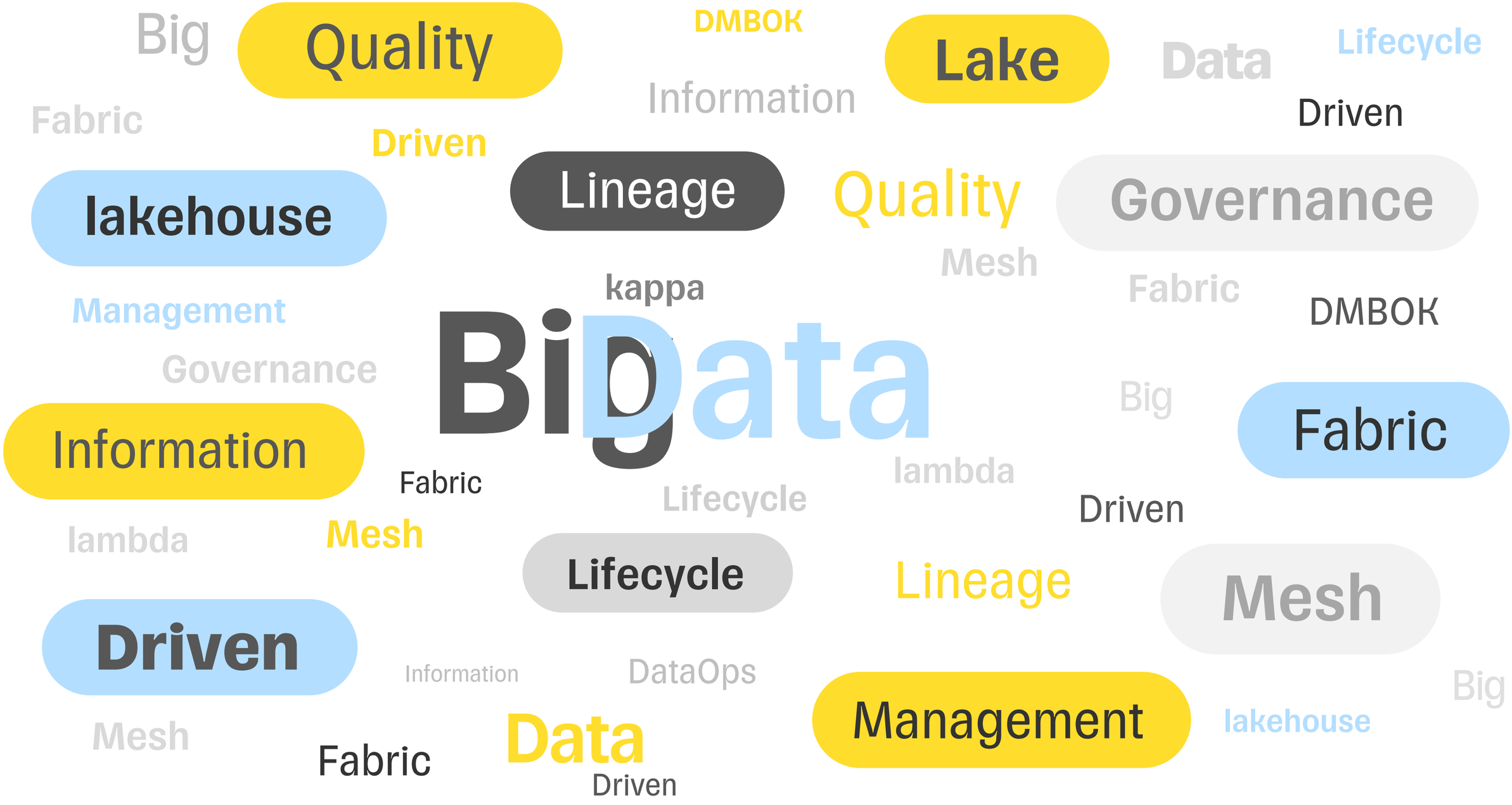
Т БАНК

ETL на ~~7000~~ 8000 процессов

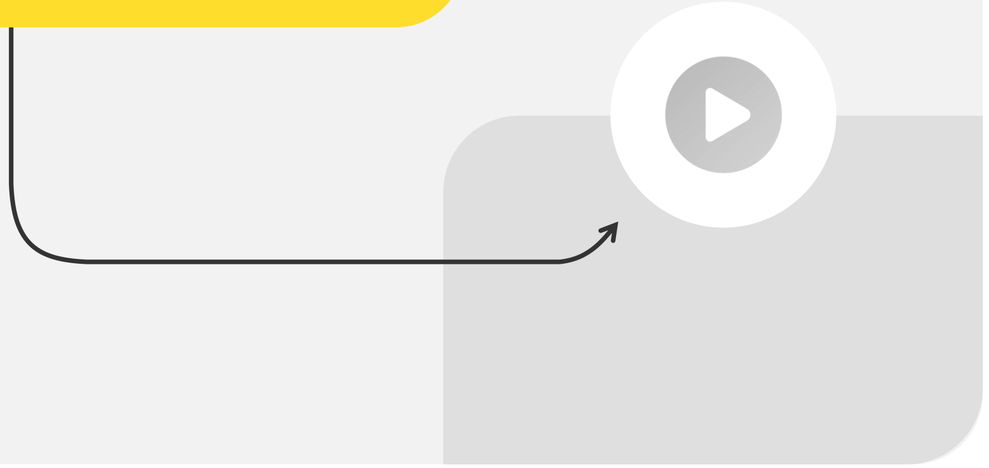

Big
Quality
DMBOK
Lake
Data
Lifecycle
Information
Fabric
Driven
Driven
Lineage
Quality
Governance
lakehouse
Mesh
kappa
Fabric
DMBOK
Management
Big Data
Governance
Big
Fabric
Information
Fabric
lambda
Lifecycle
Driven
lambda
Mesh
Lifecycle
Lineage
Mesh
Driven
Information
DataOps
Big
Management
lakehouse
Mesh
Fabric
Data
Driven

