3天挑战架构师级MySQL海量数据设计与实践

奈学教育

内容介绍

Day1 MySQL架构体系设计深入剖析篇

MySQL架构体系拆解以及设计深度剖析；
MySQL存储引擎原理拆解以及设计深度剖析；
MySQL锁实现原理拆解以及设计深度剖析；
MySQL事务实现原理拆解以及设计深度剖析；

Day 2 企业千亿级海量数据并发分库分表设计方法论提炼篇

千亿级海量数据高并发场景分库分表设计方法论；
千亿级海量数据高并发场景主键设计选择；
千亿级海量数据高并发场景分片键设计选择；
千亿级海量数据高并发场景分库实践落地方案；
千亿级海量数据高并发场景分表实践落地方案；

Day 3 企业千亿级海量数据真实案例设计与实战篇

万亿级微信消息垂直拆分真实案例实战；
企业级数据库Sharding Sphere分库分表应用设计实战；
企业级分布式事务阿里巴巴Seata应用设计实战；
企业级MySQL数据库高可用应用设计与实战；

陈东

前58集团架构师， 前转转公司架构平台部负责人、高级架构师、技术委员会核心成员， 主导了转转基础架构部门从0到1的建设工作， 负责转转RPC框架和服务治理生态的落地、消息队列的研发和多元化存储体系的建设，以及众多核心基础组件的设计研发和产品化工作，擅长后端架构、中间件、服务治理、存储等技术方向，对即时通讯系统有深刻的研究。

今晚20:00准时开课，请稍后…

奈学教育 出品

目录

NX 奈学教育

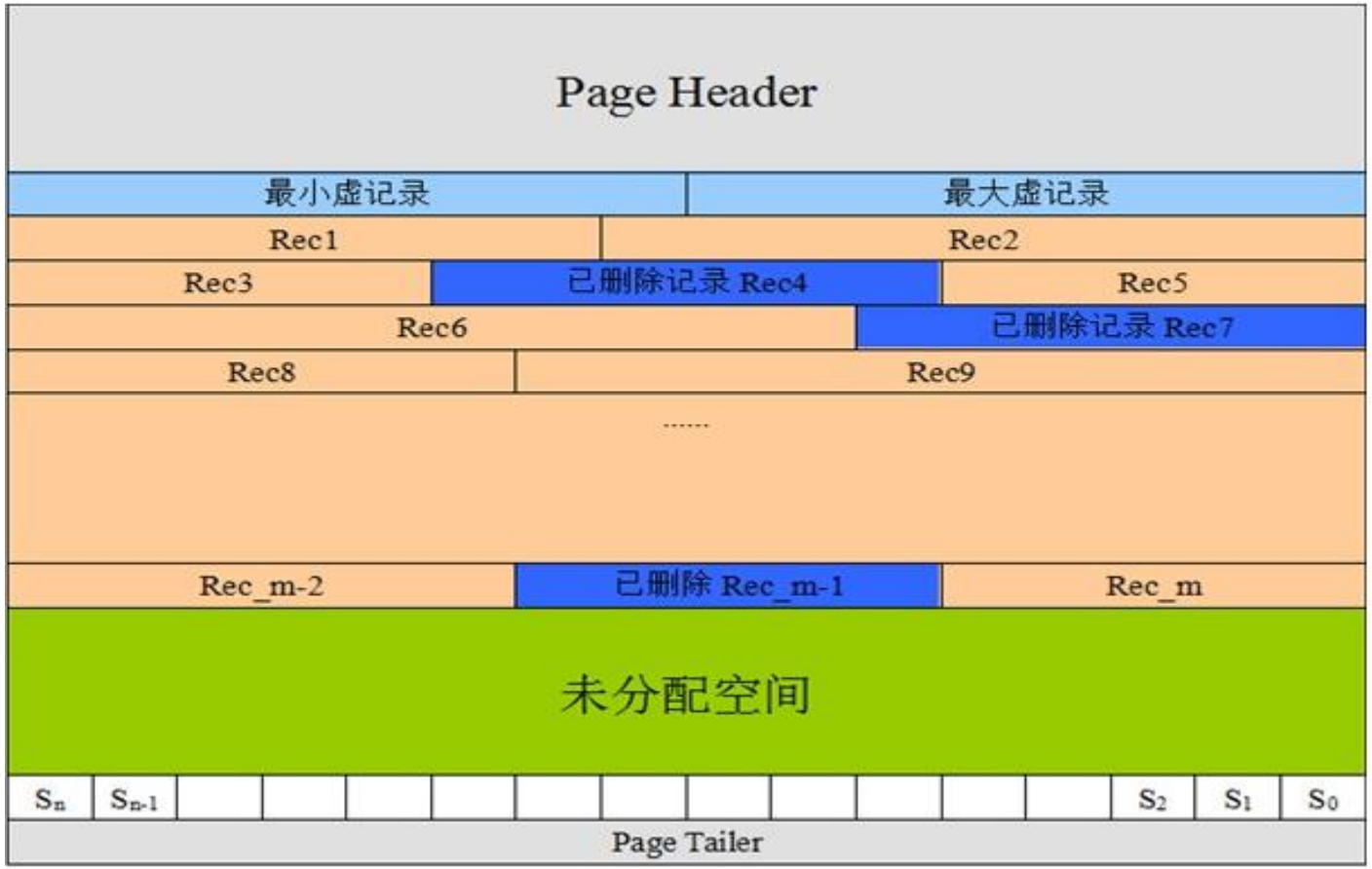
- MySQL存储引擎原理拆解以及设计深度剖析；
- MySQL锁实现原理拆解以及设计深度剖析；
- MySQL事务实现原理拆解以及设计深度剖析；

NX 奈学教育

01.MySQL存储引擎原理拆解以及设计深度剖析

MySQL记录存储

- 页头
- 虚记录
- 记录堆
- 自由空间链表
- 未分配空间
- Slot区
- 页尾



MySQL记录存储

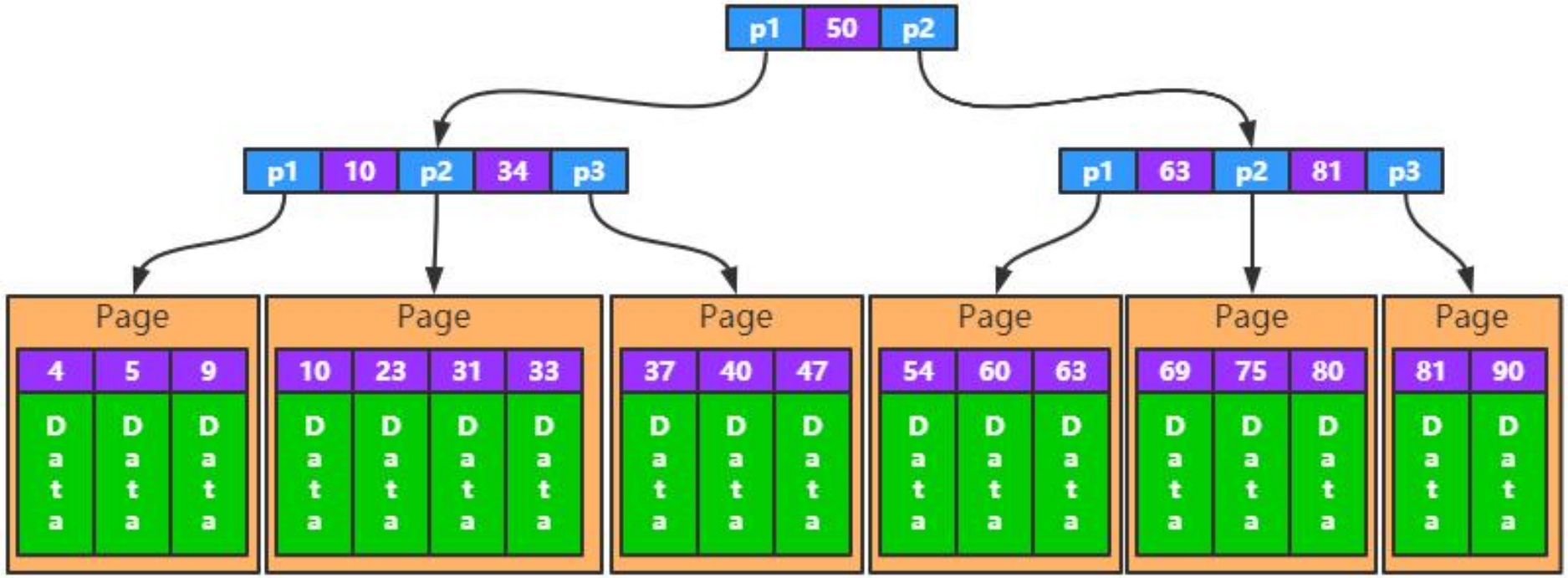
- 页头
  - 记录页面的控制信息，共占56字节，包括页的左右兄弟页面指针、页面空间使用情况等。
- 虚记录
  - 最大虚记录：比页内最大主键还大
  - 最小虚记录：比页内最小主键还小
- 记录堆
  - 行记录存储区，分为有效记录和已删除记录两种
- 自由空间链表
  - 已删除记录组成的链表

