Т БАНК

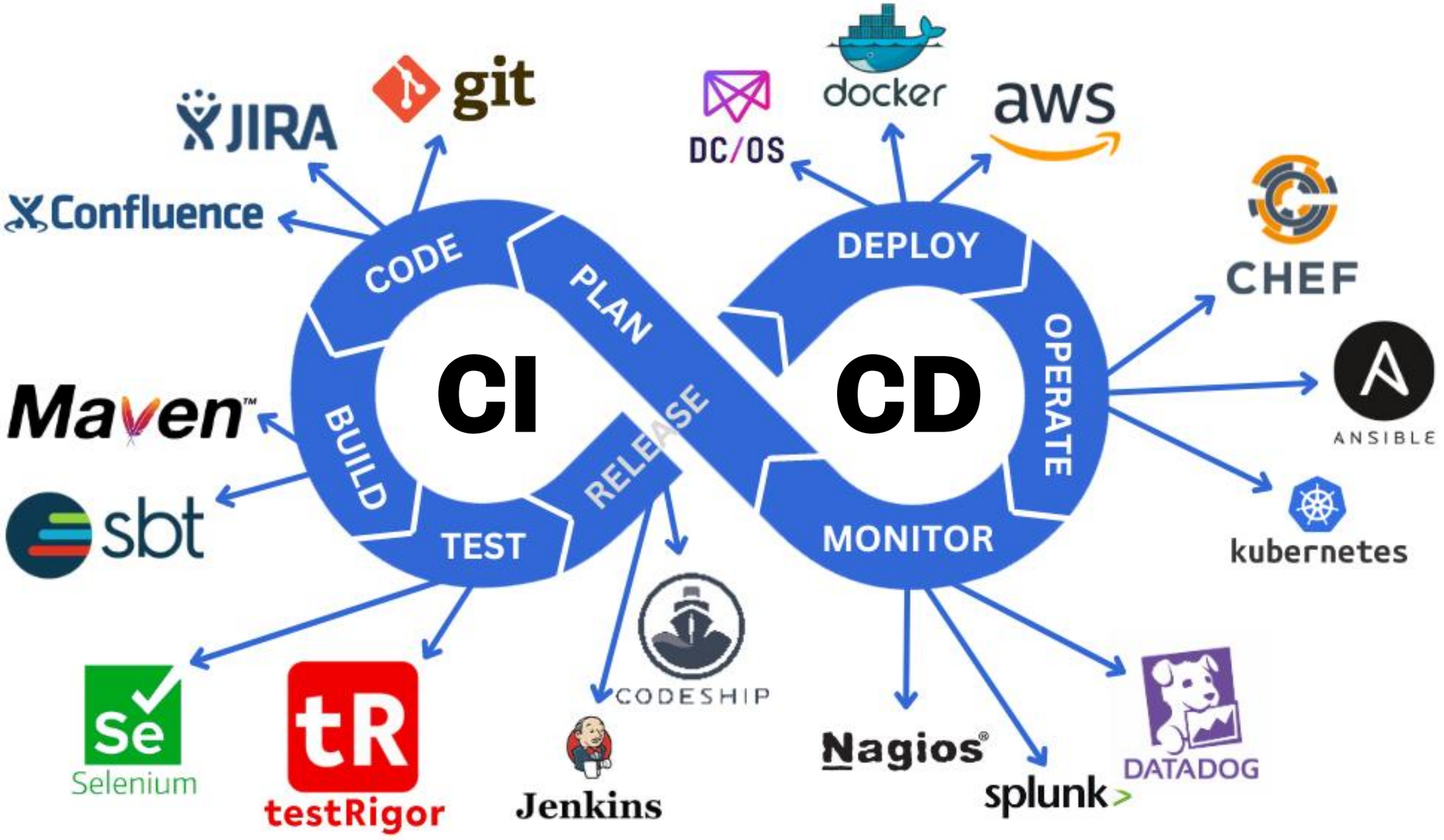
SmartData 2024

CI/CD для большого хранилища данных

Или как автоматизировать все-все-все

Арсений Бирюков

07.09.2024



https://testrigor.com/blog/what-is-cicd/

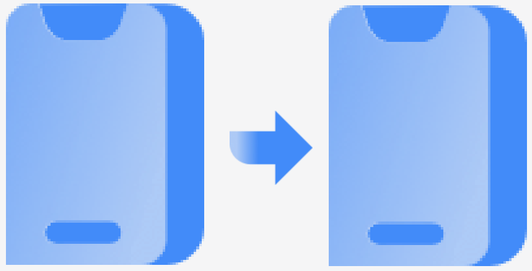
Проблемы, которые мы решили

- Где разрабатывать?
- Как положить в git?
- Как тестировать?
- Как деплоить в прод?

Где разрабатывать?

Конечно же прямо на проде!

