ПРОЕКТ PARGRESQL: РАЗРАБОТКА ПАРАЛЛЕЛЬНОЙ СУБД НА ОСНОВЕ СВОБОДНОЙ СУБД POSTGRESQL

К.С. Пан, М.Л. Цымблер

СУБД PostgreSQL [1] представляет собой свободно распространяемую реляционную СУБД с открытым исходным кодом, которая в настоящее время является одной из самых надежных и востребованных альтернатив коммерческим СУБД. Научный проект Омега [2] направлен на разработку прототипа параллельной СУБД для мультипроцессорных вычислительных систем с кластерной архитектурой.

В рамках проекта Омега в настоящее время ведется разработка параллельной СУБД, получившей название PargreSQL. Базовой идеей этой разработки является внедрение поддержки фрагментного параллелизма [3] в СУБД PostgreSQL. В данной работе описана архитектура и основные принципы разработки СУБД PargreSQL.

В основе архитектуры PostgreSQL лежит модель «клиент-сервер». В сеансе работы СУБД PostgreSQL участвуют три вида взаимодействующих процессов (см. рис. 1): приложение-клиент (frontend), серверный процесс (backend) и демон (postmaster). Демон осуществляет прием соединений, устанавливаемых клиентами, и создает отдельный серверный процесс для обработки запросов каждого отдельного клиента.

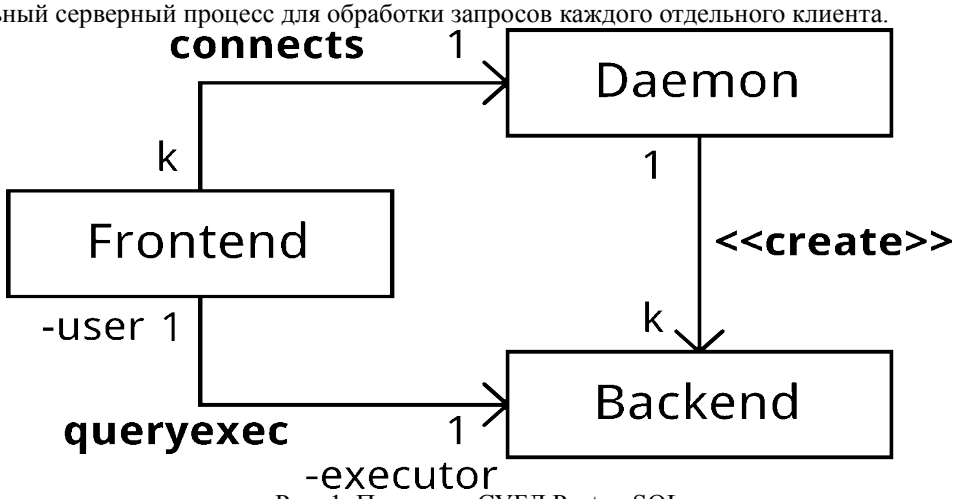

Рис. 1. Процессы СУБД PostgreSQL

Порядок взаимодействия клиента и СУБД представлен на рис. 2. Сначала клиент устанавливает соединение с демоном. Демон принимает соединение от клиента и затем с помощью системного вызова fork() создает серверный процесс. После этого клиент отправляет запрос серверному процессу, который выполняет обработку этого запроса и отправку результатов обратно клиенту.

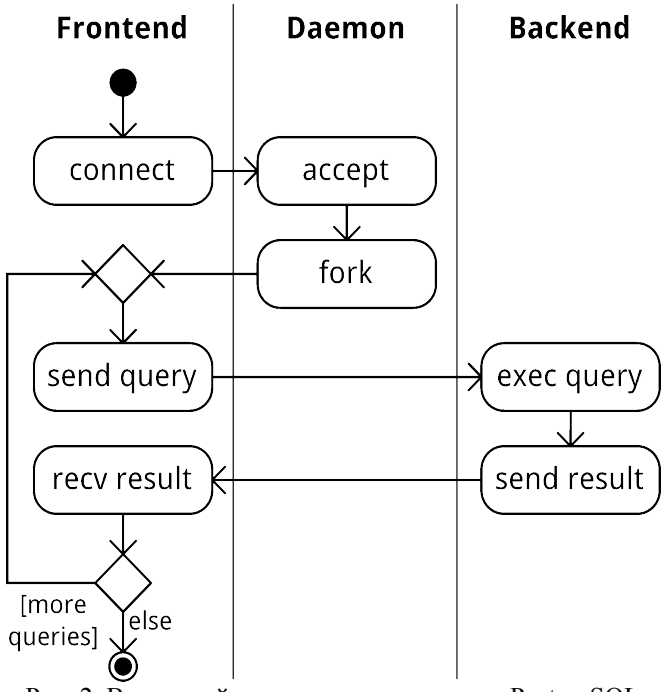

Рис. 2. Взаимодействие клиента и сервера PostgreSQL

Обработка запроса состоит из следующих этапов (см. рис. 3):

- **Parse** — разбор запроса на языке SQL;
- **Rewrite** — преобразование запроса;
- **Plan/Optimize** — составление плана запроса и его оптимизация;
- **Execute** — выполнение плана запроса.

