





慢查询诊断问题

丁奇

目录

-  慢查询相关参数和建议配置
-  如何判断 SQL 的主要消耗阶段
-  调优思路
-  案例分析

慢查询日志相关参数

- `slow_query_log + long_query_time`
- `log_slow_extra`
- `log_queries_not_using_indexes`
- `log_throttle_queries_not_using_indexes`
- `min_examined_row_limit`

参数含义及实践建议

- `slow_query_log` + `long_query_time`

- 日志开关

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

mysql> show variables like 'slow_query_log';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| slow_query_log | ON    |
+-----+-----+
1 row in set (0.01 sec)

mysql> show variables like 'long_query_time';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| long_query_time | 1.000000 |
+-----+-----+
1 row in set (0.01 sec)
```

参数含义及实践建议

- `slow_query_log + long_query_time`

- 实践建议

1. set global long_query_time=1;

2. 分析型业务, set long_query_time=N;

- long_query_time

Command-Line Format	--long-query-time=#
System Variable	<code>long_query_time</code>
Scope	Global, Session
Dynamic	Yes

参数含义及实践建议

- log_slow_extra
 - 日志格式
 - 实践建议

```
# Time: 2024-09-28T00:15:45.583492Z
# User@Host: root[root] @ localhost [127.0.0.1] Id: 12
# Query_time: 0.000774 Lock_time: 0.000010 Rows_sent: 1 Rows_examined: 1
SET timestamp=1727482545;
select * from t1 where c=2;
```

```
# Time: 2024-09-28T00:26:08.124533Z
# User@Host: root[root] @ localhost [127.0.0.1] Id: 10
# Query_time: 0.000790 Lock_time: 0.000011 Rows_sent: 1 Rows_examined: 3 Thread_id: 10 Errno: 0 Killed: 0 Bytes_received: 32 Bytes_sent: 94 Read_first: 1 Read_last: 0 Read_key: 1 Read_next: 0 Read_prev: 0 Read_rnd: 0 Read_rnd_next: 4 Sort_merge_passes: 0 Sort_range_count: 0 Sort_rows: 0 Sort_scan_count: 0 Created_tmp_disk_tables: 0 Created_tmp_tables: 0 Start: 2024-09-28T00:26:08.123743Z End: 2024-09-28T00:26:08.124533Z
SET timestamp=1727483168;
select * from t where c=2;
```

课堂练习：

- 5.7 和 8.0 关于慢查询日志差异

session1	session2
begin; update t set c=1 where id=1;	
	update t set c=2 where id=1; //blocked
select sleep(1); commit	
	//updated, affected rows=1

- 问：session2 的语句更新成功后，会不会记录到慢查询日志？

样例分析（MySQL 8.0）

session1	session2	
	lock table t2 write	
begin; update t2 set c=c+1 where id=1;		
		begin; update t2 set c=c*10 where id=1;

```
# Time: 2024-10-19T00:31:05.678840Z
# User@Host: root[root] @ localhost [127.0.0.1] Id: 69
# Query_time: 8.299680 Lock_time: 0.000032 Rows_sent: 0 Rows_examined: 1 Thread_id: 69 Errno: 0 Killed: 0 Bytes_received:
38 Bytes_sent: 52 Read_first: 0 Read_last: 0 Read_key: 1 Read_next: 0 Read_prev: 0 Read_rnd: 0 Read_rnd_next: 0
Sort_merge_passes: 0 Sort_range_count: 0 Sort_rows: 0 Sort_scan_count: 0 Created_tmp_disk_tables: 0 Created_tmp_tables: 0
Start: 2024-10-19T00:30:57.379160Z End: 2024-10-19T00:31:05.678840Z
SET timestamp=1729297857;
update t2 set c=c*10 where id=1;
```

