Обработка ошибок

Исключения

- **Исключение** (исключительная ситуация) – ошибка, возникающая во время выполнения, т.е. не относящаяся к синтаксису или семантике и не обнаруженная на этапе компиляции
  - не должна возникать при нормальном выполнении программы
- **Инициируется**
  - **Неявно** – сервером PostgreSQL => **серверные исключения**
  - **Явно** – программистом => **пользовательские исключения**
- Может быть перехвачено и обработано в секции **EXCEPTION** блока PL/pgSQL
  - перехватить можно только ошибки, возникшие в исполняемой секции блока
  - не обязательно перехватывать для обработки ошибку в том же блоке, где она возникла, можно сделать это на уровне выше

Генерация пользовательского сообщения

- Для генерации пользовательского сообщения используется оператор **raise**

```
RAISE [LEVEL] 'message (%)', arg_name [USING option];
```

  - **level** – {debug | log | notice | info | warning | **exception**}
    - указывает серьезность ошибки
    - уровень по умолчанию - **exception:** вызывает ошибку и останавливает текущую транзакцию
  - **message** - текст возвращаемого сообщения
    - для формирования динамического сообщения используются заполнители (%), которые будут заменены аргументами (**arg_name)**
    - **количество заполнителей должно совпадать с количеством аргументов**
  - **option** – тип дополнительной информации
- **Применение:**
  - В функцию пришел некорректный параметр
  - Состояние данных не соответствует требованиям
  - Необходимо прервать выполнение функции и подать сигнал «наверх» о возникшей проблеме

Уровень сообщения

- DEBUG – отладка
- LOG – запись в лог
- NOTICE – замечание
- INFO – информация
- WARNING – потенциальная опасность
- **EXCEPTION** – ошибка (исключение)
  - Откатывает текущую транзакцию (**Rollback**)
- **Параметры конфигурации**:
  - **log_min_messages –** регулирует уровень сообщений, регистрируемых в журнале сервера (по умолчанию **WARNING**)
  - **client_min_messages –** регулирует уровень сообщений, передаваемых вызывающей стороне (по умолчанию – **NOTICE**)

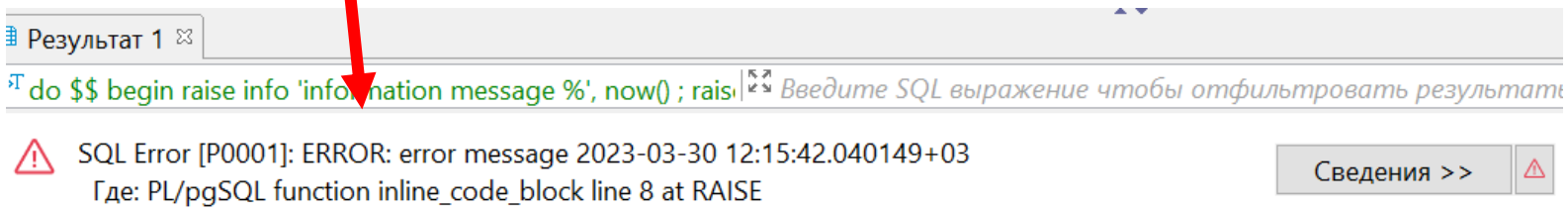
Сообщения PL/pgSQL

```
do $$
begin
  raise info 'information message %', now() ;
  raise log 'log message %', now();
  raise debug 'debug message %', now();
  raise warning 'warning message %', now();
  raise notice 'notice message %', now();
  raise exception 'error message %', now();
end $$;
```

- Обратите внимание, что не все сообщения возвращаются клиенту:
  - PostgreSQL возвращает клиенту только сообщения **info**, **warning** и **notice**

Вывод

```
information message 2023-03-22 13:24:12.71638+03
warning message 2023-03-22 13:24:12.71638+03
notice message 2023-03-22 13:24:12.71638+03
```



Дополнительная информация об ошибке

- Для добавления дополнительной информации используется выражение

```
USING option = expression
```

