Team Compaction

강상우
E-Mail : aarom416@naver.com

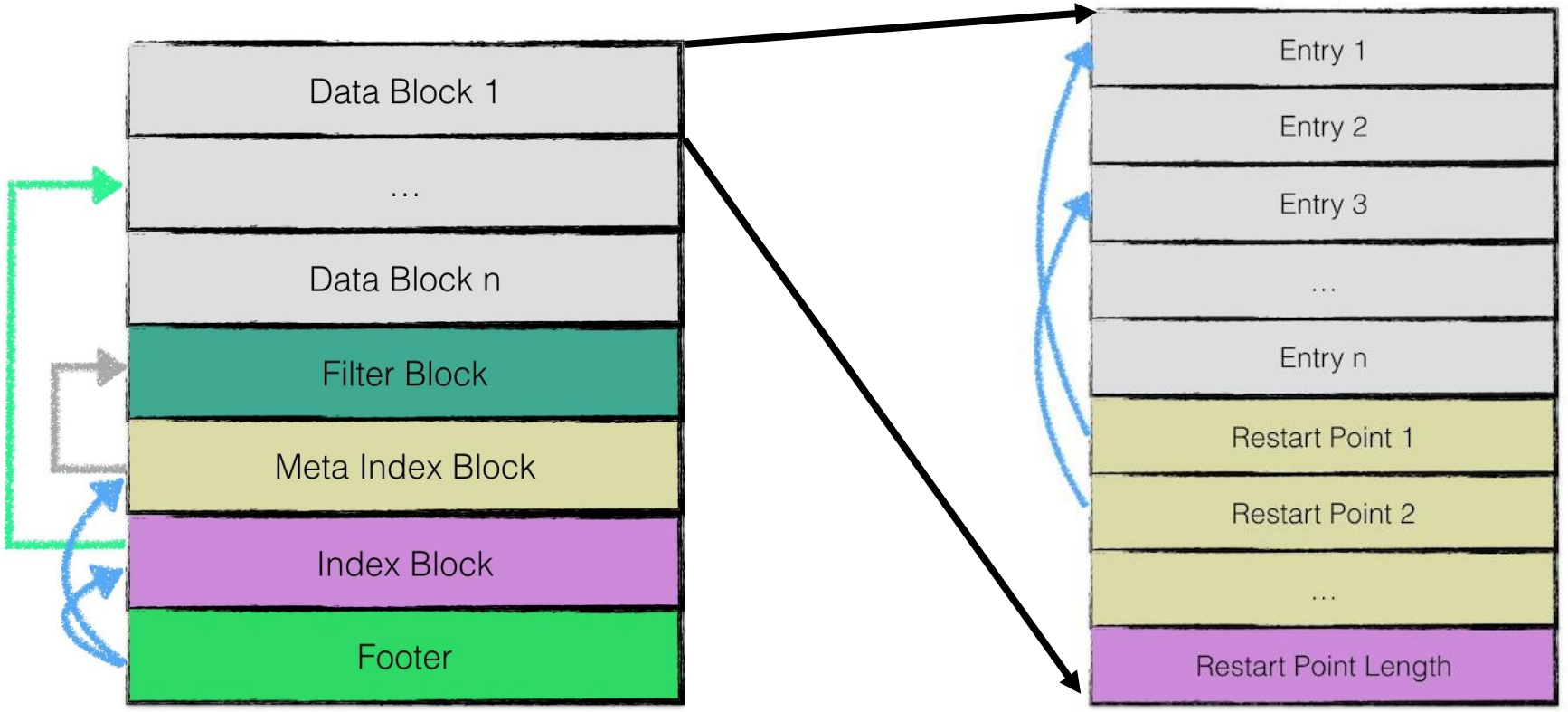
발표: 좌우꾸와쒼
E-Mail : erosbryant@dankook.ac.kr

