第十四讲：进入优化器和优化器追踪日志实现

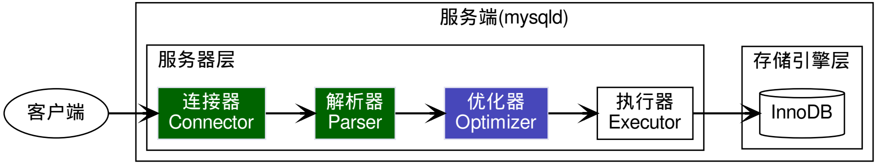
知春路遇上八里桥

<2024-06-23 Sun>

1

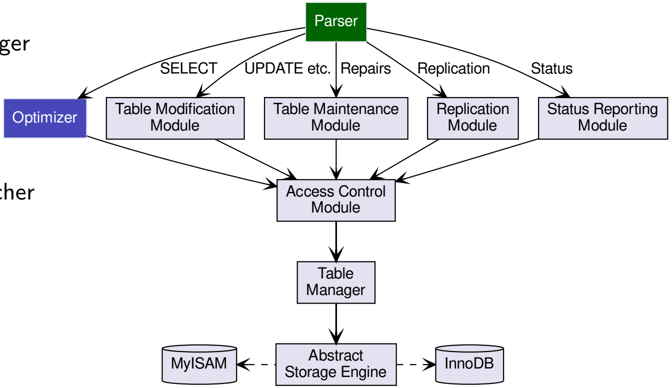
前情提要

执行流程


服务端(mysqld)
服务器层
存储引擎层
客户端
连接器 Connector
解析器 Parser
优化器 Optimizer
执行器 Executor
InnoDB

本节内容

- 连接器
  - ☑ 连接管理器 Connection Manager
  - ☑ 线程管理器 Thread Manager
  - ☑ 用户模块 User Module
- 解析器
  - ☑ 网络模块 Net Module
  - ☑ 派发模块 Commander Dispatcher
  - ☑ 词法分析 Lexical Analysis
  - ☑ 语法分析 Syntax Analysis
- 优化器
  - ☑ 准备模块 Prepare Module
  - ☐ 追踪日志 Optimizer Trace

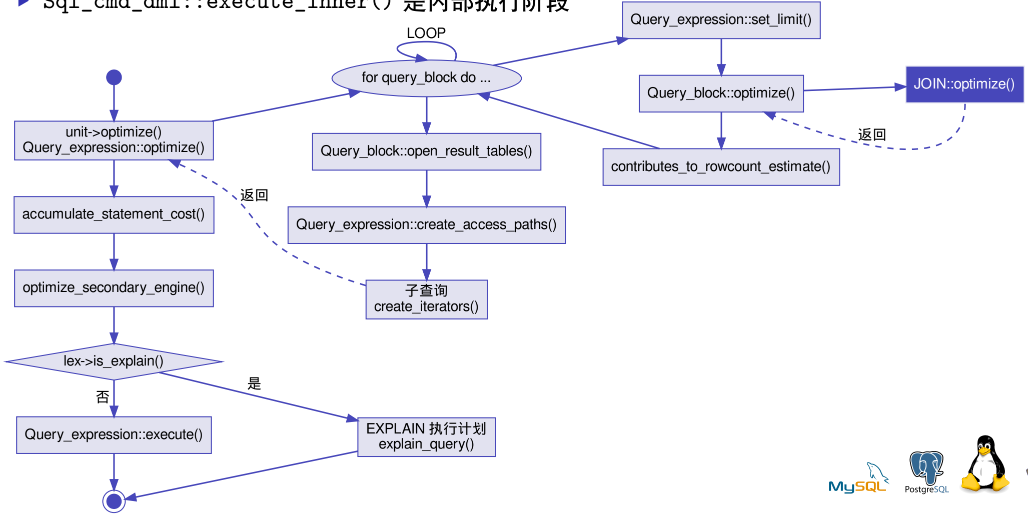


2

进入优化阶段

从 Sql_cmd_dml::execute_inner() 到 JOIN::optimize()

