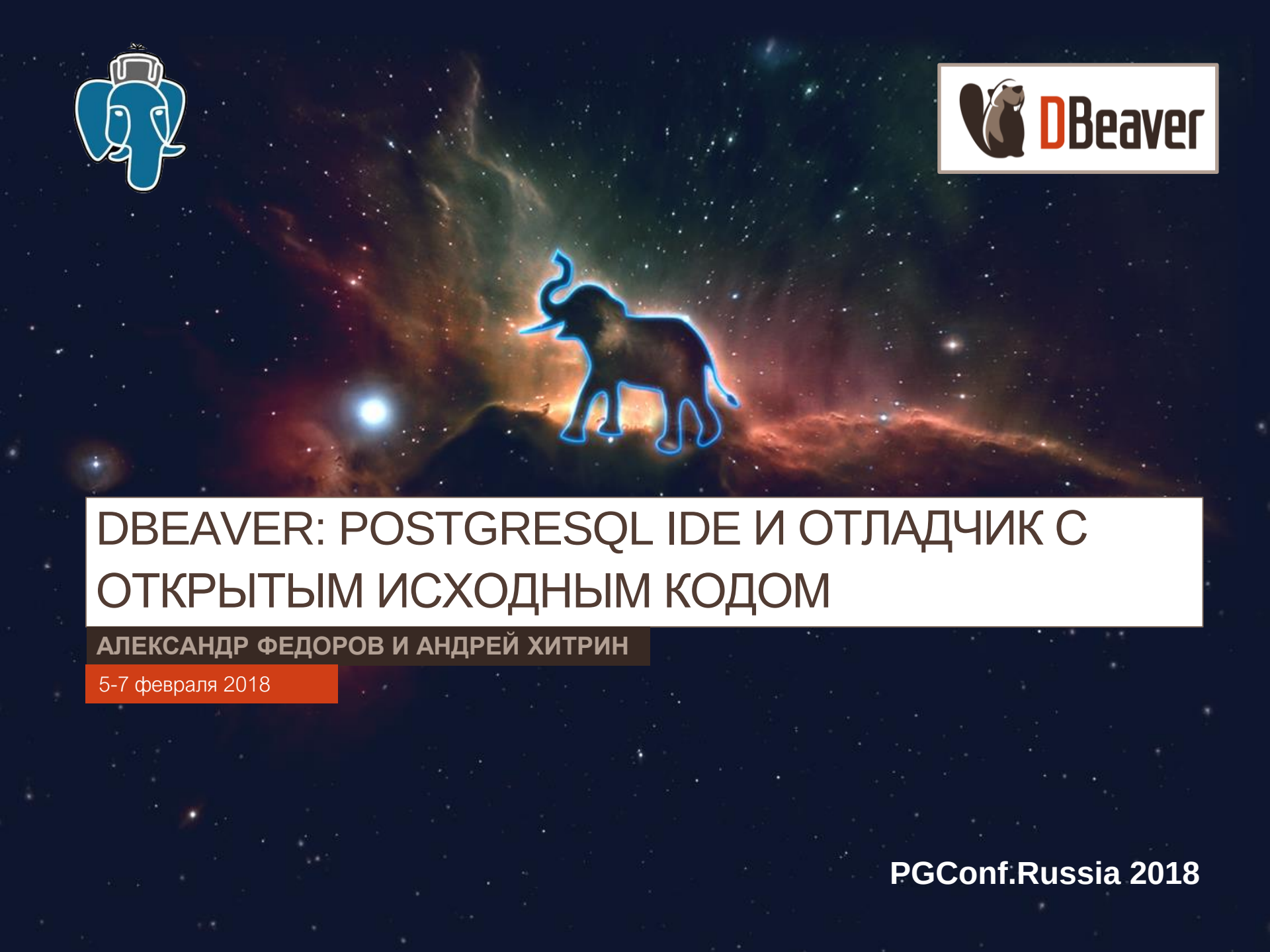


DBEAVER: POSTGRESQL IDE И ОТЛАДЧИК С ОТКРЫТЫМ ИСХОДНЫМ КОДОМ

АЛЕКСАНДР ФЕДОРОВ И АНДРЕЙ ХИТРИН