Contents

- TwoLevelIterator
- MergingIterator

TwoLevelIterator

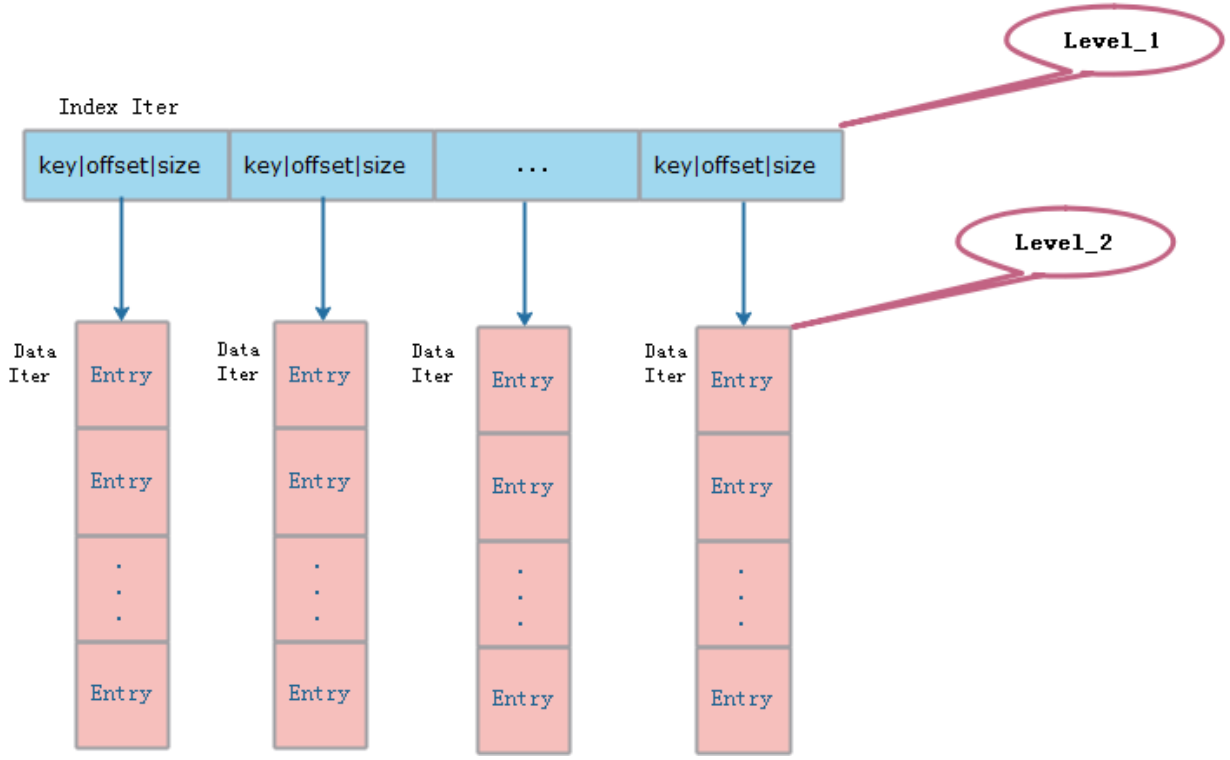
- Using in the SST



https://leveldb-handbook.readthedocs.io/zh/latest/sstable.html

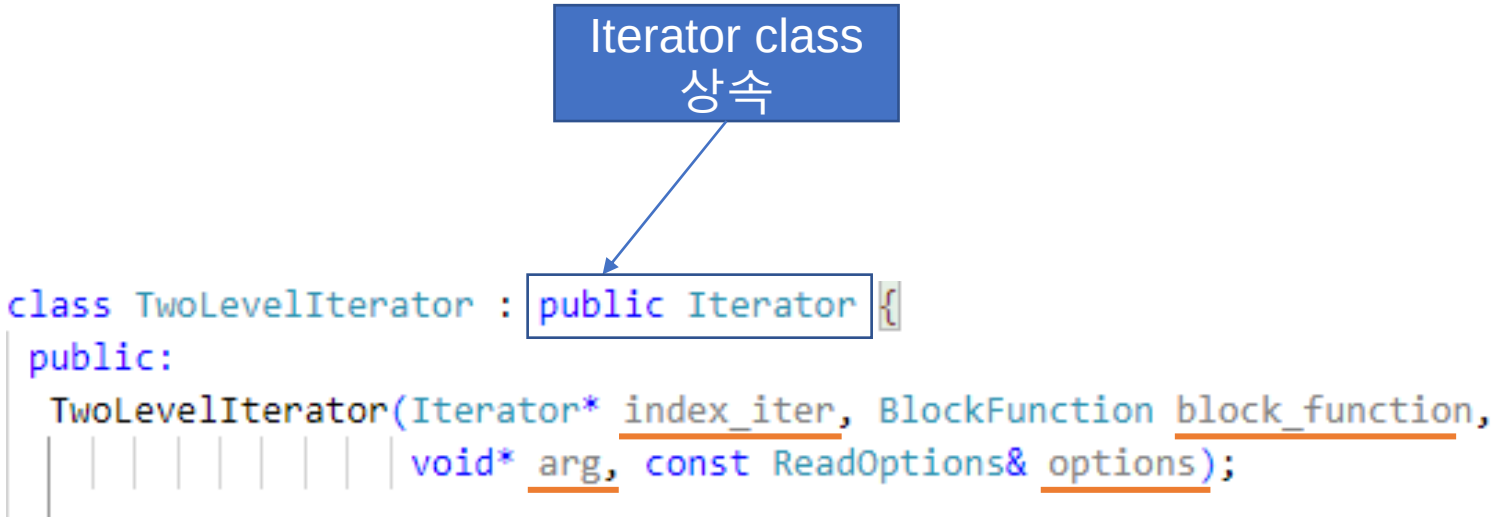
https://leveldb-handbook.readthedocs.io/zh/latest/sstable.html

