[MYSQL] 当一个PAGE里的数据全部被delete之后，它还会存在于Btree+中吗?

原创 大大刺猬 大大刺猬 2025年06月12日 18:10 上海

导读

我们知道在mysql里，delete数据只会对其打上delete_flag，不会真正的删除数据,该数据还在页内，该页也还在Btree+中；遍历树节点的时候发现有delete_flag的行就直接跳过.

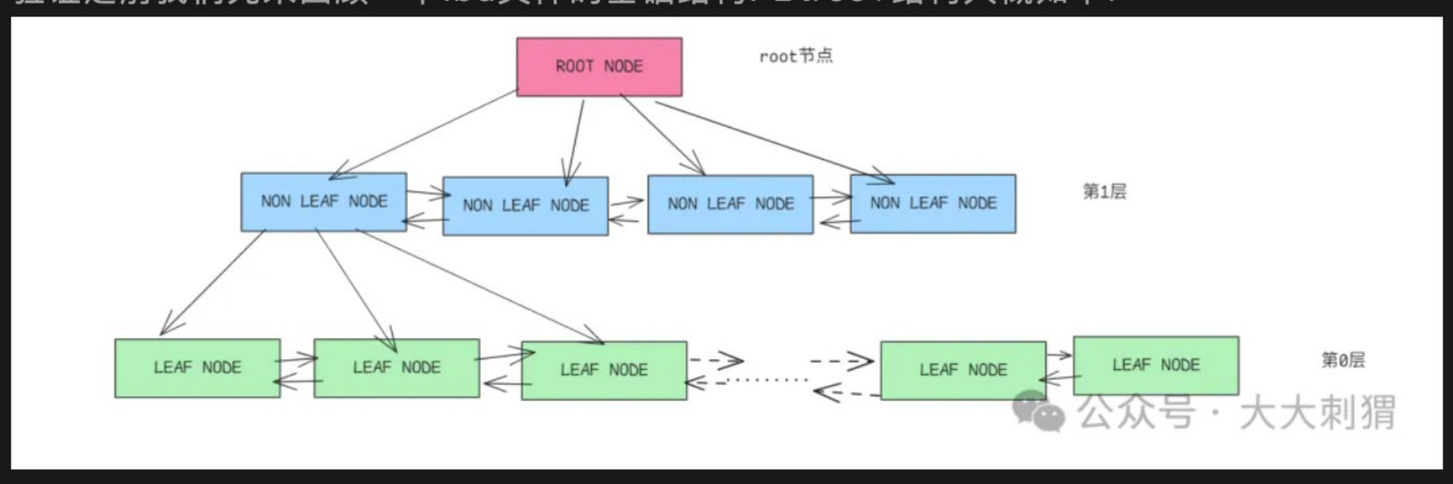
那么一个页里的数据全部被`delete`之后，这一页是否还存在于Btree+中呢?

