Integracja systemów transakcyjnych

Robert Wrembel
Politechnika Poznańska
Instytut Informatyki
`Robert.Wrembel@cs.put.poznan.pl`
`www.cs.put.poznan.pl/rwrembel`

Zaawansowana replikacja danych

- Replikacja Multimaster
- Replikacja Migawkowa 2-kierunkowa (updateable materialized views/snapshots)
- Techniki rozwiązywania konfliktów replikacji

Rodzaje replikacji

- **Multimaster**
  - **wszystkie węzły są równoważne**
  - **zmiana wprowadzona w jakimkolwiek węźle jest propagowana do pozostałych**
  - **synchroniczna lub asynchroniczna (domyślna)**
- **Migawkowa**
  - **węzeł nadrzędny - udostępnia dane do replikacji**
  - **węzeł migawkowy - replikuje dane z węzła nadrzędnego**
  - **asynchroniczna**
- **Hybrydowa**
  - **połączenie multimaster i migawkowej**

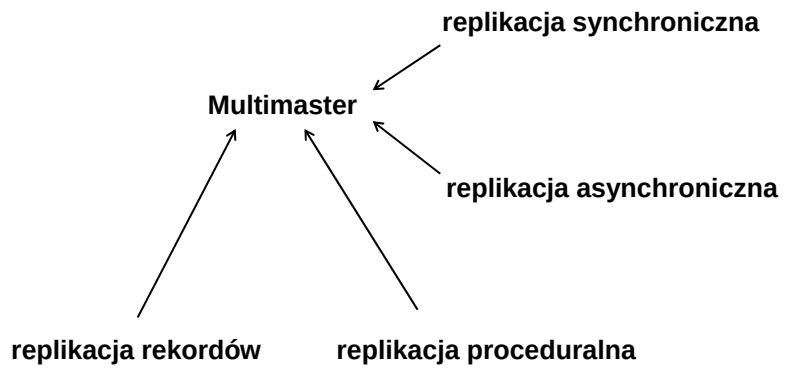
Narzędzia replikacji

- **Replication Management API**
  - **procedury i funkcje w pakietach systemowych**
- **Katalog (słownik) replikacji**
  - **metadane nt. zdefiniowanych architektur replikacji**
- **Replication Management Tool**
  - **program graficzny - moduł Enterprise Manager'a**

Definicje

- Grupa replikacji ⇨ podstawowy obiekt administracyjny
  - posiada unikalną nazwę
  - zawiera logicznie powiązane obiekty bazy danych z tego samego lub różnych schematów
  - jeden obiekt może należeć tylko do jednej grupy replikacji
- Węzeł, w którym grupa replikacji została zdefiniowana, staje się jej węzłem definicyjnym (definition site)
  - administrowanie grupą możliwe tylko w węźle definicyjnym
- Definicje grup replikacji są powielane pomiędzy węzłami
- Działanie jednej grupy jest niezależne od działania pozostałych grup (wyjątkiem jest replikacja proceduralna)

Replikacja Multimaster



Replikacja Multimaster

- Replikacja rekordów (row-level replication)
  - każda operacja DML na replice jest przesyłana do pozostałych replik
  - niska efektywność dla DML adresujących duże zbiory danych
- Replikacja proceduralna (procedural replication)
  - pomiędzy replikami są przesyłane wywołania procedur z odpowiednio skonstruowanych pakietów, które dokonują żądanych modyfikacji danych
  - do repliki są przekazywane wywołania wraz z odpowiednimi parametrami
  - wyższa efektywność

Zastosowanie

- Zwiększenie dostępności danych
  - failover
  - optymalizacja dostępu do zdalnych danych
  - równoważenie obciążenia węzłów
- Niezawodność środowiska i bezpieczeństwo danych
- Replikowanie struktur danych
  - zapewnienie identycznych schematów bd
    - baza developerska
    - baza testowa

Replikowane obiekty

- **Tabela**
  - **replikowanie operacji DDL**
  - **do węzłów zdalnych przesyłana jest każda operacja DML**
  - **nie ma możliwości ograniczenia kopiowanych danych (np. replikowanie tylko podzbioru rekordów tabeli, spełniających określony warunek) ⇨ repliki tabeli w każdym z węzłów są identyczne**
  - **musi posiadać klucz podstawowy (propagacja zmian)**
- **Indeks**
  - **przesłanie definicji (polecenia DDL) do węzłów zdalnych**
  - **indeksy tworzone automatycznie są replikowane niejawnie razem z definicją tabeli**

Replikowane obiekty

- **Procedura/funkcja**
  - **replikacja procedur i funkcji (zdefiniowanych poza pakietami) ⇨ przesłanie definicji do zdalnego węzła**
  - **nie są replikowane wywołania**
- **Pakiet**
  - **replikacja pakietu (sygnatura + implementacja) ⇨ przesłanie jego definicji do zdalnych węzłów**
  - **odpowiednio skonstruowany pakiet umożliwia wywołanie jego programów w zdalnych węzłach ⇨ replikacja proceduralna**
- **Perspektywa, synonim**
  - **replikacja definicji**

Replikowane obiekty

- **Wyzwalacz**
  - **nie są replikowane automatycznie**
  - **wymagają jawnego wskazania do replikacji (replikacja polecenia create trigger)**
  - **funkcja DBMS_REPUTIL.FROM_REMOTE**
    - **operacja na tabeli (replice) jest wynikiem propagacji zmian z węzła zdalnego ⇨ TRUE**
    - **operacja na tabeli wykonana lokalnie (np. insert) ⇨ FALSE**

```
create or replace trigger czy_zdalna
before insert or update on pracownicy for each row
  begin
  if DBMS_REPUTIL.FROM_REMOTE = false then
    ...
  end if;
end;
```


Obiekty architektury

- **Systemowe pakiety replikacji**
- **Systemowe wyzwalacze replikacji**
  - **wykrycie DML na tabeli**
  - **dla każdej operacji DML wyzwalacz tworzy wywołanie zdalnej procedury (RPC) propagującej operację ⇨ umieszczana w kolejce odroczonych transakcji**
- **Kolejka odroczonych transakcji (deferred transaction queue)**
  - **dla replikacji asynchronicznej**
  - **zawiera RPC odpowiadające DML tworzącym lokalną transakcję węzła**
  - **okresowo dedykowany proces pobiera RPC z kolejki i przesyła je do zdalnych węzłów**

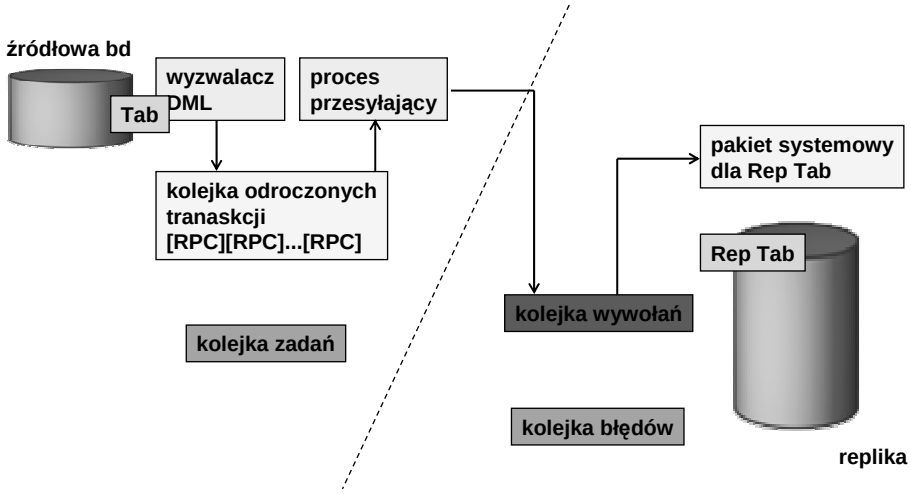
Obiekty architektury

- Odbiorcą RPC w węzłach zdalnych jest pakiet systemowy, utworzony dla repliki tabeli
- W zależności od rodzaju operacji DML, która spowodowała wygenerowanie RPC, uruchamiana jest odpowiednia procedura aplikująca lokalnie zdalną operację DML
- Wywołania replikowanych procedur w ramach odroczonych transakcji są składowane w kolejce wywołań (call queue)
  - dla każdego zmodyfikowanego rekordu z replikowanej tabeli do kolejki wywołań trafia jedno wywołanie odpowiedniej procedury z systemowego pakietu
  - dla replikacji proceduralnej w kolejce są umieszczane wywołania procedur z replikowanych pakietów

