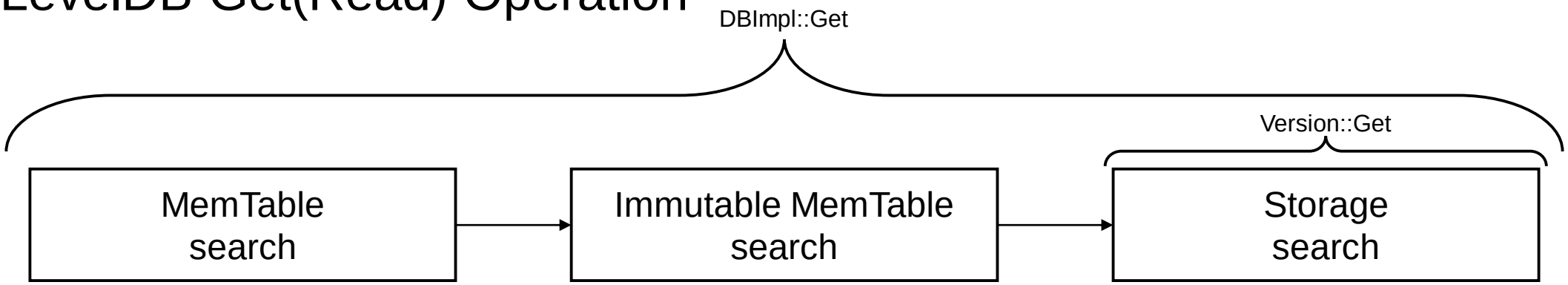
SSTable
– How to a find target key in the SSTable-

