PostgresPro

Postgres Pro AXE 2.0

Аналитика работает как обычный Postgres. Обзор новых возможностей



Дмитрий Иванкин
Старший технический консультант

PostgresPro

Аналитика данных с точки зрения бизнеса

Что нужно рынку?



Бизнес ищет аналитический стек

PostgresPro

С низкой стоимостью эксплуатации и высокой производительностью

Производительность на больших данных

47% компаний считают её триггером перехода на новую аналитическую СУБД

Простые внедрение и эксплуатация

34% организаций ориентируется на этот критерий при выборе нового решения

Снижение TCO

Работы интеграторов, масштабирование, лицензии и поддержка — ключевые статьи расходов

Realtime без обходных решений

60% опрошенных не имеют полноценной работы с realtime-данными без ограничений

Российский стек и поддержка

31% респондентов отмечают необходимость российской разработки и поддержки СУБД

Какие задачи мы ставили перед собой

PostgresPro

Основная цель: дать бизнесу удобный продукт для построения быстрой и экономичной аналитической платформы

Быстрые аналитические запросы

на ограниченных ресурсах

Работа

в привычной экосистеме

Эффективное решение задач

с меньшими затратами

Возможность эффективно обрабатывать OLTP и OLAP

нагрузку в рамках одной СУБД

Работа с сотнями ТБ данных

и гибкое масштабирование



Мы справились!

Первые пилотные проекты и внедрения показали на практике, что Postgres Pro AXE помогает нашим клиентам увеличить скорость работы аналитической платформы, при этом снизив необходимые для использования ресурсы

