МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ  
 РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Чувашский государственный университет имени И.Н. Ульянова»

Факультет информатики и вычислительной техники

Кафедра математического и аппаратного обеспечения   
 информационных систем

Лабораторная работа №5 по дисциплине “Безопасность систем баз данных”

Тема: **Триггеры**

Выполнил: студент гр. ИВТ 21-22

Иванов Н.А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил Старший преподаватель

Марков А.В. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Цель работы:

Изучить способы создания триггеров в БД PostgresSQL, создать триггеры согласно заданию.

Содержание:

Триггер является указанием, что база данных должна автоматически выполнить заданную функцию, всякий раз когда выполнен определённый тип операции. Триггеры можно использовать с таблицами (секционированными и обычными), с представлениями и с внешними таблицами.

Для обычных и сторонних таблиц можно определять триггеры, которые будут срабатывать до или после любой из команд INSERT, UPDATE или DELETE; либо один раз для каждой модифицируемой строки, либо один раз для оператора SQL. Триггеры на UPDATE можно установить так, чтобы они срабатывали, только когда в предложении SET оператора UPDATE упоминаются определённые столбцы. Также триггеры могут срабатывать для операторов TRUNCATE. Если происходит событие триггера, для обработки этого события в установленный момент времени вызывается функция триггера.

Для представлений триггеры могут быть определены для выполнения вместо операций INSERT, UPDATE и DELETE. Такие триггеры INSTEAD OF вызываются единожды для каждой строки, которая должна быть изменена в этом представлении. Именно функция триггера отвечает за то, чтобы произвести необходимые изменения в нижележащих базовых таблицах представления и должным образом возвращать изменённые строки, чтобы они появлялись в представлении. Триггеры для представлений тоже могут быть определены так, что они будут выполняться единожды для всего оператора SQL, до или после операций INSERT, UPDATE или DELETE. Однако такие триггеры срабатывают, только если для представления определён триггер INSTEAD OF. В противном случае все операторы, обращающиеся к представлению, должны быть переписаны в виде операторов, обращающихся к нижележащим базовым таблицам, и тогда будут срабатывать триггеры, установленные для этих таблиц.

Триггерная функция должна быть создана до триггера. Она должна быть объявлена без аргументов и возвращать тип trigger. (Триггерная функция получает данные на вход посредством специально переданной структуры TriggerData, а не в форме обычных аргументов.)

Триггер при изменении данных объявляется как функция без аргументов и с типом результата trigger. Заметьте, что эта функция должна объявляться без аргументов, даже если ожидается, что она будет получать аргументы, заданные в команде CREATE TRIGGER — такие аргументы передаются через TG\_ARGV, как описано ниже.

Когда функция на PL/pgSQL срабатывает как триггер, в блоке верхнего уровня автоматически создаются несколько специальных переменных:

NEW

Тип данных RECORD. Переменная содержит новую строку базы данных для команд INSERT/UPDATE в триггерах уровня строки. В триггерах уровня оператора и для команды DELETE эта переменная имеет значение null.

OLD

Тип данных RECORD. Переменная содержит старую строку базы данных для команд UPDATE/DELETE в триггерах уровня строки. В триггерах уровня оператора и для команды INSERT эта переменная имеет значение null.

TG\_NAME

Тип данных name. Переменная содержит имя сработавшего триггера.

TG\_WHEN

Тип данных text. Строка, содержащая BEFORE, AFTER или INSTEAD OF, в зависимости от определения триггера.

TG\_OP

Тип данных text. Строка, содержащая INSERT, UPDATE, DELETE или TRUNCATE, в зависимости от того, для какой операции сработал триггер.

Другие параметры доступны по ссылкам в документации ниже.

После создания триггерной функции создаётся триггер с помощью CREATE TRIGGER. Одна и та же триггерная функция может быть использована для нескольких триггеров.

PostgreSQL предлагает как построчные, так и операторные триггеры. В случае построчного триггера триггерная функция вызывается один раз для каждой строки, затронутой оператором, запустившим триггер. Операторный же триггер, напротив, вызывается только один раз при выполнении соответствующего оператора, независимо от количества строк, которые он затрагивает. В частности оператор, который не затрагивает никаких строк, всё равно приведёт к срабатыванию операторного триггера. Эти два типа триггеров также называют триггерами уровня строк и триггерами уровня оператора, соответственно. Триггеры на TRUNCATE могут быть определены только на уровне оператора, а не на уровне строк.

