

PostgresPro x BI.Qube

Быстрый старт DWH и аналитики на ваших данных

Дмитрий Иванкин
Старший технический консультант
Postgres Pro

Дмитрий Понкратенко
Архитектор
BI.Qube



- 1. Хранилища данных для аналитики:**
специфика и архитектурные подходы для импортонезависимых решений
- 2. Особенности Postgres Pro Enterprise,**
облегчающие построение хранилищ данных
- 3. Инструменты для автоматизации ELT,**
моделирования, формирования витрин, контроля качества данных
- 4. Демонстрация построения DWH на Postgres Pro Enterprise:**
от загрузки данных из 1С, Excel, REST API и Kafka до слоя витрин с контролем качества данных
- 5. Что делать, если данных стало много?**
- 6. Демонстрация создания витрин в АХЕ в инструментах BI.Qube**
- 7. Ответы на вопросы**

Хранилища данных для аналитики:

специфика и архитектурные подходы
для импортонезависимых решений

Когда нужен DWH или Data Lakehouse



Технологии

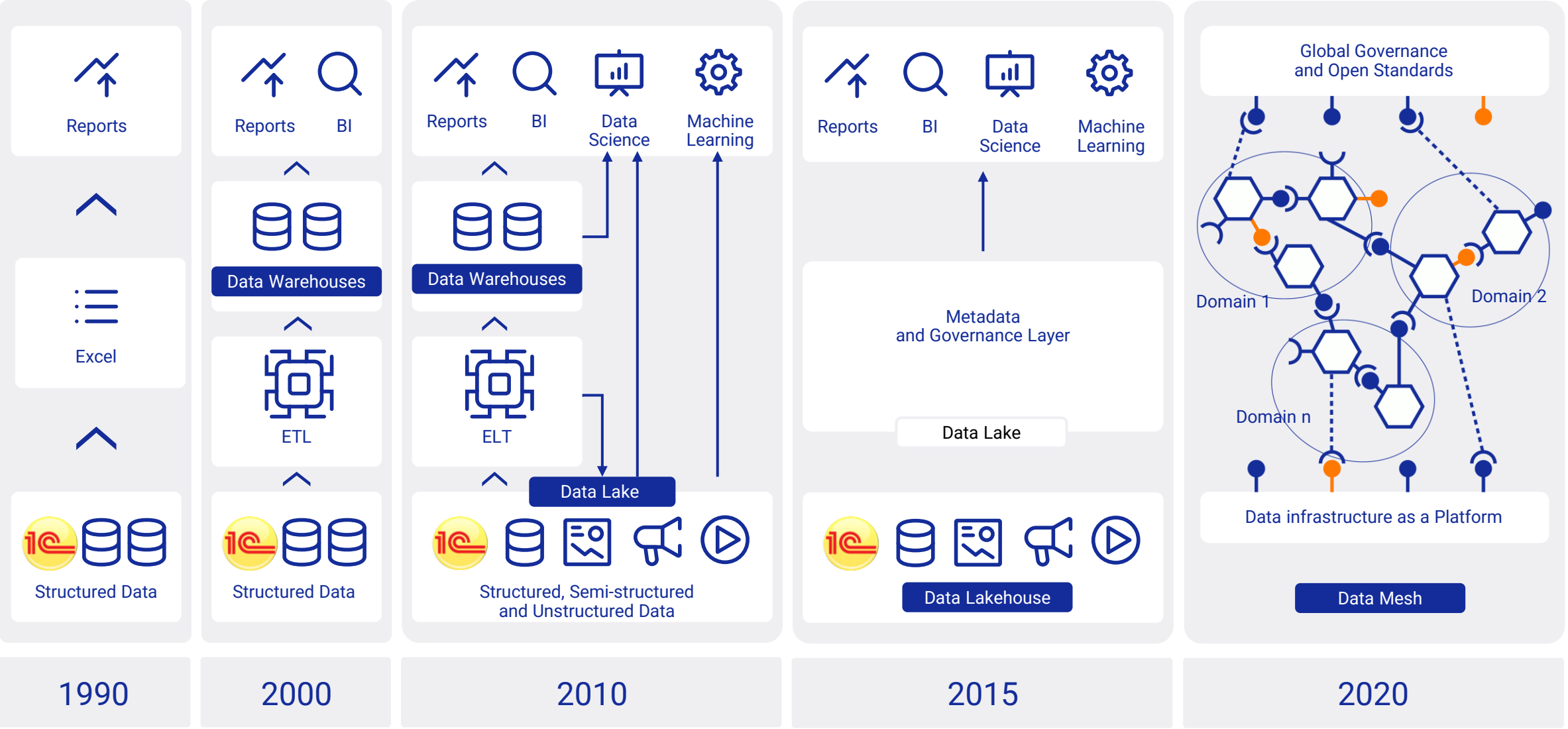
- Объем данных существенный и растёт
- Источники данных разрознены
- Нет единых мастер данных / HСИ / MDM
- Отчеты/выгрузки из 1С тормозят работу пользователей
- Низкая производительность высоконагруженных процессов
- Потребность в решении оптимизационных задач



Заказчики

- Аналитики хотят самостоятельно работать с источниками и формировать витрины и дашборды
- Дата-инженерам требуется регулярное получение актуальной информации из многих систем, включая 1С
- Бизнес хочет быстрый старт внедрения неформализованных бизнес-методик, настройку правил и алгоритмов

Эволюция подходов к анализу данных





Подключение источников

Транзакционная система тормозит во время выгрузки данных для аналитики



Быстрая обработка

Долгая переливка данных, фильтрация сквозь исторические периоды зависает



Хранение и доступность

Хранение историчных данных стоит столько же, сколько и горячих



Моделирование и масштабирование

Рост масштаба хранилища превращает данные в мусор



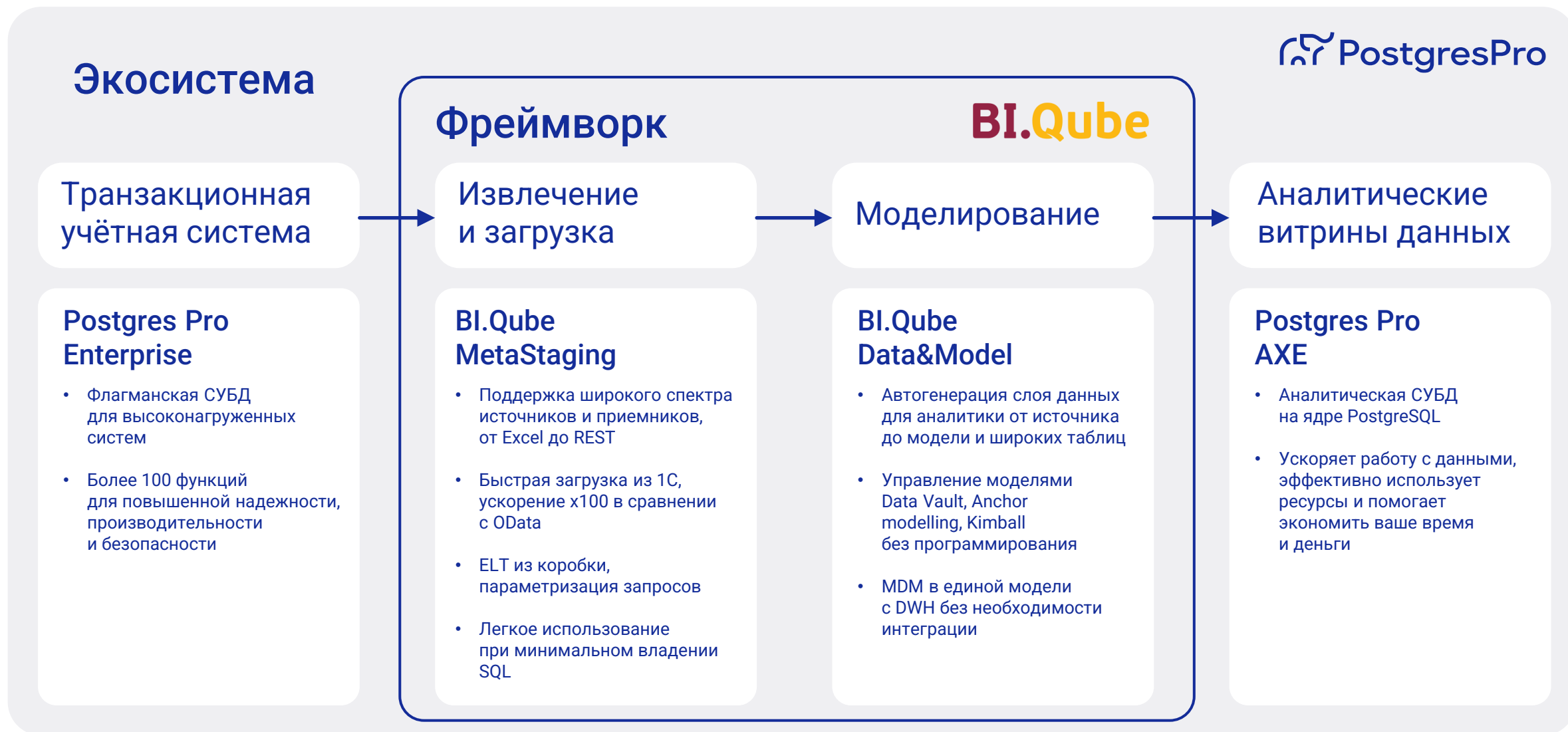
Обеспечение качества

Нет доверия к отчетам и дашбордам из-за отсутствия контроля входящих данных



Развитие и Сопровождение

Медленный time-to-market из-за отсутствия технологии внесения изменений и контроля (CI/CD)



Особенности Postgres Pro Enterprise, облегчающие построение хранилищ данных

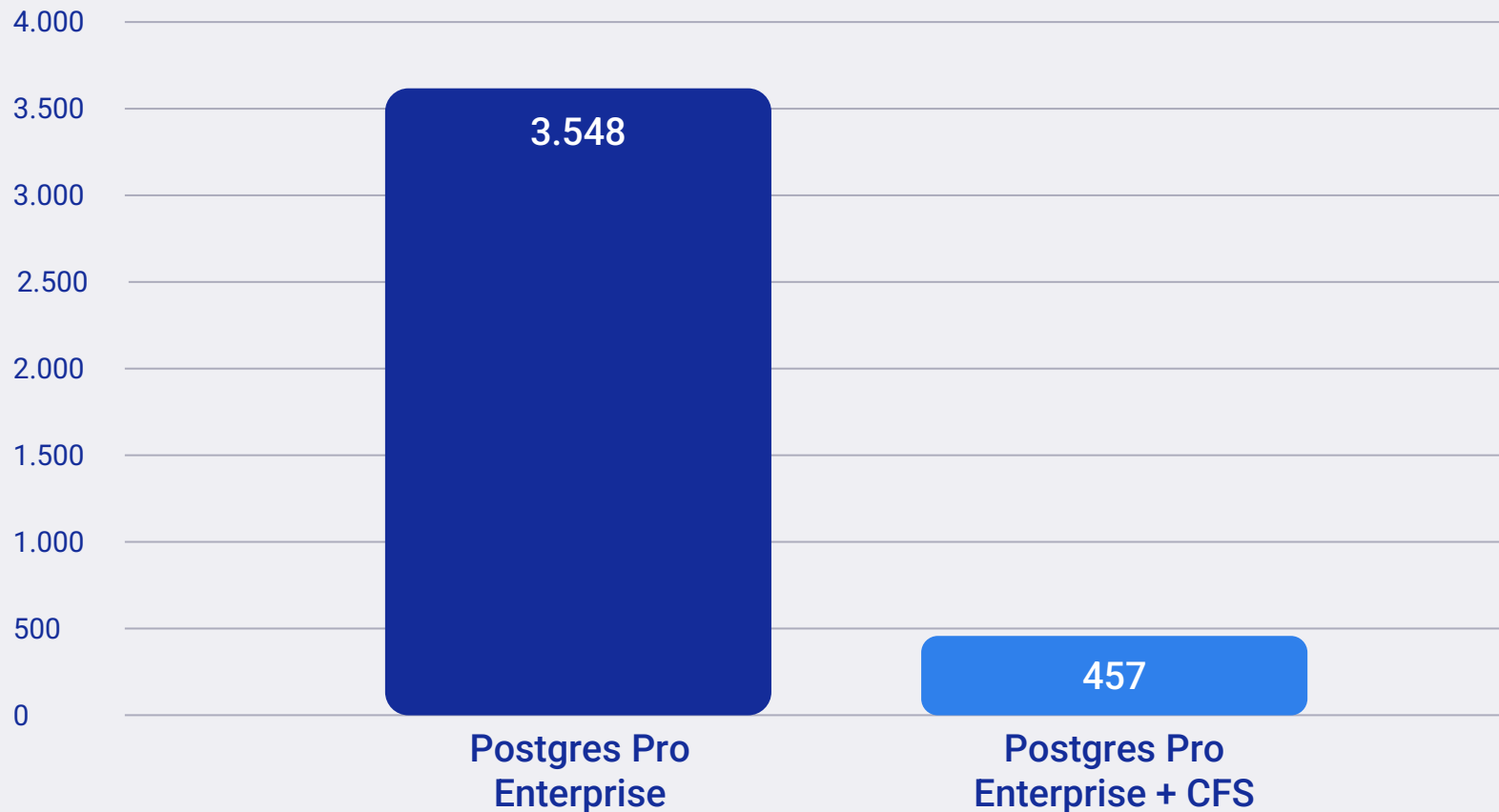
Postgres Pro Enterprise для Хранилищ Данных

1. Сжатие данных на уровне страниц (CFS)
2. ILM - управление жизненным циклом данных
3. Разделяемый Result Cache
4. Append Optimized Table
5. Мультисегментная вставка
6. Global Index
7. Тюнинг производительности (AQO, AQE)
8. Отказоустойчивость

CFS - Сжатие данных

CFS – сжатие данных

450 типовых баз 1С (БП, ЗУП, УТ, УНФ, КА)



3-5x

экономия на дисковом пространстве

Прозрачно

для приложений
и производительности

CFS – простота использования

Включается для табличных пространств

```
CREATE TABLESPACE cfs_tbs LOCATION  
'/var/data/cfs' WITH  
(compression='zstd');
```

Поддерживаются разные алгоритмы сжатия:

- Zstd
- Pglz
- zlib
- lz4

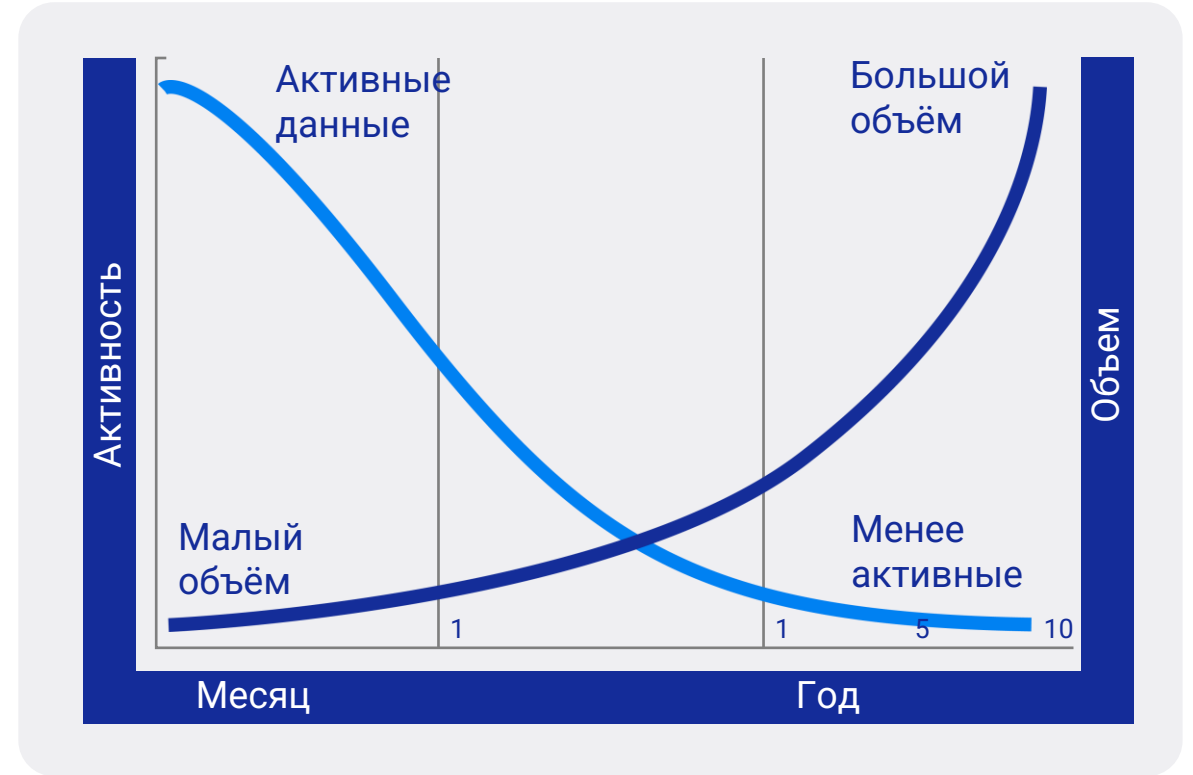
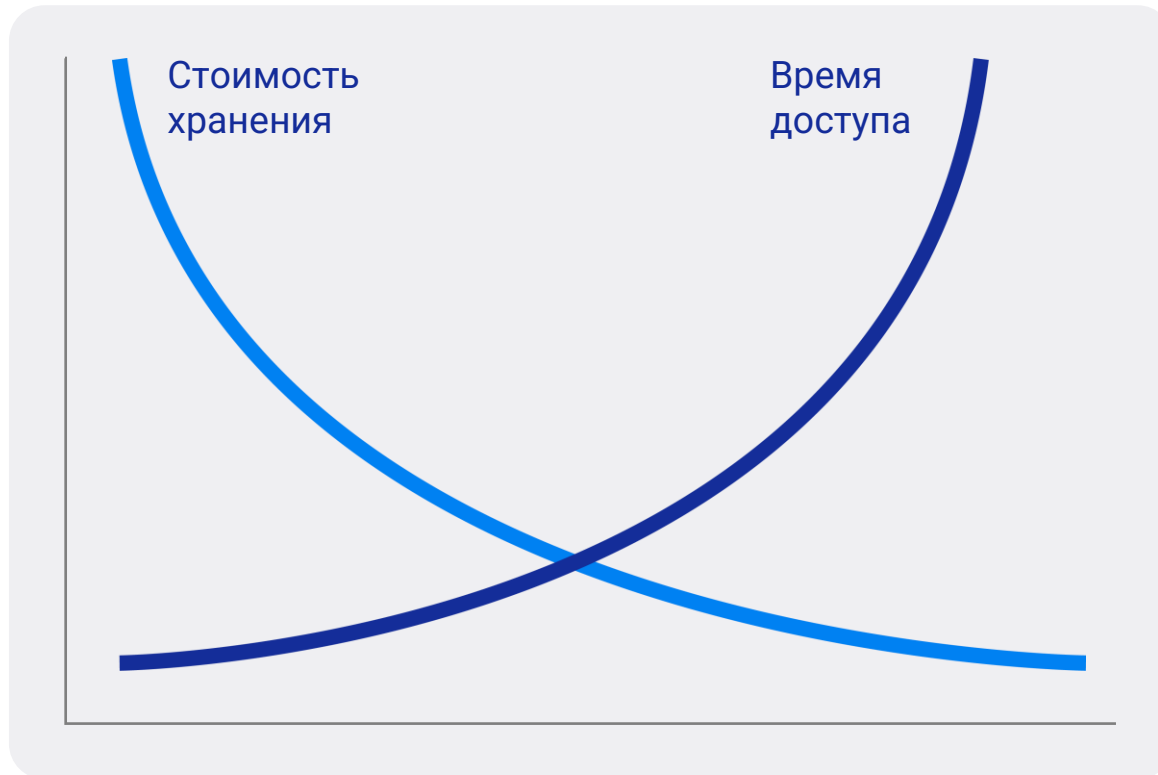
Управление уровнем сжатия
(от 0 до 19)