```
# Time: 2024-10-19T00:31:25.020243Z
# User@Host: root[root] @ localhost [127.0.0.1] Id: 71
# Query_time: 44.127009 Lock_time: 19.343139 Rows_sent: 0 Rows_examined: 1 Thread_id: 71 Errno: 0 Killed: 0
Bytes_received: 37 Bytes_sent: 52 Read_first: 0 Read_last: 0 Read_key: 1 Read_next: 0 Read_prev: 0 Read_rnd: 0
Read_rnd_next: 0 Sort_merge_passes: 0 Sort_range_count: 0 Sort_rows: 0 Sort_scan_count: 0 Created_tmp_disk_tables: 0
Created_tmp_tables: 0 Start: 2024-10-19T00:30:40.893234Z End: 2024-10-19T00:31:25.020243Z
SET timestamp=1729297840;
update t2 set c=c+1 where id=1;
```


参数含义及实践建议

- log_queries_not_using_indexes

不走索引的 SQL 语句性能问题

课堂练习:

update t3 set e=e*10 where e=2 锁多少行?

课堂练习：默认参数	
session 1	session 2
begin;	
update t3 set e=e*10 where e=1;	
	update t3 set e=e*10 where id=2;
	//现象?

```
root@devops_db 12:12: [test]> select * from t3;
+----+-----+-----+-----+
| id | c     | d     | e     |
+----+-----+-----+-----+
| 1  | 833   | 10    | 1     |
| 2  | 2     | 2     | 2     |
| 3  | 3     | 3     | 3     |
+----+-----+-----+-----+
3 rows in set (0.00 sec)
```

```
Table: t3
Create Table: CREATE TABLE `t3` (
  `id` int NOT NULL,
  `c` int DEFAULT NULL,
  `d` int DEFAULT NULL,
  `e` int DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `c` (`c`),
  KEY `d` (`d`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
1 row in set (0.01 sec)
```

log_queries_not_using_indexes导致慢查询日志过多

```
select * from information_schema.processlist;  
select * from information_schema.innodb_trx;  
show engine innodb status\G
```

```
root@devops_db 06:59: [mysql]> show create table information_schema.processlist\G  
***** 1. row *****  
      Table: PROCESSLIST  
Create Table: CREATE TEMPORARY TABLE `PROCESSLIST` (  
  `ID` bigint unsigned NOT NULL DEFAULT '0',  
  `USER` varchar(32) NOT NULL DEFAULT '',  
  `HOST` varchar(261) NOT NULL DEFAULT '',  
  `DB` varchar(64) DEFAULT NULL,  
  `COMMAND` varchar(16) NOT NULL DEFAULT '',  
  `TIME` int NOT NULL DEFAULT '0',  
  `STATE` varchar(64) DEFAULT NULL,  
  `INFO` longtext  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3
```


log_queries_not_using_indexes导致慢查询日志过多

```
select * from information_schema.processlist;
select * from information_schema.innodb_trx;
show engine innodb status\G
```

```
root@devops_db 06:58: [mysql]> show create table information_schema.innodb_trx\G
***** 1. row *****
      Table: INNODB_TRX
Create Table: CREATE TEMPORARY TABLE `INNODB_TRX` (
  `trx_id` bigint unsigned NOT NULL DEFAULT '0',
  `trx_state` varchar(13) NOT NULL DEFAULT '',
  `trx_started` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `trx_requested_lock_id` varchar(105) DEFAULT NULL,
  `trx_wait_started` datetime DEFAULT NULL,
  `trx_weight` bigint unsigned NOT NULL DEFAULT '0',
  `trx_mysql_thread_id` bigint unsigned NOT NULL DEFAULT '0',
  `trx_query` varchar(1024) DEFAULT NULL,
  `trx_operation_state` varchar(64) DEFAULT NULL,
  `trx_tables_in_use` bigint unsigned NOT NULL DEFAULT '0',
  `trx_tables_locked` bigint unsigned NOT NULL DEFAULT '0',
  `trx_lock_structs` bigint unsigned NOT NULL DEFAULT '0',
  `trx_lock_memory_bytes` bigint unsigned NOT NULL DEFAULT '0',
  `trx_rows_locked` bigint unsigned NOT NULL DEFAULT '0',
  `trx_rows_modified` bigint unsigned NOT NULL DEFAULT '0',
  `trx_concurrency_tickets` bigint unsigned NOT NULL DEFAULT '0',
  `trx_isolation_level` varchar(16) NOT NULL DEFAULT '',
  `trx_unique_checks` int NOT NULL DEFAULT '0',
  `trx_foreign_key_checks` int NOT NULL DEFAULT '0',
  `trx_last_foreign_key_error` varchar(256) DEFAULT NULL,
  `trx_adaptive_hash_latched` int NOT NULL DEFAULT '0',
  `trx_adaptive_hash_timeout` bigint unsigned NOT NULL DEFAULT '0',
  `trx_is_read_only` int NOT NULL DEFAULT '0',
  `trx_autocommit_non_locking` int NOT NULL DEFAULT '0',
  `trx_schedule_weight` bigint unsigned DEFAULT NULL
) ENGINE=MEMORY DEFAULT CHARSET=utf8mb3
```


log_queries_not_using_indexes导致慢查询日志过多

