LevelDB-Study

Team_Cache Analysis

Made by Subin Hong, Seungwon Ha

E-Mail: zed6740@dankook.ac.kr, 12gktmddnjs@naver.com

Contents

Cache flow analysis

- Specific code analysis

Cache flow analysis

```
Cache::Handle* handle = nullptr;
Status s = FindTable(file_number, file_size, &handle);
if (s.ok()) {
  Table* t = reinterpret_cast<TableAndFile*>(cache_->Value(handle))->table;
  s = t->InternalGet(options, k, arg, handle_result);
  cache_->Release(handle);
}
return s;
```

Cache flow analysis

```
~T~\ ~T~@(1000) leveldb::_GLOBAL__N_1::ShardedLRUCache::Lookup
~T~B (1000) leveldb::TableCache::FindTable
~T~B (1000) leveldb::TableCache::Get
~T~B (1000) leveldb::Version::Get::State::Match
~T~B (1000) leveldb::Version::ForEachOverlapping
~T~B (1000) leveldb::Version::Get
~T~B (1000) leveldb::DBImpl::Get
~T~B (1000) leveldb::Benchmark::ReadRandom
```

```
~T~T ~T~@(1000) leveldb::_GLOBAL__N_1::ShardedLRUCache::Lookup
 (1000) leveldb::Table::BlockReader
 (1000) leveldb::Table::InternalGet
 (1000) leveldb::TableCache::Get
 (1000) leveldb::Version::Get::State::Match
 (1000) leveldb::Version::ForEachOverlapping
 (1000) leveldb::Version::Get
 (1000) leveldb::DBImpl::Get
 (1000) leveldb::Benchmark::ReadRandom
```

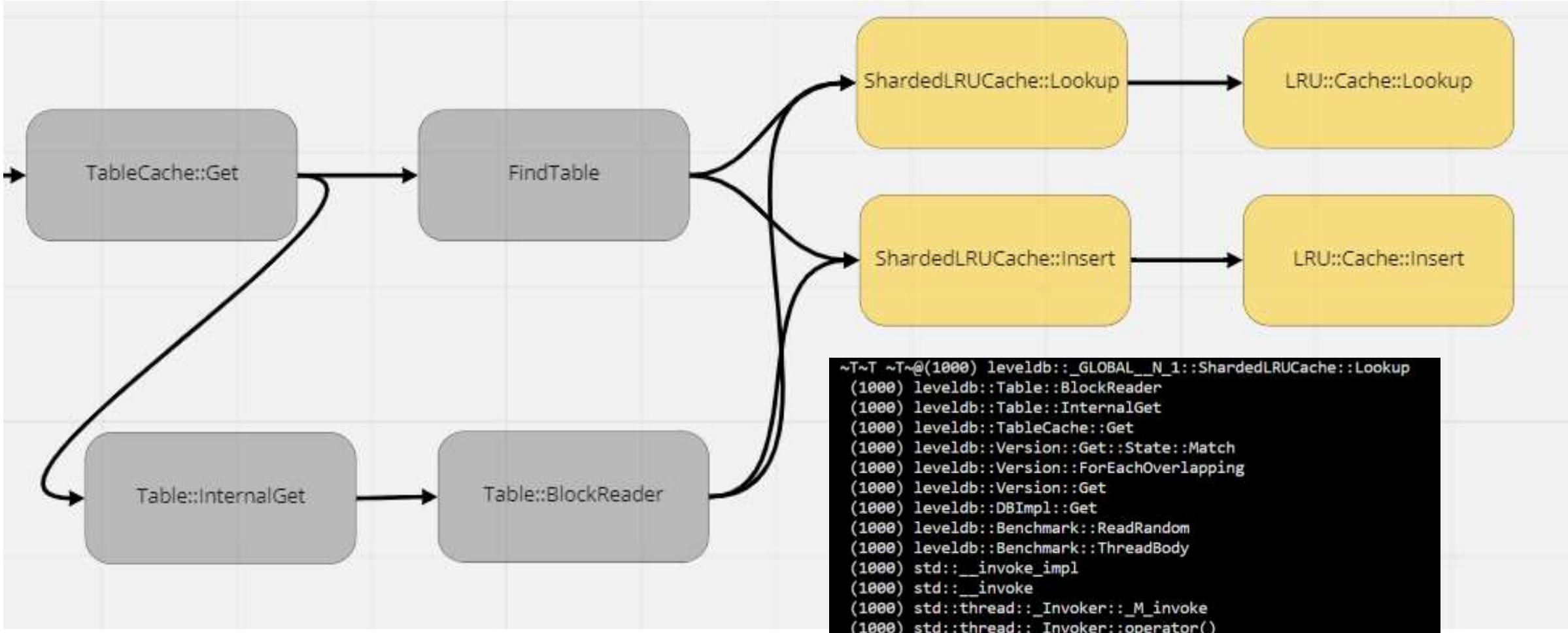
```
Status TableCache::FindTable(uint64_t file_number, uint64_t file_size,
                             Cache::Handle** handle) {
```

```
Iterator* Table::BlockReader(void* arg, const ReadOptions& options,
                             const Slice& index_value) {
```

```
*handle = cache_->Lookup(key);
```

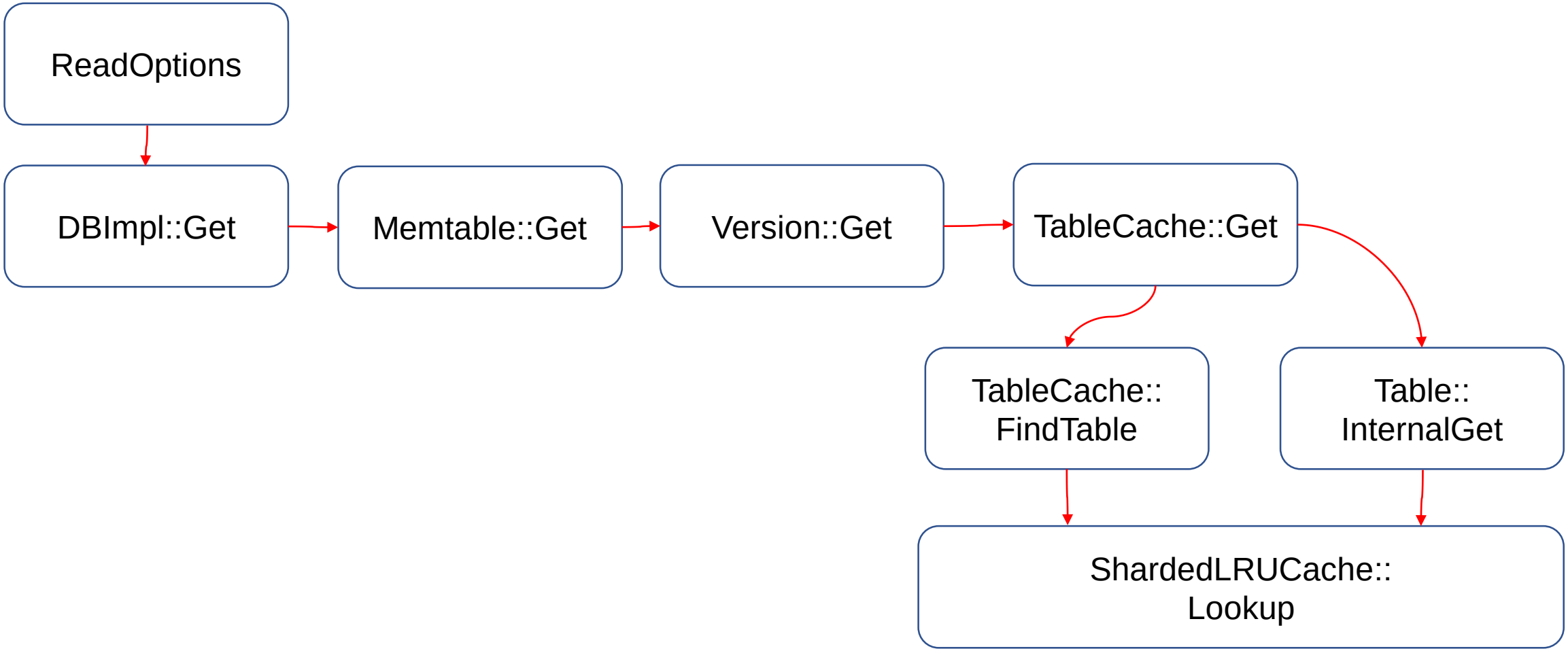
```
cache_handle = block_cache->Lookup(key);
```

Cache flow analysis

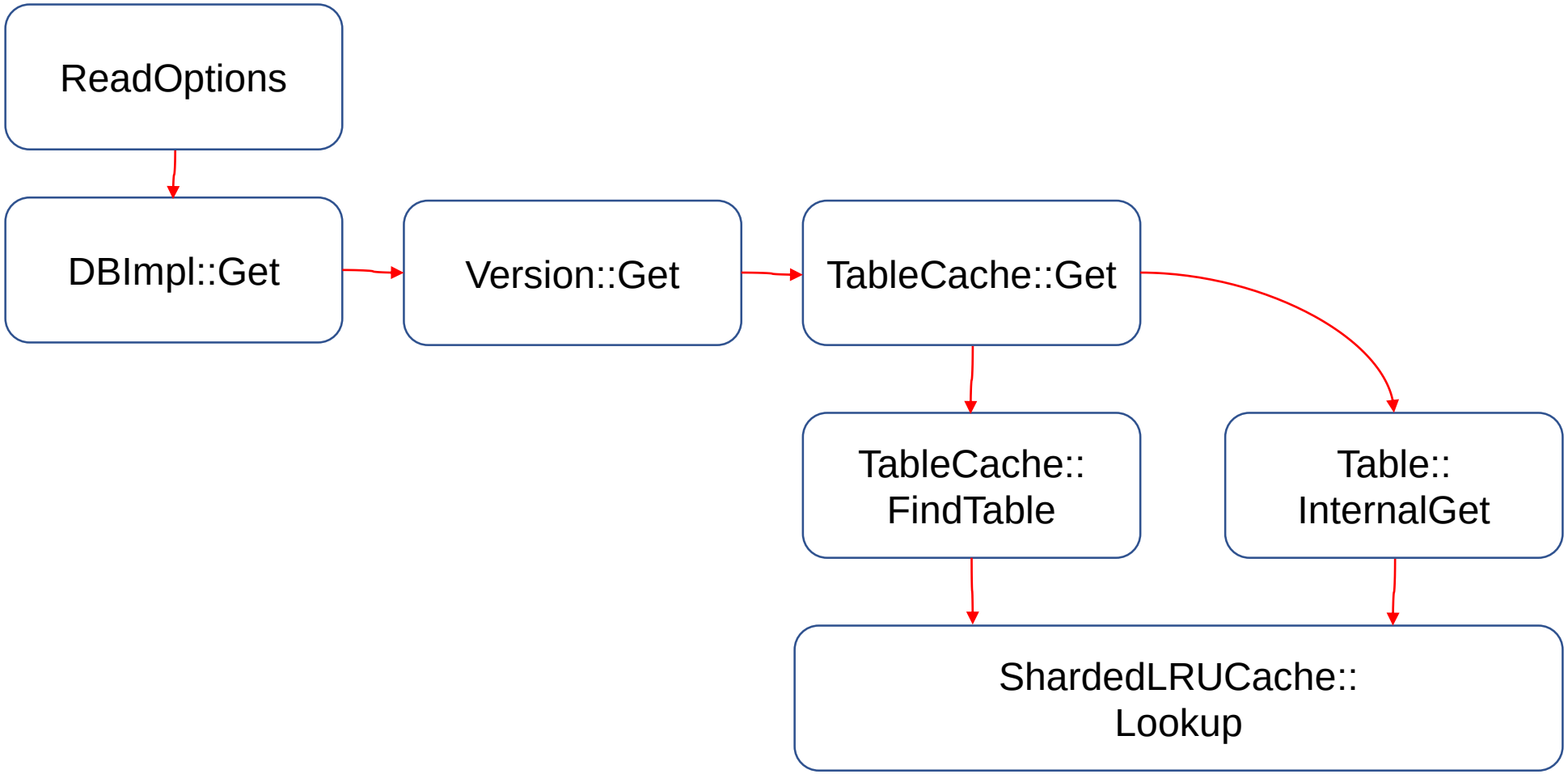



```
~T~T ~T~@(1000) leveldb::_GLOBAL__N_1::ShardedLRUCache::Lookup
 (1000) leveldb::Table::BlockReader
 (1000) leveldb::Table::InternalGet
 (1000) leveldb::TableCache::Get
 (1000) leveldb::Version::Get::State::Match
 (1000) leveldb::Version::ForEachOverlapping
 (1000) leveldb::Version::Get
 (1000) leveldb::DBImpl::Get
 (1000) leveldb::Benchmark::ReadRandom
 (1000) leveldb::Benchmark::ThreadBody
 (1000) std::__invoke_impl
 (1000) std::__invoke
 (1000) std::thread::_Invoker::_M_invoke
 (1000) std::thread::_Invoker::operator()
 (1000) std::thread::_State_impl::_M_run
```

