MVCC(Oracle/Innodb/PG)

网易杭研院——何登成

Outline

- MVCC介绍
- MVCC实现
  - Oracle
  - InnoDB
  - PostgreSQL
- 总结

MVCC简介

- 何谓MVCC
  - 多版本并发控制(MultiVersion Concurrency Control)
- MVCC的优势
  - 高并发
    - 读写不相互阻塞(读可见版本)
    - 不同数据库的可见版本定义？
  - 低加锁开销
    - 读不加锁
    - OLTP应用，8(read)/2(write)
- 支持MVCC数据库
  - 现阶段几乎所有主流数据库
  - Oracle, DB2(since 9.7), SQL Server(since 2005), Mysql Innodb, Postgres …

MVCC实现

- 实现方式
  - 基于时间戳
    - Oracle
  - 基于事务ID
    - Innodb，Postgres
- 实现粒度
  - 页面级多版本
    - Oracle
  - 行级多版本
    - Innodb，Postgres

ORACLE MVCC

- 关键词
  - 基于时间戳(SCN)
  - 页面级多版本
  - 回滚段

- SCN(System Change Number)
  - 数据库逻辑时间戳，只增加，不减小
  - 事务commit，产生新的SCN
  - SCN格式：0xFFFF.FFFFFFF
  - SCN与系统时间可以相互转换
    - Select scn_to_timestamp(1999999) from dual;
    - Select timestamp_to_scn(sysdate) from dual;

Oracle MVCC

- 页面级多版本
  - 回滚段(rollback segment)
    - 修改记录，记录的前镜像(undo)写入回滚段
    - 记录的旧版本，通过读取回滚段，获得undo，回滚获得
  - 页面级SCN
    - 页头结构中，当前页面最新提交事务的SCN (非确切说法)
    - 延时块清除(Deferred Block Cleanout)
    - Row Level SCN？
  - 页面级Undo
    - 将页面回滚到可见版本(非记录)

Oracle MVCC

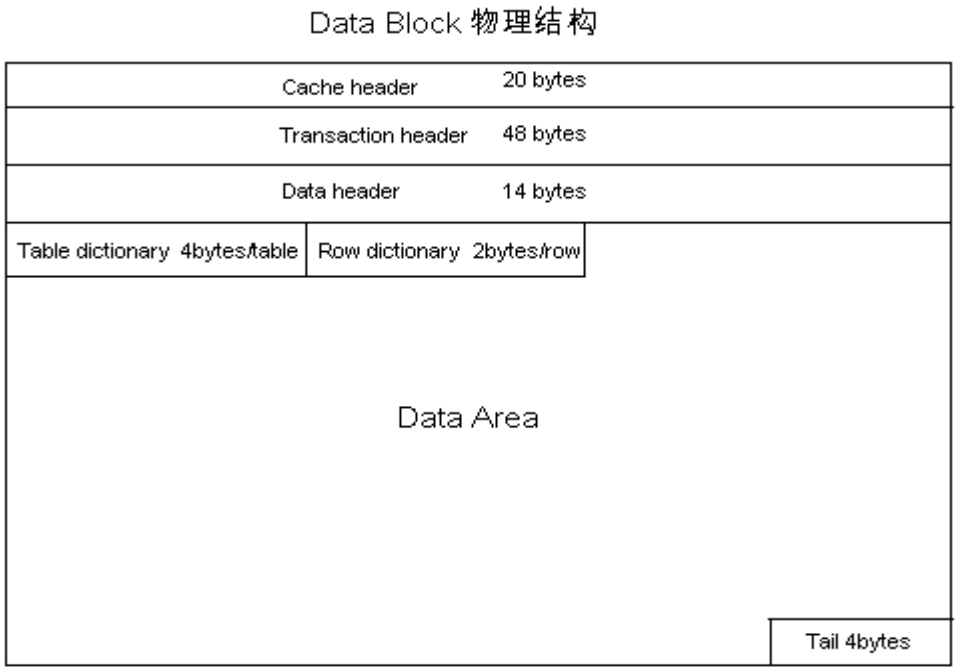
- 可见性判断
  - 给定查询SCN，回滚页面获得可见CR Block (Consistent Read：一致读)

