ГНИВЦ

Парные танцы с курсорами

Использование Python GDB API для исследования внутреннего устройства СУБД (Oracle & PostgreSQL)

Ремизов Дмитрий

ГНИВЦ

План

1. О встроенном в GDB python (hello world)
2. Общие возможности API
3. Пример использования для Oracle RDBMS
4. Пример использования для PostgreSQL RDBMS

ГНИВЦ

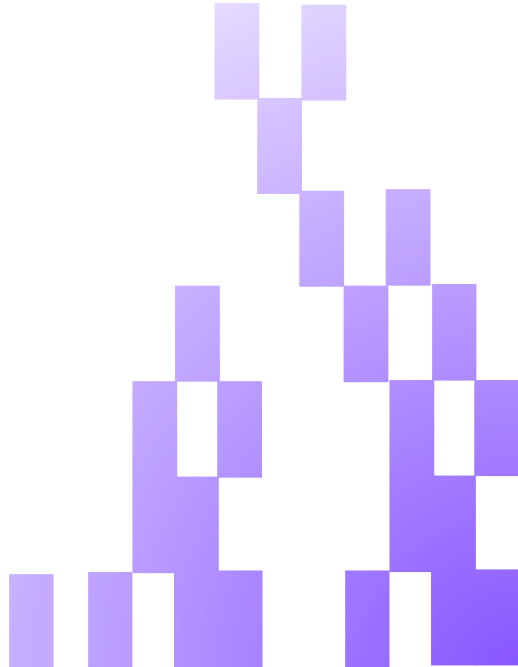
О ВСТРОЕННОМ PYTHON

HELLO WORLD

Встроенная поддержка Python в GDB

Итак, «Hello world»

```
(gdb) python
>print ("Hello world")
>end
Hello world
(gdb)
```



ГНИВЦ

ВОЗМОЖНОСТИ API

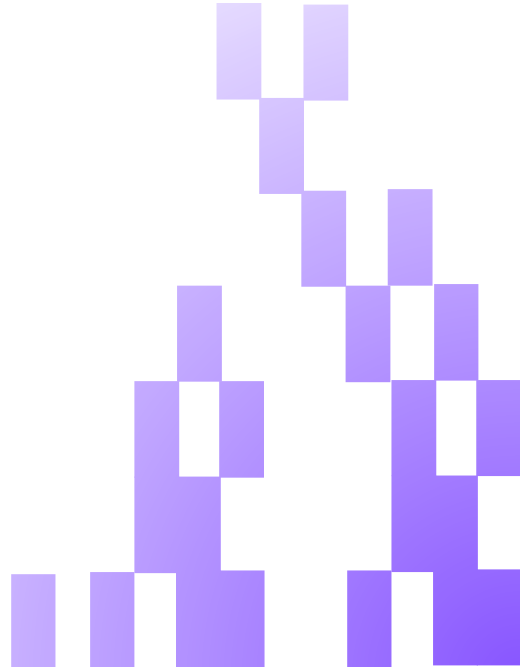
ОЧЕНЬ ОБШИРНЫЕ

Интеграционные возможности Python API

Документация по API

- Basic Python: Basic Python Functions.
- Threading in GDB: Using Python threads in GDB.
- Exception Handling: How Python exceptions are translated.
- Values From Inferior: Python representation of values.
- Types In Python: Python representation of types.
- Pretty Printing API: Pretty-printing values.
- Selecting Pretty-Printers: How GDB chooses a pretty-printer.
- Writing a Pretty-Printer: Writing a Pretty-Printer.
- Type Printing API: Pretty-printing types.
- **Frame Filter API: Filtering Frames.**
- Frame Decorator API: Decorating Frames.
- Writing a Frame Filter: Writing a Frame Filter.

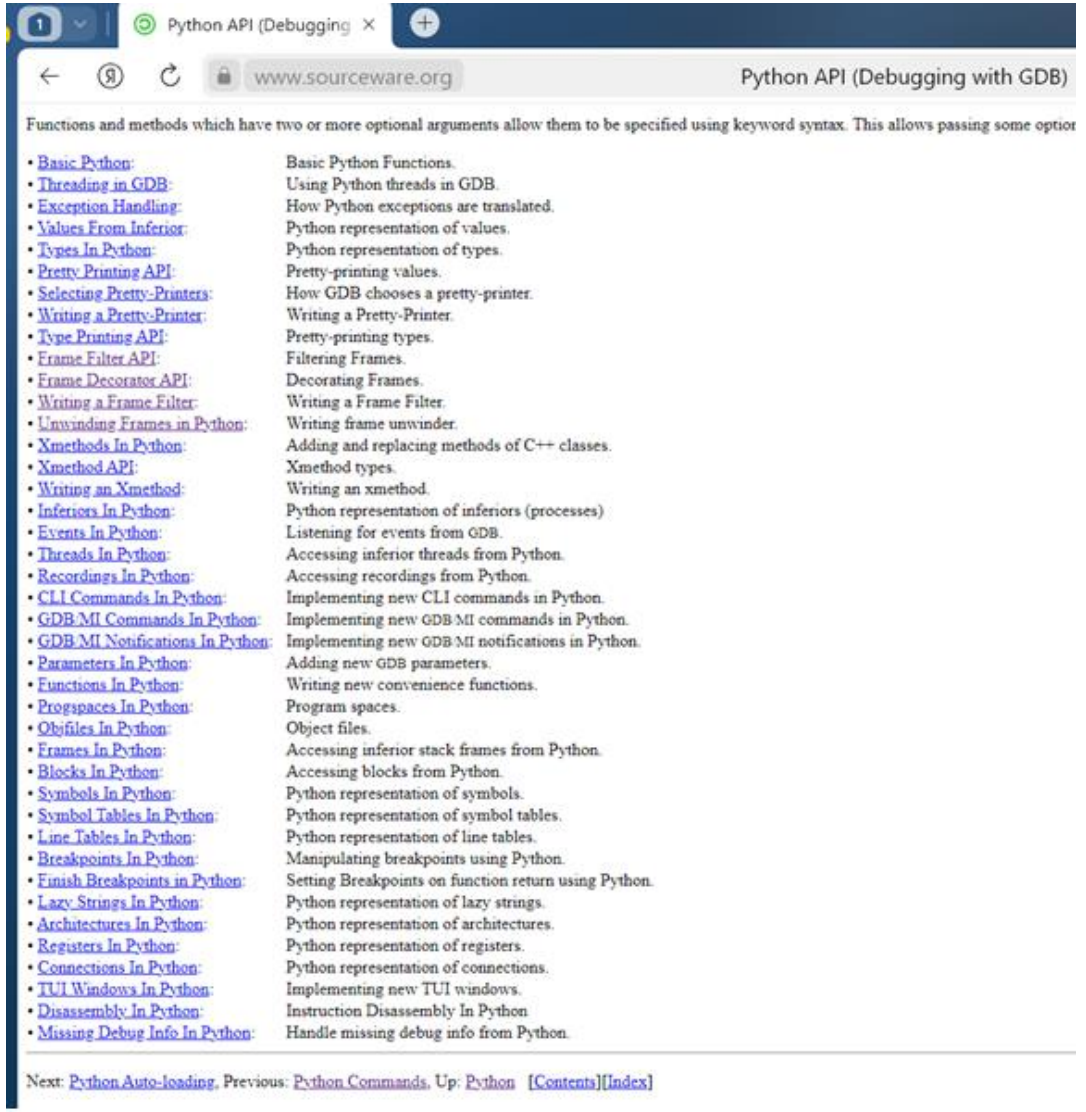
.....



