第十一讲：遍历解析树上下文构建词法单元

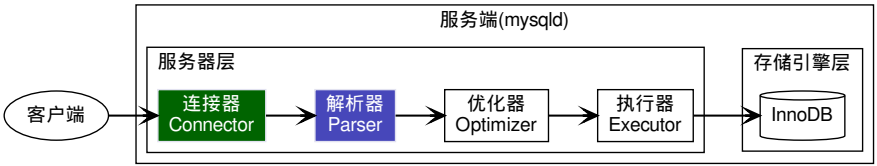
知春路遇上八里桥

<2024-06-12 Wed>

1

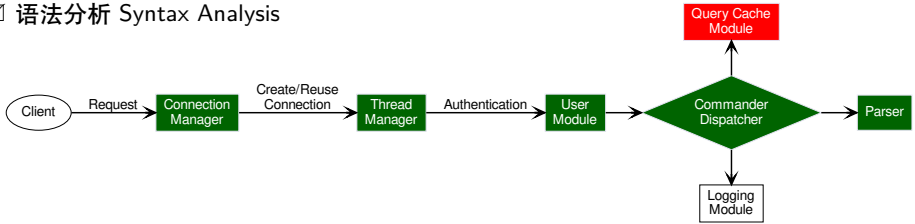
前情提要

执行流程


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



词法分析和语法分析

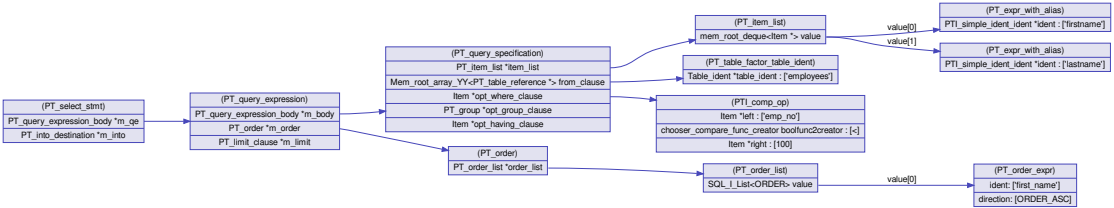
- 通过网络读取原始 Query 字符串

```
select first_name, last_name from employees where emp_no < 100 order by first_name
```

- 词法分析后形成 token 流

784/select | 484/first_name | 44/, | 484/last_name | 452/from | 484/employees | 890/where | 484/emp_no | 549/< | 628/100 | 643/order | 303/by | 484/first_name

