MySQL学习笔记（Day024：InnoDB_5 – heap_number / Buffer Pool）

MySQL学习

一. heap number

- heap_number表示页中每个 记录插入 的顺序 序号
  - 假设 插入 的数据是 `a, b, d, e, g`；则对应的 `heap_number` 为 `2，3，4，5，6`
  - `0`和`1` 被 `infimum` 和 `supermum` 所使用
    - `infimum` 对应最小的heap_number
    - `supermum` 对应最大的heap_number，随着数据的插入，该值会更新
  - update对heap_number没有影响
  - heap_number是物理的，存储在row的 `record_header` 字段中

```
            Leaf Page

+--------------------------------------+
|                                      |
|   +-----------+-------> heap number |
|   |           |              2       |
|   |   Row1    |              |       |
|   |    a      +-----+        |       |
|   +-----------+     |        |       |
|                 next|        |       |
|   +-----------+     |        3       |
|   |           <-----+        |       |
|   |   Row2    |              |       |
|   |    b      +-----+        |       |
|   +-----------+     |        |       |
|                 next|        |       |
|   +-----------+     |     a 4        |
|   |           <-----+     s |       |
|   |   Row3    |           c |       |
|   |    d      +-----+       |       |
|   +-----------+     |       |       |
|                 next|       |       |
|   +-----------+     |       5       |
|   |           <-----+       |       |
|   |   Row4    |             |       |
|   |    e      +-----+       |       |
|   +-----------+     |       |       |
|                 next|       |       |
|   +-----------+     |       6       |
|   |           <-----+       |       |
|   |   Row5    |             |       |
|   |    g      +-----+       v       |
|   +-----------+     v               |
|                  ......             |
+--------------------------------------+
```

```
-- 终端1
mysql> create table test_heap(a int primary key);
Query OK, 0 rows affected (0.13 sec)

mysql> insert into test_heap values (1);  -- 插入a=1的记录
Query OK, 1 row affected (0.03 sec)

mysql> begin;  -- 开启事物
Query OK, 0 rows affected (0.00 sec)

mysql> delete from test_heap where a=1;  -- 删除a=1的记录，此时加上了锁
Query OK, 1 row affected (0.00 sec)

-- 终端2
mysql>mysql> show variables like "%innodb_status_output_locks%";
+------------------------------+-------+
| Variable_name                | Value |
+------------------------------+-------+
| innodb_status_output_locks | OFF   |
+------------------------------+-------+

mysql> set global innodb_status_output_locks=1;
Query OK, 0 rows affected (0.00 sec)

mysql> pager less -- 使用类似linux中的less命令方式进行查看，可上下翻页
PAGER set to 'less'

mysql> show engine innodb status\G
-- ------------省略其他输出-------------
TABLE LOCK table `burn_test`.`test_heap` trx id 16943 lock mode IX
RECORD LOCKS space id 122 page no 3 n bits 72 index PRIMARY of table `burn_test`.`test_heap` trx id 16943 lock_mode X locks rec but not gap
Record lock, heap no 2 PHYSICAL RECORD: n_fields 3; compact format; info bits 32
 0: len 4; hex 80000001; asc     ;;  -- 插入的主键a=1，8的二进制1000，最高位为1，表示有符号的
 1: len 6; hex 00000000422f; asc     B/;;  -- 0x422f的 十进制就是16943 ，表示事物id（trx id）
 2: len 7; hex 2c000000450dcf; asc ,   E  ;;  -- roll pointer（回滚指针）
-- ------------省略其他输出-------------

-- space id 122  : 表空间id是122
-- page no 3     : 对应的页号是3 （表示第4个页，是root页）
-- heap no 2     : heap number是2 （表示是新插入的第一条记录）


-- heap no = 1 的一种情况
-- 终端1
mysql> rollback;
Query OK, 0 rows affected (0.00 sec)

mysql> set tx_isolation='repeatable-read';
Query OK, 0 rows affected (0.00 sec)

mysql> select * from test_heap where a>1 for update;
Empty set (0.00 sec)

-- 终端2
mysql> show engine innodb status\G
-- ------------省略其他输出-------------
Record lock, heap no 1 PHYSICAL RECORD: n_fields 1; compact format; info bits 0
 0: len 8; hex 73757072656d756d; asc supremum;;  -- 一条伪记录

Record lock, heap no 2 PHYSICAL RECORD: n_fields 3; compact format; info bits 0
 0: len 4; hex 80000001; asc     ;;
 1: len 6; hex 00000000422e; asc     B.;;
 2: len 7; hex ab000000470110; asc     G  ;;
-- ------------省略其他输出-------------
```