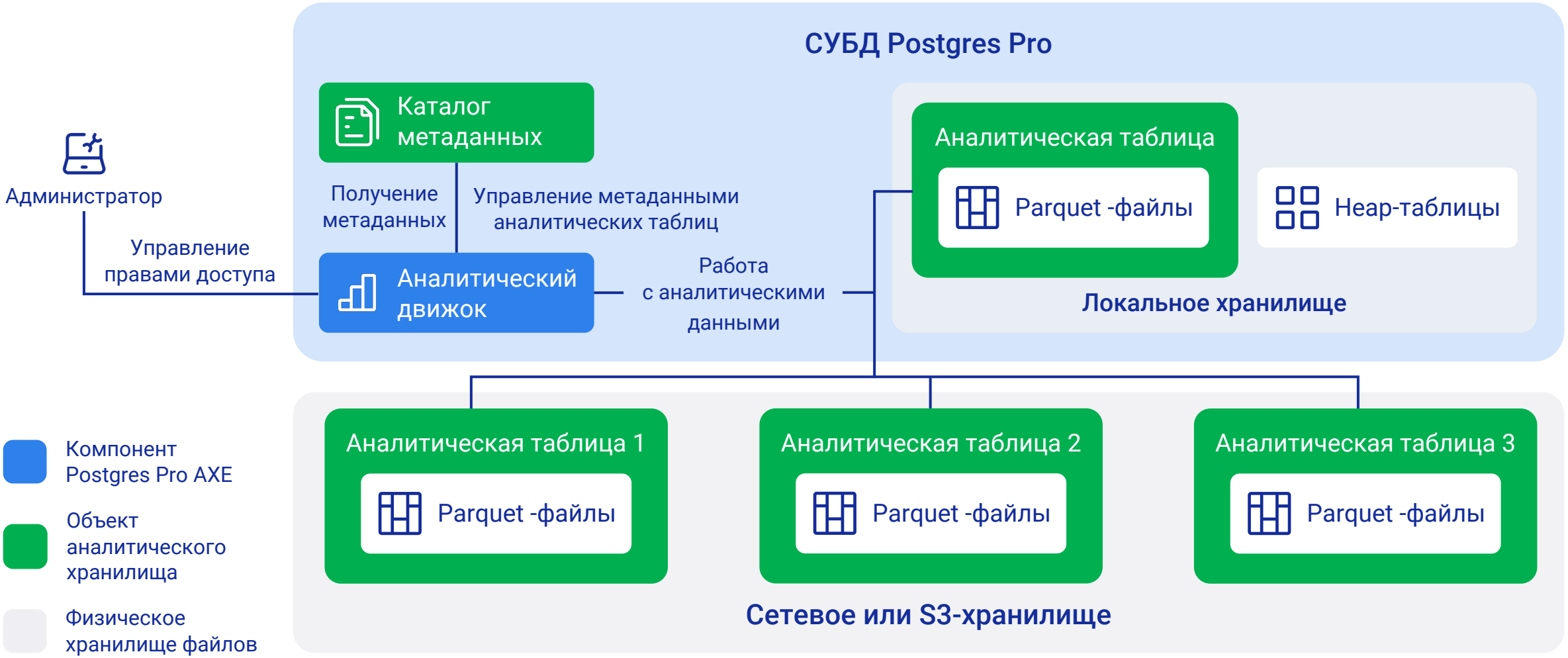


Postgres Pro AXE

За счет чего достигается такая производительность и эффективность



Архитектура Postgres Pro AXE*



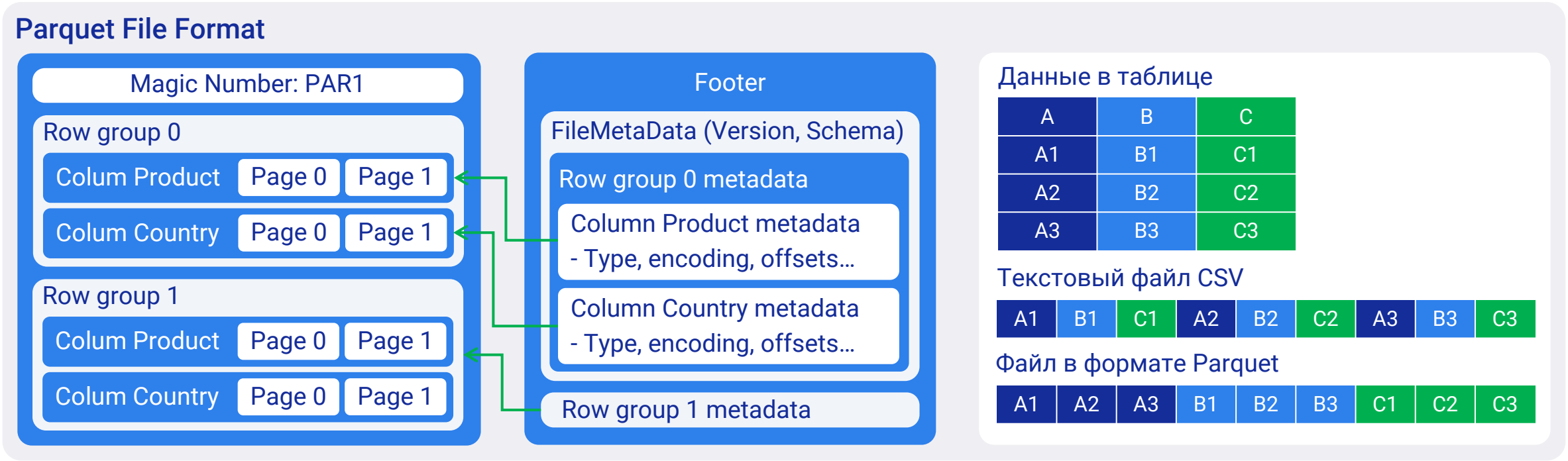
* Postgres Pro AXE может выступать как опция к Postgres Pro Enterprise 17
Аналитические данные могут храниться на локальных дисках сервера, сетевом хранилище или внешнем S3

Почему аналитические данные хранятся в Parquet-файлах?

PostgresPro

Parquet-файлы — колоночно-ориентированный формат хранения
для эффективной обработки больших объёмов данных

- Позволяет читать только необходимые столбцы
- Статистика и индексы в заголовке каждого файла
- Эффективное сжатие
- Поддерживает сложные структуры данных
- Открытый и распространенный формат