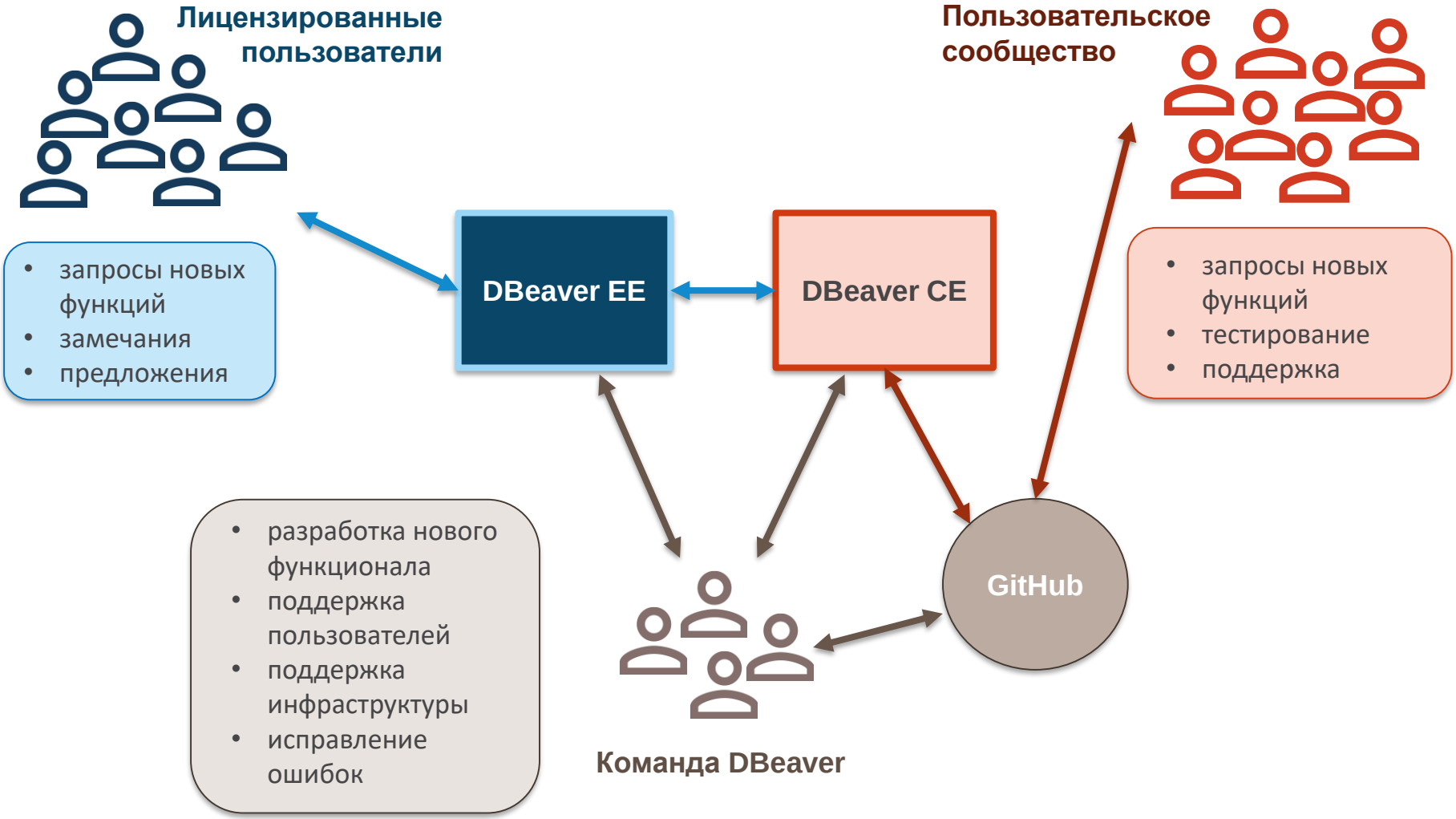
5-7 февраля 2018

PGConf.Russia 2018

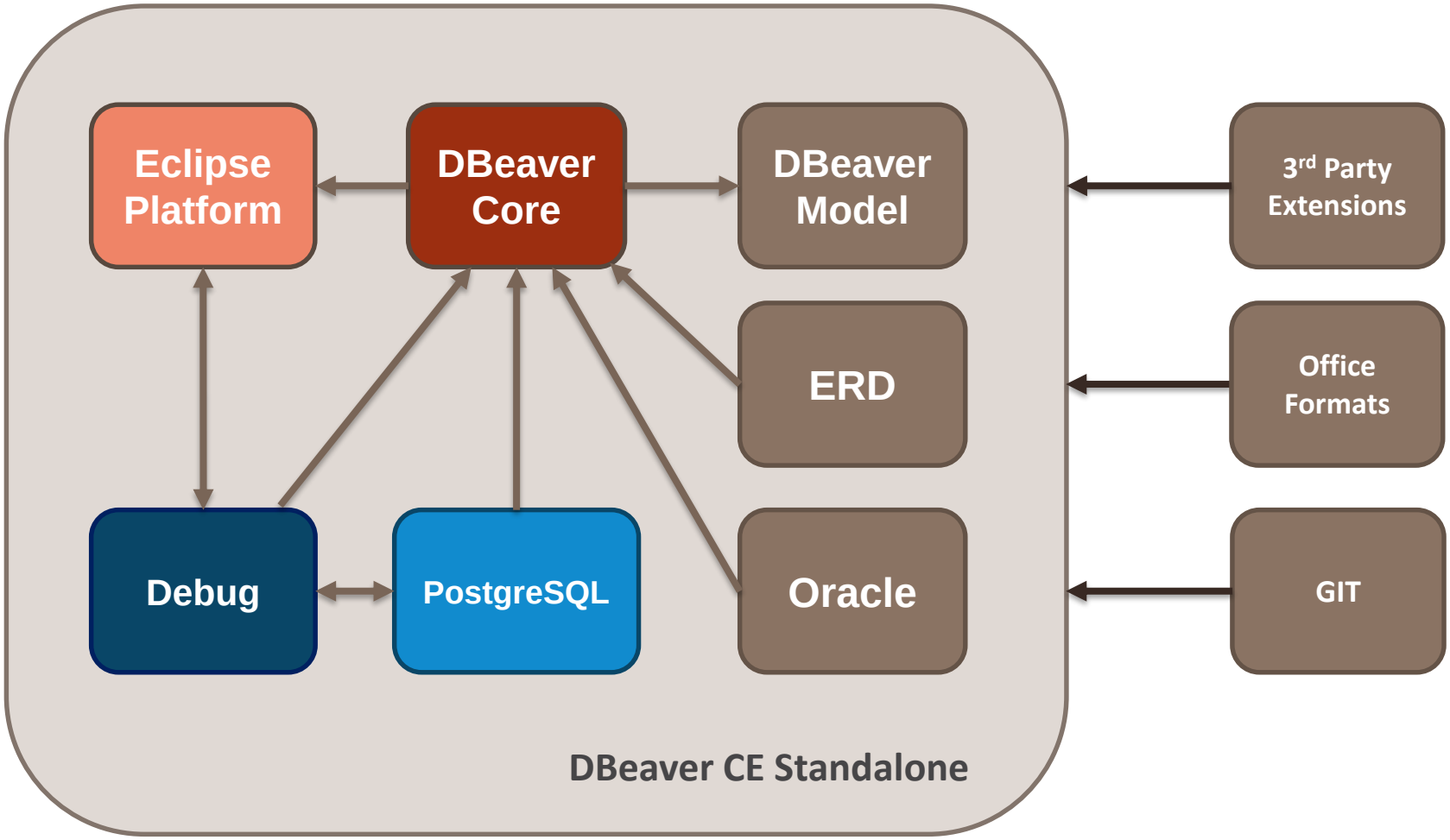
