VII Seminarium PLOUG
Warszawa
Marzec 2003

Replikacja danych w bazach danych Oracle9i

Bartosz Bębel, Robert Wrembel

Bartosz.Bebel@cs.put.poznan.pl, Robert.Wrembel@cs.put.poznan.pl

Politechnika Poznańska, Instytut Informatyki
ul. Piotrowo 3A, 60-965 Poznań

Abstrakt

Replikacja to proces kopiowania i utrzymywania obiektów (np. tabel) w połączonych bazach danych, tworzących jedno środowisko. Artykuł omawia podstawowe obiekty, biorące udział w replikacji: tabele, pakiety, migawki, grupy odświeżania i inne. Porusza zagadnienia zaawansowanej replikacji (opcji SZBD Oracle w wersji Enterprise): różne typy środowisk replikacji (replikacja multimaster, replikacja migawkowa, replikacja hybrydowa), przebieg propagacji zmian, obiekty wspomagające replikację, a także metody wykrywania i rozwiązywania konfliktów replikacji.

1. Replikacja danych

Intuicyjnie, replikacja danych polega na skopiowaniu pewnego zbioru danych z jednego miejsca, tzw. źródła, do miejsca docelowego. W kontekście relacyjnych baz danych źródłem danych jest tabela, którą dalej będziemy dalej nazywać **tabelą źródłową**. Obiektem docelowym jest również tabela, którą dalej będziemy nazywać **repliką**.

Replikację danych najczęściej wykorzystuje się w systemach rozproszonych baz danych, gdzie z jednego zdalnego węzła kopiuje się dane do innych zdalnych węzłów. Celem replikacji w tych systemach jest, po pierwsze, skrócenie czasu dostępu do danych. W tym przypadku uniezależniamy się od przepustowości i aktualnego obciążenia sieci, oraz od wydajności zdalnych węzłów i ich aktualnego obciążenia. Po drugie, dzięki replikacji uniezależniamy się od czasowej niedostępności węzłów i awarii sieci.

Wadą replikacji jest konieczność uaktualniania repliki w przypadku zmian w tabelach źródłowych. Taki proces uaktualniania będziemy również nazywali synchronizacją (ang. synchronization) lub odświeżaniem (ang. refreshing) repliki.

Replikacja wymaga utrzymywania połączeń pomiędzy węzłami. Węzeł kontaktuje się z innymi węzłami przy pomocy specjalnego obiektu schematu, noszącego nazwę łącznika bazy danych. Szczegółowy opis sposobu definiowania i stosowania łączników w *SZBD Oracle* zamieszczono w dokumentacji systemu.

1.1. Odświeżanie replik

Z odświeżaniem repliki są związane dwie kwestie techniczne: moment synchronizacji i sposób odświeżenia. Ze względu na moment synchronizacji odświeżanie dzielimy na **synchroniczne** i **asynchroniczne**. W pierwszym przypadku, zawartość repliki jest synchronizowana z zawartością jej tabel źródłowych natychmiast po modyfikacji danych źródłowych. W drugim przypadku, zawartość repliki jest synchronizowana albo na żądanie użytkownika, jest to tzw. odświeżanie manualne, albo automatycznie przez system zarządzania bazą danych – okresowo, z zadanym interwałem czasowym.

Ze względu na sposób, odświeżanie dzielimy na **pełne** (ang. complete, full) i **przyrostowe** (ang. incremental, fast). Odświeżanie pełne polega na skopiowaniu całej zawartości tabel źródłowych do repliki. Natomiast przy odświeżaniu przyrostowym do repliki kopiowane są wyłącznie zmiany dokonane w tabelach źródłowych od czasu ostatniej synchronizacji. Zmiany te będziemy dalej nazywać **deltami** (ang. deltas).

SZBD Oracle8i i *Oracle9i* wspiera mechanizm replikacji zarówno synchronicznej, jak i asynchronicznej oraz odświeżanie pełne i przyrostowe. W systemie *Oracle* replika jest implementowana jako tzw. **migawka** (ang. snapshot). W wersji *Oracle8i* wprowadzono pojęcie **perspektywy zmaterializowanej** (ang. materialized view), będącej synonimem migawki.

1.2. Migawka – perspektywa zmaterializowana

Jak wspomniano, mechanizmem wykorzystywanym w *SZBD Oracle* do replikacji danych jest migawka (perspektywa zmaterializowana). W standardowym przypadku definicja migawki składa się z następujących elementów: nazwy migawki, momentu wypełnienia migawki danymi, specyfikacji sposobu odświeżania, specyfikacji momentu rozpoczęcia automatycznego odświeżania, specyfikacji częstotliwości odświeżania, typu migawki i zapytania określającego zakres danych dostępnych w migawce.