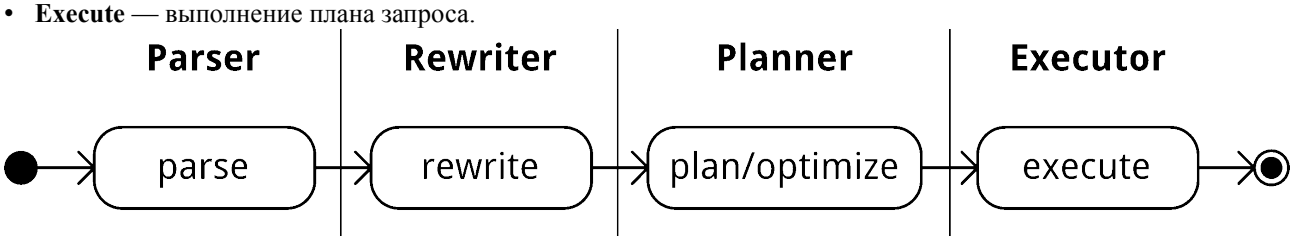

Рис. 3. Обработка запроса в СУБД PostgreSQL

СУБД PostgreSQL содержит следующие подсистемы, представленные на рис. 4:

- **Parser** — осуществляет разбор SQL-запросов;
- **Rewriter** — преобразует запрос по заданным в базе данных правилам, например, для реализации представлений;
- **Storage** — подсистема хранения данных и метаданных;
- **Planner** — составляет план запроса;
- **Executor** — исполняет план запроса;
- Библиотека **libpq** — реализует протокол взаимодействия клиента (libpq-fe) и сервера (libpq-be) PostgreSQL.