Самый простой ETL

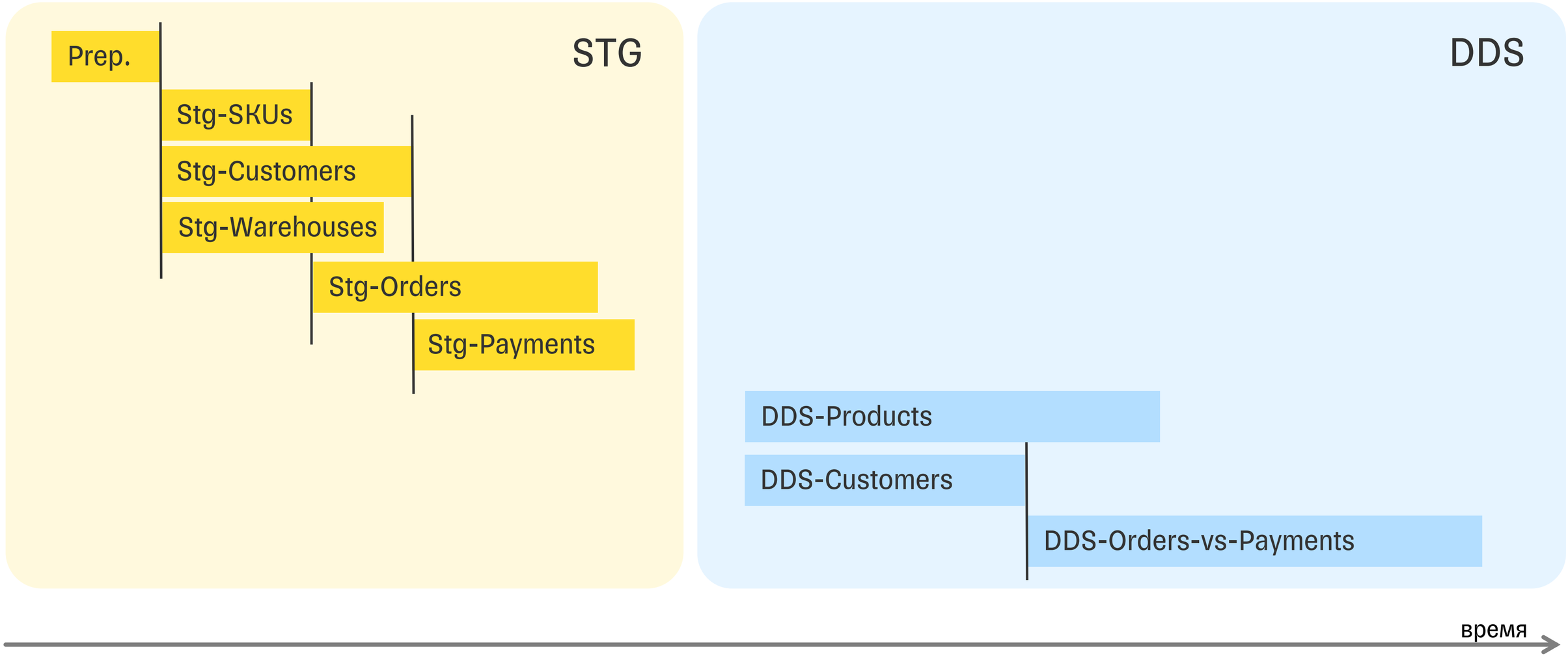
```
TRUNCATE TABLE dds.orders;

INSERT INTO dds.orders
SELECT
...
FROM
    stg.orders AS o
    INNER JOIN dds.customers AS c
        ON o.customer_id = c.customer_id
```

