



DBeaver: PostgreSQL IDE и отладчик с открытым исходным кодом

АЛЕКСАНДР ФЕДОРОВ И АНДРЕЙ ХИТРИН

5-7 февраля 2018

PGConf.Russia 2018



КОРОТКО О DBEAVER

DBeaver



Разрабатывается
с 2011



Open source



Активное сообщество
на GitHub

Пользователи



800 000+
пользователей



100+ стран



Разработчики, DBA,
аналитики данных,
службы поддержки

Разработчики



5 основных
разработчиков



30+ постоянных
разработчиков



Команда,
распределенная
по всему миру

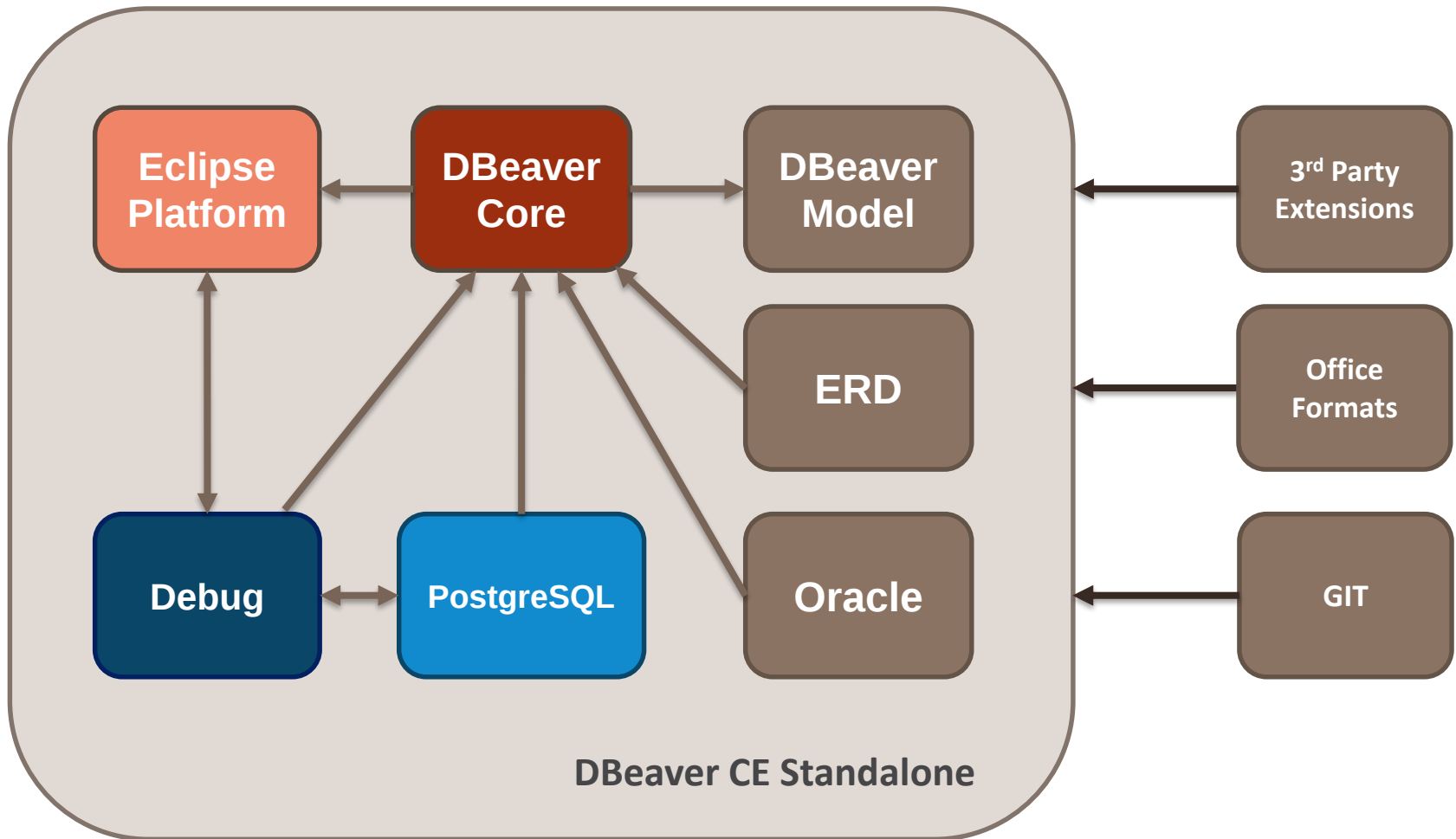


ЭКОСИСТЕМА DBEAVER



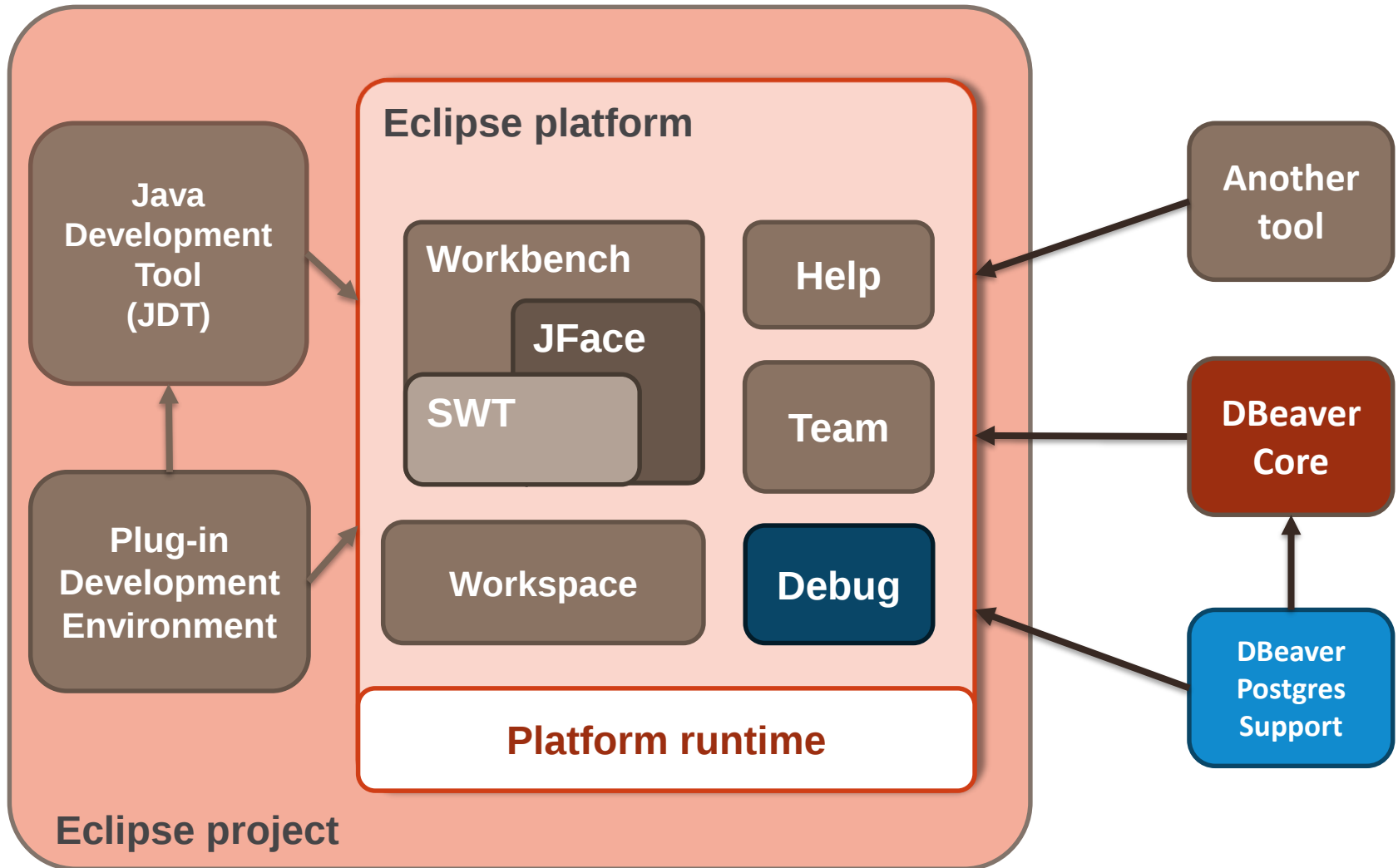


КОМПОНЕНТЫ DBEAVER





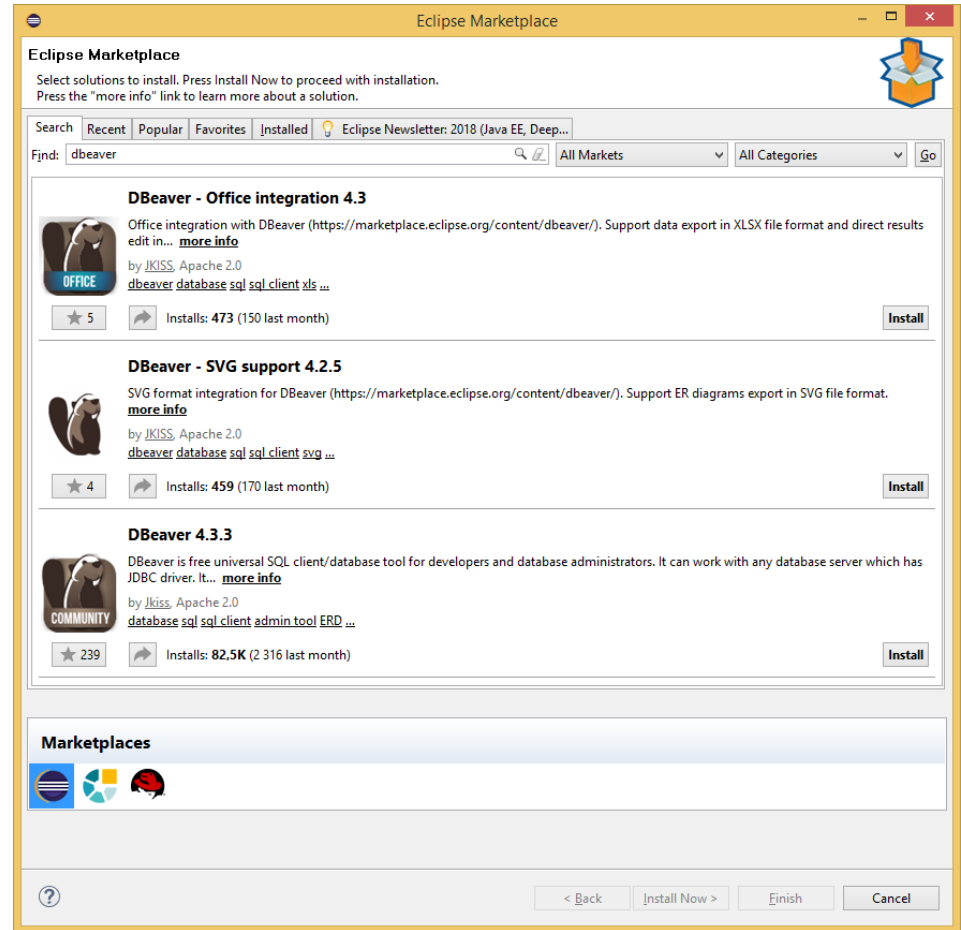
ИНФРАСТРУКТУРА ECLIPSE





ECLIPSE: ПРИЛОЖЕНИЕ VS ПЛАГИН

- Отдельное приложение:
установка в качестве отдельного приложения, построенного на базе Eclipse RCP