Sanghyun Cho, Jongki Park
E-Mail: 98shcho@naver.com
jkipark@dankook.ac.kr

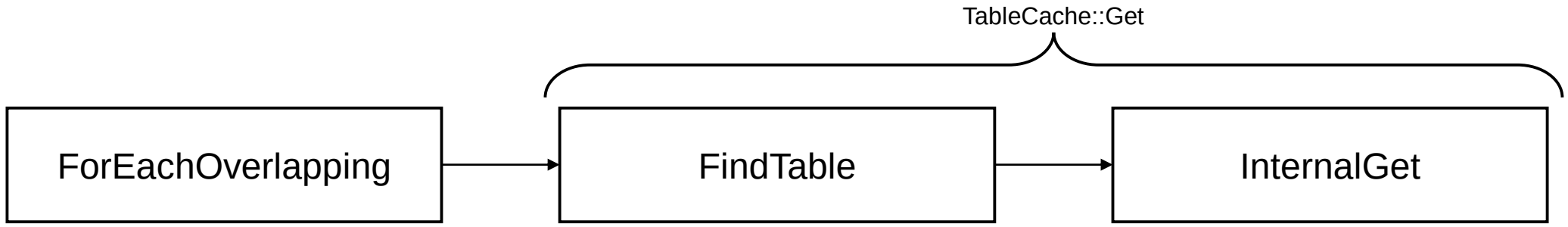
Contents

Previous Analyses

- LevelDB Get(Read) Operation



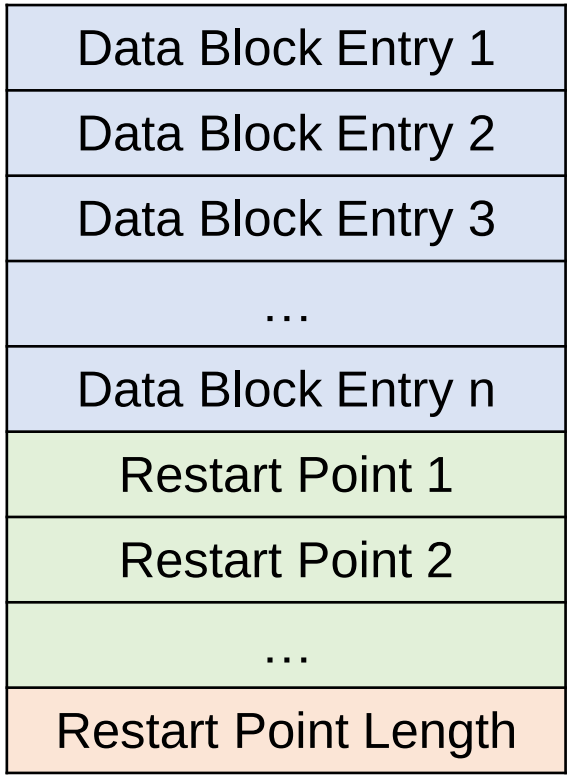
- Storage search



Now Let`s see how find keys in SSTable

Introduction of Block format

- Data Block structure
  - Saves the key-value pairs
  - Save only parts that do not overlap with the previous key instead of saving the entire key
    - **pros**: Saving memory
    - **cons**: Poor read performance (can not use binary search..)
      → solution: Restart Point
  - The entire key is stored for every few entries



Introduction of Block format

- Data Block Entry structure

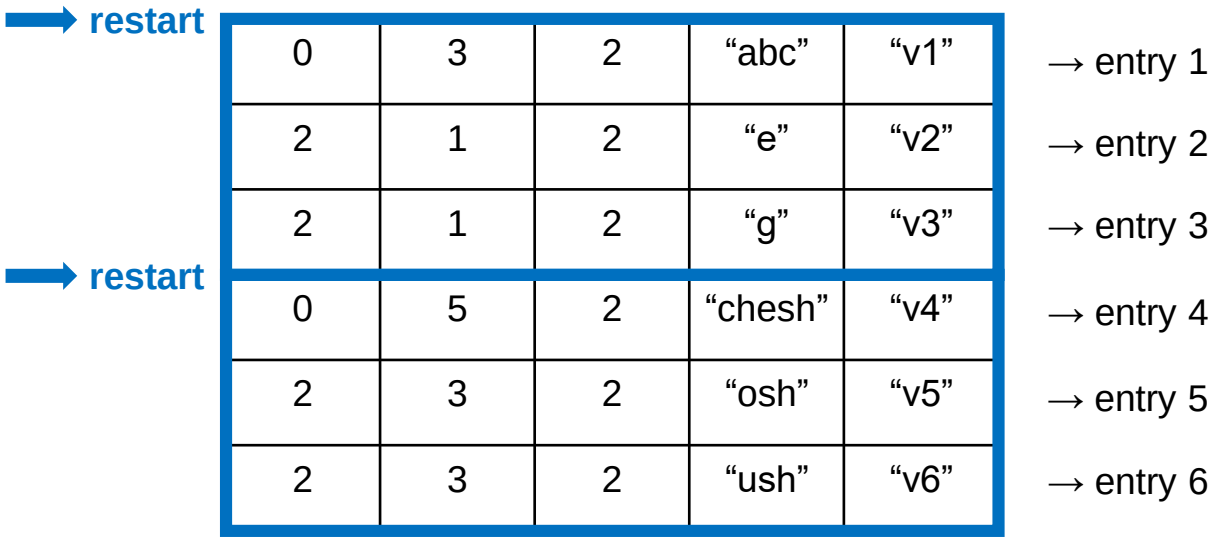
Shared key length	Unshared key length	Value length	Unshared key content	Value

- Shared key length: 이전 레코드의 키와 겹치는 부분의 길이
- Unshared key length: 이전 레코드의 키와 겹치지 않는 부분의 길이
- Value length: Value의 길이
- Unshared key content: 이전 레코드의 키와 겹치지 않는 부분의 내용
- Value: Value

Introduction of Block format

Shared key length	Unshared key length	Value length	Unshared key content	Value

- Example
  - Hypothesis
    - restart_interval = 3
    - entry 1: key = abc, value = v1
    - entry 2: key = abe, value = v2
    - entry 3: key = abg, value = v3
    - entry 4: key = chesh, value = v4
    - entry 5: key = chosh, value = v5
    - entry 6: key = chush, value = v6