ГНИВЦ

Интеграционные возможности **ГОРАЗДО ШИРЕ**

ОЧЕНЬ МНОГО возможностей



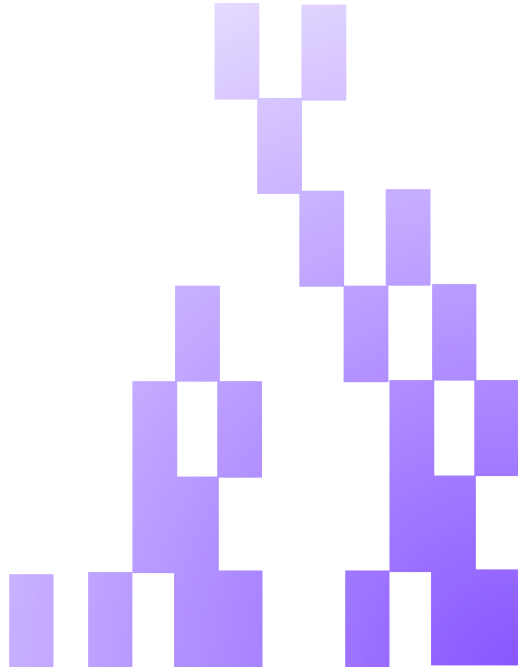
ГНИВЦ

ПИШЕМ FRAME FILTER

Frame filters are Python objects that manipulate the visibility of a frame or frames when a backtrace (see Backtrace) is printed by GDB

Три требования к FrameFilter

- Реализует API
- Само-регистрируется
- Работает с итератором



FrameFilter – пример из документации

Пример FrameFilter, который ничего не делает

```
import gdb

class FrameFilter():

    def __init__(self):
        self.name = "Foo“  # 'name' is the name of the filter that GDB will display.
        self.priority = 100  # 'priority' is the priority of the filter relative to other
        self.enabled = True # 'enabled' is a boolean that indicates whether this filter is enabled and should be executed

        # Register this frame filter with the global frame_filters dictionary.
        gdb.frame_filters[self.name] = self

    def filter(self, frame_iter):
        # Just return the iterator.
        return frame_iter
```

Конструктор

Само-регистрация

Это объект типа итератор его можно и нужно модифицировать

Метод, который ничего не делает (реализация API)

«Тихий» FrameFilter

```
python
import gdb
import itertools

btpy = [] ##a list to keep backtrace

class SilentFrameFilter():
    def __init__(self):
        self.name = "SilentFrameFilter"
        self.priority = 100
        self.enabled = True
        gdb.frame_filters[self.name] = self ## register a new frame filter

    def filter(self, frame_iter):
        del btpy[:]
        for frame in frame_iter:
            name = frame.inferior_frame().name()
            btpy.append(name) #keep names
        return list()

SilentFrameFilter()  ##need to initialize class
end
```

