, gd, fd > Class Template Reference

[StaticHashing](#) class.

```
#include <statichashing.h>
```

Public Member Functions

- [StaticHashing](#) ()
- [StaticHashing](#) ([page](#) c_bucket, [page](#) c_data)
- [~StaticHashing](#) ()
- long [getHash](#) ([value_key](#) key)
 - convert key value to hash repesctively*
- void [insert](#) (long address_register, [value_key](#) key)
 - insert a new register in the bucket respectively, search for an specific bucket a insert in it*

- [value_key next_value](#) ([value_key](#) value)
generate a next value for an specific sort of value in key, return this next value
- long [search](#) ([value_key](#) key)
search in that specific bucket is an specific register and return the register's address
- std::vector< long > [search_by_range](#) ([value_key](#) begin, [value_key](#) end)
search for a set of values what register exists and return the registers' address
- void [print](#) ()
search for a set of values what register exists and return the registers' address

Private Types

- using [page](#) = std::shared_ptr< [DiskManager](#) >
- using [value_key](#) = T
- using [Bucket](#) = [Bucket_S](#)< T, fd >

Private Attributes

- [page control_bucket](#)
- [page control_data](#)

7.8.1 Detailed Description

```
template<typename T, int gd, int fd>
class bd2::StaticHashing< T, gd, fd >
```

[StaticHashing](#) class.

Template Parameters

<i>T</i>	type of the key value
<i>gd</i>	global depth in Index
<i>fd</i>	max size in each Bucket's object

7.8.2 Member Typedef Documentation

7.8.2.1 Bucket

```
template<typename T , int gd, int fd>
using bd2::StaticHashing< T, gd, fd >::Bucket = Bucket\_S<T,fd> [private]
```

7.8.2.2 page

```
template<typename T , int gd, int fd>
using bd2::StaticHashing< T, gd, fd >::page = std::shared_ptr<DiskManager> [private]
```

7.8.2.3 value_key

```
template<typename T , int gd, int fd>
using bd2::StaticHashing< T, gd, fd >::value_key = T [private]
```

7.8.3 Constructor & Destructor Documentation

7.8.3.1 StaticHashing() [1/2]

```
template<typename T , int gd, int fd>
bd2::StaticHashing< T, gd, fd >::StaticHashing ( ) [inline]
```

7.8.3.2 StaticHashing() [2/2]

```
template<typename T , int gd, int fd>
bd2::StaticHashing< T, gd, fd >::StaticHashing (
    page c_bucket,
    page c_data ) [inline]
```

7.8.3.3 ~StaticHashing()

```
template<typename T , int gd, int fd>
bd2::StaticHashing< T, gd, fd >::~~StaticHashing ( ) [inline]
```

7.8.4 Member Function Documentation

7.8.4.1 getHash()

```
template<typename T , int gd, int fd>
long bd2::StaticHashing< T, gd, fd >::getHash (
    value_key key ) [inline]
```

convert key value to hash repectively

Parameters

<i>key</i>	key's value of the register
------------	-----------------------------

7.8.4.2 insert()

```
template<typename T , int gd, int fd>
void bd2::StaticHashing< T, gd, fd >::insert (
    long address_register,
    value_key key ) [inline]
```

insert a new register in the bucket respectively, search for an specific bucket a insert in it

Parameters

<i>address_register</i>	the adddress of register saved priorly
<i>key</i>	value of key registered

7.8.4.3 next_value()

```
template<typename T , int gd, int fd>
value_key bd2::StaticHashing< T, gd, fd >::next_value (
```

```
value_key value ) [inline]
```

generate a next value for an specific sort of value in key, return this next value

Parameters

<i>value</i>	value of key
--------------	--------------

7.8.4.4 print()

```
template<typename T , int gd, int fd>
```

```
void bd2::StaticHashing< T, gd, fd >::print ( ) [inline]
```

search for a set of values what register exists and return the registers' address

Parameters

<i>value</i>	value of key
--------------	--------------

7.8.4.5 search()

```
template<typename T , int gd, int fd>
```

```
long bd2::StaticHashing< T, gd, fd >::search (
```

```
value_key key ) [inline]
```

search in that specific bucket is an specific register and return the register's address