(полнофункциональной клиентской платформы)
- Плагин для IDE – установка в уже развернутый пакет Eclipse, такой как «Eclipse IDE for C/C++ Developers», с помощью клиента Eclipse Marketplace





ОСНОВНЫЕ ФУНКЦИИ DBeaver

/ 1 Администрирование баз данных

Широкий спектр функций для поддержки и анализа баз данных, экспорт и импорт данных, работа с пользовательскими сессиями и многое другое

/ 2 Анализ базы данных

Визуализация индивидуальных объектов базы и схемы базы в целом, изменение схемы, экспорт данных и метаданных

/ 3 Редактирование данных

Простой и интуитивно понятный интерфейс для просмотра, редактирования, сортировки и фильтрации данных

/ 4 SQL Development

Полнофункциональный и быстрый SQL-редактор для написания, применения, сохранения, экспорта и переиспользования скриптов с профилировщиком данных и функцией форматирования



СПЕЦИАЛЬНЫЕ ФУНКЦИИ DBEAVER ДЛЯ POSTGRES

Помимо универсальных функций, DBeaver поддерживает и специальные функции для Postgres:

- ⦿ PGSQL отладчик
- ⦿ Менеджер блокировок
- ⦿ Менеджер сессий
- ⦿ Бэкап и восстановление
- ⦿ Поддержка пользовательских типов данных Postgres
- ⦿ Анализ, Vacuum и многое другое



СПЕЦИАЛЬНЫЕ ФУНКЦИИ POSTGRES В ИНТЕРФЕЙСЕ DBEAVER

The image displays a collage of DBeaver interface windows for PostgreSQL, illustrating various database management functions.

Vacuum database: Shows options for vacuuming, including ☒ Full, ☐ Freeze, ☒ Analyze, and ☐ Disable page skipping.

Restore settings: Displays settings for restoring a database, including Format (Custom) and Output folder (C:\tmp).