Migawkę tworzy się poleceniem SQL `create snapshot` lub `create materialized view`, którego podstawową składnię przedstawiono poniżej. Pełną składnię tego polecenia Czytelnik znajdzie w [O9SQL].

```
create snapshot nazwa_migawki
build {immediate | deferred}
refresh sposób_odświeżania
        start with data_rozpoczęcia_odświeżania
        next częstotliwość_odświeżania
with {rowid | primary key}
as zapytanie;
```

Poniższe przykładowe polecenie definiuje migawkę o nazwie MV_SPRZEDAZ o następującej charakterystyce:

- wypełnienie danymi w momencie tworzenia (`build immediate`);
- odświeżanie pełne (`refresh complete`);
- moment rozpoczęcia odświeżania automatycznego - 1 minuta po utworzeniu migawki (`start with sysdate+(1/(24*60))`);
- częstotliwość odświeżania - 10 sekund (`next sysdate+(1/(24*60*6))`);
- udostępnia informacje o sprzedaży produktów w roku 2003 z tabeli wskazywanej łącznikiem `lab92`.

```
create materialized view mv_sprzedaz
build immediate
refresh complete
start with sysdate+(1/(24*60))
next sysdate+(1/(24*60*6))
as
select * from sprzedaz@lab92
where data like '%2003';
```

Moment wypełnienia migawki danymi

Jak wspomniano, w definicji migawki można wskazać moment jej wypełnienia danymi, to znaczy, można określić czy migawka ma zostać wypełniona danymi natychmiast po jej utworzeniu, czy też dopiero w momencie jej pierwszego odświeżenia. Do tego celu służą odpowiednio klauzule `build immediate` i `build deferred`. Pierwsza z nich jest klauzulą domyślną.

Przykładowo, poniższa migawka zostanie wypełniona danymi dopiero w momencie jej pierwszego odświeżenia, tj. 10 minut po jej utworzeniu.

```
create snapshot mv_sprzedaz_1
build deferred
refresh force
start with sysdate + (1/(24*6))
next sysdate+(1/(24*60))
with primary key
as
select produkt_id, l_sztuk, cena_jedn, data, sklep_id
from sprzedaz@lab92
where sklep_id=1;
```

Specyfikacja sposobu odświeżania

Specyfikacja sposobu odświeżania migawki (klauzula `refresh sposób_odświeżania`) umożliwia określenie, czy migawka ma być odświeżana w sposób pełny, czy przyrostowy.

Odświeżanie przyrostowe jest możliwe jeśli: (1) na tabeli źródłowej migawki został utworzony dziennik migawki (por. punkt „Dziennik migawki") i (2) migawka jest typu prostego.

Migawkę jest **typu prostego** jeśli zapytanie ją definiujące spełnia następujące warunki:

- bazuje na jednej tabeli;
- nie zawiera operacji łączenia tabeli samej z sobą;
- nie zawiera klauzul `start with` i `connect by`;
- nie zawiera operatorów zbiorowych (`union`, `union all`, `intersect`, `minus`);
- nie zawiera, w większości, przypadków funkcji grupowych.

W pewnych przypadkach migawki typu złożonego można również odświeżać przyrostowo. Jednak omówienie tych zagadnień wykracza poza zakres artykułu. Więcej informacji Czytelnik znajdzie w [WrBę03, O9Rep].

Wskazanie preferowanego sposobu odświeżania

Wybór sposobu odświeżania migawki realizuje się za pomocą klauzuli `refresh sposób_odświeżania`. Jej składnia jest następująca:

```
refresh {fast | complete | force}
```

Klauzula `refresh fast` definiuje odświeżanie przyrostowe, natomiast `refresh complete` definiuje odświeżanie pełne. Klauzula `refresh force` (przyjmowana domyślnie) umożliwia dynamiczny wybór sposobu odświeżania. Jeżeli możliwe będzie odświeżenie migawki w sposób przyrostowy, tzn. spełnione będą warunki niezbędne do takiego odświeżenia, to system zastosuje taki sposób. W przeciwnym przypadku zastosowane zostanie odświeżanie pełne.

Moment i częstotliwość odświeżania

Migawka może być odświeżana: (1) manualnie, czyli na żądanie użytkownika, (2) automatycznie po zatwierdzeniu transakcji modyfikującej dane źródłowe, (3) automatycznie z zadaną częstotliwością, lub (4) może nie być nigdy odświeżona.

W definicji migawki można określić moment i częstotliwość jej odświeżania. Służy do tego celu klauzula `refresh` o następującej składni:

```
refresh
        {fast | complete | force}] [{on demand | on commit}]
        [start with data_pierwszego_odświeżenia]
        [next częstotliwość_odświeżania]
```

Odświeżanie manualne

Użycie `on demand` w klauzuli `refresh` spowoduje utworzenie migawki odświeżanej manualnie – na żądanie użytkownika. Klauzule `fast`, `complete`, `force` można dowolnie stosować z `on demand`. Poniższy przykład prezentuje definicję migawki odświeżanej manualnie.

```
create materialized view mv_sprzedaz_2
build immediate
```

```
refresh on demand
as
select * from sprzedaz@lab92;
```

Manualne odświeżanie migawki realizuje się za pomocą procedury `REFRESH` znajdującej się w pakiecie `DBMS_SNAPSHOT`. Szersze mówienie tych zagadnień Czytelnik znajdzie w [WrBę03, O9Rep]

Odświeżanie automatyczne po zatwierdzeniu transakcji

Użycie `on commit` w klauzuli `refresh` spowoduje utworzenie migawki odświeżanej automatycznie po zatwierdzeniu transakcji modyfikującej tabele źródłowe migawki.