MySQL记录存储

- 未分配空间
  - 页面未使用的存储空间；
- 页尾
  - 页面最后部分，占8个字节，主要存储页面的校验信息；

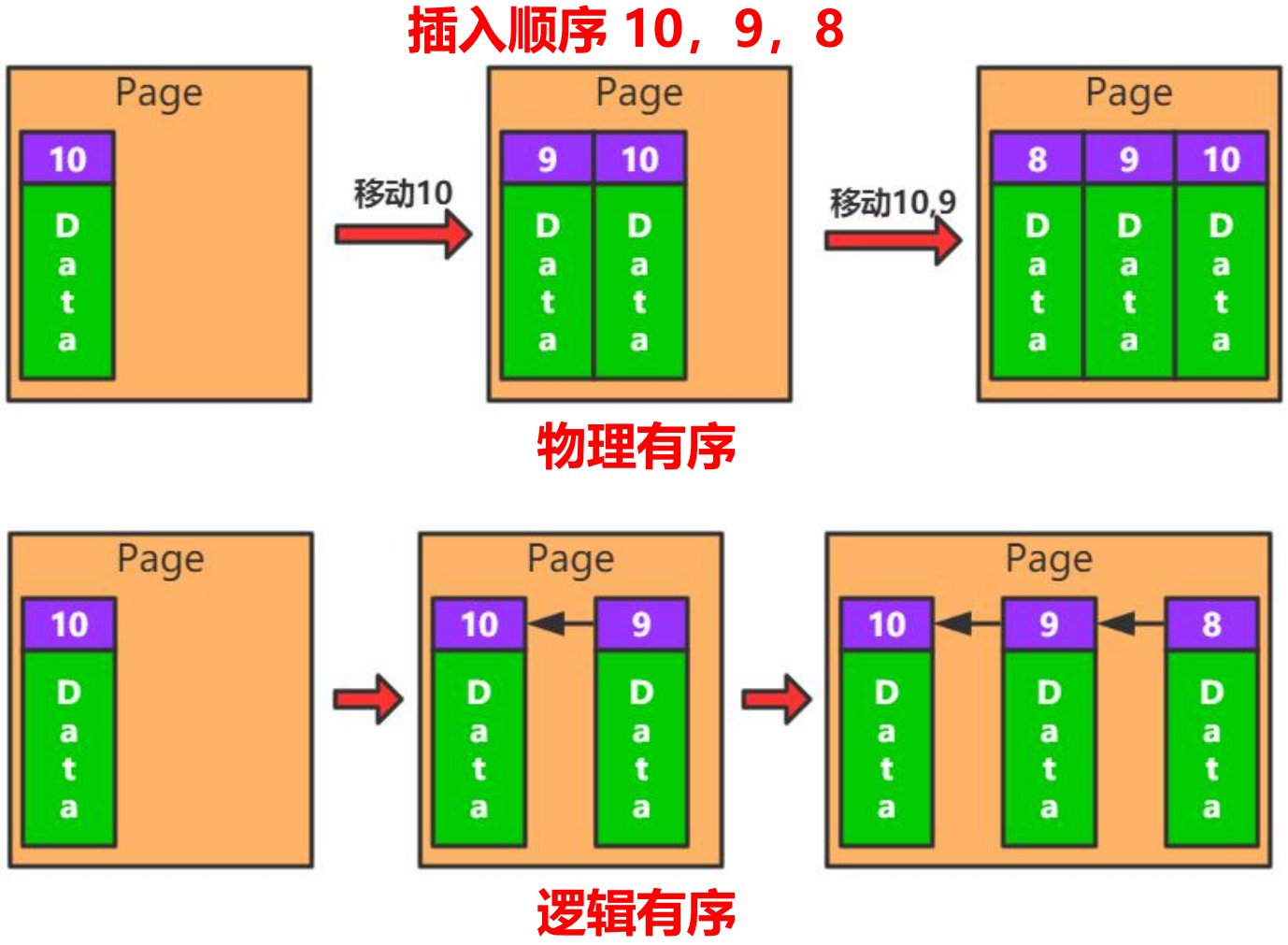
页内记录维护

- 顺序保证
- 插入策略
- 页内查询



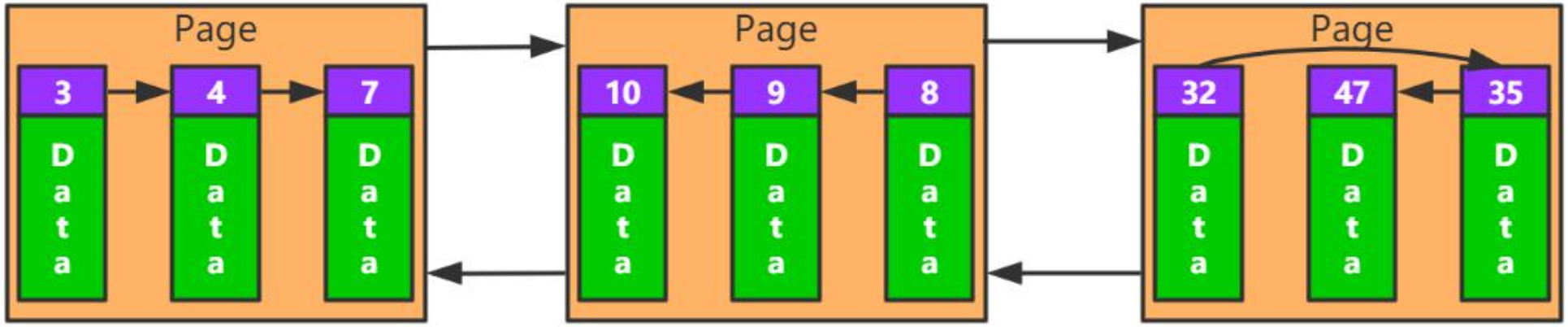
页内记录维护

- **顺序保证**
  - 物理有序
  - 逻辑有序
- 插入策略
- 页内查询



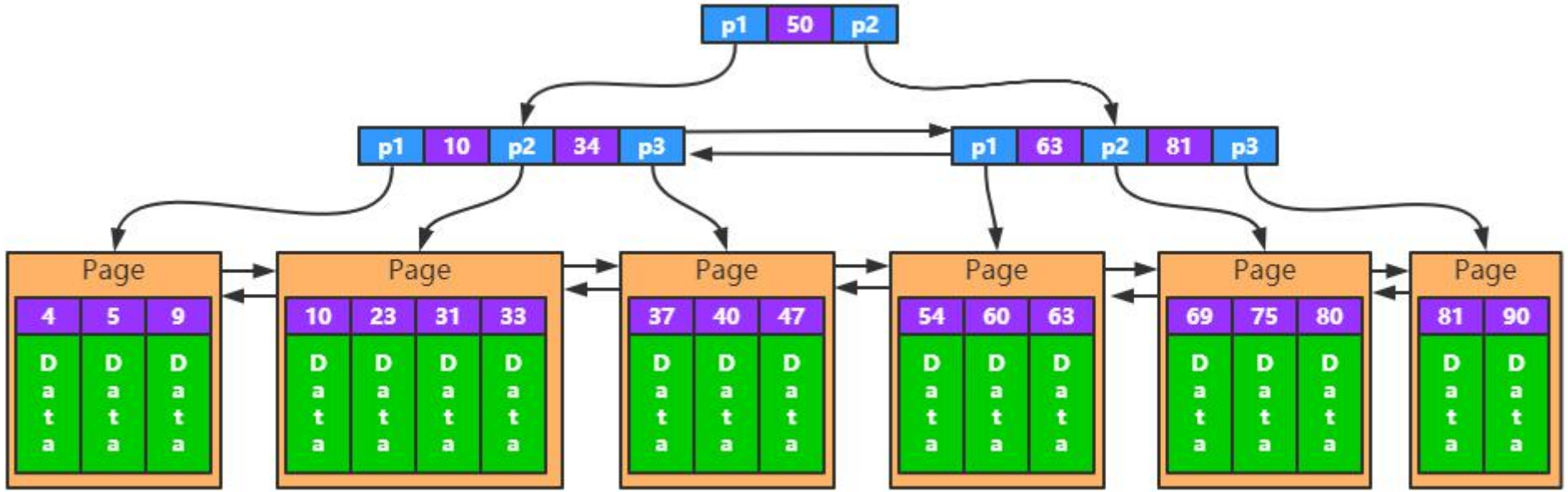
页内记录维护

- **顺序保证**
  - 物理连续
  - 逻辑连续
- 插入策略
- 页内查询



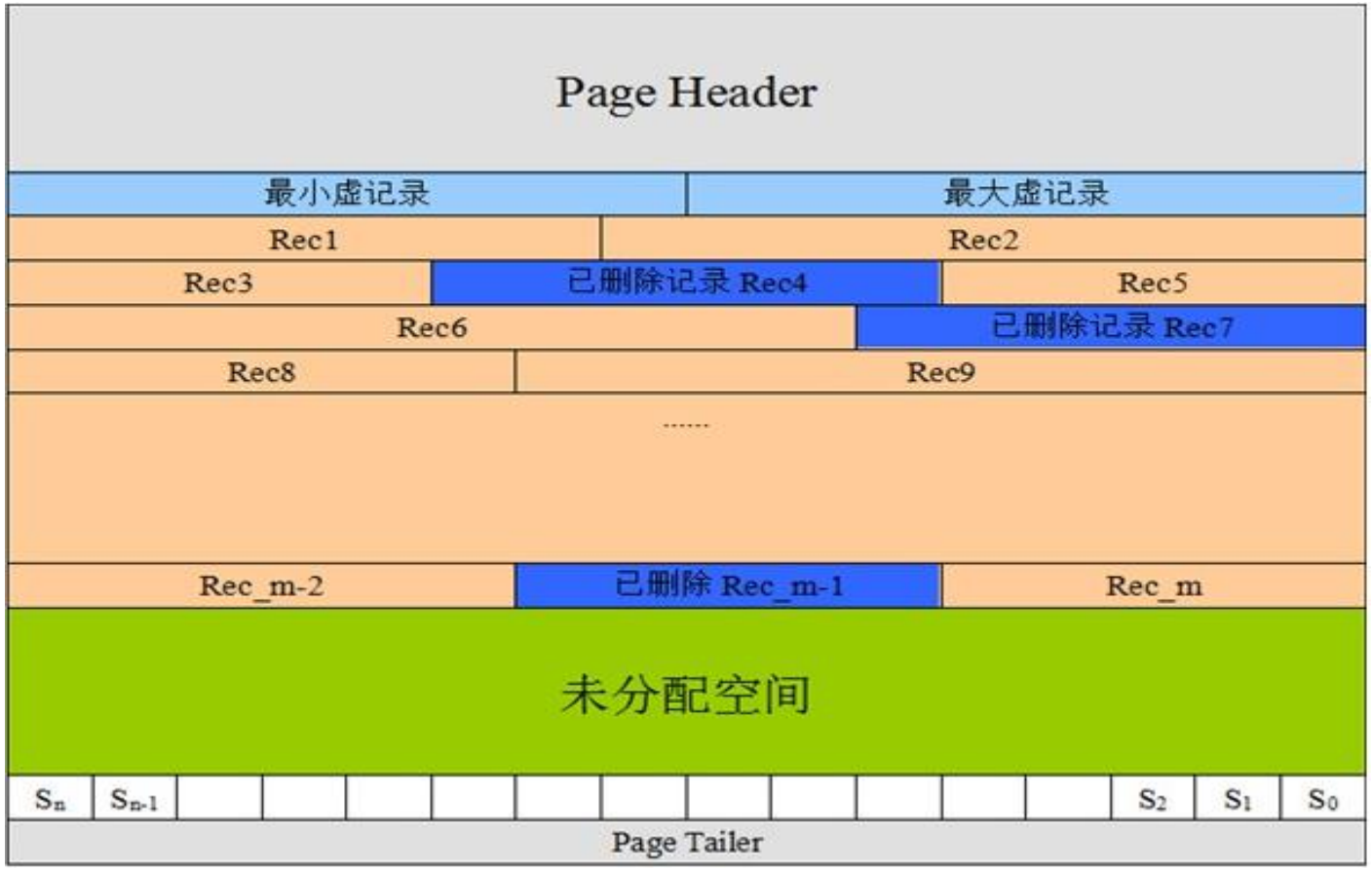
页内记录维护

- **顺序保证**
  - 物理连续
  - 逻辑连续
