

**Занятие 9. Курсоры**

**Практическое задание**

Практическое задание сдается в виде отчета, в котором для каждого задания, подразумевающего практическую работу с экземпляром Postgres, приводятся снимки экрана, на которых видны фрагменты соответствующих экранных форм, комментированные исходные коды запросов, а также результатов их выполнения на экземпляре. Также сдаётся текстовый файл \*.sql, в который включены комментированные исходные коды всех запросов по всему заданию. В случае, если задание требует использования возможности, поддержка которой в Postgres (или используемой версии Postgres) отсутствует, то необходимо отразить это в отчете и использовать альтернативный путь решения задачи.

В качестве «полигона» используйте таблицы (если необходимо), созданные в рамках подготовки ДЗ с одного из предыдущих занятий.

Курсор- это структура, содержащая набор строк и позволяющая по ним итерироваться.

КУРСОР - это указатель (хотя как такового, понятия "указатель" в **PL/SQL** нет!) на контекстную область памяти, с помощью которого программа на языке **PL/SQL** может управлять контекстной областью и ее состоянием во время обработки оператора.

Объявление курсора определяет какое выражение языка **SQL** - будет передано программе **SQL Statement Executor** (системе исполнителю выражения **SQL**). Курсор может представлять собой любое допустимое предложение языка **SQL**! Так же, курсор является основным базовым "кирпичиком" для построения блоков **PL/SQL**. Курсоры обеспечивают циклический механизм оперирования наборами данных в БД. Курсор может возвращать одну или несколько строк данных или вообще ни одной.

**Например, если запрос возвращает большой набор данных, а вы хотите обработать его постранично или выбрать только определенные строки, вы можете использовать курсор. Курсор позволяет вам выбирать и обрабатывать только те данные, которые вам нужны в данный момент, что может быть полезно для оптимизации производительности и уменьшения нагрузки на сервер.**

Курсоры очень дорогая конструкция, ресурсозатратная. И их нужно применять **осторожно**. Однако есть случаи, когда их применение оправдано. Эти случаи связаны при работе с иерархическими структурами данных в обычном реляционной SQL базе данных.

1. Создайте параметризированный курсор с вложенным курсорным выражением. Организуйте итеративную обработку записей курсора и вложенного курсорного выражения при помощи вложенных циклов.

Для этого нужно сначала создать или изменить существующую функцию, которая нам будет выводить строку.

Эта функция будет принимать число (цена на услугу) и внутри добавлять к строке название услуги, у которого функция принимает цену.

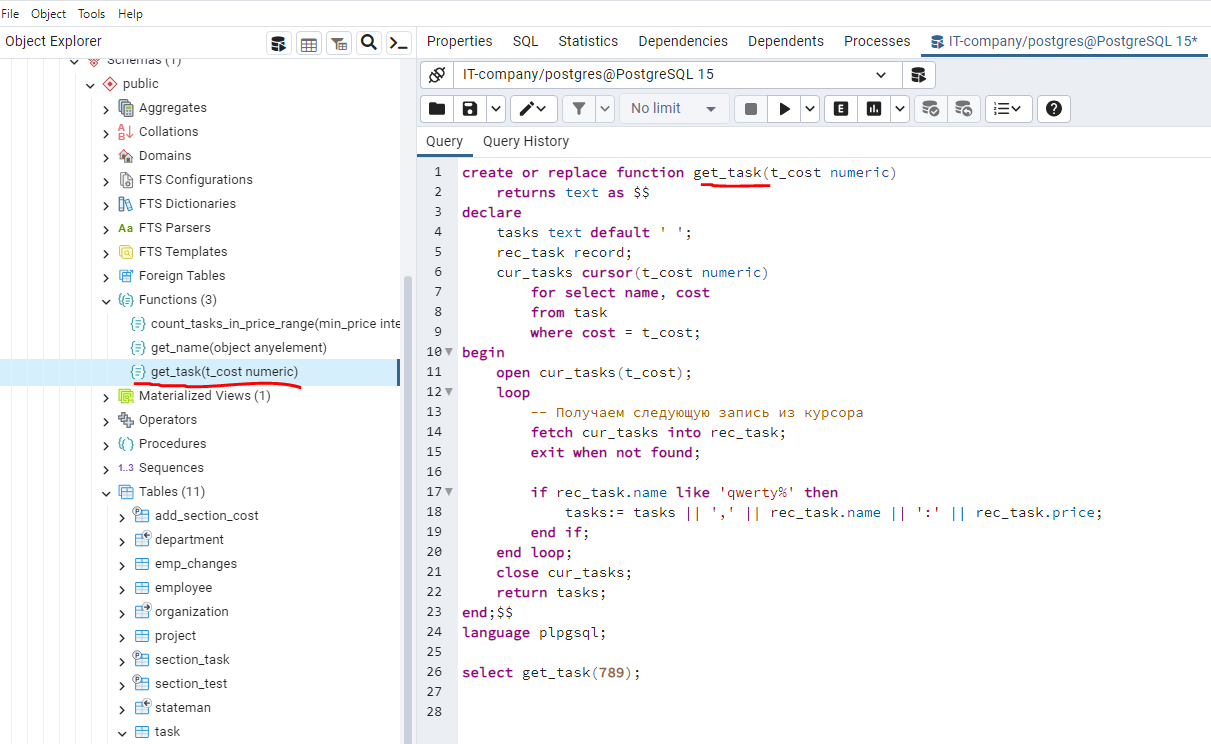
В DECLARE объявили саму строку, record (для прохода по таблице) и курсор.

