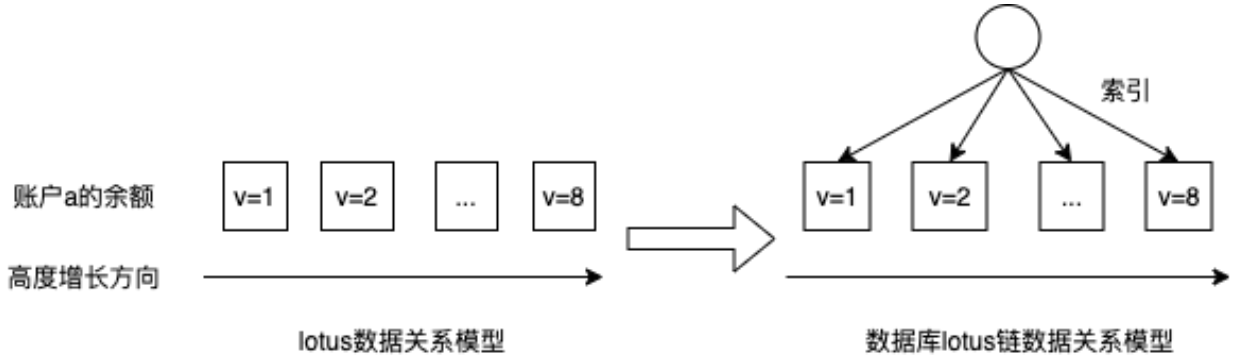
一、概述

阅读本文前需要对filecoin的数据关系要有深刻的理解，详见《filecoin数据与组织方式》

本文描述的是lotus的数据存储设计，不包含其他的业务系统的数据设计！！！



lotus记录了每个tipset下的所有数据，但是它难以对具体某一种数据进行搜索，几乎只能通过高度这一维度进行查询。比如获取账户a的余额变化记录，只能从高度0开始遍历到最新高度进行。数据库能够为具体的数据建立索引，实现各维度的数据检索。此外，通过数据分析，可以进一步挖掘出数据关系。

二、应用场景

（1）任意高度初始化

从任意高度创始是一个硬性需求，可以极大的方便开发进行功能测试。

（2）数据同步模块写入数据

整个tipset的变更集写入到数据库、head链分支切换、更新快照。

（3）数据同步模块从数据库中恢复

（4）客户端（区块浏览器、全节点功能、深度数据分析）向业务网关查询数据

客户端查询某个高度下的信息（肯定是head链的），观察某个矿工所有信息的变化，观察链数据变化，观察矿工，观察某个actor余额变化，全网信息的变化情况，比如总质押、总货币供应、区块大小变化。客户端会根据lotus给出的数据向db查询，也可能用db查询出的数据向lotus查询。所以db中要记录一些信息，这些信息可以作为向lotus查询的关键参数，而这些信息在db里是没有作为关系查询的。比如cid、stateroot。