Внутренняя коллекция для хранения стек трейса

Заполняем внутреннюю коллекцию

Возвращаем пустой лист

Всё ещё будут «ошмётки» стандартного вывода

ГНИВЦ

НАКОНЕЦ-ТО СУБДШНАЯ ЧАСТЬ

ORACLE И POSTGRESQL

Что-то вроде дисклаймера

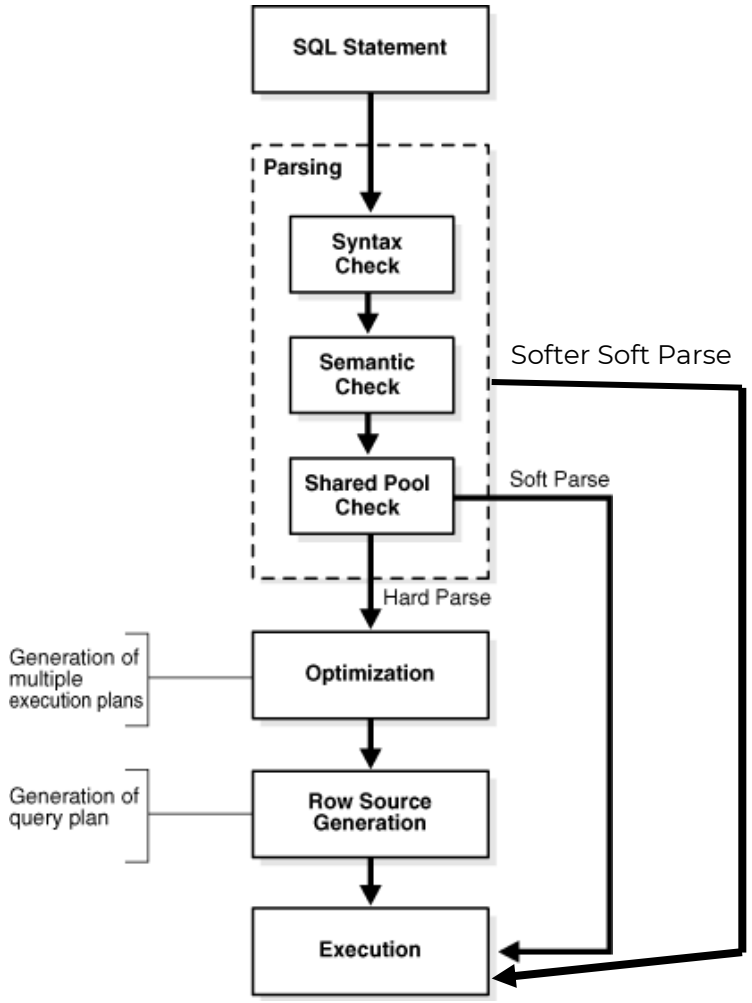
1. Представленные ниже эксперименты делались:
   - на Oracle версии 12.1.0.2 (похожие результаты и на 11.2 и др.)
   - на «ванильном» PostgreSQL версии 16 (скомпилированном без «дебаг» символов)
2. Полученные результаты не являются глубоко оригинальными:
   - пример Dion Cho: Oracle Performance Storyteller и Oracle Core
   - пример Владимир Ситников Postgres
3. Мы сфокусированы на демонстрации метода

ГНИВЦ

НАЧНЁМ С ORACLE

ЗАКРЫТАЯ, ПРОПРИЕТАРНАЯ СИСТЕМА

Этапы процессинга SQL (Oracle)



Тест со стороны сервера

Реализуем FrameFilter

```
python

import gdb
import itertools

btpy = [] ## list to keep backtrace

class SilentFrameFilter():
    def __init__(self):
        self.name = "SilentFrameFilter"
        self.priority = 100
        self.enabled = True
        gdb.frame_filters[self.name] = self ## register a new frame filter

    def filter(self, frame_iter):
        del btpy[:]
        for frame in frame_iter:
            name = frame.inferior_frame().name()
            btpy.append(name) #keep names
        return list()

SilentFrameFilter()  ##need to initialize class
end
```

Очищаем внутренний массив (для нового stack trace)

Сохраняем элементы stack trace во внутренний массив (для дальнейшего использования)

Возвращаем пустой лист, т.е. отфильтровываем/выкидываем весь стэк

Реализуем FrameFilter


```
python

import gdb
import itertools

btpy = [] ## list to keep backtrace

class SilentFrameFilter():
    def __init__(self):
        self.name = "SilentFrameFilter"
        self.priority = 100
        self.enabled = True
        gdb.frame_filters[self.name] = self ## register a new frame filter

    def filter(self, frame_iter):
        del btpy[:]
        for frame in frame_iter:
            name = frame.inferior_frame().name()
            btpy.append(name) #keep names
        return list()

SilentFrameFilter()  ##need to initialize class
end
```

Очищаем внутренний массив (для нового stack trace)

Сохраняем элементы stack trace во внутренний массив (для дальнейшего использования)

Возвращаем пустой лист, т.е. отфильтровываем/выкидываем весь стэк

Тест со стороны сервера

ГНИВЦ

Расставляем breakpoints

```
b kksParseCursor
command
silent
bt
python print('                ')
python print('                ')
python print(' -> '.join([x for x in btpy if not str(x).startswith(('opi','sou','ssth')) ]))
c
end

b kgscFindCursor
command
silent
bt
python print(' -> '.join([x for x in btpy if not str(x).startswith(('opi','sou','ssth')) ]))
c
end

b kkscsSearchChildList
command
silent
bt
python print(' -> '.join([x for x in btpy if not str(x).startswith(('opi','sou','ssth')) ]))
c
end

b kkshGetNextChild
command
silent
bt
python print(' -> '.join([x for x in btpy if not str(x).startswith(('opi','sou','ssth')) ]))
c
end

b kksParseChildCursor
command
……
```

Отключаем весь вывод, за исключением явного

Команда backtrace (которая просто заполнит внутренний python массив)

И, наконец, выводим stack trace в удобном виде

Навигация по librarycache (признак soft parse)

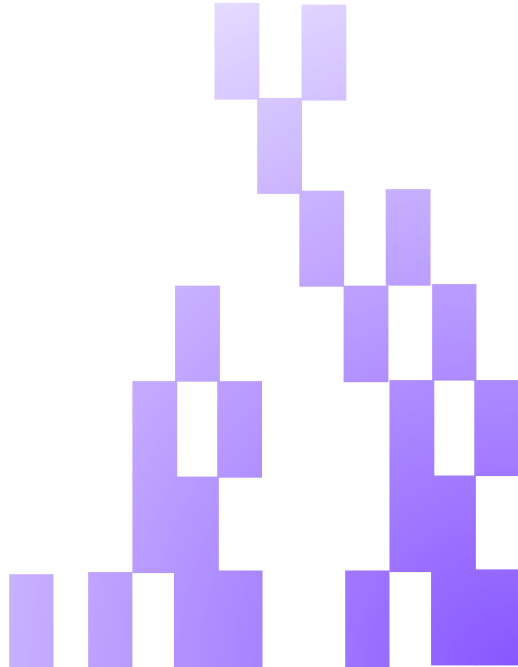
Признак hard parse (создание нового child cursor)

