Архитектура и инструменты избранных случаев интеграций по данным (часть 1)

Владимир Красильщик
JUG Ru Group

НИИ

RD Software

Continental

DaimlerChrysler

EXIGEN services

T-Mobile

LUXOFT

Deutsche Bank

Яндекс Маркет

DINS

CSC

IBS Training Center

epam

SANDOZ A Novartis Division

КОРУС КОНСАЛТИНГ

gm getmentor.dev

JUG.RU group

>h

Правило успеха 15/85

15% hard skills

85% soft skills

Правило успеха 15/85

15% разработка компонента

85% межкомпонентное взаимодействие

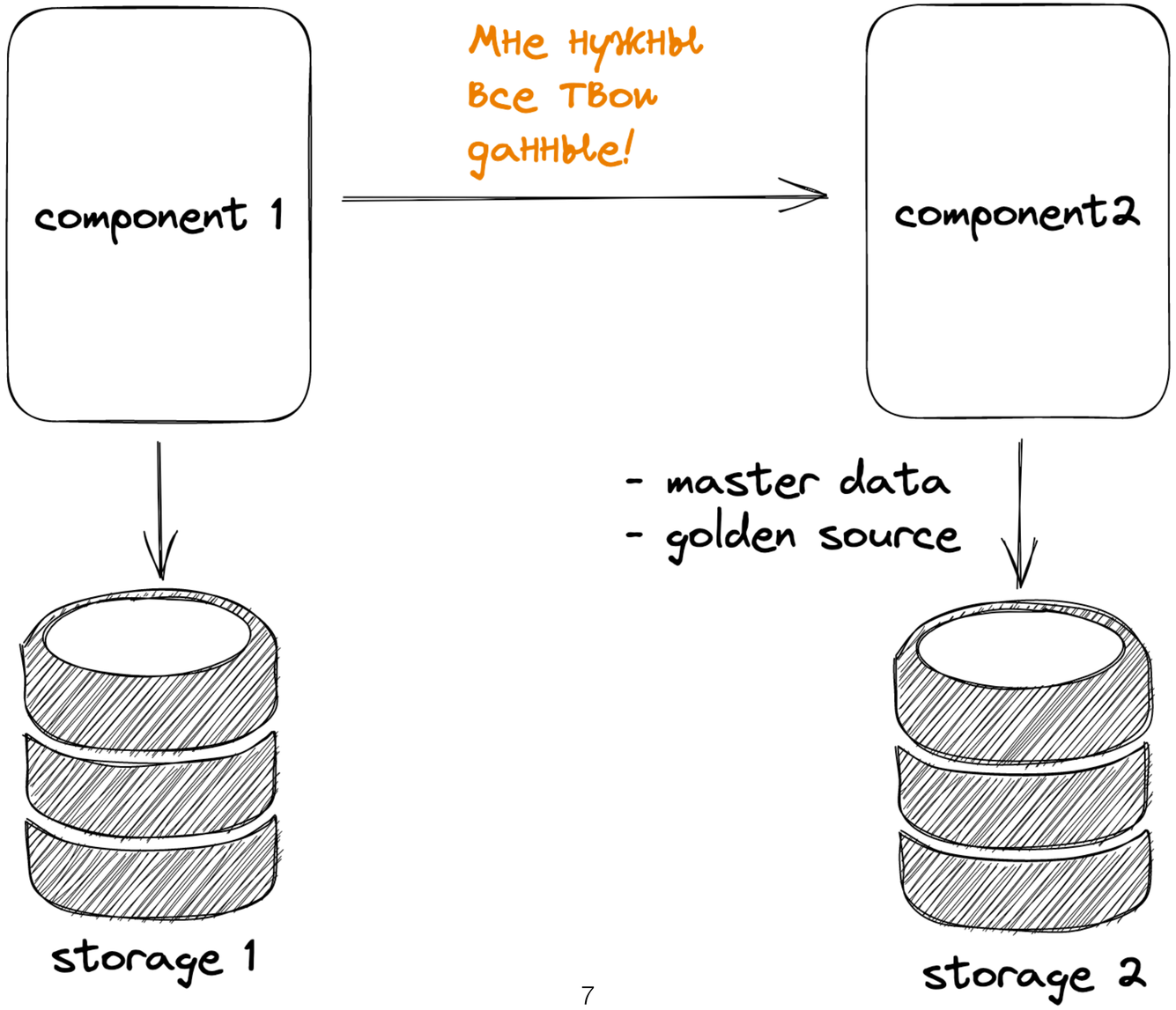
Что будет в докладе

- **Интеграция по данным** - один из видов межкомпонентного взаимодействия
- Разбор реальных кейзов интеграции по данным
- Протоколы и инструменты для интеграций по данным
- Диаграммы кейзов, sql и java код

Чего <u>не</u> будет в докладе

- Детальное погружение в инструменты
- Демо и лайвкодинг

Интеграция по данным



Интеграция по данным: примеры

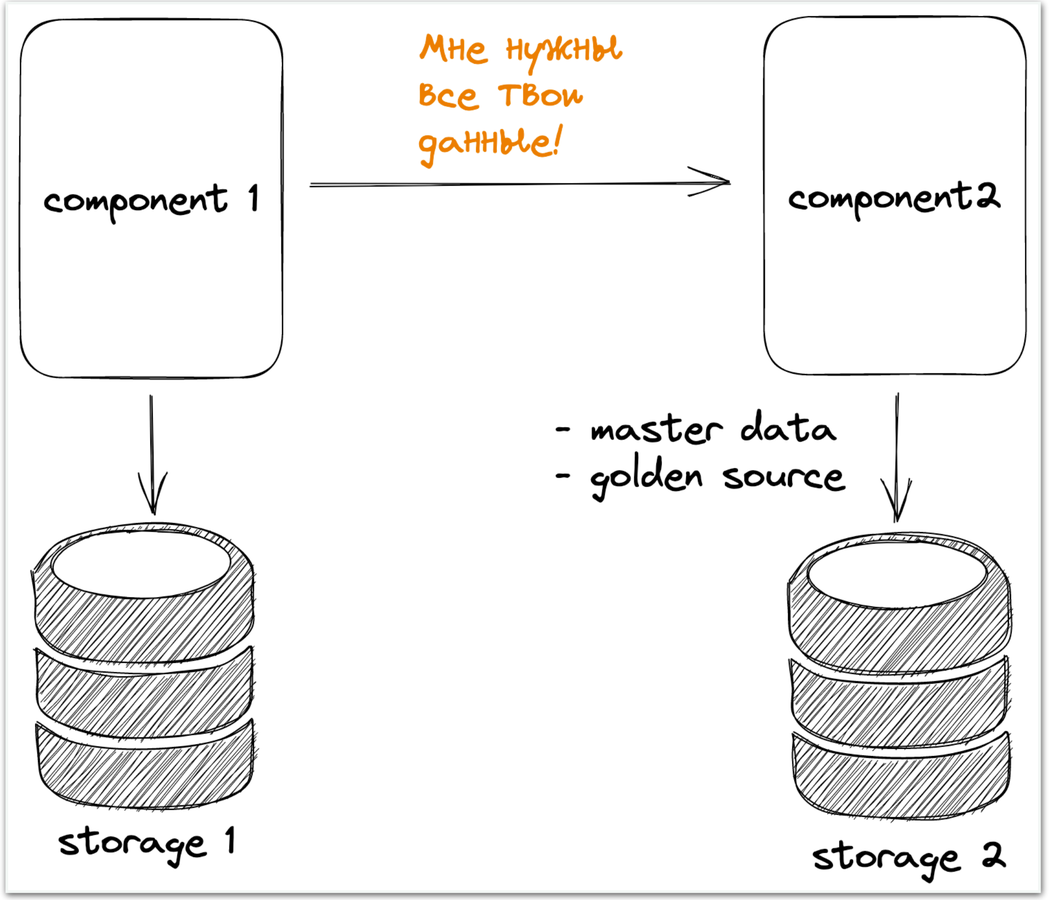
1. Мне нужны все билеты купленные Васей Пупкиным через билетного агента до даты X (это не интеграция по данным)

2. Мне нужны все билеты проданные когда-либо через билетного агента A с максимальным временем актуализации состояния 1 час (это интеграция по данным)

3. Мне постоянно нужны все актуальные данные всех поставщиков товаров на торговой площадке B (это интеграция по данным)

4. Мне прямо сейчас нужны самые актуальные данные поставщика “Рога и Копыта” (это не интеграция по данным)

Интеграция по данным



- Со стороны золотого источника:
  - выставить API для initial-load данных
  - выставить API для получения обновлений

- Со стороны потребителя данных:
  - уметь загрузить первичные данные
  - уметь подгружать обновления

Данные: характер

- Условно статичные, вероятность изменения за сутки невысокая, например:
  - контактная и платежная информация контрагентов
  - номенклатура продуктового склада

VS

- Часто меняющиеся, например:
  - биржевые котировки
  - моментальная точная статистика мировых продажи электротехники

