sys schema

开发人员利器——MYSQL 5.7 SYS库

PHPCON
www.phpconchina.com

个人介绍

- 赖明星 （mingxinglai.com）
- 网易云计算工程师，网易RDS核心开发人员
- IMG社区核心成员，多次在IMG社区、淘宝MySQL内核月报、DBA Plus社区投稿或分享
- 对关系型数据库和NoSQL数据库具有浓厚兴趣

InsideMySQL

IMG 有态度的MySQL社区

www.innomysql.net

为什么要在一个开发人员聚集的地方讲MySQL

目录

Introduction: MySQL在数据字典方面的演变历史

- MySQL 4.1 提供了information_schema 数据字典
- MySQL 5.5 提供了performance_schema 性能字典
- MySQL 5.6 默认开启performance_schema
- MySQL 5.7 提供了 sys系统数据库

Introduction: sys schema的组成和作用

- sys schema包含了一些列视图、函数和存储过程
- sys schema用以帮助DBA和开发分析定位问题

For Linux users I like to compare performance_schema to /proc, and SYS to vmstat.

参考： MySQL Server Blog

Introduction: 为什么需要sys schema

- performance_schema数据量太大，MySQL 5.6 performance_schema有52张表，MySQL 5.7有87张表，未来还可能增加
- performance_schema数据太专业
- 用户需要的是解决问题的答案，而不是一堆数据

目录

Installation

- set performance_schema=ON
- MySQL 5.6+
- 5.7默认安装

```
git clone https://github.com/MarkLeith/mysql-sys.git /tmp/sys
cd /tmp/sys
mysql -u user -p < sys_<version>.sql
```

Installation

- 检查是否安装完成
- 本次分享基于1.5.0版本

```
mysql> select * from sys.version;
+-------------+---------------+
| sys_version | mysql_version |
+-------------+---------------+
| 1.5.0       | 5.7.12-v1-log |
+-------------+---------------+
1 row in set (0.00 sec)
```

目录

Views: 两种形式

- 对于每一个视图，都有两种形式
- 一种便于人类阅读，一种便于工具处理(以”x$”开头)

```
mysql> select * from host_summary_by_file_io;
+-------------+------+------------+
| host        | ios  | io_latency |
+-------------+------+------------+
| background  | 2143 | 266.48 ms  |
| 172.17.42.1 | 1748 | 116.52 ms  |
+-------------+------+------------+
2 rows in set (0.00 sec)
```

```
mysql> select * from x$host_summary_by_file_io;
+-------------+------+--------------+
| host        | ios  | io_latency   |
+-------------+------+--------------+
| background  | 2148 | 266558291670 |
| 172.17.42.1 | 1748 | 116518395300 |
+-------------+------+--------------+
2 rows in set (0.00 sec)
```

Views: sys schema如何帮助使用

1. 谁使用了最多的资源
2. 大部分连接来自哪里
3. 在哪个文件产生了最多的IO，它的IO模式是怎么样的
4. 哪张表被访
5. 哪些语句延
6. 哪些SQL语句使用了磁盘临时表
7. 哪张表占用了最多的buffer pool空间

Views:从"使用者"角度看代价

user_summary

user_summary_by_file_io

user_summary_by_file_io_type

user_summary_by_stages

user_summary_by_statement_latency

user_summary_by_statement_type

host_summary

host_summary_by_file_io

host_summary_by_file_io_type

host_summary_by_stages

host_summary_by_statement_latency

host_summary_by_statement_type

Views: 从"使用者"角度看代价

```
mysql> select * from user_summary limit 1\G
                    user: rdsadmin
              statements: 133068443
       statement_latency: 5.17 w
   statement_avg_latency: 23.52 ms
            table_scans: 741790
               file_ios: 11062758
         file_io_latency: 3.34 h
    current_connections: 10
      total_connections: 3292
           unique_hosts: 1
         current_memory: 0 bytes
total_memory_allocated: 0 bytes
1 row in set (0.04 sec)
```

user: The client user name.