```
select * from information_schema.processlist;
select * from information_schema.innodb_trx;
show engine innodb status\G
```

```
# Time: 2024-10-19T23:01:29.314567Z
# User@Host: root[root] @ localhost [127.0.0.1] Id: 78
# Query_time: 0.002421 Lock_time: 0.000002 Rows_sent: 0 Rows_examined: 0 Thread_id: 78 Errno: 0 Killed: 0 Bytes_received: 50 Bytes_sent: 2498 Read_first: 0 Read_last: 0 Read_key: 0 Read_next: 0 Read_prev: 0 Read_rnd: 0 Read_rnd_next: 1 Sort_merge_passes: 0 Sort_range_count: 0 Sort_rows: 0 Sort_scan_count: 0 Created_tmp_disk_tables: 0 Created_tmp_tables: 1 Start: 2024-10-19T23:01:29.312146Z End: 2024-10-19T23:01:29.314567Z
use test;
SET timestamp=1729378889;
select * from information_schema.innodb_trx;

# Time: 2024-10-19T23:01:37.920034Z
# User@Host: root[root] @ localhost [127.0.0.1] Id: 78
# Query_time: 0.006295 Lock_time: 0.000002 Rows_sent: 2 Rows_examined: 2 Thread_id: 78 Errno: 0 Killed: 0 Bytes_received: 51 Bytes_sent: 776 Read_first: 1 Read_last: 0 Read_key: 1 Read_next: 0 Read_prev: 0 Read_rnd: 0 Read_rnd_next: 3 Sort_merge_passes: 0 Sort_range_count: 0 Sort_rows: 0 Sort_scan_count: 0 Created_tmp_disk_tables: 1 Created_tmp_tables: 1 Start: 2024-10-19T23:01:37.913739Z End: 2024-10-19T23:01:37.920034Z
SET timestamp=1729378897;
select * from information_schema.processlist;
```

参数含义及实践建议

- `log_queries_not_using_indexes+log_throttle_queries_not_using_indexes`

`log_throttle_queries_not_using_indexes`: 限制每分钟无主键语句的记录条数上限

参数含义及实践建议

● log_queries_not_using_indexes+min_examined_row_limit

min_examined_row_limit: 扫描行数少于这个值的语句, 不记入慢查询日志

```
Table: t3
Create Table: CREATE TABLE `t3` (
  `id` int NOT NULL,
  `c` int DEFAULT NULL,
  `d` int DEFAULT NULL,
  `e` int DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `c` (`c`),
  KEY `d` (`d`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
1 row in set (0.01 sec)
```

