NX 奈学教育

MySQL InnoDB存储引擎实现原理深入剖析与应用实践（下）

主讲人：陈东

2020.07.28

目录

奈学教育

NX 奈学教育

01.MySQL InnoDB存储引擎内存管理

MySQL事务实现原理

- MVCC
- **undo log**
  - 回滚日志
  - 保证事务原子性
  - 实现数据多版本
  - delete undo log：用于回滚，提交即清理；
  - update undo log：用于回滚，同时实现快照读，不能随便删除
- redo log

MySQL—undolog

MySQL—undolog

思考：undolog如何清理?

依据系统活跃的最小活跃事务ID Read view

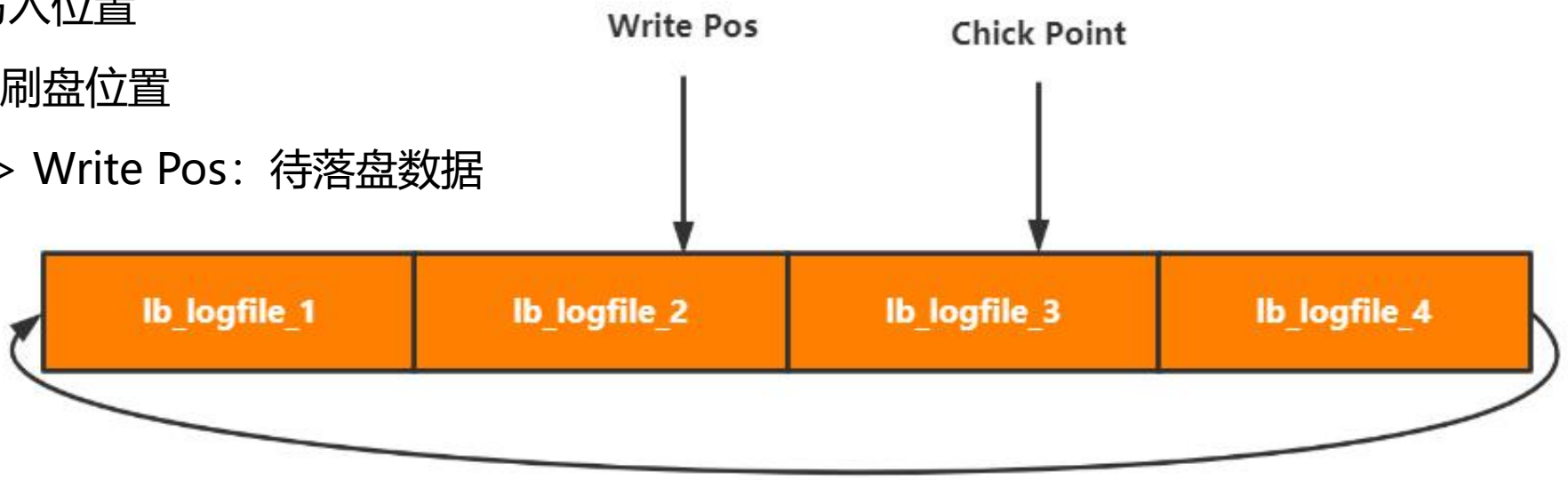
为什么InnoDB count（*）这么慢?