statements: The total number of statements for the user.

statement_latency: The total wait time of timed statements for the user.

statement_avg_latency: The average wait time per timed statement for the user.

table_scans: The total number of table scans for the user.

file_ios: The total number of file I/O events for the user.

file_io_latency: The total wait time of timed file I/O events for the user.

current_connections: The current number of connections for the user.

total_connections: The total number of connections for the user.

unique_hosts: The number of distinct hosts from which connections for the user have originated.

current_memory: The current amount of allocated memory for the user.

total_memory_allocated: The total amount of allocated memory for the user.

Views: 从"使用者"角度看代价

```
mysql> select * from host_summary\G
                host: localhost
          statements: 266159692
      statement_latency: 10.35 w
  statement_avg_latency: 23.52 ms
         table_scans: 1497844
             file_ios: 22139970
    file_io_latency: 6.68 h
 current_connections: 11
   total_connections: 3296
        unique_users: 2
      current_memory: 0 bytes
total_memory_allocated: 0 bytes
1 row in set (0.03 sec)
```

Views: 从资源角度看使用情况

io_by_thread_by_latency

io_global_by_file_by_bytes

io_global_by_file_by_latency

io_global_by_wait_by_bytes

io_global_by_wait_by_latency

latest_file_io

memory_by_host_by_current_bytes

memory_by_thread_by_current_bytes

memory_by_user_by_current_bytes

memory_global_by_current_bytes

memory_global_total

Views: 从资源角度看使用情况


```
mysql> select * from io_global_by_file_by_bytes limit 3;
+----------------------+------------+------------+---------+-------------+---------------+-----------+-----------+-----------+
| file                 | count_read | total_read | avg_read | count_write | total_written | avg_write | total     | write_pct |
+----------------------+------------+------------+---------+-------------+---------------+-----------+-----------+-----------+
| @@datadir/ibdata1    |        395 | 3.59 MiB   | 9.32 KiB |       97618 | 16.89 GiB     | 181.43 KiB | 16.89 GiB |     99.98 |
| @@datadir/ib_logfile0 |          7 | 20.50 KiB  | 2.93 KiB |     1064236 | 10.74 GiB     | 10.58 KiB  | 10.74 GiB |    100.00 |
| @@datadir/ib_logfile1 |          0 | 0 bytes    | 0 bytes  |     1014778 | 10.27 GiB     | 10.61 KiB  | 10.27 GiB |    100.00 |
+----------------------+------------+------------+---------+-------------+---------------+-----------+-----------+-----------+
3 rows in set (0.00 sec)
```

file: The file path name.

count_read: The total number of read events for the file.

total_read: The total number of bytes read from the file.

avg_read: The average number of bytes per read from the file.

count_write: The total number of write events for the file.

total_written: The total number of bytes written to the file.

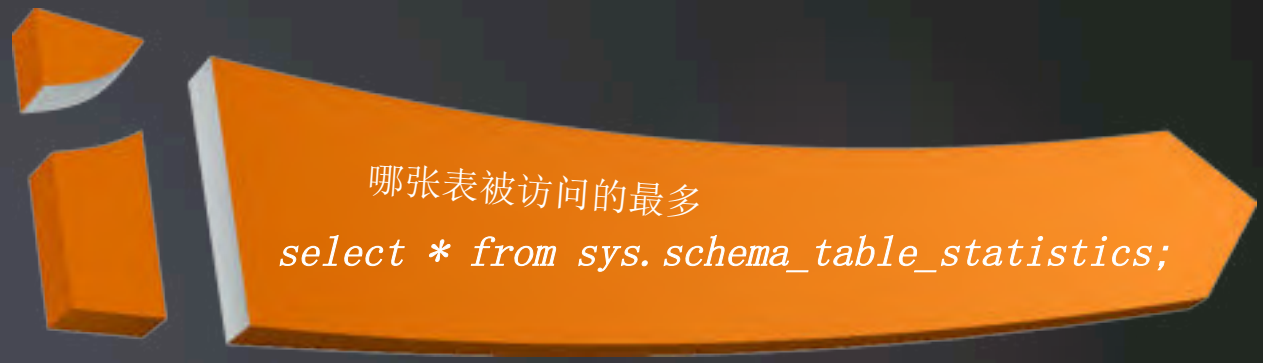
avg_write: The average number of bytes per write to the file.

total: The total number of bytes read and written for the file.

write_pct: The percentage of total bytes of I/O that were writes.

