Миграция больших БД без остановки

Как перевести БД с DB2 for z/OS на PostgreSQL с минимальным временем простоя

ВОСХОД
www.voskhod.ru

Дмитрий Погибенко

Требования к процессу миграции

- Преобразование данных
- Перенос данных без остановки БД
- Зависимости между таблицами
- Сжатые сроки перехода на PostgreSQL
- Возможность мигрировать другие БД с минимальными доработками

Требования к процессу миграции

- Преобразование данных

В процессе переноса большие бинарные объекты(BLOB) сохраняются в Ceph, а в БД сохраняются только их идентификаторы.

Требования к процессу миграции

- Преобразование данных
- Перенос данных без остановки БД

  Нужно отслеживать и обрабатывать изменения в данных

Требования к процессу миграции

- Преобразование данных
- Перенос данных без остановки БД
- Зависимости между таблицами

  Во время переноса в целевой БД существуют все внешние ключи

Требования к процессу миграции

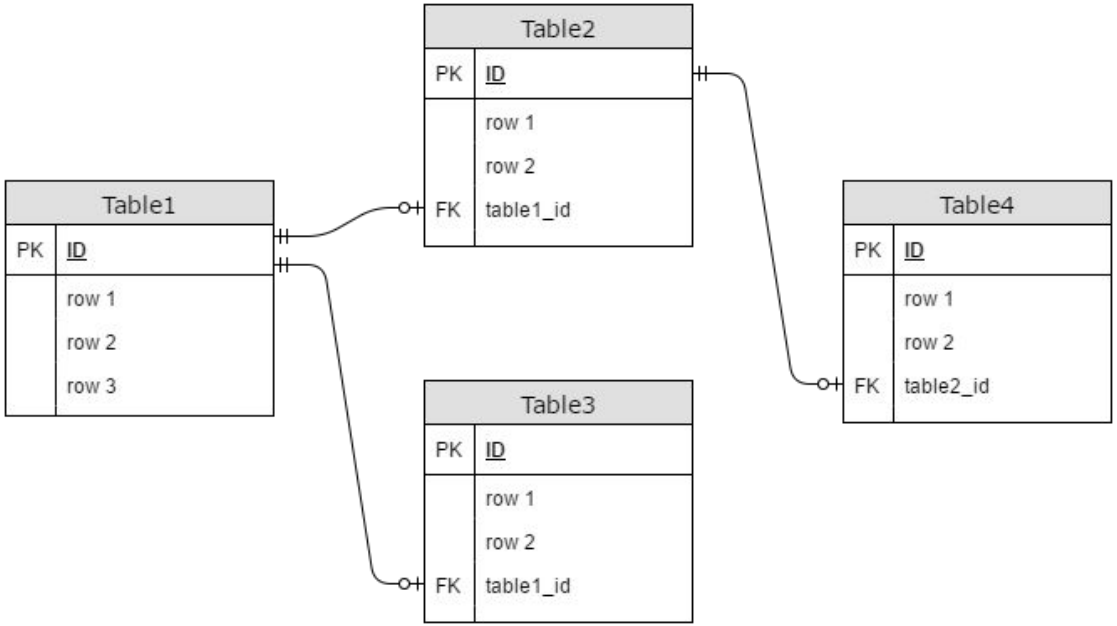
- Преобразование данных
- Перенос данных без остановки БД
- Зависимости между таблицами
- Сжатые сроки перехода на PostgreSQL