Тест со стороны клиента

JDBC

```
public static void main(String[] args) throws SQLException {

    // Create a OracleDataSource instance and set properties
    OracleDataSource ods = new OracleDataSource();
    ods.setUser("TESTJDBC");
    ods.setPassword("TESTJDBC");
    ods.setURL(JDBCURL);

    // Connect to the database
    Connection conn = ods.getConnection();
    //conn.createStatement().executeQuery("alter session set session_cached_cursors=0");

    execOnce(conn );
    execOnce(conn );
    execOnce(conn );
    execOnce(conn );
    execOnce(conn );
    execOnce(conn );
    execOnce(conn );
    execOnce(conn );
    execOnce(conn );
    execOnce(conn );
    execOnce(conn );
    // Close the connection
    conn.close();
}
```

Управляем кэшированием на стороне сервера (по умолчанию 50 на 19c)



Тест со стороны клиента

JDBC

```
private static void execOnce(Connection conn) throws SQLException {

    PreparedStatement preparedStatement = conn.prepareStatement("select id, load from T_SIMPLE where id >?");

    ((OraclePreparedStatement)preparedStatement).setDisableStmtCaching (true);

    preparedStatement.setInt(1, 1);

    // итерация по результату
    ResultSet rset = preparedStatement.executeQuery("select id, "
            + "load from T_SIMPLE");

    while (rset.next())
        System.out.println(rset.getString(1) + " " + rset.getString(2));

    // Close the ResultSet
    rset.close();
    // Close the Statement
    preparedStatement.close();
}
```

Отключаем/не отключаем механизм автоматического кэширования SQL выражений на стороне драйвера

Закроется реально или не закроется – зависит от механизма кэширования выше

На самом деле в рамках этого теста – отключаем.

Если не отключать – это “no parse” выполнение

ГНИВЦ

РЕЗУЛЬТАТЫ И ИНТЕРПРЕТАЦИЯ

САМОЕ ВКУСНОЕ☺

Результат трейсинга кэширования сессии Oracle с включённым **session_cached_cursors** (по умолчанию) и **setDisableStmtCaching** (true), чтобы исключить «no parse» execution.
После 2-го исполнения начинает инкрементиться статистика "session cursor cache hits"

kksParseCursor -> kpoal8 -> ttcpip -> main
kgscFindCursor -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main
kkscsSearchChildList -> kksfbc -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main
kkshGetNextChild -> kkscsSearchChildList -> kksfbc -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main

kksParseCursor -> kpoal8 -> ttcpip -> main
kgscFindCursor -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main
kkscsSearchChildList -> kksfbc -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main
kkshGetNextChild -> kkscsSearchChildList -> kksfbc -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main

kksParseCursor -> kpoal8 -> ttcpip -> main
kgscFindCursor -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main

kksParseCursor -> kpoal8 -> ttcpip -> main
kgscFindCursor -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main

kksParseCursor -> kpoal8 -> ttcpip -> main
kgscFindCursor -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main

kksParseCursor -> kpoal8 -> ttcpip -> main
kgscFindCursor -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main

kksParseCursor -> kpoal8 -> ttcpip -> main
kgscFindCursor -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main

kksParseCursor -> kpoal8 -> ttcpip -> main
kgscFindCursor -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main

kksParseCursor -> kpoal8 -> ttcpip -> main
kgscFindCursor -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main

kksParseCursor -> kpoal8 -> ttcpip -> main
kgscFindCursor -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main

kksParseCursor -> kpoal8 -> ttcpip -> main
kgscFindCursor -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main

Навигируем по library cache ищем подходящий child

Переключаемся в режим «softer soft parse»

Hard parse произошёл гораздо раньше в другой сессий, в library cache есть подходящий курсор

ГНИВЦ

РЕЗУЛЬТАТЫ И ИНТЕРПРЕТАЦИЯ

ОТКЛЮЧАЕМ МЕХАНИЗМ (ЧЕРЕЗ ALTER SESSION SET SESSION_CACHED_CURSORS=0)

Все исполнения **равнозначны,** "session cursor cache hits" не **инкрементится**.

kksParseCursor -> kpoal8 -> ttcpip -> main
kkscsSearchChildList -> kksfbc -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main
kkshGetNextChild -> kkscsSearchChildList -> kksfbc -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main

kksParseCursor -> kpoal8 -> ttcpip -> main
kkscsSearchChildList -> kksfbc -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main
kkshGetNextChild -> kkscsSearchChildList -> kksfbc -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main

kksParseCursor -> kpoal8 -> ttcpip -> main
kkscsSearchChildList -> kksfbc -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main
kkshGetNextChild -> kkscsSearchChildList -> kksfbc -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main

....

....

kksParseCursor -> kpoal8 -> ttcpip -> main
kkscsSearchChildList -> kksfbc -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main
kkshGetNextChild -> kkscsSearchChildList -> kksfbc -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main

kksParseCursor -> kpoal8 -> ttcpip -> main
kkscsSearchChildList -> kksfbc -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main
kkshGetNextChild -> kkscsSearchChildList -> kksfbc -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main

kksParseCursor -> kpoal8 -> ttcpip -> main
kkscsSearchChildList -> kksfbc -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main
kkshGetNextChild -> kkscsSearchChildList -> kksfbc -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main

kksParseCursor -> kpoal8 -> ttcpip -> main
kkscsSearchChildList -> kksfbc -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main
kkshGetNextChild -> kkscsSearchChildList -> kksfbc -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main

kksParseCursor -> kpoal8 -> ttcpip -> main
kkscsSearchChildList -> kksfbc -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main
kkshGetNextChild -> kkscsSearchChildList -> kksfbc -> kkspsc0 -> kksParseCursor -> kpoal8 -> ttcpip -> main

Hard parse произошёл гораздо раньше в другой сессий, в library cache есть подходящий курсор

ГНИВЦ

И НАКОНЕЦ POSTGRESQL

OPEN SOURCE И ВСЁ ВСЁ ВСЁ ☺

Что инструментируем?