Views: schema级别的统计信息

1. 对象
2. 索引使用统计
3. 表使用统计
4. 表锁信息



schema_auto_increment_columns

schema_index_statistics

schema_object_overview

schema_redundant_indexes

schema_table_lock_waits

schema_table_statistics

schema_table_statistics_with_buffer

schema_tables_with_full_table_scans

schema_unused_indexes

Views: schema级别的统计信息

```
mysql> select table_schema, table_name, index_name, rows_selected, rows_inserted, rows_updated, rows_deleted from
schema_index_statistics;
+--------------+------------+------------+---------------+---------------+--------------+--------------+
| table_schema | table_name | index_name | rows_selected | rows_inserted | rows_updated | rows_deleted |
+--------------+------------+------------+---------------+---------------+--------------+--------------+
| test         | sbtest2    | PRIMARY    |    1440833595 |             0 |      5994464 |      1998083 |
| test         | sbtest1    | PRIMARY    |    1442705405 |             0 |      6002248 |      2000660 |
| sys          | sys_config | PRIMARY    |             3 |             0 |            0 |            0 |
| test         | person     | PRIMARY    |             0 |             0 |            0 |            0 |
| test         | sbtest1    | k_1        |             0 |             0 |            0 |            0 |
| test         | sbtest2    | k_2        |             0 |             0 |            0 |            0 |
| test         | t1         | PRIMARY    |             0 |             0 |            0 |            0 |
| test         | t1         | idx_a_b    |             0 |             0 |            0 |            0 |
| test         | t1         | idx_a_b_d  |             0 |             0 |            0 |            0 |
| test         | t1         | idx_b_c    |             0 |             0 |            0 |            0 |
| test         | t1         | idx_b_c_d  |             0 |             0 |            0 |            0 |
+--------------+------------+------------+---------------+---------------+--------------+--------------+
```

Views: schema级别的统计信息

```
mysql> select data_type, column_type, is_unsigned, max_value, auto_increment_ratio from schema_auto_increment_columns;
+-----------+---------------------+-------------+----------------------+----------------------+
| data_type | column_type         | is_unsigned | max_value            | auto_increment_ratio |
+-----------+---------------------+-------------+----------------------+----------------------+
| int       | int(10) unsigned    |           1 |           4294967295 |               0.0023 |
| int       | int(11)             |           0 |           2147483647 |               0.0000 |
| bigint    | bigint(20) unsigned |           1 | 18446744073709551615 |               0.0000 |
+-----------+---------------------+-------------+----------------------+----------------------+
3 rows in set (0.15 sec)
```

Views: statement级别的统计信息

1. 执行出错
2. 全表扫描
3. 创建临时表
4. 排序


```
statement_analysis
statements_with_errors_or_warnings
statements_with_full_table_scans
statements_with_runtimes_in_95th_percentile
statements_with_sorting
statements_with_temp_tables
```

Views: statement级别的统计信息

```
select * from statement_analysis
query: SELECT DISTINCTROW `c` FROM `s ... WEEN ? AND ? + ? ORDER BY `
                db: test
         full_scan:
        exec_count: 4001667
         err_count: 0
        warn_count: 0
     total_latency: 3.18 d
       max_latency: 1.99 s
       avg_latency: 68.58 ms
      lock_latency: 10.50 m
         rows_sent: 400166900
     rows_sent_avg: 100
     rows_examined: 1200500700
 rows_examined_avg: 300
     rows_affected: 0
 rows_affected_avg: 0
        tmp_tables: 4001669
   tmp_disk_tables: 0
       rows_sorted: 400167000
 sort_merge_passes: 0
            digest: 04153ba9d7f260e56e17ea3283456351
        first_seen: 2016-06-22 15:26:11
         last_seen: 2016-06-23 11:55:39
2 rows in set (0.00 sec)
```

