Как кролик съел зеленую сливу и не умер:

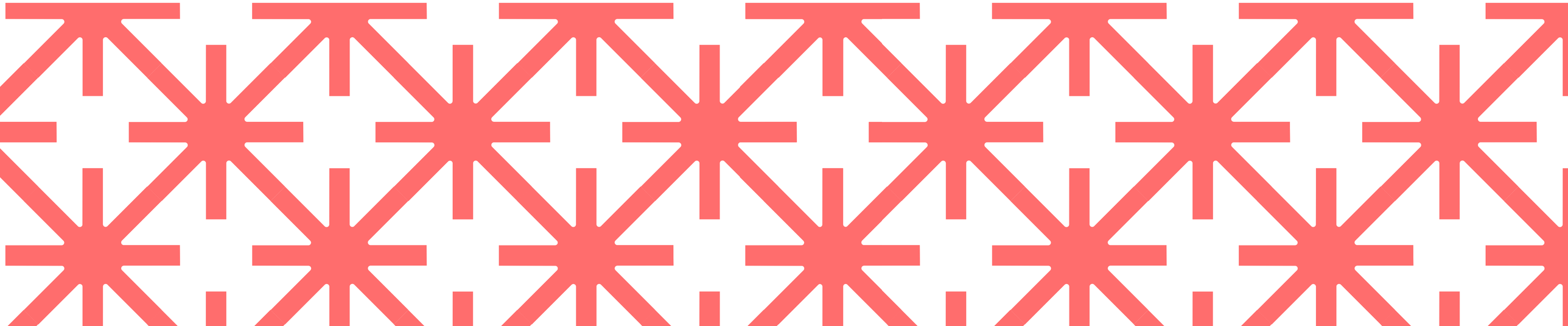
сказ о миграции на Iceberg

Валентин Пановский

Group CDO BestDoctor

BestDoctor

Введение



Краткая история DWH

70-е

Эдгар Кодд и реляционная модель данных

80-е

Термин “Business DWH” Барри Девлина и Пола Мёрфи

90-е

Появление Teradata, Oracle, IBM D2

Работы Билла Инмона и Ральфа Кимбалла

00-е

Появление Greenplum и развитие MPP подходов

10-е

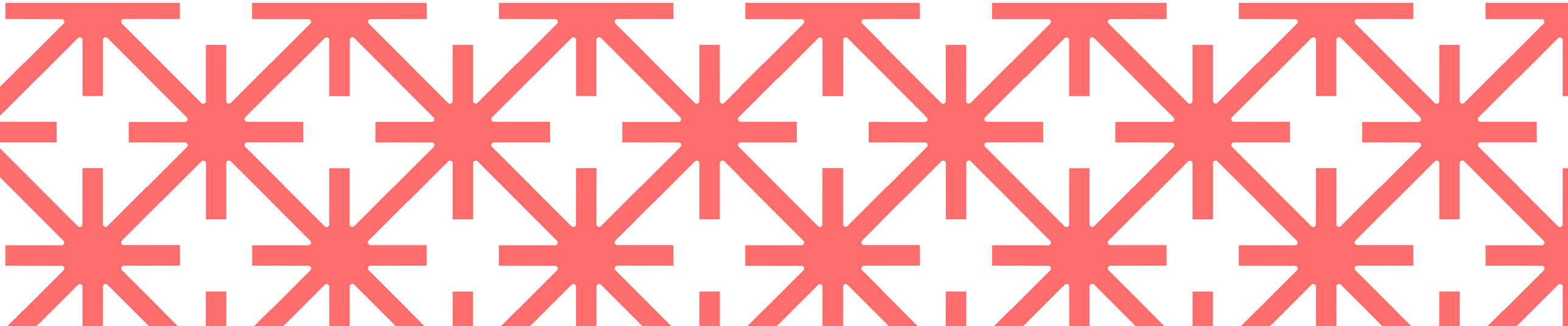
Облачные технологии в DWH: Amazon Redshift, Google BigQuery, Snowflake

20-е

Trino, Apache Doris, YugaByte,

... your future engine

Подходы к реализации DWH



Интегрированные архитектуры

(Monolithic DWH)

Хранилище данных и вычислительные ресурсы находятся в одном физическом пространстве или в одном программном решении (Greenplum, Teradata)

- Высокая производительность для специфических задач (в том числе за счёт отсутствия задержки из-за сетевых взаимодействий между компонентами)
- Упрощенная настройка и управление (по принципу “единого окна” + нет необходимости распыления в сторону освоения бОльшего технологического стека)