Векторный движок запросов

PostgresPro

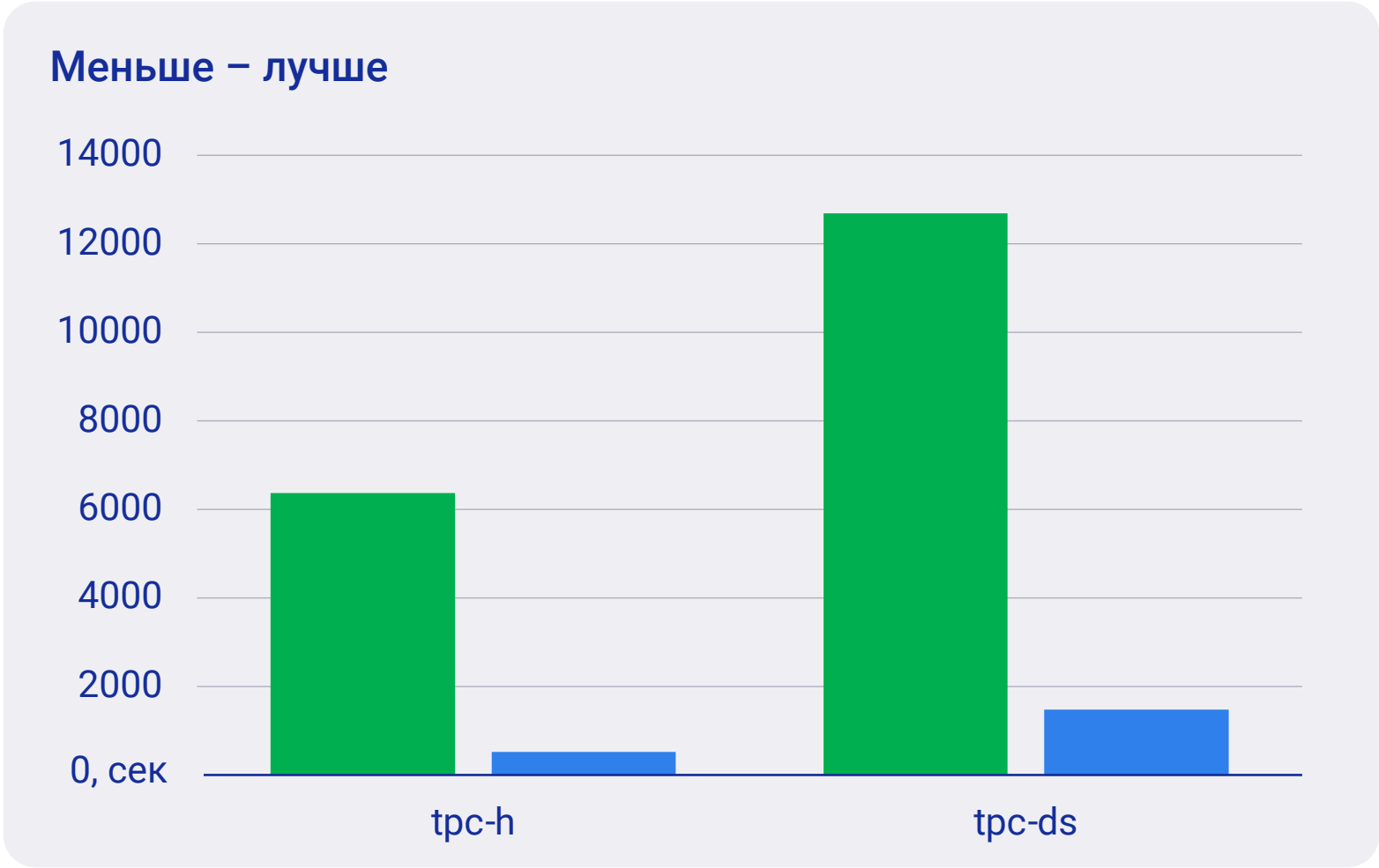
Обрабатывается не одна запись, а пакет из нескольких тысяч значений одного столбца: очень высокая производительность аналитических запросов!