Views: statement级别的统计信息


```
mysql> select * from statements_with_temp_tables limit 1\G
                    query: SELECT HOST FROM `information_ ...
slist` WHERE `command` LIKE ?
                       db: NULL
                exec_count: 193504
             total_latency: 3.67 m
       memory_tmp_tables: 193504
         disk_tmp_tables: 193504
avg_tmp_tables_per_query: 1
  tmp_tables_to_disk_pct: 100
              first_seen: 2016-06-22 12:44:53
               last_seen: 2016-06-23 16:09:24
                  digest: fdb3b36260025ebce66105473a04f0dc
1 row in set (0.00 sec)
```

first_seen: The time at which the statement was first seen.

last_seen: The time at which the statement was most recently seen.

Views: 其他

1. Buffer pool
2. 锁等待
3. 会话
4. 延迟

innodb_buffer_stats_by_schema

innodb_buffer_stats_by_table

innodb_lock_waits

wait_classes_global_by_avg_latency

wait_classes_global_by_latency

waits_by_host_by_latency

waits_by_user_by_latency

waits_global_by_latency

processlist

session

Views: 其他

```
mysql> select * from innodb_lock_waits\G
*************************** 1. row ***************************
                wait_started: 2016-06-23 09:17:19
                    wait_age: 00:00:01
               wait_age_secs: 1
                locked_table: `test`.`t2`
                locked_index: idx_a_b
                 locked_type: RECORD
              waiting_trx_id: 12138890
         waiting_trx_started: 2016-06-23 09:17:19
waiting_trx_age: 00:00:01
     waiting_trx_rows_locked: 1
   waiting_trx_rows_modified: 0
                 waiting_pid: 113911
               waiting_query: select * from t2 where a = 1 for update
             waiting_lock_id: 12138890:29:12:2
           waiting_lock_mode: X
             blocking_trx_id: 12123164
                blocking_pid: 111603
              blocking_query: NULL
            blocking_lock_id: 12123164:29:12:2
          blocking_lock_mode: X
        blocking_trx_started: 2016-06-23 09:16:01
            blocking_trx_age: 00:01:19
     blocking_trx_rows_locked: 193
   blocking_trx_rows_modified: 0
      sql_kill_blocking_query: KILL QUERY 111603
sql_kill_blocking_connection: KILL 111603
1 row in set (0.00 sec)
```

目录

Functions

- 格式化数据
- 提取对象名字
- Dump线程堆栈

```
format_time
format_bytes
format_path
format_statement
extract_table_from_file_name
extract_schema_from_file_name
ps_thread_stack
ps_is_account_enabled
```

Functions: format_time()

```
mysql> select format_time(23849723429) as time
 union select format_time(8327423749233)
 union select format_time(83274237492335);
+----------+
| time     |
+----------+
| 23.85 ms |
| 8.33 s   |
| 1.39 m   |
+----------+
```

Functions: format_bytes ()

```
mysql> select format_bytes(23423) as bytes
 union select format_bytes(23432423)
 union select format_bytes(42839479283)
 union select format_bytes(2293848203489);
+-----------+
| bytes     |
+-----------+
| 22.87 KiB |
| 22.35 MiB |
| 39.90 GiB |
| 2.09 TiB  |
+-----------+
```

