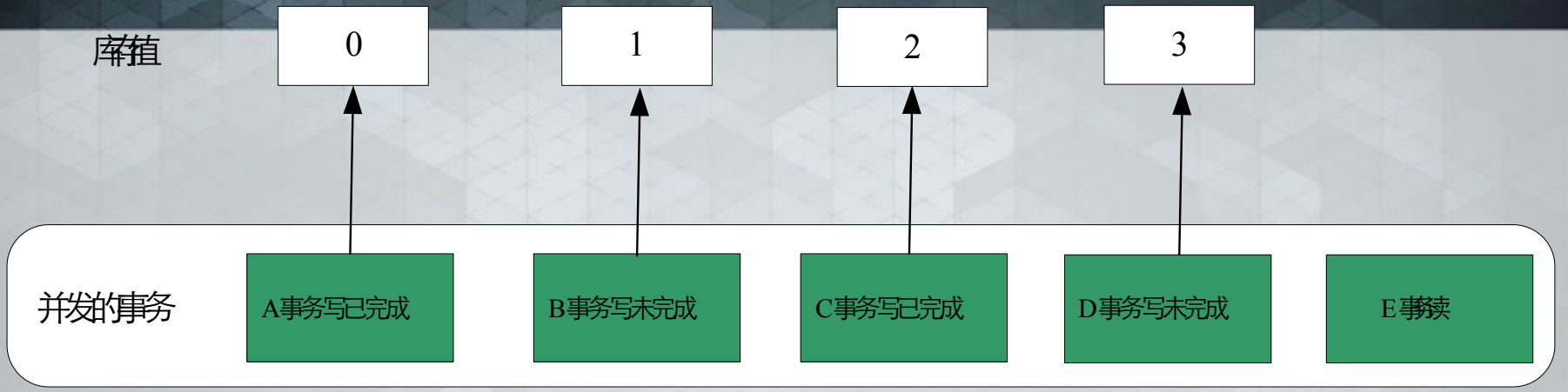
WiredTiger 存储引擎介绍

目录

CONTENTS

插件式存储引擎架构



1. E事务开始时，产生一个快照，快照里面包含B和D两个未完成的写事务
2. 如果E事务是读事务，则读取到库存值是在C事务完成后的值2，即使后面D事务提交完成了，E事务读取到的值也不会改变
3. 如果E事务为写事务，对库存值进行修改，则会发生冲突而终止（怎样检测到这种冲突是通过**MVCC**实现的），这样能防止对过期数据的修改，保证数据的一致性。

MongoDB使用快照隔离级别的事务

```
wt_transaction {
   transaction_id,
   snapshot_object,
   operation_array,
   redo_log_buf,
   state
}
```

