

ТИНЬКОФФ

Миграция данных из Oracle в PostgreSQL

Сценарии и практики

Дима Рудик
Андрей Бронников

28.03.2024

Как мигрировать данные?

- Кто поможет с оценкой нагрузки?
- К кому идти за помощью по миграции?



Что может пойти не так?

- Во время переноса данных получили деградацию сервиса
- После переключения получили сбой
- Развернули трафик обратно с потерей данных



Какие виды миграций актуальны?

- По типу источника
 - Oracle
 - PostgreSQL
- По сценарию миграции
 - Без синхронизации данных, даунтайм определяется объемом переливаемых данных
 - С синхронизацией данных, даунтайм определяется временем переключения приложения на новый источник

О чём презентация



Сценарии миграций

Как не потерять данные при сложном сценарии миграции.



Миграция с даунтаймом до 45 мин

Актуальна при выезде не более 200GB и даунтайме не более 45 минут.



Миграция с даунтаймом на cutover

При объемах на источнике более 200GB.
Актуальна при миграции критичных сервисов.

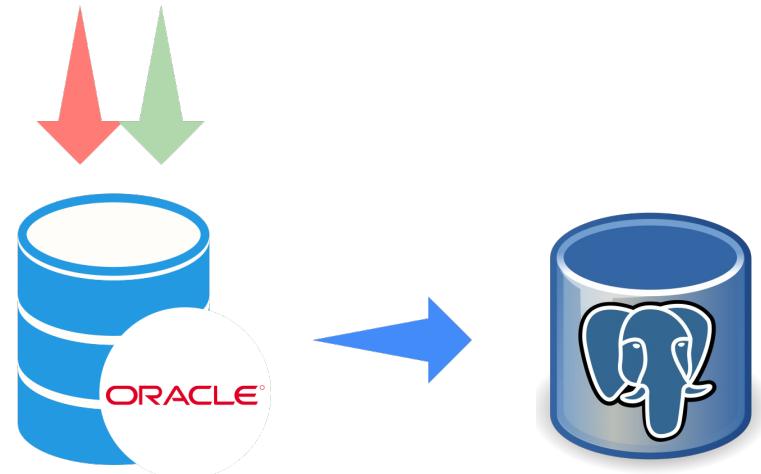
Сценарии миграций

Как переключаться

↗ - пишем и читаем
↘ - читаем

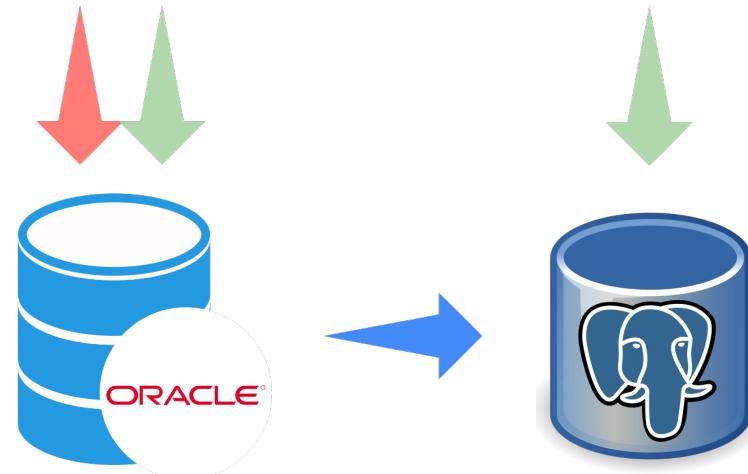
01

Синхронизировать данные



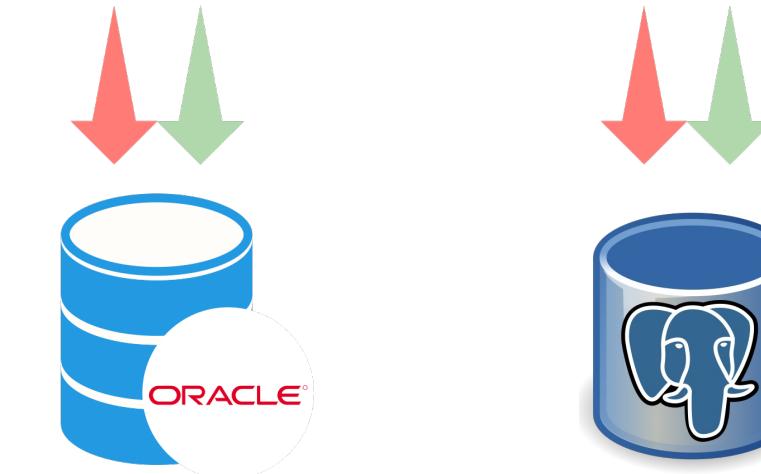
02

Продублировать RO нагрузку



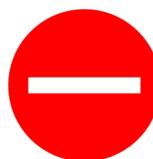
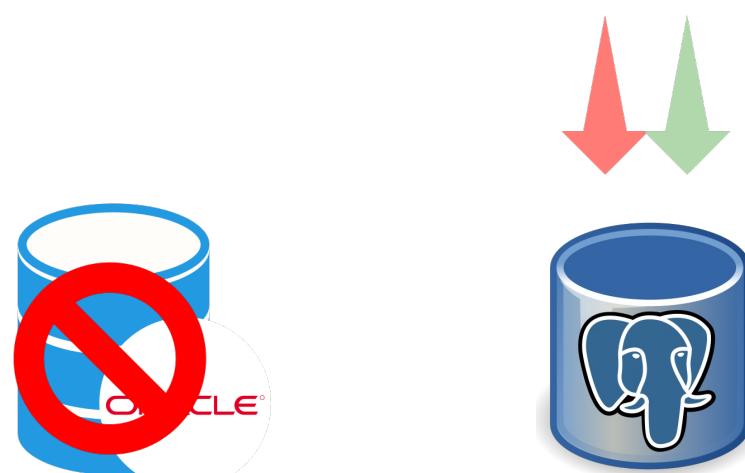
03