    1. CR Block的SCN小于查询SCN

    2. CR Block上无活跃事务

Oracle MVCC

- 页面结构

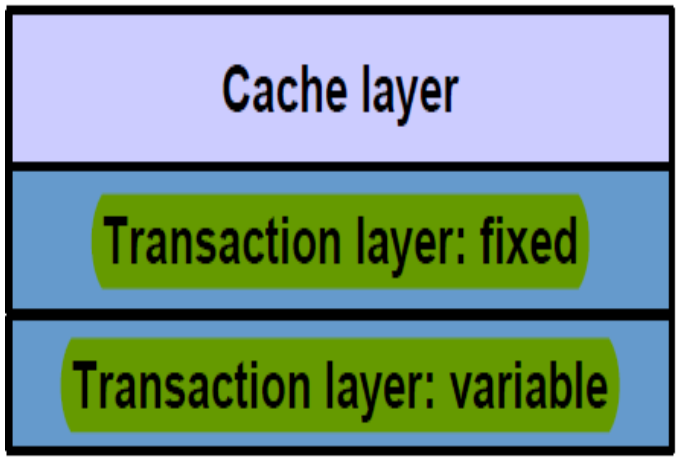


Oracle MVCC

- Data Block (cache layer)
  - Data block的第一部分，20 bytes
  - 构成
    - rdba(relative data block address)
      - 4 bytes; 前10 bits：file id；后22 bits：block id
      - 已知file Id，block id → dump block
    - scn
      - 标识block最新提交事务scn
        - » 延时块清除？
    - seq
      - 1 byte；同一scn下做的多次页面修改

Oracle MVCC

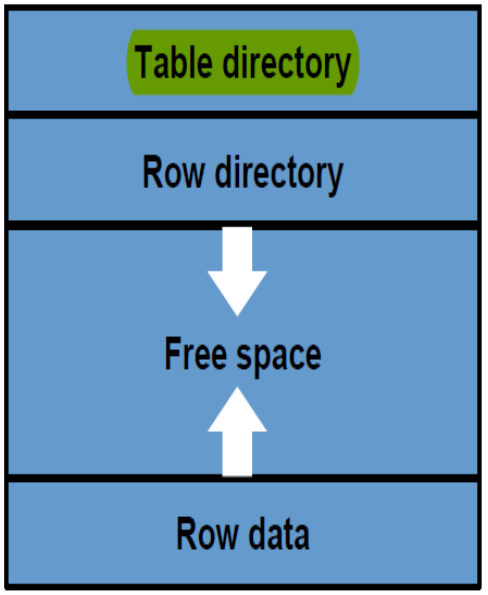
- Data Block (transaction layer)
  - Fixed部分
    - ITL数量(页面并发事务数)
      - INITRANS；MAXTRANS
    - 空闲页锁
    - Freelist指针
    - CSC： cleanout scn
  - Variable部分
    - ITLs(Interested Transaction List)




```
Itl       Xid                  Uba         Flag  Lck
Scn/Fsc
0x01   xid:  0x0005.000.00000805 uba: 0x00c02619.0304.01  ----  1
fsc 0x0000.00000000
```

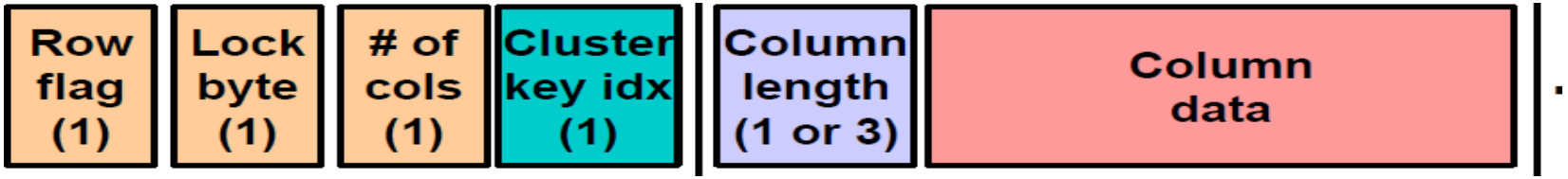
Oracle MVCC

- Data Block (Data layer)
  - Table directory
    - 聚簇表(cluster)? 作用？
  - Row directory
    - 行目录
  - Row data
    - 行数据