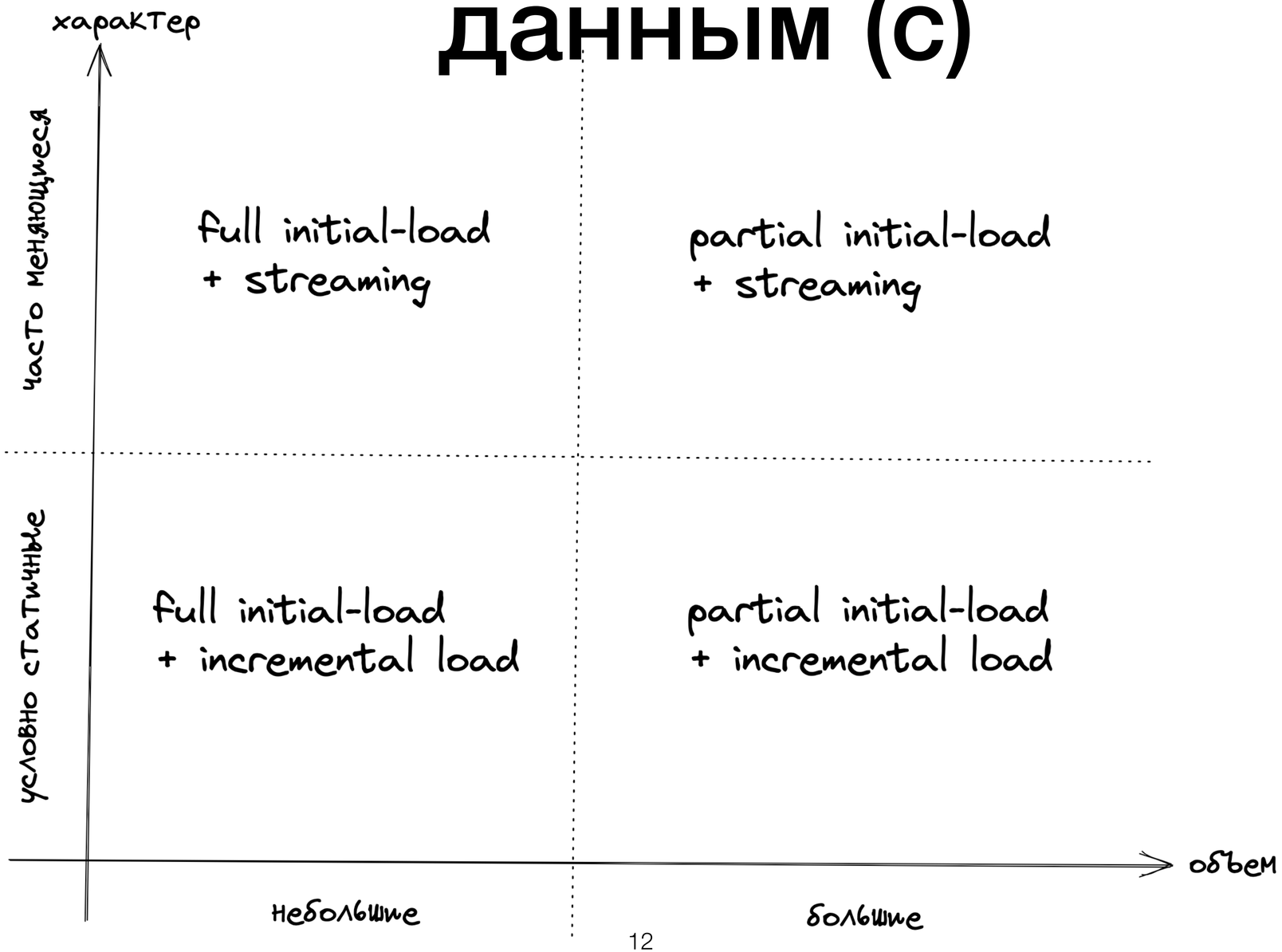
Данные: объем

- Небольшие, сотни тысяч строк и менее, например:
  - паспортные данные учителей начальных школ
  - каталог автомобильных комплектующих регионального производителя

VS

- Большие, например:
  - клики, переходы по ссылкам рекламных кампаний
  - мировые продажи глобальной фармы

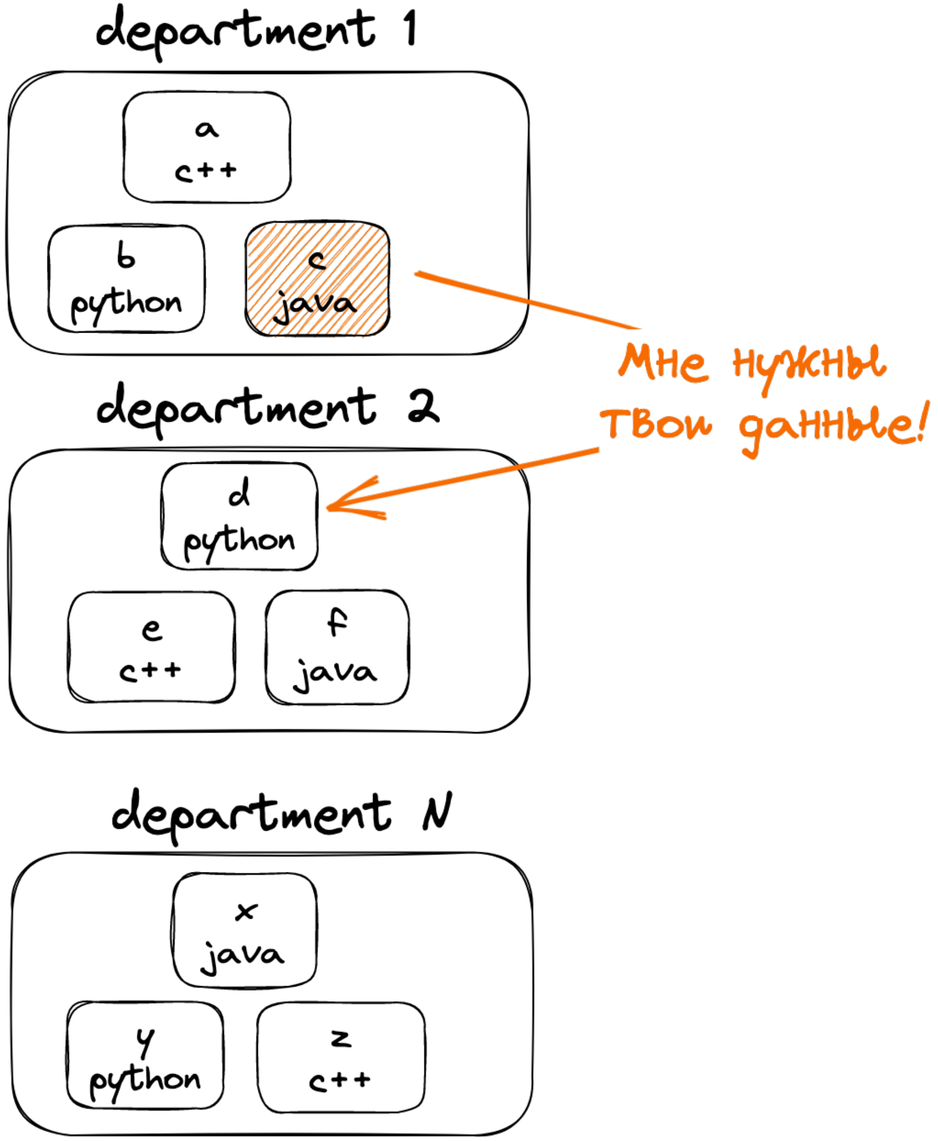
Матрица интеграций по данным (c)