  Миграция не должна занимать больше месяца

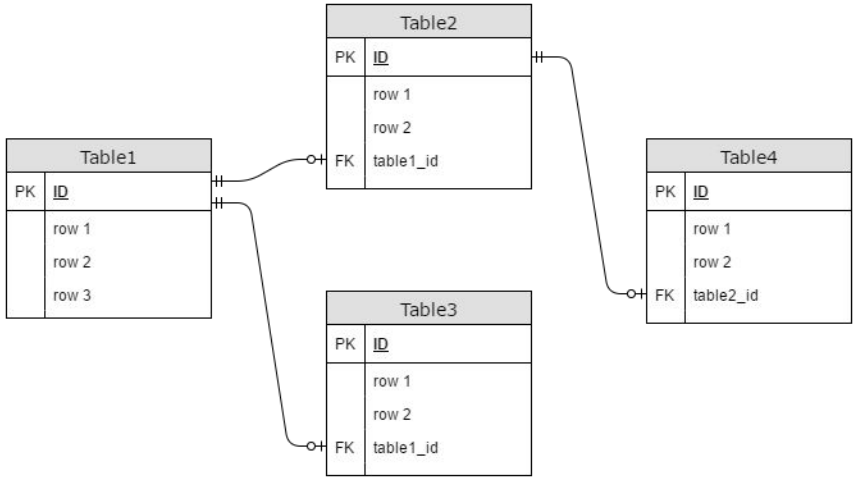
Требования к процессу миграции

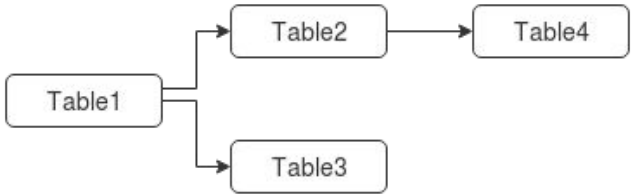
- Преобразование данных
- Перенос данных без остановки БД
- Зависимости между таблицами
- Сжатые сроки перехода на PostgreSQL
- Возможность мигрировать другие БД с минимальными доработками

Как жить с зависимостями



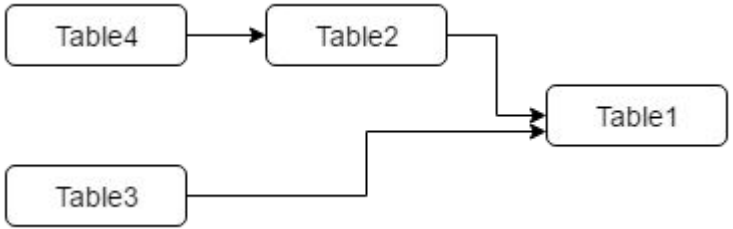
Порядок переноса данных с учетом зависимостей



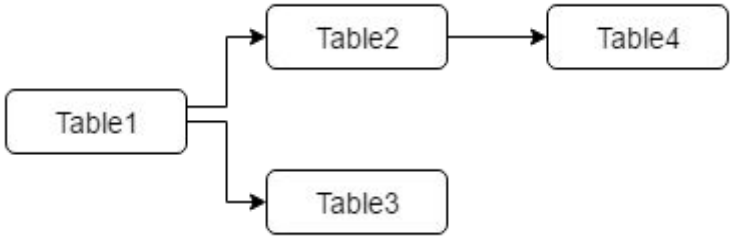


Обратный порядок расчета диапазонов

Расчет диапазонов



Перенос данных



Реальный граф зависимостей

Автоматизируем анализ зависимостей

SchemaSpy (http://schemaspy.sourceforge.net/)

- Анализирует зависимости
- Ищет циклические зависимости
- Упорядочивает таблицы от независимых к зависимым
- Открытый код. Можно почистить лишнее и прикрутить шаблонизатор

Переносим данные

Spring Batch (http://projects.spring.io/spring-batch/)

- Управление транзакциями
- Обработка по частям
- Декларативная конфигурация
- Запуск/останов/перезапуск
- Гибкие стратегии пропуска/повтора при ошибках

