mysql提升10倍count(*)的神器

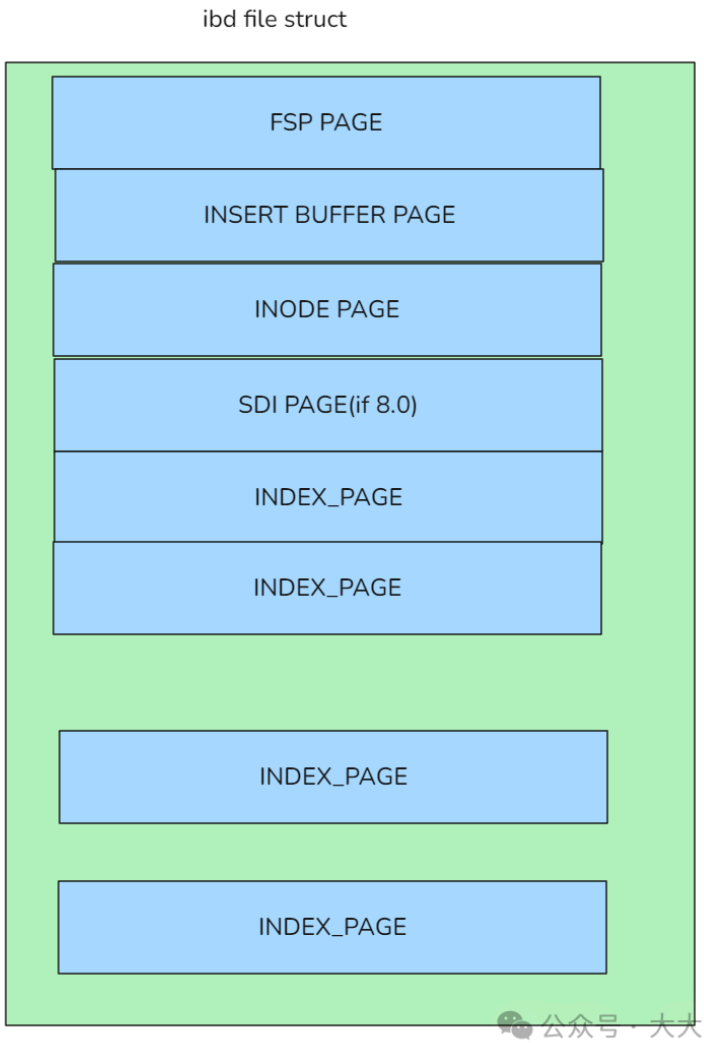
原创 大大刺猬 大大刺猬 2025年03月23日 08:03 上海

导读

之前做数据迁移之后，关于数据的一致性校验，我们是使用checksum来做的，也可以使用count(*)，但是都比较慢. 而数据校验的时候，数据实际上是静态的，没有业务使用的，欸，那我们**是不是就可以自己来统计行数呢**?

实现原理

我们来简单回顾下数据文件的结构，大概如下图:

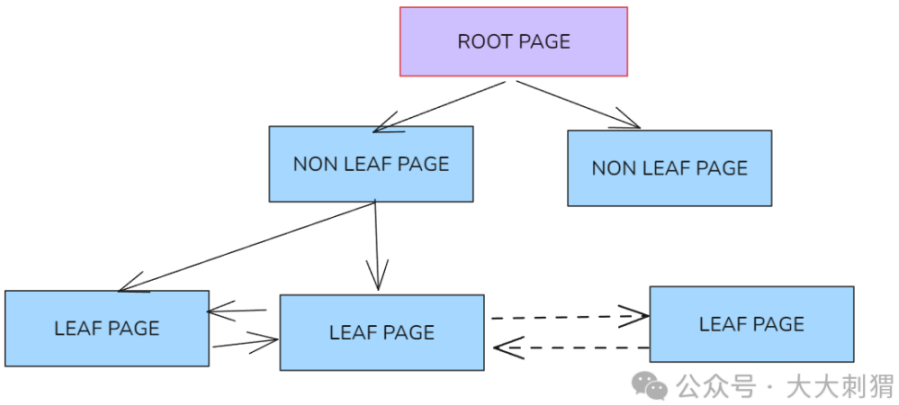


FSP PAGE主要是记录sdi信息，xdes和key之类的信息.
INODE PAGE主要是记录 segment信息，每2个segemnt为一个索引，第一对为主键索引信息，如果是8.0环境则第一对segment为sdi信息

> 主键索引存储了该表的完整数据.统计索引的行数,即为统计表的行数.

