# VII. Przetwarzanie rozproszone w SZBD Oracle

Celem zajęć jest zapoznanie się z własnościami SZBD Oracle umożliwiającymi przetwarzanie rozproszone. Własności te obejmują:

* Obsługa rozproszonych zakleszczeń.
* Prosta replikacja z wykorzystaniem materializowanych perspektyw.

## Przygotowanie środowiska

1. Zaloguj się do maszyny wirtualnej jako użytkownik rbd używając hasła RBD#7102.
2. Jeśli to konieczne to odzyskaj środowisko pracy z poprzednich zajęć:
3. Otwórz okno terminala, który nazwiemy terminalem pomocniczym.
4. Otwórz dwie kolejne zakładki terminala o nazwach ora-rbd1-0 i ora-rbd2-0. Pierwszy terminal o nazwie *rbd1* będzie służył do operowania na bazie danych *rbd1*, natomiast drugi terminal o nazwie *rbd2* będzie wykorzystywany do wykonywania operacji na bazie danych *rbd2*.
5. W terminalu ora-rbd1-0 wykonaj poniższe polecenie aby uruchomić powłokę w replice ora-rbd1-0. Replika ta będzie obsługiwać bazę danych, którą będziemy dalej nazywać rbd1.

kubectl exec -it ora-rbd1-0 -- /bin/bash

1. W terminalu rbd1 przyłącz się za pomocą narzędzia sqlplus do bazy danych RBD1:

sqlplus rbd/RBD#7102@RBD1

1. Powtórz poprzednie 2 kroki dla bazy danych rbd2.

## Rozproszone zakleszczenie

Celem tego punktu jest zapoznanie się ze sposobem rozwiązywania rozproszonych zakleszczeń. Przypomnienie: tabela *pracownicy* znajduje się w bazie danych *rbd1*, natomiast tabela *zespoly* znajduje się w bazie danych *rbd2*.

1. W bazie danych *RBD1* i RBD2 sprawdź wartość parametru [*distributed\_lock\_timeout*](https://docs.oracle.com/cd/B19306_01/server.102/b14237/initparams063.htm#REFRN10050):

show parameter distributed\_lock\_timeout

1. W bazie danych *RBD1* rozpocznij rozproszoną transakcję T1 za pomocą poniższych poleceń SQL:

update pracownicy set placa\_pod=placa\_pod+10 where id\_prac=100;

1. W bazie danych *RBD2* rozpocznij rozproszoną transakcję T2 za pomocą poniższych poleceń SQL:

update zespoly set adres='PIOTROWO 88' where id\_zesp=10;

1. W bazie danych *RBD1* kontynuuj rozproszoną transakcję T1 za pomocą poniższego polecenia SQL:

update zespoly set adres='PIOTROWO 77' where id\_zesp=10;

1. W bazie danych *RBD2* kontynuuj rozproszoną transakcję T2 za pomocą poniższego polecenia SQL:

update pracownicy set placa\_pod=placa\_pod+10 where id\_prac=100;

1. Co się stało? Poczekaj jeszcze 1 minutę. Coś się zmieniło?
2. W jaki sposób należy dokończyć transakcje T1 i T2? [Raport]

## Prosta replikacja danych

Celem zadania jest zapoznanie się z prostym mechanizmem asynchronicznej replikacji typu master-slaves implementowanej za pomocą materializowanych perspektyw.

### Materializowana perspektywa z ręcznym odświeżaniem

1. W bazie danych *rbd1*, utwórz [materializowaną perspektywę](https://docs.oracle.com/database/121/SQLRF/statements_6002.htm#SQLRF01302) REP\_ZESPOLY, będącą repliką zdalnej tabeli ZESPOLY z bazy *rbd2*, odświeżaną w trybie pełnym (COMPLETE) bez automatycznego odświeżania (odświeżanie na żądanie).

create materialized view rep\_zespoly  
refresh complete  
as select \* from zespoly@rbd@rbd2;

1. W bazie danych *rbd1* odczytaj bieżącą zawartość migawki REP\_ZESPOLY.

select \* from rep\_zespoly;

1. W bazie danych *rbd1* odczytaj informacje o utworzonych przez siebie materializowanych perspektywach:

select mview\_name, refresh\_mode, query from user\_mviews;

1. W bazie danych *rbd2* zmodyfikuj zawartość tabeli *zespoly*.

update zespoly set adres='Berdychowo 1' where id\_zesp=50;

commit;

1. W bazie danych *rbd1* odśwież „ręcznie” materializowaną perspektywę REP\_ZESPOLY w trybie pełnym (COMPLETE). Wyświetl ponownie dane z migawki REP\_ZESPOLY.

exec [dbms\_mview.refresh](https://docs.oracle.com/cd/B28359_01/appdev.111/b28419/d_mview.htm#i997194)('REP\_ZESPOLY', 'C')

select \* from rep\_zespoly;

1. Co sądzisz o wydajności procesu pełnego odświeżania materializowanych perspektyw? [Raport]

### Przyrostowe odświeżanie materializowanej pespektywy

1. W bazie danych *rbd2* utwórz [dziennik](https://docs.oracle.com/database/121/SQLRF/statements_6003.htm#SQLRF01303) tabeli *zespoly* dla przyrostowego odświeżania materializowanych perspektyw.

