Лабораторная работа №8
Перегружаемые функции, триггеры в PostgreSQL

Цель работы: Изучение перегружаемых функций и триггеров в PostgreSQL. Изучить синтаксис команд. Получение навыков работы с триггерами.

Задание. Общая часть: Ознакомиться с теоретическими сведениями о возможностях создания перегружаемых функций и триггеров в PostgreSQL. Создать перегружаемые функции и триггеры. Продемонстрировать работу триггеров на примерах вставки и удаления записей из таблицы. Если в базе нет подходящих данных, то добавить подходящие данные. Удалить триггер. По заданию преподавателя создать триггер и проверить его работоспособность. Просмотреть и проанализировать полученную в результате выполнения операций информацию.

Вариант 1: Триггер выполняется перед удалением записи из таблицы хоккейных команд. Триггер проверяет наличие в другой таблице записей, относящихся к удаляемой команде, и, если такие записи есть, удаляет их.

Триггер выполняется перед вставкой новой записи в таблицу хоккейных команд. Триггер проверяет значения, которые должна содержать новая запись и может их изменить:

- если не указано имя команды – оно генерируется по схеме – Komanda + уникальный номер из последовательности;
- если не указан город – ставится значение по умолчанию – “Omsk” ;
- если не указан рейтинг или рейтинг <=10 – устанавливается рейтинг, равный 10 для команд из города “Omsk” и 0 для всех остальных.

Вариант 2: Триггер выполняется перед удалением записи из таблицы клиента продуктового магазина. Триггер проверяет наличие в другой таблице записей, относящихся к удаляемому клиенту, и, если такие записи есть, удаляет их.

Триггер выполняется перед вставкой новой записи в таблицу клиентов продуктового магазина. Триггер проверяет значения, которые должна содержать новая запись и может их изменить:

- если не указано имя клиента – оно генерируется по схеме – Klient + уникальный номер из последовательности;
- если не указан город клиента– ставится значение по умолчанию – “Moskva” ;
- если не указан объем заказа или объем заказа <=10 – устанавливается объем заказа, равный 10 для клиентов из города “Moskva” и 0 для всех остальных.

Вариант 3: Триггер выполняется перед удалением записи из таблицы водителя. Триггер проверяет наличие в другой таблице записей, относящихся к удаляемому водителю, и, если такие записи есть, удаляет их.

Триггер выполняется перед вставкой новой записи в таблицу водителя. Триггер проверяет значения, которые должна содержать новая запись и может их изменить:

- если не указано имя водителя – оно генерируется по схеме – Voditel+ уникальный номер из последовательности;
- если не указан город водителя – ставится значение по умолчанию – “Peterburg” ;
- если не указан стаж водителя или стаж <=3 – устанавливается стаж, равный 3 для водителя из города “Peterburg” и 0 для всех остальных.

Вариант 4: Триггер выполняется перед удалением записи из таблицы автора. Триггер проверяет наличие в другой таблице записей, относящихся к удаляемому автору, и, если такие записи есть, удаляет их.

Триггер выполняется перед вставкой новой записи в таблицу авторов. Триггер проверяет значения, которые должна содержать новая запись и может их изменить:

- если не указано имя автора – оно генерируется по схеме – Avtor + уникальный номер из последовательности;
- если не указан город автора – ставится значение по умолчанию – “Samara” ;
- если не указан тираж издания или тираж <=50 – устанавливается тираж, равный 50 для автора из города “Samara” и 0 для всех остальных.

Вариант 5: Триггер выполняется перед удалением записи из таблицы производителей машин. Триггер проверяет наличие в другой таблице записей, относящихся к удаляемому производителю, и, если такие записи есть, удаляет их.

Триггер выполняется перед вставкой новой записи в таблицу производителей машин. Триггер проверяет значения, которые должна содержать новая запись и может их изменить:

- если не указано имя производителя – оно генерируется по схеме – Proizvoditel + уникальный номер из последовательности;
- если не указан город производителя– ставится значение по умолчанию – “Tokio” ;
- если не указан объем двигателя – устанавливается объем двигателя, равный 1300 $см^3$ для производителей из Tokio и 0 для всех остальных.

Вариант 6: Триггер выполняется перед удалением записи из таблицы поставщиков фруктов. Триггер проверяет наличие в другой таблице записей, относящихся к удаляемому поставщику, и, если такие записи есть, удаляет их.

