文章编号:1000-5641(2014)05-0281-09

面向 OceanBase 的存储过程实现技术研究

朱 涛, 周敏奇, 张 召
(华东师范大学 软件学院, 上海 200062)

摘要: 存储过程是一段被命名后保存在数据库服务器端,并预先编译好的代码,可以减少前台应用程序和后台数据库间的网络传输量. 本文主要研究基于静态语言和动态语言,两种典型的存储过程实现方法,来阐述存储过程实现的基本原理. 并基于此,进一步探讨了在 OceanBase 主存数据库服务器中添加存储过程模块的解决方案和技术难点.

关键词: 存储过程; 数据库系统; SQL
中图分类号: TP392 **文献标识码**: A **DOI**:10.3969/j.issn.1000-5641.2014.05.025

Study on stored procedure implementation oriented to OceanBase

ZHU Tao, ZHOU Min-qi, ZHANG Zhao
(*Software Engineer Institute, East China Normal University, Shanghai* 200062, *China*)

Abstract: A stored procedure is a pre-compiled subroutine stored in database server, which improves the efficiency of applications' database access. This paper discussed the implementation of stored procedure based on both static language and dynamic language. Besides, we gave a primary design for implementing stored procedures in OceanBase.

Key words: stored procedure; database system; SQL

1 背景简介

在各类以数据管理为核心的应用中,数据库通常扮演着底层存储的角色.应用服务器通过网络通信与数据库服务器进行交互,当某些任务需要多次访问数据库,应用服务器需要与数据库服务器进行多次的网络通信,并传送大量的中间结果. 这会影响应用对用户查询的响应速度. 为了解决这一问题,当前主流数据库 ,如 Oracle, Mysql, DB2 等,均允许用户在数据库服务器端编写程序来实现本在应用程序段完成的业务逻辑. 这段被命名后预先编译存储在数据库服务器端的代码即为存储过程[1].

1.1 存储过程简介

存储过程是大型数据库系统中,一组完成特定功能的程序集,经编译后存储在数据库

收稿日期:2014-07
基金项目:国家自然科学基金(12345678);上海市重点学科建设项目(98776654)
第一作者:朱涛,男,博士研究生,研究方向为内存数据库,分布式系统. E-mail:zhutaojs@gmail.com.
通信作者:周敏奇,男,副教授,硕士生导师,研究方向为内存数据库. E-mail:mqzhou@sei.ecnu.edu.cn.

中,上层应用调用存储过程只需要给出存储过程名和相应的参数即可. 这样可以减少应用程序与数据库间的数据交互次数和数据传输量. 因此,在一个以数据库为核心的应用系统中,需要跟数据库频繁交互的业务逻辑往往用存储过程来完成.

总的来说,存储过程具有以下三个优点:

(1) 预先编译,运行速度快. 如果跟数据库的交互工作是在前台的程序设计语言中完成,那么,每次跟数据库连接,程序中相应的 SQL 语句就要编译一次,而存储过程却可以一次编译多次运行.

(2) 前台应用程序和数据库服务器之间只传存储过程名称和参数,网络通信量少. 使用存储过程能在数据库服务器上完成全部对数据库的操作,只需返回最终结果. 从而减少了数据库服务器与应用程序之间的交互次数和数据传输总量.

(3) 对数据库访问封装,安全性高. 参数化的存储过程可以防止 SQL 语句的注入式攻击,并且可以将使用 Grant、Deny 以及 Revoke 等语句来完成 = 存储过程级别的权限管理.

然而,存储过程除了以上的三个优点以外,也存在以下的三个缺点:

(1) 程序调试困难. 由于存储过程是运行在数据库端的一段代码,并基于数据库提供的独特的编程语言来实现. 导致常用的调试工具不能直接用来调试这类代码. 因此,在开发存储过程时,通常面临调试困难的问题.

