**Лабораторная работа № 11\_1 Ошибки, исключительные ситуации POSTGRESQL**

**Обработка ошибок и исключений в pl/pgsql коде**

Обработка исключений в хранимых процедурах и анонимных блоках возможна только внутри конструкции

**BEGIN**

*-- Тут располагается потенциально опасный код*

**EXCEPTION**

**WHEN** '<идентификатор\_ошибки>' **THEN**

*-- Тут находится код, обрабатывающий исключение*

**END**

Для того, чтобы обработать исключение, необходимо указать идентификатор ошибки, который будет обрабатываться. Коды всех ошибок и их идентификаторов можно найти на странице [PostgreSQL Error Codes](http://www.postgresql.org/docs/current/static/errcodes-appendix.html) официальной документации.

К примеру, необходимо обрабатывать ситуацию, когда происходит запись двух одинаковых значений в уникальное поле таблицы

Выполнив нижеприведенный код:

**drop table test\_table;**

**DO** $$

BEGIN

CREATE TABLE test\_table(

name varchar UNIQUE

);

INSERT INTO test\_table(name) VALUES('my name');

INSERT INTO test\_table(name) VALUES('my name');

END $$;

получим исключение:

ERROR: duplicate key value violates unique constraint "test\_table\_name\_key"

DETAIL: Key (name)=(my name) already exists.

CONTEXT: SQL statement "INSERT INTO test\_table(name) VALUES('my name')"

PL/pgSQL function "inline\_code\_block" line 7 at SQL statement

по тексту исключения можно попробовать найти ошибку на странице [PostgreSQL Error Codes](http://www.postgresql.org/docs/current/static/errcodes-appendix.html), но это не всегда бывает возможно. Поэтому надо узнать код ошибки однозначно. Для этого надо вывести в лог значение специальной переменной **SQLSTATE**, которая и содержит искомый код.

**DO** $$

BEGIN

CREATE TABLE test\_table(

name varchar UNIQUE

);

INSERT INTO test\_table(name) VALUES('my name');

INSERT INTO test\_table(name) VALUES('my name');

EXCEPTION

WHEN others THEN

RAISE NOTICE 'SQLSTATE: %', SQLSTATE;

RAISE;

END $$;

в логе виден код ошибки (23505):

NOTICE: SQLSTATE: 23505

ERROR: duplicate key value violates unique constraint "test\_table\_name\_key"

DETAIL: Key (name)=(my name) already exists.

CONTEXT: SQL statement "INSERT INTO test\_table(name) VALUES('my name')"

PL/pgSQL function "inline\_code\_block" line 7 at SQL statement

другое дело. По таблице кодов ошибок сразу находим идентификатор ошибки, который надо обрабатывать

| **Error Code** | **Condition Name** |
| --- | --- |
| 23505 | unique\_violation |

изменяем предыдущий код:

**DO** $$

BEGIN

CREATE TABLE test\_table(

name varchar UNIQUE

);

INSERT INTO test\_table(name) VALUES('my name');

INSERT INTO test\_table(name) VALUES('my name');

EXCEPTION

WHEN unique\_violation THEN

RAISE NOTICE 'Illegal operation: %', SQLERRM;

END $$;

Результат:

PL/pgSQL function "inline\_code\_block" line 3 at SQL statement

NOTICE: Illegal operation: duplicate key value violates unique constraint "test\_table\_name\_key"

Исключительная ситуация обработана.

**Совет**

Бывают ситуацию, когда надо защититься от всех возможных ошибок, которые могут возникнуть. В этом случае поможет специальный идентификатор ошибки **others**. С помощью этого идентификатора можно ловить любые типы исключения, кроме **query\_canceled**

Обработка исключений Postgresql

CREATE OR REPLACE FUNCTION r\_function () RETURNS void AS $$

DECLARE

result NUMERIC;

BEGIN

result := 10/0;

EXCEPTION

WHEN division\_by\_zero THEN

RAISE EXCEPTION 'Ошибка деления на ноль !!!!!!!!!!!!!!!!!';

WHEN others THEN

RAISE EXCEPTION 'Произошла неизвестная ошибка';

END;

$$ LANGUAGE plpgsql;

SELECT r\_function() AS output;

Блоки обработки исключений могут содержать любые допустимые в языке PL/pgSQL операторы, например, операторы для вывода сообщений, записи в лог, изменения данных и т.д.

Использование блока WHEN others может быть полезным, когда нам необходимо выполнить какие-то действия для всех непредвиденных исключительных ситуаций (например, запись информации об ошибке в журнал).