Backup settings: Shows backup configuration, including Format (Custom), Compression (4), Encoding (UTF-8), and Output folder (C:\tmp).

SQL Preview: Displays the SQL command: `VACUUM (VERBOSE, FULL, ANALYZE)`.

Choose objects to export: Shows a list of objects to be exported, including Schemas, Tables, and Views.

Backup progress: Shows the progress of the backup operation, including the command: `C:\PostgresPro\10\bin\pg_dump.exe --format=c --compress=3 --encoding=windows-1251 -n public --verbose --host=10.0.3.36 --port=5432`.

Object Message: Displays a list of objects and their status, including:

- tbl: found 0 removable, 1 nonremovable row versions in 1 pages
- analyzing "public.tbl"
- tbl: scanned 1 of 1 pages, containing 1 live rows and 0 dead rows; 1 rows in sample, 1 estimated total rows
- vacuuming "pg_catalog.pg_statistic"
- pg_statistic: found 7 removable, 387 nonremovable row versions in 16 pages
- vacuuming "pg_catalog.pg_type"
- pg_type: found 6 removable, 374 nonremovable row versions in 9 pages
- analyzing "pg_catalog.pg_type"
- pg_type: scanned 9 of 9 pages, containing 372 live rows and 2 dead rows; 372 rows in sample, 372 estimated total rows
- vacuuming "pg_catalog.pg_authid"
- pg_authid: found 3 removable, 6 nonremovable row versions in 1 pages
- analyzing "pg_catalog.pg_authid"
- pg_authid: scanned 1 of 1 pages, containing 6 live rows and 0 dead rows; 6 rows in sample, 6 estimated total rows
- vacuuming "pg_catalog.pg_user_mapping"
- pg_user_mapping: found 0 removable, 0 nonremovable row versions in 0 pages
- analyzing "pg_catalog.pg_user_mapping"
- pg_user_mapping: scanned 0 of 0 pages, containing 0 live rows and 0 dead rows; 0 rows in sample, 0 estimated total rows
- vacuuming "pg_catalog.pg_attribute"
- pg_attribute: found 25 removable, 2498 nonremovable row versions in 47 pages
- analyzing "pg_catalog.pg_attribute"
- pg_attribute: scanned 46 of 46 pages, containing 2471 live rows and 27 dead rows; 2471 rows in sample, 2471 estimated total rows
- vacuuming "pg_catalog.pg_proc"
- pg_proc: found 0 removable, 2875 nonremovable row versions in 74 pages
- analyzing "pg_catalog.pg_proc"
- pg_proc: scanned 74 of 74 pages, containing 2875 live rows and 0 dead rows; 2875 rows in sample, 2875 estimated total rows
- vacuuming "pg_catalog.pg_class"
- pg_class: found 14 removable, 325 nonremovable row versions in 11 pages
- analyzing "pg_catalog.pg_class"
- pg_class: scanned 9 of 9 pages, containing 324 live rows and 5 dead rows; 324 rows in sample, 324 estimated total rows

Columns: Shows a list of columns for a table, including `id` and `value`.

Query Results: Displays the results of a query, including columns like `Entity`, `Cost`, `Rows`, `Time`, and `Condition`.



РОЛЬ PL/PGSQL В ЭКОСИСТЕМЕ POSTGRES

Широко распространены	Используются	Эксперименты
PL/pgSQL	plPerl	PL/R
plv8	plTCL	PL/Scheme
plPython		
plJava		

- 1** Валидируется при каждой загрузке, не компилируется в промежуточные формы
- 2** Трансформируется в AST, которое находится в памяти сессии
- 3** Выражения исполняются при помощи SQL



PL/PGSQL: СПОСОБЫ ОТЛАДКИ

RAISE (elog)

Серверные сообщения

LISTEN/NOTIFY

Асинхронный обмен сообщениями

Плагин (PLDBGAPI)

Интерактивный отладчик в командной строке

Другие способы

- Вывод в файлы (utl_files из ORFACE)
- Вывод в таблицы (автономные транзакции, dblink)
- Другие способы межпроцессорного взаимодействия



ИНСТАЛЛЯЦИЯ ОТЛАДЧИКА НА СЕРВЕРЕ

/ 1 Получить исходный текст

```
git clone git://git.postgresql.org/git/pldebugger.git
```

/ 2 Собрать как стандартное расширение

```
git clone git://git.postgresql.org/git/pldebugger.git  
cd pldebugger  
export USE_PGXS=1  
make  
make install
```

/ 3 Остановить сервер и добавить параметр

```
shared_preload_libraries = 'plugin_debugger'
```

