00A0 $\exists$ $\forall$ $\subseteq$ 2713x 27FA$\Longleftrightarrow$ $\surd$ $\sqrt[3]{}$ $\oplus$ $\otimes$

leveldb-handbook Documentation

Gary Rong

2022 03 29

:

CHAPTER 1

leveldb LSM (Log Structured-Merge Tree) LSM

LSM 1 2 leveldb 60MB/s, 45MB/s

leveldb

1.1

leveldb

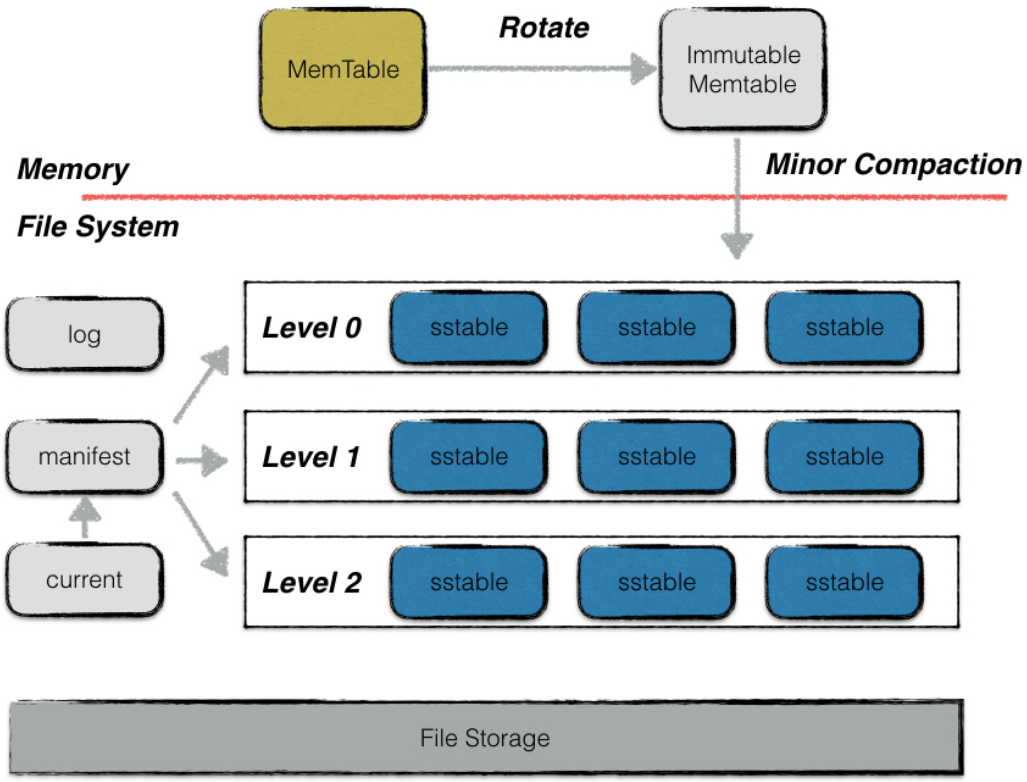
- memtable
- immutable memtable
- log(journal)
- sstable
- manifest
- current

1.1.1 memtable

leveldb memtable memtable 4MB memtable n)

1.1.2 immutable memtable

memtable memtable immutable memtable immutable memtable immutable memtable leveldb sstable


MemTable
Rotate
Immutable
Memtable
Memory
Minor Compaction
File System
log
manifest
current
Level 0
Level 1
Level 2
sstable
sstable
sstable
sstable
sstable
sstable
sstable
sstable
sstable
File Storage

1.1.3 log

leveldb leveldb leveldb log

1. log
2. log
3. write log
4. Immutable memtable
5.

log

redo

leveldb

1.1.4 sstable

leveldb leveldb sstable

memetable sstable sstable leveldb “ “ sstable compaction compac
0 level i leveldb

sstable leveldb

1.1.5 manifest

leveldb 1 2 key 3 key key compaction compaction

goleveldb key

```
// tFile holds basic information about a table.
type tFile struct {
    fd         storage.FileDesc
    seekLeft   int32
    size       int64
    imin, imax internalKey
}
```

```
type version struct {
    s *session // session - version

    levels []tFiles // file meta

    // Level that should be compacted next and its compaction score.
    // Score < 1 means compaction is not strictly needed. These fields
    // are initialized by computeCompaction()
    cLevel int // next level
    cScore float64 // current score
```

( )

( )

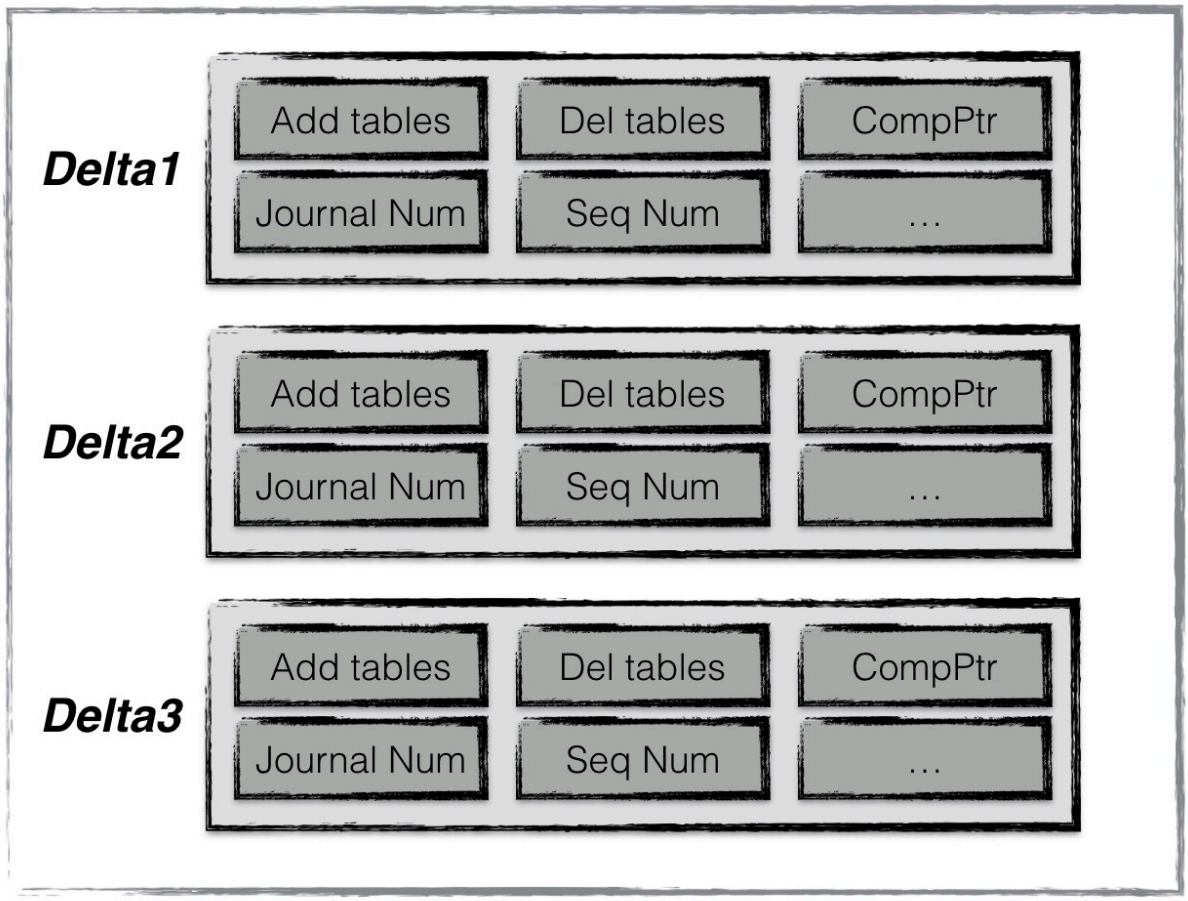
```
    cSeek unsafe.Pointer

    closing  bool
    ref      int
    released bool
}
```

compaction sstable leveldb version :

versionNew = versionOld + versionEdit

versionEdit sstable

manifest **versionEdit** versionEdit manifest manifest 3 versionEdit 1 sst 2 sst



1.1.6 current

manifest

leveldb Manifest Manifest Current Manifest Manifest

CHAPTER 2

2.1

leveldb leveldb

2.1.1

leveldb

1.
2.

: leveldb

2.1.2

leveldb 1 Put 2 Delete Delete value Put

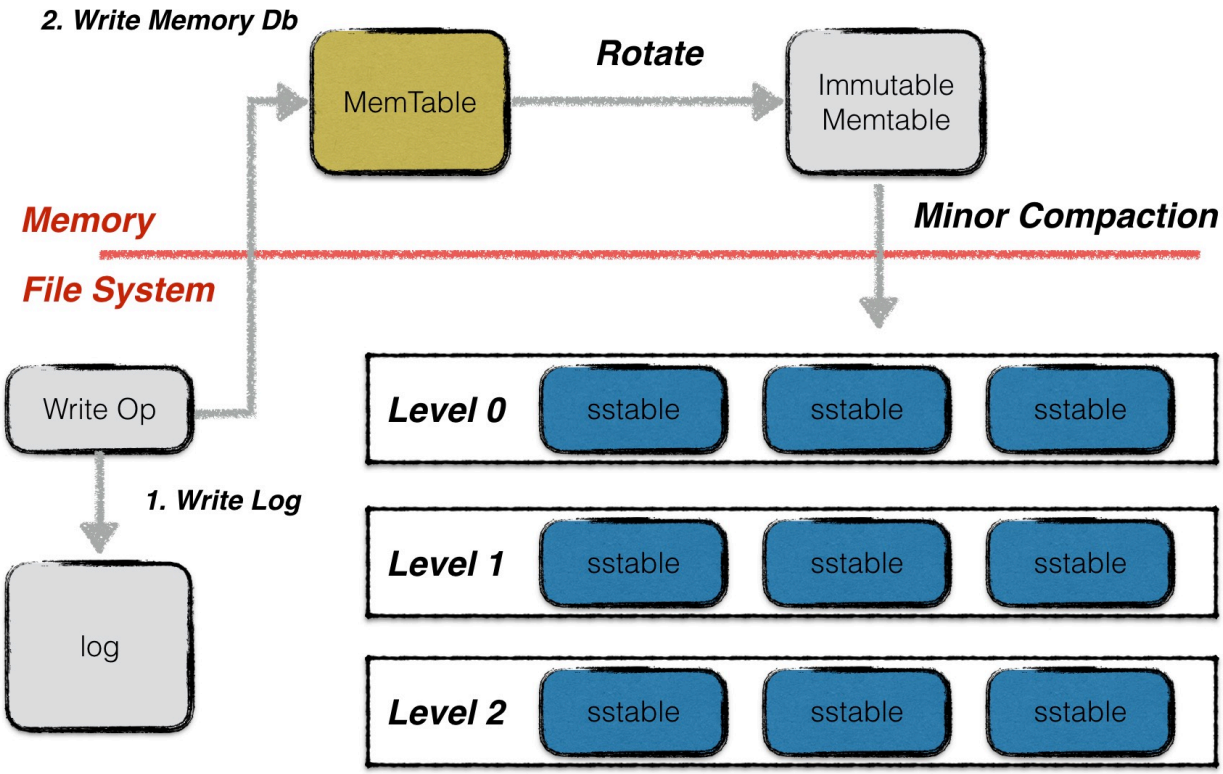
leveldb Batch Batch

2.1.3 batch

Put/Del batch batch

batch key key value value

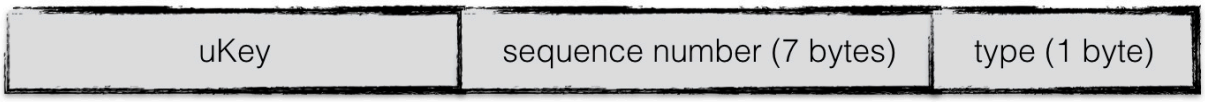
batch size size key value 8 8


2. Write Memory Db
MemTable
Rotate
Immutable Memtable
Memory
File System
Minor Compaction
Write Op
1. Write Log
log
Level 0
sstable
sstable
sstable
Level 1
sstable
sstable
sstable
Level 2
sstable
sstable
sstable


type
key length
key
value length
value

2.1.4 key

batch key leveldb key internalKey



internalkey key 8 1 sequence number 2

sequence number leveldb leveldb append key sequence number

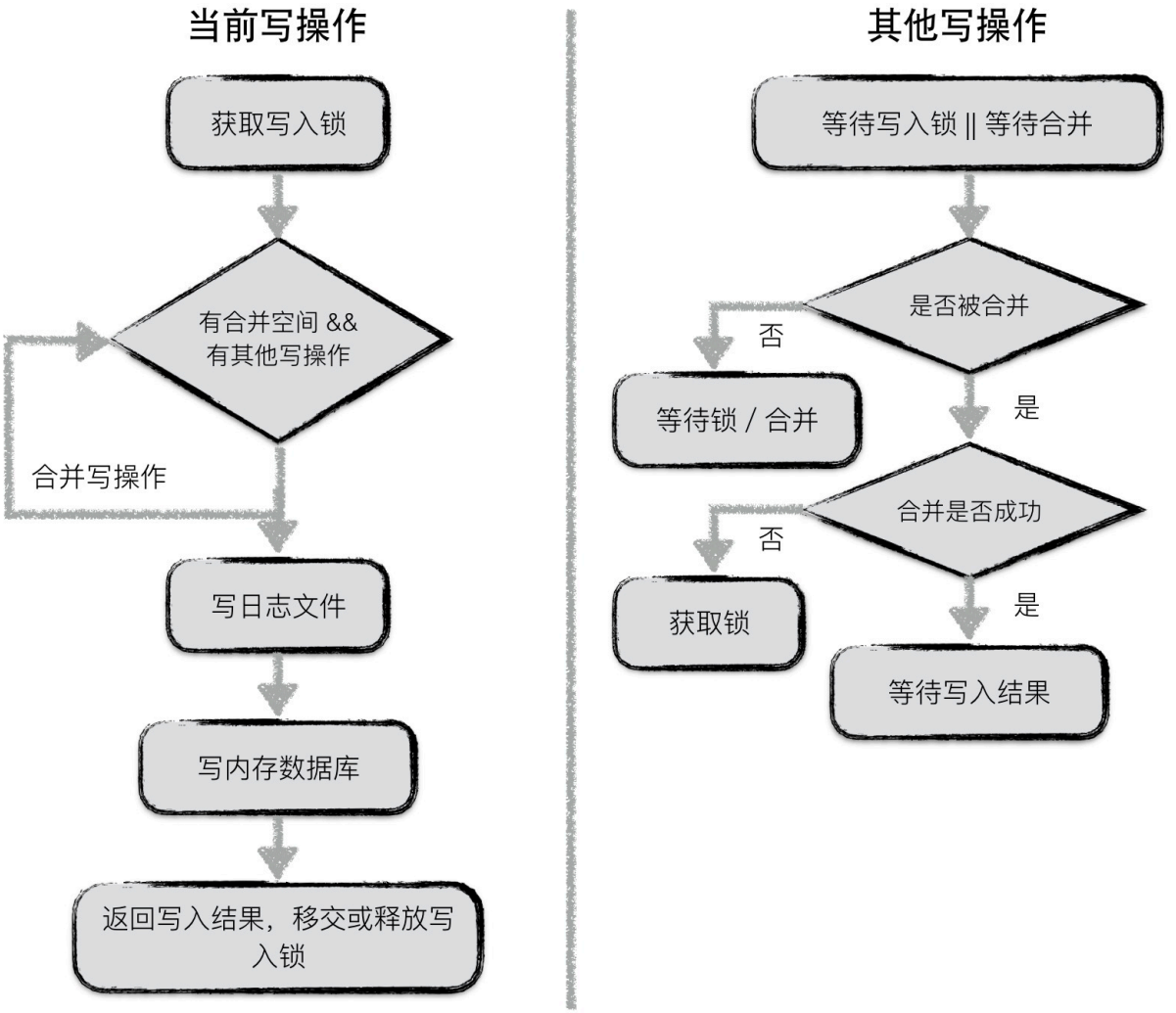
leveldb snapshot sequence number sequence number