Cache flow analysis

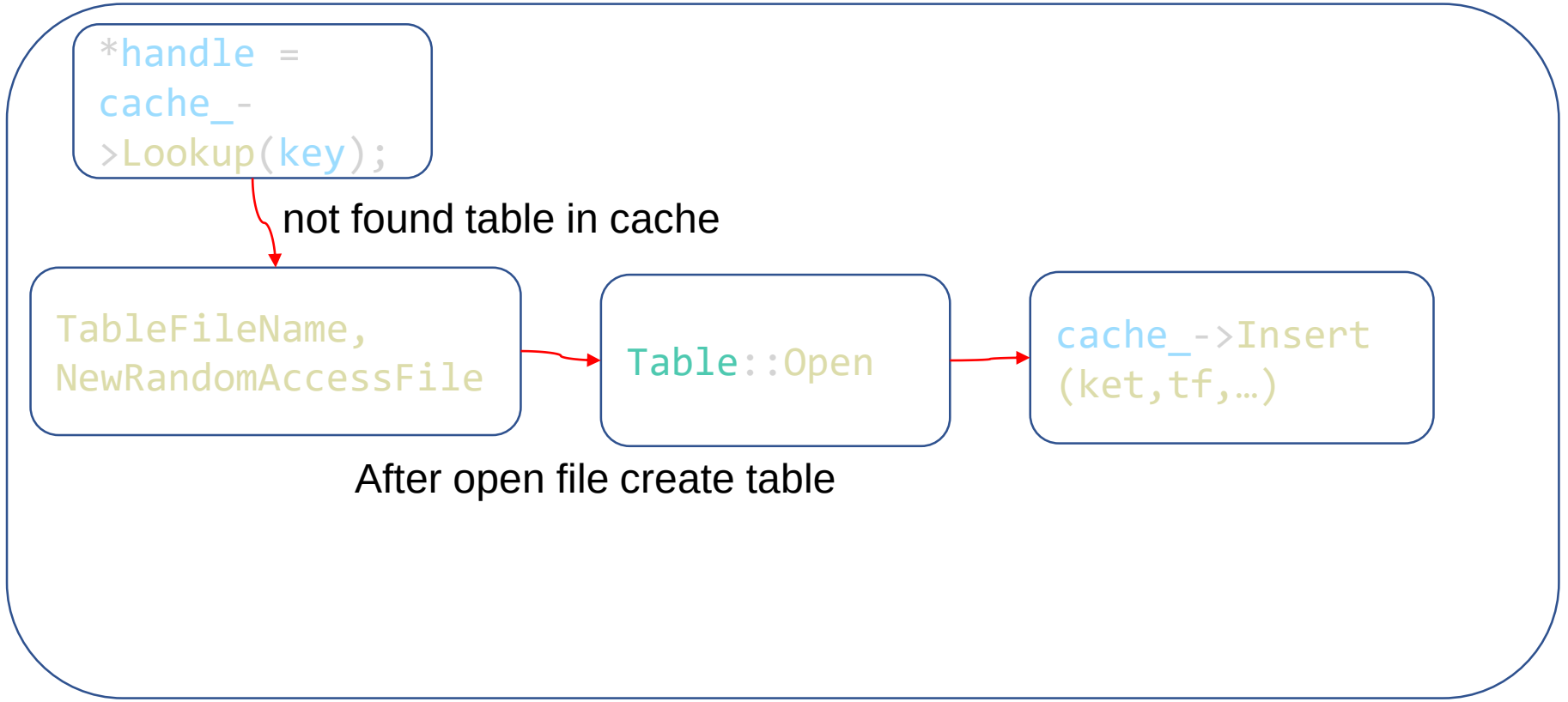


Cache flow analysis



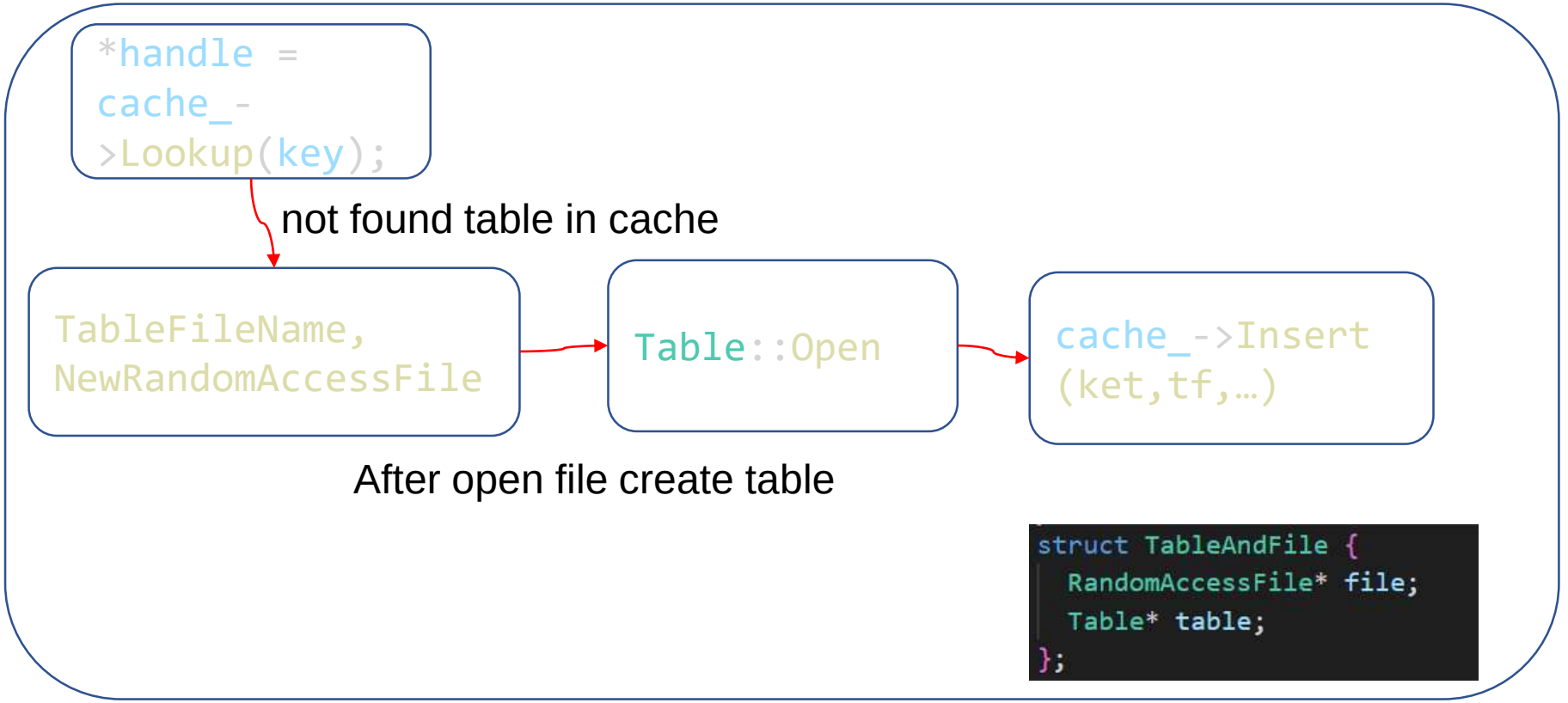
Cache flow analysis

TableCache::
FindTable



Cache flow analysis

TableCache::
FindTable



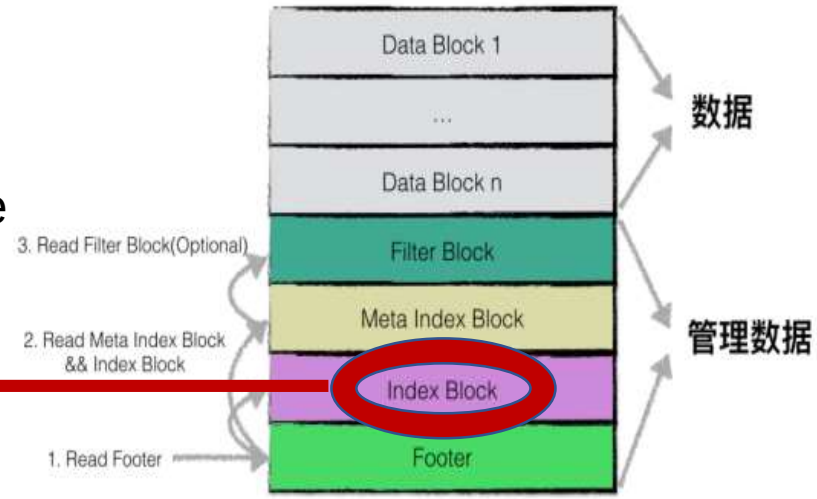
Cache flow analysis

key	value
file_number	RandomAccessFile*
	Table*

Table Cache Structure

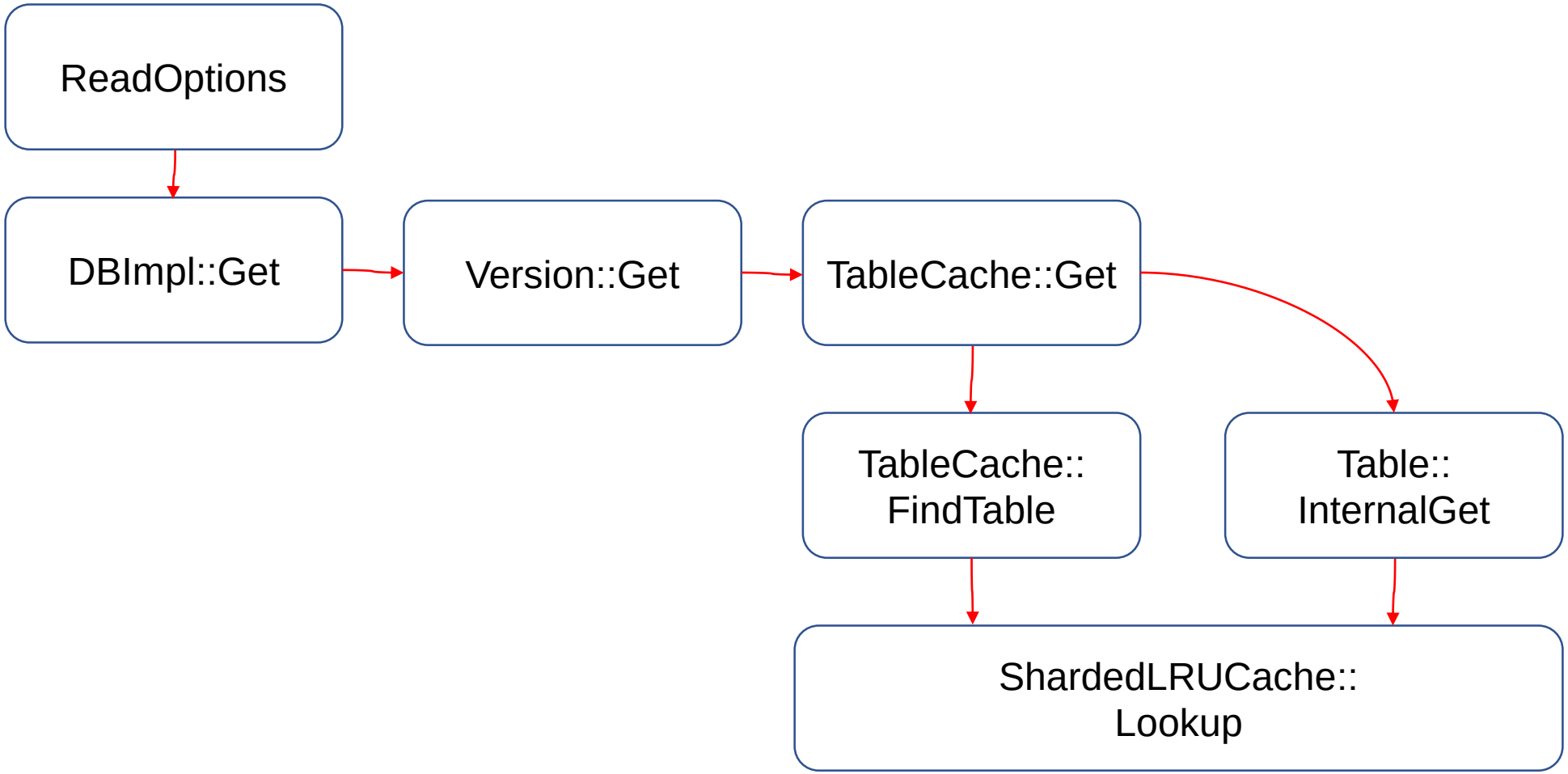
RandomAccessFile* : opened SST file

Table* the entire index block of the SST file

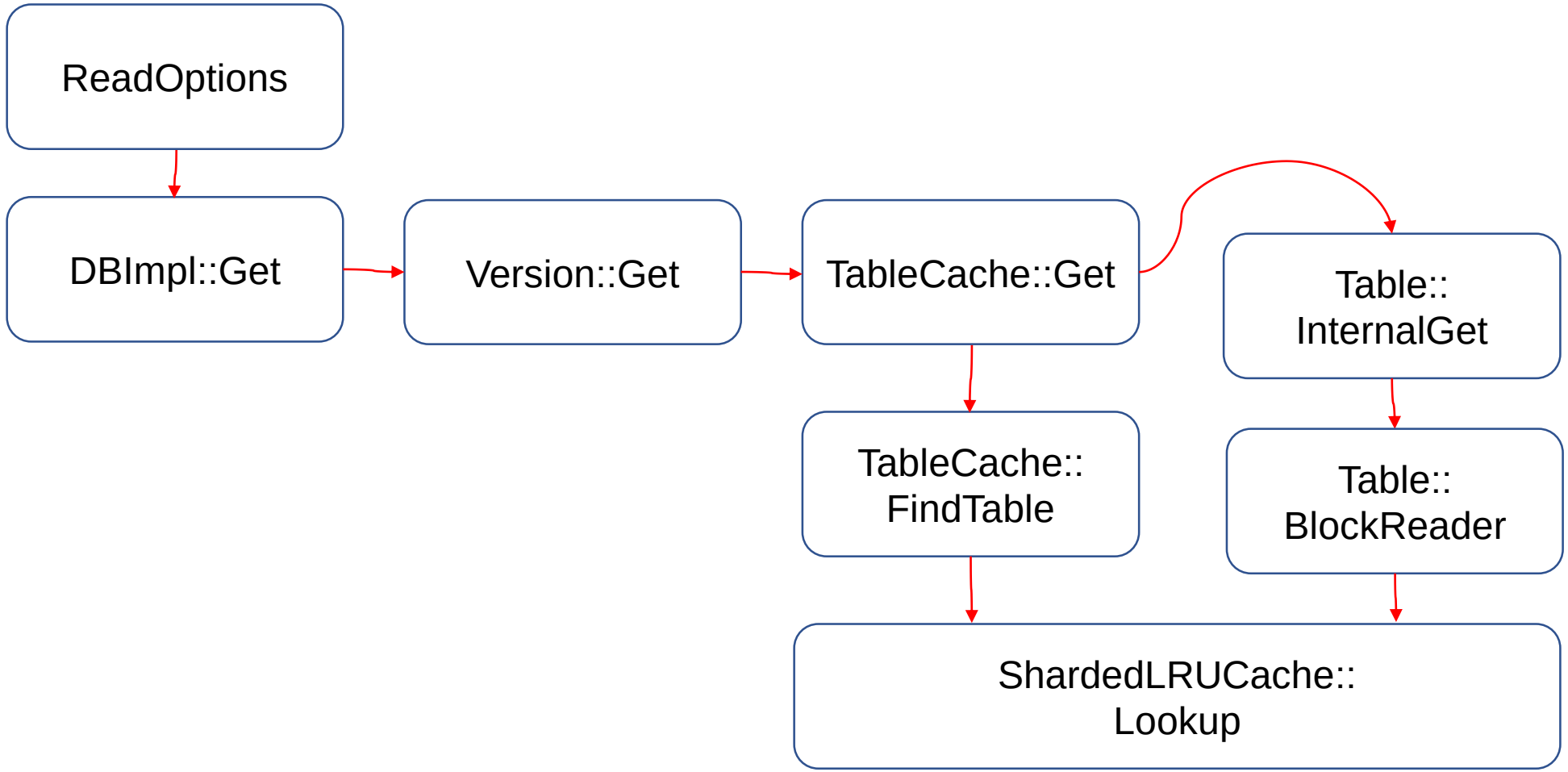


https://github.com/DKU-StarLab/leveldb-wiki/

Cache flow analysis



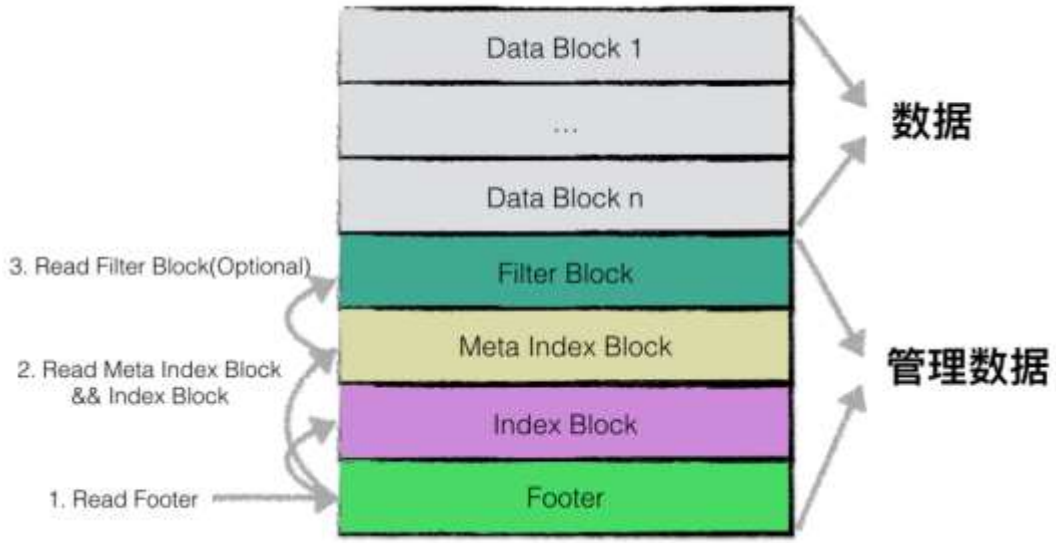
Cache flow analysis