Obiekty architektury

- Kolejka zadań (job queue)
  - przechowuje informacje o lokalnych zadaniach węzła (pobranie transakcji z lokalnej kolejki odroczonych transakcji i przesłanie jej do węzłów zdalnych, usunięcie z kolejki transakcji zaaplikowanych)
- Kolejka błędów (error queue)
  - niepowodzenia synchronizacji w węźle zdalnym (na skutek konfliktu)
  - operacja zapisywana jest w kolejce błędów (error queue) węzła docelowego
  - administrator może ponownie wykonać transakcję lub usunąć ją z kolejki

Obiekty architektury

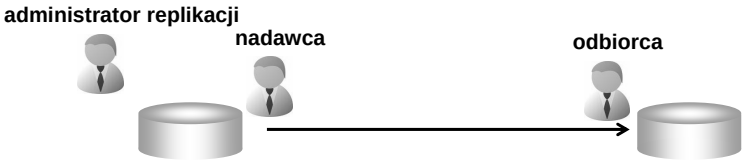


Użytkownicy replikacji

- **Administrator replikacji (replication administrator)**
  - **użytkownik wykonujący wszystkie operacje administracyjne, m.in.:**
    - **definiowanie i administrowanie grupami nadrzędnymi**
    - **dodawanie i usuwanie węzłów środowiska replikacji**
    - **administrowanie kolejkami (odroczonych transakcji, zadań i błędów)**
    - **administrowanie stanem środowiska replikacji (wstrzymywanie i wznawianie replikacji)**
- **W węźle powinien istnieć tylko jeden administrator replikacji**
  - **możliwe rozwiązanie, w którym za replikację różnych schematów będą odpowiadali różni administratorzy replikacji**

Użytkownicy replikacji

- Nadawca (propagator)
  - wysyłanie transakcji z lokalnej kolejki transakcji do węzłów zdalnych
  - węźle może istnieć tylko jeden nadawca
- Odbiorca (receiver)
  - odpowiada za odbiór zdalnych transakcji i aplikowanie ich w lokalnej bazie danych



Replikacja asynchroniczna

- INSERT/UPDATE/DELETE
  - powoduje uruchomienie systemowego wyzwalacza dla replikowanej tabeli
  - wyzwalacz tworzy odroczone zdalne wywołania procedur (RPC) i umieszcza je w lokalnej kolejce odroczonych transakcji
  - każda odroczona transakcja w kolejce posiada listę węzłów, do których zostanie przesłana
  - okresowo jest uruchomiany proces, który pobiera transakcje z kolejki odroczonych transakcji i przesyła je do węzłów docelowych
    - dla każdego węzła docelowego może zostać zdefiniowany różny moment przesłania transakcji
    - jawne wymuszenie przesłania - procedura DBMS_DEFER_SYS. PUSH

Replikacja asynchroniczna

- zdalne wywołania procedur (RPC), składające się na transakcję, zostają odebrane przez wewnętrzny pakiet i wykonane lokalnie
- jeżeli operacja nie może być wykonana lokalnie (np. naruszenie ograniczenia int.), trafia do lokalnej kolejki błędów
- konflikt skutkuje wywołaniem procedury obsługi konfliktu
- niedostępność jednego z węzłów nie wpływa na pracę pozostałych węzłów środowiska ⇨ RPC będą replikowane do tego węzła po nawiązaniu łączności

Replikacja asynchroniczna

- zaaplikowanie odroczonej transakcji we wszystkich zdalnych węzłach docelowych nie powoduje natychmiastowego usunięcia transakcji z kolejki odroczonych transakcji węzła, w którym cały proces się rozpoczął
- proces systemowy przegląda kolejkę odroczonych transakcji węzła i usuwa z niej transakcje, których propagacja zakończyła się sukcesem
  - częstotliwość uruchamiania procesu jest określana przez administratora
  - ręczne czyszczenie kolejki DBMS_DEFER_SYS.PURGE

Replikacja synchroniczna

- Przesłania zdalne w ramach tej samej transakcji, która modyfikuje dane lokalne
- Jeśli przesłanie zmian do przynajmniej jednego węzła nie jest możliwe (niedostępność węzła) ⇨ cała transakcja zostaje wycofana
- Brak konfliktów

Replikacja synchroniczna

- INSERT/UPDATE/DELETE
  - każda operacja wykonana lokalnie powoduje uruchomienie systemowego wyzwalacza, który generuje zdalne wywołania procedur (RPC)
  - zdalne wywołania procedur (RPC) zostają natychmiast przesłane do węzłów docelowych
  - systemowe procedury w węzłach zdalnych odbierają RPC i aplikują je
  - zatwierdzanie transakcji ⇨ 2PC

Stany grupy replikacji

- **NORMAL**
  - **replikacja działa poprawnie**
- **QUIESCING (wyłączanie)**
  - **stan przejściowy między NORMAL a QUIESCED**
  - **użytkownicy nie mogą rozpoczynać nowych transakcji dla obiektów grupy**
  - **trwa do momentu przesłania i zaaplikowania wszystkich transakcji z kolejki odroczonych transakcji w węzłach zdalnych - wówczas grupa przechodzi do stanu QUIESCED**

Stany grupy replikacji

- **QUIESCED (wyłączenie)**
  - **replikacja zatrzymana**
  - **brak wpisów w kolejce odroczonych transakcji**
  - **użytkownicy nie mogą adresować transakcji do obiektów grupy**
  - **przeznaczony do wykonywania czynności administracyjnych (np. dodanie nowego obiektu, zmiana definicji obiektu)**

Stany grupy replikacji

- Zmiana stanu grupy replikacji
  - zawieszenie (suspend): przejście NORMAL → QUIESCED
  - przywrówcenie (resume): przejście QUIESCED → NORMAL

Parametry konfiguracyjne

- COMPATIBLE
  - przynajmniej 9.0.1.
- GLOBAL_NAMES
  - określa, czy nazwy łączników bazy danych mają być zgodne z nazwą globalną bazy danych, na którą wskazują
  - zalecane: TRUE
- JOB_QUEUE_PROCESSES
  - określa liczbę systemowych procesów dedykowanych do obsługi zadań (np. odświeżanie perspektyw zmat.) - J000-J999
  - domyślnie: 0 - wszystkie zadania administracyjne wraz z propagacją odroczonych transakcji i usuwaniem zaaplikowanych już transakcji muszą być uruchamiane ręcznie przez administratora
  - zalecane: max. liczba zadań, jakie mogą działać równocześnie +1