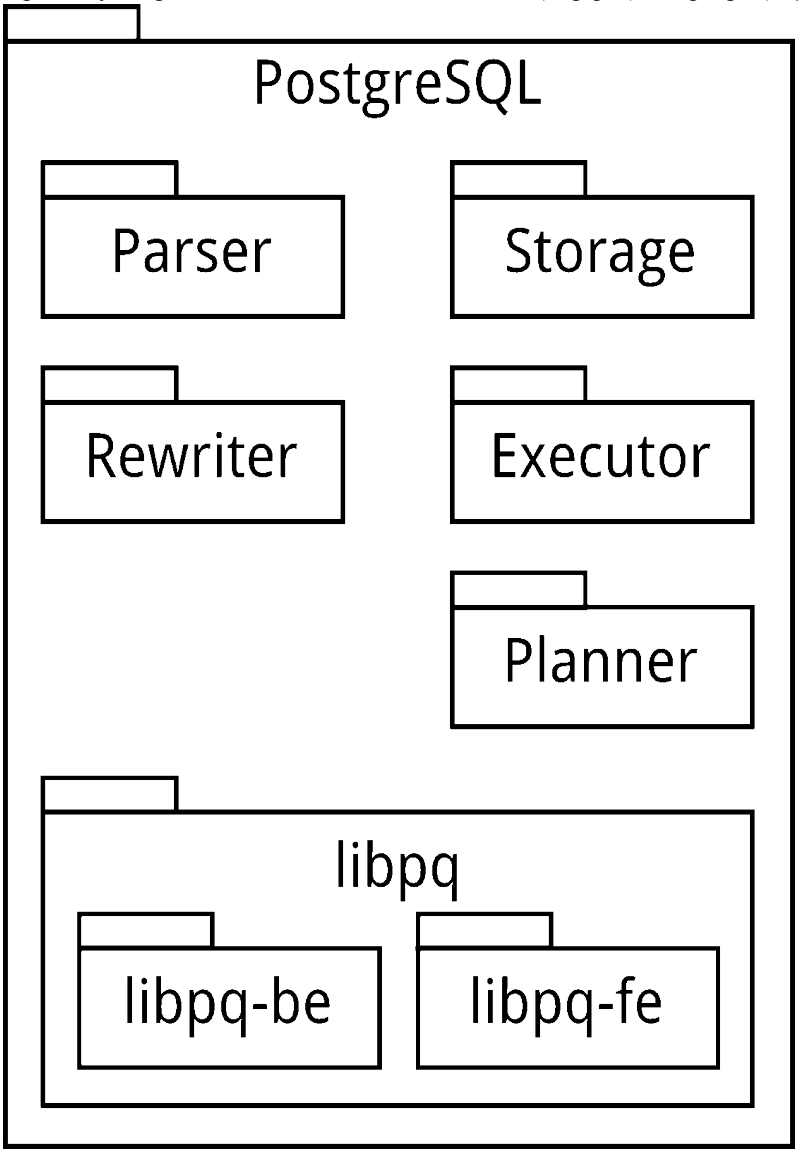

Рис. 4. Подсистемы СУБД PostgreSQL

PargreSQL использует идею фрагментного параллелизма [3]. Общая схема параллельной обработки запроса представлена на рис. 5. Каждое отношение (таблица) базы данных делится на горизонтальные **фрагменты**, распределяемые по процессорным узлам вычислительной системы. Способ фрагментации определяется **функцией фрагментации**, вычисляющей для каждого кортежа отношения номер процессорного узла, на котором должен быть размещен этот кортеж. Запрос выполняется в виде нескольких параллельных процессов (**агентов**), каждый из которых обрабатывает отдельный фрагмент отношения. Полученные фрагменты сливаются в результирующее отношение.

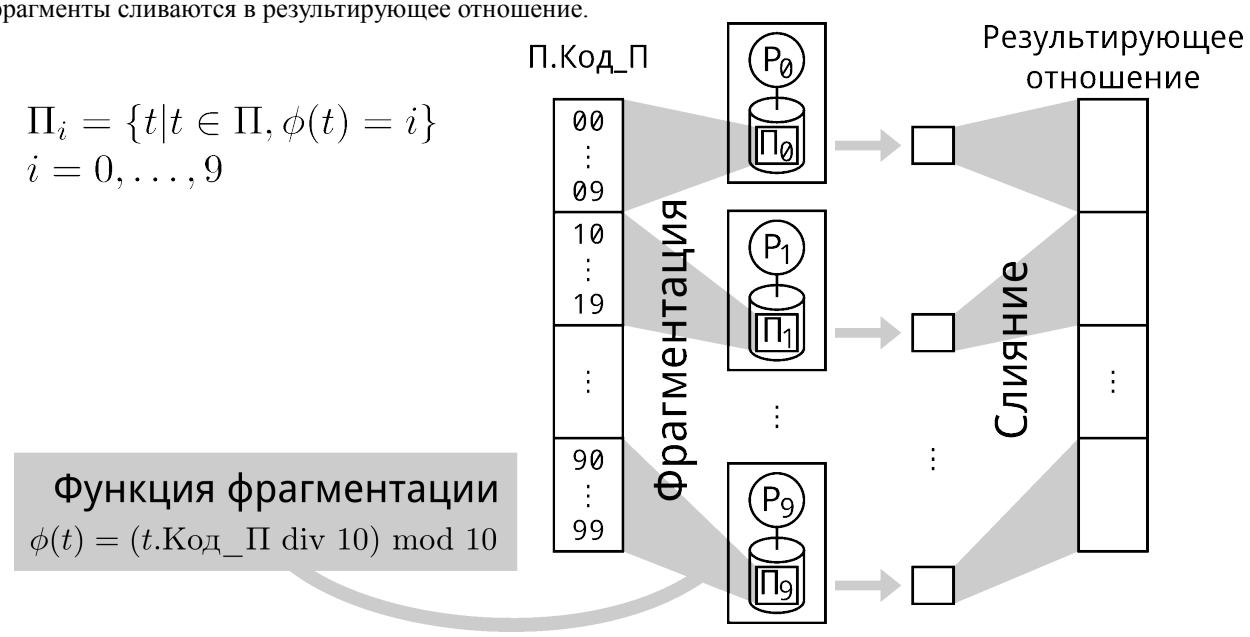

Рис. 5. Параллельная обработка запроса

Архитектура клиент-серверного взаимодействия в параллельной СУБД PargreSQL, в отличие от последовательной СУБД PostgreSQL, предполагает, что клиент взаимодействует сразу со многими серверами (см. рис. 6).