TwoLevelIterator



https://img-blog.csdnimg.cn/20200614175126109.png#pic_center?x-oss-process=image/watermark,type_ZmFuZ3poZW5naGVpdGk,shadow_10,text_aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L0g1MTQ0MzQ0ODU=,size_16,color_FFFFFF,t_70

TwoLevelIterator



SST의 경우:

1. index_iter는 index block을 가리키는 iter
2. block_function은 Table::BlockReader, 즉 블록을 읽는 것
3. arg는 SST을 가리키는 것
4. Options 읽기 옵션

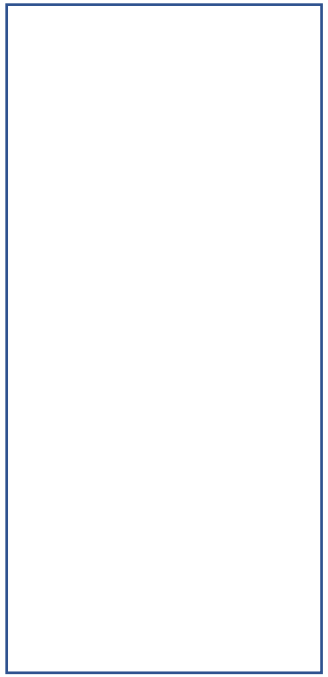
MergingIterator

- MergingIterator
  - 이름 대로 Iterator들을 merge
- NewInternalIterator
  - 핵심 함수

```
Iterator* DBImpl::NewInternalIterator(const ReadOptions& options,
                                      SequenceNumber* latest_snapshot,
                                      uint32_t* seed) {
  mutex_.Lock();
  *latest_snapshot = versions_->LastSequence();

  // Collect together all needed child iterators
  std::vector<Iterator*> list;
  list.push_back(mem_->NewIterator());
  mem_->Ref();
  if (imm_ != nullptr) {
    list.push_back(imm_->NewIterator());
    imm_->Ref();
  }
  versions_->current()->AddIterators(options, &list);
  Iterator* internal_iter =
      NewMergingIterator(&internal_comparator_, &list[0], list.size());
  versions_->current()->Ref();

  IterState* cleanup = new IterState(&mutex_, mem_, imm_, versions_->current());
  internal_iter->RegisterCleanup(CleanupIteratorState, cleanup, nullptr);

  *seed = ++seed_;
  mutex_.Unlock();
  return internal_iter;
}
```

MergingIterator

Vector<iterator*>



Memtable

Imm Memtable

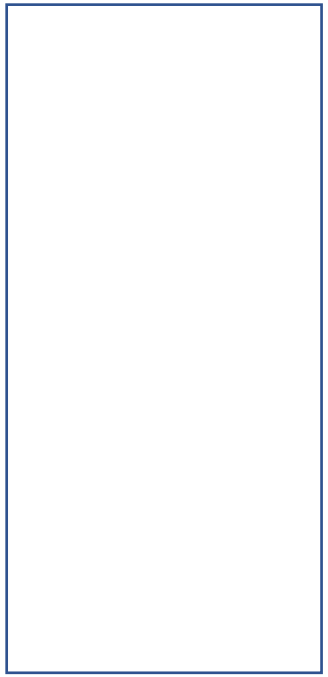
level 0

level 1 ~ N

```
void Version::AddIterators(const ReadOptions& options,
                           std::vector<Iterator*>* iters) {
  // Merge all level zero files together since they may overlap
  for (size_t i = 0; i < files_[0].size(); i++) {
    iters->push_back(vset_->table_cache_->NewIterator(
        options, files_[0][i]->number, files_[0][i]->file_size));
  }

  // For levels > 0, we can use a concatenating iterator that sequentially
  // walks through the non-overlapping files in the level, opening them
  // lazily.
  for (int level = 1; level < config::kNumLevels; level++) {
    if (!files_[level].empty()) {
      iters->push_back(NewConcatenatingIterator(options, level));
    }
  }
}
```