INDEX PAGE就是我们的索引行, 也就是数据行所在的页了. INDEX PAGE虽然结构是一样的, 但却分为**叶子节点**和**非叶子节点**供btr+使用, INODE PAGE中的每2个segment对应的就是 btr+里面的第一个非叶子节点(ROOT PAGE)和第一个叶子节点的

我们知道mysql的索引通常是btr+结构的, 比如:



叶子节点是相互连接的, 也就是找到第一个叶子节点, 即可顺藤摸瓜,找到所有的数据.

我们有必要解析每个叶子节点里面的每行数据吗? 其实没必要的, 因为叶子节点(INDEX PAGE)中的PAGE HEADER部分的PAGE_N_RECS就是记录本页数据有多少行.我们只需要统计所有叶子节点的PAGE_N_RECS即可.

理论上似乎是可行的, 那我们就来使用python实现它吧.

测试验证

编写代码的过程就略了，**我们还稍微做了下兼容性，使其能使用python2/python3直接执行，支持mysql5.7和8.0环境**

> 其它版本我这没得环境测试...

这个功能和ibd2sql比较像，我们就放到ibd2sql项目里面吧.

既然是统计行数的，那我们就叫它 `super_fast_count.py` 吧

> 希望它能对得起它的名字.

说了这么多，脚本呢？见文末.

先看看原生的mysql的count(*)速度

188W数据，耗时2秒！还是不错的成绩.

```
Your MySQL connection id is 8
Server version: 8.0.28 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

(root@127.0.0.1) [(none)]> select count(*) from db1.sbtest1;
+----------+
| count(*) |
+----------+
|  1889943 |
+----------+
1 row in set (2.13 sec)

(root@127.0.0.1) [(none)]>
```


神器

使用这个神器前，先深呼吸,吸收天地之灵气. 然后快速敲下如下命令:

```
time python3 super_fast_count.py /data/mysql_3314/mysqldata/db1/sbtest1.ibd
```

```
1 row in set (2.13 sec)

(root@127.0.0.1) [(none)]> ^DBye
16:38:59 [root@ddcw21 ei]#time python3 super_fast_count.py /data/mysql_3314/mysqldata/db1/sbtest1.ibd
TOTAL ROWS: 1889943     COST TIME: 0.17 seconds FILESIZE:0.92 GB

real    0m0.191s
user    0m0.106s
sys     0m0.084s
16:39:41 [root@ddcw21 ei]#
```

耗时为 0.19 秒. 差不多只有mysql原版的十分之一, 可喜可贺. 行数也是能对上的, 说明我们统计的结果也是对的.

> 经过测试发现, 敲命令的速度和脚本执行速度没有明显关系. 实际使用的时候可以放心的慢慢敲.

神器的兼容性测试

然后我们来测试下兼容性:
python2环境: 看起来慢一丢丢, 但无伤大雅

```
16:42:52 [root@ddcw21 ei]#time python2 super_fast_count.py /data/mysql_3314/mysqldata/db1/sbtest1.ibd
TOTAL ROWS: 1889943     COST TIME: 0.19 seconds FILESIZE:0.0 GB

real    0m0.209s
user    0m0.091s
sys     0m0.118s
16:42:54 [root@ddcw21 ei]#
```


mysql 5.7环境: 我没得大表了, 就这个10W行的意思意思吧. 0.04秒还是不错的成绩

```
16:43:37 [root@ddcw21 ei]#time python3 super_fast_count.py /data/mysql_3308/mysqldata/db1/sbtest1.ibd
TOTAL ROWS: 100000      COST TIME: 0.02 seconds FILESIZE:0.03 GB

real    0m0.048s
user    0m0.034s
sys     0m0.014s
16:43:38 [root@ddcw21 ei]#
```


总结

由于我们是直接读取的磁盘上的ibd数据文件, 所以**使用场景是有限**的, 而且使用时,会吃很多IO的.

常见使用场景:主从切换后数据快速校验, 大概估计下表的行数. 很闲的库. 没得脏数据的情况(show engine innodb看下LSN)
不建议使用的场景: 频繁更新的表, 服务器IO压力比较大.

公众号放源码不方便阅读, 可以到github上下载:

https://github.com/ddcw/ibd2sql/blob/main/super_fast_count.py