На общем dev-сервере



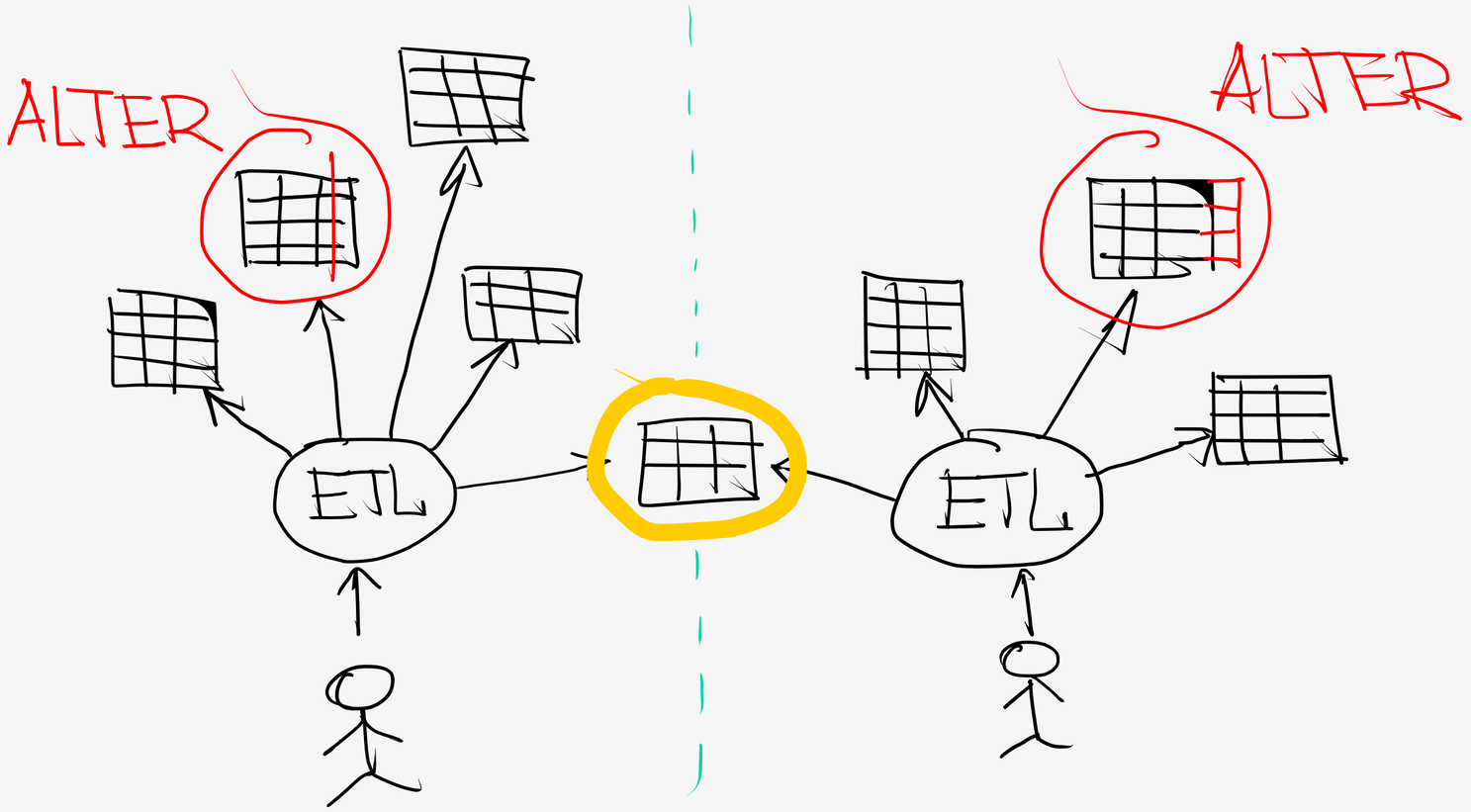
- Сломать прод нельзя
- Сервер, такой же как прод
- В таблицах есть данные

- Обновление по расписанию
- Не понятно как починить сломанное
- Разработчики мешают друг другу

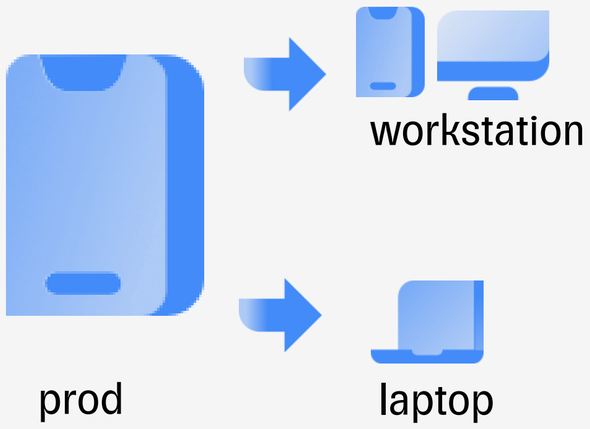
А почему мешают-то?

У каждого своя задача

Разработчики не редактируют одно и то же, но все же мешают друг другу



На персональных «серверах»



- Разработчики не мешают друг-другу
- Разработчик контролирует все

- Таблицы без данных
- Слабое железо
- Создавать все таблицы – долго
- Как отлаживать запросы – не понятно