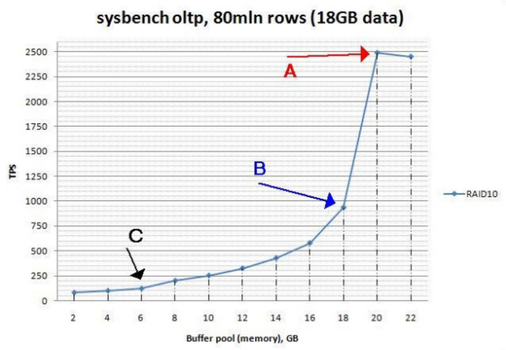
二. 缓冲池（Buffer Pool）

1. 缓冲池介绍

- 每次 读写 数据都是通过 `Buffer Pool`；
  - 当 `Buffer Pool` 中没有用户所需要的数据时，才去硬盘中获取；
- 通过 `innodb_buffer_pool_size` 进行设置总容量；
- 该值设置的越大越好；
- `innodb_buffer_pool_instances` 设置为多少个缓冲池；
  - 总容量 还是 `innodb_buffer_pool_size`
  - 设置 多个instance 可将热点打散，提高并发性能（*建议设置成CPU个数值，设置大了也没什么伤害*）
- Buffer Pool也是以 页（page） 为单位的，且大小和 `innodb_page_size` 一致；

```
                                  Clients
                                   ^   +
                          read data|   |
                     +-------------+   |
                     |                 |
                     |          +------+-------+
                     |          |   write data
                     |          |
                     |          |
               +--+--+--+--+--v--+---------------+  +--------+  +--------+     +--------+
               |  |     |  |     |       Buffer  |  | Buffer |  | Buffer |     | Buffer |
               | 16K | 16K | 16K |..... Pool 1   |  | Pool 2 |  | Pool 3 | ... | Pool N |   On Memory
               | p:5 |     | p:8 |               |  |        |  |        |     |        |
               +--^--+--+--+--+--+---------------+  +--------+  +--------+     +--------+
                  |          |
          read page|from disk |
                  |          |
+-----------------------------------------------------------------------------------------+
                  |          |
                  |      flush|to disk
                  +--+--+     |
                     |  |     |
                     +--+--+--v--+--+--+--+--+
                     |  |  |  |  |  |  |  |
  space id = 135     | 16K | 16K | 16K |.....| 16K | a.ibd         +-------+
                     | p:5 | p:8 |  |  |  |  |  |                         |
                     +--+--+--+--+--+--+--+--+--+                         |
                                                                          |
                     +--+--+--+--+--+--+--+--+--+                         |
                     |  |  |  |  |  |  |  |  |  |                         |
  space id = 166     | 16K | 16K | 16K |.....| 16K | xx.ibd        +----------+  On Disk
                     |  |  |  |  |  |  |  |  |  |                         |
                     +--+--+--+--+--+--+--+--+--+                         |
                                                                          |
                     +--+--+--+--+--+--+--+--+--+                         |
                     |  |  |  |  |  |  |  |  |  |                         |
  space id = 0       | 16K | 16K | 16K |.....| 16K | ibdata1.ibd   +-------+
                     |  |  |  |  |  |  |  |  |  |
                     +--+--+--+--+--+--+--+--+--+
```