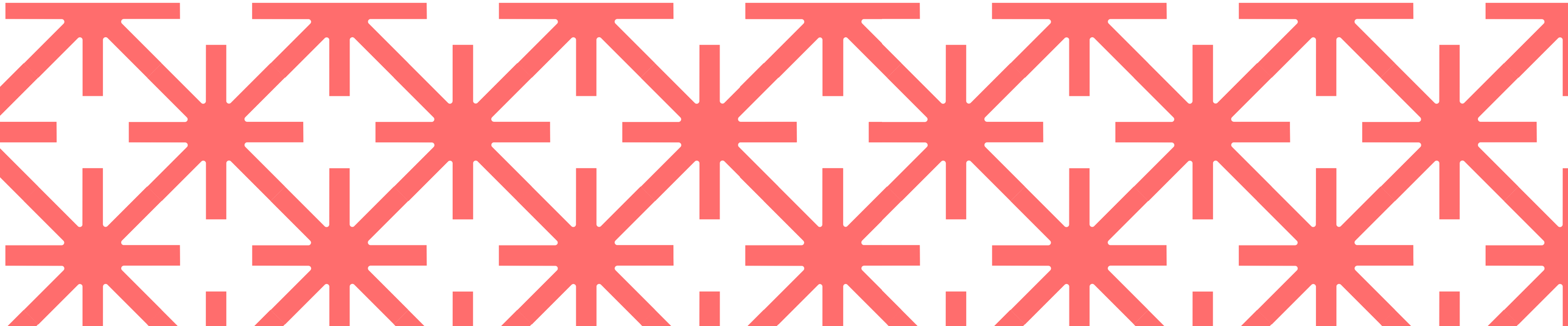
Разделенные архитектуры

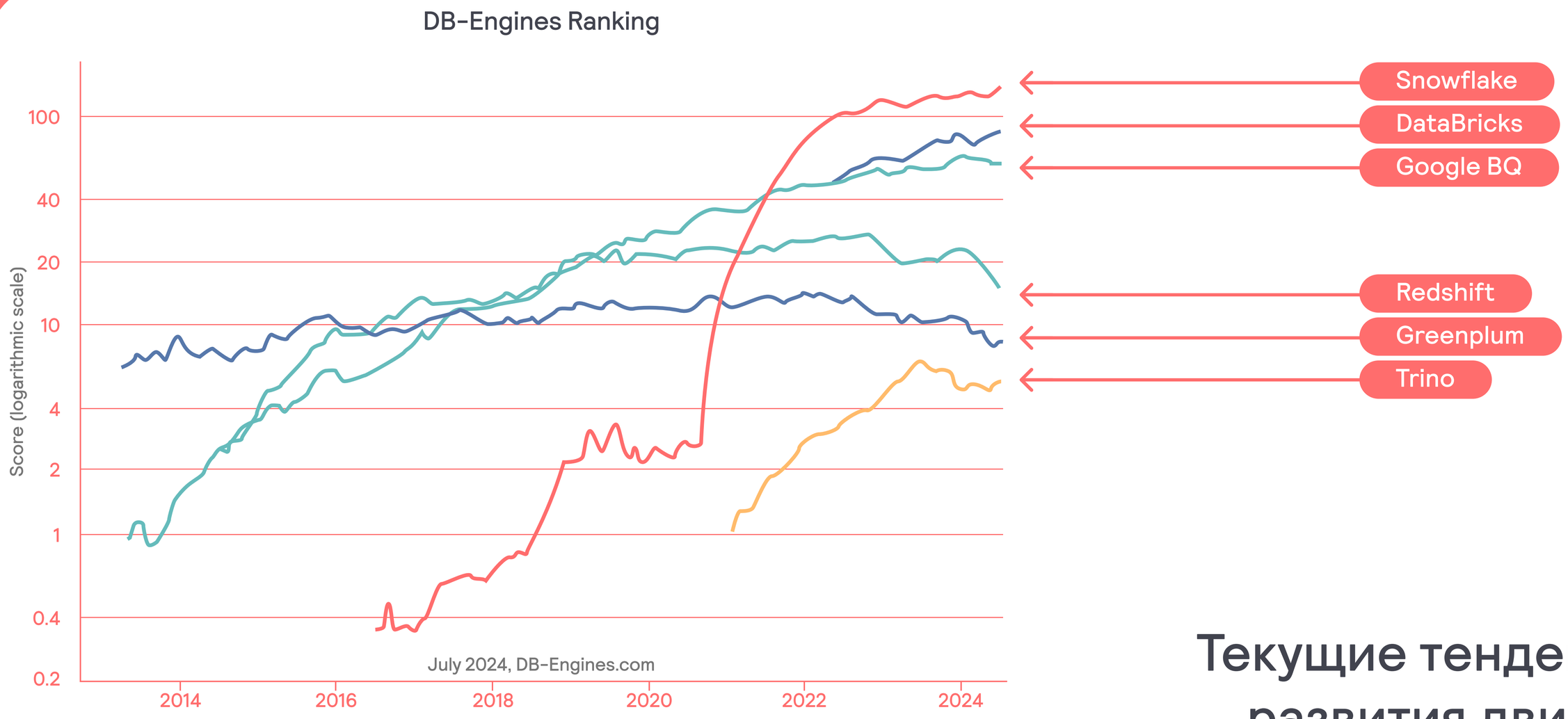
(Decoupled DWH)

В разделенных архитектурах компоненты хранения и вычислений разделены, что позволяет масштабировать их независимо друг от друга (Redshift, BigQuery, DataBricks)

- Возможность увеличивать хранилище без необходимости увеличения вычислительных мощностей, и наоборот
- Оптимизация затрат за счёт возможности интерактивного и оперативного конфигурирования слоёв обработки и хранения