- Sql_cmd_dml::execute() 是 SELECT 执行的入口函数
  - Sql_cmd_dml::prepare() 是准备阶段，前一讲已经介绍
  - Sql_cmd_dml::execute_inner() 是内部执行阶段



LOOP
for query_block do ...
Query_expression::set_limit()
Query_block::optimize()
JOIN::optimize()
返回
contributes_to_rowcount_estimate()
unit->optimize()
Query_expression::optimize()
Query_block::open_result_tables()
返回
accumulate_statement_cost()
Query_expression::create_access_paths()
optimize_secondary_engine()
子查询
create_iterators()
lex->is_explain()
是
否
Query_expression::execute()
EXPLAIN 执行计划
explain_query()

优化器核心处理函数

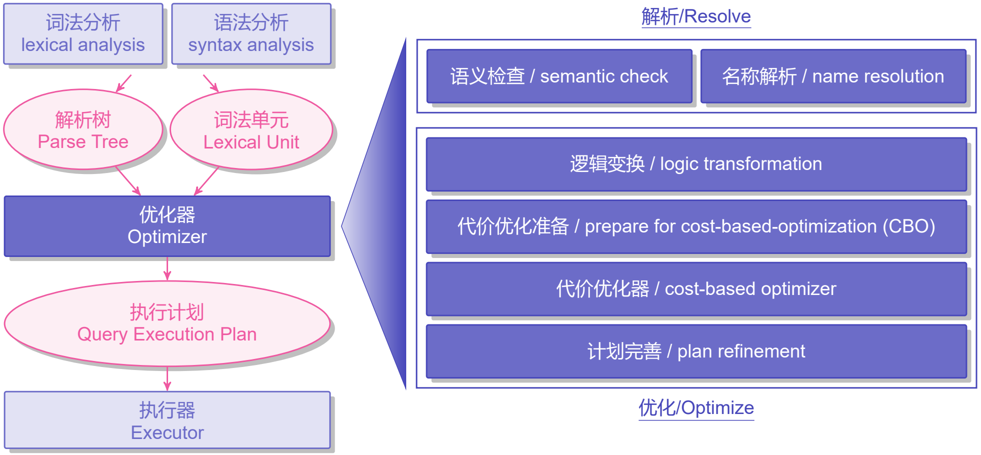
- JOIN::optimize() 是优化阶段的入口函数
  - 它主要对 Query Block 进行优化
- JOIN::optimize() 函数将 Query_block 优化成 QEP
  - ☞ sql/sql_optimizer.cc

```
bool JOIN::optimize(bool finalize_access_paths) {
  DBUG_TRACE;

  uint no_jbuf_after = UINT_MAX;
  Query_block *const set_operand_block =
      query_expression()->non_simple_result_query_block();

  assert(query_block->leaf_table_count == 0 ||
         thd->lex->is_query_tables_locked() ||
         query_block == set_operand_block);
  assert(tables == 0 && primary_tables == 0 && tables_list == (Table_ref *)1);
⋮
  set_plan_state(ZERO_RESULT);
  return false;
}
```

JOIN::optimize() 注释中的功能点

```
-# Logical transformations:
  - Outer to inner joins transformation.
  - Equality/constant propagation.
  - Partition pruning.
  - COUNT(*), MIN(), MAX() constant substitution in case of implicit grouping.
  - ORDER BY optimization.
-# Perform cost-based optimization of table order and access path selection.
   See JOIN::make_join_plan()
-# Post-join order optimization:
   - Create optimal table conditions from the where clause and the join conditions.
   - Inject outer-join guarding conditions.
   - Adjust data access methods after determining table condition (several times.)
   - Optimize ORDER BY/DISTINCT.
-# Code generation
   - Set data access functions.
   - Try to optimize away sorting/distinct.
   - Setup temporary table usage for grouping and/or sorting.
```

