# Asynchroniczna replikacja dwukierunkowa SZBD Postgres

Replikacja z wykorzystaniem materializowanych perspektyw oraz replikacja strumieniowa jest replikacją typu *single-master*. W takiej replikacji zmiany wprowadzane tylko do bazy danych *master* są replikowane do pozostałych baz danych. Pozostałe bazy danych pracują w trybie tylko do odczytu lub wprowadzane do nich zmiany nie są propagowane do bazy danych *master*. Niniejszy tutorial poświęcony jest replikacji *multi-master* w środowisku systemu Postgresql. Replikacja *multi-master* umożliwia propagowanie zmian wprowadzanych we wszystkich replikach danych.

Celem zajęć jest zapoznanie się z dwukierunkowa replikacją udostępnianą przez odgałęzienie (ang. fork) projektu *Postgresql* o nazwie [Bi-Directional Replication for PostgreSQL](https://www.2ndquadrant.com/en/resources/bdr/) (Postgres-BDR lub BDR). Odgałęzienie to jest rozwijane w formie otwartego kodu przez firmę [2ndQuadrant](https://2ndquadrant.com/). Najnowsza stabilna wersja bazuje na *Posgresql* w wersji 9.4. Rozwiązanie to analogicznie do replikacji strumieniowej korzysta z informacji zapisywanych w dzienniku bazy danych.

## Przygotowanie środowiska

1. Zaloguj się do maszyny wirtualnej jako użytkownik postgres używając hasła RBD#7102.
2. Otwórz okno terminala, który nazwiemy terminalem pomocniczym.
3. W pierwszym kroku pobierz plik manifestów za pomocą poniższego polecenia:

wget www.cs.put.poznan.pl/jjezierski/RBDv2/bdr.yaml

1. Otwórz plik manifestów w celu jego przeglądnięcia za pomocą polecenia:

less bdr.yaml

Zwróć uwagę, że manifesty są bardzo podobne do tych, które zostały wykorzystane do przygotowania węzłów roboczych klastra Citus.

1. Rozpocznij wdrożenie komponentów z pliku manifestów za pomocą polecenia:

kubectl apply -f bdr.yaml

1. Obserwuj postęp wdrożenia *StatefulSet* wykorzystując poniższe polecenie:

kubectl get sts --watch

1. Skopiuj katalog *loggers* do katalogu */data* replik pgsql-bdr-bdr-0 i pgsql-bdr-bdr-1. Użyj poniższych poleceń:

kubectl cp loggers pgsql-bdr-sts-0:/data

kubectl cp loggers pgsql-bdr-sts-1:/data

1. Otwórz dwie kolejne zakładki w oknie terminala wybierając z menu *File* pozycję *New Tab*. Nazwij te zakładki nazwami kolejnych replik od pgsql-rbd-sts-0 do pgsql-rbd-sts-1, wykorzystaj w tym celu pozycję *Set Title* z menu *Terminal*. W terminalach pgsql-bdr-sts-0 i pgsql-bdr-sts-1 będziemy wykonywać operacje na bazach danych uruchomionych w replikach pgsql-bdr-bdr-0 1 i pgsql-bdr-bdr-1,które dalej będziemy odpowiednio nazywać bdr0 i bdr1.
2. W terminalu pgsql-bdr-sts-0 wykonaj poniższe polecenie aby uruchomić powłokę w replice pgsql-bdr-sts-0. Replika ta będzie obsługiwać bazę danych, którą będziemy dalej nazywać bdr0.

kubectl exec -it pgsql-bdr-sts-0 -- /bin/bash

1. W tym samym terminalu uruchom narzędzie *psql* w celu przyłączenia się do bazy danych bdr0, wykorzystaj następujące polecenie:

psql -U postgres

1. Analogicznie jak punkcie 10, w zakładce pgsql-bdr-sts-1 uruchom powłokę repliki pgsql-bdr-sts-1.
2. Tak samo jak punkcie 10, w zakładkach pgsql-bdr-sts-1 uruchom narzędzie psql.
3. W bazie danych *bdr0* utwórz rozszerzenie *bdr*:

CREATE EXTENSION btree\_gist;

CREATE EXTENSION bdr;