Cache flow analysis

cache_id + block_offset	Block Data (Data Block)
cache_id + block_offset	Block Data
cache_id + block_offset	Block Data

Block Cache Structure

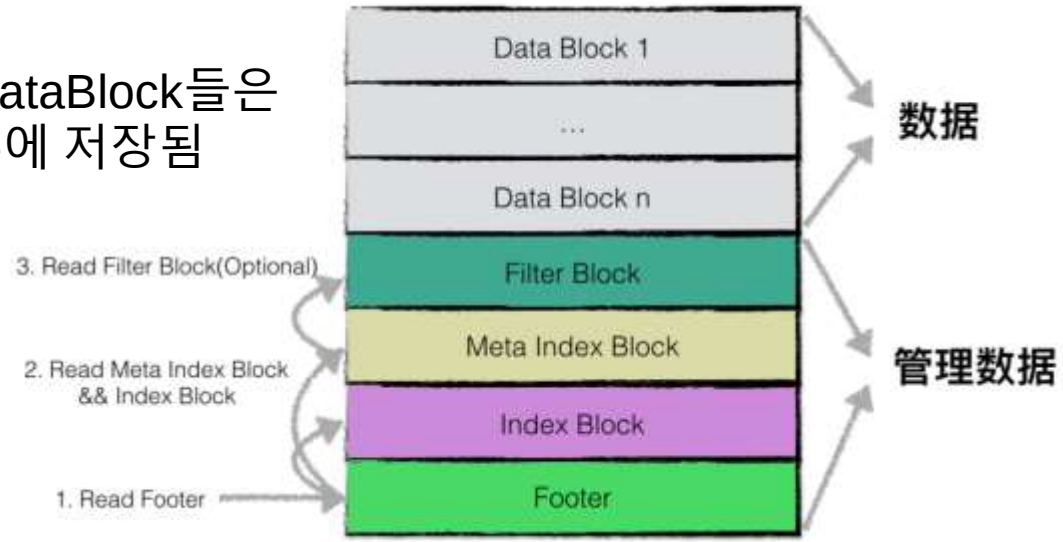


Cache flow analysis

key	value
cache_id + block_offset	**Block Data (Data Block)**
cache_id + block_offset	Block Data
cache_id + block_offset	Block Data

Block Cache Structure

열린 sst파일의 DataBlock들은
전역 BlockCache에 저장됨

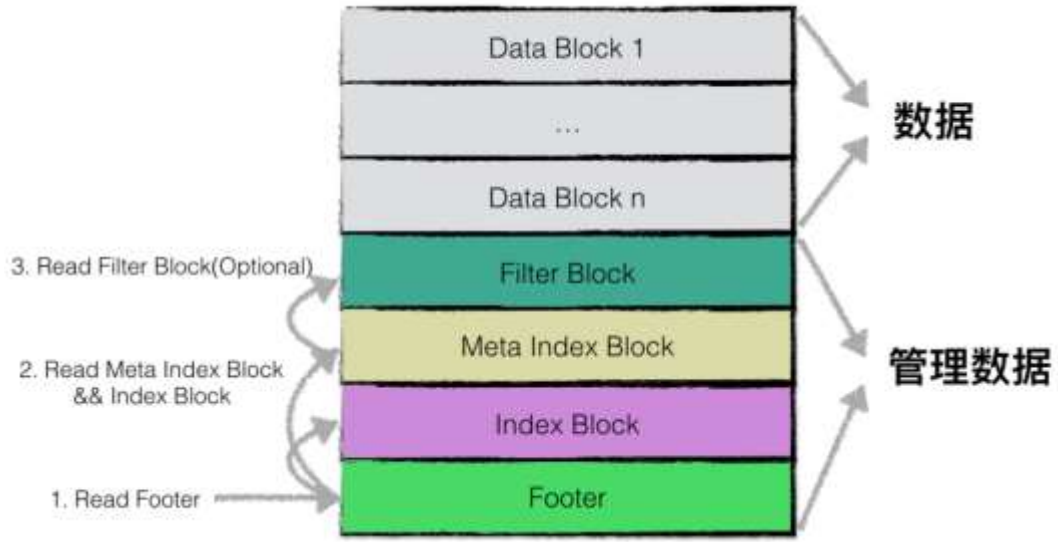


Cache flow analysis

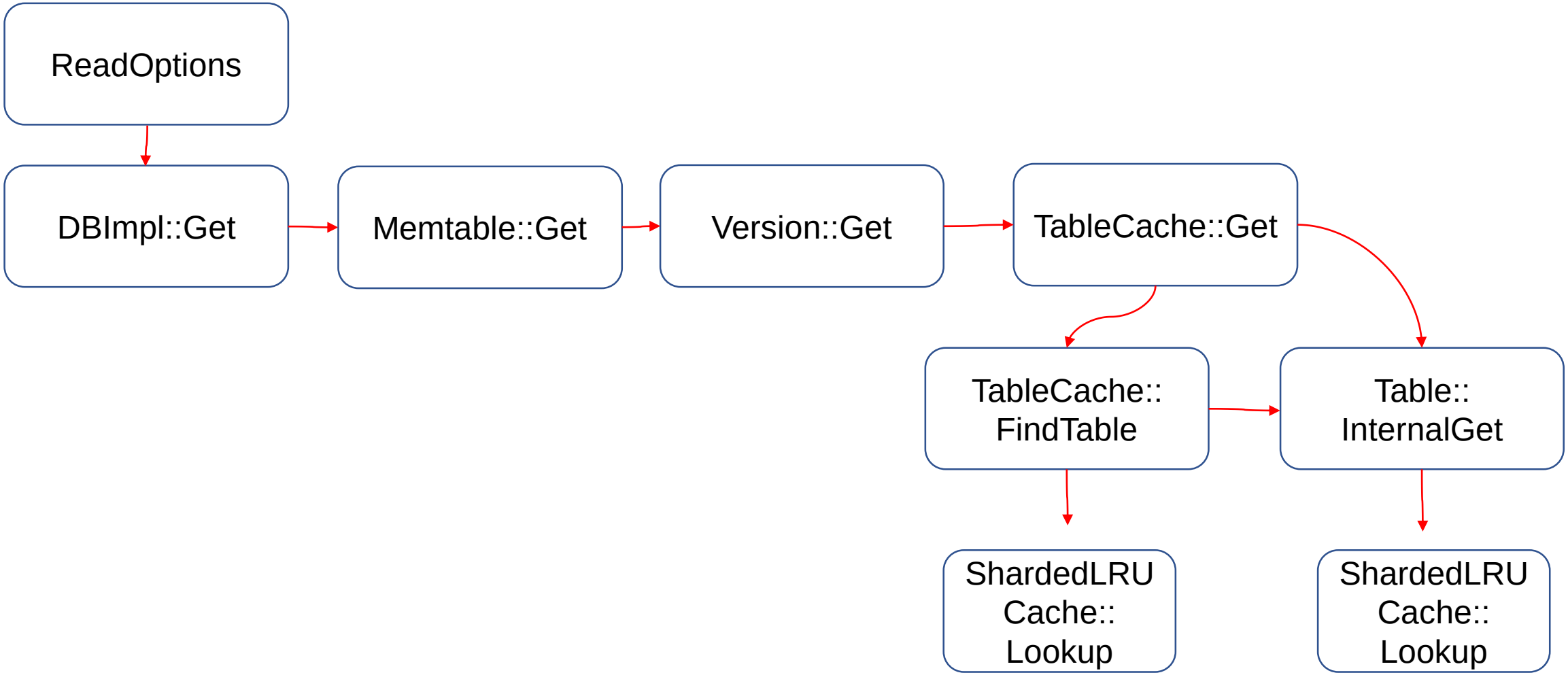
cache_id + block_offset	Block Data (Data Block)
cache_id + block_offset	Block Data
cache_id + block_offset	Block Data

