Рекурсивные запросы в PosDB

Фирсов Михаил Александрович

Группа: 20.Б08-мм
Научный руководитель:
ассистент кафедры ИАС Чернышев Г.А.

Декабрь 2021

Постановка задачи

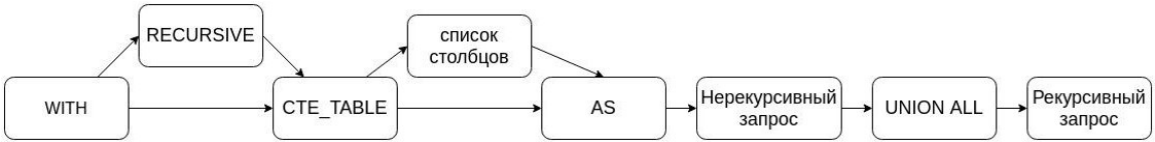
Целью работы является реализация поддержки рекурсивных запросов в РСУБД PosDB. Для её выполнения были поставлены следующие задачи:

1. Сделать обзор семантики рекурсивных запросов и её реализаций в других РСУБД.
2. Разработать и реализовать эффективный алгоритм обработки рекурсивных запросов с учётом архитектуры PosDB.
3. Написать тесты, проверяющие корректность и эффективность этого алгоритма.

Синтаксис и семантика рекурсивных запросов

```
WITH [RECURSIVE] <имя CTE запроса>
[(<список столбцов 0>) //может быть выведен
из запроса 1
AS (<запрос 1>) //обычный запрос
UNION ALL
<основной запрос 2> //рекурсивный запрос
```

Common Table Expression (CTE) — обобщенное табличное выражение, которое можно использовать множество раз в запросе



Зачем?

Рекурсивные запросы позволяют удобно работать с иерархически упорядоченными данными, хранящимися в таблицах. Можно, задавая условие на начальные данные и условие на подчинение одних строк таблицы другим, получить необходимую информацию в иерархически упорядоченном виде.

Пример

Есть следующая таблица. Требуется найти всех косвенных подчиненных начальника (работника с Master_ID = NULL) вплоть до работников с неявным подчинения через 3 промежуточных начальников.

```
WITH RECURSIVE person(Worker_ID, Master_ID)
AS SELECT Table1.Worker_ID, Table1.Master_ID FROM
Table1 WHERE Table1.Master_ID = NULL
UNION ALL
SELECT Table1.Worker_ID, Table1.Master_ID FROM person
JOIN Table1
ON Table1.Master_ID = person.Worker_ID
OPTION (MAXRECURSION 3);
```

Worker_ID	Master_ID
1	NULL
2	5
3	1
4	3
5	4
6	5
7	5

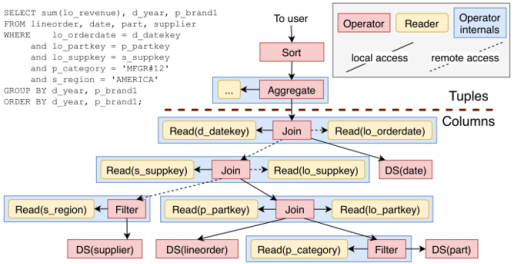
Устройство PosDB

PosDB — распределённая колоночная СУБД. Хранит каждый атрибут отдельно от других. Все запросы, проходя через парсер, переводятся в специальный план запроса.

План запроса — это диаграмма перехода табличных данных через специальные операторы. Каждый оператор в плане пропускает через себя поток данных.

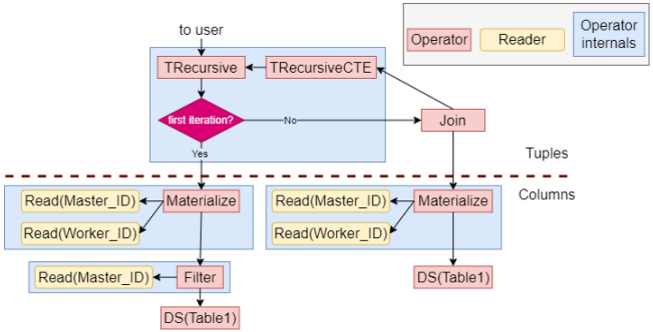
Операторы могут обрабатывать блоки кортежей или позиций. Позиционные операторы имеют специальные Reader-ы для чтения отдельных атрибутов таблицы.

Момент преобразования позиций в кортежи — материализация.



Реализация

Введем 2 оператора:
Recursive хранит в себе указатели на RecursiveCTE, ChildOperator и RecursiveChildOperator. ChildOperator используется для нерекурсивной части запроса, с его помощью получим стартовые строки или позиции. RecursiveChildOperator — обычный оператор, но внутри себя он получает данные от RecursiveCTE.
RecursiveCTE хранит указатель на Recursive, от которого просит новые строки (позиции) для их передачи методом Next() в RecursiveChildOperator.

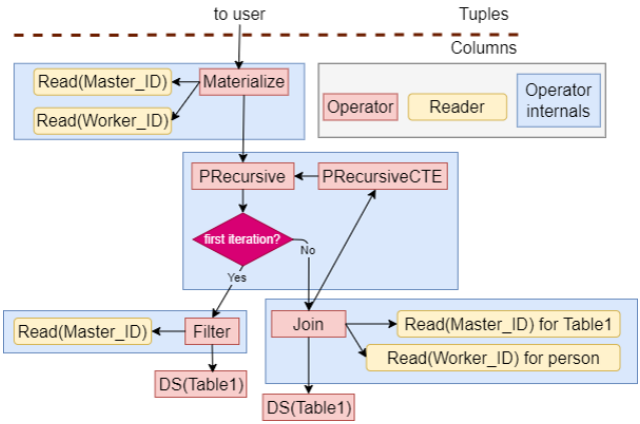


Детали

Для обработки кортежей введём операторы TRecursive и TRecursiveCTE, для обработки позиций — PRecursive и PRecursiveCTE.

XRecursive хранит все блоки данного шага рекурсии и по запросу передает их по одному в XRecursiveCTE. Для получения всех блоков очередной итерации XRecursive вызывает новый блок из RecursiveChildOperator до тех пор, пока не получит пустой блок.

Есть параметр maxRecursion — максимальное число рекурсивных итераций запроса.



Заключение. Что сделано?

1. Проанализированы семантика рекурсивных запросов в других СУБД: MS SQL Server, Oracle, PostgreSQL, MariaDB, SQLite. Написан обзор.
2. Реализованы операторы TRecursive, TRecursiveCTE, PRecursive, PRecursiveCTE.
3. Написан набор тестов, проверяющий корректность работы операторов, в том числе с использованием параметра maxRecursion и бесконечным циклом в рекурсивной части запроса.

Планы на будущее.

1. Написать набор тестов, проверяющий производительность рекурсивных операторов.
2. Подумать о “ненаивных” стратегиях для вычисления запросов такого типа.