

[MYSQL] mysql空间问题案例分享

原创 大大刺猬 大大刺猬 2025年02月20日 14:30 上海

导读

某环境自上线以来, 空间使用越来越多. 总是扩空间也不是办法啊. 于是只能看能不能从数据库层面来释放一部分空间了.

分析思路

1. 首先想到的是归档, 即把不需要的历史数据备份到其它地方. 这个需要业务层配合, 实施难度大.
2. 对表进行压缩也是个方法, 但是这就影响到读写速度了. 牺牲性能来保证空间. 先做保留.
3. 还有个办法就是干掉未使用的索引. 可行性较高, 先查询下

我这里没得碎片问题, 实际场景可能还有碎片问题, 需要注意下.

处理过程

查询数据量

我们登录数据库查询下, 数据量和索引量的比较

```
1 SELECT
2     ROUND(SUM(data_length) / 1024 / 1024 / 1024, 2) AS DATA_GB,
3     ROUND(SUM(index_length) / 1024 / 1024 / 1024,
4           2) AS INDEX_GB,
5     ROUND(SUM(data_free) / 1024 / 1024 / 1024, 2) AS FREE_GB
6 FROM
7     information_schema.tables
8 WHERE
9     table_schema NOT IN ('sys' , 'performance_schema',
10                          'mysql',
11                          'information_schema');
```

实际环境查询出来 数据和索引大小差不多, data_free基本上为空

生产环境不方便截图, 我模拟的环境, 量太小, 不好看...

查询未使用的索引量

也就是说可能存在未使用的索引的, 然后我们查询下 `sys.schema_unused_indexes` 看下未使用的索引

```
1 select count(*) from sys.schema_unused_indexes;
```

查询出来发现有上万个未使用的索引... 数据库启动时间已经超过半年, 说明这部分索引超过半年未使用, 是可以释放的.

其实看对应的表名就能猜出来这部分表是应用程序自动建的, 索引应该也是自动建的, 而建这部分索引的时候并没有考虑后续是否使用...

查询未使用的索引大小

虽然删除这部分索引不会释放空间(OS层面), 但是后续对该表的数据更改操作, 会先使用这释放出来的那部分空间(数据库层面). 那我们就来统计下这部分空间有多大吧.

我们可以查询 `mysql.innodb_index_stats` 表看对应索引的大小, 虽然是统计信息, 但误差不会太大.

主键索引不算在索引大小里面, 而是算在数据大小里面的

我们查询 `stat_name=size` 的就是索引的page数量, 再乘上 `@@innodb_page_size` 就是索引大小了. 实际上 `information_schema.tables` 就是这么计算的.

```
1 -- information_schema.tables 中索引的大小
2 select data_length, index_length from information_schema.tables where ta
3
4 -- mysql.innodb_index_stats 中索引大小
5 select sum(stat_value)*@@innodb_page_size from mysql.innodb_index_stats
6
7 -- mysql.innodb_index_stats 中数据大小
8 select sum(stat_value)*@@innodb_page_size from mysql.innodb_index_stats
```

```
(root@127.0.0.1) [(none)]> select data_length,index_length from information_schema.tables where table_schema='db1' and table_name='sbtest1';
+-----+-----+
| DATA_LENGTH | INDEX_LENGTH |
+-----+-----+
| 456048640 | 509771776 |
+-----+-----+
1 row in set (0.00 sec)

(root@127.0.0.1) [(none)]> select sum(stat_value)*@@innodb_page_size from mysql.innodb_index_stats where database_name='db1' and table_name='sbtest1' and stat_name='size' and index_name = 'PRIMARY';
+-----+
| sum(stat_value)*@@innodb_page_size |
+-----+
| 456048640 |
+-----+
1 row in set (0.01 sec)

(root@127.0.0.1) [(none)]> select sum(stat_value)*@@innodb_page_size from mysql.innodb_index_stats where database_name='db1' and table_name='sbtest1' and stat_name='size' and index_name != 'PRIMARY';
+-----+
| sum(stat_value)*@@innodb_page_size |
+-----+
| 509771776 |
+-----+
1 row in set (0.00 sec)
```