- 插入策略
- 页内查询



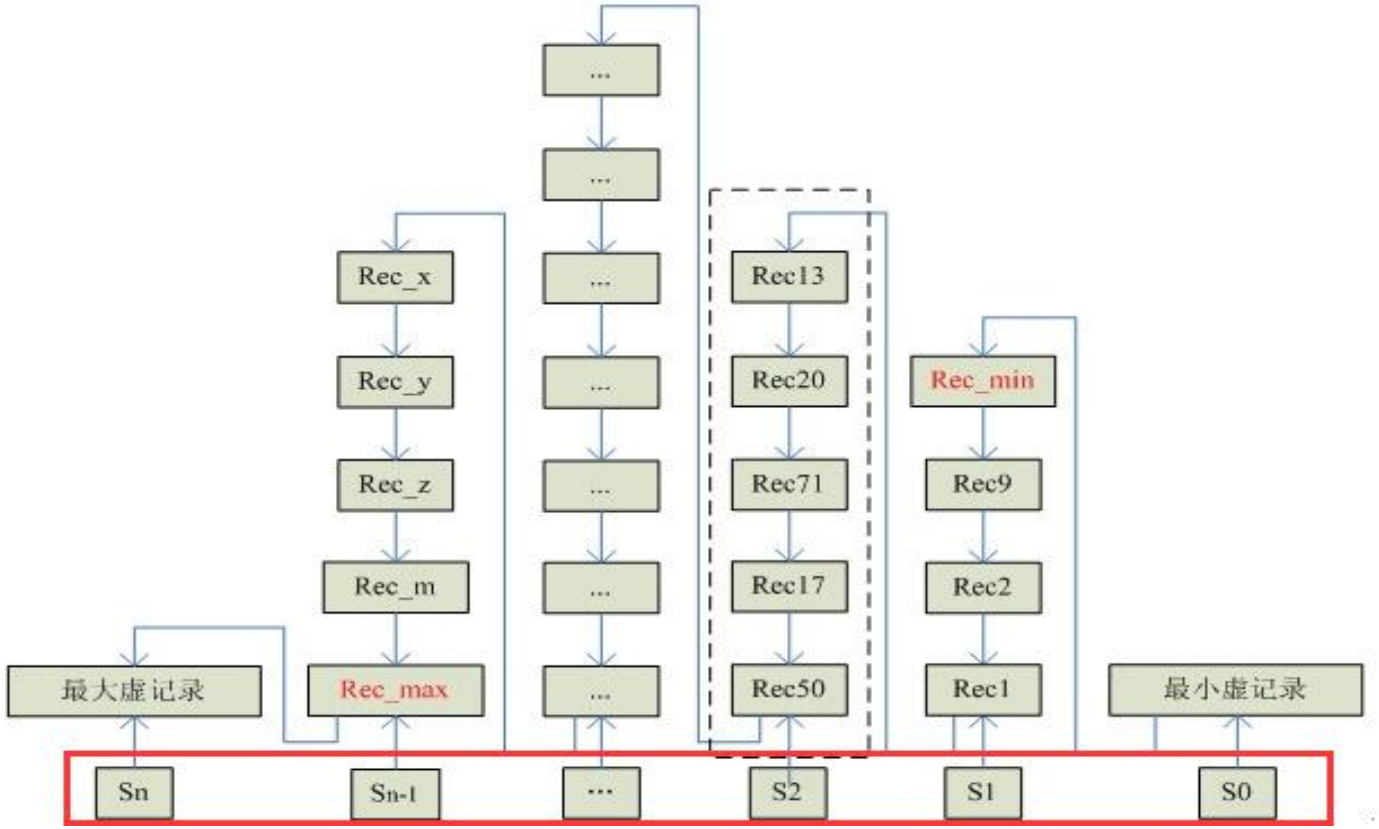
页内记录维护

- 顺序保证
- **插入策略**
  - 自由空间链表
  - 未使用空间
- 页内查询



页内记录维护

- 顺序保证
- 插入策略
- **页内查询**
  - 遍历
  - 二分查找

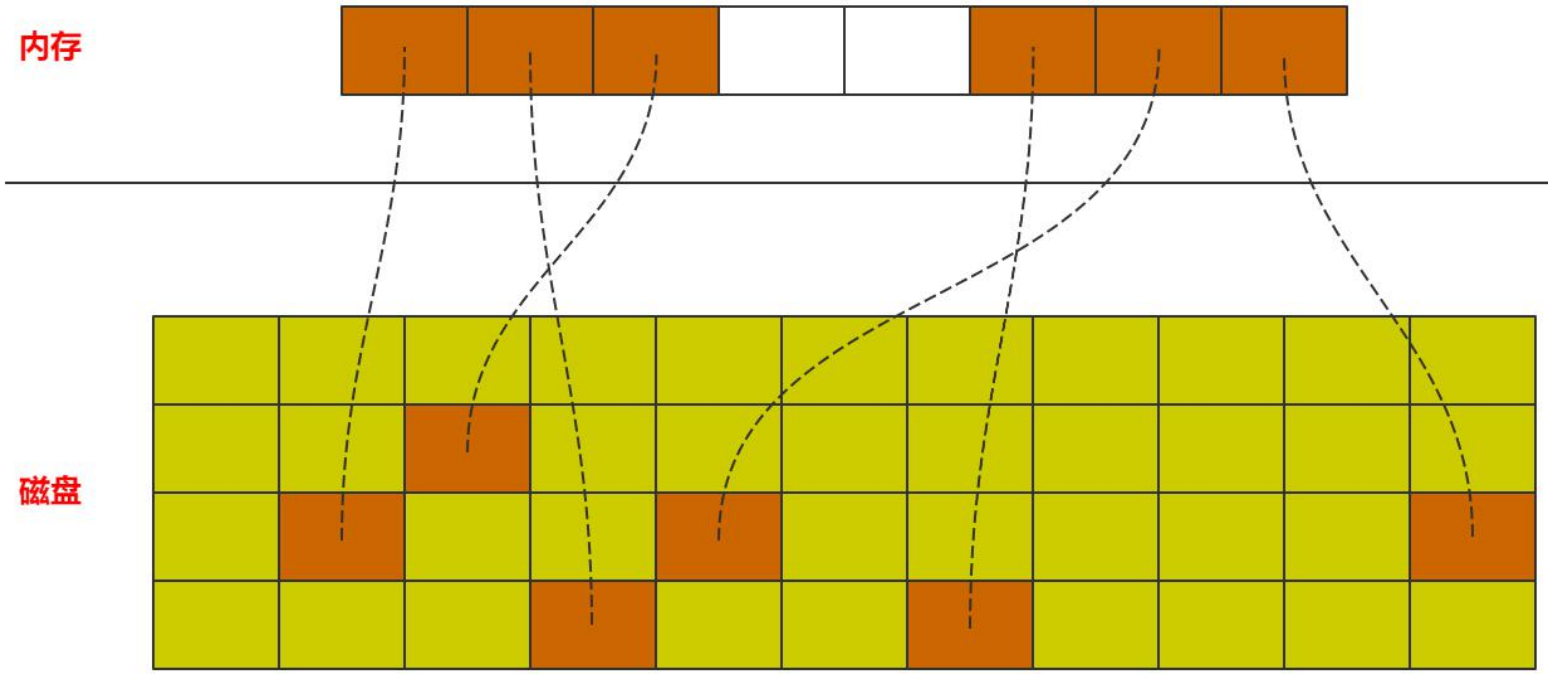


NX 奈学教育

02.MySQL InnoDB存储引擎内存管理

InnoDB内存管理

- 预分配内存空间
- 数据以页为单位加载
- 数据内外存交换



InnoDB内存管理—技术点

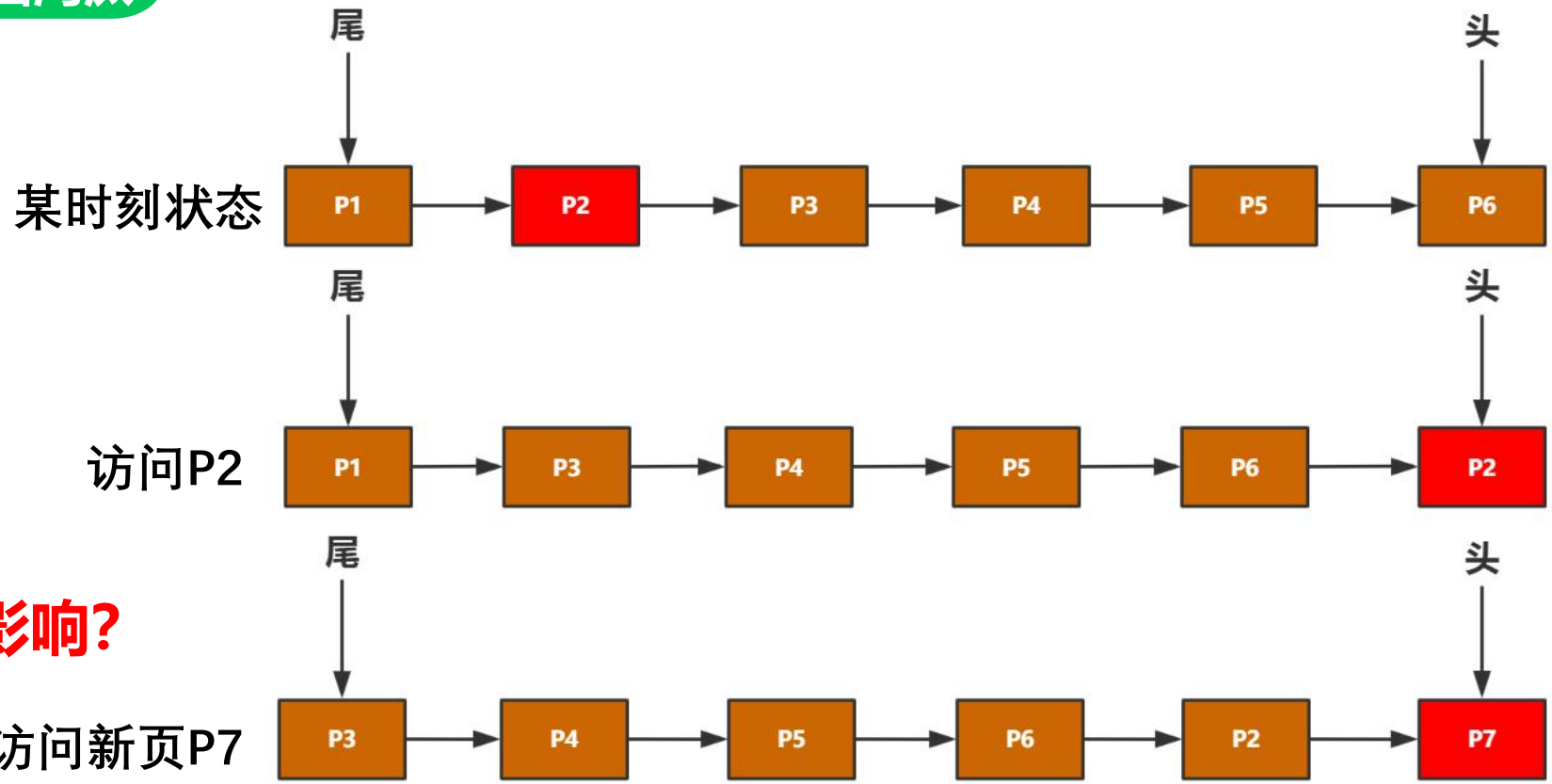
- 内存池
- 内存页面管理
  - 页面映射
  - 页面数据管理
- 数据淘汰
  - 内存页都被使用
  - 需要加载新数据

