Отчет по ЛР 4
“Метаданные”

Выполнил студент группы ПМИ-34-20

Семенов Сергей

Дата сдачи: 21.11.2022

Задание:

Решить проблему неконтролируемых ошибок разработанной ранее хранимой процедуры (ХП) в случае, если в качестве параметров передана несуществующая таблица, несуществующий столбец существующей таблицы или столбец, который не имеет целочисленный тип, посредством проверки корректности параметров на основе метаданных базы данных. В случае некорректных параметров самостоятельно порождать ошибку.

Внести такие изменения в код ХП, чтобы имя триггера формировалось по шаблону "{имя таблицы}_{имя столбца}_{номер триггера}", где номер триггера выбирается как количество существовавших ранее триггеров для этой таблицы + 1. Перед созданием требуется проверить существование другого объекта в базе данных с предлагаемым именем. В случае, если такой объект уже есть, требуется сформировать имя любым другим образом.

Решение:

Для решения этой л/р нам понадобится снова изменить нашу хранимую процедуру. Первым делом добавим проверки на корректность входных данных. Для этого нам понадобятся следующие метаданные:

information_schema.columns – содержит информацию обо всех столбцах таблиц (или столбцах представлений) в базе данных.

information_schema.tables – содержит все таблицы и представления, содержащиеся в текущей базе данных.

information_schema.triggers - показывает все триггеры, определённые в текущей базе данных для таблиц и представлений, к которым имеет доступ текущий пользователь.

Всякий раз, когда нам будет приходить неверное название таблицы/столбца или столбец с неправильным типом данных, наша функция будет выдавать соответствующую ошибку и возвращать 0.

Для этого были добавлены следующие проверки:

```
IF NOT EXISTS(SELECT * FROM information_schema.tables
                                WHERE table_schema = _schema_name AND table_name = _table_name)
```

```
    THEN
        RAISE NOTICE 'таблицы с таким именем не существует!';
        RETURN 0;
    END IF;

    IF NOT EXISTS(SELECT * FROM information_schema.columns
        WHERE table_schema = _schema_name AND table_name = _table_name AND column_name = _column_name)
    THEN
        RAISE NOTICE 'Столбца с таким именем не существует!';
        RETURN 0;
    END IF;

    IF NOT EXISTS(SELECT * FROM information_schema.columns
        WHERE table_schema = _schema_name AND table_name = _table_name AND column_name = _column_name AND data_type = 'integer')
    THEN
        RAISE NOTICE 'Передан столбец, тип данных которого не целочисленный';
        RETURN 0;
    END IF;
```

Так же, пришлось изменить создание названия тригера. Теперь будет формировать его следующим образом: "{имя таблицы}_{имя столбца}_{номер триггера}", где номер триггера выбирается как количество существовавших ранее триггеров для этой таблицы + 1.

Количество уже существующих тригеров получаем следующим образом:

```
triggerCount = (SELECT count(*) + 1 FROM
    (SELECT trigger_name FROM information_schema.triggers
        WHERE event_object_schema = _schema_name AND event_object_table = _table_name)
    AS triggers);
```

Далее проверим, не существует ли тригер у рассматриваемой таблицы с вычисленным id(кол-во триггеров) в конце названия. Если такой имеется, увеличиваем счетчик тригера

```
LOOP
    IF EXISTS(SELECT * FROM information_schema.triggers
```

```
                                        WHERE event_object_schema = _schema_name
AND event_object_table = _table_name AND
                                        trigger_name = _table_name || '_' ||
_column_name || '_' || triggerCount)
                                        THEN
                                                triggerCount = triggerCount + 1;
                                        ELSE
                                                EXIT;
                                END IF;
                        END LOOP;
```

И, непосредственно, создаем триггеры, увеличивая счетчик, после создания первого триггера:

```
        EXECUTE format('CREATE TRIGGER %I AFTER INSERT ON %s
                                        FOR EACH STATEMENT
                                        EXECUTE FUNCTION
upd_spec_maxvalue(%s, %s);',
                                        _table_name || '_' || _column_name ||
'_' || triggerCount, _table_name, _table_name, _column_name);


                        triggerCount = triggerCount + 1;


                        EXECUTE format ('CREATE TRIGGER %I AFTER UPDATE ON %s
                                        FOR EACH STATEMENT
                                        EXECUTE FUNCTION upd_spec_maxvalue(%s,
%s);',
                                _table_name || '_' || _column_name || '_' ||
triggerCount, _table_name, _table_name, _column_name);
```