优化器功能点

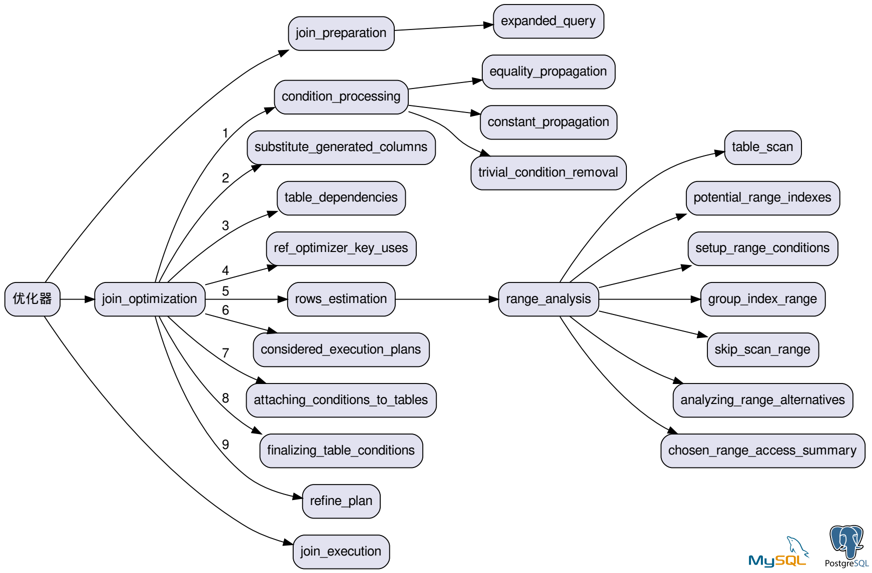


词法分析 lexical analysis
语法分析 syntax analysis
解析树 Parse Tree
词法单元 Lexical Unit
优化器 Optimizer
执行计划 Query Execution Plan
执行器 Executor
解析/Resolve
语义检查 / semantic check
名称解析 / name resolution
逻辑变换 / logic transformation
代价优化准备 / prepare for cost-based-optimization (CBO)
代价优化器 / cost-based optimizer
计划完善 / plan refinement
优化/Optimize

3

优化器追踪日志

Opt_trace 日志结构



源代码和 Opt_trace 日志 JSON 输出

- 通过代码搜索 "considered_execution_plans" 字符串

```
mysql-server $ grep -nR '"considered_execution' sql
sql/sql_planner.cc:2004:          "considered_execution_plans",
```

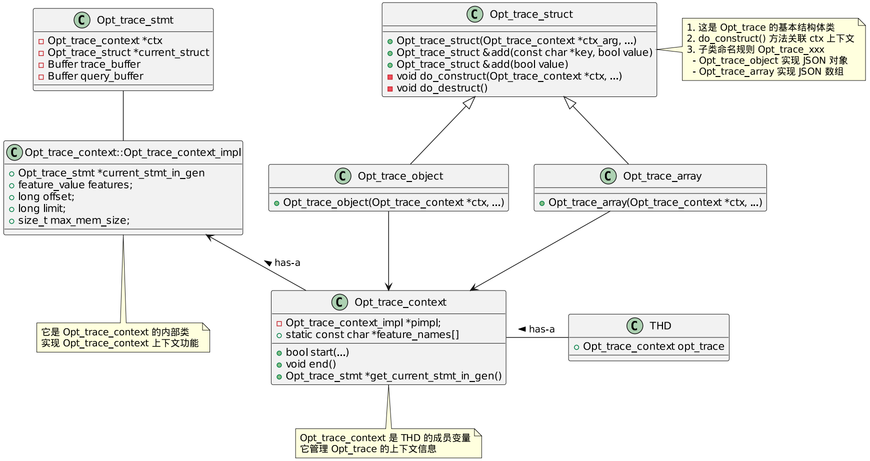
- 原始代码 ☞ sql/sql_planner.cc

```
2002  Opt_trace_object wrapper(&join->thd->opt_trace);
2003  Opt_trace_array trace_plan(&join->thd->opt_trace,
2004                             "considered_execution_plans",
2005                             Opt_trace_context::GREEDY_SEARCH);
2006
2007  if (thd->optimizer_switch_flag(OPTIMIZER_SWITCH_COND_FANOUT_FILTER) &&
2008      join->where_cond) {
2009    for (uint idx = join->const_tables; idx < join->tables; ++idx)
2010      bitmap_clear_all(&join->best_ref[idx]->table()->cond_set);
2011
2012    /*
2013      Set column bits for all columns involved in predicates in
2014      cond_set. Used to avoid calculating condition filtering in
2015      best_access_path() et al. when no filtering effect is possible.
2016    */
2017    join->where_cond->walk(&Item::add_field_to_cond_set_processor,
2018                           enum_walk::POSTFIX, nullptr);
2019  }
```

