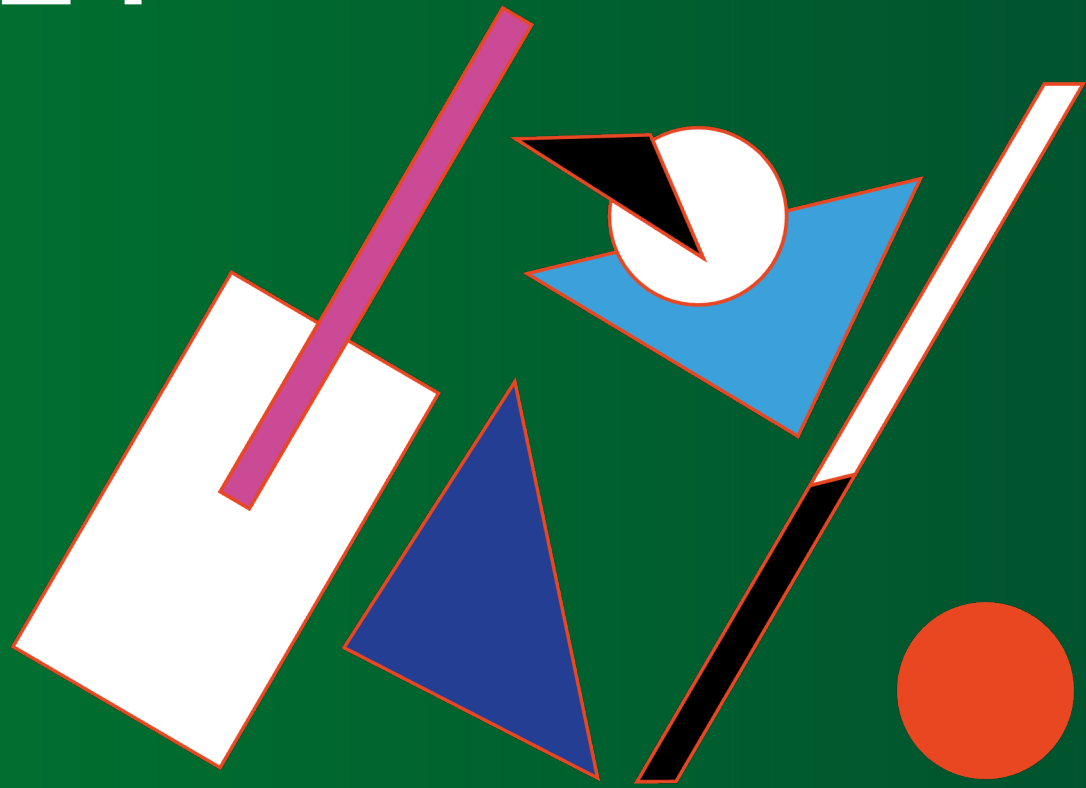
PostgresPro

PGCONF.RUSSIA 2024

Управление планами запросов: новые возможности



Александр Котин

Старший технический менеджер продукта

a.kotin@postgrespro.ru

PostgresPro

Что такое «план запроса»?

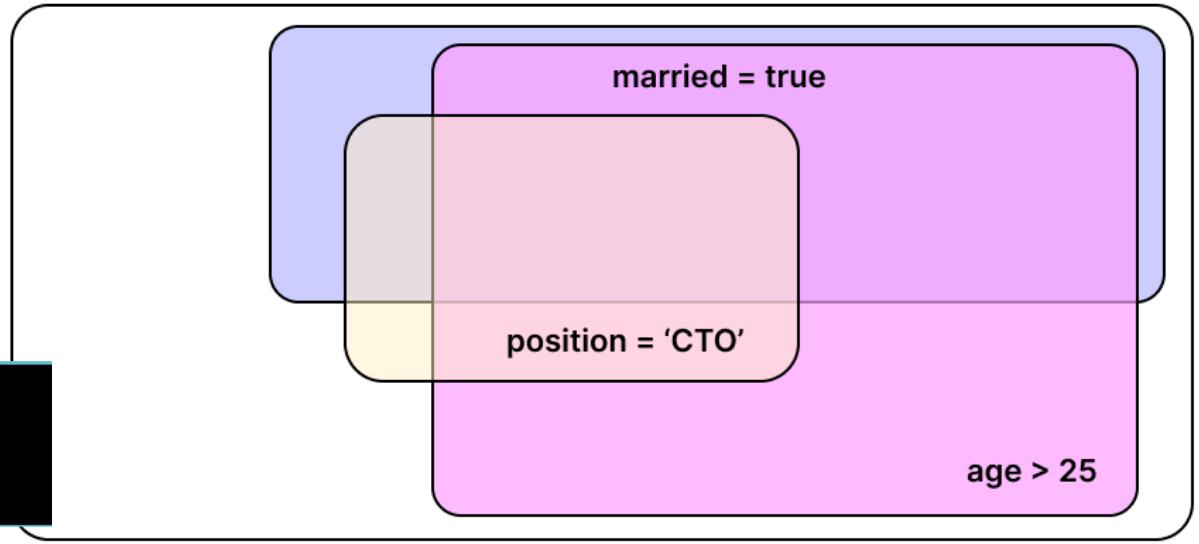
```
EXPLAIN SELECT *
FROM users AS u1, messages AS m, users AS u2
WHERE u1.id = m.sender_id AND m.receiver_id = u2.id;
                                    QUERY PLAN
-------------------------------------------------------------------------------
 Hash Join  (cost=540.00..439429.44 rows=10003825 width=27)
   Hash Cond: (m.receiver_id = u2.id)
   ->  Hash Join  (cost=270.00..301606.84 rows=10003825 width=23)
         Hash Cond: (m.sender_id = u1.id)
         ->  Seq Scan on messages m  (cost=0.00..163784.25 rows=10003825 width=19)
         ->  Hash  (cost=145.00..145.00 rows=10000 width=4)
               ->  Seq Scan on users u1  (cost=0.00..145.00 rows=10000 width=4)
   ->  Hash  (cost=145.00..145.00 rows=10000 width=4)
         ->  Seq Scan on users u2  (cost=0.00..145.00 rows=10000 width=4)
(9 rows)
```

PostgresPro

Проблематика: зачем нужно управлять планами запросов?