2.1.5

leveldb leveldb “ ” “ ”

-
- pending
- pending
-

-
- oversize
-



2.1.6

leveldb

1.
2.

redo

2.1.7

leveldb - leveldb -

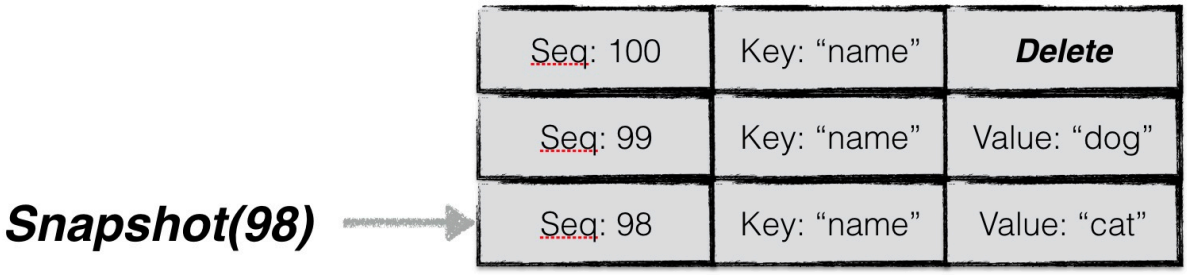
2.2

leveldb

1. `Get`
2. snapshot snapshot Get
   snapshot snapshot
   snapshot

2.2.1

leveldb
leveldb key compaction
leveldb
leveldb 98 key “name” ”name” ”dog” “cat”



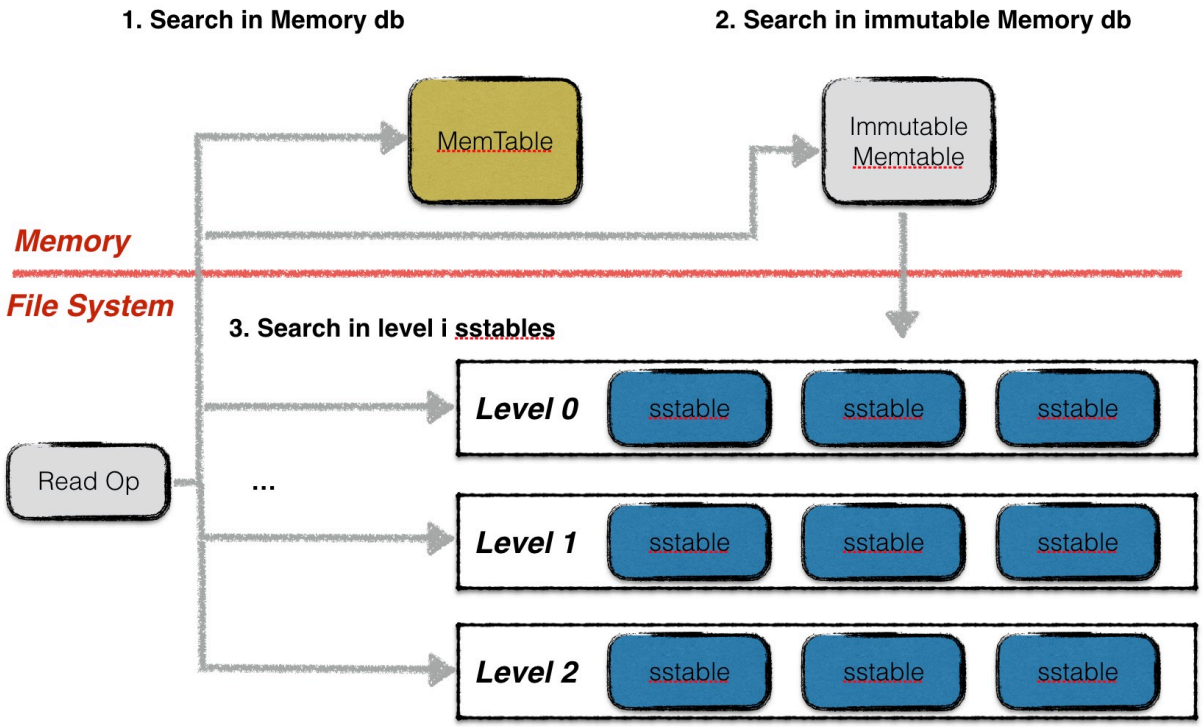
leveldb key internalKey internalKey seq seq **seq**

2.3

leveldb

1. memory db key
2. memory db key
3. level i sstable key Not Found

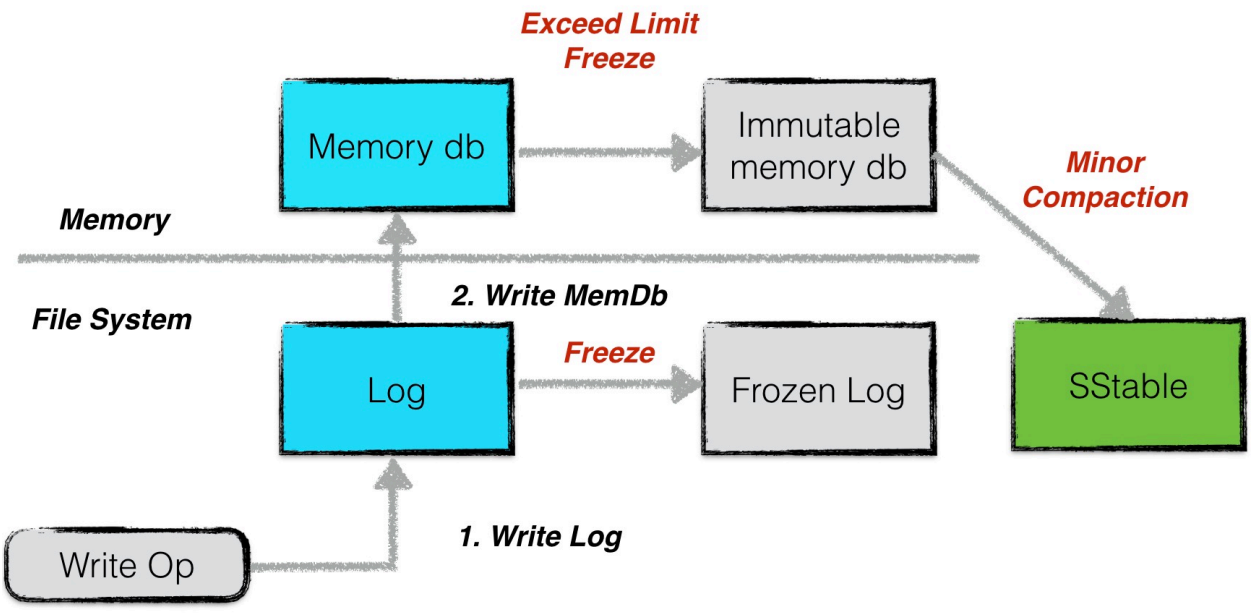
: leveldb sstable sstable
0 0 key 0 sstable sstable
0 key leveldb sstable key sstable



memory db sstable internalKey key seq seq sstable

CHAPTER 3

leveldb



leveldb memory db memory db db memory db frozen log immutable memory db minor compaction sstable frozen log

3.1

block block 32KiB block chunk

chunk chunk 7 header 4 chunk 2 chunk chunk checksum chunk data

chunk full first middle last chunk chunk full chunk chunk first, last 0 middle chunk

block 32KiB block first block block middle block last

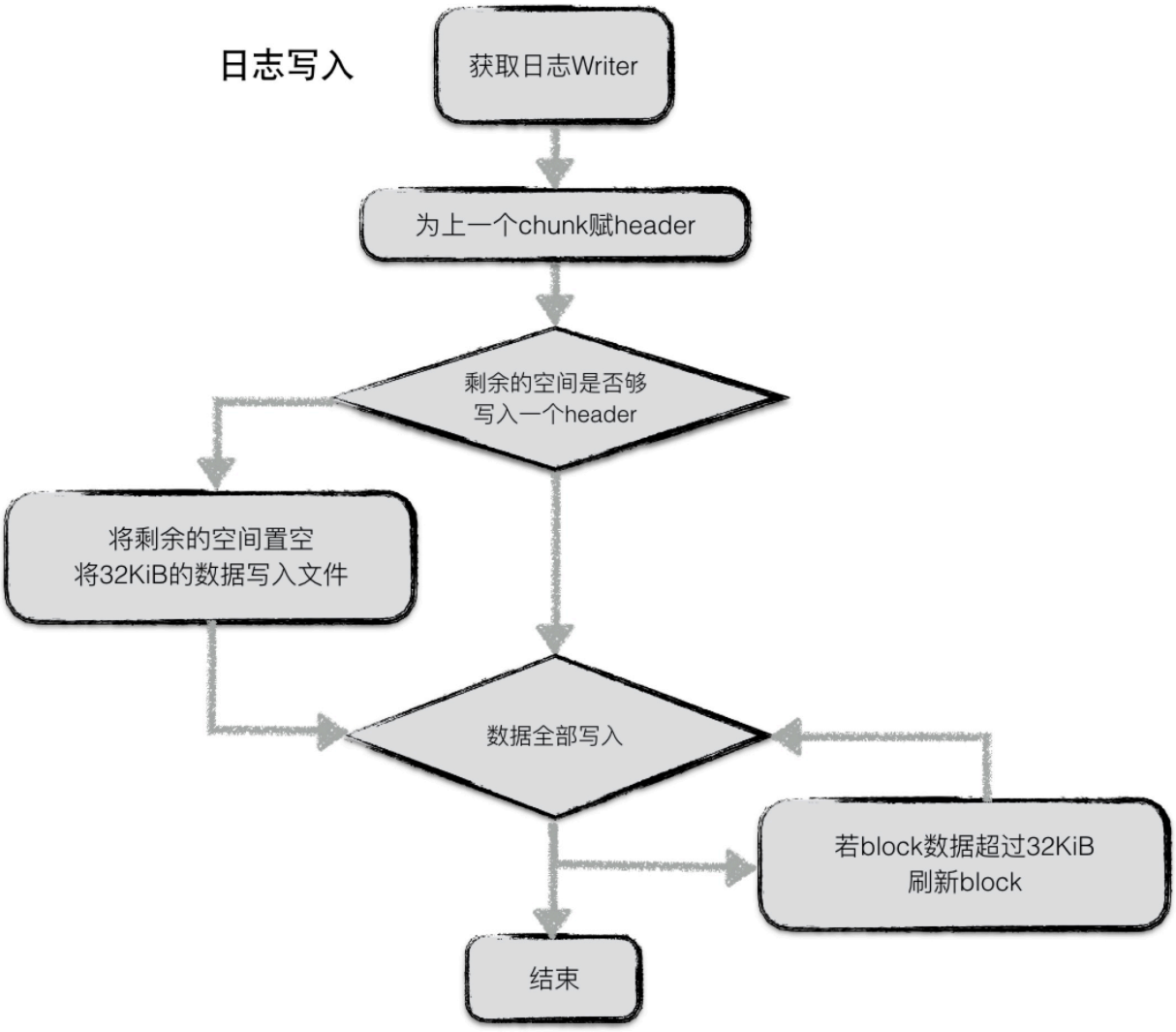
3.2

batch

- Header
- Data

header 1 db sequence number 2 put/del

batch batch .


日志写入
获取日志Writer
为上一个chunk赋header
剩余的空间是否够
写入一个header
将剩余的空间置空
将32KiB的数据写入文件
数据全部写入
若block数据超过32KiB
刷新block
结束

3.3

leveldb journal writer Next singleWriter singleWriter **journal**

singleWriter chunk writer buffer 32KiB chunk chunk header chunk reset buffer chunk

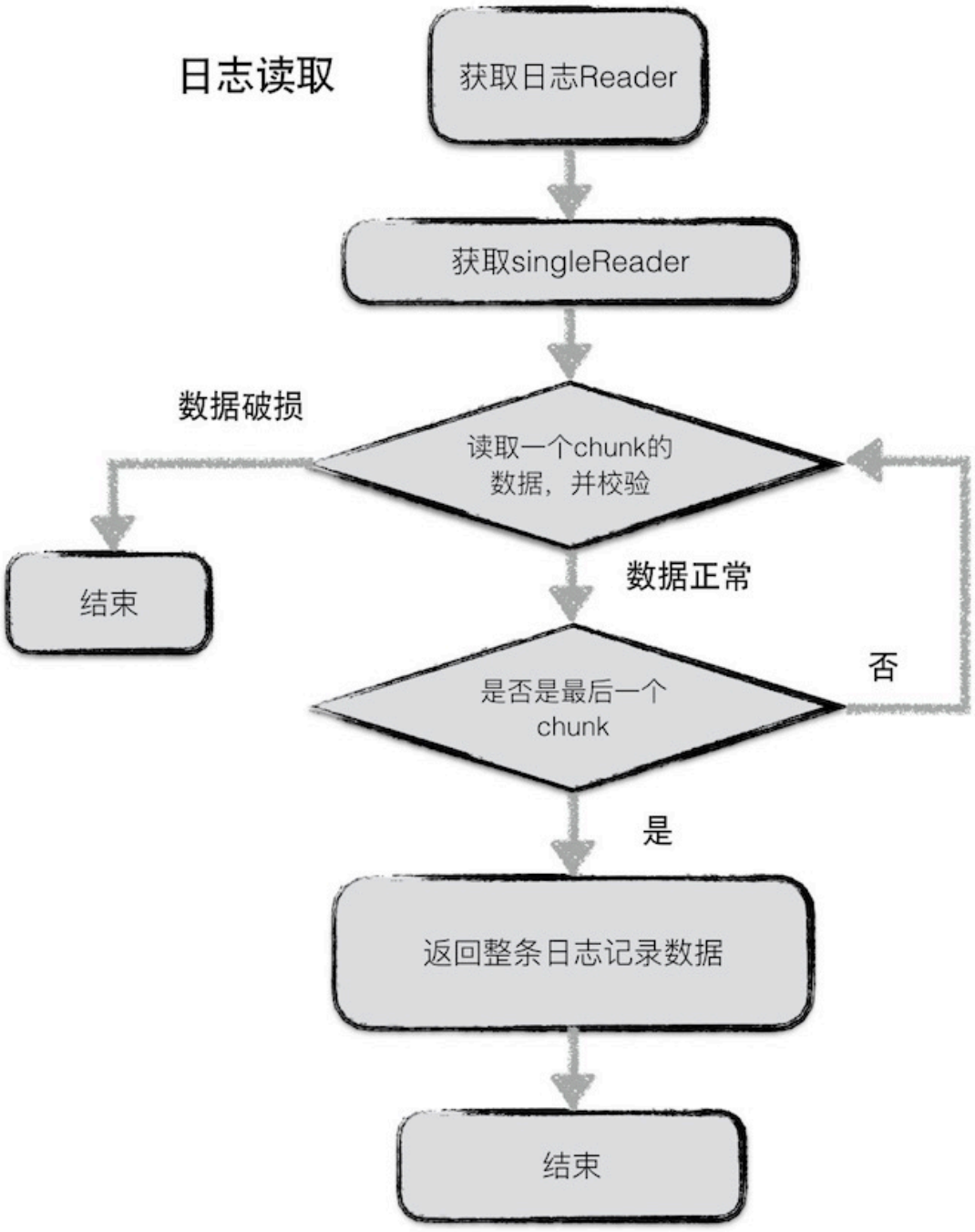
journal chunk block

3.4

IO block 32KiB

reader Next singleReader singleReader Read chunk chunk chunk

singleReader read Last chunk


日志读取
获取日志Reader
获取singleReader
数据破损
读取一个chunk的
数据，并校验
结束
数据正常
否
是否是最后一个
chunk
是
返回整条日志记录数据
结束