Klauzule `fast`, `complete`, `force` można stosować z `on commit`. Należy jednak zwrócić uwagę na fakt, że stosowanie klauzuli `on commit` jest ograniczone. Po pierwsze, nie można stosować tej klauzuli jeżeli zapytane definiujące perspektywę odwołuje się do tabel źródłowych za pośrednictwem łączników bazy danych. Po drugie, `on commit` w można stosować jedynie dla:

- migawek opartych o jedną tabelę, bez wyliczania agregatów;
- migawek, których zapytanie wyznacza agregaty w oparciu o pojedynczą tabelę;
- migawek których zapytanie wykorzystuje łączenie tabel, ale bez wyliczania agregatów.

Klauzula `on commit` nie może być stosowana ani z klauzulą `start with` ani z `next`.

Jeżeli perspektywa jest definiowana z klauzulami `on commit` i `with primary key`, wówczas zapytanie musi wybierać wszystkie atrybuty wchodzące w skład klucza podstawowego jej tabeli bazowej.

Odświeżanie automatyczne z zadaną częstotliwością

Jeżeli migawka ma być odświeżana automatycznie z zadaną częstotliwością, wówczas w jej definicji należy umieścić klauzule: `start with` i `next`. Składnia klauzuli `refresh` jest w tym przypadku następująca:

```
refresh {fast | complete | force}
start with data
next częstotliwość
```

Klauzula `start with data` umożliwia określenie momentu rozpoczęcia odświeżania automatycznego, po utworzeniu migawki. Klauzula `next częstotliwość` umożliwia wyspecyfikowanie częstotliwości odświeżania. Jeżeli zostanie pominięta klauzula `start with`, wówczas odświeżanie automatyczne rozpoczyna się natychmiast po utworzeniu migawki.

Przykładowo, poniższy fragment definicji migawki specyfikuje: pierwsze odświeżenie 20 sekund po utworzeniu migawki i odświeżanie co 5 sekund.

```
start with sysdate + (1/(24*60*3))
next sysdate + (1/(24*60*12))
```

Kolejny przykład ilustruje fragment definicji migawki odświeżanej każdej środy o godzinie 18:30, przy czym pierwsze odświeżenie migawki nastąpi następnego dnia, licząc od momentu jej utworzenia, o godzinie 9:00.

```
start with trunc (sysdate + 1) + 9/24
next trunc (next_day (sysdate, 'Wednesday')) + 18/24 + (1/(24*2))
```

Należy zwrócić uwagę, że ani z klauzulą `start with`, ani z `next` nie może wystąpić klauzula `on commit`. Natomiast klauzula `on demand` może wystąpić z klauzulami `fast`, `complete`, `force` oraz `start with` i `next`.

Brak odświeżania

Możliwe jest również zdefiniowanie migawki, która nigdy nie zostanie odświeżona. Wykorzystuje się w tym celu klauzulę `never refresh`. Tak utworzona migawka zostanie wypełniona danymi raz – w momencie jej utworzenia. Manualne wywołanie procedury odświeżającej dla takiej migawki jest możliwe, lecz nie powoduje jej odświeżenia. Klauzula `never refresh` (jeśli jest stosowana) zastępuje klauzulę `refresh`.

Przykład migawki wykorzystującej klauzulę `never refresh` przedstawiono poniżej.

```
create materialized view mv_test
never refresh
as
select * from sklepy@lab92;
```

Typ migawki

Typ migawki (klauzula `with typ_migawki`) określa sposób identyfikowania rekordów w tabeli źródłowej i w migawce. Możliwe są tu dwa sposoby identyfikacji. Pierwszy z nich wykorzystuje fizyczne adresy rekordów na dysku – ROWID, a migawkę wykorzystującą te adresy będziemy dalej nazywać migawką typu *ROWID*. Ponieważ ROWID zawiera w sobie adres pliku, numer bloku w ramach tego pliku i adres rekordu w bloku, więc przeniesienie tabeli np. z jednego pliku do innego wiąże się ze zmianą adresów rekordów tej tabeli. Podobnie, eksport tabeli wraz z jej zawartością, a następnie jej import powodują zmianę wartości ROWID. Jeśli na takiej eksportowanej tabeli była zdefiniowana migawka, to po imporcie tej tabeli gubione są powiązania między rekordami tabeli, a odpowiadającymi im rekordami migawki. W rezultacie, migawki nie da się odświeżyć przyrostowo, a jedynym sposobem jej synchronizacji po imporcie tabeli jest odświeżenie pełne.

Takiej wady nie posiada migawka typu *primary key*. Tu, do identyfikacji rekordów źródłowych i migawki są stosowane wartości klucza podstawowego tabeli źródłowej. Począwszy od wersji *Oracle rel. 8.0* udostępnione zostały migawki typu *primary key*, jako domyślny typ. Stosowanie tego typu migawek jest zalecane przez producenta oprogramowania.