(2) 代码移植困难. 不像 SQL 语言,存储过程语言并没有被标准化. 不同的数据库支持存储过程的开发语言都不尽相同. 当 Web 应用使用的数据库系统发生变化时,在两个数据库系统上移植相应的存储过程需要重新开发和调试.

(3) 数据库负担重. 存储过程虽然减少了应用和数据库之间的交互,但同时增加了数据库的计算负担. 例如,原本对查询结果集的遍历访问是在应用层完成的,数据库只需要提供相应的查询结果即可. 但在存储过程中,数据库服务器不仅要提供查询结果,还需要完成遍历访问以及一些必要的数据加工和计算,并产生最终的结果返回给上层应用.

1.2 已有数据库系统对存储过程的支持

虽然主流的数据库管理系统都支持存储过程,但是它们支持的语法和使用的方法却有很大区别. 这是因为存储过程还没有形成统一的规范,并且存储过程在各个系统中的实现也受限于系统本身的设计. 表 1 列举了当前一些数据库支持的存储过程语言,其中的数据来源于各自的网站. 从中不难看出,不同数据库所采用的存储过程语言大不一样. 即便是被多个数据库同时采用的 Java 语言,其实际的存储过程编程接口还是会有很大区别.

表 1 主流数据库系统存储过程开发语言

Tab. 1 Programming language of stored procedure in modern database systems

数据库系统	存储过程支持的语言
CUBRID	Java
DB2	SQL PL、Java
Firebird	PSQL
Informix	SPL、Java
Microsoft SQL Server	Transact-SQL 和其他. Net Framwork 语言
MySQL	SQL/PSM
Oracle	PL/SQL、Java
PostgreSQL	PL/pgSQL
Sybase ASE	Transact-SQL
VoltDB	Java

事实上,根据所使用的开发语言的性质,存储过程的实现可以分为两类:(1)采用静态过程语言实现,例如 SQL PL,PSQL,SPL,SQL/PSM,PL/SQL[2] 等;(2)采用已有的支持反射机制的动态编程语言实现,例如 Java,. Net Framework 语言. 在第 3 节和第 4 节,我们将以两个分别使用静态过程语言和已有动态编程语言的开源数据库中存储过程的实现来分别分析这两类实现方案.

2　存储过程支持的主要功能

存储过程是一段运行在数据库端的程序代码,能够被数据库系统动态地加载,编译和执行. 它支持数据库访问,变量定义,流程控制等功能. 一个数据库管理系统如果需要实现存储过程功能,需要考虑以下几个问题.

2.1　基本语法

存储过程需要支持的功能主要包括,变量声明、变量赋值、表达式计算、流程控制以及 SQL 语句执行,当然,也需要包括为了解决数据库和程序设计语言之间阻抗不匹配而引入的游标. 下面将对以上功能进行逐一阐述.

变量声明　存储过程支持的变量包括数据库本身支持的变量类型,以及表的行记录,游标等. 例如 integer,numeric(5,2),varchar,tablename%ROWTYPE 和 cursor 等. 在执行存储过程时,变量的取值,变量的类型等信息需要存储在过程的执行上下文中.

代码块定义　代码块可以用于逻辑分组或者把变量局部化为作用于一个比较小的语句组,在块内能定义临时变量,执行若干语句等.

表达式计算表达式计算需要处理包含常量和变量的表达式,需要产生正确的结果及结果类型. 表达式计算可以有单独的功能模块支持,也可以直接继承数据库提供的表达式计算功能.

变量赋值　变量赋值要求除了能够处理一般的数据类型,例如 integer, varchar 等,同时也要支持复合类型的赋值,如关系表中的行记录. 当然,变量赋值也应支持数据库中特有的游标类型的赋值.

流程控制　流程控制是提供编写复杂的业务逻辑所必须的功能. 主要包括条件语句、循环语句以及过程返回语句. 其中,循环语句需要支持对关系表记录的迭代.