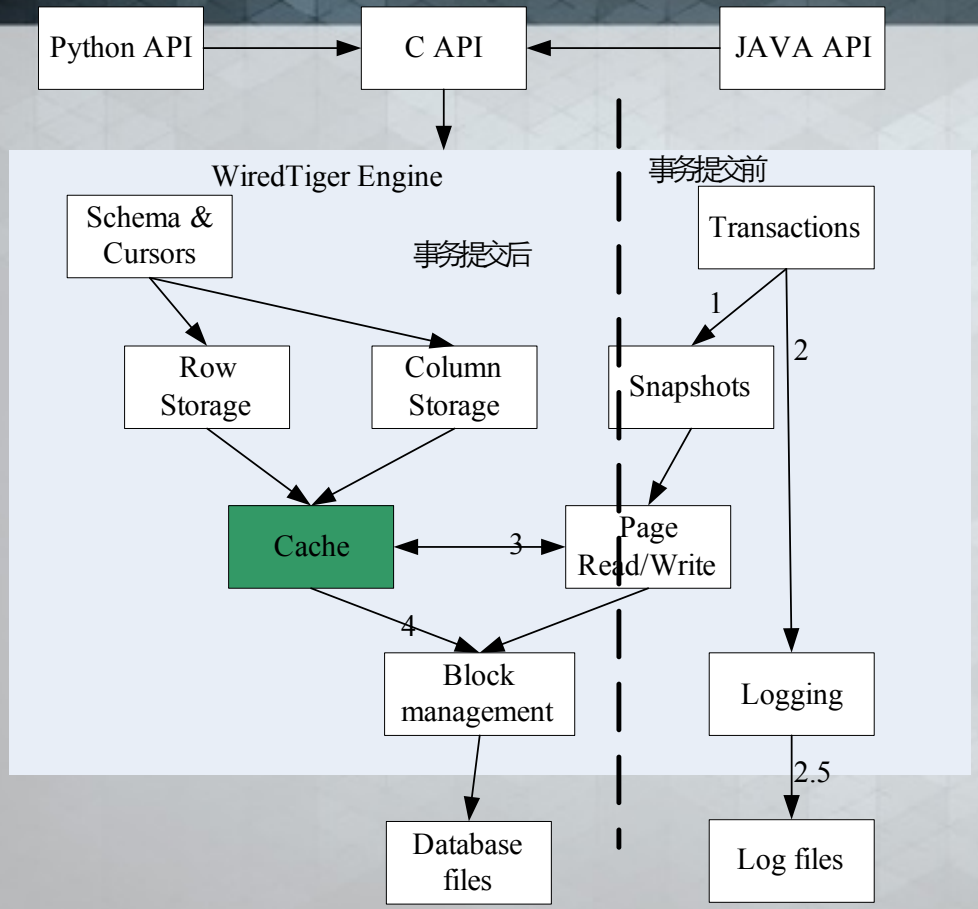
transaction_id为事务的唯一标识一直递增，由存储引擎内部的全局事务管理模块进行分配

snapshot_object为事务开始执行时其他正在执行写操作还未提交的事务，用于事务快照

operation_array为本次事务中执行的操作动作，用于事务回滚。

redo_log_buf为事务对应的重做日志缓冲区，会被持久化

state为事物的当前状态

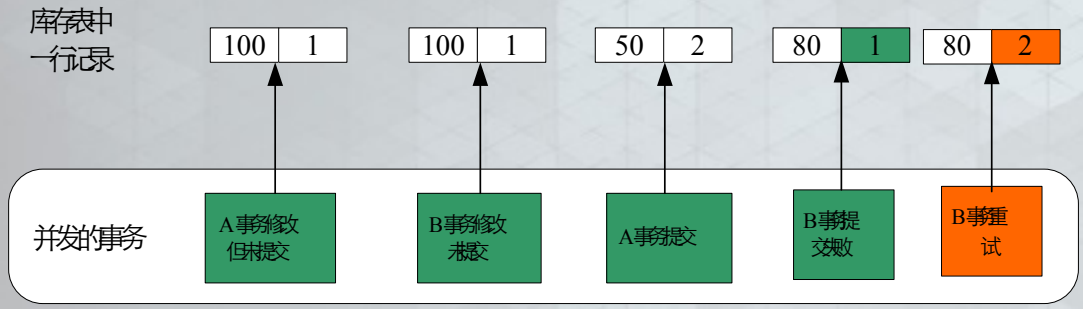


1. 写操作事务开始执行之前，对所有正在执行的还未提交事务进行快照。

2. 将本次写操作的动作保存到operation_array中，可以从中提取出动作进行回滚；其次将修改的数据以日志形式记录下来，记录到redo_log_buf日志缓冲区。

3. 写操作事务提交，首先将重做日志缓冲区中的数据刷到磁盘上，写入到log文件，数据库意外宕机恢复时需要读取这个文件，重演文件里面的动作。

4. 写操作引起的数据变化，首先写入到WiredTiger存储引擎的cache中，cache中的数据以Btree结构组织，Btree的叶子节点是真正存放数据的page，当数据发生更改时page就变为“脏页”（在内存中）；存储引擎每60s将“脏页”中的数据写到物理磁盘上进行持久化。