MergingIterator

Vector<iterator*>



Memtable

Imm Memtable

level 0

level 1 ~ N

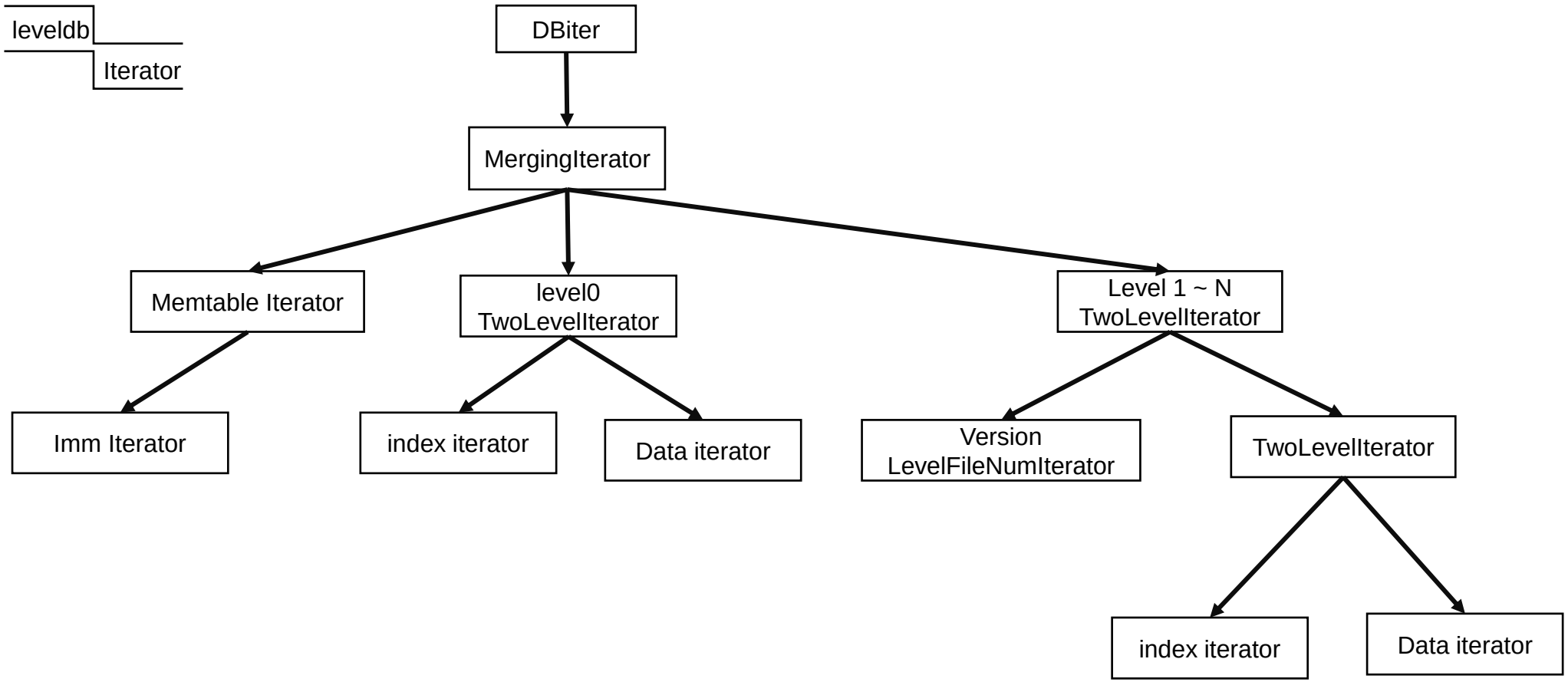
```
void Version::AddIterators(const ReadOptions& options,
                           std::vector<Iterator*>* iters) {
  // Merge all level zero files together since they may overlap
  for (size_t i = 0; i < files_[0].size(); i++) {
    iters->push_back(vset_->table_cache_->NewIterator(
        options, files_[0][i]->number, files_[0][i]->file_size));
  }

  // For levels > 0, we can use a concatenating iterator that sequentially
  // walks through the non-overlapping files in the level, opening them
  // lazily.
  for (int level = 1; level < config::kNumLevels; level++) {
    if (!files_[level].empty()) {
      iters->push_back(NewConcatenatingIterator(options, level));
    }
  }
}
```