- Векторная обработка данных на уровне движка запросов
- Применение векторных вычислений на уровне CPU (SIMD команды)
- Мощный оптимизатор
- Быстрее большинства аналогов



Тесты tpc-h, tpc-ds:
полное время, 1 клиент, sf=1 000 Гб

PostgresPro



до 10 раз
быстрее

в 4 раза
меньше ресурсов

Greenplum
1 координатор, 3 сегмента аналитики. Итого:
120 CPU | 1840GB RAM

Postgres Pro AXE
1 сервер. Итого:
30 CPU | 460 GB RAM

Целостность данных и безопасность

PostgresPro

Postgres Pro AXE обеспечивает целостность и согласованность данных за счет использования снимков данных (snapshot). Доступ к данным аналитических таблиц осуществляется в единой ролевой модели Postgres Pro

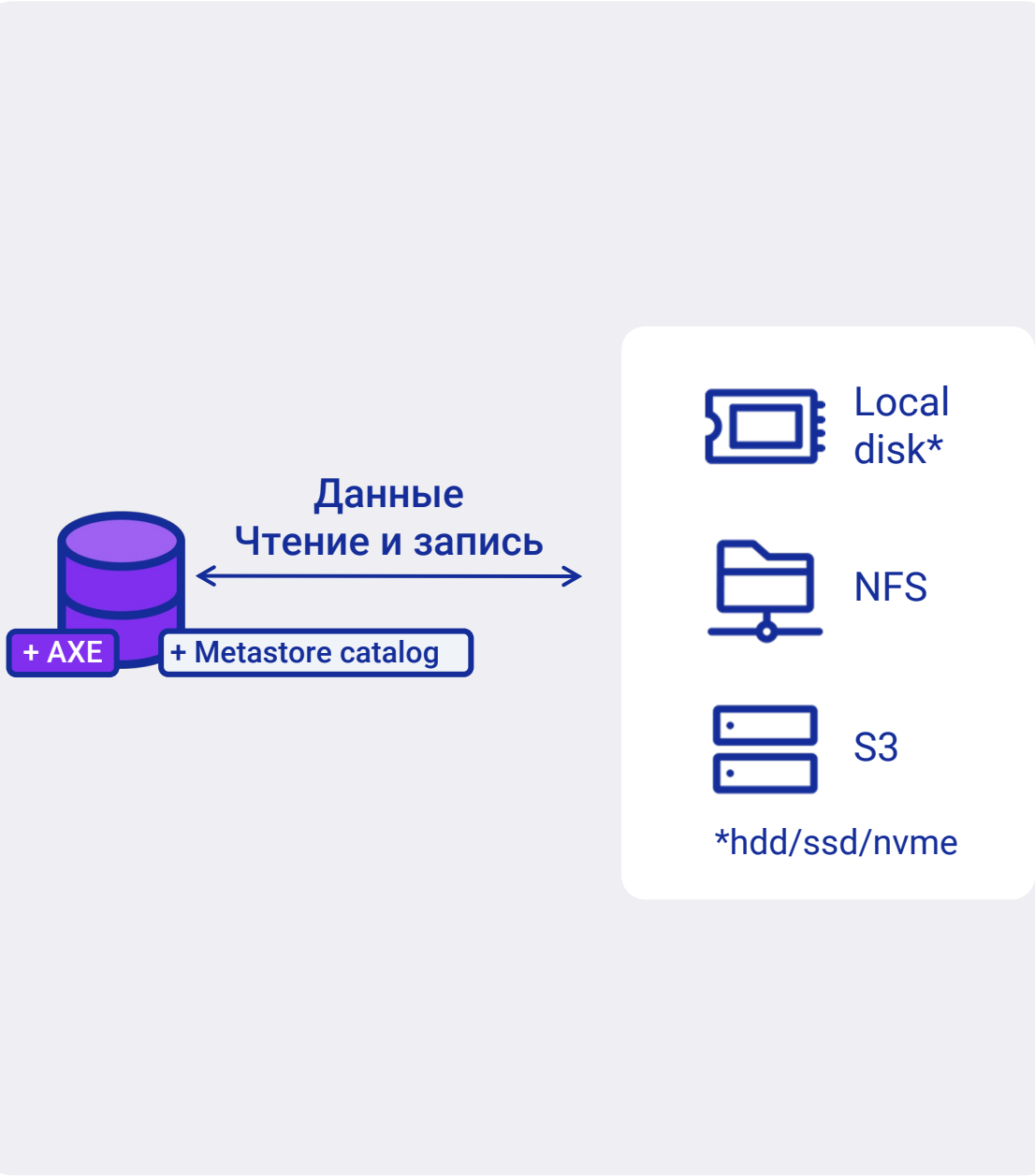
- Управление правами доступа
- Поддержка ACID
- Реализация TimeTravel
- Согласованность метаданных при использовании нескольких реплик Postgres Pro