- **option** – определяет тип дополнительной информации об ошибке:
  - **message** - сообщение об ошибке
  - **hint** - сообщение-подсказка, чтобы было легче обнаружить основную причину ошибки
  - **detail** - подробная информация об ошибке
  - **errcode** - код ошибки
- **expression** – текст дополнительной информации

```
DO $$
BEGIN
    RAISE exception USING
      message = 'Сообщение об ошибке',
      detail  = 'При выполнении кода произошла ошибка',
      hint = 'Обратитесь к системному администратору',
      ERRCODE='ERR01';
END;
$$;
```

```
SQL Error [ERR01]: ERROR: Сообщение об ошибке
  Подробности: При выполнении кода произошла ошибка
  Подсказка: Обратитесь к системному администратору
```

Сведения об ошибках

- Северная ошибка идентифицируется:
  - Именем
  - Кодом (**SQLSTATE**)
- Код исключения - строка из пяти символов:
  - Класс ошибки – 2 символа
  - Номер ошибки – 3 символа
- Для определения **пользовательского** кода ошибки:
  - **USING** ERRCODE='код_ошибки'
  - **RAISE** exception **SQLSTATE** 'код_ошибки' **USING**…

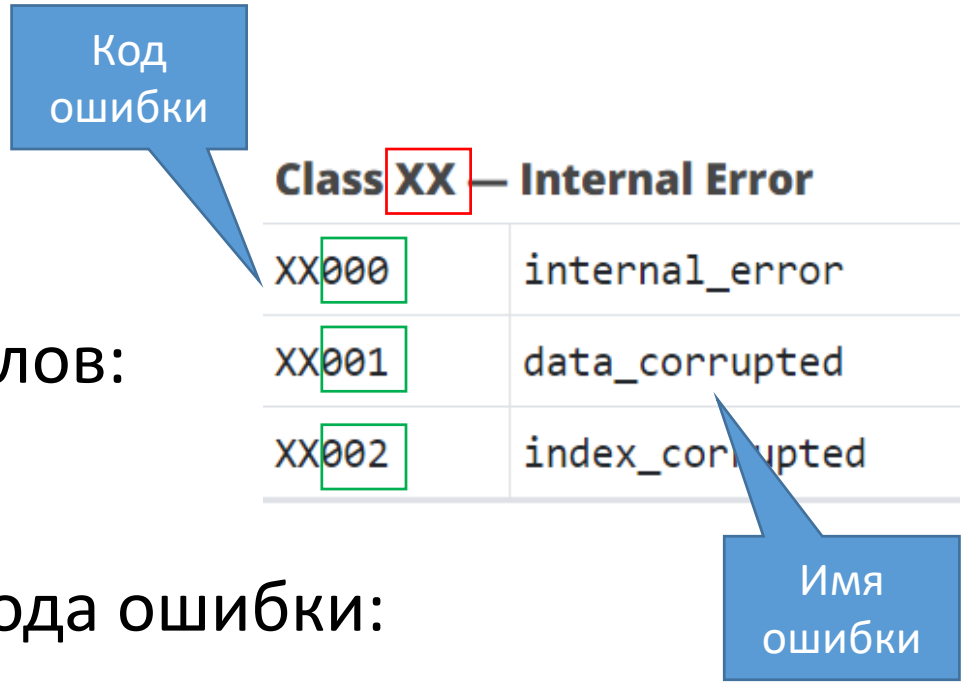


Таблица серверных кодов ошибок и названий состояний – https://www.postgresql.org/docs/current/errcodes-appendix.html

Пример

```
CREATE OR REPLACE FUNCTION functions.get_season(month_number int)
RETURNS text AS $$
DECLARE
    season text;
BEGIN
    IF month_number NOT BETWEEN 1 AND 12 THEN
        RAISE EXCEPTION SQLSTATE '12882'
                        USING HINT='Allowed from 1 up to 12',
                              MESSAGE = 'invalid month. You passed: ' || month_number::text ;
    END IF;
    season = arr[month_number] from string_to_array('зима,зима,весна,весна,весна,лето,
                                                     лето,лето,осень,осень,осень,зима',',') as arr;
    RETURN season;
END;
$$ LANGUAGE plpgsql;
```

```
select functions.get_season(15);
```

SQL Error [12882]: ERROR: invalid month. You passed: (15)
Подсказка: Allowed from 1 up to 12
Где: PL/pgSQL function functions.get_season(integer) line 6 at RAISE