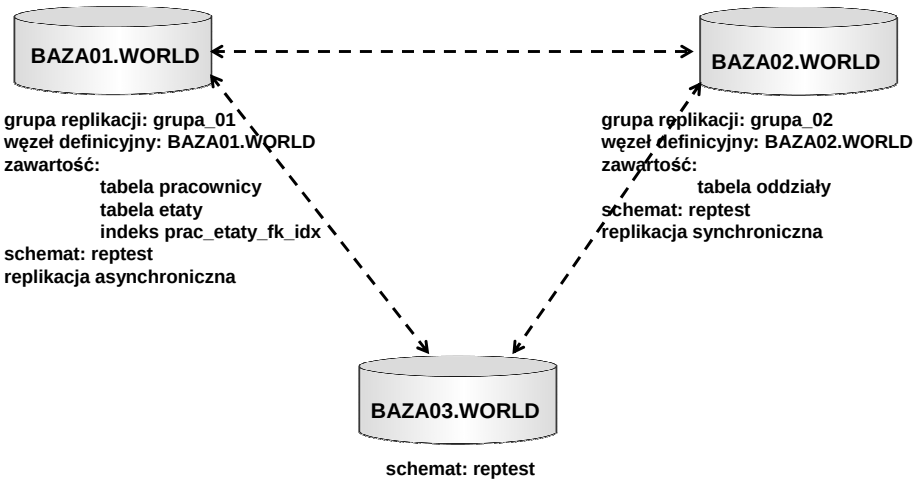
Parametry konfiguracyjne

- **OPEN_LINKS**
  - **max. liczba równocześnie aktywnych łączników bd**
  - **domyślnie: 4, max.: 255**
  - **zalecane: liczba węzłów**
- **PROCESSES**
  - **max. liczba wszystkich procesów obsługiwanych przez instancję**
  - **min.: 6, max.: limit systemu operacyjnego**

Replikacja multimaster

Replikacja rekordów
Replikacja asynchroniczna i synchroniczna

Konfigurowanie



Konfigurowanie

- Poniższe kroki wykonać dla wszystkich baz danych
- Administrator węzła nadrzędnego

```
create user repadmin identified by repadmin;
```

- Uprawnienia

```
DBMS_REPCAT_ADMIN.GRANT_ADMIN_ANY_SCHEMA(username => 'repadmin');
```

- Uprawnienia dla graficznej aplikacji zarządzania replikacją

```
grant select any dictionary to repadm;
```

Konfigurowanie

- repadm jest nadawcą

```
DMBS_DEFER_SYS.REGISTER_PROPAGATOR(username => 'repadmin');
```

- Procedura DBMS_DEFER_SYS.REGISTER_PROPAGATOR
  - rejestruje użytkownika wskazanego parametrem username jako nadawcę transakcji dla węzła
  - nadaje mu przywileje CREATE SESSION, CREATE PROCEDURE, CREATE DATABASE LINK, EXECUTE ANY PROCEDURE

Konfigurowanie

- repadm jest odbiorcą

```
DBMS_REPCAT_ADMIN.REGISTER_USER_REPGROUP(
username => 'repadmin',
privilege_type => 'receiver',
list_of_gnames => null);
```

- Procedura DBMS_REPCAT_ADMIN.REGISTER_USER_REPGROUP
  - wskazanemu parametrem username użytkownikowi nadaje uprawnienia określone parametrem privilege_type
    - receiver - dla użytkownika-odbiorcy transakcji z węzła zdalnego
    - proxy_snapadmin - dla użytkownika-pełnomocnika administratora migawek (repl. migawkowa)
  - list_of_gnames wskazuje grupy replikacji, dla których użytkownik zostaje zarejestrowany
    - null powoduje, że użytkownik zostaje zarejestrowany dla wszystkich grup replikacji, nawet tych, które zostaną dopiero utworzone

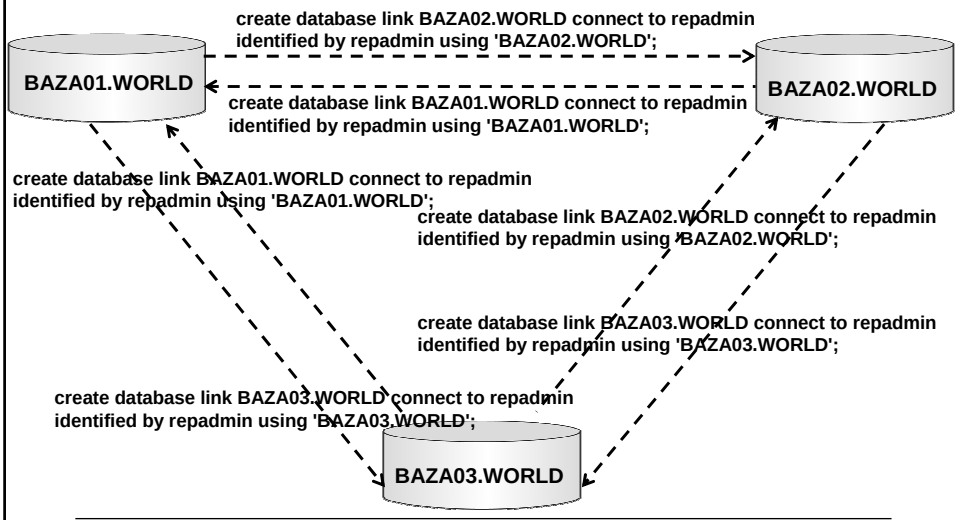
Konfigurowanie

- **Zadanie okresowego czyszczenia lokalnej kolejki odroczonych transakcji (czyszczenie z transakcji zaaplikowanych do węzłów zdalnych)**

```
DBMS_DEFER_SYS.SCHEDULE_PURGE(
next_date => SYSDATE, interval => 'SYSDATE + 1/(24*6)',
delay_seconds => 0, execution_seconds => 0);
```

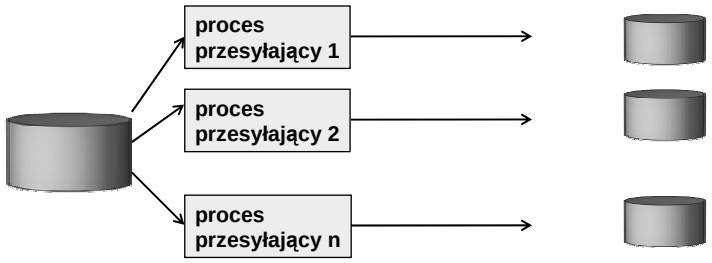
  - **interval: proces uruchamiany co 10 minut**
  - **delay_seconds: czas aktywności procesu**
    - **=0 ⇨ jeśli kolejka pusta to proces kończy swoje działanie**
    - **> 0 ⇨ jeśli kolejka pusta to proces pozostaje aktywny przez n sekund**
  - **execution_seconds: czas działania procesu**
    - **= 0, wówczas proces działa tak długo, aż nie usunie wszystkich transakcji z kolejki**
    - **> 0 ⇨ działanie procesu zakończy się dokładnie po wskazanym czasie (nawet jeśli w kolejce jeszcze istnieją transakcje do usunięcia)**

Konfigurowanie



Konfigurowanie

- W każdym węźle definiujemy procesy, których zadaniem będzie pobieranie odroczonych transakcji z kolejki i przesyłanie ich do węzłów zdalnych
- Dla każdego węzła zdalnego, do którego węzeł będzie przesyłał swoje lokalne transakcje definiuje się osobny proces