InnoDB内存管理—页面管理

- 空闲页
- 数据页
- 脏页

MySQL内存管理—页面淘汰

- LRU



思考：
全表扫描对内存的影响？

MySQL内存管理—页面淘汰

解决问题

- 避免热数据被淘汰

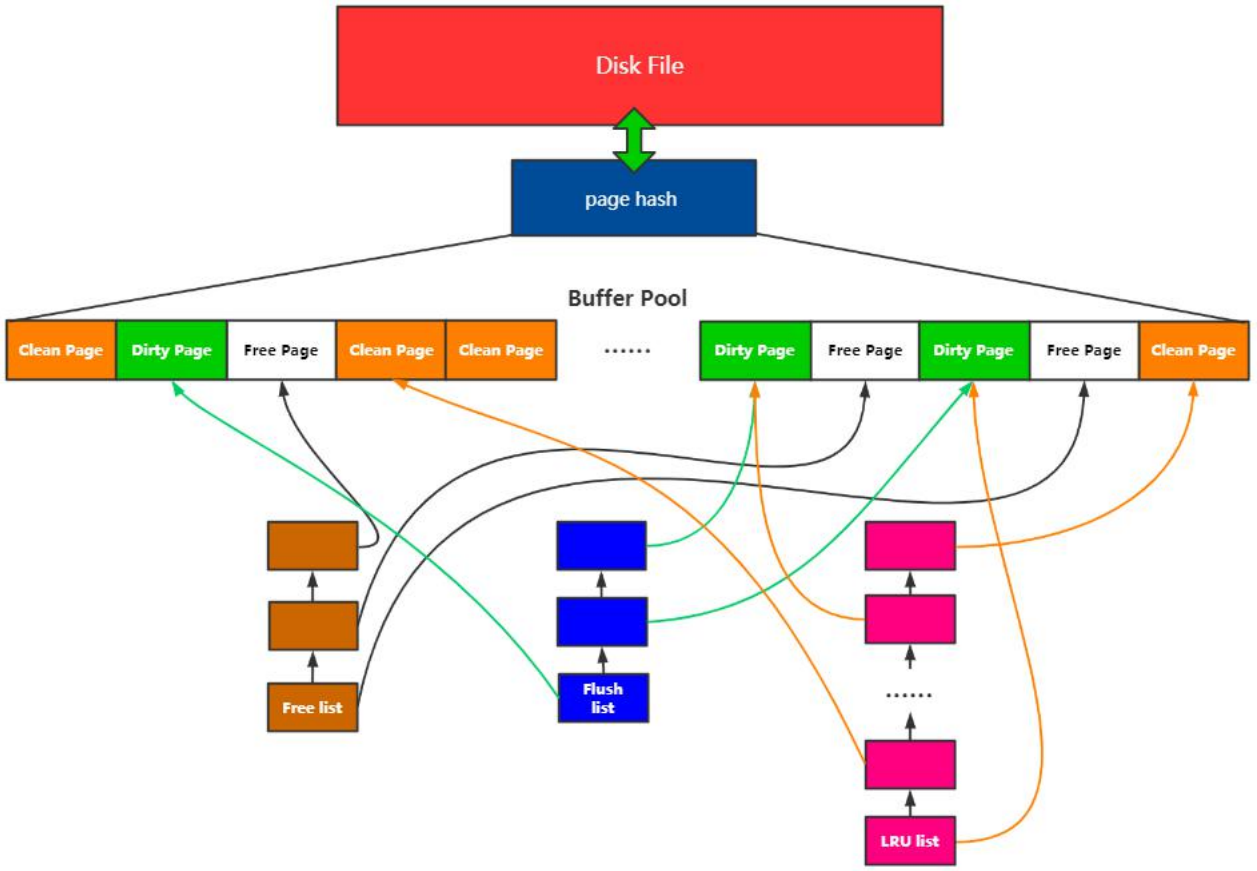
思路

- **访问时间 + 频率?**
- **两个LRU表?**

MySQL是如何解决的.....

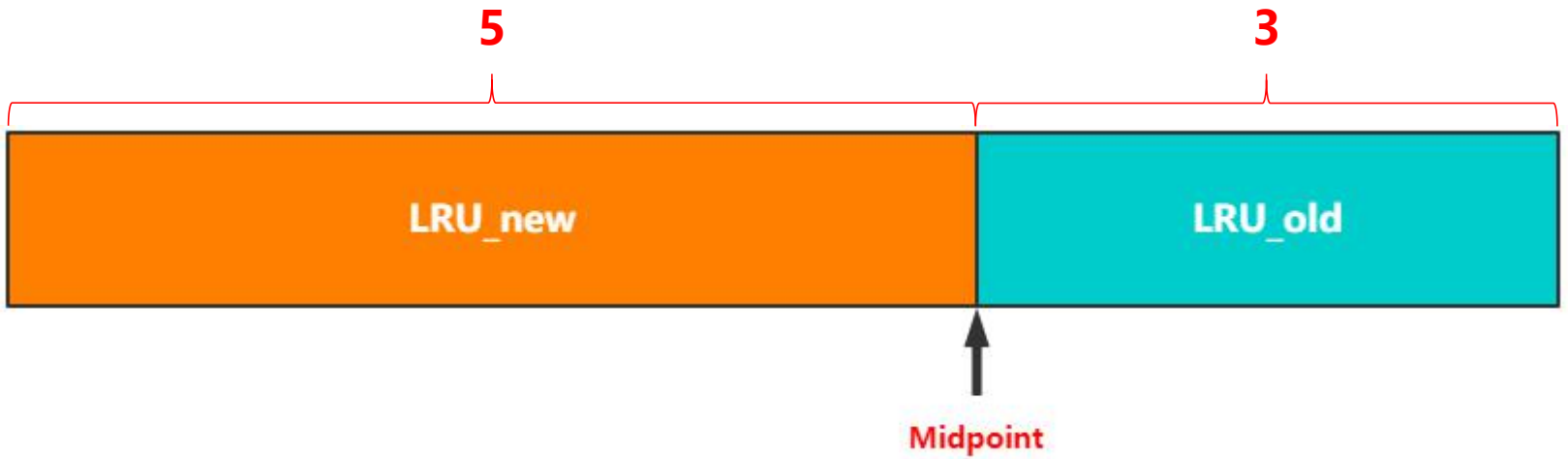
MySQL内存管理

- Buffer Pool
  - 预分配的内存池
- Page
  - Buffer Pool的最小单位
- Free list
  - 空闲Page组成的链表
- Flush list
  - 脏页链表
- Page hash 表
  - 维护内存Page和文件Page的映射关系
- LRU
  - 内存淘汰算法