Należy zwrócić uwagę, że migawka typu *primary key* musi udostępniać wszystkie atrybuty wchodzące w skład klucza tabeli źródłowej, w przeciwnym przypadku tworzenie migawki zakończy się błędem.

Implementacja migawki

Migawka w *Oracle8i* i *Oracle9i* jest implementowana jako: (1) tabela o nazwie identycznej z nazwą migawki i (2) indeks założony na atrybucie jednoznacznie identyfikującym rekordy tej tabeli. Przykładowo, rozważmy migawkę typu *ROWID* zdefiniowaną jak niżej.

```
create materialized view mv_sprzedaz
refresh force
start with sysdate
next sysdate + (1/(24*60*30))
with rowid
as
select produkt_id, l_sztuk, cena_jedn, data, sklep_id
from sprzedaz@lab92;
```

W rezultacie wykonania tego polecenia, w bazie danych, zostanie utworzona tabela MV_SPRZEDAZ i indeks. Tabela MV_SPRZEDAZ zawiera atrybuty wyspecyfikowane w klauzuli `select` i jeden dodatkowy atrybut o nazwie M_ROW$$. Atrybut ten pojawia się w tabeli MV_SPRZEDAZ ponieważ utworzono migawkę z klauzulą `with rowid`. Wartość M_ROW$$ jest wykorzystywana do identyfikacji rekordów migawki i odpowiadających im rekordów w tabeli źródłowej. Indeks jest zdefiniowany na atrybucie M_ROW$$.

W przypadku utworzenia perspektywy z klauzulą `with primary key` rekordy tabeli źródłowej i rekordy migawki są identyfikowane za pomocą atrybutów wchodzących w skład klucza tabeli źródłowej. W tym przypadku tabela implementująca migawkę nie zawiera atrybutu M_ROW$$, a indeks jest zakładany na atrybutach kluczowych.

Słownik bazy danych opisujący migawki

Informacje na temat migawek istniejących w systemie bazy danych można uzyskać za pomocą zapytania skierowanego do perspektyw słownika bazy danych: *DBA_*, *ALL_*, *USER_MVIEWS*, *DBA_*, *ALL_*, lub *USER_SNAPSHOTS* [O9SQL].

Informacje o wszystkich migawkach zarejestrowanych w bazie zdalnej są dostępne za pomocą perspektywy słownika bazy danych *DBA_*, *ALL_*, *USER_REGISTERED_MVIEWS*, *DBA_*, *ALL_*, lub *USER_REGISTERED_SNAPSHOTS*.

Informacje na temat czasu ostatniego odświeżenia migawki są dostępne za pomocą perspektywy słownika danych *DBA_*, *ALL_*, *USER_MVIEW_REFRESH_TIMES*, *DBA_*, *ALL_*, lub *USER_SNAPSHOT_REFRESH_TIMES*.

1.3. Dziennik migawki

Dziennik migawki (ang. snapshot log, materialized view log) służy do rejestrowania zmian dokonanych w zawartości tabeli – delt - dla której zdefiniowano taki dziennik. Tabelę tę będziemy dalej nazywać **tabelą bazową dziennika**. Tabela bazowa może posiadać wyłącznie jeden dziennik. Każde polecenie DML jest rejestrowane w dzienniku. Informacje te są następnie wykorzystywane do odświeżania przyrostowego migawki korzystającej z tabeli bazowej dziennika.

Dziennik migawki tworzy się poleceniem SQL `create snapshot log` lub `create materialized view log`, których podstawową składnię przedstawiono poniżej. Pełną składnię tego polecenia Czytelnik znajdzie w [O9SQL].

```
create snapshot log
on tabela_bazowa
[with {      primary key |
      ROWID |
      primary key, ROWID }];
```

Domyślna klauzula `with primary key` służy do zdefiniowania dziennika dla migawki typu *primary key*, natomiast klauzula `with ROWID` powoduje utworzenie dziennika dla migawki typu *ROWID*.

Informacje na temat dzienników migawek istniejących w systemie bazy danych można uzyskać za pomocą zapytania skierowanego do perspektyw słownika bazy danych: *DBA_*, *ALL_*, *USER_MVIEW_LOGS*, bądź *DBA_*, *ALL_*, *USER_SNAPSHOT_LOGS*.

2. Zaawansowana replikacja

Zaawansowana replikacja (ang. advanced replication) to opcja *SZBD Oracle8i* i *Oracle9i* w wersji *Enterprise*, umożliwiająca zbudowanie wydajnych środowisk replikacji o różnych konfigu-

racjach. Należy podkreślić, że środowisko mogą tworzyć instancje baz danych *Oracle* na różnych platformach systemowych.

Przedstawione teraz zostaną definicje podstawowych składników systemu replikacji: obiektów replikacji, grup replikacji i węzłów replikacji.

