Лекция 4

1. PL/pgSQL

ИТМО ВТ

Пользовательские функции

- функции на языке запросов (функции, написанные на языке SQL);
- функции на процедурных языках (функции, написанные, например, на PL/pgSQL или PL/Tcl);
- функции на языке C;

ИТМО ВТ

PL/pgSQL

- PL/pgSQL это процедурный язык для СУБД PostgreSQL.

- Функции PL/pgSQL могут использоваться везде, где допустимы встроенные функции.

ИТМО ВТ

Создание функции

Добавление пользовательских функций:

CREATE FUNCTION newFunc(int)

RETURNS int

AS 'body logic'

LANGUAGE chosenLanguage;

При добавлении функции можно использовать **CREATE OR REPLACE** — чтобы заменить ранее загруженную функцию.

ИТМО ВТ

SQL-функция

Функция для изменения группы студента:

```
CREATE FUNCTION updateStudentGroup(int, int)
    RETURNS void AS
        'UPDATE STUDENT
            SET GroupID=$2 WHERE StudID=$1;'
    LANGUAGE SQL;
```

- Аргументы могут использоваться как значения, но не идентификаторы.

SQL-функция, параметры

- Начиная с PostgreSQL 9.2:

```
CREATE FUNCTION updateStudentGroup(
    StudIDParam int, GroupIDParam int)
    RETURNS void AS
        'UPDATE STUDENT
            SET GroupID=GroupIDParam
            WHERE StudID=StudIDParam;'
    LANGUAGE SQL;
```

Есть ли проблема в приведенном коде?

```
CREATE FUNCTION updateStudentGroup(
 GroupIDParam int )
 RETURNS void AS
        'UPDATE STUDENT
                SET GroupID=GroupIDParam
                WHERE StudName='Petr';'
LANGUAGE SQL;
```

ИТМО ВТ

Экранирование строк

```
CREATE FUNCTION updateStudentGroup(
 GroupIDParam int )
 RETURNS void AS
        'UPDATE STUDENT
                SET GroupID=GroupIDParam
                WHERE StudName=''Petr'';'
LANGUAGE SQL;
```

Синтаксис для записи строк через $$

Строка:

string of characters with ' symbol

может быть представлена:

- 'string of characters with '' symbol'
- $someId$string of characters with ' symbol$someId$

Синтаксис для записи строк через $$

```
CREATE FUNCTION updateStudentGroup(
 GroupIDParam int )
 RETURNS void AS $$
        UPDATE STUDENT
                SET GroupID=GroupIDParam
                WHERE StudName='Petr';
$$ LANGUAGE SQL;
```

Вызов функции

Для вызова функции нужен SELECT:

- SELECT updateStudentGroup(950, 345);

ИТМО ВТ

Процедура

Добавление процедуры:

```
CREATE PROCEDURE newProc(integer)
AS $$ body logic $$
LANGUAGE chosenLanguage;
```

- В процедуре отсутствует RETURNS часть.
- Для вызова процедуры можно использовать CALL:

```
CALL newProc(42);
```

ИТМО ВТ

Введение

PL/pgSQL это блочно-структурированный язык. Текст определения функции должен быть блоком.

Структура блока:

```
[ <<метка>> ]
[ DECLARE
    объявления ]
BEGIN
    операторы
END [ метка ];
```

ИТМО ВТ

Пример

```
CREATE FUNCTION somefunc() RETURNS integer AS $$ -- SQL command
<< outerblock »                                  -- pl/pgSQL
DECLARE
    quantity integer := 30;
BEGIN
    -- Создаем вложенный блок
    DECLARE
        quantity integer := 80;
    BEGIN
        RAISE NOTICE 'Inner quantity = %', quantity;  -- Выводится 80
        RAISE NOTICE 'Outer quantity = %', outerblock.quantity;  -- Выводится 30
    END;
    RAISE NOTICE 'Сейчас quantity = %', quantity;  -- Выводится 30
    RETURN quantity;
END;
$$ LANGUAGE plpgsql;
```

ИТМО ВТ

Анонимный блок

```
DO $$
<<studentBlock>>
DECLARE
  studCount integer := 0;
BEGIN
   SELECT COUNT(*)
   INTO studCount
   FROM STUDENT;
   RAISE NOTICE 'Students: %', studCount;
end studentBlock $$;
```

2. Ограничения целостности: триггеры

ИТМО ВТ

Ограничения целостности

- Типы данных — один из способов ограничивать данные, но его не всегда **достаточно**.