Oracle MVCC

- Row Format

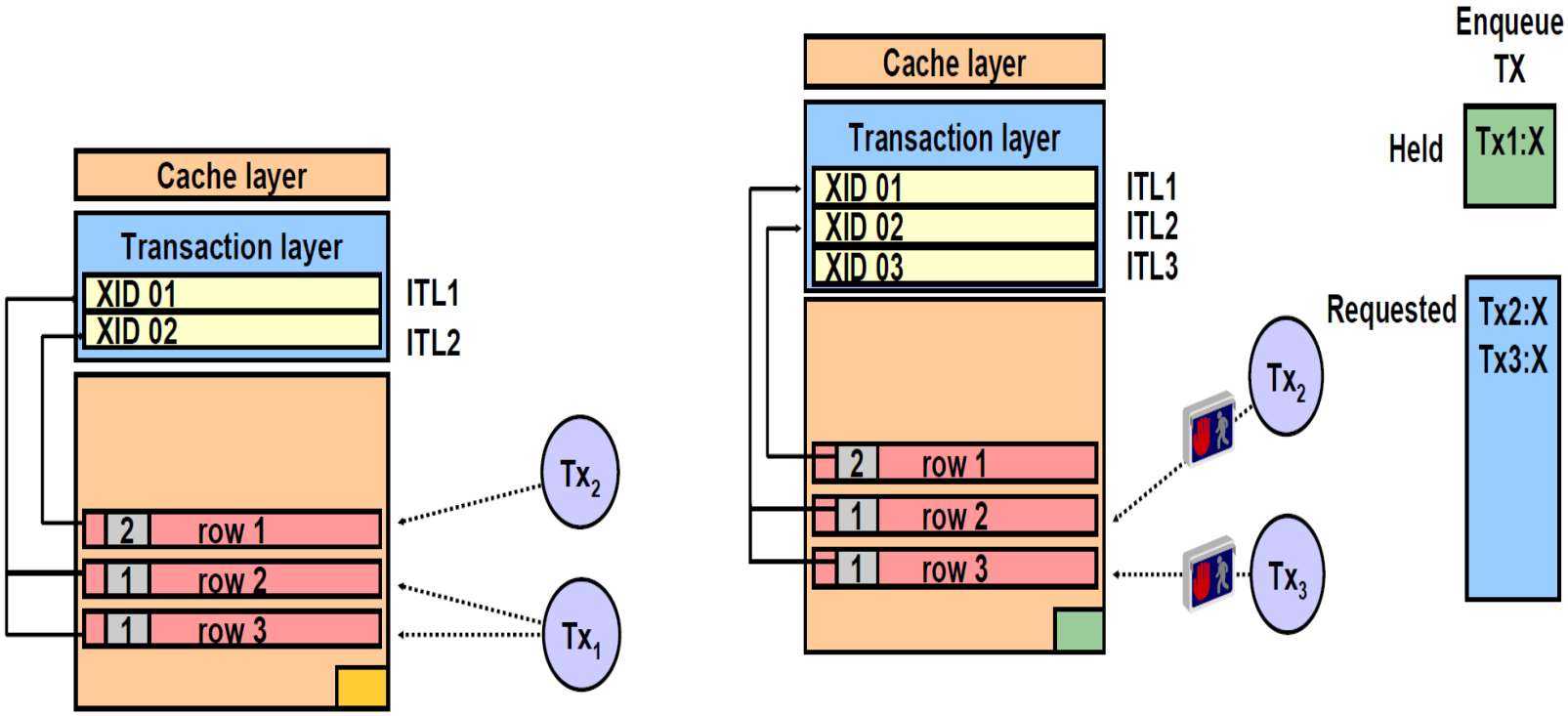


  - tl: total length
  - fb: flag byte
    - 行类型
    - KCHDFLPN
  - lb: lock byte
    - ITL number
  - cc: column count
    - NULL?

```
tab 0, row 0, @0x6f
tl: 6 fb: --H-FL-- lb: 0x0 cc: 1
col  0: [ 2]  c1 02
tab 0, row 1, @0x6d6
tl: 226 fb: --H-FL-- lb: 0x0 cc: 2
col  0: [ 2]  c1 03
col  1: [219] 6e 6f 4b 4c 4d ...
```

