MySQL学习笔记（Day025：InnoDB_6 – Buffer Pool与压缩页/CheckPoint/LSN）

MySQL学习

一. 思考题解析

- **查看Buffer Pool中的Flush List**

> **不要在线上操作该SQL语句，开销较大**

```
SELECT
    pool_id,
    lru_position,
    space,
    page_number,
    table_name,
    oldest_modification,
    newest_modification
FROM
    information_schema.INNODB_BUFFER_PAGE_LRU
WHERE
    oldest_modification <> 0
        AND oldest_modification <> newest_modification; -- 如果没有脏页，结果集为空
```

二. Buffer Pool与压缩页

2.1. 查找Buffer Pool中的压缩页

```
mysql> desc information_schema.INNODB_BUFFER_PAGE_LRU;
+---------------------+---------------------+------+-----+---------+-------+
| Field               | Type                | Null | Key | Default | Extra |
+---------------------+---------------------+------+-----+---------+-------+
| POOL_ID             | bigint(21) unsigned | NO   |     | 0       |       |
| LRU_POSITION        | bigint(21) unsigned | NO   |     | 0       |       |
| SPACE               | bigint(21) unsigned | NO   |     | 0       |       |
| PAGE_NUMBER         | bigint(21) unsigned | NO   |     | 0       |       |
| PAGE_TYPE           | varchar(64)         | YES  |     | NULL    |       |
| FLUSH_TYPE          | bigint(21) unsigned | NO   |     | 0       |       |
| FIX_COUNT           | bigint(21) unsigned | NO   |     | 0       |       |
| IS_HASHED           | varchar(3)          | YES  |     | NULL    |       |
| NEWEST_MODIFICATION | bigint(21) unsigned | NO   |     | 0       |       |
| OLDEST_MODIFICATION | bigint(21) unsigned | NO   |     | 0       |       |
| ACCESS_TIME         | bigint(21) unsigned | NO   |     | 0       |       |
| TABLE_NAME          | varchar(1024)       | YES  |     | NULL    |       |
| INDEX_NAME          | varchar(1024)       | YES  |     | NULL    |       |
| NUMBER_RECORDS      | bigint(21) unsigned | NO   |     | 0       |       |
| DATA_SIZE           | bigint(21) unsigned | NO   |     | 0       |       |
| COMPRESSED_SIZE     | bigint(21) unsigned | NO   |     | 0       |       | -- 压缩页的大小
| COMPRESSED          | varchar(3)          | YES  |     | NULL    |       | -- 该页是否被压缩
| IO_FIX              | varchar(64)         | YES  |     | NULL    |       |
| IS_OLD              | varchar(3)          | YES  |     | NULL    |       |
| FREE_PAGE_CLOCK     | bigint(21) unsigned | NO   |     | 0       |       |
+---------------------+---------------------+------+-----+---------+-------+
20 rows in set (0.00 sec)
```

```
mysql> select
    ->     table_name, space, page_number,
    ->     index_name, compressed, compressed_size
    -> from
    ->     information_schema.INNODB_BUFFER_PAGE_LRU
    -> where
    ->     compressed = 'yes' limit 10;
+------------+-------+-------------+------------+------------+-----------------+
| table_name | space | page_number | index_name | compressed | compressed_size |
+------------+-------+-------------+------------+------------+-----------------+
| NULL       |   104 |        2669 | NULL       | YES        |            4096 |
| NULL       |   104 |        2671 | NULL       | YES        |            4096 |
| NULL       |   104 |        2674 | NULL       | YES        |            4096 |
| NULL       |   104 |        2677 | NULL       | YES        |            4096 |
| NULL       |   104 |        2679 | NULL       | YES        |            4096 |
| NULL       |   104 |        2682 | NULL       | YES        |            4096 |
| NULL       |   104 |        2685 | NULL       | YES        |            4096 |
| NULL       |   104 |        2687 | NULL       | YES        |            4096 |
| NULL       |   104 |        2686 | NULL       | YES        |            4096 |
+------------+-------+-------------+------------+------------+-----------------+
10 rows in set (0.04 sec)

mysql> select
    ->     table_id, name, space, row_format, zip_page_size
    -> from
    ->     information_schema.INNODB_SYS_TABLES
    -> where
    ->     space = 104;
+----------+-----------------------------+-------+------------+---------------+
| table_id | name                        | space | row_format | zip_page_size |
+----------+-----------------------------+-------+------------+---------------+
|      104 | employees/employee_comps_1 |   104 | Compressed |          4096 |
+----------+-----------------------------+-------+------------+---------------+
1 row in set (0.00 sec)

mysql> show create table employees.employee_comps_1\G
*************************** 1. row ***************************
       Table: employee_comps_1
Create Table: CREATE TABLE `employee_comps_1` (
  `emp_no` int(11) NOT NULL,
  `birth_date` date NOT NULL,
  `first_name` varchar(14) NOT NULL,
  `last_name` varchar(16) NOT NULL,
  `gender` enum('M','F') NOT NULL,
  `hire_date` date NOT NULL,
  PRIMARY KEY (`emp_no`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=4 -- 之前确实是指定压缩
1 row in set (0.00 sec)
```