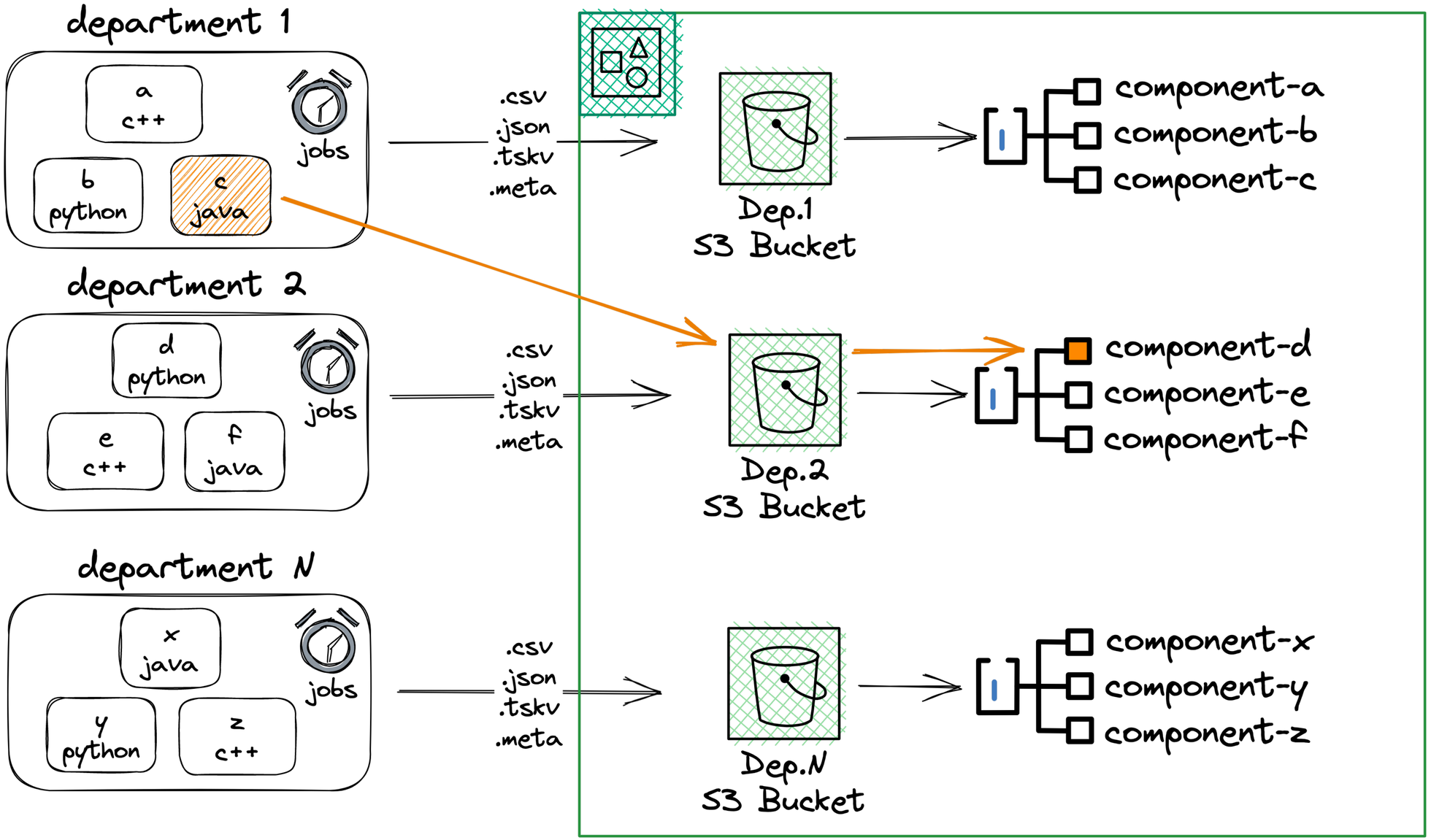
Кейз №1

Яндекс Маркет

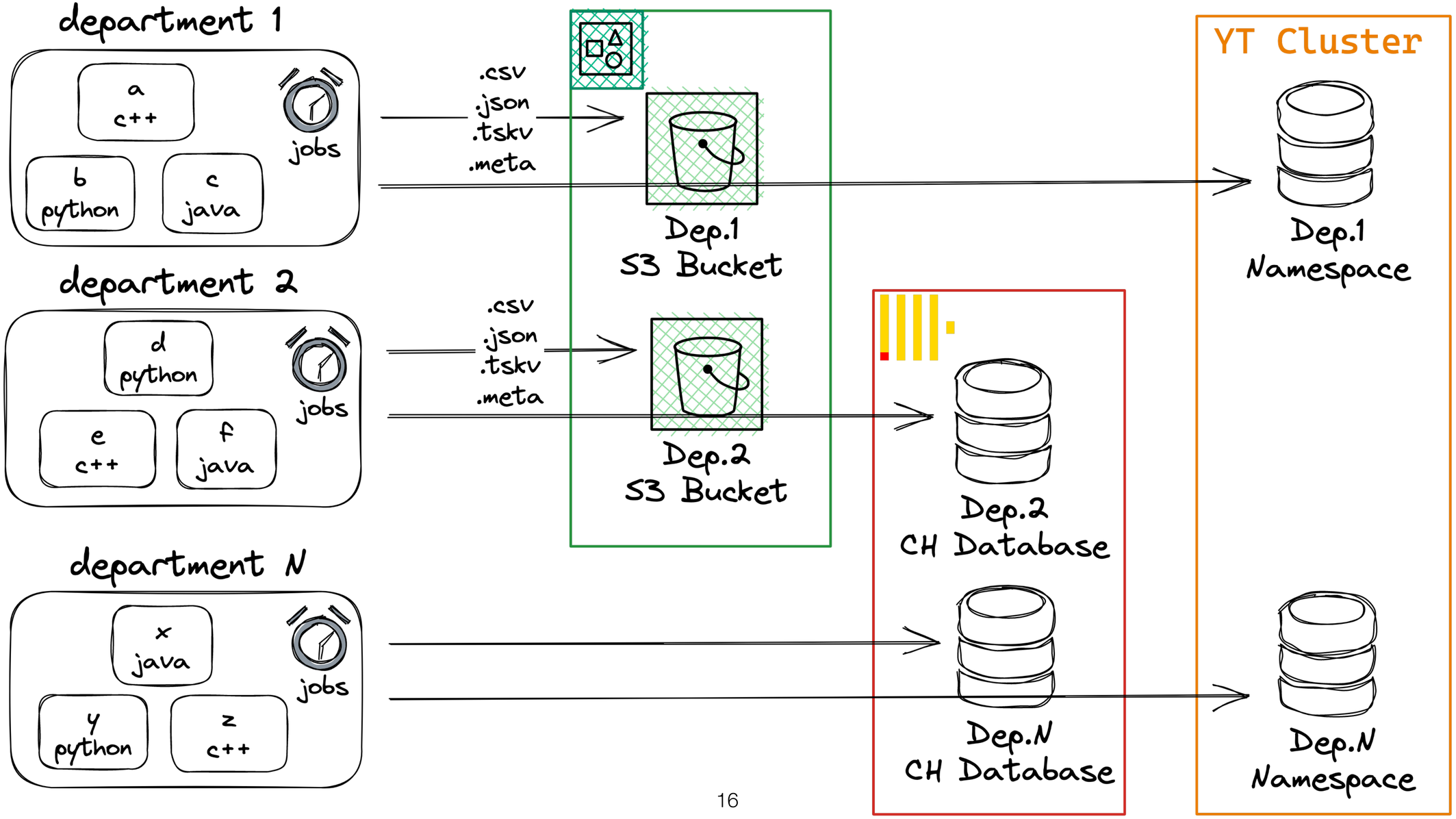
Кейз №1


department 1
a
c++
b
python
c
java
Мне нужны
твои данные!
department 2
d
python
e
c++
f
java
department N
x
java
y
python
z
c++

Кейз №1


department 1
a
c++
b
python
c
java
jobs
.csv
.json
.tskv
.meta
Dep.1
S3 Bucket
component-a
component-b
component-c
department 2
d
python
e
c++
f
java
jobs
.csv
.json
.tskv
.meta
Dep.2
S3 Bucket
component-d
component-e
component-f
department N
x
java
y
python
z
c++
jobs
.csv
.json
.tskv
.meta
Dep.N
S3 Bucket
component-x
component-y
component-z

Кейз №1


department 1
a
c++
b
python
c
java
jobs
department 2
d
python
e
c++
f
java
jobs
department N
x
java
y
python
z
c++
jobs
.csv
.json
.tskv
.meta
Dep.1
S3 Bucket
.csv
.json
.tskv
.meta
Dep.2
S3 Bucket
Dep.2
CH Database
Dep.N
CH Database
YT Cluster
Dep.1
Namespace
Dep.N
Namespace

Почему бы просто не ходить за интеграционными данными в “золотой источник” по-требованию?

1. Интеграционные данные нужны нам в нашей базе для джойнов с нашими “родными” данными

2. Отказоустойчивость

3. Аудит изменений интеграционных данных

Аудит изменений интеграционных данных

- в 98% кейзов нам просто нужны самые последние версии данных

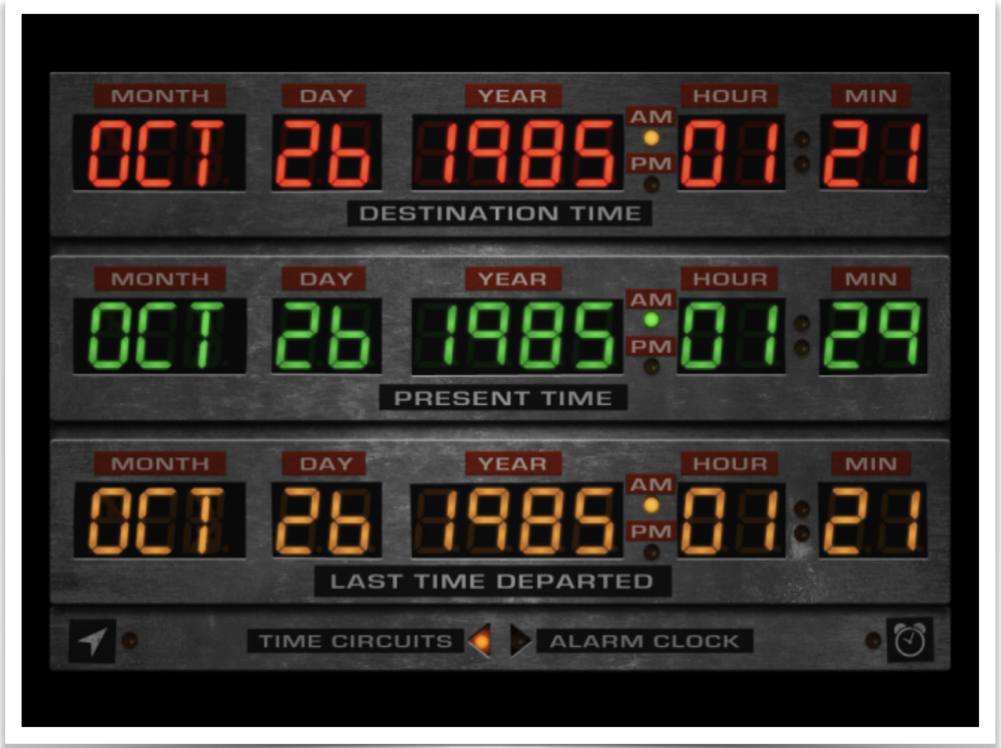
- в 1% кейзов важно фиксировать критические изменения величины для разбора полетов и фиксов, например:
  - кто-то вставил html в поле CMS и теперь не собирается сайт конференции

- в 1% кейзов по бизнесу важно фиксировать все изменения величины во времени, например:
  - расчет стоимости покупки в кредит валютного опциона в рублях на момент подачи заявки по моментальным значениям кредитной ставки и валютной пары

Аудит изменений интеграционных данных

https://youtu.be/FkUqLZtIP1U

Аудит изменений интеграционных данных



Назад в будущее современной банковской системы

https://youtu.be/FkUqLZtlP1U

Аудит изменений на уровне БД

Аудит изменений на уровне БД: основная таблица

```
create table lk_user
(
    id            bigint       not null
        constraint pk_user primary key,
    lang          varchar(255),
    password_hash varchar(255),
    status        int2         not null default 0,
    comment       varchar(255),
    created_date  timestamp    not null default now(),
    created_by    varchar(255) not null,
    modified_date timestamp    not null default now(),
    modified_by   varchar(255) not null
);
```

Аудит изменений на уровне БД: аудитная таблица

```
create table lk_user_audit
(
    operation      char(1) not null,
    id             int8    not null,
    lang           varchar(255),
    password_hash  varchar(255),
    status         int2,
    comment        varchar(255),
    created_date   timestamp,
    created_by     varchar(255),
    modified_date  timestamp,
    modified_by    varchar(255)
);
```

Аудит изменений на уровне БД: триггеры

```
CREATE TRIGGER lk_user_audit_trigger
    AFTER INSERT OR UPDATE OR DELETE
    ON lk_user
    FOR EACH ROW
EXECUTE PROCEDURE process_lk_user_audit();
```

Аудит изменений на уровне БД: триггеры

```
CREATE OR REPLACE FUNCTION process_lk_user_audit() RETURNS TRIGGER AS
$lk_user_audit_trigger$
BEGIN
    IF (TG_OP = 'DELETE') THEN
        INSERT INTO lk_user_audit SELECT 'D', OLD.*; -- TODO add date
        RETURN OLD;
    ELSIF (TG_OP = 'UPDATE') THEN
        INSERT INTO lk_user_audit SELECT 'U', NEW.*;
        RETURN NEW;
    ELSIF (TG_OP = 'INSERT') THEN
        INSERT INTO lk_user_audit SELECT 'I', NEW.*;
        RETURN NEW;
    END IF;
    RETURN NULL;
END;
$lk_user_audit_trigger$ LANGUAGE plpgsql;
```

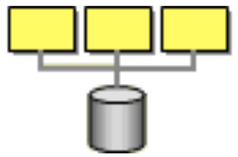
Чеклист компонента

Яндекс Маркет

1. Таблицы для интеграционных данных
2. Таблицы для аудита интеграционных данных
3. Триггеры для аудита интеграционных данных
4. Периодические джобы для выгрузок своих данных
5. Периодические джобы для загрузки интеграционных данных
6. В кластерном варианте предохранитель от множественного параллельного выполнения джобы загрузки интеграционных данных

Получается дорого!
Что если нет времени
на такую красоту?