Проверка утверждений

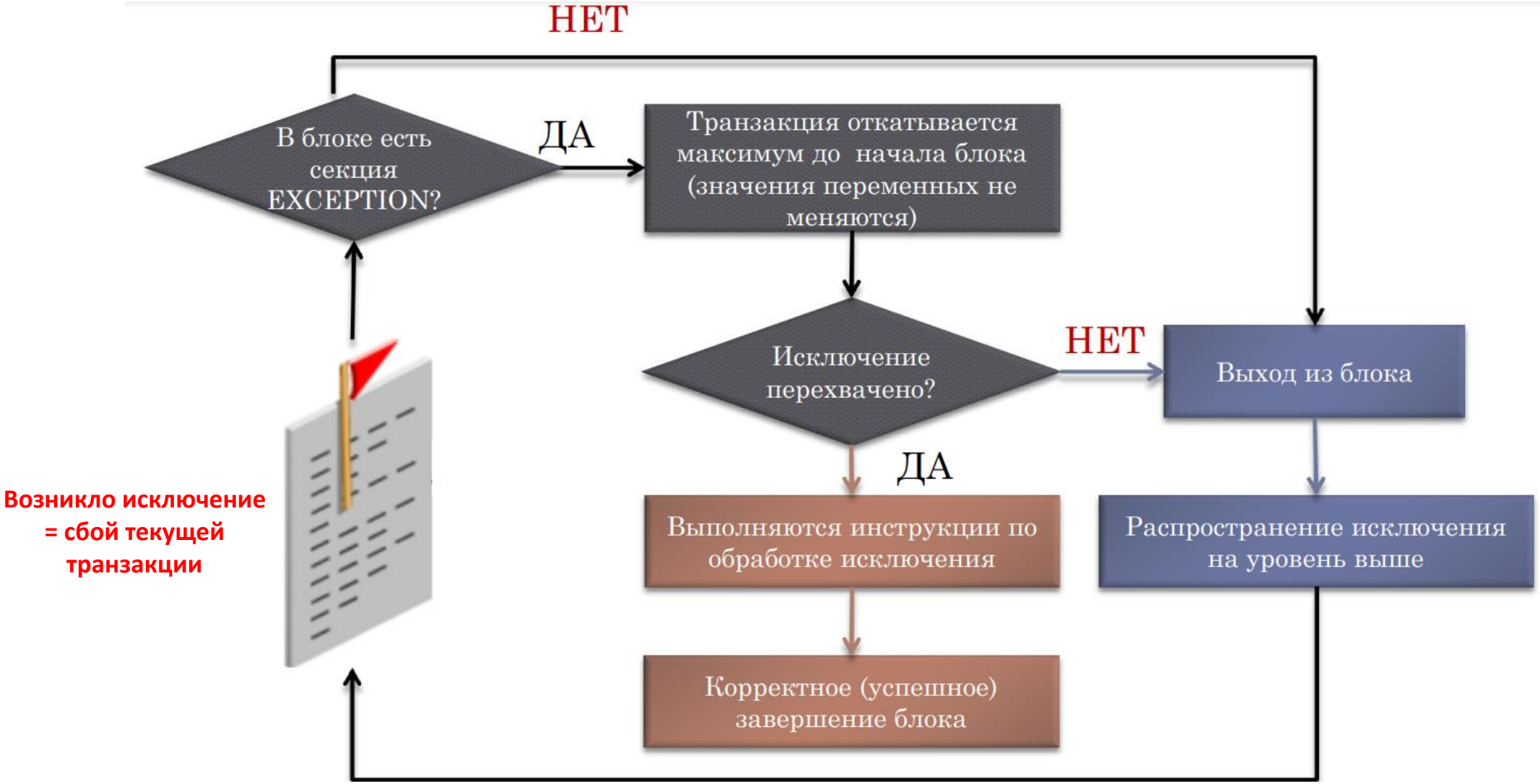
- Для выполнения отладки и выявления программных дефектов можно использовать оператор ASSERT

```
ASSERT условие [ , сообщение ]
```

- условие — логическое выражение, которое, как ожидается, всегда будет возвращать значение true.
  - Если условие равно true, оператор assert ничего не делает
  - Если условие равно false или null, PostgreSQL выдает assert_failure исключение
  - Если ошибка происходит при вычислении условия, она выдаётся как обычная ошибка
- Сообщение – текст возвращаемого сообщения в случае, если условие не выполняется. Если сообщение не указано – возвращается сообщение об ошибке по умолчанию «assertion failed» (нарушение истинности).
- Параметр конфигурации:
  - **plpgsql.check_asserts = on** – включение проверки утверждений

```
do $$
declare
          emp_count integer;
begin
     select count(*) into emp_count
     from public.emp1 ;
    assert emp_count > 0, 'employees not found, check the public.emp1 table';
end$$;
```