MySQL事务实现原理

- MVCC
- undo log
- **redo log**
  - 实现事务持久性

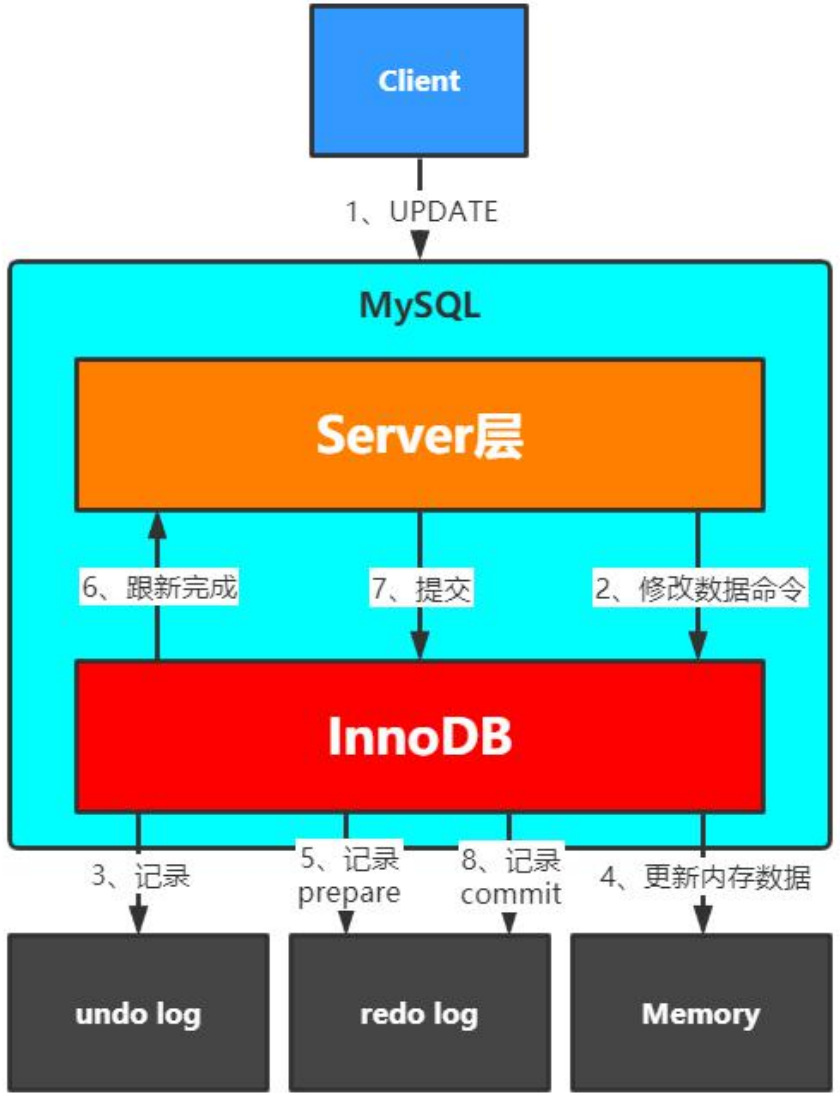
MySQL—redolog

- 记录修改
- 用于异常恢复
- 循环写文件
  - Write Pos：写入位置
  - Chick Point：刷盘位置
  - Chick Point -> Write Pos：待落盘数据



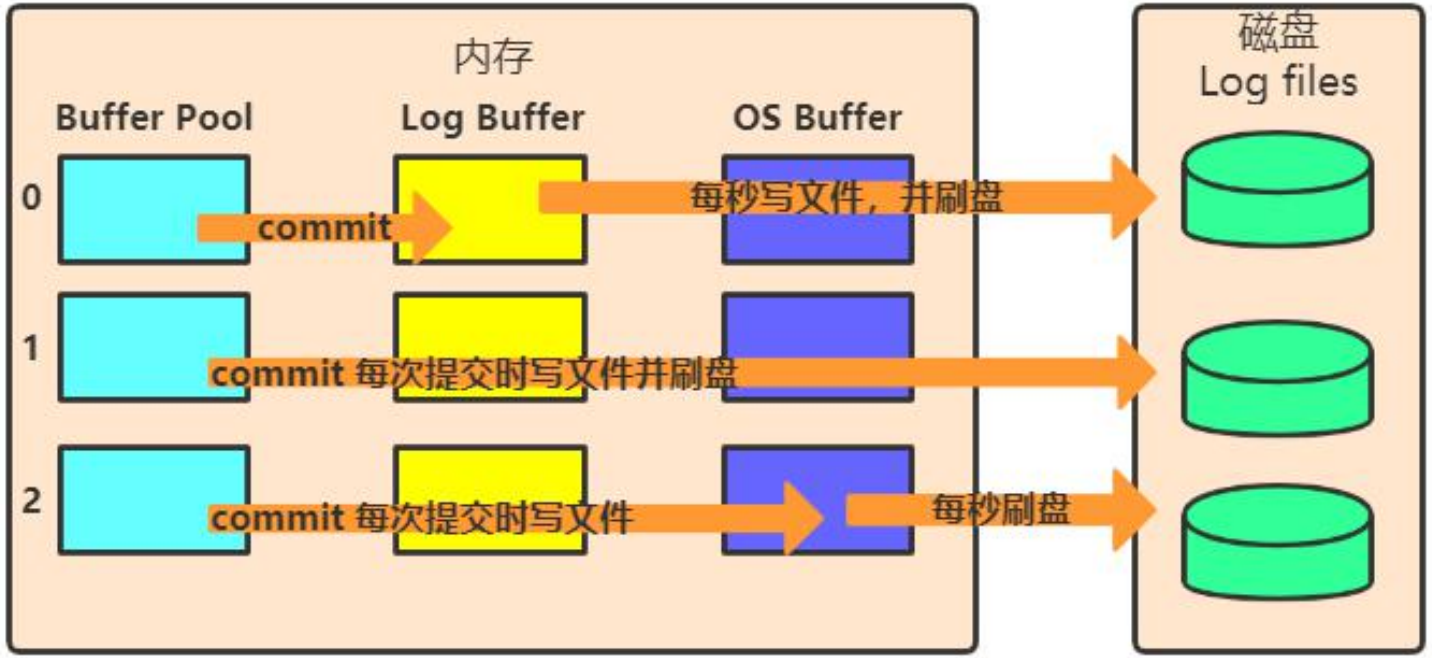
MySQL—redolog

- 写入流程
  - 记录页的修改，状态为prepare
  - 事务提交，讲事务记录为commit状态



MySQL—redolog

- 刷盘时机
  - innodb_flush_log_at_trx_commit



MySQL—redolog

- 意义
  - 体积小，记录页的修改，比写入页代价低
  - 末尾追加，随机写变顺序写，发生改变的页不固定

NX 奈学教育

02.MySQL InnoDB锁机制详解

InnoDB锁种类

- 锁粒度
  - 行级锁
  - 间隙锁
  - 表级锁
- 类型
  - 共享锁（S）
    - 读锁，可以同时被多个事务获取，阻止其他事务对记录的修改；
  - 排他锁（X）
    - 写锁，只能被一个事务获取，允许获得锁的事务修改数据；