MergingIterator

- MergingIterator
  - 이름 대로 Iterator들을 merge
- NewInternalIterator
  - 핵심 함수

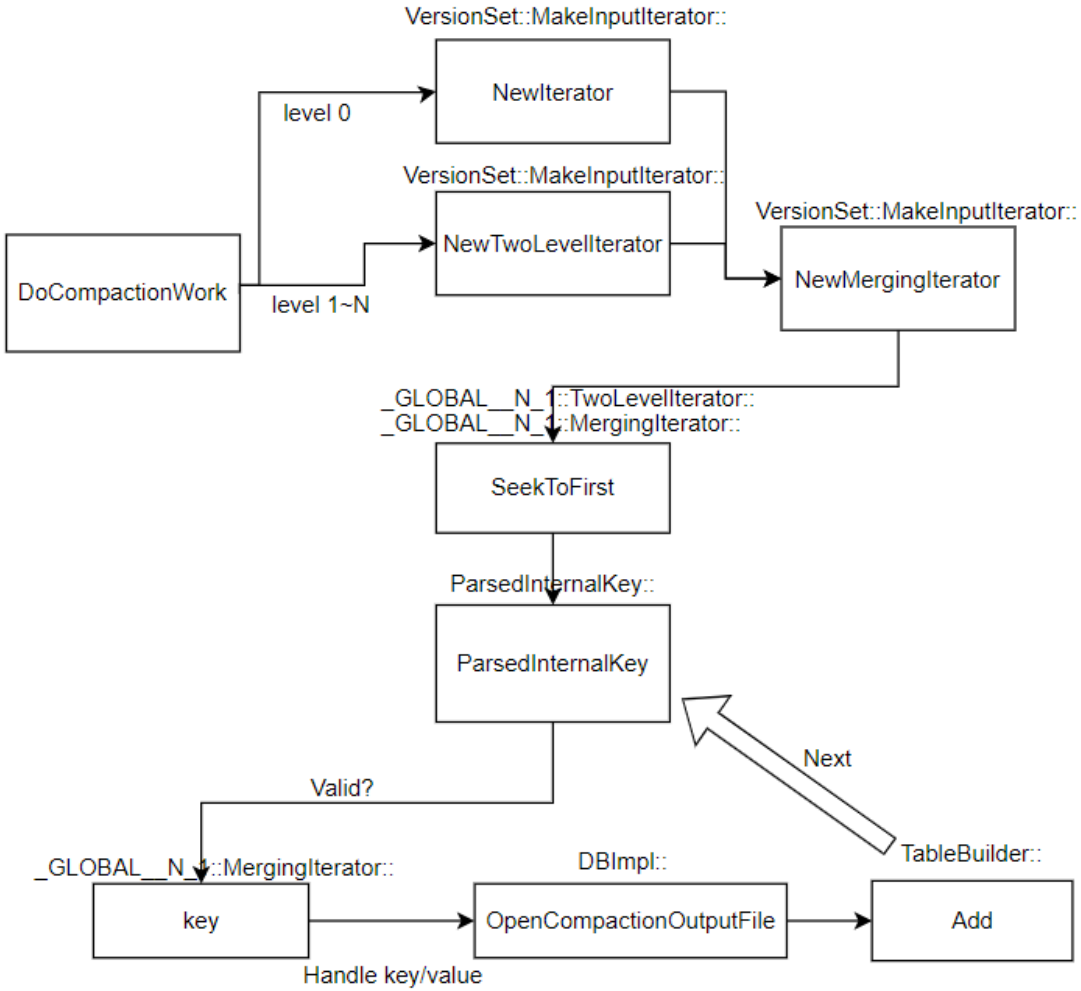
```
Iterator* DBImpl::NewInternalIterator(const ReadOptions& options,
                                      SequenceNumber* latest_snapshot,
                                      uint32_t* seed) {
  mutex_.Lock();
  *latest_snapshot = versions_->LastSequence();

  // Collect together all needed child iterators
  std::vector<Iterator*> list;
  list.push_back(mem_->NewIterator());
  mem_->Ref();
  if (imm_ != nullptr) {
    list.push_back(imm_->NewIterator());
    imm_->Ref();
  }
  versions_->current()->AddIterators(options, &list);
  Iterator* internal_iter =
      NewMergingIterator(&internal_comparator_, &list[0], list.size());
  versions_->current()->Ref();

  IterState* cleanup = new IterState(&mutex_, mem_, imm_, versions_->current());
  internal_iter->RegisterCleanup(CleanupIteratorState, cleanup, nullptr);

  *seed = ++seed_;
  mutex_.Unlock();
  return internal_iter;
}
```

Iterator 기능적 흐름도



https://www.jianshu.com/p/9e63dea36af0

DoCompactionWork



Thank you