Продублировать RO + RW нагрузку

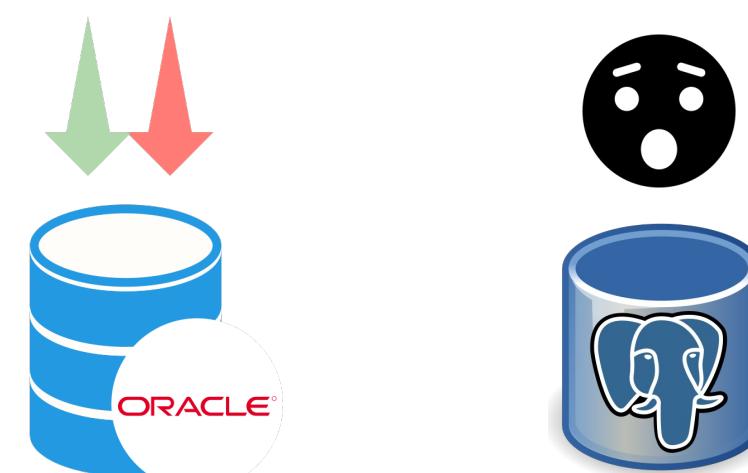


04

Переключиться на новый источник



Переключиться обратно по пункту 03



- 02 – применим только при CDC
- 02 и 03 – можно поменять местами

Сценарий для **Insert Only** нагрузки без синхронизации

01

Перелить основную часть

```
[  
{  
    "fromSchemaName" : "TEST",  
    "fromTableName" : "TABLE1",  
    "toSchemaName" : "PUBLIC",  
    "toTableName" : "TABLE1",  
    "fetchHintClause" : "/*+ no_index(TABLE1) */",  
    "fetchWhereClause" : " ID <= 100000 ",  
    "fromTaskName" : "TABLE1_TASK",  
    "columnToColumn" : {  
    }  
}
```

02

Перелить подготовительную
часть для выполнения
короткого даунтайма.

Построить индексы

```
{  
...  
    "toTableName" : "TABLE1",  
    "fetchHintClause" : "/*+ no_index(TABLE1) */",  
    "fetchWhereClause" : " ID > 100000 and ID <= 100007 ",  
    "fromTaskName" : "TABLE1_TASK",  
    "columnToColumn" : {  
    }  
}
```

03

Остановить нагрузку,
перелить остаток, запустить
приложение

```
{  
...  
    "fetchHintClause" : "/*+ no_index(TABLE1) */",  
    "fetchWhereClause" : " ID > 100007 ",  
...  
}
```

ID	NAME	VALUE
1	A	10
2	B	2
3	C	53
4	D	4678
5	E	25
6	F	247
7	G	4
8	H	235
...
100000	J	12
100001	K	4
100002	L	90
100003	M	32
100004	N	894
100005	O	
100006	P	3
...
100007	Q	45634
100008	R	0
100009	S	35
100010	T	24

Доливка исторических данных

- Возможна, если исторические данные доступны в прежнем источнике
- Для уменьшения времени создания индексов

```
[  
  
{  
    "fromSchemaName" : "TEST",  
    "fromTableName" : "TABLE1",  
    "toSchemaName" : "PUBLIC",  
    "toTableName" : "TABLE1",  
    "fetchHintClause" : "/*+ no_index(TABLE1) */",  
    "fetchWhereClause" : " ID <= 100000 ",  
    "fromTaskName" : "TABLE1_TASK",  
    "columnToColumn" : {  
        ...  
    }  
}  
]
```

ID	NAME	VALUE
1	A	10
2	B	2
3	C	53
4	D	4678
5	E	25
6	F	247
7	G	4
8	H	235
...
100000	J	12
100001	K	4
100002	L	90
100003	M	32
100004	N	894
100005	O	
100006	P	3

Сценарий для CRUD нагрузки без синхронизации

01

Перелить неизменяемую
часть. Построить индексы

```
[  
{  
    "fromSchemaName" : "TEST",  
    "fromTableName" : "TABLE1",  
    "toSchemaName" : "PUBLIC",  
    "toTableName" : "TABLE1",  
    "fetchHintClause" : "/*+ no_index(TABLE1) */",  
    "fetchWhereClause" : " ID <= 100000 ",  
    "fromTaskName" : "TABLE1_TASK",  
    "columnToColumn" : {  
        ...  
    }  
}
```

02

Остановить нагрузку,
перелить изменяемый
остаток, запустить
приложение

```
{  
...  
    "toTableName" : "TABLE1",  
    "fetchHintClause" : "/*+ no_index(TABLE1) */",  
    "fetchWhereClause" : " ID > 100000 ",  
    "fromTaskName" : "TABLE1_TASK",  
    "columnToColumn" : {  
        ...  
    }  
}
```

ID	NAME	VALUE
1	A	10
2	B	2
3	C	53
4	D	4678
5	E	25
6	F	247
7	G	4
8	H	235
...
100000	J	12
100001	K	4
100002	L	90
100003	M	32
100004	N	894
100005	O	
100006	P	3
...
100007	Q	45634

Сценарий для CRUD нагрузки с синхронизацией



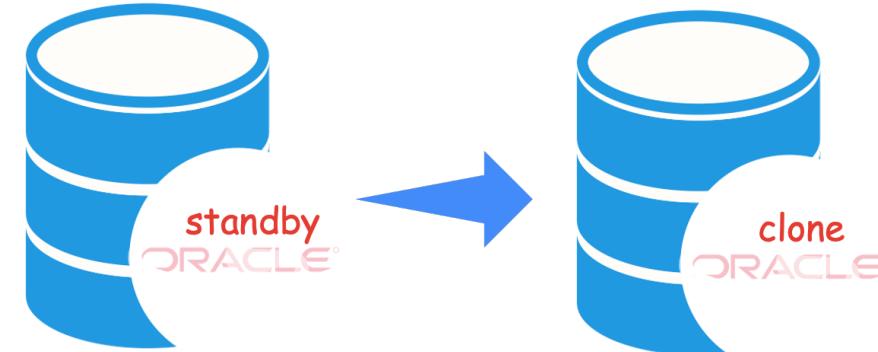
01

Создать Extract в Oracle



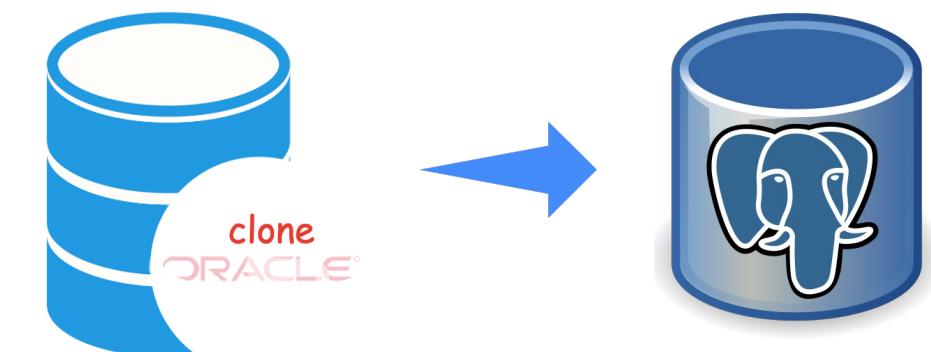
02

Создать клон с реплики Oracle и
активировать его (запомнить SCN)