先说答案: 不在btree+中了,且数据可能也被mv一部分了

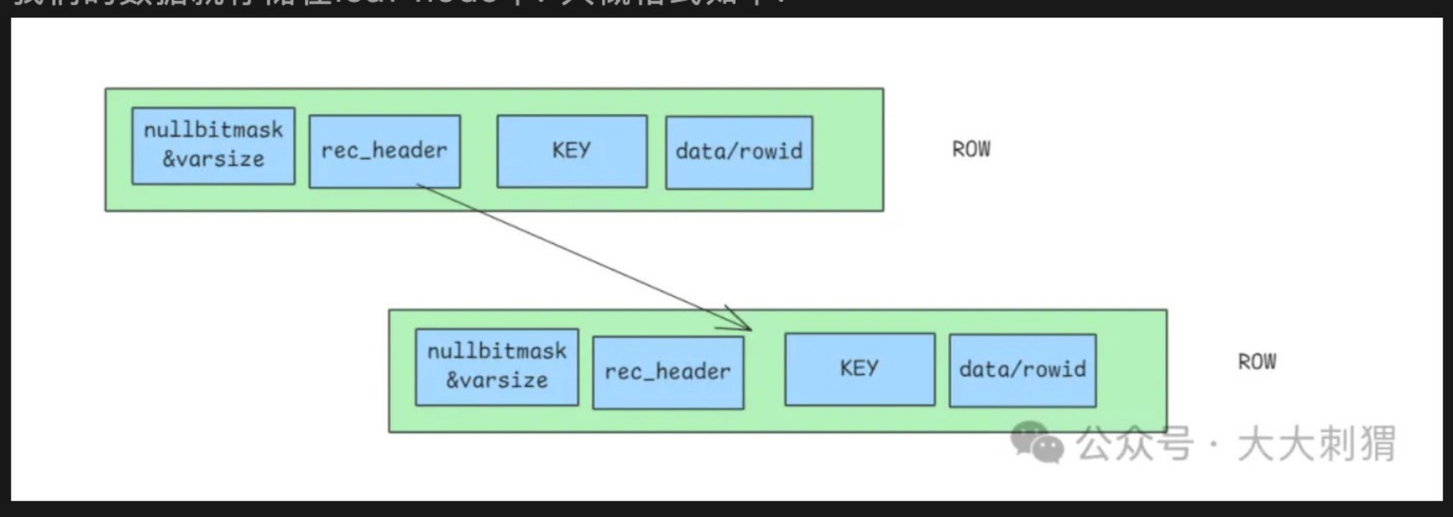
> 被无情的抛弃了

基础回顾

验证之前我们先来回顾一下ibd文件的基础结构. Btree+结构大概如下:



我们的数据就存储在leaf node中. 大概格式如下:



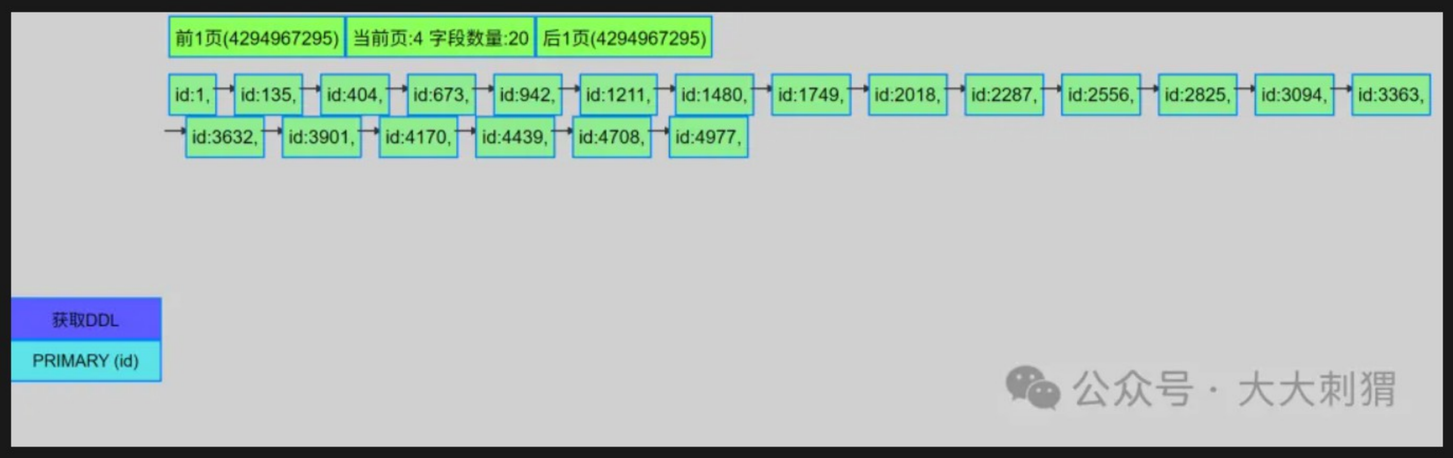
准备测试数据

然后我们来准备点测试数据

```
-- create procedure
delimiter //
create procedure pro_insert_Nrows( IN rows1 int)
begin
declare n int;
set n=1;
set autocommit=off;
while n <= rows1
do
insert into db1.t20250612_2 values(n,md5(n));
set n=n+1;
end while;
commit;
end//
delimiter ;

-- create table
create table db1.t20250612_2 (id int primary key, c2 varchar(200));

-- call procedure
call pro_insert_Nrows(5000);
```

我们再使用ibd2sql查看下当前的页分布情况

```
python3 ibd2sql_web.py /data/mysql_3314/mysqldata/db1/t20250612_2.ibd
```



我这测试数据比较少(5000), 就只有1层(root&leaf page).
第1页的数据范围是1-134
第2页的数据范围是135-403
第3页的数据范围是404-672
...