Konfigurowanie

- Procedura uruchamiana jako repadm w każdym z węzłów (poniżej dla węzła 1)

```
DBMS_DEFER_SYS.SCHEDULE_PUSH(
     destination => 'BAZA02.WORLD',
     next_date => SYSDATE,
     interval => 'SYSDATE + 1/144',
     parallelism => 1,
     execution_seconds => 0,
     delay_seconds => 0);


DBMS_DEFER_SYS.SCHEDULE_PUSH(
     destination => 'BAZA03.WORLD',
     next_date => SYSDATE,
     interval => 'SYSDATE + 1/144',
     parallelism => 1,
     execution_seconds => 0,
     delay_seconds => 0);
```

Konfigurowanie

- destination: nazwa węzła, do którego proces będzie przesyłał transakcje z lokalnej kolejki transakcji
- next_date: moment uruchomienia procesu
- interval: okres uruchamiania (10 min)
- parallelism: zrównoleglenie propagacji
  - 0: propagacja sekwencyjna
  - n > 1: do propagacji jest wykorzystywanych n procesów
- execution_seconds i delay_seconds: znaczenie, jak dla DBMS_DEFER_SYS.SCHEDULE_PURGE

Konfigurowanie

- Jako repadmin tworzymy nadrzędną grupę replikacji o nazwie grupa_01
- Jej węzłem definicyjnym jest BAZA01.WORLD

```
DBMS_REPCAT.CREATE_MASTER_REPGROUP(gname => 'grupa_01');
```

- Informacje o grupach: DBA_REPGROUP

Konfigurowanie

- Dodanie obiektów do grupy

```
DBMS_REPCAT.CREATE_MASTER_REPOBJECT(gname => 'grupa_01',
oname => 'etaty', type => 'TABLE', sname => 'reptest',
use_existing_object => true, copy_rows => false);

DBMS_REPCAT.CREATE_MASTER_REPOBJECT(gname => 'grupa_01',
oname => 'pracownicy', type => 'TABLE', sname => 'reptest',
use_existing_object => true, copy_rows => false);

DBMS_REPCAT.CREATE_MASTER_REPOBJECT(gname => 'grupa_01',
oname => 'prac_etaty_fk_idx', type => 'INDEX', sname => 'reptest',
use_existing_object => true, copy_rows => false);
```

- Zbiór obiektów w grupie replikacji ⇨ DBA_REPOBJECT

Konfigurowanie

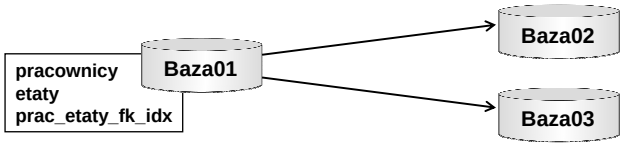
- Wskazanie węzłów do których będą replikowane obiekty wskazanej grupy
- DBMS_REPCAT.ADD_MASTER_DATABASE
  - dodaje do środowiska replikacji węzeł, którego nazwę podano w parametrze master
  - węzeł ten będzie replikował obiekty z grupy o nazwie w gname
  - propagation_mode: ASYNCHRONOUS/SYNCHRONOUS
- Zbiór węzłów środowiska replikacji ⇨ DBA_REPSITES

Konfigurowanie

- Dla grupy replikacji definiujemy listę węzłów, do których będą replikowane obiekty należące do grupy

```
DBMS_REPCAT.ADD_MASTER_DATABASE(gname => 'grupa_01',
master => 'BAZA02.WORLD',
use_existing_objects => true,
copy_rows => false,
propagation_mode => 'ASYNCHRONOUS');

DBMS_REPCAT.ADD_MASTER_DATABASE(gname => 'grupa_01',
master => 'BAZA03.WORLD',
use_existing_objects => true,
copy_rows => false,
propagation_mode => 'ASYNCHRONOUS');
```



Konfigurowanie

```
select oname, type, sname, generation_status
from   DBA_REPOBJECT where  gname = 'grupa_01';

ONAME                TYPE          SNAME              GENERATION_STATUS
-------------------- ------------- ------------------ -----------------
ETATY                TABLE         REPTEST            NEEDSGEN
PRAC_ETATY_FK_IDX    INDEX         REPTEST
PRACOWNICY           TABLE         REPTEST            NEEDSGEN
```

- oname: nazwa obiektu
- type: typ replikowanego obiektu
- sname: nazwa schematu w którym znajduje się replikowany obiekt
- generation_status:
  - jeśli obiekt wymaga utworzenia obiektów wspomagających replikację ⇨ NEEDSGEN
  - jeśli obiekty wspomagające są poprawne ⇨ GENERATED
- gname: nazwa grupy

Konfigurowanie

- Utworzenie obiektów wspomagających replikację dla obiektów dla których generation_status='NEEDSGEN'

```
DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT(
oname => 'pracownicy',
sname => 'reptest',
type => 'TABLE',
min_communication => true);

DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT(
oname => 'etaty',
sname => 'reptest',
type => 'TABLE',
min_communication => true);
```


Konfigurowanie

- Obiekty wspomagające zostaną automatycznie utworzone we wszystkich węzłach środowiska (również w węźle definicyjnym)
- Parametr min_communication określa, czy wielkość danych, przesyłanych pomiędzy węzłami, ma zostać zmniejszona do minimum

Konfigurowanie

```
SQL> select oname, type, sname, generation_status, min_communication
  2  from   DBA_REPOBJECT
  3  where  gname = 'grupa_01';

ONAME                TYPE          SNAME              GENERATION_STATUS M
-------------------- ------------- ------------------ ---------------- -
ETATY                TABLE         REPTEST            GENERATED        Y
ETATY$RP             PACKAGE       REPTEST
ETATY$RP             PACKAGE BODY  REPTEST
PRAC_ETATY_FK_IDX    INDEX         REPTEST
PRACOWNICY           TABLE         REPTEST            GENERATED        Y
PRACOWNICY$RP        PACKAGE       REPTEST
PRACOWNICY$RP        PACKAGE BODY  REPTEST
```

- **Uruchomienie replikacji**

```
DBMS_REPCAT.RESUME_MASTER_ACTIVITY(gname => 'grupa_01');
```


Konfigurowanie - BAZA02

- **Jako repadmin tworzymy nadrzędną grupę replikacji o nazwie grupa_02**
- **Jej węzłem definicyjnym jest BAZA02.WORLD**

```
DBMS_REPCAT.CREATE_MASTER_REPGROUP(gname => 'grupa_02');
```

- **Do utworzonej grupy dodajemy tabelę**

```
DBMS_REPCAT.CREATE_MASTER_REPOBJECT(gname => 'grupa_02',
oname => 'oddzialy',
type => 'TABLE',
sname => 'reptest',
use_existing_object => true,
copy_rows => false);
```

Konfigurowanie - BAZA02

- Definiujemy węzły zdalne, do których będą replikowane obiekty z nadrzędnej grupy replikacji - propagacja synchroniczna

```
DBMS_REPCAT.ADD_MASTER_DATABASE(gname => 'grupa_02',
master => 'BAZA01.WORLD',
use_existing_objects => true,
copy_rows => false,
propagation_mode => 'SYNCHRONOUS');
```

```
DBMS_REPCAT.ADD_MASTER_DATABASE(gname => 'grupa_02',
master => 'BAZA03.WORLD',
use_existing_objects => true,
copy_rows => false,
propagation_mode => 'SYNCHRONOUS');
```


Konfigurowanie - BAZA02

- Utworzenie obiektów wspomagających replikację

```
DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT(
oname => 'oddzialy',
sname => 'reptest',
type => 'TABLE',
min_communication => true);
```

- Uruchomienie replikacji dla grupy

```
DBMS_REPCAT.RESUME_MASTER_ACTIVITY(gname => 'grupa_02');
```