InnoDB锁种类

- **行级锁**
  - 作用在索引上
  - 聚簇索引 & 二级索引
- 间隙锁
- 表级锁

uid（PK）	100	101	102	103	104	105
phone	135	138	151	134	133	178

delete from user where uid = 103

InnoDB锁种类

- **行级锁**
  - 作用在索引上
  - 聚簇索引 & 二级索引
- 间隙锁
- 表级锁

uid (PK)	100	101	102	103	104	105
phone (Index)	135	138	151	134	133	178

delete from user where phone = 134

phone	135	138	151	134	133	178
uid	100	101	102	103	104	105

唯一索引

InnoDB锁种类

- **行级锁**
  - 作用在索引上
  - 聚簇索引 & 二级索引
- 间隙锁
- 表级锁

唯一索引/非唯一索引 * RC/RR
4种情况逐一分析

InnoDB锁种类

- **行级锁**
  - 作用在索引上
  - 聚簇索引 & 二级索引
- 间隙锁
- 表级锁

delete from user where phone = 134

phone	135	138	151	134	133	134
uid	100	101	102	103	104	105

uid（PK）	100	101	102	103	104	105
phone (Index)	135	138	151	134	133	134

RC隔离级别

InnoDB锁种类

- **行级锁**
  - 作用在索引上
  - 聚簇索引 & 二级索引
- 间隙锁
- 表级锁

delete from user where phone = 134

phone	133	134	134	135	138	151
uid	104	103	105	100	101	102

uid（PK）	100	101	102	103	104	105
phone (Index)	135	138	151	134	133	134

RR隔离级别

InnoDB锁种类

- 行级锁
- **间隙锁**
  - 解决可重复读模式下的幻读问题；
  - GAP锁不是加在记录上；
  - GAP锁锁住的位置，是两条记录之间的GAP；
  - 保证两次当前读返回一致的记录；
- 表级锁

两次当前读之前，其他的事务不会插入新的满足条件的记录

InnoDB锁种类

- 行级锁
- **间隙锁**
- 表级锁

1、134，X<130

2、134，130<X<150

3、134，X>150

delete from user where phone = 134

phone	131	134	134	137	138	151
uid	140	130	150	100	110	120

1 2 3

uid（PK）	100	110	120	130	140	150
phone (Index)	137	138	151	134	131	134

InnoDB锁种类

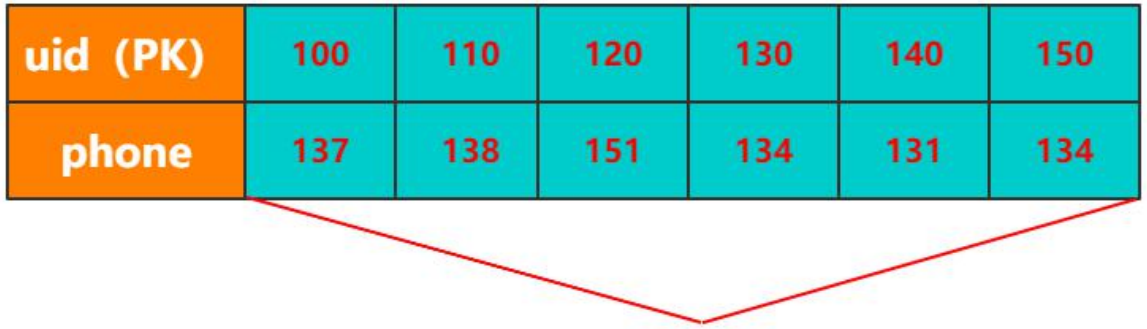
- 行级锁
- 间隙锁
- **表级锁**
  - lock tables
  - 元数据锁（meta data lock，MDL）

InnoDB锁种类

- 行级锁
- 间隙锁
- **表级锁**
  - 全表扫描“表锁”