Хранение данных

Postgres Pro AXE позволяет хранить parquet файлы как в локальном хранилище (рекомендуем быстрые NVME диски), так и в разделяемом хранилище S3 или сетевом хранилище NFS

- Одновременно можно использовать разные хранилища для разных данных
- Гибкий подход к хранению - реализация разных сценариев использования данных
- Простое горизонтальное масштабирование по хранению
- Позволяет эффективно хранить сотни ТБ данных

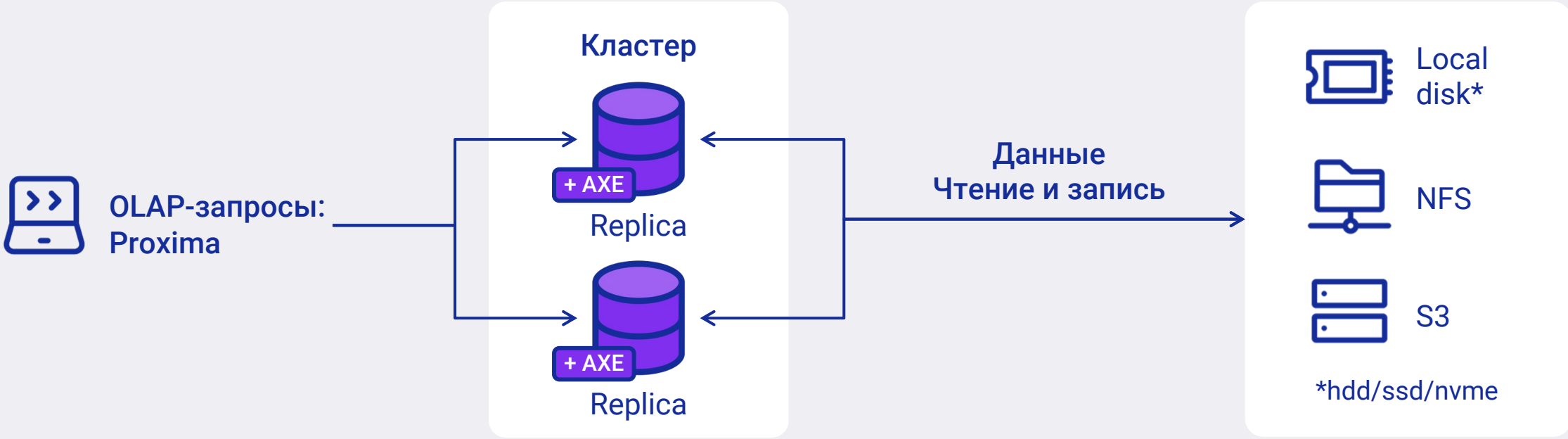


Отказоустойчивость и Масштабируемость

PostgresPro

Postgres Pro AXE использует встроенный инструмент BiHA для построения распределенных геолокационных кластеров и обеспечения гибкой масштабируемости

- Использование разделяемых хранилищ данных для хранения Parquet
- Масштабирование нагрузки за счет реплик



Один продукт — решение основных задач аналитики

PostgresPro

Классический подход

- Используется набор разных технологических решений
- Дополнительные задержки
- Накладные расходы на содержание разных СУБД
- Дополнительные затраты на перекладку данных между разными СУБД

Postgres Pro AXE

- Один продукт вместо несколько баз
- Быстрая аналитика над данными
- Наследование опыта и инструментария Postgres Pro