События при возникновении исключения



Обработка ошибок

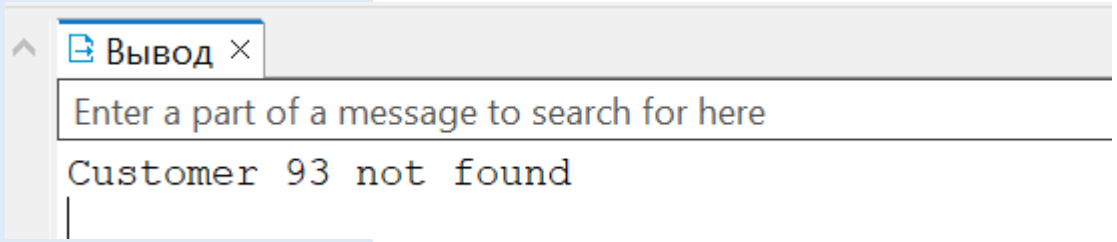
- Чтобы «поймать» исключение используется секция **EXCEPTION**

```
BEGIN
…
EXCEPTION -- работает аналогично CASE, но без раздела ELSE
          WHEN условие [OR условие]… THEN …
          [ WHEN условие [OR условие] ... THEN …]…
          [ WHEN others THEN …]
END;
```

- В качестве условия в обработчике можно использовать:
  - Код ошибки или код класса ошибки
  - Имя ошибки или имя класса ошибки
  - Специальное имя **OTHERS**
    - Позволяет перехватывать любую ошибку, кроме самых фатальных (**QUERY_CANCELED** и **ASSERT_FAILURE**)
- Условия (обработчики) просматриваются сверху вниз
  - выбирается **первая подходящая ветвь** и выполняются ее операторы
  - при использовании в обработчике **кода или имени класса ошибки** будет перехвачена **любая** ошибка данного класса!

Пример обработки исключения no_data_found

```
do
$$
declare
    rec record;
    v_cust_id int = 93;
begin
-- select a customer
    select companyname, contactname
    into strict rec
    from "Sales"."Customers"
    where custid = v_cust_id;
    -- catch exception
    exception
        when sqlstate 'P0002' then --no_data_found then
        raise notice 'Customer % not found', v_cust_id;
end;
$$;
```



Пример обработки нескольких исключений

```
do
$$
declare
    rec record;
    v_custid int = 90;
begin
-- select a film
    select orderid,orderdate
    into strict rec
    from "Sales"."Orders"
    where custid = v_custid;
-- catch exception
exception
    when sqlstate 'P0002' then
        raise notice 'The Customer % not have any order', v_custid;
    when sqlstate 'P0003' then
        raise notice 'The customer % have too many orders', v_custid;
end;
$$;
```

Получение информации об ошибке

- Для получения в обработчике:
  - кода ошибки – переменная **SQLSTATE**
  - текста сообщения – переменная **SQLERRM**
- Для получения дополнительной информации используется команда

```
GET STACKED DIAGNOSTICS variable { = | := } item [ , ... ];
```

  - Специальные элементы диагностики (**item**) позволяют получить:
    - **message_text** – текста ошибки **(message)**
    - **pg_exception_detail** – дополнительной информации об ошибке **(detail)**
    - **pg_exception_hint** – текста сообщения-подсказки **(hint)**

```
GET STACKED DIAGNOSTICS
            _message = message_text,
            _detail = pg_exception_detail,
            _hint = pg_exception_hint;
```

- **Все эти данные не доступны вне обработчиков исключений!**

https://www.postgresql.org/docs/current/plpgsql-control-structures.html

Пример получения доп. информации

```
DO $$
DECLARE
_message text;
BEGIN
        RAISE NOTICE 'res =%', 5/0;
EXCEPTION
    WHEN sqlstate '22012' then
    GET STACKED DIAGNOSTICS _message = message_text;
        RAISE NOTICE E'Ошибка #: % \nТекст %',   sqlstate, _message;
END;
$$;
```