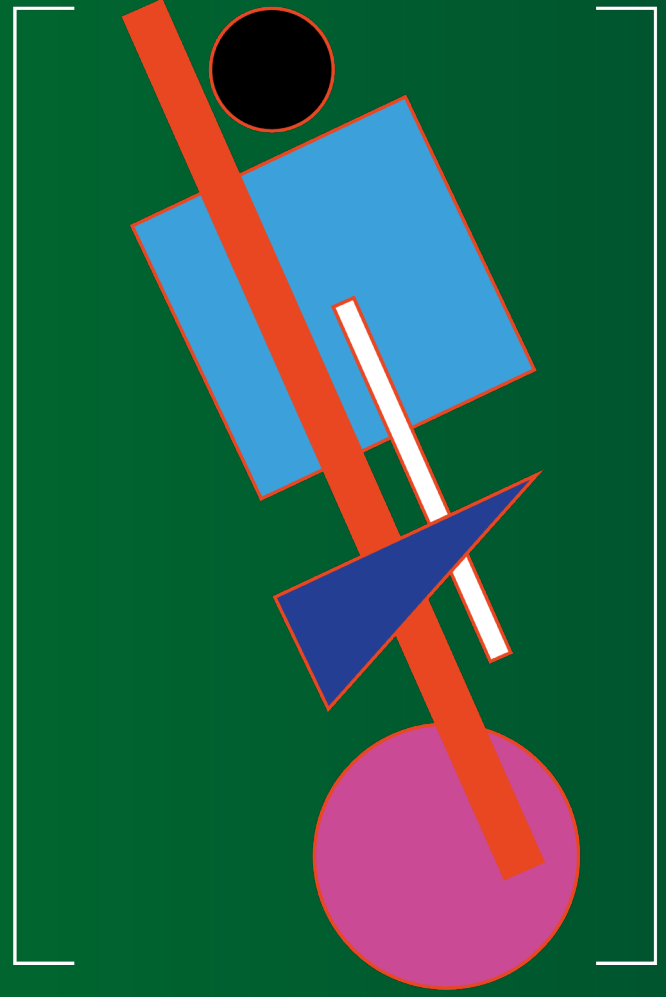
- Потери на планировании
- Контроль за ключевыми запросами
- Стабильность производительности
- Предсказуемость результата
- Статистика отстает от данных
- Оптимизатор ошибается

```
SELECT * FROM users
WHERE age > 25 AND married = true
AND position = 'CTO';
```



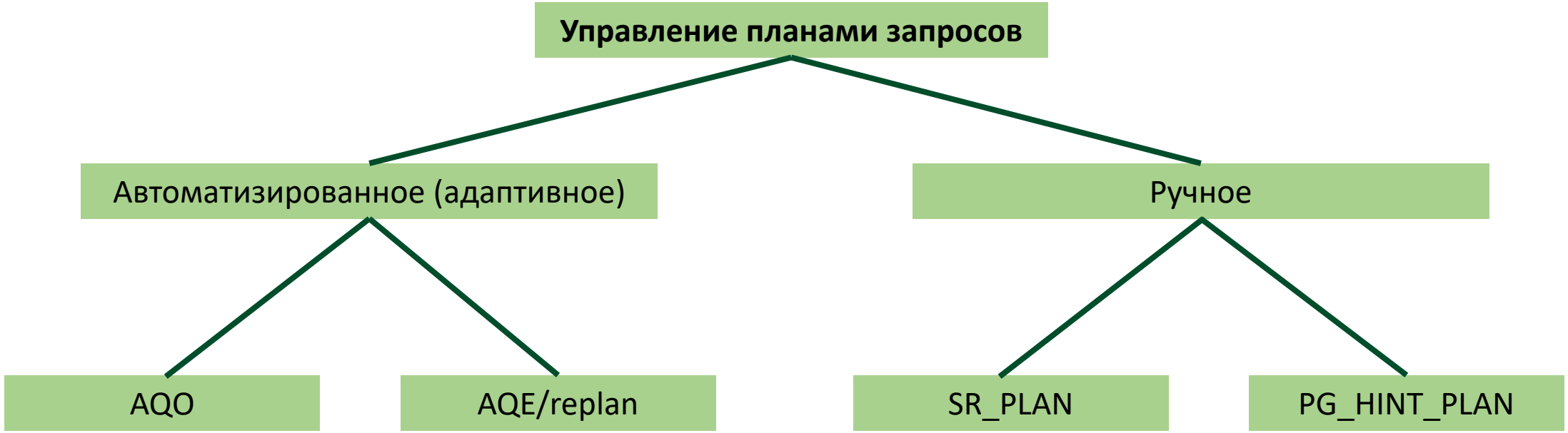
PostgresPro

Инструменты управления планами запросов



PostgresPro

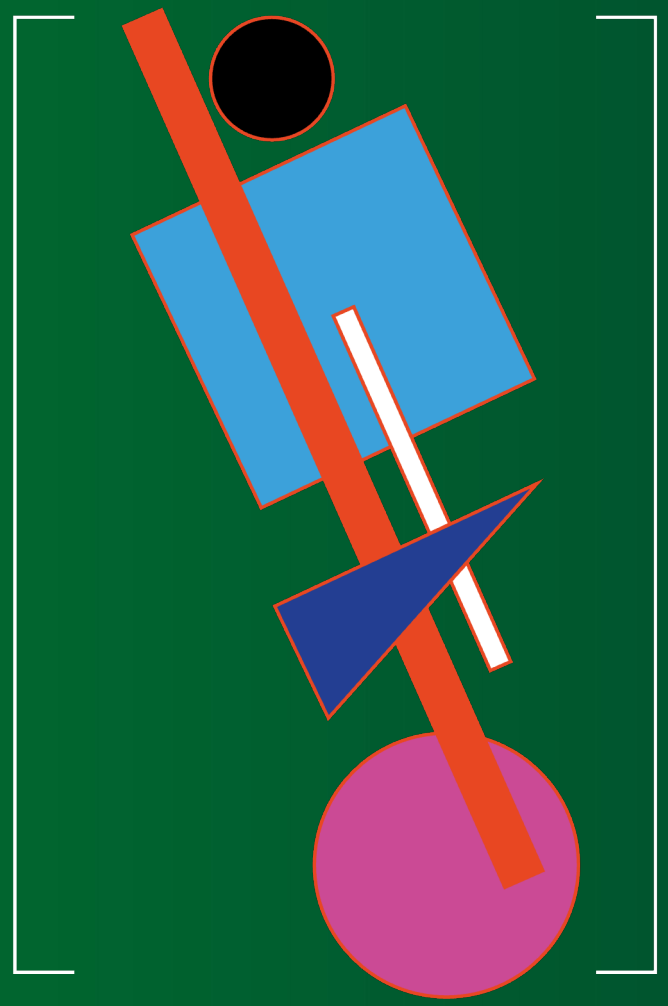
Инструменты Postgres Pro для управления планами запросов