Триггеры также классифицируются в соответствии с тем, срабатывают ли они до, после или вместо операции. Они называются триггерами BEFORE, AFTER и INSTEAD OF, соответственно. Триггеры BEFORE уровня оператора срабатывают до того, как оператор начинает делать что-либо, тогда как триггеры AFTER уровня оператора срабатывают в самом конце работы оператора. Эти типы триггеров могут быть определены для таблиц, представлений или сторонних таблиц. Триггеры BEFORE уровня строки срабатывают непосредственно перед обработкой конкретной строки, в то время как триггеры AFTER уровня строки срабатывают в конце работы всего оператора (но до любого из триггеров AFTER уровня оператора). Эти типы триггеров могут определяться только для таблиц, в том числе сторонних, но не для представлений. Триггеры INSTEAD OF могут определяться только для представлений и только на уровне строк: они срабатывают для каждой строки сразу после того как строка представления идентифицирована как подлежащая обработке.

CREATE TRIGGER создаёт новый триггер, а CREATE OR REPLACE TRIGGER создаёт новый триггер или заменяет существующий. Триггер будет связан с указанной таблицей, представлением или сторонней таблицей и будет выполнять заданную функцию при определённых операциях с этой таблицей.

Для удаления триггера применяется команда DROP TRIGGER.

В postgres предпочтитетельнее создавать именно триггеры на pl/pgSQL, пример в приложенных к работе файлах.

Более полная информация по работе с триггерами доступна <https://postgrespro.ru/docs/postgresql/15/trigger-definition>,

<https://postgrespro.ru/docs/postgresql/15/sql-createtrigger>, <https://postgrespro.ru/docs/postgresql/15/plpgsql-trigger>

**Задание:**

1. Создать, запустить и проверить работу скрипта create\_triggers.sql, который создаёт следующие триггеры: - при добавлении/удалении записи в таблице payments, меняет поле баланс в таблице accounts на значение суммы для счета плательщика и счёта получателя в отрицательную и положительную сторону соответственно(исправить триггер, приведённый в приложенных файлах);   
   - при добавлении записи в таблицу payments, создаёт такую же запись в таблице archive\_payments;  
   - при изменении записи в таблице payments, меняет поле баланс в таблице accounts на значение суммы для счета плательщика и счёта получателя в отрицательную и положительную сторону соответственно (измениться может не только сумма, но и плательщик и получатель, в таком случае надо пересчитать все участвующие записи их accounts);  
   - при изменении записи в таблице payments, создаёт такую же запись в таблице archive\_payments;  
   - при удалении записи в таблице pages удаляет все дочерние записи;
2. Создать и запустить скрипт check\_triggers.sql, который выполняет запросы к БД, демонстрирующие работоспособность и правильность триггеров из п.1.
3. Создать и запустить скрипт drop\_triggers.sql, который удаляет триггеры из п.1.

Результат:

**<https://github.com/bwl-andromeda/db/tree/lab-5>**

**Вывод:**

В ходе лабораторной работы были изучены методы создания и использования триггеров в базе данных PostgreSQL. Были рассмотрены типы триггеров, включая триггеры уровня строки и оператора, а также различия между триггерами BEFORE, AFTER и INSTEAD OF. Также изучены особенности работы с триггерами в PL/pgSQL и порядок создания триггерных функций.

В рамках задания был разработан скрипт create\_triggers.sql, который создаёт несколько триггеров для работы с таблицами платежей, архивных платежей и таблицей аккаунтов. Триггеры реализуют автоматическое обновление балансов счётов при добавлении, изменении и удалении записей в таблице платежей, а также синхронизацию с архивными таблицами. Особое внимание было уделено корректному пересчёту балансов при изменении записей, что позволяет сохранить целостность данных в базе.

В дополнение был создан скрипт check\_triggers.sql для проверки работоспособности триггеров и скрипт drop\_triggers.sql для удаления триггеров после выполнения всех проверок. Проверка продемонстрировала корректную работу всех триггеров согласно требованиям задания.

Таким образом, цель работы была достигнута — были получены практические навыки по созданию триггеров в PostgreSQL, а также понимание их использования для автоматизации процессов обновления данных и поддержания целостности данных в таблицах базы данных.