1. Powtórz poprzedni krok w bazie *bdr1*.

## Replikacja dwukierunkowa

1. W bazie danych *bdr0* utwórz grupę replikacji:

SELECT bdr.bdr\_group\_create(

local\_node\_name := 'bdr0\_node',

node\_external\_dsn

:= 'port=5432 dbname=postgres host=pgsql-bdr-sts-0.pgsql-rbd-bdr user=postgres password=rbd-bdr'

);

1. W bazie danych *bdr0* włącz oczekiwanie na dołączenie drugiej bazy danych*:*

SELECT bdr.bdr\_node\_join\_wait\_for\_ready();

1. W bazie danych *bdr1* dołącz do grupy replikacji.

SELECT bdr.bdr\_group\_join(

local\_node\_name := 'bdr1\_node',

node\_external\_dsn

:= 'port=5432 dbname=postgres host=pgsql-bdr-sts-1.pgsql-rbd-bdr user=postgres password=rbd-bdr',

join\_using\_dsn

:= 'port=5432 dbname=postgres host=pgsql-bdr-sts-0.pgsql-rbd-bdr user=postgres password=rbd-bdr ');

1. W bazie danych *bdr1* włącz oczekiwanie na drugiej bazy danych*:*

SELECT bdr.bdr\_node\_join\_wait\_for\_ready();

1. W bazie danych *bdr0* sprawdź status baz danych znajdujących się w grupie replikacji:

select node\_name,

case node\_status when 'r' then 'ready' when 'k' then 'killed/removed' when 'i' then 'init' end

as status, node\_read\_only from bdr.bdr\_nodes;

1. W bazie danych *bdr0* uruchom skrypt /data/loggers/loggers.sql:

\i /data/loggers/loggers.sql

1. W bazie danych *bdr1* sprawdź dostępne tabele: [Raport]

\dt

1. W bazie danych *bdr1* załaduj dane do utworzonych tabel:

\i /data/loggers/organizations.dmp

\i /data/loggers/loggers.dmp

\i /data/loggers/measurements.dmp

1. W bazie danych *bdr0* sprawdź zawartość tabeli *organizations*.
2. W bazie danych *bdr0* zmień nazwę organizacji o identyfikatorze 50 na *Replikowana org1*.
3. W bazie danych *bdr1* sprawdź nazwę organizacji o identyfikatorze 50. [Raport]
4. W bazie danych *bdr1* zmień nazwę organizacji o identyfikatorze 50 na *Replikowana org2*.
5. W bazie danych *bdr1* sprawdź nazwę organizacji o identyfikatorze 50. [Raport]

## Konflikt typu *insert* vs *insert*

Replikacja asynchroniczna może prowadzić do konfliktów jeżeli dane są współbieżnie modyfikowane w różnych bazach danych. Poniższe ćwiczenia mają na celu zilustrowanie takich konfliktów.

1. W bazie danych *bdr0* wstrzymaj aplikowanie zmian z replik:

select bdr.bdr\_apply\_pause();

1. W bazie danych *bdr1* wstrzymaj aplikowanie zmian z replik:

select bdr.bdr\_apply\_pause();

1. W bazie danych *bdr0* utwórz organizację *Replikowana org3* o identyfikatorze -10.
2. W bazie danych *bdr1* utwórz organizację *Replikowana org4* o identyfikatorze -10.
3. W bazie danych *bdr0* wznów aplikowanie zmian z replik:

select bdr.bdr\_apply\_resume();

1. W bazie danych *bdr1* wznów aplikowanie zmian z replik:

select bdr.bdr\_apply\_resume ();

1. W bazie danych *bdr0* sprawdź nazwę organizacji o identyfikatorze -10. [Raport]
2. W bazie danych *bdr1* sprawdź nazwę organizacji o identyfikatorze -10. [Raport]

## Konflikt typu *update* vs *update*

1. W bazie danych *bdr0* wstrzymaj aplikowanie zmian z replik:

select bdr.bdr\_apply\_pause();

1. W bazie danych *bdr1* wstrzymaj aplikowanie zmian z replik:

select bdr.bdr\_apply\_pause();

1. W bazie danych *bdr0*  rozpocznij poniższą transakcję:

begin;

select me\_temperature from measurements where me\_id=203308395 for update;

update measurements set me\_temperature=me\_temperature+1 where me\_id=203308395;

1. W bazie danych *bdr1*  rozpocznij poniższą transakcję:

begin;

select me\_temperature from measurements where me\_id=203308395 for update;

update measurements set me\_temperature=me\_temperature+2 where me\_id=203308395;