2. Buffer Pool 性能测试



> 18G的测试数据,80M条记录；
>
> 1. 当 `Buffer Pool` 使用的内存超过 数据库的大小 时，比如20G（库中所有数据都在内存中），此时的性能有了很大的提升；
> 2. 该图测试的是 `TPS`（每秒事物数），sysbench中一个事物由18条SQL语句组成，即这里的QPS为4.5W
> 3. **内存减少 `10%` ，性能下降 `60%`**

3. Buffer Pool的管理

3.1 Buffer Pool 的组成

1. **Free List**
   - `Buffer Pool` 刚启动时，有一个个16K的空白的页，这些页就存放（链表串联）在 `Free List` 中
2. **LRU List**
   - 当读取一个数据页的时候，就从 `Free List` 中取出一个页，存入数据，并将该页放入到 `LRU List` 中
3. **Flush List**
   - 当 `LRU List` 中的页 第一次 被修改了，就将该页的 指针（page number） 放入了 `Flush List`（只要修改过，就放入，不管修改几次）
   - `Flush List` 中包含脏页（*数据经过修改，但是未刷入磁盘的页*）
   - `Flush list` 中存放的不是一个页，而是页的指针（page number）

3.2 查看Buffer Pool的状态

1. 使用命令 `show engine innodb status\G` 配合 `pager less`

```
mysql> show engine innodb status\G
-- ---------------省略其他输出-----------------

---BUFFER POOL 0
Buffer pool size   16383     -- 该Buffer Pool中有多少个页
Free buffers       16357     -- 该Buffer Pool中有多少个空白页（Free List），线上可能看到为0
Database pages     41        -- 该Buffer Pool中使用了多少页（LRU List）
Old database pages 0         -- old pages（见3.4）
Modified db pages  0         -- 脏页
Pending reads 0
Pending writes: LRU 0, flush list 0, single page 0
Pages made young 0, not young 0
0.00 youngs/s, 0.00 non-youngs/s  -- young表示old-->new的状态
Pages read 41, created 0, written 20
0.00 reads/s, 0.00 creates/s, 0.00 writes/s
No buffer pool page gets since the last printout
Pages read ahead 0.00/s, evicted without access 0.00/s, Random read ahead 0.00/s
LRU len: 41, unzip_LRU len: 0
I/O sum[0]:cur[0], unzip sum[0]:cur[0]

-- ---------------省略其他输出-----------------
```

