基于 PostgreSQL 解析器模块的底层研究与分析

曹 阳[1],张 冰[2]
(1.同济大学 电子信息与工程学院,上海 201804;
2.煤炭科学研究总院经济与信息研究分院,北京 100013)

摘 要:介绍了 PostgreSQL 的整体架构,着重针对其解析器模块进行底层研究,分析查询请求在解析器模块中的查询处理过程和其内部各种函数的调用关系,最后就其解析器模块的开源代码做出的剖析以全面理解该模块的运作,并通过一个子查询程序调试实现了一个基本的查询请求。

关键词:数据库;解析器;查询处理;转换处理
中图分类号:TP311.132.4 **文献标识码**:A **文章编号**:1674-7720(2011)21-0001-05

Deep study and analysis on the parser module of PostgreSQL database

Cao Yang[1], Zhang Bing[2]
(1.Department of Electronics Information and Engineering, Tongji University, Shanghai 201804, China;
2.Economy and Information Research Branch of China Coal Science Research Institute, Beijing 100013, China)

Abstract: The paper introduced the integrated framework of PostgreSQL firstly. Then, put an emphasis on its parser, and analysed the query processing procedure of the query requests in parser and the calling relationship of various functions internally. Finally, dissected the open-source code of parser in order to understand the movement in parser in a round. Last but not least, we realized an essential query request by debugging a subquery programme.

Key words: database; parser; query processing; conversion processing

PostgreSQL[1]始自于美国加州大学伯克利分校的数据库研究计划“Ingres 项目”,最早被命名为 Postgres。历经多年的改善和发展,于 2006 年成立了 EnterpriseDB 公司,其目的为了让 PostgreSQL 能更好更方便地为企业级用户服务。现在,PostgreSQL 数据库已经成为当今世界上特色最鲜明、功能最强大、内容最丰富和开发人员最多的开源数据库系统之一,是一种复杂的关系型数据库管理系统(ORDBMS)。它的很多特性正是当今许多商业数据库的前身。

PostgreSQL 数据库支持多版本并发控制,可以支持所有广为使用的 SQL 结构,并且兼容时下最流行的计算机开发语言,如 C、C++、Java、Perl 等。与其他主流数据库一样,PostgreSQL 拥有诸多现代数据库特征,其特征特性涵盖了 SQL-2/SQL-92 和 SQL-3/SQL-99。作为一款开源软件,PostgreSQL 数据库不断被完善,拥有如下诸多优势:

(1)事务支持更彻底。对拥有海量存储的数据库,若一个查询请求长时间运行无果,就很可能会导致阻碍表的更新。MySQL 对于无事务的 MyISAM 表所采取的处理方式为“表锁定”,而 PostgreSQL 则根本不存在这类问题。

(2)完美支持存储过程。通过存储过程,PostgreSQL 便能轻松完成商业逻辑封装,极大减少了服务器的运转负荷。独有的内在机制会自行优化设计原存储过程,最大限度地避免传输大量原始查询语句,从而提高运行效率。

(3)支持子查询和触发器。使用子查询语句使得 PostgreSQL 在高效运行的同时,拥有更高的程序可读性;而触发器则更有利于其商业逻辑封装,可减少应用程序对同一商业逻辑的重复控制,从而保证数据的完整性。

(4)支持多种特殊数据类型和自定义扩展需求。一般来说,特殊行业的数据会比商业数据更为复杂多变,而 PostgreSQL 将多维数据的集合体描述成一个对象类型,并作为属性存储在表中。这样能保证用户自定义数据类型与原有操作符运算规则一致,并兼容现有数据类型。

尽管 PostgreSQL 有着无可比拟的优势,但仍不可避

免地存在缺点。如其稳定性和效能上有待提高;欠缺一些高端数据库管理系统所需要的特性(如联机热备份、数据库集群等)。

1 PostgreSQL 开源数据库的体系架构

就架构技术而言,PostgreSQL 数据库采用了经典的客户端/服务器(Client/Server)模型,其客户端进程与服务器端进程一一对应。很多工作经由客户端处理后再提交给服务器,极大地提高了客户端的响应速度,从而充分发挥客户端的处理能力。