Интеграция на общей БД по jdbc



Shared Database

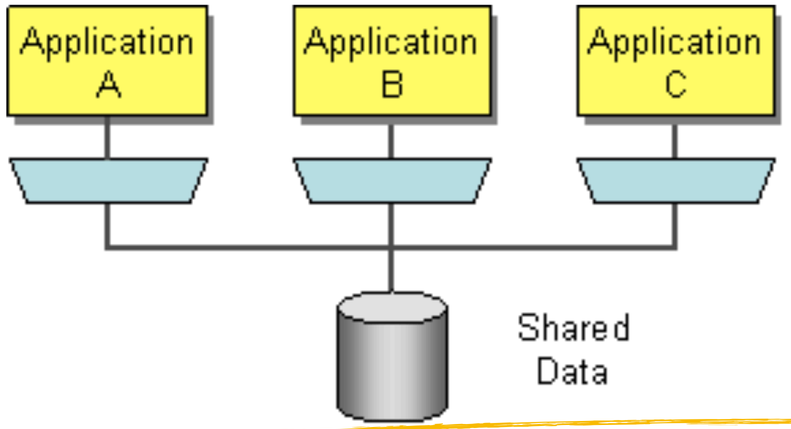
Messaging Patterns » Integration Styles » Shared Database

Messaging Patterns

Previous Next

An enterprise has multiple applications that are being built independently, with different languages and platforms. The enterprise needs information to be shared rapidly and consistently.

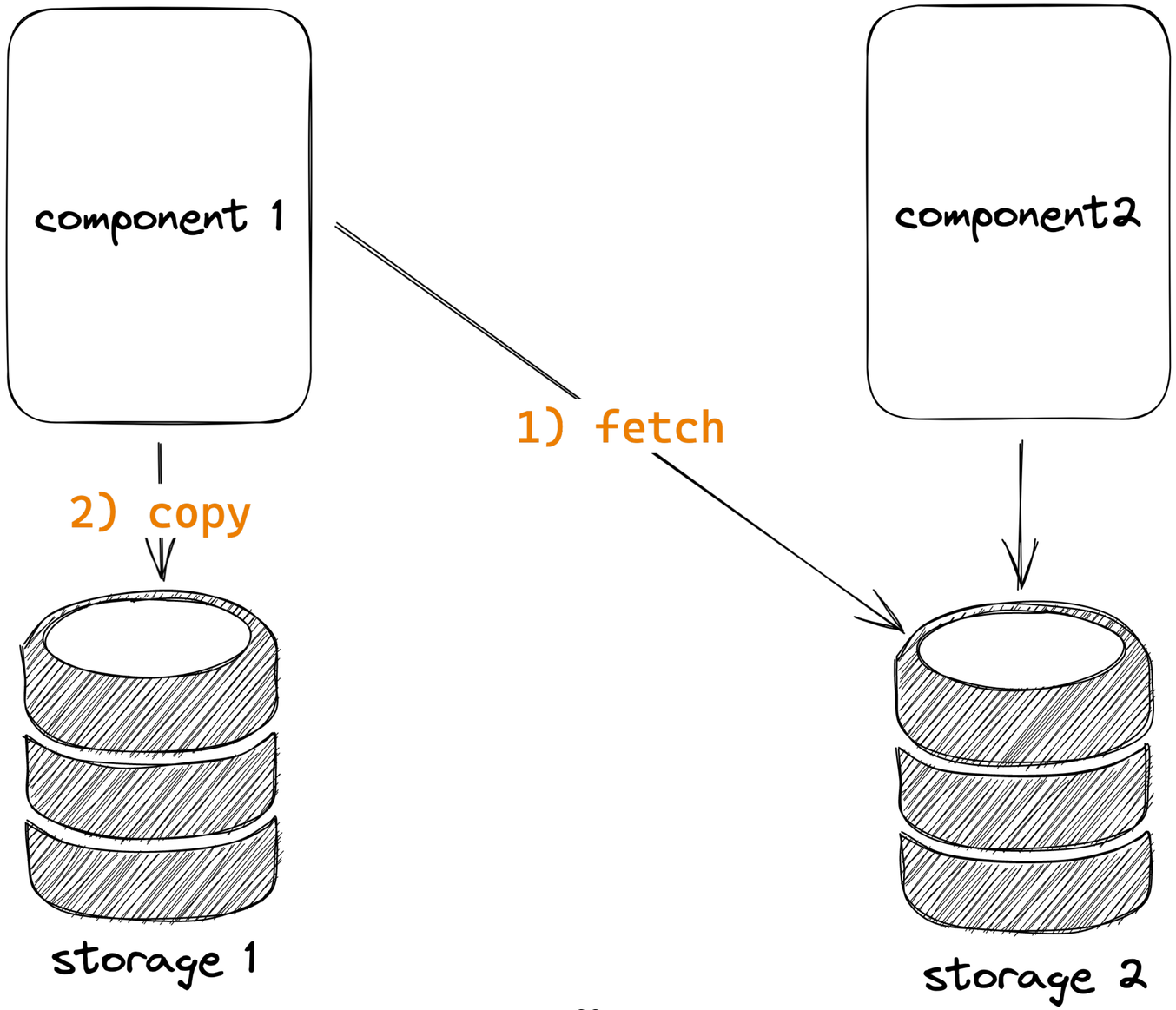
How can I integrate multiple applications so that they work together and can exchange information?



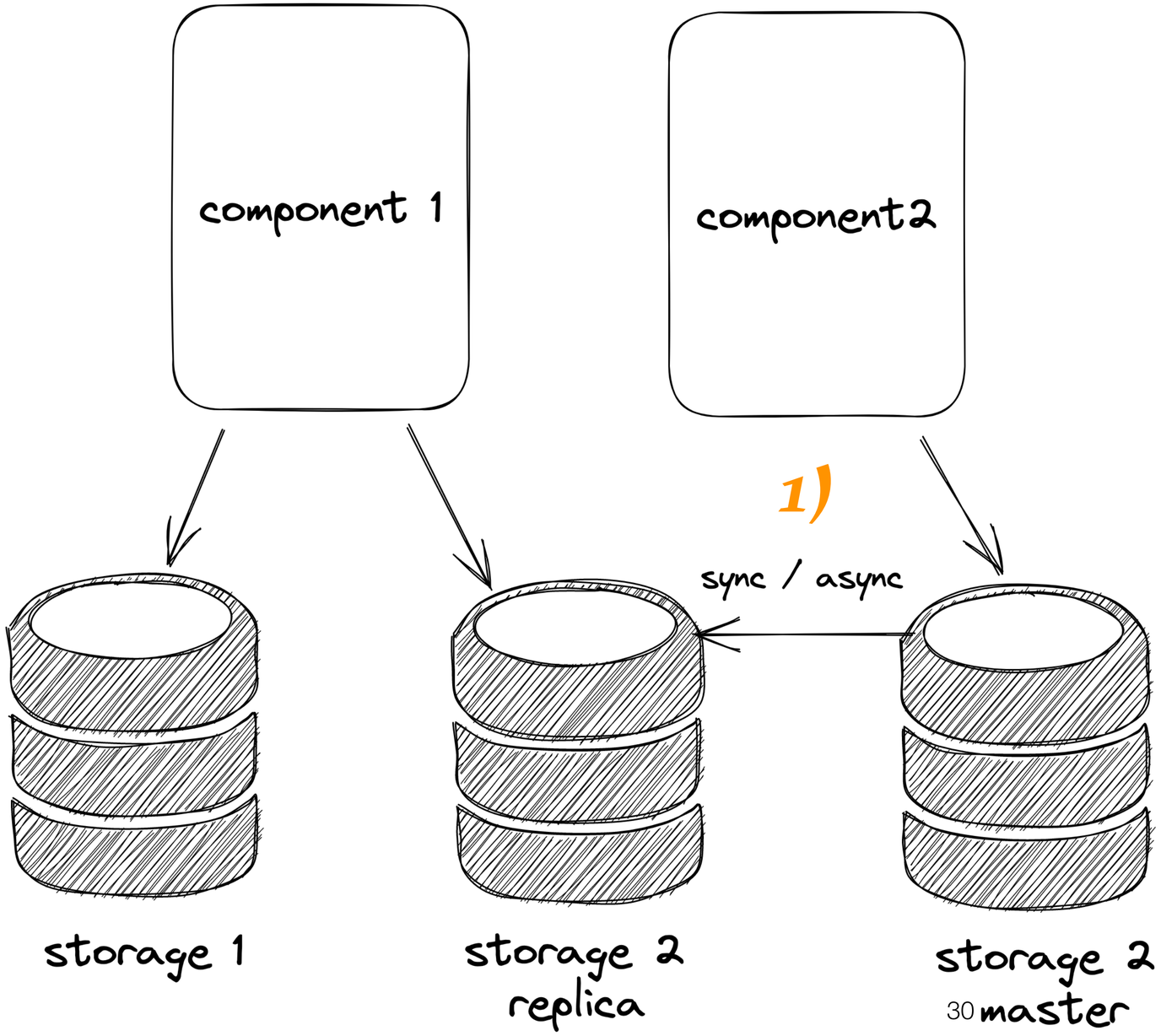
Integrate applications by having them store their data in a single *Shared Database*.