- SQL позволяет определять ограничения для колонок и таблиц:
  - CHECK, NOT NULL, UNIQUE;
  - триггеры;

ИТМО ВТ

Триггеры

Триггер — сочетание хранимой в базе данных процедуры и события, которое заставляет ее выполняться.

Могут быть объявлены как для таблицы из пользовательской БД, так и для представления (PostgreSQL).

ИТМО ВТ

Запуск триггеров

События:

- ввод новой строки;
- изменение значений одного или нескольких столбцов таблицы;
- удаление строк таблицы;

ИТМО ВТ

Запуск триггеров

При возникновении события:

- производится проверка условий срабатывания триггера;
- запускаются созданные триггеры;

ИТМО ВТ

Триггеры

Триггеры дают возможность:

- Реализовывать **сложные ограничения целостности** данных, которые невозможно реализовать через ограничения, устанавливаемые при создании таблицы *(CHECK …)*.

- Контролировать информацию, хранимую в таблице, посредством **регистрации вносимых изменений** и пользователей, производящих эти изменения.

ИТМО ВТ

Создание триггера (PostgreSQL)

Триггеры создаются на основе триггерной функции:

- функция без аргументов;
- возвращает тип *trigger, event_trigger*.

CREATE OR REPLACE FUNCTION **fname()** …

RETURNS trigger ...;

CREATE TRIGGER name …

EXECUTE PROCEDURE **fname()**;

ИТМО ВТ

Синтаксис

CREATE TRIGGER *name*

{ BEFORE | AFTER | INSTEAD OF } { event [OR ...] }

ON table_name

...

[FOR [EACH] { ROW | STATEMENT }]

[WHEN (condition)]

EXECUTE PROCEDURE **function_name** ( )

Где event:

INSERT, UPDATE [OF column_name [, ...]], DELETE, TRUNCATE

ИТМО ВТ

Виды триггеров

- **Строковый триггер** — запускается один раз для каждой строки, обрабатываемой активизирующим предложением.
- **Табличный триггер** — выполняется только один раз, независимо от того, сколько строк содержит запускающее его предложение.

Выбор того или иного типа триггера зависит от требований, определенных для базы данных.

ИТМО ВТ

Строковые/Табличные триггеры

- **ROW** — процедура будет вызываться для каждой модифицируемой записи
- **STATEMENT** — процедура будет вызываться один раз для всех обрабатываемых записей в рамках команды

CREATE TRIGGER *trigger1* BEFORE UPDATE ON table FOR **EACH ROW** EXECUTE ...;

Колоночный триггер:

CREATE TRIGGER *trigger2* BEFORE **UPDATE OF** *attr1* **ON** table **FOR EACH ROW** EXECUTE ...;

ИТМО ВТ

Типы триггеров

- Триггеры DML (обычные и условные).
- Триггеры замещения (instead of).
- Событийные триггеры (event triggers).

ИТМО ВТ

Триггеры DML

Триггеры DML активизируются предложениями ввода, обновления и удаления информации (INSERT , UPDATE , DELETE) до или после выполнения предложения, на уровне строки или таблицы.

CREATE TRIGGER *trname* BEFORE **DELETE** ON *table*

ИТМО ВТ

Триггеры DML

CREATE TRIGGER *name* { **BEFORE** | **AFTER** | ...}

- **BEFORE** — процедура вызывается перед выполнением операции (включая проверки ограничений целостности, реализуемые при выполнении команд INSERT и DELETE).
- **AFTER** — процедура вызывается после завершения операции, приводящей в действие триггер.

