# 13 为什么表数据删掉一半，表文件大小不变？

一个InnoDB表包含两部分，即表结构定义和数据，在MySQL8.0版本以前，表结构是存在以.frm为后缀的文件里，而MySQL8.0版本，则已经吧表结构定义放在系统数据表中了，因为表结构定义占用的空间很小，所以主要讨论表数据

**参数innodb\_file\_per\_table**

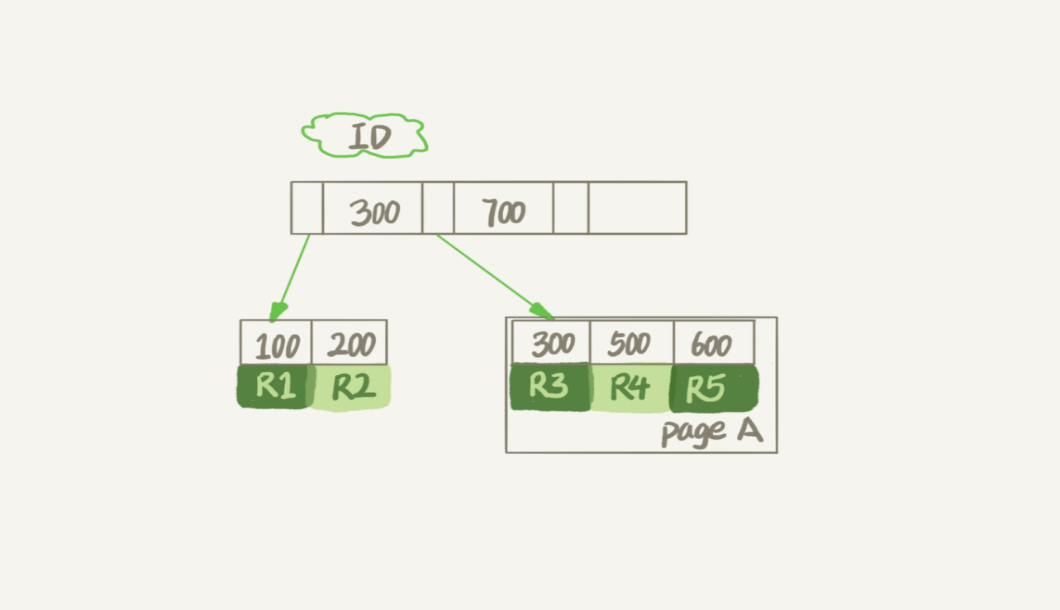
表数据即可以存在共享表空间里，也可以单独的文件，这个行为是有参数

Innodb\_file\_per\_table控制的

1. 这个参数设置为OFF表示的是，表的数据放在系统共享表空间，也就是跟数据字典放在一起
2. 这个参数设置为ON表示的是，每个InnoDB表数据存储在一个以.ibd为后缀的文件中

从mysql 5.6 版本开始，默认On了，建议设置为On,因为一个单独存储为一个文件更容易管理，而且在不需要这个表的时候，通过drop table命令，系统就会直接删除这个文件。而如果放在共享表空间中，即使是删除表，空间也不会回收。

**数据删除流程**



假设要删除R4这个记录，InnoDB引擎会把R4记录标记为删除，如果再插入一个ID在300和600之间的记录时，可能会复用这个位置，但是，磁盘文件的大小并不会缩小。

**如果是删除一个数据页上的记录，整个数据页可以被复用**

**数据页的复用跟记录的复用是不同的**

记录的复用，只限于符合范围条件的数据，比如上面的这个例子，R4这条记录被删除，如果插入一个ID=400的行，可以直接复用这个空间。但是要插入ID=800的行，就不能复用这个位置。

而当整个也从B+树里摘除掉，可以复用到任何位置，以图1为例，如果将数据页pageA上的所有记录删除，pageA被标记为可复用。此时如果要插入ID=500的记录需要使用新的页，pageA可以被复用的。