КОРОТКО О DBEAVER

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

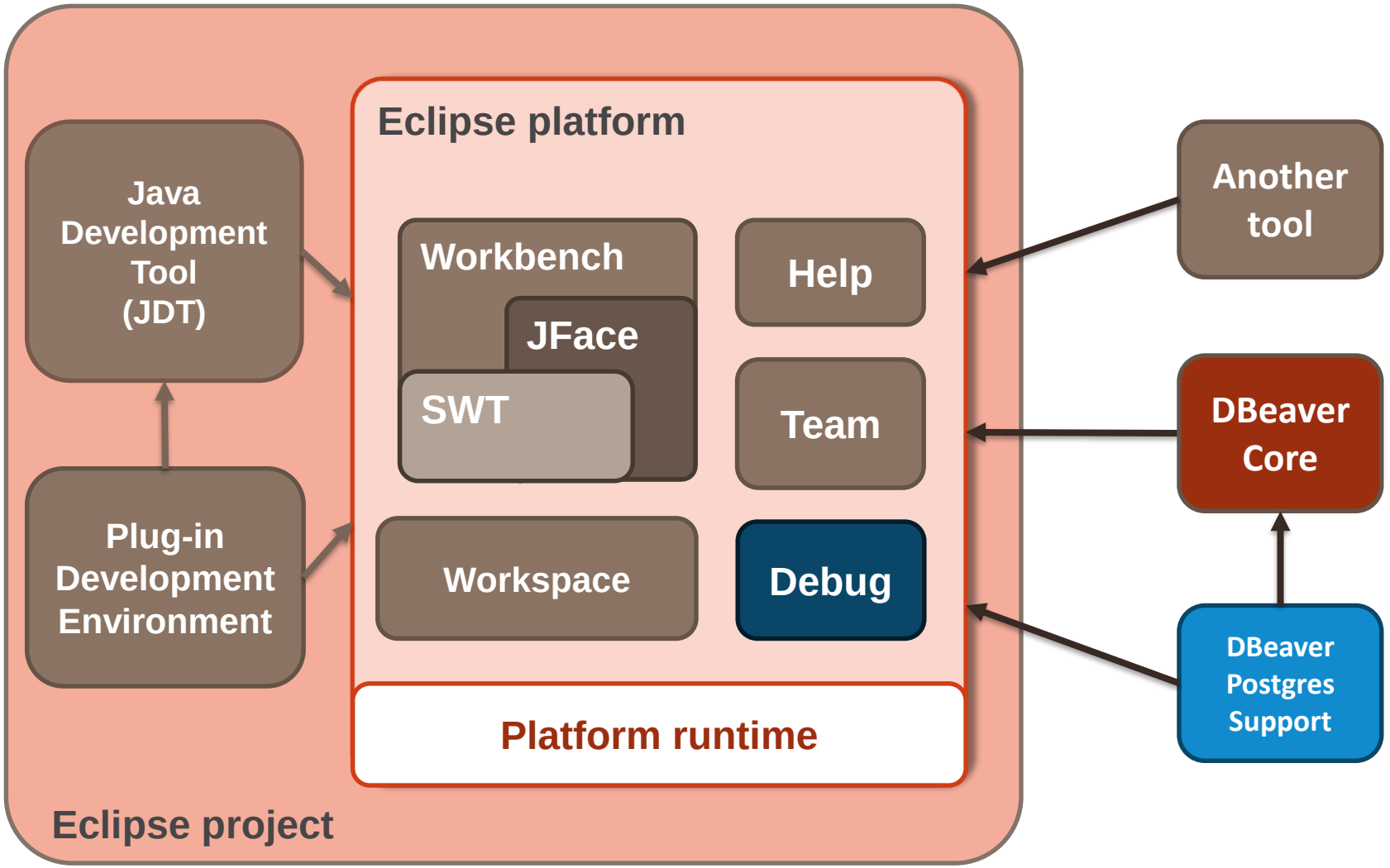
ЭКОСИСТЕМА DBEAVER