Триггер выполняется перед вставкой новой записи в таблицу поставщиков фруктов. Триггер проверяет значения, которые должна содержать новая запись и может их изменить:

- если не указано имя поставщика – оно генерируется по схеме – Postavshik + уникальный номер из последовательности;
- если не указан город поставщика– ставится значение по умолчанию – “Krasnodar” ;
- если не указан объем поставок или объем поставок <=10 кг – устанавливается объем поставок, равный 10 для поставщика из города “ Krasnodar” и 0 для всех остальных.

Вариант 7: Триггер выполняется перед удалением записи из таблицы печатной продукции. Триггер проверяет наличие в другой таблице записей, относящихся к удаляемой печатной продукции, и, если такие записи есть, удаляет их.

Триггер выполняется перед вставкой новой записи в таблицу печатной продукции. Триггер проверяет значения, которые должна содержать новая запись и может их изменить:

- если не указано название печатной продукции – оно генерируется по схеме – Kniga + уникальный номер из последовательности;
- если не указан город заказчика – ставится значение по умолчанию – “Barnaul” ;
- если не указан объем заказа или объем заказа <=50 экземпляров – устанавливается объем заказа, равный 50 для печатной продукции из города “ Barnaul” и 0 для всех остальных.

Вариант 8: Триггер выполняется перед удалением записи из таблицы данных о ремонте. Триггер проверяет наличие в другой таблице записей, относящихся к удаляемой записи, и, если такие записи есть, удаляет их.

Триггер выполняется перед вставкой новой записи в таблицу данных о ремонте. Триггер проверяет значения, которые должна содержать новая запись и может их изменить:

- если не указано наименование ремонта – оно генерируется по схеме – Remont + уникальный номер из последовательности;
- если не указан город у мастера – ставится значение по умолчанию – “Kemerovo” ;
- если не указана длительность ремонта или длительность ремонта <=30 дней – устанавливается длительность ремонта, равная 30 для мастера из города “Kemerovo” и 0 для всех остальных.

Вариант 9: Триггер выполняется перед удалением записи из таблицы мастеров. Триггер проверяет наличие в другой таблице записей, относящихся к удаляемому мастеру, и, если такие записи есть, удаляет их.

Триггер выполняется перед вставкой новой записи в таблицу мастеров. Триггер проверяет значения, которые должна содержать новая запись и может их изменить:

- если не указано имя мастера – оно генерируется по схеме – Master + уникальный номер из последовательности;
- если не указан город мастера– ставится значение по умолчанию – “Irkutsk” ;
- если не указан стаж мастера или стаж <=5 – устанавливается стаж, равный 5 для мастера из города “Irkutsk” и 0 для всех остальных.

Вариант 10: Триггер выполняется перед удалением записи из таблицы данных о туре. Триггер проверяет наличие в другой таблице записей, относящихся к удаляемой записи, и, если такие записи есть, удаляет их.

Триггер выполняется перед вставкой новой записи в таблицу данных о туре. Триггер проверяет значения, которые должна содержать новая запись и может их изменить:

- если не указано наименование тура – оно генерируется по схеме – Tur + уникальный номер из последовательности;
- если не указан город у тура – ставится значение по умолчанию – “Kaliningrad” ;
- если не указана длительность тура или длительность тура <=7 дней – устанавливается длительность тура, равная 7 для туров в город “Kaliningrad” и 0 для всех остальных.

Содержание отчета

Отчет должен содержать титульный лист, цель работы, задание, коды команд создания триггеров, результаты работы триггеров, демонстрация правильности работы триггеров, тексты запросов и их результаты, выводы и анализ результатов работы.

Контрольные вопросы

1. Что такое триггер, для чего он нужен?
2. Синтаксис определения триггера.
3. Дайте краткое описание компонентов определения триггеров.
4. С помощью какой команды можно удалить триггер?

5. Для чего нужны перегружаемые функции?
6. Что означает ключевое слово BEFORE в определении триггера?
7. Что означает ключевое слово AFTER в определении триггера?
8. Каким образом обращаются к отдельным полям записей NEW и OLD в триггерных процедурах?
9. Что такое триггерная функция?
10. Синтаксис определения триггерной функции.
11. Каким образом можно проверить правильность работы триггера?

7.1. Использование перегружаемых функций в СУБД PostgreSQL

PostgreSQL позволяет создавать *перегружаемые функции.* Под этим понимается определение нескольких функций с одинаковыми именами при условии, что каждая из них получает уникальный набор аргументов.