OLTP СУБД

Источники данных

Загрузка данных из источников

Postgres Pro AXE

Сырые данные • Детальные данные
Витрины данных

BI–платформа

Визуализация данных

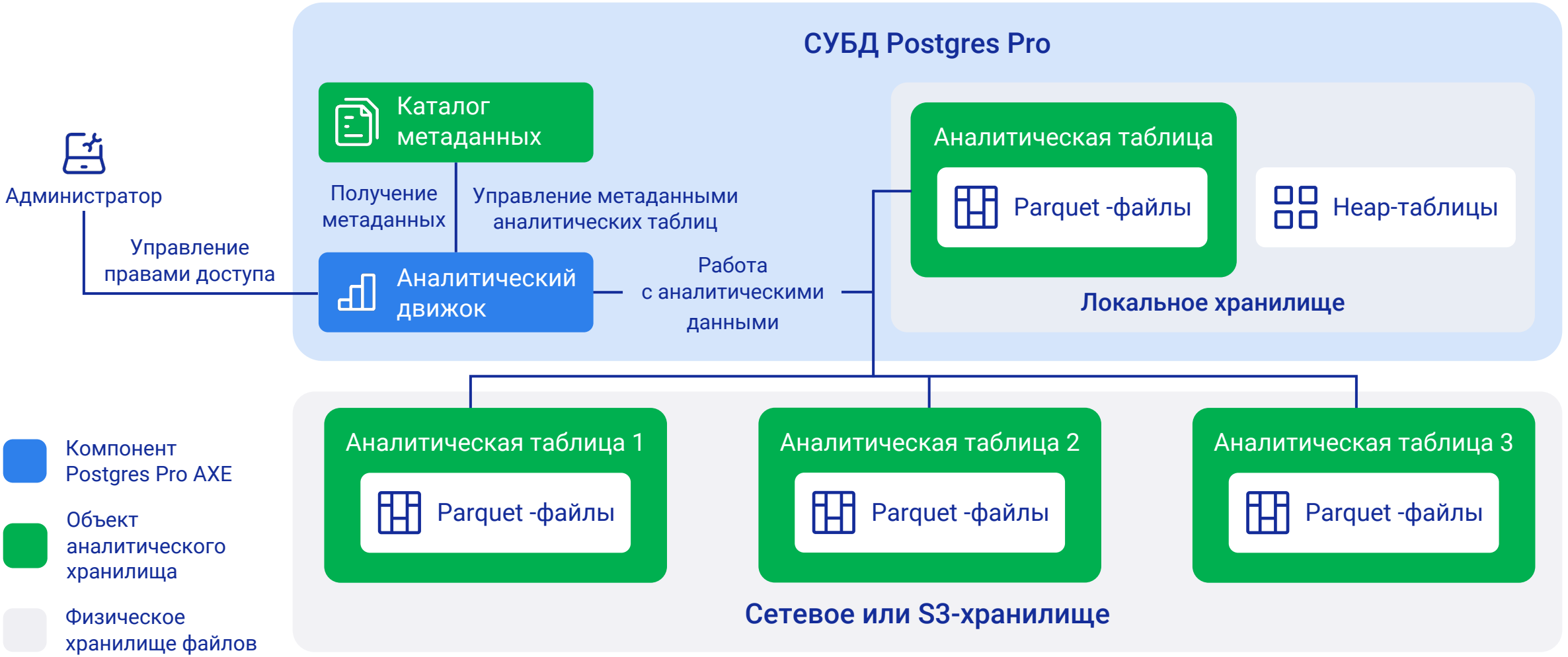
PostgresPro

Postgres Pro AXE 2.0

Направление развития

Концепция остается той же*

PostgresPro



* Postgres Pro AXE может выступать как опция к Postgres Pro Enterprise 17
Аналитические данные могут храниться на локальных дисках сервера, сетевом хранилище или внешнем S3

Аналитика — это не только быстрые запросы

PostgresPro

Быстрые аналитические запросы

на ограниченных ресурсах

Возможность эффективно обрабатывать OLTP и OLAP

нагрузку в рамках одной СУБД

Работа с сотнями ТБ данных

и гибкое масштабирование

Эффективное решение задач

с меньшими затратами

Работа

в привычной экосистеме

Аналитика — это не только быстрые запросы

PostgresPro

Быстрые аналитические запросы
на ограниченных ресурсах

Возможность эффективно обрабатывать OLTP и OLAP
нагрузку в рамках одной СУБД

Работа с сотнями ТБ данных
и гибкое масштабирование

Эффективное решение задач
с меньшими затратами

Работа
в привычной экосистеме

Удобство и простота
работы с данными

PostgresPro

Postgres Pro AXE 2.0

Обзор новой функциональности

Аналитика — это не только быстрые запросы

PostgresPro

От Parquet файлов к Table Access Method

Postgres Pro AXE 1.0	Postgres Pro AXE 2.0
Создание аналитической таблицы через специальные методы Metastore	Использование классического синтаксиса CREATE TABLE
SELECT через промежуточное представление	SELECT напрямую к таблице
Изменения данных через перезапись целого файла/партиции	Изменение данных через Insert/Update/Delete