Parameters

<i>key</i>	value of key
------------	--------------

7.8.4.6 search_by_range()

```
template<typename T , int gd, int fd>
```

```
std::vector<long> bd2::StaticHashing< T, gd, fd >::search_by_range (
```

```
value_key begin,
```

```
value_key end ) [inline]
```

search for a set of values what register exists and return the registers' address

Parameters

<i>begin</i>	lower bound of searched keys
<i>end</i>	upper bound of searched keys

7.8.5 Member Data Documentation

7.8.5.1 control_bucket

```
template<typename T , int gd, int fd>
```

```
page bd2::StaticHashing< T, gd, fd >::control_bucket [private]
```

7.8.5.2 control_data

```
template<typename T , int gd, int fd>  
page bd2::StaticHashing< T, gd, fd >::control_data [private]
```

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

- [statichashing.h](#)

Chapter 8

File Documentation

8.1 b_plus_tree.h File Reference

B+Tree Index Implementation based on the starting template for a B-Tree implementation by Alexander Ocsa in ADA 2019-2.

```
#include "disk_manager.h"
#include "b_plus_tree_node.h"
#include "b_plus_tree_iterator.h"
#include <memory>
#include <iostream>
#include <vector>
#include <cmath>
```

Include dependency graph for b_plus_tree.h:

8.2 b_plus_tree_iterator.h File Reference

B+Tree Iterators Implementation, operators ++, –, dereference * were implented.

```
#include "disk_manager.h"
#include <memory>
#include <iostream>
#include <vector>
#include <cmath>
```

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

Classes

- class [bd2::BPlusTree< T, ORDER >](#)
BPlusTree class.
- class [bd2::BPlusTreeIterator< T, ORDER >](#)
B Plus Tree Iterator Object.

Namespaces

- [bd2](#)

8.2.1 Detailed Description

B+Tree Iterators Implementation, operators ++, –, dereference * were implented.

Author

Juan Vargas Castillo (juan.vargas@utec.edu.pe)
Giordano Alvitez Falcón (giordano.alvitez@utec.edu.pe)
Roosevelt.Ubaldo Chavez (roosevelt.ubaldo@utec.edu.pe)

Version

0.1

Date

2020-05-12

Copyright

Copyright (c) 2020

8.3 `b_plus_tree_node.h` File Reference

This graph shows which files directly or indirectly include this file:

Classes

- class `bd2::BPlusTree< T, ORDER >`
BPlusTree class.
- class `bd2::BPlusTreeIterator< T, ORDER >`
B Plus Tree Iterator Object.
- class `bd2::Node< T, ORDER >`

Namespaces

- `bd2`

8.4 `data_base_manager.h` File Reference

Database manager, it permit to insert records using indexes like Static Hashing or B+Tree. It is possible to insert without indexes.

```
#include "b_plus_tree.h"  
#include "statichashing.h"  
#include <string>  
#include <fstream>  
#include <sstream>  
#include <utility>  
#include <thread>
```

Include dependency graph for `data_base_manager.h`:

Classes

- class `bd2::DataBase< Record, Key, gd, fd >`
Database Manager object.

Namespaces

- `bd2`

Macros

- `#define B_ORDER 1000`

8.4.1 Detailed Description

Database manager, it permit to insert records using indexes like Static Hashing or B+Tree. It is possible to insert without indexes.

Author

Juan Vargas Castillo (juan.vargas@utec.edu.pe)

Giordano Alvitez Falcón (giordano.alvitez@utec.edu.pe)

Roosevelt.Ubaldo Chavez (roosevelt.ubaldo@utec.edu.pe)

Version

0.1

Date

2020-05-15

Copyright

Copyright (c) 2020

8.4.2 Macro Definition Documentation

8.4.2.1 B_ORDER

```
#define B_ORDER 1000
```

8.5 disk_manager.h File Reference

Disk Manager Implementation, is used to read and write on file streams.

```
#include <cstdlib>
#include <fstream>
#include <iostream>
#include <string>
```

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

Classes

- class [bd2::DiskManager](#)

Namespaces

- [bd2](#)

8.5.1 Detailed Description

Disk Manager Implementation, is used to read and write on file streams.