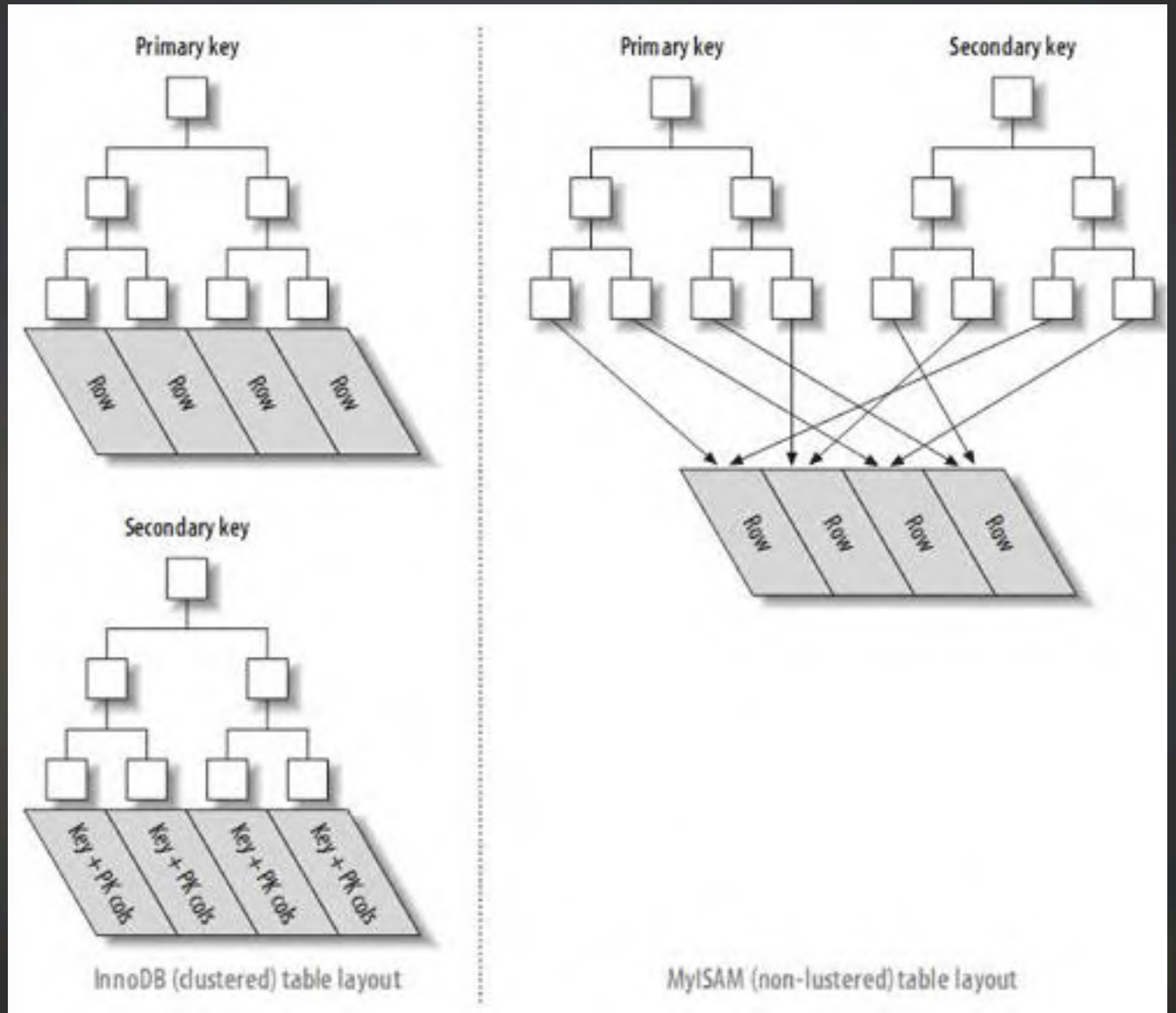
Procedures

- Performance Schema Config Helper Procedures

```
ps_setup_show_disabled() / ps_setup_show_enabled()
ps_setup_disable_thread() / ps_setup_enable_thread()
ps_setup_disable_background_threads() /
ps_setup_enable_background_threads()
……
```