2.2. 压缩页在内存中的存放

- **压缩页存在于 `unzip_LRU` 中**

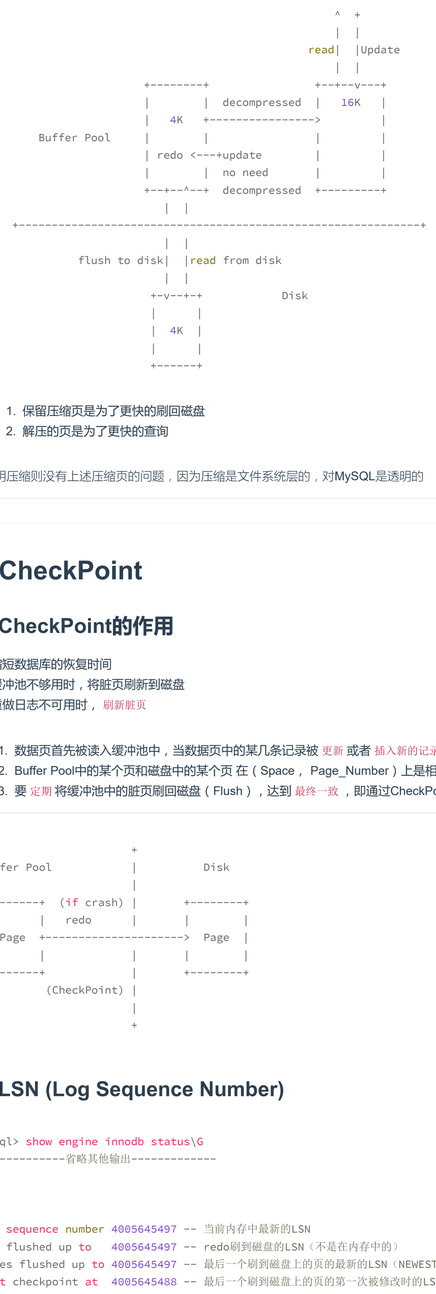
```
mysql> show engine innodb status\G
-- ----------省略其他输出------------
---BUFFER POOL 0
Buffer pool size   16383
Free buffers       15540
Database pages     651
Old database pages 237
Modified db pages  0
Pending reads 0
Pending writes: LRU 0, flush list 0, single page 0
Pages made young 0, not young 0
0.00 youngs/s, 0.00 non-youngs/s
Pages read 589, created 62, written 124
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: 651, unzip_LRU len: 382  -- 压缩页LRU的长度在 buffer pool 1 中的长度是382
I/O sum[0]:cur[0], unzip sum[0]:cur[0]
-- ----------省略其他输出------------
```

- **伙伴算法**

  - 磁盘中存放压缩页（row_format=compressed），假设 `key_block_size=8K` ， `Buffer Pool` 的页大小是 `16K`
  - 向 `Free List` 中申请空闲的页，如果没有空闲页，则向 `LRU List` 申请页，如果LRU满了，则找LRU中最后的一个页，如果最后的页是 `脏页` ，则做 `flush` 操作，最后得到一个空白的页（16K）
  - 该16K的空白页，就给8K的压缩页使用，这样就 `多出一个8K的空间` ，该空间会 `移到8K的Free List` 中去
  - 如果有一个4K的压缩页，就把8K的Free list中的空白页给他用，然后 `多余的4K的空间移到4K的Free List` 中去

> 1. 通过上述方式，`不同大小的页可以在同一个Buffer Pool中使用`（*可以简单的认为Free List是 按照页大小 来进行 划分 的*）。
> 2. `不能` 根据页大小来划分缓冲池，缓冲池中页的大小就是 `固定的大小` （ `等于innodb_page_size` ）
> 3. LRU List和Flush List 不需要按照页大小划分，都是统一的 `innodb_page_size` 大小