（5）离线数据分析

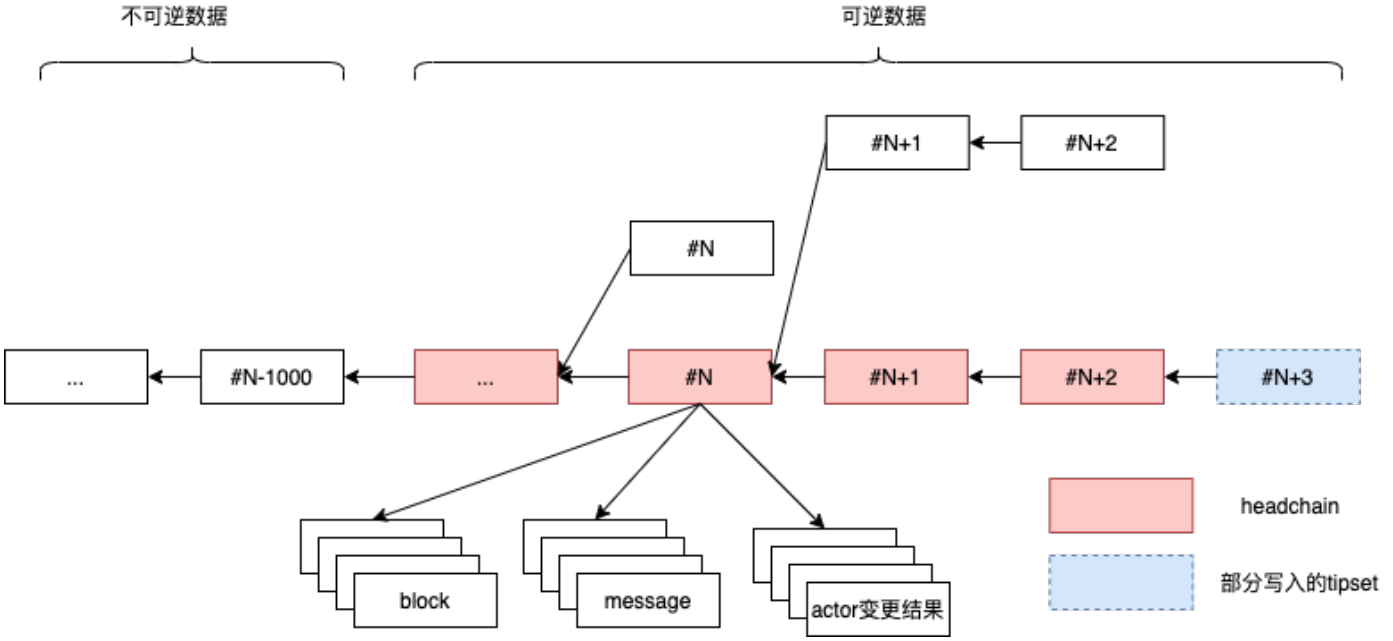
通过OLAP工具进行数据分析。

（6）实时数据分析

实时数据分析不会直接访问数据库，但是它的数据也是从数据库通过消息队列出来的。

（7）补数据工具

三、数据内容



以数据库的形式记录了一条链。除记录tipset、block、message外，还记录了每个tipset、每个block产生的所有的变化信息，比如actor状态与变更记录，每条消息的gas费，奖励和惩罚。大多数信息可以直接从Actor里照搬进来，有些为了满足一些格式要求要做处理，还有一些信息是通过Message加工出来的（比如gas费、奖励和惩罚）。

抽象来看，可以划分为链式结构数据以及业务数据。链式结构是业务最核心的部分，包括了链的生长、分支切换、快照更新，而业务数据（block、message、actor变更）是附庸在某个tipset下的数据。业务数据可以进行进一步的划分，分为输入、输出、状态。

链式结构：tipset、区块。

业务数据：

（1）输入

消息

（2）输出

收据、消息执行结果

（3）状态

链算力、矿工信息、矿工算力等

四、数据引用关系

4.1 tipset间引用

区块链是前后引用关系，后一个状态数据可能会引用历史上某个状态数据，比如后面的数据记录前面某个业务数据的id。

为了正确的描述这种关系，如果后面的数据引用前面数据，需要用业务上的记录id进行记录，并用不可逆区间+可逆区间的headchain进行搜索，防止重复。

用业务id还有一个好处是如果是从任意高度初始化，只是找不到数据，不影响系统逻辑。还能避免写入tipset数据时向数据库查询。

4.2 tipset内引用

同一个tipset内的所有数据的都有从属关系，比如区块和消息的关系、所有tipset内的数据都属于tipset、消息和收据的关系，可以用数据库自增id+外键（正式环境不设置外键约束）来引用。并且在插入数据的时候就能返回数据库记录id。

五、数据写入模式

本质上，tipset是有数据依赖关系，但是这个关系在lotus上就已经正确处理了，当在内存中形成FullTipSet时，就已经用业务标识正确的表示了和历史数据的关系（就是tipset间引用）。所以写入FullTipSet时，不需要读取历史tipset的任何数据，只写不读。这确保了从任意高度初始化时，系统依旧能正常运行。

注：有一个例外，就是AddressIDMap，因为是需要通过两个tipset的initActor对比差异才能计算出本次新增的AddressIDMap，但是这样要内存里维护前一次的信息，太过麻烦，所以可以通过查询数据库来获取前一次的信息。（从lily项目看可以改成在内存中计算好，通过向lotus查询即可，从而不需要向数据库查询历史的，后续改掉）