Запускаем по расписанию
с помощью

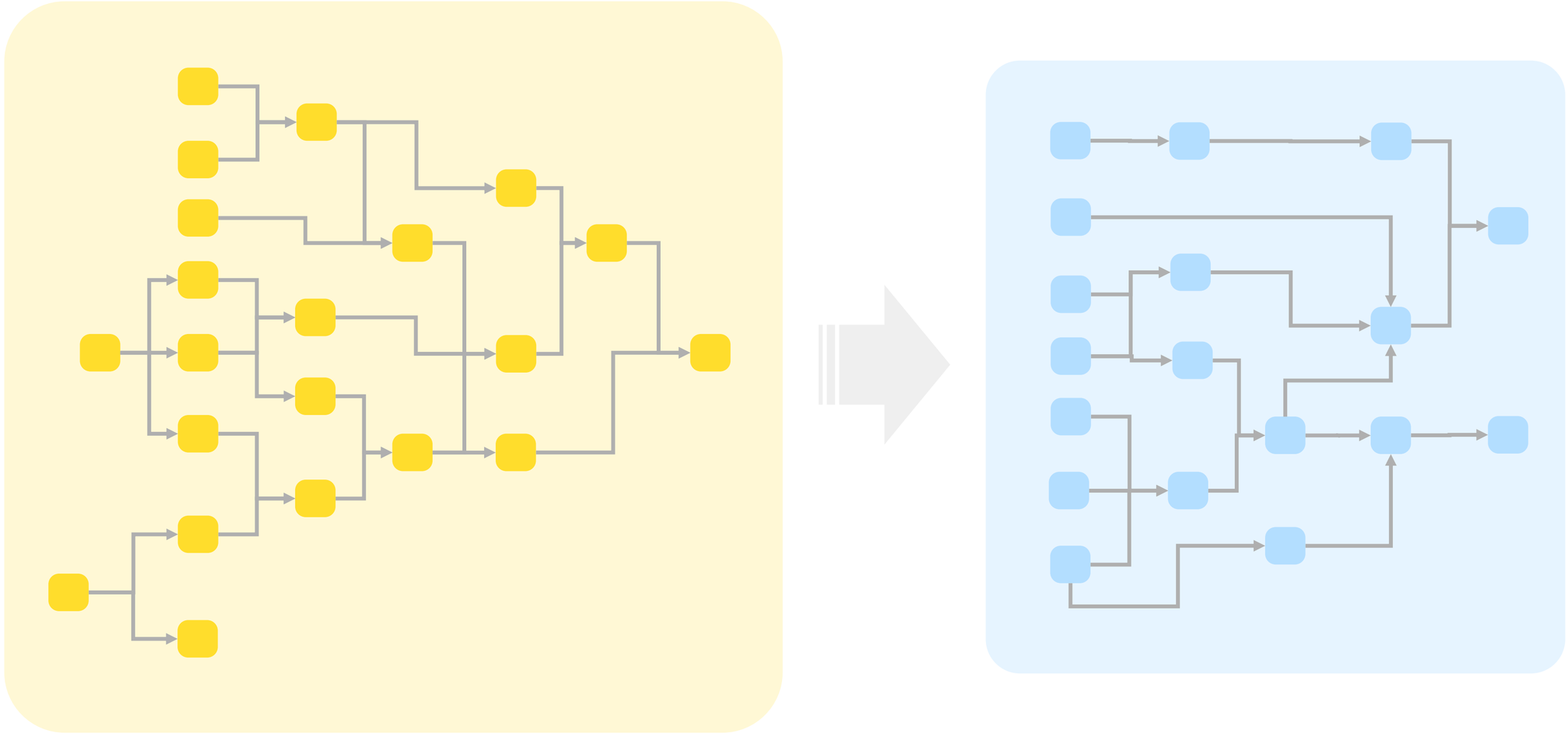
psql + cron



Много процессов



Направленный ациклический граф



Как запускать процессы лучше?

У нас более 8000 ETL процессов

Мы не можем нарисовать оптимальный граф руками

Нам не хватает ночного времени для построения хранилища

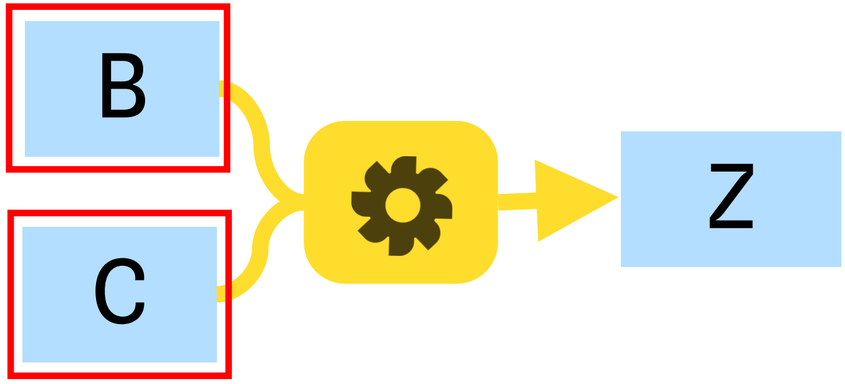
Автоматическое формирование последовательности выполнения