create materialized view log on zespoly;

1. W bazie danych *rbd1* zmień sposób odświeżania materializowanych perspektyw na przyrostowy.

alter materialized view rep\_zespoly

refresh fast;

1. W bazie danych *rbd1* odśwież „ręcznie” materializowaną perspektywę REP\_ZESPOLY w trybie pełnym (COMPLETE). Jak sądzisz, dlaczego po utworzeniu dziennika jest potrzebne odświeżenie w trybie pełnym? [Raport]

exec [dbms\_mview.refresh](https://docs.oracle.com/cd/B28359_01/appdev.111/b28419/d_mview.htm#i997194)('REP\_ZESPOLY', 'C')

1. W bazie danych *rbd2* zmodyfikuj zawartość tabeli *zespoly*.

update zespoly set adres='Berdychowo 2' where id\_zesp=50;

commit;

1. W bazie danych *rbd1* odśwież „ręcznie” materializowaną perspektywę REP\_ZESPOLY w trybie przyrostowym (FAST). Wyświetl ponownie dane z migawki REP\_ZESPOLY.

exec dbms\_mview.refresh('REP\_ZESPOLY', 'F')

select \* from rep\_zespoly;

1. Czy przyrostowe odświeżanie materializowanych perspektyw wpływa na wydajność procesu odświeżania? [Raport]

### Materializowana perspektywa z odświeżaniem automatycznym

1. W bazie danych *RBD1* i RBD2 sprawdź wartość parametru [job\_queue\_processes](http://docs.oracle.com/cd/B19306_01/server.102/b14237/initparams089.htm#REFRN10077):

show parameter job\_queue\_processes

1. W bazie danych *rbd1* utwórz [dziennik](https://docs.oracle.com/database/121/SQLRF/statements_6003.htm#SQLRF01303) tabeli *pracownicy* dla przyrostowego odświeżania materializowanych perspektyw.
2. create materialized view log on pracownicy;
3. W bazie danych *rbd2* utwórz materializowaną perspektywę REP\_PLACE replikującą identyfikatory, nazwiska, płace podstawowe i dodatkowe ze zdalnej tabeli PRACOWNICY. Migawka powinna być odświeżana w trybie przyrostowym (FAST) automatycznie co 1 minutę, pierwsze wypełnienie danymi migawki ma zajść zaraz po jej utworzeniu.

create materialized view rep\_place

refresh fast

next sysdate + 1/(24\*60)

as

select id\_prac, nazwisko, placa\_pod, placa\_dod, etat

from pracownicy@rbd@rbd1;

1. W bazie danych *rbd2* wyświetl zawartość materializowanej perspektywy *rep\_place.*
2. W bazie danych *rbd1* zmodyfikuj zawartości tabeli *pracownicy*.

update pracownicy set placa\_pod=777 where id\_prac=150;  
commit;

1. W bazie danych *rbd2* wyświetl dane z migawki REP\_PLACE. Poczekaj 1 minutę i ponownie wyświetl dane migawki. Czy widzisz modyfikację płacy? [Raport]

### Grupy odświeżania

Grupy odświeżania służą do odświeżania grupy materializowanych perspektyw w jednej transakcji. Dzięki temu mechanizmowi mamy gwarancję, że zawartość materializowanych perspektyw jest wzajemnie spójna.

1. Pobierz skrypt SQL służący do tworzenia tabeli *etaty* wykonując w terminalu pomocniczym polecenie:

wget www.cs.put.poznan.pl/jjezierski/RBDv2/etaty.sql

1. W terminalu pomocniczym skopiuj plik etaty.sql do katalogu /opt/oracle/oradata repliki ora-rbd1-0. Użyj poniższego polecenia:

kubectl cp etaty.sql ora-rbd1-0:/opt/oracle/oradata

1. W bazie danych *rbd1* uruchom pobrany skrypt:

@ /opt/oracle/oradata /etaty.sql

1. W bazie danych *rbd1* utwórz dziennik materializowanej perspektywy dla tabeli  *etaty*:

create materialized view log on etaty;

1. W bazie danych *rbd2* utwórz materializowaną perspektywę dla zdalnej tabeli etaty:

create materialized view rep\_etaty  
refresh fast

as select \*

from etaty@rbd@rbd1;

1. W bazie danych *rbd2* utwórz grupę odświeżania o nazwie RG\_KADRY, zawierającą materializowane perspektywy REP\_ETATY i REP\_PLACE (pierwsze odświeżenie: natychmiast po utworzeniu, okres odświeżania: co 2 minuty, zmiana grupy odświeżania dla migawek już automatycznie odświeżanych). Zatwierdź transakcję.

exec DBMS\_REFRESH.MAKE (name => 'rg\_kadry', -

list => 'rep\_etaty, rep\_place', -

next\_date => sysdate, -

interval => 'sysdate + 1/(24\*30)', -

lax => true)

commit;

1. Sprawdź działanie grupy odświeżania RG\_KADRY. [Raport]