如果相邻的数据页的利用率都很小，系统就会把这两个页上的数据合并到其中一个页上，另一个数据页被标记为可复用。Delete命令把整个表的数据删除，所有的数据页都不被标记可复用，但是磁盘上，文件大小不会变

Delete 命令只是把记录的位子标记为可复用，或者数据页标记为“可复用”，

但磁盘文件的大小不会变的，也就是通过delete命令不会回收表空间，这些可以复用的，而没有被使用的空间，看起来像是“空洞”。

**不止删除数据会造成空洞，插入数据也会**

如果数据是按照递增的顺序插入的，那么索引是紧凑的，但如果数据时随机插入的，就可能造成索引的数据页分裂。

假设图1中pageA已经满了，这时要在插入一行数据，会怎么样呢

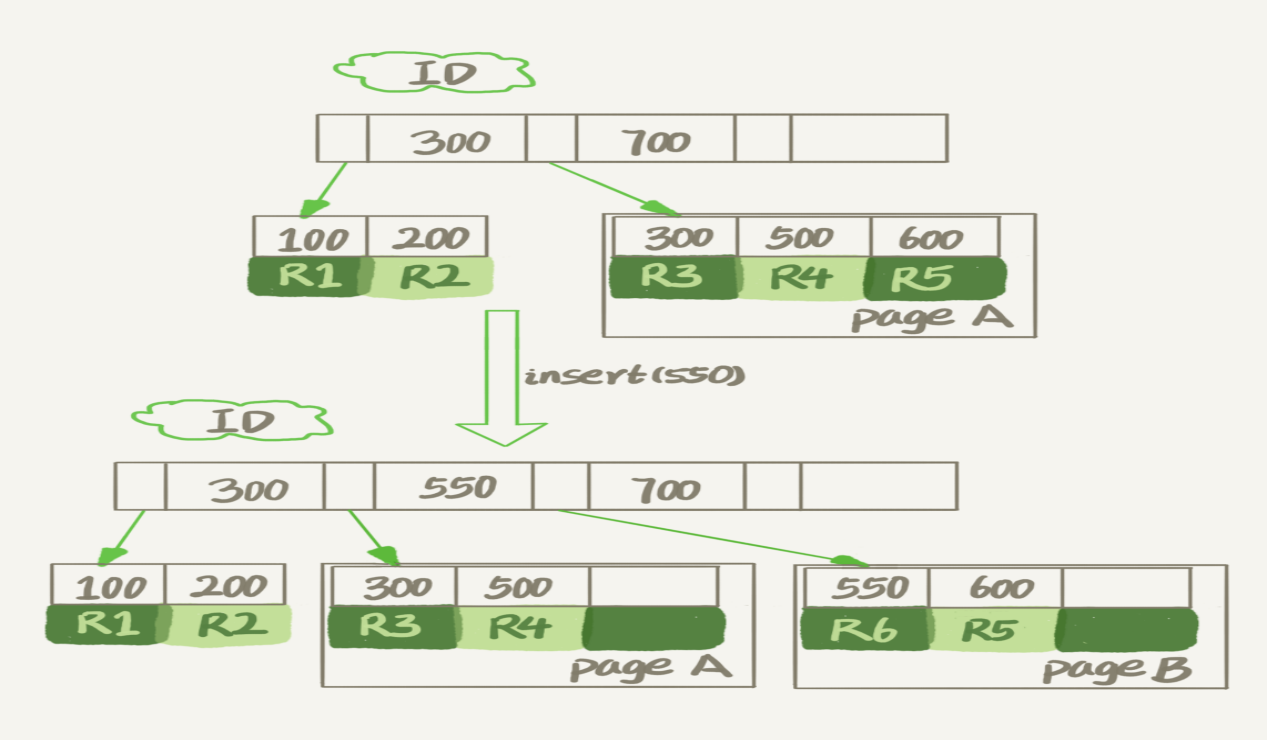


图2 ，插入数据导致页分裂

由于pageA满了，在插入一个ID是550的数据时，就不得不在申请一个新的页面pageB来存数据，页面分裂完成后，pageA末尾就留下了空间（注意：实际上，可能不止一个记录的位置空洞）