Где расставляем breakpoints

1. Инструментируем основные фазы исполнения SQL (Query execution stages):
   - Parse
   - Rewrite (transformation)
   - Plan
   - Execute

2. Как определяем точки для breakpoints (мониторинга)

   Всё уже сделано за нас ☺
   В исходном коде PostgreSQL расставлены так называемые USDT (User Statically-Defined Tracing) пробы. Например:

```
TRACE_POSTGRESQL_QUERY_PARSE_START(query_string); --в pg_parse_query
```

Тест со стороны сервера

ГНИВЦ

Расставляем breakpoints

```
break pg_parse_query
command
silent
bt
python print(' -> '.join([x for x in btpy if not str(x).startswith(('PostmasterMain','ServerLoop','PostgresMain')) ]))
c
end

break pg_rewrite_query
command
silent
bt
python print(' -> '.join([x for x in btpy if not str(x).startswith(('PostmasterMain','ServerLoop','PostgresMain')) ]))
c
end

break pg_plan_query
command
silent
bt
python print(' -> '.join([x for x in btpy if not str(x).startswith(('PostmasterMain','ServerLoop','PostgresMain')) ]))
c
end


break PortalRun
command
silent
bt
python print(' -> '.join([x for x in btpy if not str(x).startswith(('PostmasterMain','ServerLoop','PostgresMain')) ]))
c
end
```

Фаза parse

Фаза rewrite

Фаза plan

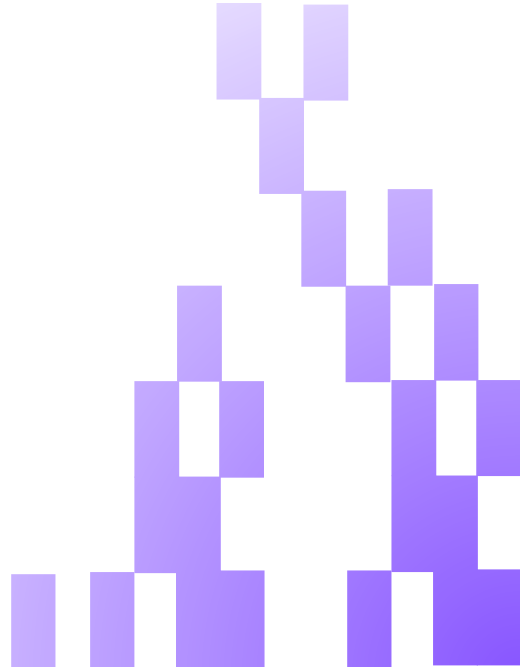
Фаза execute

FrameFilter абсолютно идентичный

Тест со стороны клиента

JDBC

```
public class Main {
public static final String JDBCURL = "jdbc:postgresql://158.160.77.253:5432/postgres";


    public static void main(String[] args) throws SQLException {
        Properties props = new Properties();
        props.setProperty("user", "testjdbc");
        props.setProperty("password", "TESTJDBC");
        // Connect to the database
        Connection conn = DriverManager.getConnection(JDBCURL, props);

        execOnce(conn);
        execOnce(conn);
        execOnce(conn);
        execOnce(conn);
        execOnce(conn);
        execOnce(conn);
        execOnce(conn);
        execOnce(conn);
        execOnce(conn);
        execOnce(conn);
        execOnce(conn);
        execOnce(conn);
        // Close the connection
        conn.close();

    }
```



Тест со стороны клиента

JDBC

```
private static void execOnce(Connection conn) throws SQLException {
    PreparedStatement preparedStatement = conn.prepareStatement("select id, load from T_SIMPLE where id >=?");

    //org.postgresql.PGStatement pgstmt = preparedStatement.unwrap(org.postgresql.PGStatement.class);
    //pgstmt.setPrepareThreshold(1);

    preparedStatement.setInt(1, 1);

    ResultSet rset = preparedStatement.executeQuery();

    // Iterate through the result
    while (rset.next())
        System.out.println(rset.getString(1) + " " + rset.getString(2));
    // Close the ResultSet
    rset.close();
    // Close the Statement
    preparedStatement.close();
}
```

Управление механизмом кэширования

ГНИВЦ

РЕЗУЛЬТАТЫ И ИНТЕРПРЕТАЦИЯ

ВСЁ ПО УМОЛЧАНИЮ

--выполнение с PrepareThreshold по умолчанию (5 раз, на 6-ой)

pg_parse_query -> main
pg_rewrite_query -> main
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_parse_query -> main
pg_rewrite_query -> main
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_parse_query -> main
pg_rewrite_query -> main
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_parse_query -> main
pg_rewrite_query -> main
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_parse_query -> main
pg_rewrite_query -> main
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

---закэшировали курсор

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

PortalRun -> main

5

Parse & rewrite фаза включена

Результаты и интерпретация

PrepareThreshold=1

-- выполнение с server side prepared statements с явно выставленным
-- pgstmt.setPrepareThreshold(1);

pg_parse_query -> main
pg_rewrite_query -> main
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main
---закэшировали курсор
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

PortalRun -> main
PortalRun -> main
PortalRun -> main
PortalRun -> main
PortalRun -> main