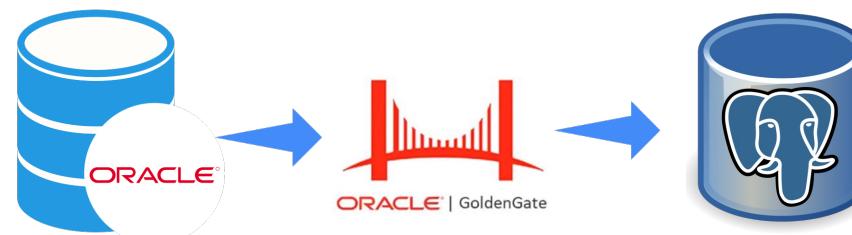
03

Мигрировать статичные данные с
клона



04

Запустить Replicat на применение с
SCN от шага 2.



05

Остановить Replicat. Выполнить
переключение приложения.



06

Удалить Extract и Replicat



Сценарий для **CRUD** нагрузки с синхронизацией

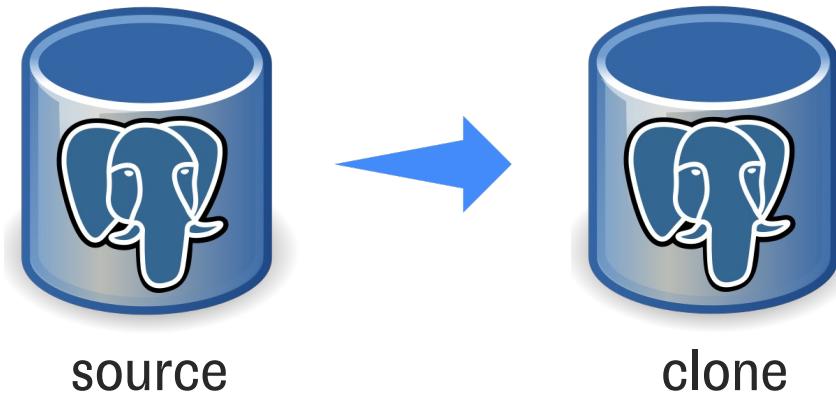


01

Создать slot, publication на источнике

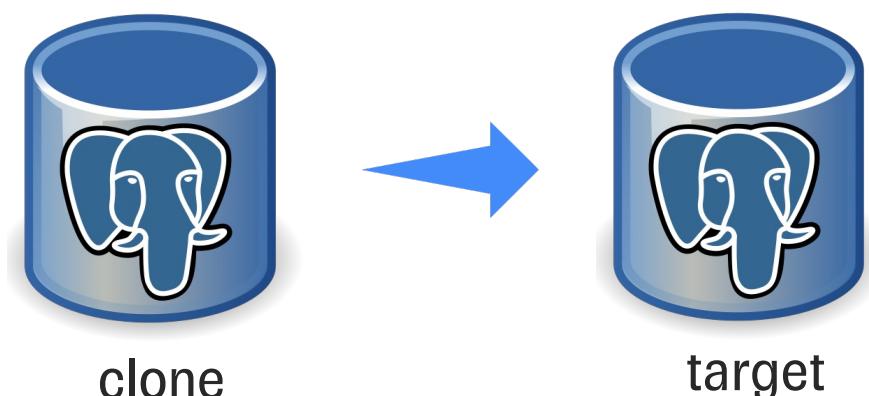
02

Создать клон с реплики и
активировать его (запомнить LSN)



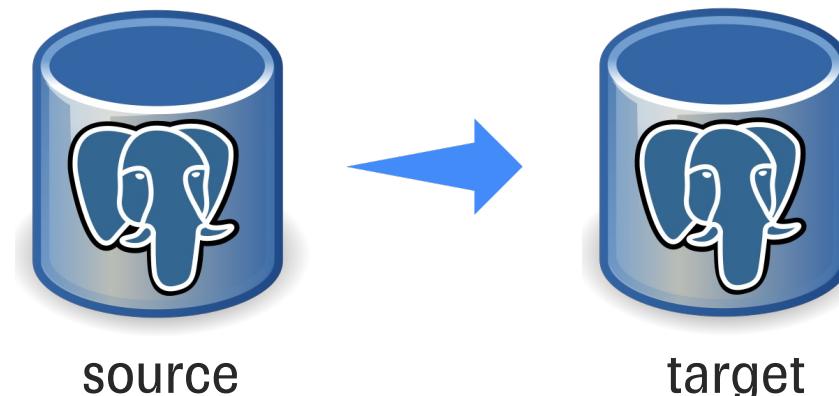
03

Мигрировать статичные данные с
клона



04

Запустить Subscription на применение
с LSN от шага 2.



05

Остановить Subscription. Выполнить
переключение приложения.