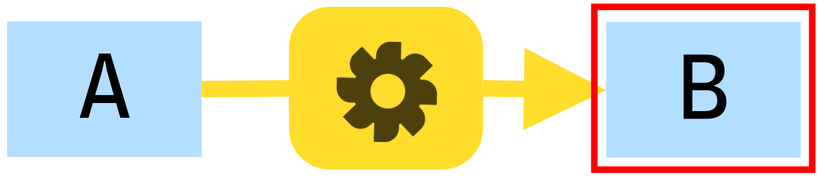
Data Lineage

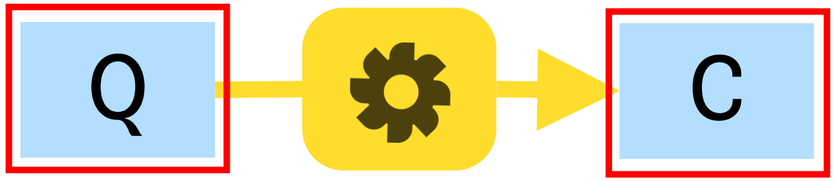
+

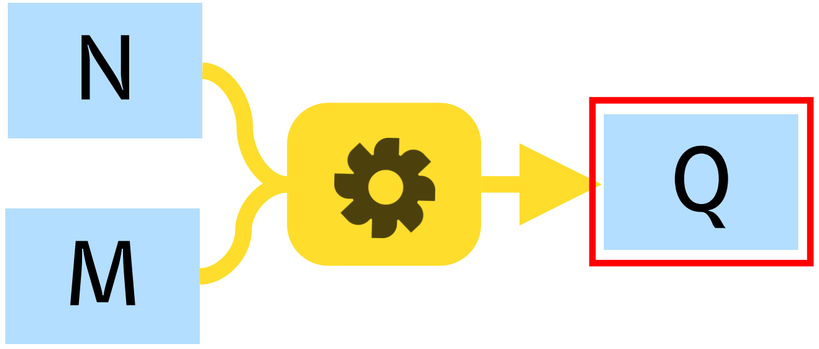
Свой планировщик

Идея простыми словами







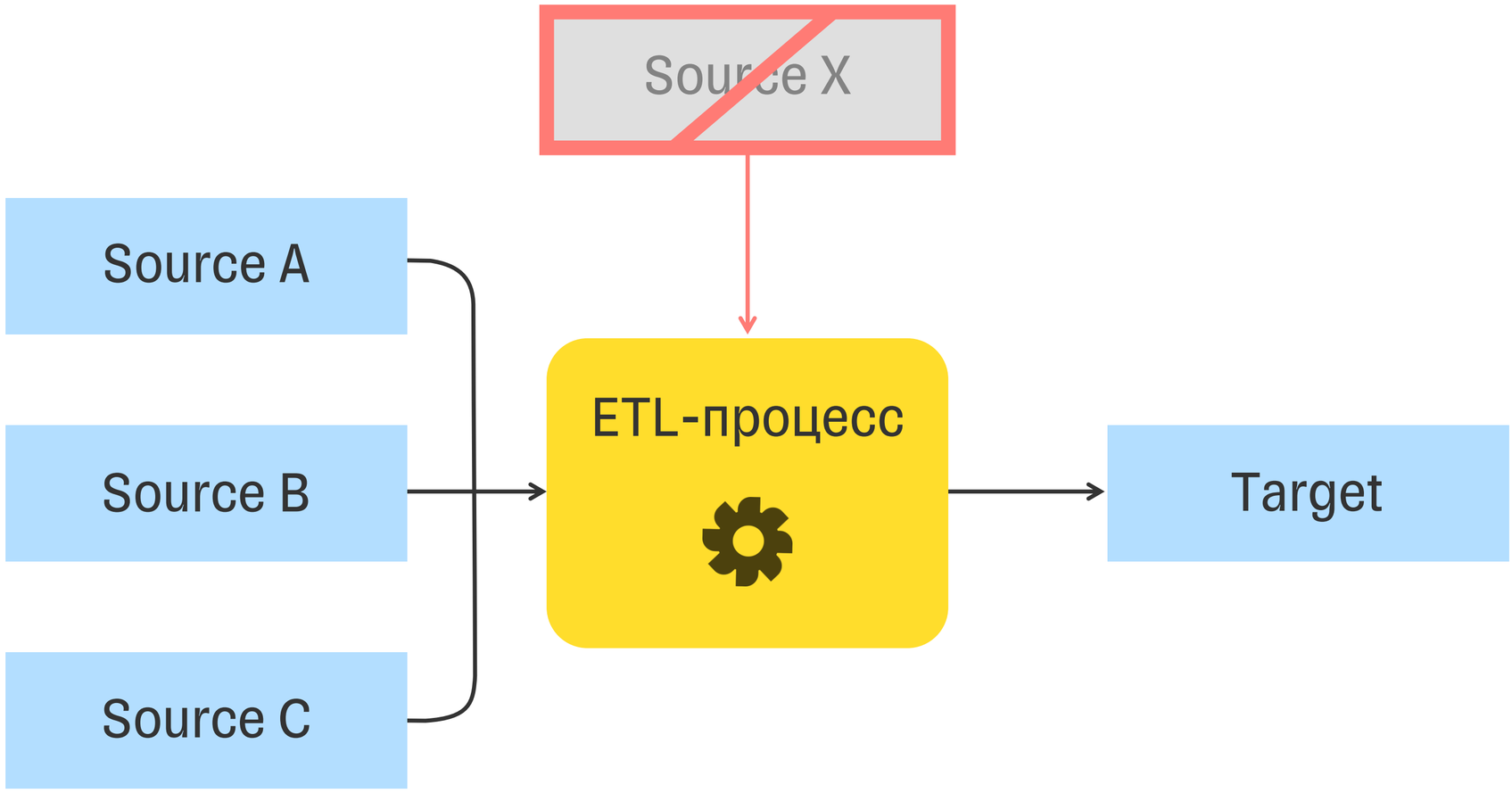


Lineage

Достаточно lineage по таблицам, без столбцов

Про каждый процесс нужно знать	Как правило, этой информации нет
Какие нужны источники (таблицы)	Airflow – нет такой фичи
Куда (в какую таблицу) он пишет данные	Microsoft Integration Services – можно было бы, но нет

Lineage в Т-Банке



Как это работает у нас?

У нас есть свой ETL-инструмент, **tedi**

- Он знает источники и приемники каждого процесса
- Невозможно сделать процесс, который не сообщает lineage

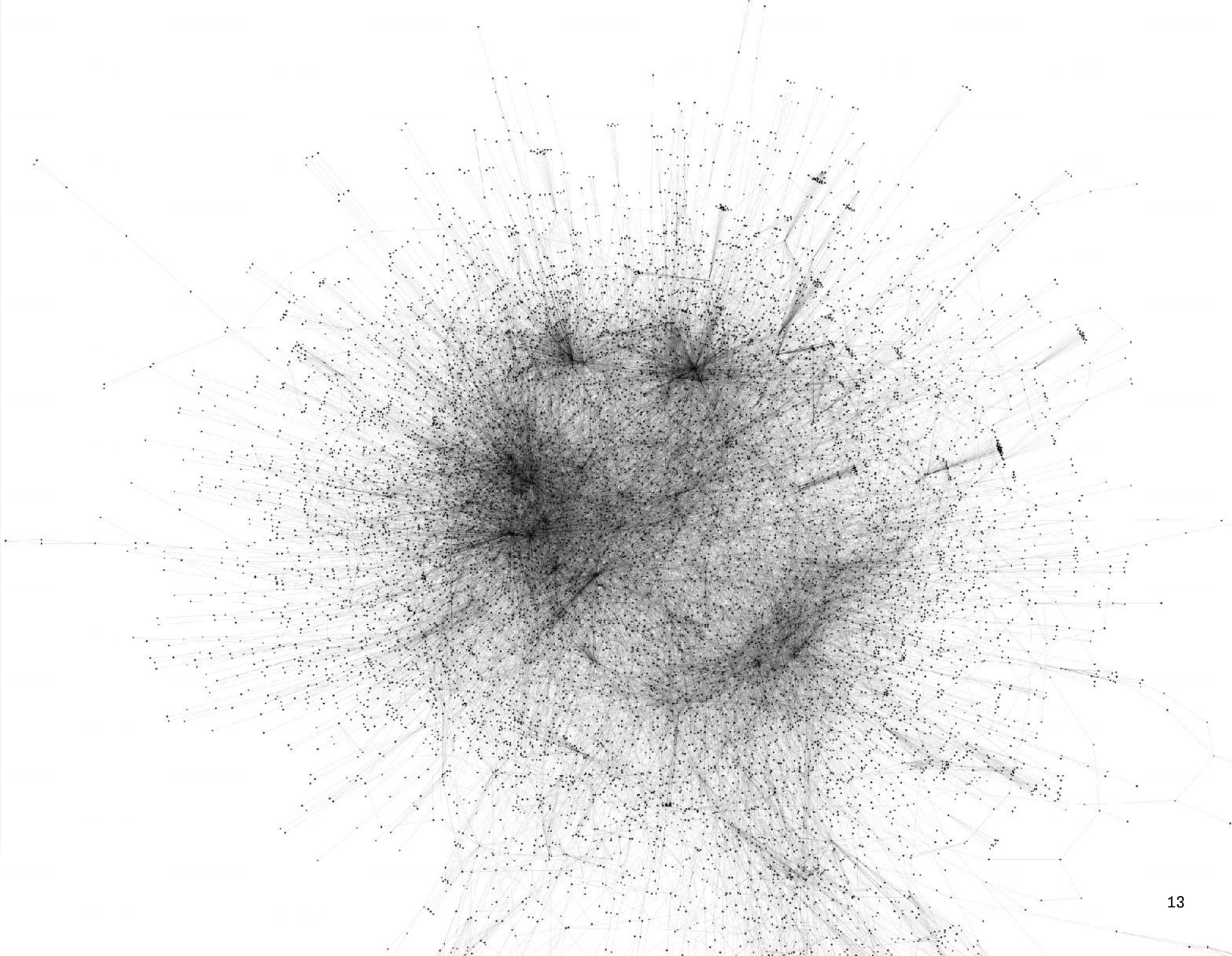
У нас есть свой планировщик – **moebius**