Наше решение

Общий сервер

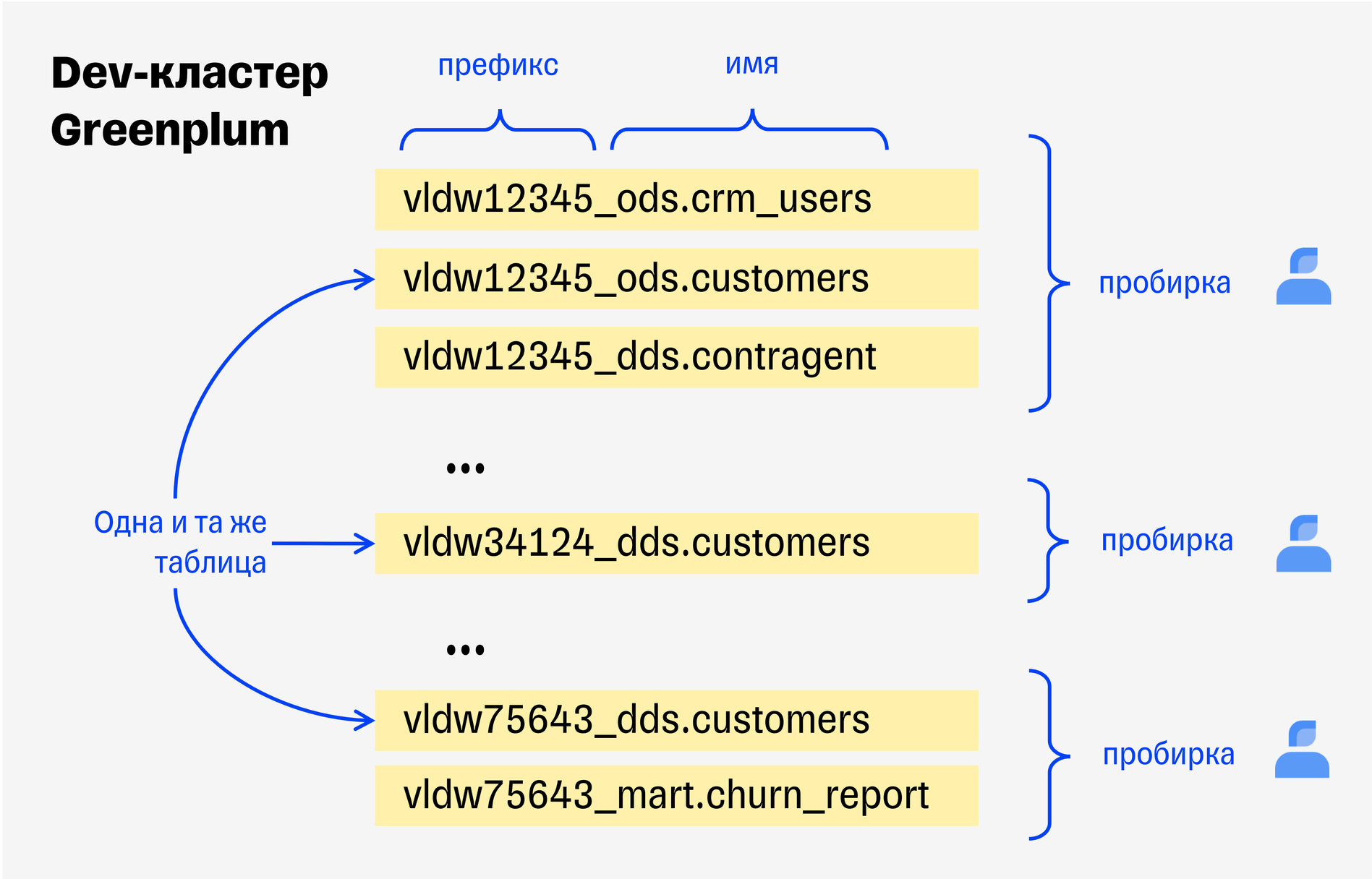
+

Персональные таблицы

- **Сломать прод нельзя**
- **Мощное железо**
- **Таблицы с данными**
- **Разработчики не мешают друг-другу**
- **Можно откатить изменения**

- **Сложно**

Подробности реализации



А что для этого нужно?

UI для выбора таблиц

Разработчики где то должны указывать, что им нужно

Копировать таблицы с прода

Вроде бы просто, но есть нюансы

Переключать префиксы в запросах

Самое сложное, но у нас есть свой ETL-инструмент

Оптимизация копирования

Копировать по сети – долго

Копировать по сети
большую таблицу целиком – очень долго

Копировать по сети
большую таблицу целиком
каждый раз – совсем неприемлемо

Раз в неделю копируем все таблицы на dev-сервер

Копии для пользователей создаем локально

Плюс: данные консистентны

Оптимизация места

Пользователи быстро съели все место

Зачем копировать RO-таблицы?

Создаем VIEW

Как узнать, RO или RW?