PostgresPro

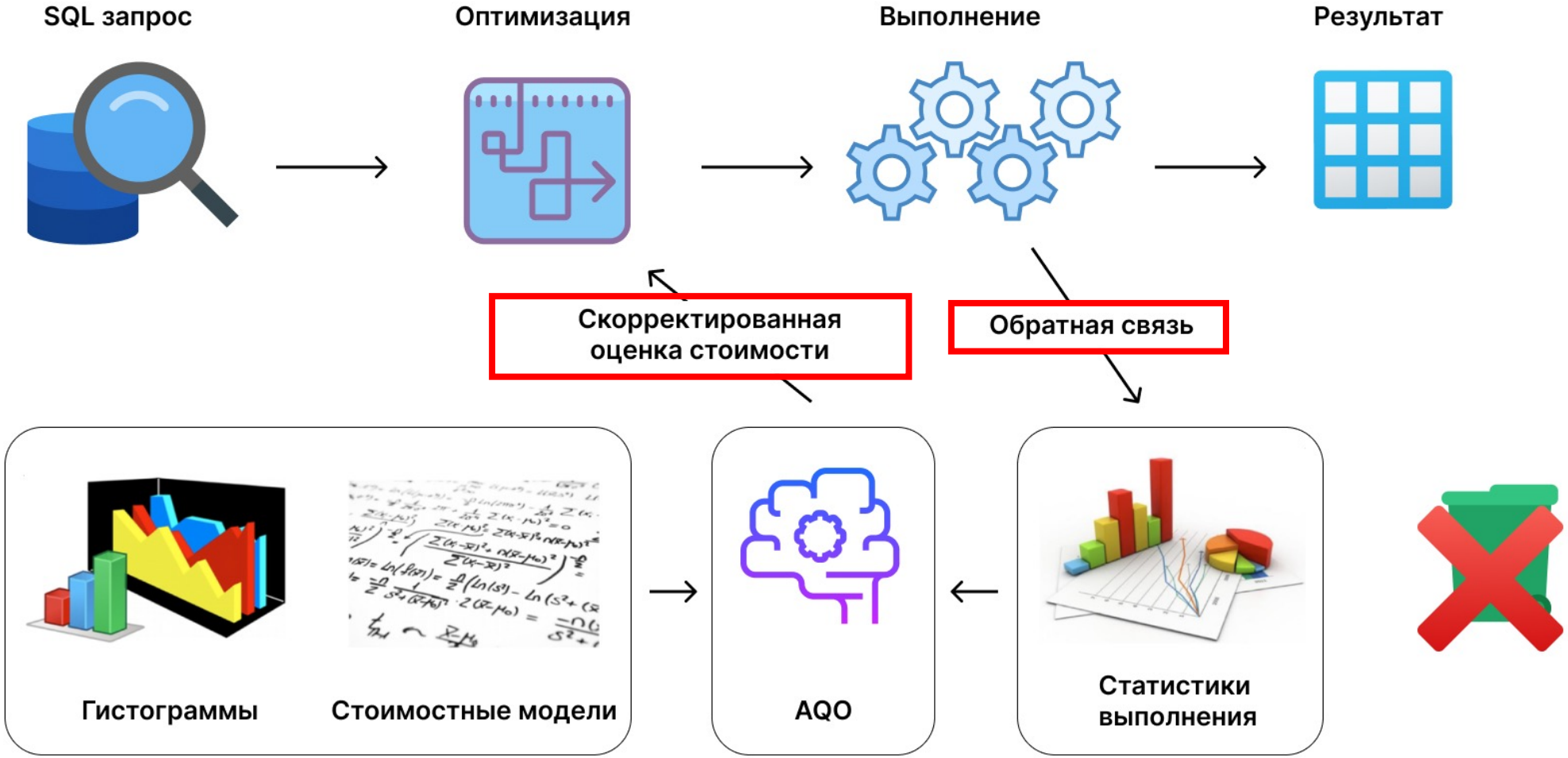
Adaptive Query Optimizer

Машинное обучение на предыдущих запросах



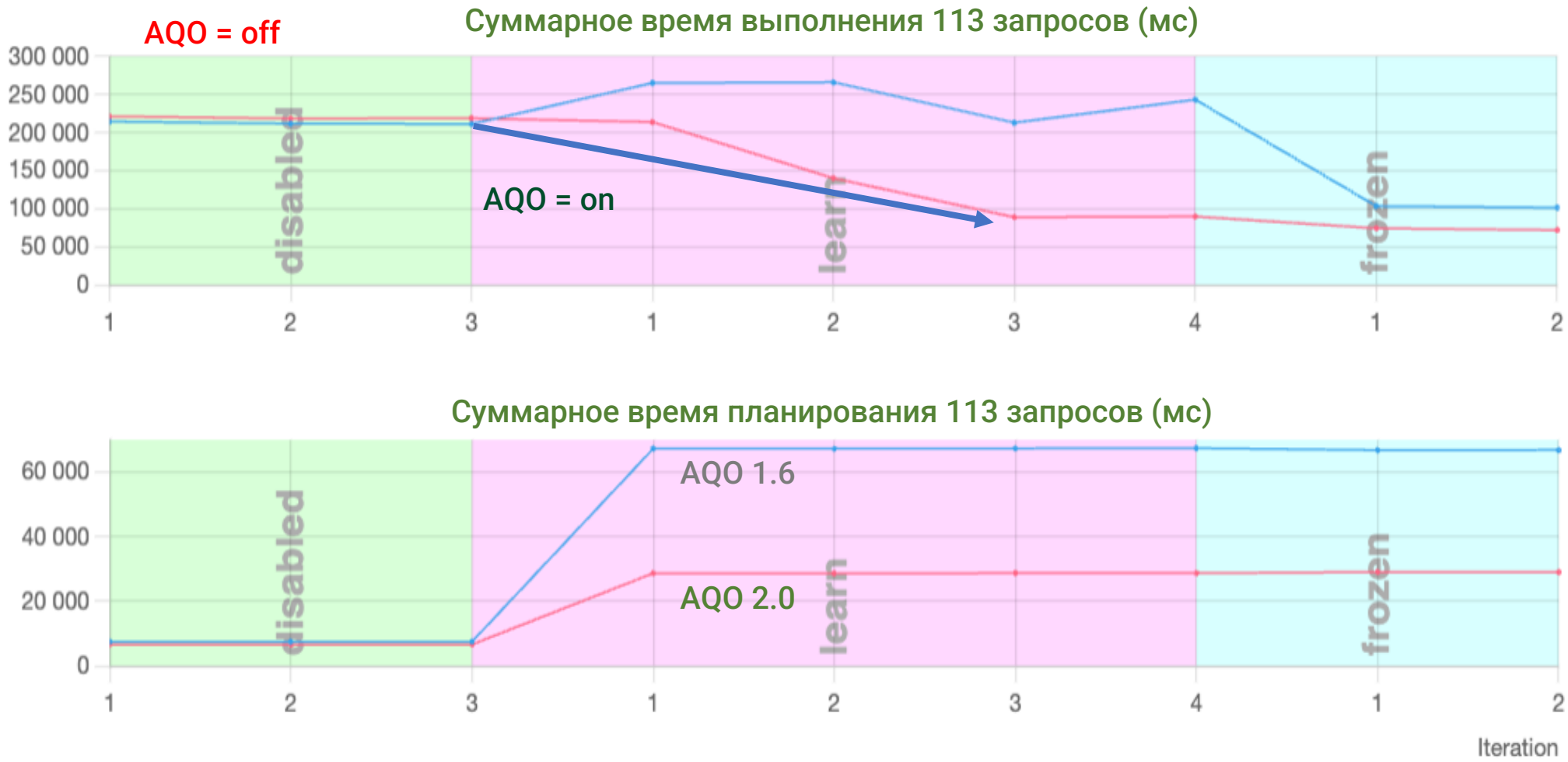
PostgresPro

AQO - Adaptive Query Optimizer



PostgresPro

AQO 2.0 - польза



PostgresPro

AQO 2.0 – что изменилось?