Текущие тенденции развития движков DWH

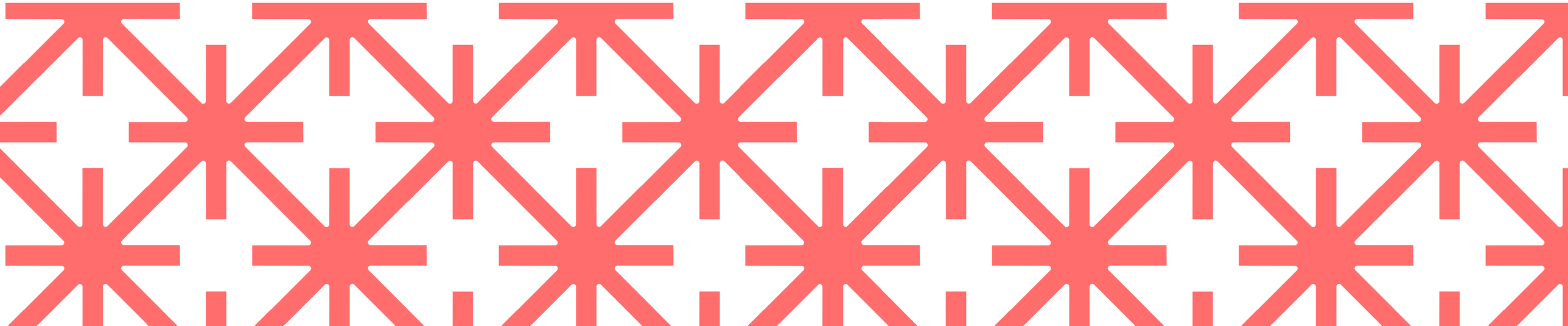




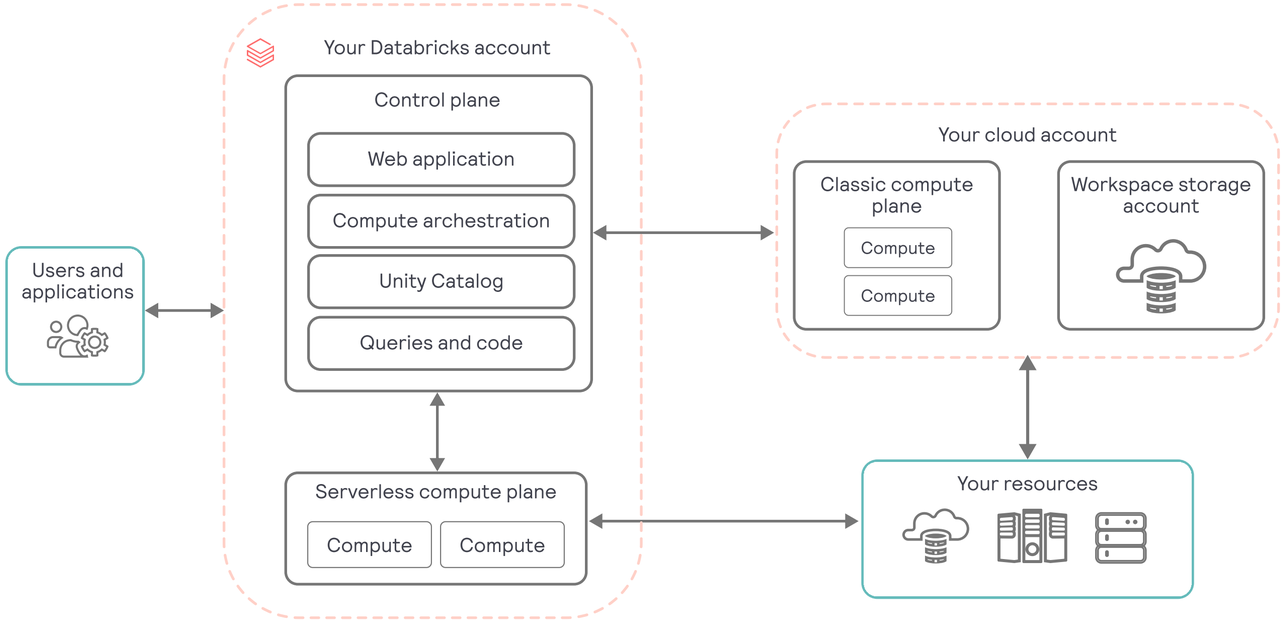
Текущие тенденции развития движков DWH

Текущие SaaS реализации разделённого подхода

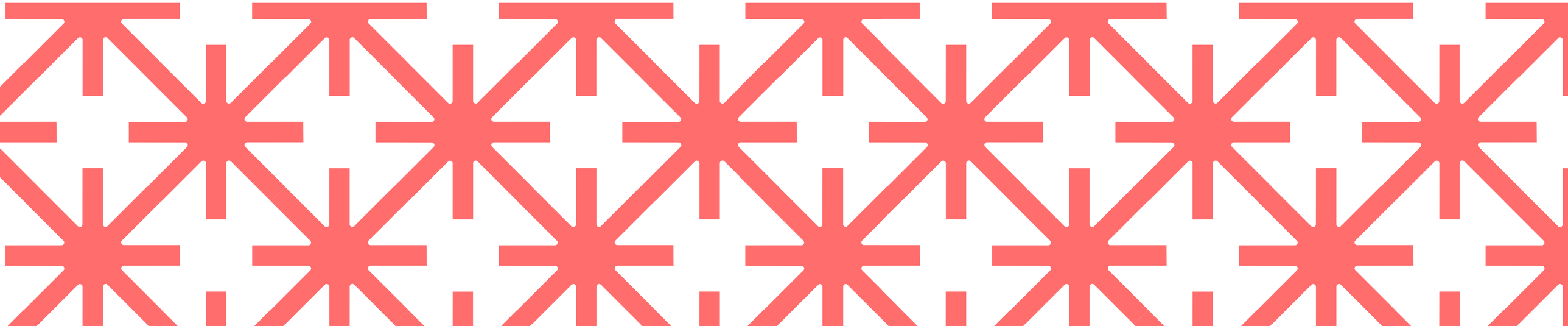
(на примере DataBricks)