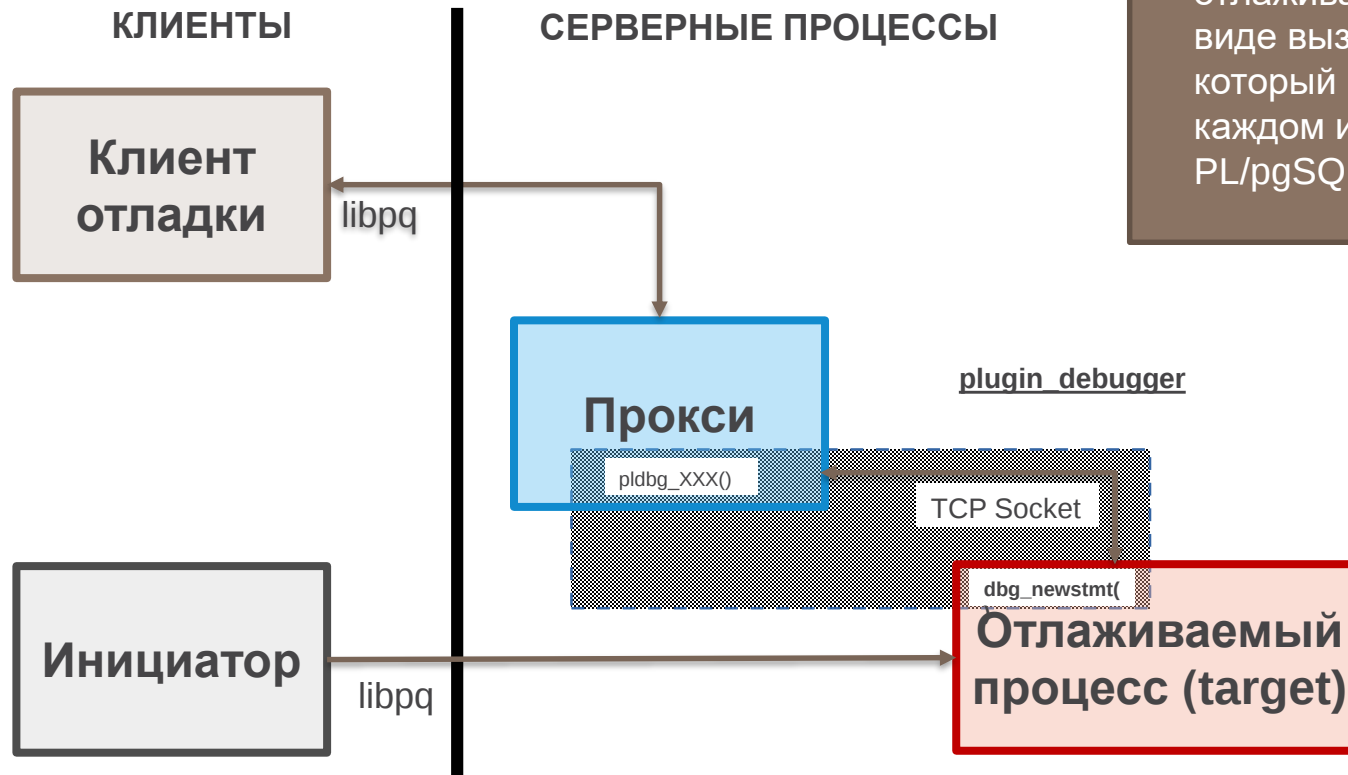
/ 4 Запустить сервер и создать расширение

```
CREATE EXTENSION pldbgapi;
```