- Чтобы начать использование – достаточно выставить параметр aqo.enable = on
- Знания по умолчанию общие. Их можно изолировать по запросам, выставив advanced = on
- Всего 3 режима – Learn, Controlled, Frozen
- Drop Extension = мгновенно выключает AQO на текущей базе

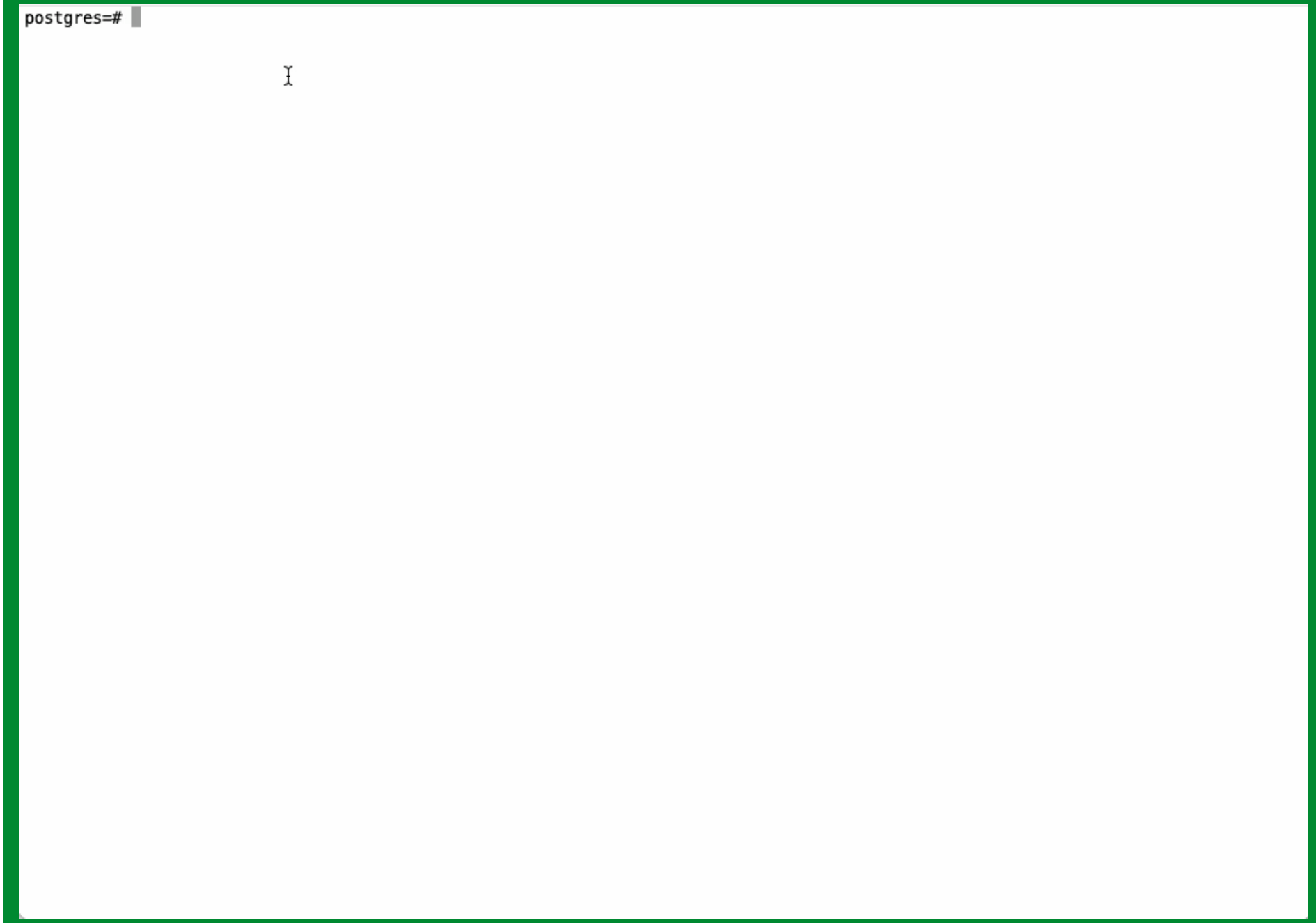
PostgresPro

AQO – плюсы и минусы

- Плюсы
  - Есть во всех редакциях Postgres Pro
  - «Искусственный интеллект»
  - Автоматическое исправление проблем производительности
  - Не используются таблицы -> дружит с 1С
  - Размещается в shmem – максимально на чтение
- Минусы
  - Оверхед при планировании на всех запросах
  - Нужно дообучаться при изменении данных
  - Не используются таблицы –> нет репликации через WAL (но...)

PostgresPro

AQO – демо



PostgresPro

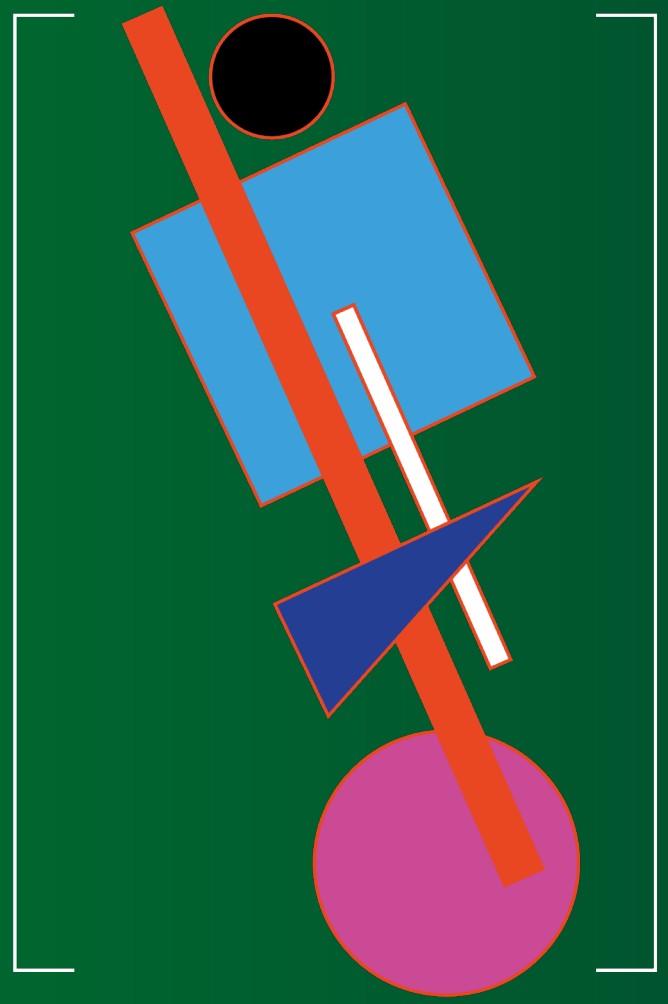
AQO – как использовать?