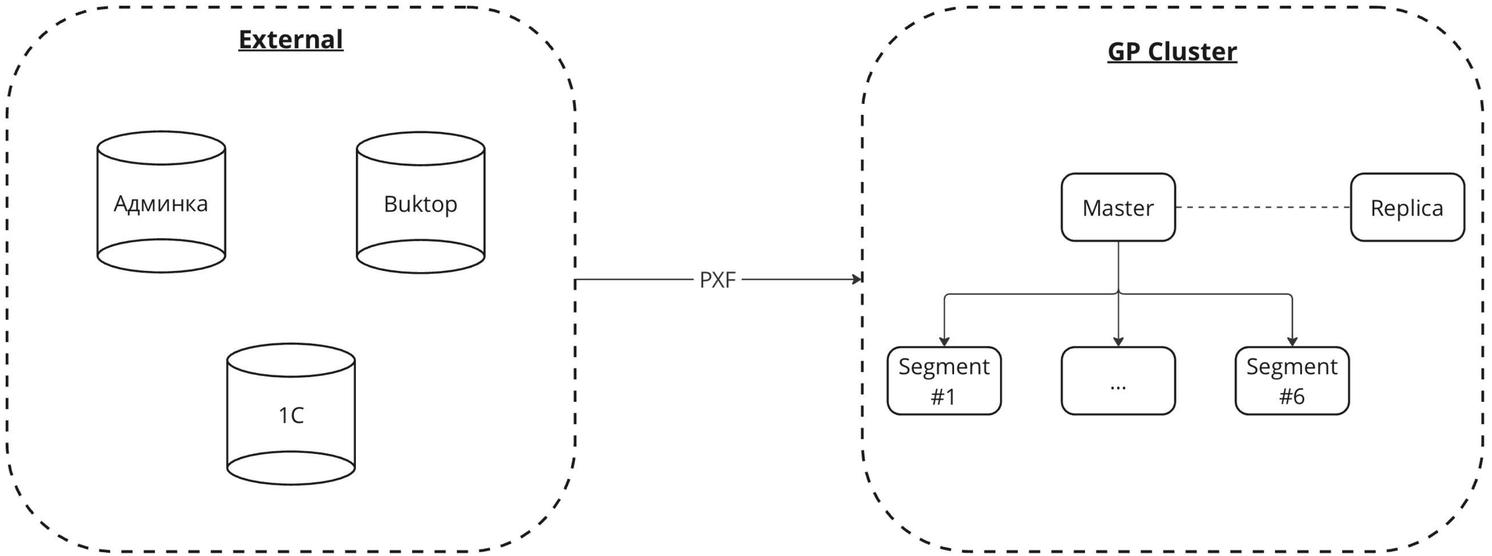
Текущие SaaS реализации разделённого подхода
(на примере DataBricks)



Архитектура DWH в BestDoctor



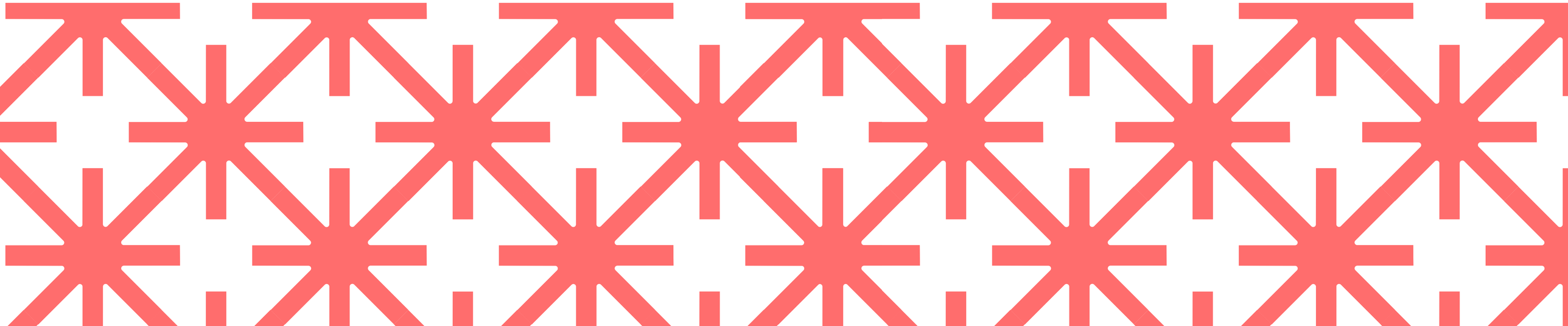
Архитектура DWH в BestDoctor



Недостатки

- Высокая стоимость из-за лицензирования + требования к конфигурациям серверов
- Необходимость гомогенного конфигурирования серверов (стреляет в ногу, если узкие места задействуют только один аспект ресурсов)
- Адаптер для DBT всё ещё в community режиме
- Наличие большого числа нюансов для партиционирования
- Нехватка современных процедур оптимизации аналитических запросов

Основные компоненты разделённого DWH



Технологические компоненты: Storage Layer

В концепции разделённых DWH слой хранения отвечает за физическое представление данных, над которым впоследствии выстраивается слой абстракции в виде Metastore

При этом последний позволяет однозначно смаппить пользовательский запрос на конкретные файлы, которые должны быть обработаны

Какие технологии могут быть использованы:

- Object/S3 Storage
- HDFS
- MapR FS
- NAS

Технологические компоненты: Compute Layer

Compute Layer, как и следует из названия, отвечает за преобразование и выполнение аналитического запроса в реалиях движка обработки (это может быть проксирование запроса на источник, обработка “сырых” файлов или собственные сценарии вычислений над данными)

Из существующих технологий:

- **Trino** - Fast distributed SQL query engine (на его базе разрабатывается CedrusData)
- **Dremio** - Unified Lakehouse Platform for Self-Service Analytics
- **Drill** - Schema-free SQL Query Engine
- **Doris** - Open Source, Real-Time Data Warehouse