MySQL内存管理—LRU

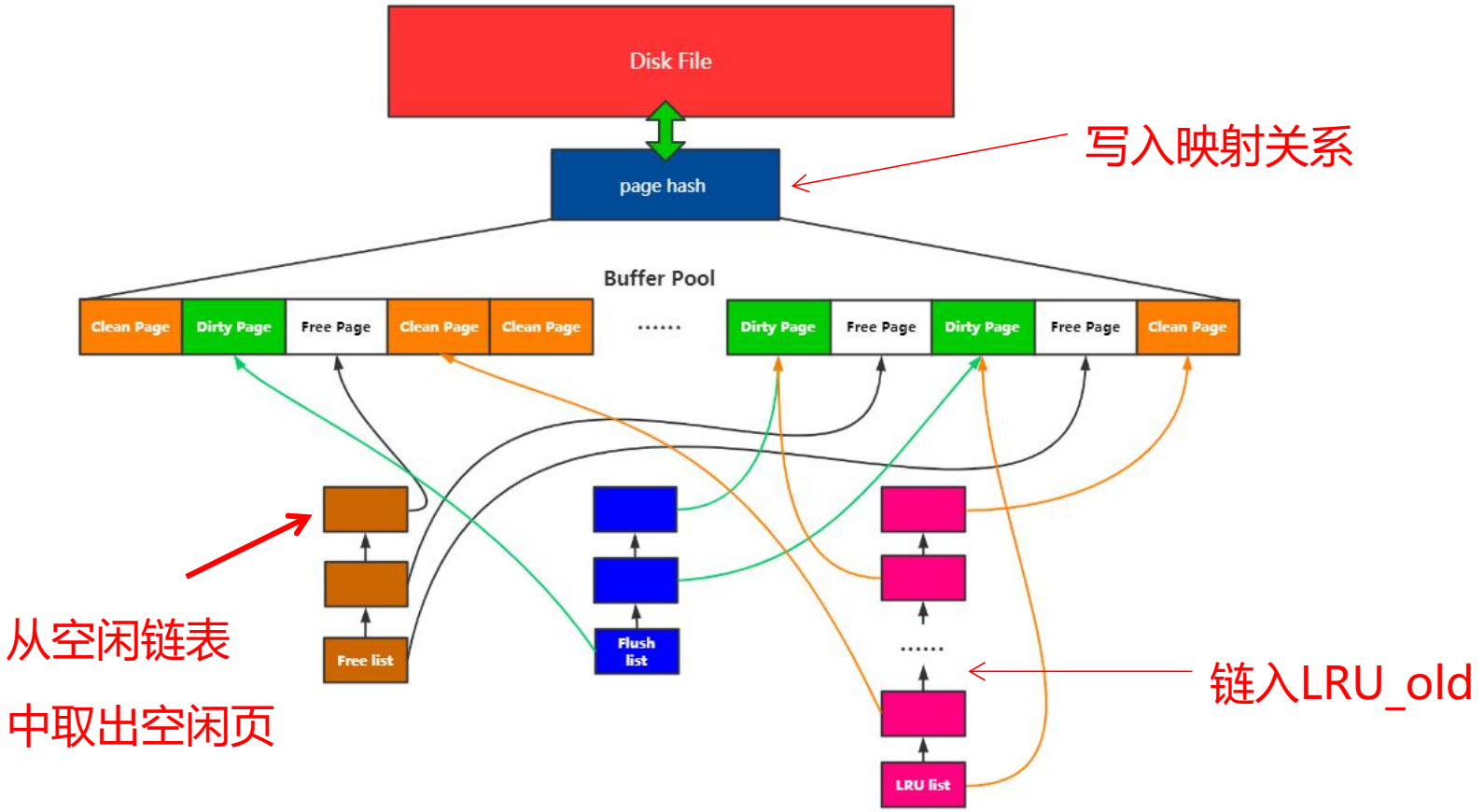
- LRU_new
- LRU_old
- Midpoint



冷热分离

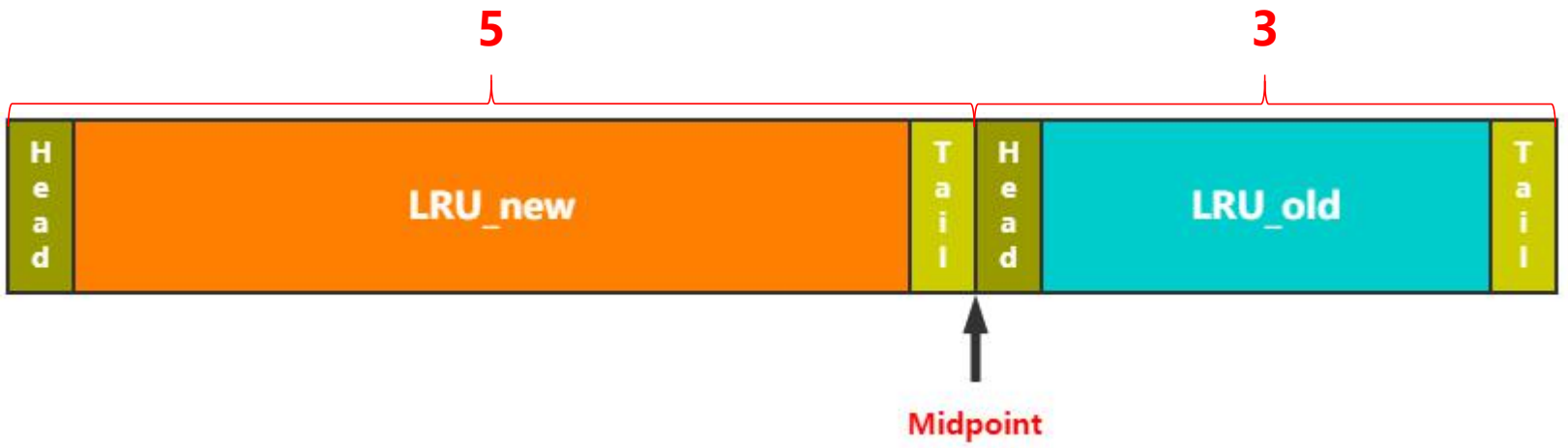
MySQL内存管理—LRU

- **页面装载**
  - 磁盘数据到内存
- 页面淘汰
- 位置移动
- LRU_new的操作



MySQL内存管理—LRU

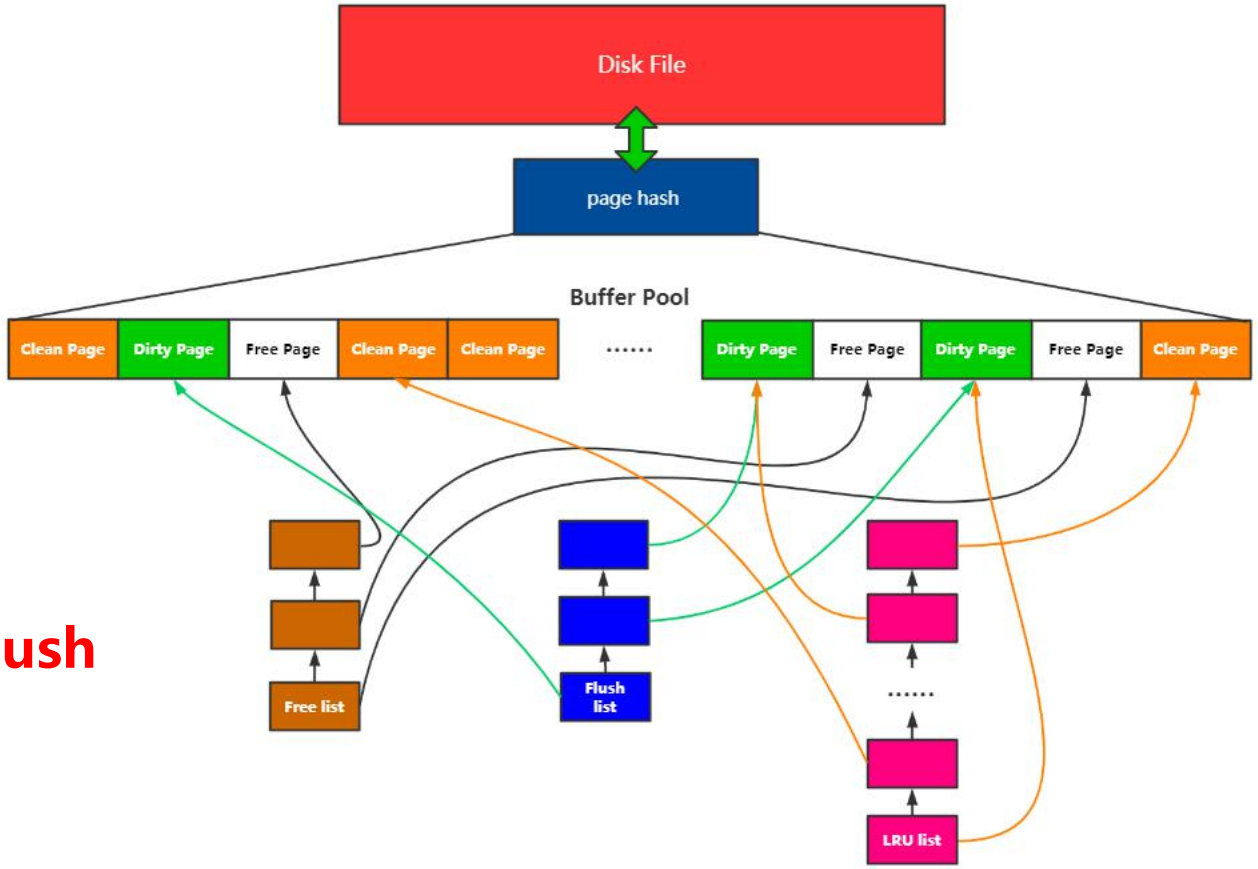
- **页面装载**
  - 磁盘数据到内存
- 页面淘汰
- 位置移动
- LRU_new的操作



没有空闲页怎么办?

MySQL内存管理—LRU

- **页面装载**
  - 磁盘数据到内存
- 页面淘汰
- 位置移动
- LRU_new的操作



Free list中取 > LRU中淘汰 > LRU Flush

MySQL内存管理—LRU

- 页面装载
- **页面淘汰**
  - LRU尾部淘汰
  - Flush LRU淘汰

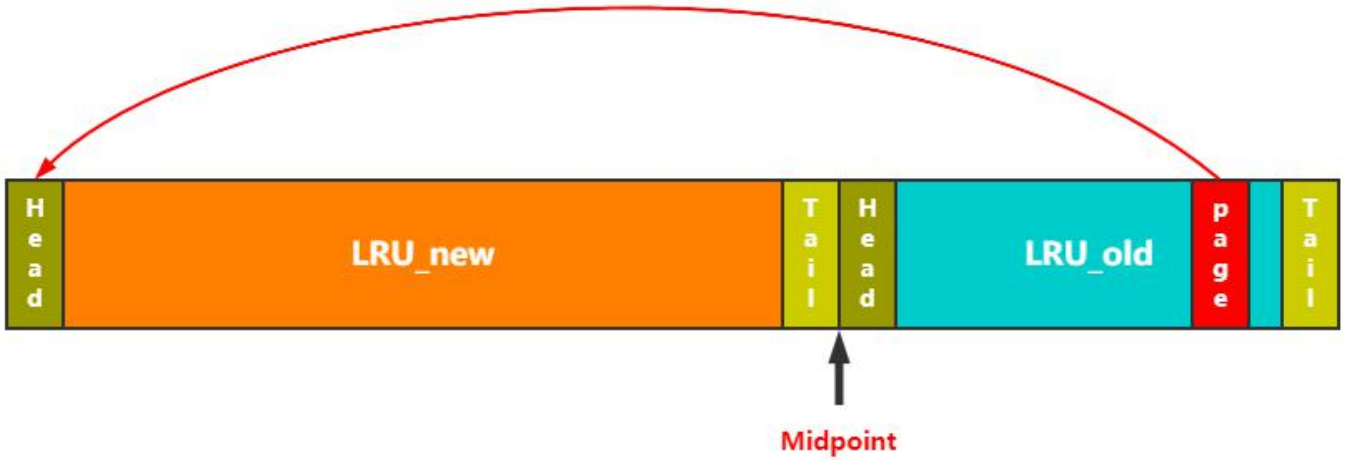
LRU链表中将第一个脏页刷盘并“释放”