> **注意：**
> `Free buffers` + `Database pages` < `Buffer pool size`

2. 使用元数据表 `information_schema.INNODB_BUFFER_POOL_STATS`

```
mysql> select * from information_schema.INNODB_BUFFER_POOL_STATS\G
*************************** 1. row ***************************
                         POOL_ID: 0
                       POOL_SIZE: 16383  -- 该Buffer Pool中有多少个页
                  FREE_BUFFERS: 16357  -- 该Buffer Pool中有多少个空白页（Free List），线上可能看到为0
                DATABASE_PAGES: 41     -- 该Buffer Pool中使用了多少页（LRU List）
            OLD_DATABASE_PAGES: 0      -- old pages （见3.4）
        MODIFIED_DATABASE_PAGES: 0      -- 脏页
             PENDING_DECOMPRESS: 0
                  PENDING_READS: 0
              PENDING_FLUSH_LRU: 0
             PENDING_FLUSH_LIST: 0
               PAGES_MADE_YOUNG: 0
           PAGES_NOT_MADE_YOUNG: 0
          PAGES_MADE_YOUNG_RATE: 0
      PAGES_MADE_NOT_YOUNG_RATE: 0
             NUMBER_PAGES_READ: 41
          NUMBER_PAGES_CREATED: 0
          NUMBER_PAGES_WRITTEN: 20
               PAGES_READ_RATE: 0
             PAGES_CREATE_RATE: 0
            PAGES_WRITTEN_RATE: 0
              NUMBER_PAGES_GET: 1041
                      HIT_RATE: 0
  YOUNG_MAKE_PER_THOUSAND_GETS: 0
NOT_YOUNG_MAKE_PER_THOUSAND_GETS: 0
       NUMBER_PAGES_READ_AHEAD: 0
     NUMBER_READ_AHEAD_EVICTED: 0
            READ_AHEAD_RATE: 0
    READ_AHEAD_EVICTED_RATE: 0
               LRU_IO_TOTAL: 0
             LRU_IO_CURRENT: 0
           UNCOMPRESS_TOTAL: 0
         UNCOMPRESS_CURRENT: 0

-- ---------------省略其他输出-----------------


mysql> select * from information_schema.INNODB_BUFFER_PAGE_LRU limit 1\G
*************************** 1. row ***************************
          POOL_ID: 0
      LRU_POSITION: 0
             SPACE: 0    -- space id 表空间号
       PAGE_NUMBER: 7    -- 对应的页号
         PAGE_TYPE: SYSTEM
        FLUSH_TYPE: 1
         FIX_COUNT: 0
         IS_HASHED: NO
NEWEST_MODIFICATION: 4005630175 -- 该页最近一次（最新）被修改的LSN值
OLDEST_MODIFICATION: 0          -- 该页在Buffer Pool中第一次被修改的LSN值，FLushList是根据该值进行排序的
                                -- 该值越小，表示该页应该最先被刷新
       ACCESS_TIME: 729305074
        TABLE_NAME: NULL
        INDEX_NAME: NULL
    NUMBER_RECORDS: 0
         DATA_SIZE: 0
   COMPRESSED_SIZE: 0
        COMPRESSED: NO
            IO_FIX: IO_NONE
            IS_OLD: NO
   FREE_PAGE_CLOCK: 0
1 row in set (0.01 sec)
-- ---------------省略其他输出-----------------
```

3.3 Buffer Pool 在线调整

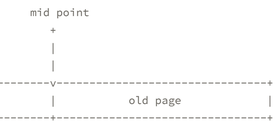
从 `MySQL 5.7` 开始，可以在线修改 `innodb_buffer_pool_size`

```
mysql> show variables like "%innodb_buffer_pool_size%";
+-------------------------+------------+
| Variable_name           | Value      |
+-------------------------+------------+
| innodb_buffer_pool_size | 1073741824 |
+-------------------------+------------+
1 row in set (0.00 sec)

mysql> set global innodb_buffer_pool_size=2*1024*1024*1024;  -- 扩大
Query OK, 0 rows affected (0.00 sec)

mysql> show variables like "%innodb_buffer_pool_size%";
+-------------------------+------------+
| Variable_name           | Value      |
+-------------------------+------------+
| innodb_buffer_pool_size | 2147483648 |
+-------------------------+------------+
1 row in set (0.00 sec)

mysql> set global innodb_buffer_pool_size=1*1024*1024*1024; -- 缩小，没修改的页被丢弃，修改的需要刷回磁盘
Query OK, 0 rows affected (0.00 sec)

mysql> show variables like "%innodb_buffer_pool_size%";
+-------------------------+------------+
| Variable_name           | Value      |
+-------------------------+------------+
| innodb_buffer_pool_size | 1073741824 |
+-------------------------+------------+
1 row in set (0.01 sec)
```

> `MySQL 5.7之前` 的版本，修改该值，需要 重启

3.4 LRU List 的管理

- **使用 `mid point` 的LRU算法**
  - 当该页被 第一次读取 时，将该页先放在 `mid point` 的位置（*因为无法保证一定是活跃*）；
  - 当被读到 第二次 时，才将改页放入到 `new page` 的首部；
  - `innodb_old_blocks_pct` 参数控制 `mid point` 的位置，默认是 `37` ，即 `3/8` 的位置