当然我们还可以使用如下python代码来查看,更方便点

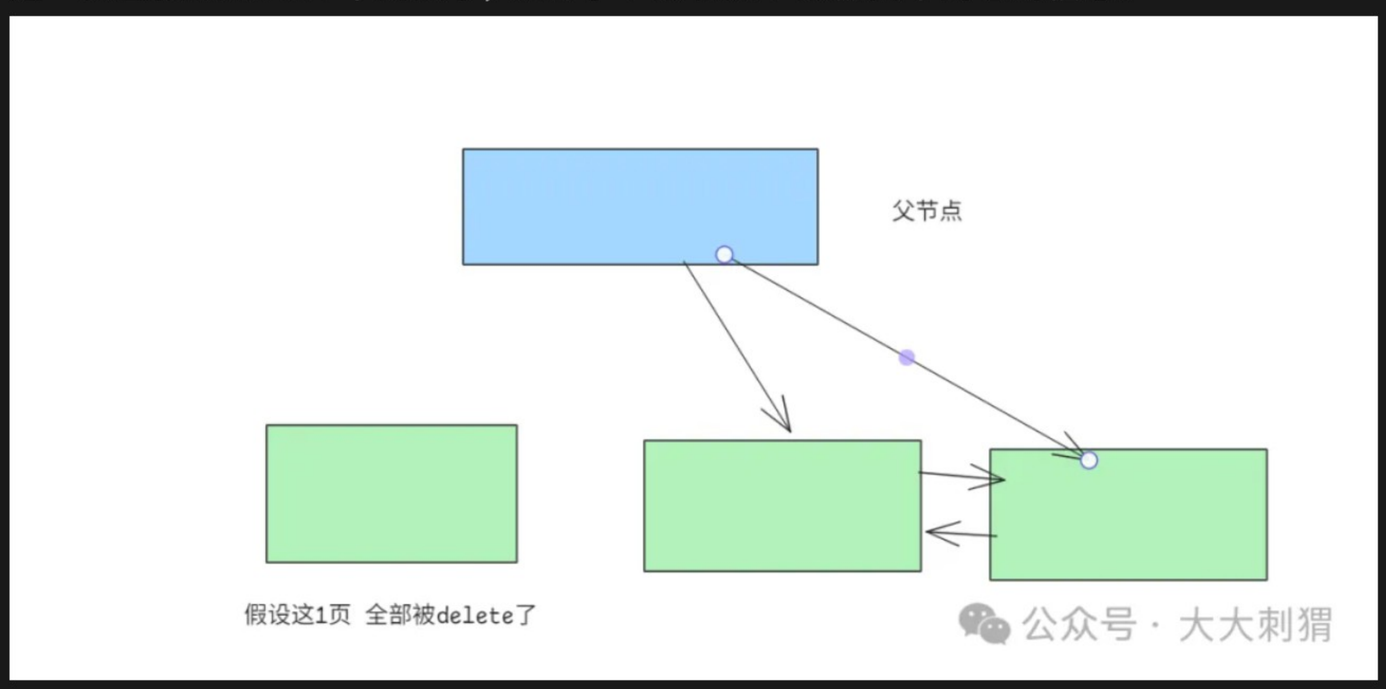
```
import struct
f = open('/data/mysql_3314/mysqldata/db1/t20250612_2.ibd','rb')
f.seek(4*16384,0)
data = f.read(16384)
offset = 99
for x in range(10): # only view 10 rows
offset += struct.unpack('>h',data[offset-2:offset])[0]
        pk,pageid = struct.unpack('>LL',data[offset:offset+8])
        print(x,'PK:',pk-2**31, "PAGE ID:",pageid)