CHAPTER 4

leveldb key-value O(log n)

4.1

4.1.1

SkipList William Pugh Skip lists: a probabilistic alternative to balanced trees
O(log n)

> *Skip lists are a data structure that can be used in place of balanced trees. Skip lists use probabilistic balancing rather than strictly enforced balancing and as a result the algorithms for insertion and deletion in skip lists are much simpler and significantly faster than equivalent algorithms for balanced trees.*

a N
b 2 2 n/2 + 1
c b 4 4 n/4 + 2
2^ i O(log n)

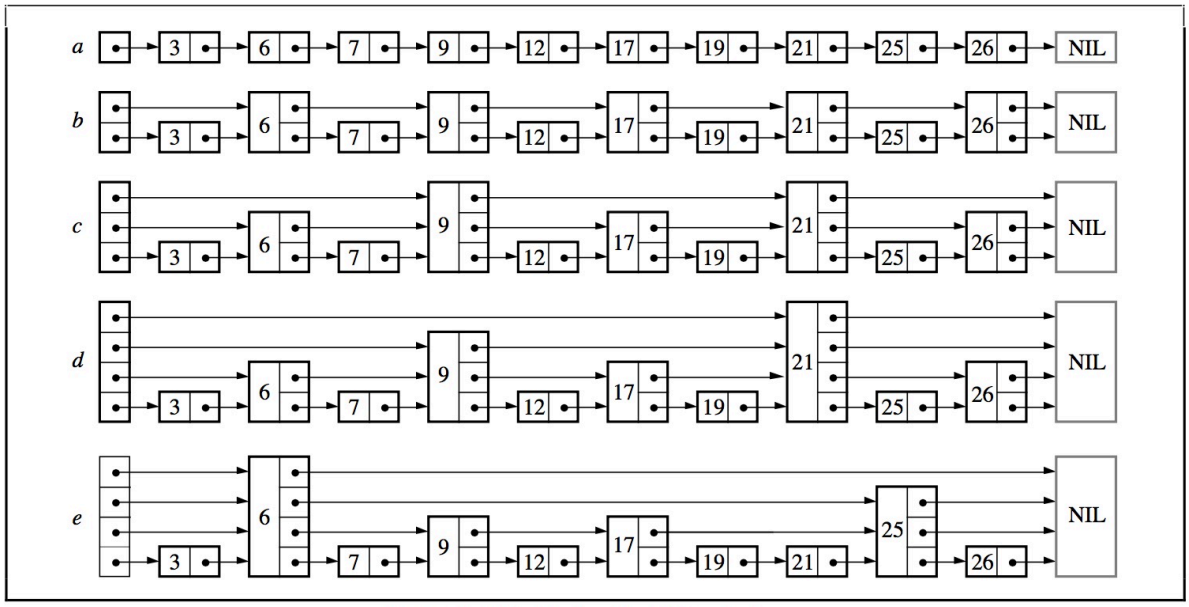
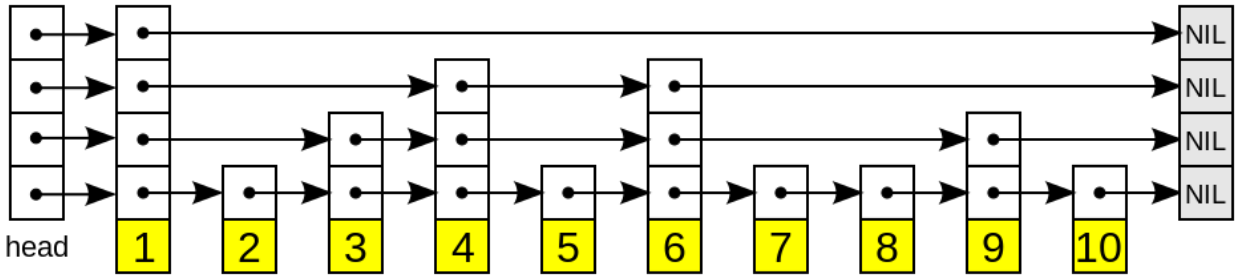


FIGURE 1 - Linked lists with additional pointers

k k *level k node* 50% 1 25% 2 12.5% 3 e

4.1.2



” ” i p (0.5 0.25) i+1 1/(1-p)
O(log1/p n)

4.1.3

17

-
-
-

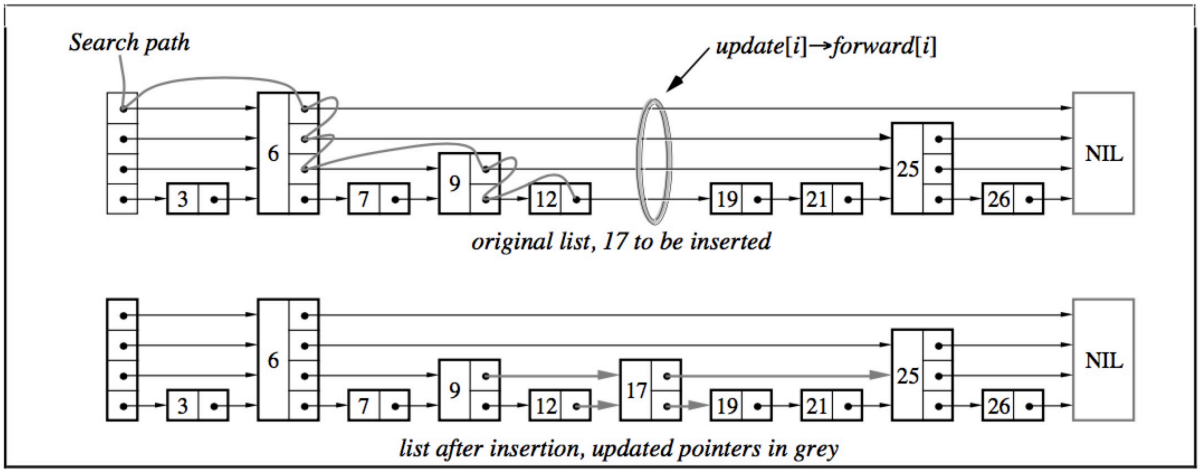


FIGURE 3 - Pictorial description of steps involved in performing an insertion

- 0 0 key key

4.1.4

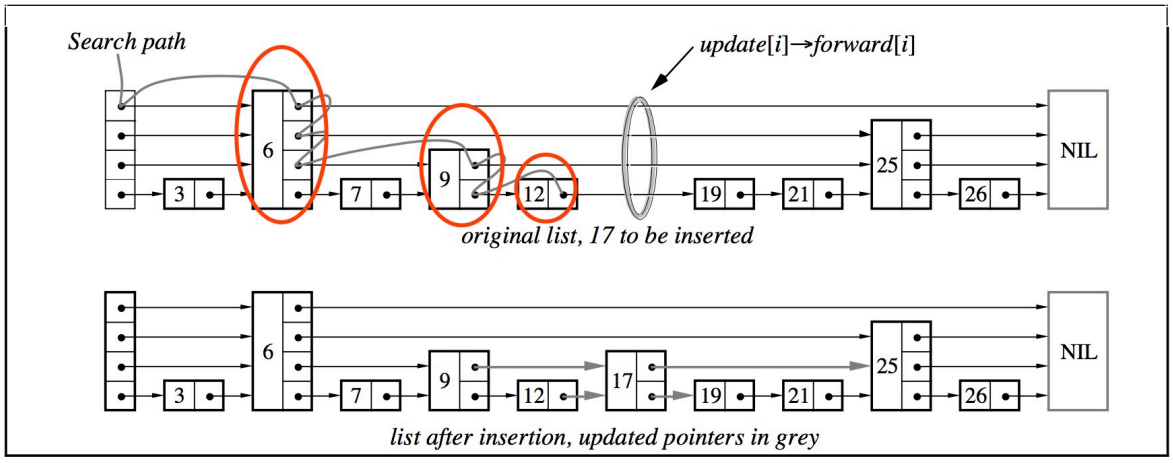


FIGURE 3 - Pictorial description of steps involved in performing an insertion

-
- p
- 12 19

Next Next Next

```
func (p *DB) randHeight() (h int) {
    const branching = 4
```

( )

( )

```
    h = 1
    for h < tMaxHeight && p.rnd.Int()%branching == 0 {
        h++
    }
    return
}
```

4.1.5

Next Next

4.1.6

- [Start, Limit)
-
- keyvalue key value keyvalue

- [Start, Limit
- key key
- keyvalue key value keyvalue

4.2

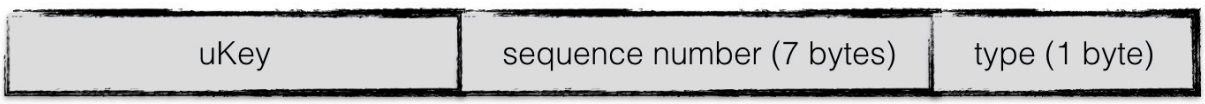
leveldb

4.2.1

kv key

key internalKey

- key key key
- leveldb sequence number leveldb key
-



4.2.2

- key ukey key internalKey
- key internalKey

key key

4.2.3

goleveldb

```
type DB struct {
    cmp comparer.BasicComparer
    rnd *rand.Rand

    mu     sync.RWMutex
    kvData []byte
    // Node data:
    // [0]         : KV offset
    // [1]         : Key length
    // [2]         : Value length
    // [3]         : Height
    // [3..height] : Next nodes
    nodeData  []int
    prevNode  [tMaxHeight]int
    maxHeight int
    n         int
    kvSize    int
}
```

kvData key-value nodeData

nodeData

- key-value kvData
- key
- value
-
-

4.2.4

Put Get Delete Iterator

CHAPTER 5

sstable

5.1

leveldb LSM (Log Structured-Merge Tree) leveldb memtable leveldb checkpoint memtable memtable immutable memory db memtable immutable memory db minor compaction

: minor compaction

leveldb LSM Minor Compaction

1.
2.

memory db leveldb sstable sstable

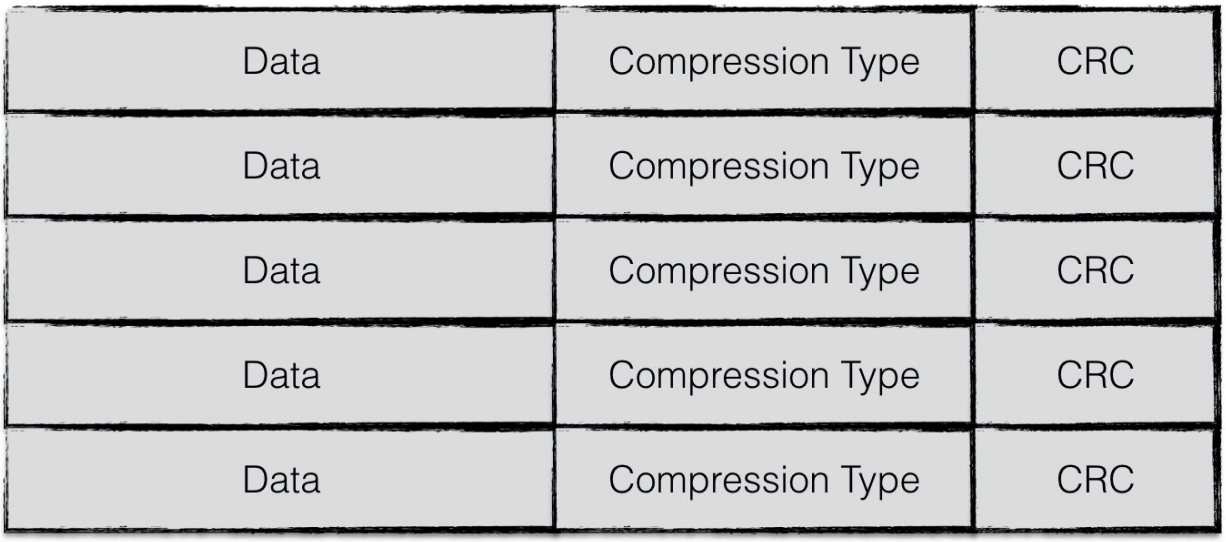
5.2 SStable

5.2.1

sstable 4KiB Block

1.
2. CRC

Block leveldb Snappy CRC



5.2.2

leveldb sstable

1. **data block**: key value
2. **filter block**: leveldb leveldb block
3. **meta Index block**: filter block sstable
4. **index block** index block data block
5. **footer**: meta index block index block