- 位置移动
- LRU_new的操作

放到LRU尾部？直接放FreeList？

MySQL内存管理—LRU

- 页面装载
- 页面淘汰
- 位置移动
  - **old 到 new**
  - new 到 old
- LRU_new的操作

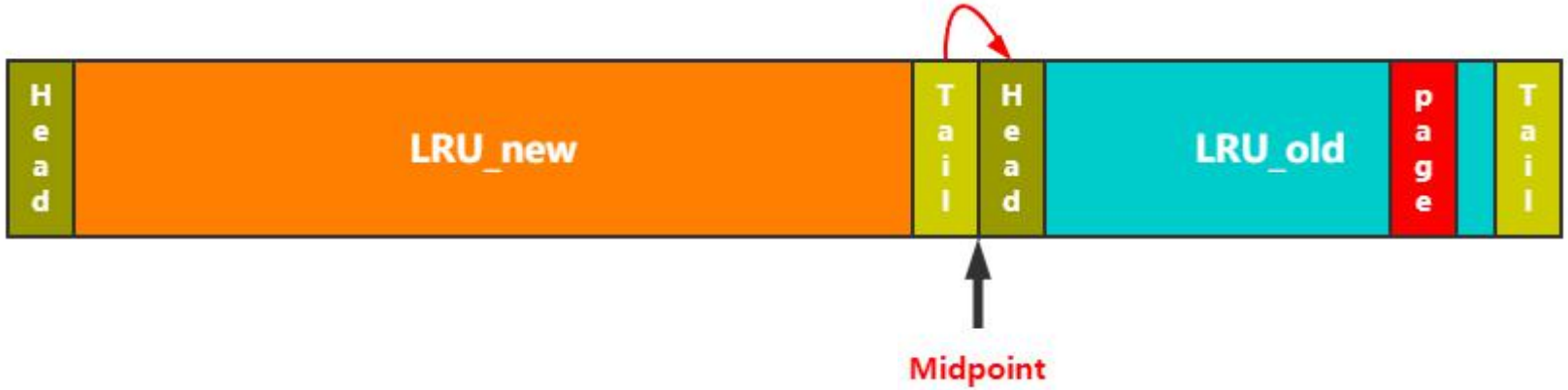


思考：移动时机

innodb_old_blocks_time
old区存活时间，大于此值，有机会进入new区

MySQL内存管理—LRU

- 页面装载
- 页面淘汰
- 位置移动
  - old 到 new
  - **new 到 old**
- LRU_new的操作



Midpoint：指向5/8位置

MySQL内存管理—LRU

- 页面装载
- 页面淘汰
- 位置移动
- **LRU_new的操作**

链表操作效率很高，有访问移动到表头？

Lock！！！

MySQL设计思路：减少移动次数

两个重要参考：1、freed_page_clock：Buffer Pool淘汰页数

2、LRU_new长度1/4

当前freed_page_clock - 上次移动到Header时freed_page_clock

>

LRU_new长度1/4

NX 奈学教育

03.MySQL事务实现原理拆解以及设计深度剖析

MySQL事务基本概念

- 事务特性
- 并发问题
- 隔离级别

MySQL事务基本概念

- **事务特性**
  - A（Atomicity原子性）：全部成功或全部失败
  - I（Isolation隔离性）：并行事务之间互不干扰
  - D（Durability持久性）：事务提交后，永久生效
  - C（Consistency一致性）：通过AID保证
- 并发问题
- 隔离级别

MySQL事务基本概念

- 事务特性
- **并发问题**
  - **脏读(Drity Read)**：读取到未提交的数据
  - **不可重复读(Non-repeatable read)**：两次读取结果不同
  - **幻读(Phantom Read)**：select 操作得到的结果所表征的数据状态无法支撑后续的业务操作
- 隔离级别

MySQL事务基本概念

- 事务特性
- 并发问题
- **隔离级别**
  - **Read Uncommitted**（读取未提交内容）：最低隔离级别，会读取到其他事务未提交的数据，脏读；
  - **Read Committed**（读取提交内容）：事务过程中可以读取到其他事务已提交的数据，不可重复读；
  - **Repeatable Read**（可重复读）：每次读取相同结果集，不管其他事务是否提交，幻读；
  - **Serializable**（串行化）：事务排队，隔离级别最高，性能最差；

MySQL事务实现原理

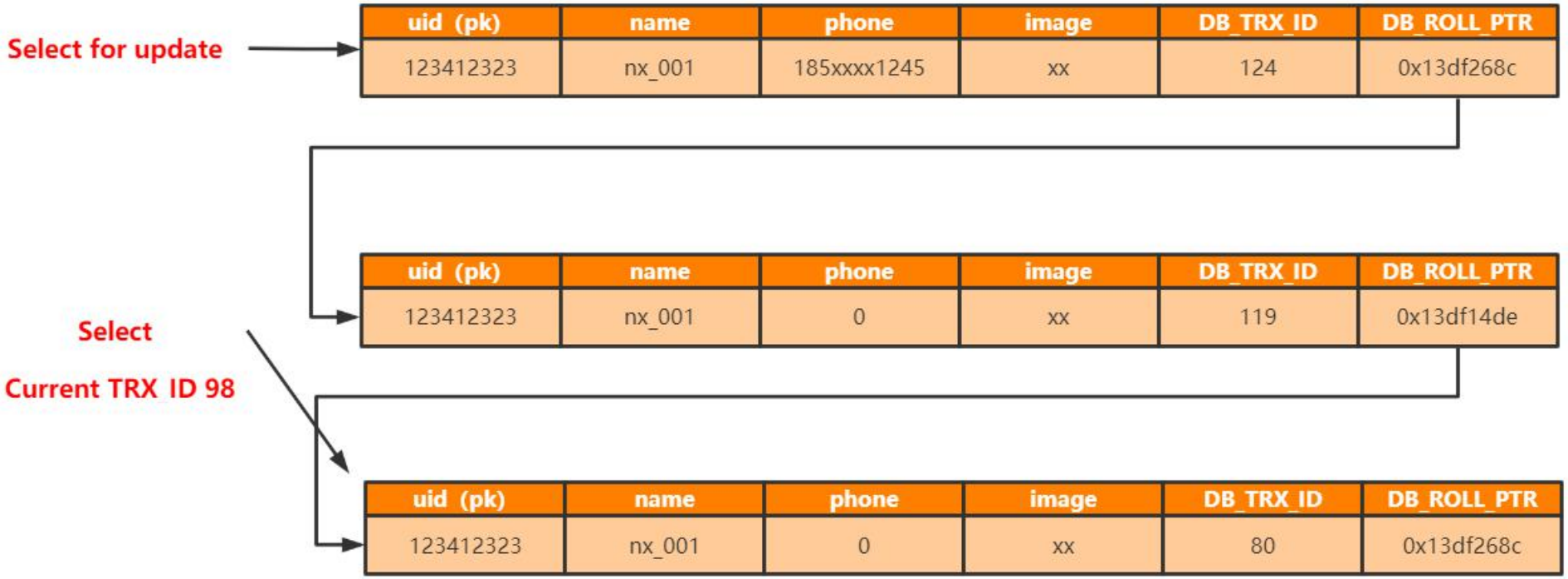
- MVCC
- undo log
- redo log

MySQL事务实现原理

- **MVCC**
  - 多版本并发控制
  - 解决读-写冲突
  - 隐藏列
- undo log
- redo log

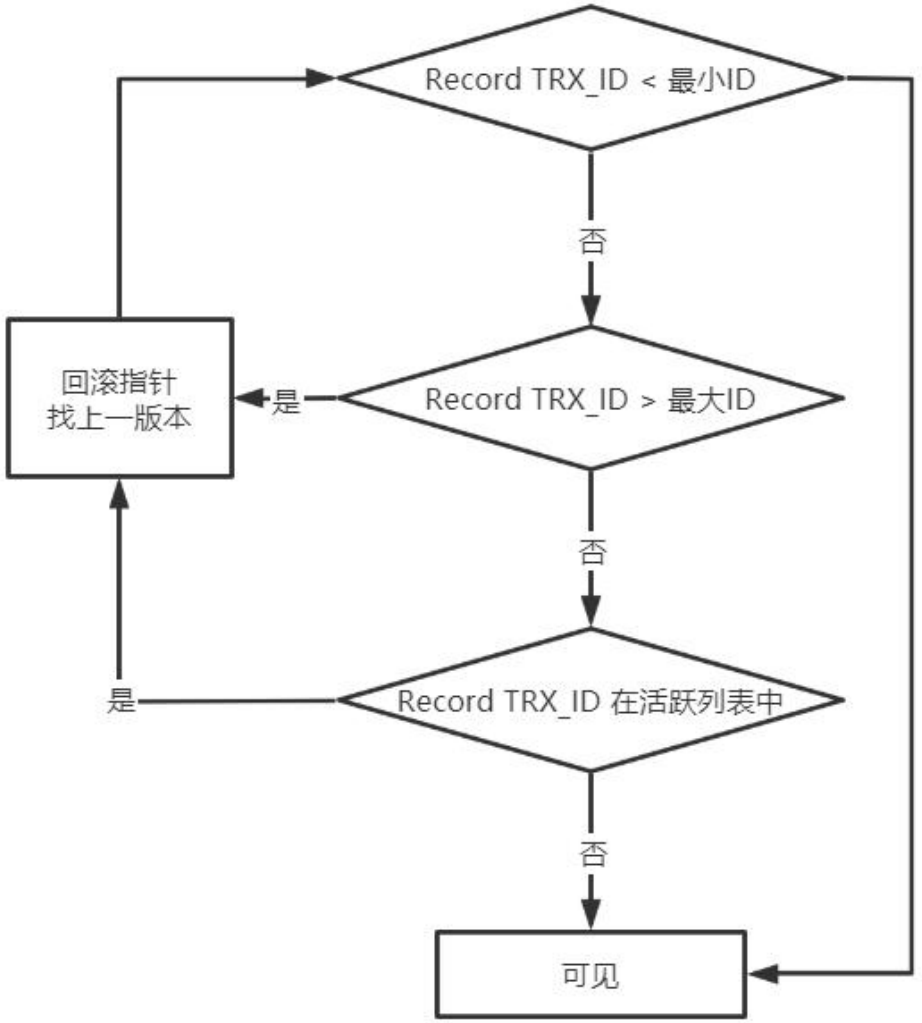
MySQL—MVCC

- 当前读
- 快照读



MySQL—MVCC

- 可见性判断
  - 创建快照这一刻，还未提交的事务；
  - 创建快照之后创建的事务；
- Read View
  - 快照读 活跃事务列表
  - 列表中最小事务ID
  - 列表中最大事务ID