2.1. Podstawowe definicje

Obiekt replikacji to obiekt schematu, powielony w bazach danych tworzących środowisko replikacji. Operacje, wykonane na obiekcie replikacji w jednej bazie danych środowiska, zostają zaaplikowane kopiom tego obiektu, istniejącym w pozostałych bazach środowiska. Możliwe jest replikowanie następujących typów obiektów: tabele (zwykłe, ale także partycjonowane, zorganizowane jako indeksy, obiektowe, zawierające kolumny bazujące na typach zdefiniowanych przez użytkownika), indeksy, perspektywy (zwykłe i obiektowe), pakiety wraz z ciałami, procedury i funkcje, typy zdefiniowane przez użytkownika oraz indeksowe, wyzwalacze, synonimy oraz operatory.

Celem ułatwienia zadań administracyjnych, logicznie powiązane ze sobą obiekty replikacji łączy się w zbiory, nazywane **grupami replikacji**. Jedna grupa najczęściej zawiera obiekty jednego schematu (np. wykorzystywane przez jedną aplikację), jednak nie ma żadnych przeszkód, aby grupa replikacji zawierała obiekty z różnych schematów. Dla jednego schematu można również zdefiniować wiele grup replikacji, jednak jeden obiekt może należeć tylko do jednej grupy.

Środowisko zaawansowanej replikacji tworzą dwa rodzaje węzłów: **węzły nadrzędne** (ang. master site) oraz **węzły migawkowe** (ang. materialized view site). Jak już wcześniej wspomniano, grupy replikacji są definiowane w węzłach replikacji. Węzeł środowiska może być węzłem nadrzędnym dla jednej grupy replikacji i jednocześnie węzłem migawkowym dla innej grupy replikacji. Jednak jeden węzeł nie może być jednocześnie węzłem nadrzędnym i węzłem migawkowym dla tej samej grupy replikacji. Dokładne definicje węzła nadrzędnego i węzła migawkowego zostaną przedstawione przy omawianiu poszczególnych typów środowisk replikacji.

2.2. Typy środowisk replikacji

W zależności od wzajemnych relacji między węzłami replikacji wyróżniamy następujące typy środowisk replikacji: **replikację multimaster**, **replikację migawkową** oraz **replikację hybrydową**.

W środowisku **replikacji multimaster** (ang. multimaster replication) wszystkie węzły replikacji są równoważne - każdy węzeł jest węzłem nadrzędnym. Oznacza to, że zmiana wprowadzona w którymkolwiek węźle zostaje zaaplikowana we wszystkich pozostałych węzłach środowiska, a replikowane obiekty są identyczne we wszystkich węzłach - zarówno pod względem struktury, jak i zawartości.

Środowisko **replikacji migawkowej** (ang. materialized view replication) najczęściej tworzy jeden węzeł nadrzędny oraz wiele węzłów migawkowych. Grupa replikacji w węźle nadrzędnym zawiera obiekty (tabele), będące źródłami danych dla migawek, należących do grup replikacji w węzłach migawkowych. Węzłem nadrzędnym może być również węzeł migawkowy, wówczas źródłami danych są zdefiniowane w nim migawki (jest to tzw. architektura wielowarstwowa).

Replikacja hybrydowa (ang. hybrid replication) jest kombinacją dwóch wcześniej wymienionych typów środowisk replikacji. Środowisko replikacji hybrydowej zawiera wiele węzłów nadrzędnych wraz z wieloma węzłami migawkowymi.

2.3. Przesłanie i zaaplikowanie zmian

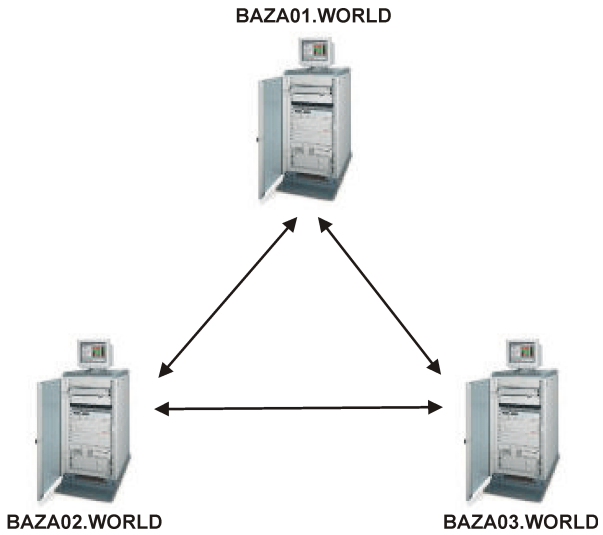
Jak już wcześniej wspomniano, ze względu na moment synchronizacji replikację możemy podzielić na synchroniczną i asynchroniczną. W zaawansowanej replikacji przy **replikacji synchronicznej** proces przesłania i zastosowania zmian jest częścią transakcji węzła lokalnego. Pomyślne zakończenie transakcji lokalnej (zatwierdzenie) oznacza, że zmiany zostały z powodzeniem przesłane i zaaplikowane we wszystkich pozostałych węzłach środowiska. Wystąpienie błędów przy przesyłaniu zmian do któregokolwiek z węzłów (np. na skutek awarii sieci) powoduje niemożność dokończenia lokalnej transakcji węzła, który zapoczątkował cały proces i w konsekwencji jej wycofanie. Stąd replikacja synchroniczna wymaga stałych i pewnych połączeń pomiędzy węzłami środowiska