目录

案例: InnoDB的索引结构



案例: 索引的好处

- 通过索引过滤，减少需要扫描的记录数量
- 索引可以帮助服务器避免排序和临时表
- 索引可以将随机IO变为顺序IO
- 通过索引覆盖，加快查询

参考资料：高性能MySQL第3版152页

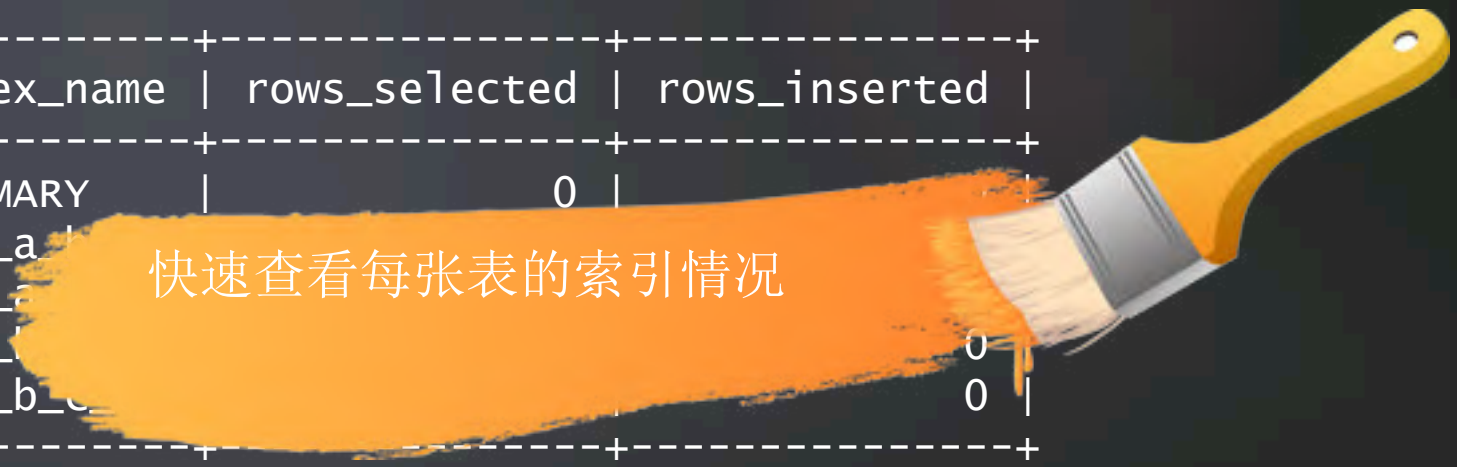
案例: 索引的坏处

- 占用磁盘空间
- 增加了记录的修改（插入、删除、修改）代价

案例: 索引统计

- 很多革命前辈的经验，一张表不宜超过6个索引，这个经验已经被证明不靠谱，这里只是说明索引太多的坏处

```
mysql> select table_schema, table_name, index_name, rows_selected, rows_inserted
from schema_index_statistics;
+--------------+------------+------------+---------------+---------------+
| table_schema | table_name | index_name | rows_selected | rows_inserted |
+--------------+------------+------------+---------------+---------------+
| test         | t1         | PRIMARY    |             0 |             0 |
| test         | t1         | idx_a      |             0 |             0 |
| test         | t1         | idx_       |             0 |             0 |
| test         | t1         | idx_       |             0 |             0 |
| test         | t1         | idx_b_c    |             0 |             0 |
+--------------+------------+------------+---------------+---------------+
11 rows in set (0.00 sec)
```



参考资料：数据库索引设计与优化中文版第6页

案例: 重复索引

```
mysql> CREATE TABLE test(
    ->     ID INT NOT NULL PRIMARY KEY,
    ->     A INT NOT NULL,
    ->     B INT NOT NULL,
    ->     UNIQUE(ID),
    ->     INDEX(ID))ENGINE=InnoDB;
```