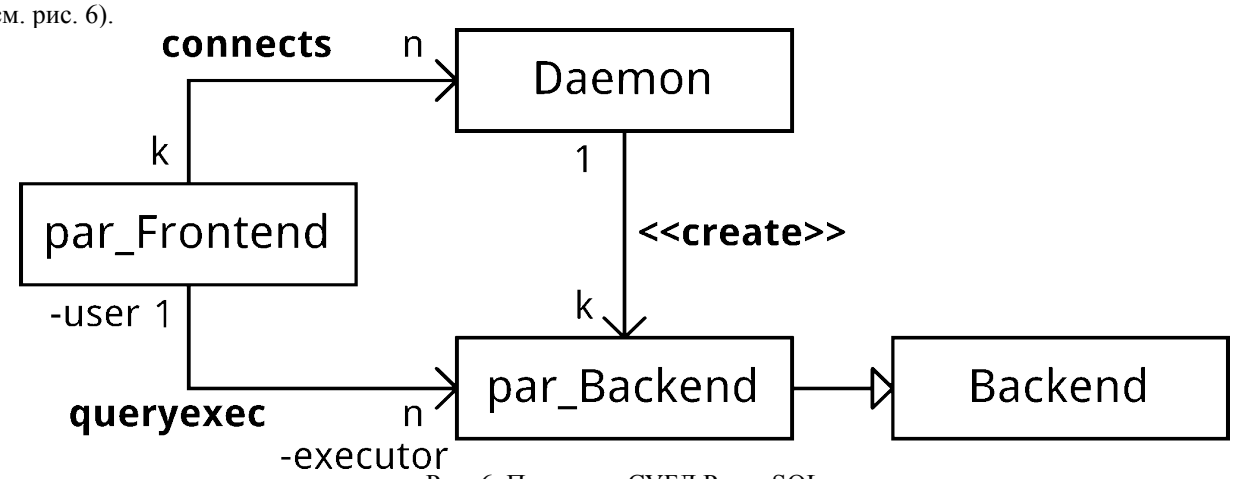

Рис. 6. Процессы СУБД PargreSQL

Порядок взаимодействия клиента и СУБД PargreSQL представлен на рис. 7. Клиентское приложение подключается сразу ко всем экземплярам СУБД, отправляет им одинаковый запрос. Подсистема составления плана запроса в каждом из серверных процессов составляет последовательный план, а подсистема параллелизации плана делает из него паралельный план путем вставки оператора exchange.

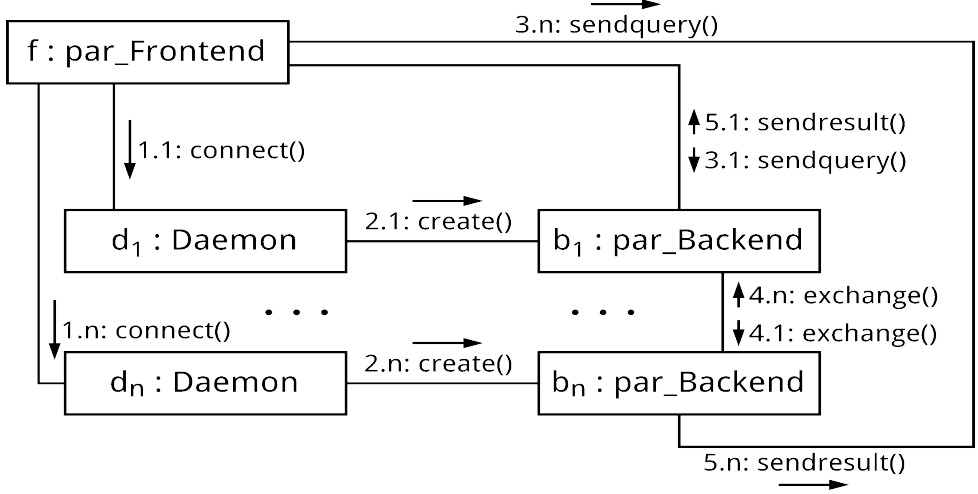

Рис. 7. Взаимодействие клиента и серверов PargreSQL

Параллельная обработка запроса состоит из следующих этапов (см. рис. 8):

- **Parse** — разбор запроса на языке SQL;
- **Rewrite** — преобразование запроса;
- **Plan/Optimize** — составление последовательного плана запроса и его оптимизация;
- **Parallelize** — составление параллельного плана запроса;
- **Execute** — выполнение плана запроса.

Рис. 8. Обработка запроса в СУБД PargreSQL

Архитектура PargreSQL представлена на рис. 9. PostgreSQL является подсистемой в рамках системы PargreSQL. Изменения коснулись подсистем Storage, Executor и Planner. Добавлены следующие подсистемы:

- **par_Storage** — подсистема хранения данных о фрагментации отношений;
- **par_Exchange** — реализация узла exchange, служащего для обмена кортежами между экземплярами СУБД;
- **par_Parallelizer** — параллелизатор плана запроса;
- **par_libpq-fe** — надстройка над libpq-fe, реализующая тиражирование запроса;
- **par_Compat** — подсистема, реализующая прозрачное для приложения подключение par_libpq-fe.