另外跟新索引上的值，可以理解为删除一个旧的值，在插入一个新值，也会造成空洞。

经过大量增删改查的表，都是可能是存在空洞的，如果能够把这些空洞去掉，

就能达到收缩表空间的目的，**重建表可以达到此目的**

**重建表**

可以新建一个与表A结构相同的表B，然后按照主键ID递增的顺序，把数据一行行的从表A里读出来在插入到表B中。

由于表B 是新建的表，所以表A的主键索引上的空洞，在表B中就都不存在了，显然的，表B的索引跟紧凑，数据页的利用率也更高。如果我们把表B作为临时表，数据从表A导入表B的操作完成后，用表B 代替表A,从效果上看，就起到了收缩表A的作用。

可以使用alter table A engine=InnoDB命令来重建表，在MySQL5.5版本以前，这个命令的执行跟我们前面描述的差不多，区别只是这个临时表B不需要自己创建，MySQL会自动完成转存数据，交换表名，删除旧的表操作。

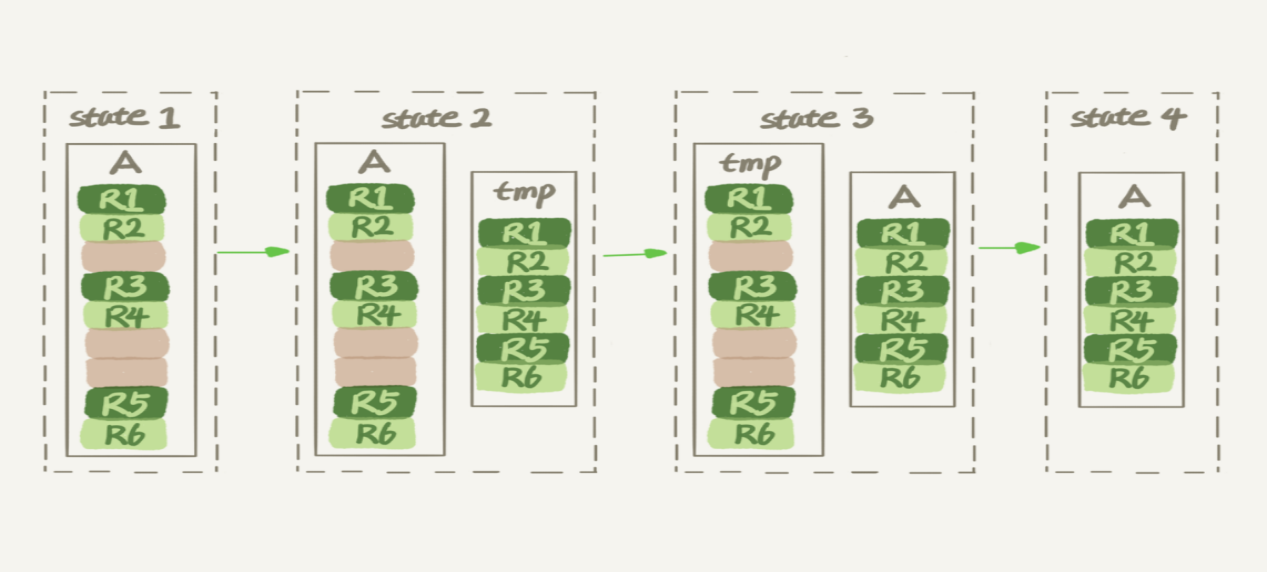


图3 改锁表DDL

显然，花时间最多的步骤是往临时表插数据的过程，如果在这过程中，有新的数据要写入到表A中，就会造成数据丢失。因此整个DDL过程中，A表不能有更新，这个DDL不是Online的。