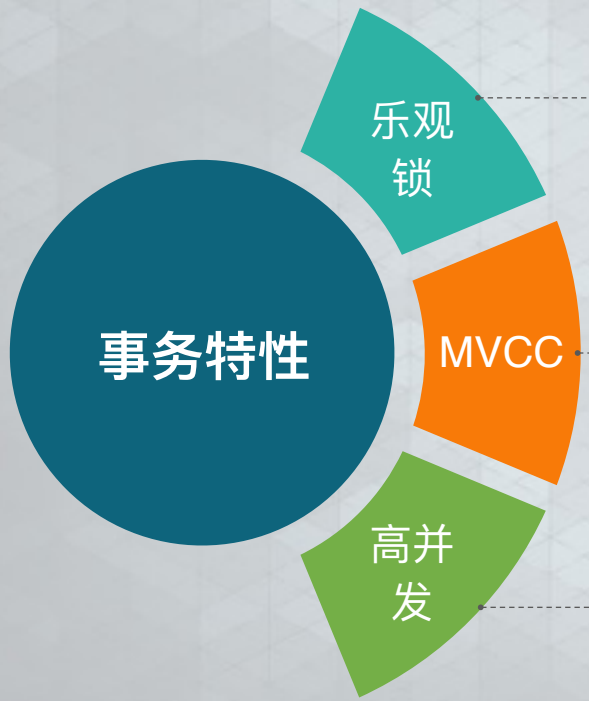
1. A事务首先从表中读取到要修改的行数据，读取到库存值为100，行记录的版本号为1

2. B事务也从中读取到要修改的相同行数据，读取库存值为100，行记录版本号为1。

3. A事务修改库存值后提交，同时行记录版本号加1，即变为2，大于一开始读取到的版本号1，A事务可以提交。

4. 但B事务提交时发现此时行记录版本号已经变为2，产生冲突，B事务提交失败。

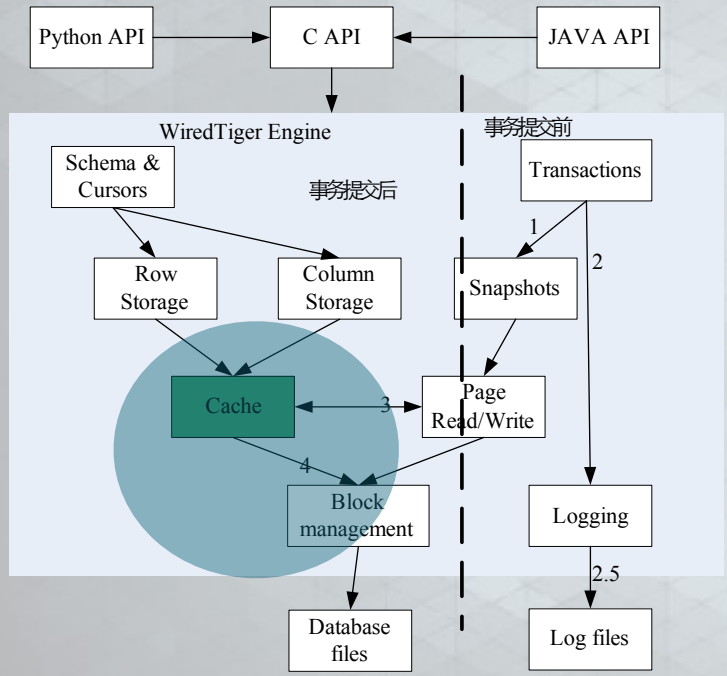
5. B事务尝试重新提交，此时读取到的版本号为2，加1，即变为3，不会产生冲突正常提交。