LRU 中new page和old page是在一个链表上的，想像成排队，访问多的就从mid point排到了链表的前面
然后后冷的页就慢慢被挤到了old page中，如果old中的数据继续被多次访问，还是会回到new中

```
1 : mid --> new
2 : mid --> old --> new
3 : mid --> old --> 刷回磁盘
4 : new --> old --> 刷回磁盘
```

> 1. 当Free List中没有空余的页时，就需要从 `old page` 中最后的页（被淘汰的页）给取出，给新的查询所使用
> 2. 如果被淘汰的页是脏页（page number在Flush List中），则需要先刷回磁盘后，再给新的查询使用

```
mysql> show variables like "%innodb_old_blocks_pct%";
+-----------------------+-------+
| Variable_name         | Value |
+-----------------------+-------+
| innodb_old_blocks_pct | 37    |
+-----------------------+-------+
1 row in set (0.00 sec)
```

- **避免扫描语句污染LRU**
  当使用 `select * from tablename;` 时，该语句会读取某个页很多次（即该页可能被读取了两次以上， 读取一条记录，就需要读一次页 ）
  - `innodb_old_blocks_time`

    ```
    mysql> show variables like "%innodb_old_blocks_time%";
    +------------------------+-------+
    | Variable_name          | Value |
    +------------------------+-------+
    | innodb_old_blocks_time | 1000  |  -- 设置为1s
    +------------------------+-------+
    1 row in set (0.00 sec)
    ```

> 1. 当该页被 第一次 读取时，将该页放在 `mid point` 位置，但是随后 无论你读多少次 ，我在这 `innodb_old_blocks_time` 的时间内都不管（ 都视作只读取了一次 ），等这个时间过去了（时间到），如果该页 还是被读取 了，我才把这个页放到 `new page` 的首部。
> 2. 通常 `select *` 扫描操作不会高于1秒，一个页很快就被扫完了。

4. Buffer Pool 的预热

> Buffer Pool预热

- 在MySQL启动后（*MySQL5.6之前*），Buffer Pool中页的数据是空的，需要大量的时间才能把磁盘中的页读入到内存中，导致启动后的一段时间性能很差。

```
-- 使用该方法预热，强制扫描，将数据刷入buffer pool，但是不能真正将热点数据放入buffer pool
select count(1) from table force index(PRIMARY)
select count(1) from table FORCE index(index name)
```

- 在 `MySQL 5.6` 以后，可以在 停机 的时候 `dump` 出 `buffer pool` 的数据（space，page number），然后在 启动 的时候 `Load` 进 `buffer pool`
- 该功能可以让MySQL启动时 自动预热 ，无需人工干预。

```
mysql> show variables like "%innodb_buffer_pool%";
+-------------------------------------+----------------+
| Variable_name                       | Value          |
+-------------------------------------+----------------+
| innodb_buffer_pool_chunk_size       | 134217728      |
| innodb_buffer_pool_dump_at_shutdown | ON             |  -- 在停机时dump出buffer pool中的（space,page）
| innodb_buffer_pool_dump_now         | OFF            |  -- set 一下，表示现在就从buffer pool中dump
| innodb_buffer_pool_dump_pct         | 40             |  -- dump的百分比，是每个buffer pool文件，而不是整体
| innodb_buffer_pool_filename         | ib_buffer_pool |  -- dump出的文件的名字
| innodb_buffer_pool_instances        | 8              |
| innodb_buffer_pool_load_abort       | OFF            |
| innodb_buffer_pool_load_at_startup  | ON             |  -- 启动时加载dump的文件，恢复到buffer pool中
| innodb_buffer_pool_load_now         | OFF            |  -- set一下，表示现在加载 dump的文件
| innodb_buffer_pool_size             | 1073741824     |
+-------------------------------------+----------------+
10 rows in set (0.00 sec)


shell> head ib_buffer_pool  # dump出来的文件
120,3
120,2
120,1
120,0
106,4539
106,4538
106,4537
106,4536
106,4535
106,4534
```