Przy **replikacji asynchronicznej** lokalne zmiany z zatwierdzonej transakcji węzła są najpierw składowane w lokalnej strukturze, zwanej kolejką transakcji, z której następnie są okresowo pobierane i przesyłane do pozostałych węzłów środowiska. Operacje przesłania i zaaplikowania zmian są realizowane jako osobna transakcja. Dlatego też utrata połączenia pomiędzy niektórymi węzłami środowiska nie powoduje wstrzymania replikacji pomiędzy pozostałymi węzłami (nie wstrzymuje również lokalnych operacji w węźle).

W następnej części artykułu rozpoczniemy szczegółowy opis pierwszego typu środowiska – replikacji multimaster.

2.4. Replikacja multimaster

W środowisku **replikacji multimaster** wszystkie węzły środowiska są równoważne. Każdy węzeł środowiska jest węzłem nadrzędnym i utrzymuje połączenie ze wszystkimi pozostałymi węzłami przy pomocy łączników bazodanowych. Przykładowe środowisko replikacji przedstawiono na rysunku 1. Składa się ono z trzech węzłów: **BAZA01.WORLD**, **BAZA02.WORLD** oraz **BAZA03.WORLD**.



Rysunek 1. Przykład środowiska replikacji multimaster

Jeśli w jednym z węzłów zostanie zatwierdzona transakcja modyfikująca obiekt replikacji, wówczas rozpoczyna się proces, którego celem jest zaaplikowanie takich samych zmian w kopiach tego obiektu we wszystkich pozostałych węzłach środowiska. Węzły współpracują ze sobą w celu utrzymania **identycznego stanu** (struktury i zawartości) obiektów replikacji w każdym węźle. Czas, po którym replikowane schematy staną się identyczne we wszystkich węzłach zależy od wy-

branego trybu pracy środowiska (replikacja asynchroniczna lub replikacja synchroniczna). Szczegółowy przebieg procesu wysyłania i aplikowania lokalnych zmian węzła w pozostałych węzłach środowiska zostanie przedstawiony w dalszej części tego rozdziału.

Domyślnym sposobem działania replikacji multimaster jest **replikacja na poziomie rekordu**. Rozważmy następującą sytuację. Użytkownik w bazie **BAZA01.WORLD** wykonał operację *update*, modyfikującą 100 rekordów replikowanej tabeli PRACOWNICY. Do każdego z pozostałych węzłów środowiska zostanie przesłana informacja o zmianie każdego pojedynczego rekordu, czyli każdy węzeł otrzyma 100 wiadomości o zmianie rekordów w relacji PRACOWNICY. Alternatywnym sposobem działania replikacji multimaster jest **replikacja proceduralna** – w tym wypadku replikowane są wywołania procedur, które realizują operacje na obiektach (replikacja proceduralna zostanie szerzej opisana w dalszej części artykułu).

Architektura węzła replikacji multimaster

Opiszemy teraz architekturę węzła replikacji multimaster. Najpierw szczegółowo scharakteryzujemy obiekty, które biorą udział w replikacji.

Obiekty replikacji

Tabela

Tabela jest podstawowym i najczęściej wykorzystywanym obiektem replikacji. Replikacja obsługuje zarówno zmiany zawartości tabeli (czyli modyfikacje danych), jak i zmiany jej struktury. Nie ma możliwości ograniczenia zakresu replikacji danych – przez to zawartość replikowanych tabel we wszystkich węzłach środowiska replikacji multimaster jest zawsze taka sama. Każda tabela, która ma brać udział w replikacji, powinna posiadać zdefiniowany klucz podstawowy (jest niezbędny do jednoznacznej identyfikacji rekordów w replikowanych tabelach). W przypadku jego braku należy wskazać klucz alternatywny.

Indeks

Replikacja indeksu polega jedynie na przesłaniu jego definicji do węzła zdalnego. Tam, po pomyślnym utworzeniu, indeks zachowuje się tak samo, jak indeks lokalny (nie będący obiektem replikacji).

Pakiet i ciało pakietu

Przy replikacji pakietu lokalne wywołanie procedury z pakietu na jednym z węzłów replikacji jest przesyłane do wszystkich pozostałych węzłów środowiska. Taki tryb działania replikacji nosi nazwę replikacji proceduralnej (zostanie ona szerzej omówiona w dalszej części artykułu).

Procedura i funkcja

Replikacja procedury lub funkcji, zdefiniowanych poza pakietem, skutkuje jedynie rozesłaniem definicji obiektu do wszystkich węzłów replikacji i jego utworzeniem w lokalnych schematach węzłów. Jednak lokalne wywołanie takiej procedury lub funkcji nie jest przesyłane do pozostałych węzłów środowiska. Replikowana procedura lub funkcja zachowuje się tak, jak lokalny obiekt węzła.