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

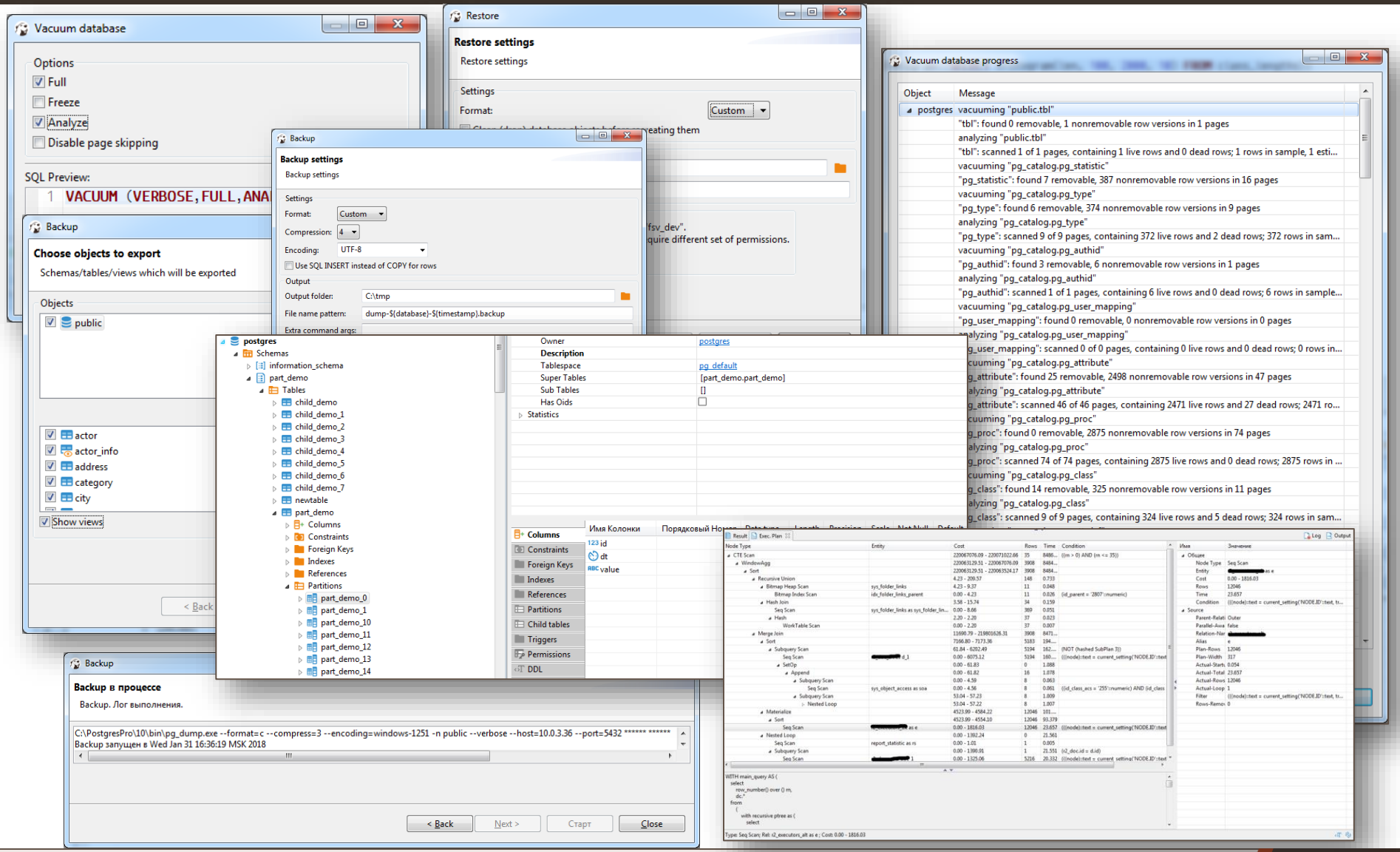
Полнофункциональный и быстрый SQL-редактор для написания, применения, сохранения, экспорта и переиспользования скриптов с профилировщиком данных и функцией форматирования