Управление жизненным циклом информации — ILM

Information Lifecycle Management

- Актуальность данных и активность работы с ними со временем снижается
- Самый большой объём занимают исторические данные

- Активные операции над данными производятся только в течение первых месяцев жизни данных



Проблема больших баз данных

Active — Часто меняющиеся данные (OLTP)

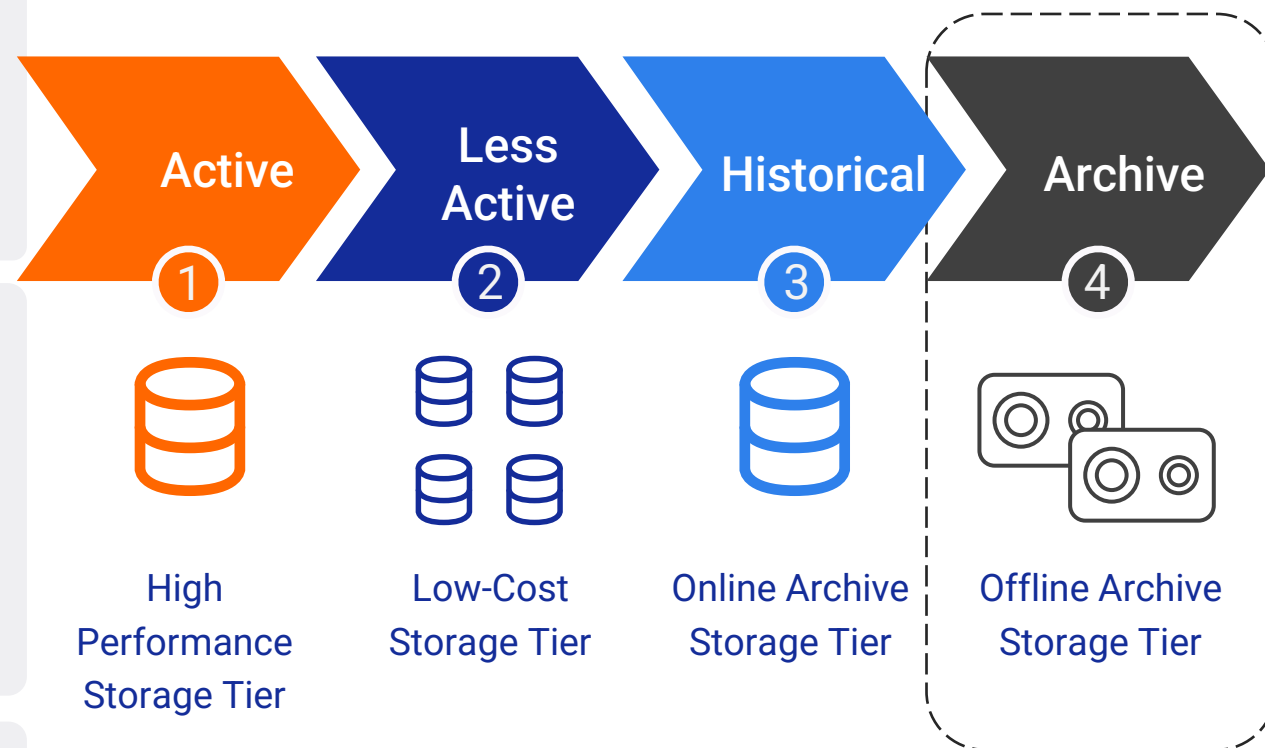
- Очень много чтений в случайных местах
- Пример: таблица финансовых транзакций в АБС, секция (partition) текущего месяца

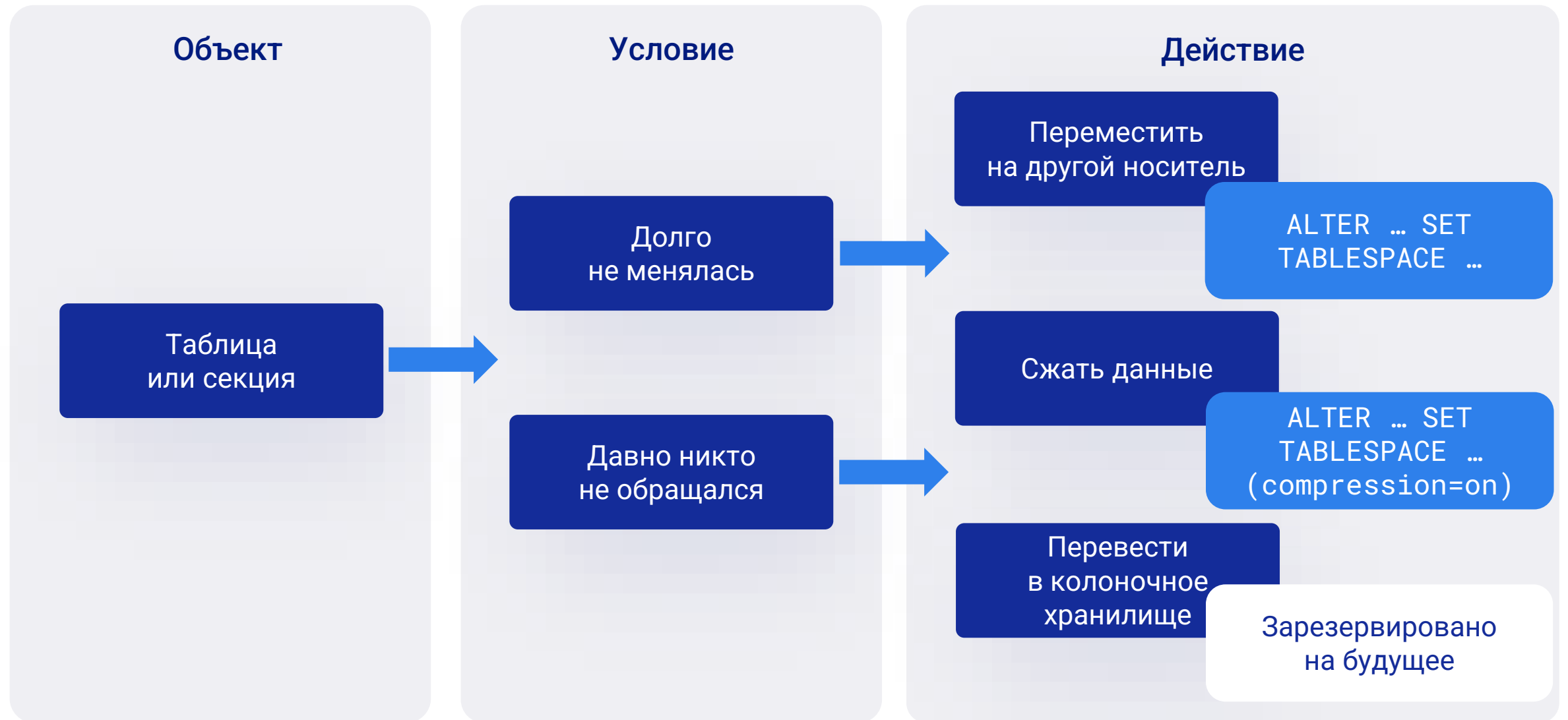
Less Active — Редко меняющиеся данные (OLTP, OLAP)

- Чтения преимущественно сканирующие по колонкам
- Пример: история финансовых транзакций, секции (partitions) текущего года

Historical — Данные не меняются (OLAP)

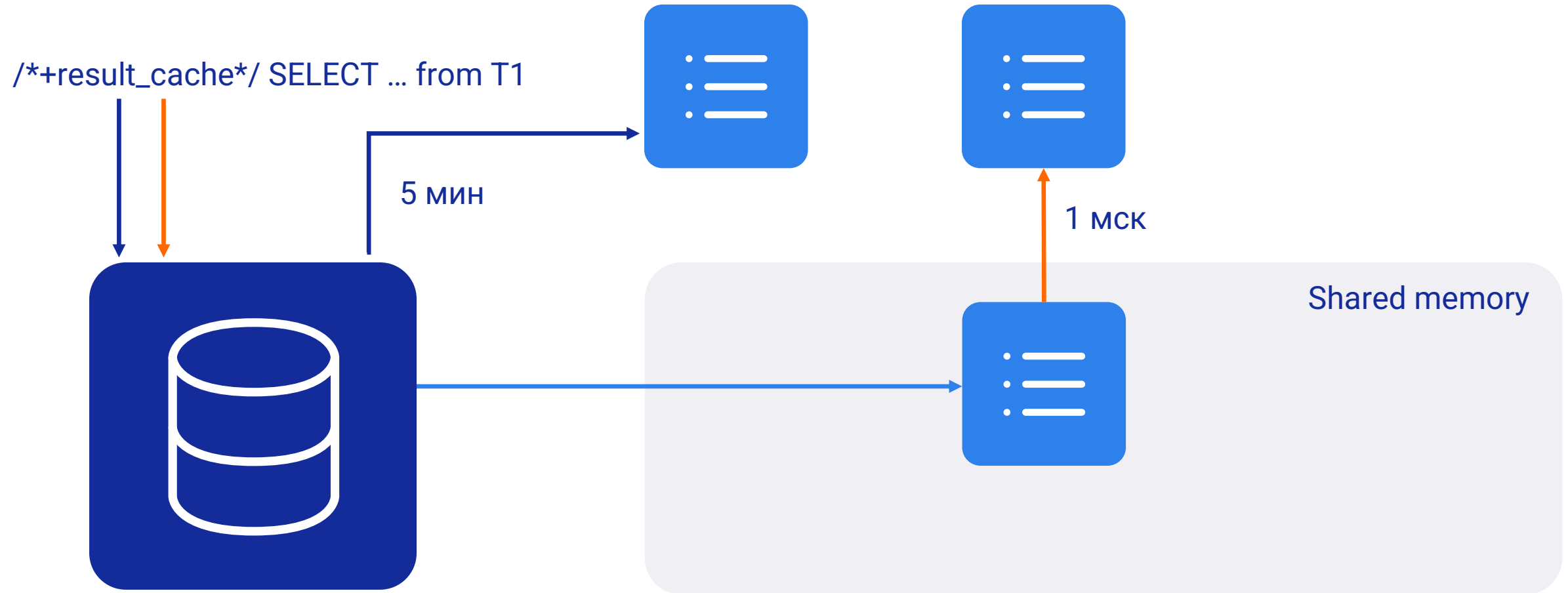
- Редко выполняются сканирующие чтения по колонкам
- Пример: историческая информация о финансовых транзакциях, секции (partitions) за старые годы. Данные старше 3 лет