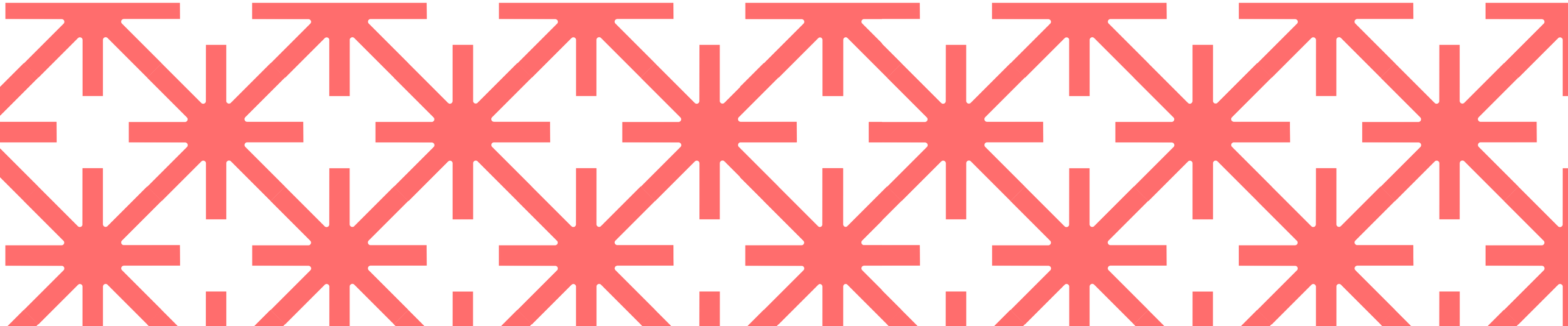
Технологические компоненты: Catalog/Metastore

Не стоит путать Catalog в смысле Metastore с каталогом данных как сущностью Data Governance! В контексте части DWH Catalog является хранилищем метаинформации о том, каким образом табличные данные (логический слой абстракции) связаны с физической системой хранения

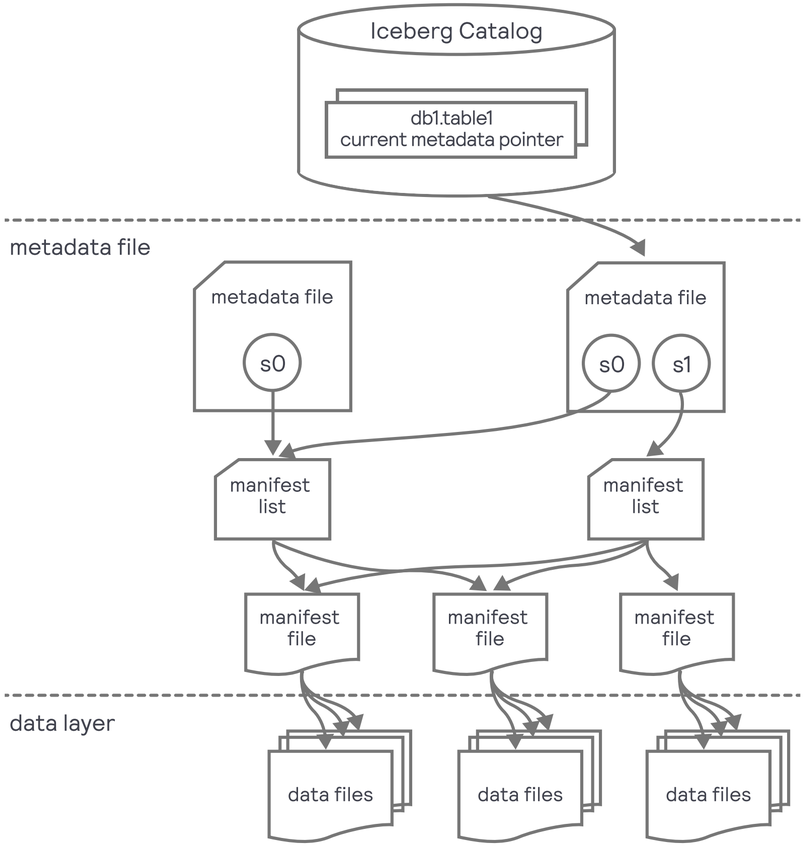
Способы реализации:

- Hive
- Iceberg REST Catalog
- Nessie server
- JDBC catalog

Что такое Iceberg Catalog?