公众号 · 大大刺猬

哈哈，扯远了。我们回到刚才的问题：查看未使用的索引的大小。

```
1  -- 查询未使用的索引的大小明细
2  SELECT
3      a.object_schema,
4      a.object_name,
5      a.index_name,
6      b.stat_value * @@innodb_page_size / 1024 / 1024 AS IDX_SIZE_MB
7  FROM
8      sys.schema_unused_indexes a
9      JOIN
10     mysql.innodb_index_stats b ON a.object_schema = b.database_name
11     AND a.object_name = b.table_name
12     AND a.index_name = b.index_name
13 WHERE
14     b.stat_name = 'size';
```

如果要总和的话，改为sum即可。

```
1  -- 查询未使用的索引的大小之和
2  SELECT round(sum(b.stat_value * @@innodb_page_size)/1024/1024/1024,2) as
3  FROM
4      sys.schema_unused_indexes a
5      JOIN
6      mysql.innodb_index_stats b ON a.object_schema = b.database_name
7      AND a.object_name = b.table_name
8      AND a.index_name = b.index_name
9  WHERE
10     b.stat_name = 'size';
```

这部分表很多其实都没得主键, 所以还涉及到主键的新增, 而主键的新增就涉及到表的重建了, 那么索引那部分空间就能释放出来了. 可能会有小伙伴问: (不新增字段)添加主键的话, 空间会不会增加啊? 答:不会, 甚至会减少, 因为没得索引的时候, mysql会自动新增个rowid来作为主键, 而人工指定主键的话, 就不会有这个rowid字段, 所以空间就会减少下来.

然后将上面的结果发给开发,让他们去做即可.

啰嗦一句

再啰嗦一句, 上面的 mysql.innodb_index_stats 中的信息我们我们也可以使用 ibd2sql_web 去验证. 该工具可以在浏览器上查看ibd的结构, 表中描述为 Number of leaf pages in the index 的page实际上是PAGE_LEVEL=1的PAGE, 其实该算为LEAF PAGE的...

前1页(4294967295) 当前页:400 字段数量:544 后1页(410)

k:521421,id:110920, k:526
k:621637,id:642674, k:630
k:663984,id:419791, k:669
k:693439,id:239272, k:697
k:714882,id:178855, k:717
k:731520,id:419923, k:734
k:746421,id:179464, k:748
k:760045,id:292230, k:762
k:771848,id:155675, k:775

```
{
  "fil_header": {
    "FIL_PAGE_SPACE_OR_CHKSUM": 2509921193,
    "FIL_PAGE_OFFSET": 400,
    "FIL_PAGE_PREV": 4294967295,
    "FIL_PAGE_NEXT": 410,
    "FIL_PAGE_LSN": 2862221049,
    "FIL_PAGE_TYPE": 17855,
    "FIL_PAGE_FILE_FLUSH_LSN": 0,
    "FIL_PAGE_SPACE_ID": 50363
  },
  "page_header": {
    "PAGE_N_DIR_SLOTS": 89,
    "PAGE_HEAP_TOP": 15896,
    "PAGE_N_HEAP": 33698,
    "PAGE_FREE": 4868,
    "PAGE_GARBAGE": 6528,
    "PAGE_LAST_INSERT": 15238,
    "PAGE_DIRECTION": 5,
    "PAGE_N_DIRECTION": 0,
    "PAGE_N_RECS": 544,
    "PAGE_MAX_TRX_ID": 0,
    "PAGE_LEVEL": 1,
    "PAGE_INDEX_ID": 56749,
    "PAGE_BTR_SEG_LEAF": {
      "SPACE_ID": 0,
      "PAGE_ID": 0,
      "PAGE_OFFSET": 0
    }
  }
}
```

公众号 · 大大刺猬

前1页(400) 当前页:410 字段数量:591 后1页(401)

公众号 · 大大刺猬

前1页(410) 当前页:401 字段数量:479 后1页(412)

公众号 · 大大刺猬

前1页(401) 当前页:412 字段数量:477 后1页(4294967295)

公众号 · 大大刺猬

db1	sbtest1	k_1	2025-02-17 17:02:24	n_diff_pfx01	354685	20	k
db1	sbtest1	k_1	2025-02-17 17:02:24	n_diff_pfx02	1746194	20	k,id
db1	sbtest1	k_1	2025-02-17 17:02:24	n_leaf_pages	2091	NULL	Number of leaf pages in the index
db1	sbtest1	k_1	2025-02-17 17:02:24	size	2405	NULL	Number of pages in the index

16 rows in set (0.00 sec)

(root@127.0.0.1) [sys]> select 544+591+479+477;

544+591+479+477
2091

公众号 · 大大刺猬

这统计信息准得离谱...