```

```
>>> import struct
>>> f = open('/data/mysql_3314/mysqldata/db1/t20250612_2.ibd','rb')
>>> f.seek(4*16384,0)
65536
>>> data = f.read(16384)
>>> offset = 99
>>> for x in range(10): # only view 10 rows
...     offset += struct.unpack('>h',data[offset-2:offset])[0]
...     pk,pageid = struct.unpack('>LL',data[offset:offset+8])
...     print(x,'PK:',pk-2**31, "PAGE ID:",pageid)
...
0 PK: 1 PAGE ID: 5
1 PK: 135 PAGE ID: 6
2 PK: 404 PAGE ID: 7
3 PK: 673 PAGE ID: 8
4 PK: 942 PAGE ID: 9
5 PK: 1211 PAGE ID: 10
6 PK: 1480 PAGE ID: 11
7 PK: 1749 PAGE ID: 12
8 PK: 2018 PAGE ID: 13
9 PK: 2287 PAGE ID: 14
>>>
```

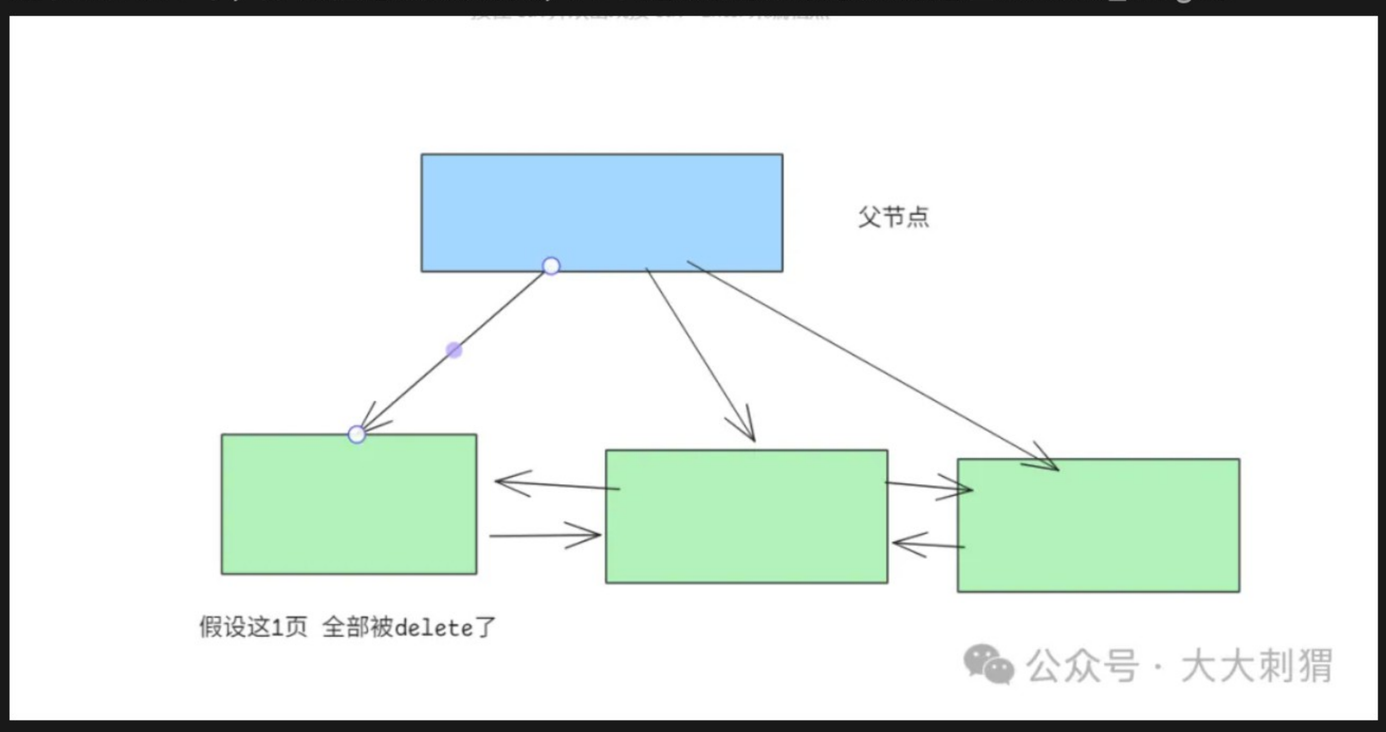

猜想

假设我们没有看到开头的结论, 然后要删除某一个页的数据; 这时有4种可能(排列组合.):