**MySQL 5.6版本引入的OnlineDDL,对这个操作流程做了优化**

1. 建立临时表，扫描表A主键的所有数据页。
2. 用数据页中表A的记录生成B+树，存储到临时文件中
3. 生成临时文件的过程中，将所有对A的操作记录在一个日志文件（row log）中，对应的是图中state2的状态
4. 临时文件生成后，将日志文件中的操作应用到临时文件，得到一个逻辑数据上与表A相同的数据文件，对应就是图中state3的状态
5. 用临时文件替换表A的数据文件。



图4 Online DDL

与图3过程的不同之处在于，由于日志文件和重放操作这个功能的存在这个方案重建表的过程中，允许对表A做增删改操作，这就是Online DDL名字的来源。

1. DDL之前要拿MDL写锁的，这样还能叫Online DDL吗?

确实，图4中流程，alter 语句的启动的时候需要获取MDL写锁，但是这个写锁在真正拷贝数据之前就退化成读锁了。

为什么退化呢？为了实现Online,MDL读锁不会阻塞增删改操作

那为什么不干脆直接解锁呢？为了保护自己，禁止其他线程对这个表同时做DDL

而对于一个大表来说，OnlineDDL最耗时的过程就是拷贝到临时表的过程，这个步骤的执行期间接受增删改操作，所以，相对整个DDL过程来说，锁的时间非常短，对业务来说，就可以是Online的。

上述的这些重建方法会扫描原表数据和构建临时文件，对于很大的表来说，这个操作很消耗IO和CPU资源，如果是线上业务，要小心操作，推荐使用GibHub开源的gh-os来做。

Online 和inplace

在图3中，把表A中的数据导出来存放的位置叫做tmp\_table,这是一个临时表是server层创建的

在图4中，根据表A重建出来的数据是放在tmp\_file里，这个临时文件是InnoDB在内部创建出来的。整个DDL过程都在InnoDB内部完成，对于server层来说，没有把数据挪动到临时表，是一个“原地”操作，这个就是inplace名称的来源。

所以，如果一个1TB的表，现在磁盘空间是1.2TB.能不能做一个inplace的DDL呢？

答案是不能，因为tmp\_file也是需要占用临时空间的

重建表alter table t engine= InnoDB，其隐含意思是；

alter table t engine=innodb,ALGORITHM=inplace;

跟 inplace 对应的就是拷贝表的方式了，用法是：

alter table t engine=innodb,ALGORITHM=copy;

当使用ALGORITHM=copy时，表示强制拷贝表，对应的流程就是图3 的操作过程

但这样可能觉的，inplace和Online是不是一个意思？

其实不是，只是在重建表这个逻辑中刚好是这样的

比如要给InnoDB表的一个字段加全文索引，写法是

alter table t add FULLTEXT(field\_name);

这个过程是inplace的，但会阻塞增删改操作，是非OnLine的

这两个逻辑之间的关系是什么可以概括为：

1. DDL过程如果是Online的，一定是inplace的，
2. 反过来未必，也就是inplace的DDL,有可能不是OnLine的，截止到MySQL8.0，添加全文索引（FULLTEXT index）和空间索引（SPATIAL index）就属于这种情况

Optimize table\analyze table和 alter table 这三种方式重建表的区别，

1.从MySQL 5.6版本开始，alter table t engine=InnoDB(也就是recreate)默认就是上面图4的流程

2.Analyze table t 其实不是重建表，只是对表的索引信息做重新统计，没有修改数据，这个过程加了MDL锁。

3.Optimize table 等于 recreate+ analyze

**思考题**

一个“想要收缩表空间，结果适得其反”的情况

1. 一个表t文件大小1TB
2. 对这个表 alter table t engine = InnoDB
3. 发现执行完成后，空间不仅没有缩小，还稍微大了一点儿，比如1.01TB

可能是什么愿意呢？