06

Удалить slot и publication

Multiple Data Sources

Практики по реализации нескольких DataSource для Java

- JdbcTemplate и MyBatis не вызывают проблем
- JPA Hibernate – необходимы доработки

Spring Dynamic DataSource Routing

- <https://spring.io/blog/2007/01/23/dynamic-datasource-routing> (`AbstractRoutingDataSource`)
- <https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/jdbc/datasource/lookup/AbstractRoutingDataSource.html>
- <https://www.baeldung.com/spring-abstract-routing-data-source>

```
@Component
@Slf4j
@Getters
public class DataSourceRouter extends AbstractRoutingDataSource {
    private final DataSourceContextHolder dataSourceContextHolder;
    private final HikariProperties hikariProperties;

    public DataSourceRouter(DataSourceContextHolder dataSourceContextHolder,
                           DataSourceMap dataSourceMap,
                           HikariProperties hikariProperties) {
        this.hikariProperties = hikariProperties;
        this.dataSourceContextHolder = dataSourceContextHolder;
        Map<Object, Object> map = dataSourceMap.initMap();
        this.setTargetDataSources(map);
        this.setDefaultTargetDataSource(map.get(this.hikariProperties.getSource()));
    }

    @Override
    protected Object determineCurrentLookupKey() {
        return dataSourceContextHolder.getContext();
    }
}
```

ThreadLocal<T> AbstractRoutingDataSource

- <https://docs.oracle.com/en%2Fjava%2Fjavase%2F21%2Fdocs%2Fapi%2Fjava.base/java/lang/ThreadLocal.html>

```
@Component
public class DataSourceContextHolder {
    private static ThreadLocal<String> CONTEXT = new ThreadLocal<>();

    public AutoCloseable setContext(String dataSourceContext) {
        Assert.notNull(dataSourceContext, "clientDatabase cannot be null");
        CONTEXT.set(dataSourceContext);
        return () -> CONTEXT.remove();
    }

    public String getContext() {
        return CONTEXT.get();
    }

    public void clear() {
        CONTEXT.remove();
    }
}
```

application.properties

```
spring.jpa.show-sql = true
spring.jpa.open-in-view = false

hikari.source = datasource1
hikari.target = datasource2

hikari.config.datasource1.jdbcurl = jdbc:oracle:thin:@(description=(address=(host=localhost)(protocol=tcp)(port=1521))(connect_data=(service_name=xepdb1)))
hikari.config.datasource1.username = test
hikari.config.datasource1.password = test

hikari.config.datasource2.jdbcurl = jdbc:postgresql://localhost/postgres
hikari.config.datasource2.username = test
hikari.config.datasource2.password = test
hikari.config.datasource2.connectionTimeout = 500
hikari.config.datasource2.validationTimeout = 1000
```

```
@Component
@ConfigurationProperties(prefix = "hikari")
@Data
public class HikariProperties {
    private String source;
    private String target;
    private Map<String, HikariConfig> config;
}
```

Доступность источника данных

- Если недоступен target, приложение должно стартовать

```
public Map<Object, Object> initMap() {  
    hikariProperties.getConfig().forEach((k, v) -> {  
        try {  
            HikariDataSource hikariDataSource = new HikariDataSource(v);  
            map.put(k, hikariDataSource);  
        } catch (HikariPool.PoolInitializationException e) {  
            if (k.equals(hikariProperties.getSource())) {  
                throw new RuntimeException();  
            }  
            log.error("Pool {} cannot be initialized", k);  
        }  
    });  
    return map;  
}
```

Синхронизация ID сущностей

- При переключении на target необходимо сдвинуть последовательности
- Значения ID должны быть синхронны во время параллельной записи в два источника данных

```
@Entity
@Table(name = "users_tab")
@Data
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "users_tab_seq")
    @GenericGenerator(name = "users_tab_seq", type = UserIdGenerator.class)
    private long id;
    ...
}
```

```
public class UserIdGenerator extends SequenceStyleGenerator {

    @Override
    public Object generate(SharedSessionContractImplementor sharedSessionContractImplementor, Object o) {
        if (dataSourceContextHolder().getContext() == null
            || dataSourceContextHolder().getContext().equals(hikariProperties().getSource())) {
            return super.generate(sharedSessionContractImplementor, o);
        }
        return ((User) o).getId();
    }

    @Bean
    private HikariProperties hikariProperties(){
        return new HikariProperties();
    }

    @Bean
    private DataSourceContextHolder dataSourceContextHolder() {
        return new DataSourceContextHolder();
    }
}
```

Target Timeout

- Необходимо прерывать запрос на target по таймауту

```
@Repository
public interface UserRepository extends JpaRepository<User, Long> {

    @Query(value = "select id, create_at, name, update_at, gender, pg_sleep(5) from users_tab where id = 1", nativeQuery = true)
    Optional<User> findUserWithTimeOut();
}
```

```
public Optional<User> findUserWithTimeOut() {
    try (AutoCloseable a = dataSourceContextHolder.setContext(hikariProperties.getTarget())) {
        transactionTemplate.setTimeout(3);
        return transactionTemplate.execute(transactionStatus -> userRepository.findUserWithTimeOut());
    } catch (Exception e) {
        log.error("", e);
    }
    return Optional.empty();
}
```

Проблемы

- `@Transactional` будет работать только с дефолтным контекстом
- Синтаксис запросов аннотаций `@Query` должен работать в разнородных источниках имеющих разный диалект

Пишем в два источника по паттерну Outbox

```
public Optional<User> multiSaveUser(User user, RequestMethod method) {  
    Outbox outbox = new Outbox();  
    outbox.setMethod(method);  
  
    Optional<User> optionalUser = transactionTemplate.execute(transactionStatus -> {  
        Optional<User> o = saveUser(user);  
        try {  
            ObjectMapper objectMapper = JsonMapper.builder()  
                .findAndAddModules()  
                .build();  
            outbox.setMessage(objectMapper.writeValueAsString(o.get()));  
        } catch (JsonProcessingException e) {  
            throw new RuntimeException(e);  
        }  
        outboxRepository.save(outbox);  
        return o;  
    });  
  
    if (dataSourceMap.hasDataSource(hikariProperties.getTarget())) {  
        try (AutoCloseable a = dataSourceContextHolder.setContext(hikariProperties.getTarget())) {  
            saveUser(user);  
        } catch (Exception e) {  
            log.error("", e);  
            return optionalUser;  
        }  
        outboxRepository.delete(outbox);  
    }  
  
    return optionalUser;  
}
```

Пример

- <https://github.com/dimarudik/multi-datasource>
- Oracle & PostgreSQL in docker

The screenshot shows a GitHub README file with several code snippets:

- `git clone https://github.com/dimarudik/multi-datasource.git`
`cd ./multi-datasource`
- `./dockerfiles/buildContainerImage.sh -x -v 21.3.0`
- `docker run --name oracle \`
`-p 1521:1521 -p 5500:5500 \`
`-e ORACLE_PWD=oracle_4U \`
`-v ./dockerfiles/scripts:/docker-entrypoint-initdb.d \`
`-d oracle/database:21.3.0-xe`
- `sqlplus test/test@(description=(address=(host=localhost)(protocol=tcp)(port=1521))(connect_dat`
- `docker run --name postgres \`
`-e POSTGRES_USER=postgres \`
`-e POSTGRES_PASSWORD=postgres \`
`-e POSTGRES_DB=postgres \`
`-p 5432:5432 \`
`-v ./sql/init.sql:/docker-entrypoint-initdb.d/init.sql \`
`-v ./sql/.psqlrc:/var/lib/postgresql/.psqlrc \`
`-d postgres \`
`-c shared_preload_libraries="pg_stat_statements,auto_explain" \`
`-c max_connections=200 \`
`-c logging_collector=on \`
`-c log_directory=pg_log \`
`-c log_filename=%u_%a.log \`
`-c log_min_duration_statement=3 \`
`-c log_statement=all \`
`-c auto_explain.log_min_duration=0 \`
`-c auto_explain.log_analyze=true`

**Миграция с даунтаймом до 45
минут**

Когда может понадобиться миграция с даунтаймом без синхронизации данных?

- При выезде нескольких таблиц из коммунальной базы в качестве нового сервиса
- Когда объем мигрируемых данных на источнике не превышает ~ 200GB
- Когда мы можем позволить себе даунтайм в пределах 45 минут (время ограничено)

Типовой сценарий для миграции

```
[  
  
{  
    "fromSchemaName" : "TEST",  
    "fromTableName" : "TABLE1",  
    "toSchemaName" : "PUBLIC",  
    "toTableName" : "TABLE1",  
    "fetchHintClause" : "/*+ no_index(TABLE1) */",  
    "fromTaskName" : "TABLE1_TASK",  
    "columnToColumn" : {  
        "id" : "id",  
        "name" : "name",  
        "create_at" : "create_at",  
        "update_at" : "update_at",  
        "gender" : "gender",  
        "byteablob" : "byteablob",  
        "textblob" : "textblob"  
    }  
}  
]  

```

ID	NAME	VALUE
1	A	10
2	B	2
3	C	53
4	D	4678
5	E	25
6	F	247
7	G	4
8	H	235
...
100000	J	12
100001	K	4
100002	L	90
100003	M	32
100004	N	894
100005	O	
100006	P	3

Общий подход для переливки данных

- Длинный запрос на Oracle, скорее всего, приведет к ORA-01555 Snapshot Too Old
- Табличку надо разделить на порции (чанки) для переливки
- Переливку осуществлять по чанкам размером ~ 100k строк (зависит от длины строки и наличия больших объектов LOB, bytea)
- При делении на чанки по первичному ключу доступ к таблице будет не самым эффективным
- Наиболее эффективным способом будет деление таблицы на чанки по внутреннему идентификатору строки. Oracle – ROWID; PostgreSQL – CTID
- Переливать чанки без «приземления» в файл
- <https://jdbc.postgresql.org/documentation/publicapi/org/postgresql/copy/CopyManager.html>
- На приемнике необходимо убрать все индексы
- Добавление данных производить с помощью COPY в бинарном формате

Делим табличку на чанки в Oracle

- https://docs.oracle.com/en/database/oracle/oracle-database/19/arpls/DBMS_PARALLEL_EXECUTE.html#GUID-D13B6975-09B5-4711-AD43-45F68228C1CC

```
begin
    dbms_parallel_execute.create_chunks_by_rowid ( task_name      => 'TABLE1_TASK',
                                                table_owner   => 'TEST',
                                                table_name    => 'TABLE1',
                                                by_row        => TRUE,
                                                chunk_size    => 100000 );
end;
/
```

OOOOOOFFFBBBBBBRRR

Object ID
4 bytes
Datafile
Number
1.5 Bytes
Block
Number
2.5 Bytes
Row
Number
2 Bytes

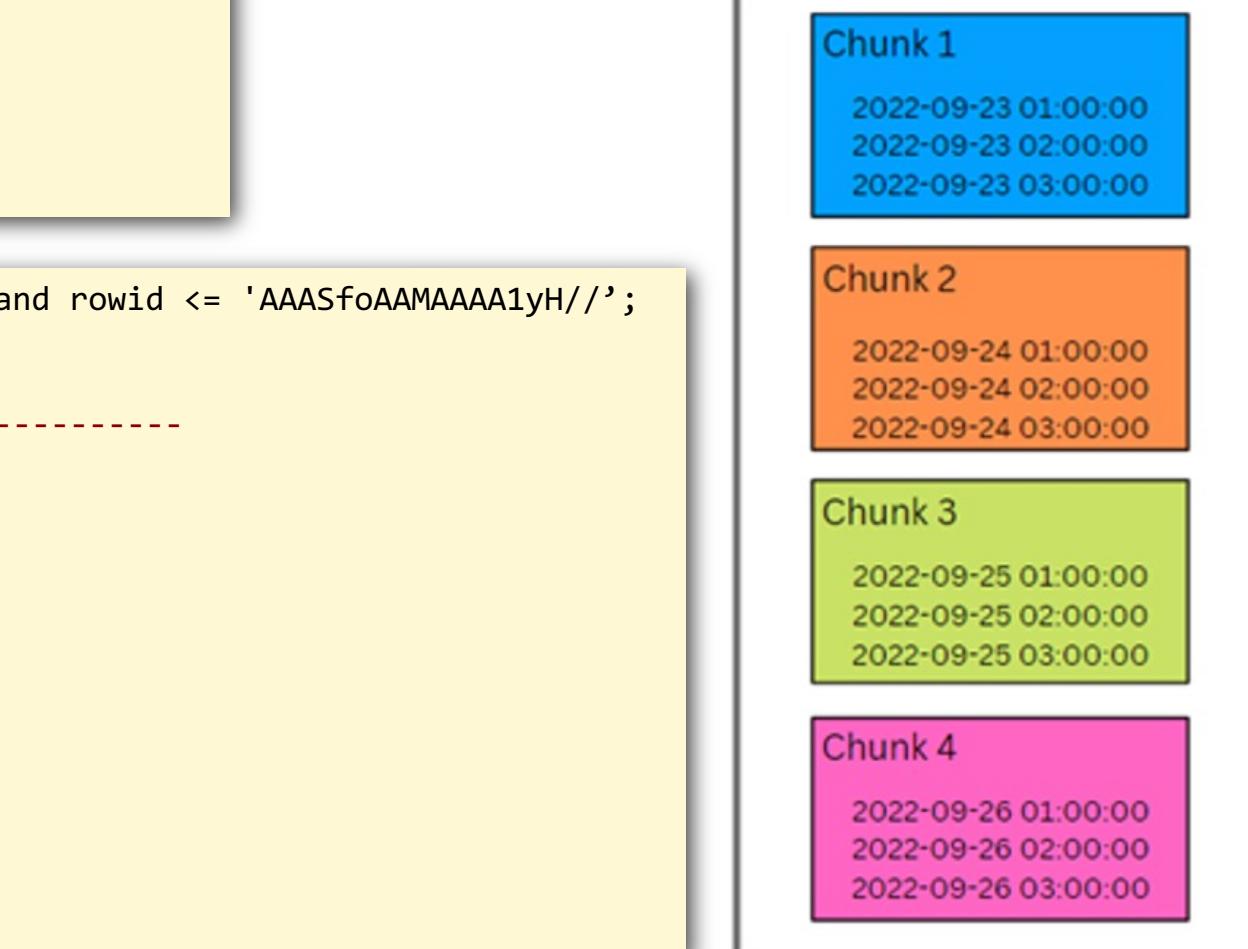
```
TEST@xe>select * from user_parallel_execute_chunks;
```