1-4 1.1 CRC

5.3 data block

data block leveldb keyvalue data block CRC

keyvalue sstable keyvalue leveldb keyvalue key **key** key

keyvalue key 16 key Restart point

: leveldb Restart point sstable

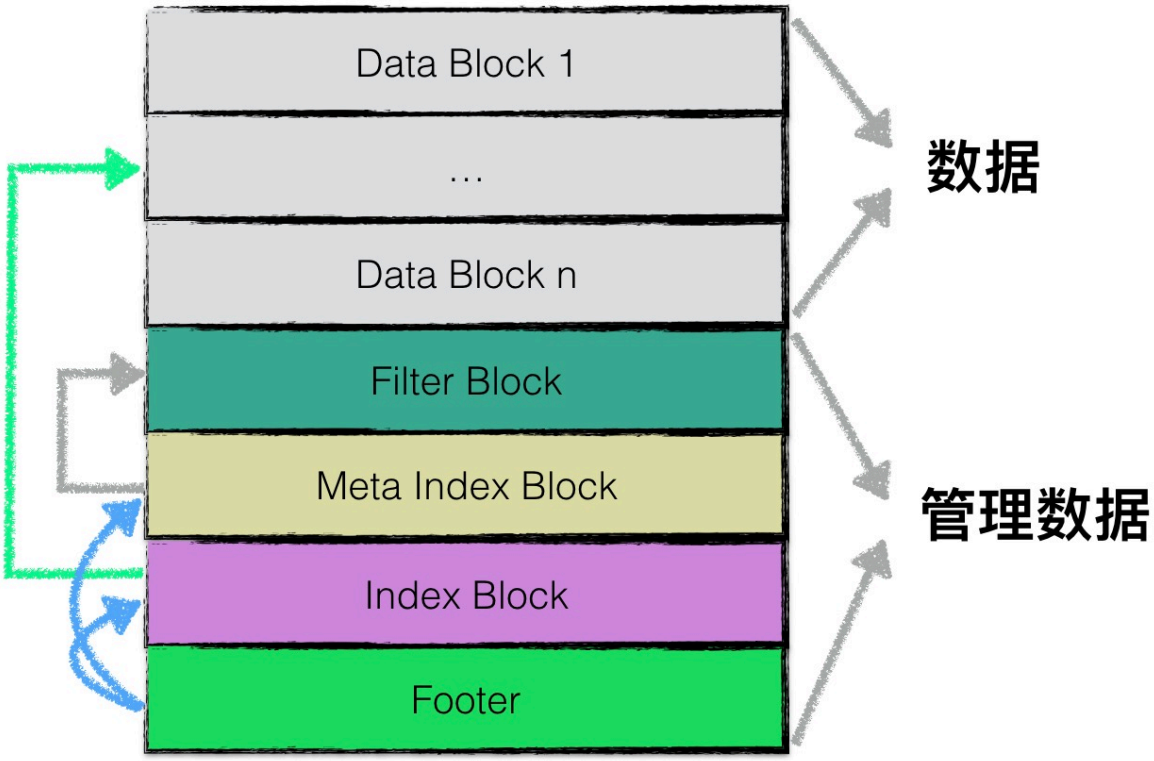
Restart point key sstable restart point

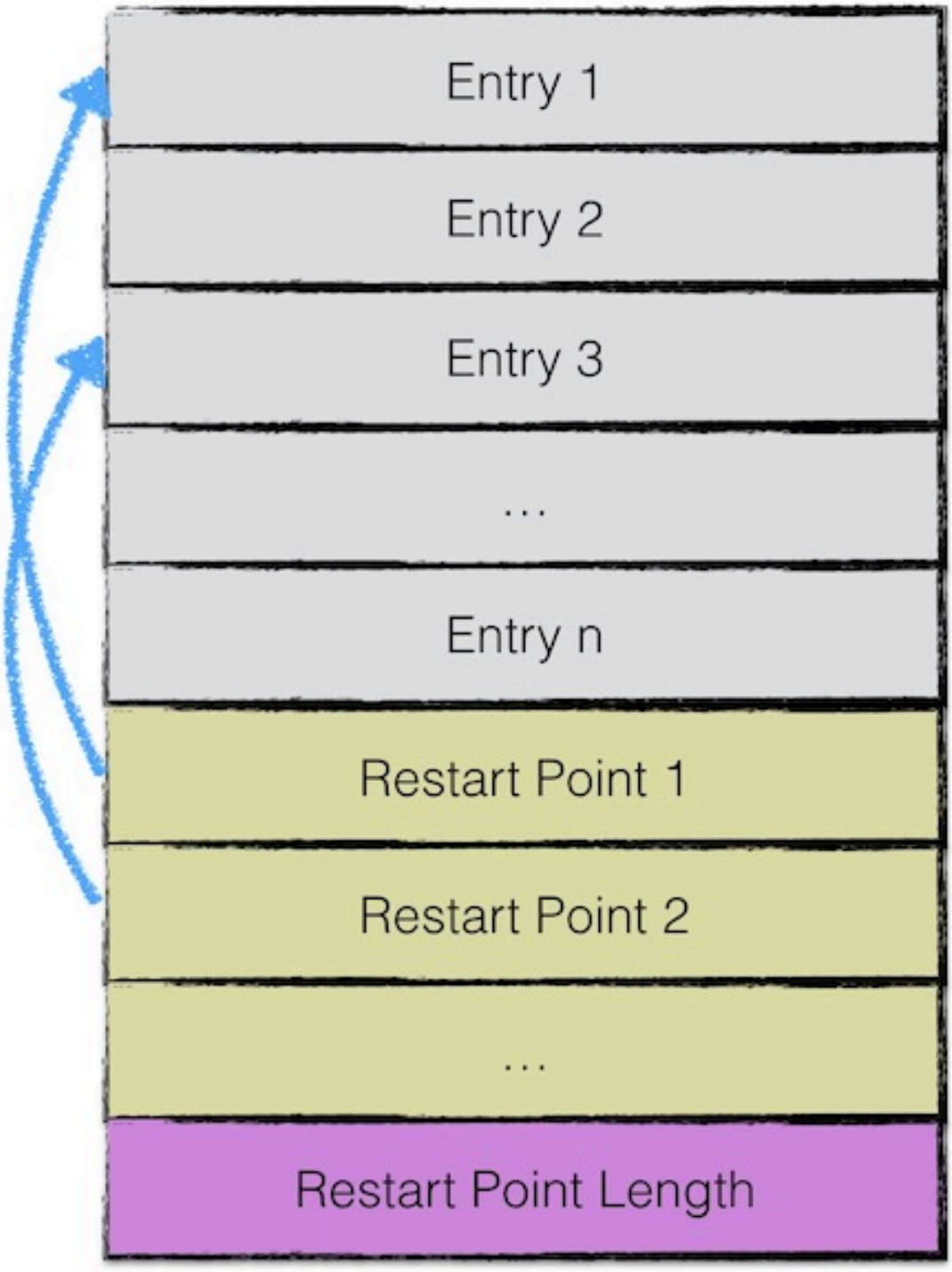
key

---

entry 5

1. key

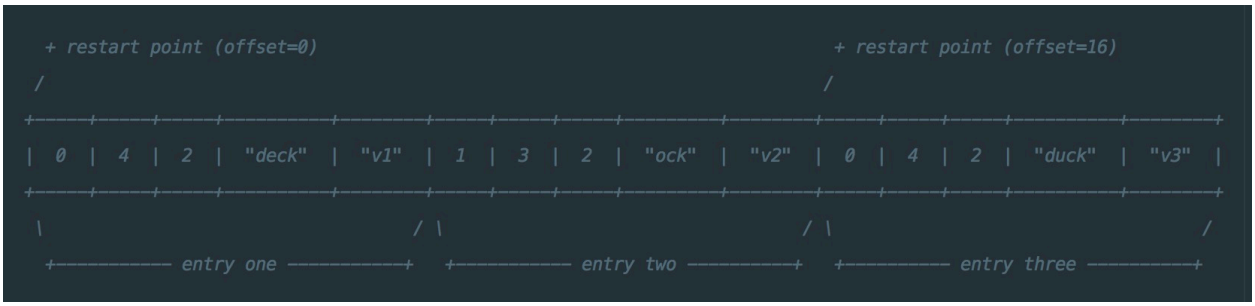

Data Block 1
...
Data Block n
Filter Block
Meta Index Block
Index Block
Footer
数据
管理数据



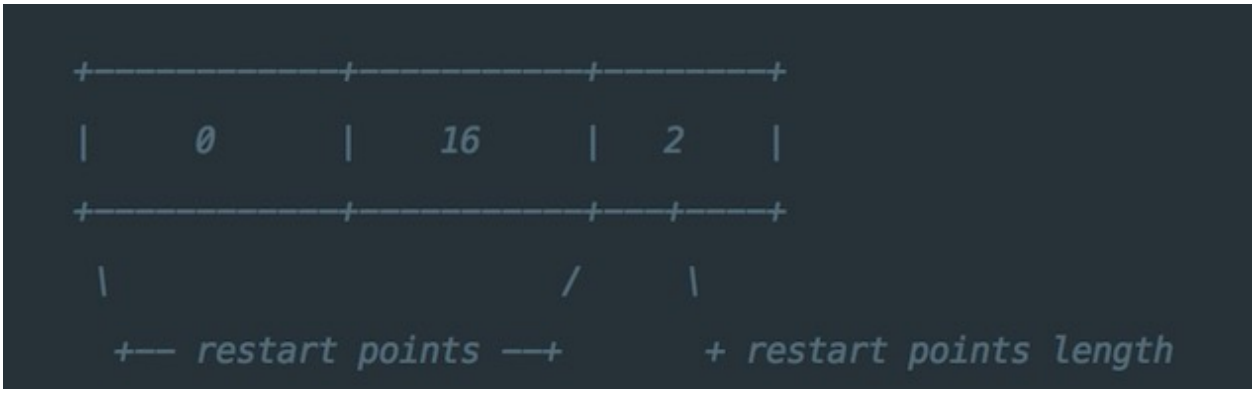
Shared key length	Unshared key length	Value length	Unshared key content	Value

2. key
3. value
4. key
5. value

```
restart_interval=2
entry one  : key=deck,value=v1
entry two  : key=dock,value=v2
entry three: key=duck,value=v3
```



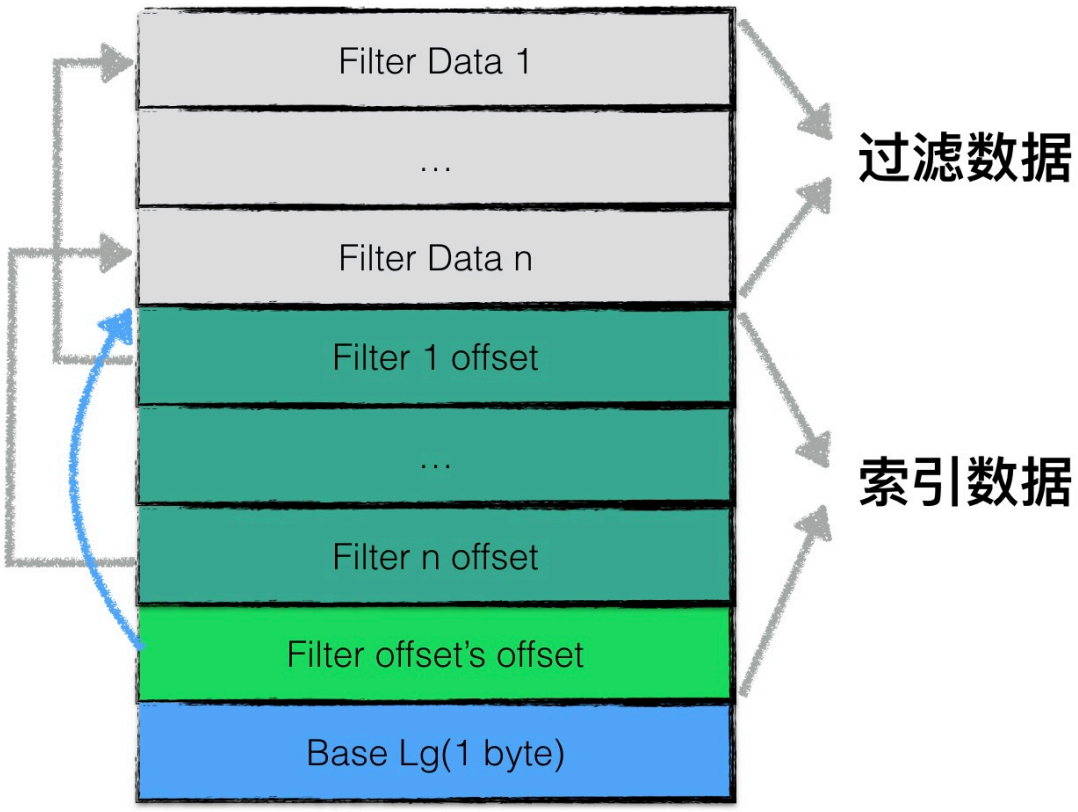
entry restart_interval 2 entry restart point key entry3 key
restart point 0 restart point 16 restart point datablock



restart point restart point

5.4 filter block

data block filter block
sstable datablock leveldb filter block datablock datablock
filter block data block Leveldb -
filter block 1 2
`filter i offset` i filter data filter block `filter offset's offset` filter block filter block-


Filter Data 1
...
Filter Data n
Filter 1 offset
...
Filter n offset
Filter offset's offset
Base Lg(1 byte)
过滤数据
索引数据

filter block `filter offset's offset` `filter i offset` offset `filter data`

Base Lg 11 2KB

sstable filter block block filter . filter_data_k [base*k, base*(k+1)] block key filter block block key block key

: leveldb sstable

BloomFilter

5.5 meta index block

meta index block filter block sstable

meta index block

key "filter."