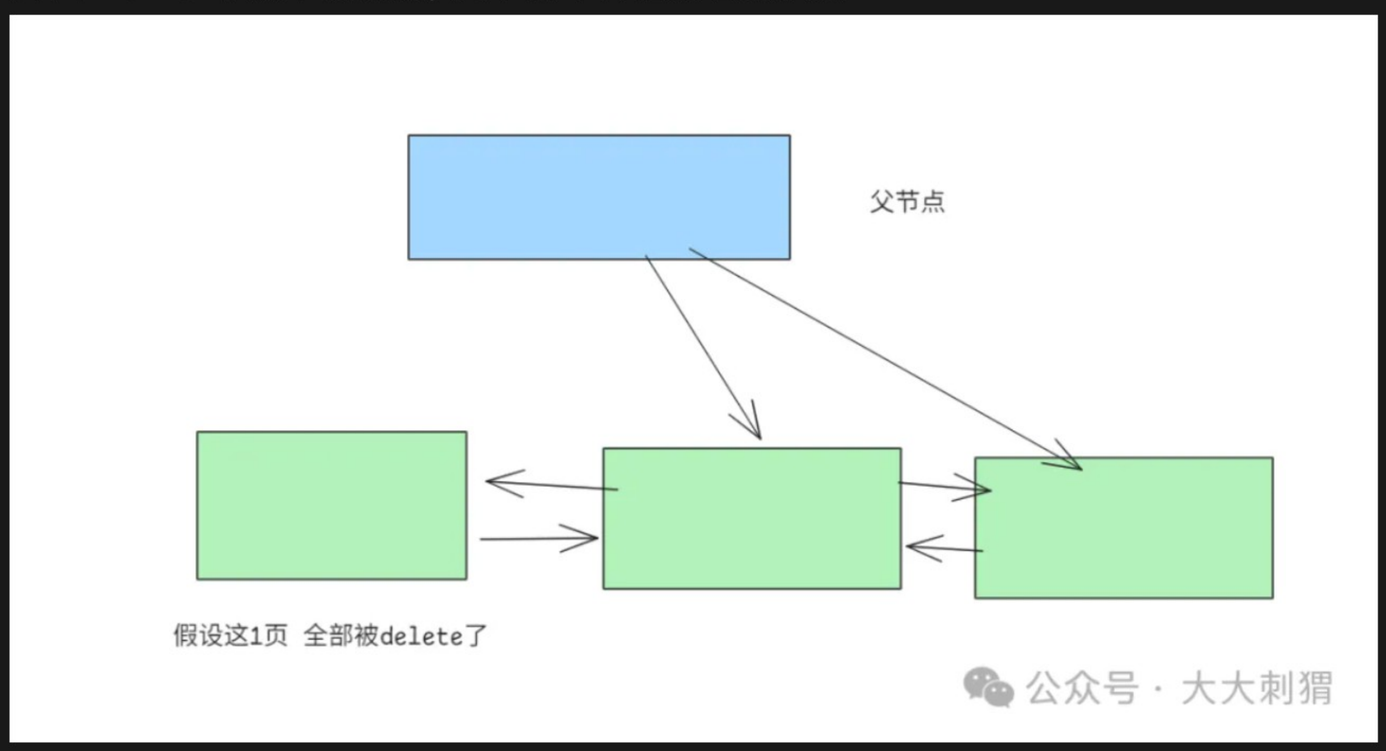
1. 这一页直接从btree+中消失了, 即叶子节点和父节点都找不到它的信息.