PostgreSQL 查询语句的执行流程如图 1 所示。当出现查询连接请求时,主进程(Main)派生一个服务器监听进程(Postmaster),以等待从 TCP/IP 端口送入的连接请求。客户应用端通过调用库函数(Libpq)发出连接请求,并把用户请求反馈给 Postmaster,后者派生出后台服务进程(Postgres),使其与客户端进程直接对接(Libpq 仅支持一个客户端进程对接多个后台服务进程)。随后,客户端进程和后台服务进程不通过 Postmaster 而直接通信。Postmaster 和 Postgres 运行在数据库服务器中,而客户端应用则可运行在任何机器上。服务器进程之间利用信号标志和共享内存进行通信,确保并发数据访问过程的数据完整性。一个查询请求经历的完整过程可分为 5 个阶段:

(1)连接阶段:客户端向服务器发出查询请求,通过 Postmaster 与 Postgres 建立通信对接。

(2)解析阶段:解析器分析查询请求,核对语法无误后创建一个查询树。

(3)重写阶段:在系统目录(pg_rewrite)中匹配重写规则,重写系统根据规则体进行重写转换,随后创建解析树作为结果输出。

(4)优化阶段:优化器根据解析树创建查询规划。首先确定同一查询结果的所有查询路径,然后计算不同查询路径的执行成本并选择最优路径,最后将其拓展为完整查询规划树。

(5)执行阶段:执行器对查询规划树进行递归扫描,按其指定的执行方式检索数据记录。在扫描关系时,使用存储系统执行排序和连接操作,计算条件后反馈最终查询结果[2]。

2 PostgreSQL 解析器模块的内部结构

PostgreSQL 数据库的解析器模块由解析器和转换处理器组成。解析器又分为词法分析器和语法分析器两部分,分别由 lex 和 yacc 创建,定义在 scan.l 和 gram.y 中。解析器生成原始解析树(即为一种数据结构)后,转换处理器再对其进行修正和增补,最终生成查寻树。

词法分析器(yylex())的主要功能是做查询语句词法检测,用来识别其中的标识符、SQL 关键字等。它对每个关键字或标识符都会生成一个标记符号并将该标记结果反馈至语法分析器。语法分析器(yyparse())主要功能

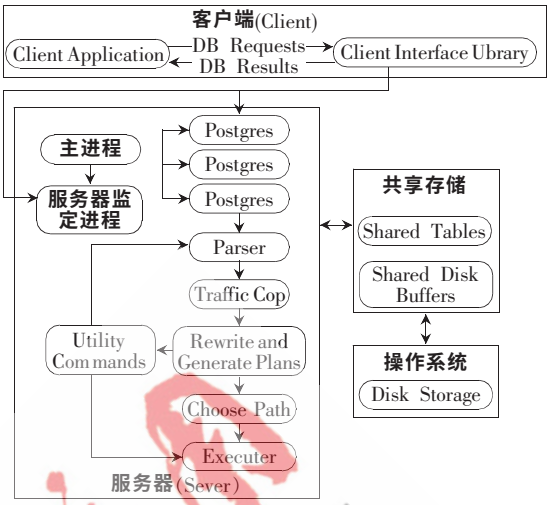

图 1 PostgreSQL 查询语句的执行流程

是对查询语句做语法检测。它由一套语法规则和触发规则组成,当任意一条规则被触发时,两者将被执行。

所谓语法语义解析,即将来自库函数的 SQL 查询请求转化为解析树,供优化器/执行器或指令集使用。在解析阶段,当以 ASCII 码组成的查询字符串到达解析器后,首先对其执行词法分析,将其转换为解析器所能识别的关键字、标识符和常量;再执行语法分析,检查该查询字符串的语法的有效性,并生成命令形式的查询结构(即解析树)。若语法正确,解析器将会反馈生成的解析树;若语法错误,则返回一个错误标志或空值;随后,解析树被分离、检查和传送到指令集中的处理函数,或转换为结点链表供优化器/执行器处理。

在解析器阶段,解析器只依据与 SQL 语法结构相关的固定规则创建解析树。由于自身不会自动查询任何系统目录,因此就不能理解查询语句的详细语义。故查询语句被转换为解析树后,转换处理器将对其做进一步处理,分析查询语句所引用的表、函数和操作符的语义,然后生成查询树(Query Tree),用来表示解析树具体信息的数据结构。把原始解析(仅包含查询语句的原始信息)和语义分析分成两个过程是因为系统目录查询操作只能在一个事务中执行,并且无法在接收到查询字符串后就立即发起一个事务。服务器一旦确认正在处理一个查询操作,那么就可以发起一个事务。此时,转换处理器才能被顺利调用。