- 对应的追溯日志 ✂ OPT/Trace

```
{
  "considered_execution_plans": [
    {
      "plan_prefix": [],
      "table": "`employees`",
      "best_access_path": {
        "considered_access_paths": [
          {
            "rows_to_scan": 9,
            "access_type": "range",
            "range_details": {
              "used_index": "PRIMARY"
            },
            "resulting_rows": 9,
            "cost": 2.81039,
            "chosen": true
          }
        ]
      },
      "condition_filtering_pct": 100,
      "rows_for_plan": 9,
      "cost_for_plan": 2.81039,
      "chosen": true
    }
  ]
}
```

Opt_trace 追踪优化器



Opt_trace_stmt
Opt_trace_context *ctx
Opt_trace_struct *current_struct
Buffer trace_buffer
Buffer query_buffer
Opt_trace_struct
Opt_trace_struct(Opt_trace_context *ctx_arg, ...)
Opt_trace_struct &add(const char *key, bool value)
Opt_trace_struct &add(bool value)
void do_construct(Opt_trace_context *ctx, ...)
void do_destruct()
1. 这是 Opt_trace 的基本结构体类
2. do_construct() 方法关联 ctx 上下文
3. 子类命名规则 Opt_trace_xxx
- Opt_trace_object 实现 JSON 对象
- Opt_trace_array 实现 JSON 数组
Opt_trace_context::Opt_trace_context_impl
Opt_trace_stmt *current_stmt_in_gen
feature_value features;
long offset;
long limit;
size_t max_mem_size;
Opt_trace_object
Opt_trace_object(Opt_trace_context *ctx, ...)
Opt_trace_array
Opt_trace_array(Opt_trace_context *ctx, ...)
has-a
Opt_trace_context
Opt_trace_context_impl *pimpl;
static const char *feature_names[]
bool start(...)
void end()
Opt_trace_stmt *get_current_stmt_in_gen()
has-a
THD
Opt_trace_context opt_trace
它是 Opt_trace_context 的内部类
实现 Opt_trace_context 上下文功能
Opt_trace_context 是 THD 的成员变量
它管理 Opt_trace 的上下文信息

```
(gdb) set print elements 400
(gdb) p thd->opt_trace.get_current_stmt_in_gen()->trace_buffer.c_ptr_safe()
$83 = 0x7fff3019dde0 "{\n  \"steps\": [\n    {\n      \"join_preparation\": {\n
  \"select#\": 1,\n        \"steps\": [\n          {\n", ' ' <repeats 12 times>,
  "\"expanded_query\": \"/* select#1 */ select `employees`.`emp_no` AS `emp_no`,`em...
```

4

MySQL 执行追溯日志

原始日志分析 - 壹

- T@8 其中 8 是连接 ID (Connection id)
- 每个函数的调用开始和结束通过 < 和 > 包裹
- 函数调用栈通过 | 缩进显示
- 中间夹杂着 DBUG_PRINT("info", ...) 的输出

```
T@8: | | | >lex_start
T@8: | | | >alloc_query
T@8: | | | | thd_query: thd->thread_id():8 thd:0x7fff30001050 query:SELECT * FROM employees WHERE emp_no > 1234 LIMIT 1
T@8: | | | >parse_sql
T@8: | | | >mysql_execute_command
T@8: | | | | >Table_ref*, enum_sql_command, List<set_var_base>*, const char*, size_t, sp_printable*, const CHARSET_INFO*
T@8: | | | | | >Opt_trace_context::start
T@8: | | | | | | opt: new stmt 0x7fff3000fe20 support_I_S 0
T@8: | | | | | | >Opt_trace_context::purge_stmts
T@8: | | | | | | <Opt_trace_context::purge_stmts
T@8: | | | | | | opt: rc 1048576 max_mem_size 1048576
T@8: | | | | | <Opt_trace_context::start
T@8: | | | | | >{anonymous}::opt_trace_disable_if_no_tables_access
T@8: | | | | | <{anonymous}::opt_trace_disable_if_no_tables_access
T@8: | | | | >open_temporary_tables
T@8: | | | | | >open_temporary_table
T@8: | | | | | | enter: table: 'employees'.'employees'
T@8: | | | | | <open_temporary_table
T@8: | | | | <open_temporary_tables
```