数据库访问　其中包括执行 SQL 语句,使用游标等. 在这个模块,存储过程需要能够向主数据库引擎提交 SQL 查询,缓存 SQL 解析后的执行计划,定义和使用游标,将查询结果存储到变量中. 这个模块通常需要设计数据库的内部接口,供服务器端编程使用.

2.2　过程编译、执行与存储

存储过程的编译[6] 发生在过程初次编译阶段和过程执行阶段. 这两部分工作分别是由过程编译模块和 SQL 编译模块完成. 前者主要负责诸如变量定义赋值,流程控制等内容的编译工作,而后者负责 SQL 语句的编译工作. 通常,这部分工作是在存储过程被首次调用时完成的. 例如当首次执行 SQL 语句时,数据库内核会完成对 SQL 的解析,并产生执行计划. 这部分编译结果会由存储过程模块缓存.

过程的执行同样可以分为两个部分:(1)由存储过程模块执行的语句;(2)由数据库引擎执行的语句. 前者需要定义和实现每类语句的执行函数,完成语句的执行. 后者需要定义访问数据库的接口,通过访问接口来完成语句执行. 返回的结果需要复制到存储过程模块

执行的上下文中,并交由该模块进行下一步的处理.

与保存在数据库中其他信息一样,数据库管理系统也要保证存储过程在系统故障恢复后,依然可以正常使用. 因此,如何将过程序列化到非易失性存储器尤为重要. 根据采用的过程语言类型,当前主要有两种存储方案. 其中,第一种方案是,存储过程的源代码会存储在系统表中,编译后的目标代码存储在系统缓存中. 第二种方案是,在系统表中只记录存储过程的调用信息,例如过程名,参数信息等,而不存储过程源代码,编译后的结果存储在系统外部,当需要调用时,系统根据存储路径加载,并在系统内部缓存.

在下一节中,我们将以 PostgreSQL 为例,重点介绍设计自定义过程语言实现存储过程模块的机制.

3 静态语言存储过程实现机制

PostgreSQL[5,7]是一个自由的对象-关系数据库服务器(数据库管理系统),它在灵活的 BSD-风格许可证下发行. 它提供了相对其他开放源代码数据库系统(比如 MySQL 和 Firebird),和专有商用系统(比如 Oracle、Sybase、IBM 的 DB2 和 Microsoft SQL Server)之外的另一种选择.

PostgreSQL 虽然支持使用内置过程语言 PL/pgSQL、脚本语言、编译语言 C 和 C++ 等编写存储过程. 但本节重点以 PostgreSQL 为例,阐述静态语言存储过程实现的原理和机制.

3.1 PL/pgSQL 简介

PL/pgSQL 是 PostgreSQL 数据库系统的一个可装载的过程语言,其语法类似于 Oracle 的 PL/SQL. PL/pgSQL 的设计目标是创建一种可装载的过程语言:(1)可用于创建函数和触发器过程;(2)为 SQL 语言增加控制结构;(3)可以执行复杂的计算;(4)继承所有用户定义类型,函数和操作符.

为了支持 PL/pgSQL,Postgre 需要提供该过程语言的编译功能、执行功能、存储和数据库访问功能.

3.2 编译与执行

3.2.1 词法解析与语法解析

在存储过程的编译阶段需要将过程源代码编译为系统内部指令树. 但需要注意的是,在函数内使用到的独立的 SQL 表达式和 SQL 命令的编译则是在过程首次调用阶段完成,其由 PostgreSQL 的 SQL 解析模块负责编译,并产生执行计划.

正如传统的 SQL 一样,对过程源码的编译也需要经过词法解析和语法解析. 在 PostgreSQL 中采用开源软件 Flex 与 Bison 完成词法解析与语法解析. Flex 进行词法分析,这需要我们设计过程语言的语法后,为其设计正则表达式来匹配源码中出现的标记(常量,数学符号,括号,中括号,变量名,保留字等),并定义规则将标记映射为内部标志符. 之后,Bison 来对词法分析后的内部标志符序列进行语法分析,形成抽象语义树. 我们需要为每个语义单元设计识别规则,并为其创建一个相应的结构保存.