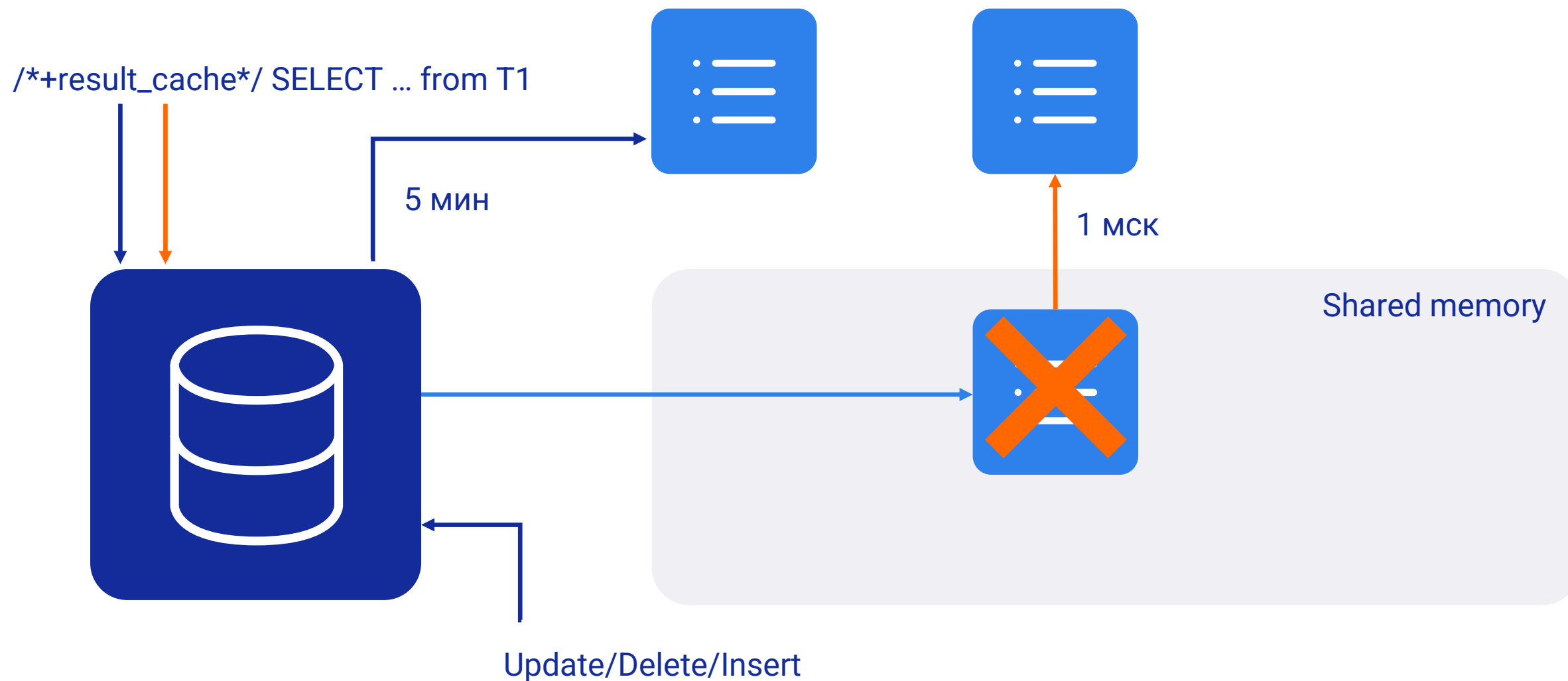


Result Cache

Result cache - быстрое выполнение одинаковых запросов разных сессий

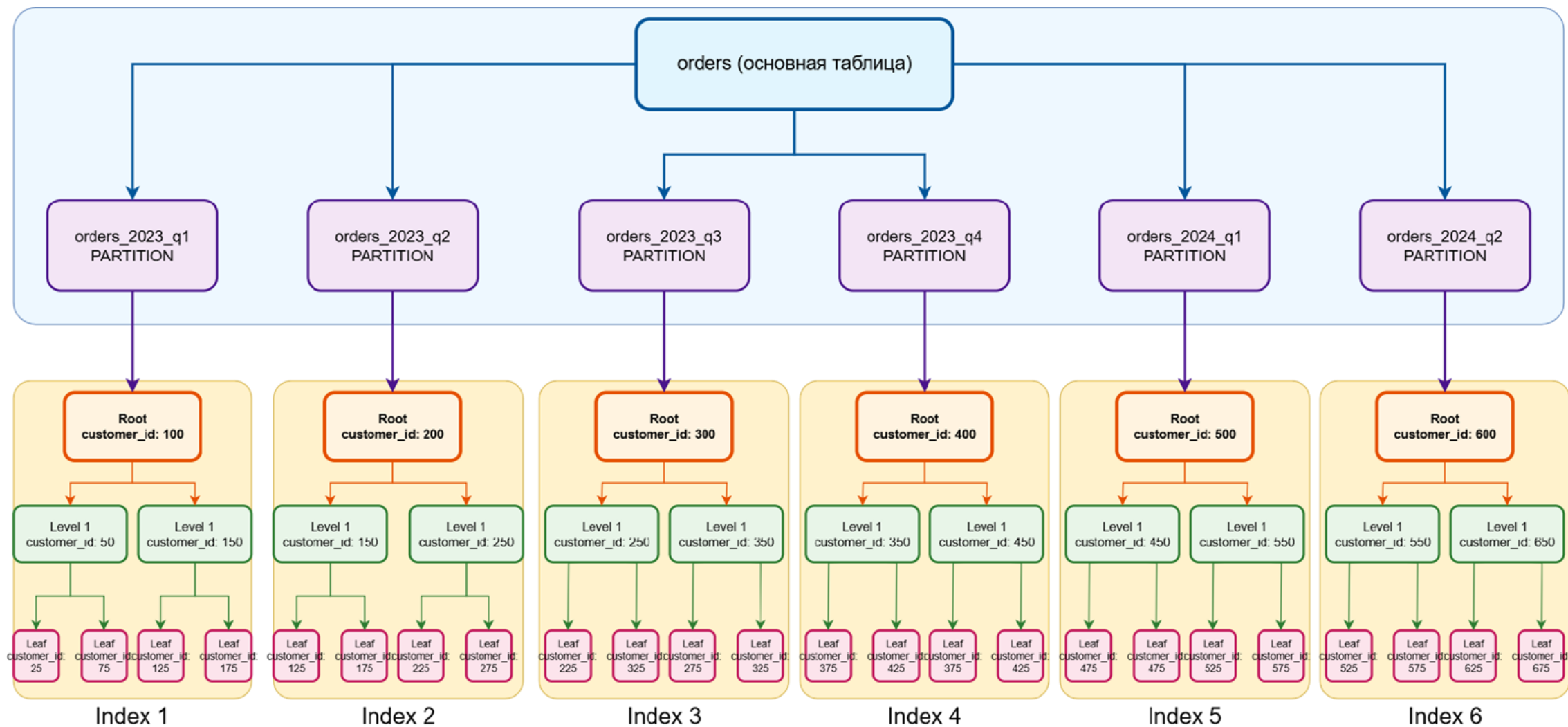


Result cache - быстрое выполнение одинаковых запросов разных сессий

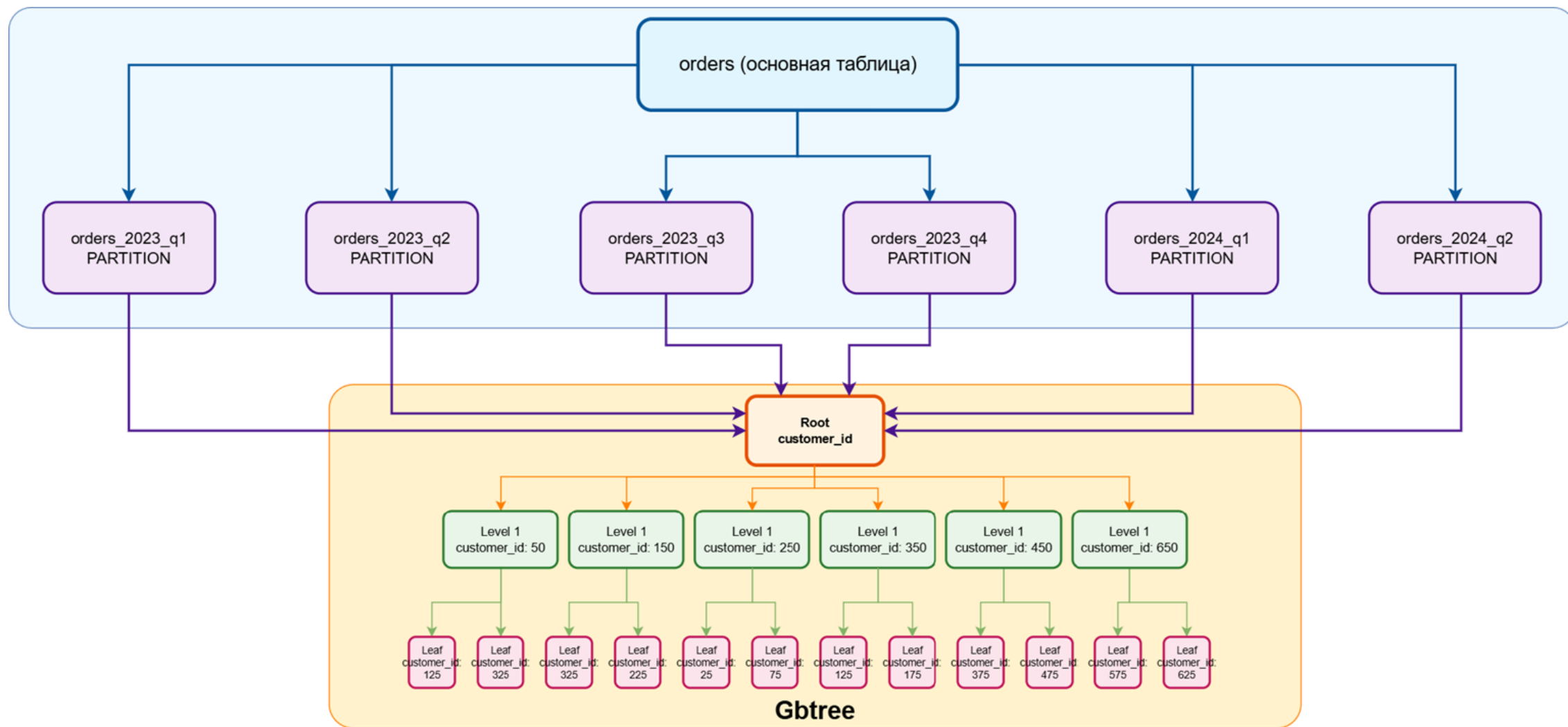


Global Index

Обычный индекс на секционированную таблицу = множество локальных индексов на секции



Глобальный индекс = одно B-дерево на все секции



```
-- Создание расширения pgpro_gbtrees
CREATE EXTENSION pgpro_gbtrees;

-- Создание секционированной таблицы с тремя секциями
CREATE TABLE t_int (i int PRIMARY KEY, v int, x int) PARTITION BY RANGE (i);
CREATE TABLE t_int_1 PARTITION OF t_int FOR VALUES FROM (0) TO (100);
CREATE TABLE t_int_2 PARTITION OF t_int FOR VALUES FROM (100) TO (200);
CREATE TABLE t_int_3 PARTITION OF t_int FOR VALUES FROM (200) TO (300);

-- Создание глобального индекса для столбца v
CREATE INDEX t_int_v ON t_int USING gbtrees (v);
```

Global Index - Пример использования

17.4

QUERY PLAN

```
-----
Index Scan using t_int_v on t_int  (cost=0.12..8.14 rows=0 width=0)
  Index Cond: (v > 100)
```

```
-- План запроса с условием
для столбца с глобальным
индексом
EXPLAIN SELECT * FROM t_int
WHERE v > 100;
```

QUERY PLAN

```
-----
Append  (cost=9.42..93.97 rows=2040 width=12)
-> Bitmap Heap Scan on t_int_1  (cost=9.42..27.92 rows=680 width=12)
    Recheck Cond: (x > 100)
    -> Bitmap Index Scan on t_int_1_x_idx  (cost=0.00..9.25 rows=680 width=0)
        Index Cond: (x > 100)
-> Bitmap Heap Scan on t_int_2  (cost=9.42..27.92 rows=680 width=12)
    Recheck Cond: (x > 100)
    -> Bitmap Index Scan on t_int_2_x_idx  (cost=0.00..9.25 rows=680 width=0)
        Index Cond: (x > 100)
-> Bitmap Heap Scan on t_int_3  (cost=9.42..27.92 rows=680 width=12)
    Recheck Cond: (x > 100)
    -> Bitmap Index Scan on t_int_3_x_idx  (cost=0.00..9.25 rows=680 width=0)
        Index Cond: (x > 100)
```

```
-- План запроса с условием
для столбца с обычным
индексом
EXPLAIN SELECT * FROM t_int
WHERE x > 100;
```

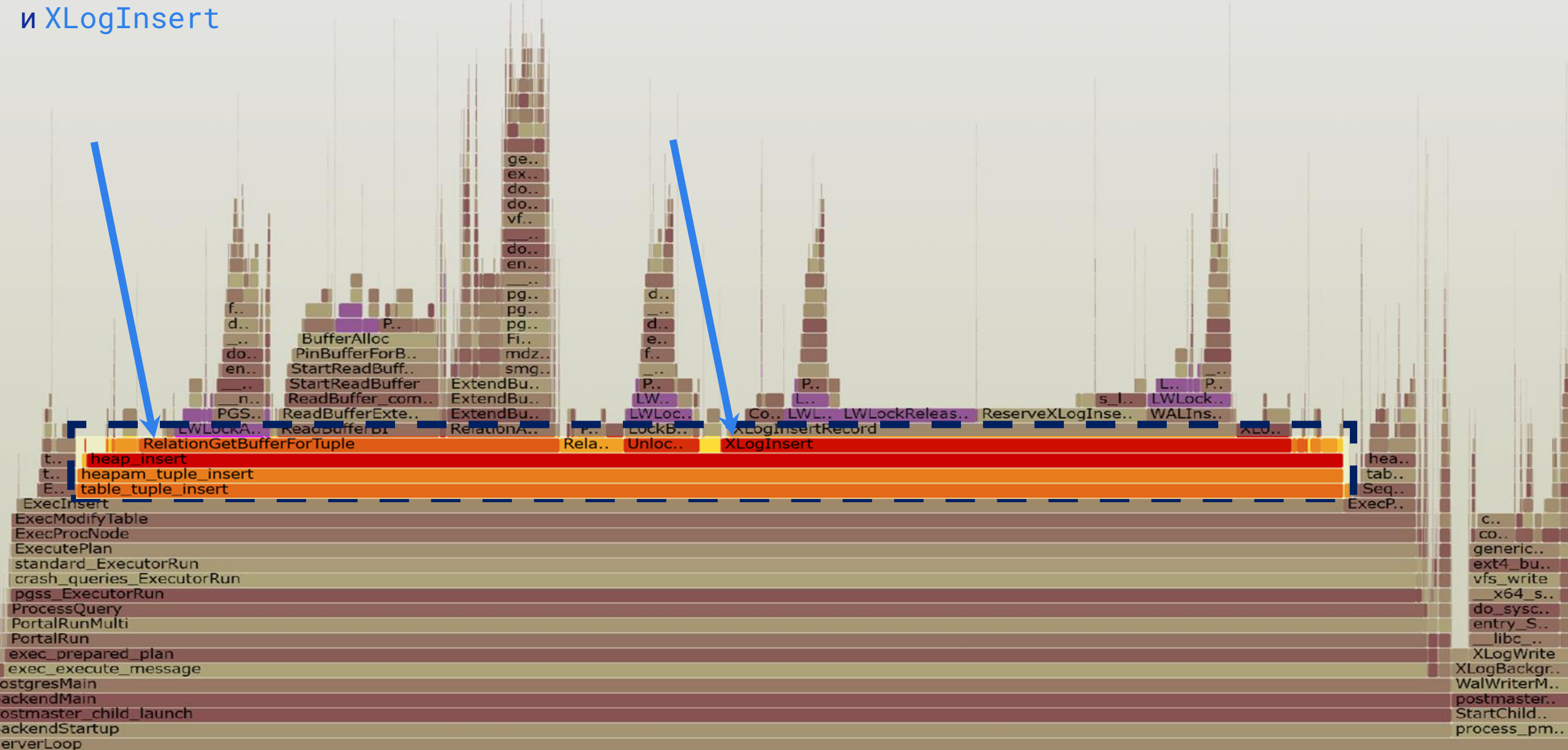
Append Optimized Table

Внутренний механизм Postgres Pro, предназначенный для ускорения массовой вставки данных в таблицы.

Он оптимизирует стандартный процесс вставки, снижая накладные расходы на работу с буферами и WAL, и повышает эффективность при больших пакетах вставки.

На что расходуется больше всего времени при вставке строк?

-> практически всё время от `table_tuple_insert` занимают `RelationGetBufferForTuple` и `XLogInsert`



Append Optimized Table

Цель: повысить производительность операций массовой вставки

За счёт чего:

- Снизить количество обращений к буферному менеджеру
- Уменьшить объём записей в WAL
- Сократить число блокировок на уровне WAL write
- Сохранить совместимость с существующими SQL-запросами

Каким образом:

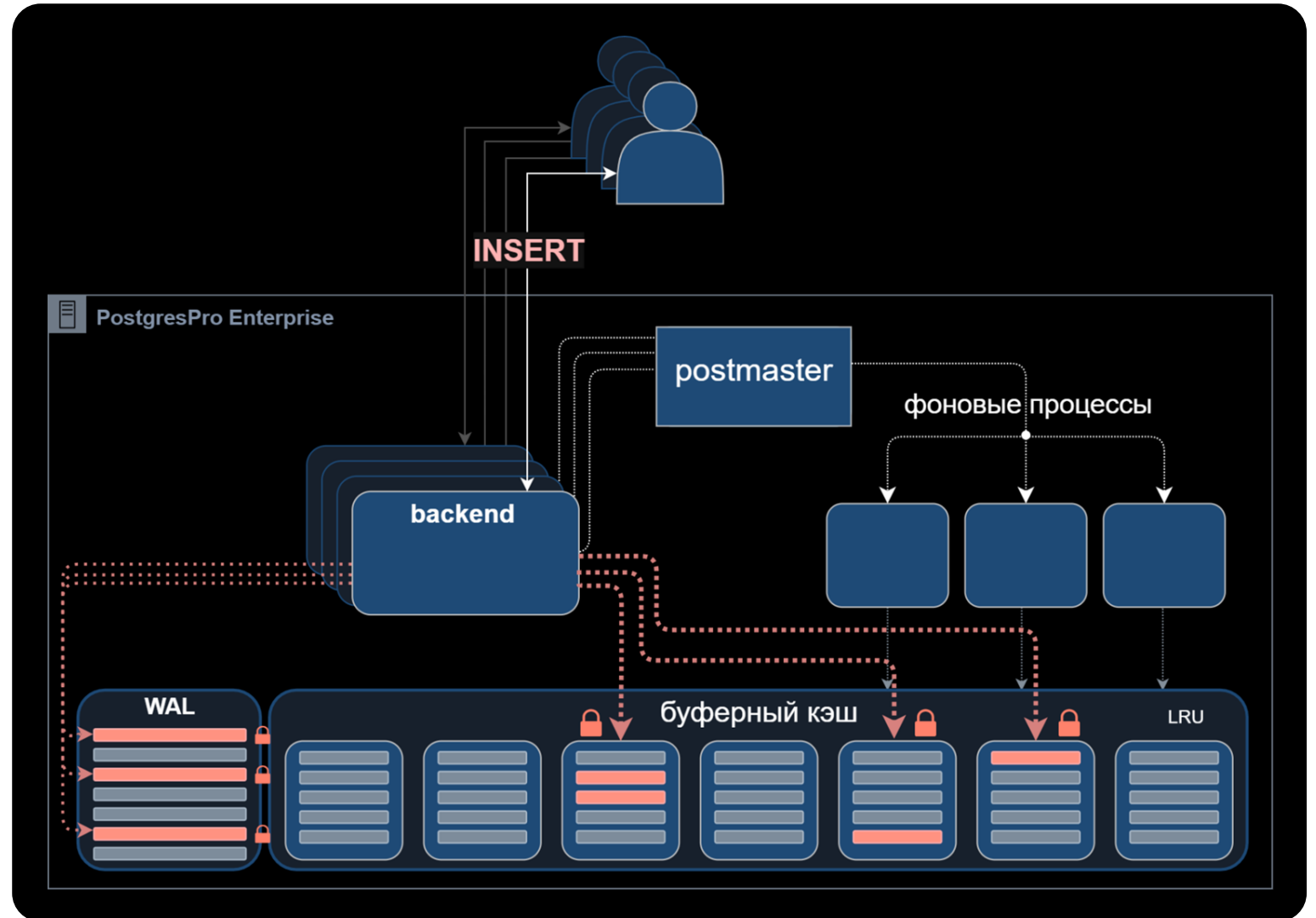
- Использование своего локального буфера бэкенда для формирования пачки строк
- Запись сразу пачки строк в табличную страницу
- Запись в WAL всего одной записи о вставке пачки строк

Обычная процедура вставки

Много обращений
к буферному менеджеру

Много блокировок буферов

На каждую вставку
формируется запись в WAL,
которая сопровождается
блокировкой места в журнале