CHUNK_ID	TASK_NAME	STATUS	START_ROWID	END_ROWID
1	TABLE1_TASK	UNASSIGNED	AAASfoAAMAAAACAAAA	AAASfoAAMAAA1yH//
2	TABLE1_TASK	UNASSIGNED	AAASfoAAMAAA1zAAA	AAASfoAAMAAAB1lH//

```
TEST@xe>select rowid, t.id, t.name, t.gender, t.textclob from test.table1 t where rowid >= 'AAASfoAAMAAAACAAAA' and rowid <= 'AAASfoAAMAAA1yH//';
```

ROWID	ID	NAME	GENDER	TEXTCLOB
AAASfoAAMAAAACDAAA	1	Hi, I'm using varchar2 to varchar	1	Hi, I'm using CLOB to text
AAASfoAAMAAAACDAAB	2	Hi, I'm using varchar2 to varchar	0	Hi, I'm using CLOB to text
AAASfoAAMAAAACDAAC	3	Hi, I'm using varchar2 to varchar	1	Hi, I'm using CLOB to text
AAASfoAAMAAAACDAAD	4	Hi, I'm using varchar2 to varchar	0	Hi, I'm using CLOB to text
AAASfoAAMAAAACDAAE	5	Hi, I'm using varchar2 to varchar	1	Hi, I'm using CLOB to text
AAASfoAAMAAAACDAAF	6	Hi, I'm using varchar2 to varchar	0	Hi, I'm using CLOB to text
AAASfoAAMAAAACDAAG	7	Hi, I'm using varchar2 to varchar	1	Hi, I'm using CLOB to text
AAASfoAAMAAAACDAAH	8	Hi, I'm using varchar2 to varchar	0	Hi, I'm using CLOB to text
AAASfoAAMAAAACDAAI	9	Hi, I'm using varchar2 to varchar	1	Hi, I'm using CLOB to text
AAASfoAAMAAAACDAAJ	10	Hi, I'm using varchar2 to varchar	0	Hi, I'm using CLOB to text

10 rows selected.

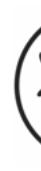


Делим табличку на чанки в PostgreSQL

```
postgres=# select ctid, t.id, t.name, t.gender, t.textclob from table1 t limit 10;  
  
ctid | id | name | gender | textclob  
-----+---+-----+-----+-----  
(0,1) | 1 | Hi, I'm using varchar2 to varchar | t | Hi, I'm using CLOB to text  
(0,2) | 2 | Hi, I'm using varchar2 to varchar | f | Hi, I'm using CLOB to text  
(0,3) | 3 | Hi, I'm using varchar2 to varchar | t | Hi, I'm using CLOB to text  
(0,4) | 4 | Hi, I'm using varchar2 to varchar | f | Hi, I'm using CLOB to text  
(0,5) | 5 | Hi, I'm using varchar2 to varchar | t | Hi, I'm using CLOB to text  
(0,6) | 6 | Hi, I'm using varchar2 to varchar | f | Hi, I'm using CLOB to text  
(0,7) | 7 | Hi, I'm using varchar2 to varchar | t | Hi, I'm using CLOB to text  
(0,8) | 8 | Hi, I'm using varchar2 to varchar | f | Hi, I'm using CLOB to text  
(0,9) | 9 | Hi, I'm using varchar2 to varchar | t | Hi, I'm using CLOB to text  
(0,10) | 10 | Hi, I'm using varchar2 to varchar | f | Hi, I'm using CLOB to text  
  
(10 rows)
```

- **reltuples** – число всех строк в таблице
- **relopages** – количество страниц в таблице
- **reltuples / 100000 = chunks** (количество чанков)
- **relopages / chunks = pages in chunk**

```
postgres=# select ctid, t.id, t.name, t.gender, t.textclob from table1 t where ctid between '(0,2)' and '(0,7)';  
  
ctid | id | name | gender | textclob  
-----+---+-----+-----+-----  
(0,2) | 2 | Hi, I'm using varchar2 to varchar | f | Hi, I'm using CLOB to text  
(0,3) | 3 | Hi, I'm using varchar2 to varchar | t | Hi, I'm using CLOB to text  
(0,4) | 4 | Hi, I'm using varchar2 to varchar | f | Hi, I'm using CLOB to text  
(0,5) | 5 | Hi, I'm using varchar2 to varchar | t | Hi, I'm using CLOB to text  
(0,6) | 6 | Hi, I'm using varchar2 to varchar | f | Hi, I'm using CLOB to text  
(0,7) | 7 | Hi, I'm using varchar2 to varchar | t | Hi, I'm using CLOB to text  
  
(6 rows)  
  
postgres=# explain (analyze, buffers) select ctid, t.id, t.name, t.gender, t.textclob from table1 t where ctid between '(0,2)' and '(0,7)';  
QUERY PLAN  
-----  
Tid Range Scan on table1 t (cost=0.01..4.07 rows=6 width=76) (actual time=0.010..0.012 rows=6 loops=1)  
TID Cond: ((ctid >= '(0,2)::tid) AND (ctid <= '(0,7)::tid))  
Buffers: shared hit=1  
Planning Time: 0.062 ms  
Execution Time: 0.118 ms  
  
(5 rows)
```