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

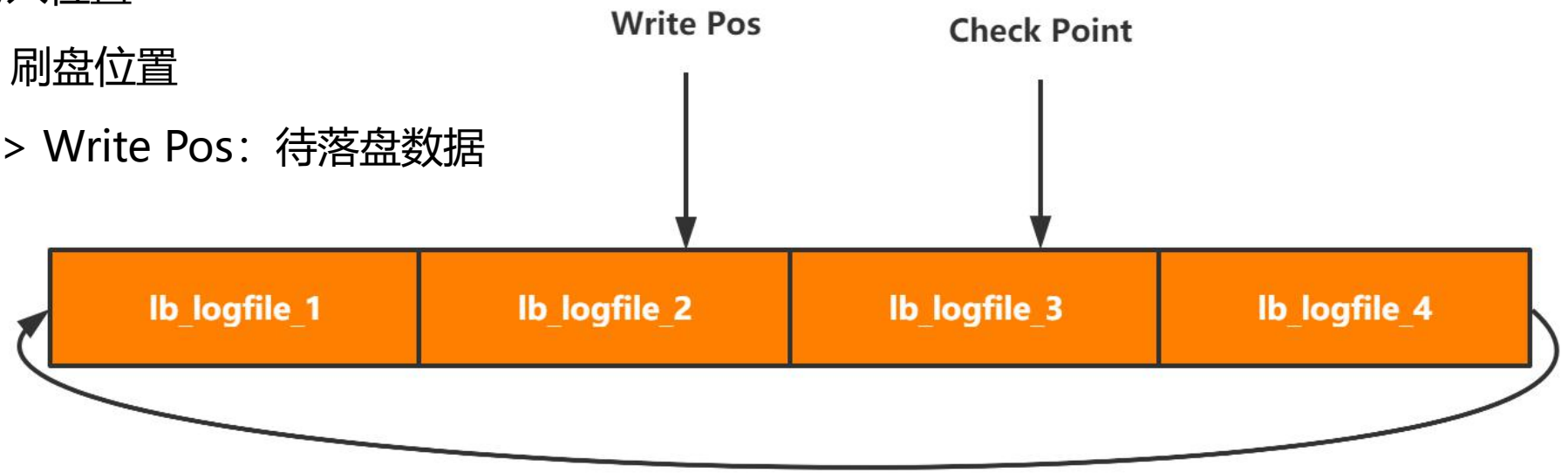
为什么InnoDB count（*）这么慢?

MySQL事务实现原理

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

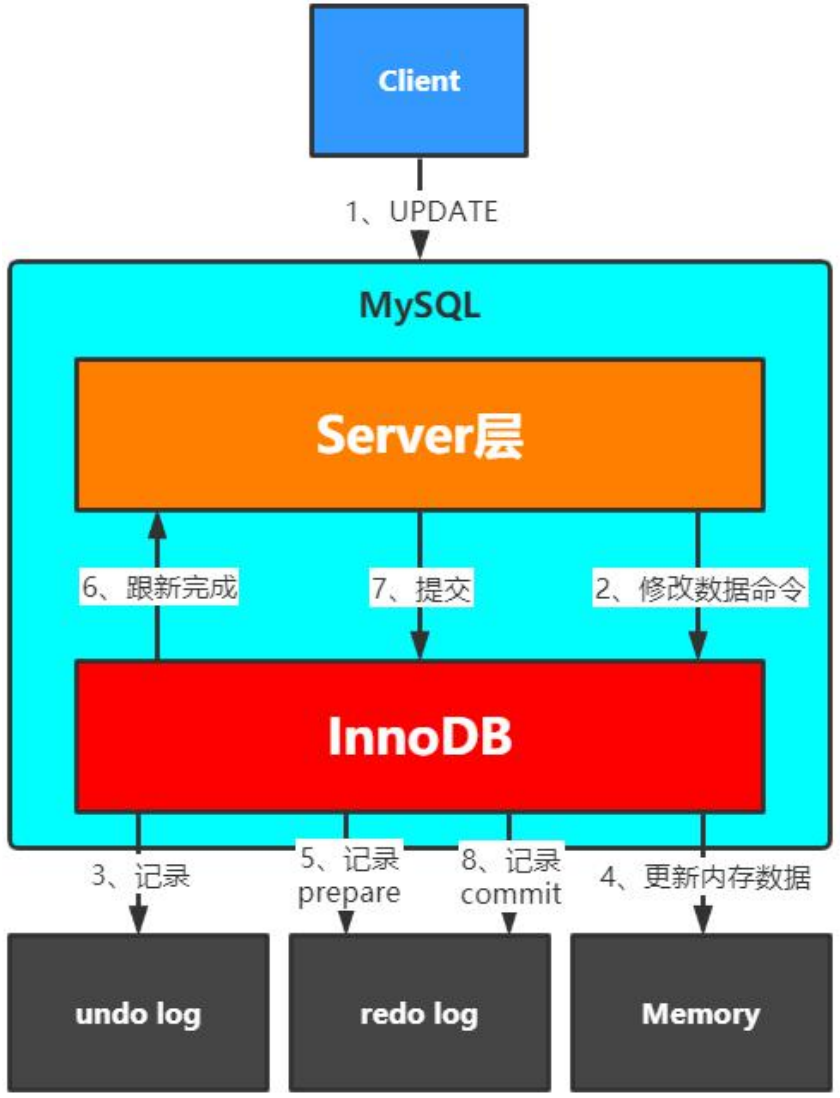
MySQL—redolog

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



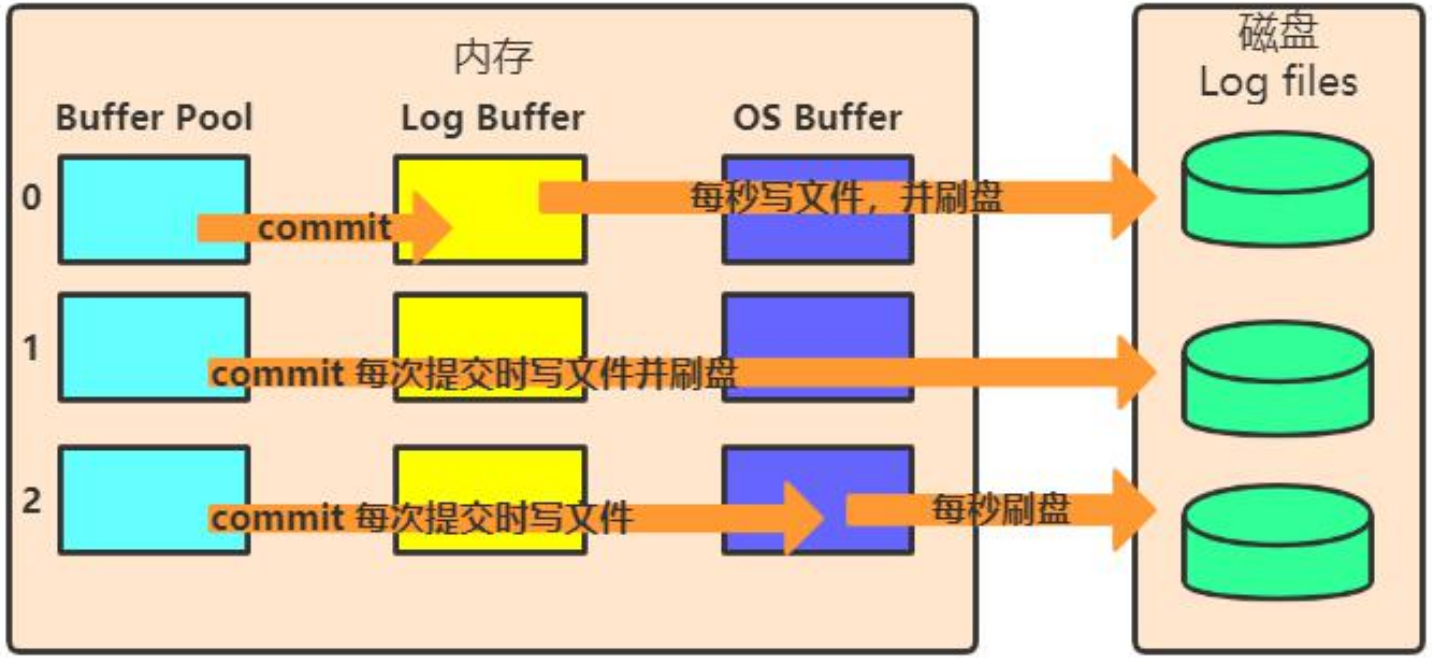
MySQL—redolog

- 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

04.MySQL锁实现原理拆解以及设计深度剖析

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
- 间隙锁
- 表级锁

所有当前读加排他锁，都有哪些是当前读?