Пример. Создание перегружаемой функции.

```
CREATE FUNCTION n11 (integer,integer) RETURNS integer AS '
BEGIN
if $1 is null then return $2;
else return $1;
end if;
END;
' LANGUAGE plpgsql;


CREATE FUNCTION n11 (character, character) RETURNS character AS '
BEGIN
if $1 is null then return $2;
else return $1;
end if;
END;
' LANGUAGE plpgsql;


CREATE FUNCTION n11 (text,text) RETURNS text AS '
BEGIN
if $1 is null then return $2;
else return $1;
end if;
END;
' LANGUAGE plpgsql;


CREATE FUNCTION n11 (real,real) RETURNS text AS '
BEGIN
if $1 is null then return $2;
else return $1;
end if;
END;
' LANGUAGE plpgsql;
```

Синтаксис команды

CREATE [OR REPLACE] **FUNCTION**
имя_функции **([[** *метод_аргумента* **][***имя_аргумента* **]** *тип_ аргумента* **[,…]])**
RETURNS *тип_возвращаемого_значения* **AS** '*определение*'
LANGUAGE '*язык*'

- **CREATE FUNCTION** *имя_функции* ([[*метод_аргумента*] [*имя_аргумента*] *тип_ аргумента* [,…]]) - после ключевых слов CREATE FUNCTION указывается имя создаваемой функции, после чего в круглых скобках перечисляются аргументы, разделенные запятыми. Для каждого аргумента достаточно указать только тип, но при желании можно задать метод (in, out, inout; по умолчанию in) и имя. Если список в круглых скобках пуст, функция вызывается без аргументов (хотя сами круглые скобки обязательно должны присутствовать как в определении функции, так и при ее использовании). Ключевые слова OR REPLACE используются для изменения уже существующей функции.

- **RETURNS** *тип_возвращаемого_значения* - тип данных, возвращаемый функцией.

- **AS** '*определение*' - программное определение функции. В процедурных языках (таких, как PL/pgSQL) оно состоит из кода функции. Для откомпилированных функций C указывается абсолютный системный путь к файлу, содержащему объектный код.

- **LANGUAGE** '*язык*'. Название языка, на котором написана функция.

7.2. Использование триггеров в СУБД PostgreSQL

Триггер определяет операцию, которая должна выполняться при наступлении некоторого события в базе данных. Триггеры срабатывают при выполнении с таблицей команды SQL INSERT, UPDATE или DELETE.
В PostgreSQL триггеры создаются на основе существующих функции, т.е. сначала командой CREATE FUNCTION определяется триггерная функция, затем на ее основе командой CREATE TRIGGER определяется собственно триггер.

Синтаксис определения триггера
CREATE TRIGGER ***триггер***
BEFORE | AFTER } { ***событие*** [OR ***событие***] } ON ***таблица***
FOR EACH { ROW | STATEMENT }
EXECUTE PROCEDURE ***функция*** (***аргументы***)

Ниже приводятся краткие описания компонентов этого определения.

- **CREATE TRIGGER *триггер***. В аргументе *триггер* указывается произвольное имя создаваемого триггера. Имя может совпадать с именем триггера, уже существующего в базе данных при условии, что этот триггер установлен для другой таблицы. Кроме того, по аналогии с большинством других несистемных объектов баз данных, имя триггера (в сочетании с таблицей, для которой он устанавливается) должно быть уникальным лишь в контексте базы данных, в которой он создается
- **{ BEFORE | AFTER }.** Ключевое слово BEFORE означает, что функция должна выполняться *перед* попыткой выполнения операции, включая все встроенные проверки ограничений данных, реализуемые при выполнении команд INSERT и DELETE. Ключевое слово AFTER означает, что функция вызывается после завершения операции, приводящей в действие триггер.
- **{** *событие* **[OR** *событие ...* **] }.** События SQL, поддерживаемые в PostgreSQL: INSERT, UPDATE или DELETE. При перечислении нескольких

событий в качестве разделителя используется ключевое слово OR.

- **ON *таблица.*** Имя таблицы, модификация которой заданным событием приводит к срабатыванию триггера.
- **FOR EACH { ROW | STATEMENT }.** Ключевое слово, следующее за конструкцией FOR EACH и определяющее количество вызовов функции при наступлении указанного события. Ключевое слово ROW означает, что функция вызывается *для каждой модифицируемой записи.* Если функция должна вызываться всего один раз для всей команды, используется ключевое слово STATEMENT.
- **EXECUTE PROCEDURE** *функция* **(** *аргументы* **).** Имя вызываемой функции с аргументами.