https://www.enterpriseintegrationpatterns.com/SharedDataBaseIntegration.html

Интеграция на общей БД по jdbc



Интеграция на общей БД по jdbc



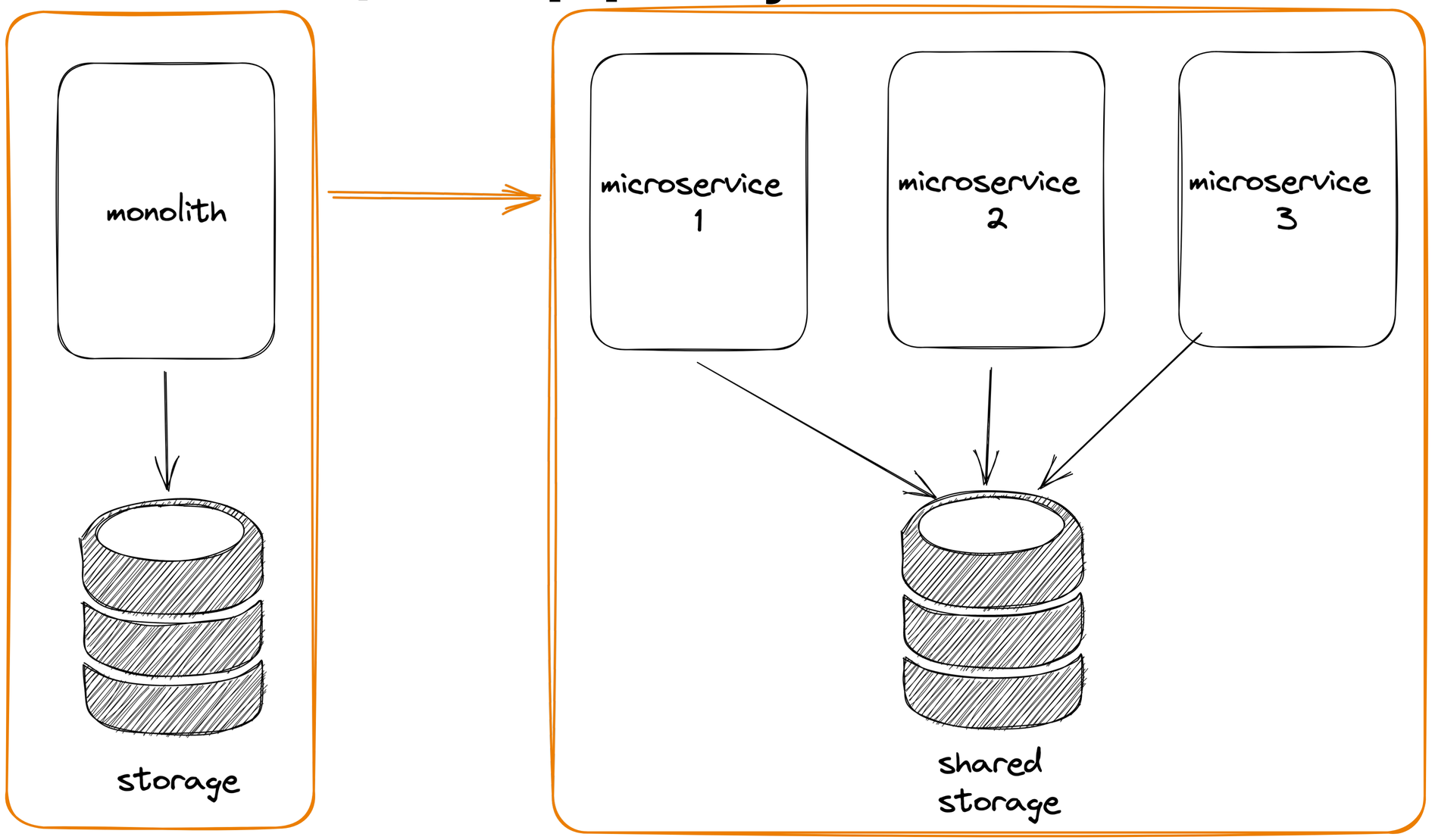
2)

```
-| db
---| datamart_schema
------| view_1
------| view_2
… …
------| view_N
```

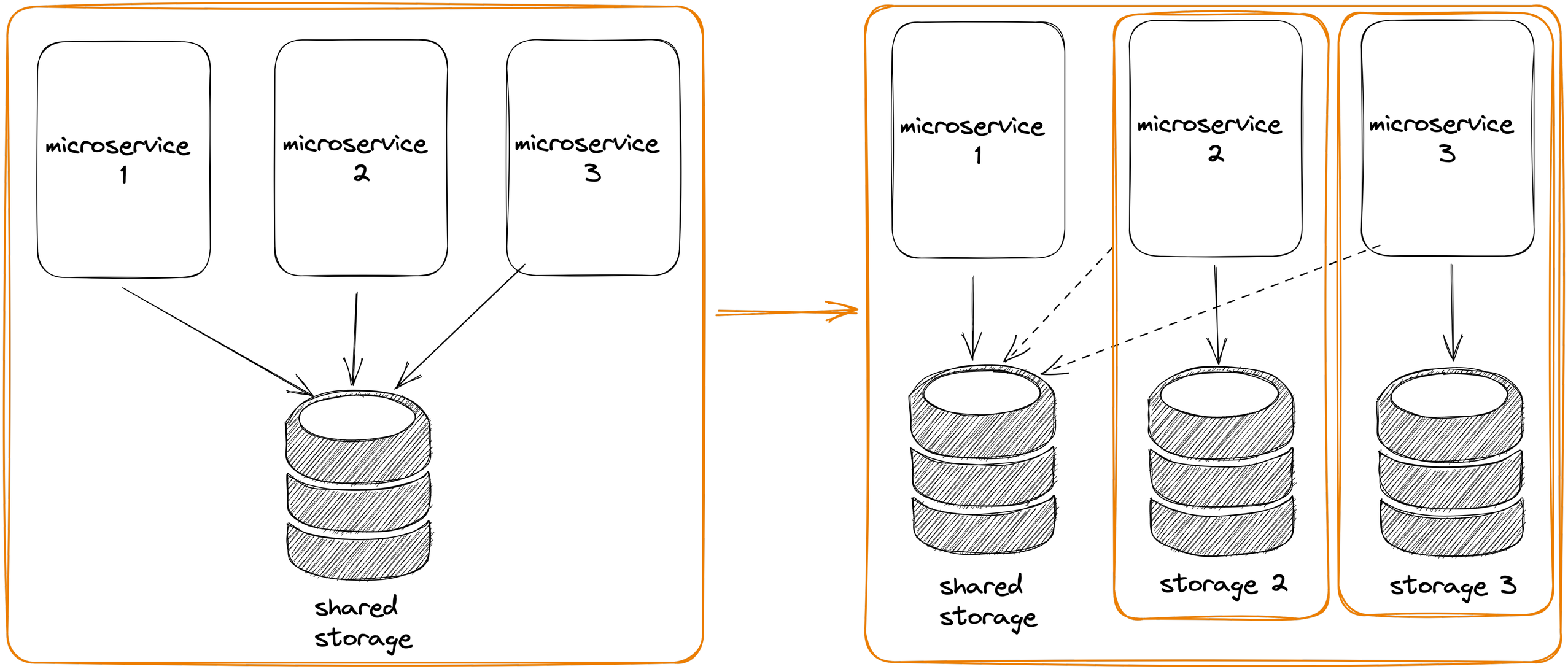
Интеграция на общей БД по jdbc

```
create view datamart_schema.sales_kz_restricted_view as
select s.id                          as sale_id,
       s.amount                      as sale_amount,
       s.item_id                     as sale_item,
       hashing_function(s.buyer)     as buyer,
       s.sum                         as sale_sum
from main_schema.sales s
where s.region = 'KZ';

grant select on datamart_schema.sales_kz_restricted_view to
analytic_kz_restricted;
```

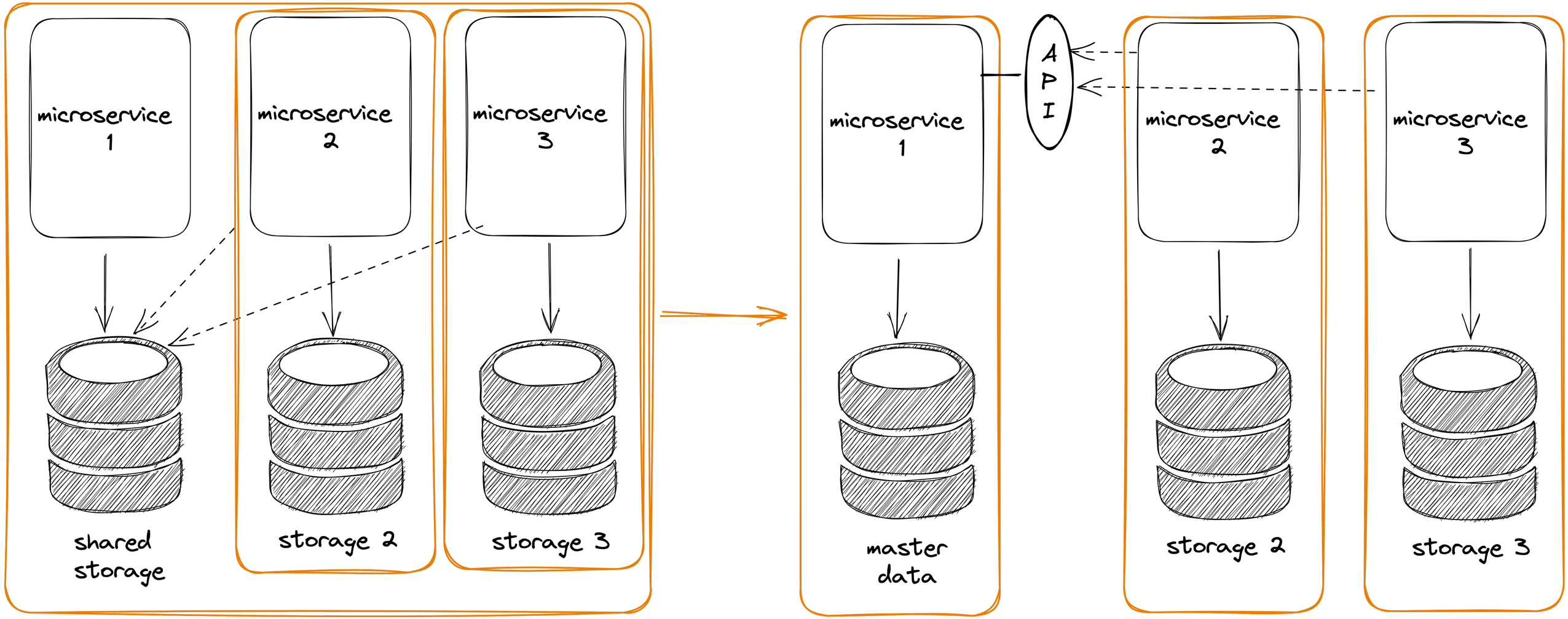
Вынужденная интеграция на общей БД по jdbc: MSA