3.2.2 过程执行

过程的执行实际上是一个遍历语法树的过程,根据当前访问的指令,转到对应的指令函数进行执行. 图 1 以 IF 语句为例,来说明这个过程.

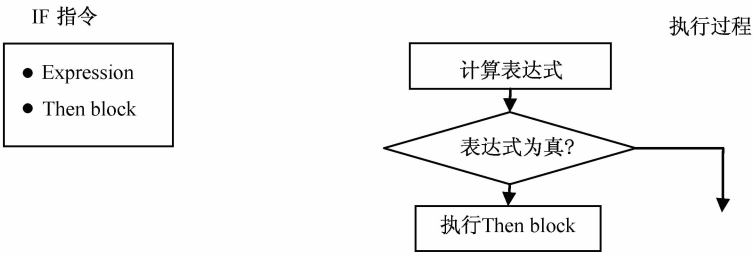

图 1　IF 语句的指令函数

Fig. 1　Handler for IF statment

对于每个语法树中的节点,均有一个对应的指令函数负责该指令的解释运行. 整个过程的执行是从指令树的根节点开始遍历,并在运行时决定执行的路径.

3.2.3　数据库访问

在 PostgreSQL 中,对于需要访问数据库的指令,指令函数需要访问系统提供的内部访问接口来完成数据库访问操作,例如,执行一条查询语句、插入数据或者计算某个表达式的结果. 在 PostgreSQL 内部,提供了服务端编程接口(Server Programming Interface)来接收来自系统内部的数据操作和访问请求. 在下表中,表 2 列举了接口包含的部分重要访问函数:

表 2　SPI 访问接口

Tab. 2　Server Programming Interface

函数名称	函数功能
SPI_connect	开启过程对 SPI 的访问,保存存储过程模块和主数据库引擎的上下文.
SPI_finish	关闭过程对 SPI 的访问,释放临时资源.
SPI_prepare	解析 SQL,产生执行计划.
SPI_execute	执行一条 SQL 指令.
SPI_cursor_open	打开游标
SPI_cursor_fetch	操作游标,fetch
SPI_cursor_close	关闭游标

3.3　存储策略

存储过程的存储,即是对过程的持久化,要保证过程在系统重启后依旧能够使用. 在 PostgreSQL 中,过程存储分为两个部分,过程的源代码和编译后的指令集.

过程的源代码会被存储在系统表 pg_proc 中. 该系统表用于存储关于函数或者过程的信息. 表 3 列举了需要存储的部分重要属性. 从表 3 可以看出,系统表主要存储了过程的原始信息,包括名字,源码,参数,调用方式等. 由于这些信息是存储在系统表中的,从而保证了过程存储的持久化.

过程会在创建或第一次调用的时候被编译. 编译后的执行计划会被存储在位于系统缓存中的 hash 表中. 每个过程对应一条记录,主键为根据过程名和参数计算得到的 hash 值,映射的对象是过程编译后的指令集.

对过程的调用,首先会从系统表中找到对应的过程记录,然后在系统缓存中寻找是否存在已编译的指令集,如果没有,那么根据系统表中的 prosrc 字段重新编译,并存储到系统缓存中.