СПЕЦИАЛЬНЫЕ ФУНКЦИИ DBEAVER ДЛЯ POSTGRES

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

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

СПЕЦИАЛЬНЫЕ ФУНКЦИИ POSTGRES В ИНТЕРФЕЙСЕ DBEAVER



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
```

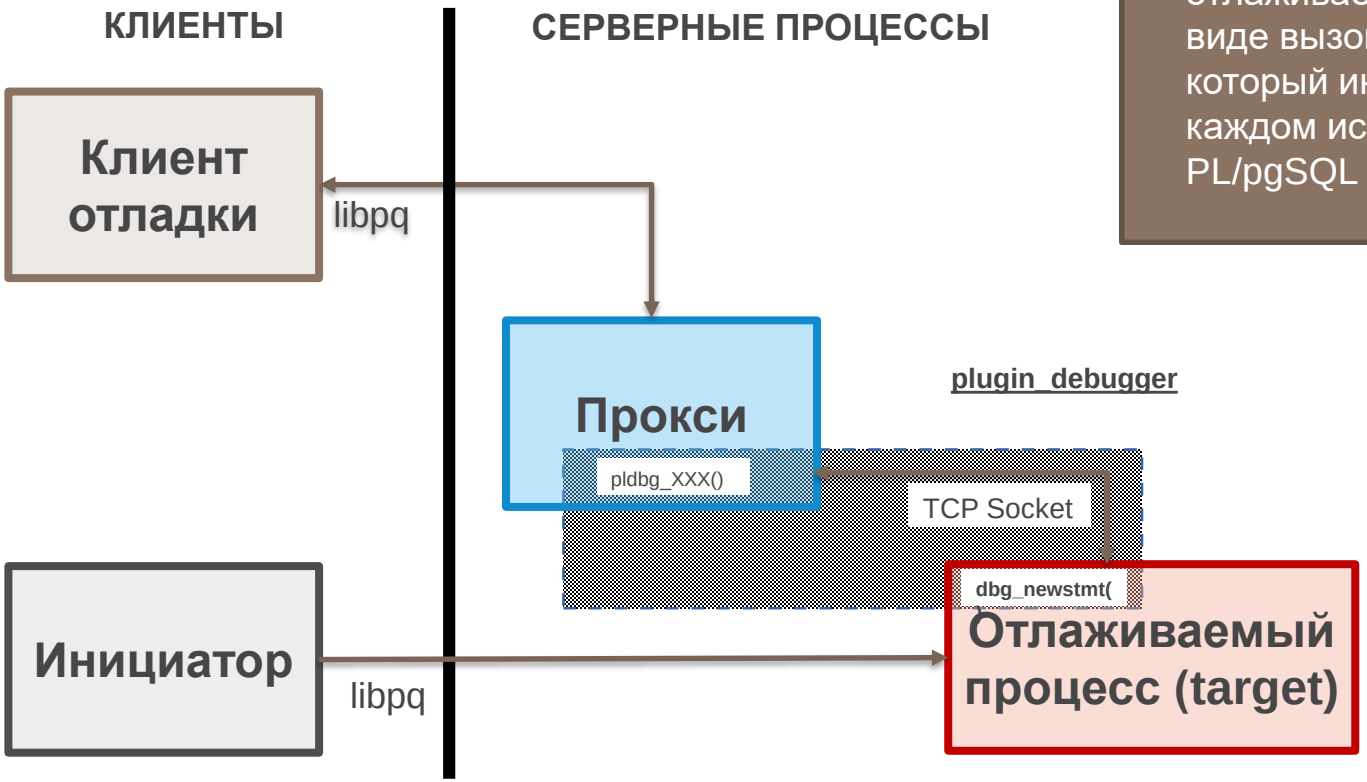
3 **Остановить сервер и добавить параметр**