从数据结构来说,转换处理器生成的查询树与原始解析树十分类似,但是结构细节上仍有差异。例如:在一个解析树中,一个 FuncCall 节点可能在语句构成上被理解为函数调用功能,也可能根据引用名是普通函数还是聚合函数被转换成 FuncExpr 节点或 Aggref 节点。类似地,查寻树也可能添加了具体数据类型的信息(如相关字段或表达式)。本文以 transformFuncCall 调用为例:

(1)对每一个参数调用 transformExpr;

(2)调用 ParseFuncOrColumn:

①调用 ParseComplexProjection 辨别 Column Projection 函数：关系型参数和复合类型参数。

②调用 func_get_detail(普通函数)：

• 调用 FuncnameGetCandidates，在 Cache PROCNAME-ARGSNSP 中找到符合查询条件的候选函数列表 FuncCandidateList；

• 返回无需做参数类型转换的项；

• 若只有一个参数，则尝试对系统未定义的类型转换型函数做二进制兼容的强制转换；

• 若为一般函数，则在候选函数中找最匹配函数 func_match_argtypes；

• 尝试解决存在的冲突调用 func_select_candidate；

• 返回值。

③依据返回值采取相应措施；

④生成相应节点并返回：FuncExpr 节点或 Aggref 节点[1]。

3 PostgresSQL 解析器模块的源码剖析

PostgreSQL 开源数据库的源代码相对路径为 Postgresql\source\src\backend，其代码整体编译流程大致如下：PostgreSQL 在 exec_simple_query 函数中处理与用户交互的简单查询请求，其查询操作使用两种协议：Simple query protocol 和 Extended query protocol。Extended query protocol 包含 Parse、Bind、Execute 三个步骤，并定义了 Prepared Statement 和 Portal 两种重要数据结构。Prepared Statement 用来表示词法解析、查询规划等结果，Portal 用来表示可以被执行或已部分执行完毕的语句。由于本文侧重研究解析阶段，故只对 Simple query protocol 协议进行具体分析，第三步生成物理查询规划和第四步执行物理规划由于不是本文的侧重点在此不赘述。

(1)编译查询

通过 pg_parse_query 函数返回一系列查询解析树，并生成列表文件 parsetree_list。具体工作由 raw_parse 函数通过调用 base_yyparse 函数对查询语句进行词法分析完成。

(2)生成逻辑规划与查询重写(QueryRewrite)

通过语义分析将已有解析树转化为所需查询规划，建立最优查询规划。具体由 pg_analyze_and_rewrite 函数调用 parse_analyze 函数转换完成。部分代码如下：

```
foreach(parsetree_item, parsetree_list)
 {
 Node*parsetree=(Node*)lfirst
 (parsetree_item);
 …
}
```

再执行 parse_analyze 函数中的 query=transformStmt(pstate, parseTree)；最后，通过调用语句 querytree_list = pg_rewrite_query(query)；完成查询重写。pg_rewrite_query 函数通过调用查询重写器的入口函数 QueryRewrite 来完成操作，而查询重写的作用就是根据系统或用户自定义规则来重写规划。

PostgreSQL 解析器模块所接受的查询请求表现形式为查询字符串，经过解析生成原始解析树，并返回列表文件(parsetree_list)，其每一个节点都是一个完整的语法解析树。一个原始解析树由多个节点构成，一个节点表示一个类型(由 NodeTag 定义)。PostgreSQL 解析器模块内部函数调用关系如图 2 所示。

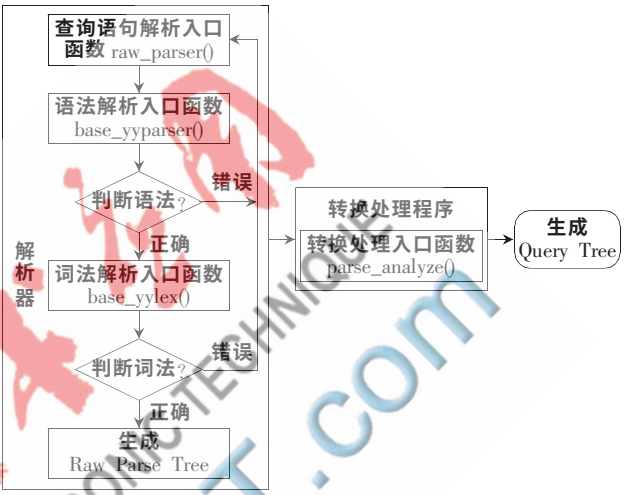

图 2 PostgreSQL 解析器模块内部函数的调用模型

解析器模块有 3 个主要函数，分别是：raw_parser()、free_parser()和 filtered_base_yylex()，可以在.\Parser\Parser.c 文件中找到其定义。下面就其主要源码进行剖析和理解：