- Добавляем в shared preload libraries
- Create extension aqo в базе
- В своей или в сессии от приложения выставляется aqo.mode = learn; (уже стоит по дефолту)
- aqo.enable = on;
- Мониторим планы запросов и после достижения результата применяем для всех
- Alter system set aqo.enable = on;
- Alter system set aqo.mode = frozen;
- select pg_reload_conf();

PostgresPro

Adaptive Query Executor /replan

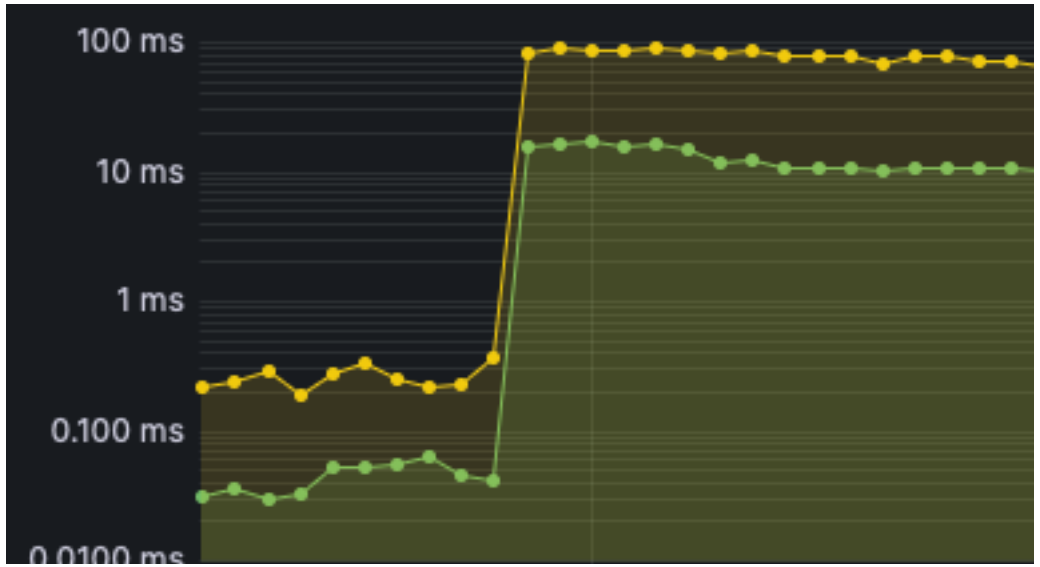
Перепланирование запросов «на лету»



PostgresPro

Adaptive Query Executor (AQE/replan) - польза

replan = on



replan = off

PostgresPro

AQE/replan – плюсы и минусы

- Плюсы
  - Встроено в Postgres Pro 16 Enterprise
  - Работает на всем потоке запросов
  - Уникальная технология в мире Postgres
  - Может использоваться как защита последней надежды от безнадежных запросов
  - Или для защиты критических запросов

- Минусы
  - Сложно подобрать единое время перепланирования
  - Бесполезен для 1С - нет поддержки extended protocol и prepared statements
  - Нельзя выставить отсечку через хинт pg_hint_plan

PostgresPro

AQE/replan

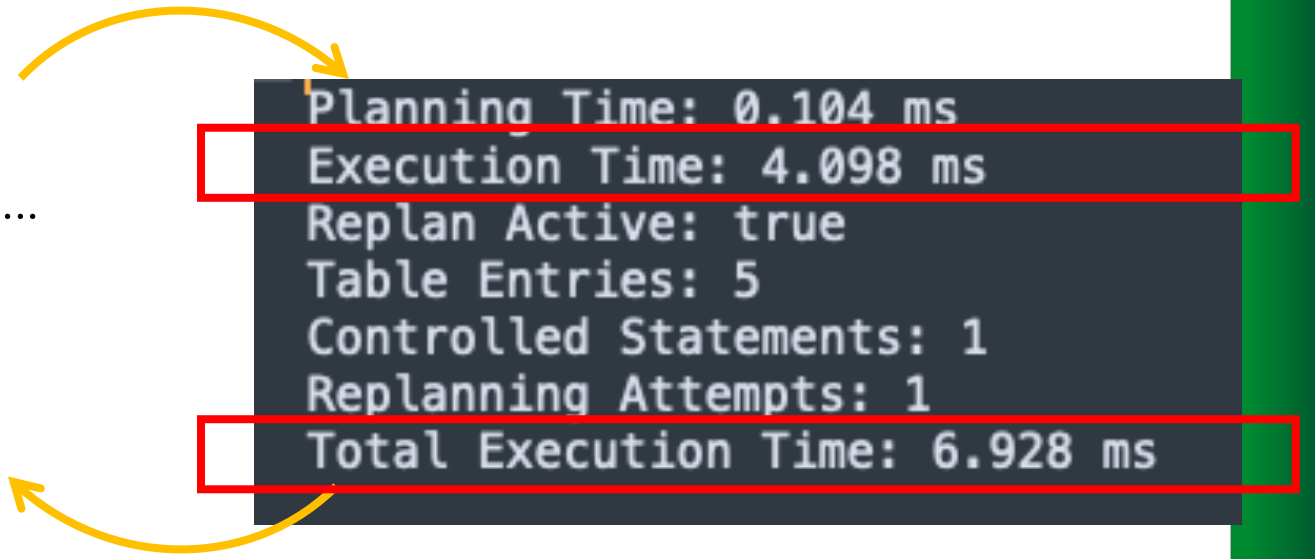
демо

```
postgres=#
```

PostgresPro

AQE/replan – как использовать?