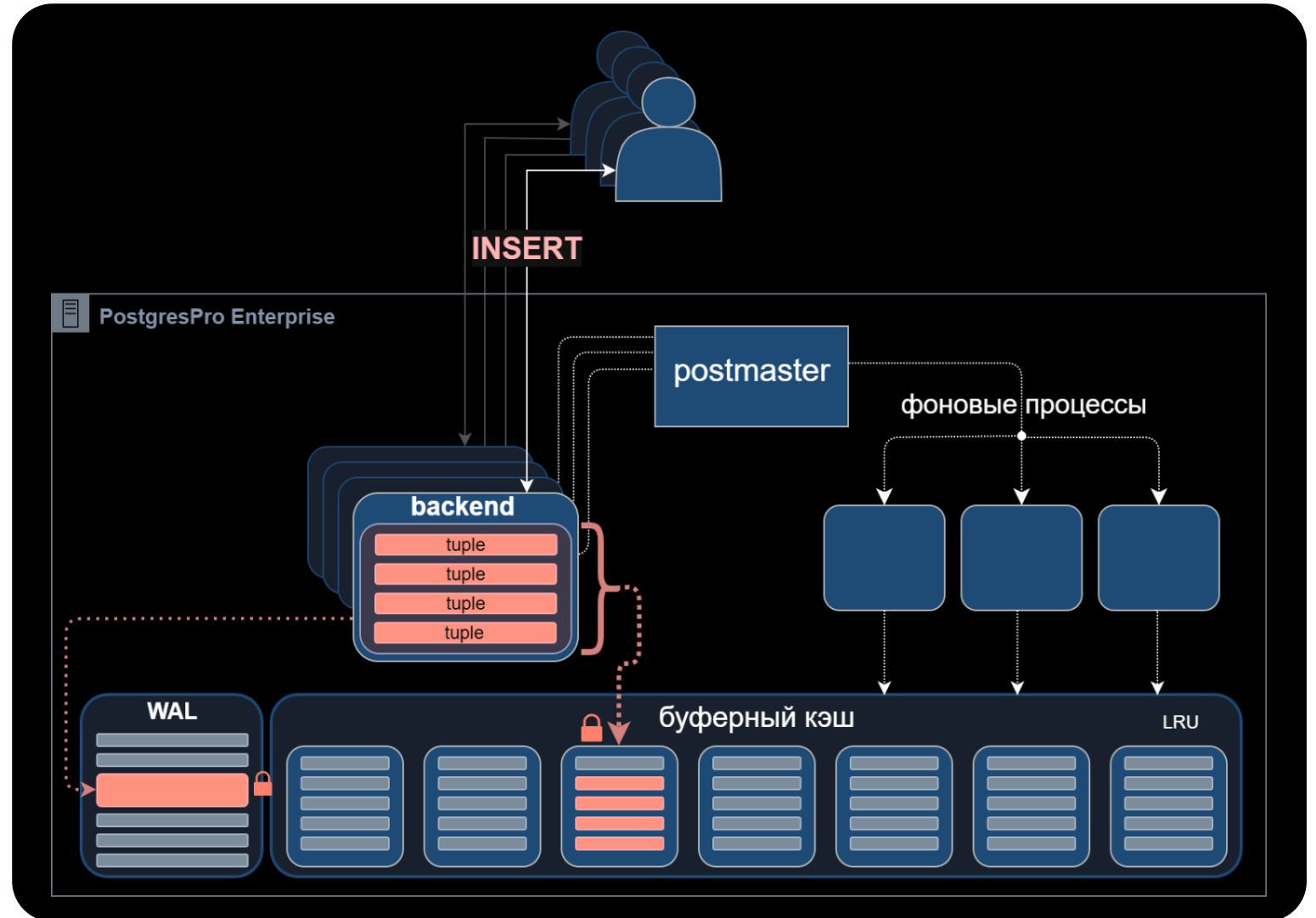
AOT – append-optimized tables

Наполнение локального
буфера (8КБ)

Меньше блокировок буферов

Меньше объем записи
в WAL

Меньше блокировок
WALInsertLock



Мульти сегментная вставка

Это механизм Postgres Pro, предназначенный для ускорения массовой загрузки (доливки) данных в таблицу

Multisegment insert (append table)

Цель: повысить производительность операций массовой заливки данных в таблицу

За счёт чего:

- Ликвидировать узкое место в виде блокировок буферов в буферном менеджере
- Уменьшить объём записей в WAL
- Сократить число блокировок "extend file segment"

Каким образом:

- Каждый backend пишет в свой выделенный сегмент таблицы(файл) в обход буферного менеджера
- Каждый backend организует структуру блоков и сам собирает сегмент
- В валы пишутся только целиком заполненные блоки (full page write) + информация о создании сегмента

Мультисегментная вставка

```
ALTER SYSTEM SET  
multi_segment_relname = tablename ;
```

Любой процесс, пишущий в таблицу получает выделенный сегмент (INSERT/COPY/MERGE)

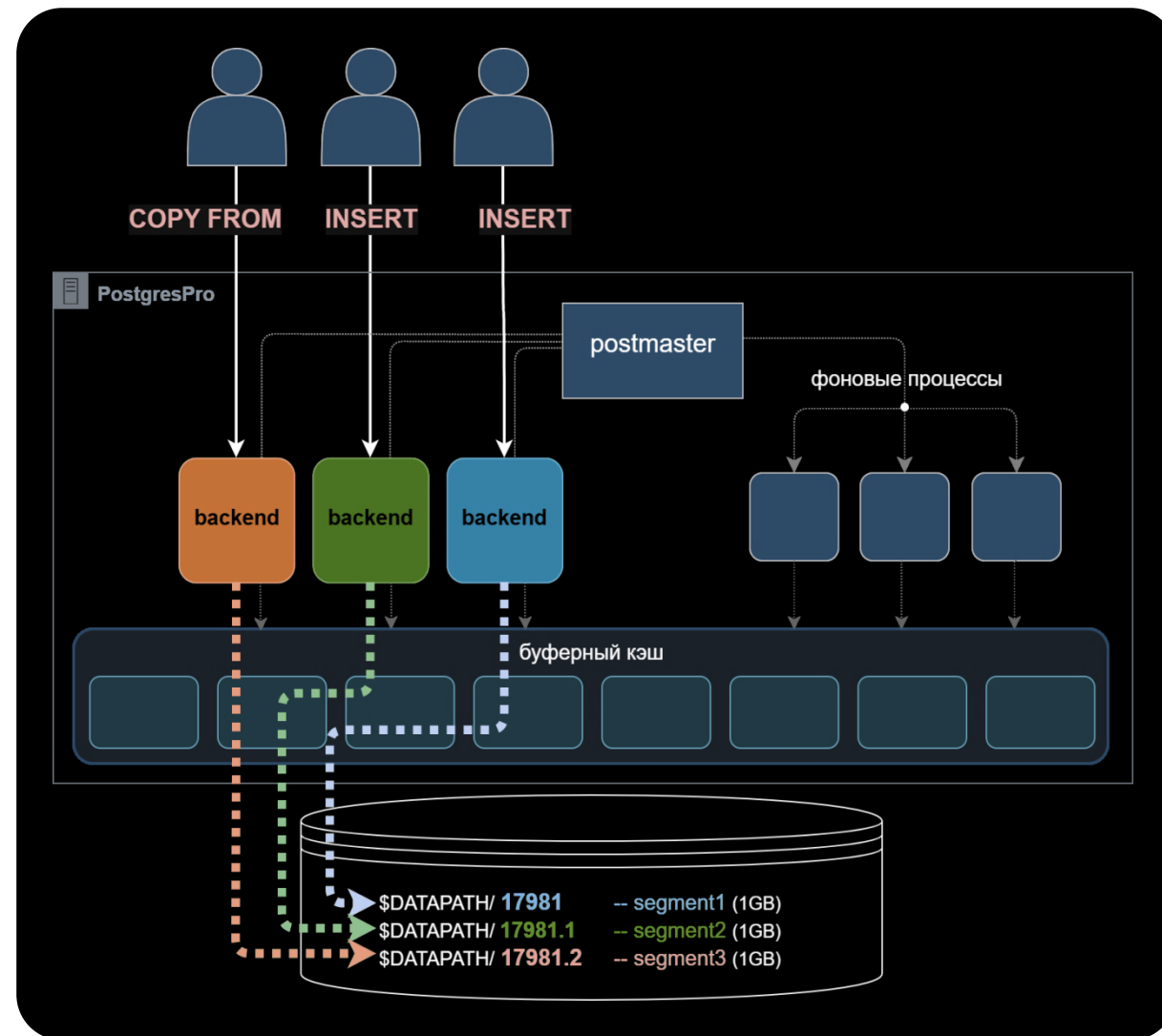
Сегмент выделяется сразу 1ГБ, чтобы минимизировать блокировки

Изменения передаются на реплики и в бэкап, т.к. в WAL информация пишется

Все ограничения целостности работают исправно

Блоки собираются в локальной памяти обслуживающего процесса (bulk insert по 32 страницы)

Данные нужно готовить – нужно придумать как их логически разделить на N примерно равных порций



Инструменты оптимизации

Модуль, позволяющий управлять планом выполнения.

Расширение поддерживает два варианта работы:

1. Фразы-указания в комментариях особого вида, записанных в теле целевого оператора SQL

```
postgres=# /*+
postgres*#      SeqScan(t1)
postgres*#      IndexScan(t2 t2_pkey)
postgres*#      */
postgres=# SELECT * FROM table1 t1 JOIN table2 t2 ON (t1.key = t2.key);
```

2. Таблица указаний:

query_id	hint	application_name
-7164653396197960701	HashJoin(a b) SeqScan(a)	ALL

AQO (Adaptive query optimization) - адаптивная оптимизация запросов

Для каждого класса запросов модуль аqо сохраняет:

- для машинного обучения качество оценки количества строк
- время планирования
- время и статистику выполнения

На основе этих данных аqо строит новый план выполнения и использует его для следующего запроса того же класса.

AQO показал значительное увеличение производительности для сложных запросов.



AQE (Adaptive query execution)

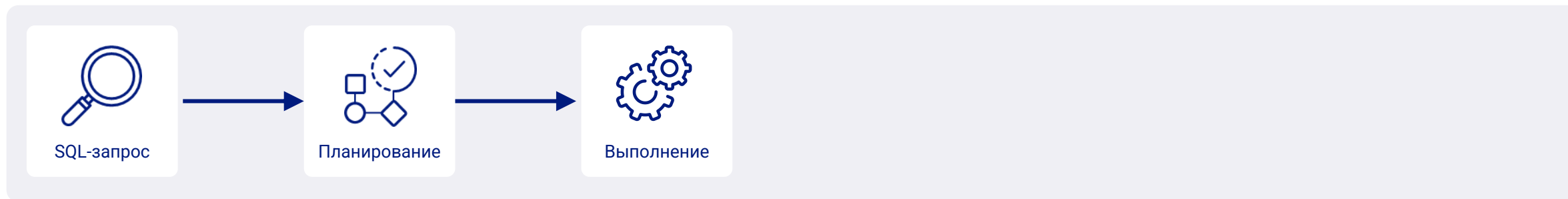
Перепланирование запросов в реальном времени.

Перепланирование запросов в реальном времени (AQE) позволяет перепланировать запросы, если во время их выполнения какой-либо триггер указывает на неоптимальное выполнение, то есть предпочтительно найти более оптимальный план.

Частично выполненный запрос игнорируется, и предпринимается попытка собрать информацию для дальнейшего перепланирования.

Сравнение процедур обработки запроса в PostgreSQL и Postgres PRO

Обычная процедура обработки запроса:

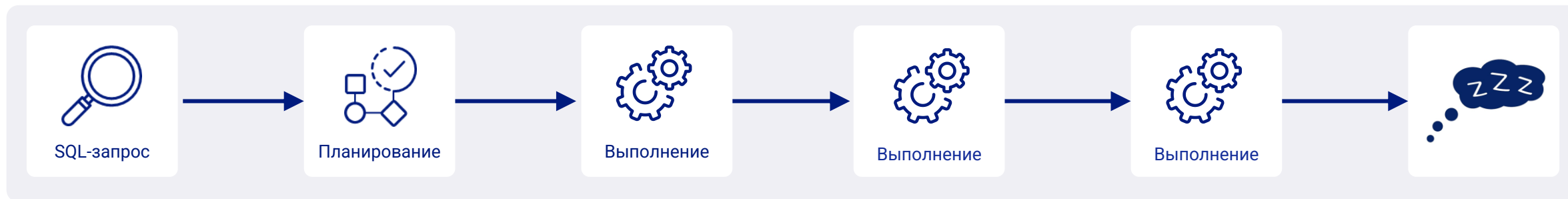


AQE:



Сравнение процедур обработки запроса в PostgreSQL и Postgres PRO

Обычная процедура обработки запроса:

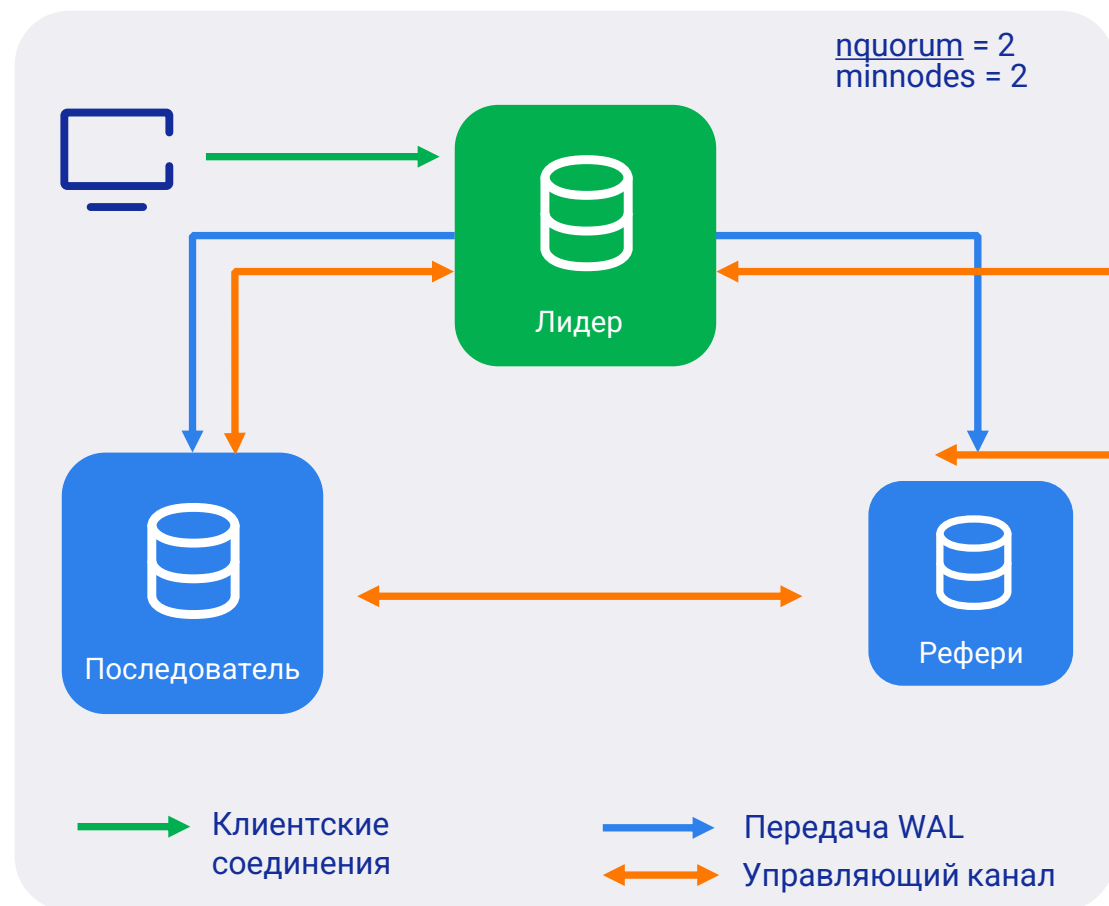


AQE: 



BiHA – Built-in High Availability

BiHA – Built-in High Availability



Кластер с физической репликацией и встроенным аварийным переключением узлов, отказоустойчивостью и автоматическим восстановлением после отказа узлов

Встроен во все редакции

Отдельный управляющий канал

Простая установка и настройка

Не требует дополнительного ПО

Кластерный кворум

Отсутствие split-brain

Приоритеты выбора нового лидера

Функции-обработчики смены состояний

Узел рефери и рефери с WAL

Георезервирование

Каскадная репликация

Сервисный режим

Postgres Pro Enterprise для Хранилищ Данных

Хранилище данных - не только СУБД.
Нам нужны удобные инструменты для того, чтобы работать
с моделью данных, загружать данные из систем-источников
и строить отчеты/дашборды.

Инструменты для автоматизации ELT, моделирования, формирования витрин, контроля качества данных

Ограничения аналитики внутри учётных систем на примере 1С

Производительность запросов

Язык запросов 1С генерирует сложный SQL.

