EOS的ChainBase实现

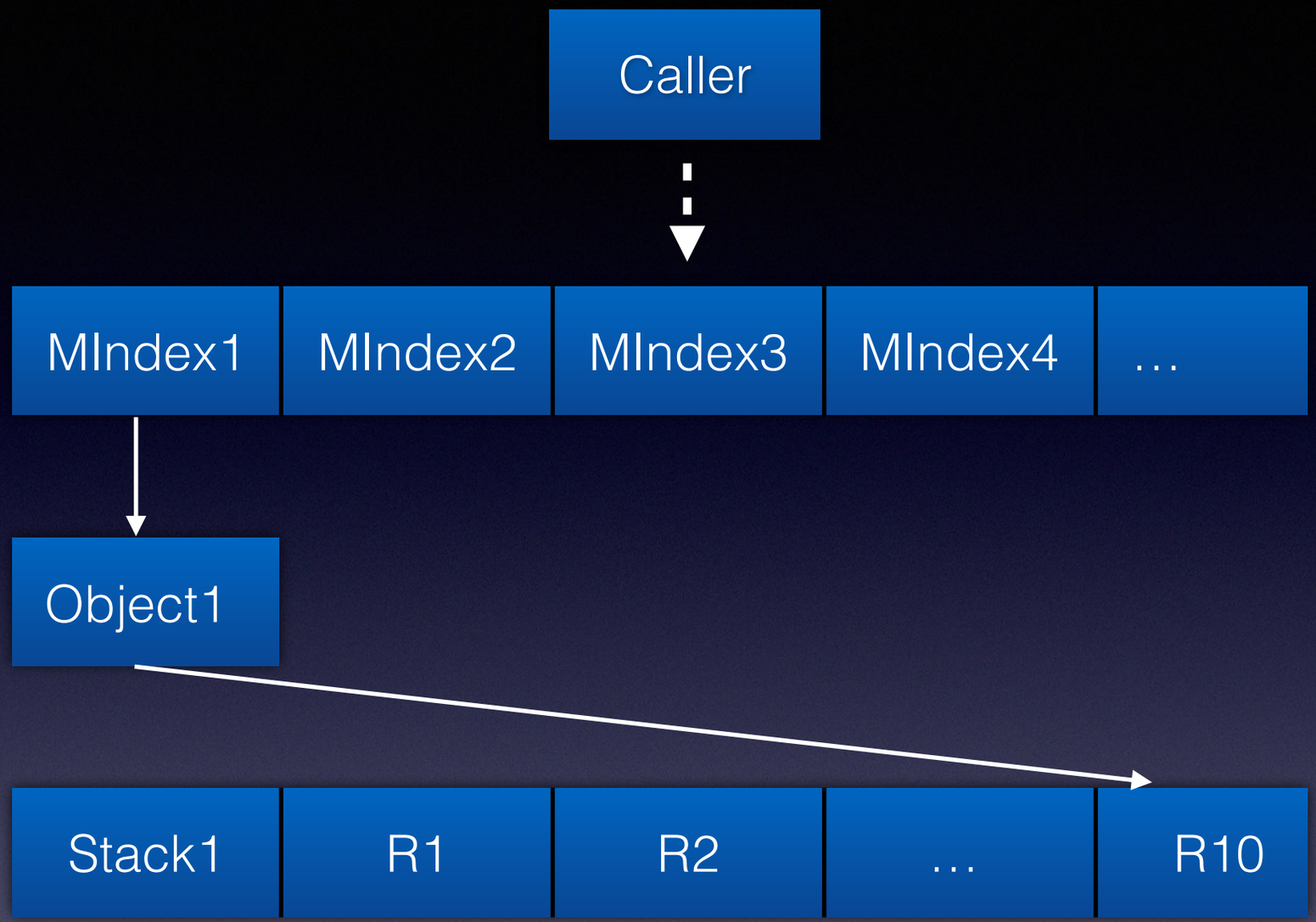
陈渊

来源

- https://github.com/steemit/steem/tree/master/libraries/chainbase
- https://github.com/eosio/chainbase
- ByteMaster手写的一个数据库

特性

- 内存映射(mmap)
- 多索引结构(boost::multi_index_container)
- 版本无限*Undo
- 单线程写入加锁
- 无备份机制，容量有限


Caller
MIndex1
MIndex2
MIndex3
MIndex4
…
Object1
Stack1
R1
R2
…
R10

Index

```
   */
  void undo() {      Daniel Larimer, 2 years ago • Initial checkin
     if( !enabled() ) return;

     const auto& head = _stack.back();

     for( auto& item : head.old_values ) {
        auto ok = _indices.modify( _indices.find( item.second.id ), [&]( value_type& v ) {
           v = std::move( item.second );
        });
        if( !ok ) BOOST_THROW_EXCEPTION( std::logic_error( "Could not modify object, most li
     }

     for( auto id : head.new_ids )
     {
        _indices.erase( _indices.find( id ) );
     }
     _next_id = head.old_next_id;

     for( auto& item : head.removed_values ) {
        bool ok = _indices.emplace( std::move( item.second ) ).second;
        if( !ok ) BOOST_THROW_EXCEPTION( std::logic_error( "Could not restore object, most l
     }

     _stack.pop_back();
     --_revision;
  }
```

Undo

EOS为什么造轮子?

- 安全性可控?
- 事务性场景不同
- litetree - sqlite with commits & branches
  - https://github.com/aergoio/litetree

```
/* The block log is an external append only log of the blocks. Blocks should only be written
 * to the log after they irreverisble as the log is append only. The log is a doubly linked
 * list of blocks. There is a secondary index file of only block positions that enables O(1)
 * random access lookup by block number.
 *
 * +---------+---------------+---------+---------------+-----+------------+-------------------+
 * | Block 1 | Pos of Block 1 | Block 2 | Pos of Block 2 | ... | Head Block | Pos of Head Block |
 * +---------+---------------+---------+---------------+-----+------------+-------------------+
 *
 * +---------------+---------------+-----+-------------------+
 * | Pos of Block 1 | Pos of Block 2 | ... | Pos of Head Block |
 * +---------------+---------------+-----+-------------------+
 *
 * The block log can be walked in order by deserializing a block, skipping 8 bytes, deserializing a
 * block, repeat... The head block of the file can be found by seeking to the position contained
 * in the last 8 bytes the file. The block log can be read backwards by jumping back 8 bytes, following
 * the position, reading the block, jumping back 8 bytes, etc.
 *
 * Blocks can be accessed at random via block number through the index file. Seek to 8 * (block_num - 1)
 * to find the position of the block in the main file.
 *
 * The main file is the only file that needs to persist. The index file can be reconstructed during a
 * linear scan of the main file.
 */
```

Block Log