```
shared_preload_libraries = 'plugin_debugger'
```

4 **Запустить сервер и создать расширение**

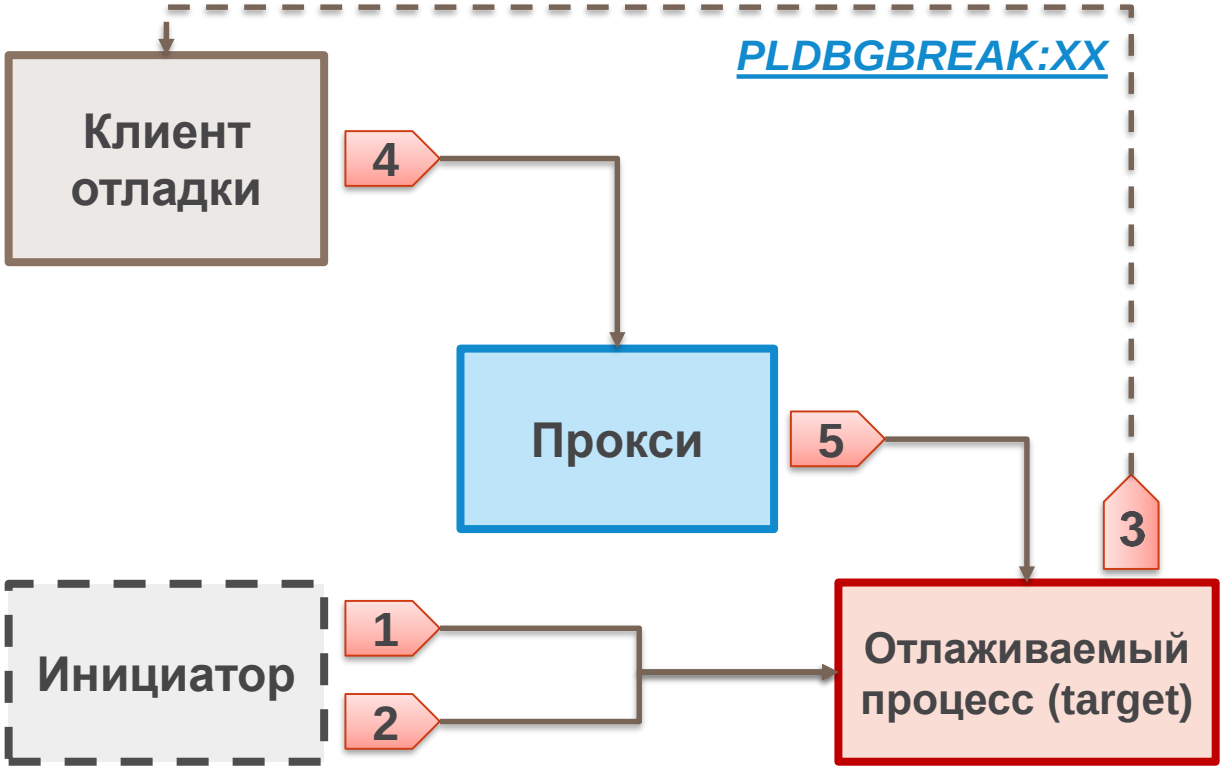
```
CREATE EXTENSION pldbgapi;
```

УСТРОЙСТВО ОТЛАДКИ



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

ОТЛАДКА С ЛОКАЛЬНОЙ ТОЧКОЙ ОСТАНОВА

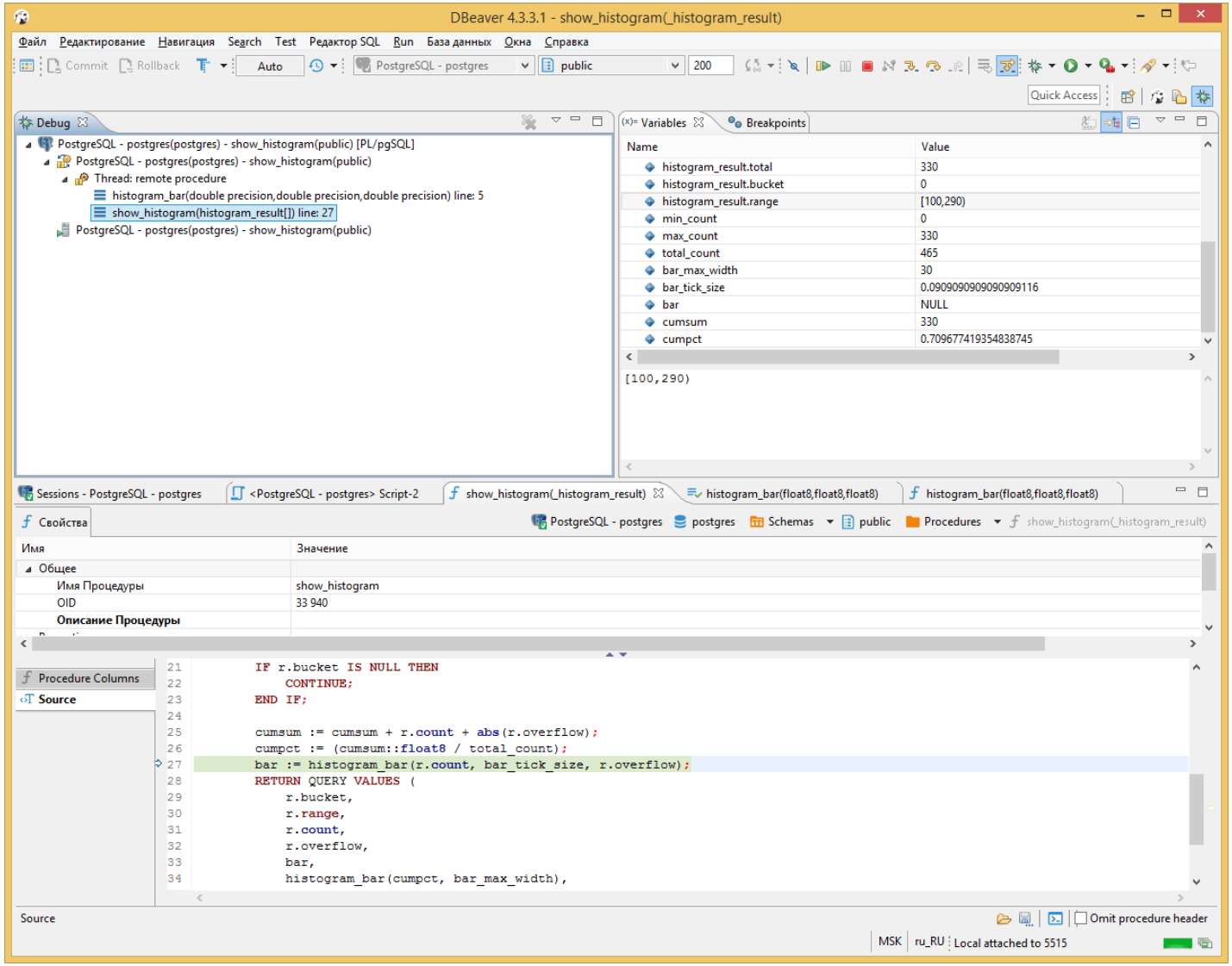


1. Точка останова
2. Запуск процедуры
3. NOTICE + блокировка
4. Установка соединения
5. Создание сеанса отладки

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



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

HOLD

WAIT

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
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
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

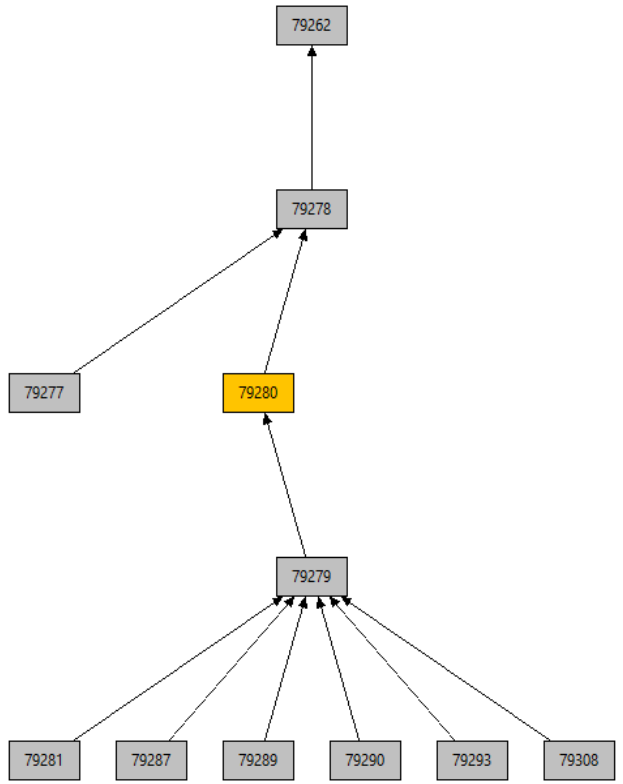
Wait PID	Wait User	Hold PID	Hold User	Wait Statement	Hold Statement
79 262	postgres	0		/*ROOT 1 */ ...	
79 265	postgres	0		/*ROOT 4 */ ...	
79 264	postgres	0		/*ROOT 3 */ ...	
79 267	postgres	0		/*ROOT 6 */ ...	
79 266	postgres	0		/*ROOT 5 */ ...	
79 268	postgres	0		/*ROOT 7 */ ...	
79 277	postgres	79 278	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

Schema	Type	Rel.	Mode	TID	Page/Tpl	+
postgres	relation	usr_pkey	RowExclusiveLock		-/-	☑
postgres	relation	usr	RowExclusiveLock		-/-	☑
	virtualxid		ExclusiveLock		-/-	☑
	transactionid		ExclusiveLock	892	-/-	☑
postgres	tuple	usr	ExclusiveLock		0/1	☐

Hold - 79278

Schema	Type	Rel.	Mode	TID	Page/Tpl	+
postgres	relation	usr_pkey	RowExclusiveLock		-/-	☑
postgres	relation	usr	RowExclusiveLock		-/-	☑
	virtualxid		ExclusiveLock		-/-	☑
postgres	tuple	usr	ExclusiveLock		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