Запрос к виртуальной таблице регистра может превратиться в **JOIN 10+ физических таблиц**



Агрегация больших объёмов данных становится неприемлемо дорогой

Модель хранения данных

Данные разбиты на множество взаимосвязанных таблиц **_AccumRg**, **_InfoRg**

Для аналитики нужна:

- широкая схема
- денормализация
- колоночное хранение



Преобразование «на лету» не справляется

Необходимость консолидации источников

Отчёт по УТ и ЗУП означает:

- медленные межбазовые запросы
- или ручное копирование данных



Теряется целостность и актуальность данных

Конкуренция за ресурсы

Аналитические запросы мешают вводу данных и наоборот

Дублирование запросов

Данные для отчёта каждый раз извлекаются заново, и могут отличаться для разных отчётов

Сравнение вариантов интеграции данных 1С: Предприятие

	Стандартные механизмы 1С COM, OData, HTTP-сервисы	Прямой доступ к БД 1С SELECT из таблиц СУБД или репликация CDC
Производительность	 x1 Тысячи записей в секунду	 x100 Сотни тысяч записей в секунду
Нагрузка на сервер приложений	 Высокая	 Отсутствует
Сложность	 Можно использовать все функции BCL (экспертиза программиста 1С)	 Нужно использовать SQL (экспертиза программиста 1С + разработчика баз данных + DBA)
Безопасность	 Работа через штатный API	 Работа напрямую с БД на чтение
Развитие	 Новая выгрузка для каждого изменения в отчете	 Высокая сложность и хрупкость
Когда использовать	 Для единичных документов	 Для промышленного ETL

Специфика хранения и загрузки данных 1С

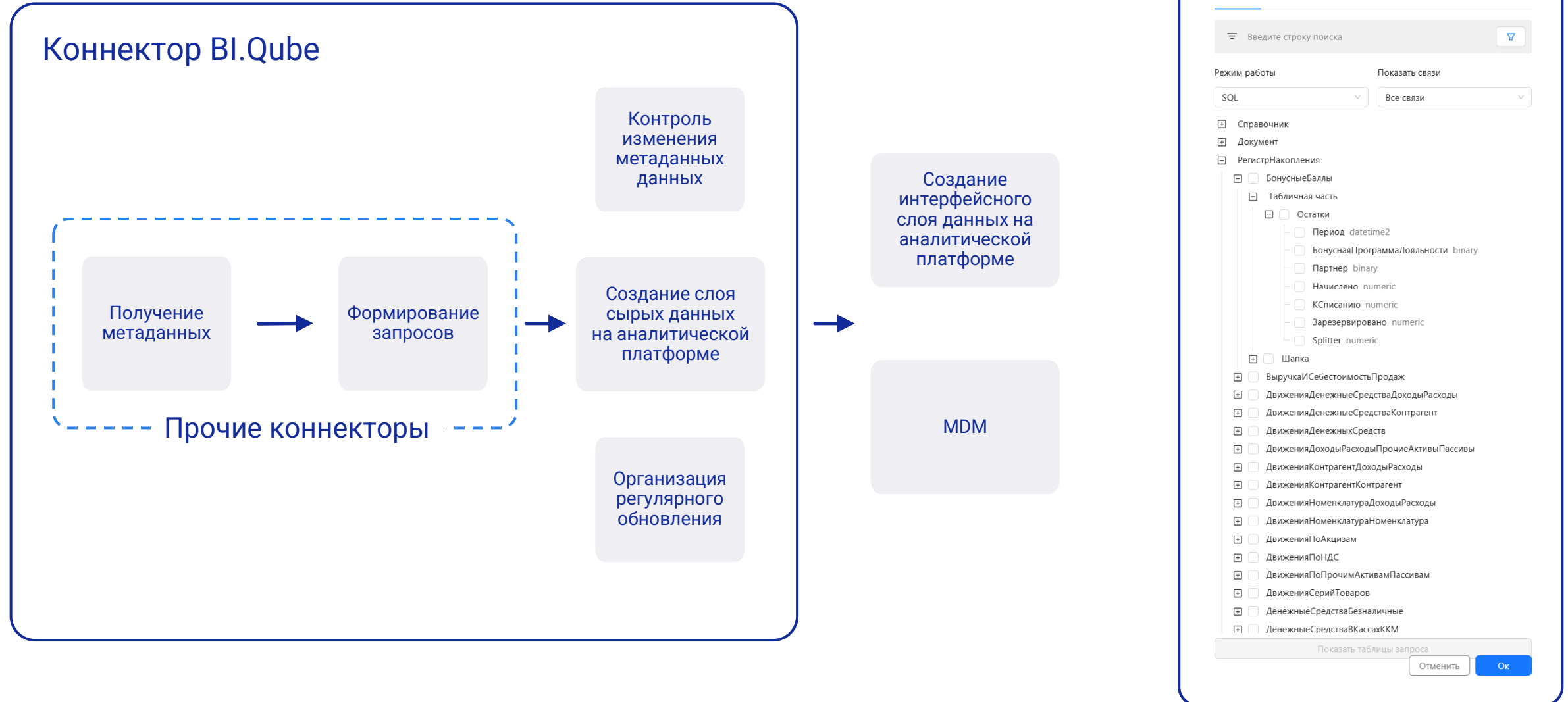


Полная перезагрузка
для компактных объектов

Загрузка изменений
для объектов с версионированием

Скользящее окно
для фактических данных связанных с календарём

Коннектор к 1С



Шины данных и веб сервисы, ФГИСы

Веб-сервис, шина данных, ФГИС
(JSON, XML, TSV, CSV)

https://www.cbr.ru/scripts/XML_dynamic.asp?date_req1=01/01/2025&date_req2=16/04/2026&VAL_NM_RQ=R01235

BI.Qube коннектор
(REST, Kafka, ФГИС)

BI.Qube парсер
(JSON, XML, TSV, CSV)

SELECT [Nominal], [Value], [VunitRate], [Date], [Id], [ValCurs_Id] FROM xml 'https://www.cbr.ru/scripts/XML_dynamic.asp?date_req1=01/01/2025&date_req2=16/04/2026&VAL_NM_RQ=R01235' Проверить запрос Выполнить запрос

Nominal	Value	VunitRate	Date	Id	ValCurs_Id
1	102,2911	102,2911	10.01.2025	R01235	0
1	101,9146	101,9146	11.01.2025	R01235	0
1	102,7081	102,7081	14.01.2025	R01235	0
1	103,4380	103,438	15.01.2025	R01235	0
1	102,8078	102,8078	16.01.2025	R01235	0
1	102,3762	102,3762	17.01.2025	R01235	0

Данные Источник Параметры

Введите адрес файла

https://www.cbr.ru/scripts/XML_dynamic.asp?date_req1=01/01/2025&date_req2=16/04/2026&VAL_NM_RQ=R01235

Метод

GET

Заголовки запроса Тело запроса Исходный JSON Преобразовать

```
<?xml version="1.0" encoding="windows-1251"?><ValCurs ID="R01235"
DateRange1="01.01.2025" DateRange2="16.04.2026" name="Foreign Currency
Market Dynamic"><Record Date="10.01.2025" Id="R01235">
<Nominal>1</Nominal><Value>102,2911</Value>
<VunitRate>102,2911</VunitRate></Record><Record Date="11.01.2025"
Id="R01235"><Nominal>1</Nominal><Value>101,9146</Value>
<VunitRate>101,9146</VunitRate></Record><Record Date="14.01.2025"
Id="R01235"><Nominal>1</Nominal><Value>102,7081</Value>
```

Данные Источник Параметры

Tables

- ValCurs
- ☒ Record
 - ☒ Nominal
 - ☒ Value
 - ☒ VunitRate
 - ☒ Date
 - ☒ Id
 - ☒ ValCurs_Id

Система управления НСИ (MDM)

Управление справочниками

- Ведение пользовательских справочников
- Редактирование данных, загруженных из учетных систем
- Нативная интеграция загруженных и пользовательских данных

Интеграция и обогащение

- Догрузка из внешних источников с автоматическим и ручным маппингом полей, с возможностью преобразования данных
- Синхронизация с Excel, связанные справочники отображаются в Excel выпадающими списками
- Поддержка API: CLI, SQL, C#

Золотая запись

- Объединение и дедубликация
- Гибкие правила консолидации, в т.ч. по комбинации атрибутов
- Учет приоритета при формировании атрибутов
- Двухнаправленная связь с исходными записями

Управление качеством данных

- Бизнес-правила с контролем ввода и подсветкой по условиям
- Поддержка ссылочной целостности
- Полная история изменений
- Массовое редактирование

Удобство для пользователя

- Web-интерфейс для работы с данными в справочниках, аналог Microsoft MDS
- Многопользовательская работа с историей и контролем изменений

Функциональность фреймворка BI.Qube

PostgresPro x BI.Qube

BI.Qube 2.0



MetaStaging

Для интеграции данных из 1С, СУБД, ФГИСов, маркетплейсов



MetaVault

Для построения единой масштабируемой модели



MetaMasterData

Для обогащения справочников, управления мастер-данными, формирования «золотых записей»



MetaFact

Для эффективного формирования фактов



MetaCube

Для переноса модели данных в витрины



MetaControl

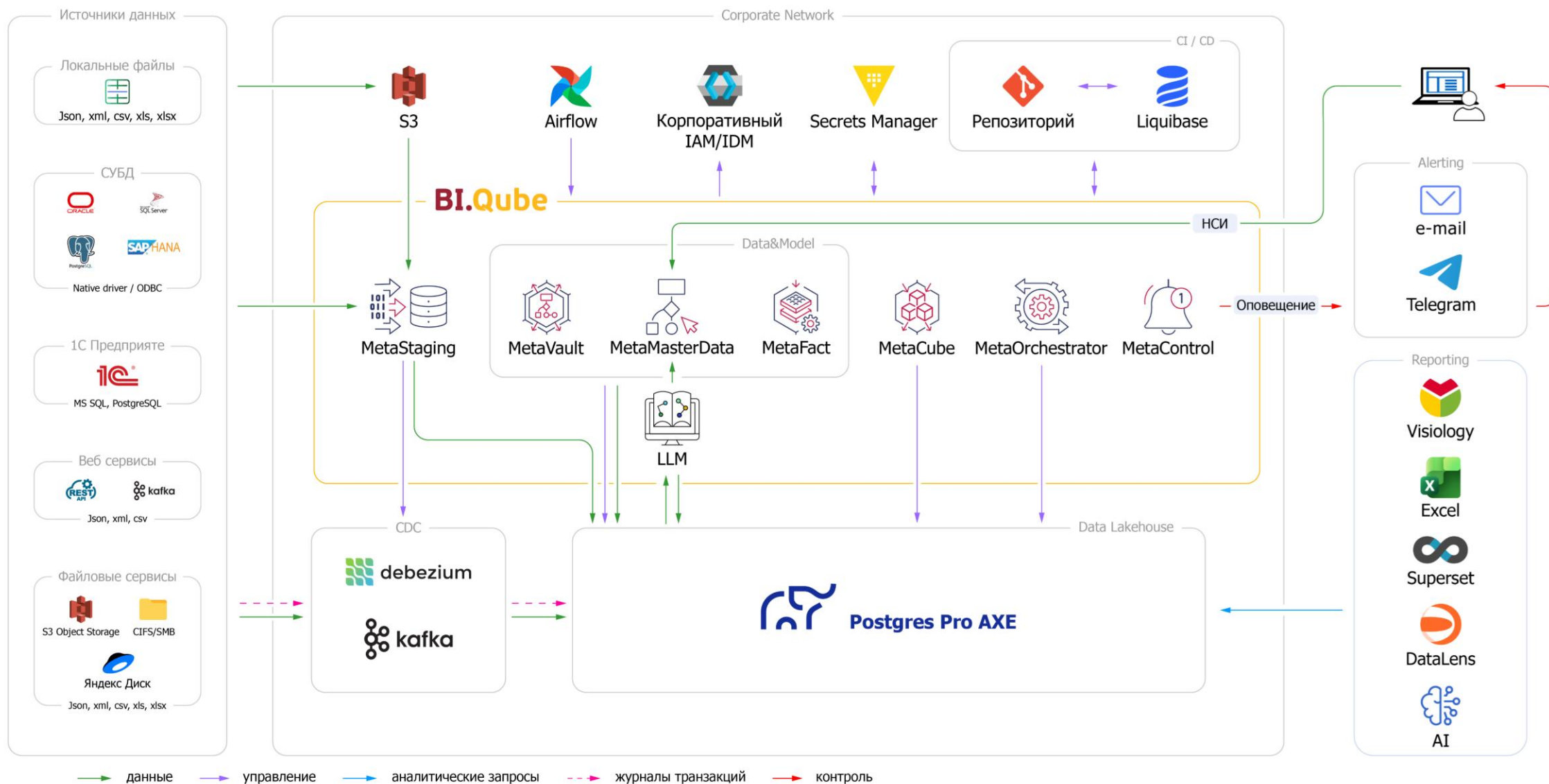
Для обеспечения качества данных и уведомления о бизнес-событиях



MetaOrchestrator

Для управления регулярными процессами загрузки и обработки данных ELT

BI.Qube в ИТ-ландшафте



Эффект автоматизации загрузки данных

Разработка

Кодирование
вручную

15 минут

полной

60 минут

инкрементальной

120 минут

скользящим
окном

Выполнение



производительность



потребление памяти

Эффект автоматизации загрузки данных

Разработка

Кодирование
вручную

Конфигурирование
в MetaStaging

15 минут

полной

1 минута

60 минут

инкрементальной

1 минута

120 минут

скользящим
окном

5 минут

Выполнение



производительность



в 2 раза быстрее



потребление памяти



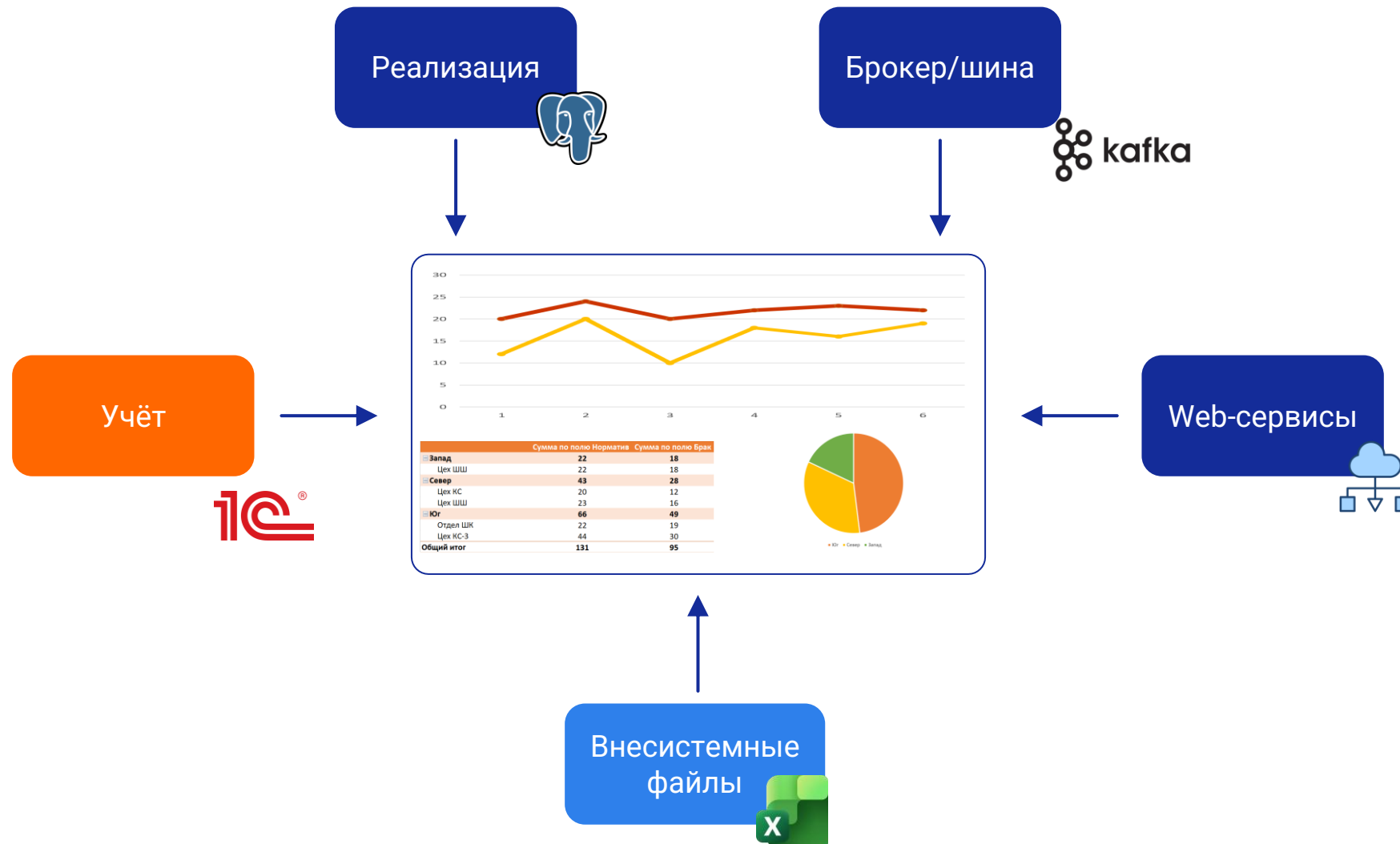
в 10 раз меньше

Демонстрация построения DWH на Postgres Pro Enterprise:

от загрузки данных из 1С, Excel, REST API и Kafka
до слоя витрин с контролем качества данных

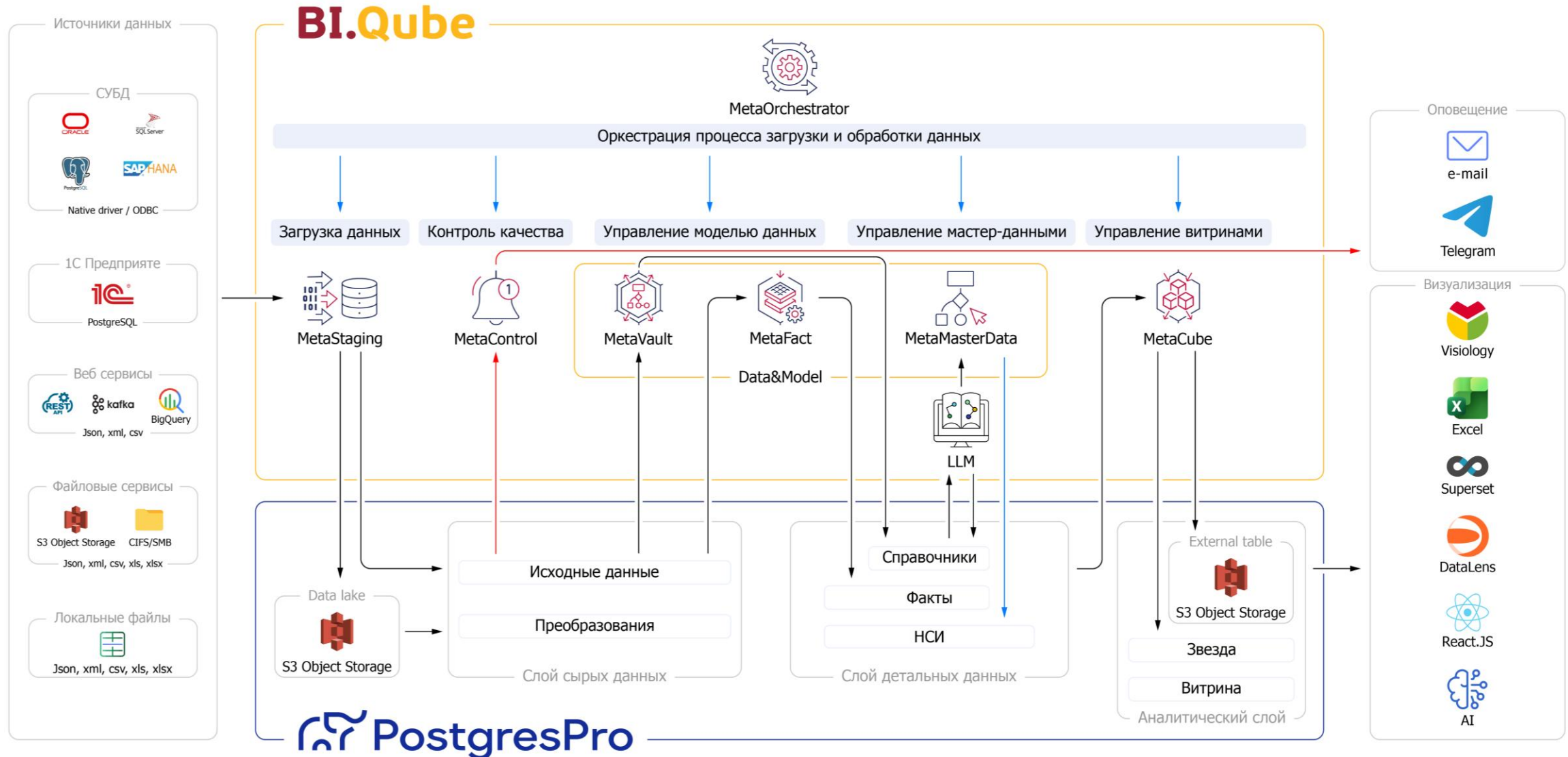
Практическая задача

На примере кейса «аналитика торгового предприятия»