Block Cache Structure

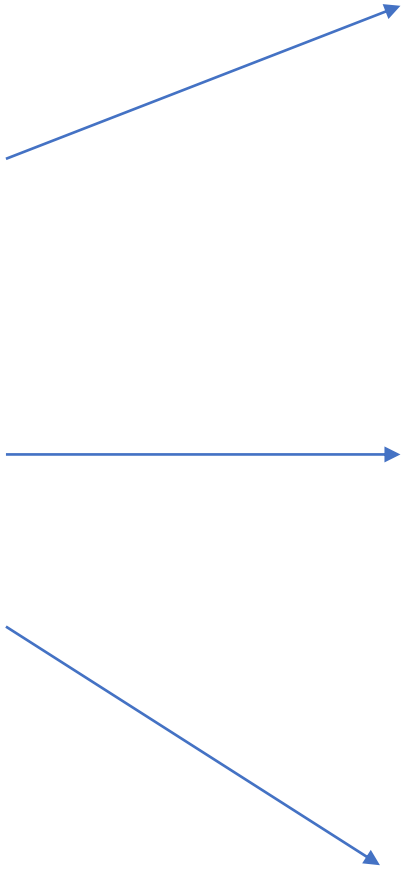
다른 sst파일의 Data Block offset이 동일할 수 있으므로 구별을 위해, 각 sst파일에 고유한 cache_id를 조합하여 key를 구성함