允许冲突发生，冲突发生后，数据库会再次尝试提交动作

引入MVCC多版本并发控制机制解决乐观锁里面事务的冲突检测

避免了长事务中的数据库加锁开销，大大提升了并发性



刷盘周期

内存中被修改的数据按一定周期(默认60s)或一定条件（如果开启了Journal，则日志文件大小达到2G）时能刷到磁盘上进行持久化

触发Checkpoint

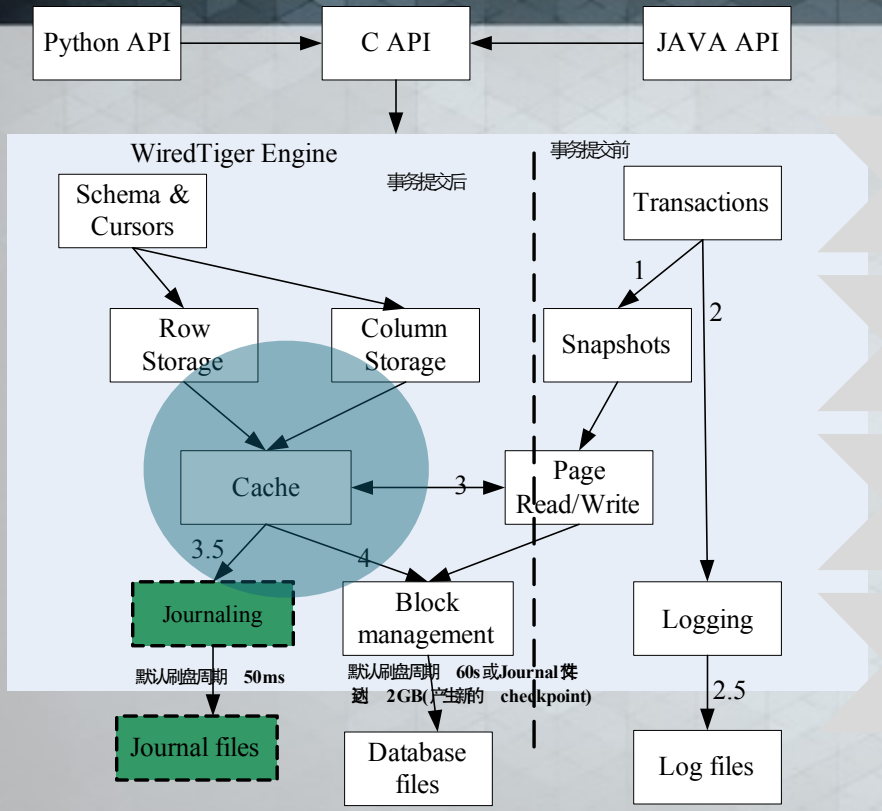
当刷盘条件触发时，持久化内存中的修改page，就会产生一个checkpoint点在数据文件上

数据恢复

当数据库意外宕机恢复时，只需要从最新的checkpoint点进行恢复，这样节省恢复时间

清理内存

一旦新的checkpoint完成后，内存中旧的checkpoint点到新的checkpoint点间的数据占用的page可以释放掉



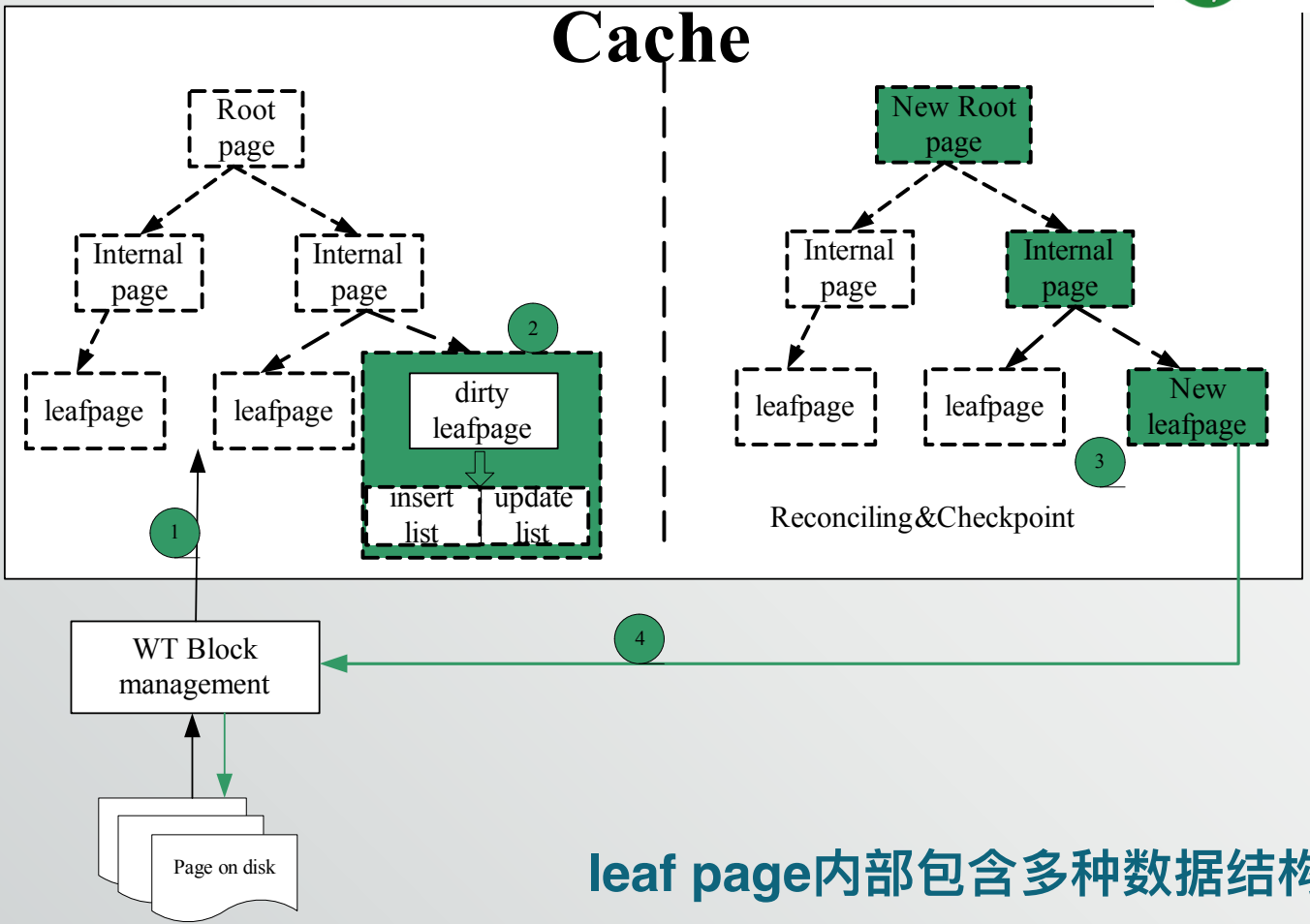
如果开启了Journal，存储引擎在将数据持久化到磁盘之前，会将每一个写操作修改日志记录到Journal对应的内存中