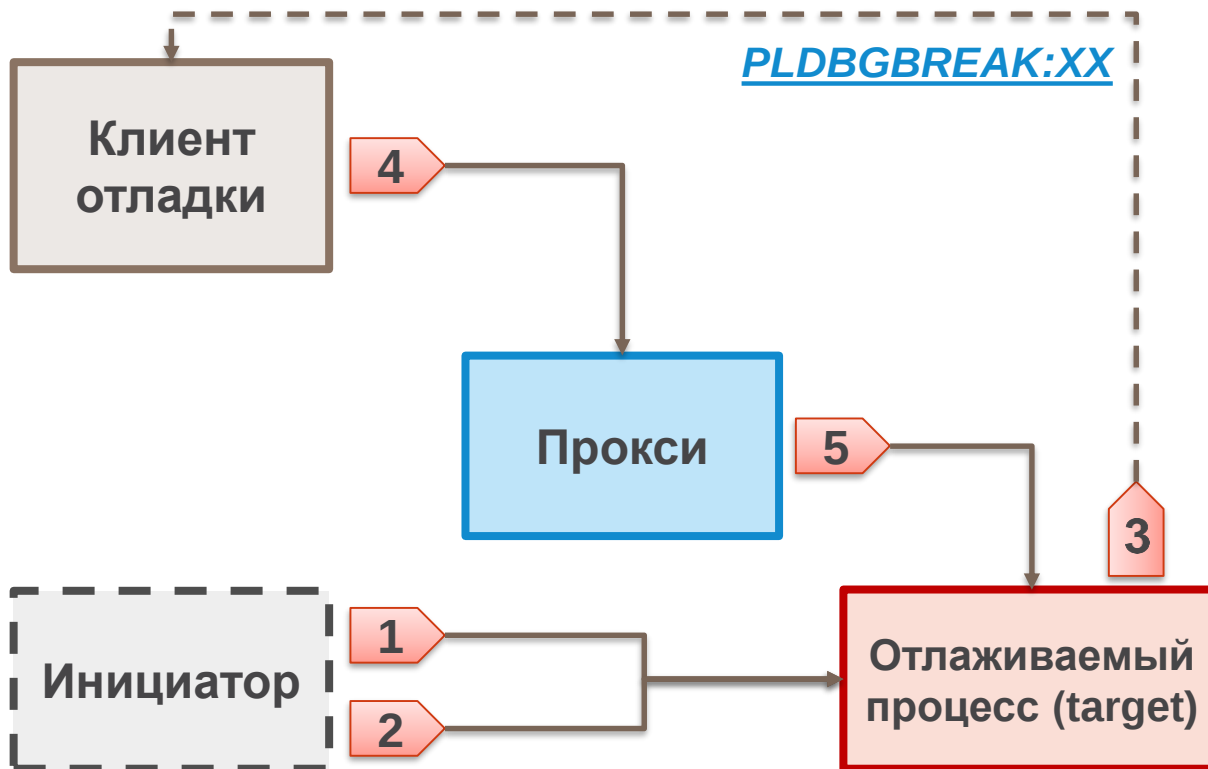
УСТРОЙСТВО ОТЛАДКИ

Отладчик встраивается в отлаживаемый процесс в виде вызова `dbg_newstmt()`, который иницируется при каждом исполнении строки PL/pgSQL





ОТЛАДКА С ЛОКАЛЬНОЙ ТОЧКОЙ ОСТАНОВА



1. Точка останова
2. Запуск процедуры
3. NOTICE + блокировка
4. Установка соединения
5. Создание сеанса отладки



ОТЛАДКА С ГЛОБАЛЬНОЙ ТОЧКОЙ ОСТАНОВА



1. Точка останова (команда для прокси), ожидание вызова (блокировка)
2. Создание точки останова
3. Инициатор запускает процесс
4. Проверка: установлена ли на строке глобальная точка останова
5. Точка нашлась - создание сеанса отладки



ОТЛАДЧИК В ECLIPSE

DBeaver 4.3.3.1 - show_histogram(_histogram_result)

Файл Редактирование Навигация Search Test Редактор SQL Run База данных Окна Справка

Commit Rollback T Auto PostgreSQL - postgres public 200

Quick Access

Debug

- PostgreSQL - postgres(postgres) - show_histogram(public) [PL/pgSQL]
 - PostgreSQL - postgres(postgres) - show_histogram(public)
 - Thread: remote procedure
 - histogram_bar(double precision,double precision,double precision) line: 5
 - show_histogram(histogram_result[]) line: 27

(x) Variables Breakpoints

Name	Value
histogram_result.total	330
histogram_result.bucket	0
histogram_result.range	[100,290)
min_count	0
max_count	330
total_count	465
bar_max_width	30
bar_tick_size	0.0909090909090909116
bar	NULL
cumsum	330
cumpct	0.709677419354838745

[100, 290)

Sessions - PostgreSQL - postgres <PostgreSQL - postgres> Script-2 show_histogram(_histogram_result) histogram_bar(float8,float8,float8) histogram_bar(float8,float8,float8)

Свойства

Имя	Значение
Общее	
Имя Процедуры	show_histogram
OID	33 940
Описание Процедуры	

Procedure Columns

Source

```
21 IF r.bucket IS NULL THEN
22     CONTINUE;
23 END IF;
24
25 cumsum := cumsum + r.count + abs(r.overflow);
26 cumpct := (cumsum::float8 / total_count);
27 bar := histogram_bar(r.count, bar_tick_size, r.overflow);
28 RETURN QUERY VALUES (
29     r.bucket,
30     r.range,
31     r.count,
32     r.overflow,
33     bar,
34     histogram_bar(cumpct, bar_max_width),
```