Iceberg vs Parquet

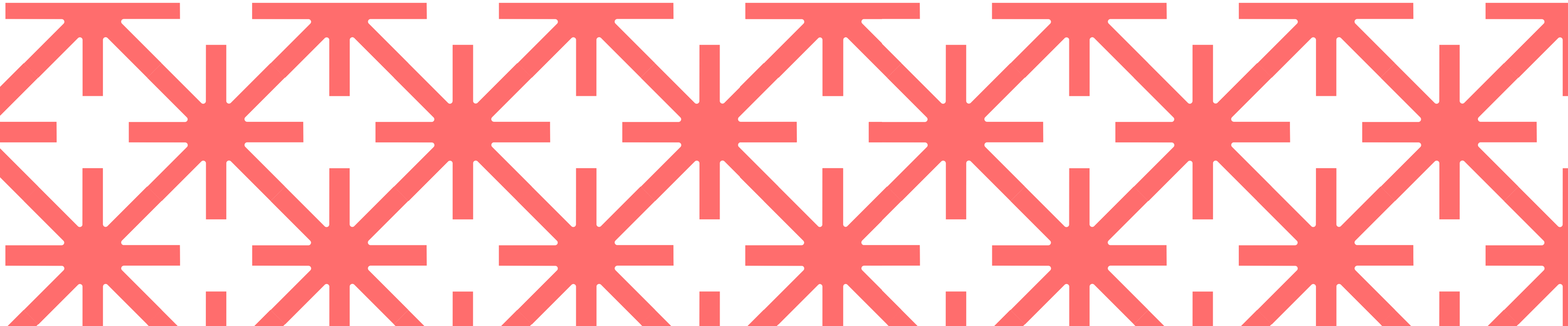


Пример работы с Iceberg REST Catalog

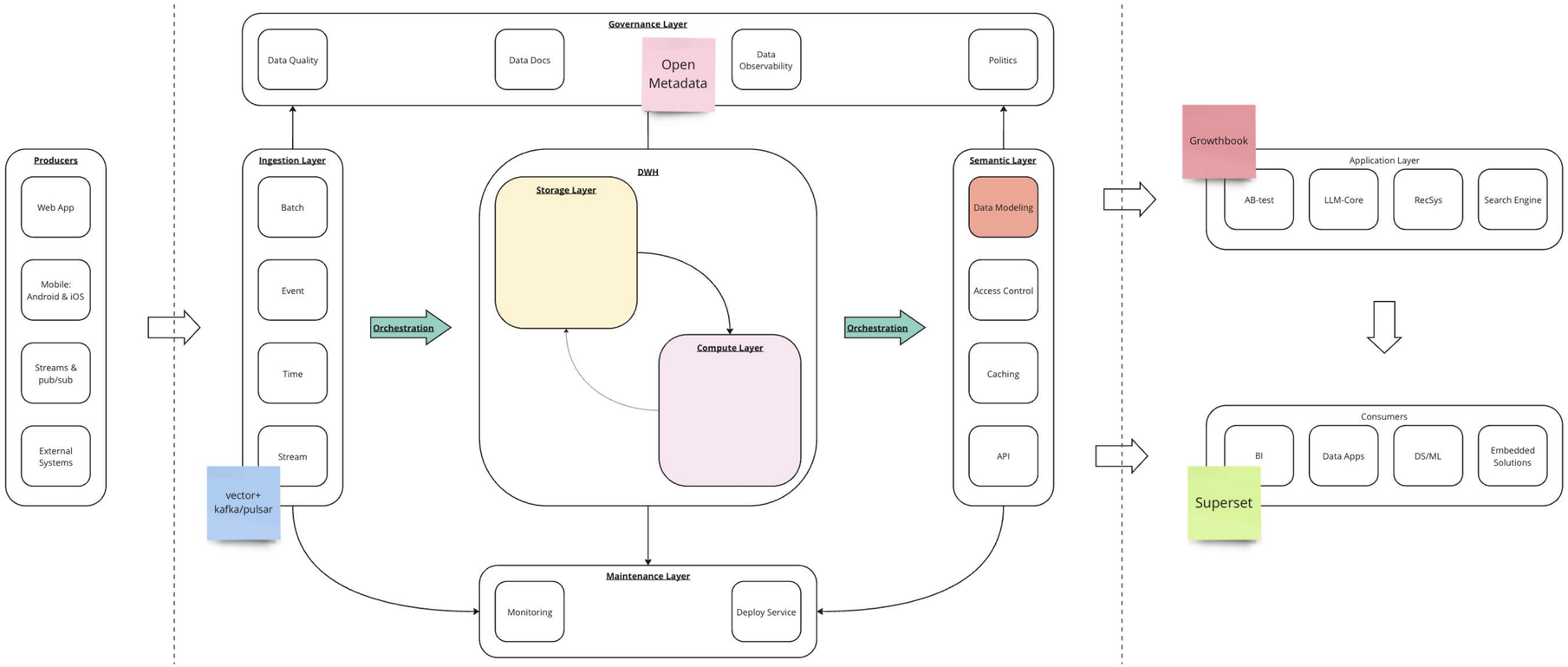
```python
from dotenv import load_dotenv
load_dotenv()

import os
from pyiceberg.catalog.rest import RestCatalog

catalog = RestCatalog(
    name="bd-catalog",
    uri=os.environ.get ("ICEBERG_REST_CATALOG_URL")
)

metainfo = catalog.load_table(os.environ.get("DEMO_TABLE"))
print(metainfo)
```

```
  1: insurer_id: optional string,
  2: order_id: optional string,
  3: local_order_date: optional date,
  4: local_order_time: optional timestamp,
  5: local_order_end_date: optional date,
  6: local_order_end_time: optional timestamp,
  7: number_of_passengers: optional int,
  8: request_type: optional string,
  9: date_query: optional date,
  10: load_ts: optional timestamp
),
partition by: [local_order_date],
sort order: [],
snapshot: Operation.APPEND: id=7705771013511283294, parent_id=3764969941883322876, schema_id=0
```

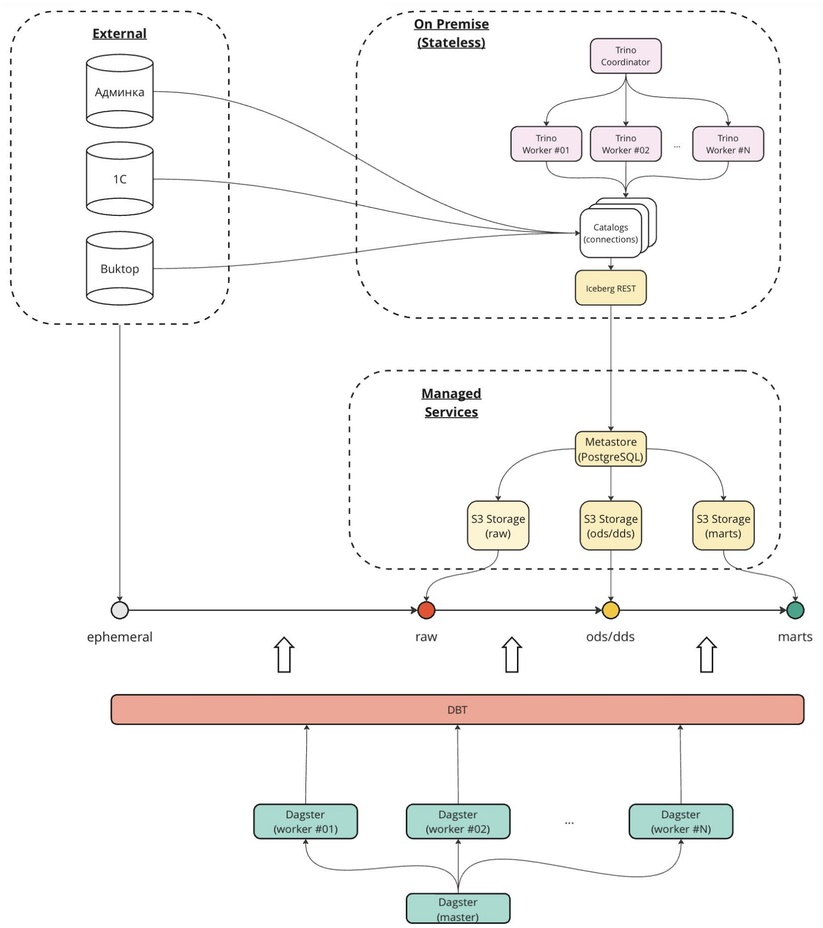
Архитектура новой платформы данных в BestDoctor