六、数据规模估算

名	行	数据长度	引擎	创建日期	修改日期	排序规则	注释
access_logs	3	16.00 KB	InnoDB	2022-02-07 09:38:35	2022-02-07 09:51:40	utf8mb4_0900_ai...	
block_header_list	857	1552.00 KB	InnoDB	2022-02-07 09:38:35	2022-02-07 10:18:20	utf8mb4_0900_ai...	
block_message_relation	227,403	14896.00 KB	InnoDB	2022-02-07 09:38:35	2022-02-07 10:18:21	utf8mb4_0900_ai...	
branch_tipset	185	160.00 KB	InnoDB	2022-02-07 09:38:35	2022-02-07 10:18:21	utf8mb4_0900_ai...	
chain_config	1	16.00 KB	InnoDB	2022-02-07 09:38:35	2022-02-07 09:38:52	utf8mb4_0900_ai...	
chain_power_history	186	80.00 KB	InnoDB	2022-02-07 09:38:35	2022-02-07 10:18:20	utf8mb4_0900_ai...	
chain_reward_history	186	80.00 KB	InnoDB	2022-02-07 09:38:35	2022-02-07 10:18:20	utf8mb4_0900_ai...	
common_actor_history	70,661	15920.00 KB	InnoDB	2022-02-07 09:38:35	2022-02-07 10:18:20	utf8mb4_0900_ai...	
execution_trace	287,617	28224.00 KB	InnoDB	2022-02-07 09:38:35	2022-02-07 10:18:21	utf8mb4_0900_ai...	
execution_trace_messages	215,553	253776.00 KB	InnoDB	2022-02-07 09:38:35	2022-02-07 10:18:20	utf8mb4_0900_ai...	
execution_trace_receipts	278,982	28224.00 KB	InnoDB	2022-02-07 09:38:35	2022-02-07 10:18:21	utf8mb4_0900_ai...	
genesis_tipset	1	16.00 KB	InnoDB	2022-02-07 09:38:35	2022-02-07 09:38:35	utf8mb4_0900_ai...	
head_chain	185	96.00 KB	InnoDB	2022-02-07 09:38:35	2022-02-07 10:18:21	utf8mb4_0900_ai...	
init_actor	1,193	144.00 KB	InnoDB	2022-02-07 09:38:35	2022-02-07 10:14:06	utf8mb4_0900_ai...	
invoke_result	82,449	27200.00 KB	InnoDB	2022-02-07 09:38:35	2022-02-07 10:18:20	utf8mb4_0900_ai...	
irreversible	1	16.00 KB	InnoDB	2022-02-07 09:38:35	2022-02-07 09:38:52	utf8mb4_0900_ai...	
market_deal_proposal	0	16.00 KB	InnoDB	2022-02-07 09:38:35		utf8mb4_0900_ai...	
market_deal_state	0	16.00 KB	InnoDB	2022-02-07 09:38:35		utf8mb4_0900_ai...	
messages	82,533	110240.00 KB	InnoDB	2022-02-07 09:38:35	2022-02-07 10:18:20	utf8mb4_0900_ai...	
miner_history	40,401	5648.00 KB	InnoDB	2022-02-07 09:38:35	2022-02-07 10:18:20	utf8mb4_0900_ai...	
miner_power_history	40,323	4624.00 KB	InnoDB	2022-02-07 09:38:35	2022-02-07 10:18:20	utf8mb4_0900_ai...	
miner_sector_event	0	16.00 KB	InnoDB	2022-02-07 09:38:35		utf8mb4_0900_ai...	
multi_sigs	8	16.00 KB	InnoDB	2022-02-07 09:38:35	2022-02-07 10:16:59	utf8mb4_0900_ai...	
payment_channel	0	16.00 KB	InnoDB	2022-02-07 09:38:35		utf8mb4_0900_ai...	
receipts	83,853	6672.00 KB	InnoDB	2022-02-07 09:38:35	2022-02-07 10:18:20	utf8mb4_0900_ai...	
sector_deal_event	0	16.00 KB	InnoDB	2022-02-07 09:38:35		utf8mb4_0900_ai...	
sector_infos	0	16.00 KB	InnoDB	2022-02-07 09:38:35		utf8mb4_0900_ai...	
sector_precommit_info	0	16.00 KB	InnoDB	2022-02-07 09:38:35		utf8mb4_0900_ai...	
tipset_list	186	80.00 KB	InnoDB	2022-02-07 09:38:35	2022-02-07 10:18:20	utf8mb4_0900_ai...	

目前是150万高度，假设系统运行10年，按照现在的区块增长速度，10年后是1500万tipset，区块数量是6900万，消息数66亿，trace是220亿，区块消息关系183亿，账户变更数56亿。

七、表设计

整体设计上要以最精简的数据结构和操作逻辑来表示整个过程，并严格保证逻辑的正确性。

逻辑正确性

（1）head链的正确性

为了获取到head链上的数据，可以通过两次操作，首先获取到headchain，分析出最后一个不可逆高度，然后向业务表查询小于不可逆高度以及在headchain上的数据并按高度排序。即使两步骤是间隔很久，但是操作本身就是存在一定延迟，只不过查出来的是历史的一个branch上的数据，不影响准确性。

（2）tipset间引用关系

基于业务id，配合head链的搜索方式就能实现。

（3）tipset暴露

为了满足tipset写集过大无法在事务中一次提交，需要分批写入，如果还没写入完毕，就不能对外暴露。

业务表考虑到后续可能会增加字段，因此具体的数据内容字段都要求不强制填充，这样可以保证按照一定的顺序升级而不会产生影响。升级顺序，数据库增加字段，新的porter程序往新增字段上加数据，往历史数据的新增字段上补数据，最后升级业务网关读取新增字段。

7.1 链式结构

对于用户，往往只关心HeadChain，而不关心其他分支，此时对外表现的链必须是HeadChain，并且也能向其他业务表获取HeadChain的数据。可以通过高度获取业务表中的数据，再和数据TipSet表中属于HeadChain的记录进行Join。

TipSet表

```
type TipSet struct {
  ID             uint64 `gorm:"primaryKey;autoIncrement:true"`
  Height         int64  `gorm:"index;"`
  ParentTipSetID uint64 `gorm:"column:parent_tipset_id"` // ref TipSet's id
  ParentHeight   int64  `gorm:"column:parent_height"`
  Timestamp      int64

  BlockCount              int
  TotalMessageCount       int
  DedupedMessageCount     int
  UnProcessedMessageCount int
  /*
    BaseFeeBurn         string
    OverEstimationBurn  string
    MinerPenalty        string
    MinerTip            string
    Refund              string
    GasRefund           int64
    GasBurned           int64
  */

  ParentStateRoot string          `gorm:"type:text"`
  ParentWeight    decimal.Decimal `gorm:"type:DECIMAL(38,0)"`
  ParentBaseFee   decimal.Decimal `gorm:"type:DECIMAL(38,0)"`

  Reward  decimal.Decimal `gorm:"type:DECIMAL(38,0)"`
  Penalty decimal.Decimal `gorm:"type:DECIMAL(38,0)"`

  IsValid       bool `gorm:"type:bool;default:false;column:is_valid"`
  InHeadChain   bool `gorm:"type:bool;default:false;column:in_head_chain"`
  HasActorState bool `gorm:"type:bool;default:false;column:has_actor_state"` // for
client access
}

func (TipSet) TableName() string {
```

```
  return "tipset_list"
}
```