- BISON 语法分析得到解析树 PT



2

词法单元

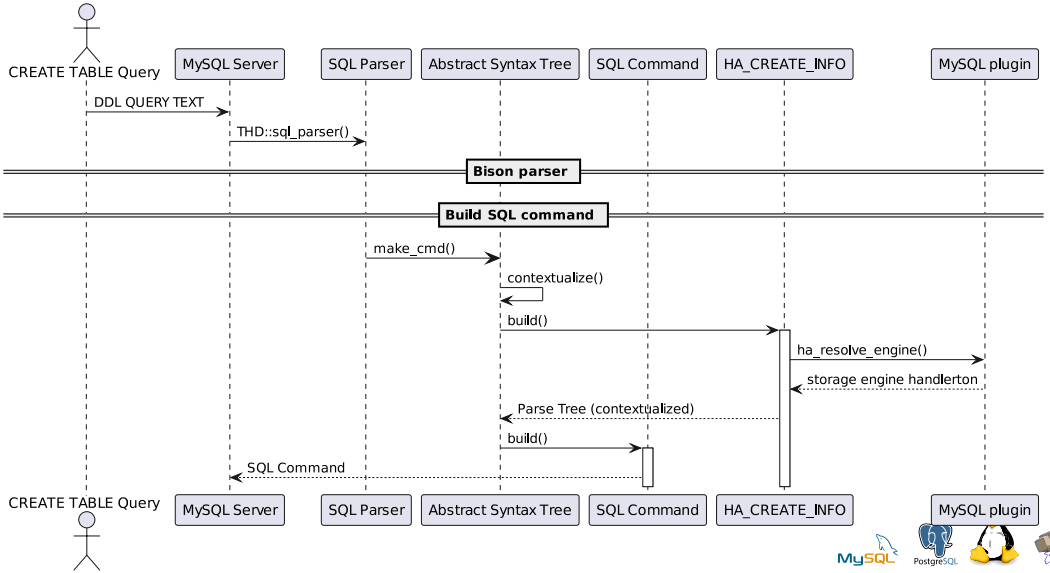
词法单元

- 聊一下 MySQL 词法单元代码发展的历史
  - 词法单元早期由 SELECT_LEX_UNIT 和 SELECT_LEX 类交替构成
  - 这部分代码在 8.0.x 版本中进行了重构，规范了类名，具体做法如下：
    1. SELECT_LEX ⇒ Query_block
    2. SELECT_LEX_UNIT ⇒ Query_expression
  - 这部分实际记录在 thd->lex->unit 变量中，故称为词法单元 (Lexical Unit) [1]
- 词法单元是先创建，然后填充，填充后完成后才生效
  - 填充发生在解析树构建完成后，即 MYSQLparse(this, &root) 调用完成后
  - 在调用 lex->make_sql_cmd() 时，遍历解析树进行增量填充
  - 这个填充过程重复调用 Parse_tree_node::contextualize() 方法，故也称 上下文化
- Query_block 是代表查询块，Query_expression 是包含多个查询块的查询表达式
- 查询块和查询表达式有以下特点 [2]
  - 查询块和查询表达式交替构成
  - 同级别只包含同类的实例，它们之间通过 next / prev 指针关联
  - 不同级别包含不同类的实例，它们通过 master / slave 进行关联

[1] 也有人把这个词法单元称为 MySQL 的抽象语法树 (AST)

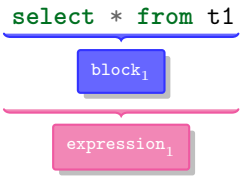
[2] 参考第十讲最后的 Query_block 和 Query_expression 的复杂关系图

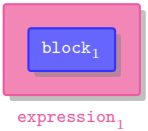
建表语句上下文化时序图



词法单元示例（壹）

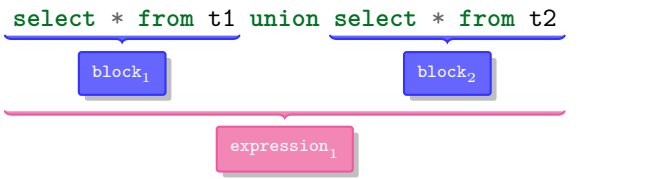
- 如果仅有一个 `select` 只会产生一个 expression ， 并通过一个 block 包裹

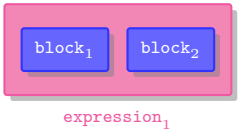




词法单元示例（贰）

- union 关键字将两个 block 合并成一个 expression

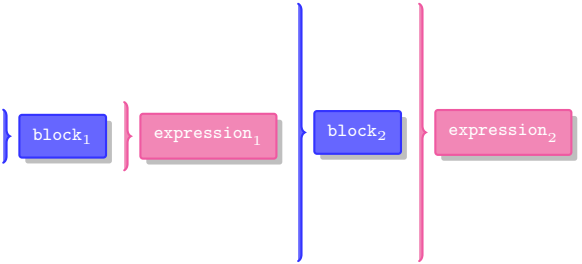




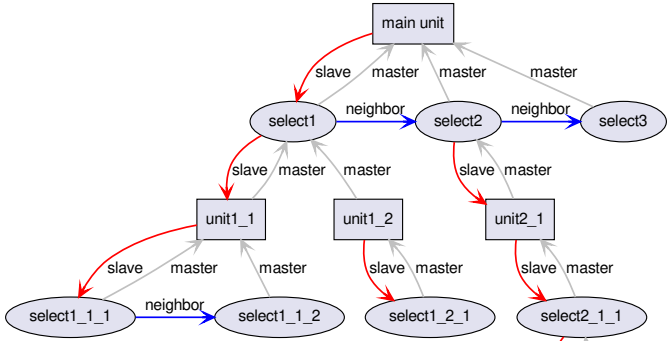
词法单元示例（叁）

- 包含子查询的会被解析成 expression 嵌套

```
1  select *
2    from
3      (
4        select * from t1
5         order by a limit 10
6      ) t2
7   order by b
8   limit 5
```



最复杂的词法单元结构




```
select *
  from table1
  where table1.field IN (select * from table1_1_1 union
                         select * from table1_1_2)
  union
select *
  from table2
  where table2.field=(select (select f1 from table2_1_1_1_1
                              where table2_1_1_1_1.f2=table2_1_1.f3)
                      from table2_1_1
                      where table2_1_1.f1=table2.f2)
  union
select * from table3;
```

3

代码分析

基础工具类

1. mem_root_deque 为了 MEM_ROOT 部分实现了双向队列 (std::deque),
   - ⋆ include/mem_root_deque.h

```
109 template <class Element_type>
110 class mem_root_deque {
111  public:
112   /// Used to conform to STL algorithm demands.
```