- **压缩页在内存中保留**
  - 被压缩的页需要在Buffer Pool中 解压 。
  - 原来的压缩页 保留 在Buffer Pool中。
  - 缺点是压缩页占用了Buffer。 Pool的空间，对于热点数据来说，相当于内存小了，可能造成性能下降（热点空间变小）。
    - 所以在开启了压缩后，Buffer Pool的空间要相应增大；
    - 如果启用压缩后节省的磁盘IO能够 抵消 掉Buffer； Pool"空间变小"所带来的性能下降，那整体性能还是会上涨；
    - 启用压缩的前提是，内存尽可能的大；
  - 压缩页保留的原因是为了在更新数据的时候，将 redo 添加到压缩页的空闲部分，如果要刷回磁盘，可以 直接 将该压缩页刷回去。如果该页被写满，则做一次 reorganize 操作（在此之前也要做解压），真的写满了才做分裂。



1. 保留压缩页是为了更快的刷回磁盘
2. 解压的页是为了更快的查询

> 透明压缩则没有上述压缩页的问题，因为压缩是文件系统层的，对MySQL是透明的

三. CheckPoint

3.1. CheckPoint的作用

- 缩短数据库的恢复时间
- 缓冲池不够用时，将脏页刷新到磁盘
- 重做日志不可用时， 刷新脏页

> 1. 数据页首先被读入缓冲池中，当数据页中的某几条记录被 更新 或者 插入新的记录 时，所有的操作都是在Buffer Pool 先 完成的；
> 2. Buffer Pool中的某个页和磁盘中的某个页 在（Space， Page_Number）上是相同的，但是其内容可能是不同的（Buffer Pool中的被更新过了），形成了 脏页 ；
> 3. 要 定期 将缓冲池中的脏页刷回磁盘（Flush），达到 最终一致 ，即通过CheckPoint机制来刷脏页；

3.2. LSN (Log Sequence Number)

```
mysql> show engine innodb status\G
-- ----------省略其他输出------------
---
LOG
---
Log sequence number 4005645497 -- 当前内存中最新的LSN
Log flushed up to   4005645497 -- redo刷到磁盘的LSN（不是在内存中的）
Pages flushed up to 4005645497 -- 最后一个刷到磁盘上的页的最新的LSN（NEWEST_MODIFICATION）
Last checkpoint at  4005645488 -- 最后一个刷到磁盘上的页的第一次被修改时的LSN（OLDEST_MODIFICATION）
```

> **`LSN(Log Sequence Number)` 是一个字节数。**
>
> **注意：**
>
> 1. **`Log sequence number`** 和 **`Log flushed up`** 这两个LSN可能会不同，运行过程中后者可能会 小于 前者，因为redo日志也是先在内存中更新，再刷到磁盘的。
>
> 2. **`Pages flushed up`** 与 **`Last checkpoint`** 其实都指向了 最后一个 刷新到磁盘的 页 ，只是 `Pages flushed up` 代表了页中的 `NEWEST_MODIFICATION` ，而 `Last checkpoint` 代表了页中的 `OLDEST_MODIFICATION` 。
>   - `FLUSH LIST` 使用 `OLDEST_MODIFICATION` 进行记录并排序，那在刷新脏页时， `CheckPoint` 的 `LSN` 值就对应的是 当前刷新到某个页 的 `OLDEST_MODIFICATION` ；
>   - 当某个页 只被修改过一次 ，则 `Pages flushed up` 与 `Last checkpoint` 会相等，反之多次修改，则 `Pages flushed up` 大于 `Last checkpoint` ；
>   - 在恢复时，从 `CheckPoint` 开始恢复，如果 当前页的LSN大于CheckPoint的LSN ，则表示不需要恢复了；