原始日志分析 - 贰

- 进入优化器之前的准备阶段 `Query_block::prepare()`

```
T@8: | | | | >bool Sql_cmd_dml::execute
T@8: | | | | | >bool Sql_cmd_dml::prepare
T@8: | | | | | | >check_table_access
T@8: | | | | | | | info: table: employees derived: 0  view: 0
T@8: | | | | | | | >check_access
T@8: | | | | | | | | enter: db: employees  want_access: 1  master_access: 2147483647
T@8: | | | | | | | | THD::enter_stage: 'checking permissions' /opt/src/mysql-server/sql/auth/sql_authorization.cc:2146
T@8: | | | | | | >open_tables_for_query
T@8: | | | | | | | >open_tables
T@8: | | | | | | | | THD::enter_stage: 'Opening tables' /opt/src/mysql-server/sql/sql_base.cc:5797
T@8: | | | | | | | <open_tables
T@8: | | | | | | >Query_block::prepare
T@8: | | | | | | | opt: (null): starting struct
T@8: | | | | | | | opt: join_preparation: starting struct
T@8: | | | | | | | opt: select#: 1
T@8: | | | | | | | opt: steps: starting struct
T@8: | | | | | | | >Query_block::setup_tables
T@8: | | | | | | | <Query_block::setup_tables
T@8: | | | | | | | >Query_block::setup_wild
T@8: | | | | | | <Query_block::prepare
T@8: | | | | | <bool Sql_cmd_dml::prepare
T@8: | | | | | THD::enter_stage: 'init' /opt/src/mysql-server/sql/sql_select.cc:772
```

原始日志分析 - 叁

- optimize_cond() 函数执行的细节

```
T@8: | | | | | >Query_expression::optimize
T@8: | | | | | | >Query_block::optimize
T@8: | | | | | | | >JOIN::optimize
T@8: | | | | | | | | THD::enter_stage: 'optimizing' /opt/src/mysql-server/sql/sql_optimizer.cc:354
T@8: | | | | | | | | opt: (null): starting struct
T@8: | | | | | | | | opt: join_optimization: starting struct
T@8: | | | | | | | | opt: select#: 1
T@8: | | | | | | | | opt: steps: starting struct
T@8: | | | | | | | | >optimize_cond
T@8: | | | | | | | | | opt: (null): starting struct
T@8: | | | | | | | | | opt: condition_processing: starting struct
T@8: | | | | | | | | | opt: condition: "WHERE"
T@8: | | | | | | | | | opt: original_condition: "(`employees`.`emp_no` > 1234)"
T@8: | | | | | | | | | opt: steps: starting struct
T@8: | | | | | | | | | opt: (null): starting struct
T@8: | | | | | | | | | opt: transformation: "equality_propagation"
T@8: | | | | | | | | | opt: subselect_evaluation: starting struct
T@8: | | | | | | | | | opt: subselect_evaluation: ending struct
T@8: | | | | | | | | | opt: resulting_condition: "(`employees`.`emp_no` > 1234)"
T@8: | | | | | | | | | opt: (null): ending struct
T@8: | | | | | | | | | opt: (null): starting struct
T@8: | | | | | | | | | opt: transformation: "constant_propagation"
T@8: | | | | | | | | | opt: subselect_evaluation: starting struct
T@8: | | | | | | | | | opt: subselect_evaluation: ending struct
T@8: | | | | | | | | | opt: resulting_condition: "(`employees`.`emp_no` > 1234)"
```