表 3 pg_proc 表的部分重要属性

Tab. 3 Part of important attributes in pg_proc

名字	类型	描述
Proname	name	函数名字
prolang	oid	函数的实现语言、调用接口
pronargs	Int2	参数数目
prorettype	oid	返回值的数据类型
proargtypes	Oidvec-tor	一个存放函数参数的数据类型的数组. 数组里只包括输入参数(包括 INOUT 参数),因此代表该函数的调用签名(接口).
proargnames	Text[]	一个保存函数参数的名字的数组. 没有名字的参数在数组里设置为空字串. 如果没有一个参数有名字,这个字段将是空.
prosrc	text	这个字段告诉函数句柄如何调用该函数. 它实际上对于解释语言来说就是函数的源程序,或者一个链接符号,一个文件名,或者是任何其他的东西,具体取决于语言/调用习惯的实现.
probin	bytea	关于如何调用该函数的附加信息. 同样,其含义也是和语言相关的.

3.3 静态语言实现存储过程小结

PostgreSQL 采用的存储过程实现方式是定制了自己的过程语言 PL/pgSQL,在系统内部实现过程的编译、运行和存储. 这种策略的主要工作事实上可以分为两个部分:(1)提供过程语言的编译和运行机制;(2)设计过程语言访问数据库系统的接口. 而大量的工作在于提供过程语言的基本功能. 在下一节中,我们将介绍一种更为简明的,利用动态语言中的反射机制来实现存储过程的策略.

4 动态语言存储过程实现机制

VoltDB[8,9]是一个内存数据库[10],提供了 NoSQL 数据库的可伸缩性和传统关系数据库系统的 ACID 一致性. 使用 Java 编写的存储过程进行数据操作. 我们以 VoltDB 系统为例来阐述使用动态语言来实现存储过程的原理,VoltDB 在上层采用的是 Java 语言开发,负责系统的集群管理,事务调度,SQL 解析,接受客户端连接等功能;下层采用的 C++ 语言开发,负责 SQL 的执行和数据的存取. 存储过程实现在 Java 层.

4.1 反射机制

从效果来看,存储过程是一段动态添加到系统中执行的代码. 而 Java 提供的反射机制具备这样的功能. 反射机制允许 Java 在运行时加载、探知、使用编译期间完全未知的类. Java 可以加载一个运行时才得知名称的类,获悉其完整构造,并生成对象实体、或修改成员变量,或调用方法.

VoltDB 利用该机制,为应用程序提供了可扩展的编程接口,应用程序员通过继承系统提供的过程基类,定制应用访问数据库的逻辑,并在系统外部使用 Java 编译器进行编译,并最终交由系统在运行时加载和调用.

4.2 过程的编译与执行

VoltDB 中存储过程的流程控制,变量定义等功能完全由 Java 提供,而对于需要访问数据库的操作,设计了相应的接口. 每一个存储过程都需要继承这个接口,并使用相应的接口方法来提交查询,获得结果.

下面我们给出一个 VoltDB 中存储过程的样例.

```
public class KvUdpate extends VoltProcedure {
    public final SQLStmtaddRecord = new SQLStmt
        "insert into kvstore values(?,?);"
    );
    publicVoltTable[] run(int a, int b) throws Exception{
        voltQueueSQL(addRecord, a, b);
        voltExecuteSQL(true);
        return null;
    }
}
```

这段代码会被 Java 编译器编译. 例如:

$javac -cp $"CLASSPATH:/opt/voltdb/voltdb/*"UpdatePeople.java

这里 Java 编译器只负责编译 Java 语法相关的内容. 而对于 SQL 语句是由 VoltDB 的 SQL 解析引擎在过程调用时负责编译和缓存.

VoltProcedure 主要提供了如表 4 所示的接口函数:

表 4　VoltProcedure 主要接口函数

Tab. 4　Main interface functions of VoltProcedure

函数名	函数功能
voltQueueSQL	提交一个 SQL 请求
voltExecuteSQL	执行提交的请求
setAppStatusString	设置对客户端的返回信息

需要注意的是,在 VoltProcedure 中并没有提供游标等功能,对表的迭代访问是使用 VoltTable 提供的相应操作方法. 编译与执行的整体框架如图 2 所示.