Replikacja multimaster

Replikacja proceduralna
Replikacja asynchroniczna i synchroniczna

Replikacja proceduralna

- **Pomiędzy węzłami replikowane są wywołania procedury składowanej, która modyfikuje zawartość tabeli**
  - **nie są replikowane operacje modyfikacji tabeli źródłowej**
- **Zastosowanie**
  - **SRBD z dużą liczbą transakcji modyfikującej duże wolumeny danych**
- **W przypadku stosowania replikacji proceduralnej w środowisku hybrydowym wywołania procedur są przesyłane wyłącznie do węzłów nadrzędnych**
  - **węzły migawkowe nie uczestniczą w propagacji**

Replikacja proceduralna

- W danym momencie może działać tylko jedna replikowana procedura
- Replikacja synchroniczna
  - wywołania procedury są natychmiast przesłane do węzłów docelowych
- Replikacja asynchroniczna
  - wywołania procedury są składowane w lokalnej kolejce transakcji

Replikacja proceduralna

- Wymaga utworzenia pakietu, który będzie służył do modyfikowania danych
- Wymagania odnośnie do pakietu
  - może zawierać wyłącznie procedury
  - parametry procedur mogą być wyłącznie trybu IN
  - w ciałach procedur pakietu, których wywołanie będzie replikowane, musi znaleźć się kod wyłączający replikację na poziomie rekordu
  - procedury nie mogą odwoływać się do funkcji SQL, których wykonanie jest zależne od stanu lokalnego środowiska (np. funkcji SYSDATE)

Replikacja proceduralna

- Przykładowy pakiet utworzony w schemacie reptest

```
create or replace package pracownicy_rep
is
procedure usun_pracownikow(p_nazwa_etatu IN VARCHAR2);
end pracownicy_rep;

create or replace package body pracownicy_rep
is
procedure usun_pracownikow(p_nazwa_etatu IN VARCHAR2)
is
  begin
    DBMS_REPUTIL.REPLICATION_OFF;
    delete from pracownicy
    where  pr_et_nazwa = p_nazwa_etatu;
    DBMS_REPUTIL.REPLICATION_ON;
    end usun_pracownikow;
end pracownicy_rep;
```


Replikacja proceduralna

- Po zdefiniowaniu pakietu i wskazaniu go jako obiekt replikacji należy zainicjować generowanie obiektu wspomagającego w postaci systemowego pakietu (wrappera)
- Systemowy pakiet zostaje utworzony w schemacie administratora węzła replikacji
  - przedrostek pakietu: DEFER_
- W schemacie, w którym został zdefiniowany oryginalny pakiet powstaje synonim wskazujący na systemowy pakiet
- Systemowy pakiet zawiera
  - procedury o identycznych nazwach, jak procedury z pakietu oryginalnego
  - inne niż oryginalne ciała procedur
  - dodatkowe parametry wywołania: call_local i call_remote

Replikacja proceduralna

- **call_local=Y: wywołanie procedury z pakietu ma zostać wykonane lokalnie ⇨ wywołanie oryginalnej procedury (domyślnie N)**
- **call_remote=Y: wywołanie procedury z pakietu ma zostać replikowane do węzłów zdalnych (domyślnie Y) ⇨ procedura tworzy zdalne wywołania (RPC) dla wszystkich węzłów zdalnych środowiska, które powodują uruchomienie tam procedury z repliki pakietu**
- **Replikowanie: wywołanie procedury z pakietu systemowego**

Replikacja proceduralna

➲ **Przykładowe wywołanie procedury pakietu systemowego (jako repadmin)**

```
DEFER_pracownicy_rep.usun_pracownikow(
p_nazwa_etatu => 'ADMINISTRATOR',
call_local => 'Y',
call_remote => 'Y');
```

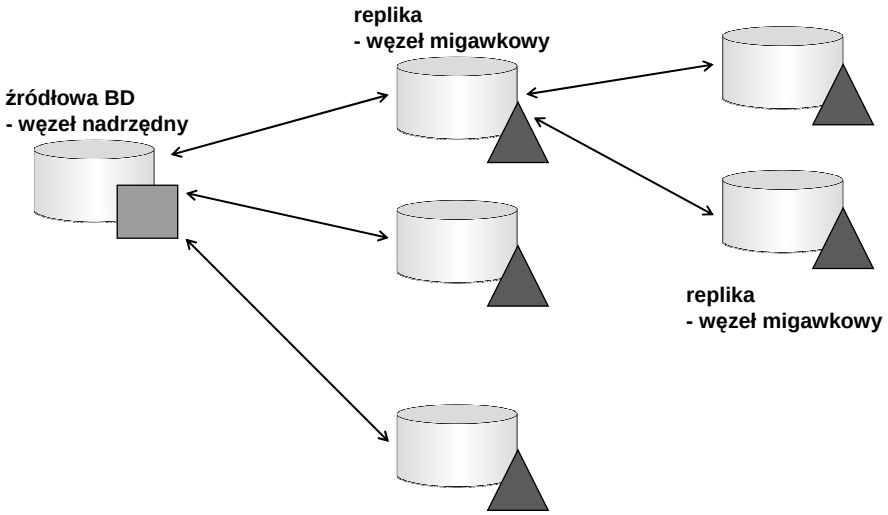
Replikacja migawkowa

Replikacja rekordów
Replikacja asynchroniczna

Replikacja migawkowa

- **Replikacja standardowa rozszerzona o migawki modyfikowalne**
- **Replikacja 2-kierunkowa**
- **Replikacja rekordów**
- **Rodzaje węzłów**
  - **nadrzędny (master site)**
  - **migawkowy (materialized view site) - replikuje dane z węzła nadrzędnego (źródłem danych jest tabela lub inna migawka)**
- **Każdy węzeł migawkowy jest połączony z dokładnie jednym węzłem nadrzędnym**
  - **węzeł nadrzędny może być połączony z wieloma węzłami migawkowymi**
  - **węzeł migawkowy może być węzłem nadrzędnym dla innych węzłów migawkowych**

Replikacja migawkowa



Replikacja migawkowa

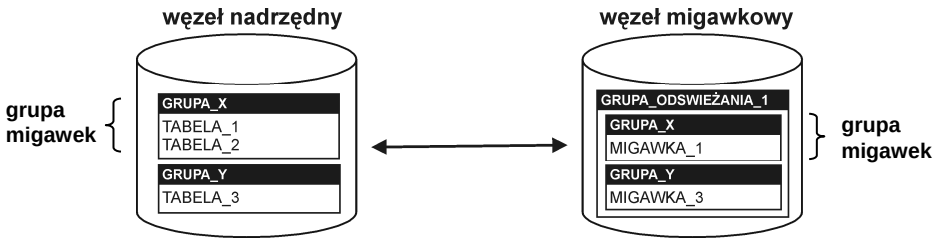
- Odświeżanie okresowe
- Ograniczenie replikowanych danych
  - kolumny
  - podzbiór rekordów
- Możliwość pracy bez połączeń między węzłami (pomiędzy odświeżeniami)

Replikacja migawkowa

- **Obiekty**
  - **migawka modyfikowalna (klauzula for update)**
    - **musi zostać przydzielona do grupy migawek**
  - **dziennik migawki modyfikowalnej (updateable snapshot log)**
    - **tworzony automatycznie**
    - **nazwa: USLOG$_nazwa_migawki**
    - **systemowy wyzwalacz ⇨ zapis w dzienniku operacji DML na migawce modyfikowalnej**
    - **wykorzystywany do określenia, które rekordy mają zostać nadpisane lub usunięte z migawki podczas odświeżania przyrostowego**
  - **łącznik bd**