```
root@devops_db 12:12: [test]> select * from t3;
+----+-----+-----+-----+
| id | c     | d     | e     |
+----+-----+-----+-----+
| 1  | 833   | 10    | 1     |
| 2  | 2     | 2     | 2     |
| 3  | 3     | 3     | 3     |
+----+-----+-----+-----+
3 rows in set (0.00 sec)
```

课堂练习: select count(*) from t3; 的 rows_examined 是多少?

参数含义及实践建议

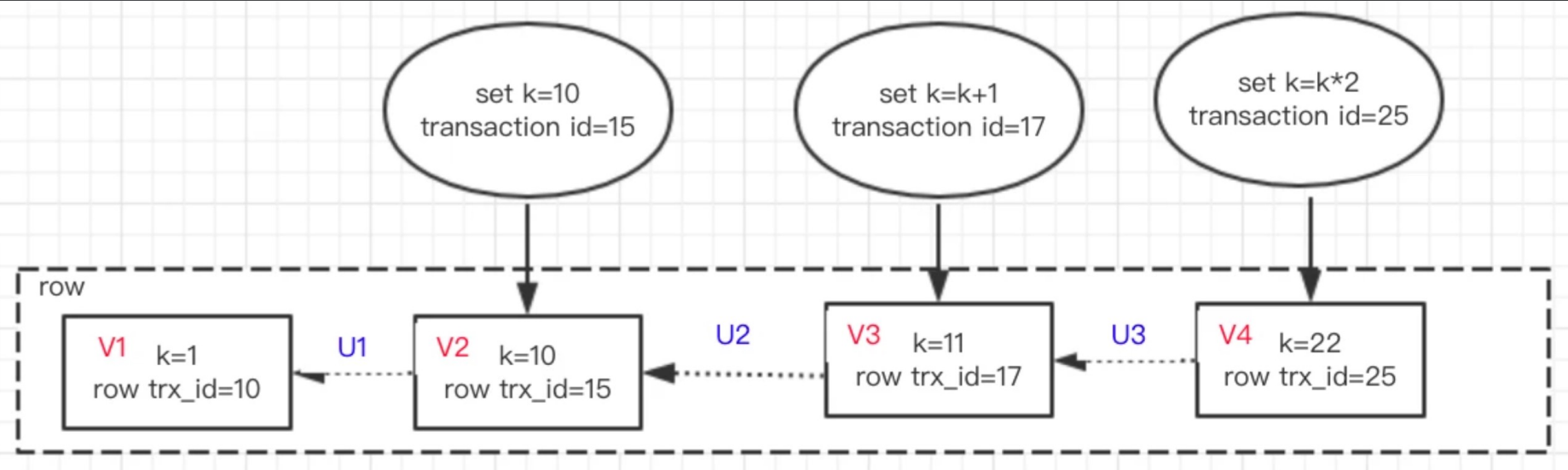
● log_queries_not_using_indexes+min_examined_row_limit

```
delimiter ;;
create procedure udata3()
begin
declare i int; set i=1;
while(i<=50000) do
  update t set c=c+1 where id=3;
  set i=i+1;
end while;
end;;
delimiter ;//准备一个存储过程，对id=3这一行做5万次更新
```

session1 (隔离级别RR)	session2
begin; select * from t where id=1;	
	call udata3()
select * from t where id=3; //查询Q	

1. 查询Q的rows_examined是多少?

2. 查询 Q 的查询时间是更接近哪个数值?
A: 0.01s B:1.01s



参数含义及实践建议

- log_queries_not_using_indexes

- 内核改进方案的思路

1. 用户名过滤

set global log_queries_not_using_indexes_white_user = 'xxx' (模拟)

2. 改为 global,session 变量

set global log_queries_not_using_indexes=on

监控应用端:

set log_queries_not_using_indexes=off (模拟)

慢查询日志影响性能吗？

A. 影响

B. 不影响

如何分析？

慢查询日志影响性能吗？

课堂练习：执行错误的语句记不记入慢查询日志？

A. 记录

B. 不记录

如何验证？

慢查询日志影响性能吗？

课堂练习：慢查询日志是在语句执行的哪个阶段写入的？

- A. 语句执行开始阶段
- B. 语句执行结束，发查询结果给客户端前
- C. 语句执行结束，发查询结果给客户端后

如何验证？

慢查询日志影响性能吗？