Source

MSK ru_RU Local attached to 5515



ПРИМЕР ДЛЯ ДЕМОНСТРАЦИИ ОТЛАДКИ

REQUEST

```
WITH class_lengths AS (  
  select reltuples len from pg_class  
)  
SELECT bucket,range,count,cumbar FROM show_histogram  
((SELECT histogram(len, 100, 2000, 10) FROM class_lengths))
```

RESULT

bucket	range	count	cumbar
0	[100,290)	31	=====
1	[290,480)	11	=====
2	[480,670)	3	=====
3	[670,860)	8	=====
4	[860,1050)	99	=====
5	[1050,1240)	0	=====
6	[1240,1430)	0	=====
7	[1430,1620)	3	=====
8	[1620,1810)	0	=====
9	[1810,2000)	0	=====



ПОСТРОЕНИЕ ГИСТОГРАММЫ

TYPE

```
histogram_result AS ( count INTEGER,  
overflow INTEGER, total INTEGER,  
bucket INTEGER, range numrange);
```

AGGREGATE

```
CREATE AGGREGATE histogram(float8, float8, float8, INTEGER) (  
SFUNC = histogram_sfunc,STYPE = histogram_result[]);
```

FUNCTION

```
FUNCTION histogram_sfunc(state histogram_result[], val float8, min float8, max float8,  
nbuckets INTEGER) RETURNS histogram_result[]
```

```
FUNCTION histogram_bar(v float8, tick_size float8, overflow float8 = 0)
```

```
FUNCTION show_histogram(h histogram_result[])
```

Весь код доступен по адресу - <https://github.com/wolever/pg-histogram>



БЛОКИРОВКИ

- 1** Предназначены для обеспечения конкурентного доступа к ресурсам
- 2** Часто являются источниками проблем и пристального внимания как разработчиков, так и DBA
- 3** Существующие решения для мониторинга не уделяют внимания количественному мониторингу блокировок
- 4** DBeaver предлагает плагин для Postgres для структурного отображения блокировок

	locktype	database	relation	page	tuple	pid	mode	granted
	relation	13951	17018			79322	RowExclusiveLock	true
	virtualxid					72923	ExclusiveLock	true
	tuple	13951	17012	0	1	79278	ExclusiveLock	true
	transactionid					79320	ExclusiveLock	true
HOLD ➔	tuple	13951	17012	0	6	79309	ExclusiveLock	true
	transactionid					79283	ExclusiveLock	true
	transactionid					79315	ExclusiveLock	true
	tuple	13951	17012	8	3	79313	ExclusiveLock	false
	tuple	13951	17012	0	7	79314	ExclusiveLock	false
WAIT ➔	tuple	13951	17012	0	6	79326	ExclusiveLock	false
	transactionid					79310	ExclusiveLock	true
	transactionid					79284	ExclusiveLock	true
	tuple	13951	17012	9	6	79302	ExclusiveLock	false