(1)raw_parser()

```
raw_parser(const  char  *str)
 {
  Int  yyresult;
  …
  return  parsetree;
 }
```

输入：为输入的查询字符串定义一个指向字符常数的指针。

功能：扫描查询字符串并返回相应值。

返回值：若无语法错误，则建立解析树(Parser Tree)，并将查询语句存储在该解析树中；否则，向解析器返回空指针(NIL)。

所调用函数：pool_memory_create ()； scanner_init()； parser_init()； base_yyparse()

(2)pg_parse_string_token()

```
pg_parse_string_token(const  char  *token)
 {
  Int  ctoken;
  …
  return  base_yylval.str;
 }
```

输入:为所给字符串定义一个指向字符常数的指针。

功能:获取所给字符串的值。

返回值:base_yylval.str;

调用函数:scanner_init(); base_yylex(); scanner_finish()。

(3)filtered_base_yylex()

```
filtered_base_yylex(void)
{
 Int  cur_token;
 …
 return  cur_token;
}
```

输入:无输入值。

功能:作为解析器的中间件,对查询语句执行优化处理,过滤部分累赘语句。例如:当查询语句存在多个空字符时,该函数会自动将其合并。

返回值:无返回值。

调用函数:base_yylex()。

(4)free_parser(void)

输入:无输入值。

功能:释放已生成的语法解析树所占用的内存空间。

返回值:无返回值。

调用函数:pool_memory_delete()。

对查询语句做转换处理是为了把生成的解析树都转换成根为 Query 型节点的查寻树。转换处理工作在 parse_analyze 函数中进行:首先定义 ParseState *pstate 函数,用以记录转换处理过程中的状态信息,然后调用 transformStmt 函数进行正式的转换处理工作,它会根据解析树不同类型的根节点调用相应的函数。部分代码如下:

```
SelectStmt*n=(SelectStmt*)parseTree;
 if(n->valuesLists)
  result=transformValuesClause(pstate,n);
  else  if(n->op==SETOP_NONE)
  result=transformSelectStmt(pstate,n);
 else
  result=transformSetOperationStmt(pstate,n);
```

Select statement 分为简单型和复合型。复合型内含集合操作符,在这种语法树中,Select 语句都处于叶节点位置,而内部节点则表示集合操作符。简单型也可分为两类:一类有 VALUES,一类无 VALUES。

把原始解析树转换处理到查寻树的目标就是为了从系统关系表中得到相关信息并据此进行相关节点的转换和信息添加。得到了查寻树后,它将被执行重写、优化、执行等操作,至此完成了一次完整的查询请求。

4 PostgreSQL 子查询调试

由于 PostgreSQL 是个开源数据库,可以支持多种操作系统的编译与执行。本文选用以 Linux 为内核的 Ubuntu v11.04 作为调试环境,其内核版本为 2.6.38-8-generic。操作系统加载的 PostgreSQL 版本为 v8.4.8。搭建系统中,所选用的调试工具为 DDD,其版本号为 v3.3.12。所选用编译器为 gcc,其版本为 v4.4.5,库版本为 GNU libc v2.12。

执行子查询调试步骤如下:

首先,在 PostgreSQL 数据库中建立一张新表,表名为“film”,该表中存放电影相关信息字段:

```
Create  table  "film"
(
 "id"  Serial  NOT  NULL  UNIQUE,
 "filmname"  Varchar(20)NOT  NULL,
 "director"  Varchar(10)NOT  NULL,
 "actor"  Varchar(10)NOT  NULL,
 "production"  Varchar(40)NOT  NULL,
 "description"  Varchar(255),
 UNIQUE(filmname),
 PRIMARY  KEY("id")
);
Create  index  "film_index"  on  "film"  using  btree("id","filmname");
```