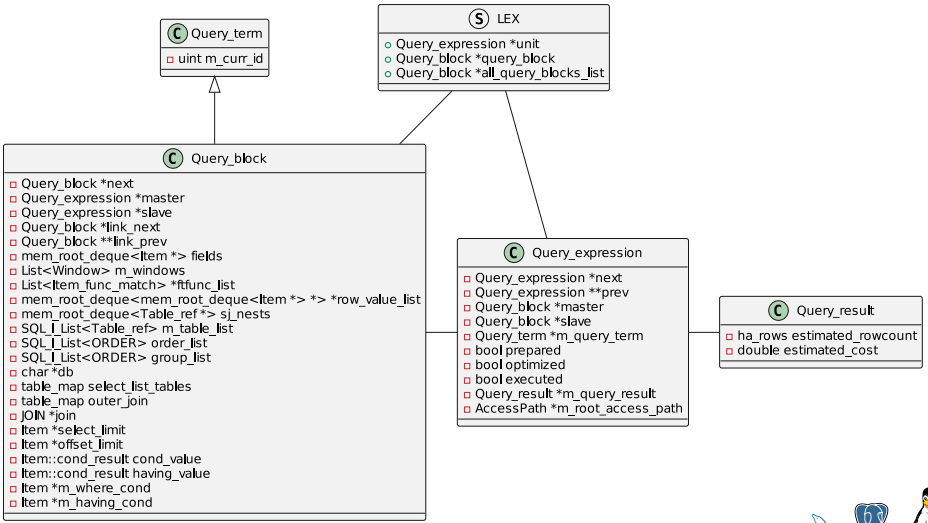
   - 支持通过迭代器进行访问

```
438 iterator begin() { return iterator{this, m_begin_idx}; }
439 iterator end() { return iterator{this, m_end_idx}; }
```

2. SQL_I_List 单项链表的模版类
   - ⋆ sql/sql_list.h

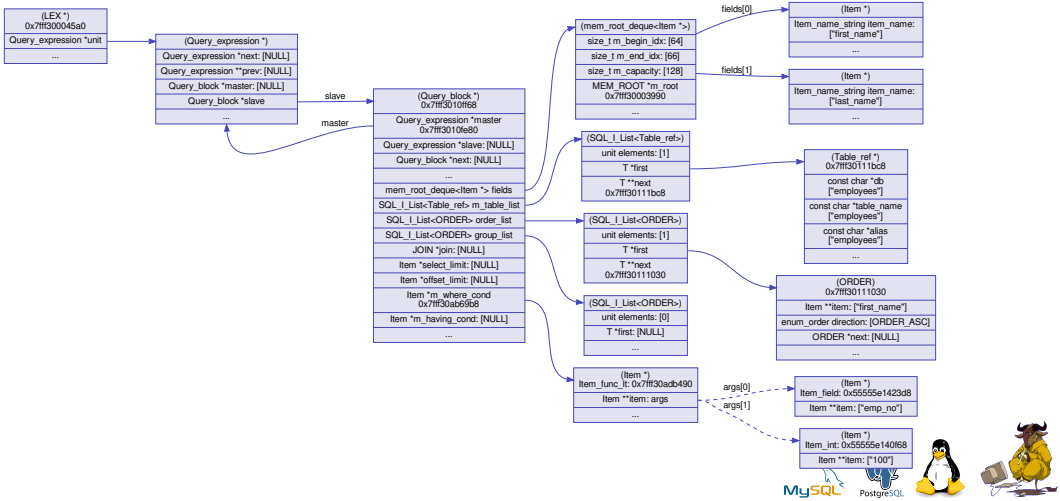
```
45 template <typename T>
46 class SQL_I_List {
47  public:
48   uint elements;
49   /** The first element in the list. */
50   T *first;
51   /** A reference to the next element in the list. */
52   T **next;
```

Query_expression 和 Query_block 类关系



生成词法单元的数据结构

```
select first_name, last_name from employees where emp_no < 100 order by first_name
```



解析 where 条件的参数过程

- m_where_cond 是一个 Item_func_lt 的实例 [3]，其父类 Item 变量 args 记录参数列表

```
#0  THD::sql_parser (this=0x7fff30001050) at /opt/src/mysql-server/sql/sql_class.cc:3071
(gdb) set $a = ((Item_func_lt *)($b->m_where_cond))->args
(gdb) p (**($a)).item_name
$17 = {
  <Name_string> = {
    <Simple_cstring> = {
      m_str = 0x7fff30110be8 "emp_no",
      m_length = 6
    }, <No data fields>},
  members of Item_name_string:
  m_is_autogenerated = true
}
(gdb) p (**($a+1)).item_name
$18 = {
  <Name_string> = {
    <Simple_cstring> = {
      m_str = 0x7fff30110cb8 "100",
      m_length = 3
    }, <No data fields>},
  members of Item_name_string:
  m_is_autogenerated = true
}
(gdb)
```

[3] https://dev.mysql.com/doc/dev/mysql-server/8.0.37/classItem___func___lt.html

更多示例语句

1. union 子句示例

```
select first_name from employees where emp_no < 100
 union
select last_name from employees where emp_no < 100
```

2. 子查询示例

```
select *
  from (select first_name, last_name
          from employees
         where emp_no < 100) a
 order by last_name
```

结束