Вынужденная интеграция на общей БД по jdbc: MSA



Вынужденная интеграция на общей БД по jdbc: MSA



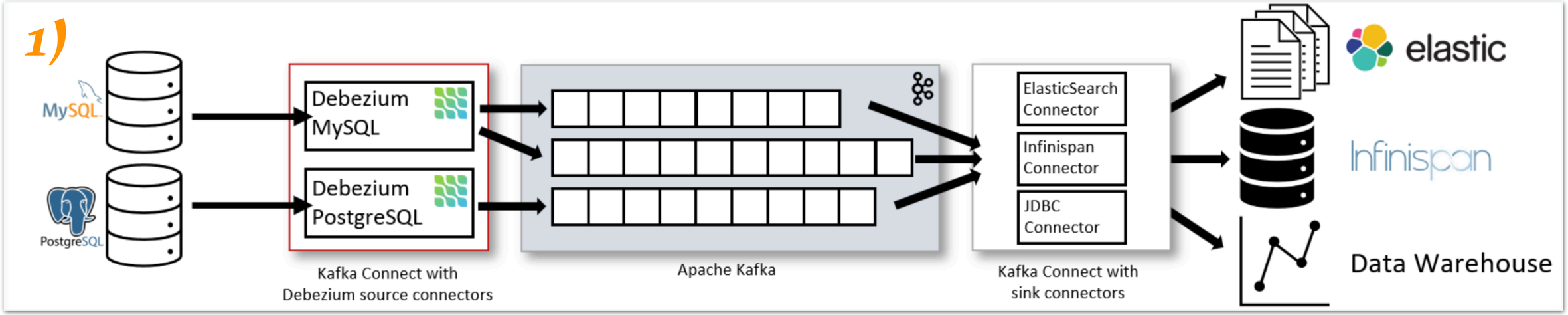
А можно ли без рефакторингов? Может есть готовые решения для интеграции по данным на БД?

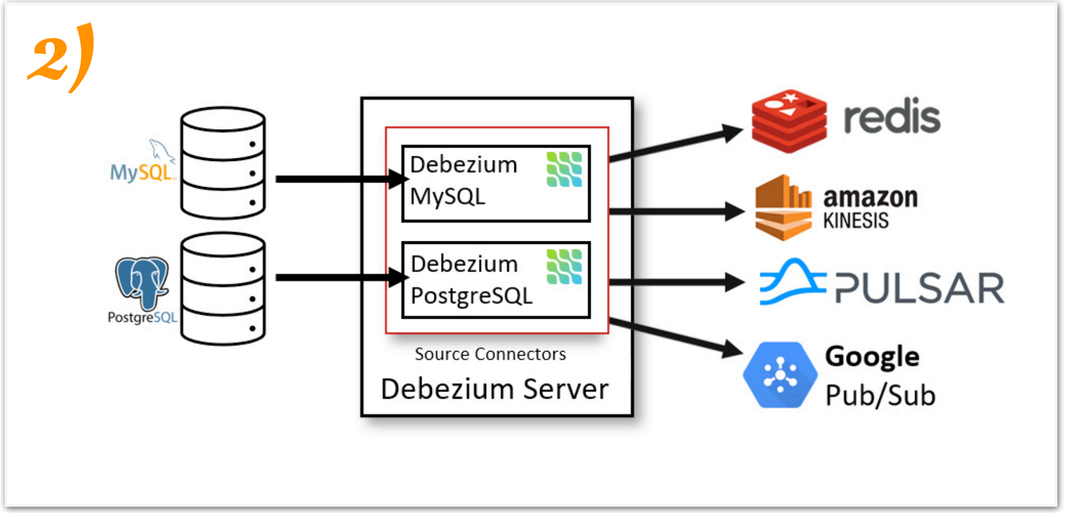
Готовые решения для интеграций по данным: debezium

- Бесплатный
- Настраивается на БД и поставляет данные в виде потока, CDC
- Работает с MySQL, MongoDB, PostgreSQL, Oracle, SQL Server, Db2, Cassandra
- Построен и разворачивается как набор коннекторов для источников под Kafka Connect либо как standalone server
- Может быть настроен в embedded режиме просто как либа
- Может фильтровать, маскировать, трансформировать поток и запускать снепшоты

debezium

Готовые решения для интеграций по данным: debezium





https://debezium.io/documentation/reference/stable/architecture.html

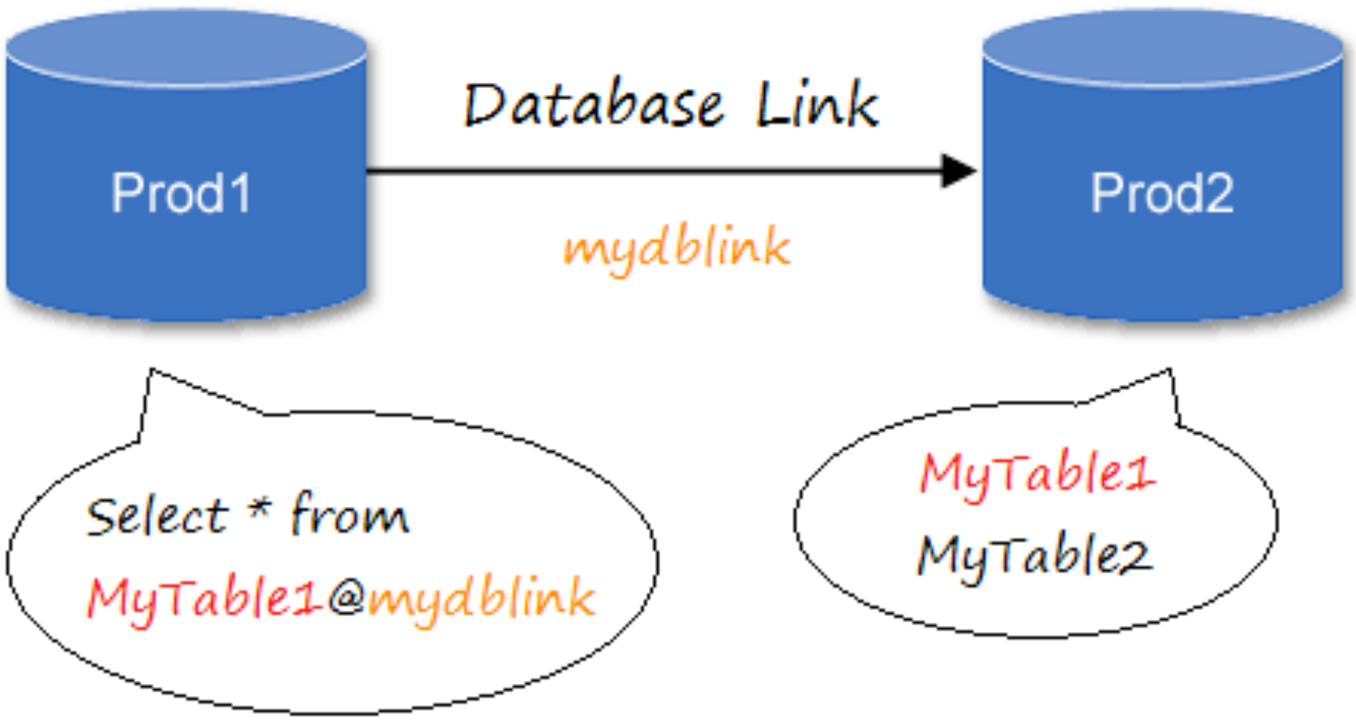
Готовые решения для интеграций по данным: debezium

```
final Properties props = config.asProperties(); // < — 1)
props.setProperty("name", "engine");
props.setProperty("connector.class",
“io.debezium.connector.mysql.MySqlConnector");
…
try (DebeziumEngine<ChangeEvent<String, String>> engine =
DebeziumEngine.create(Json.class)                // < -- 2)
        .using(props)
        .notifying(record -> {
                                                 // < -- 4)
        })
        .build()) {
}

ExecutorService executor = Executors.newSingleThreadExecutor();
executor.execute(engine);                        // < -- 3)
```