默认以50ms周期将Journal日志从内存持久化到磁盘或者创建新的Journal文件时；如果写操作带有写关注选项，则WiredTiger存储引擎会强制刷新内存中的Journal日志到磁盘

由于Journal日志文件大小默认为100MB，当达到这个限制时，存储引擎会创建新的Journal日志文件，同时刷新内存中的日志到磁盘

数据库恢复时，存储引擎会先从最新的Checkpoint开始，查找Journal日志文件中的修改记录，最后再进行恢复

双重保险



leaf page内部包含多种数据结构

leafpage

```
struct __wt_page{
    WT_ROW *row;                    /* Key/value pairs */
    uint32_t entries;               /* Leaf page entries */
    WT_PAGE_MODIFY *modify;         /* modified information. */
}
```

```
struct __wt_page_modify {
    size_t bytes_dirty;             /* Dirty bytes added to the cache. */
    WT_INSERT_HEAD  **insert;       /* Inserted items for row-store. */
    WT_UPDATE       **update;       /* Updated items for row-stores. */
}
```

```
struct __wt_update {
    uint64_t txnid;                 /* update transaction */
    WT_UPDATE *next;                /* forward-linked list */
    uint32_t size;                  /* update length */
    upd;                            /*updata data*/
}
```

```
struct __wt_insert {
    WT_UPDATE *upd;                 /* value */
    uint32_t offset;                /* row-store key data start */
    uint32_t size;                  /* row-store key data size */
    WT_INSERT *next[0];             /* forward-linked skip list */
}
```

源码

MMAPV 1

```
> db.customers.stats()
{
        "ns" : "customers.customers",
        "count" : 999998,
        "size" : 111999776,
        "avgObjSize" : 112,
        "numExtents" : 12,
        "storageSize" : 174735360,
```

Wired Tiger

```
> db.customers.stats()
{
        "ns" : "customers.customers",
        "count" : 999998,
        "size" : 63999872,
        "avgObjSize" : 64,
        "storageSize" : 20430848,
        "capped" : false,
        "wiredTiger" : {
```

WiredTiger默认采用Snappy压缩

MongoDB 中文社区

IT大咖说 知识分享平台

报告完毕 感谢聆听