**ODBC log / trace parser**

Парсер предназначен для извлечения SQL-запросов и автоматической подстановки параметров из ODBC-лога или трассы. Для работы нужно скопировать фрагмент лога/трассы в файл log.txt и запустить скрипт (необходим Python 3): py process\_log.py

Парсер работает с двумя форматами:

### Формат ODBC-лога:

Строки начинаются с ‘[’, блок содержит строки с маркерами:

[2023-Nov-17 17:13:54.323608] <NOTE> **NSQL**:  исходный запрос, который приходит из прикладного кода. (NATIVE\_SQL)

[2023-Nov-17 17:13:54.408382] <NOTE> **PHSQL**:  запрос, обработанный конвертером, там пронумерованы placeholder. (PLACEHOLDER\_SQL)

[2023-Nov-17 17:13:54.416360] <NOTE> **bind map**:  карта placeholder

[2023-Nov-17 17:13:54.416360] <NOTE> **SQL**:  запрос pgsql из конвертера, обработанный “перестановщиком placeholder”, все placeholder заменены на ‘?’

[2023-Nov-17 17:13:54.417357] <NOTE> **to RECORD**:  строки с параметрами.

### Формат трассы:

Первая строка трассы начинается с любого символа, кроме ‘[’. Для парсинга Oracle-трассы следует на первой строке входного файла log.txt поставить строку 'ORACLE'. Расположение полей в трассе у разных пользователей может различаться, в связи с этим может потребоваться доработка парсера.

Пример форматов трассы, с которой работает эта реализация.

#1, type RSDLONG, value: 1 RsdCmdSp.cpp (line 1710) Parameter\Input 4 RSDODBC 12:22:45.955 20-11-2023

или:

RSDODBC Parameter\Input #1, type RSDLONG, value: 505 4 11:40:27.029 23-11-2023 RsdCmdSp.cpp (line 1710)

Строка с параметром определяется по наличию маркера 'Parameter'

#1, type RSDSHORT, value: -32768 RsdCmdSp.cpp (line 1561) Parameter\Input

Тело запроса считывается со строки, следующей за последним параметром:

RSDODBC Parameter\Input #15, type RSDDATE, value: 8.12.2025 4   
RSDODBC SQL\Execute select t.t\_infoid, t.t\_direct, t.t\_trn, t.t\_r 1

Запрос считывается до строки, содержащей Result=

RSDODBC SQL\Execute Result=

При парсинге трассы скрипт обрабатывает все встреченные блоки #1 ... Result= , последовательно сохраняя все найденные запросы в выходной файл.

### Начало работы:

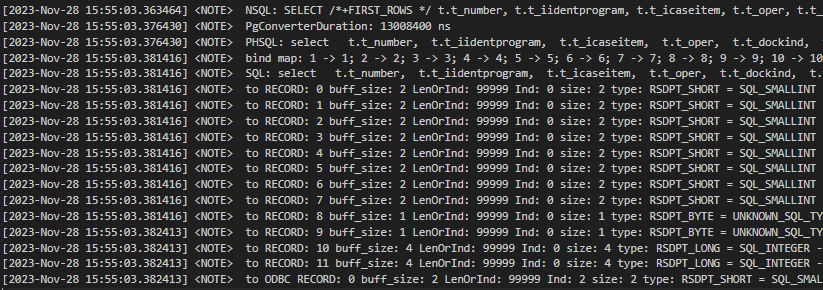
Для начала работы следует убедиться, что в системе установлен Python 3. Для этого в командной строке выполните команду