Тестирование:

```
        --создадим таблицу spec
        CREATE TABLE spec
        (
          id integer NOT NULL,
          table_name character varying(30) NOT NULL,
```

```
    column_name character varying(30) NOT NULL,
    cur_max_value integer NOT NULL
);
```

--добавим изначальные значения

```
INSERT INTO spec VALUES (1, 'spec', 'id', 1);
```

--создадим таблицу test

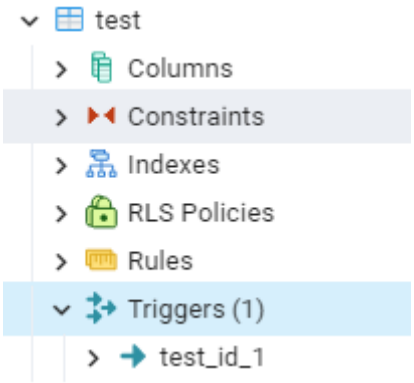
```
CREATE TABLE test
(
    id integer NOT NULL
);
```

--добавим в столбец id таблицы тест значение 30

```
INSERT INTO test VALUES (30)
```

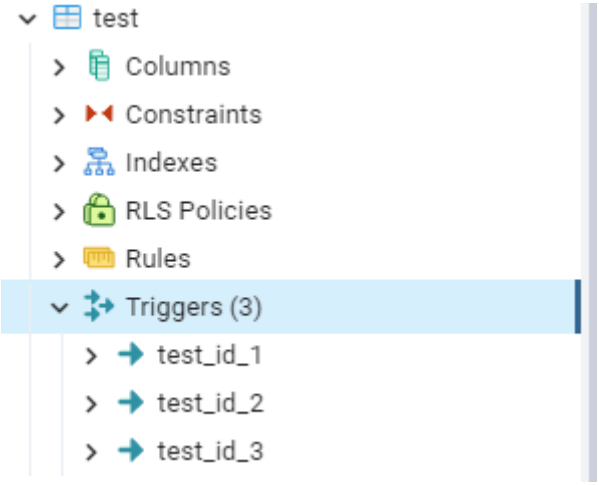
--создадим триггер вне функции и дадим название test_id_1

```
CREATE TRIGGER test_id_1 AFTER INSERT ON test
                FOR EACH STATEMENT
        EXECUTE FUNCTION upd_spec_maxvalue(test, id);
```



--вызовем хранимую процедуру с параметрами test id public

```
SELECT xp('test', 'id', 'public')
```



Создались 2 новых триггера с номерами 2 и 3, т.к. первый номер уже был занят.

--вызовем хранимую процедуру с названием несуществующей таблицы

SELECT xp('NAN', 'id', 'public')

```
124  --вызовем хранимую процедуру с названием несуществующей таблицы
125  SELECT xp('NAN', 'id', 'public')
126
```

Data output | Notifications | Messages

	xp integer
1	0

```
ЗАМЕЧАНИЕ:  таблицы с таким именем не существует!

Successfully run. Total query runtime: 50 msec.
1 rows affected.
```

--вызовем хранимую процедуру с несуществующим столбцом

SELECT xp('test', 'NAN', 'public')

```
127  --вызовем хранимую процедуру с несуществующим столбцом
128  SELECT xp('test', 'NAN', 'public')
129
130
131
132
133
```

Data output | Notifications | Messages

	xp integer
1	0

```
ЗАМЕЧАНИЕ:  Столбца с таким именем не существует!

Successfully run. Total query runtime: 46 msec.
1 rows affected.
```

