V. 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 (Postgres-BDR lub BDR). Odgałęzienie to jest rozwijane w formie otwartego kodu przez firmę 2ndQuadrant. Najnowsza stabilna wersja bazuje na *Posgresql* w wersji 9.4. Rozwiązanie to analogicznie do replikacji strumieniowej korzysta z informacji zapisywanych w dzienniku bazy danych.

1. 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
3. 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.
4. Rozpocznij wdrożenie komponentów z pliku manifestów za pomocą polecenia:
   kubectl apply -f bdr.yaml
5. Obserwuj postęp wdrożenia *StatefulSet* wykorzystując poniższe polecenie:
   kubectl get sts --watch
7. 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
```

8. 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`.
9. 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
```

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

```
psql -U postgres
```

11. Analogicznie jak punkcie 10, w zakładce `pgsql-bdr-sts-1` uruchom powłokę repliki `pgsql-bdr-sts-1`.
12. Tak samo jak punkcie 10, w zakładkach `pgsql-bdr-sts-1` uruchom narzędzie psql.
13. W bazie danych *bdr0* utwórz rozszerzenie *bdr*:
    ```
    CREATE EXTENSION btree_gist;
    CREATE EXTENSION bdr;
    ```
14. Powtórz poprzedni krok w bazie *bdr1*.

2. 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'
   );
   ```
2. W bazie danych *bdr0* włącz oczekiwanie na dołączenie drugiej bazy danych*:*
   ```
   SELECT bdr.bdr_node_join_wait_for_ready();
   ```
3. 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 ');
   ```
4. W bazie danych *bdr1* włącz oczekiwanie na drugiej bazy danych*:*
   ```
   SELECT bdr.bdr_node_join_wait_for_ready();
   ```
5. 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;
   ```
6. W bazie danych *bdr0* uruchom skrypt /data/loggers/loggers.sql:
   ```
   \i /data/loggers/loggers.sql
   ```
7. W bazie danych *bdr1* sprawdź dostępne tabele: [Raport]
   ```
   \dt
   ```
8. W bazie danych *bdr1* załaduj dane do utworzonych tabel:
   ```
   \i /data/loggers/organizations.dmp
   \i /data/loggers/loggers.dmp
   \i /data/loggers/measurements.dmp
   ```
9. W bazie danych *bdr0* sprawdź zawartość tabeli *organizations*.
10. W bazie danych *bdr0* zmień nazwę organizacji o identyfikatorze 50 na *Replikowana org1*.
11. W bazie danych *bdr1* sprawdź nazwę organizacji o identyfikatorze 50. [Raport]
12. W bazie danych *bdr1* zmień nazwę organizacji o identyfikatorze 50 na *Replikowana org2*.
13. W bazie danych *bdr1* sprawdź nazwę organizacji o identyfikatorze 50. [Raport]

3. 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();
2. W bazie danych *bdr1* wstrzymaj aplikowanie zmian z replik:
   select bdr.bdr_apply_pause();
3. W bazie danych *bdr0* utwórz organizację *Replikowana org3* o identyfikatorze -10.
4. W bazie danych *bdr1* utwórz organizację *Replikowana org4* o identyfikatorze -10.
5. W bazie danych *bdr0* wznów aplikowanie zmian z replik:
   select bdr.bdr_apply_resume();
6. W bazie danych *bdr1* wznów aplikowanie zmian z replik:
   select bdr.bdr_apply_resume ();
7. W bazie danych *bdr0* sprawdź nazwę organizacji o identyfikatorze -10. [Raport]
8. W bazie danych *bdr1* sprawdź nazwę organizacji o identyfikatorze -10. [Raport]

4. Konflikt typu *update* vs *update*

1. W bazie danych *bdr0* wstrzymaj aplikowanie zmian z replik:
   select bdr.bdr_apply_pause();
2. W bazie danych *bdr1* wstrzymaj aplikowanie zmian z replik:
   select bdr.bdr_apply_pause();
3. 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;
4. 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;
5. W bazie danych *bdr0* zatwierdź transakcję.
6. W bazie danych *bdr1* zatwierdź transakcję.
7. W bazie danych *bdr0* wznów aplikowanie zmian z replik:
   select bdr.bdr_apply_resume();
8. W bazie danych *bdr1* wznów aplikowanie zmian z replik:
   select bdr.bdr_apply_resume();
9. W bazie danych *bdr0* odczytaj zmodyfikowany pomiar: [Raport]
   select me_temperature from measurements where me_id=203308395;
10. W bazie danych *bdr1* odczytaj zmodyfikowany pomiar: [Raport]
    select me_temperature from measurements where me_id=203308395;

5. 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
```

3. W bazie danych *bdr0* utwórz procedurę *measurements_conflict_handler_upd_upd*:
   ```
   \i /data/measurements_conflict_handler_upd_upd.sql
   ```
4. 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');
   ```
5. Sprawdź czy w bazie danych *bdr1* należy powtórzyć kroki 3 i 4. [Raport]
6. Powtórz kroki z ćwiczenia 4 "Konflikt typu *update* vs *update*".
7. Na czym polega rozwiązanie tego konfliktu? [Raport]

6. Niedostępność bazy danych z grupy replikacji

6.1. 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
```

6.2. W bazie danych *bdr0* zmień nazwę organizacji z identyfikatorem -10 na wartość ChangedOnBdr1. Czy operacja się powiodła? [Raport]

6.3. 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]

6.4. 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
```

6.5. W zakładce `pgsql-bdr-sts-1` uruchom ponownie powłokę repliki `pgsql-bdr-sts-1`. Przyłącz się w niej ponownie do bazy danych *bdr1*.

6.6. Sprawdź czy zmiana nazwy organizacji z identyfikatorem -10 zreplikowała się do bazy danych *bdr1*. [Raport]

6.7. W zakładce `pgsql-bdr-sts-1` sprawdź jaki jest status wykonania polecenia ALTER TABLE. [Raport]

6.8. W bazie danych *bdr1* sprawdź schemat tabeli *organizations*. [Raport]

7. 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;');
   ```
2. W bazie danych bdr1 sprawdź istnienie użytkownika RBD.
   ```
   \du RBD
   ```