一个经验不足的用户，可能想创建一个主键，先加上唯一限制，然后再加上索引以供查询使用。事实上，MySQL的唯一限制和主键限制，都是通过索引实现的，因此，这里的建表语句，在同一个列上创建了三个重复索引。

查看并修复重复索引

```
mysql> select table_name, redundant_index_name, sql_drop_index from
schema_redundant_indexes limit 2;
+------------+-----------------------+-----------------------------------------+
| table_name | redundant_index_name  | sql_drop_index                          |
+------------+-----------------------+-----------------------------------------+
| test       | ID_2                  | ALTER TABLE `test`.`test` DROP INDEX `ID_2` |
| test       | ID                    | ALTER TABLE `test`.`test` DROP INDEX `ID`   |
+------------+-----------------------+-----------------------------------------+
```

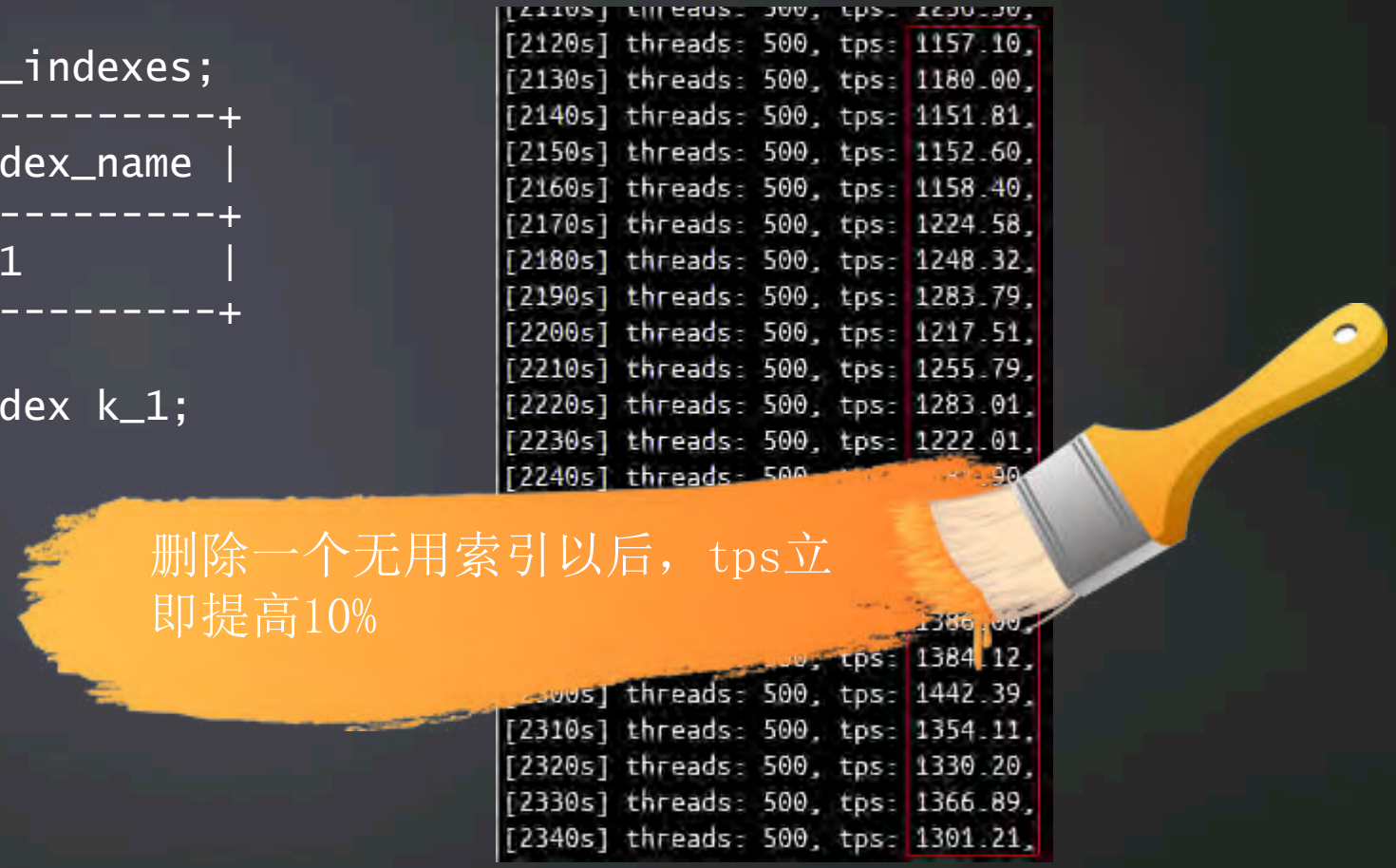
参考资料：高性能MySQL第3版178页

案例: 无用索引

```
SELECT * FROM schema_unused_indexes;
```

```
mysql> select * from schema_unused_indexes;
+---------------+-------------+------------+
| object_schema | object_name | index_name |
+---------------+-------------+------------+
| test          | sbtest1     | k_1        |
+---------------+-------------+------------+

mysql> alter table sbtest1 drop index k_1;
```



删除一个无用索引以后，tps立即提高10%

综合案例

```
mysql> show create table t1\G
*************************** 1. row ***************************
       Table: t1
Create Table: CREATE TABLE `t1` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `a` int(11) DEFAULT NULL,
  `b` int(11) DEFAULT NULL,
  `c` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=71663 DEFAULT CHARSET=utf8mb4
```

```
select a from t1 where a < 8;
```

综合案例: 全表扫描

```
mysql> select * from statements_with_full_table_scans where db='db2' limit 1\G
*************************** 1. row ***************************
                  query: SELECT `a` FROM `t1` WHERE `a` = ?
                     db: db2