1. W bazie danych *bdr0* zatwierdź transakcję.
2. W bazie danych *bdr1* zatwierdź transakcję.
3. W bazie danych *bdr0* wznów aplikowanie zmian z replik:

select bdr.bdr\_apply\_resume();

1. W bazie danych *bdr1* wznów aplikowanie zmian z replik:

select bdr.bdr\_apply\_resume();

1. W bazie danych *bdr0* odczytaj zmodyfikowany pomiar: [Raport]

select me\_temperature from measurements where me\_id=203308395;

1. W bazie danych *bdr1* odczytaj zmodyfikowany pomiar: [Raport]

select me\_temperature from measurements where me\_id=203308395;

## Obsługa konfliktu

1. Otwórz terminal pomocniczy.
2. W terminalu pomocniczym pobierz źródło procedury *measurements\_conflict\_handler\_upd\_upd* i zapoznaj się z nim:

curl \  
https://www.cs.put.poznan.pl/jjezierski/RBDv2/measurements\_conflict\_handler\_upd\_upd.sql \

-o ~/measurements\_conflict\_handler\_upd\_upd.sql

kubectl cp ~/measurements\_conflict\_handler\_upd\_upd.sql pgsql-bdr-sts-0:/data

1. W bazie danych *bdr0* utwórz procedurę *measurements\_conflict\_handler\_upd\_upd*:

\i /data/measurements\_conflict\_handler\_upd\_upd.sql

1. W bazie danych *bdr0* wykorzystaj procedurę *measurements\_conflict\_handler\_upd\_upd* do obsługi konfliktu *update vs update* na tabeli *measurements*:

select \* from bdr.bdr\_create\_conflict\_handler(

ch\_rel := 'measurements',

ch\_name := 'measurements\_upd\_upd\_handler',

ch\_proc := 'public.measurements\_conflict\_handler\_upd\_upd(public.measurements, public.measurements, text, regclass, bdr.bdr\_conflict\_type)',

ch\_type := 'update\_update');

1. Sprawdź czy w bazie danych *bdr1* należy powtórzyć kroki 3 i 4. [Raport]
2. Powtórz kroki z ćwiczenia 4 "Konflikt typu update vs update".
3. Na czym polega rozwiązanie tego konfliktu? [Raport]

## Niedostępność bazy danych z grupy replikacji

* 1. W celu wyłączenia bazy danych *bdr1* zmniejsz liczbę replik do jeden, w tym celu w terminalu pomocniczym wykonaj poniższe polecenie:

kubectl scale sts pgsql-bdr-sts --replicas 1

* 1. W bazie danych *bdr0* zmień nazwę organizacji z identyfikatorem -10 na wartość ChangedOnBdr1. Czy operacja się powiodła? [Raport]
  2. Dodaj nową kolumnę do tabeli *organizations*, w tym celu w bazie danych *bdr0* wykonaj następujące polecenie:

alter table organizations add column location varchar(50);

Co się stało? [Raport]

* 1. W celu włączenia bazy danych *bdr1* zwiększ liczbę replik do dwóch, w tym celu w terminalu pomocniczym wykonaj poniższe polecenie:

kubectl scale sts pgsql-bdr-sts --replicas 2

* 1. W zakładce pgsql-bdr-sts-1 uruchom ponownie powłokę repliki pgsql-bdr-sts-1. Przyłącz się w niej ponownie do bazy danych *bdr1.*
  2. Sprawdź czy zmiana nazwy organizacji z identyfikatorem -10 zreplikowała się do bazy danych *bdr1.* [Raport]
  3. W zakładce pgsql-bdr-sts-1 sprawdź jaki jest status wykonania polecenia ALTER TABLE. [Raport]
  4. W bazie danych *bdr1* sprawdź schemat tabeli *organizations*. [Raport]

## Replikacja operacji DDL

Replikacja operacji DDL na obiektach znajdujących się w grupie replikacji jest wykonywana automatycznie. Wyjątkiem są operacje, które dotyczą całej instancji bazy danych, przykładowo utworzenie użytkownika. W takim przypadku należy użyć funkcji *bdr.bdr\_replicate\_ddl\_command*.

1. W bazie danych *bdr0* utwórz użytkownika RBD.

SELECT bdr.bdr\_replicate\_ddl\_command('CREATE USER RBD;');

1. W bazie danych bdr1 sprawdź istnienie użytkownika RBD.

\du RBD