--создадим таблицу test со столбцом строкового типа

CREATE TABLE test2

(

name character varying(30) NOT NULL

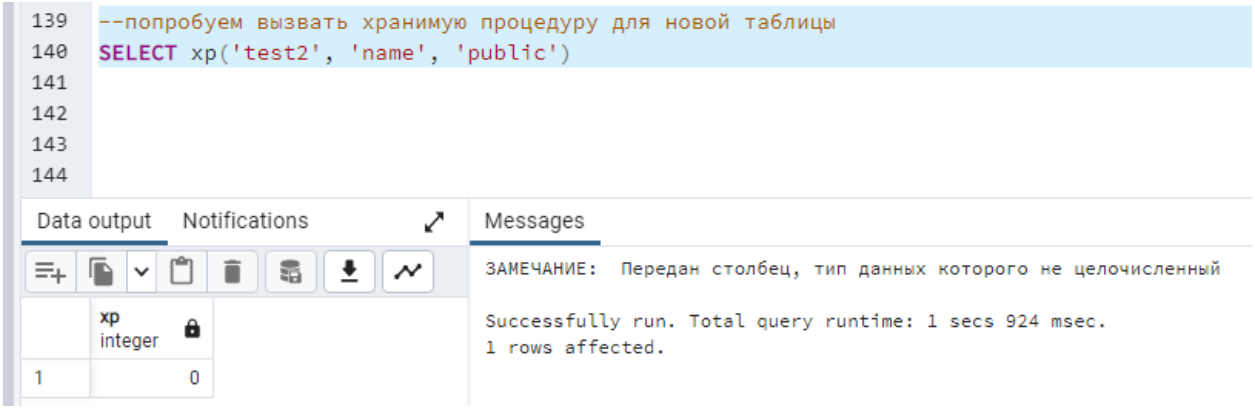
);

--добавим запись в новую таблицу

INSERT INTO test2 VALUES ('newValue')

--попробуем вызвать хранимую процедуру для новой таблицы

SELECT xp('test2', 'name', 'public')



--также проверим работоспособность триггера

--вставим в таблицу test значение большее, чем максимальное (30 на данном этапе)

INSERT INTO test VALUES (50)

--и посмотрим таблицу spec после вставки

select * from spec

```
147  --и посмотрим таблицу spec после вставки
148  select * from spec
149
150
151
```

Data output | Notifications

	id integer	table_name character varying (30)	column_name character varying (30)	cur_max_value integer
1	1	spec	id	2
2	2	test	id	50

Триггер успешно сработал

Код исполняемого файла:

```
--создаем тригерную функцию
CREATE OR REPLACE FUNCTION upd_spec_maxvalue()
RETURNS trigger AS
$$
DECLARE
   maxValue integer;
BEGIN
   EXECUTE format('select max(%s) from %s', tg_argv[1], tg_argv[0]) INTO maxValue;
   UPDATE spec
   SET cur_max_value = maxValue
   WHERE table_name = tg_argv[0] AND column_name = tg_argv[1] AND maxValue > cur_max_value;
   RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

```
--наша хранимая процедура, в которую добавили создание тригера
CREATE OR REPLACE FUNCTION xp (_table_name text, _column_name text, _schema_name text)
RETURNS integer
AS $$
DECLARE
  maxValue integer := 0;
  triggerCount integer := 0;
BEGIN

        IF NOT EXISTS(SELECT * FROM information_schema.tables
                                WHERE table_schema = _schema_name AND table_name = _table_name)
    THEN
                RAISE NOTICE 'таблицы с таким именем не существует!';
                RETURN 0;
        END IF;

    IF NOT EXISTS(SELECT * FROM information_schema.columns
                                WHERE table_schema = _schema_name AND table_name = _table_name AND column_name = _column_name)
    THEN
                RAISE NOTICE 'Столбца с таким именем не существует!';
                RETURN 0;
        END IF;

    IF NOT EXISTS(SELECT * FROM information_schema.columns
            WHERE table_schema = _schema_name AND table_name = _table_name AND column_name = _column_name AND data_type = 'integer')
```

```
        THEN
                RAISE NOTICE 'Передан столбец, тип данных которого не целочисленный';
                RETURN 0;
        END IF;