             exec_count: 3
          total_latency: 59.17 ms
      no_index_used_count: 3
no_good_index_used_count: 0
        no_index_used_pct: 100
                rows_sent: 810
            rows_examined: 147456
            rows_sent_avg: 270
        rows_examined_avg: 49152
               first_seen: 2016-06-23 17:51:19
                last_seen: 2016-06-23 17:51:23
                   digest: 1eb0b504cd6bcf28384d910f0573ad2d
1 row in set (0.00 sec)

alter table t1 add index idx_a(a);
```

综合案例: 记录排序

```
SELECT `a` , `b` FROM `t1` WHERE a > 282475235 ORDER BY a, b;
```


```
mysql> select * from statements_with_sorting where db = 'db
*************************** 1. row ***************************
            query: SELECT `a` , `b` FROM `t1` WHERE `a` < ? ORDE
               db: db2
       exec_count: 4
    total_latency: 543.65 ms
sort_merge_passes: 4
  avg_sort_merges: 1
sorts_using_scans: 0
 sort_using_range: 4
     rows_sorted: 153480
  avg_rows_sorted: 38370
       first_seen: 2016-06-23 17:58:29
        last_seen: 2016-06-23 17:59:00
           digest: 13231c72167afb97841f57444562b0c3
1 row in set (0.00 sec)
```

综合案例: 记录排序

为了避免排序，增加一个（a，b）的索引

```
alter table t1 add index idx_a_b(a, b);
```

综合案例: 冗余索引

增加（a，b）的索引以后，(a)就成了一个冗余索引

```
mysql> select * from schema_redundant_indexes where table_schema = 'db2'\G
*************************** 1. row ***************************
               table_schema: db2
                 table_name: t1
         redundant_index_name: idx_a
      redundant_index_columns: a
redundant_index_non_unique: 1
          dominant_index_name: idx_a_b
       dominant_index_columns: a,b
 dominant_index_non_unique: 1
              subpart_exists: 0
              sql_drop_index: ALTER TABLE `db2`.`t1` DROP INDEX `idx_a`
```

综合案例: 总结

全表扫描

添加索引避免全表扫描

新的SQL语句需要排序

新建索引避免排序

发现冗余索引并修复

thanks