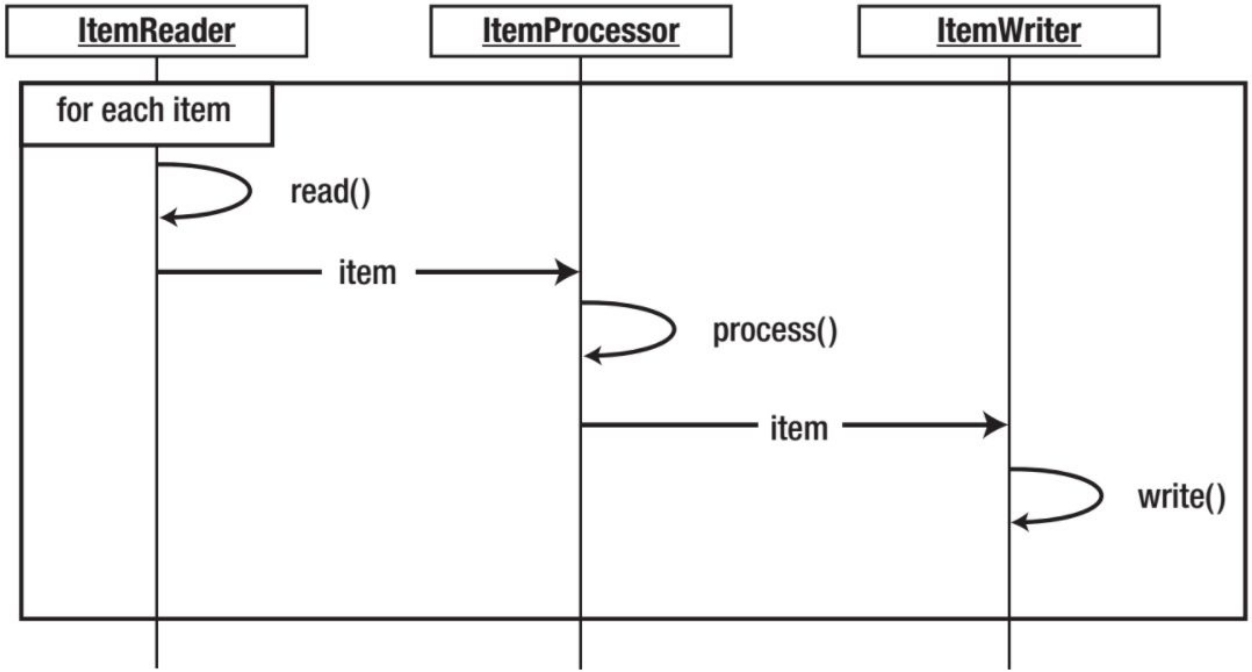
Декларативная конфигурация Spring Batch

```
<batch:job id="sampleJob" job-repository="jobRepository">
    <batch:step id="data_step">
        <batch:tasklet>
            <batch:chunk reader="data_reader"
                processor="data_processor"
                writer="data_writer"
                commit-interval="100"/>
        </batch:tasklet>
    </batch:step>
</batch:job>
```