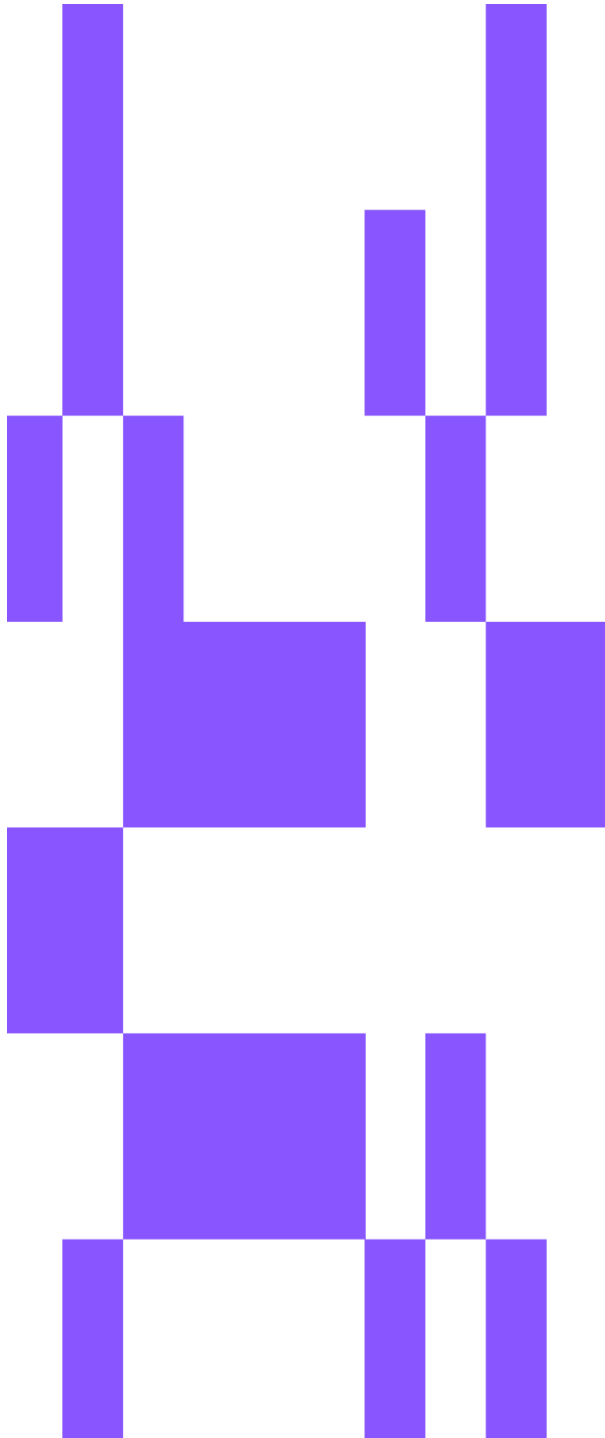
1

Parse & rewrite фаза включена

ГНИВЦ

ФОКУС НА 2 МЕХАНИЗМ

PrepareThreshold=5 (по умолчанию)



--выполнение с PrepareThreshold по умолчанию (5 раз, на 6-ой)
pg_parse_query -> main
pg_rewrite_query -> main
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_parse_query -> main
pg_rewrite_query -> main
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_parse_query -> main
pg_rewrite_query -> main
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_parse_query -> main
pg_rewrite_query -> main
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_parse_query -> main
pg_rewrite_query -> main
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main
---закэшировали курсор
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

PortalRun -> main

5

Только plan & execute

ГНИВЦ

ФОКУС НА 2 МЕХАНИЗМ

PrepareThreshold=1

-- выполнение с server side prepared statements с явно выставленным
-- pgstmt.setPrepareThreshold(1);

pg_parse_query -> main
pg_rewrite_query -> main
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main
---закэшировали курсор
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

PortalRun -> main
PortalRun -> main
PortalRun -> main
PortalRun -> main
PortalRun -> main

1 — Parse & rewrite фаза включена

5 — Только plan & execute

Только execute

ГНИВЦ

ФОКУС НА 2 МЕХАНИЗМ

PrepareThreshold=2

-- выполнение с server side prepared statements с явно выставленным
-- pgstmt.setPrepareThreshold(2);

pg_parse_query -> main
pg_rewrite_query -> main
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_parse_query -> main
pg_rewrite_query -> main
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

PortalRun -> main
PortalRun -> main
PortalRun -> main
PortalRun -> main

PortalRun -> main
PortalRun -> main
PortalRun -> main
PortalRun -> main
PortalRun -> main

2

5

Только plan & execute

ГНИВЦ

ФОКУС НА 2 МЕХАНИЗМ

PrepareThreshold=3

-- выполнение с server side prepared statements с явно выставленным
-- pgstmt.setPrepareThreshold(3);

pg_parse_query -> main
pg_rewrite_query -> main
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_parse_query -> main
pg_rewrite_query -> main
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_parse_query -> main
pg_rewrite_query -> main
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

3

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

5

Только plan & execute

PortalRun -> main
PortalRun -> main
PortalRun -> main
PortalRun -> main

PortalRun -> main
PortalRun -> main
PortalRun -> main
PortalRun -> main
PortalRun -> main

ЧТО ЖЕ ЭТО ЗА МЕХАНИКА?

ГИПОТЕЗА

plan_cache_mode = (**auto**|force_custom_plan|force_generic_plan)