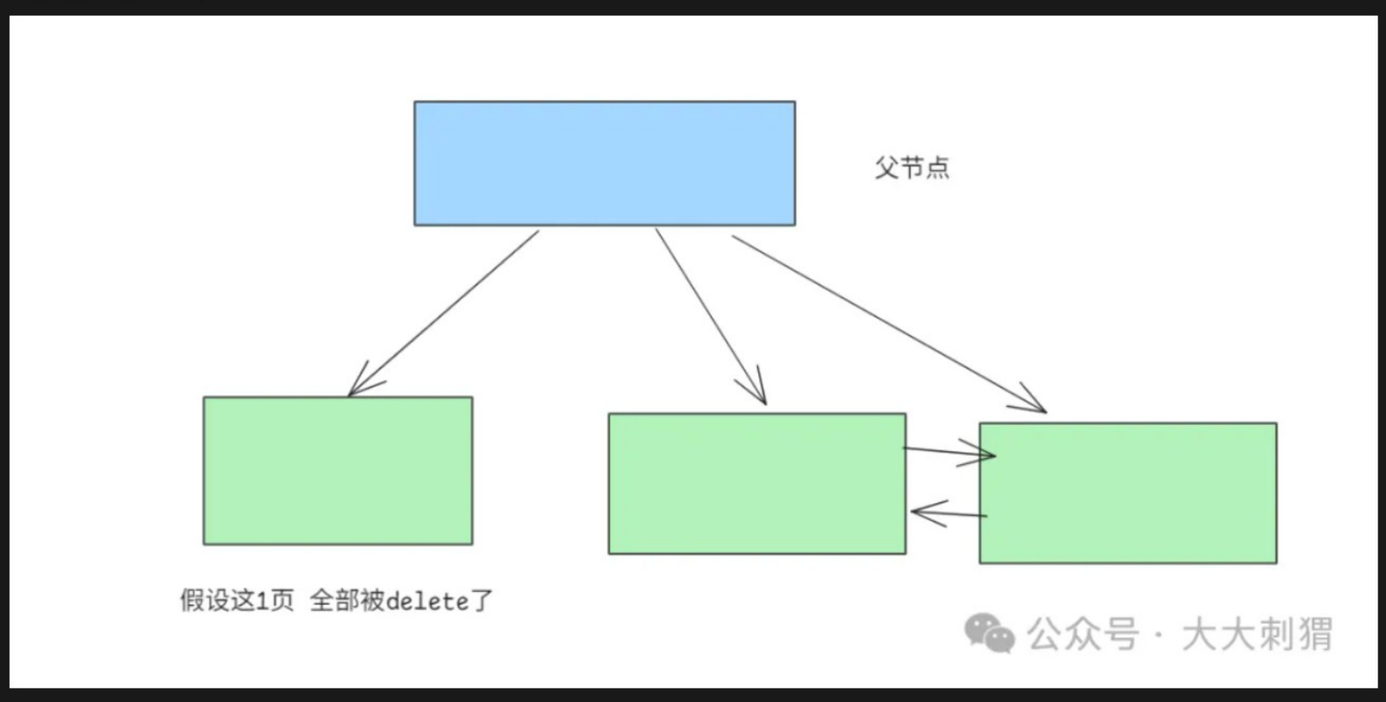
2. 还在btree+中, 什么信息都没有变, 只不过读数据的时候跳过含delete_flag的



3. 只在Brother节点中有记录，父节点不再记录它的信息.



4. 只在父节点记录，Brother节点不记录.（可以直接排除这种情况，不然遍历的时候数据就完整了...）



其实从上图就能看出来，只有猜想1,2的btree+是完整的，3,4两种的btree+都不完整了.

验证

我们就直接删除某个范围的数据，比如: 135—>403的

```
1 delete from db1.t20250612_2 where id>=135 and id<404;
2 flush tables;
```

```
>>> import struct
>>> f = open('/data/mysql_3314/mysqldata/db1/t20250612_2.ibd','rb')
>>> f.seek(4*16384,0)
65536
>>> data = f.read(16384)
>>> offset = 99
>>> for x in range(10): # only view 10 rows
...     offset += struct.unpack('>h',data[offset-2:offset])[0]
...     pk,pageid = struct.unpack('>LL',data[offset:offset+8])
...     print(x,'PK:',pk-2**31, "PAGE ID:",pageid)
...
0 PK: 1 PAGE ID: 5
1 PK: 404 PAGE ID: 7
2 PK: 673 PAGE ID: 8
3 PK: 942 PAGE ID: 9
4 PK: 1211 PAGE ID: 10
5 PK: 1480 PAGE ID: 11
6 PK: 1749 PAGE ID: 12
7 PK: 2018 PAGE ID: 13
8 PK: 2287 PAGE ID: 14
9 PK: 2556 PAGE ID: 15
>>>
```