Author

Juan Vargas Castillo (juan.vargas@utec.edu.pe)
Giordano Alvitez Falcón (giordano.alvitez.com)
Roosevelt.Ubaldo Chavez (roosevelt.ubaldo@utec.edu.pe)

Version

0.1

Date

2020-05-12

Copyright

Copyright (c) 2020

8.6 README.md File Reference

8.7 statchashing.h File Reference

```
#include "disk_manager.h"  
#include <memory>  
#include <queue>  
#include <vector>  
#include <iostream>
```

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

Classes

- class [bd2::Bucket_S< T, fd >](#)
[Bucket_S](#) class.
- class [bd2::StaticHashing< T, gd, fd >](#)
[StaticHashing](#) class.

Namespaces

- [bd2](#)

Index

- ~BPlusTree
 - bd2::BPlusTree< T, ORDER >, 15
- ~DiskManager
 - bd2::DiskManager, 36
- ~StaticHashing
 - bd2::StaticHashing< T, gd, fd >, 43
- address
 - bd2::Bucket_S< T, fd >, 29
- B_ORDER
 - data_base_manager.h, 49
- b_plus_tree.h, 47
- b_plus_tree_iterator.h, 47
- b_plus_tree_node.h, 48
- bd2, 11
- bd2::BPlusTree< T, ORDER >, 13
 - ~BPlusTree, 15
 - begin, 16
 - BPlusTree, 15
 - createNode, 16
 - disk_manager, 22
 - diskManager, 15
 - end, 17
 - find, 17
 - findKey, 17
 - getRecordIdByKeyValue, 18
 - header, 22
 - insert, 18
 - isKeyPresent, 19
 - iterator, 15
 - node, 15
 - NORMAL, 15
 - null, 19
 - OVERFLOW, 15
 - print, 19
 - range_search, 20
 - readNode, 20
 - search, 21
 - showTree, 21
 - splitNode, 22
 - splitRoot, 22
 - state, 15
 - writeNode, 22
- bd2::BPlusTree< T, ORDER >::Header, 38
 - disk_id, 38
 - n_nodes, 38
- bd2::BPlusTreeIterator< T, ORDER >, 23
 - BPlusTree< T, ORDER >, 27
 - BPlusTreeIterator, 24, 25
- disk_manager, 27
- diskManager, 24
- getRecordId, 25
- keys_pos, 27
- node, 24
- node_disk_id, 28
- operator!=, 25
- operator*, 25
- operator++, 25, 26
- operator--, 26
- operator=, 26
- operator==, 27
- readNode, 27
- bd2::Bucket_S< T, fd >, 28
 - address, 29
 - Bucket_S, 29
 - keys, 29
 - NextBucket, 29
 - size, 29
 - value_key, 28
- bd2::DataBase< Record, Key, gd, fd >, 29
 - btree, 30
 - bucketManager, 34
 - DataBase, 31
 - diskManager, 31
 - findWithoutIndex, 31
 - index, 34
 - indexManager, 35
 - indexSH, 35
 - insertThread, 32
 - insertWithBPlusTreeIndex, 32
 - insertWithoutIndex, 32
 - insertWithStaticHashing, 32
 - insertWithThreads, 33
 - kind_of_index, 35
 - loadFromExternalFile, 33
 - n_records, 35
 - readRecord, 33
 - readRecord_SH, 33
 - readRecordRange, 34
 - recordManager, 35
 - showStaticHashingIndex, 34
 - showTreeIndex, 34
 - staticHashing, 31
- bd2::DiskManager, 35
 - ~DiskManager, 36
 - DiskManager, 36
 - empty, 38
 - filePath, 38