因为tipset的key是属于该tipset的所有block的cid，每个cid至少64字节，通常一个tipset有6-7个块，如果用tipset的key作为每个数据的索引，会极大增加数据库大小，甚至可能key就占了数据库的一半大小。因此，tipset的主键是自增id，其他表用外键指向TipSet的ID。此外，tipset中没有记录tipsetkey，一方面是没有这种需求，另外即使真的要查询，通过block_header表查询。porter程序通过reversible_tipset表获得tipset的id。

如果有获取headchain的需求，通过判断其中的IsValid和InHeadChain标记，就能获取正确的headchain，并且也能方便多表join。

tipset是链式结构的参考基准，任何数据如何要查询，建议配合通过tipset做联表查询。

Irreversible表

```
type Irreversible struct {
  ID       uint64 `xorm:"pk autoincr 'id'"`
  TipSetID uint64 `xorm:"'tipset_id'"` // ref TipSet's id
  Height   int64
}
```

记录最后一个不可逆tipset。网关服务不应该使用这个表。

ReversibleTipSet表

```
// for inner use only!!
type ReversibleTipSet struct {
  ID                uint64 `gorm:"primaryKey;autoIncrement:true"`
  TipSetID          uint64 `gorm:"type:bigint;column:tipset_id"` // ref TipSet's id
  Height            int64  `gorm:"type:bigint;column:height"`
  ParentHeight      int64  `gorm:"type:bigint;column:parent_height"`
  Status            uint8
  Tsk               string          `gorm:"type:longtext"`                      // it
can not transform to TipSetKey directly. For read and find only.
  ParentTsk         string          `gorm:"type:longtext;column:parent_tsk"`     // it
can not transform to TipSetKey directly. For read and find only.
  ParentTipSetID    uint64          `gorm:"type:bigint;column:parent_tipset_id"` //
ref TipSet's id, but must not use forigen key constraint, maybe parent not exist if
height=0
  ParentWeight      decimal.Decimal `gorm:"type:DECIMAL(38,0);column:parent_weight"`
  ChangedAddressList string          `gorm:"type:longtext;column:changed_addr_list"`
}

func (ReversibleTipSet) TableName() string {
  return "reversible_tipset"
}

const (
```

```
  TipSetStatusWriting  uint8 = 0
  TipSetStatusDeleting uint8 = 1
  TipSetStatusValid    uint8 = 2
)
```

记录所有的可逆tipset的一些摘要信息，Status标识位表示这个tipset当前的状态，该表只是内部用于处理链生长的控制，不对外使用。

> 追加数据时，只会基于该表的状态进行增长，避免向tipset表查询，以及去重功能。

BlockHeader表

```
type BlockHeader struct {
  ID     uint64 `gorm:"primaryKey;autoIncrement:true"`
  Height int64  `gorm:"index"`
  // no parent info, because block may incoming but it's parent not exist in local
  Cid          string `gorm:"uniqueIndex;type:varchar(255)"`
  Miner        string `gorm:"index;type:varchar(255)"`
  Size         int
  Reward       decimal.Decimal `gorm:"type:DECIMAL(38,0)"`
  Penalty      decimal.Decimal `gorm:"type:DECIMAL(38,0)"`
  WinCount     int64
  MessageCount int
  IsOrphan     bool `gorm:"type:bool;default:false;column:is_orphan"`

  Raw []byte
}
```

记录原始区块信息。孤块标记会该高度变为不可逆后进行检查并设置。

TipSetBlock

```
type TipSetBlock struct {
  ID                          uint64          `gorm:"primaryKey;autoIncrement:true"`
  Height                      int64           `gorm:"index"`
  TipSetID                    uint64          `gorm:"column:tipset_id"`       // ref
TipSet's id
  BlockID                     uint64          `gorm:"index;column:block_id"`  // ref
Block's id
  IndexInTipSet               uint16          `gorm:"column:index_in_tipset"` // array
index of BlockHeader in the tipset which belongs to, just like
tipset.Blocks[IndexInTipSet]
  Reward                      decimal.Decimal `gorm:"type:DECIMAL(38,0)"`
  Penalty                     decimal.Decimal `gorm:"type:DECIMAL(38,0)"`
  UniqueMessageCountInTipSet int
`gorm:"column:unique_message_count_in_tipset"`
  UnProcessedMessageCount     int             `gorm:"column:unprocessed_msg_count"`
}
```

记录在tipset下的block，包含一些加工出来的数据，区块奖励、罚金等，这些数据是通过message加工出来的。因为只有产生了下一个块，才会有当前块所属的tipset，才能获取到系统消息产生的区块奖励、罚金等。

7.2 输入

UserMessage表

```
type UserMessage struct {
  ID uint64 `gorm:"primaryKey;autoIncrement:true"`

  Size    int    `gorm:"column:size"`
  Cid     string `gorm:"uniqueIndex;type:varchar(255)"`
  Version uint64
  Nonce   uint64
  From    string          `gorm:"index;type:varchar(255)"`
  To      string          `gorm:"index;type:varchar(255)"`
  Value   decimal.Decimal `gorm:"type:DECIMAL(38,0)"`

  GasLimit   int64
  GasFeeCap  decimal.Decimal `gorm:"type:DECIMAL(38,0)"`
  GasPremium decimal.Decimal `gorm:"type:DECIMAL(38,0)"`

  Method uint64
  Params []byte

  Signature []byte
}
```