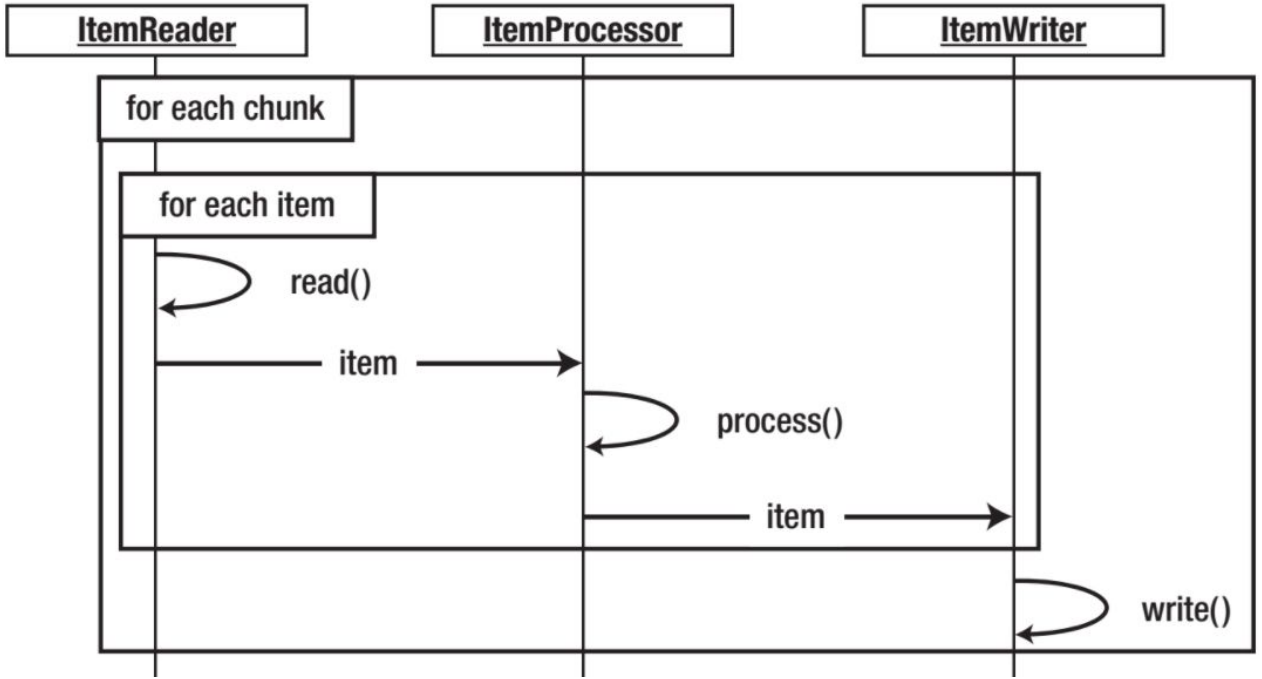
Центральные интерфейсы Spring Batch

```
public interface ItemReader<T> {
    T read() throws Exception;
}

public interface ItemProcessor<I, O> {
    O process(I item) throws Exception;
}

public interface ItemWriter<T> {
    void write(List<? extends T> items) throws Exception;
}
```

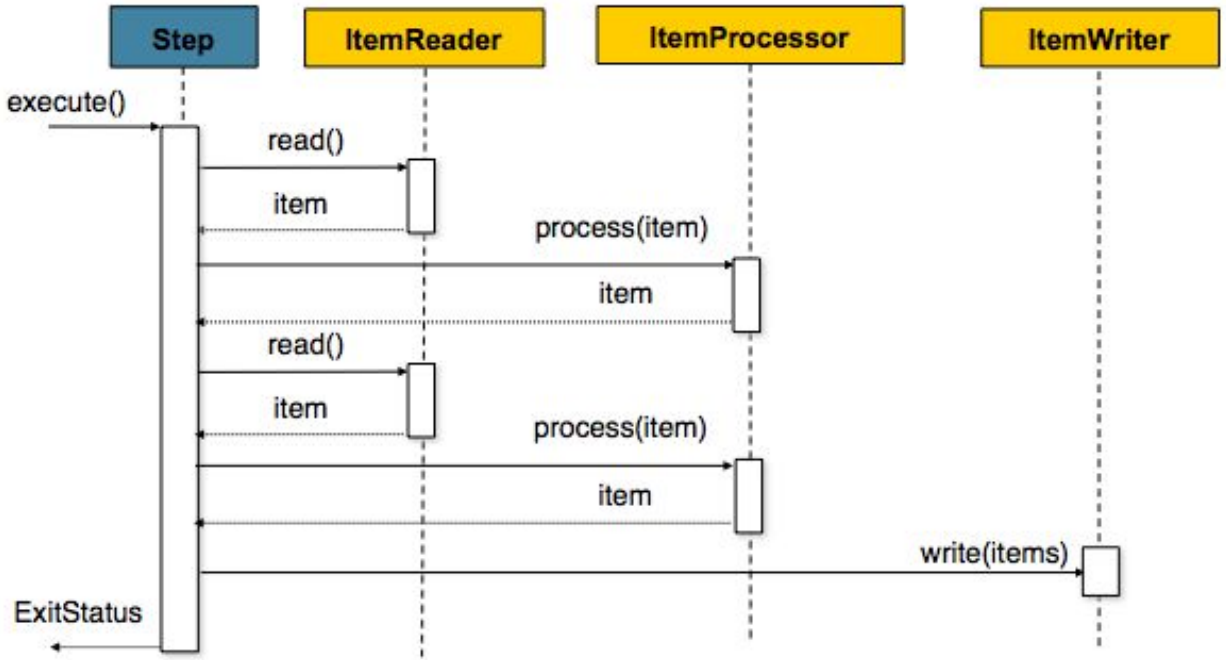
Item-ориентированная обработка в Spring Batch