- is_empty, 36
- retrieve_record, 36
- write_record, 37
- write_record_to_ending, 37
- bd2::Node< T, ORDER >, 38
 - BPlusTree< T, ORDER >, 40
 - BPlusTreeliterator< T, ORDER >, 40
 - children, 40
 - disk_id, 40
 - initChildrensWithZeros, 39
 - insertKeyInPosition, 40
 - is_leaf, 41
 - isOverflow, 40
 - keys, 41
 - n_keys, 41
 - next_node, 41
 - Node, 39
 - prev_node, 41
 - records_id, 41
- bd2::StaticHashing< T, gd, fd >, 41
 - ~StaticHashing, 43
 - Bucket, 42
 - control_bucket, 44
 - control_data, 44
 - getHash, 43
 - insert, 43
 - next_value, 43
 - page, 42
 - print, 44
 - search, 44
 - search_by_range, 44
 - StaticHashing, 43
 - value_key, 42
- begin
 - bd2::BPlusTree< T, ORDER >, 16
- BPlusTree
 - bd2::BPlusTree< T, ORDER >, 15
- BPlusTree< T, ORDER >
 - bd2::BPlusTreeliterator< T, ORDER >, 27
 - bd2::Node< T, ORDER >, 40
- BPlusTreeliterator
 - bd2::BPlusTreeliterator< T, ORDER >, 24, 25
- BPlusTreeliterator< T, ORDER >
 - bd2::Node< T, ORDER >, 40
- btree
 - bd2::DataBase< Record, Key, gd, fd >, 30
- Bucket
 - bd2::StaticHashing< T, gd, fd >, 42
- Bucket_S
 - bd2::Bucket_S< T, fd >, 29
- bucketManager
 - bd2::DataBase< Record, Key, gd, fd >, 34
- children
 - bd2::Node< T, ORDER >, 40
- control_bucket
 - bd2::StaticHashing< T, gd, fd >, 44
- control_data
 - bd2::StaticHashing< T, gd, fd >, 44
- createNode
 - bd2::BPlusTree< T, ORDER >, 16
- data_base_manager.h, 48
 - B_ORDER, 49
- DataBase
 - bd2::DataBase< Record, Key, gd, fd >, 31
- disk_id
 - bd2::BPlusTree< T, ORDER >::Header, 38
 - bd2::Node< T, ORDER >, 40
- disk_manager
 - bd2::BPlusTree< T, ORDER >, 22
 - bd2::BPlusTreeliterator< T, ORDER >, 27
- disk_manager.h, 49
- DiskManager
 - bd2::DiskManager, 36
- diskManager
 - bd2::BPlusTree< T, ORDER >, 15
 - bd2::BPlusTreeliterator< T, ORDER >, 24
 - bd2::DataBase< Record, Key, gd, fd >, 31
- empty
 - bd2::DiskManager, 38
- end
 - bd2::BPlusTree< T, ORDER >, 17
- filePath
 - bd2::DiskManager, 38
- find
 - bd2::BPlusTree< T, ORDER >, 17
- findKey
 - bd2::BPlusTree< T, ORDER >, 17
- findWithoutIndex
 - bd2::DataBase< Record, Key, gd, fd >, 31
- getHash
 - bd2::StaticHashing< T, gd, fd >, 43
- getRecordId
 - bd2::BPlusTreeliterator< T, ORDER >, 25
- getRecordIdByKeyValue
 - bd2::BPlusTree< T, ORDER >, 18
- header
 - bd2::BPlusTree< T, ORDER >, 22
- index
 - bd2::DataBase< Record, Key, gd, fd >, 34
- indexManager
 - bd2::DataBase< Record, Key, gd, fd >, 35
- indexSH
 - bd2::DataBase< Record, Key, gd, fd >, 35
- initChildrensWithZeros
 - bd2::Node< T, ORDER >, 39
- insert
 - bd2::BPlusTree< T, ORDER >, 18
 - bd2::StaticHashing< T, gd, fd >, 43
- insertKeyInPosition
 - bd2::Node< T, ORDER >, 40
- insertThread
 - bd2::DataBase< Record, Key, gd, fd >, 32