Чтобы запустить процесс, ему нужно сообщить, какие нужны источники

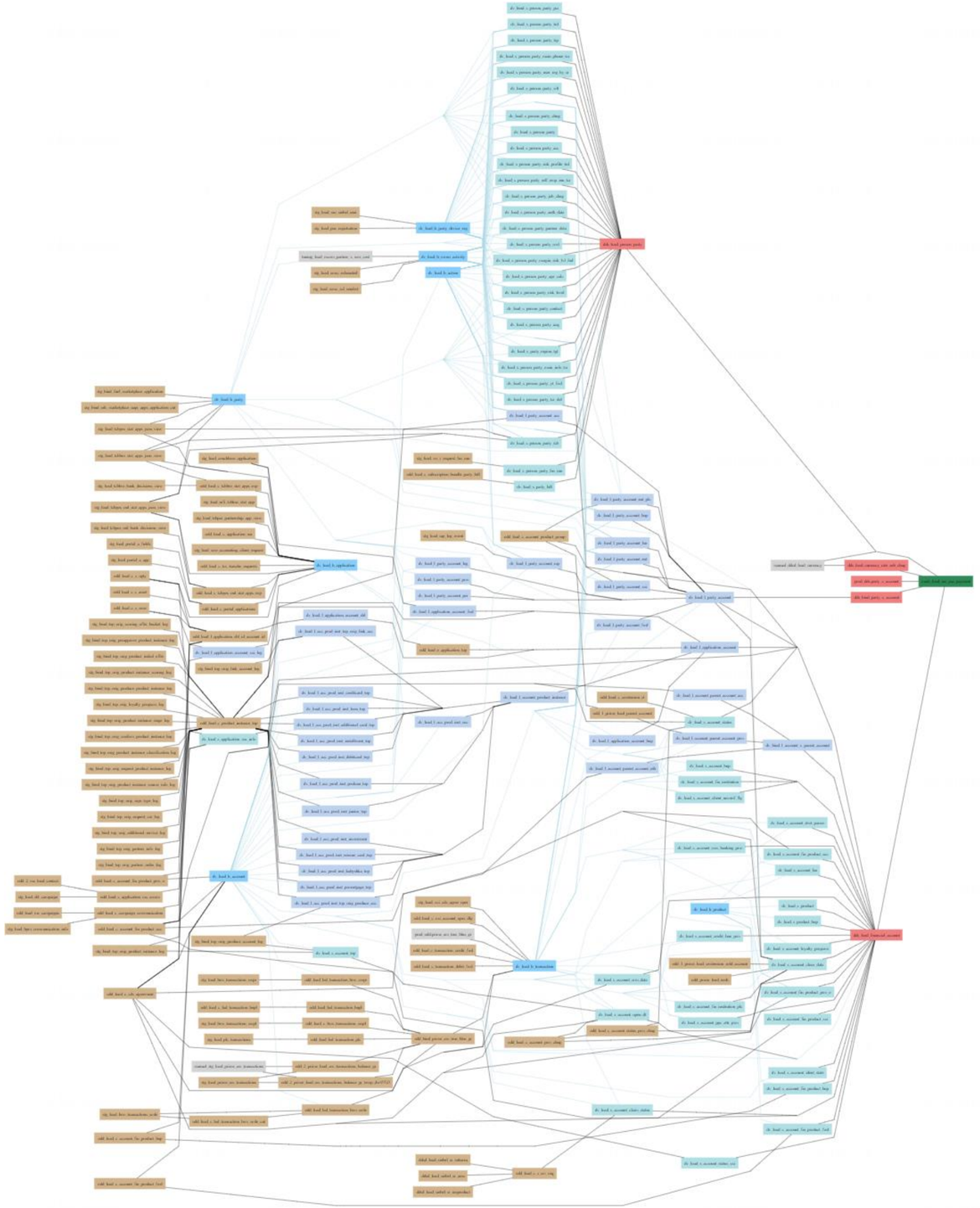
У нас есть сервис **актуальности** таблиц

- Обновил данные в таблице?
- Сообщи всем заинтересованным!

Все целиком



Одна витрина



Усложнение второе

Много данных

Уже 9 утра,
а ETL-процессы
не завершились

А раньше
успевали...

WTF? Мы каждый раз пересчитываем все заново!

Полная загрузка

```
TRUNCATE TABLE dds.orders;


INSERT INTO dds.orders
SELECT
...
FROM
    stg.orders AS o
    INNER JOIN dds.customers AS c
        ON o.customer_id = c.customer_id
```

Инкрементальная загрузка

```
TRUNCATE TABLE dds.orders;


INSERT INTO dds.orders
SELECT
...
FROM
    stg.orders AS o
    INNER JOIN dds.customers AS c
        ON o.customer_id = c.customer_id
```

Проблемы инкрементальной загрузки

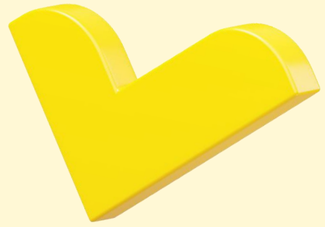
Инкремент выделить сложно

Все полные загрузки похожи друг на друга, каждая инкрементальная загрузка инкрементальна по-своему