Wyzwalacz

Wyzwalacze, zdefiniowane lokalnie dla replikowanej tabeli, nie są automatycznie replikowane razem z tabelą. Należy jawnie wskazać, że wyzwalacz ma być również obiektem replikacji. Dopiero wtedy definicja wyzwalacza zostanie przesłana do węzłów zdalnych i wyzwalacz zostanie utworzony na zdalnych replikach tabeli. Będzie zachowywał się w taki sam sposób, jak wyzwalacz zdefiniowany lokalnie.

Perspektywa, perspektywa obiektowa, synonim, typ i operator zdefiniowany przez użytkownika, typ indeksowy

Replikacja wymienionych obiektów polega tylko na rozesłaniu ich definicji do wszystkich węzłów środowiska. Po utworzeniu replikowany obiekt zachowuje się identycznie jak obiekt lokalny.

Obiekty wspomagające replikację

Zmiany, jakie wystąpią w replikowanym schemacie w jednym z węzłów środowiska muszą zostać przechwycone i przesłane do pozostałych węzłów. Do realizacji tych zadań *SZBD Oracle* tworzy **obiekty wspomagające replikację**. Są to **wewnętrzne wyzwalacze** (ang. internal trigger) i **wewnętrzne procedury** (ang. internal procedure). Wewnętrzne procedury zostają zgrupowane w pakiet, którego nazwa koresponduje z nazwą obiektu replikacji (np. dla replikowanej tabeli PRACOWNICY *SZBD Oracle* tworzy pakiet `PRACOWNICY$RP`). Generacja obiektów wspomagających replikację odbywa się automatycznie po wskazaniu obiektu bazodanowego jako obiektu replikacji. Omówimy teraz obiekty wspomagające replikację, a także przedstawimy sposób ich współdziałania w procesie replikacji.

Replikacja danych w *SZBD Oracle* jest realizowana przez przesyłanie pomiędzy węzłami **zdalnych wywołań procedur** (ang. remote procedure call – **RPC**). Zadaniem wewnętrznego wyzwalacza tabeli jest przechwytywanie lokalnych operacji modyfikacji danych tabeli. Dla każdej operacji DML, składającej się na lokalną transakcję, wyzwalacz generuje zdalne wywołania procedury (RPC). RPC zawiera definicję operacji, jaka ma zostać wykonana w zdalnym węźle, wraz z porcją danych, których ta operacja dotyczy. Wszystkie RPC dla lokalnej transakcji tworzą **odroczoną transakcję** (ang. deferred transaction) i są przechowywane w lokalnej **kolejce odroczonych transakcji** (ang. deferred transaction queue). Każdy węzeł posiada tylko jedną kolejkę odroczonych transakcji, niezależnie od liczby replikowanych obiektów. Odbiorcą RPC w zdalnym węźle jest wewnętrzna procedura. Dotarcie RPC do węzła powoduje wykonanie wewnętrznej procedury, która aplikuje lokalnie zmiany, pochodzące ze zdalnego węzła środowiska replikacji. Ponieważ wszystkie RPC pamiętają, z której transakcji pochodzą, zaaplikowanie zmian w węźle zdalnym również odbywa się w ramach transakcji.

Oprócz kolejki transakcji w węźle utrzymywana jest również **kolejka wywołań** (ang. call queue). Zapisywane są w niej wszystkie wywołania replikowanych procedur w ramach odroczonych transakcji. W propagacji zmian w replikowanej tabeli do kolejki trafia wywołanie procedury wewnętrznej, odpowiedzialnej za replikację zmian, natomiast w przypadku replikacji proceduralnej – jawne wywołanie replikowanej procedury.

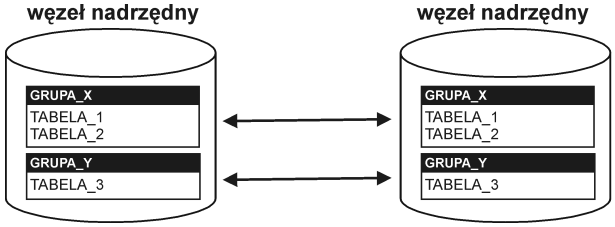
Kolejki odroczonych transakcji i wywołań mają zastosowanie jedynie przy replikacji asynchronicznej.

Kolejnym obiektem, pełniącym ważną rolę w procesie replikacji, jest **kolejka zadań** (ang. job queue). Jest to tabela, przechowująca informacje o lokalnych zadaniach węzła. Typowe zadania w replikacji to pobranie transakcji z lokalnej kolejki odroczonych transakcji i przesłanie jej do węzłów zdalnych oraz usunięcie z kolejki transakcji odroczonych transakcji z powodzeniem zastosowanych w węzłach zdalnych.

Jeśli przesłanie lokalnej transakcji do węzła zdalnego i tam jej zaaplikowanie nie zakończy się powodzeniem, informacja o błędzie zostaje zapamiętana w **kolejce błędów** (ang. error queue) węzła docelowego. Administrator, przeglądając kolejkę błędów, może zdecydować o ponownej próbie zastosowania transakcji lub o jej usunięciu.