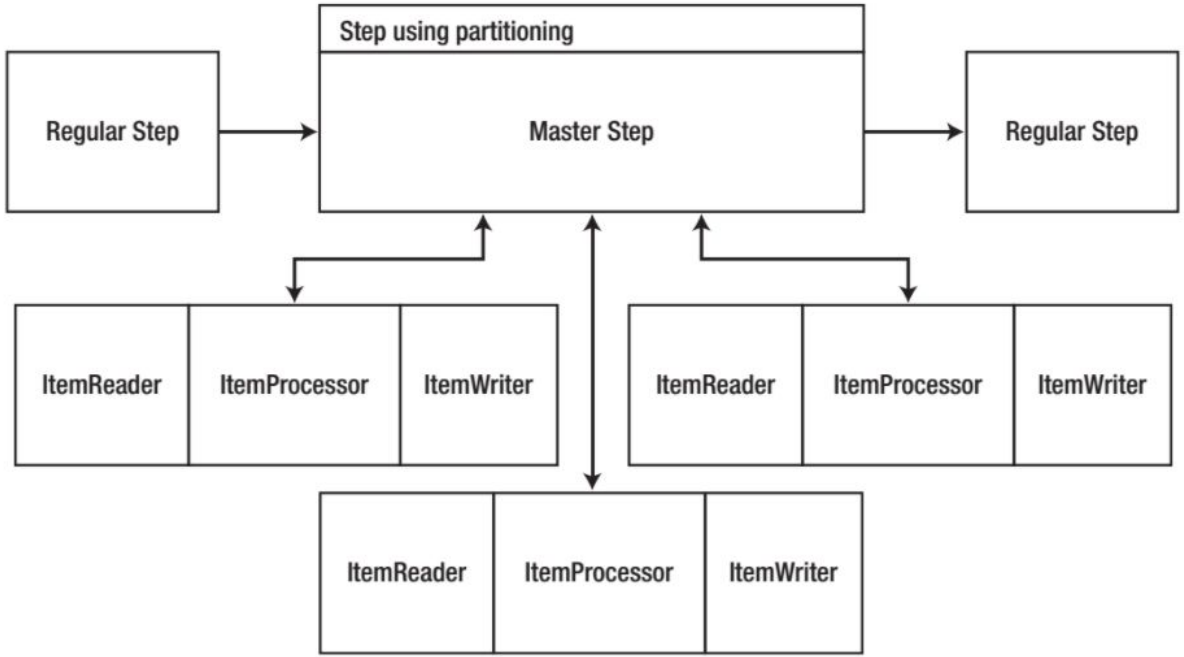
Chunk-ориентированная обработка в Spring Batch