只有用户消息，不记录高度。

From和To这两个地址作索引，cid作为唯一索引。

SystemMessage

```
type SystemMessage struct {
  ID       uint64 `gorm:"primaryKey;autoIncrement:true"`
  TipSetID uint64 `gorm:"column:tipset_id"` // ref TipSet's id
  Height   int64  `gorm:"index"`

  Size    int    `gorm:"column:size"`
  Cid     string `gorm:"index;type:varchar(255)"`
  Version uint64
  Nonce   uint64
  From    string          `gorm:"index;type:varchar(255)"`
  To      string          `gorm:"index;type:varchar(255)"`
  Value   decimal.Decimal `gorm:"type:DECIMAL(38,0)"`

  GasLimit   int64
  GasFeeCap  decimal.Decimal `gorm:"type:DECIMAL(38,0)"`
  GasPremium decimal.Decimal `gorm:"type:DECIMAL(38,0)"`

  Method uint64
  Params []byte
}
```

系统消息只有在tipset中才会有。

BlockMessageRelation表

```
type BlockMessageRelation struct {
  ID                uint64 `xorm:"pk autoincr 'id'"`
  TipSetID          uint64 `xorm:"index 'tipset_id'"`      // ref TipSet's id
  BlockHeaderID     uint64 `xorm:"index 'blockheader_id'"` // ref BlockHeader's id
  MessageID         uint64 `xorm:"index 'message_id'"`     // ref Message's id
  IndexInBlockHeader int
}
```

用于表示区块与消息的关联关系，因为block和message是多对多的关系。BlockHeaderID指向区块id，MessageID指向用户消息id。

7.3 输出

MessageInvokeResult表

```
type MessageInvokeResult struct {
  ID        uint64 `gorm:"primaryKey;autoIncrement:true"`
  Height    int64  `gorm:"index"`
  TipSetID  uint64 `gorm:"column:tipset_id"`        // ref TipSet's id
  MessageID uint64 `gorm:"index;column:message_id"` // ref Message's id

  MsgType uint8
  // from Invoke Result
  // gas cost
  GasUsed            decimal.Decimal `gorm:"type:DECIMAL(38,0)"`
  BaseFeeBurn        decimal.Decimal `gorm:"type:DECIMAL(38,0)"`
  OverEstimationBurn decimal.Decimal `gorm:"type:DECIMAL(38,0)"`
  MinerPenalty       decimal.Decimal `gorm:"type:DECIMAL(38,0)"`
  MinerTip           decimal.Decimal `gorm:"type:DECIMAL(38,0)"`
  Refund             decimal.Decimal `gorm:"type:DECIMAL(38,0)"`
  TotalCost          decimal.Decimal `gorm:"type:DECIMAL(38,0)"`

  GasRefund decimal.Decimal `gorm:"type:DECIMAL(38,0)"`

  // other result
  Error    string `gorm:"type:text"`
  Duration int64

  // from Receipt
  ExitCode int64
  Return   []byte
}
```

用于记录一条消息（用户消息或系统消息）的调用结果，包含了收据。

ExecutionTraceMessage表

```
type ExecutionTraceMessage struct {
  ID       uint64 `gorm:"primaryKey;autoIncrement:true"`
  TipSetID uint64 `gorm:"column:tipset_id"` // ref TipSet's id
  Height   int64  `gorm:"index"`
  // the user or system message which belong to
  MessageID uint64 `gorm:"index;column:message_id"` // ref Messages's id

  MsgType uint8

  // when common.ExecutionTrace breaks down into model.ExecutionTrace rows
  // these elements can describe the tree and rebuild it.
  Level       int
  Index       int
  ParentIndex int
```

```
  // message
  Size    int    `gorm:"column:size"`
  Cid     string `gorm:"type:varchar(255)"`
  Version uint64
  Nonce   uint64
  From    string          `gorm:"index;type:varchar(255)"`
  To      string          `gorm:"index;type:varchar(255)"`
  Value   decimal.Decimal `gorm:"type:DECIMAL(38,0)"`

  GasLimit   int64
  GasFeeCap  decimal.Decimal `gorm:"type:DECIMAL(38,0)"`
  GasPremium decimal.Decimal `gorm:"type:DECIMAL(38,0)"`

  Method uint64
  Params []byte

  // from ExecutionTrace
  Error    string `gorm:"type:text"`
  Duration int64
  //GasCharge *types.GasTrace // level 0 is nil, level 0 is describe in invoke_result

  ExitCode int64
  Return   []byte
  GasUsed  int64
}
```

表示的是每一条用户消息和系统消息的调用跟踪过程。因为用户消息和系统消息在内部生效是通过发送内部消息给各个actor来实现的。

7.4 状态

这个状态就是系统内部每个actor的状态，每个高度也是记录增量状态。在数据库里，为了引用历史上未变化的状态数据，需要用业务的外键关联。为了简化业务数据的按高度搜索的联表查询，每张表都要记录高度，并且以高度作为索引。实际上，状态都是由后一个tipset产生后，从而获取到当前的tipset，才能根据后一个tipset（或区块？）的tipsetkey获取当前块的状态。