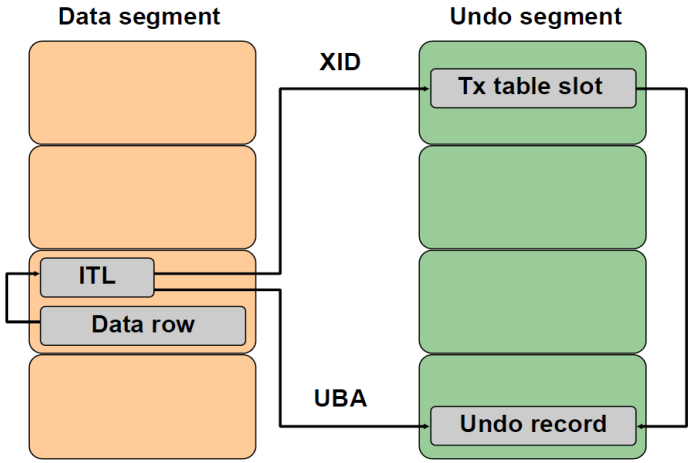
Oracle MVCC

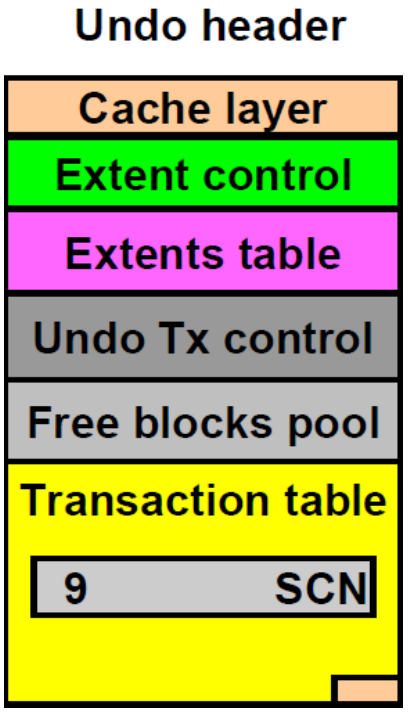
- 加锁/锁等待



Oracle MVCC

- 回滚段 & 事务





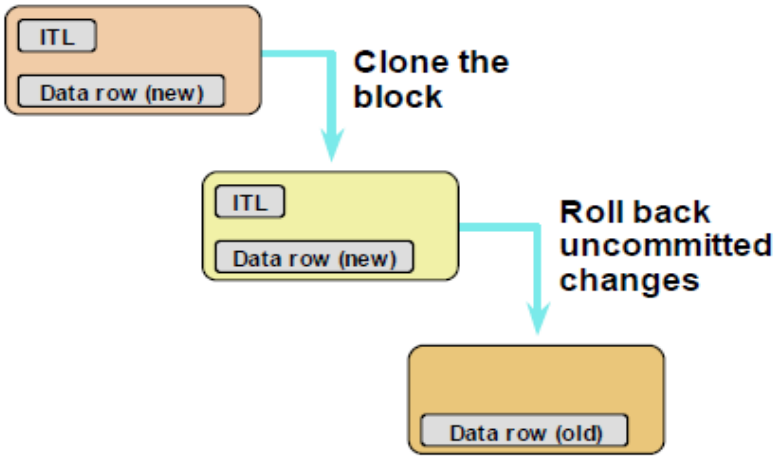
– 事务ID
  - 8 bytes　　xid: 0x0004.00e.0000449b
  - 0004：　回滚段号
  - 00e：　回滚段中的事务slot号
  - 0000449b：　事务槽被重用的次数

Oracle MVCC

- 块清除(Block Cleanout)
  - 清除什么？
    - 清除数据块上的事务操作：行锁；ITL；设置CSC；Block SCN ...
  - 何时清除
    - fast block cleanout
      - 记录修改块，提交时清除
      - 数量？Cache？日志？
    - deferred block cleanout(延时块清除)
      - 正确性保证：commit时更新回滚段头事务表
      - 操作：等待下次事务型操作(why？)访问此块
  - 为什么需要清除
    - 不清除时怎么做？

Oracle MVCC

- Undo
  - Block Undo
    - read current block
    - clone
    - read ITL
    - undo transaction



  - Consistent Read (CR读)
    - 将block上所有提交scn > query scn的事务undo
  - Ora-01555
    - snapshot too old ?
    - undo_retention = 10800 ?