我们发现第二页(135-403)的page信息没了(父节点没有记录这个信息了)，那么这个信息就真的没了吗？我们再瞅瞅brother是否记录这个PAGEID=6的信息.

```
f.seek(16384*5,0)
data = f.read(16384)
pre_pageno,next_pageno = struct.unpack('>LL',data[8:16])
print(pre_pageno,next_pageno)

f.seek(16384*7,0)
data = f.read(16384)
pre_pageno,next_pageno = struct.unpack('>LL',data[8:16])
print(pre_pageno,next_pageno)

f.seek(16384*6,0)
data = f.read(16384)
pre_pageno,next_pageno = struct.unpack('>LL',data[8:16])
PAGE_N_RECS = struct.unpack('>H',data[38:38+56][16:18])
print(pre_pageno,next_pageno,PAGE_N_RECS)

```

```
>>>
>>> f.seek(16384*5,0)
81920
>>> data = f.read(16384)
>>> pre_pageno,next_pageno = struct.unpack('>LL',data[8:16])
>>> print(pre_pageno,next_pageno)
4294967295 7
>>>
>>> f.seek(16384*7,0)
114688
>>> data = f.read(16384)
>>> pre_pageno,next_pageno = struct.unpack('>LL',data[8:16])
>>> print(pre_pageno,next_pageno)
5 8
>>>
>>> f.seek(16384*6,0)
98304
>>> data = f.read(16384)
>>> pre_pageno,next_pageno = struct.unpack('>LL',data[8:16])
>>> PAGE_N_RECS = struct.unpack('>H',data[38:38+56][16:18])
>>> print(pre_pageno,next_pageno,PAGE_N_RECS)
5 7 (146,)
>>>
```


我们发现PAGE-5记录的下一页是7. PAGE-7记录的上1页是5. 也就是说brother也不在记录这个全部被delete的数据行了. 也就**证明了是猜想1正确**了.

而PAGE-6虽然还是指向5和7, 但是已经不算数了. 而且我们发现page6的数据范围本来应该是`404 - 135 = 269`的, 但是现在只剩下146了(不应该是全部被删除了吗???), 那剩下的123条数据去哪了呢? 不在这个page-6里面了?

我们使用ibd2sql解析下被标记为delete的数据瞅瞅呢.

```
python3 main.py /data/mysql_3314/mysqldata/db1/t20250612_2.ibd --sql --d
```

```
14:19:52 [root@ddcw21 ibd2sql-main]#python3 main.py /data/mysql_3314/mysqldata/db1/t20250612_2.ibd --sql --delete | wc -l
146
14:19:53 [root@ddcw21 ibd2sql-main]#
```

发现被删除的数据为146条.

我们再看看这些delete的数据在那个page吧.