value filter block sstable 1 sstable 2

5.6 index block

meta index block index block data block

indexblock data block

1. data block i key
2. data block sstable
3. data block

: data block i key index block key

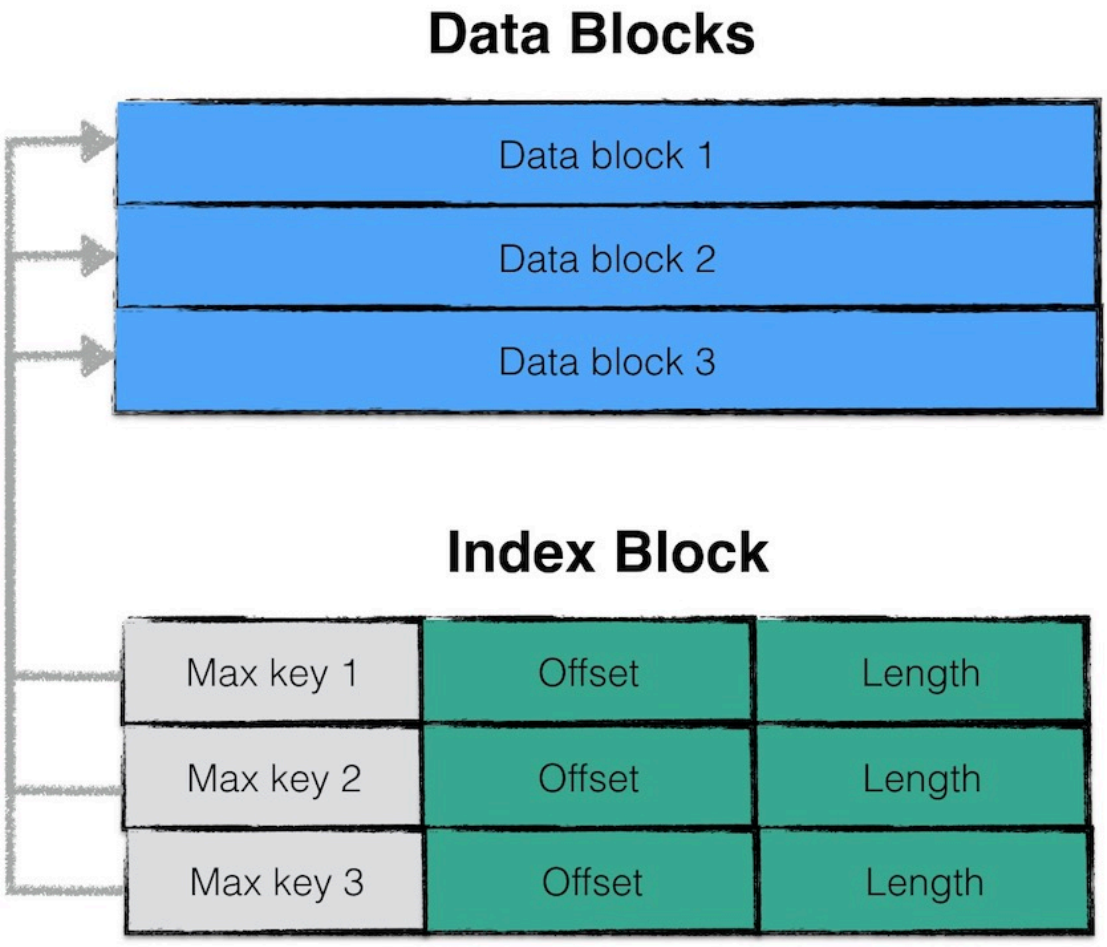
index block key data block

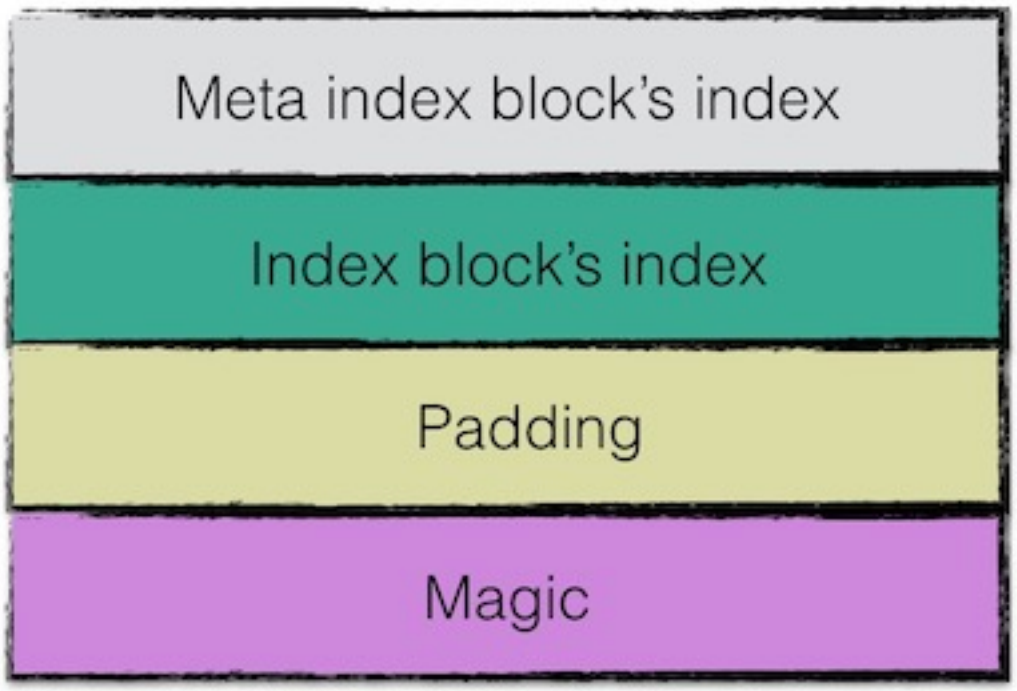
5.7 footer

footer 48 meta index block index block sstable magic word "http://code.google.com/p/leveldb/" sha1 8

5.8

sstable


Data Blocks
Data block 1
Data block 2
Data block 3
Index Block
Max key 1
Offset
Length
Max key 2
Offset
Length
Max key 3
Offset
Length



5.8.1

sstable

- memory db sstable
- leveldb compaction sstable sstable

sstable tWriter

```
// tWriter wraps the table writer. It keep track of file descriptor
// and added key range.
type tWriter struct {
    t *tOps

    fd storage.FileDesc //
    w  storage.Writer   //   writer
    tw *table.Writer

    first, last []byte
}
```

sstable writer sstable key tableWriter

sstable tableWriter `Append` sstable

sstable

: sstable 1 2 3 key 4 key

tableWriter `Append`

tableWriter

append tableWriter

```
// Writer is a table writer.
type Writer struct {
    writer io.Writer
    // Options
    blockSize   int //   4KiB

    dataBlock   blockWriter // data Writer
    indexBlock  blockWriter // indexBlock Writer
    filterBlock filterWriter // filter Writer
    pendingBH   blockHandle
    offset      uint64
    nEntries    int // key-value
}
```

blockWriter filterWriter writer blockWriter data filterWriter pendingBH dataBlock dataBlock indexBlock

Append

append

1. dataBlock dataBlock
2. keyvalue datablock;
3. filterBlock
4. datablock datablock

```
func (w *Writer) Append(key, value []byte) error {
    w.flushPendingBH(key)
    // Append key/value pair to the data block.
    w.dataBlock.append(key, value)
    // Add key to the filter block.
    w.filterBlock.add(key)

    // Finish the data block if block size target reached.
    if w.dataBlock.bytesLen() >= w.blockSize {
        if err := w.finishBlock(); err != nil {
            w.err = err
            return w.err
        }
    }
    w.nEntries++
    return nil
}
```

dataBlock.append

kv dataBlock buffer restart restart point

filterBlock.append

kv key Leveldb -

finishBlock

datablock

1. dataBlock restart point
2. dataBlock snappy dataBlock
3. checksum
4. dataBlock offset length
5. datablock buffer
6. filterBlock buffer

Close

tableWriter Close

1. buffer datablock
2. filterBlock
3. filterBlock metaIndexBlock
4. indexBlock
5. footer

sstable **sstable** **memory db** **Compaction** sstable **leveldb**

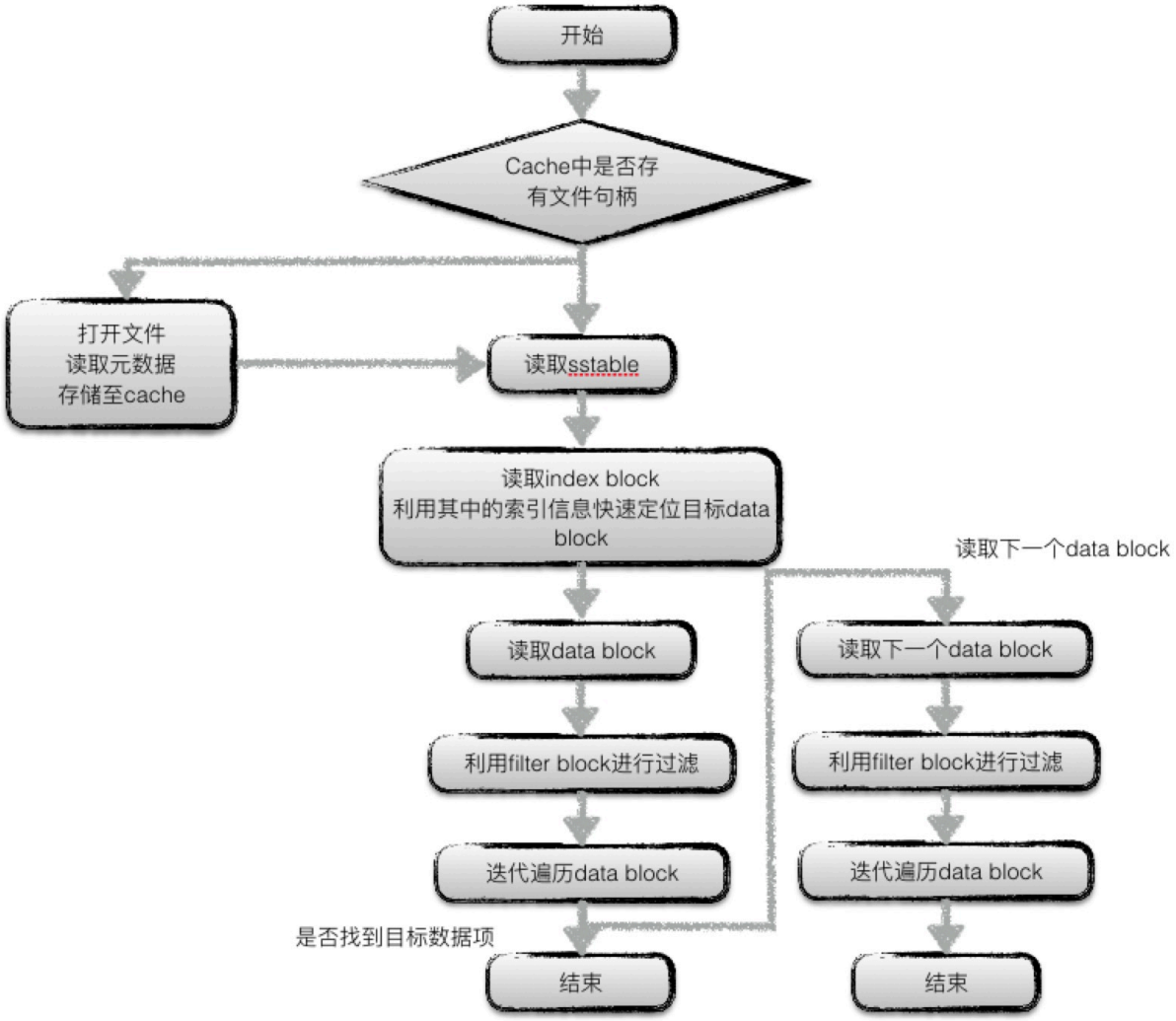
5.9

sstable

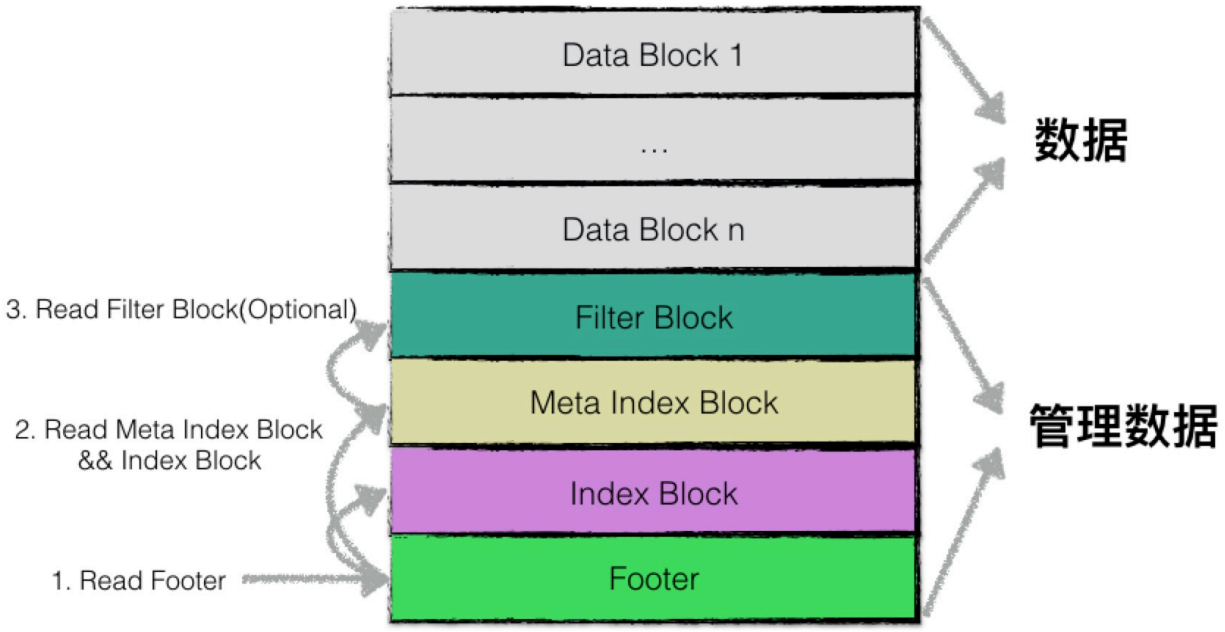
1. “ ”cache sstable cache sstable cache
2. sstable index block data block
3. index block data block
4. filter block data block data block data block
5. data block data block 4
6. data block `Not Found`

leveldb cache

- sstable
- data block


开始
Cache中是否存
有文件句柄
打开文件
读取元数据
存储至cache
读取sstable
读取index block
利用其中的索引信息快速定位目标data
block
读取下一个data block
读取data block
读取下一个data block
利用filter block进行过滤
利用filter block进行过滤
迭代遍历data block
迭代遍历data block
是否找到目标数据项
结束
结束