Oracle MVCC

- 分析
  - 优势
    - 页面中不保存版本信息
    - 支持闪回查询
      - select count(*) from flashback_query_test as of timestamp to_timestamp('20011-11-17 13:34:12', 'yyyy-mm-dd hh24:mi:ss');
  - 缺点
    - 页面级别多版本，访问冲突
    - 著名的ora-01555!
      - 大表的全表扫描处理：并行查询
      - select /* +parallel (t1 8) */ * from huge_table t1;

Innodb MVCC

- 关键词
  - 基于事务ID
  - 行级多版本
  - 回滚段

- 事务ID
  - 唯一标识一个事务
  - 递增产生
    - 新事务，对应更大的事务ID
  - 64位，8 bytes

Innodb MVCC

- MVCC扩展结构
  - 聚簇索引(主键索引)
    - 记录扩展(系统列)
      - DB_TRX_ID
      - ROLLBACK_PTR
      - Delete_Bit
  - 二级索引(辅助索引)
    - 记录扩展
      - Delete_Bit
    - 页面扩展
      - DB_MAX_ID
        - » 作用？Index only scan

Innodb MVCC

- 更新处理
  - Delete
    - 聚簇索引
      - 设置Delete_Bit位，前项写入回滚段
      - 前项包括：系统列，二级索引属性列
    - 二级索引
      - 设置Delete_Bit位
  - Update
    - 聚簇索引
      - 原地更新，前项写入回滚段
    - 二级索引
      - clone & update，原项设置Delete_Bit (不物理删除)

Innodb MVCC

- 可见性判断
  - 原理
    - 给定事务Tx，所有在事务Tx开始时已经提交的事务做的更新是可见的(snapshot read)
  - ReadView
    - 每个事务，都有read_view结构，通过此判断可见性
    - low_limit_id: 事务id >= 的所做的修改，均不可见
    - up_limit_id: 事务id < 的所做的修改，一定可见
    - trx_ids: 事务开始时，活跃事务链表

Innodb MVCC

- Undo
  - 事务Undo
    - insert_undo链表
    - update_undo链表
  - 可见版本构造(记录级)
    1. read 最新版本，判断可见性
    2. 根据ROLLBACK_PTR回滚到前一版本
    3. 继续步骤1

Innodb MVCC

- Purge
  - 功能
    - 回收聚簇/二级索引上的删除项
  - 方式
    - 定期唤醒purge线程
    - 遍历undo日志，构造索引记录，查找并删除
    - 一次回收20项undo pages
      - n_pages_handled = 20
  - 不足
    - 为了能够删除二级索引记录，undo中必须记录完整索引项
  - 思考
    - 为什么需要做purge？

Innodb MVCC

- 分析
  - 优势
    - 实现简单(相对于Oracle)
    - 行级多版本，不存在false violation
  - 缺点
    - 记录开销增加(系统列)
    - 记录不能实时物理删除(行标识)
    - 更新时，undo中需要记录二级索引所有属性
    - purge性能较差
      - 过期版本不能及时回收
      - undo日志不能被覆盖
    - 不支持闪回查询

Postgres MVCC

- 关键词
  - 基于事务ID
  - 行级多版本
  - 无回滚段，行内存储
    - 一次update操作，产生记录的两个版本
    - 两个版本，都存储于页面内部

Postgres MVCC

- MVCC扩展结构
  - 行结构(系统列)
    - xmin
      - 记录创建的事务ID
      - Insert，Update
    - xmax
      - 记录过期的事务ID
      - Delete，Update，Row Locks
      - Transaction Status Array
    - **cmin/cmax(单列)**
      - 用于标识多语句事务的statement顺序
      - **想到了什么？**
      - Combo Command ids

Postgres MVCC

- xmin/xmax
  - insert

```
 xmin | xmax | val
------+------+-----
 5409 |    0 |   1
```

  - delete

```
 xmin | xmax | val
------+------+-----
 5411 | 5412 |   1
```

  - update

```
 xmin | xmax | val
------+------+-----
 5414 |    0 |   2
```

```
 xmin | xmax | val
------+------+-----
 5413 | 5414 |   1
```