Cache flow analysis



과목	성적
국어	85
수학	97
영어	88
생명	95
지구과학	75
물리	87
화학	70



100~90

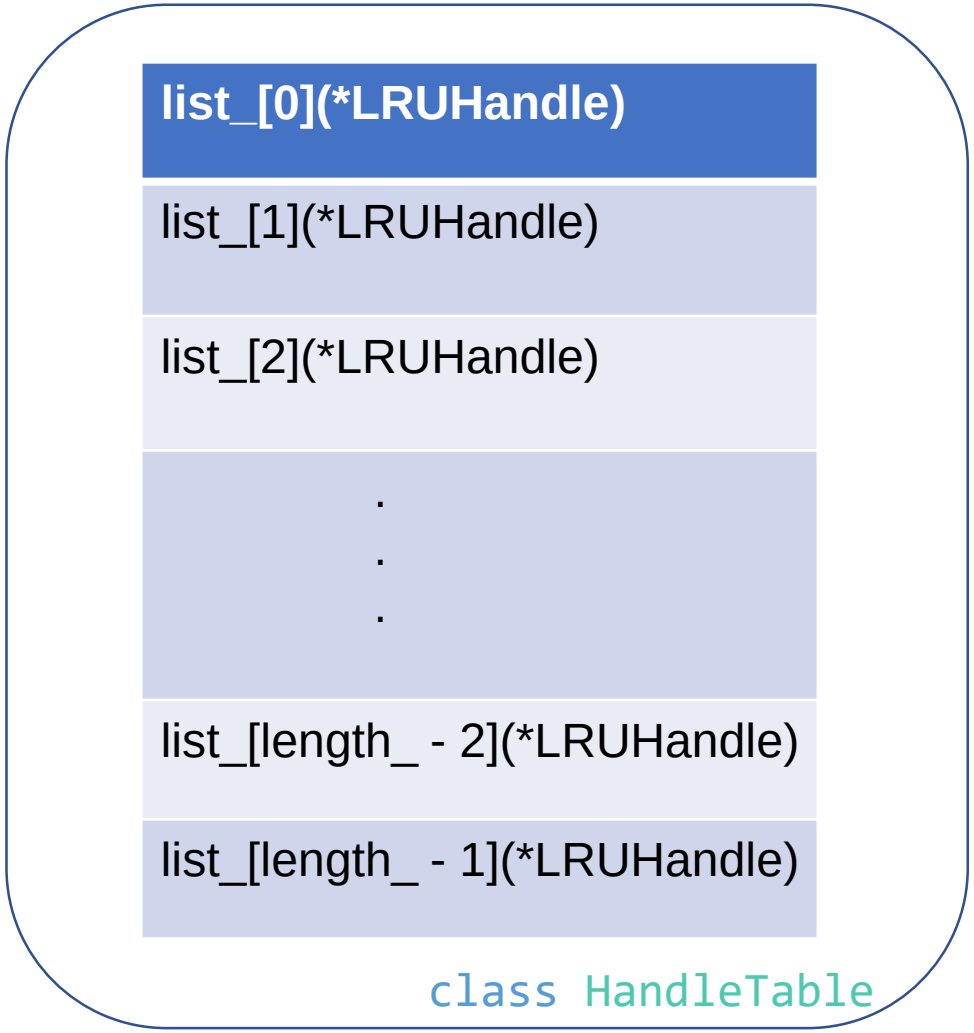
과목	성적
수학	97
생명	95

90~80

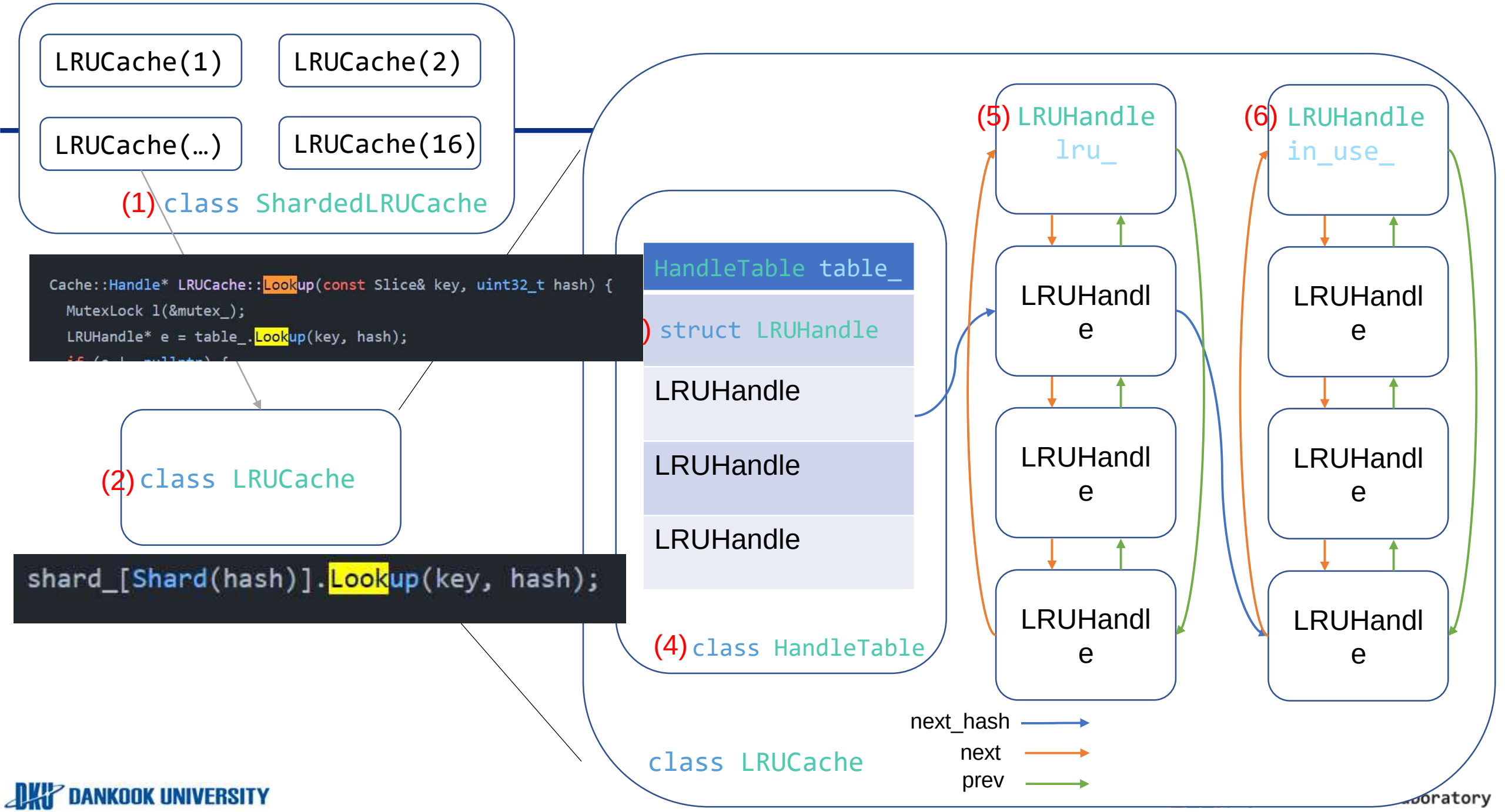
과목	성적
국어	85
영어	88
물리	87

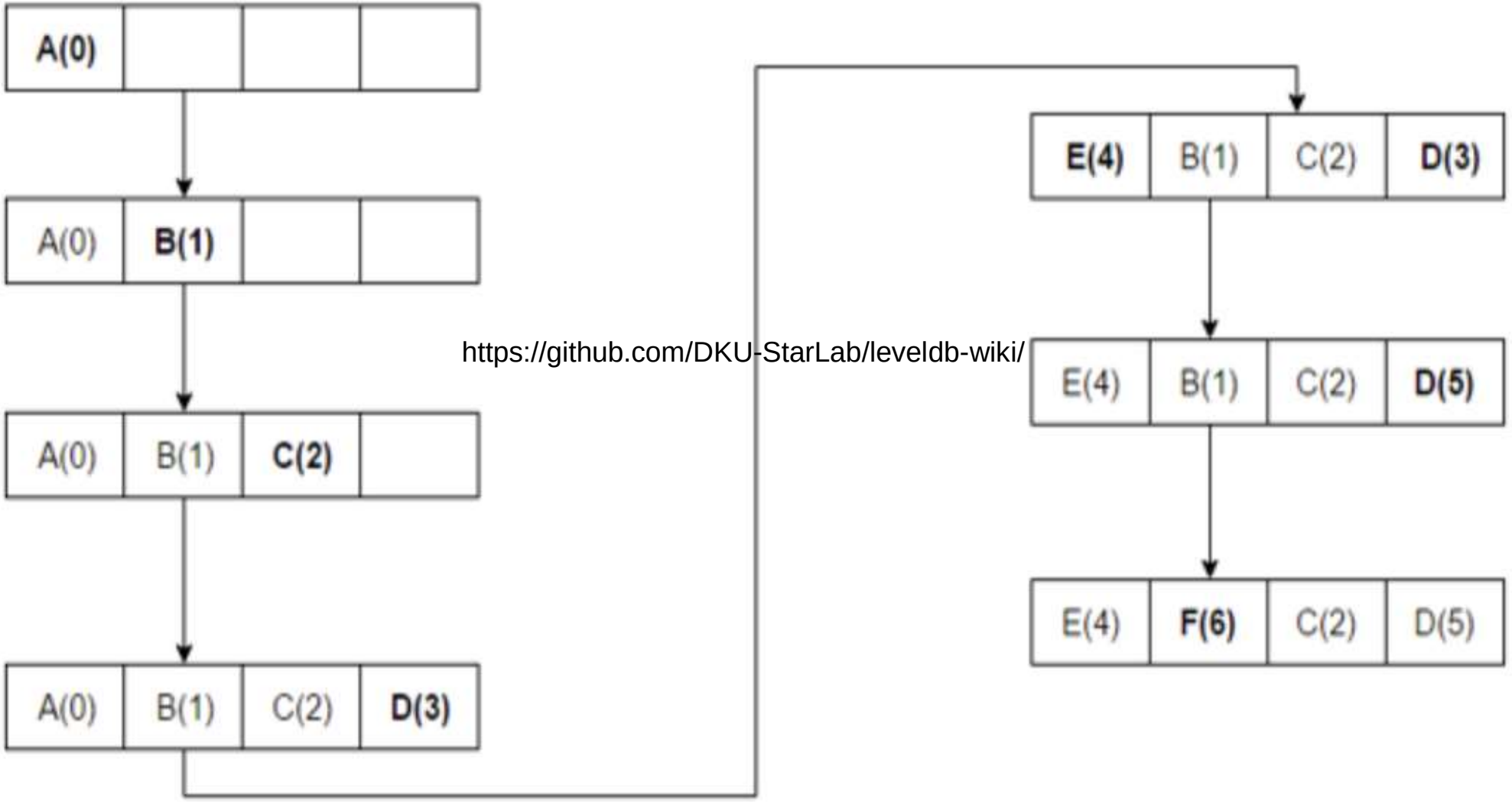
80~70

과목	성적
지구과학	75
화학	70

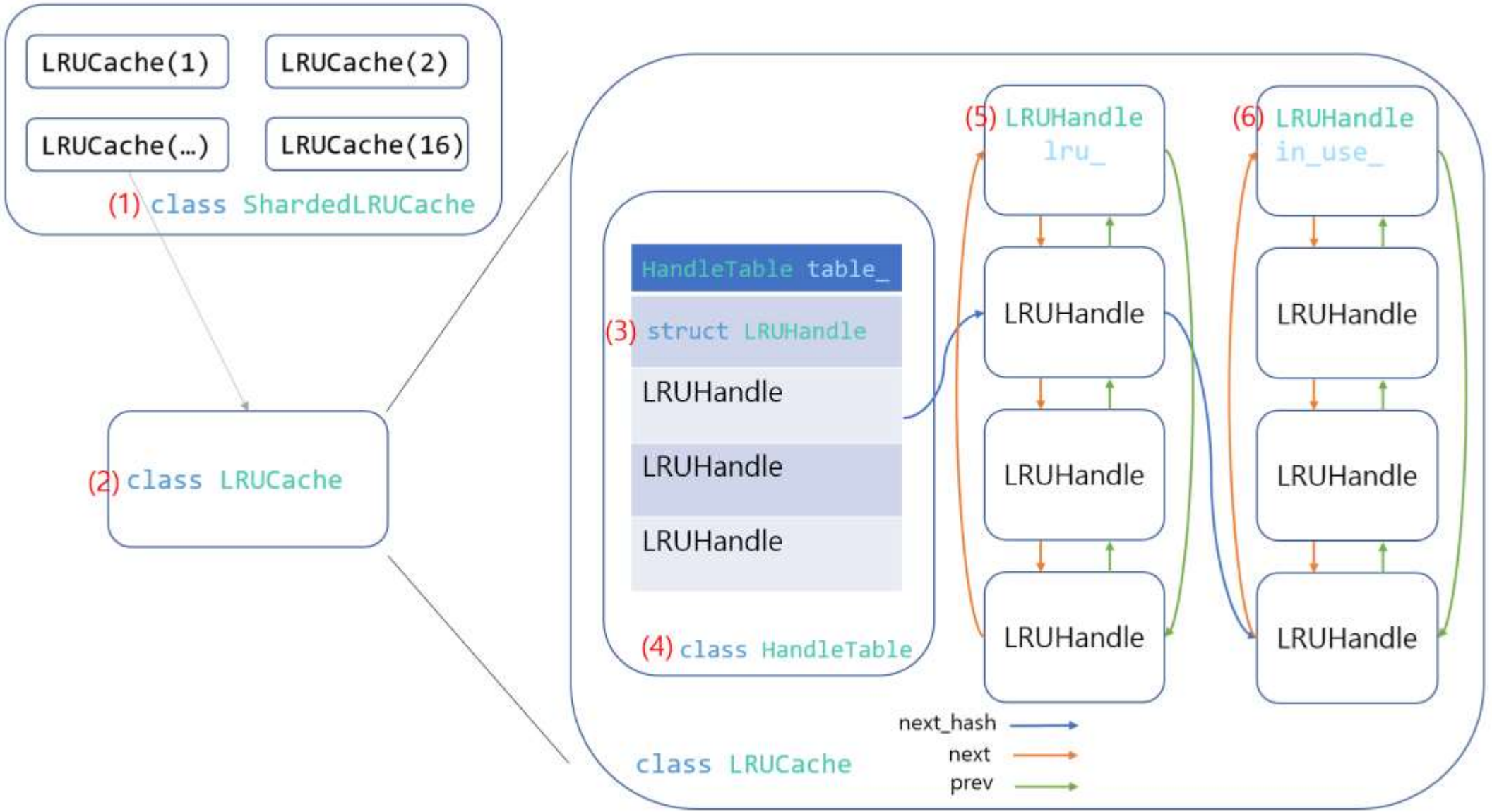


https://github.com/DKU-StarLab/leveldb-wiki/

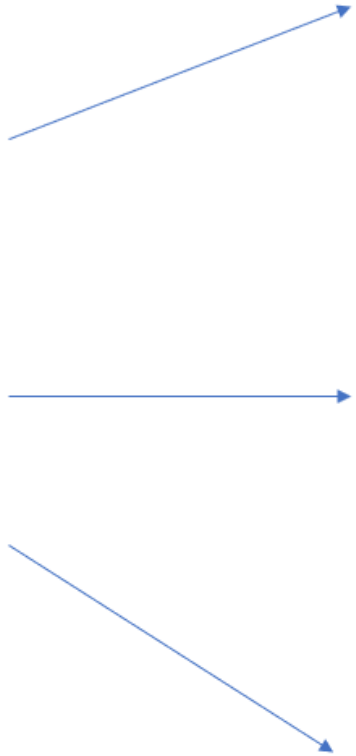

LRUCache(1)
LRUCache(2)
LRUCache(…)
LRUCache(16)
(1) class ShardedLRUCache
Cache::Handle* LRUCache::Lookup(const Slice& key, uint32_t hash) {
MutexLock l(&mutex_);
LRUHandle* e = table_.Lookup(key, hash);
(2) class LRUCache
shard_[Shard(hash)].Lookup(key, hash);
HandleTable table_
(3) struct LRUHandle
LRUHandle
LRUHandle
LRUHandle
(4) class HandleTable
(5) LRUHandle lru_
(6) LRUHandle in_use_
LRUHandle
LRUHandle
LRUHandle
LRUHandle
LRUHandle
LRUHandle
LRUHandle
LRUHandle
next_hash
next
prev
class LRUCache



https://github.com/DKU-StarLab/leveldb-wiki/


LRUCache(1)
LRUCache(2)
LRUCache(…)
LRUCache(16)
(1) class ShardedLRUCache
(2) class LRUCache
HandleTable table_
(3) struct LRUHandle
LRUHandle
LRUHandle
LRUHandle
(4) class HandleTable
(5) LRUHandle lru_
(6) LRUHandle in_use_
LRUHandle
LRUHandle
LRUHandle
LRUHandle
LRUHandle
LRUHandle
LRUHandle
LRUHandle
next_hash
next
prev
class LRUCache