ChainPower表

```
type ChainPower struct {
  ID                                uint64          `xorm:"pk autoincr 'id'"`
  TipSetID                          uint64          `xorm:"index 'tipset_id'"` // ref
TipSet's id
  Height                            int64           `xorm:"index"`
  TotalRawBytes                     decimal.Decimal `xorm:" DECIMAL(128,0)"`
  TotalRawBytesCommitted            decimal.Decimal `xorm:" DECIMAL(128,0)"`
  TotalQualityAdjustedBytes         decimal.Decimal `xorm:" DECIMAL(128,0)"`
```

```
  TotalQualityAdjustedBytesCommitted decimal.Decimal `xorm:" DECIMAL(128,0)"`
  TotalPledgeCollateral              decimal.Decimal `xorm:" DECIMAL(128,0)"`
  QaSmoothedPositionEstimate         decimal.Decimal `xorm:" DECIMAL(128,0)"`
  QaSmoothedVelocityEstimate         decimal.Decimal `xorm:" DECIMAL(128,0)"`
  MinerCount                         int64
  MinerCountAboveMinimumPower        int64
}
```

记录链算力，tipset级别的数据。

ChainReward表

```
type ChainReward struct {
  ID                           uint64          `xorm:"pk autoincr 'id'"`
  TipSetID                     uint64          `xorm:"index 'tipset_id'"` // ref
TipSet's id
  Height                       int64           `xorm:"index"`
  CumSumBaselinePower          decimal.Decimal `xorm:" DECIMAL(128,0)"`
  CumSumRealizedPower          decimal.Decimal `xorm:" DECIMAL(128,0)"`
  EffectiveNetworkTime         int64
  EffectiveBaselinePower       decimal.Decimal `xorm:" DECIMAL(128,0)"`
  NewBaselinePower             decimal.Decimal `xorm:" DECIMAL(128,0)"`
  NewBaseReward                decimal.Decimal `xorm:" DECIMAL(128,0)"`
  NewSmoothingPositionEstimate decimal.Decimal `xorm:" DECIMAL(128,0)"`
  NewSmoothingVelocityEstimate decimal.Decimal `xorm:" DECIMAL(128,0)"`
  TotalMinedReward             decimal.Decimal `xorm:" DECIMAL(128,0)"`
  Timestamp                    int64
}
```

记录链奖励，tipset级别的数据。

CommonActor表

```
type CommonActor struct {
  ID       uint64 `xorm:"pk autoincr 'id'"`
  TipSetID uint64 `xorm:"index 'tipset_id'"` // ref TipSet's id
  Address  string `xorm:"index"`
  Height   int64  `xorm:"index"`
  Head     string // may need to repair data, it's easy to read data by stateroot
  Nonce    uint64
  Balance  decimal.Decimal `xorm:" DECIMAL(128,0)"`
}
```

所有Actor的基本状态信息，每次产生tipset时，就记录本次变更的actor。从状态树中读出来的时候就是这些基础信息。包含了普通用户和系统actor。实际上就是以太坊的account。

InitActor表

```
type InitActor struct {
  ID       uint64 `gorm:"primaryKey;autoIncrement:true"`
  TipSetID uint64 `gorm:"column:tipset_id"` // ref TipSet's id
  Height   int64  `gorm:"index"`
  Address  string `gorm:"index;type:varchar(255)"`
  ActorID  string `gorm:"index;type:varchar(255);column:actor_id"`
  // TODO how to describe id->pubkey
}
```

记录了Address和ActorID之间的映射关系

Miner表

```
type Miner struct {
  ID       uint64 `xorm:"pk autoincr 'id'"`
  TipSetID uint64 `xorm:"index 'tipset_id'"` // ref TipSet's id
  Height   int64  `gorm:"index"`

  MinerID    string `xorm:"index text"`
  OwnerAddr  string `xorm:"text"`
  WorkerAddr string `xorm:"text"`
  PeerID     string `xorm:"text"`
  SectorSize uint64
}
```

矿工表，这些矿工信息一般变化少。

MinerPower表

```
type MinerPower struct {
  ID       uint64          `xorm:"pk autoincr 'id'"`
  TipSetID uint64          `xorm:"index 'tipset_id'"` // ref TipSet's id
  Miner    string          `xorm:"index text"`
  Height   int64           `xorm:"index"`
  RawPower decimal.Decimal `xorm:" DECIMAL(128,0)"`
  QalPower decimal.Decimal `xorm:" DECIMAL(128,0)"`
}
```

记录矿工算力变化历史。

SectorInfo表

```
type SectorInfo struct {
  ID       uint64 `xorm:"pk autoincr 'id'"`
  TipSetID uint64 `xorm:"index 'tipset_id'"` // ref TipSet's id

  MinerID         string `xorm:"index text"`
  SectorID        uint64
  SealedCid       string `xorm:"text"`
  ActivationEpoch int64
  ExpirationEpoch int64

  DealWeight         decimal.Decimal `xorm:" DECIMAL(128,0)"`
  VerifiedDealWeight decimal.Decimal `xorm:" DECIMAL(128,0)"`

  InitialPledge         decimal.Decimal `xorm:" DECIMAL(128,0)"`
  ExpectedDayReward     decimal.Decimal `xorm:" DECIMAL(128,0)"`
  ExpectedStoragePledge decimal.Decimal `xorm:" DECIMAL(128,0)"`
}
```