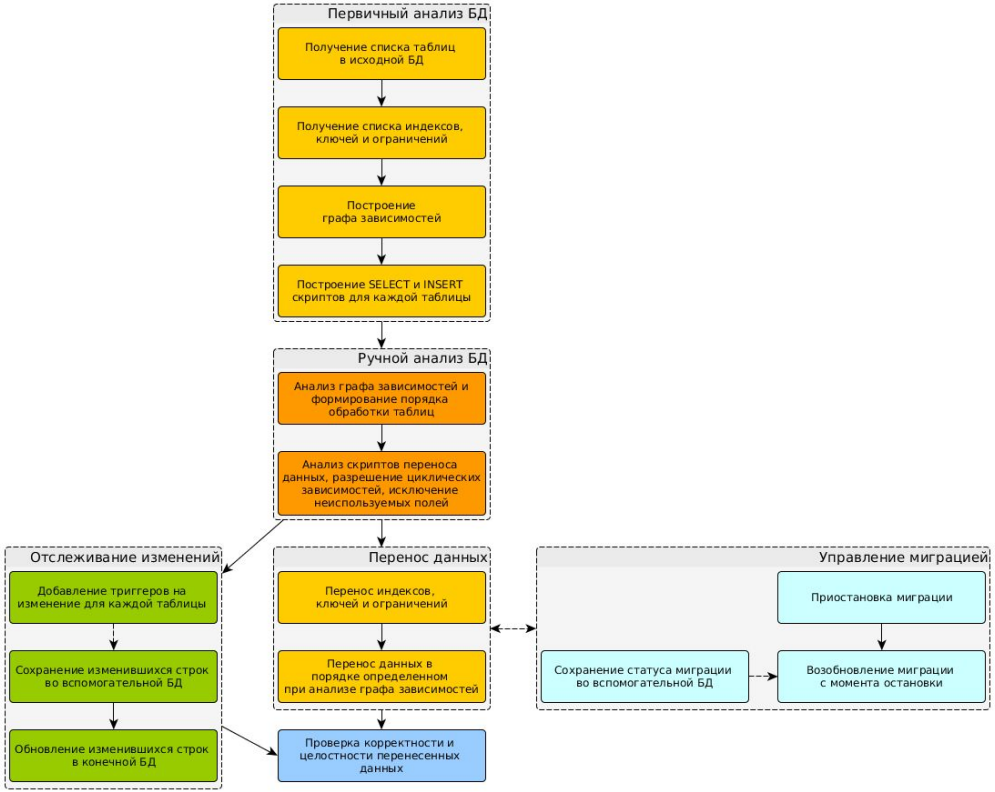
Chunk-ориентированная обработка в Spring Batch



Партиционирование в Spring Batch



Процесс миграции



Отслеживание и перенос изменений

- Изменения в данных отслеживаются триггерами
- Идентификаторы измененных, добавленных и удаленных строк записываются в служебные таблицы
- Система переноса данных периодически читает служебные таблицы и переносит соответствующие строки