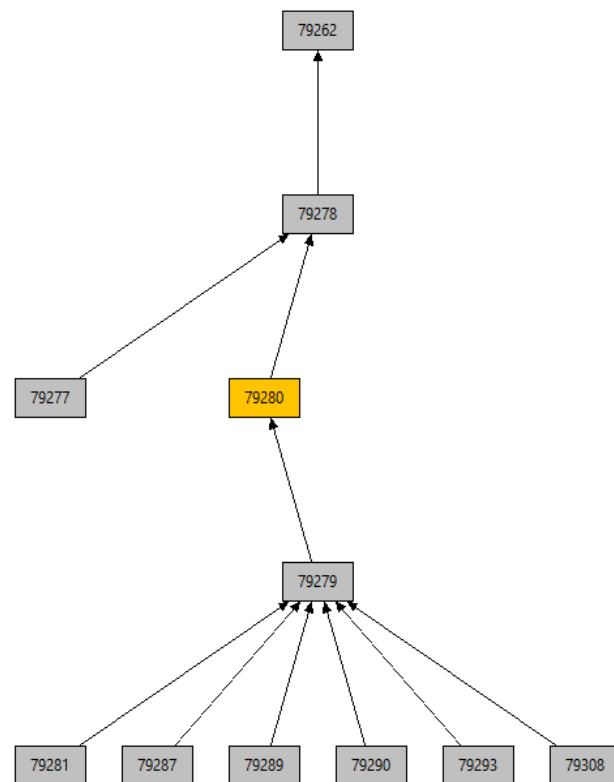


ГРАФ БЛОКИРОВОК

Wait PID	Wait User	Hold PID	Hold User	Wait Statement	Hold Statement
79 262	postgres	0		/*ROOT 1 */ ...	
79 265	postgres	0		/*ROOT 4 */ ...	
79 264	postgres	0		/*ROOT 3 */ ...	
79 267	postgres	0		/*ROOT 6 */ ...	
79 266	postgres	0		/*ROOT 5 */ ...	
79 268	postgres	0		/*ROOT 7 */ ...	
79 277	postgres	79 278	postgres	/*[79277] 1->...	/*[79278] 1->1 (0) */ update usr set v = 100500 where field = \$1
79 279	postgres	79 280	postgres	/*[79279] Sub...	/*[79280] 1->1 (3) */ update usr set v = 100500 where field = \$1
79 278	postgres	79 262	postgres	/*[79278] 1->...	/*ROOT 1 */ update usr set v = 100500 where field = \$1
79 281	postgres	79 279	postgres	/*[79281] Sub...	/*[79279] Sublock 64 for 1 -> 1 (1) */ update usr set v = 100500 where field = \$1
79 280	postgres	79 278	postgres	/*[79280] 1->...	/*[79278] 1->1 (0) */ update usr set v = 100500 where field = \$1
79 283	postgres	79 282	postgres	/*[79283] 2->...	/*[79282] 2->0 (1) */ update usr set v = 100500 where field = \$1
79 282	postgres	79 263	postgres	/*[79282] 2->...	/*ROOT 2 */ update usr set v = 100500 where field = \$1
79 285	postgres	79 282	postgres	/*[79285] Sub...	/*[79282] 2->0 (1) */ update usr set v = 100500 where field = \$1
79 284	postgres	79 282	postgres	/*[79284] 2->...	/*[79282] 2->0 (1) */ update usr set v = 100500 where field = \$1
79 287	postgres	79 279	postgres	/*[79287] Sub...	/*[79279] Sublock 64 for 1 -> 1 (1) */ update usr set v = 100500 where field = \$1
79 286	postgres	79 285	postgres	/*[79286] Sub...	/*[79285] Sublock 59 for 2 -> 0 (2) */ update usr set v = 100500 where field = \$1
79 289	postgres	79 279	postgres	/*[79289] Sub...	/*[79279] Sublock 64 for 1 -> 1 (1) */ update usr set v = 100500 where field = \$1
79 288	postgres	79 282	postgres	/*[79288] 2->...	/*[79282] 2->0 (1) */ update usr set v = 100500 where field = \$1
79 291	postgres	79 264	postgres	/*[79291] 3->...	/*ROOT 3 */ update usr set v = 100500 where field = \$1

Wait - 79280							Hold - 79278						
Schema	Type	Rel.	Mode	TID	Page/Tpl	+	Schema	Type	Rel.	Mode	TID	Page/Tpl	+
postgres	relation	usr_pkey	RowExclusiveLock	-/-	-/-	☒	postgres	relation	usr_pkey	RowExclusiveLock	-/-	-/-	☒
postgres	relation	usr	RowExclusiveLock	-/-	-/-	☒	postgres	relation	usr	RowExclusiveLock	-/-	-/-	☒
	virtualxid		ExclusiveLock	-/-	-/-	☒		virtualxid		ExclusiveLock	-/-	-/-	☒
	transactionid		ExclusiveLock	892	-/-	☒		tuple	usr	ExclusiveLock	0/1	-/-	☒
postgres	tuple	usr	ExclusiveLock	0/1	0/1	☐		transactionid		ExclusiveLock	891	-/-	☒
								transactionid		ShareLock	881	-/-	☐

Wait - 79280 (postgres) Hold - 79278 (postgres)





ТВОРЧЕСКИЕ ПЛАНЫ, КОНТАКТЫ, ВОПРОСЫ

НАШИ ПЛАНЫ

- Визуализация (и не только) плана исполнения
- Сравнение и обновление структуры баз данных
- Управление Postgres Row Level Security
- Мониторинг и отчеты производительности

ПРИСОЕДИНЯЙТЕСЬ

- GitHub: <https://github.com/dbeaver/dbeaver>
- DBeaver CE: <https://dbeaver.jkiss.org/>
- DBeaver EE: <https://dbeaver.com/>

ВАШИ ВОПРОСЫ



Если мы не успели
ответить на ваш вопрос,
напишите нам:

alexander.fedorov@dbeaver.com

andrew@jkiss.org

serge@dbeaver.com

tati@dbeaver.com