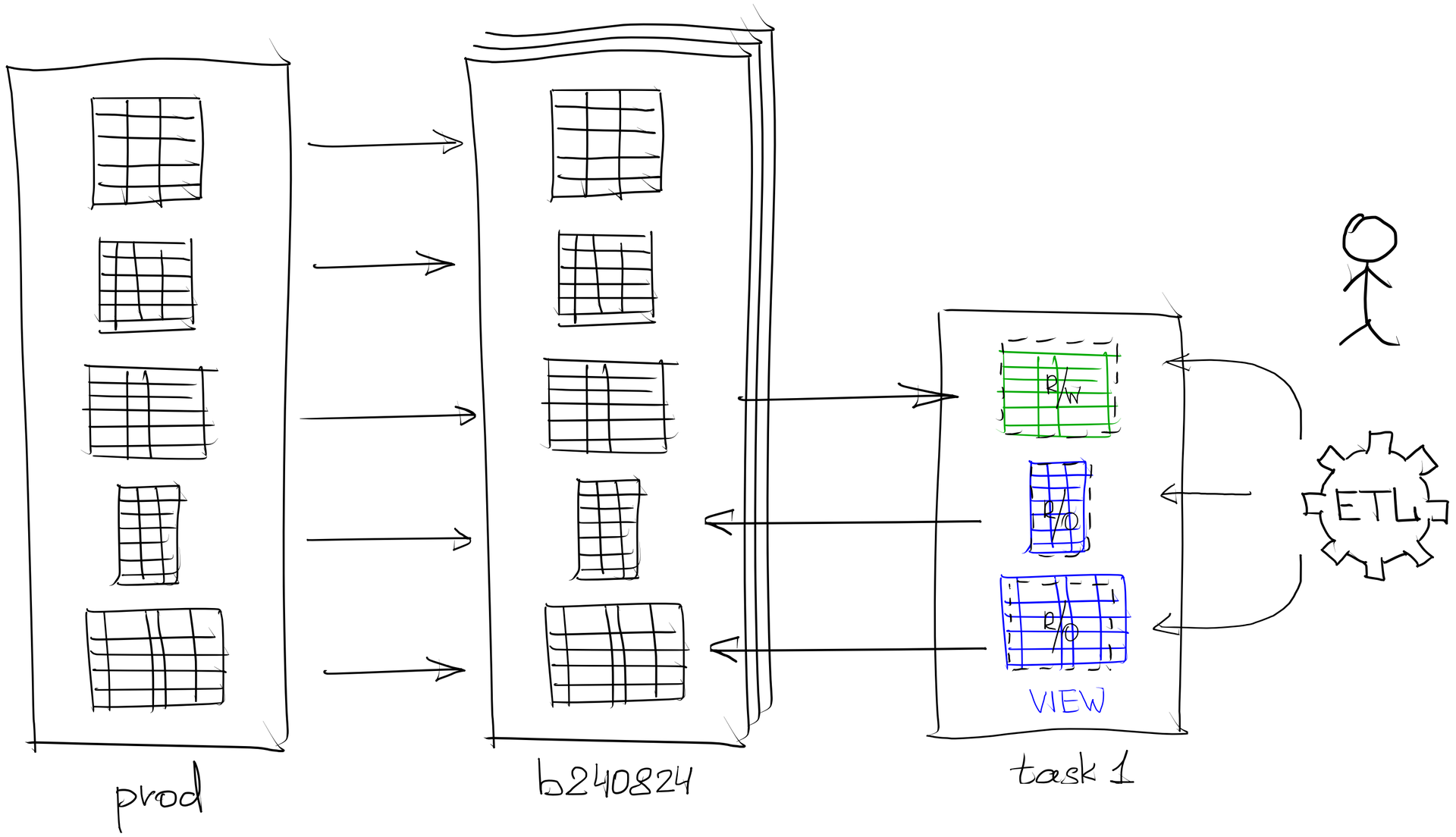
Оптимизация User eXperience

Вручную добавлять много таблиц – неудобно

Указывать для каждой таблицы тип (RO/RW) – еще хуже

Но ведь у нас же ETL-процессы!

Общая картина



Т БАНК

Как положить хранилище в git?

Кладем CREATE TABLE в git

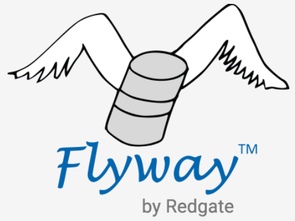
Храним в репозитории только CREATE TABLE (VIEW, FUNCTION, etc)

Нужно что-то изменить – меняем CREATE

- ODS
  - Customers.sql
  - Accounts.sql
  - Transactions.sql
- DV
  - H_Contragent .sql
  - S_Contragent_accounting .sql
  - S_Contragent_CRM.sql
- Marts
  - И т.п.

Не понятно как модифицировать существующую БД

Кладем в git
миграции



Простая последовательность SQL-скриптов:

- V1_create_ODS_core_tables.sql
- V2_add_crm_user_ODS.sql
- V3_add_online_customers.sql
- V4_change_all_names_to_plural.sql
- V5_create_contragents_DDS.sql
- ...
- V2345_refactor_to_data_vault.sql
- ...
- V7542_ensure_transactions_quality_check.sql

Каждая миграция **изменяет** существующую БД

Миграции это круто, но...

- Последовательное выполнение
- Непрозрачность
- Непонятно, как откатить обратно
- Только СУБД

А из чего состоит хранилище?

Сами данные

Нужно создавать и изменять таблицы, мигрировать данные в новые таблицы, переименовывать их

ETL-процессы

Создание новых ETL процессов и изменение существующих.
Оптимизация производительности
Перезаливка данных.

Вспомогательные сервисы

Нужно настраивать различные вспомогательные системы, например для управления жизненным циклом данных

Наше решение

Пакет

=

миграция на стероидах

Содержимое пакета

- Метаданные
- Сценарий релиза
- Артефакты
- Результаты тестов

Это просто папка, мы кладем ее в git

Метаданные пакета

Автор/владелец

Описание

Список изменяемых объектов

- Таблицы
- ETL-процессы

Поиск

Аппрувы

Создание пробирок

Преодолеваем недостатки миграций

Сценарий релиза

- Сделай бэкап
- Переименуй таблицу
- Выполни SQL-скрипт
- Сделай merge
- Скопируй файл
- Вызови REST-сервис

Миграции внутри миграций

В каждом пакете сценарий разный

Откуда берется сценарий?