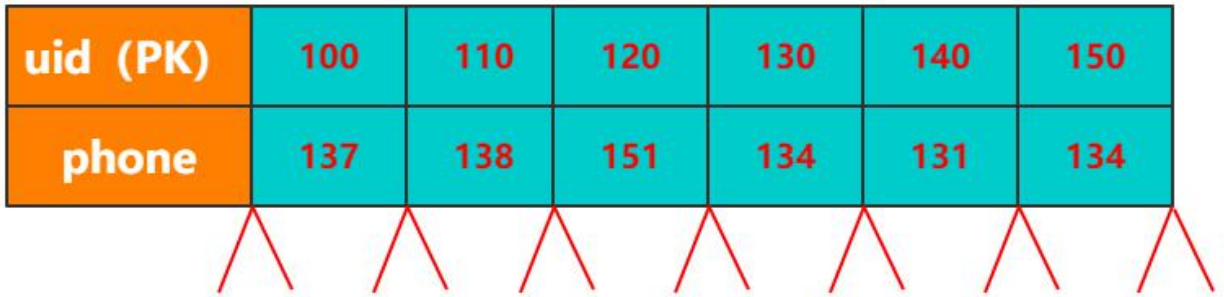
delete from user where phone = 134

RC



所有记录加锁后返回，然后由MySQL Server层进行过滤

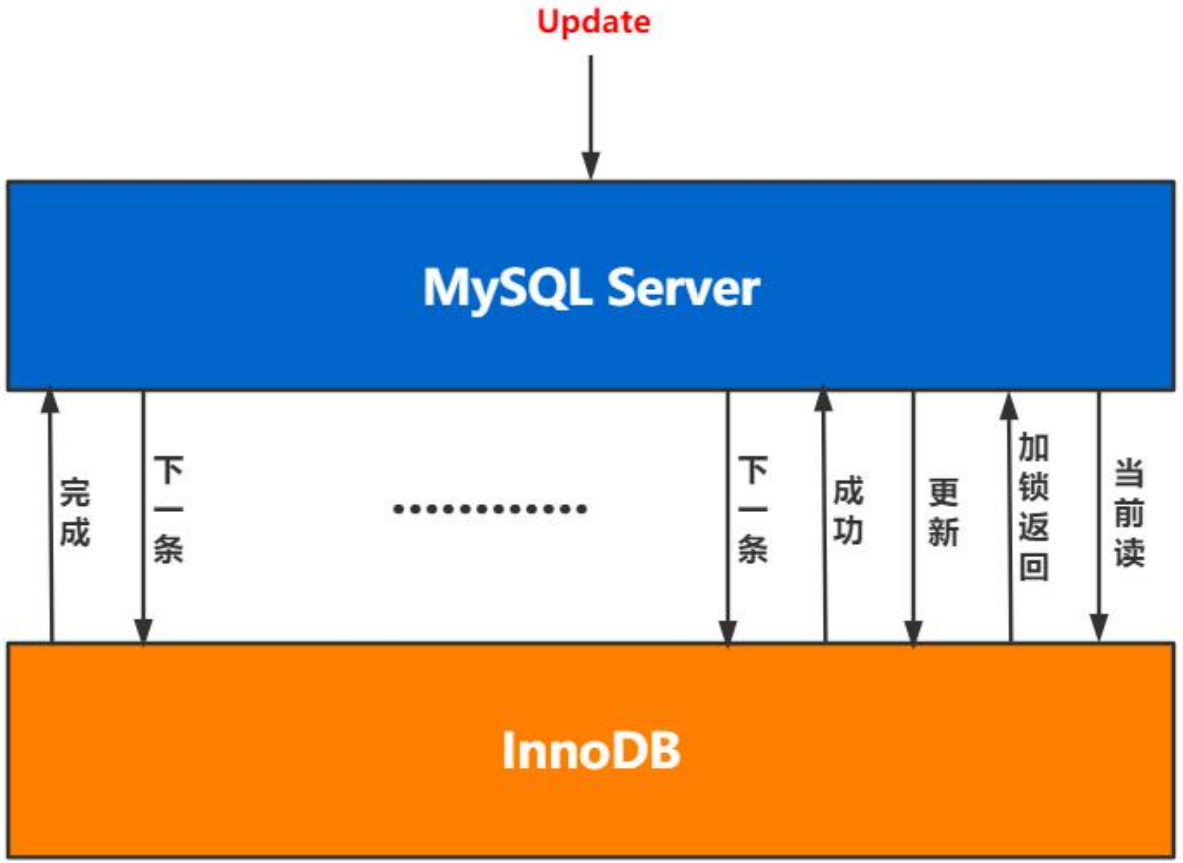
RR



NX 奈学教育

03.MySQL InnoDB数据加锁过程剖析

InnoDB加锁过程



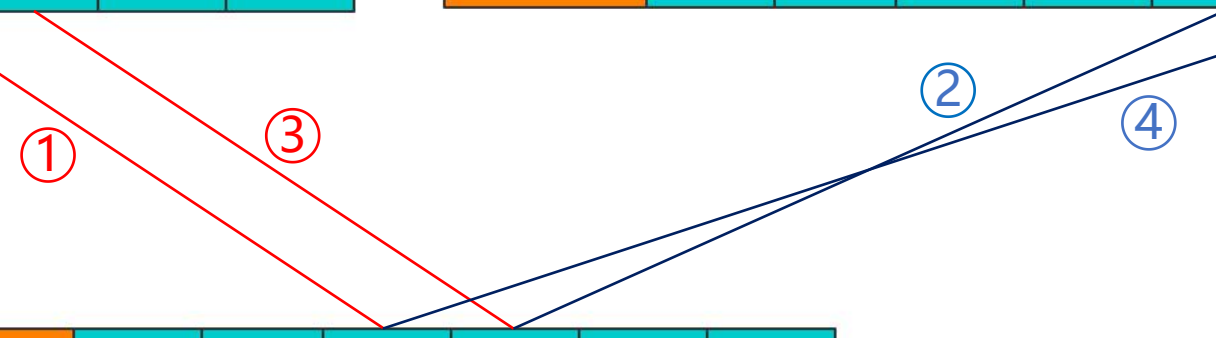
InnoDB加锁过程

T1：UPDATE t_user SET XX=XX WHERE name= ‘f’ ;

T2：SELECT * FROM t_user WHERE age > 33 FOR UPDATE;

name	a	e	f	f	h	x
uid	100	110	120	130	140	150

age	20	23	25	31	35	42
uid	140	150	100	110	130	120



uid（PK）	100	110	120	130	140	150
name (Index)	a	e	f	f	h	x
age (Index)	25	31	42	35	20	23

NX 奈学教育

04.MySQL使用实践经验分享

MySQL使用经验分享

- 索引及数据类型
- 分库分表

索引及数据类型

- 联合索引：优于多列独立索引
- 索引顺序：选择性高的在前面
- 覆盖索引：Key里面包含要查询的数据
- 索引排序：索引同时满足查询和排序

索引及数据类型

- 数据库字符集使用utf8mb4;
- VARCHAR按实际需要分配长度;
- 文本字段建议使用VARCHAR;
- 时间字段建议使用long;
- bool字段建议使用tinyint;
- 枚举字段建议使用tinyint;
- 交易金额建议使用long;
- 禁止使用“%”前导的查询;
- 禁止在索引列进行数学运算，会导致索引失效;
  - select * from t1 where id+1 >1121 不会使用索引
  - select * from t1 where id >1121 - 1 会使用索引