과목	성적
국어	85
수학	97
영어	88
생명	95
지구과학	75
물리	87
화학	70



100~90

과목	성적
수학	97
생명	95

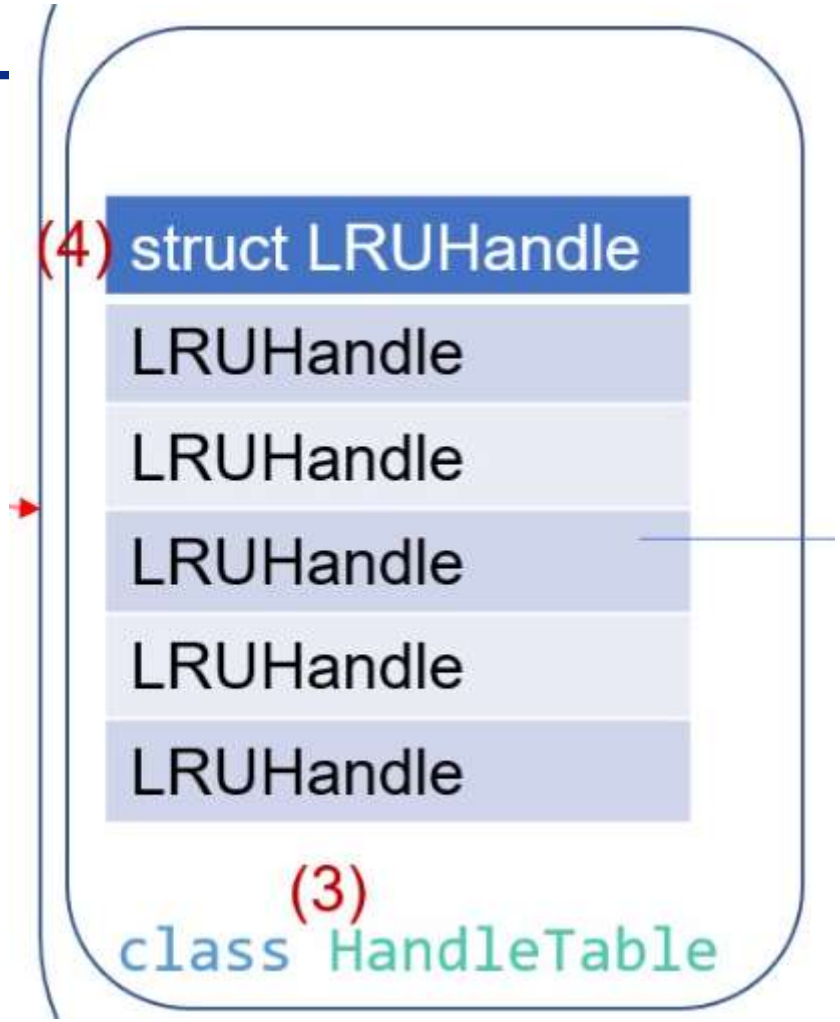
90~80

과목	성적
국어	85
영어	88
물리	87

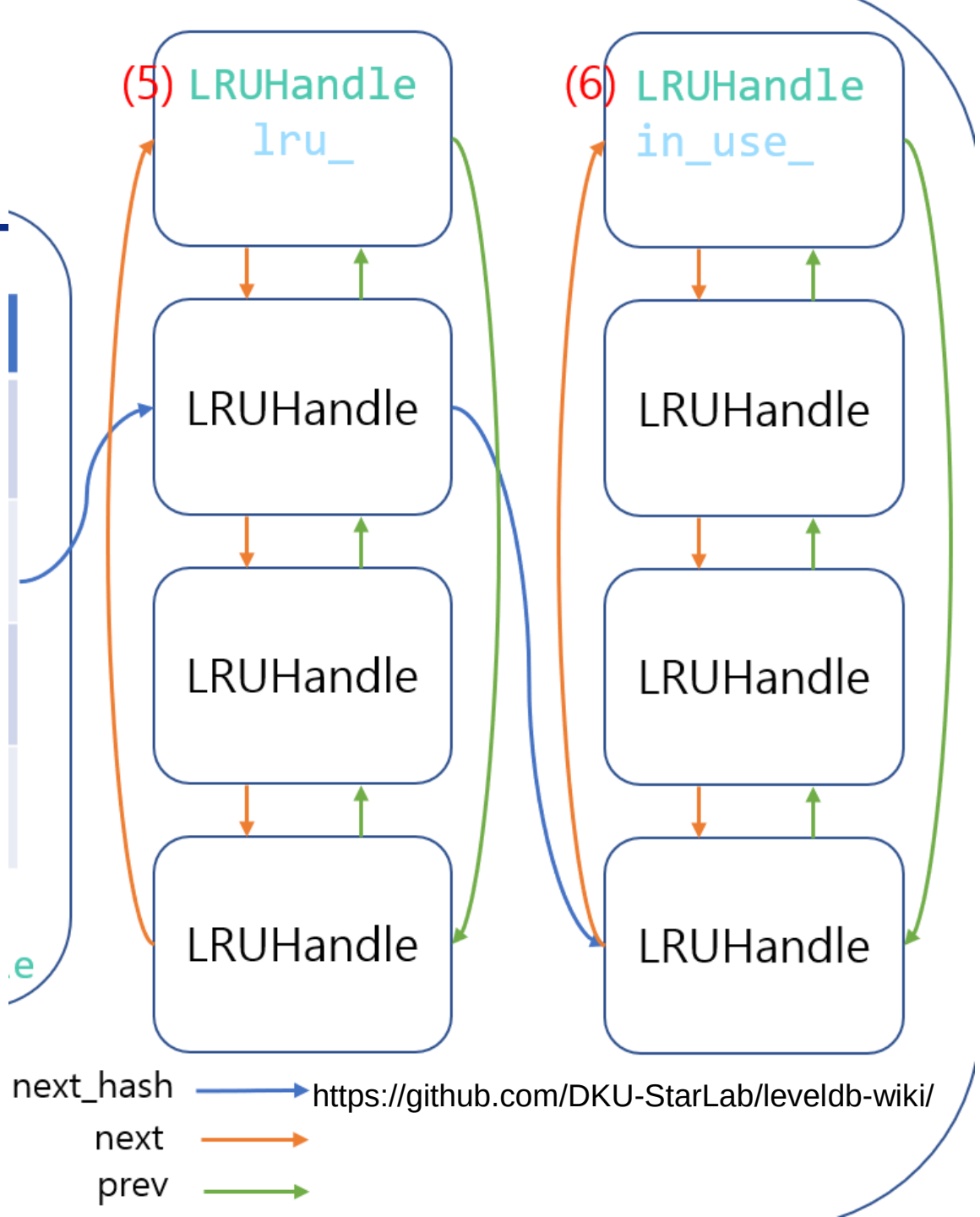
80~70

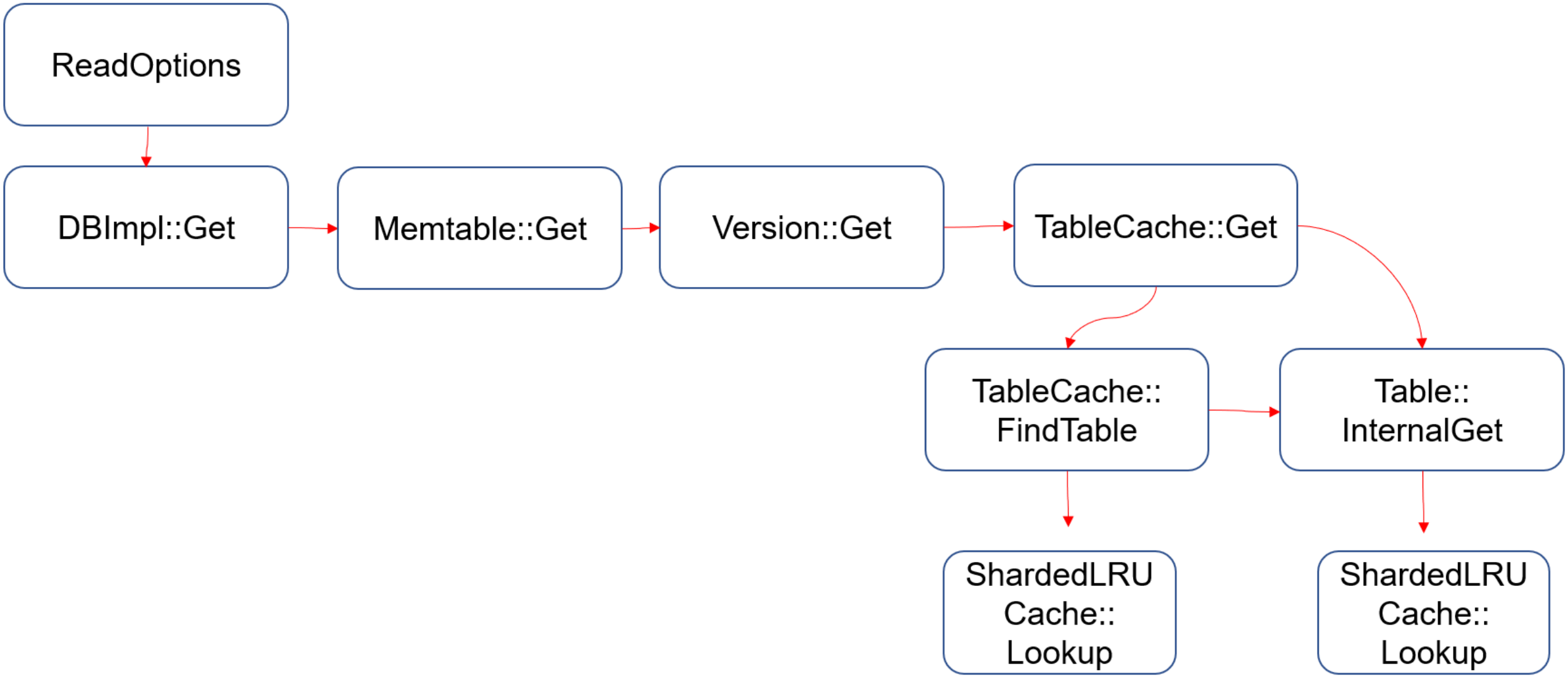
과목	성적
지구과학	75
화학	70

https://github.com/DKU-StarLab/leveldb-wiki/



https://github.com/DKU-StarLab/leveldb-wiki/


(5) LRUHandle
lru_
(6) LRUHandle
in_use_
LRUHandle
LRUHandle
LRUHandle
LRUHandle
LRUHandle
LRUHandle
LRUHandle
LRUHandle
next_hash
next
prev
https://github.com/DKU-StarLab/leveldb-wiki/

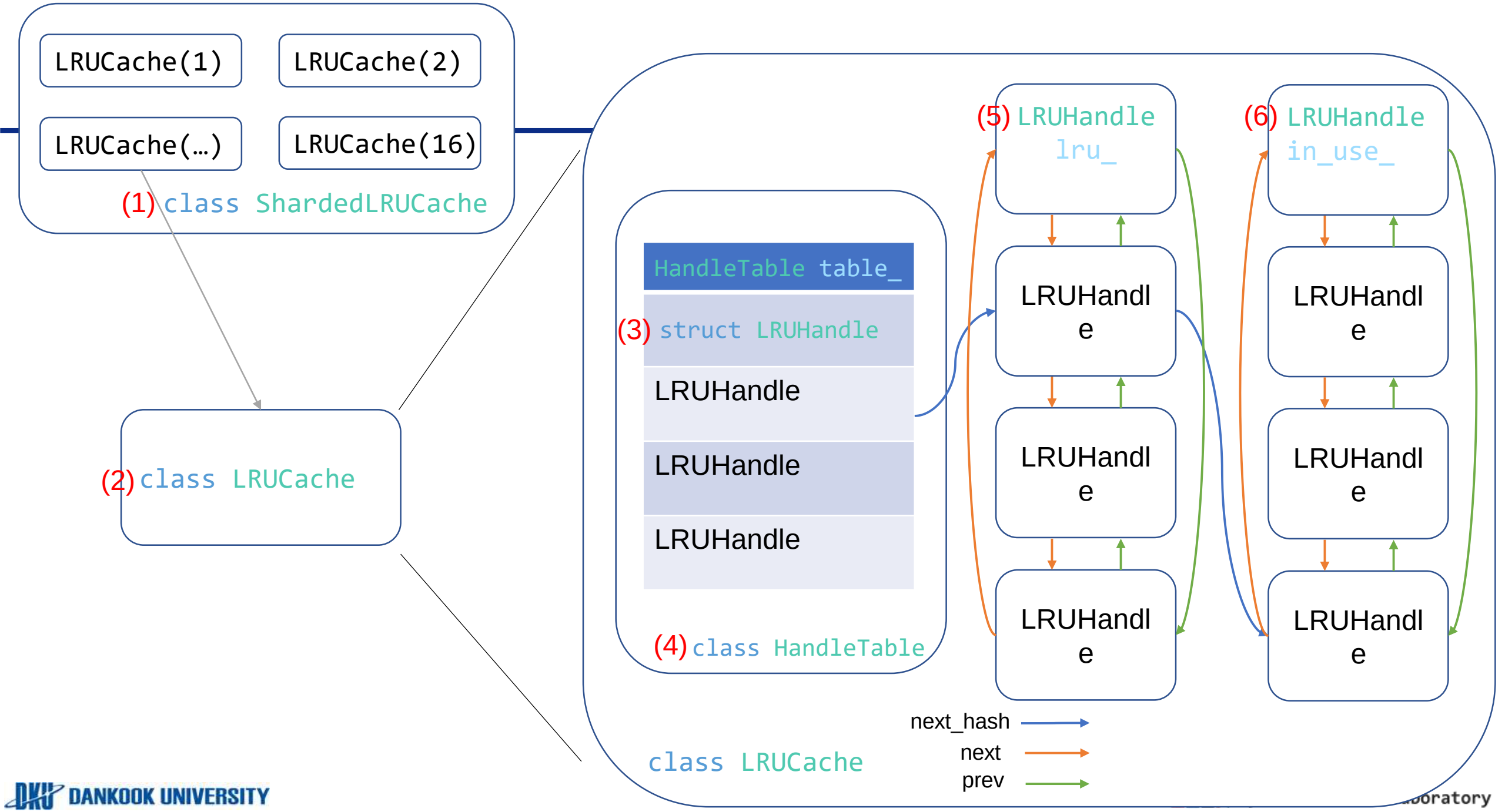


https://github.com/DKU-StarLab/leveldb-wiki/

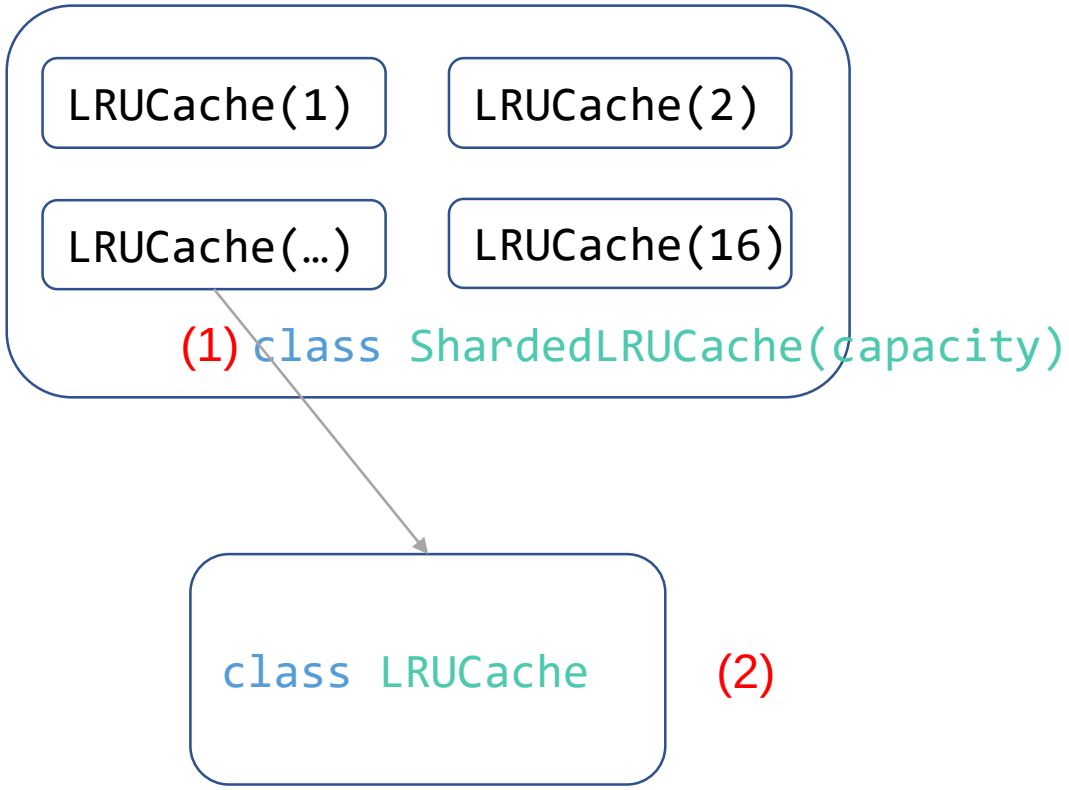
key	value
cache_id + block_offset	**Block Data (Data Block)**
cache_id + block_offset	Block Data
cache_id + block_offset	Block Data