> 1. **dump的越多，启动的越慢**
> 2. 频繁的手工dump（ `set innodb_buffer_pool_dump_now = 1` ），会导致Buffer Pool中的数据越来越少，是因为设置了 `innodb_buffer_pool_dump_pct`
> 3. 如果做了高可用，可以定期dump，然后将该dump的文件传送到slave上，然后直接load（ `set innodb_buffer_pool_load_now = 1` ）（*slave上的（Space，Page）和Master上的 大致相同* ）
> 4. `load now` 和 `dump now` 都是 异步 在后台加载的，返回的速度很

```
# mysql 启动
sehll> cat error.log
## ---------------省略其他输出-----------------
2016-01-10T20:17:31.263901+08:00 0 [Note] InnoDB: Loading buffer pool(s) from /data/mysql_data/5.7/ib_buffer_pool
## ---------------省略其他输出-----------------
2016-01-10T20:17:31.365768+08:00 0 [Note] InnoDB: Buffer pool(s) load completed at 160110 20:17:31  # 速度还是很快的

# mysql 停机
shell> cat error.log
## ---------------省略其他输出-----------------
2016-01-11T08:47:20.067308+08:00 0 [Note] InnoDB: Dumping buffer pool(s) to /data/mysql_data/5.7/ib_buffer_pool
2016-01-11T08:47:20.067730+08:00 0 [Note] InnoDB: Buffer pool(s) dump completed at 160111  8:47:20
## ---------------省略其他输出-----------------


# 查看当前buffer pool中的数据的条数
shell> wc -l ib_buffer_pool
524 ib_buffer_pool


mysql> set global innodb_buffer_pool_dump_now=1;
Query OK, 0 rows affected (0.00 sec)

mysql> show status like 'Innodb_buffer_pool_dump_status';
+---------------------------------+-------------------------------------------------+
| Variable_name                   | Value                                           |
+---------------------------------+-------------------------------------------------+
| Innodb_buffer_pool_dump_status | Buffer pool(s) dump completed at 160111  9:04:21 |
+---------------------------------+-------------------------------------------------+
1 row in set (0.00 sec)
-- 已经完成


mysql> wc -l ib_buffer_pool
524 ib_buffer_pool  # 发现和原来一致，并没有减少为原来的40，似乎和innodb_buffer_pool_dump_pct设置的不符合


-- 由于innodb_buffer_pool_dump_pct是针对每个buffer pool的百分比，应该是由于我的数据太小，没有达到该比例（40%），所以这里全部dump出来了
-- 测试的时候可以设置小一点
mysql> set global innodb_buffer_pool_dump_pct=2; -- 设置成2%
Query OK, 0 rows affected (0.00 sec)

mysql> set global innodb_buffer_pool_dump_now=1;
Query OK, 0 rows affected (0.00 sec)

mysql> show status like 'Innodb_buffer_pool_dump_status';
+----------------------------------+-------------------------------------------------+
| Variable_name                    | Value                                           |
+----------------------------------+-------------------------------------------------+
| Innodb_buffer_pool_load_status | Buffer pool(s) load completed at 160111  9:05:38 |
+----------------------------------+-------------------------------------------------+
1 row in set (0.00 sec)


shell> wc -l ib_buffer_pool  # dump前
524 ib_buffer_pool
shell> wc -l ib_buffer_pool  # dump后
23 ib_buffer_pool
```

> innodb_buffer_pool_dump_pct
> 该百分比（N<100）不是你当前buffer pool的总的数据（总页数）的N%，而是你每个buffer pool实例中最近使用的页的N%
> *if there are 4 buffer pools with 100 pages each, and innodb_buffer_pool_dump_pct is set to 25, the 25 most recently used pages from each buffer pool are dumped*