  - row lock

```
 xmin | xmax | val
------+------+-----
 5416 | 5417 |   1
```

Postgres MVCC

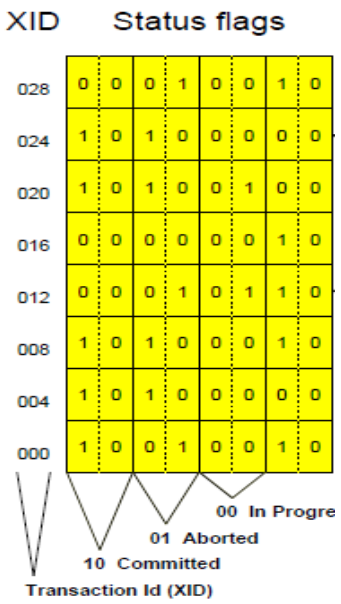
- cmin/cmax

xmin	cmin	xmax	val
5419	0	0	1
5419	1	0	2
5419	2	0	3

xmin	xmax	cmax	val
5421	5421	0	1
5421	5421	1	2
5421	5421	2	3

Postgres MVCC

- Transaction Status Array
  - 事务数组，记录事务状态
  - why need？
    - 行事务信息不清理



- Combo Command Id
  - cmin/cmax是单列
  - 如何处理同一事务同时Insert/Delete多行？
    - 此时既需要cmin，又需要cmax

Postgres MVCC

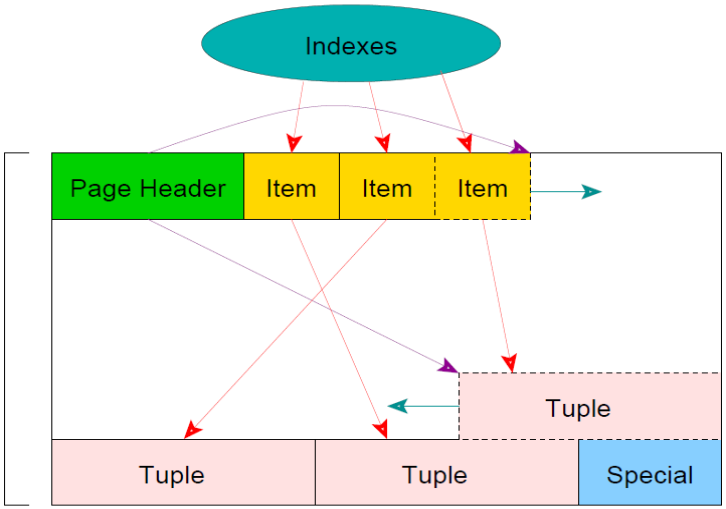
- 可见性判断

PostgreSQL 9.1.1 src/backend/utils/time/tqual.c

```
((Xmin == my-transaction &&                inserted by the current transaction
  Cmin < my-command &&                     before this command, and
  (Xmax is null ||                         the row has not been deleted, or
   (Xmax == my-transaction &&              it was deleted by the current transaction
    Cmax >= my-command)))                  but not before this command,
||                                         or
 (Xmin is committed &&                     the row was inserted by a committed transaction, and
     (Xmax is null ||                      the row has not been deleted, or
      (Xmax == my-transaction &&           the row is being deleted by this transaction
       Cmax >= my-command) ||              but it's not deleted "yet", or
      (Xmax != my-transaction &&           the row was deleted by another transaction
       Xmax is not committed))))           that has not been committed
```

Postgres MVCC

- 过期版本回收



  - 回收对象
    - 堆记录；堆记录指针；索引记录
    - 顺序：堆记录->索引记录->记录指针
  - 回收方法
    - 页面级：页面访问时回收
    - 表级/系统级：autovacuum；vacuum cmd

Postgres MVCC

- 分析
  - 优势
    - 实现简单(Postgres原来不支持多版本)
    - 过期版本页内存储，无需rollback获取
  - 不足
    - 空间消耗：过期版本页内存储；系统列
    - 性能消耗：Vacuum
    - 可扩展：vacuum不及时
    - 不支持闪回查询

MVCC总结

- 时间戳 vs 事务ID
- 回滚段 vs 页内存储
- 可见性判断
- 过期版本回收
- 闪回查询

Q&A

谢谢大家