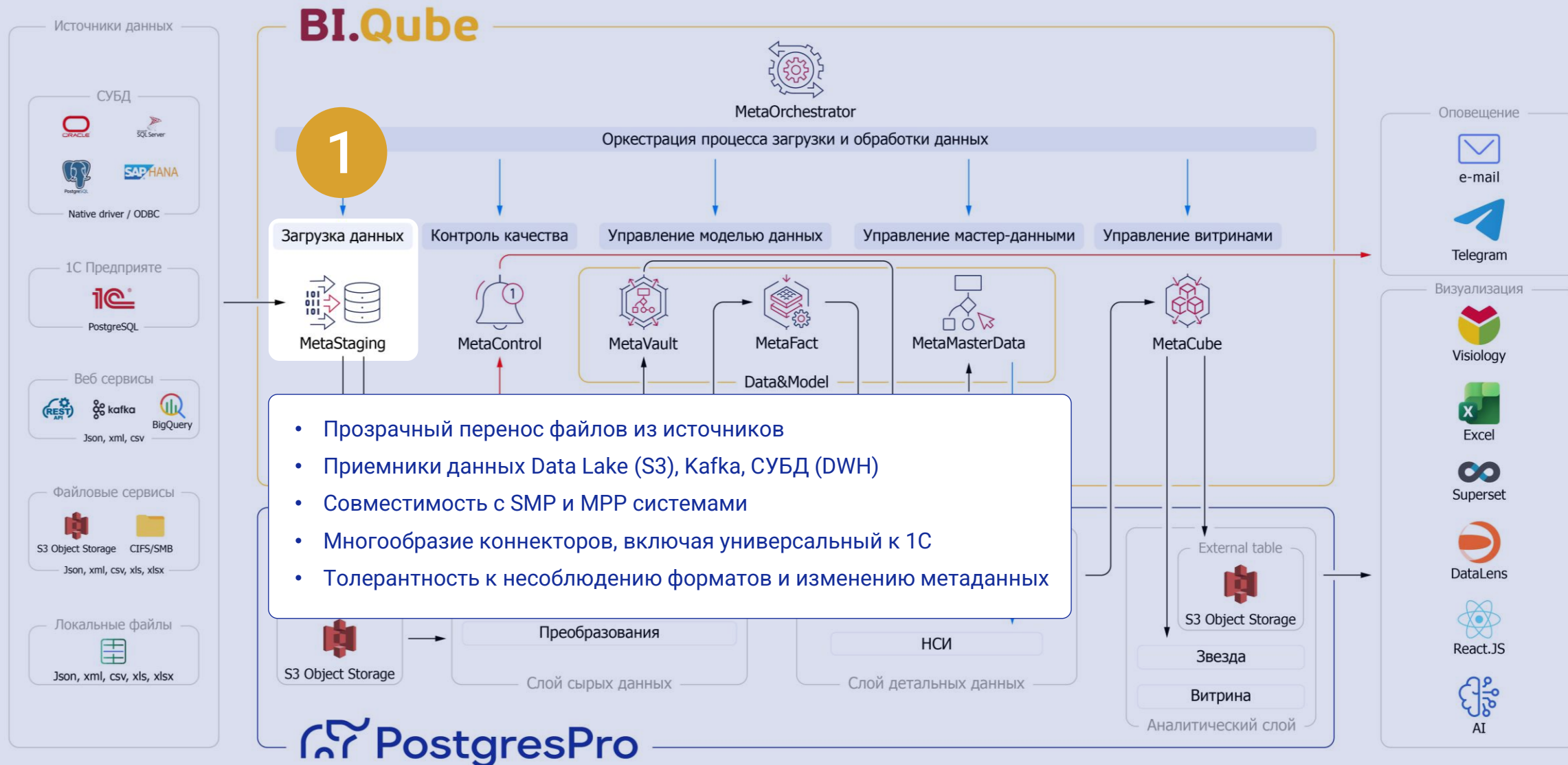


Решение практической задачи на Postgres Pro и BI.Qube

PostgresPro x BI.Qube



Загрузка данных



- Общее
- Staging
- Data & Model
- Control
- Cube
- Конструктор веб-форм
- Data Lineage
- Orchestrator

Домашняя страница

Доска информации метаконпонентов

Frontend

Версия: release005:19

Дата сборки: 08.05.2026 08:03:44

Номер спринта: 90

MetaMasterData & MetaVault

Версия: release005:37

Дата сборки: 07.05.2026 15:20:33

Номер спринта: 90

MetaStaging

Версия: release005:37

Дата сборки: 07.05.2026 15:20:33

Номер спринта: 90

Внешние сервисы

Keycloak

Система аутентификации и авторизации

HashicorpVault

Сервис хранения учетных данных подключений

Дополнительная информация

Всего пользователей : 69

Всего команд стейджинг : 0

Всего моделей : 1

- факты : 4
- справочники : 10

Всего контролей : 15

Всего витрин данных : 2

- Элементов витрин данных : 14

Время последнего запуска : 08.05.2026, 08:01:42

Всего подключений : 7

- недоступные : 1
- доступные : 6
- нет данных : 0



00:00

00:56



- Общее
- Staging
- Data & Model
- Control
- Cube
- Конструктор веб-форм
- Data Lineage
- Orchestrator

Домашняя страница

Доска информации метаконпонентов

Frontend

Версия: release005:19

Дата сборки: 08.05.2026 08:03:44

Номер спринта: 90

MetaMasterData & MetaVault

Версия: release005:37

Дата сборки: 07.05.2026 15:20:33

Номер спринта: 90

MetaStaging

Версия: release005:37

Дата сборки: 07.05.2026 15:20:33

Номер спринта: 90

Внешние сервисы

Keycloak

Система аутентификации и авторизации

HashicorpVault

Сервис хранения учетных данных подключений

Дополнительная информация

Всего пользователей : 70

Всего команд стейджинг : 0

Всего моделей : 1

- факты : 4
- справочники : 10

Всего контролей : 15

Всего витрин данных : 2

- Элементов витрин данных : 14

Время последнего запуска : 08.05.2026, 08:01:42

Всего подключений : 7

- недоступные : 1
- доступные : 6
- нет данных : 0



00:00

01:04



Модели

Создать

Проверка целостности

Введите строку поиска

Название	Подключение	Префикс	Базовая схема	Описание
Модель для вебинара	PostgresPro	pgpro		
Без моделей				



00:00

00:54



4-75 Быстрый старт DWH и аналитики на ваших данных - Загрузка из REST

BI.Qube

Общее

Staging

Команды

Трансформации

Профили

Сессии

Данные

Data & Model

Control

Cube

Конструктор веб-форм

Data Lineage

Orchestrator

Модели

Создать

Проверка целостности

Введите строку поиска

Название	Подключение	Префикс	Базовая схема	Описание
Модель для вебинара	PostgresPro	pgpro		
Без моделей				



00:00

02:22



Настройки



Контроль качества данных



Модели

Создать

Проверка целостности

Введите строку поиска

Название	Подключение	Префикс	Базовая схема	Описание
Модель для вебинара	PostgresPro	pgpro		
Без моделей				



00:00

00:49



Формирование модели данных



- Общее
- Staging
- Data & Model
- Модели**
- Профили
- ER-Модель
- Сессии
- Control
- Cube
- Конструктор веб-форм
- Data Lineage
- Orchestrator

Модели

[Создать](#) [Проверка целостности](#)

Введите строку поиска

Название	Подключение	Префикс	Базовая схема	Описание
Модель для вебинара	PostgresPro	pgpro		



Описание

Префикс

Базовая схема

pgpro_dds

Внутренние объекты Data Vault: pgpro_dds

Промежуточные таблицы: pgpro_dds_stg

Интерфейсные объекты Data Vault: pgpro_dds_v



Сущности [Назад](#) [Справочник](#) [Факт](#) [Золотая запись](#)

[Создать](#) [Удалить](#) [Собрать](#) [Очистить](#)

Введите строку поиска

Сохраняемый фильтр

Начальная ... → Конечная д...

Выберите теги

Название	Описание	Профили	Теги
СтруктураПредприятия			
Организации_33			
СправочникИзPG			
СправочникИз1C			
Организации			
КодыВалют			
СправочникСтруктураПредприятия			
СправочникОрганизации			
СправочникОрганизацииИзPG			
СправочникОрганизацийИз1C			
СправочникВалют			
СправочникНоменклатура			
СправочникНоменклатураИзKafka			
СправочникНоменклатураИз1C			



Профили

Описание

Описание

Базовая схема

Базовая схема

Таблица-источник

Целевая таблица-источник

Выбрать

Ключи

Нет ключей



Поля

Нет полей

02:06

Создать поле

Добавить

00:00



1 20 / стр.



Добавить



Управление Мастер-Данными



Сущности [Назад](#) [Справочник](#) [Факт](#) [Золотая запись](#)

Создать

Удалить

Собрать

Очистить

Введите строку поиска

☐ Больше информации

Название

Описание

tags

СправочникОрганизации

СправочникНоменклатура

Выберите теги

Профиль

Описание

Описание

Базовая схема

Базовая схема

* Справочники

Справочники

Ключи соединения

Нет ключей соединения

Атрибуты

Нет атрибутов



Домашняя страница

Доска информации метаконпонентов

Frontend

Версия: release005:33

Дата сборки: 03.06.2026 08:22:44

Номер спринта: 92

MetaMasterData & MetaVault

Версия: release005:69

Дата сборки: 03.06.2026 11:21:59

Номер спринта: 92

MetaStaging

Версия: release005:69

Дата сборки: 03.06.2026 11:21:59

Номер спринта: 92

Внешние сервисы

Keycloak

Система аутентификации и авторизации

HashicorpVault

Сервис хранения учетных данных подключений

Дополнительная информация

Всего пользователей : 50

Всего команд стейджинг : 0

Всего моделей : 1

- факты : 5
- справочники : 30

Всего контролей : 15

Всего витрин данных : 2

- Элементов витрин данных : 9

Время последнего запуска : 10.06.2026, 14:32:13

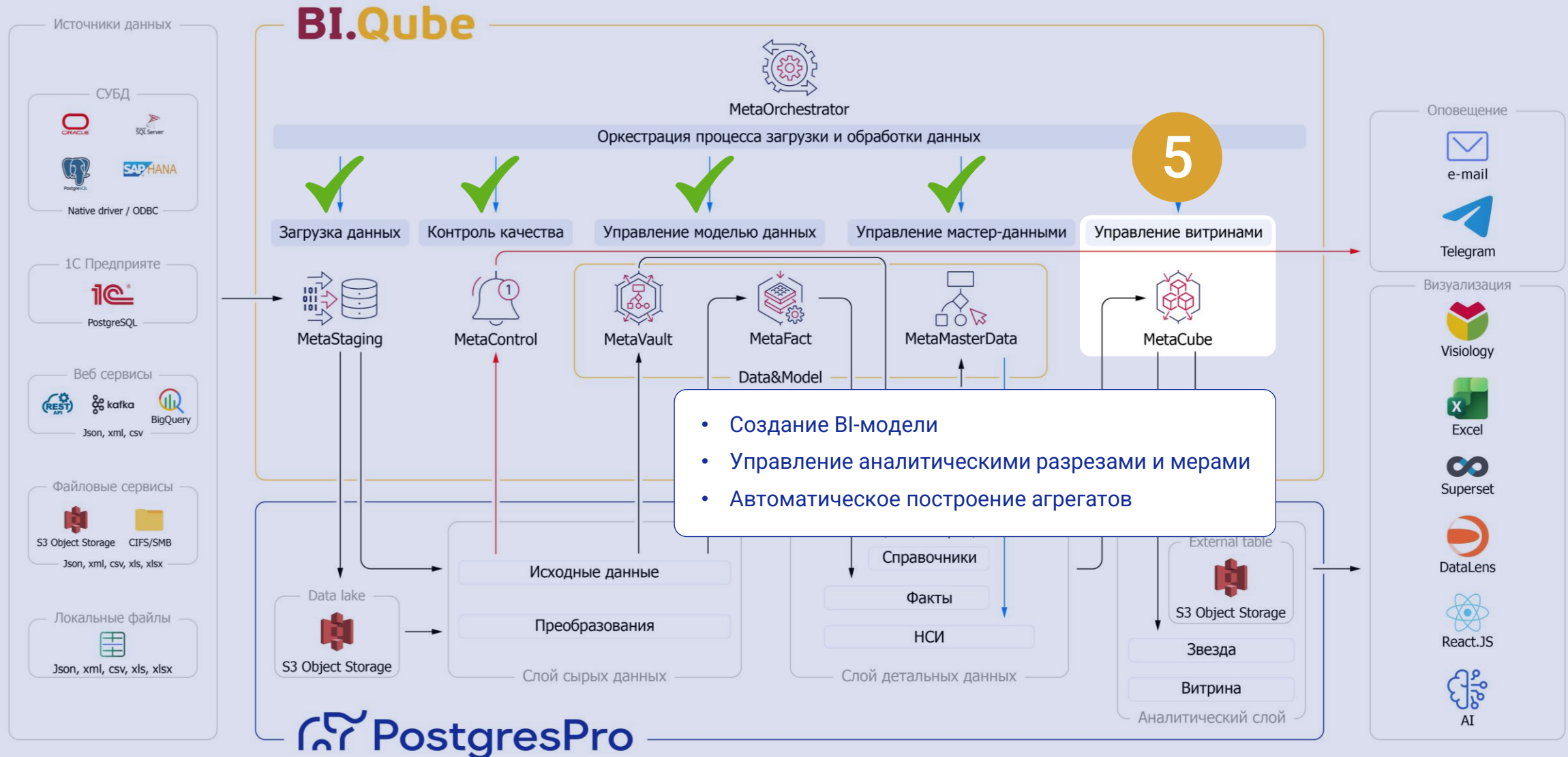
Всего подключений : 10

- недоступные : 1
- доступные : 9
- нет данных : 0



Формирование реляционных витрин в Postgres Pro

PostgresPro x BI.Qube



- Общее
- Staging
- Data & Model
- Control
- Cube
- Витрины**
- Профили
- Сессии
- Конструктор веб-форм
- Data Lineage
- Orchestrator

Витрины

[Создать](#) [Удалить](#)

Введите строку поиска ☐ Больше информации

Имя витрины	Адрес витрины	Домены модели	Описание
Витрина Продажи товаров	PostgresPro	pgpro_webinar	



PostgresPro

* Адрес витрины ?

Адрес витрины

Описание

Описание



Оркестрация и DataLineage

6

PostgresPro x BI.Qube



- Общее
- Staging
- Data & Model
- Control
- Cube
- Конструктор веб-форм
- Data Lineage
- Ресурсы и зависимости
- Orchestrator

Ресурсы и зависимости

Показать зависимости

Модель

+

-

↻

🔒

📶

👁



00:00

01:06

↶ 10

▶

↷ 10

🔊

Postgres Pro + BI.Qube = равноценная замена проприетарных платформ для аналитики

Задача	Microsoft	BI.Qube + экосистема/продукт Партнера
Подключение источников	-	BI.Qube = подключение к 1С, СУБД, ERP/CRM, Excel, Kafka, REST API
DWH	MS SQL	BI.Qube + Postgres Pro
ETL/ELT	Integration Services	BI.Qube = генерация ETL/ELT pipeline, оркестрация, расписание
OLAP	Analysis Services	BI.Qube + Lakehouse Postgres Pro AXE
Моделирование данных	-	BI.Qube = Kimball, Data Vault, Anchor modeling
Дашборды	Power BI	BI.Qube + Superset / Datalens / Visiology
Excel	Excel	BI.Qube + Visiology
Репортинг	SSRS / Power BI Report Builder	BI.Qube = интеграция с используемыми заказчиком инструментами
MDM	Master Data Services	BI.Qube = справочники, внесистемный учет, золотая запись. Полнофункциональный аналог MDS

Ускорение проектов

- Автоматизация развертывания
- Быстрое наполнение DWH
- Короткий time-to-market

Снижение требований к экспертизе

- Генерация ELT по метаданным
- Low-code/no-code для типовых задач DWH
- Сквозной Data Governance, CI/CD

Повышение эффективности

- Ускорение разработки в 2-5 раз
- Сокращение трудозатрат
- Минимизация ошибок в коде
- Уменьшение лицензионных платежей
- Снижение TCO за счет упрощения поддержки и внесения изменений

Быстрый старт

- Развертывание фреймворка BI.Qube в Docker или кластере Kubernetes
- Единая аналитическая платформа в экосистеме Postgres Pro вместо интеграции разрозненных продуктов open source (PostgreSQL + Greenplum + ClickHouse)

Что делать, если данных стало много?