SectorPrecommitInfo表

```
type SectorPrecommitInfo struct {
  ID       uint64 `xorm:"pk autoincr 'id'"`
  TipSetID uint64 `xorm:"index 'tipset_id'"` // ref TipSet's id

  MinerID   string
  SectorID  uint64
  SealedCid string

  SealRandEpoch    int64
  ExpirationEpoch  int64
  PrecommitDeposit decimal.Decimal `xorm:" DECIMAL(128,0)"`
  PrecommitEpoch   int64

  DealWeight         decimal.Decimal `xorm:" DECIMAL(128,0)"`
  VerifiedDealWeight decimal.Decimal `xorm:" DECIMAL(128,0)"`

  IsReplaceCapacity bool

  ReplaceSectorDeadline  uint64
  ReplaceSectorPartition uint64
  ReplaceSectorNumber    uint64
}
```

MinerSectorEvent表

```
type MinerSectorEvent struct {
  ID       uint64 `xorm:"pk autoincr 'id'"`
  TipSetID uint64 `xorm:"index 'tipset_id'"` // ref TipSet's id

  MinerID  string
  SectorID uint64
  Event    SectorLifecycleEvent
}
```

SectorDealEvent表

```
type SectorDealEvent struct {
  ID       uint64 `xorm:"pk autoincr 'id'"`
  TipSetID uint64 `xorm:"index 'tipset_id'"` // ref TipSet's id

  DealID   uint64 // which is deal table?
  MinerID  string
  SectorID uint64
}
```

dealid是业务id，对应的deal有自己的成交号。

PaymentChannel表

```
type PaymentChannel struct {
  ID       uint64 `xorm:"pk autoincr 'id'"`
  TipSetID uint64 `xorm:"index 'tipset_id'"` // ref TipSet's id
  Height   int64  `xorm:"index"`

  From       string `xorm:"index text"`
  To         string `xorm:"index text"`
  SettlingAt int64
  ToSend     string
  LaneCount  uint64
  //LaneState  map[uint64]model.LaneState
}
```

记录的是支付记录而不是支付通道的中间状态。

MultiSig表

```
type MultiSig struct {
  ID       uint64 `gorm:"primaryKey;autoIncrement:true"`
  TipSetID uint64 `gorm:"column:tipset_id"` // ref TipSet's id
  Height   int64  `gorm:"index"`

  MultisigID    string `gorm:"type:text"`
  TransactionID int64

  // Transaction State
  To       string          `gorm:"type:varchar(255)"`
  Value    decimal.Decimal `gorm:"type:DECIMAL(38,0)"`
  Method   uint64
  Params   []byte
  Approved string `gorm:"type:longtext"`
}
```

多签事件包括新增、移除和修改，对应MultiSig Actor。

似乎对数据分析没用?

7.5 其他

GenesisTipSet表

```
type GenesisTipSet struct {
  ID  uint64 `xorm:"pk autoincr 'id'"`
  Raw []byte
}
```

记录创世块，可以和lotus节点对比，防止数据库链和lotus节点的创世信息不同导致出现奇怪问题。

EmptyHeight

```
type EmptyHeight struct {
  ID        uint64 `gorm:"primaryKey;autoIncrement:true"`
  Height    int64  `gorm:"uniqueIndex"`
  Timestamp int64
}
```

OrphanBlock

```
type OrphanBlock struct {
  ID     uint64 `gorm:"primaryKey;autoIncrement:true"`
  Height int64  `gorm:"index"`
  Cid    string `gorm:"uniqueIndex;type:varchar(255)"`
  Miner  string `gorm:"index;type:varchar(255)"`
}
```

记录孤块，当tipset变为不可逆后，才把没有child的块设置为孤块。

ChainConfig表

```
type ChainConfig struct {
  ID                  uint64 `xorm:"pk autoincr 'id'"`
  ReversibleRangeSize int
}
```

记录了一些配置信息

PidFile表

```
type PidFile struct {
  ID   uint64 `gorm:"primaryKey;autoIncrement:true"`
  Info string
}
```

启动时检查并创建该表，如果表已存在，退出，否则创建该表。该表表示是否有程序正在写数据库，类似与本地的一个pid文件，表示进程是否在运行。

八、索引

考虑到数据规模太大，从有些维度进行搜索时涉及的数据量非常庞大，而且分页数据需要先排序再分页。为了简化这一实现，通过缓存来记录，而且区块链数据是不会变化的（但是会分支切换），可以一直保存。