Кейз №2

Кейз №2

censored

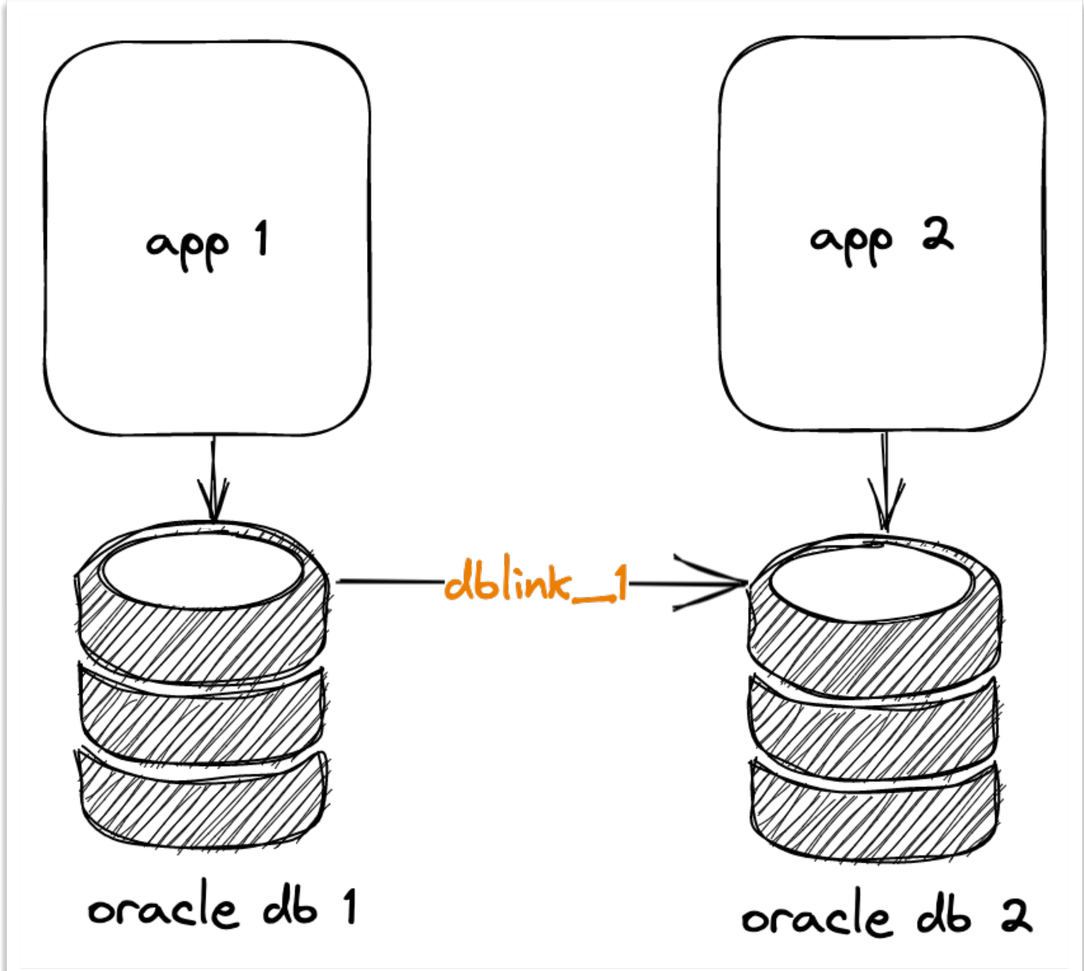


https://betacode.net/10585/oracle-database-link-synonym

Кейз №2

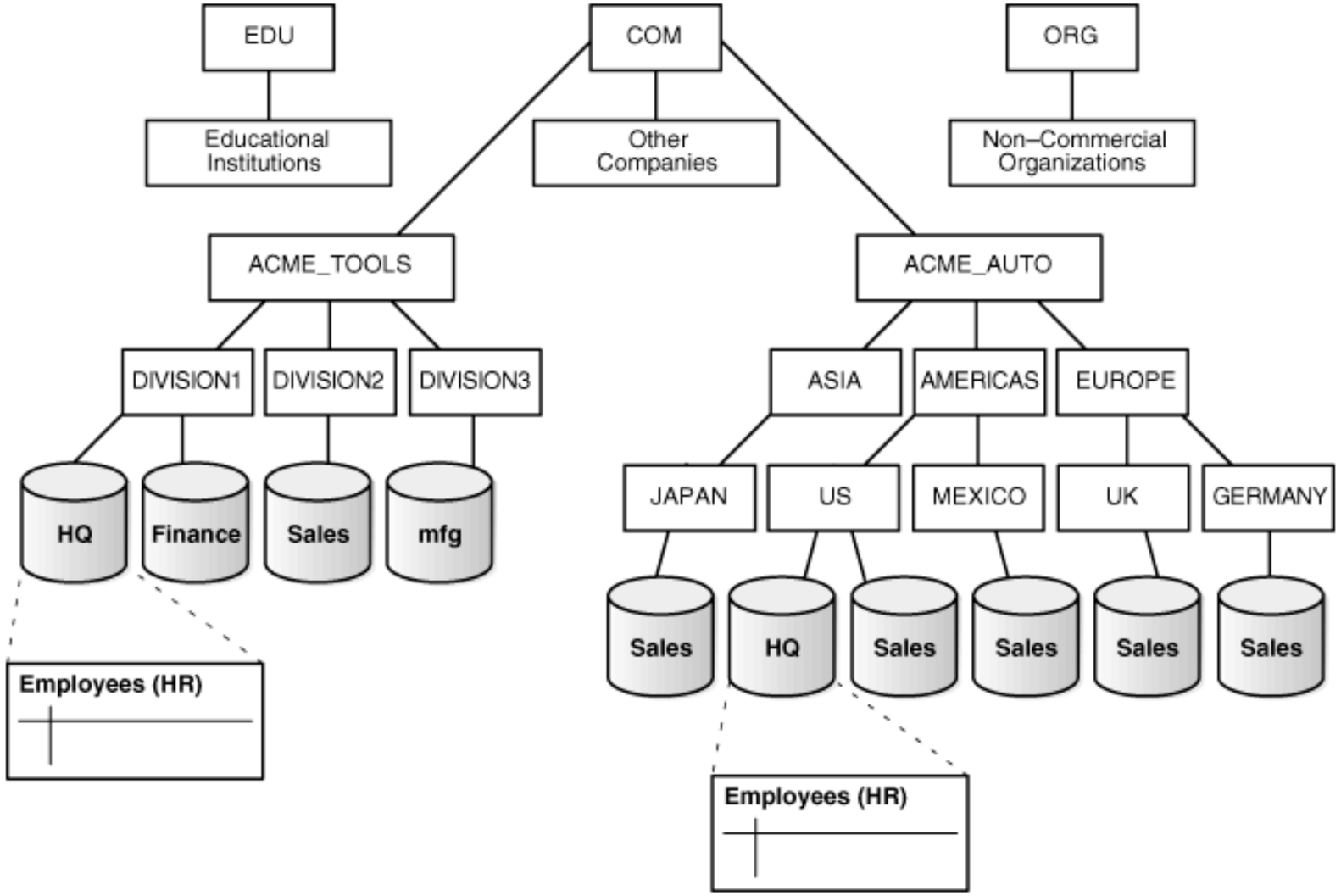
censored

```
SELECT B.BIC  BIK,
       B.NAME BANK_NAME,
       B.KS   CORR_ACC
FROM BANK.TFC_BANKS@SYBASE_ENGINE B
WHERE (:B1 IS NULL)
   OR (B.BIC = :B1);

SELECT SUM(T.ACCOUNT_CURRENCY_AMOUNT)
FROM BACKOFFICE.AUTH_LIST@SYBASE_ENGINE T
WHERE T.ACCOUNT_NO = :B2
  AND T.OPER_DATE <= TRUNC(:B1);

SELECT NVL(MAX(T.ACC_MIR), CHR(1))
FROM BACKOFFICE.DIVISION@SYBASE_ENGINE T
WHERE DIV_CODE = :B1;
```