«БулиК» - утилита для переноса данных

- Oracle → PostgreSQL
- PostgreSQL → PostgreSQL

```
threadCount: 10

fromProperties:
  url: jdbc:oracle:thin:@(description=(address=(host=localhost)(protocol=tcp)(port=1521))(connect_data=(service_name=xepdb1)))
  user: test
  password: test
toProperties:
  url: jdbc:postgresql://localhost:5432/postgres
  user: test
  password: test
```

```
[
  {
    "fromSchemaName" : "TEST",
    "fromTableName" : "TABLE1",
    "toSchemaName" : "PUBLIC",
    "toTableName" : "TABLE1",
    "fetchHintClause" : "+ no_index(TABLE1) /*",
    "fetchWhereClause" : "1 = 1",
    "fromTaskName" : "TABLE1_TASK",
    "columnToColumn" : {
      ...
    }
  }
]
```

<https://github.com/dimarudik/bublik>



README



Tool for Data Transfer from Oracle to PostgreSQL or from PostgreSQL to PostgreSQL

This tool facilitates the efficient transfer of data from Oracle to PostgreSQL or from PostgreSQL to PostgreSQL.

The quickest method for extracting data from Oracle is by using `ROWID` (employing `dbms_parallel_execute` to segment the data into chunks). In case of PostgreSQL, we should split a table into chunks by CTID.

As you know, the fastest way to input data into PostgreSQL is through the `COPY` command in binary format.

Supported Types

ORACLE	Postgresql (possible types)
char, varchar, varchar2	char, bpchar, varchar, text
CLOB	varchar, text
BLOB	bytea
date	date, timestamp, timestamptz
timestamp	timestamp, timestamptz
timestamp with time zone	timestamptz
number	numeric, smallint, bigint, integer, double precision

1. Oracle To PostgreSQL

The objective is to migrate table `TABLE1` from an Oracle schema `TEST` to a PostgreSQL database.

Step 1

- Prepare Oracle environment in docker for testing

Миграция с даунтаймом на cutover

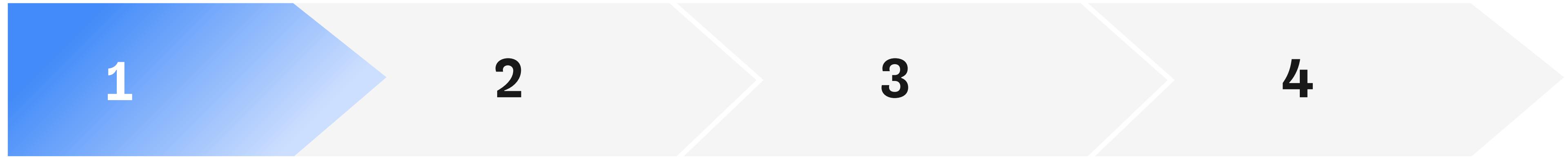
Когда может понадобиться миграция с даунтаймом на cutover?

- Когда объем мигрируемых данных на источнике превышает 200GB
- Когда необходимо перенести данные без остановки приложения. Даунтайм необходим будет в момент переключения приложения на СУБД PostgreSQL
- Когда необходимо синхронизировать данные с момента создания копии до момента переключения

Архитектура GoldenGate



Что необходимо сделать



Подготовка

Добавление
экстрактора в БД Oracle
Подготовка целевой БД

Заливка данных

Начальная заливка
данных из Oracle в
PostgreSQL

Синхронизация

Добавление репликата
Настройка
синхронизации

Переключение

Шаг 1. Создание экстрактора и подготовка целевой БД PostgreSQL

- Oracle DBA создают новый экстрактор. (момент времени T1)
- В целевой БД необходимо создать структуру
- Учесть следующие моменты:
 - Построить только primary key или unique index (после начальной заливки данных)
 - Отключить триггеры
 - Удалить Foreign Key

Подводные камни - Триггеры и Foreign Key



insert into tableA

триггер: insert into tableB



insert into tableA
insert into tableB

insert into tableA
insert into tableB

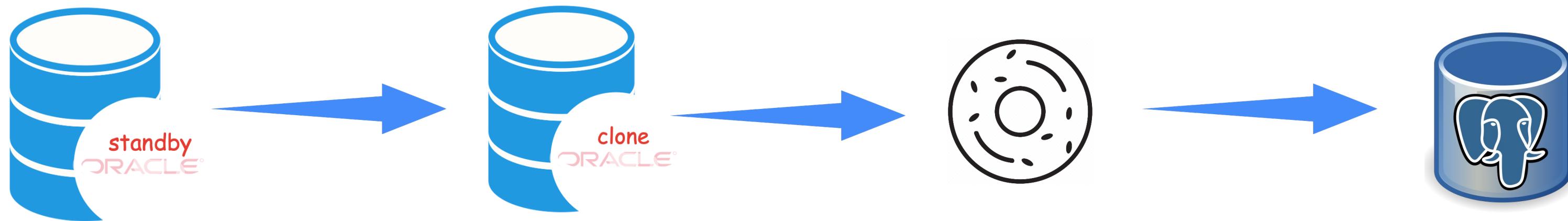


insert into tableA
триггер: insert into tableB
insert into tableB



DML выполнится дважды

Шаг 2. Заливка данных



- Oracle DBA создают клон. (момент времени $T_2 > T_1$)
- Утилитой «Бублик» выполняется заливка данных в БД PostgreSQL. Данные синхронизированы на момент времени T_2 .