В основной части открываем курсор и начинаем цикл.

Проходимся по таблице, ищем подходящие строки, и добавляем их нужные значения к строке.

Закрываем курсор и выводим значение с заданным условием, какие именно услуги должны выводиться, на экран.

Проверяем работоспособность с помощью SELECT:



https://lh7-us.googleusercontent.com/CpH1GKuvlpPfXJPfSM0WdYSKrNVC1hCUeo3WWZH8d-Lb2vThI-AoG3Fdelurv5ZlWh9fd5CdfkBd-j-pave91JRy-SjxGRmjDukyo-pVqG6R2QNek_jKzu25QwQn6UdrWCYG4t1uYapn

https://lh7-us.googleusercontent.com/ymEwaheY0wBkkmMn4TeVUj5l0WyZfXz-OD-izZM73Am_hqAx_3TVxUisF04UkwRW9eBVnMlBf_CRuX6gKESfTErJV8z8Xxifbzz4q_WtfN6UUFhG8oin-XqUXynUuVZkwhBn1qS0RI8t

Код создает функцию get\_task, которая принимает числовой параметр с именем t\_cost и возвращает текстовое значение. Функция сначала объявляет три переменные:

задачи: текстовая переменная, инициализируемая пустой строкой. Эта переменная будет использоваться для хранения названий и стоимости задач, соответствующих указанным критериям.

Rec\_task: переменная записи, которая будет использоваться для хранения каждой строки данных, извлекаемых из курсора.

cur\_tasks: курсор, который будет использоваться для выборки строк из таблицы задач, стоимость которых равна значению параметра t\_cost.

Затем функция открывает курсор и входит в цикл, который извлекает строки из курсора по одной. Для каждой извлекаемой строки функция проверяет, соответствует ли столбец имени строки регулярному выражению «qwerty%». Если имя соответствует регулярному выражению, функция добавляет имя и стоимость задачи к переменной задач

Наконец, функция закрывает курсор и возвращает значение переменной Tasks.

Оператор select, используемый в конце фрагмента кода, вызывает функцию get\_task с параметром t\_cost, равным 789. Это вернет имена и стоимость всех задач в таблице задач, стоимость которых равна 789 и имя которых начинается с строка qwerty.

1. Сделайте так, чтобы записи курсора разместились в коллекции TABLE OF. Рассмотрите вариант с построчным внесением записей в коллекцию, а также вариант массовой вставки BULK COLLECT INTO. Подумайте и напишите, чем предпочтительнее тот или иной вариант в разных ситуациях.

You can create a temporary table in your PL/PgSQL code and use that for the desired purpose. Temp tables in PL/PgSQL are a little bit annoying because the names are global within the session, but they work correctly in PostgreSQL 8.4 and up.

A better alternative for when you're doing all the work within a single SQL statement is to use a common table expression (CTE, or WITH query). This won't be suitable for all situations.

1. Немного отдохните. Это сложная и интересная тема. Вы проделали большую работу ☺ Но мы, как обычно, затронули только лишь верхушку.