- Set replan_enable = on
  - replan_query_execution_time 1, 2…
  - replan_max_attempts 100..10…



- pg_backend_set_config
- replan_show_signature on
- replan_regression_mode off
- replan_overrun_limit -1

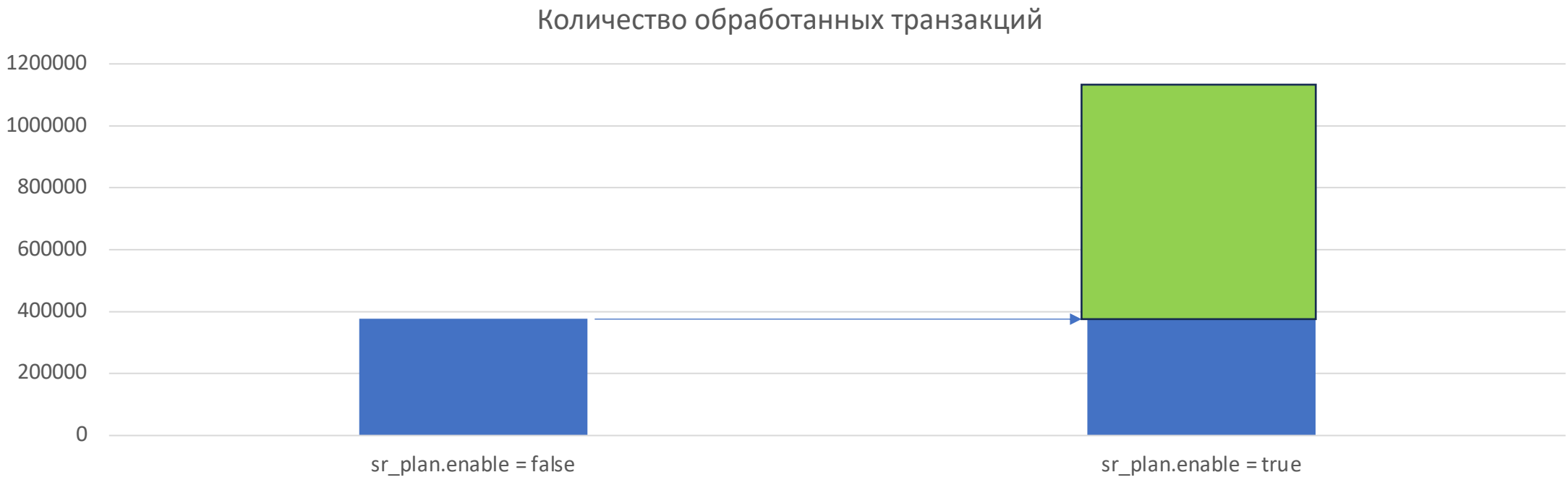
PostgresPro

SR_PLAN

Save & repeat plan

PostgresPro

sr_plan - польза



./pgbench -n -r -c 1 -T 60 -f test_complx.sql

```
select count(*) from pgbench_tellers pt, pgbench_branches pb, pgbench_accounts pa
WHERE pt.tid = pb.bid AND pb.bid = pa.aid;
```

PostgresPro

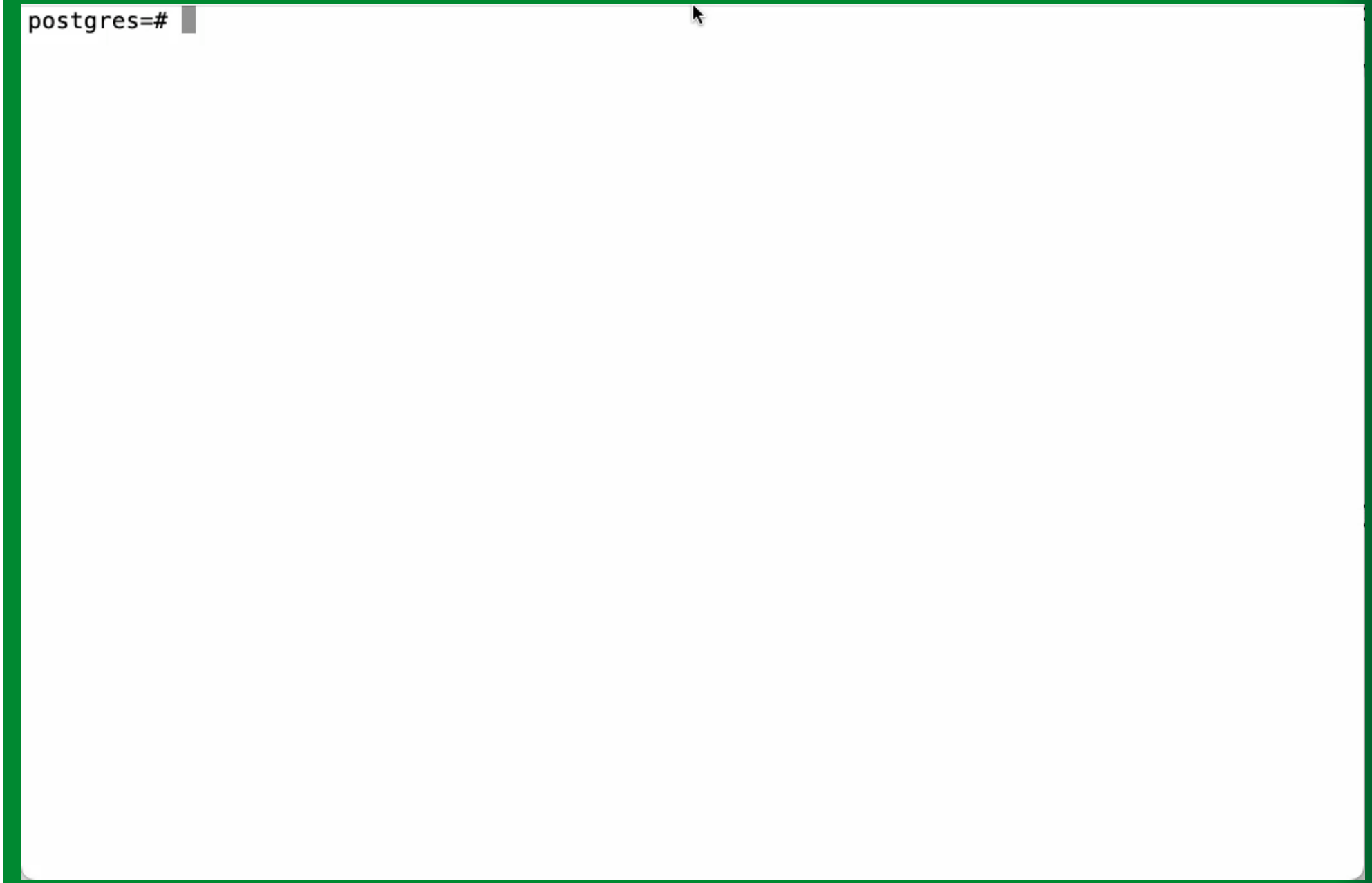
sr_plan - плюсы и минусы

- Плюсы
  - Экономия на планировании
  - Полный контроль за планом
  - Работает с prepared statements
  - Дружит с 1С - размещается в shmem и кешируется на диск
- Минусы
  - План может не влезать в кеш бекенда
  - Прожорлив к памяти
  - Нет переносимости планов (привязаны к OIDs), но...
  - Нет поддержки репликации, но...