Block Cache Structure

https://github.com/DKU-StarLab/leveldb-wiki/

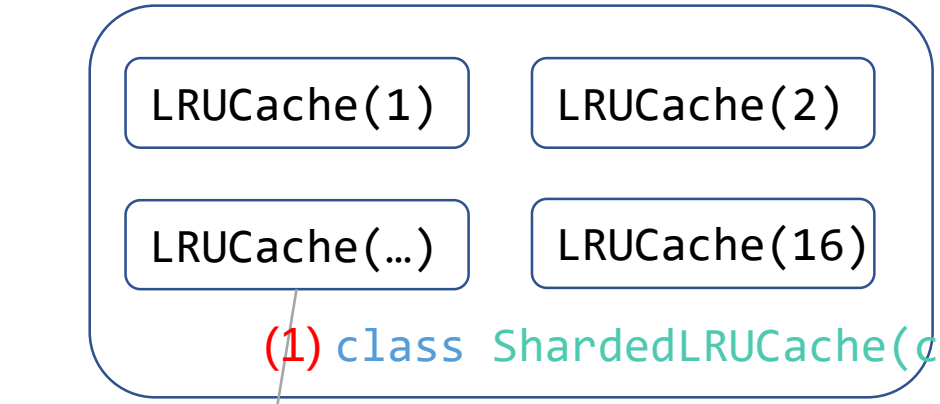

LRUCache(1)
LRUCache(2)
LRUCache(…)
LRUCache(16)
(1) class ShardedLRUCache
(2) class LRUCache
HandleTable table_
(3) struct LRUHandle
LRUHandle
LRUHandle
LRUHandle
(4) class HandleTable
(5) LRUHandle lru_
(6) LRUHandle in_use_
LRUHandle
LRUHandle
LRUHandle
LRUHandle
LRUHandle
LRUHandle
LRUHandle
LRUHandle
next_hash
next
prev
class LRUCache

./db_bench --benchmarks="fillrandom,readrandom" --cache_size=1000000




```
static const int kNumShardBits = 4;
static const int
kNumShards = 1 << kNumShardBits; // 16

class ShardedLRUCache : public Cache

LRUCache shard_[kNumShards];
```

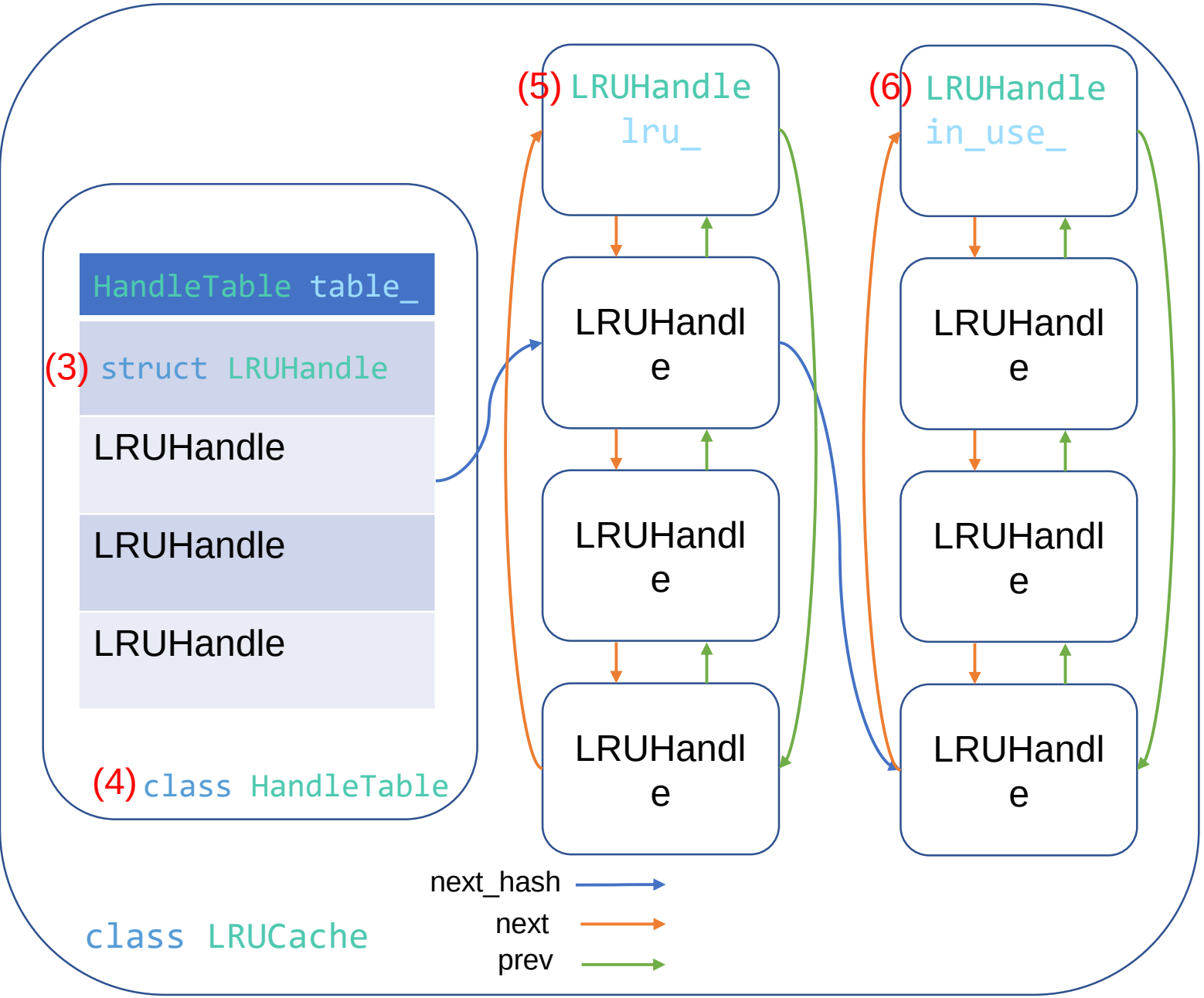
./db_bench --benchmarks="fillrandom,readrandom" --cache_size=1000000 --open_files=1000(default)



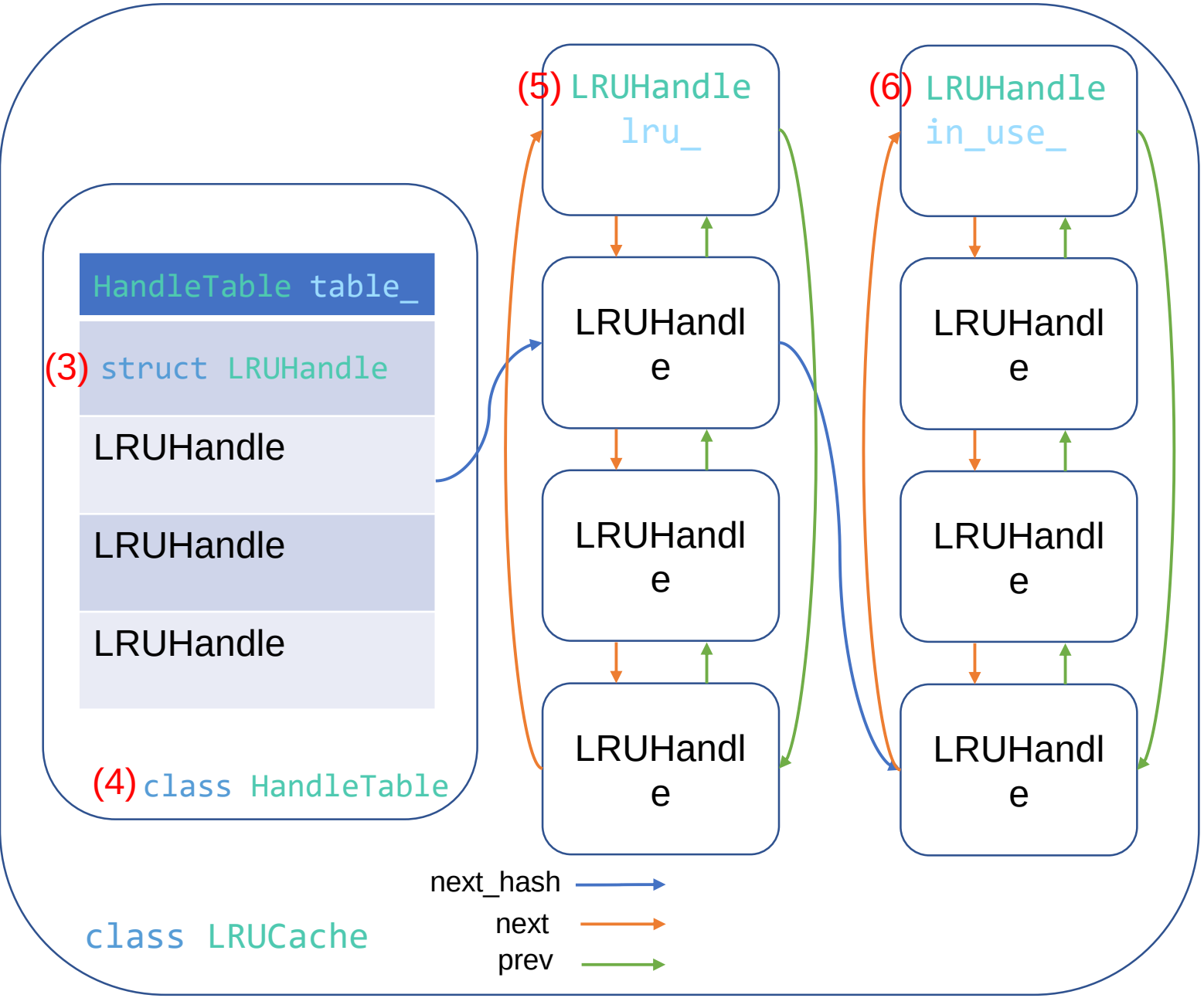
(1) class ShardedLRUCache(capacity)

```
explicit ShardedLRUCache(size_t capacity) : last_id_(0) {
    const size_t per_shard = (capacity + (kNumShards - 1)) /;
kNumShards
    for (int s = 0; s < kNumShards; s++) {
      shard_[s].SetCapacity(per_shard);
⇒ SetCapacity 는 LRUCache배열의 버퍼 즉, LRUCache 각각의 capacity를 설정
⇒ Cache_size가 클수록 _shard배열의 각 버퍼인 LRUCache의 capacity가 커짐
    }
  }
```

class LRUCache



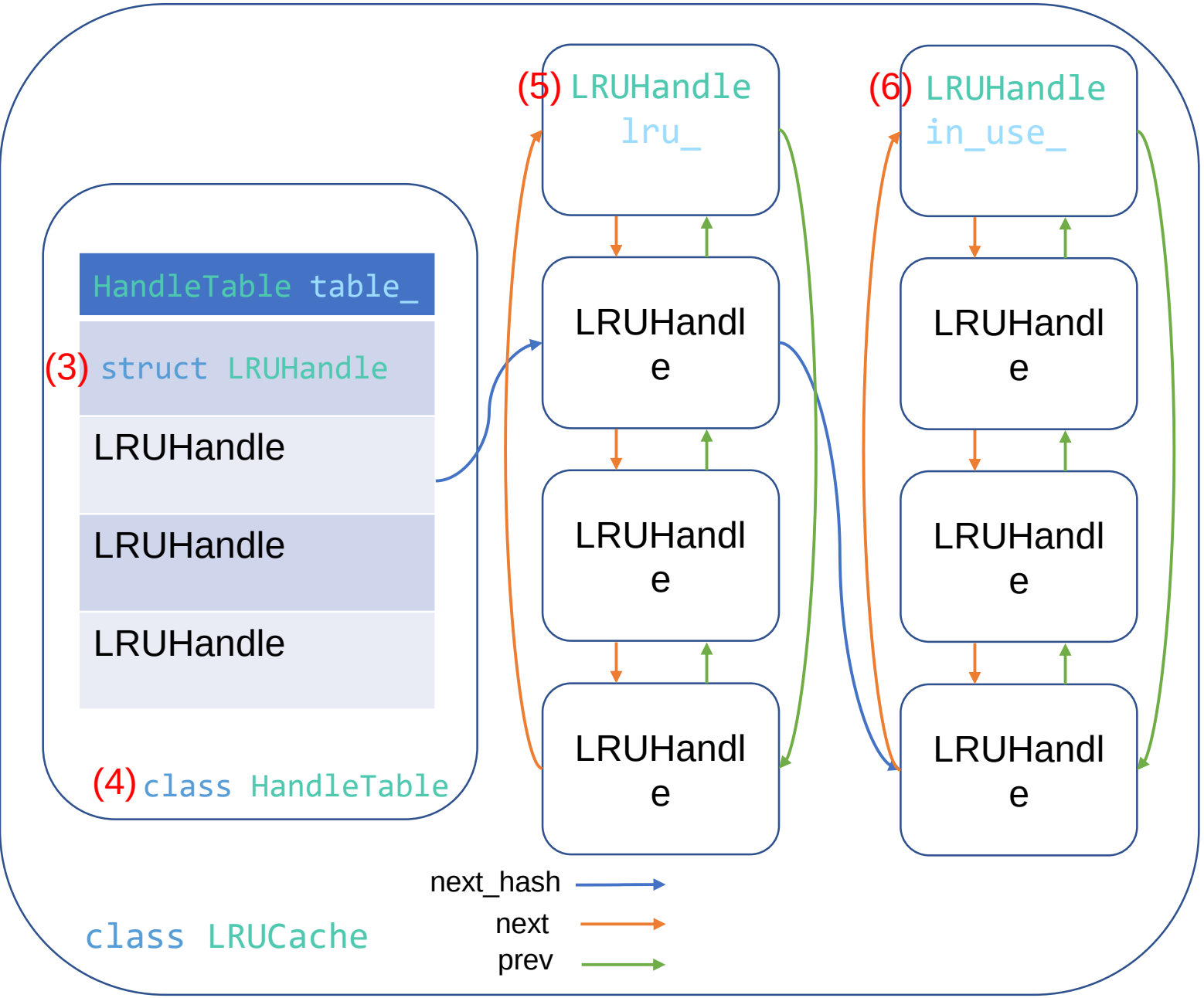

```
size_t capacity_;
//SetCapacity를 통해 각
LRUCache의 capacity를 담는
변수임 (용량)

size_t usage_;
// (사용량)
```




```
Cache::Handle*
LRUCache::Insert(const
Slice& key, uint32_t hash,
void* value, size_t
charge)
```