cache sstable



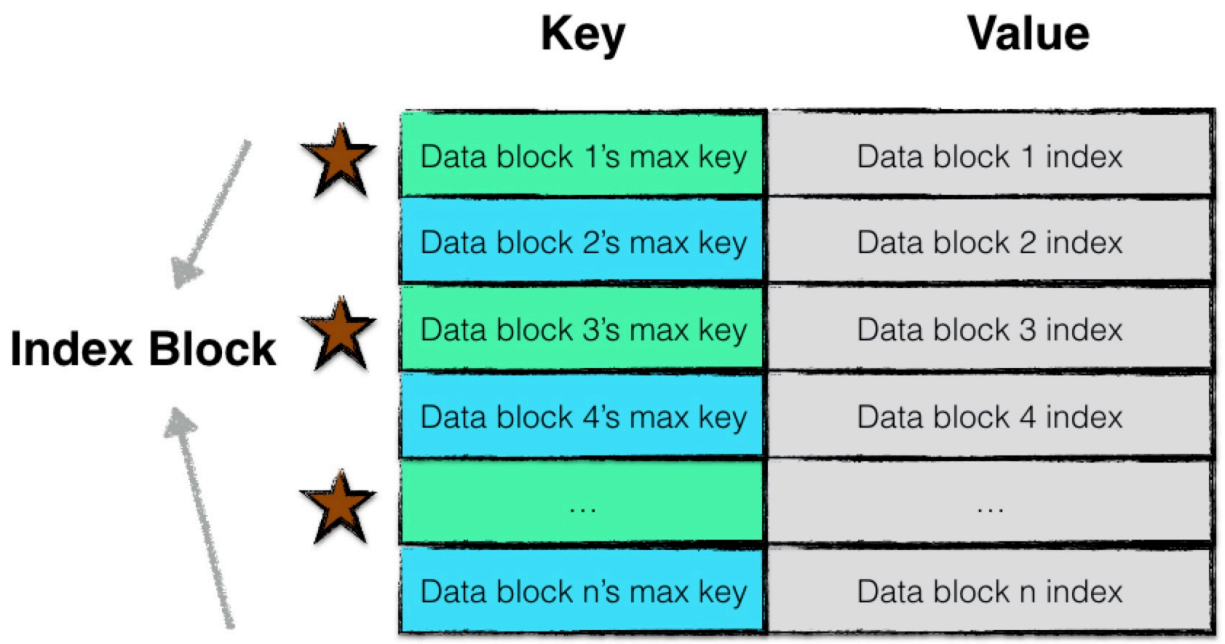
sstable sstable

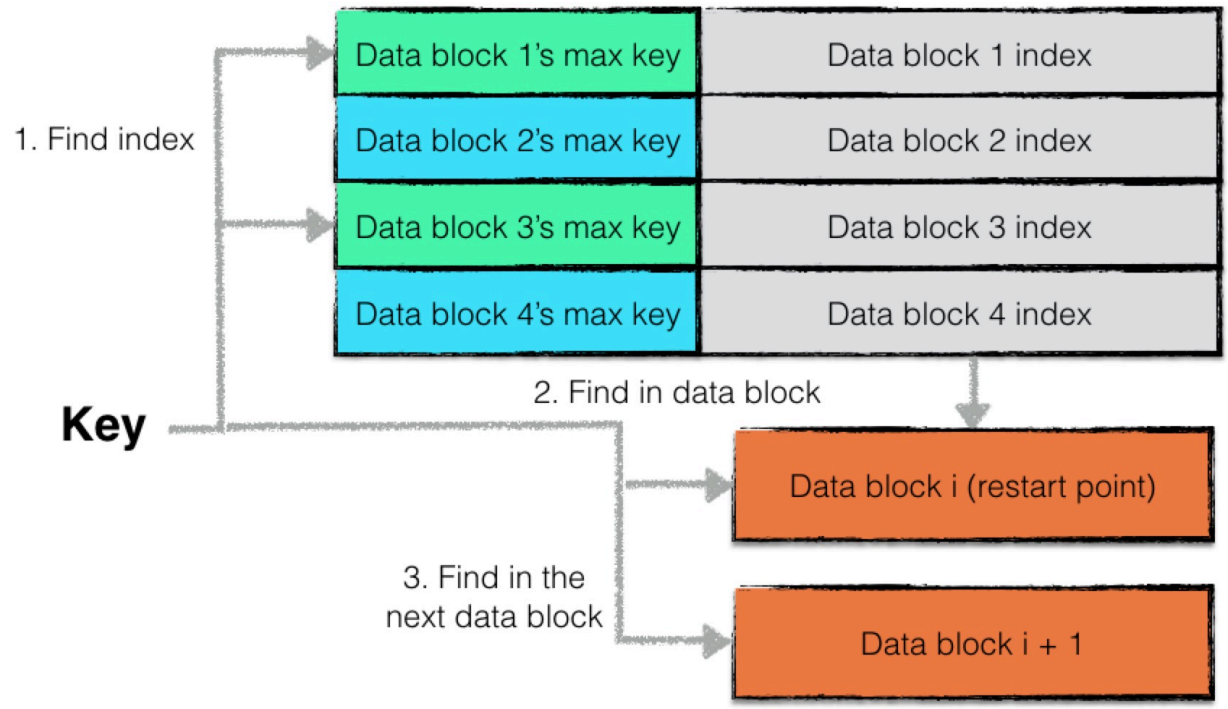
1. 48 **Footer**
2. Footer (1) Meta Index Block(2) Index Block
3. meta index block
4. meta index block “ ” filter block sstable

sstable data block “ ” sstable index block data block
index block
index block data block
key data block key value data block offset, length
index block key data block key data block data block-

: data block index block key key key key data block-
16 index block 2
index block “ “ key key key
” “data block

data block


Key
Value
Index Block
Data block 1's max key
Data block 1 index
Data block 2's max key
Data block 2 index
Data block 3's max key
Data block 3 index
Data block 4's max key
Data block 4 index
...
...
Data block n's max key
Data block n index

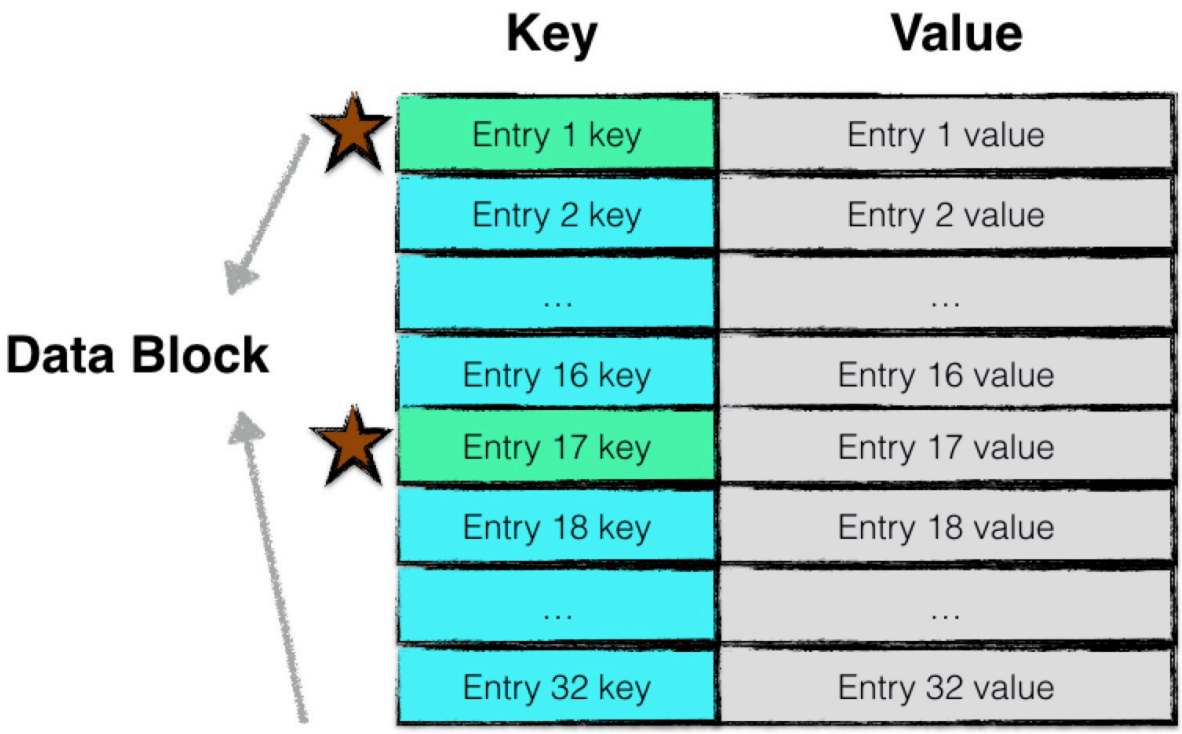

1. Find index
Data block 1's max key
Data block 1 index
Data block 2's max key
Data block 2 index
Data block 3's max key
Data block 3 index
Data block 4's max key
Data block 4 index
2. Find in data block
Key
Data block i (restart point)
3. Find in the
next data block
Data block i + 1

sstable data block data block “ ” data block

- data block data block
- data block data block

data block

data block



data block data block “ ”

index block data block 16 entry 1 key key entry 17

data block

sstable restart point index block filter block max key

5.10

5.10.1

sstable compaction

5.10.2

sstable

-
-

sstable

5.10.3

sstable

leveldb 0

5.10.4 Cache

sstable cache sstable

CHAPTER 6

leveldb IO

Leveldb LRUCache

- sstable
- sstable dataBlock

cache IO leveldb cache

6.1 Cache

leveldb cache LRUcache

- Hash table
- LRU

Hash table Yujie Liu Dynamic-Sized Nonblocking Hash Table hash O(1)

hash hash resize hash bucket

hash table resize

LRU Least Recently Used cache LRU cache

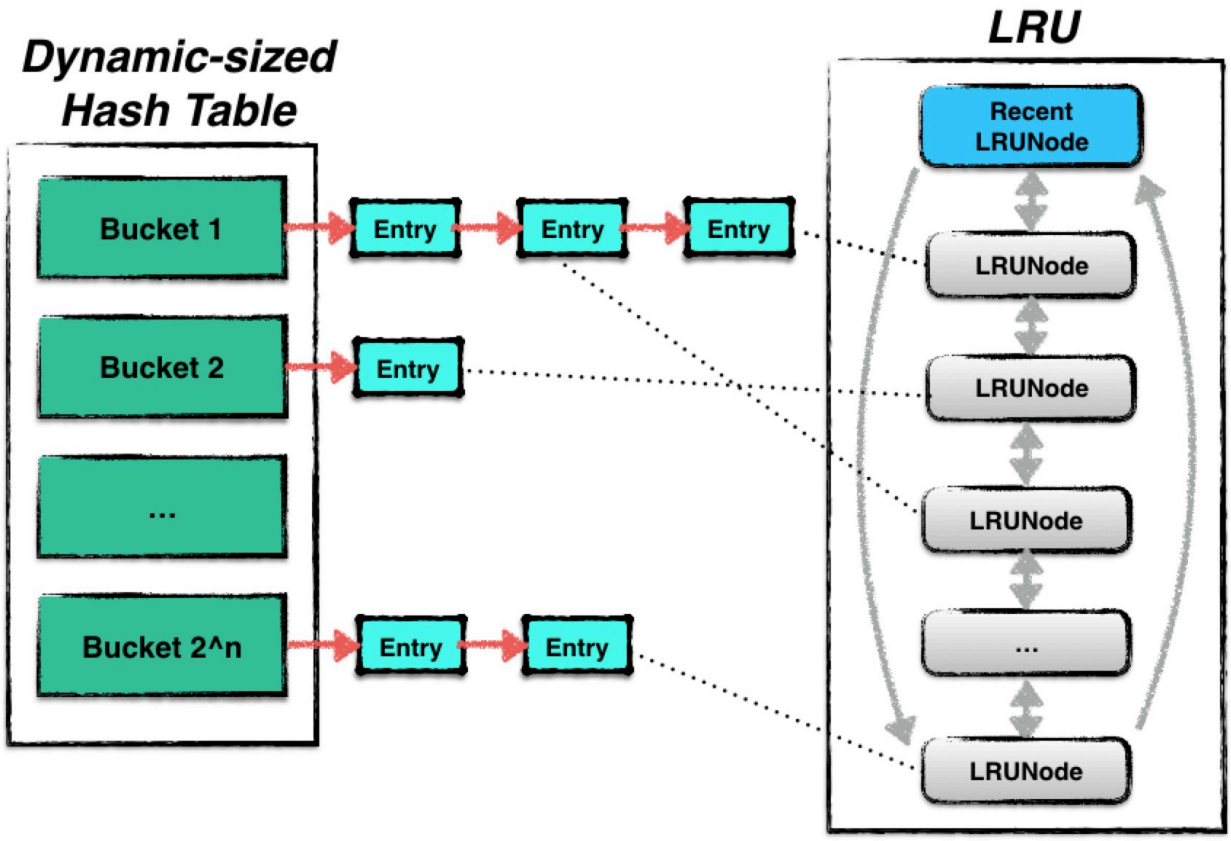
6.2 Dynamic-sized NonBlocking Hash table

hash resize Lock-Free

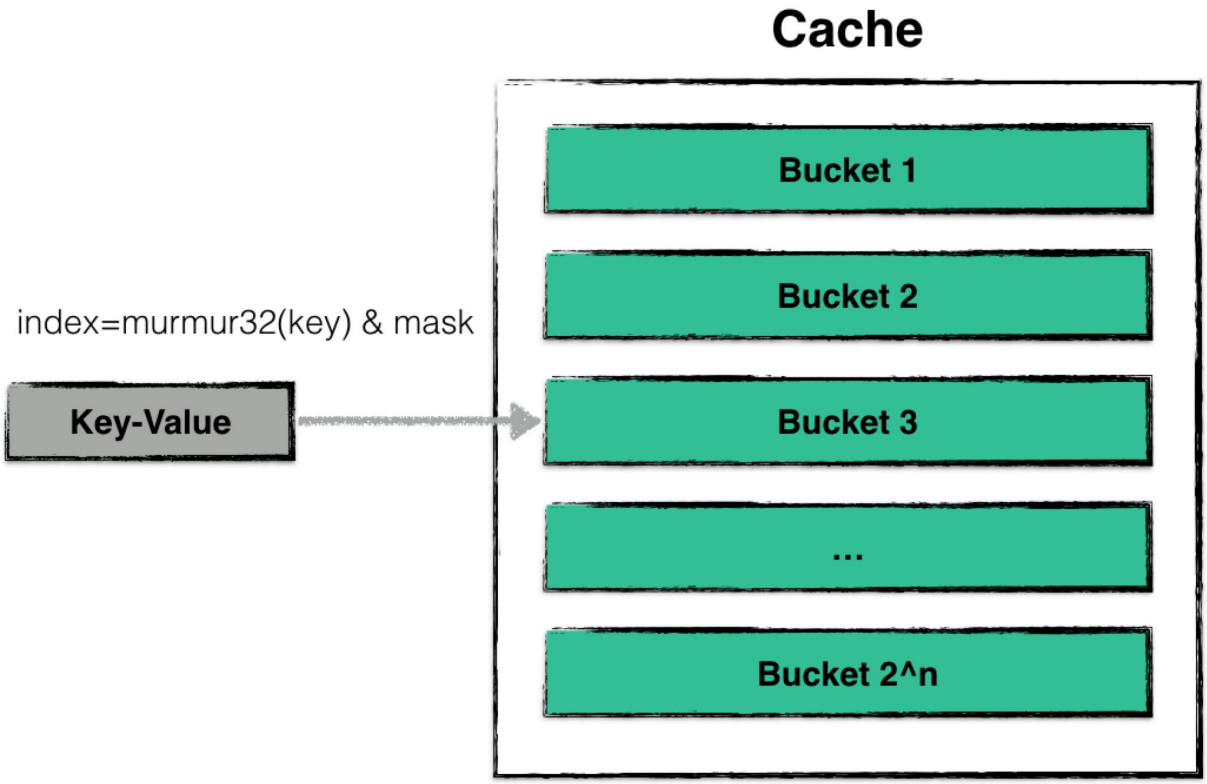
hash bucket bucket hash resize “ ” “ ”

liu **bucket**

hash resize bucket


Dynamic-sized
Hash Table
Bucket 1
Bucket 2
...
Bucket 2^n
Entry
Entry
Entry
Entry
Entry
LRU
Recent
LRUNode
LRUNode
LRUNode
LRUNode
...
LRUNode