1.从账户的角度来搜索数据

账户发送的消息

账户收到的消息

账户资金变更明细

2.记录矿工出的所有块

3.主链的所有tipset

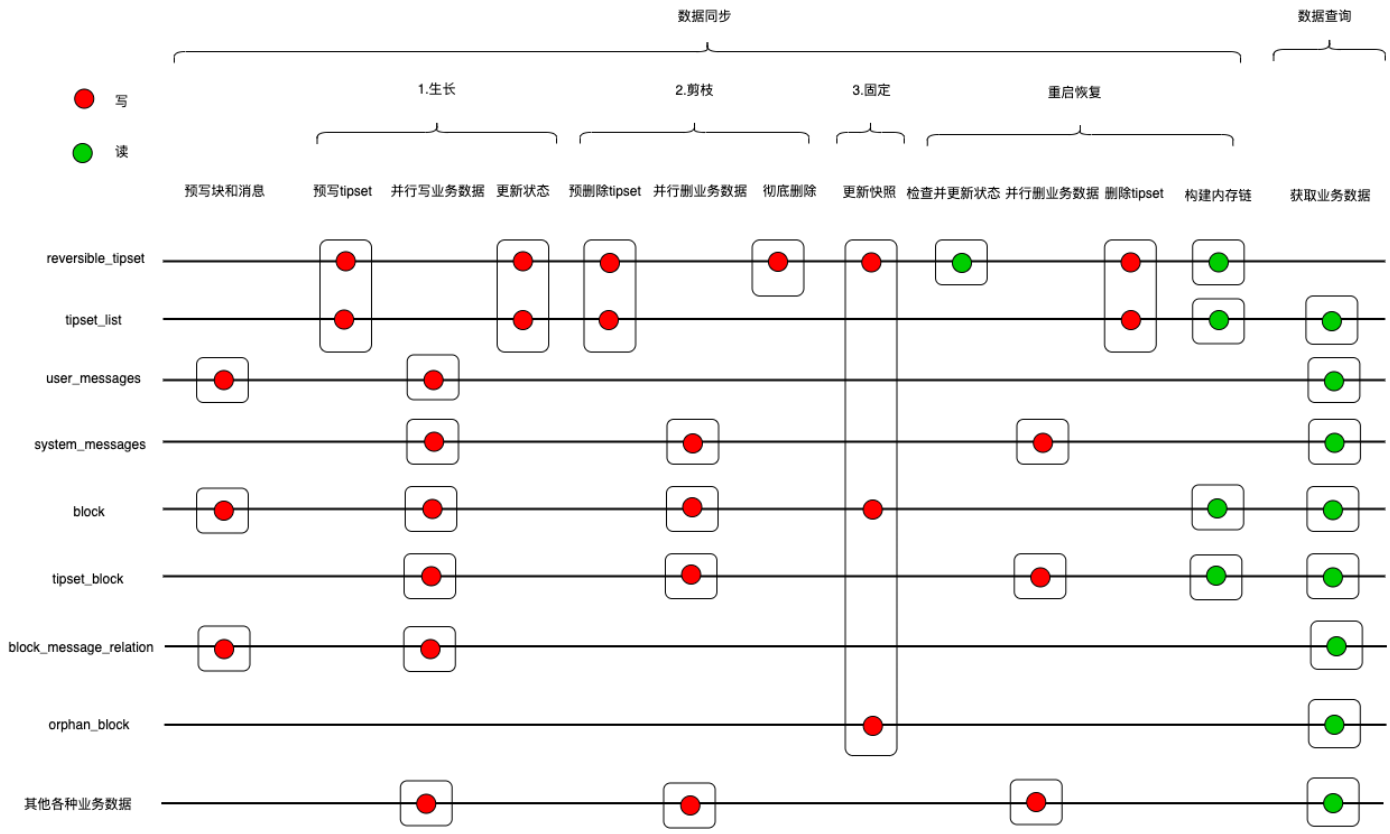
九、数据一致性

数据库数据的一致性，数据库写入时，会设置tipset的is_valid标记，查询时需要带上该标记，和其他业务数据一起事务查询，可以确保数据是一致的。

缓存数据的一致性，采用tx_pipeline提交，确保分支切换的数据都是一致的。

对外表现的一致性，一次性从缓存中获取链时，它实际上就表明了是当时的一个headchain，即使延迟一定时间，它退化成一个branch，但是这个branch是前后衔接的，也还是可以从数据库里根据branch里的tipset_id数据一步步下降获取到对应的区块和消息，依旧保证了一致性。

十、数据竞争



数据同步的具体操作需要看《数据同步详细设计》

通过数据库中的mvcc来表示读写事务的偏序关系。从图中可以看到，对于数据查询，它的事务操作几乎不会和数据同步产生竞争。也正因为如此，数据同步操作不会对数据查询操作的性能产生影响。至于数据同步内部的几个操作，只有预写块和消息会产生少量的竞争，其他都是顺序的。

十一、数据库选型

从以上的要求看，必须是关系型数据库，事务提交，且能各种扩展，性能要高，维护成本低。这些基本特性大部分数据库都能满足。

官方的chainwatch使用的就是PostgreSQL，而且zksync（公司另一个项目）使用的也是PostgreSQL。从统一管理的角度选择PG最合适，但是从数据规模看，PG应该是不满足要求的。

按照数据库的发展趋势，现代化的数据库应该支持可水平扩展、自动分表、HTAP（兼顾事务和分析），比如oceanbase、tidb，从而把精力放在业务上。其中tidb是兼容mysql协议，而且操作尽可能采用orm框架，目前oceanbase是不支持orm框架的。所以选择orm+tidb是合理的选择。

十二、数据库初始化

数据库初始化的时候，要写入初始信息到genesis_tipset、tipset_list、reversible_tipset、headchain、block_header_list，确保数据库链和lotus链是相同的创世块，并且确保数据同步能正常追加数据到数据库。

为了满足快速开发、验证，可以从任意高度进行初始化，初始化高度的tipset是headchain里最后一个tipset。为了简化内部实现，要求不可逆tipset不能为空，高度至少是0，也就是说，初始化高度必须大于等于1.

注意，设置初始化tipset时，要确保该块是不会被回滚的，否则程序可能会出现新的数据无论如何都无法插入到数据库链中。