```
gdb -p <pid of mysqld>
```

```
(gdb) b my_error
```

```
(gdb) c
```

```
root@devops_db 15:07: [test]> select a;
```

```
Thread 38 "connection" hit Breakpoint 1, my_error (nr=1054, MyFlags=0) at /data1/dingqi/mysql-8.0.36/mysys/my_error.cc:215
215     void my_error(int nr, myf MyFlags, ...) {
(gdb) █
```

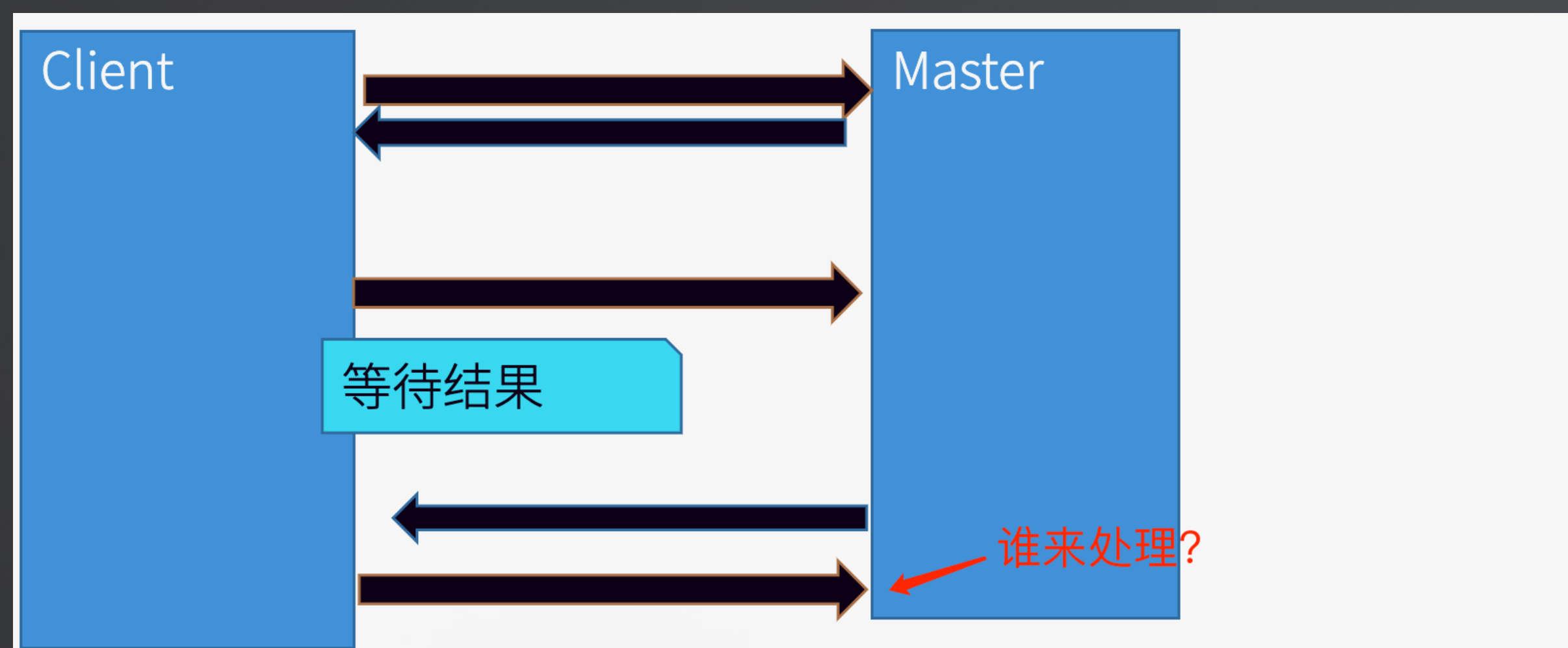
看此时 slow log 还没有输出 select a 这个语句

结论：slowlog 是在语句结果返回给客户端后再输出的

慢查询日志影响性能吗？

结论：slowlog 是在语句结果返回给客户端后再输出的

疑问：那为什么开 slow log 会影响性能？



怎么减少突发慢查询对系统的影响

- 提前发现坏查询

- 业务回归测试时提前发现

测试环境: `set global long_query_time = 0;`

`set global log_slow_extra = on;`

`rows_sent / rows_examined / sort_rows / created_tmp_tables ...`

- 审计日志采集

- 分析和预警

怎么判断慢查询语句是否有优化空间？

- 了解语句的执行流程
- 对比执行过程的消耗，跟输出结果

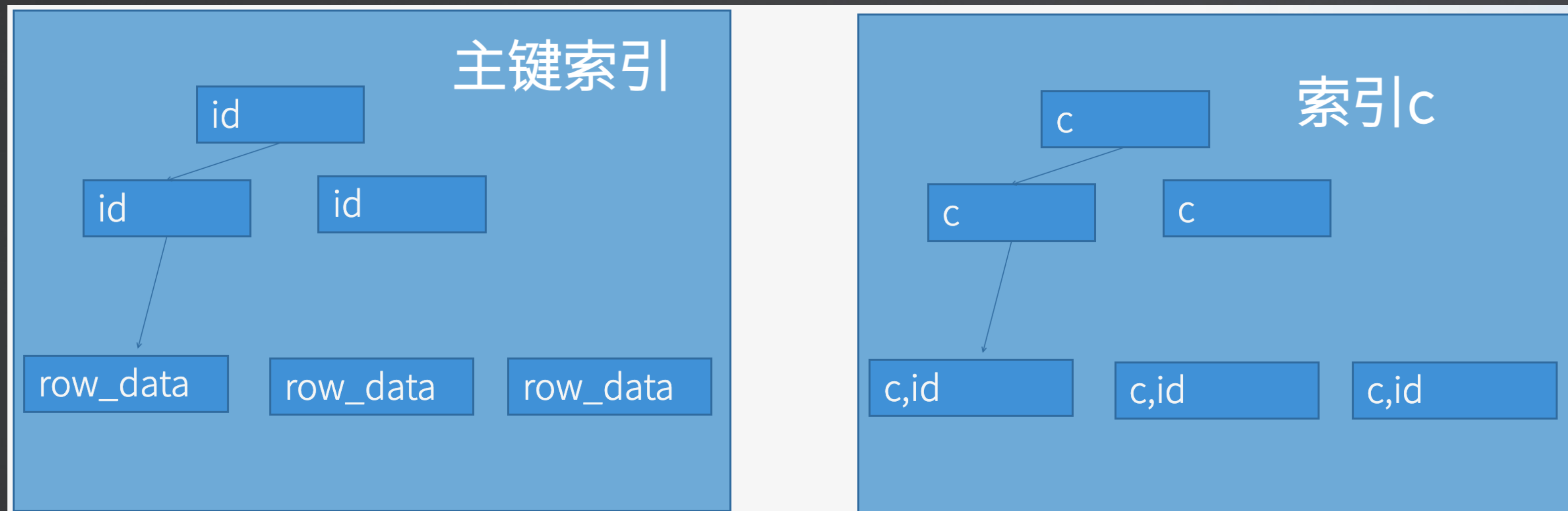
```
create table a(id int primary key, c int , d int,e text,  
index(c))engine=innodb;  
for i in [1,100]:  
  insert into a select id, id, id, repeat('a', 16000) from t;
```