Replikacja migawkowa

- **Grupa migawek (materialized view group)**
  - **tworzona w węźle migawkowym**
  - **nie musi zawierać migawek dla wszystkich tabel z grupy nadrzędnej**
- **Migawka modyfikowalna musi należeć do grupy migawek o nazwie identycznej z nazwą nadrzędnej grupy replikacji**



Replikacja migawkowa

- Grupa migawek jest automatycznie rejestrowana w węźle nadrzędnym
  - zbiór migawek dostępny w DBA_REGISTERED_MVIEW_GROUPS
- Migawki z jednej grupy migawek mogą należeć do różnych grup odświeżania,
- Grupa odświeżania może zawierać migawki z różnych grup migawek

Użytkownicy - węzeł nadrzędny

- Proxy materialized view administrator (ProxyMVA)
  - wykonuje w węźle nadrzędnym operacje w imieniu administratora migawek z węzła migawkowego
  - podstawowy zestaw uprawnień
- Proxy refresher (ProxyR)
  - wykonuje operacje w imieniu użytkownika odświeżającego migawki w węźle migawkowym

Użytkownicy - węzeł migawkowy

- Administrator migawek (MView administrator)
  - odpowiada za konfigurowanie i zarządzanie węzłem migawkowym
  - odpowiednik administratora replikacji multimaster
- Nadawca (propagator)
  - przesyła lokalne transakcje z lokalnej kolejki odroczonych transakcji do węzła nadrzędnego
  - wykorzystywany, gdy węzeł migawkowy zawiera migawki modyfikowalne

Użytkownicy - węzeł migawkowy

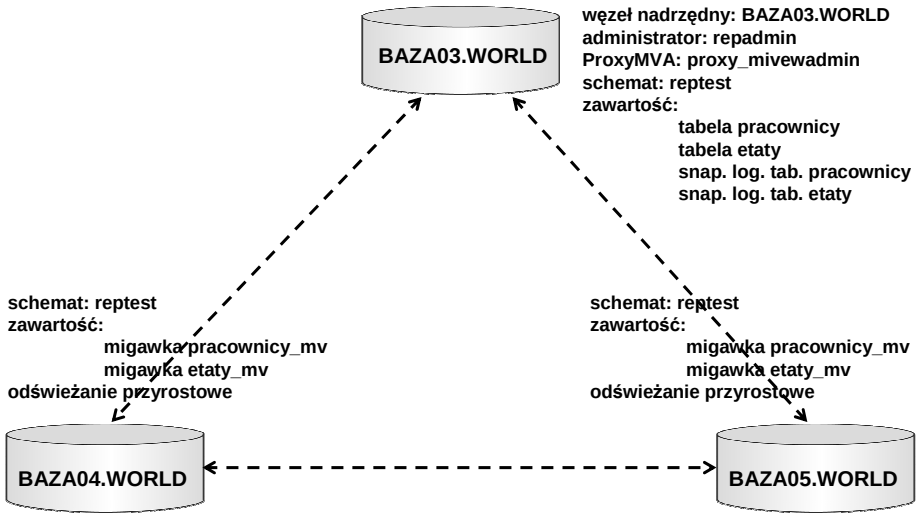
- Użytkownik odświeżający migawki (refresher)
  - aplikowanie zmian, jakie zaszły w tabelach lub migawkach węzła nadrzędnego
- Odbiorca (receiver)
  - odbiera odroczone transakcje, przesyłane przez nadawców z innych węzłów migawkowych
  - wykorzystywany tylko, gdy węzeł migawkowy jest węzłem nadrzędnym dla innych węzłów migawkowych

Odświeżanie

- **Wszystkie transakcje, które dokonały modyfikacji danych migawki, są zapisywane w kolejce odroczonych transakcji**

1. **Przesłanie zawartości kolejki odroczonych transakcji do węzła nadrzędnego (lub węzła migawkowego w przypadku architektury wielowarstwowej) i zaaplikowanie ich w tabeli źródłowej ⇨ mogą wystąpić konflikty**
2. **Przesłanie danych z tabeli źródłowej z węzła nadrzędnego do migawki ⇨ odświeżenie migawki**

Konfigurowanie



Konfigurowanie

- **BAZA03.WORLD**
  - **utworzenie użytkownika proxy_mviewadmin**
  - **nadanie uprawnień**

```
DBMS_REPCAT_ADMIN.REGISTER_USER_REPGROUP(
username => 'proxy_mviewadmin',
privilege_type => 'proxy_snapadmin',
list_of_gnames => null);
```

  - **utworzenie użytkownika proxy_refresher**
  - **nadanie uprawnień**

```
grant create session, select any table to proxy_refresher;
```


Konfigurowanie

- **BAZA04.WORLD**
  - **utworzenie administratora węzła ⇨ mviewadmin**
  - **nadanie uprawnień**

```
DBMS_REPCAT_ADMIN.GRANT_ADMIN_ANY_SCHEMA(
username => 'mviewadmin');
```

```
grant comment any table, lock any table,
select any dictionary
to mviewadmin;
```

  - **utworzenie nadawcy lokalnych transakcji do węzła nadrzędnego ⇨ propagator**
  - **nadanie uprawnień**

```
DMBS_DEFER_SYS.REGISTER_PROPAGATOR(
username => 'propagator');
```

Konfigurowanie

- **BAZA04.WORLD**
  - **utworzenie użytkownika odpowiedzialnego za proces odświeżania migawek zawartością tabel z węzła nadrzędnego ⇨ refresher**
  - **nadanie uprawnień**

```
grant create session,
      alter any materialized view
to refresher;
```

- **Dla węzła migawkowego nadrzędnego konieczny jest użytkownik odpowiedzialny za odbiór odroczonych transakcji przesyłanych z węzłów migawkowych ⇨ administrator węzła (mviewadmin)**

```
DBMS_REPCAT_ADMIN.REGISTER_USER_REPGROUP(
username => 'mviewadmin',
privilege_type => 'receiver',
list_of_gnames => null);
```


Konfigurowanie

- **Łącznik z BAZA04 w schemacie mviewadmin**

```
create database link BAZA03.WORLD
connect to proxy_mviewadmin identified by proxy_mviewadmin
using 'BAZA03.WORLD';
```

- **Łącznik z BAZA04 w schemacie propagator**

```
create database link BAZA03.WORLD
connect to repadmin identified by repadmin
using 'BAZA03.WORLD';
```

- **Łącznik z BAZA04 w schemacie reptest**

```
create database link BAZA03.WORLD
connect to proxy_refresher identified by proxy_refresher
using 'BAZA03.WORLD';
```

Konfigurowanie

- **BAZA04.WORLD**
  - **zdefiniowanie zadania okresowego czyszczenia lokalnej kolejki odroczonych transakcji**
  - **jako mviewadmin**

```
DBMS_DEFER_SYS.SCHEDULE_PURGE(
next_date => SYSDATE,
interval => 'SYSDATE + 1/24',
delay_seconds => 0,
execution_seconds => 0);
```

  - **zdefiniowanie automatycznego przesyłania modyfikacji z migawki do węzła nadrzędnego (tabeli źródłowej)**

```
DBMS_DEFER_SYS.SCHEDULE_PUSH(
destination => 'BAZA03.WORLD',
next_date => SYSDATE,
interval => 'SYSDATE + 1/24',
parallelism => 0);
```

- **Uwaga: powyższe operacje wykonać dla wszystkich węzłów migawkowych**

Konfigurowanie

- **BAZA03.WORLD**
  - **utworzenie dzienników migawek dla tabel źródłowych**

```
create materialized view log on reptest.etaty;
create materialized view log on reptest.pracownicy;
```

