**Task 21 (SQL vs PLpgSQL)**

Папка задания в репозитории:

<https://github.com/artemgur/AdvancedPostgreSQL/tree/master/Task%2021%20(SQL%20vs%20PLpgSQL)>

**Создание функций**

Функция SQL:

| CREATE OR REPLACE FUNCTION abc\_sql(x int) RETURNS int AS $$  SELECT b FROM abc WHERE a = x LIMIT 1; $$ LANGUAGE sql |
| --- |

Функция PLpgSQL:

| CREATE OR REPLACE FUNCTION abc\_plpgsql(x int) RETURNS int AS $$ BEGIN  RETURN (SELECT b FROM abc WHERE a = x LIMIT 1); END $$ LANGUAGE plpgsql |
| --- |

Таблица abc — таблица, которая использовалась в задании 16 (BTree Index Usage). Эта таблица имеет следующий вид:

| CREATE TABLE abc (  a integer,  b integer,  c integer ) |
| --- |

Для таблицы abc существуют BTree-индексы для каждой комбинации столбцов.

**PgBench**

В случае SQL с помощью команды:

| pgbench -U postgres -T 300 -nc 10 -f PgBenchSQL.sql |
| --- |

запускался скрипт:

| \set a random(1, 1000) SELECT abc\_sql(:a) |
| --- |

В случае PLpgSQL с помощью команды:

| pgbench -U postgres -T 300 -nc 10 -f PgBenchPLpgSQL.sql |
| --- |

запускался скрипт:

| \set a random(1, 1000) SELECT abc\_plpgsql(:a) |
| --- |

**Результаты**

SQL:

* 0.469 ms — latency average
* 21336.982954 — tps (including connections establishing)
* 21338.945532 — tps (excluding connections establishing)

PLpgSQL:

* 0.195 ms — latency average
* 51259.579938 — tps (including connections establishing)
* 51264.810862 — tps (excluding connections establishing)

Таким образом, функция PLpgSQL работает намного быстрее.

**Дальнейшее исследование**

Чтобы понять, почему PLpgSQL работает гораздо быстрее, я добавил к команде pgbench флаг -C, чтобы для каждой транзакции устанавливалось новое подключение. Также для этого бенчмарка я уменьшил количество клиентов до 1.

Были получены следующие результаты:

SQL:

* 4.351 ms — latency average
* 229.822989 — tps (including connections establishing)
* 596.814583 — tps (excluding connections establishing)

PLpgSQL:

* 5.584 ms — latency average
* 179.072750— tps (including connections establishing)
* 350.512620 — tps (excluding connections establishing)

В этом случае SQL-функция работает быстрее.

**Выводы**

В документации PostgreSQL указано, что план выполнения функций PLpgSQL кэшируется:

<https://postgrespro.ru/docs/postgresql/14/plpgsql-implementation#PLPGSQL-PLAN-CACHING>

Проведенный выше бенчмарк подтверждает это.

Таким образом, при первом выполнении в сессии функции PLpgSQL работают медленнее, в том числе из-за затрат на кэширование. При последующих выполнениях функции PLpgSQL выполняются гораздо быстрее, так как план выполнения уже составлен.

При выборе языка программирования для функции нужно учитывать, как эта функция будет использоваться. Если функция будет использоваться 1 раз в сессию, то при прочих равных стоит использовать SQL. Если функция будет выполняться несколько раз за сессию, стоит использовать PLpgSQL.