Метаданные

Изменения в пробирках

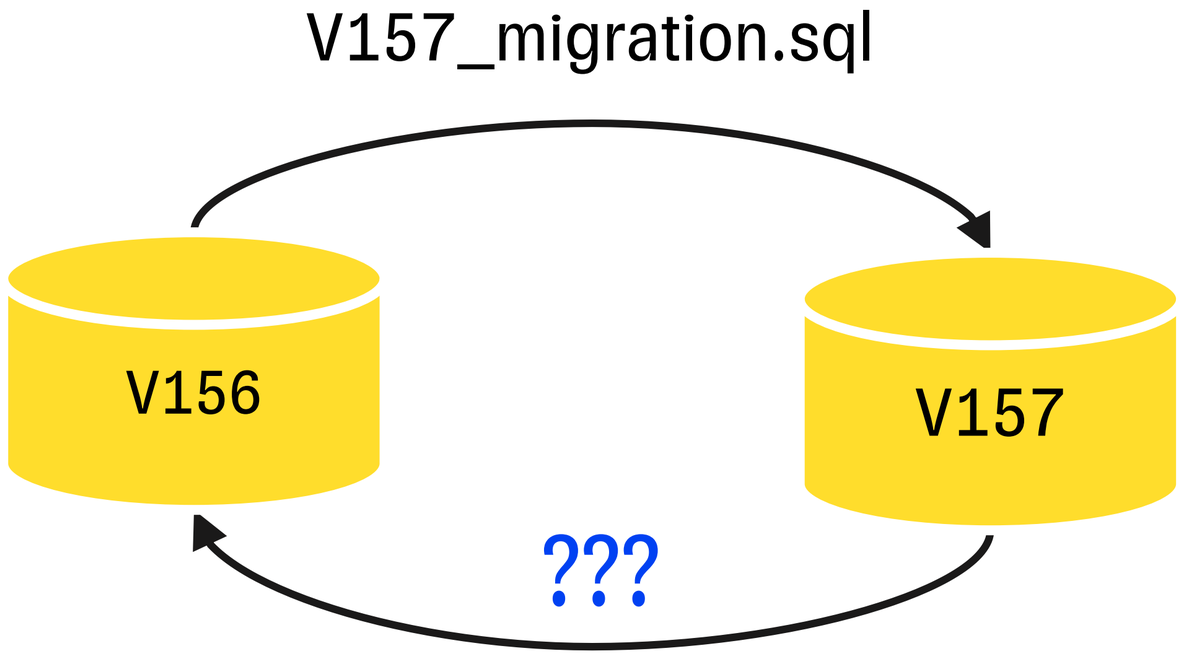
Артефакты

Автоматически созданный сценарий

Упрощаем жизнь ETL-разработчиков

Умные шаги

Обычные миграции «не знают» как откатить изменения



Наши шаги запоминают, что нужно сделать, чтобы вернуться назад

Готовимся к дерелизу

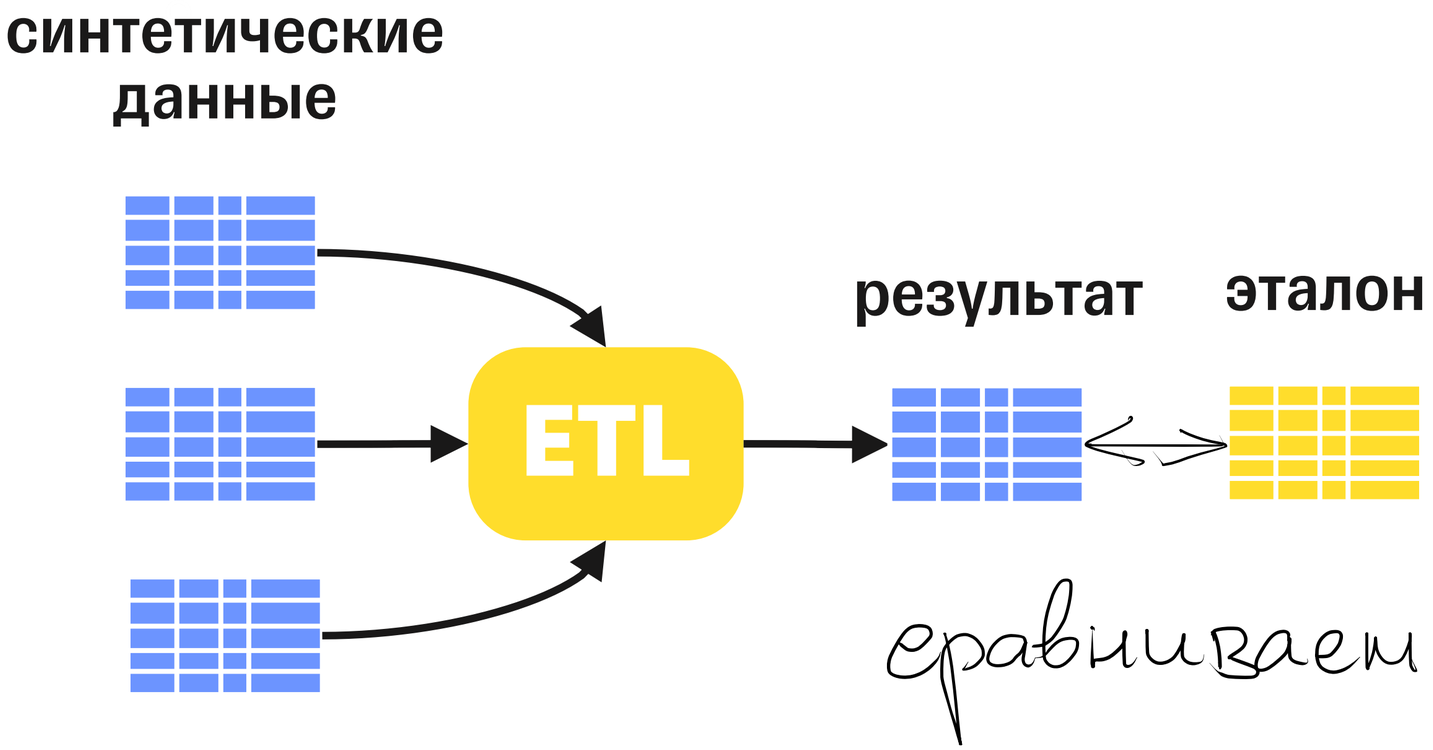
Наш CI процесс

Т БАНК

Протестируем?

Идея:

UNIT-тесты



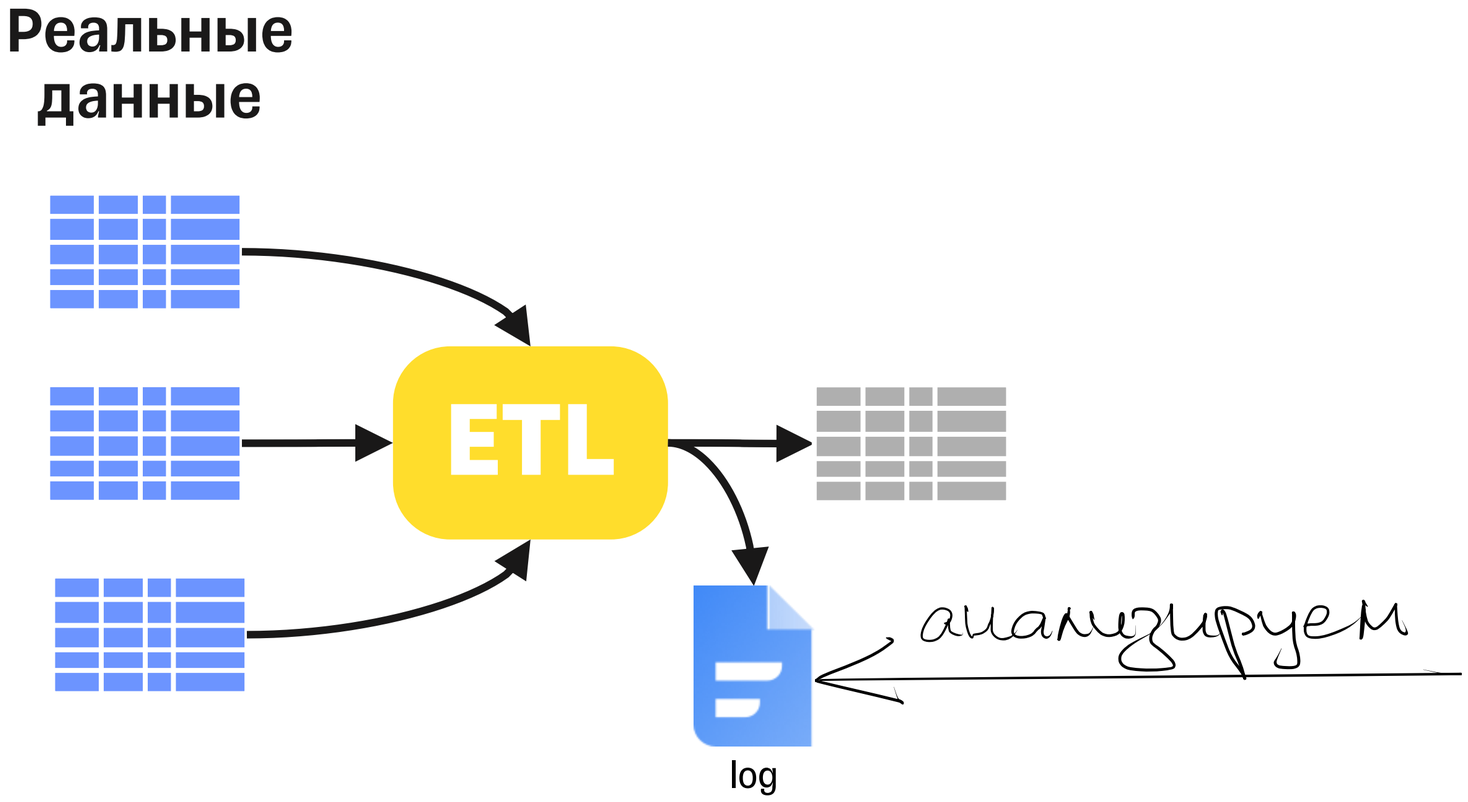
Факт: это не работает

Статический анализ

Можно проверять пакет на соответствие правилам

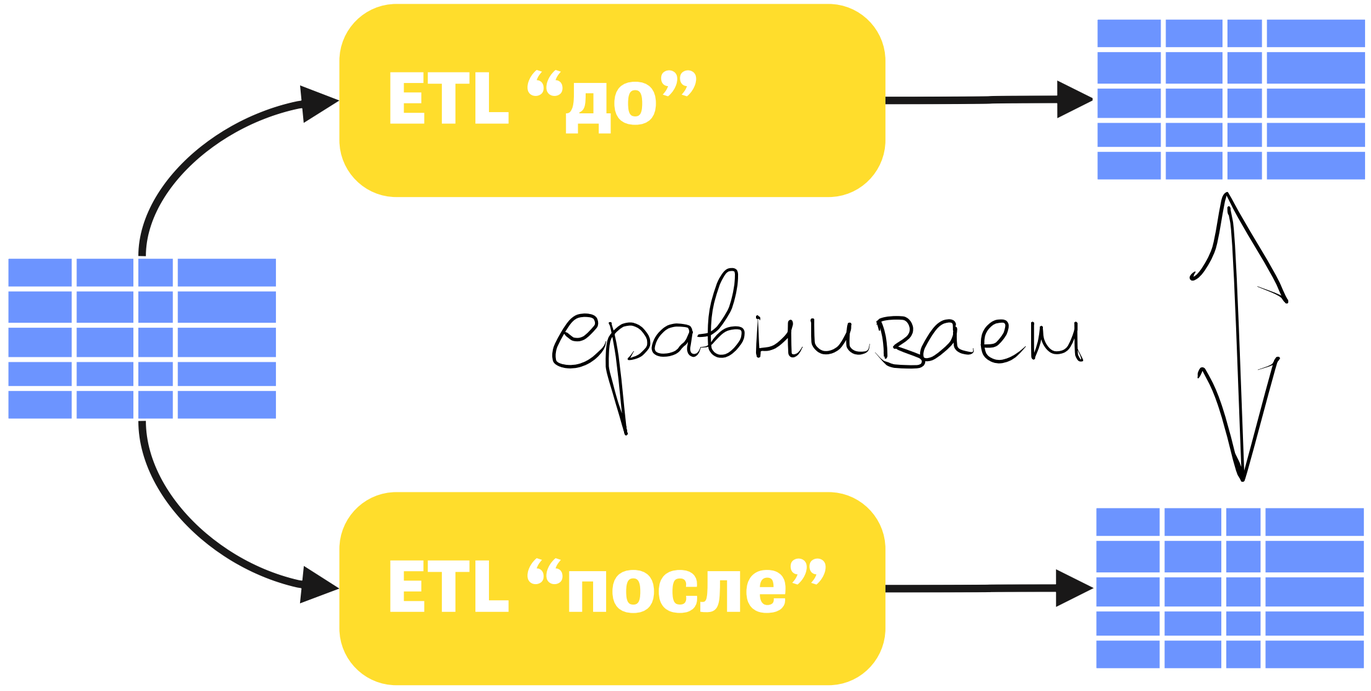
- **SQL-скрипт:**
  Имена таблиц соответствуют шаблону
- **ETL-процесс**
  Предикаты в запросах ищут по индексу
- **Метаданные**
  Не пытаемся переименовывать новые таблицы

Анализ логов



Планы запросов, перекос данных, спилы и т.п.