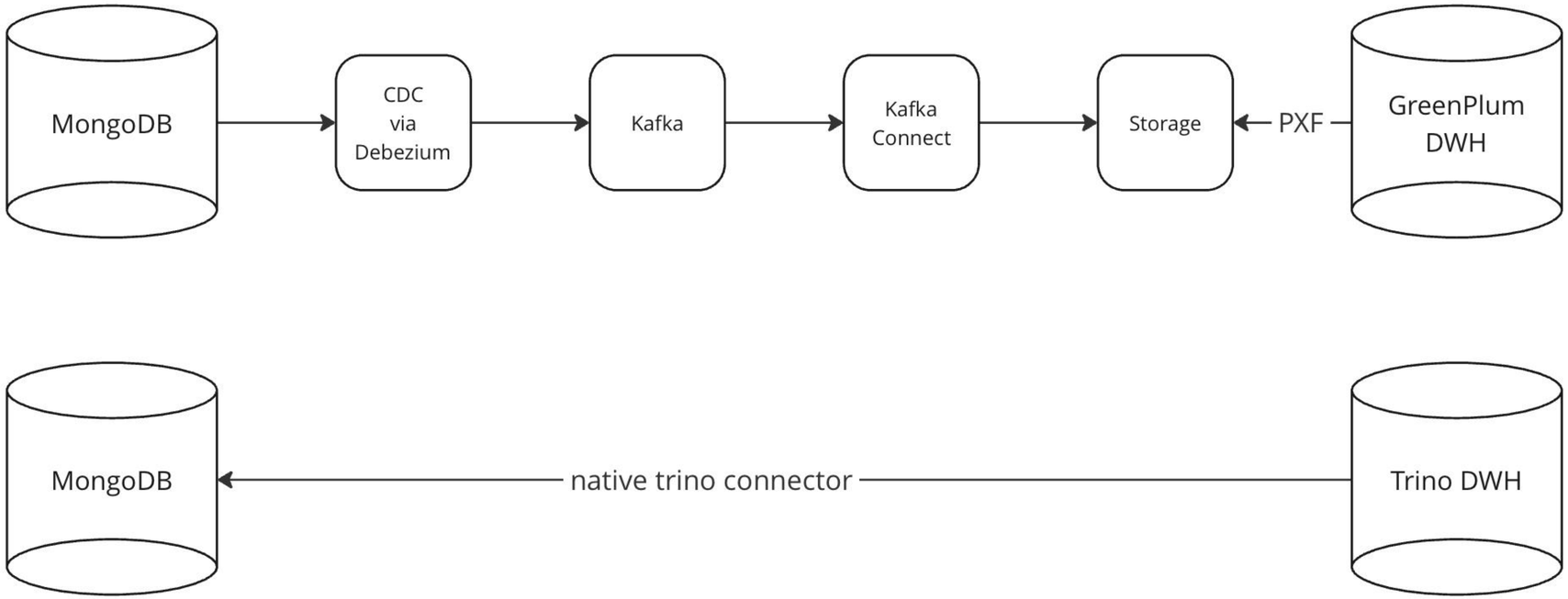
Архитектура Платформы Данных в BestDoctor