Работа с колоночными данными как с обычной таблицей

PostgresPro

- Достаточно при создании таблицы (CREATE TABLE) указать USING axe
- Для работы с данными в такой таблице можно использовать все привычные нам SQL команды
- Пользователь может даже не знать, что работает с аналитической таблицей, данные которой хранятся в Parquet-файлах
- Удобная модификация данных при помощи INSERT/UPDATE/DELETE операторов

```
CREATE TABLE axe_example (
  id INT,
  product_name TEXT,
  price NUMERIC(10,2),
  created_at TIMESTAMP
) USING axe;

CREATE TABLE axe_table USING axe
AS SELECT * FROM public.axe_heap_table
```

Целостность данных и безопасность

PostgresPro

Postgres Pro AXE обеспечивает целостность и согласованность данных за счет использования снимков данных (snapshot). Доступ к данным аналитических таблиц осуществляется в единой ролевой модели Postgres Pro

- Управление правами **доступа при помощи единого и привычного инструмента (GRANT/REVOKE)**
- Поддержка ACID
- Реализация TimeTravel
- Согласованность метаданных при использовании нескольких реплик Postgres Pro



PostgresPro

Как это работало в Postgres Pro AXE 1.0

Посмотрим на примере

Создание аналитической таблицы и работа с ней в версии 1.0

PostgresPro



1
```
SELECT metastore.add_table(
 'demo_axe_mart',  -- имя новой аналитической таблицы
 'demo_storage',   -- хранилище
 'bookings.pg_mart_part' -- наследуем структуру су-щей таблицы
);
```

2
```
select metastore.copy_table('demo_axe_mart', $$select * from
bookings.pg_mart_part$$);
```

3
```
select metastore.create_view('demo_axe_mart');
select * from bookings.demo_axe_mart;
```

4
```
select * from bookings.demo_axe_mart;
```

Аналитическая таблица в 1.0 и технические ограничения

PostgresPro

- Пользователь не взаимодействует напрямую с аналитической таблицей
- Работа с данными через промежуточное view
- Использование функций metastore вместо привычного SQL
- Отсутствие простого способа модифицировать данные



PostgresPro

Как это работает в Postgres Pro AXE 2.0

Посмотрим на примере



Создание аналитической таблицы и работа с ней в версии 2.0

PostgresPro

Создание аналитической таблицы

Работа с данными напрямую

1

```
CREATE TABLE axe_table USING axe
AS SELECT * FROM public.axe_heap_table;
```