6.2.1



32

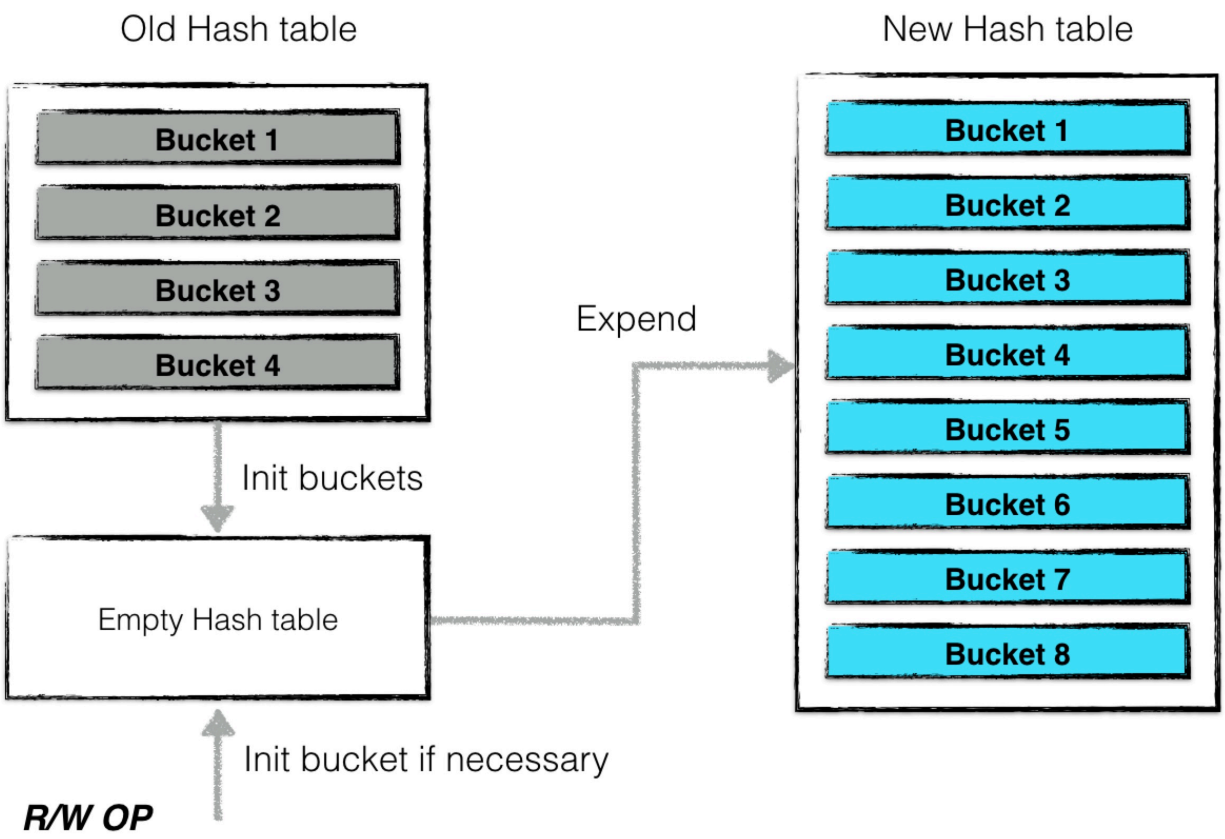
6.2.2

cache “cache ”

- cache node 16 32
- cache 32 128

1.
2. “ ” “ ”
3. “ ” “ ”

“ ”


Old Hash table
New Hash table
Bucket 1
Bucket 2
Bucket 3
Bucket 4
Expend
Bucket 1
Bucket 2
Bucket 3
Bucket 4
Bucket 5
Bucket 6
Bucket 7
Bucket 8
Init buckets
Empty Hash table
Init bucket if necessary
R/W OP

1. key
2. ID

key

6.2.3

2

6.3 LRU

leveldb LRU

Leveldb LRU `LRUNode`

```
type lruNode struct {
    n   *Node // customized node
    h   *Handle
    ban bool

    next, prev *lruNode
}
```

`LRUNode` LRU “ ”

LRU

- Promote

hash `LRUNode` `LRUNode`

hash `LRUNode` `LRUNode`

LRU

- Ban

hash `LRUNode` “ ”

0

6.4

leveldb cache

- cache sstable 500
- bcache sstable dataBlock 8MB ;

sstable cache 1 indexBlock 2 metaIndexBlock

6.5

- ”Dynamic-Sized Nonblocking Hash Tables”, by Yujie Liu, Kunlong Zhang, and Michael Spear. ACM Symposium on Principles of Distributed Computing, Jul 2014.

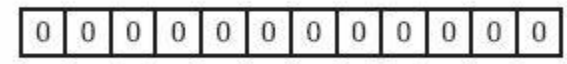
CHAPTER 7

Bloom Filter Bloom Filter false positive Bloom Filter “ ” Bloom Filter

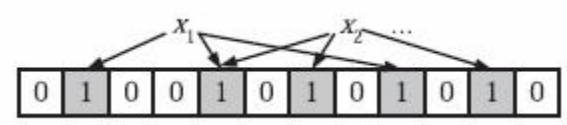
leveldb key sstable key

7.1

bloom 0



x k 3 x bloom 1



k 1 “ ” 0 y1 y2

7.2

http://blog.csdn.net/jiaomeng/article/details/1495500

- k
- m;
- n;

1. k = ln2 * (m/n)
2. m 1.44

7.3

leveldb goleveldb filter/bloom.go

bloom int

```
type bloomFilter int
```

key 2 m/n 1.44 10

```
func NewBloomFilter(bitsPerKey int) Filter {
    return bloomFilter(bitsPerKey)
}
```

generator, k k = f * ln2 f = m/n 1

generator key generate key

```
func (f bloomFilter) NewGenerator() FilterGenerator {
    // Round down to reduce probing cost a little bit.
    k := uint8(f * 69 / 100) // 0.69 =~ ln(2)
    if k < 1 {
        k = 1
    } else if k > 30 {
        k = 30
    }
    return &bloomFilterGenerator{
        n: int(f),
        k: k,
    }
}
```

generator

1. Add
2. Generate

Add key

```
func (g *bloomFilterGenerator) Add(key []byte) {
    // Use double-hashing to generate a sequence of hash values.
    // See analysis in [Kirsch,Mitzenmacher 2006].
    g.keyHashes = append(g.keyHashes, bloomHash(key))
}
```

Generate key key

key key

k

```
func (g *bloomFilterGenerator) Generate(b Buffer) {
    // Compute bloom filter size (in both bits and bytes)
    // len(g.keyHashes)    n  g.n   m/n
    // nBits   m
    nBits := uint32(len(g.keyHashes) * g.n)
    // For small n, we can see a very high false positive rate.  Fix it
    // by enforcing a minimum bloom filter length.
    if nBits < 64 {
        nBits = 64
    }
    nBytes := (nBits + 7) / 8
    nBits = nBytes * 8

    dest := b.Alloc(int(nBytes) + 1)
    dest[nBytes] = g.k

    for _, kh := range g.keyHashes {
        // Double Hashing
        delta := (kh >> 17) | (kh << 15) // Rotate right 17 bits
        for j := uint8(0); j < g.k; j++ {
            bitpos := kh % nBits
            dest[bitpos/8] |= (1 << (bitpos % 8))
            kh += delta
        }
    }

    g.keyHashes = g.keyHashes[:0]
}
```

Contain key

```
func (f bloomFilter) Contains(filter, key []byte) bool {
    nBytes := len(filter) - 1
    if nBytes < 1 {
        return false
    }
    nBits := uint32(nBytes * 8)

    // Use the encoded k so that we can read filters generated by
    // bloom filters created using different parameters.
    k := filter[nBytes]
    if k > 30 {
```

( )

( )

```
        // Reserved for potentially new encodings for short bloom filters.
        // Consider it a match.
        return true
    }

    kh := bloomHash(key)
    delta := (kh >> 17) | (kh << 15) // Rotate right 17 bits
    for j := uint8(0); j < k; j++ {
        bitpos := kh % nBits
        if (uint32(filter[bitpos/8]) & (1 << (bitpos % 8))) == 0 {
            return false
        }
        kh += delta
    }
    return true
}
```

7.4

- http://blog.csdn.net/jiaomeng/article/details/1495500
- https://en.wikipedia.org/wiki/Double_hashing

CHAPTER 8

compaction

Compaction leveldb leveldb leveldb compaction compaction

8.1 Compaction

8.1.1

leveldb LSM leveldb ***Minor Compaction***

minor compaction 0 sstable 0 overlap

8.1.2

leveldb O(log n)

leveldb O(log n) 0 0 overlap

leveldb 0 leveldb ***Major Compaction*** 0 1

leveldb compaction

8.1.3

minor compaction major compaction major compaction

major compaction 0 leveldb

- 0 `SlowdownTrigger`
- 0 `PauseTrigger` Major Compaction

compaction

8.1.4

leveldb key leveldb

leveldb major compaction

8.2 Compaction

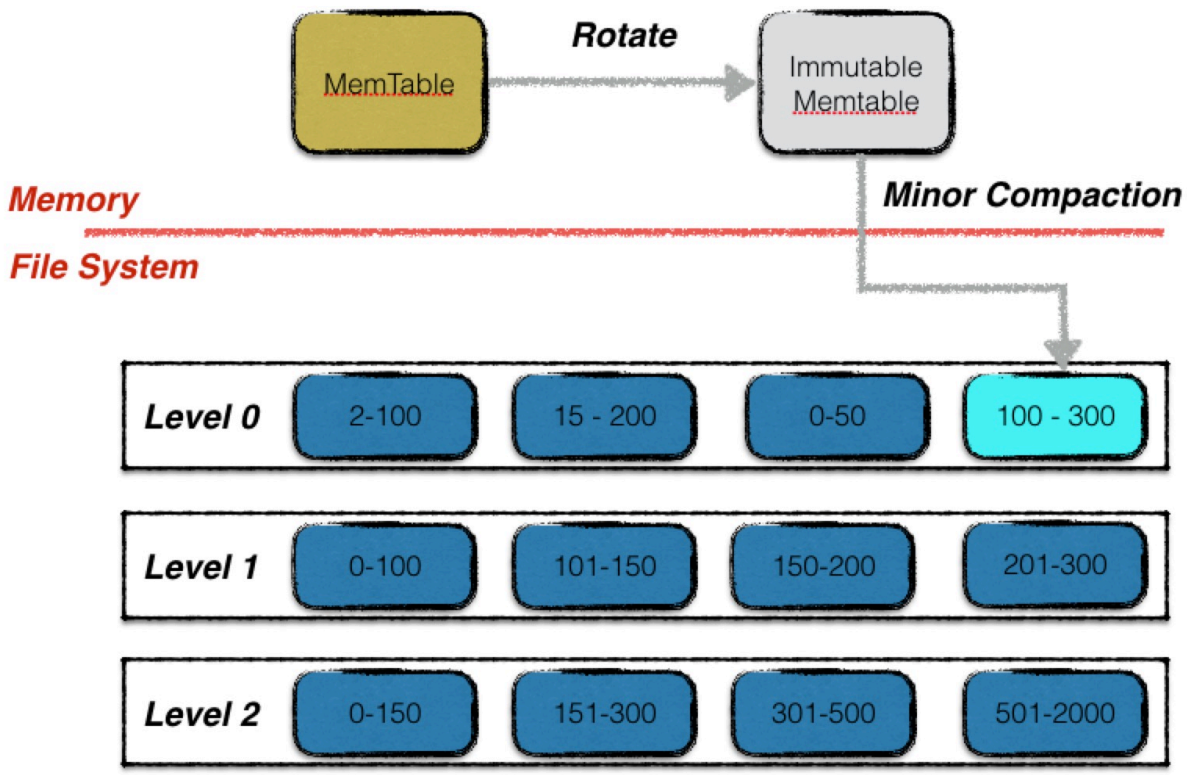
compaction

- minor compaction
- major compaction

compaction

8.2.1 Minor Compaction

minor compaction

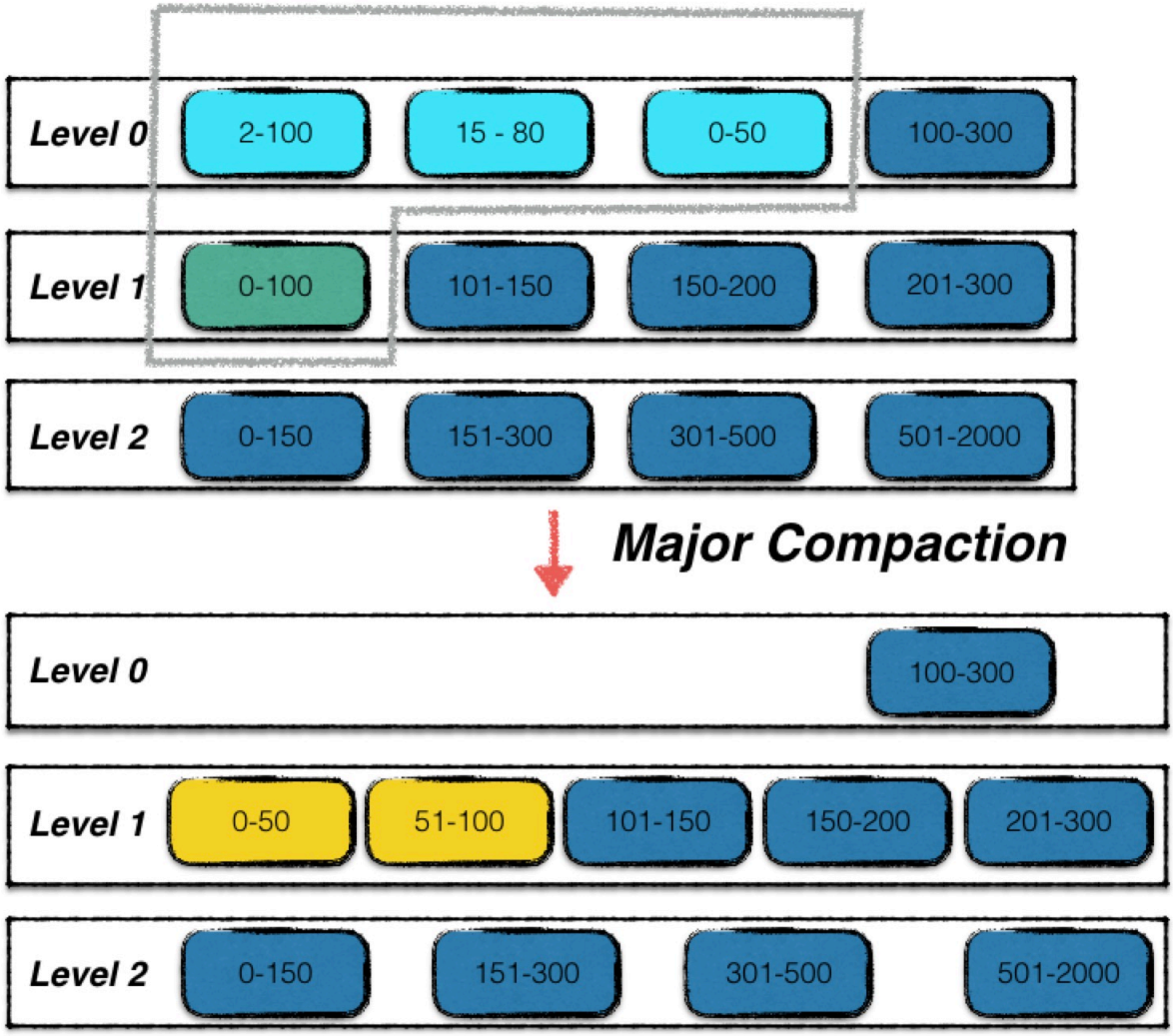


minor compaction sstable **Leveldb**

minor compaction **minor compaction** **major compaction** minor compaction major compaction major compaction