Нужны две версии ETL-процесса: полная и инкрементальная

Перемешивать выделение инкремента и бизнес-логику — неправильно

Альтернативный способ



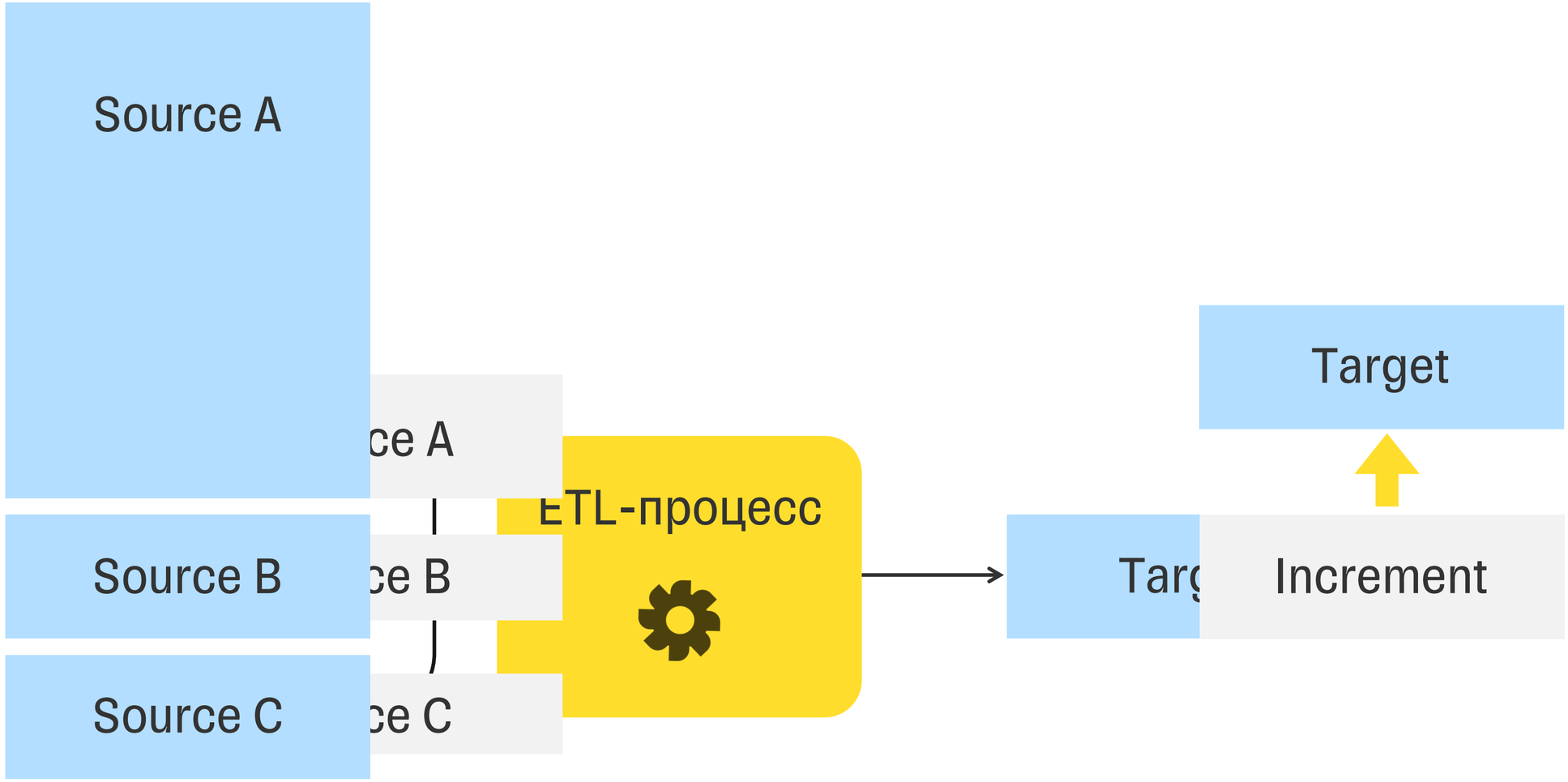
Формирование
консистентного среза

В ETL процесс
не вмешиваемся

Вместо полных данных
подсовываем на вход **инкремент**

Truncate делать
нельзя

Формирование консистентного среза



Как это работает у нас?

ETL инструмент умеет подменять таблички

Это возможно, потому что все входы и выходы жестко контролируются

Есть отдельный сервис, который формирует консистентный срез

- Откуда мы знаем, как его сделать?
- Это нужно настроить вручную

Усложнение третье

Место закончилось

В хранилище куча каких-то табличек

И главное, можно ли их удалить?

- Мы не знаем, что это за таблички
- Мы не знаем, актуальные ли там данные
- Мы не знаем, кто пользуется этими данными

Как ответить на эти вопросы?

Кто выполняет запросы к данным?

Мы поймем, кому они нужны

Кто дорабатывал ETL-процессы?

Мы поймем, кто технический владелец

Кто ставил задачи в Jira?

Мы поймем, кто бизнес-владелец

Как мы это делаем?

Data Detective

Централизованный источник информации о таблицах и процессах

Lifecycle management

У таблицы есть владелец

Просим подтвердить, что данные нужны

Guillotine

Старые данные удаляются по расписанию

Усложнение четвертое

Набежали пользователи

Наше чудесное хранилище стало очень популярно у пользователей

- Утилизация CPU 100%
- Disk latency > 100ms
- Память кончилась

Масштабирование

Мы тут

Сюда мы стремимся

Level 1

SMP → MPP СУБД

или

SMP → Hadoop

Level 2

MPP → много MPP

Level 3

Compute / storage separation

Massively Parallel Processing

Вместо одного сервера ставим кластер

Данные равномерно размазываем по узлам

Profit!