Триггер удаляется с помощью команды

DROP TRIGGER имя_триггера ON имя_таблицы

Синтаксис определения триггерной функции

```
CREATE FUNCTION функция () RETURNS trigger AS '
DECLARE
    объявления ;
BEGIN
команды;
END; '
LANGUAGE  plpgsql;
```

В триггерных функциях используются специальные переменные, содержащие информацию о сработавшем триггере. С помощью этих переменных триггерная функция работает с данными таблиц.

Имя	Тип	Описание
NEW	RECORD	Новые значения полей записи базы данных, созданной командой INSERT или обновленной командой UPDATE, при срабатывании триггера уровня записи (ROW). Переменная используется для модификации новых записей. Переменная NEW доступна только при операциях INSERT и UPDATE. Поля записи NEW могут быть изменены триггером.
OLD	RECORD	Старые значения полей записи базы данных, содержавшиеся в записи перед выполнением команды DELETE или UPDATE при срабатывании триггера уровня записи (ROW. Переменная OLD доступна только при операциях DELETE и UPDATE. Поля записи OLD можно использовать только для чтения, изменять нельзя.
TG_NAME	name	Имя сработавшего триггера.
TG_WHEN	text	Строка BEFORE или AFTER в зависимости от момента срабатывания триггера, указанного в определении (до или после операции).
TG_LEVEL	text	Строка ROW или STATEMENT в зависимости от уровня триггера, указанного в определении.
TG_OP	text	Строка INSERT, UPDATE или DELETE в зависимости от операции, вызвавшей срабатывание триггера.
TG_RELID	oid	Идентификатор объекта таблицы, в которой сработал триггер.
TG_RELNAME	name	name Имя таблицы, в которой сработал триггер.

К отдельным полям записей NEW и OLD в триггерных процедурах обращаются следующим образом: NEW.names , OLD.rg.

Примеры создания триггеров

Пример 1. Триггер выполняется перед удалением записи из таблицы поставщиков T1. Триггер проверяет наличие в таблице поставок S1 записей, относящихся к удаляемому поставщику, и, если такие записи есть, удаляет их.

Создание триггерной функции

```
CREATE FUNCTION trigger1 () RETURNS trigger AS '
BEGIN
if (select count(*) from S1 a where trim(a.ns)=trim(OLD.ns))>0
then delete from S1 where trim(S1.ns)=trim(OLD.ns);
end if;
return OLD;
END;
' LANGUAGE  plpgsql;
```

Создание триггера

```
CREATE TRIGGER trig1
BEFORE DELETE ON T1 FOR EACH ROW
EXECUTE PROCEDURE trigger1();
```

Проверка работы триггера

Delete from T1 where ns=’S2’;

Пример 2. Создание триггера-генератора для таблицы поставщиков T1.

Триггер выполняется перед вставкой новой записи в таблицу поставщиков T1. Триггер проверяет значения, которые должна содержать новая запись (record NEW) и может их изменить:

- если не указан номер поставщика – он генерируется по схеме – S+ уникальный номер из последовательности;
- если не указан рейтинг или рейтинг <=0 – устанавливается рейтинг = 10 для поставщиков из Tomska и 0 для всех остальных.

Создание последовательности

```
CREATE SEQUENCE seq1 INCREMENT BY 1 START WITH 25;
```

Создание триггерной функции

(В этой функции вызывается перегружаемая функция n11).

```
CREATE FUNCTION trigger2 () RETURNS trigger AS '
BEGIN
NEW.ns=n11(NEW.ns,’’S’’||trim(to_char(nextval(’’seq1’’),’’99999’’)));

 if (n11(NEW.rg,0)<=0)  then
      If  NEW.town= ’’Tomsk’’   then  NEW.rg=10;
     else NEW.rg=0;
     end if;
end if;
return NEW;
```

```
END;
' LANGUAGE  plpgsql;
```

Создание триггера

```
CREATE TRIGGER trig2
BEFORE INSERT ON T1 FOR EACH ROW
EXECUTE PROCEDURE trigger2 ();
```

Проверка работы триггера

```
insert into T1 values(null,null,null,null);
```