- insertWithBPlusTreeIndex
 - bd2::DataBase< Record, Key, gd, fd >, [32](#)
- insertWithoutIndex
 - bd2::DataBase< Record, Key, gd, fd >, [32](#)
- insertWithStaticHashing
 - bd2::DataBase< Record, Key, gd, fd >, [32](#)
- insertWithThreads
 - bd2::DataBase< Record, Key, gd, fd >, [33](#)
- is_empty
 - bd2::DiskManager, [36](#)
- is_leaf
 - bd2::Node< T, ORDER >, [41](#)
- isKeyPresent
 - bd2::BPlusTree< T, ORDER >, [19](#)
- isOverflow
 - bd2::Node< T, ORDER >, [40](#)
- iterator
 - bd2::BPlusTree< T, ORDER >, [15](#)
- keys
 - bd2::Bucket_S< T, fd >, [29](#)
 - bd2::Node< T, ORDER >, [41](#)
- keys_pos
 - bd2::BPlusTreeIterator< T, ORDER >, [27](#)
- kind_of_index
 - bd2::DataBase< Record, Key, gd, fd >, [35](#)
- loadFromExternalFile
 - bd2::DataBase< Record, Key, gd, fd >, [33](#)
- n_keys
 - bd2::Node< T, ORDER >, [41](#)
- n_nodes
 - bd2::BPlusTree< T, ORDER >::Header, [38](#)
- n_records
 - bd2::DataBase< Record, Key, gd, fd >, [35](#)
- next_node
 - bd2::Node< T, ORDER >, [41](#)
- next_value
 - bd2::StaticHashing< T, gd, fd >, [43](#)
- NextBucket
 - bd2::Bucket_S< T, fd >, [29](#)
- Node
 - bd2::Node< T, ORDER >, [39](#)
- node
 - bd2::BPlusTree< T, ORDER >, [15](#)
 - bd2::BPlusTreeIterator< T, ORDER >, [24](#)
- node_disk_id
 - bd2::BPlusTreeIterator< T, ORDER >, [28](#)
- NORMAL
 - bd2::BPlusTree< T, ORDER >, [15](#)
- null
 - bd2::BPlusTree< T, ORDER >, [19](#)
- operator!=
 - bd2::BPlusTreeIterator< T, ORDER >, [25](#)
- operator*
 - bd2::BPlusTreeIterator< T, ORDER >, [25](#)
- operator++
 - bd2::BPlusTreeIterator< T, ORDER >, [25](#)
 - bd2::BPlusTreeIterator< T, ORDER >, [26](#)
- operator--
 - bd2::BPlusTreeIterator< T, ORDER >, [26](#)
- operator=
 - bd2::BPlusTreeIterator< T, ORDER >, [26](#)
- operator==
 - bd2::BPlusTreeIterator< T, ORDER >, [27](#)
- OVERFLOW
 - bd2::BPlusTree< T, ORDER >, [15](#)
- page
 - bd2::StaticHashing< T, gd, fd >, [42](#)
- prev_node
 - bd2::Node< T, ORDER >, [41](#)
- print
 - bd2::BPlusTree< T, ORDER >, [19](#)
 - bd2::StaticHashing< T, gd, fd >, [44](#)
- range_search
 - bd2::BPlusTree< T, ORDER >, [20](#)
- README.md, [50](#)
- readNode
 - bd2::BPlusTree< T, ORDER >, [20](#)
 - bd2::BPlusTreeIterator< T, ORDER >, [27](#)
- readRecord
 - bd2::DataBase< Record, Key, gd, fd >, [33](#)
- readRecord_SH
 - bd2::DataBase< Record, Key, gd, fd >, [33](#)
- readRecordRange
 - bd2::DataBase< Record, Key, gd, fd >, [34](#)
- recordManager
 - bd2::DataBase< Record, Key, gd, fd >, [35](#)
- records_id
 - bd2::Node< T, ORDER >, [41](#)
- retrieve_record
 - bd2::DiskManager, [36](#)
- search
 - bd2::BPlusTree< T, ORDER >, [21](#)
 - bd2::StaticHashing< T, gd, fd >, [44](#)
- search_by_range
 - bd2::StaticHashing< T, gd, fd >, [44](#)
- showStaticHashingIndex
 - bd2::DataBase< Record, Key, gd, fd >, [34](#)
- showTree
 - bd2::BPlusTree< T, ORDER >, [21](#)
- showTreeIndex
 - bd2::DataBase< Record, Key, gd, fd >, [34](#)
- size
 - bd2::Bucket_S< T, fd >, [29](#)
- splitNode
 - bd2::BPlusTree< T, ORDER >, [22](#)
- splitRoot
 - bd2::BPlusTree< T, ORDER >, [22](#)
- state
 - bd2::BPlusTree< T, ORDER >, [15](#)
- StaticHashing
 - bd2::StaticHashing< T, gd, fd >, [43](#)
- staticHashing

bd2::DataBase< Record, Key, gd, fd >, [31](#)
statichashing.h, [50](#)

value_key
bd2::Bucket_S< T, fd >, [28](#)
bd2::StaticHashing< T, gd, fd >, [42](#)

write_record
bd2::DiskManager, [37](#)

write_record_to_ending
bd2::DiskManager, [37](#)

writeNode
bd2::BPlusTree< T, ORDER >, [22](#)