PostgresPro

Sr_plan

демо



PostgresPro

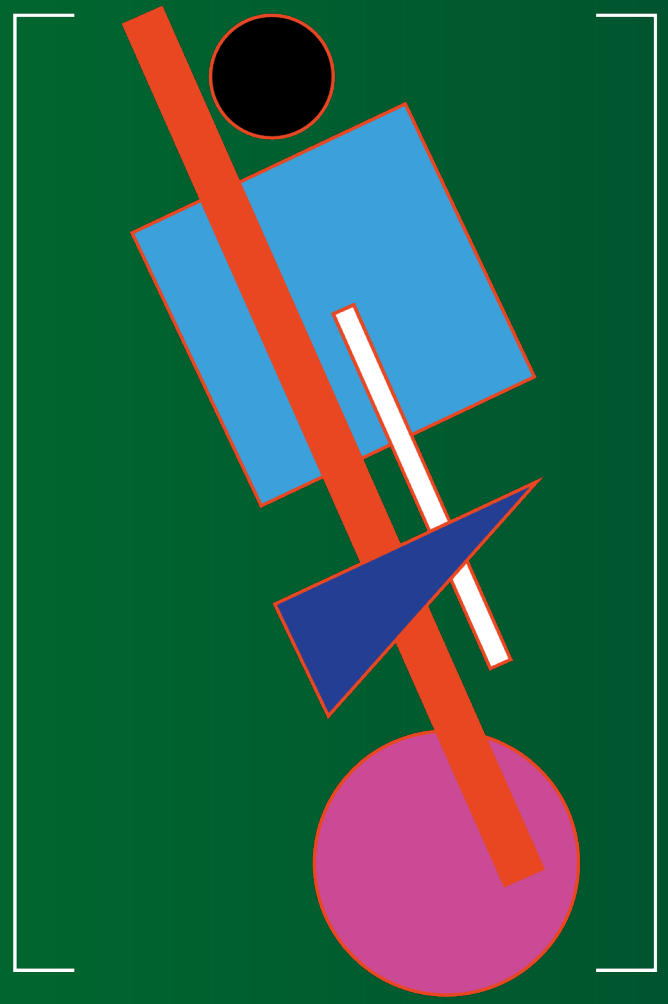
sr_plan - как использовать?

- Вносим sr_plan в shared preload libraries
- Create extension sr_plan(но не обязательно)
- Sr_plan.enable = on;
- Sr_plan.auto_tracking = on;
- Смотрим, что нам не нравится в плане запроса, пробуем разные GUC планировщика, чтобы добиться нужного результата
- Замораживаем запрос: SELECT sr_plan_freeze(QueryID, 0);

PostgresPro

PG_HINT_PLAN

Хинты «как в Oracle»



PostgresPro

pg_hint_plan

- /*+ SeqScan(a) */.
- Типы хинтов:
  - Методы сканирования
  - Методы соединения
  - Порядок соединения
  - Поправка кардинальности
  - Хинты для параллельных планов
  - Выставление GUC во время планирования

PostgresPro

pg_hint_plan - как использовать?

- Вносим pg_hint_plan в shared preload libraries
- Create extension pg_hint_plan
- Открываем документацию pg_hint_plan
- Смотрим, что нам не нравится в плане запроса, пробуем разные наборы хинтов, пока результат не устроит
- Вносим итоговый набор хинтов вместе с параметризованным запросом в таблицу pg_hint_plan.hints

PostgresPro

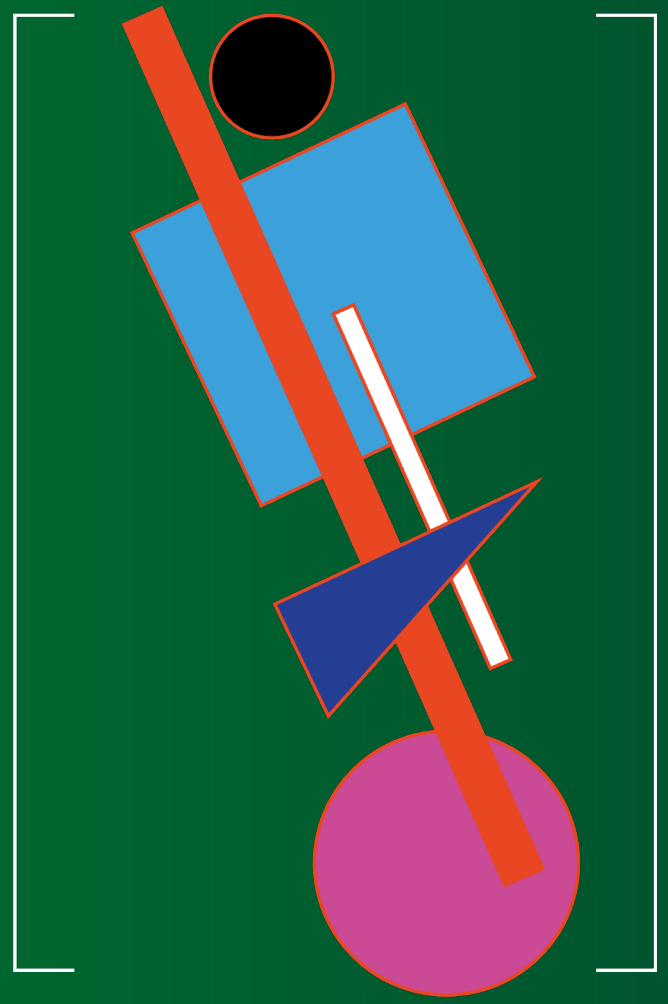
pg_hint_plan - плюсы и минусы

- Плюсы
  - Неинвазивный
  - Нет оверхеда
  - Нет привязки к OIDs
  - Таблица pg_hint_plan.hints
- Минусы
  - Таблица pg_hint_plan.hints
  - Синтаксис
  - Сложно писать комплексные хинты