Модифицируем тест

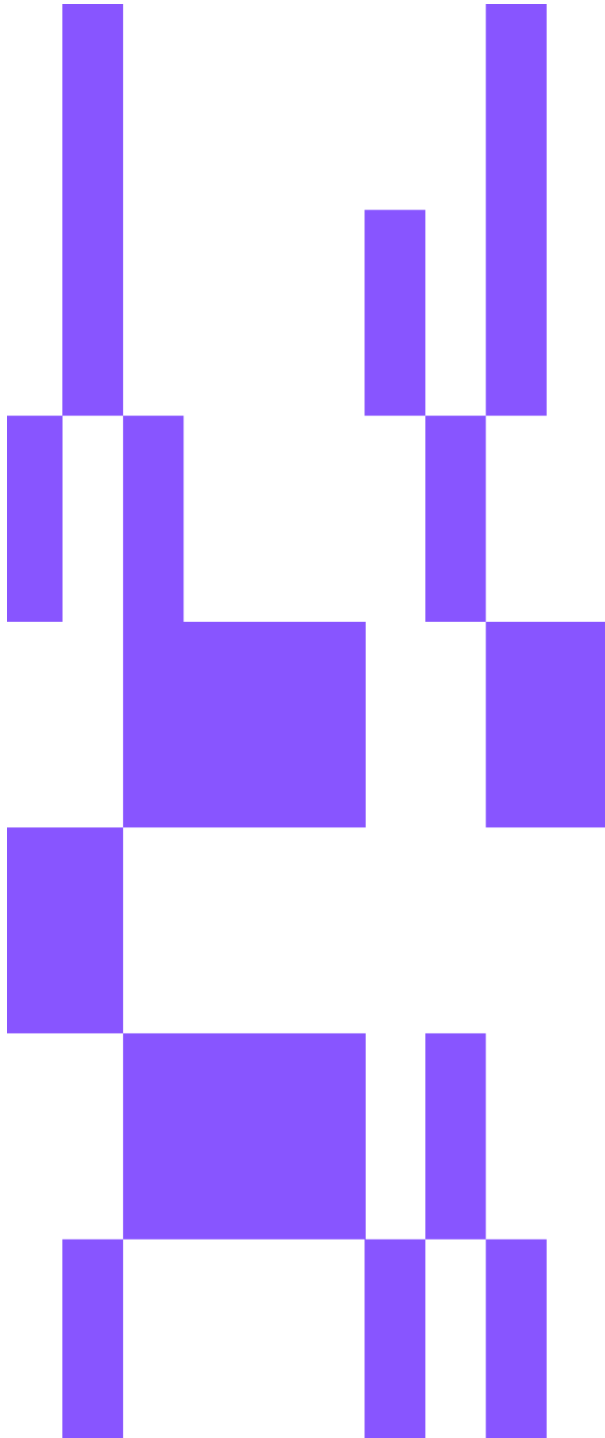
PrepareThreshold=1

```
    public class Main {
public static final String JDBCURL = "jdbc:postgresql://51.250.23.87:5432/postgres";

    public static void main(String[] args) throws SQLException {

        Properties props = new Properties();
        props.setProperty("user", "testjdbc");
        props.setProperty("password", "TESTJDBC");
        Connection conn = DriverManager.getConnection(JDBCURL, props);
        conn.createStatement().execute("set plan_cache_mode=force_generic_plan");
        //conn.createStatement().execute("set plan_cache_mode=force_custom_plan");

        execOnce(conn);
        execOnce(conn);
        execOnce(conn);
        execOnce(conn);
        execOnce(conn);
        execOnce(conn);
        execOnce(conn);
        execOnce(conn);
        execOnce(conn);
        execOnce(conn);
        execOnce(conn);
        // Close the connection
        conn.close();
    }
```

```
 private static void execOnce(Connection conn) throws SQLException {
        PreparedStatement preparedStatement = conn.prepareStatement("select id, load from T_SIMPLE
where id >=?");

        org.postgresql.PGStatement pgstmt = preparedStatement.unwrap(org.postgresql.PGStatement.class);
        pgstmt.setPrepareThreshold(3);

        preparedStatement.setInt(1, 1);
```

FORCE_GENERIC_PLAN

PrepareThreshold=3



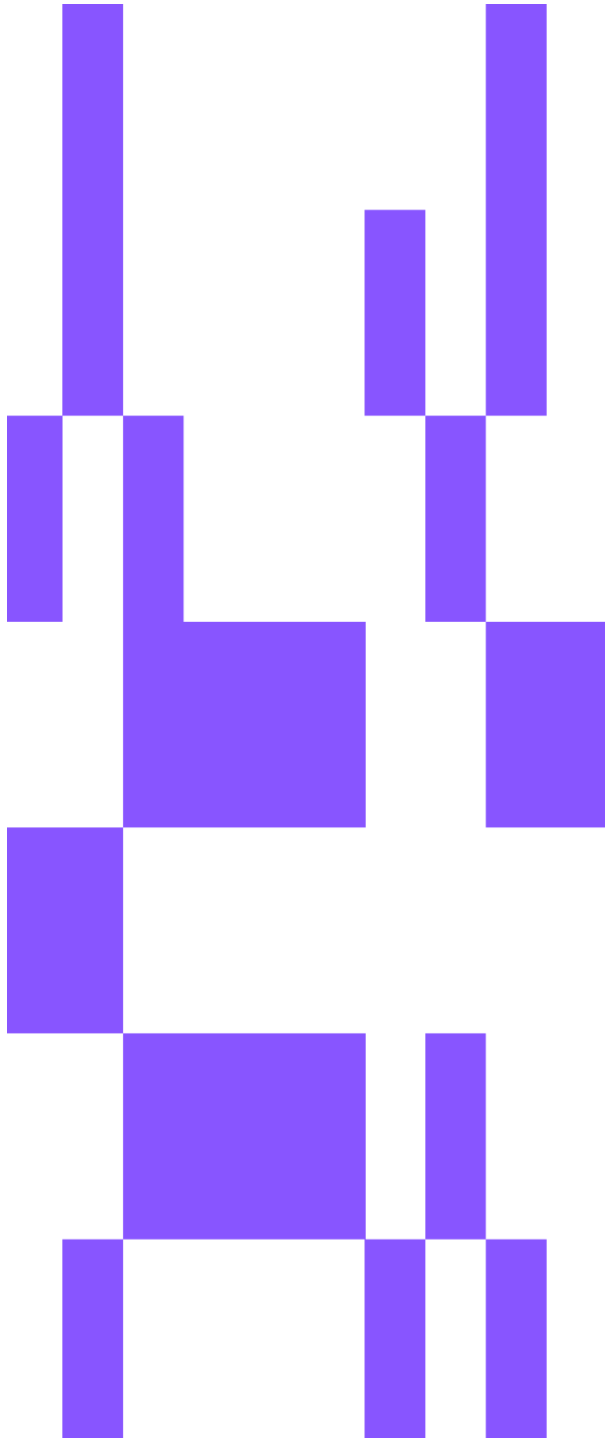
```
 -- выполнение с server side prepared statements с явно выставленным
-- pgstmt.setPrepareThreshold(3);
-- (force generic plan):
pg_parse_query -> main
pg_rewrite_query -> main
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_parse_query -> main
pg_rewrite_query -> main
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_parse_query -> main
pg_rewrite_query -> main
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

PortalRun -> main
PortalRun -> main
PortalRun -> main
PortalRun -> main
PortalRun -> main
PortalRun -> main
PortalRun -> main
PortalRun -> main
PortalRun -> main
```

3

Нет этапа
«только plan & execute»
сразу только execute

FORCE_CUSTOM_PLAN

PrepareThreshold=3



-- выполнение с server side prepared statements с явно выставленным
-- pgstmt.setPrepareThreshold(3); (force custom plan):