```
select * from a;
```

```
select * from a where c>10 and c<20 and d>=15;
```


怎么判断慢查询语句是否有优化空间？

```
select * from a where c>10 and c<20 and d>=15;
```



```
select * from a where id in (select id from a where c>10 and c<20 and d>=15)
```

课堂练习

RR隔离级别下，表t的建表结构和初始化数据如下：

```
1 create table t(id int primary key,c int) engine=innodb;  
2 insert into t values(1,1),(11,11),(21,21);
```

在会话1 执行如下语句：

```
1 begin;  
2 select * from t lock in share mode;
```

那么，会话2的以下哪些语句会被进入“等待行锁”的状态？

- A: insert into t values(15,15);
- B: update t set c=c+1 where id=15;
- C: delete from t where id=15;
- D: alter table t add d int;

课堂练习

表t1使用InnoDB引擎， 以下哪个场景会导致语句Q1: `select * from t1 limit 1` 被堵住？

A: 另一个线程在Q1执行之前，执行了 `alter table t1 add index(f1)`， 当前处于“拷贝数据到临时表”阶段

B: 另一个线程在Q1执行之前，执行了 `truncate table t1`， 当前处于waiting for metadata lock 阶段

C: 另一个线程在Q1执行之前，执行了 `delete from t1`， 且未执行完成

D: 另一个线程在Q1执行之前，执行了 `lock table t1 write`， 并执行完成

Q & A

THANKS