    IF
                (SELECT COUNT(*)
                FROM spec
                WHERE column_name = _column_name AND table_name = _table_name) > 0
    THEN
                UPDATE spec
                SET cur_max_value = cur_max_value + 1
                WHERE column_name = _column_name AND table_name = _table_name;

                RETURN cur_max_value FROM spec
                                WHERE column_name = _column_name AND table_name = _table_name;

        ELSE
                EXECUTE format('SELECT MAX(%s)
                                                FROM %s ',
                                                _column_name, _table_name)
                                                INTO maxValue;
                IF maxValue IS null THEN maxValue := 1; ELSE maxValue := maxValue + 1; END IF;
```

```
                EXECUTE format('INSERT INTO spec
                                VALUES (%s, ''%s'', ''%s'', %s)',
                                (SELECT xp('spec', 'id', _schema_name)),
_table_name, _column_name, maxValue);
                --получаем кол-во триггеров в рассматриваемой таблице
                triggerCount = (SELECT count(*) + 1 FROM
                                (SELECT trigger_name FROM
information_schema.triggers
                                WHERE event_object_schema =
_schema_name AND event_object_table = _table_name) AS triggers);
                --проверим, существует ли уже тригер для
рассматриваемой таблицы и столбца, с полученным номером на конце,
увеличим номер по необходимости
                LOOP
                IF EXISTS(SELECT * FROM information_schema.triggers
                                WHERE event_object_schema =
_schema_name AND event_object_table = _table_name AND
                                trigger_name = _table_name || '_' ||
_column_name || '_' || triggerCount)
                                THEN
                                triggerCount = triggerCount + 1;
                                ELSE
                                EXIT;
                END IF;
                END LOOP;
                --создадим триггеры
                EXECUTE format('CREATE TRIGGER %I AFTER INSERT ON %s
                                FOR EACH STATEMENT
                                EXECUTE FUNCTION
upd_spec_maxvalue(%s, %s);',
```

```
                                                    _table_name || '_' || _column_name
|| '_' || triggerCount, _table_name, _table_name, _column_name);

                        --увеличим счетчик триггеров на 1, после создания первого
триггера

                        triggerCount = triggerCount + 1;


                        EXECUTE format ('CREATE TRIGGER %I AFTER UPDATE ON %s

                                                    FOR EACH STATEMENT

                                                    EXECUTE FUNCTION
upd_spec_maxvalue(%s, %s);',

                                                    _table_name || '_' || _column_name || '_'
|| triggerCount, _table_name, _table_name, _column_name);


                RETURN maxValue;

        END IF;

END;

$$ LANGUAGE plpgsql;


--создадим таблицу spec

CREATE TABLE spec

(

   id integer NOT NULL,

   table_name character varying(30) NOT NULL,

   column_name character varying(30) NOT NULL,

   cur_max_value integer NOT NULL

);

--добавим изначальные значения

INSERT INTO spec VALUES (1, 'spec', 'id', 1);

--создадим таблицу test
```

```
CREATE TABLE test
(
    id integer NOT NULL
);
--добавим в столбец id таблицы тест значение 30
INSERT INTO test VALUES (30)
--создадим триггер вне функции и дадим название test_id_1
CREATE TRIGGER test_id_1 AFTER INSERT ON test
                    FOR EACH STATEMENT
                    EXECUTE FUNCTION upd_spec_maxvalue(test, id);
--вызовем хранимую процедуру с параметрами test id public
SELECT xp('test', 'id', 'public')

--вызовем хранимую процедуру с названием несуществующей таблицы
SELECT xp('NAN', 'id', 'public')

--вызовем хранимую процедуру с несуществующим столбцом
SELECT xp('test', 'NAN', 'public')

--создадим таблицу test со столбцом строкового типа
CREATE TABLE test2
(
    name character varying(30) NOT NULL
);

--добавим запись в новую таблицу
INSERT INTO test2 VALUES ('newValue')
```

```
--попробуем вызвать хранимую процедуру для новой таблицы

SELECT xp('test2', 'name', 'public')


--также проверим работоспособность триггера

--вставим в таблицу test значение большее, чем максимальное (30 на данном этапе)

INSERT INTO test VALUES (50)


--и посмотрим таблицу spec после вставки

select * from spec
```