索引及数据类型

- 表必须有主键，建议使用业务主键；
- 单张表中索引数量不超过5个；
- 单个索引字段数不超过5个；
- 字符串索引使用前缀索引，前缀长度不超过10个字符；

MySQL使用经验分享

- 索引及数据类型
- **分库分表**

分库分表

- 是否分表
  - 建议单表不超过1KW
- 分表方式
  - 取模：存储均匀&访问均匀
  - 按时间：冷热库
- 分库
  - 按业务垂直分
  - 水平查分多个库

分库分表—案例分享

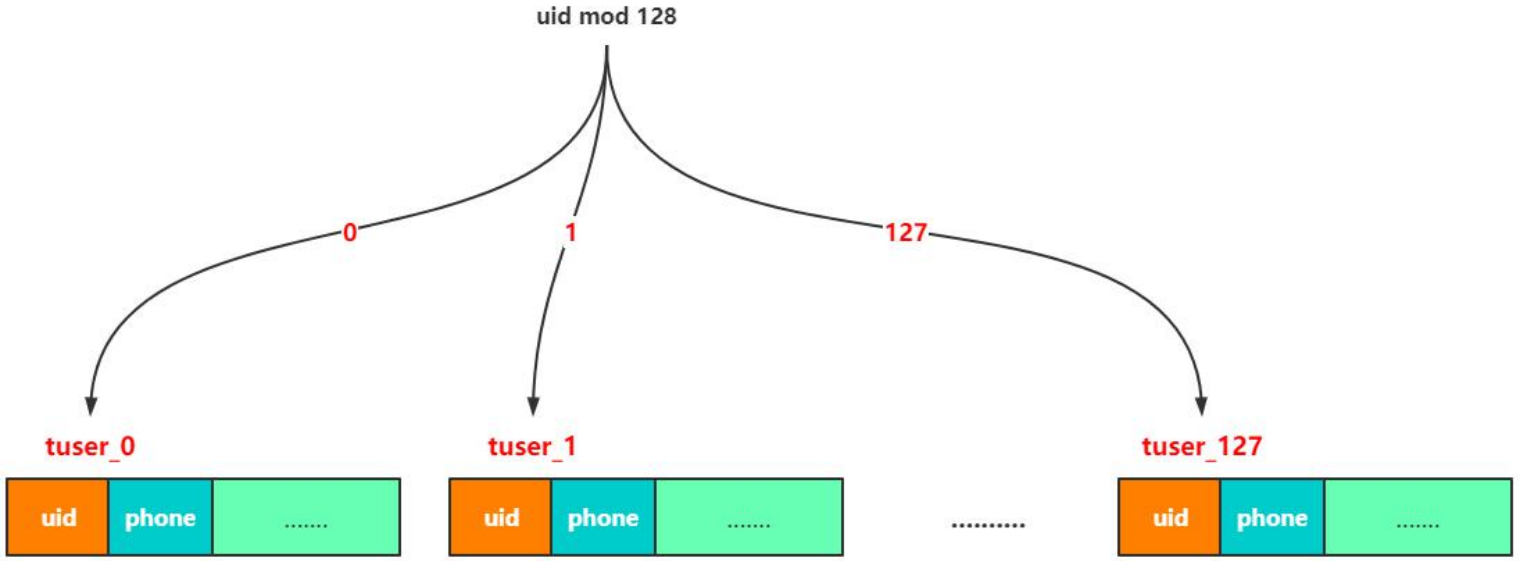
- 用户库分表
- 商品库分表
- 系统消息库分表

分库分表—案例分享

- **用户库分表**
- 商品库分表
- 系统消息库分表

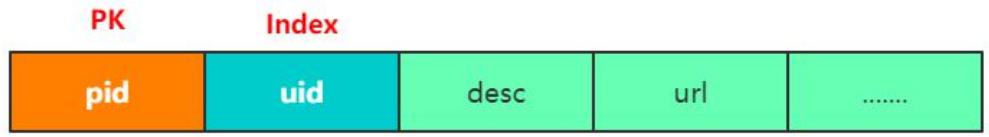
存储均匀&访问均匀 -> 取模

手机号查询怎么办?