Postgres Pro AXE

Аналитическое решение для работы с большими
гибридными нагрузками



Postgres Pro AXE

 PostgresPro x BI.Qube

Продукт для построения быстрой и экономичной аналитической платформы

Хранение петабайтов данных

Эффективная обработка сотен
ТБ данных в рамках одного запроса

Масштабируемость



Быстрое и эффективное аналитическое решение на базе СУБД Postgres Pro !



Работа в привычной экосистеме Postgres Pro. Один продукт вместо «зоопарка» opensource технологий



Снижение TCO (совокупная стоимость владения) за счет эффективного использования ресурсов и уменьшение издержек на внедрение / сопровождение

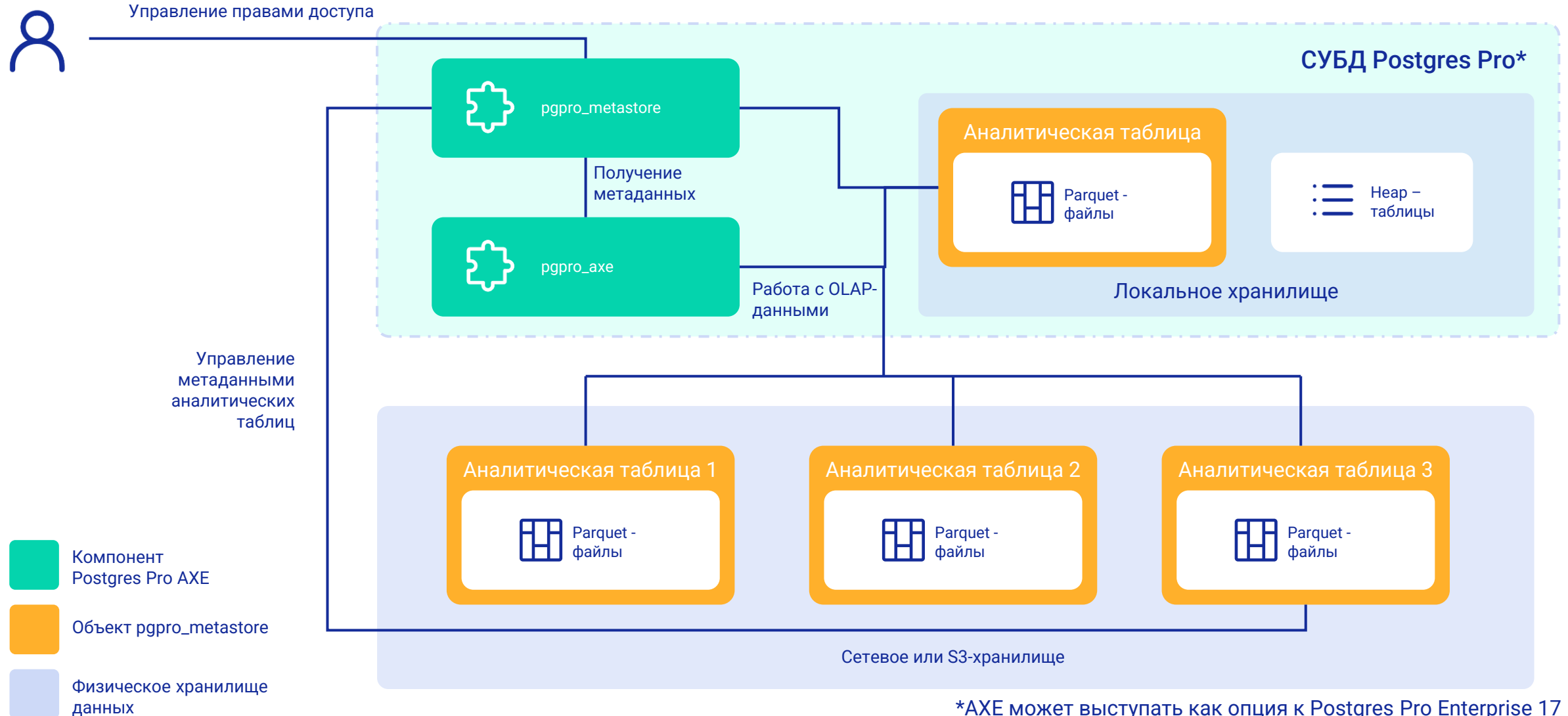


Позволяет строить гибридные системы, объединяющие транзакционную и аналитическую нагрузку

Postgres Pro AXE

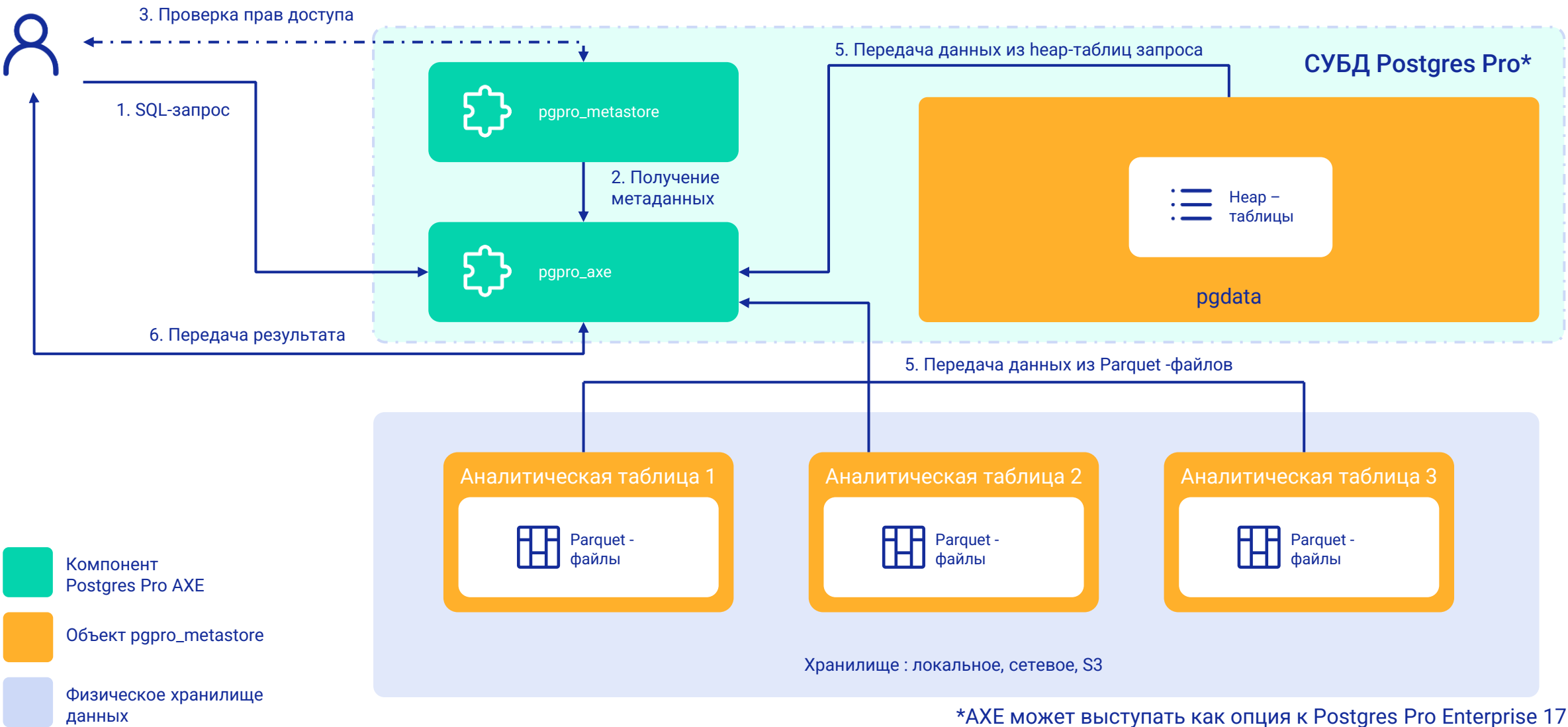
За счет чего достигается такая производительность
и эффективность

Архитектура Postgres Pro AXE



*AXE может выступать как опция к Postgres Pro Enterprise 17

Обработка запроса в Postgres Pro AXE



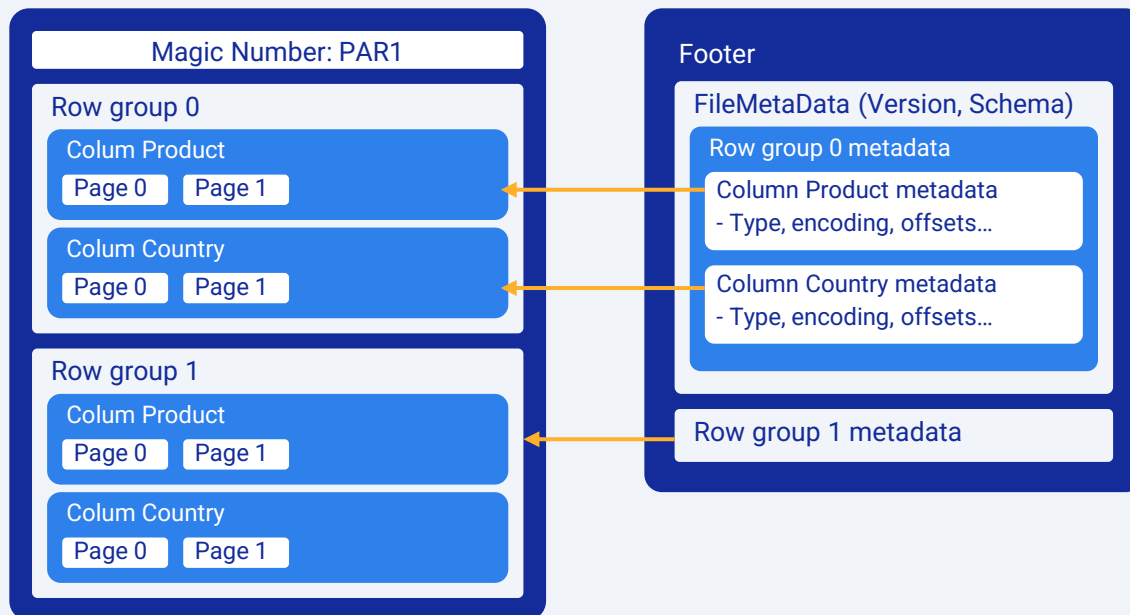
*AXE может выступать как опция к Postgres Pro Enterprise 17

Создание parquet-файла

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 распространённый формат

Parquet File Format



Данные в таблице

A	B	C
A1	B1	C1
A2	B2	C2
A3	B3	C3

Текстовый файл CSV

A1	B1	C1	A2	B2	C2	A3	B3	C3
----	----	----	----	----	----	----	----	----

Файл в формате Parquet

A1	A2	A3	B1	B2	B3	C1	C2	C3
----	----	----	----	----	----	----	----	----

Работа с Parquet файлом как с обычной таблицей

Над колоночными данными, лежащими в parquet файлах можно создать представление (view) и работать с ним в запросах как с обычной таблицей Postgres Pro

- Автоматическое создание View через функцию
- Не нужно переписывать существующие запросы
- Пользователь продолжает работать с данными привычным способом
- Внешние системы не заметят подмены, но начнут получать данные гораздо быстрее
- Возможность в рамках одного запроса работать одновременно и с обычными heap-таблицами и с parquet файлами

```
CREATE OR REPLACE VIEW mt_items
AS
SELECT
    r['item_id']::integer AS item_id,
    r['product_name']::text AS
product_name,
    r['status']::text AS status
FROM read_parquet ('/dir/items/*') r;
```

Векторный движок запросов

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



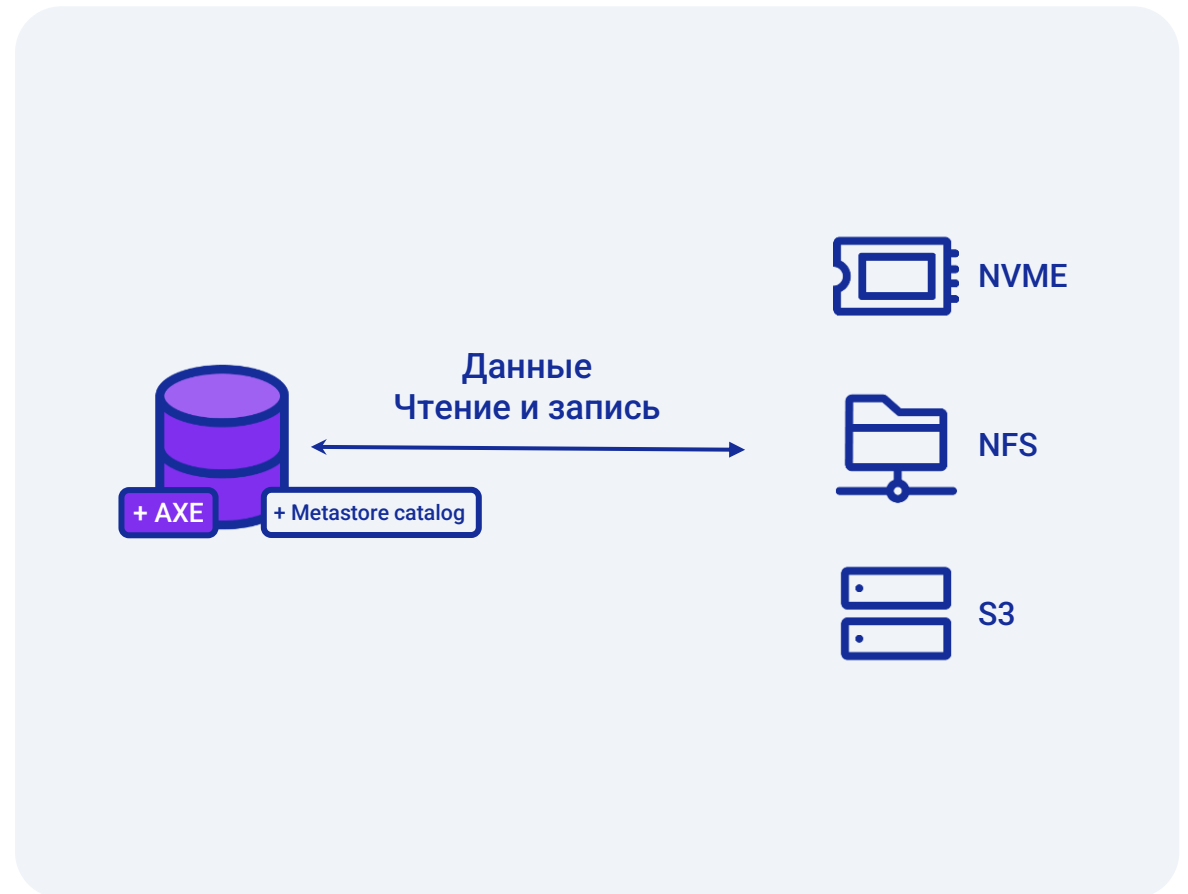
Быстрее большинства аналогов



Гибкость: хранение данных

Postgres Pro AXE позволяет хранить parquet файлы как в локальном хранилище (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



Гибкость: использование данных

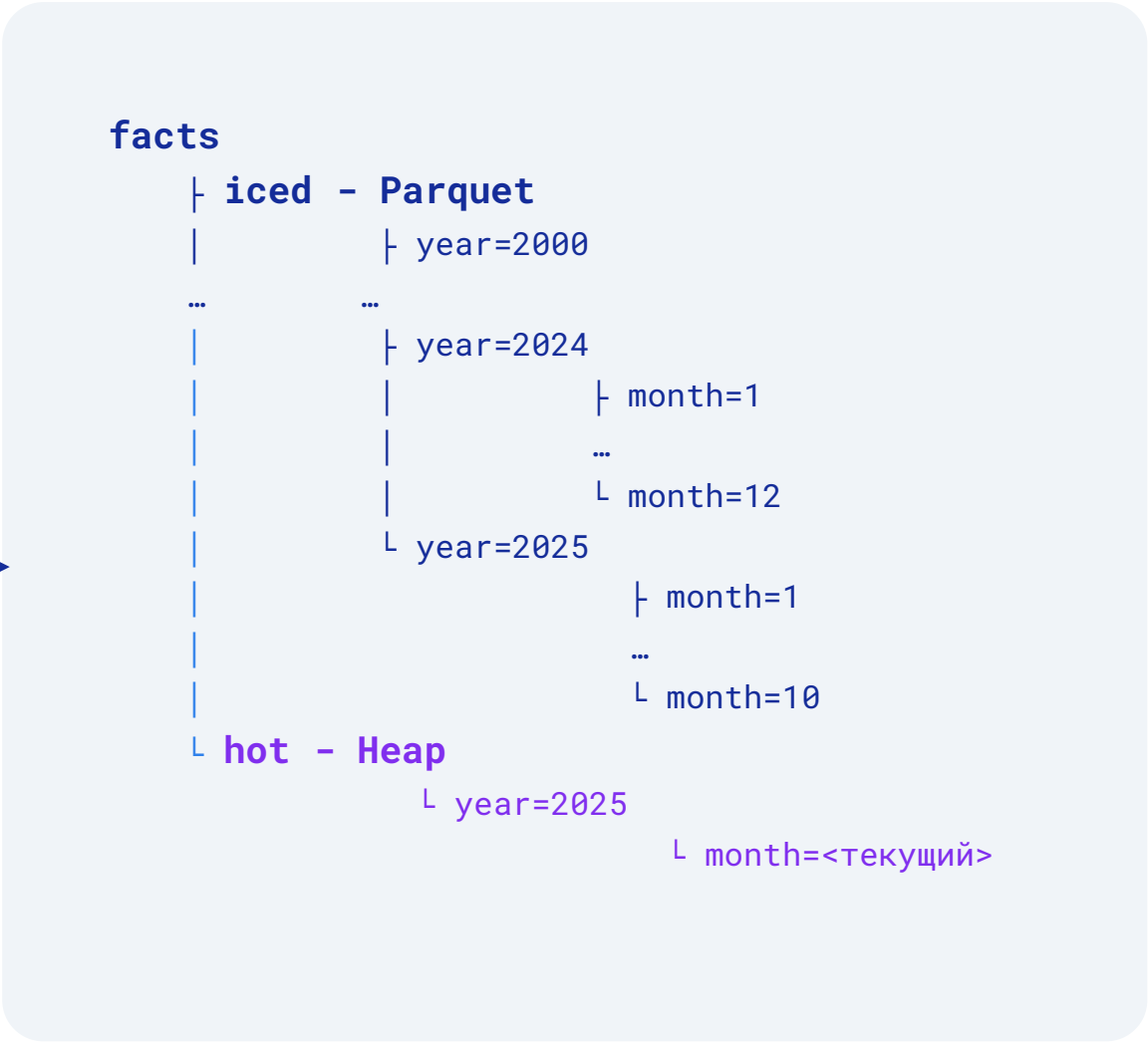
Postgres Pro AXE позволяет реализовать различные сценарии использования данных благодаря возможности работать одновременно и с heap-таблицами, и с parquet файлами