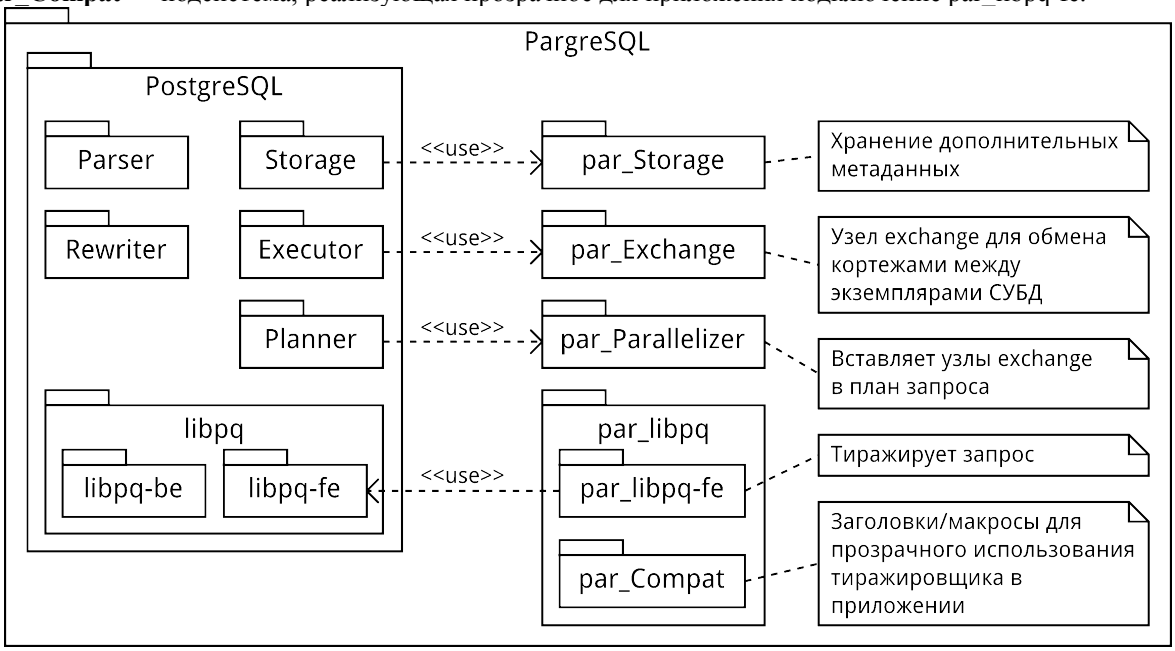

Рис. 9. Архитектура СУБД PargreSQL

Задача подсистемы par_Parallelizer заключается в добавлении операторов exchange [3] в нужные места последовательного плана выполнения запроса.

Оператор exhange обеспечивает пересылку кортежей между экземплярами СУБД. Задача оператора exchange — передать все кортежи, которые должны быть обработаны ядрами PargreSQL на других вычислительных узлах, и получить все кортежи, предназначенные для обработки ядром PargreSQL на данном узле.

PargreSQL разрабатывается в соответствии со следующими тремя основными принципами.

1. Параллельная СУБД PargreSQL, запущенная на одном вычислительном узле, работает так же, как последовательная СУБД PostgreSQL.
2. В исходные тексты PostgreSQL вносятся минимальные изменения. Изменения в структурах данных и алгоритмах инкапсулируются в новых файлах исходных текстов, подключаемых к исходным текстам PostgreSQL.
3. Использование PargreSQL прозрачно для пользовательских приложений. В прикладных программах, использующих PostgreSQL, подключение PargreSQL производится с минимальными изменениями в исходных кодах приложения.

Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (проект 09-0700241-а).

ЛИТЕРАТУРА:

1. Stonebraker M., Kemnitz G. The POSTGRES next-generation database management system // Communications of the ACM. Oct. 1991. Vol. 34, No. 10. P. 78-92.
2. Sokolinsky L., Axenov O., Gutova S. Omega: The Highly Parallel Database System Project // Proceedings of the First East-European Symposium on Advances in Database and Information Systems (ADBIS'97), St.-Petersburg, September 2-5, 1997. St.-Petersburg: Nevsky Dialect. 1997. Vol. 2. P. 88-90.
3. Соколинский Л.Б. Организация параллельного выполнения запросов в многопроцессорной машине баз данных с иерархической архитектурой // Программирование. 2001. № 6. С. 13-29.