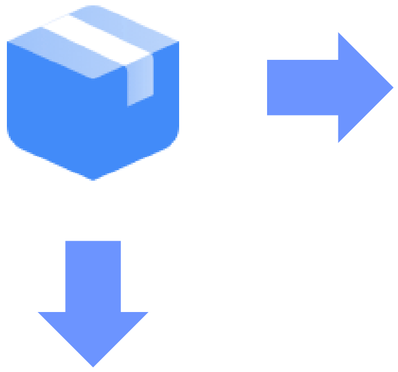
Регресс-тестирование



Профилируем результаты

Тестовый релиз

Пакет



Dev ETL-сервер

Load_dds_contragent

Dev-кластер Greenplum

vldw12345_ods.crm_users

vldw12345_ods.customers

vldw12345_dds.contragent

Свежие таблицы с прода

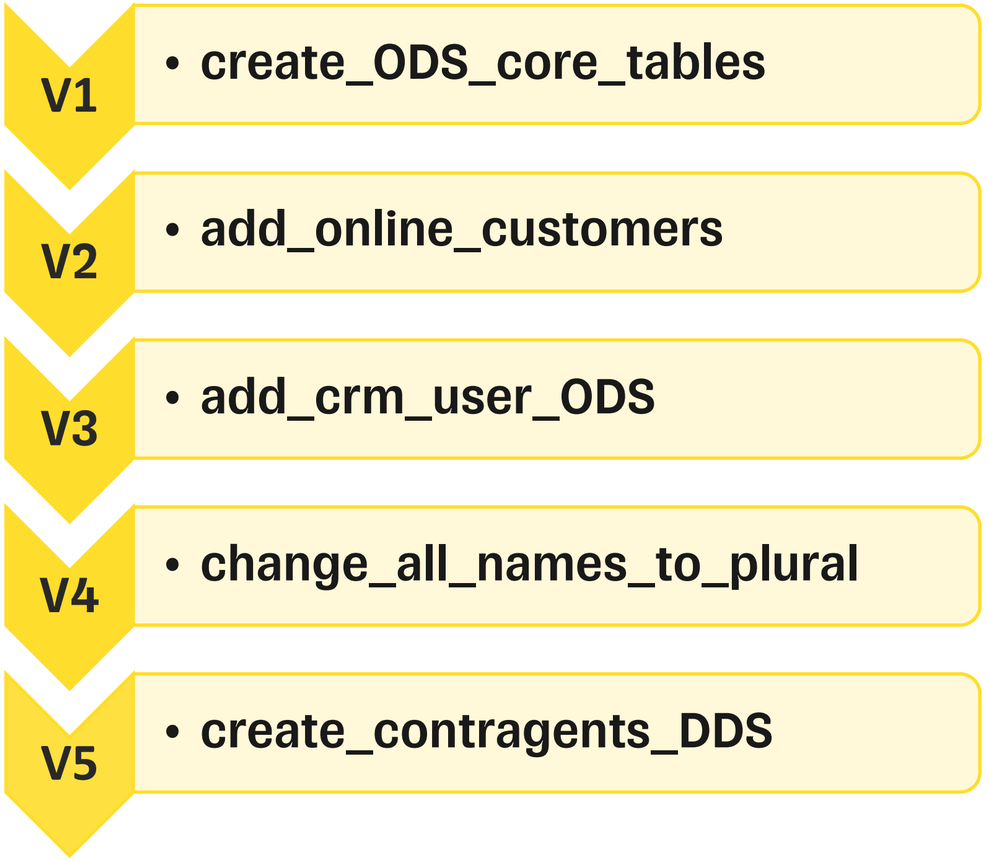
Ключ к успеху - автоматизация

- Тесты запускаются автоматически
- Результаты кладутся в Allure
- Релизить без тестов нельзя

Т БАНК

Пора релизить?

Накат миграций на базу данных - flyway

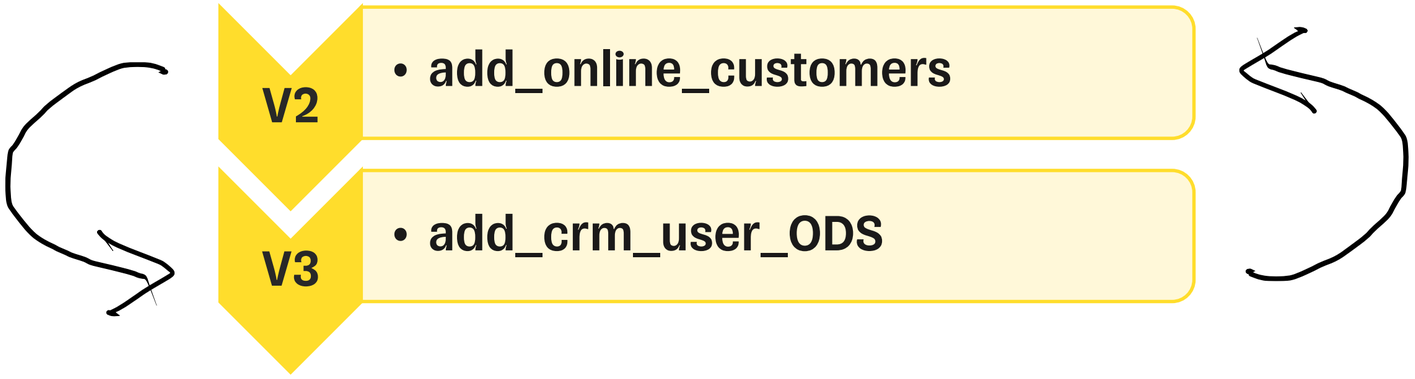


Migration_name	dttm
create_ODS_core_tables	2024-07-04
add_online_customers	2024-07-09
add_crm_user_ODS	2024-07-15

Нужно выполнить

Недостатки

Откуда берется линейный порядок?



А что если кто-то внес изменения вручную?

- Таблица ods.crm_user уже создана

Можно ли накатыватать пакет на прод?

Не упадет ли релиз на прод?

Тестовый релиз

- На тест раскатилось?
- Ну наверное и на проде получится!

Изменялись ли объекты?

Кто-то менял таблицы или ETL?

- Нет? Релизим!