Шаг 3. Синхронизация

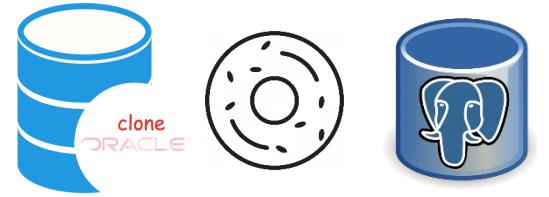
T1

Запуск
экстрактора



T2

Начальная
загрузка



T3

Запуск
репликата



Репликат запускается
с момента времени
=T2

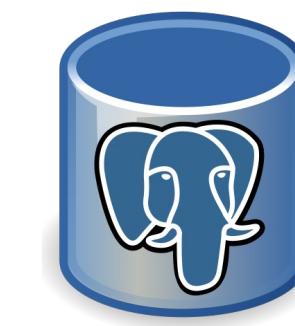
T4

Синхронная
копия



T5

Построить
необходимые
индексы



Шаг 4. Переключение

- Включаются необходимые триггеры
- Достраиваются Foreign Key с опцией NOT VALID

Ограничения

- Отсутствует репликация DDL.
- Размер реплицируемого LOB-а не может превышать 1GB – ограничение PostgreSQL (bytea, text).
- Оптимальный сценарий OLTP нагрузка – маленькие транзакции.

Возможные сценарии

- Репликация с фильтрацией
- Репликация с разрешением конфликтов
- Параллельная репликация в несколько БД

Репликация с фильтрацией

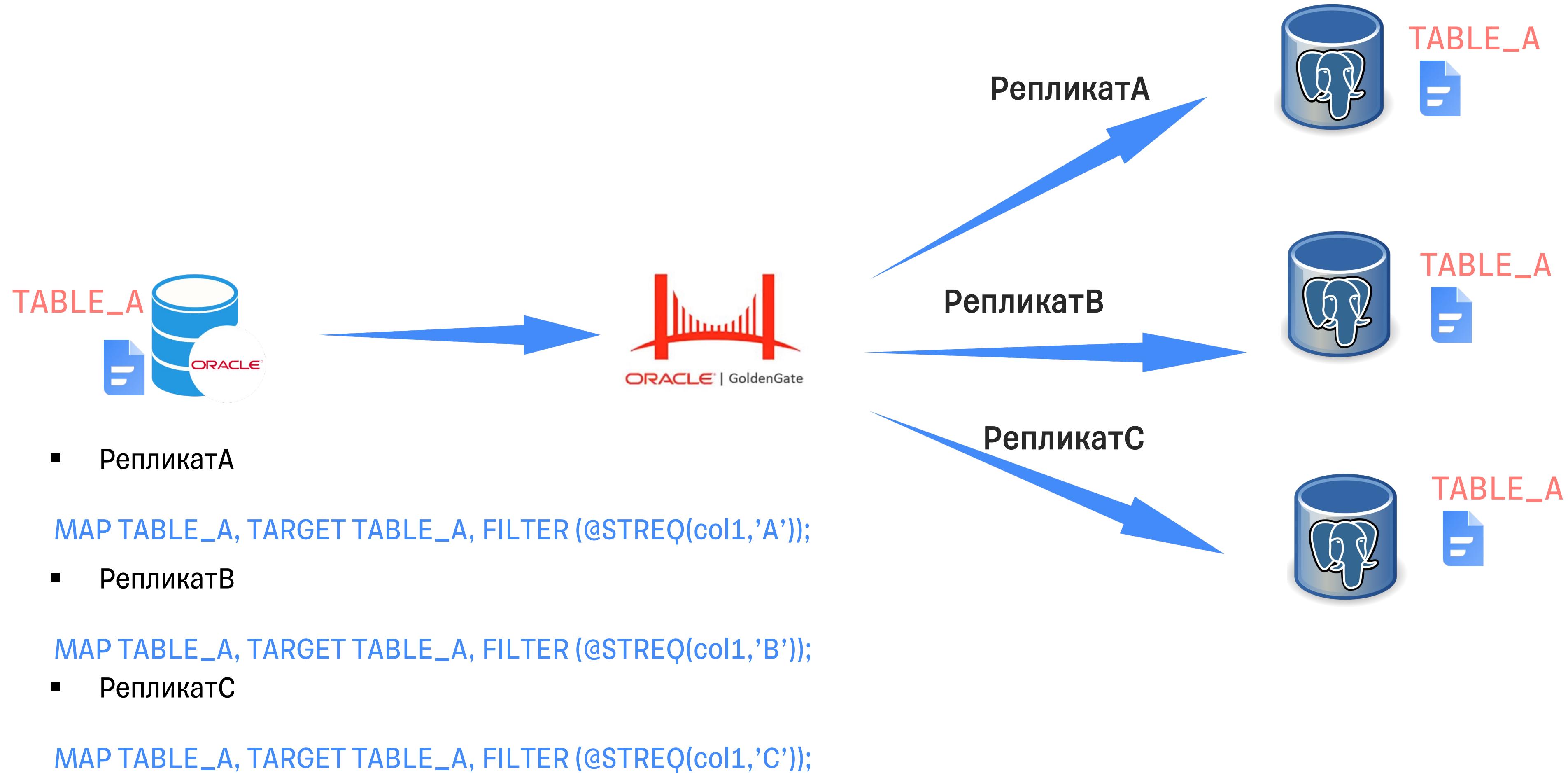
- фильтры можно включать как в экстракторе, так и в репликате

```
TABLE TCTDBS.GLOBALPARAMETERVALUES, WHERE (PRODUCTSERNO <> 0);
```

Репликация с разрешением конфликтов

- При частичной первоначальной заливке данных в целевую БД PostgreSQL в процессе репликации можно получить ошибку **NO DATA FOUND**, если транзакция попытается выполнить dml над данными, которых нет в БД.
- Разрешается путем добавления параметра **INSERTMISSINGUPDATES** в процесс репликации, таким образом отсутствующая строка будет добавлена в целевую БД.
- **Таблица не должна содержать LOB. Также появляются накладные расходы на логирование изменений в источнике.**

Параллельная репликация в несколько БД



SLA

- Не можем позволить длительную репликацию
- Репликация «от начала и до конца» не должна быть дольше недели.

ТИНЬКОФФ

Спасибо!