Служебные таблицы отслеживания изменений

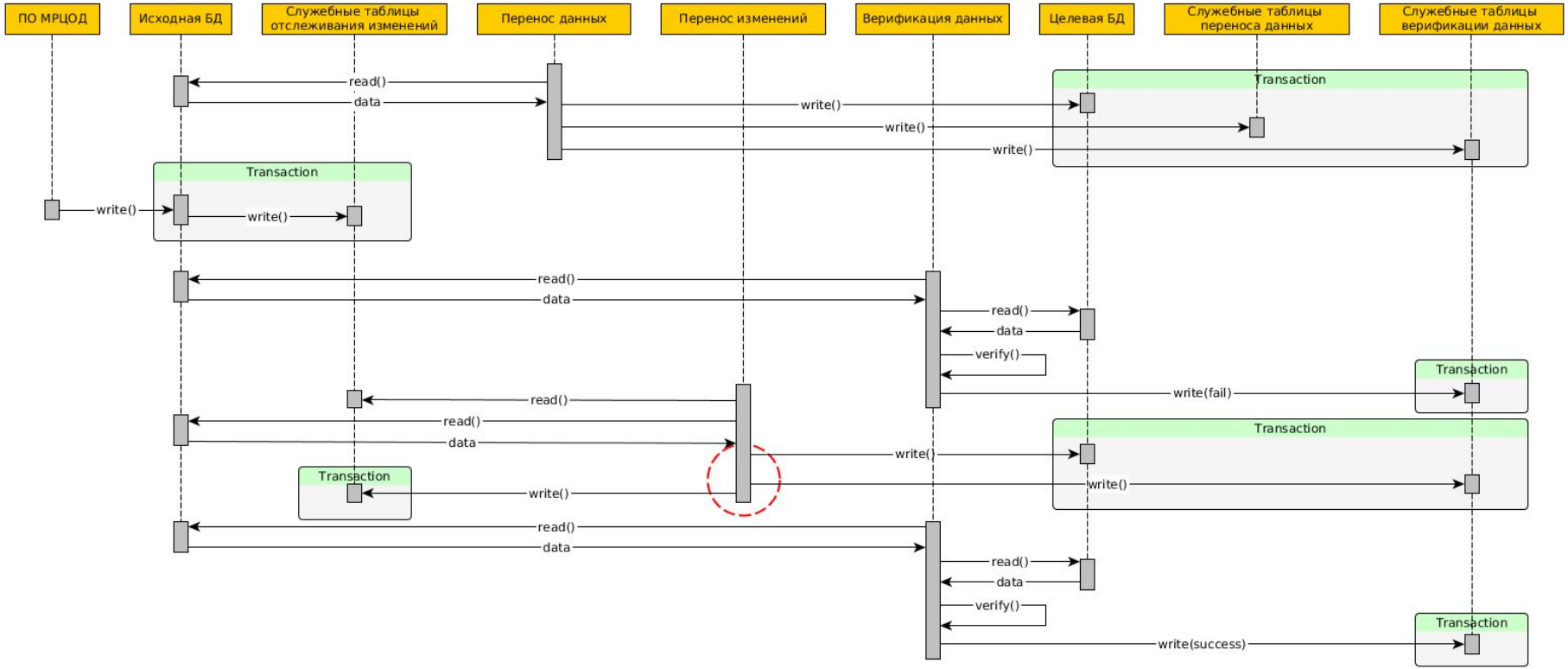
Table1	
PK	ID
	row 1
	row 2
	row 3

Table1_changes	
PK	ID
	optimestamp
	operation

Table2	
PK	ID
	row 1
	row 2
	row 3

Table2_changes	
PK	ID
	optimestamp
	operation

Упрощенная диаграмма работы системы



Разбиение данных на диапазоны(простое)

1. Получаем минимальный и максимальный id
2. Все значения id между ними делим на равные диапазоны

Разбиение данных на диапазоны(простое)

Плюсы:

- Быстро работает
- Легко реализовать

Минусы:

- Разное количество строк в диапазонах при неравномерном распределении id

Разбиение на диапазоны с равным числом строк

1. Получаем число строк в таблице
2. Делим количество строк в таблице на число диапазонов - это количество строк в каждом диапазоне
3. Получаем первый и последний элемент каждого диапазона запросами вида:

```
SELECT id as min_id FROM tablename
    WHERE id > :prev_id ORDER BY id LIMIT 1;

SELECT id as max_id FROM tablename
    WHERE id >= :min_id ORDER BY id LIMIT 1 OFFSET partitionSize;
```

Разбиение на диапазоны с равным числом строк

Плюсы:

- Одинаковое количество строк в каждом диапазоне

Минусы:

- Медленно работает
- Зависит от диалекта SQL

Решения

- Изменения данных отслеживаются триггерами
- Зависимости анализируются SchemaSpy
- Данные переносятся Spring Batch

Каждая миграция имеет свои особенности

- Снести индексы и восстановить после переноса. Быстрее или нет?

  В нашем случае нет.

- Размер коммита и количество потоков нужно подбирать экспериментально.

  В нашем случае оптимальным оказалось 16 потоков(по числу ядер на сервере Эльбрус) и 1000 строк в каждой транзакции.

Базы данных, которые легко мигрировать

- Нет бизнес-логики в хранимых процедурах
- Во всех таблицах есть первичный ключ. Желательно равномерно распределенный и не составной
- Нет циклических зависимостей
- Основная операция - INSERT, а UPDATE и DELETE сравнительно мало