Сделаем тестовый релиз

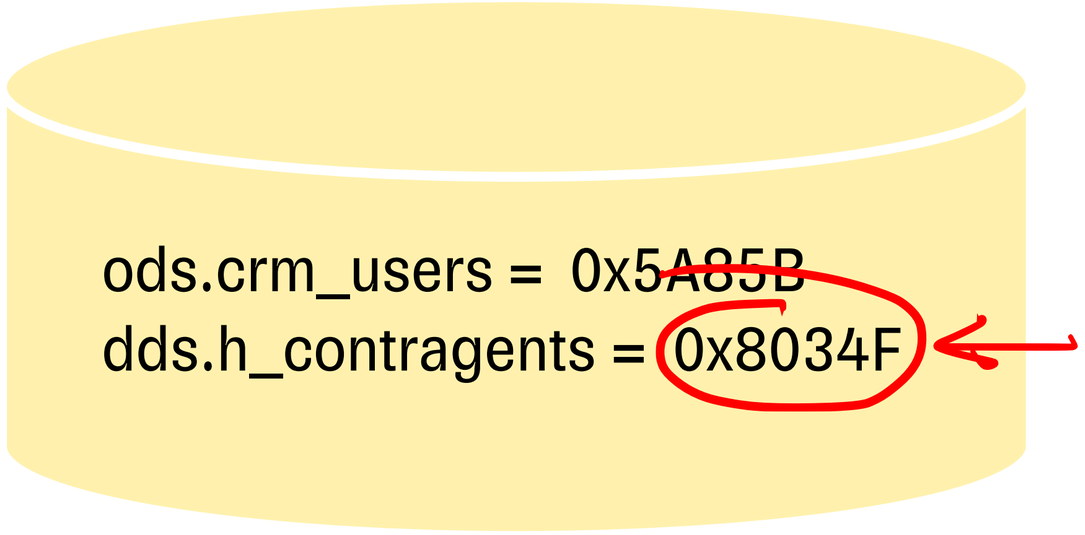
- Создадим еще один набор пробирок
- Положим туда самые-самые свежие данные
- Выполним сценарий пакета на этой пробирке

- В пробирке можно сделать не все

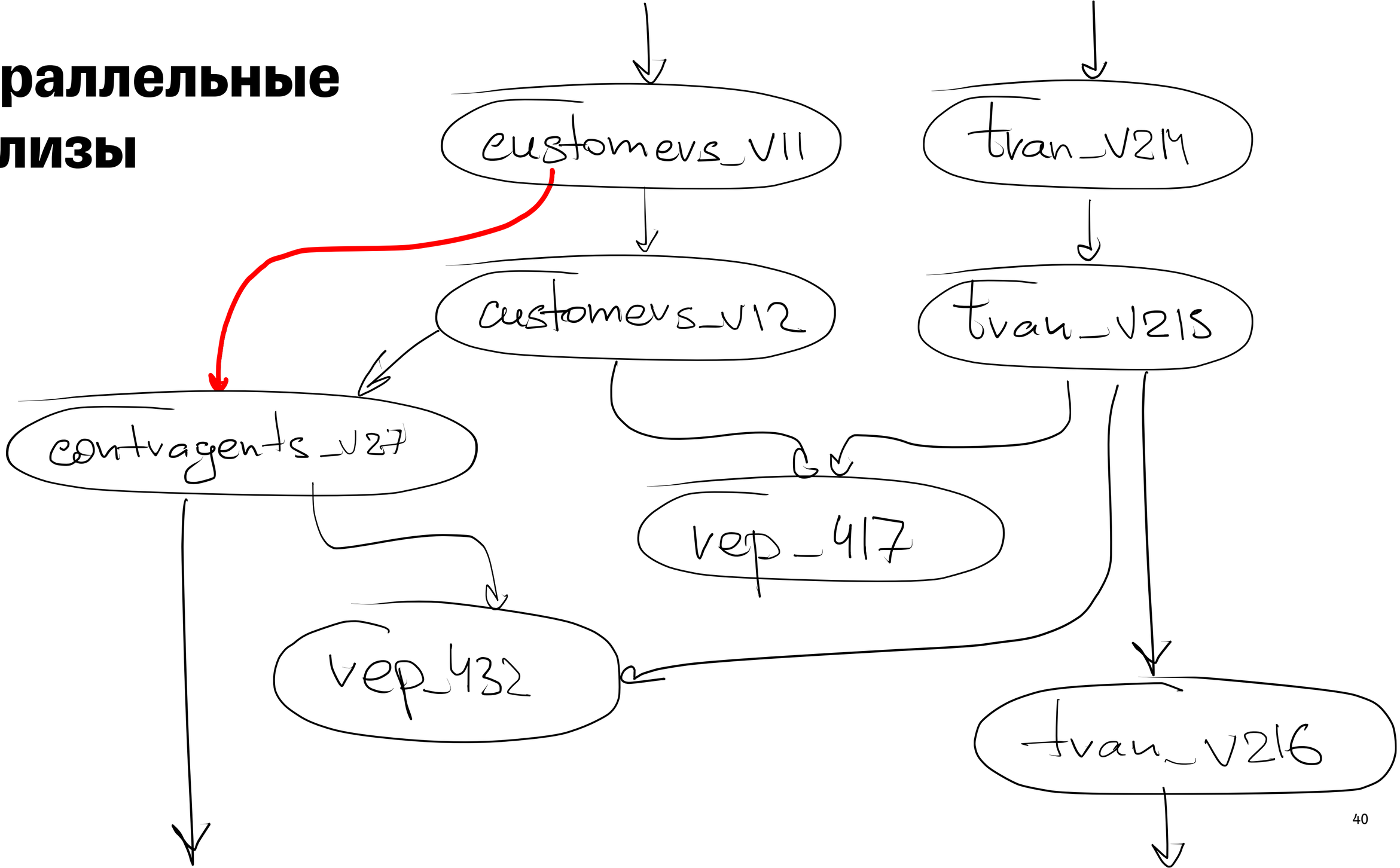
По версиям объектов

Charge_contragents_package

before	name	after
0x5A85B	ods.crm_users	0x456AF
0x123BA	dds.h_contragents	0x0F63A
0xF43A0	Load_h_contragents	0x45299



Параллельные релизы



Т БАНК

Спасибо!