MPP хуже чем SMP

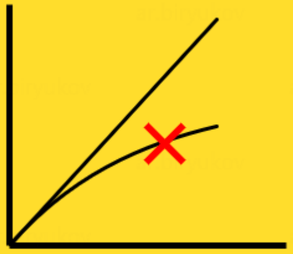
Основная проблема – данные нужно передавать по сети между нодами

Перекос

Мастер-нода

Много MPP

Пытаемся преодолеть нелинейное масштабирование



Одни и те же данные на всех кластерах

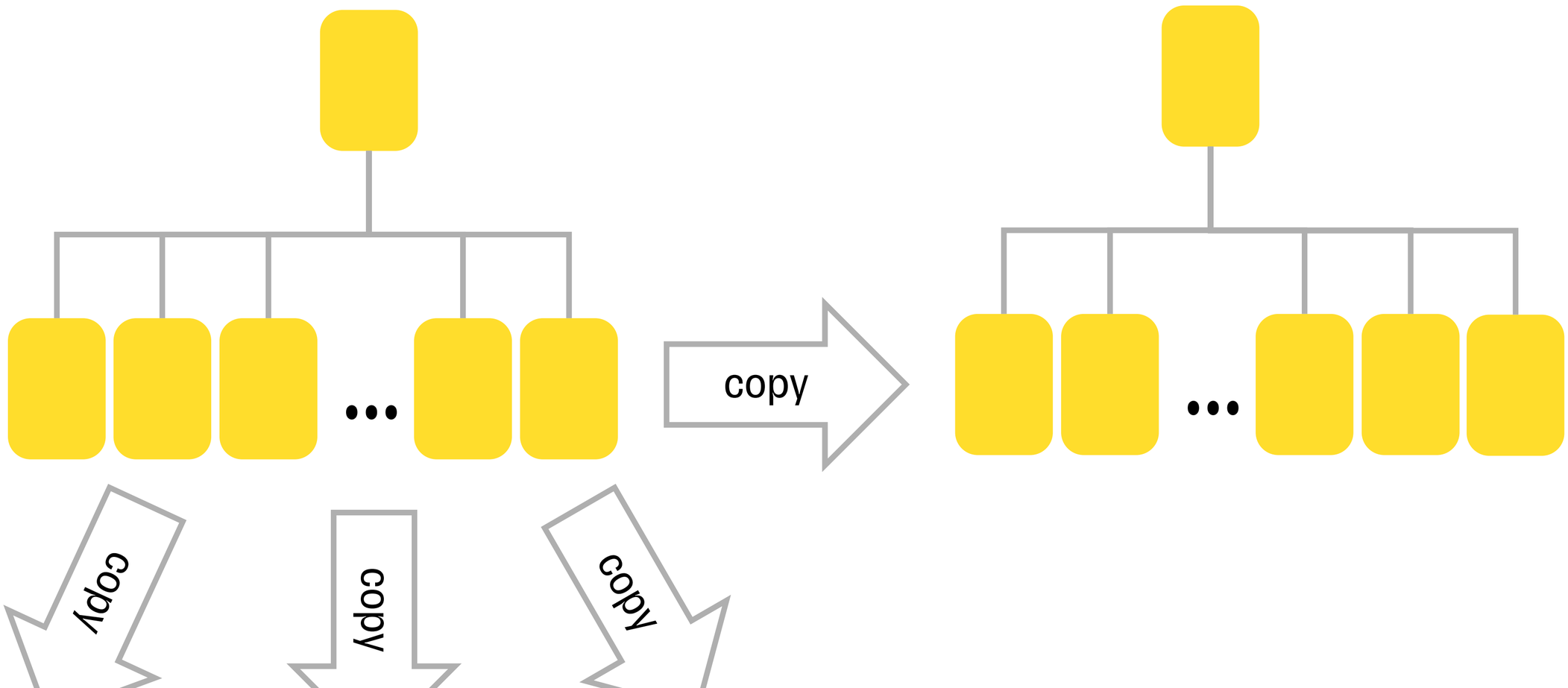
~Data Fabric

Разные данные на всех кластерах

~Data Mesh

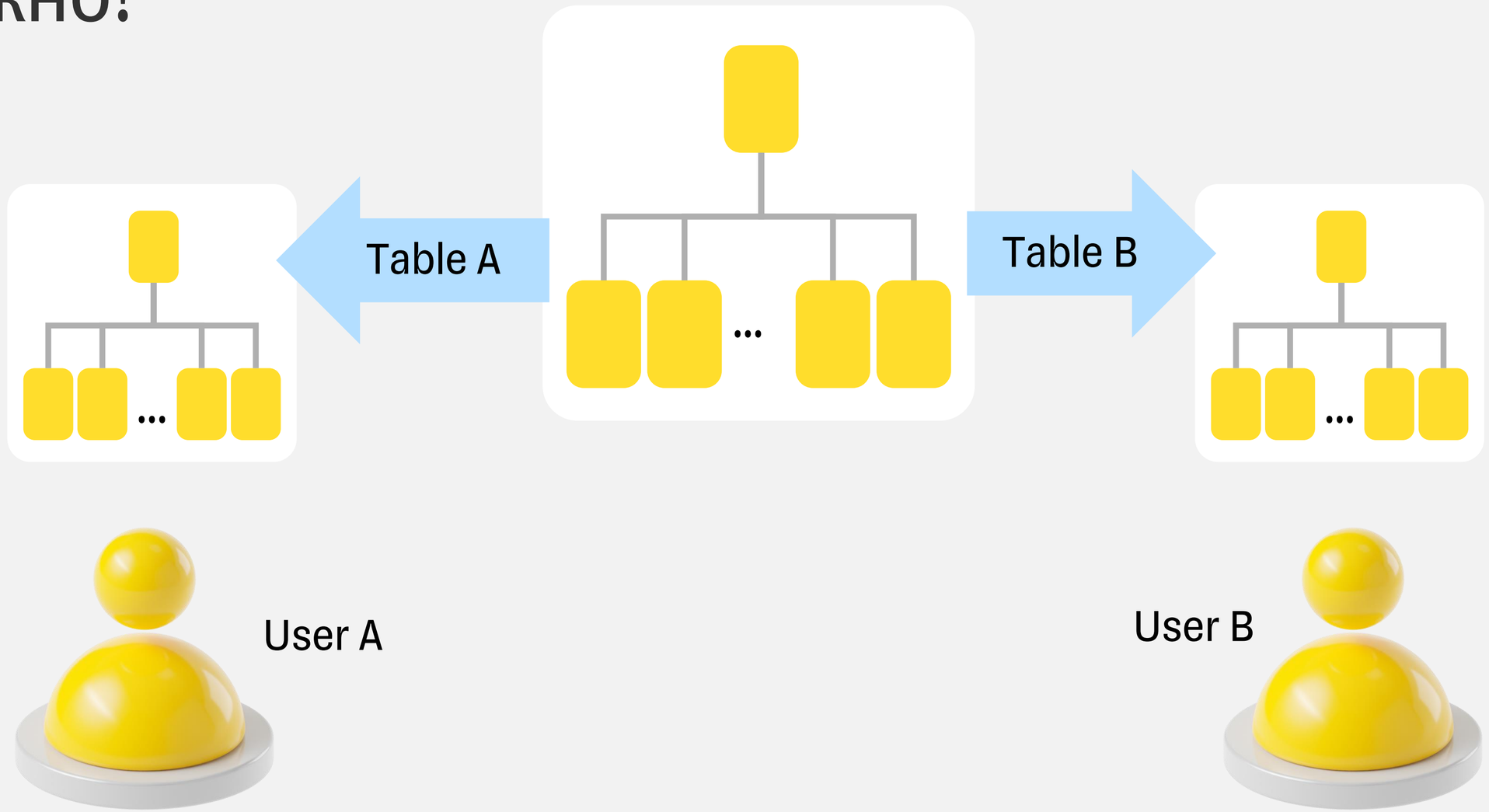
Много MPP

Одни и те же данные на всех кластерах



А можно все не копировать?

Можно!

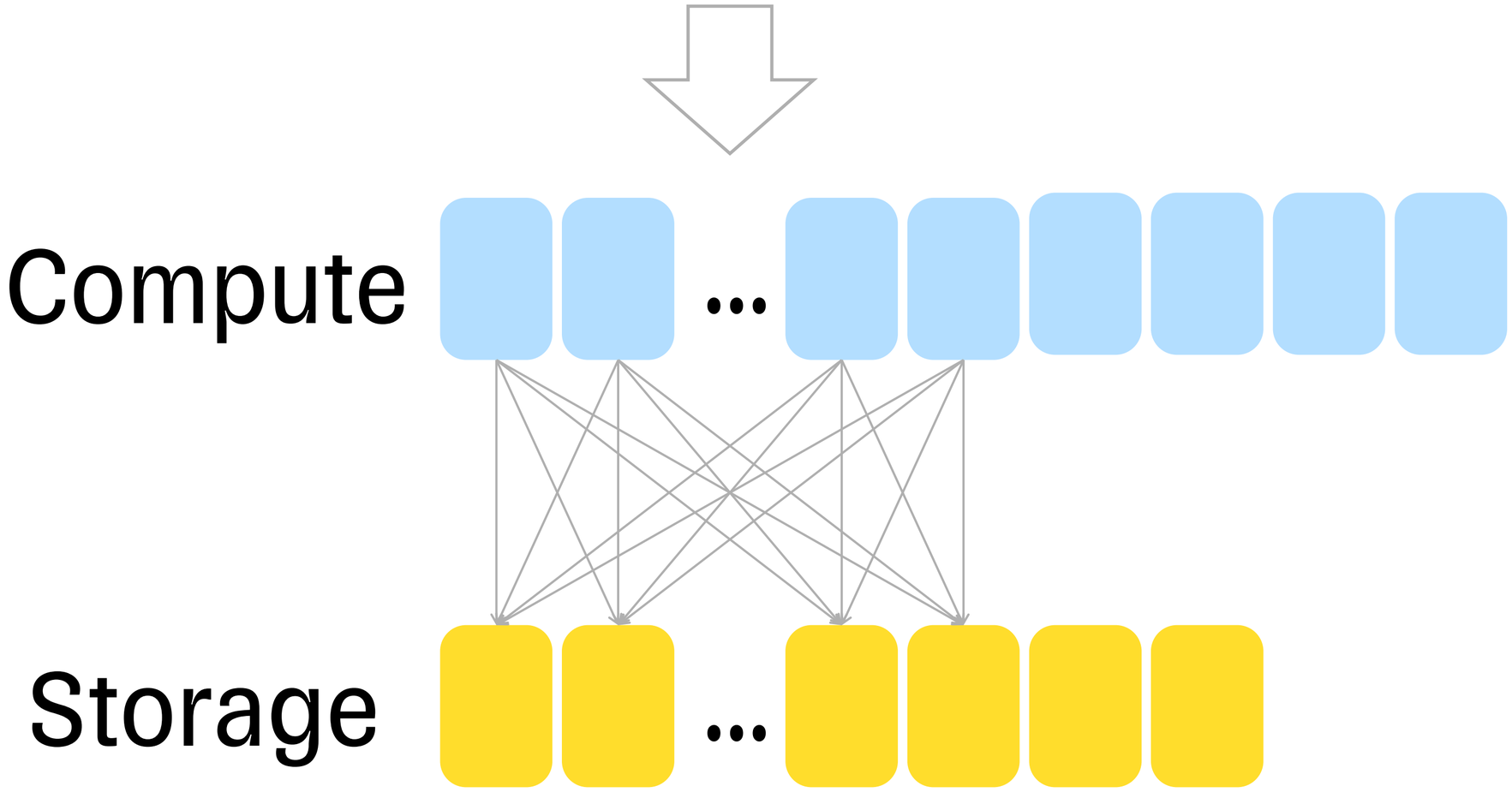


Compute-storage separation

Храним данные
на одних серверах,
обрабатываем
на других

Greenplum так
не умеет

Compute-storage separation



Усложнение последнее

ETL-разработчики

Реальность:

- У нас много данных
- У нас много ETL-процессов
- У нас куча каких-то вспомогательных сервисов

Откуда все это взялось?

- У нас много ETL-разработчиков
- Они постоянно что-то разрабатывают

Где разрабатывать?

Общая dev-среда

- Синхронизируем с продом по расписанию
- Таблицы с данными
- Мощный сервер
- Можно сделать 1:1 прод

- **Конфликты**

Локальная среда

- На машине разработчика
- Таблицы без данных
- Актуализируем сами
- Виртуализация
- Слабое железо
- Всегда отличается от прода

Виртуальные изолированные среды

Создаются под каждую доработку

На dev-серверах

Но полностью изолированы

Занимают мало места

Автоматически наполняются данными

Тесты, тесты, тесты...

Continuous Integration

Continuous Deployment

DWH ETL Tools

Moebius

Consumers

Duet2

Unicorn

Actuality

Cut2

Tedi

Guillotine

Data Detective

Chimera

BODS

SDP

Т БАНК

Спасибо!