ИТМО ВТ

Активация триггеров

Активация триггеров DML — при выполнении предложений INSERT, UPDATE, DELETE.

Для каждого предложения может быть создано 4 триггера: табличный BEFORE, табличный AFTER, строковый BEFORE, строковый AFTER.

ИТМО ВТ

Порядок активации

Порядок активации триггеров:

1. Табличный триггер BEFORE
2. Строковый триггер BEFORE
3. Строковый триггер AFTER
4. Табличный триггер AFTER

ИТМО ВТ

Переменные триггерной процедуры

Когда PL/pgSQL функция вызывается триггером, создается ряд переменных:

- NEW — (RECORD) содержит новую строку базы данных для команд INSERT/UPDATE в строковых триггерах.
- OLD — (RECORD) содержит старую строку базы данных для команд UPDATE/DELETE в строковых триггерах.
- TG_NAME — имя сработавшего триггера.
- TG_WHEN — BEFORE, AFTER или INSTEAD OF, в зависимости от определения триггера.

Пример

```
CREATE TABLE EMPLOYEE(
   ID              INT              PRIMARY KEY,
   NAME            TEXT             NOT NULL,
   ADDR            CHAR(50),
   SALARY          REAL
);

CREATE TABLE AUDIT(
    EMP_ID              INT         NOT NULL,
    ENTRY_DATE          TEXT        NOT NULL
);
```

ИТМО ВТ

Пример

```
CREATE TRIGGER my_trigger
AFTER INSERT ON EMPLOYEE
FOR EACH ROW EXECUTE PROCEDURE auditfunc();

CREATE OR REPLACE FUNCTION auditfunc()
RETURNS TRIGGER AS $$
    BEGIN
        INSERT INTO AUDIT(EMP_ID, ENTRY_DATE)
                VALUES (NEW.ID, current_timestamp);
        RETURN NEW;
    END;
$$ LANGUAGE plpgsql;
```

ИТМО ВТ

Триггеры замещения

- Триггеры замещения (**instead of**) можно создавать только для представлений.
- В отличие от триггеров DML триггеры замещения выполняются вместо предложений DML, вызывающих их срабатывание.
- Триггеры замещения должны быть строковыми триггерами.

ИТМО ВТ

Событийные триггеры

- Срабатывают на предложения DDL (CREATE/ALTER/DROP).
- Процедура, которая вызывается как событийный триггер, должна объявляться без аргументов. Типом возвращаемого значения должен быть event_trigger.

Событийные триггеры

CREATE OR REPLACE FUNCTION **eventtest()** RETURNS *event_trigger* AS $$

BEGIN

    RAISE NOTICE 'eventtest: %', tg_event;

END;

$$ LANGUAGE plpgsql;

CREATE EVENT TRIGGER **eventtest** ON **ddl_command_start** EXECUTE PROCEDURE **eventtest()**;

Событийные триггеры

tg_event — событие, вызывающее триггер;

```
$ create table SOMETABLE (myattr int4);

NOTICE:  eventtest: ddl_command_start CREATE

CREATE TABLE
```

ИТМО ВТ

Триггеры: общие правила

- Триггерная функция должна вернуть либо NULL, либо запись/строку, соответствующую структуре таблицы, для которой сработал триггер.
- Если строковый триггер с BEFORE возвращает NULL, то все дальнейшие действия с этой строкой прекращаются (не срабатывают последующие триггеры, команда INSERT/UPDATE/DELETE для этой строки не выполняется).

Удаление триггеров

DROP TRIGGER *name*;

ИТМО ВТ

Литература

При подготовке презентации использовались материалы из:

- Введение в реляционные базы данных / В. В. Кириллов, Г. Ю. Громов, Издательство: BHV, 2009 г.
- Документация PostgreSQL.

  https://www.postgresql.org/about/licence/