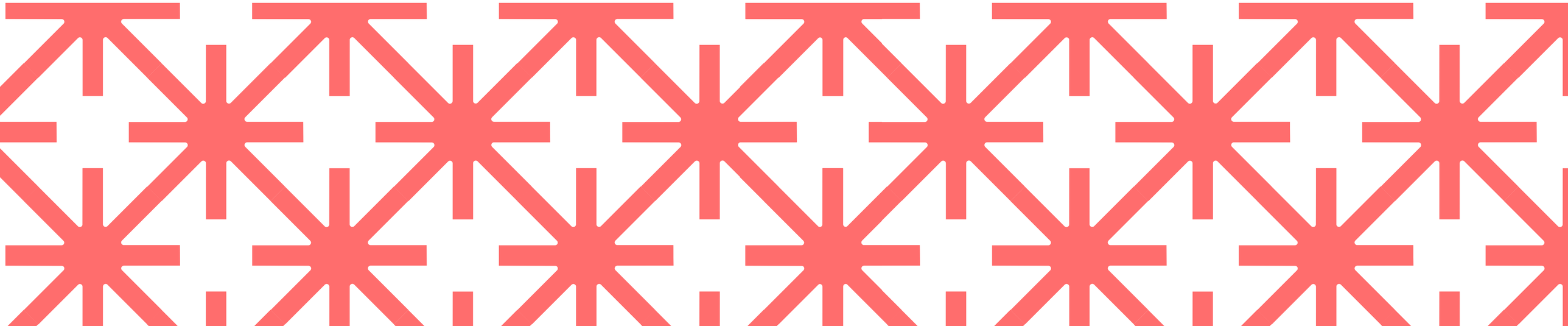
Архитектура DWH в BestDoctor



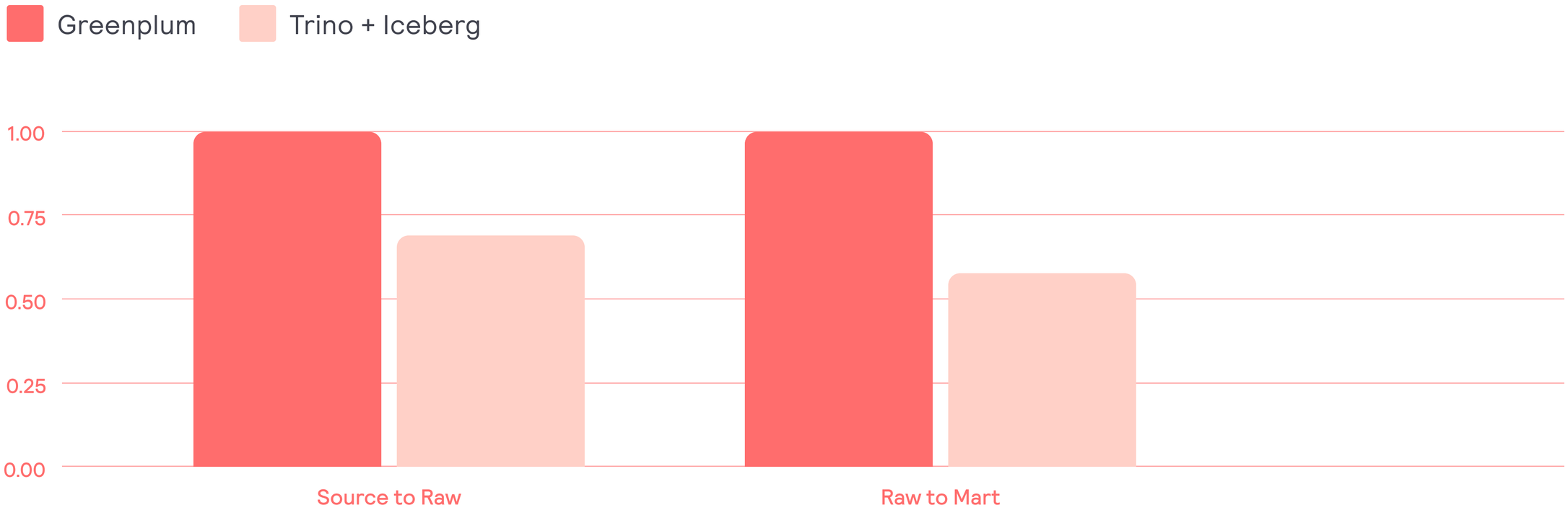
Сценарий облегчения процесса работы



Опыт эксплуатации



Оптимизация по времени исполнения процедур



Негативные кейсы

- Деградация взаимодействия с DWH в non-OLAP сценариях
- Сложность в миграции актуарных витрин и процедур расчёта из-за широкого уровня проникновения хранимых процедур
- Много boilerplate’а при работе с данными из 1С за счёт использования в последней обилия кастомных типов

Полезные ссылки

OpenAPI Spec для REST Catalog

Перейти

Плагин для Kafka Connect’a

Перейти

Интеграция ClickHouse с Iceberg’ом

Перейти

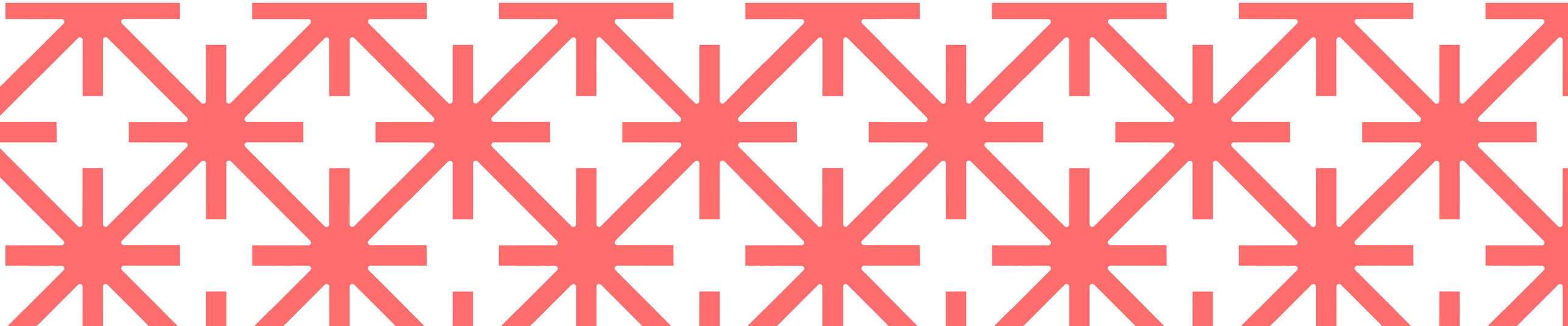
Плагин для Flink’a

Перейти

БД, ориентированная на работу со snapshot’ами данных

Перейти

Заключение и вопросы



Спасибо за внимание!

btw let’s keep in touch (:

Валентин Пановский

Group CDO BestDoctor

@PANOVSKIY_V

BestDoctor