2

```
SELECT * FROM axe_table;
```

PostgresPro

Обновления данных

Сложный вопрос при работе с Parquet



Обновление данных аналитической таблицы в версии 1.0

PostgresPro



Обновление данных аналитической таблицы в версии 1.0

PostgresPro

Найти снапшоты, в которых нужно обновить данные

↓

Выгрузить снапшоты во временную локальную таблицу

↓

Выполнить нужные DML операции над временной таблицей

↓

Залить обновленные данные в аналитическую таблицу как новый снапшот

↓

Пометить старый снапшот как истекший

```
do
$sql$
declare
  snapshots_ids bigint[];
  example_parquet_file text;
  query text;
  rec record;
begin

  -- временно включи флаг, чтобы можно было создать axe-таблицу
  set local duckdb.unsafe_allow_mixed_transactions to true;

  -- найдём снапшоты, которые необходимо извлечь
  with w_cte as materialized(
    select df.begin_snapshot, fcs.min_value, fcs.max_value
    from axe_catalog.pga_table t
    join axe_catalog.pga_data_file df on t.table_id = df.table_id
    join axe_catalog.pga_file_column_statistics fcs on fcs.data_file_id = df.data_file_id
    join axe_catalog.pga_column c on c.column_id = fcs.column_id
    where
      t.table_name = 'repostat_mt_table_local' -- название таблицы в каталоге
      and df.end_snapshot is null
      and c.column_name = 'event_date' )      -- название поля для фильтрации в каталоге
  select coalesce( array_agg( distinct begin_snapshot ), array[]::bigint[])
  into snapshots_ids
  from w_cte wc
  where
    tsrange(
      '2025-01-01',   -- минимальное значение поля
      '2025-01-05',   -- максимальное значение поля
      '[)'
    )
    &&
    tsrange(
      date '1970-01-01' + interval '1d' * wc.min_value::int, -- преобразование date::int из паркета
      date '1970-01-01' + interval '1d' * wc.max_value::int, -- преобразование date::int из паркета
      '[]'
    );
  raise info 'Снапшоты для извлечения: %', snapshots_ids;

  -- найдём последний паркет-файл для этой анатилической табилцы
  select df.path
  into example_parquet_file
  from axe_catalog.pga_table t
  join axe_catalog.pga_data_file df on t.table_id = df.table_id
  where
    t.table_name = 'repostat_mt_table_local' -- название таблицы в каталоге
    and df.end_snapshot is null
  order by df.data_file_id desc
  limit 1;
  raise info 'Файл для извлечения структуры: %', example_parquet_file;

  -- создаём временную таблицу, в которую сохраним данные из снапшотов
  drop table if exists tmp_table_for_axe;
  execute format($q1$ create temp table tmp_table_for_axe using duckdb as select * from read_parquet(%L) with no data $q1$, example_parquet_file);

  -- добавляем во временную таблицу данные из снапшотов
  for rec in
    select df.path
    from axe_catalog.pga_table t
    join axe_catalog.pga_data_file df on t.table_id = df.table_id
    where
      t.table_name = 'repostat_mt_table_local' -- название таблицы в каталоге
      and df.end_snapshot is null
      and df.begin_snapshot = any(snapshots_ids)
  loop

    -- загружаем данные из файла
    query := format(
      $q1$
        insert into tmp_table_for_axe
        select %s from read_parquet(%L) r
      $q1$,
      (
        SELECT string_agg( format('r[%L]::%s as %I', column_name, data_type, column_name), ', ')
        FROM INFORMATION_SCHEMA.COLUMNS
        WHERE table_name = 'tmp_table_for_axe'
      ),
      rec.path
    );
    raise info 'query: %', query;
    execute query;

  end loop;

  -- выполняем нужные нам DML-запросы
  update tmp_table_for_axe set data5 = -1;

  -- заливаемомодифицированные данные в аналитическую таблицу
  perform metastore.copy_table('repostat_mt_table_local', 'select * from tmp_table_for_axe');
  -- А старые снапшоты, в которых лежат данные до модификации удаляем
  perform metastore.expire_snapshot_on_ids(snapshots_ids);
  perform metastore.delete_expired_files();
end;
$sql$;
```

Обновление данных аналитической таблицы в версии 2.0

PostgresPro

Выполняем обычный UPDATE

1

```
UPDATE testtable
SET data1 = 123 WHERE id = 1;
```



Аналитическая таблица в версии 2.0

PostgresPro

- Пользователь работает напрямую с аналитической таблицей
- Стандартные DML операции (INSERT/UPDATE/DELETE)
- Все операции выполняются при помощи стандартных SQL инструкций
- Набор встроенных служебных процедур для обслуживания аналитических таблиц



Postgres Pro AXE 2.0

Перейдем к демонстрации

Демонстрационный стенд и сценарий

PostgresPro

Параметры стенда

32 CPU, 94 GB RAM

Демо база авиаперевозок

14 млн записей (1 год данных)

Наглядно покажем ускорение

аналитических запросов
в AXE по сравнению
с обычным Postgres Pro

Рассмотрим на практике

сценарий создания
аналитической витрины
в Postgres Pro AXE

Продемонстрируем сценарии

изменения данных
аналитических таблиц

Покажем как обновление и удаление данных

работает на уровне parquet
файлов

Что мы увидели

PostgresPro

Скорость создания аналитической витрины

14 млн записей: 14 секунд

Выполнение аналитического запроса

4 секунды в Postgres Pro AXE

против

63 секунд в OLTP Postgres

Быстрое обновление данных в аналитической витрине

3,5 млн записей: 3 секунды



На чем фокус дальше?

Автоматизация подбора параметров для оптимальной настройки

Оптимизация использования ресурсов

PostgresPro

Ответим на ваши вопросы!

Почта
info@postgrespro.ru

Сайт
https://postgrespro.ru



Postgres Pro AXE