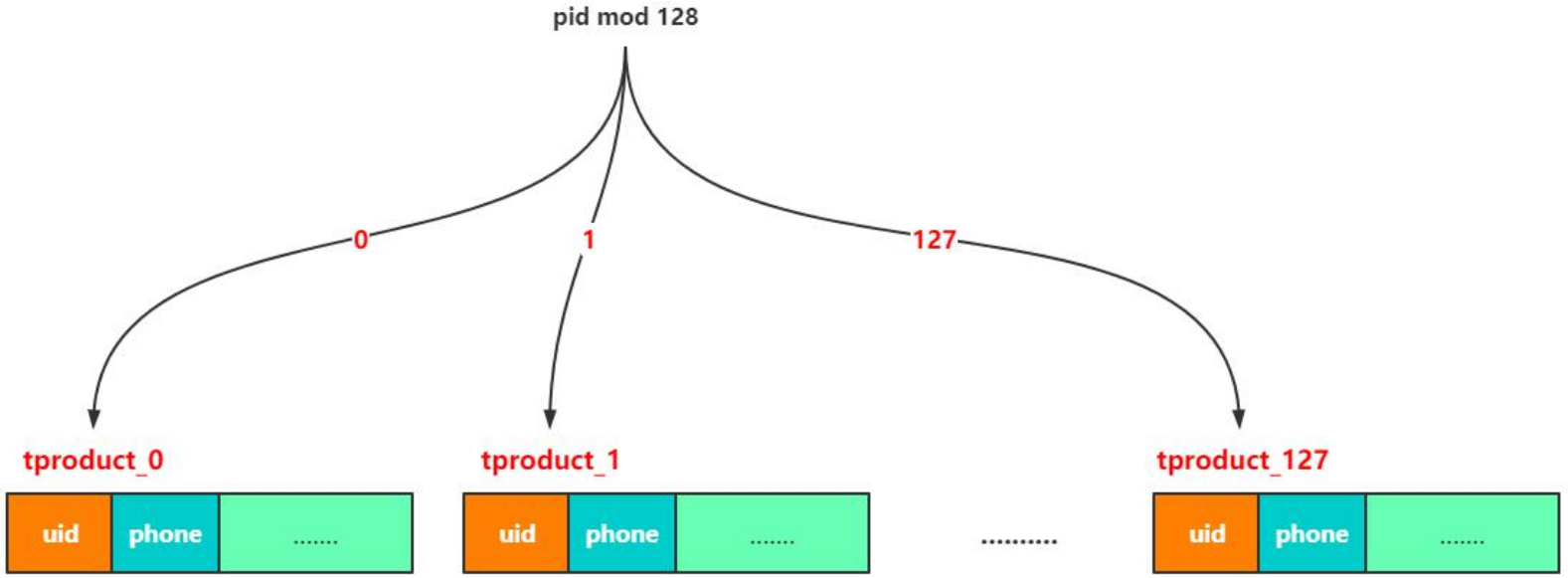


分库分表—案例分享

- 用户库分表
- **商品库分表**
- 系统消息库分表

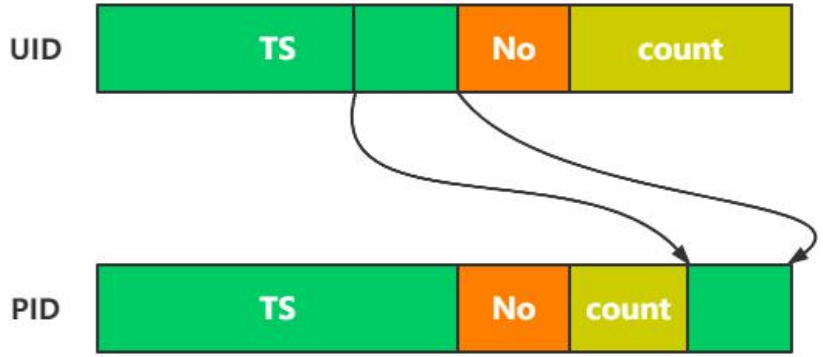