Пример получения доп. информации

```
DO $$
DECLARE
  _hint text ;
  _detail text;
  _message text;
BEGIN
    RAISE exception  using
      message = 'User error',
      detail  = 'При выполнении кода произошла ошибка',
      hint = 'Обратитесь к системному администратору',
      ERRCODE='22011';
EXCEPTION
    WHEN sqlstate '22011'  then
    GET STACKED DIAGNOSTICS  _message = message_text,
                             _detail = pg_exception_detail,
                             _hint = pg_exception_hint;
        RAISE NOTICE E'Ошибка #: % \nТекст  % \nДоп.инф. % \nХинт %',
                        sqlstate, _message, _detail, _hint;
END;
$$;
```

Вложенность блоков

- Обработчик исключения не может возвратить управление обратно в блок
- Чтобы продолжить выполнение, нужно поместить блок кода с обработчиком ошибок в **подблок**

```
DO$$
BEGIN
…
… -- возникло исключение
… -- никогда не выполнится
EXCEPTION
… --обработали
END $$
-- после обработки вышли из блока
```

```
DO$$
BEGIN
    BEGIN
    … -- возникло исключение
    … -- никогда не выполнится
    EXCEPTION
    … --обработали
    END;
… -- после обработки перешли сюда
END $$
```

Распространение исключений из подблоков

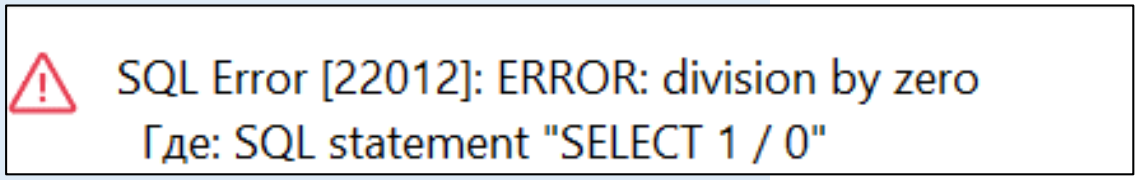
- Поиск обработчика ошибки происходит «изнутри наружу» в порядке вложенности блоков и вызова функций
  - ошибка «поднимается» на уровень выше
  - для получения в обработчике описания стека вызовов в момент исключения - элемент **pg_exception_context** в команде **GET STACKED DIAGNOSTICS**
- Если ни один из обработчиков не сработал:
  - **сообщение об ошибке попадает в журнал сообщений сервера**
  - информация об ошибке передается клиенту, который инициировал вызов кода

```
DO $$
DECLARE
_stack text;
BEGIN
    BEGIN
    --код блока
    RAISE USING
       errcode = '12121',
       message = 'Ошибка внутреннего блока';
    END;
EXCEPTION
    WHEN no_data_found THEN
        RAISE NOTICE 'Сообщение об ошибке';
    WHEN OTHERS then
        GET STACKED diagnostics
            _stack = pg_exception_context;
        RAISE E'[%]: % \n %',
                SQLSTATE, sqlerrm, _stack;
END; $$
```

Обработка ошибок. Замечания

- При возникновении ошибки:
  - выполняется **автоматический откат** к неявной точке сохранения, которая устанавливается в начале блока
  - процедуры лишаются возможности использовать команды **COMMIT** и **ROLLBACK**
- Ошибку невозможно перехватить, если она произошла:
  - в секции **DECLARE**
  - внутри блока **EXCEPTION**

```
DO $$
DECLARE
    _number integer := 1 / 0; -- данная ошибка не перехватывается
BEGIN
    RAISE NOTICE 'Все успешно';
EXCEPTION
    WHEN sqlstate '22012' THEN
        RAISE NOTICE 'Деление на ноль';
END;
$$;
```



Обработка ошибок. Замечания

- Наличие секции **EXCEPTION** в блоке увеличивает накладные расходы
  - на вход/выход из блока
  - установку неявной точки сохранения
  - откат к точке сохранения
- Не нужно стремиться обработать все возможные ошибки в серверном коде
  - бывает полезно передать возникшую ошибку клиенту, если невозможно предусмотреть адекватную обработку в возникшей ситуации
- Если требуется (и возможно) вернуть серверную ошибку с дополнительными пояснениями, можно воспользоваться пользовательским исключением