Кейз №2

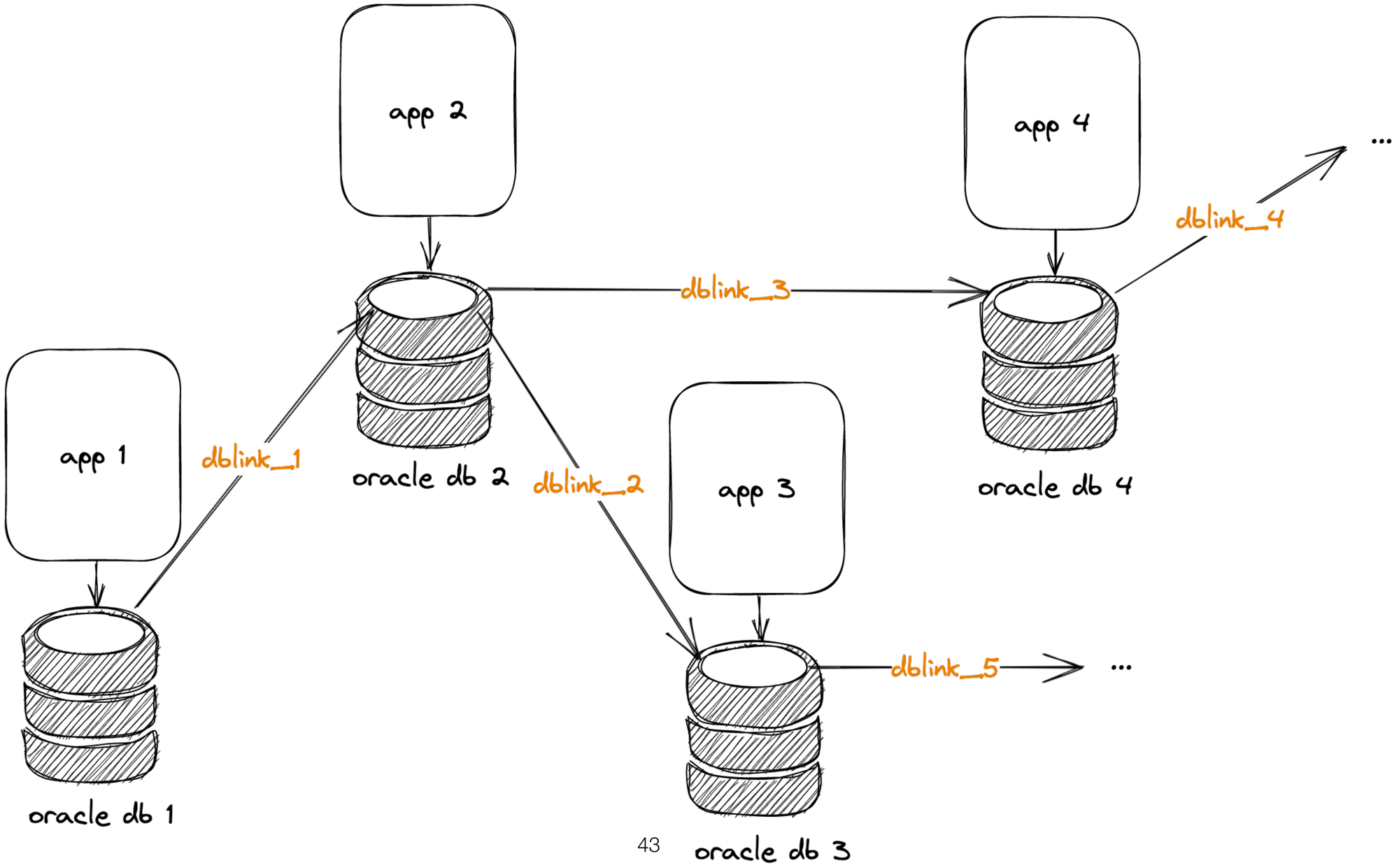
censored



https://docs.oracle.com/cd/E18283_01/server.112/e17120/ds_concepts002.htm

Кейз №2

censored



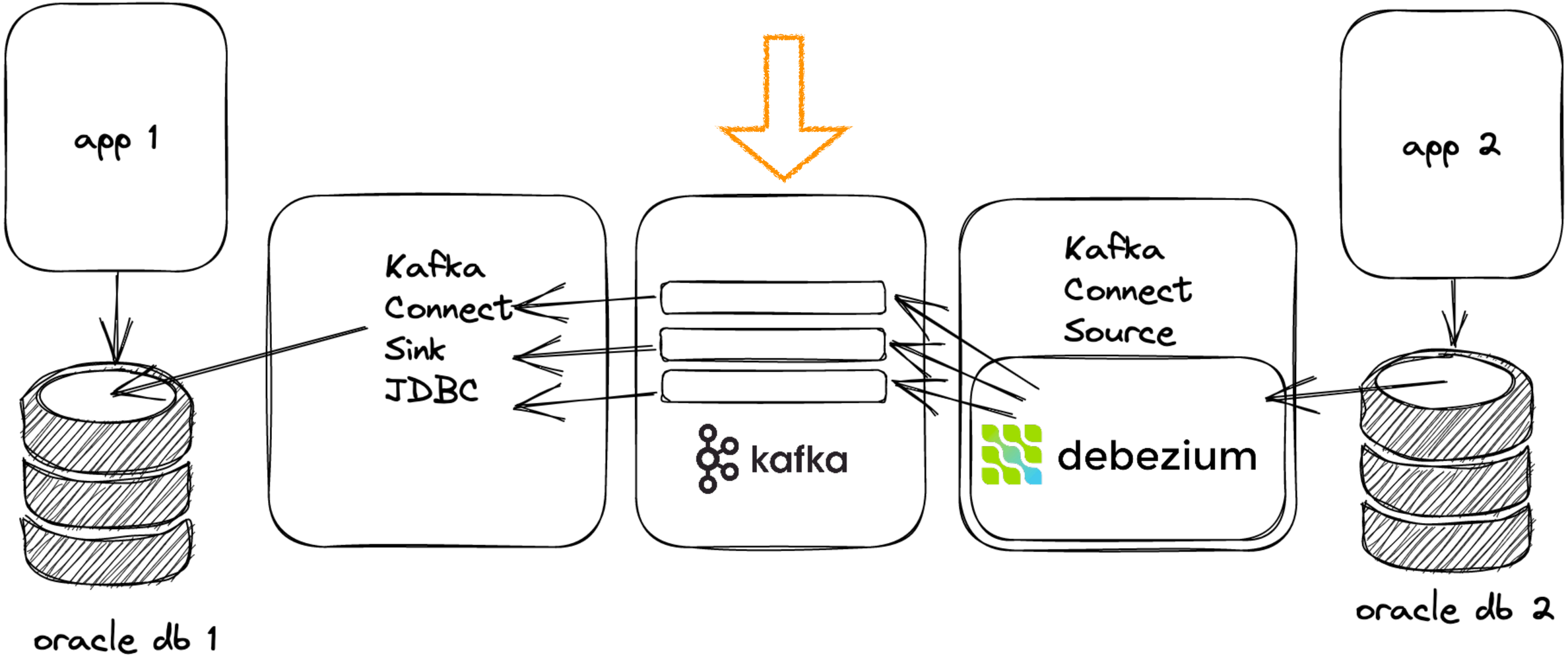
Кейз №2

censored

1. Снимаем статистику всех запросов на всех приложениях идущих по dblink
2. Понимаем в каких приложениях находятся “золотые источники”, а в каких нужно создать локальные таблицы для интеграционных данных
3. Для каждого “золотого источника” настраиваем push-pipeline на базе debezium и Kafka Connect (смотрим следующий слайд)

Кейз №2

censored



Кейз №2

censored

4. Делаем initial load интеграционных данных через materialized view, затягивая данные из “золотого источника” по dblink и перекладываем в интеграционную таблицу
5. Включаем push-pipeline в холостом режиме одновременно с работающим dblink и старой логикой
6. Проводим тестирование данных приходящих по dblink и push-pipeline на основании статистики запросов
7. Переводим логику с dblink на локальные запросы к интеграционным данным
8. Выключаем dblink и удаляем старую логику

Готовые решения для интеграций по данным: StreamSets

- Платный
- “Швейцарский нож” в мире интеграций по данным, ingestion
- Push и pull коннекторы практически ко всему https://streamsets.com/products/connectors/
- Визуальный конструктор пайплайнов
- Фильтрация, трансформация данных и т п

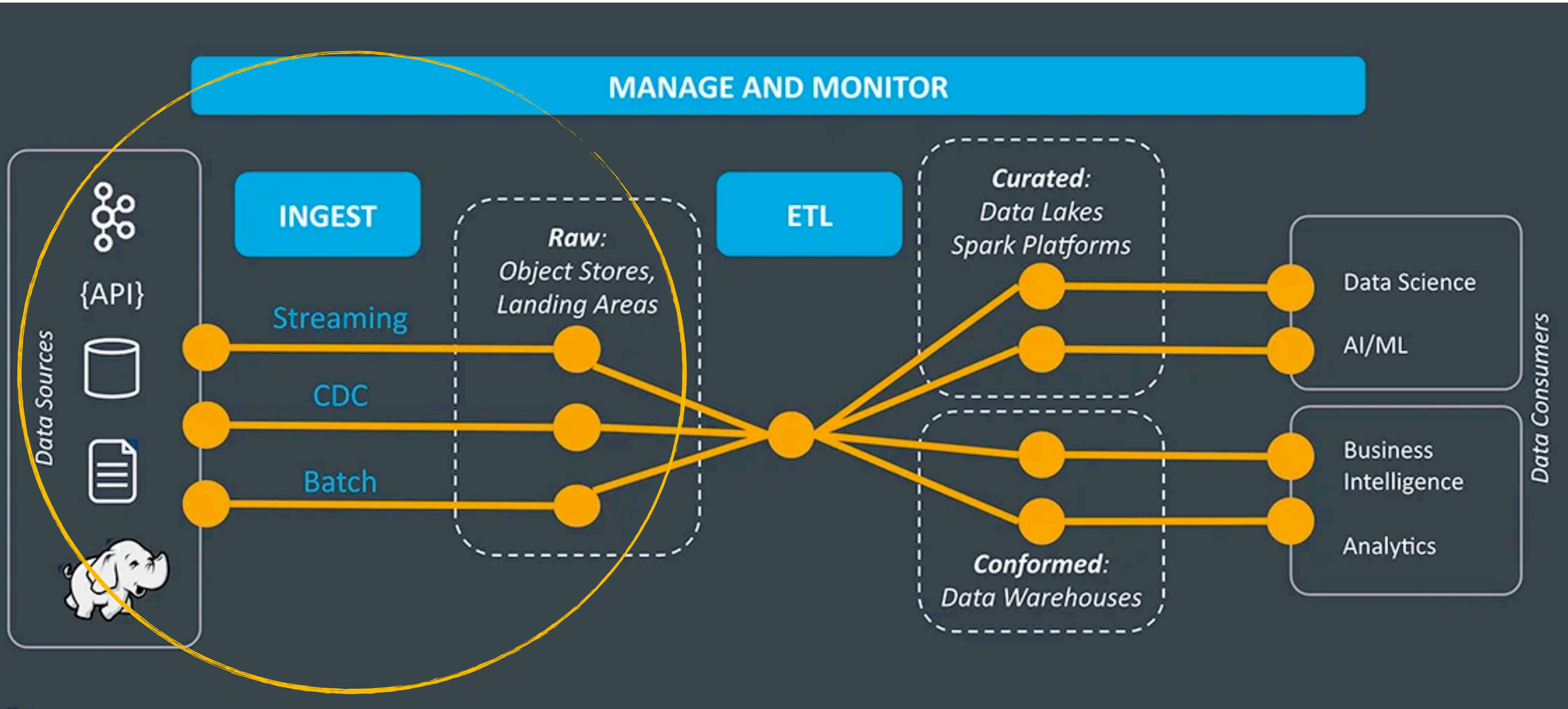
StreamSets
A SOFTWARE AG COMPANY

Кейз №3

SANDOZ A Novartis Division

Кейз №3

SANDOZ A Novartis Division



Выводы (часть 1)

1. Понятие интеграции по данным

2. Объем и характер данных влияет на API, см. матрицу интеграции по данным

3. Подход с выгрузками в общий S3, CH, YT (Кейз 1)

4. debezium для рефакторинга dblink (Кейз 2)

5. Streamsets для наполнения озера данных (Кейз 3)

6. Shared database и доступом по jdbc в витрину на реплике

7. Аудит данных на тригеррах БД

Спасибо и до встречи на второй части!

@vlakra
vladimir.krasilschik@gmail.com