SELECT FOR UPDATE

UPDATE

DELETE

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

delete from user where phone = 134

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

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

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

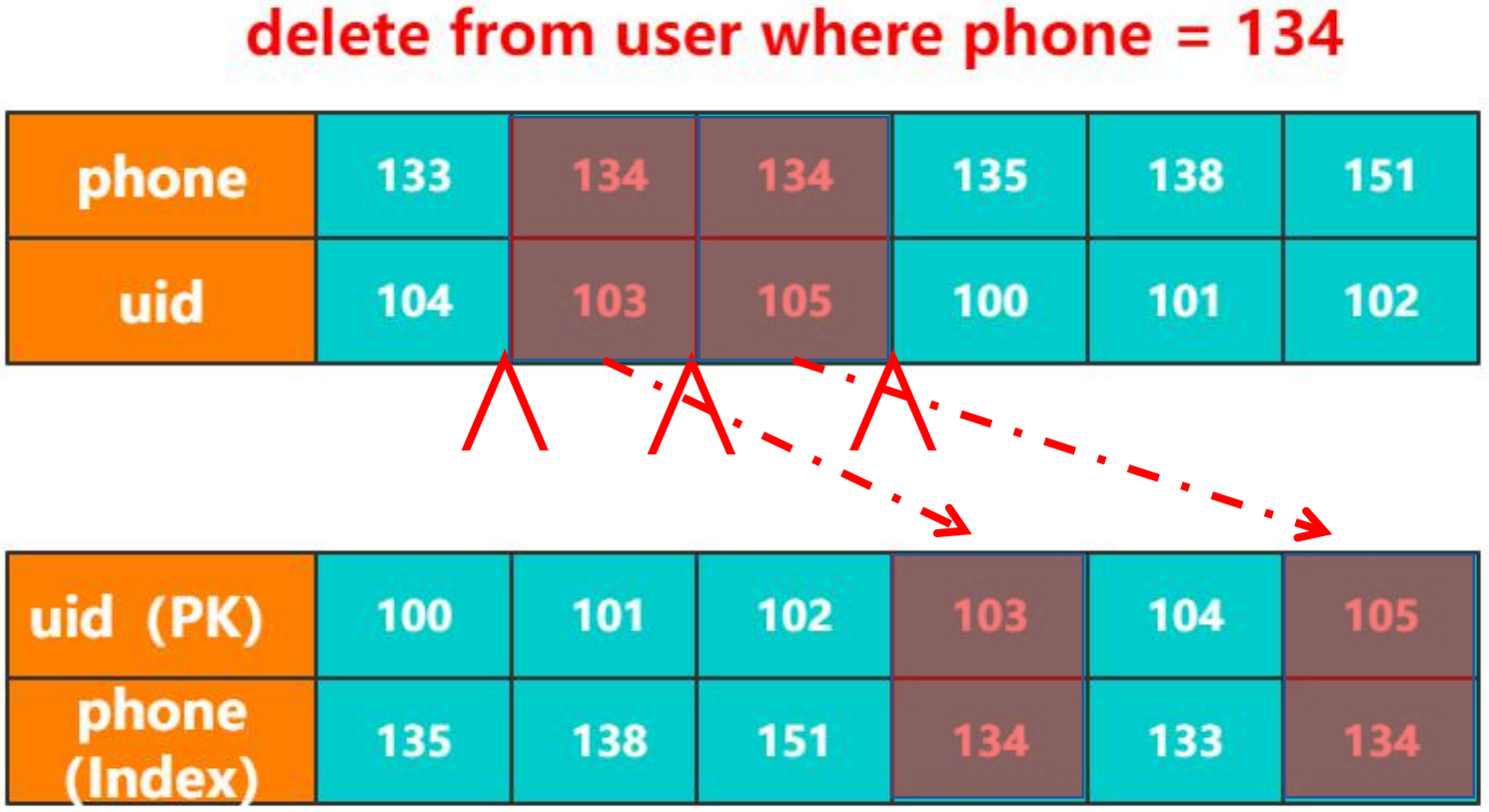
phone	133	134	134	135	138	151
uid	104	103	105	100	101	102

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

RC隔离级别，非唯一索引

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



RR隔离级别，非唯一索引

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

两次当前读之间，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

delete from user where phone = 134

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

1 2 3

1、[131,140]~[134,130]

2、[134,130]~ [134,150]

3、[134,150]~[137,100]

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

InnoDB锁种类

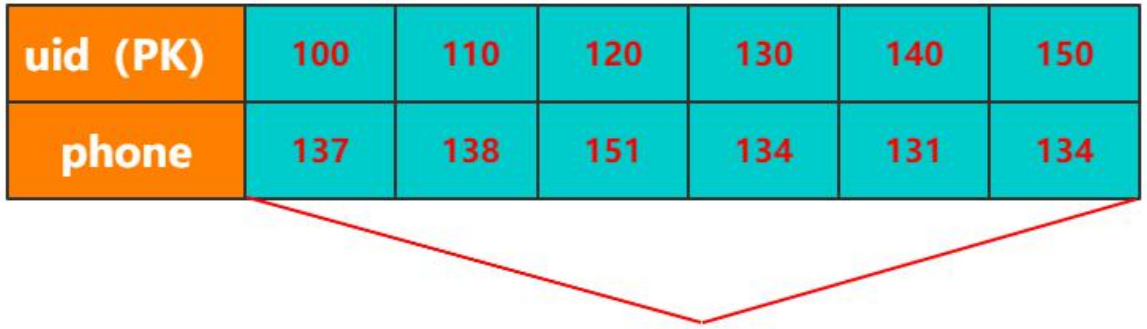
- 行级锁
- 间隙锁
- **表级锁**
  - lock tables
  - 元数据锁（meta data lock，MDL）

InnoDB锁种类

- 行级锁
- 间隙锁
- **表级锁**
  - 全表扫描“表锁”

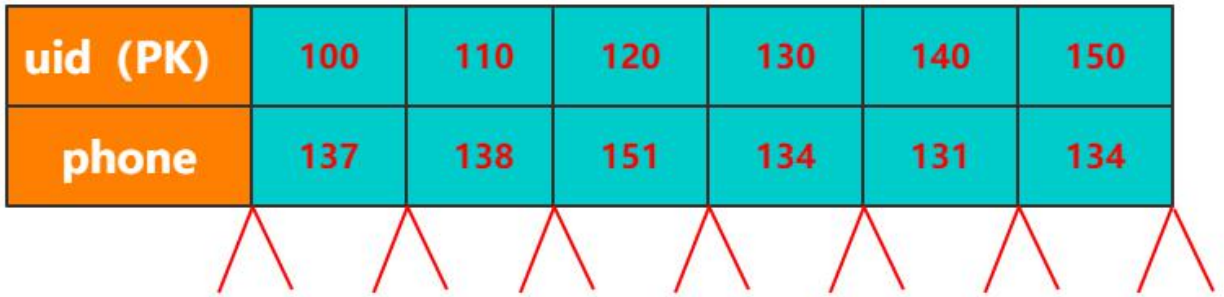
delete from user where phone = 134

RC

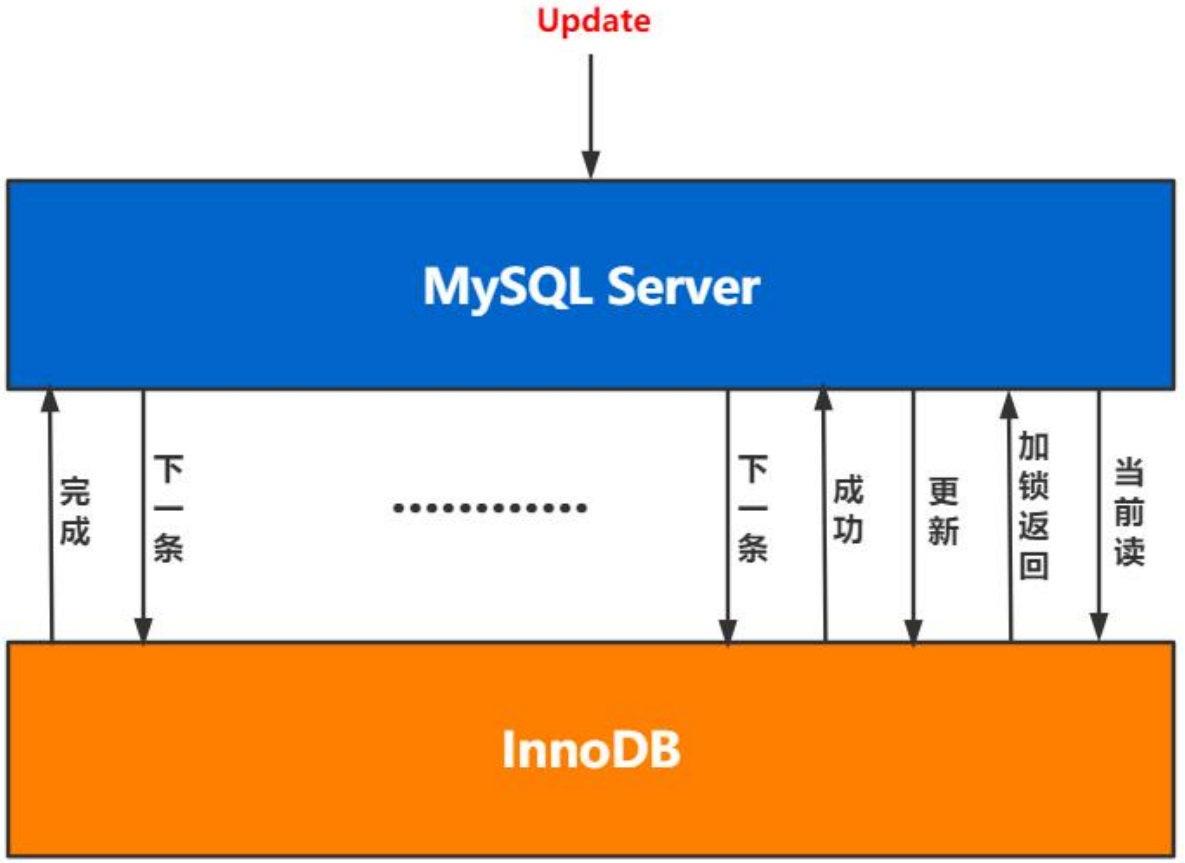


所有记录加锁后返回，然后由MySQL Server层进行过滤

RR



InnoDB加锁过程

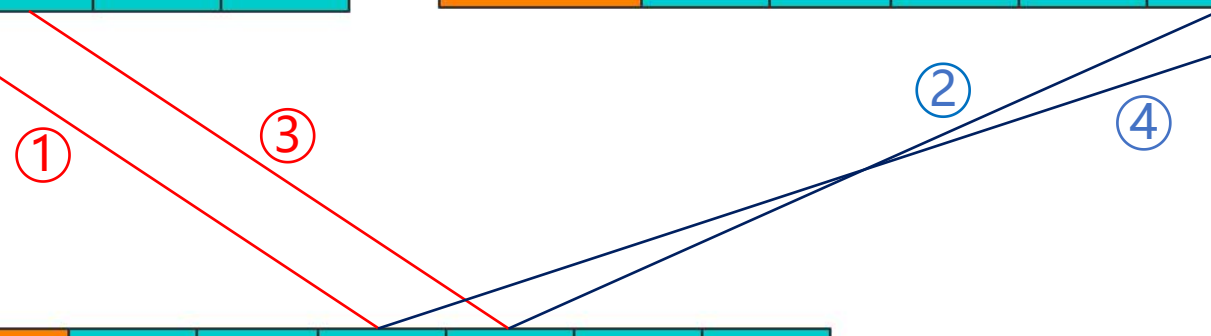


InnoDB加锁过程

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

扫码参与问卷调查
给我们一个活动反馈吧~

期待每一次课都能与你相遇