**python –V**

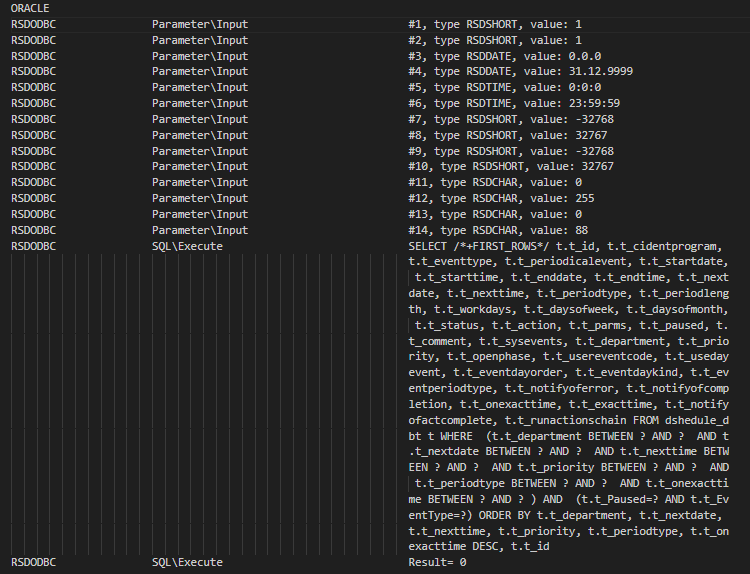
**Python 3.11.3**

Если не установлен – с [официального сайта](https://www.python.org/downloads/windows/) берём подходящий релиз Python 3 (для Windows 7 это будет Python 3.8.6, для более новых ОС - Python 3.12.0) и устанавливаем.

После этого нужно сохранить скрипт и файл log.txt в отдельную папку. В log.txt скопируйте данные для парсинга. При этом обратите внимание, что фрагмент ODBC лога должен содержать 1 запрос, начинаться с символа “[”, содержать маркеры NSQL, PHSQL, bind map и параметры запроса с маркерами to RECORD, число параметров совпадает с bind map. Поддерживаются типы RSDPT\_CHAR, RSDPT\_LPSTR, RSDPT\_SHORT, RSDPT\_LONG, RSDPT\_TIMESTAMP, RSDPT\_DATE, RSDPT\_TIME. Встретив неизвестный тип, парсер обрабатывает запрос, оставляя “?” вместо этого параметра.



Фрагмент трассы начинается со строки “ORACLE”, если это нужно, далее идёт перечисление параметров. Поддерживаются RSDSHORT, RSDLONG, RSDCHAR, RSDLPSTR, RSDTIMESTAMP, RSDDATE, RSDTIME. Неизвестные типы игнорируются, значения этих типов не подставляются в запрос. Сразу после параметров парсер начинает читать текст запроса, чтение запроса заканчивается на строке с “ Result= ”

Обратите внимание, что в копируемом фрагменте трассы может быть несколько блоков с параметрами и запросами. Все они будут обработаны и последовательно сохранены в выходном файле.   


Далее запускается скрипт

**py process\_log.py**

В консоль выводятся найденные параметры и запросы, а обработанные результаты сохраняются в файлы:

req\_NSQL.sql – исходный запрос с пронумерованными плейсхолдерами;

req\_OR\_PARAMS.sql – исходный запрос с подстановкой параметров;

req\_PG\_PARAMS.sql – запрос Postgres с подстановкой параметров;

req\_PHSQL.sql - запрос Postgres с пронумерованными плейсхолдерами;

req\_TRACE\_PARAMS.sql – запросы из трассы с подстановкой параметров.

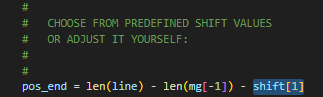
После того, как скрипт отработал, запросы можно скопировать или открыть в toad, dbeaver или других менеджерах БД. Для меня, однако, наиболее удобно работать в простом VSCode, где выполняются все операции – редактирование лога, запуск скрипта, при необходимости правки получившихся запросов и их запуск (при наличии настроенного соединения VSCode с БД при помощи, например, [SQLTools](https://marketplace.visualstudio.com/items?itemName=mtxr.sqltools))

#### Примечание:

Поскольку формат трассы может различаться у разных пользователей, может потребоваться в функции **find\_req\_trace()** подобрать параметр **shift**. Его значение показывает, сколько символов разделяют в трассе поле с запросом и следующее за ним:



По умолчанию заданы 2 значения - 3 и 7, соответствующие приведённым выше примерам трассы. Переключение между ними происходит в строке:



Для данного примера shift[1] = 7. Можно подставить своё значение в текст скрипта.

Разумеется, скрипт не полностью универсален,