- Справочники и маленькие таблицы можно хранить в Heap
- Данные которые часто меняются обрабатываем в heap структуре
- «Остывшие» данные перекладываем в parquet
- Единое view объединяющее горячие и холодные данные

```
SELECT
  f.route_no,
  f.scheduled_departure,
  t.book_ref,
  s.price
FROM vw_local_parquet_flights f -- Представление Parquet файла на локальном диске сервера
JOIN vw_s3_parquet_segments s ON s.flight_id = f.flight_id -- Представление Parquet файла на S3
JOIN bookings.tickets t ON t.ticket_no = s.ticket_no; -- HEAP таблица
```

Пример гибкого использования данных

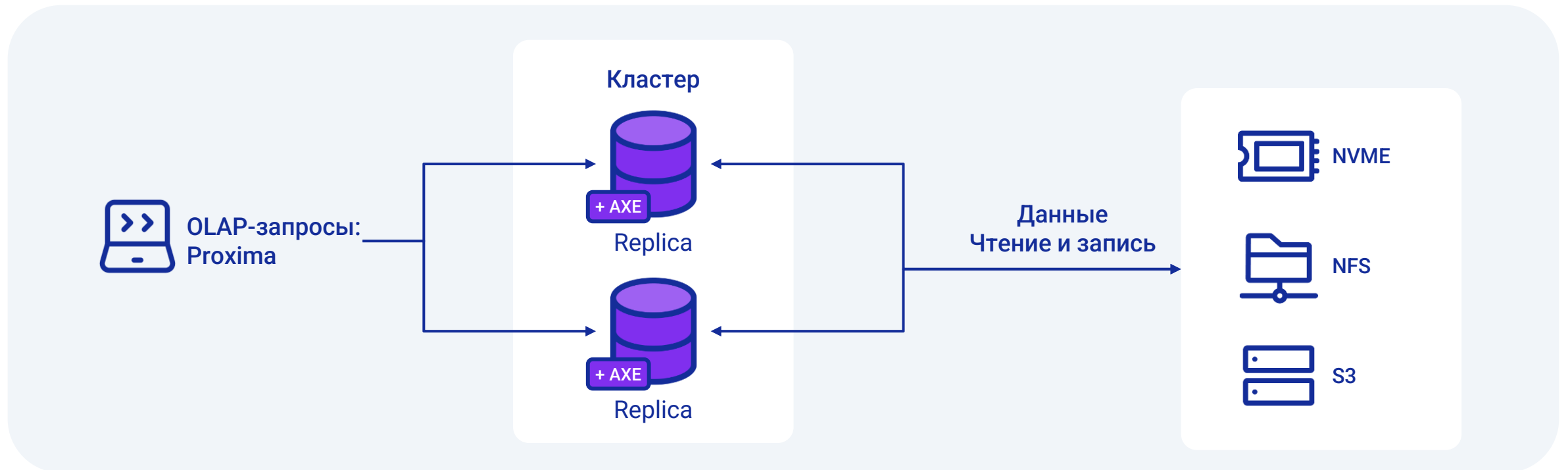
Данные	Год	Месяц
Холодные	2000	1
	...	...
	2024	12
	2025	1
	...	...
	2025	8
	2025	9
	2025	10
Горячие	2025	11



Масштабируемость

Современные аналитические решения должны иметь гибкие возможности для масштабирования как по хранению (storage), так и по вычислению (compute)

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



В десятки раз быстрее heap-таблиц Postgres Pro

 PostgresPro x BI.Qube

Запрос N	PG tables	AXE	Ускорение в
1	7.4186s	0.128s	57
2	4.0139s	0.175s	22
3	1.6309s	0.0373s	43
4	1.9727s	0.0423s	46
5	1.5942s	0.0359s	44
6	1.9371s	0.0346s	55
7	3.4457s	0.0713s	48
8	5.2353s	0.167s	31
9	1.6342s	0.0567s	28
10	2.0166s	0.0426s	47
11	1.6746s	0.0408s	41

ClickBench

64 ядра

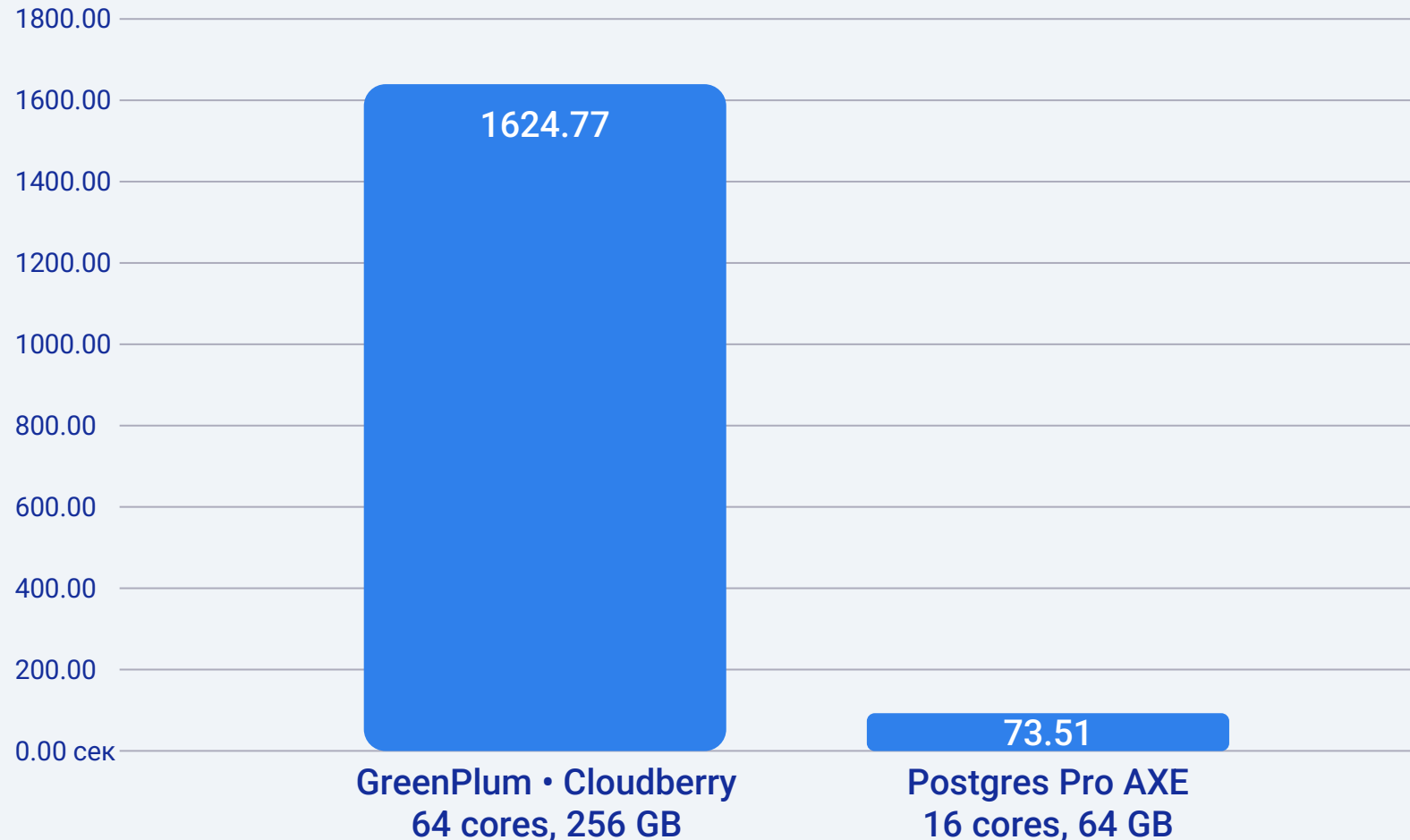
64 Гб RAM

6 млн строк

до **60** раз
ускорение

TPC-H • SF=100 ГБ, Полное время

Меньше лучше



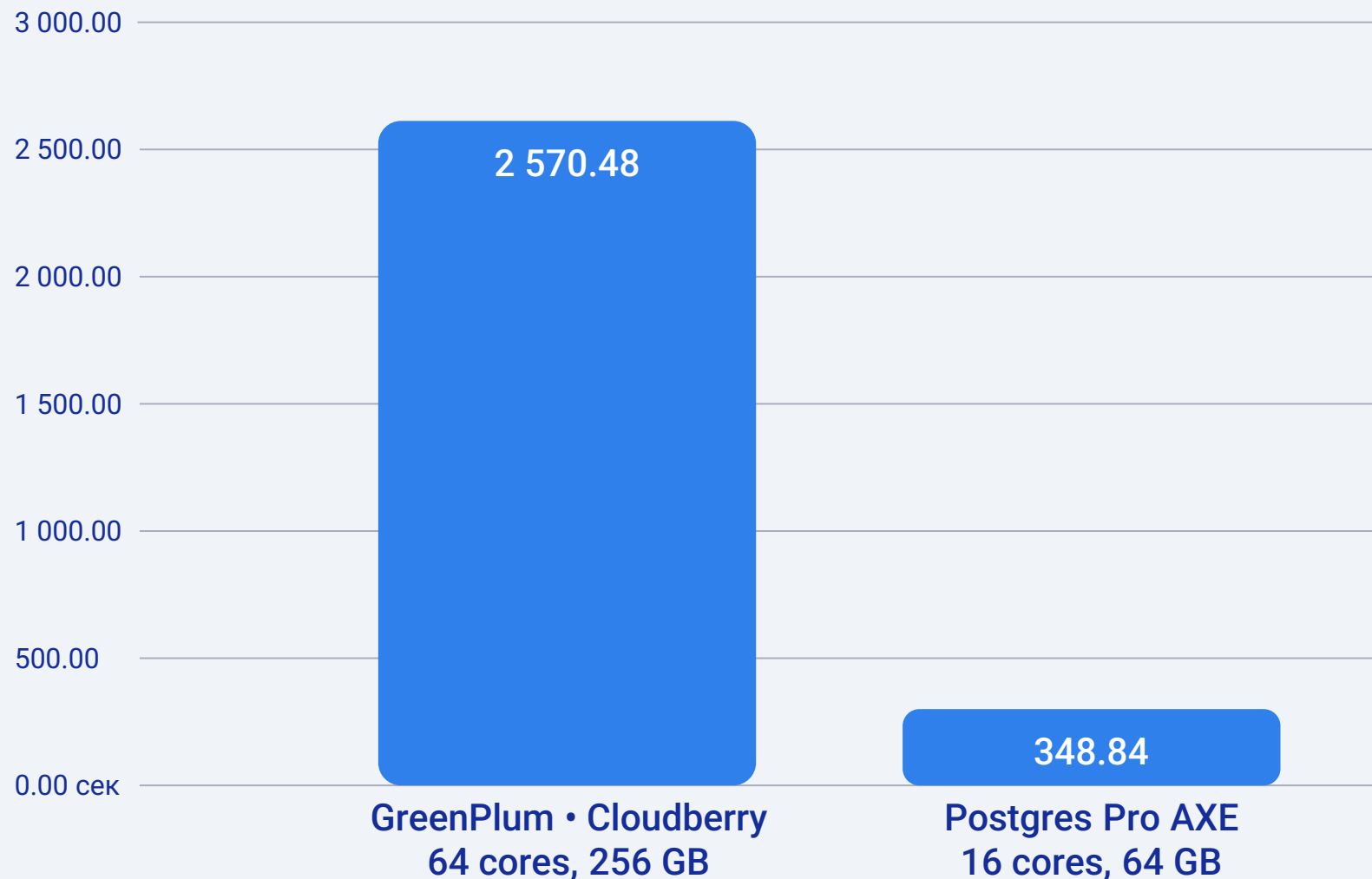
в 22 раз
выше скорость

до 10 раз
меньше ресурсов

TPC-DS • SF=100 ГБ, Полное время

PostgresPro x BI.Qube

Меньше лучше



в 7 раз
выше скорость

до 10 раз
меньше ресурсов

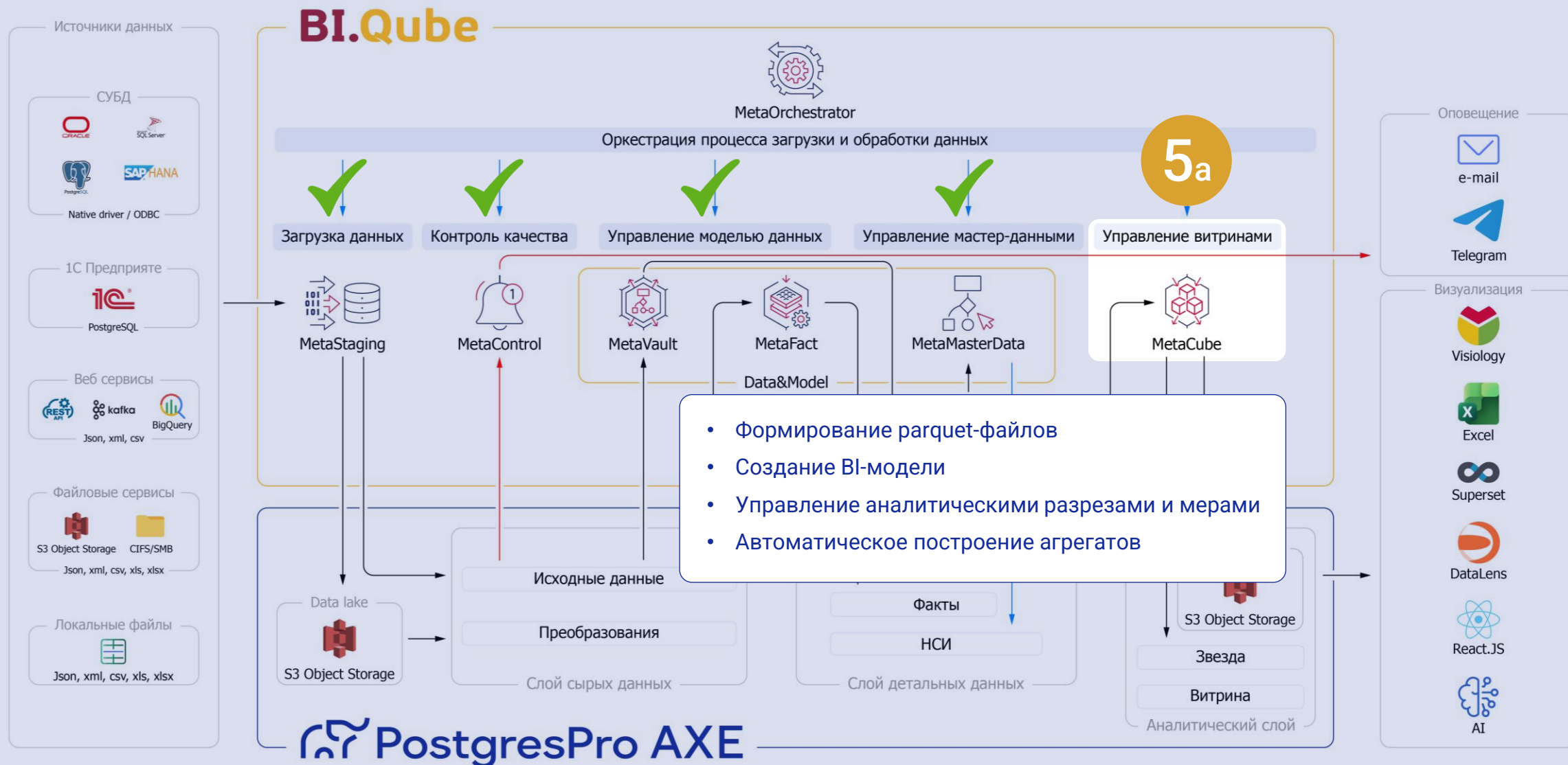
Давайте посмотрим на практике

Демонстрация создания витрин в АХЕ в инструментах BI.Qube

Демонстрация создания витрин в АХЕ в инструментах BI.Qube

Формирование реляционных витрин в Postgres Pro AXE

PostgresPro x BI.Qube



- Общее
- Staging
- Data & Model
- Control
- Cube
- Витрины**
- Профили
- Сессии
- Конструктор веб-форм
- Data Lineage
- Orchestrator

Витрины

Создать Удалить

Введите строку поиска ☐ Больше информации

Имя витрины	Адрес витрины	Домены модели	Описание
Витрина Продажи товаров	PostgresPro	pgpro_webinar	



Адрес модели

* Адрес витрины ?

Адрес витрины

Описание

Описание



00:00

00:41



Ответы на вопросы

Протестируйте самостоятельно
или закажите пилот:

Postgres Pro AXE

Фреймворк BI.Qube

Помощь консультантов
BI.Qube

От извлечения данных из источника
до построения витрины или дашборда

Оставьте заявку



postgrespro.ru

Благодарим за внимание!

 PostgresPro x **BI.Qube**

 PostgresPro



postgrespro.ru

BI.Qube



biqube.ru