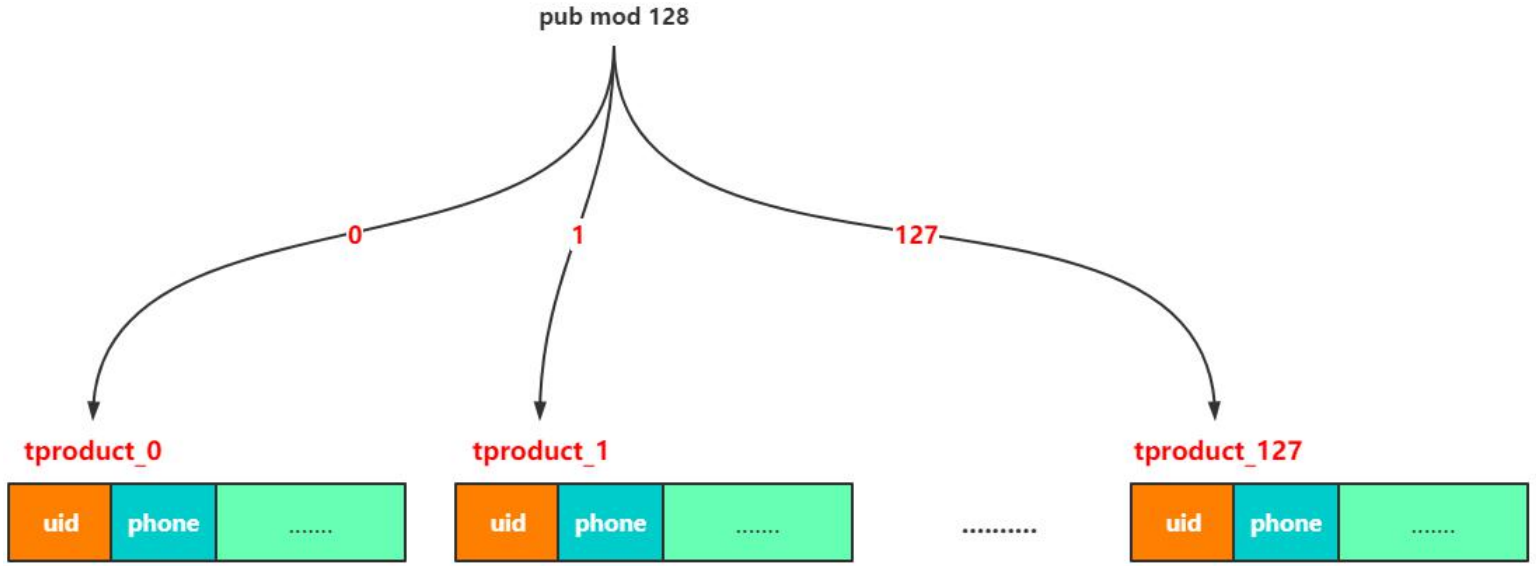
查询自己发布的商品



分库分表—案例分享

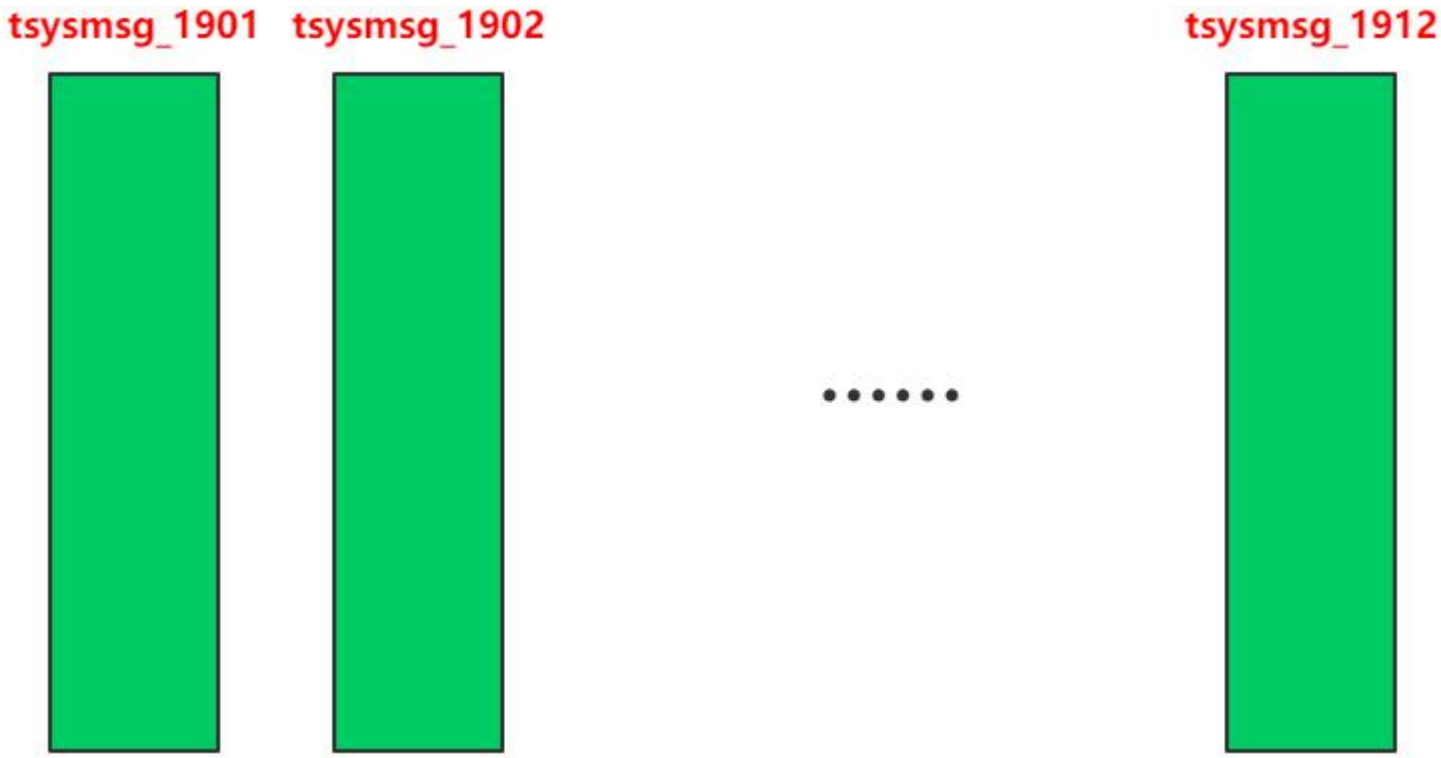
- 用户库分表
- **商品库分表**
  - 两个维度查询必须满足
  - 映射表太重
- 系统消息库分表