Introduction of Block format

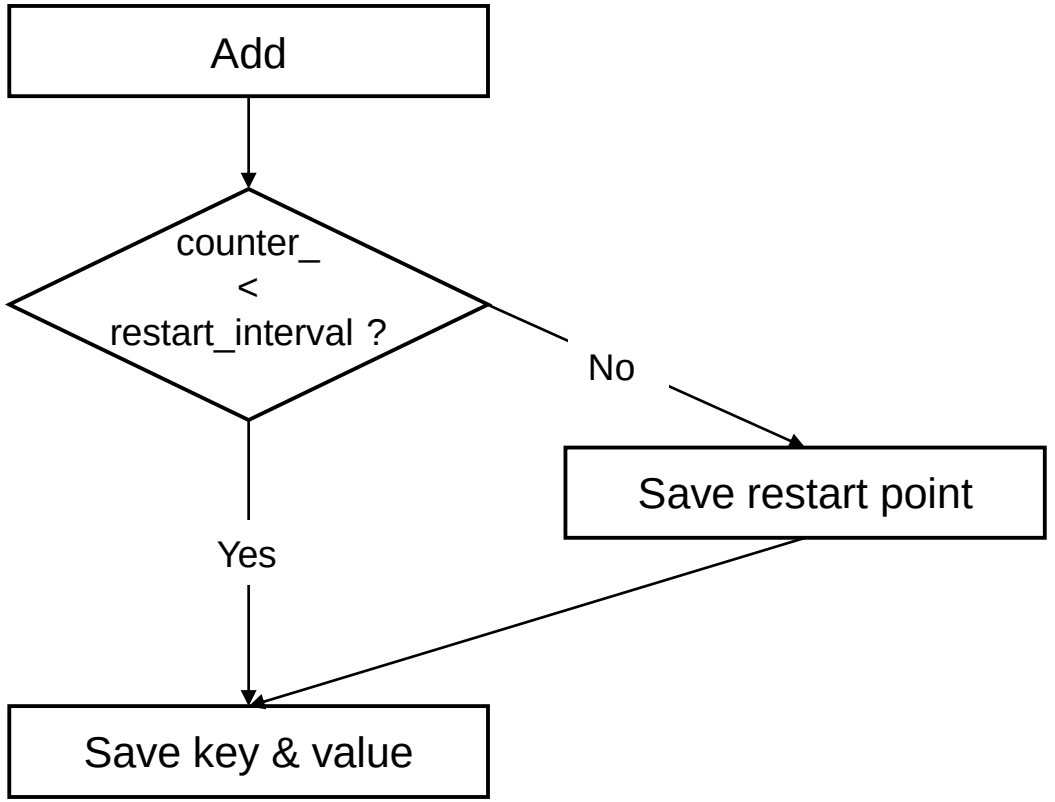
- Index Block structure
  - Saves the Index info of each data blocks
  - Each Entry has
    1. Max key of Data Block i
    2. Offset of Data Block i
    3. Size of Data Block