Konfigurowanie

- **BAZA04.WORLD**
  - **utworzenie grupy migawek w schemacie mviewadmin**
  - **uwaga: nazwa grupy musi być zgodna z nazwą nadrzędnej grupy replikacji z węzła nadrzędnego, zawierającej tabele bazowe**

```
DBMS_REPCAT.CREATE_MVIEW_REPGROUP(
gname => 'grupa_01',
master => 'BAZA03.WORLD',
propagation_mode => 'ASYNCHRONOUS');
```

  - **grupa jest automatycznie rejestrowana w węźle nadrzędnym ⇨ DBA_REGISTERED_MVIEW_GROUPS**
  - **informacje słownikowe o utworzonych grupach ⇨ DBA_REPGROUP**

Konfigurowanie

- **BAZA04.WORLD**
  - **utworzenie pustej grupy odświeżania**

```
DBMS_REFRESH.MAKE(
name => 'mviewadmin.grod_01',
list => '',
next_date => SYSDATE,
interval => 'SYSDATE + 1/24',
implicit_destroy => false,
push_deferred_rpc => true,
refresh_after_errors => false,
parallelism => 1);
```

- **push_deferred_rpc =true ⇨ modyfikacje, jakie zostały wykonane na migawce, mają zostać przesłane do tabeli bazowej przed wykonaniem operacji właściwego odświeżenia migawki (jeśli false, wówczas zmiany w migawce są tracone)**
- **refresh_after_errors = false ⇨ migawka modyfikowalna nie zostanie odświeżona w przypadku konfliktów (perspektywa DEFERRORS)**

Konfigurowanie

- **BAZA04.WORLD**
  - **utworzenie migawek**

```
create materialized view reptest.etaty_mv
refresh fast for update
as select * from reptest.etaty@BAZA03.WORLD;

create materialized view reptest.pracownicy_mv
refresh fast for update
as select * from reptest.pracownicy@BAZA03.WORLD;
```

  - **dodanie migawek do grupy migawek**

```
DBMS_REPCAT.CREATE_MVIEW_REPOBJECT(
gname => 'grupa_01',
sname => 'reptest',
oname => 'etaty_mv',
type => 'SNAPSHOT',
min_communication => true);
```

```
DBMS_REPCAT.CREATE_MVIEW_REPOBJECT(
gname => 'grupa_01',
sname => 'reptest',
oname => 'pracownicy_mv',
type => 'SNAPSHOT',
min_communication => true);
```


Konfigurowanie

- **BAZA04.WORLD**
  - **dodanie migawek do grupy odświeżania**

```
DBMS_REFRESH.ADD(
name => 'grod_01',
list => 'reptest.etaty_mv',
lax => false);
```

```
DBMS_REFRESH.ADD(
name => 'grod_01',
list => 'reptest.pracownicy_mv',
lax => false);
```

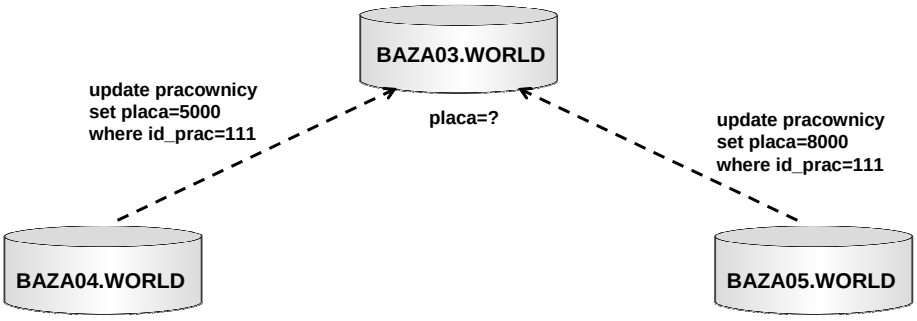
  - **lax=true ⇨ umożliwia przenoszenie migawek pomiędzy grupami**

Konflikty

- Występują w replikacji asynchronicznej
- Wykrywanie i rozwiązywanie konfliktów
  - w replikacji rekordowej i migawkowej ⇨ zbiór możliwych do wykorzystania predefiniowanych technik
  - w replikacji proceduralnej ⇨ procedura użytkownika

Rodzaje konfliktów

- Aktualizacja
  - gdy transakcje, pochodzące z różnych węzłów środowiska, dokonują w tym samym czasie operacji aktualizacji tego samego rekordu w replikach
  - wartość rekordu po synchronizacji zależy od kolejności propagacji zmian



Rodzaje konfliktów

- **Naruszenie unikalności**
  - **gdy replikacja rekordu powoduje naruszenie ograniczeń klucza podstawowego lub klucza unikalnego w węzłach docelowych**
- **Usunięcie**
  - **gdy jedna transakcja usuwa rekordy z replikowanej tabeli, które są w tym samym momencie modyfikowane przez transakcję z innego węzła**

Wykrywanie konfliktów

- **Aktualizacja**
  - **dane wysyłane do węzła w procesie synchronizacji zawierają**
    - **nowe wartości kolumn z replikowanego rekordu (rekord po modyfikacji)**
    - **stare wartości kolumn (sprzed modyfikacji)**
  - **jeżeli istnieje różnica pomiędzy starą wartością replikowanego rekordu, przesłaną wraz z operacją z węzła zdalnego, a wartością aktualną odpowiadającego mu lokalnego rekordu ⇨ konflikt**

Wykrywanie konfliktów

- **Unikalność**
  - **w węźle docelowym błąd systemowy naruszenia ograniczenia integralnościowego PK lub UNIQUE**
- **Usunięcie**
  - **w węźle docelowym nie można odnaleźć rekordów do uaktualnienia (wymaganych synchronizacją z węzłem zdalnym)**
- **Informacje o nierozwiązanych konfliktach Aktualizacja, Unikalność, Usunięcie ⇨ perspektywa DEFERROR**
  - **ID transakcji w lokalnym węźle, która spowodowała błąd**
  - **miejsce pochodzenia tej transakcji (nazwa globalna węzła)**
  - **numer i tekst błędu**

Ręczne rozwiązywanie konfliktów

- **Analiza zawartości DEFERROR**
- **Zidentyfikowanie transakcji lokalnych powodujących konflikt**
- **Pozostawienie jednej transakcji jako poprawnej, dla pozostałych transakcji należy wykonać tzw. transakcje kompensujące przywracające dane sprzed rozpoczęcia tych transakcji**

```
DBMS_REPUTIL.REPLICATION_OFF;
insert ...;
...
update ...;
...
delete ...;
...
commit;
DBMS_REPUTIL.REPLICATION_ON;
```

wyłączenie propagacji transakcji kompensującej

transakcja kompensująca

włączenie propagacji

Ręczne rozwiązywanie konfliktów

- Po wykryciu konfliktu wszystkie transakcje trafiają do kolejki błędów
- Usunięcie transakcji skompensowanych z kolejki błędów (dla wszystkich węzłów z transakcjami skompensowanymi)

```
DBMS_DEFER_SYS.DELETE_ERROR
        (deferred_tran_id => 'local_tran_id');
```

- Zaaplikowanie transakcji wybranej jako poprawna do węzła zdalnego (zdalnych)

```
DBMS_DEFER_SYS.EXECUTE_ERROR(
deferred_tran_id => 'local_tran_id',
destination=>'baza.zdalna')
```