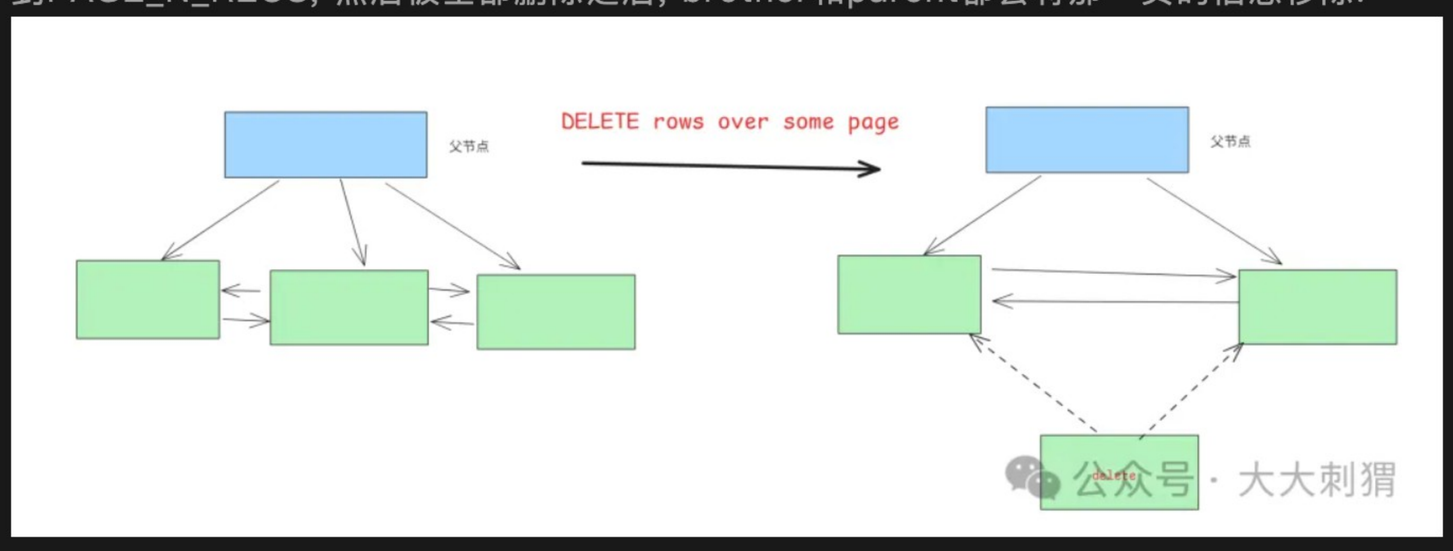
```
14:24:10 [root@ddcw21 ibd2sql-main]#python3 main.py /data/mysql_3314/mysqldata/db1/t20250612_2.ibd --sql --delete --page-start
=5 | wc -l
146
14:24:13 [root@ddcw21 ibd2sql-main]#
14:24:13 [root@ddcw21 ibd2sql-main]#python3 main.py /data/mysql_3314/mysqldata/db1/t20250612_2.ibd --sql --delete --page-start
=6 | wc -l
123
14:24:16 [root@ddcw21 ibd2sql-main]#
14:24:16 [root@ddcw21 ibd2sql-main]#
14:24:16 [root@ddcw21 ibd2sql-main]#python3 main.py /data/mysql_3314/mysqldata/db1/t20250612_2.ibd --sql --delete --page-start
=7 | wc -l
0
14:24:19 [root@ddcw21 ibd2sql-main]#
14:24:20 [root@ddcw21 ibd2sql-main]#
```

当page-start=5时，被delete的数据有146条．page-5的下一页是page-7，也就是说page-5和page-7记录的被删除的数据有146条
当page-start=6时，被delete的数据有123条，page-6的下一页是page-7，也就是说page-6和page-7记录的被删除的数据有123条
当page-start=7时，被delete的数据有0条，page-7的下一页是page-8，也就是说page-7和page-8记录的被删除的数据有0条.

综上也就是说PAGE-5记录了146条被删除的数据，PAGE-6记录了123条被删除的数据，PAGE-7记录了0条被删除的数据．也就是当PAGE-6发现自己要被干掉的时候,赶忙把数据转移到了PAGE-5里面(虽然那部分数据还是被标记为delete了)

总结

当某个页的数据要被全部删除时，可能会把一页数据转移到其它page,并记录转移的数据量到PAGE_N_RECS; 然后被全部删除之后，brother和parent都会将那一页的信息移除.



那么我们大量删除数据时，怎么使用ibd2sql去恢复呢？毕竟btree+中都不记录那些PAGE_ID了啊，虽然实际上那个PAGE里的信息还是有的.

很简单: 遍历所有page,不管是否是btree+的一部分，只要是符合索引信息的page,我们全部遍历，然后解析被标记为删除的数据即可，正好最新版(> v1.10)`--force`的逻辑就是这个．我已我们只要加上一force即可.

参考:

https://github.com/ddcw/ibd2sql

https://github.com/ddcw/ibd2sql/issues/66