PostgresPro

УЛУЧШАЕМ И ПЕРЕНОСИМ ПЛАН

AQO + sr_plan + pg_hint_plan



PostgresPro

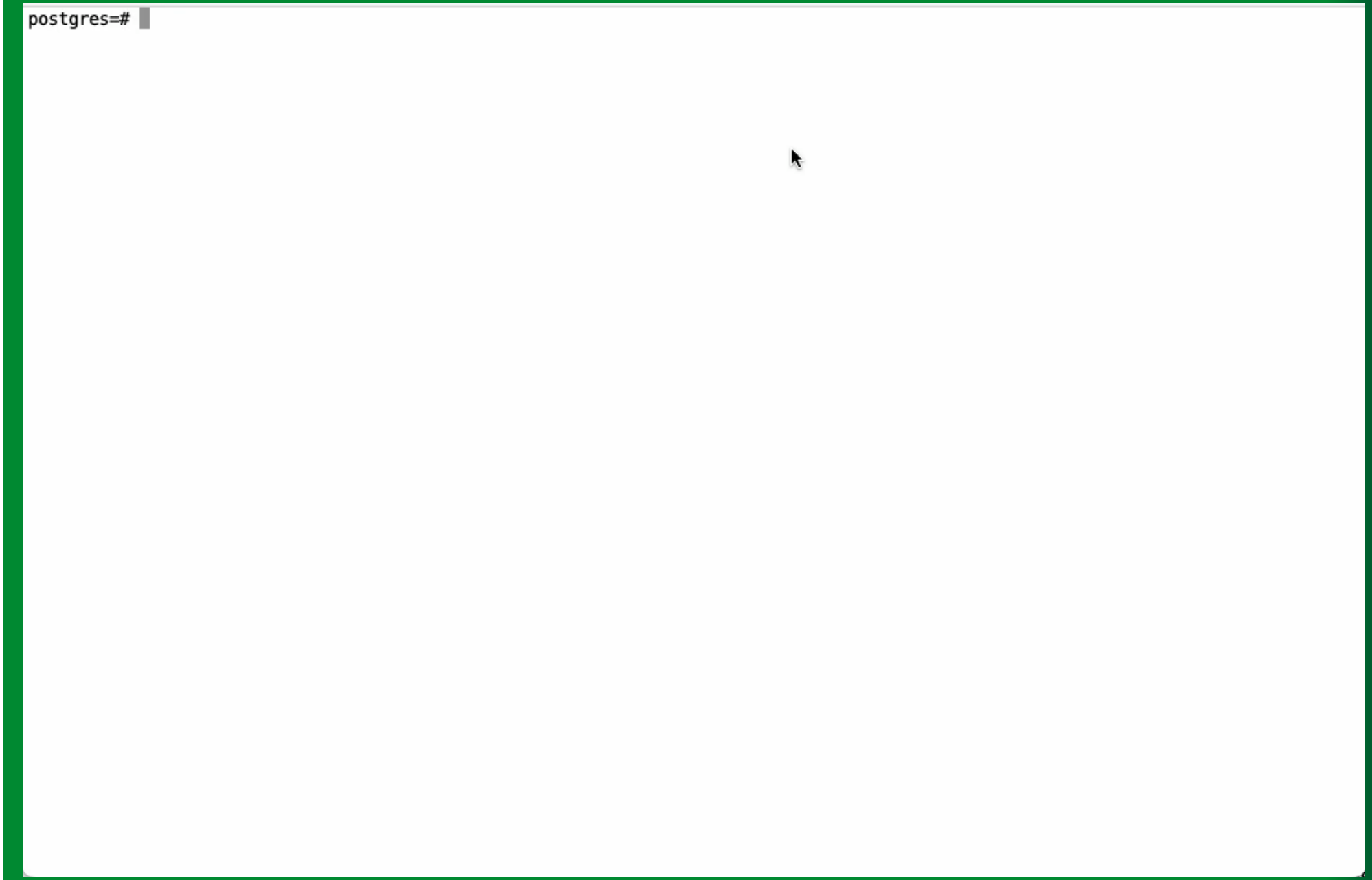
Sr_plan 0.5 – hintset plans!

```
/*+
Leading((((((((((cn mc )ci )mk )k )mi )n )t )mi_idx )it2 )it1 ))
NestLoop(cn mc) NestLoop(ci cn mc) NestLoop(ci cn mc mk) NestLoop(ci cn k mc mk) NestLoop(ci cn k mc mi mk) NestLoop(ci cn k mc mi mk n) NestLoop(ci cn k mc mi mk n t) NestLoop(ci cn k mc mi mi_idx mk n t) HashJoin(ci cn it2 k mc mi mi_idx mk n t) HashJoin(ci cn it1 it2 k mc mi mi_idx mk n t)
IndexScan(ci movie_id_cast_info) SeqScan(cn) SeqScan(it1) SeqScan(it2) IndexScan(k keyword_pkey) IndexScan(mc company_id_movie_companies) IndexScan(mi movie_id_movie_info) IndexScan(mi_idx movie_id_movie_info_idx) IndexScan(mk movie_id_movie_keyword) IndexScan(n name_pkey) IndexScan(t title_pkey)
*/
```

PostgresPro

Hintset plans

демо



PostgresPro

AQO + sr_plan + pg_hint_plan

- !!! Порядок такой:

shared_preload_libraries = 'pg_hint_plan, sr_plan, aqo'

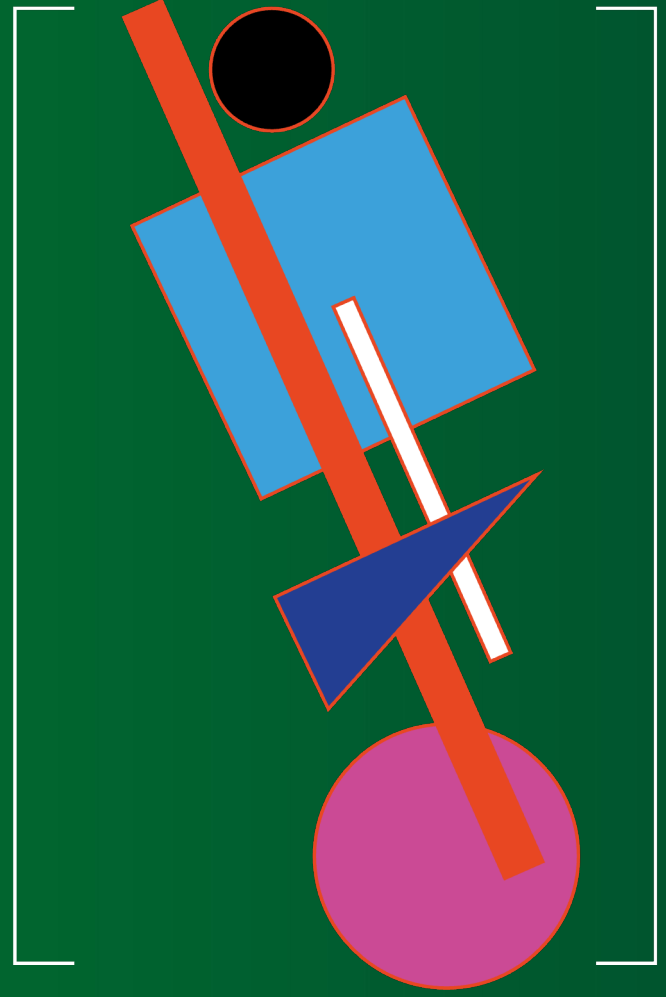
PostgresPro

/Комбинированная техника/
sr_plan + pg_hint_plan и AQO для plan discovery

- Когда стоит применять?
  - Мажорный апгрейд
  - Бекап планов
  - Поиск планов на логической реплике и перенос на прод
  - Хинты по queryID (без таблицы hints)
  - 1С

PostgresPro

Ближайшие планы



PostgresPro

Ближайшие планы

- aqo.wal_rw
- sr_plan.wal_rw
- aqo.use_on_standby
- sr_plan.use_on_standby
- sr_plan query capture
- SQL_Hash
- aqo autonomous mode

PostgresPro

Q&A