СНИВЦ

pg_parse_query -> main
pg_rewrite_query -> main
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_parse_query -> main
pg_rewrite_query -> main
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_parse_query -> main
pg_rewrite_query -> main
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

3

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

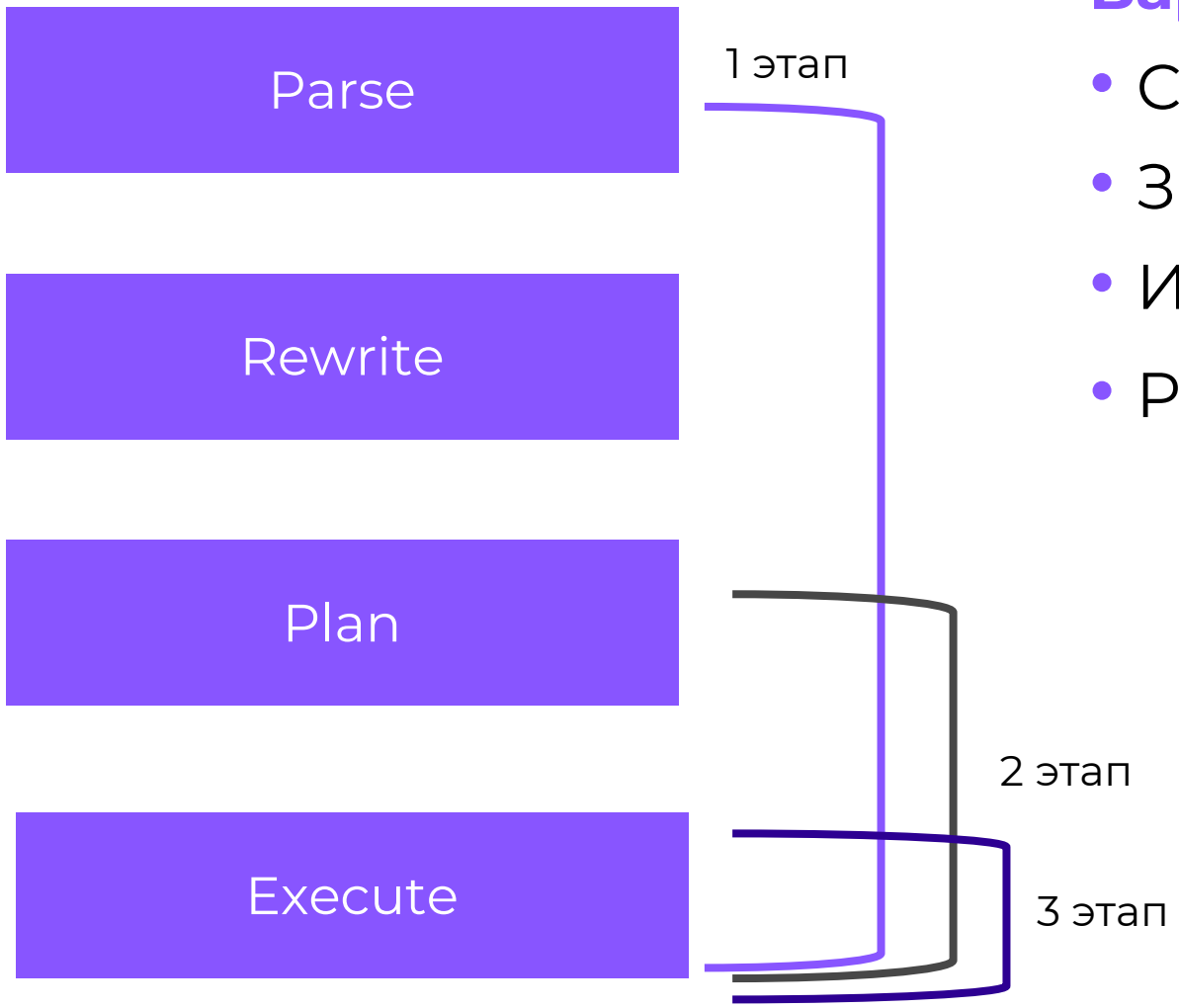
pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

pg_plan_query -> pg_plan_queries -> BuildCachedPlan -> GetCachedPlan -> main
PortalRun -> main

Нет этапа
Только execute

Что узнали



Варианты исполнения SQL

- Сначала Parse-Rewrite-Plan-Execute
- Затем Plan-Execute
- И наконец Execute
- Postgres очень любит число 5 ☺

Управляется через:

- PrepareThreshold
- plan_cache_mode

ГНИВЦ

Заключение

1. Мы увидели удивительное сходство(и отличия) механизмов исполнения SQL на двух довольно различных СУБД.
   Т.е. механизмы динамической адаптации, оптимизации, способность выкидывать большие куски кода при повторном исполнении SQL.
2. Мы продемонстрировали метод, который позволяет наглядно визуализировать эти механизмы
3. Знание данных механизмов может помочь вам оптимизировать ваше приложение, в реальной жизни ситуация будет гораздо сложнее, кеши на стороне сервера и клиента не бесконечны, в том числе вам есть что «по-архитектурить».

Для тех, кто решит поэкспериментировать самостоятельно

1. Удобно положить весь gdb'шный код в файлик .gdbinit в home-директории вашего юзера исполняющего gdb. В этом случае код загрузится автоматом при старте gdb.

2. Нужно добавить следующие инструкции в начале файлика:
   - set auto-load safe-path /
   - set breakpoint pending on – для того, чтобы breakpoint'ы создались без проблем (в pending)

Ремизов Дмитрий

Лидер Центра Компетенций Базы Данных
25+ лет в индустрии
Работал с Oracle, PostgreSQL, Sybase, MSSQL .....
Сертификации Oracle и MSSQL
Участник групп RuOUG и BGOUG
Спикер на международных конференциях

Вопросы?

Extra materials (to not present)

1. https://dev.to/yugabyte/postgres-query-execution-jdbc-prepared-statements-51e2
2. https://www.npgsql.org/doc/api/Npgsql.NpgsqlCommand.html
   .Net prepare statement