Max key 1	Offset	Length
Max key 2	Offset	Length
Max key 3	Offset	Length

Introduction of Block format

- More detail of “Write” - BlockBuilder::Add

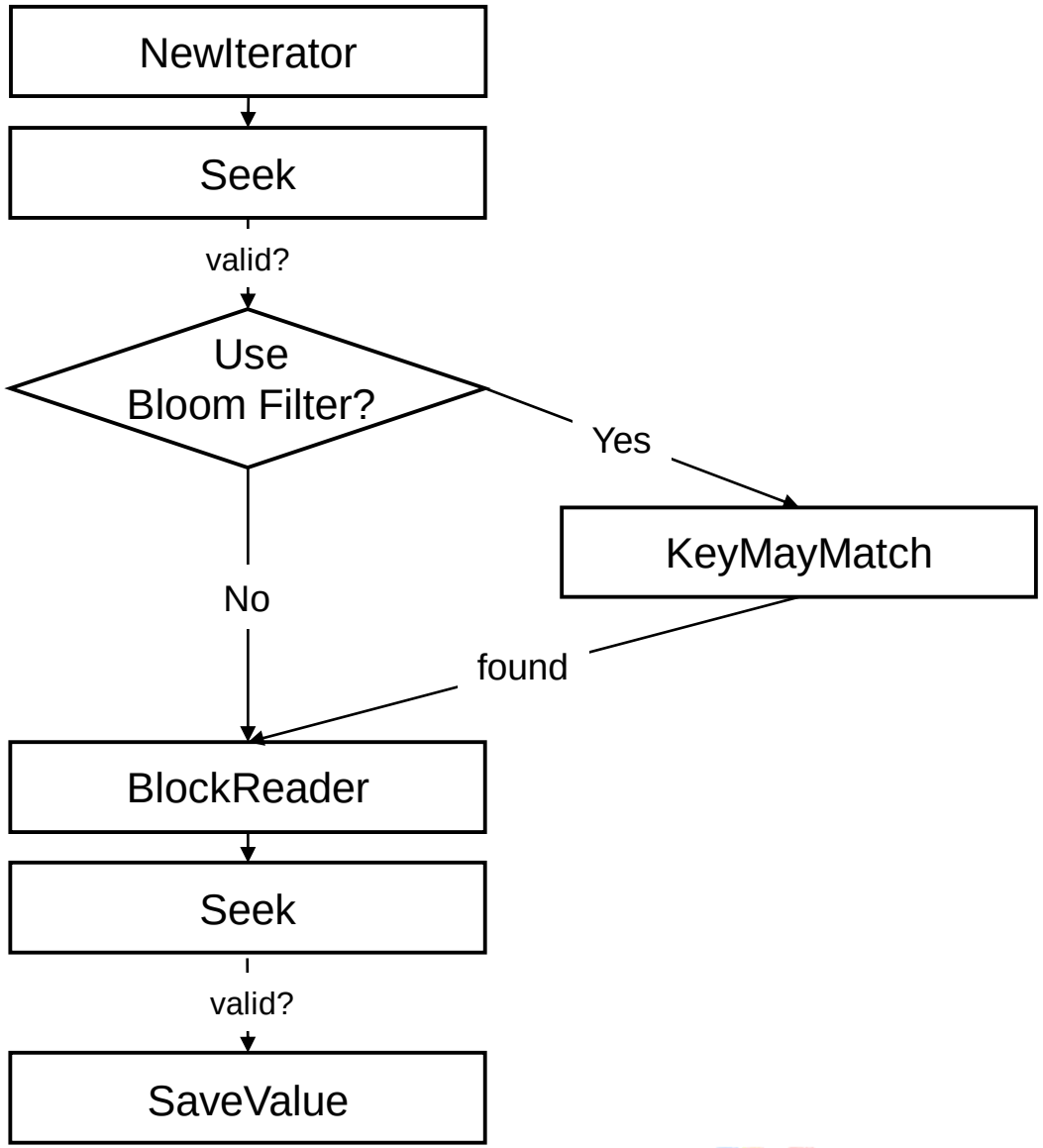
```
void BlockBuilder::Add(const Slice& key, const Slice& value) {
  // …
  if (counter_ < options_->block_restart_interval) {
    // …
  } else {
    restarts_.push_back(buffer_.size());
    counter_ = 0;
  }
  const size_t non_shared = key.size() - shared;

  // Add "<shared><non_shared><value_size>" to buffer_
  PutVarint32(&buffer_, shared);
  PutVarint32(&buffer_, non_shared);
  PutVarint32(&buffer_, value.size());

  // Add string delta to buffer_ followed by value
  buffer_.append(key.data() + shared, non_shared);
  buffer_.append(value.data(), value.size());

  // …

  counter_++;
}
```



InternalGet: Finding keys in SSTable

Overall Process

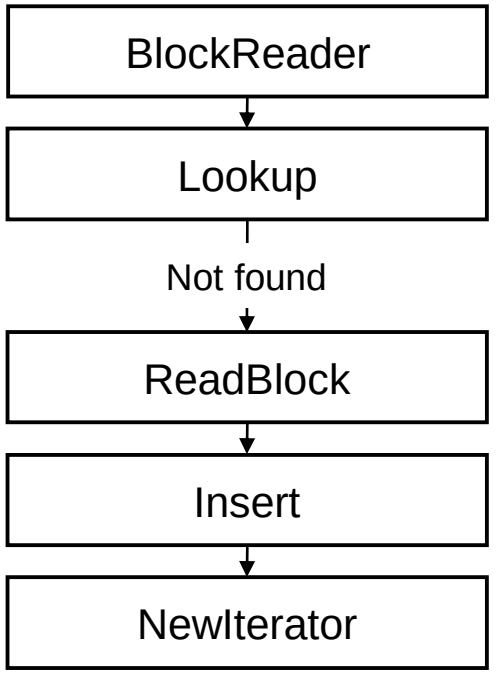
1. Find Index Block that may have a target key
2. If using Bloom Filter, check if there a target key
3. If there is a target key, create an Iterator for the Data Block that Index Block refers to
4. Find a target key in the Data Block
5. Save value if found