1. **日志（redo）中的LSN：**
   - 假设当前的LSN为 `C` ，此时对某个页做修改，则会产生 `M` 个字节的 日志 （需要写入M个字节的日志），那此时的 `LSN` 则为 `C+M` 。依次类推，LSN是一个 单调递增 的值（字节数）。
   - 日志中的LSN代表了日志一共写入了多少个字节。
2. **页中的LSN：**
   页中也存在LSN，表示该页被修改的时候，多应的日志的LSN是多少；

```
mysql> desc information_schema.INNODB_BUFFER_PAGE_LRU;
+---------------------+---------------------+------+-----+---------+-------+
| Field               | Type                | Null | Key | Default | Extra |
+---------------------+---------------------+------+-----+---------+-------+
| POOL_ID             | bigint(21) unsigned | NO   |     | 0       |       |
| LRU_POSITION        | bigint(21) unsigned | NO   |     | 0       |       |
| SPACE               | bigint(21) unsigned | NO   |     | 0       |       |
| PAGE_NUMBER         | bigint(21) unsigned | NO   |     | 0       |       |
| PAGE_TYPE           | varchar(64)         | YES  |     | NULL    |       |
| FLUSH_TYPE          | bigint(21) unsigned | NO   |     | 0       |       |
| FIX_COUNT           | bigint(21) unsigned | NO   |     | 0       |       |
| IS_HASHED           | varchar(3)          | YES  |     | NULL    |       |
| NEWEST_MODIFICATION | bigint(21) unsigned | NO   |     | 0       |       | -- 该页最近一次（最新）被修改的LSN值
| OLDEST_MODIFICATION | bigint(21) unsigned | NO   |     | 0       |       | -- 该页在Buffer Pool中第一次被修改的LSN值
| ACCESS_TIME         | bigint(21) unsigned | NO   |     | 0       |       |
| TABLE_NAME          | varchar(1024)       | YES  |     | NULL    |       |
| INDEX_NAME          | varchar(1024)       | YES  |     | NULL    |       |
| NUMBER_RECORDS      | bigint(21) unsigned | NO   |     | 0       |       |
| DATA_SIZE           | bigint(21) unsigned | NO   |     | 0       |       |
| COMPRESSED_SIZE     | bigint(21) unsigned | NO   |     | 0       |       |
| COMPRESSED          | varchar(3)          | YES  |     | NULL    |       |
| IO_FIX              | varchar(64)         | YES  |     | NULL    |       |
| IS_OLD              | varchar(3)          | YES  |     | NULL    |       |
| FREE_PAGE_CLOCK     | bigint(21) unsigned | NO   |     | 0       |       |
+---------------------+---------------------+------+-----+---------+-------+
20 rows in set (0.00 sec)
```

> Page中的LSN主要用在 恢复 的时候
> Page中的LSN放在页头

3. **CheckPoint LSN**
   每个数据库中也有一个LSN，表示 最后一个刷新到磁盘的页的LSN ，表明了该LSN之前的数据都刷回到磁盘了，且如果要做恢复操作，也只要从当前这个 `CheckPoint LSN` 开始恢复。

> CheckPoint LSN 写在redo log 的 前2K 空间中

1. 日志中的**`LSN` = `CheckPoint`的`LSN` ，则表示所有页都已经刷回磁盘**
2. 日志中的**`LSN` > `CheckPoint`的`LSN` ，则表示还有页没刷到磁盘；如果是宕机，则需要用日志恢复。**
3. 日志中的**`LSN` < `CheckPoint`的`LSN` ，则报错**

3.3. CheckPoint的分类

- Sharp CheckPoint
  - 将所有的脏页刷新回磁盘
  - 通常在数据库关闭的时候
  - 刷新时系统hang住
  - `innodb_fast_shutdown={1|0}`
- Fuzzy CheckPoint
  - 将部分脏页刷新回磁盘
  - 对系统影响较小
  - `innodb_io_capacity`
    - 最小限制为100
    - 一次最多刷新脏页的能力，与IOPS相关
      - SSD 可以设置在4000-8000
      - SAS 最多设置在800多（IOPS在1000左右）

3.4. 刷新

1. **Master Thread Checkpoint**
   - 从 `FLUSH_LIST` 中刷新
2. **FLUSH_LRU_LIST Checkpoint**
   - 从 `LRU_LIST` 中刷新（即使不在脏页链表中）
     - 5.5以前需要保证在 `LRU_LIST` 尾部要有100个空闲页（可替换的页），即 刷新一部分数据 ，保证有100个空闲页
   - `innodb_lru_scan_depth` – **每次进行 `LRU_LIST` 刷新的脏页的数量**
     - 应用到 每个 Buffer Pool实例，总数即为该值乘以Buffer Pool的实例个数，如果超过 `innodb_io_capacity` 是不合理的
     - 建议该值不能超过 `innodb_io_capacity` / `innodb_buffer_pool_instances`
3. **Async/Sync Flush Checkpoint**
   - 重做日志重用
4. **Dirty Page too much Checkpoint**
   - `innodb_max_dirty_pages_pct` 参数控制