然后,再在数据库中建立第二张表,表名为“cinema”,该表中存放有某一部电影的放映信息:

```
Create  table  "cinema"
(
 "id"  Serial  NOT  NULL  UNIQUE,
 "cinema"  Varchar(40)NOT  NULL,
 "releasedate"  Timestamp  Default  now(),
 UNIQUE  (cinema),
 PRIMARY  KEY  ("id")
);
Create  index  "cinema_index"  on  "cinema"  using  btree("id","cinema");
```

在所建的两张表中,同时建立了 B-tree 类型的索引。B-tree 索引主要处理那些按照某种顺序存储的数据查询,它能为索引的每个列(字段)声明一个操作符表。PostgreSQL 的 create index 语句一般都默认使用 B-tree 索引类型。

最后,再新建一张链接表,表名为“announcement”。该表通过建立外部关键字(foreign key)链接生成,主要用来显示所需要查询的某部电影的相关信息:

```
Create  table  "announcement"
(
"id"  Serial  NOT  NULL  UNIQUE,
"fid"  integer  NOT  NULL  Default  0,
"cid"  integer  NOT  NULL  Default  0,
PRIMARY  KEY  ("id")
);
Alter  table  "announcement"  add  foreign  key  ("fid")  references  "film"("id")on  update  restrict  on  delete  restrict;
```

```
Alter table "announcement" add foreign key ("cid") references "cinema"("id")on update restrict on delete restrict;
```

在更改好表“announcement”中的相关字段属性后，通过新建视图表来完成所查询请求，并显示相关结果：

```
CREATE VIEW v_announcement AS
 SELECT * from announcement where cid =1
  FROM "film" f ,"cinema" c, "announcement"a
  Where f.id = a.fid and c.id = a.cid
ORDER BY a.id;
```

子查询的结果如下：

```
filmdb=# select*from announcement where cid=1;
id | fid | cid
---+---+---
4 | 2 | 1
5 | 7 | 1
6 | 8 | 1
(3 rows)
```

该结果中，字段 fid 即为表“film”中的 id 字段，每一个 id 字段值对应唯一一部电影；字段 cid 即为表“cinema”中的 id 字段，每一个字段值所对应的电影都被标示着相应的放映信息。在本次查询请求的建立表中，表“cinema”中的字段 id=1 所对应的 cinema=Wanda，即表示所进行的查询请求就是显示所有在万达影城播放的电影信息。更多具体电影和放映信息在表中略去，但通过增加字段来完整显示即可。

PostgresSQL 开源数据库目前已被成功应用于包括医疗、电子商务等在内的广阔领域中。经过二十多年的快速发展，PostgreSQL 的内部结构和工作机制已经相当成熟，直到今天依然在全球众多开源程序员的努力中不断地被改进和优化。

本文主要对 PostgreSQL 开源数据库的解析器做了底层研究并对该模块的源码做了一定分析，并通过一个子查询调试实验实现了一个基本的查询请求。针对 PostgreSQL 开源数据库的源码分析在国内尚处起步阶段，而源码分析对深入理解 PostgreSQL 开源数据库的开发思想有着里程碑的意义。

参考文献

[1] PostgreSQL 中文之家. [2010-05].http://www.pgsqldb.org.

[2] The PostgreSQL Global development Group. PostgreSQL 8.4.8 documentation [S/OL]. (2010-05-17) [2011-04-02]. http://www.postgresql.org/docs/8.4/static/index.html.

[3] 陈景峰.PostgreSQL 实用实例参考 [R/OL](2004-5-20). [2011-04-20].

[4] 郭龙江，李金宝.PostgreSQL 分析器的研究[J].黑龙江大学自然科学学报，2001，18(4)：49-52.

[5] DOUGLAS K, DOUGLAS S. PostgreSQL: the comprehensive guide to building, programming, and administering PostgreSQL databases, second edition[M]. [S.l.]: Sams Publishing,2005.

[6] 陈文星，付继宗.Linux 下数据库 PostgreSQL 分析与应用[J].电脑开发与应用，2006，19(11)：56-57.

(收稿日期：2011-06-04)

作者简介：

曹阳，男，1987 年生，硕士研究生，主要研究方向：数据库应用，电子商务及工人智能。

张冰，男，1979 年生，硕士研究生，工程师，主要研究方向：企业信息化建设，Web 开发技术及数据集成。