InternalGet: Finding keys in SSTable

BlockReader

- ✓ Returns an Iterator for the Data Block referenced by the Index Block Entry that we found
- ✓ Check if the corresponding data block is cached
- ✓ If not, read the block contents and put it in the cache

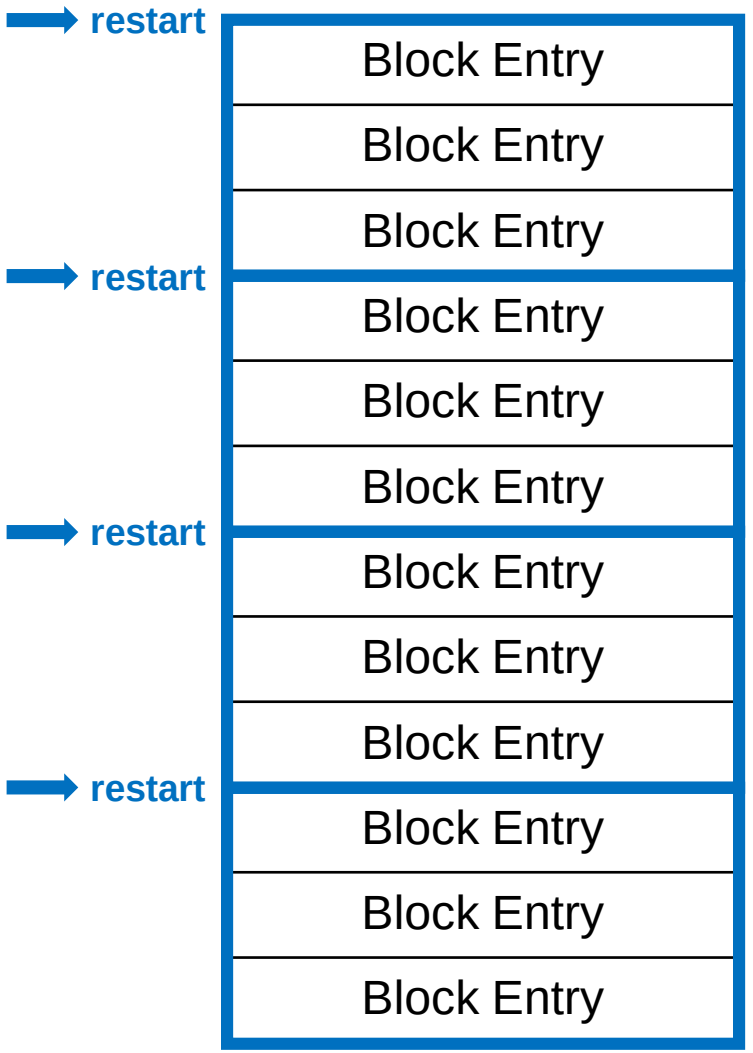


InternalGet: Finding keys in SSTable

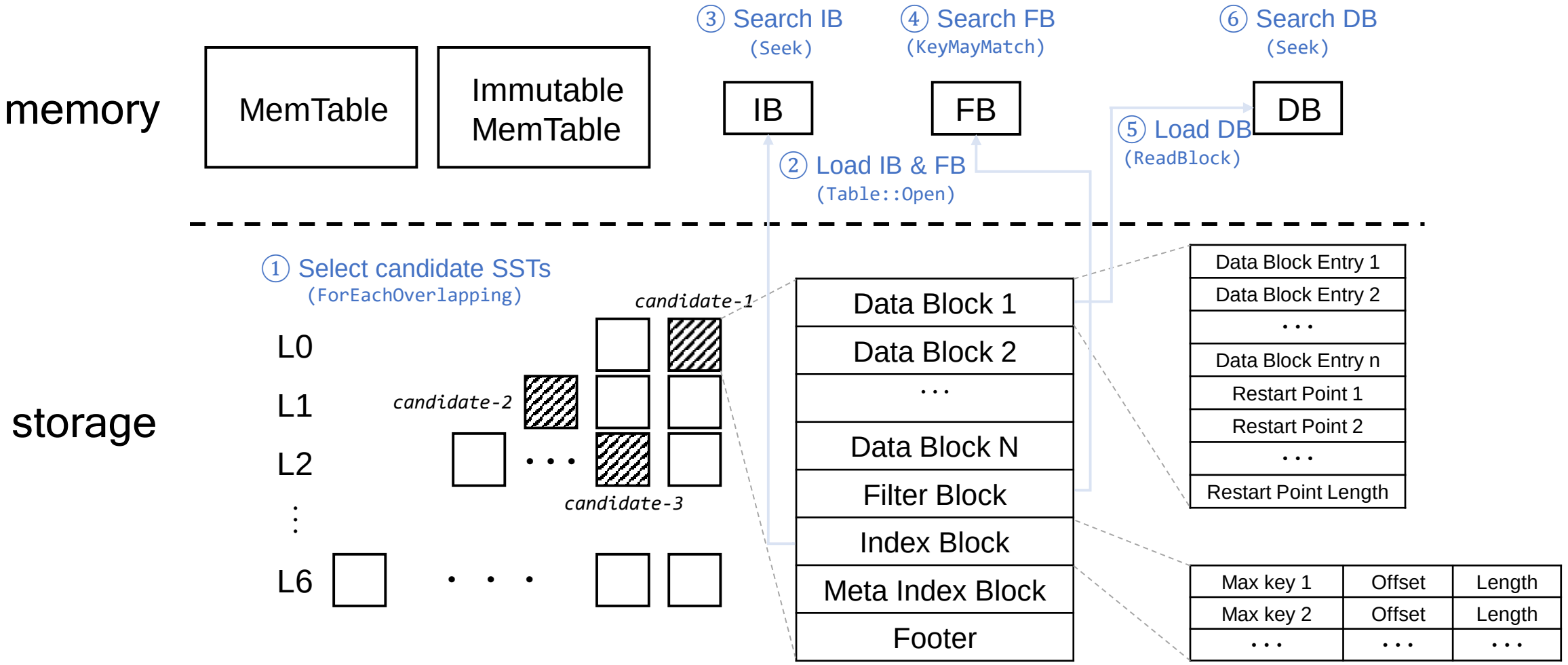
Seek

- ✓ Entire key is stored for each restart point
  - It means that entries in the Block form a kind of area
- ✓ Use Binary search to find the area where target key may be located
- ✓ And use Linear search to find target key in the area

```
void Seek(const Slice& target) override {
 // …
  // Binary search in restart array to find the last restart point
  while (left < right) {
    uint32_t mid = (left + right + 1) / 2;
  // …
    if (Compare(mid_key, target) < 0) {
       left = mid;
    } else {
    right = mid - 1;
    }
  }
 // …
   // Linear search (within restart block) for first key >= target
  while (true) {
    if (!ParseNextKey()) return;
    if (Compare(key_, target) >= 0) return;
  }
}
```



Summary


memory
MemTable
Immutable MemTable
③ Search IB
(Seek)
④ Search FB
(KeyMayMatch)
⑥ Search DB
(Seek)
IB
FB
DB
⑤ Load DB
(ReadBlock)
② Load IB & FB
(Table::Open)
storage
① Select candidate SSTs
(ForEachOverlapping)
candidate-1
candidate-2
candidate-3
L0
L1
L2
L6
Data Block 1
Data Block 2
Data Block N
Filter Block
Index Block
Meta Index Block
Footer
Data Block Entry 1
Data Block Entry 2
Data Block Entry n
Restart Point 1
Restart Point 2
Restart Point Length
Max key 1
Offset
Length
Max key 2
Offset
Length

Reference

- https://leveldb-handbook.readthedocs.io/zh/latest/
- https://zhuanlan.zhihu.com/p/149796078