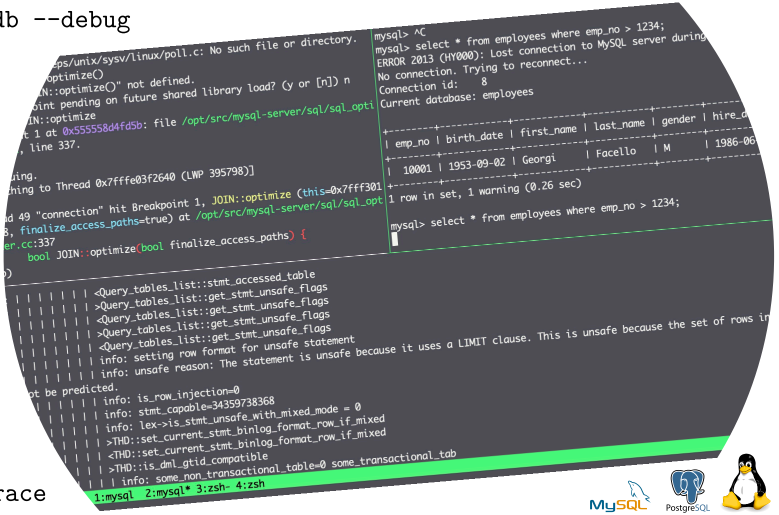
原始日志获取

- 启动时添加 --debug 选项

```
gdb --args mysqld --gdb --debug
```

gdb 调试

trace 日志



```
eps/unix/sysv/linux/poll.c: No such file or directory.
optimize()
IN::optimize()" not defined.
oint pending on future shared library load? (y or [n]) n
IN::optimize
t 1 at 0x555558d4fd5b: file /opt/src/mysql-server/sql/sql_opti
, line 337.

uing.
hing to Thread 0x7fffe03f2640 (LWP 395798)]

d 49 "connection" hit Breakpoint 1, JOIN::optimize (this=0x7fff301
8, finalize_access_paths=true) at /opt/src/mysql-server/sql/sql_opt
er.cc:337
    bool JOIN::optimize(bool finalize_access_paths) {
)
```

```
mysql> ^C
mysql> select * from employees where emp_no > 1234;
ERROR 2013 (HY000): Lost connection to MySQL server during
No connection. Trying to reconnect...
Connection id:    8
Current database: employees

+--------+------------+------------+-----------+--------+-----------
| emp_no | birth_date | first_name | last_name | gender | hire_d
+--------+------------+------------+-----------+--------+-----------
|  10001 | 1953-09-02 | Georgi     | Facello   | M      | 1986-06
+--------+------------+------------+-----------+--------+-----------
1 row in set, 1 warning (0.26 sec)

mysql> select * from employees where emp_no > 1234;
```

```
| | | | | | | <Query_tables_list::stmt_accessed_table
| | | | | | | >Query_tables_list::get_stmt_unsafe_flags
| | | | | | | <Query_tables_list::get_stmt_unsafe_flags
| | | | | | | >Query_tables_list::get_stmt_unsafe_flags
| | | | | | | <Query_tables_list::get_stmt_unsafe_flags
| | | | | | | info: setting row format for unsafe statement
| | | | | | | info: unsafe reason: The statement is unsafe because it uses a LIMIT clause. This is unsafe because the set of rows in
ot be predicted.
| | | | | | | info: is_row_injection=0
| | | | | | | info: stmt_capable=34359738368
| | | | | | | info: lex->is_stmt_unsafe_with_mixed_mode = 0
| | | | | | | >THD::set_current_stmt_binlog_format_row_if_mixed
| | | | | | | <THD::set_current_stmt_binlog_format_row_if_mixed
| | | | | | | >THD::is_dml_gtid_compatible
| | | | | | | info: some_non_transactional_table=0 some_transactional_tab
1:mysql  2:mysql* 3:zsh- 4:zsh
```

- 查看 trace 日志输出

```
tail -f /tmp/mysqld.trace
```

结束