Metody predefiniowane

- Procedura DBMS_REPCAT.ADD_UPDATE_RESOLUTION, parametr METHOD
- Konflikty uaktualnienia ⇨ predefiniowane metody
  - minimum
  - maximum
  - latest timestamp
  - earliest timestamp
  - additive
  - average
  - priority group
  - site priority
  - overwrite
  - discard

Metody predefiniowane

- Konflikty naruszenia unikalności ⇨ predefiniowane metody
  - append site name
  - append sequence
  - discard
- Konflikty usunięcia ⇨ brak predefiniowanych metod

Metody predefiniowane

- Latest timestamp
  - wykorzystuje znacznik czasowy polecenia update
  - wybierane polecenie o największym znaczniku (najmłodsze)
  - wymaga zaimplementowania w każdym z węzłów mechanizmu oznaczania czasu wykonania każdego polecenia update dla danych replikowanej tabeli ⇨ dodatkowa kolumna w tabeli
    - wypełniana przez aplikację lub wyzwalacz
  - problem synchronizacji zegarów

Latest timestamp - konfiguracja

- Rozwiązanie konfliktu modyfikacji wartości atrybutu pracownicy.etat
- Węzeł: BAZA01.WORLD
- Użytkownik: repadmin
- Tabela pracownicy w grupie replikacji grupa_01

- Zatrzymanie replikacji

```
DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY(gname => 'grupa_01');
```

- Dodanie do tabeli znacznika czasowego

```
DBMS_REPCAT.ALTER_MASTER_REPOBJECT(
sname => 'reptest', oname => 'pracownicy', type => 'TABLE',
ddl_text => 'alter table reptest.pracownicy add
            (pr_znacznik_czasowy DATE)');
```


Latest timestamp - konfiguracja

- Nadawanie znacznika czasowego ⇨ wyzwalacz

```
DBMS_REPCAT.CREATE_MASTER_REPOBJECT(
gname => 'grupa_01', type => 'TRIGGER'',
oname => 'czas_modyfikacji', sname => 'reptest',
ddl_text => 'create trigger reptest.czas_modyfikacji
             before insert or update on pracownicy
             for each row
             begin
               if DBMS_REPUTIL.FROM_REMOTE = false then
                 :new.pr_znacznik_czasowy := SYSDATE;
               end if;
             end;');
```

Latest timestamp - konfiguracja

- Zdefiniowanie grupy kolumn o nazwie pracownicy_etat z kolumnami etat i pr_znacznik_czasowy

```
DBMS_REPCAT.MAKE_COLUMN_GROUP(
sname => 'reptest',
oname => 'pracownicy',
column_group => 'pracownicy_etat',
list_of_column_names => 'etat, pr_znacznik_czasowy');
```

- Zdefiniowanie dla grupy kolumn metody Latest timestamp

```
DBMS_REPCAT.ADD_UPDATE_RESOLUTION(
sname => 'reptest',
oname => 'pracownicy',
column_group => 'pracownicy_etat',
sequence_no => 1,
method => 'LATEST TIMESTAMP',
parameter_column_name => 'pr_znacznik_czasowy');
```


Latest timestamp - konfiguracja

- sequence_no ⇨ kolejność aplikowania wskazanej metody rozwiązywania konfliktu

- Utworzenie obiektów wspomagających replikację dla tabeli

```
DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT(
sname => 'reptest',
oname => 'pracownicy',
type => 'TABLE',
min_communication => TRUE);
```

- Włączenie replikacji

```
DBMS_REPCAT.RESUME_MASTER_ACTIVITY(gname => 'grupa_01');
```

Metody predefiniowane

- **Site priority**
  - **nadanie priorytetów węzłom**
  - **w sytuacji konfliktu wygrywa węzeł z największym priorytetem**
- **Additive**
  - **dla kolumn numerycznych**
  - **x_new:=x+delta**

Site priority - konfiguracja

1. **Do replikowanej tabeli dodać atrybut SITE przechowujący nazwę globalną bazy danych (perspektywa GLOBAL_NAME)**
2. **Do replikowanej tabeli dodać wyzwalacz nadający wartość SITE wstawianemu/modyfikowanemu rekordowi**
3. **Utworzyć grupę kolumn (column group) zawierającą atrybut SITE i pozostałe atrybuty, których watości mogą być konfliktowe**

```
DBMS_REPCAT.MAKE_COLUMN_GROUP (
      sname => ...,
      oname => ...,
      column_group => 'gr_kol1',
      list_of_column_names => '..., site');
```

Site priority - konfiguracja

4. **Zdefiniować grupę priorytetową dla węzła (site priority group)**

```
DBMS_REPCAT.DEFINE_SITE_PRIORITY (
      gname => 'nadrzędna grupa replikacji',
      name => 'site priority group');
```

5. **Nadać priorytety węzłom**

```
DBMS_REPCAT.ADD_SITE_PRIORITY_SITE (
      gname => 'nadrzędna grupa replikacji',
      name => 'site priority group',
      site => 'baza03.world',
      priority => 90);
```


Site priority - konfiguracja

6. **Wskazać metodę rozwiązywania konfliktów**

```
DBMS_REPCAT.ADD_UPDATE_RESOLUTION (
      sname => ...,
      oname => ...,
      column_group => 'gr_kol1',
      sequence_no => 1,
      method => 'SITE PRIORITY',
      parameter_column_name => 'site',
      priority_group => 'site priority group');
```

7. **Wygenerować systemowe obiekty wspierające replikację**

```
DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
      sname => ...,
      oname => ...,
      type => 'TABLE',
      min_communication => TRUE);
```

Metody użytkownika

- **Procedura DBMS_REPCAT.ADD_UPDATE_RESOLUTION, parametr FUNCTION_NAME**
  - **wskazanie nazwy metody rozwiązywania konfliktów zaimplementowanej przez użytkownika**

Monitorowanie pracy

- **Perspektywy słownikowe DEF...**
  - **kolejka odroczonych transakcji ⇨ DEFTRAN**
  - **lista węzłów dla odroczonych transakcji ⇨ DEFTRANDEST**
  - **kolejka błędów ⇨ DEFERROR**
  - **lista wywołań operacji synchronizacji ⇨ DEFCALL**
  - **harmonogram odświeżania ⇨ DEFSCHEDULE**

Modyfikowanie środowiska

- Zmiana węzła definicyjnego dla nadrzędnej grupy replikacji

```
DBMS_REPCAT.RELOCATE_MASTERDEF(
gname => 'grupa_01',
old_masterdef => 'BAZA01.WORLD',
new_masterdef => 'BAZA02.WORLD');
```

- Usunięcie węzła nadrzędnego

```
DBMS_REPCAT.REMOVE_MASTER_DATABASE(
gname => 'grupa_01',
master_list => 'BAZA02.WORLD');
```

- Usunięcie obiektu z nadrzędnej grupy replikacji

```
DBMS_REPCAT.DROP_MASTER_REPOBJECT(
oname => 'pracownicy',
sname => 'reptest',
type => 'TABLE');
```


Modyfikowanie środowiska

- Usunięcie nadrzędnej grupy replikacji

```
DBMS_REPCAT.DROP_MASTER_REPGROUP(
gname => 'grupa_02',
drop_contents => true,
all_sites => true);
```

- Usunięcie migawki z grupy migawek

```
DBMS_REPCAT.DROP_MVIEW_REPOBJECT(
sname => 'reptest',
oname => 'pracownicy_mv',
type => 'SNAPSHOT',
drop_objects => true);
```

- Usunięcie grupy migawek

```
DBMS_REPCAT.DROP_MVIEW_REPGROUP(
gname => 'grupa_01',
drop_contents => true);
```