分库分表—案例分享

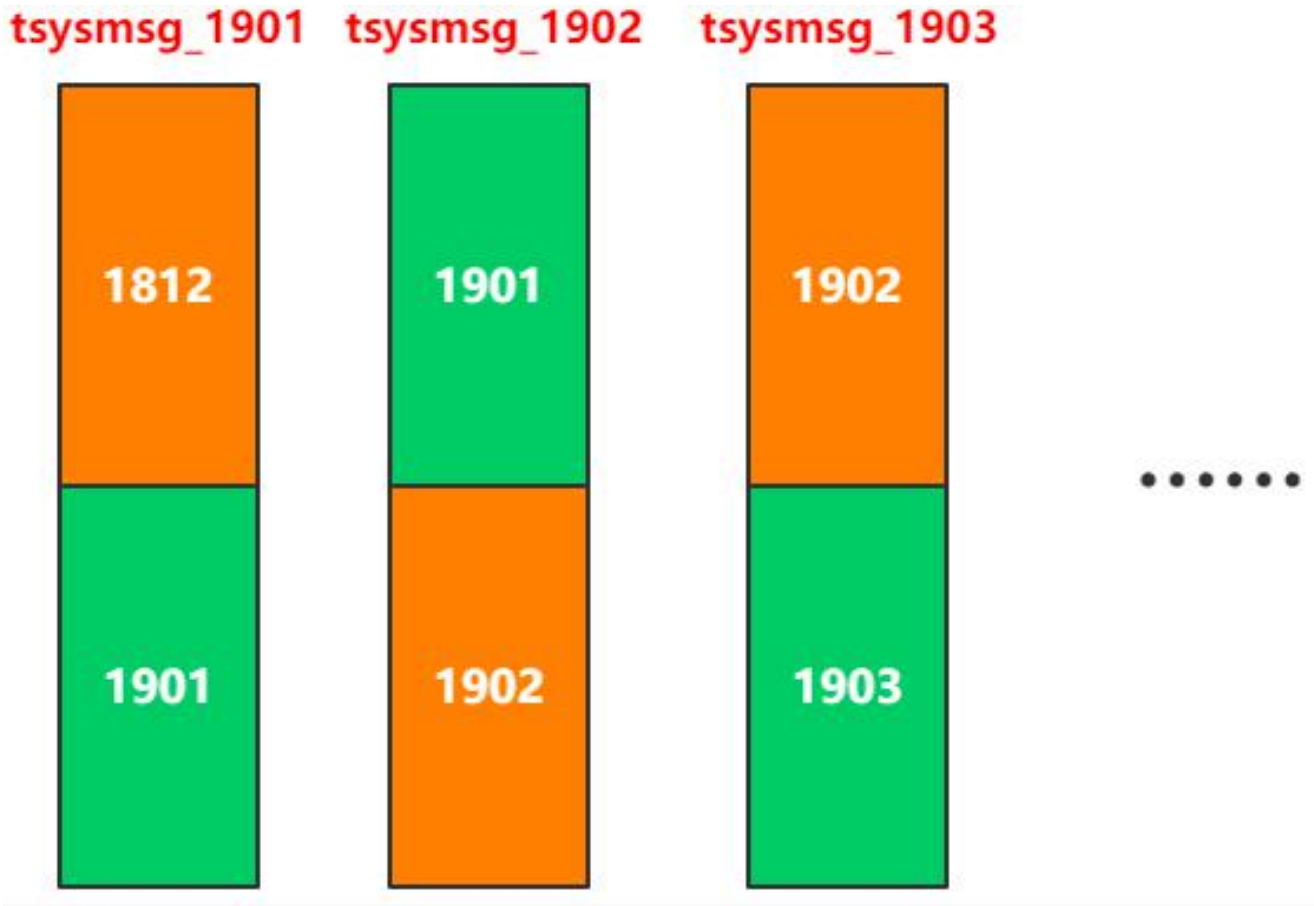
- 用户库分表
- 商品库分表
- **系统消息库分表**
  - 时效性强
  - 冷热数据拆分



思考：有效期30天

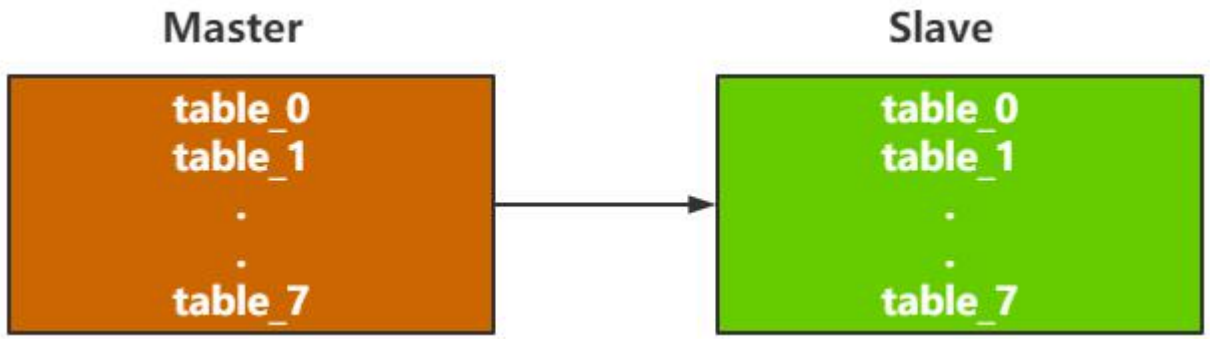
分库分表—案例分享

- 用户库分表
- 商品库分表
- **系统消息库分表**
  - 时效性强
  - 冷热数据拆分

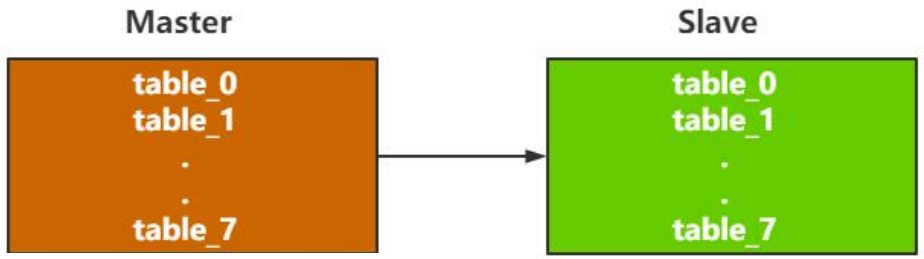


分库分表—案例分享

- 分表分少了怎么办?



分库分表—案例分享



业务修改路由算法，后台清理数据

NX 奈学教育

欢迎关注本人公众号
“架构之美”