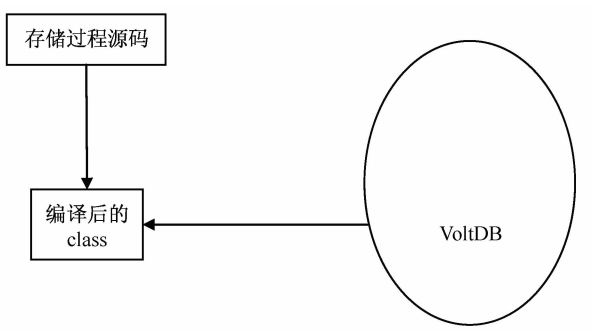

图 2　VoltDB 存储过程框架

Fig. 2　Framework of the stored procedure in VoltDB

4.3　过程的存储

存储过程采用 Java 编写,在系统外进行编译和存储,关于过程的信息,例如过程名,参数,使用的 SQL 语句会被存储在系统表中. 在外部的类文件被调用后,系统内部会缓存类信息,已备下一次的调用.

编译后的存储过程会在系统表中进行注册. 例如:

CREATE PROCEDURE FROM CLASS UpdatePeople;

PARTITION PROCEDURE UpdatePeople ON TABLE people COLUMN state_num;

运行以上的命令后,系统表中就记录了这个过程的相关信息,从而方便之后的再调用. VoltDB 对过程的调用是通过系统表中提供的过程名和相应的存储路径,并据此,从外部加载相应的类,通过反射机制生成该类的实例,调用相应的 run()方法.

4.4 支持反射的动态语言实现存储过程小结

利用动态语言(例如,Java)的反射机制能够非常便捷地实现存储过程. 它不需要实现存储过程的编译和执行,从而大大降低了开发难度. 然而,这取决于底层数据库采用的实现语言是否能够提供反射机制,因而已有利用的动态语言来实现存储过程有一定的局限性.

5 OceanBase 中的实现思路

OceanBase[3,4] 是阿里集团研发的可扩展的关系数据库. 目前的版本暂时不支持存储过程的编写,本小节我们将探讨在 OceanBase 中添加存储过程模块的关键技术. OceanBase 整机架构分为四个模块:主控服务器 RootServer,更新服务器 UpdateServer,基线数据服务器 ChunkServer 以及合并服务器 MergeServer.

◆ 客户端:用户使用 OB 的方式和 MySQL 数据库完全相同,支持 JDBC、C 客户端访问.

◆ RootServer: 管理集群中的所有服务器,Tablet 数据分布以及副本管理. RootServer 一般为一主一备,主备之间数据强同步.

◆ UpdateServer:存储 OB 系统的增量更新数据. UpdateServer 一般为一主一备,UpdateServer 和 RootServer 一般部署在同一服务器中.

◆ ChunkServer:存储 OB 系统的基线数据. 基准数据一般存储两份或者三份.

◆ MergeServer:接收并解析用户的 SQL 请求,经过词法分析、语法分析、查询优化等一系列操作后转发给相应的 ChunkServer. 如果请求数据分布在多台 ChunkServer 上,MergeServer 还需对多台 ChunkServer 返回的结果进行合并.

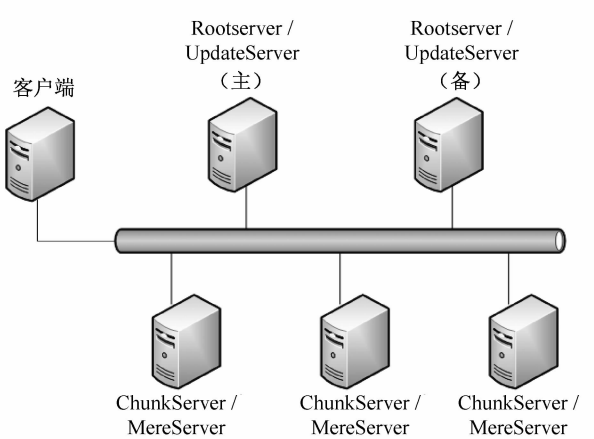

图 3 OceanBase 架构

Fig. 3 Architecture of OceanBase

正如之前所讨论的,存储过程的模块的编写主要有以 PostgreSQL 为代表的静态语言过程语言的方式,以及以 VoltDB 为代表的基于动态语言反射机制的方式.