8.2.2 Major Compaction

minor compaction major compaction major compaction



0 sstable 1 sstable 1 sstable

leveldb major compaction

- 0 4
- level i (10 ^ i) MB
-

0

compaction leveldb 0 major compaction

0

level i i >= 1 sstable compaction **compaction IO**

leveldb 1 **0 key 0 1 1** 0 compaction IO
leveldb 1 10MB 2 100MB 7

0 1
0-n

- source
- source source+1
- source source+2 10

souce source+1
“ ” leveldb ” “
1 2
1 Leveldb 10ms,
2 1MB 25ms 1MB 25
1MB 1 source 1MB 2 source+1 10-12MB 3 source+1 10-12MB
25MB IO 100MB s IO 25ms

sstable `seekLeft` 16KB
leveldb 1 Get 2

sstable `seekLeft`
`seekLeft` 0
compaction

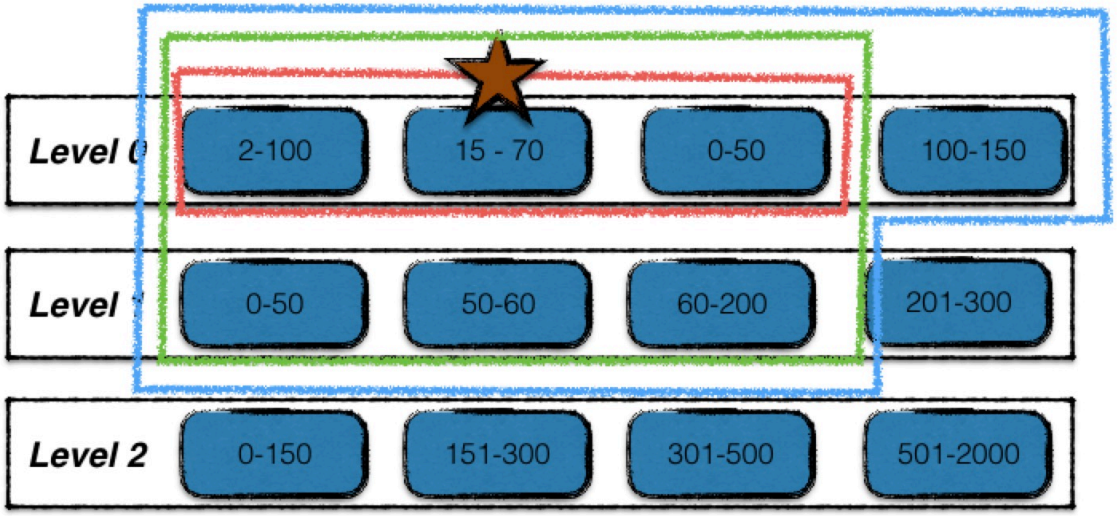
8.2.3

compaction

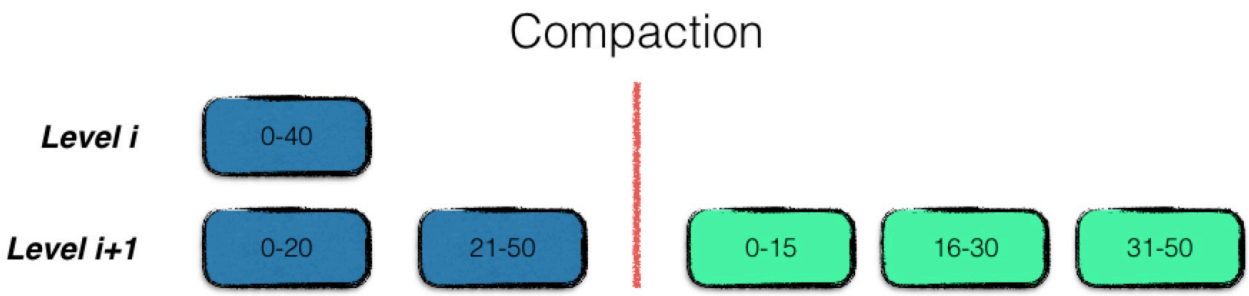
1.
2. key
3.
4.

level 0 *level i* key key

1.
2. level i key level i
3. level i level i+1 key level i i+1
4. level i+1 level i key



level i level i+1 level i+1



compaction source source+1 leveldb

- 0 4
- 0

1

8.3

leveldb compaction trivial move leveldb IO

CHAPTER 9

Leveldb sstable sstable

sstable leveldb

leveldb

9.1 Manifest

manifest leveldb

Manifest Session Record Session Record

:

1 sstable

2 sstable compaction

3 journal

Manifest leveldb

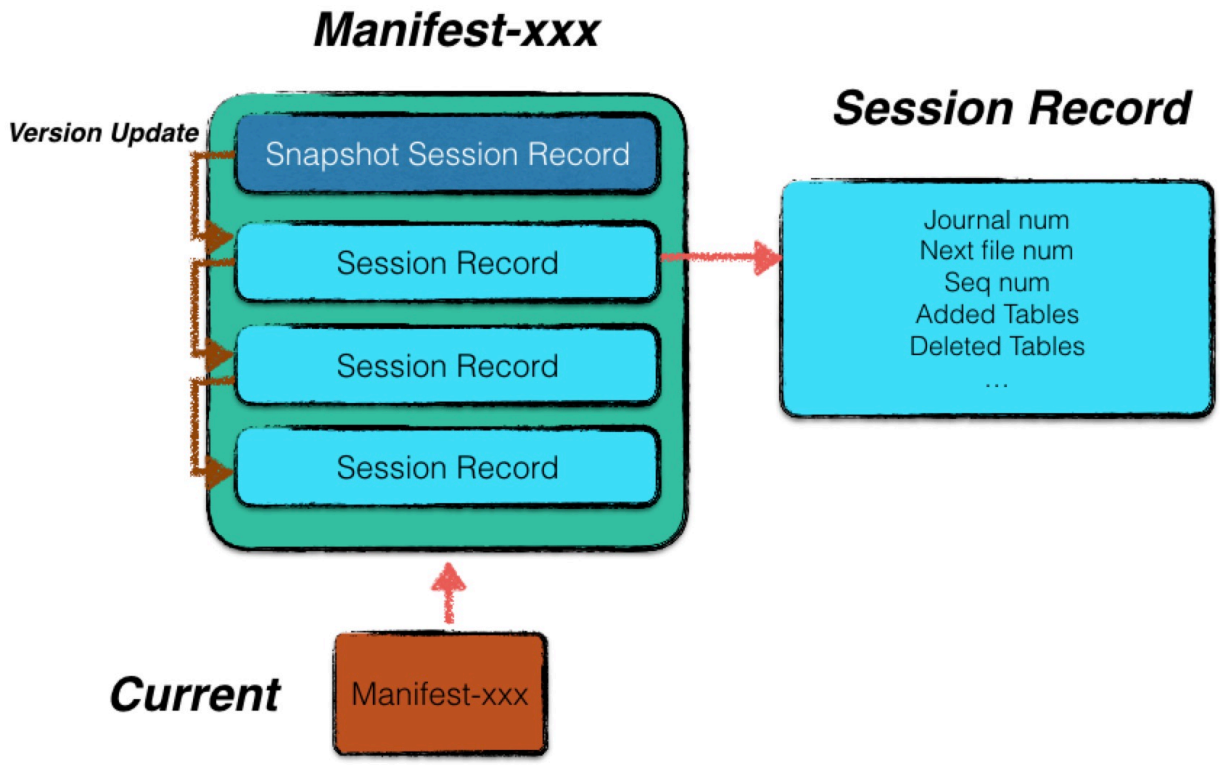
Manifest

Manifest Session Record **Session Record** leveldb Session Record

manifest Session Record

Session Record

- Comparer
- journal
-

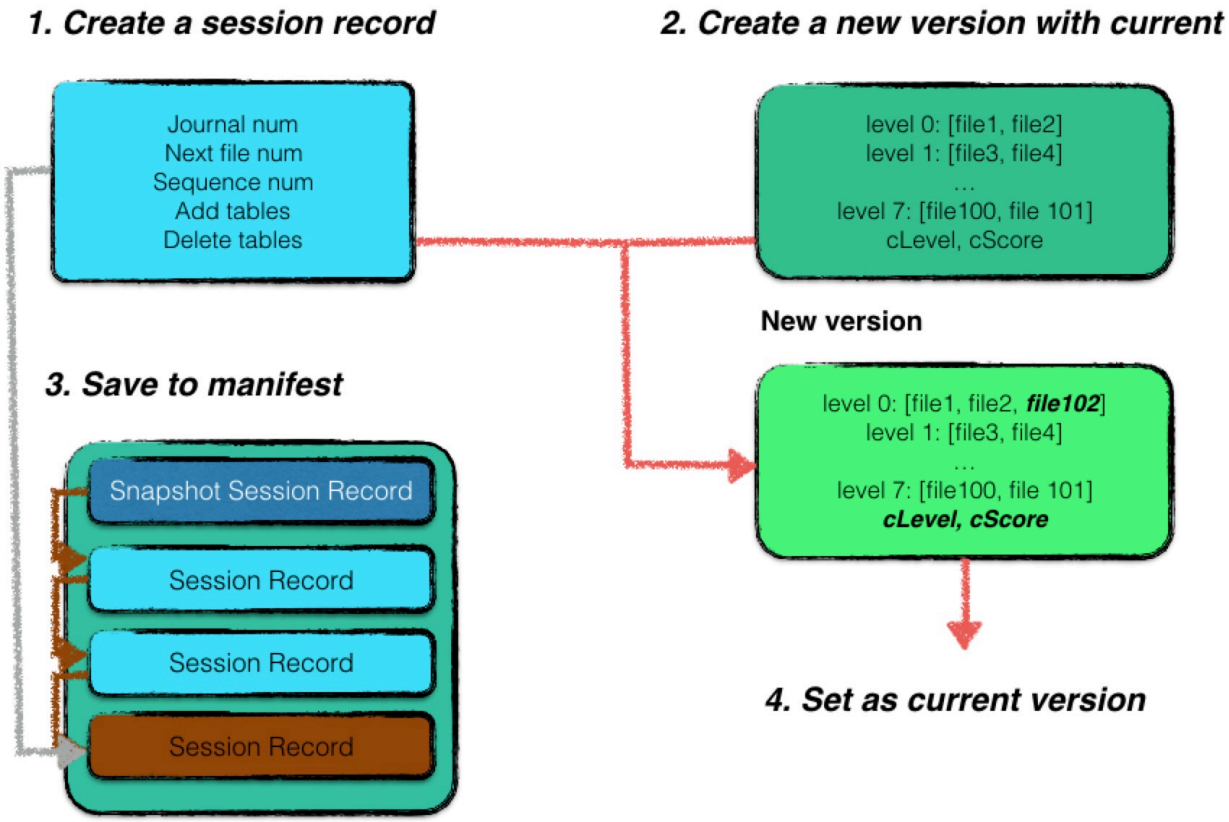


- sequence number
-
-
- compaction

9.2 Commit

1 major compaction 2 minor compaction 0 leveldb

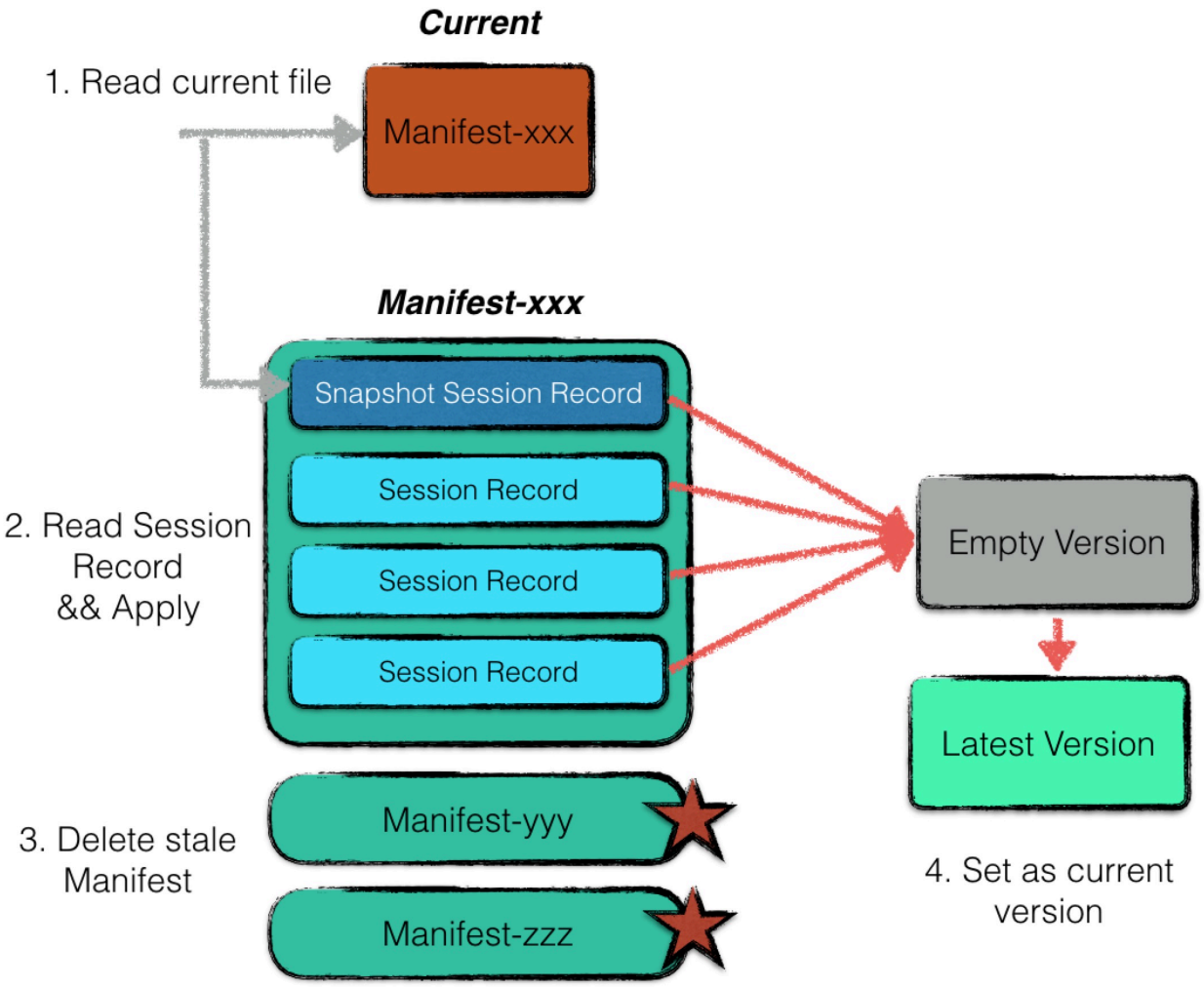
1. session record
2. minor compaction replay sstable session record journal sequence number
3. major compaction session record
4. session record 1 2
5. session record
6. session record manifest session record manifest current manifest
7. manifest session record
8. version version


1. Create a session record
Journal num
Next file num
Sequence num
Add tables
Delete tables
2. Create a new version with current
level 0: [file1, file2]
level 1: [file3, file4]
…
level 7: [file100, file 101]
cLevel, cScore
New version
level 0: [file1, file2, file102]
level 1: [file3, file4]
…
level 7: [file100, file 101]
cLevel, cScore
3. Save to manifest
Snapshot Session Record
Session Record
Session Record
Session Record
4. Set as current version

:	leveldb	sstable					
		manifest	session record			sstable	compaction
	leveldb	version	version				

9.3 Recover

recover Manifest version



1. Current manifest
2. version manifest session record apply version manifest session record version session record
3. current manifest
4. version version

: leveldb manifest session record leveldb manifest session record version manifest recover leveldb manifest manifest

9.4 Current

Manifest leveldb manifest current manifest manifest

9.5

manifest leveldb

leveldb manifest leveldb `Repairer`

1. sstable key sequence number
2. sstable 0 0 key
3. manifest 0 4 compaction

9.6

leveldb MVCC

sstable major compaction sstable leveldb leveldb

1. sstable compaction sstable sstable
2. sstable compaction sstable
3. compaction
4. compaction sstable 0 0