Insert되는 블록들은 charge를 갖고 Insert함수를 호출하며 호출될 때마다 usage에 charge를 더해감

Insert시에 LRUHandle을 lru_에 먼저 넣으며(refs=1)
Capacity_가 양수일 경우
Refs를 1 올리고
in_use에 LRUHandle을 넣고
Lru_에서 LRUHandle을 빼고
Usage에 charge를 더함



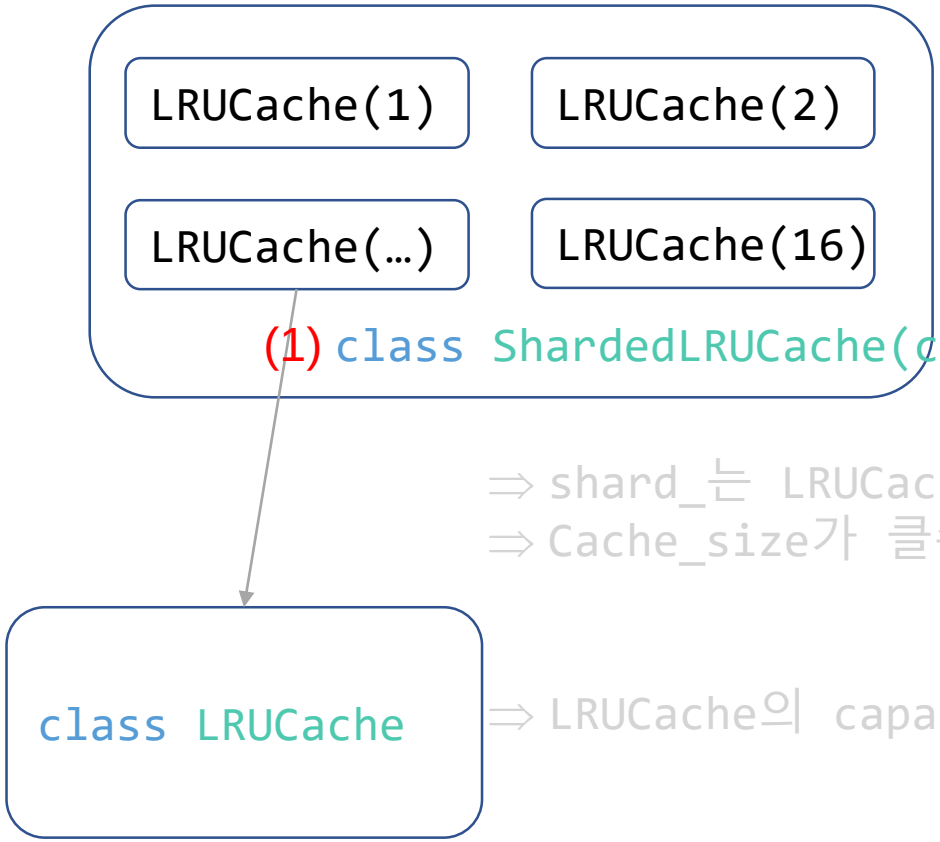
Usage_가 capacity_보다 커질 경우 lru_에서 오래된 노드(LRUHandle)을 제거함

usage_에서 제거된 노드의 charge를 뺌

=> LRU정책

./db_bench --benchmarks="fillrandom“ –use=0

H



⇒ shard_는 LRUCache 배열 즉, LRUCache배열의 각 인덱스의 capacity를 설정
⇒ Cache_size가 클수록 _shard배열의 각 버퍼인 LRUCache의 capacity가 커짐

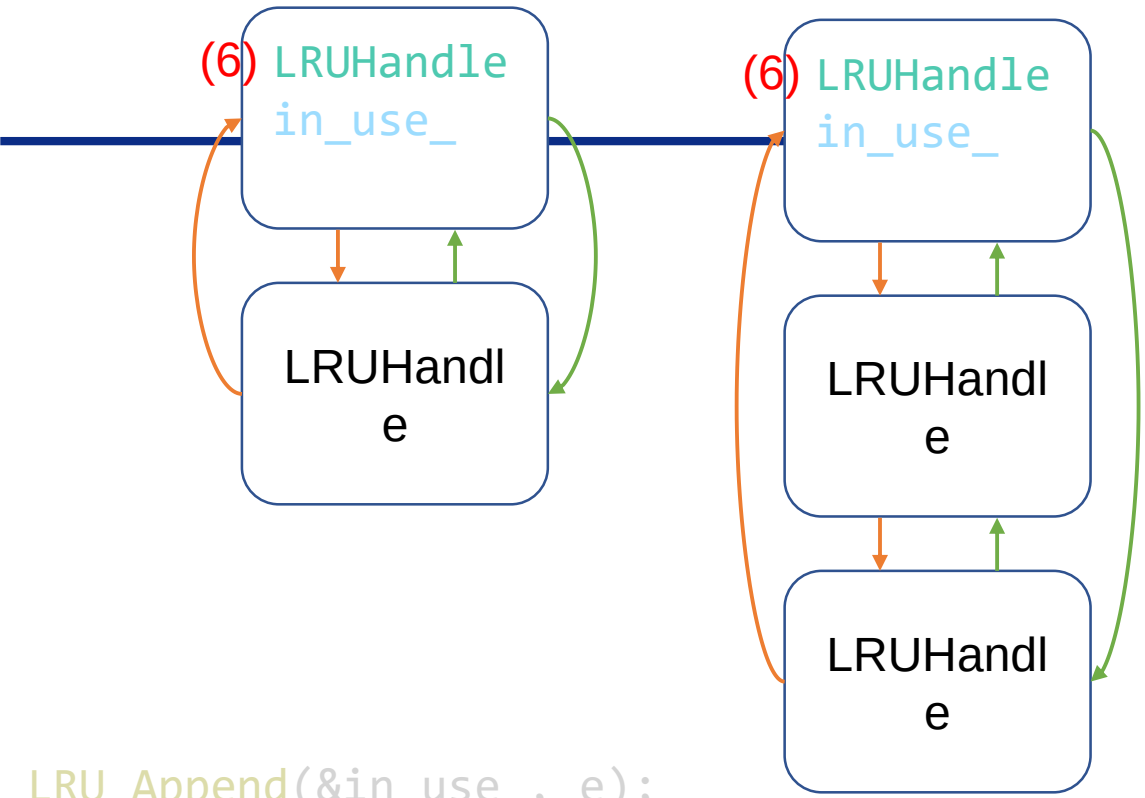
⇒ LRUCache의 capacity가 커지면 장점?

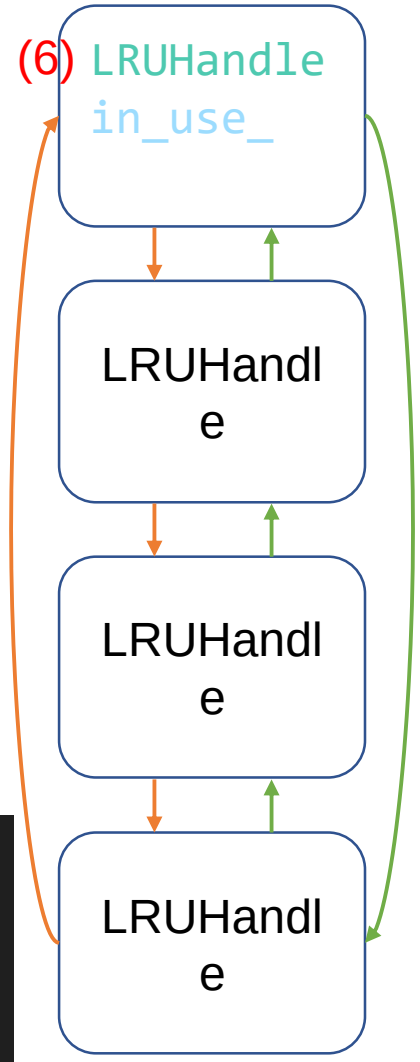
⇒ shard_는 LRUCache 배열 즉, LRUCache배열의 각 인덱스의 capacity를 설정
⇒ Cache_size가 클수록 _shard배열의 각 버퍼인 LRUCache의 capacity가 커짐

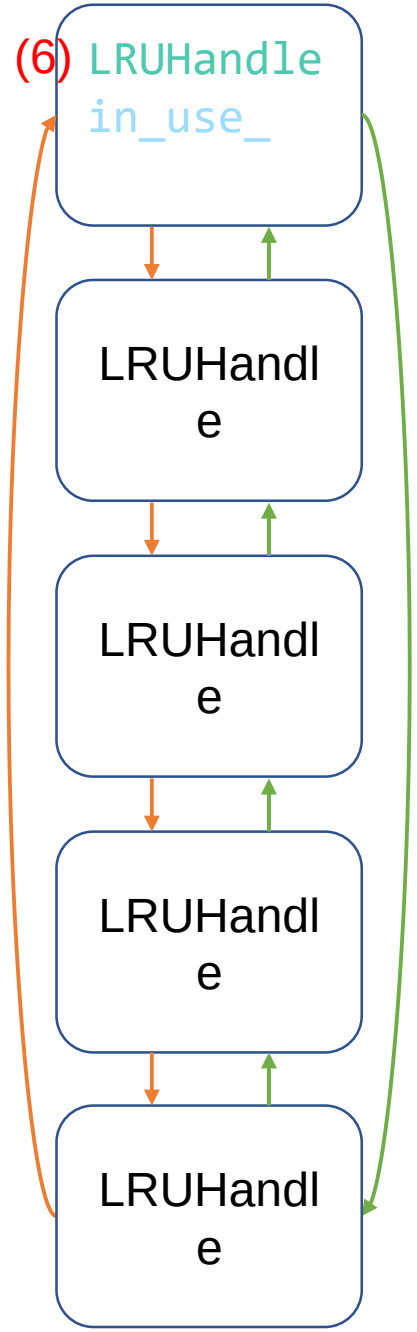
⇒ LRUCache의 capacity가 커지면 장점?

```
Cache::Handle* LRUCache::Insert(const Slice& key, uint32_t hash, void*
value, size_t charge, void (*deleter)(const Slice& key, void* value))
```

=> cache에 넣을 block의 key-value, hashtable의 hash, deleter, cache size인 charge








```
LRU_Append(&in_use_, e);
```

```
void LRUCache::LRU_Append(LRUHandle* list, LRUHandle* e) {
  // Make "e" newest entry by inserting just before *list
  e->next = list;
  e->prev = list->prev;
  e->prev->next = e;
  e->next->prev = e;
}
```

각 double linked list의 LRUHandle에는 HandleTable(HashTable)을 통해 바로 접근이 가능