由于 OceanBase 是采用 C++语言开发的,所以实现存储过程的策略主要参考 Postgre 的实现方案. MergerServer 是 OB 查询的入口,因而存储过程实现在 MergeServer 上. 添加存储过程模块,需要增加存储过程编译模块,存储过程执行模块,Obsql 对存储过程定义和调用的支持以及存储过程访问 SQL 引擎的接口,其基本框架如图 4 所示:

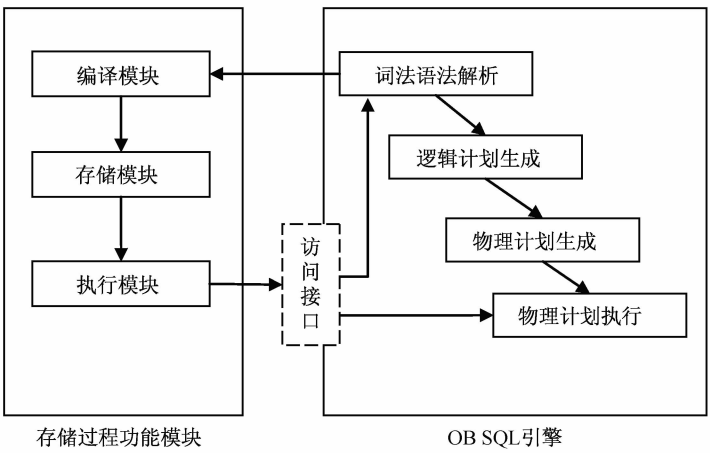

图 4　存储过程基本框架设计

Fig. 4　The design of the basic framework of the stored procedure

- 定义存储过程的指令首先被 MergerServer 接收,MergerServer 调用 SQL 词法语法解析模块,对过程定义的部分会转到存储过程的编译模块完成.
- 编译后的结果可以缓存在系统内部,过程源代码可以存储在系统表中.
- 调用一个存储过程通过 SQL 查询的入口点,转到存储过程的执行模块,当遇到需要访问数据库的操作时,通过访问接口调用 SQL 引擎的编译或者执行模块.

在 MergeServer 上,需要封装存储过程访问 SQL 引擎的访问接口. 该接口需要提供如表 5 所示的基本访问功能.

表 5　OceanBase 的服务端编程接口需要的基本功能

Tab. 5　Basic functions needed in server programming interface of OceanBase

访问接口	功能
解析带参数的 SQL,生成执行计划	使存储过程能够缓存 SQL 解析的结果,并能以不同的参数进行调用
执行 SQL,返回能被存储过程访问的结果	提供诸如执行动态生成的 SQL 语句,计算表达式结果等功能
执行 SQL 计划	执行已经缓存编译结果的 SQL 语句. 减少 SQL 编译的次数
生成游标,访问游标,关闭游标	提供存储过程对执行结果的迭代访问

6　总　　结

存储过程是用户使用数据库的重要功能. 它大大改善了前台应用访问数据库的性能. 现有的主流数据库都支持这项功能. 存储过程的实现方案主要取决于数据库系统的设计和实现方案. 本文主要阐述了基于静态语言和基于动态语言的反射机制实现存储过程的基本原理. 并进一步以采用 C 开发的 PostgreSQL 和以 Java 开发的 VoltDB 为例,主要讨论了存储过程在不同系统中的实现原理. 最后,对 OceanBase 中实现存储过程这项功能给出了初步的设计.

(下转第页 319)