Nadrzędne grupy replikacji

W replikacji multimaster grupa replikacji nosi nazwę **grupy nadrzędnej** (ang. master group). Jest podstawowym obiektem administracyjnym. Przykład grup nadrzędnych został przedstawiony na rysunku 2.



Rysunek 2. Grupy nadrzędne w replikacji multimaster

Definicja grupy nadrzędnej zawiera listę obiektów replikacji wraz z listą węzłów docelowych, do których wybrane obiekty będą replikowane. Grupę zostaje zdefiniowana w wybranym węźle środowiska, nazywanym **węzłem definicyjnym** (ang. definition site), natomiast pozostałe węzły kopiują schemat grupy z węzła definicyjnego. Wszystkie czynności administracyjne grupy nadrzędnej realizuje się tylko w jej węźle definicyjnym. Środowisko replikacji może składać się z wielu grup nadrzędnych, zdefiniowanych w różnych węzłach środowiska (czyli posiadających różne węzły definicyjne). Grupy replikacji są od siebie niezależne – sposób działania jednej nie wpływa na działanie pozostałych.

Łączniki bazodanowe

Każdy węzeł środowiska replikacji multimaster musi być połączony ze wszystkimi pozostałymi węzłami. Stąd, jeśli w środowisku mamy N węzłów, każdy z nich musi posiadać N-1 łączników bazodanowych (dla przykładu z rysunku 1 – 3 węzły, każdy z nich posiada 2 łączniki bazodanowe).

Minimalna konfiguracja wymaga, aby został zdefiniowany łącznik bazodanowy pomiędzy schematem administratora replikacji w węźle lokalnym a schematem administratora replikacji w węźle zdalnym. Jednak w przypadku konfiguracji, w której propagację transakcji realizuje inny użytkownik niż administrator, musi on mieć swój własny zestaw łączników bazodanowych.

Mechanizmy administracyjne replikacji multimaster

Funkcje administracyjne

Wymienione teraz zostaną funkcje, których realizacja jest niezbędna do właściwego działania procesu replikacji multimaster. Funkcje te mogą być realizowane przez jednego użytkownika lub rozdzielone pomiędzy dwóch lub trzech użytkowników. Rozdzielenie funkcji pomiędzy kilku użytkowników może przyczynić się do zwiększenia poziomu bezpieczeństwa środowiska.

Administrator replikacji (ang. replication administrator) to użytkownik, wykonujący wszystkie operacje administracyjne środowiska replikacji. Są to 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).

Sugeruje się, aby w węźle istniał tylko jeden administrator replikacji, niezależnie od liczby replikowanych schematów. Możliwe jest jednak również takie rozwiązanie, w którym za replikację różnych schematów będą odpowiadali różni użytkownicy – administratorzy replikacji.

Nadawca (ang. propagator) to użytkownik, którego zadaniem jest wysyłanie transakcji z lokalnej kolejki transakcji do węzłów zdalnych. W węźle może istnieć tylko jeden użytkownik, będący nadawcą.

Z kolei **odbiorca** (ang. receiver) to użytkownik odpowiedzialny za odbiór zdalnych transakcji i zaaplikowanie ich lokalnie.

Stany replikacji multimaster

Środowisko replikacji może znajdować się w jednym z trzech stanów:

- **normalne działanie** (ang. normal) – replikacja jest włączona, lokalne transakcje węzłów są propagowane do węzłów zdalnych.
- **wyłączanie** (ang. quiescing) – jest trybem przejściowym pomiędzy normalnym działaniem replikacji a wyłączeniem. W trybie tym użytkownicy w węzłach środowiska nie mogą rozpoczynać żadnych nowych lokalnych transakcji dla replikowanych obiektów (czyli operacji DML dla replikowanych tabel i wywołań replikowanych procedur). Dozwolone jest jedynie wykonywanie zapytań. Stan wyłączania trwa do czasu, aż wszystkie transakcje z kolejki odroczonych transakcji zostaną z powodzeniem przesłane do węzłów zdalnych. Wtedy replikacja przechodzi w stan wyłączenia.
- **wyłączenie** (ang. quiesced) – replikacja zostaje zatrzymana. W kolejce odroczonych transakcji nie ma już żadnych transakcji, które nie zostały z powodzeniem przesłane do węzłów zdalnych. Podobnie jak w stanie wyłączania, użytkownicy nie mogą definiować żadnych transakcji z udziałem obiektów replikacji. Ma to zapobiec utracie zgodności stanów odpowiadających sobie obiektów replikacji w różnych węzłach środowiska. Stan wyłączenia służy głównie do wykonywania czynności administracyjnych (np. dodania nowego obiektu do grupy replikacji).

Zmiany stanów replikacji dokonuje się na poziomie grupy nadrzędnej w jej węźle definicyjnym, niezależnie od innych grup nadrzędnych. Oznacza to, że jeśli środowisko replikacji tworzy kilka grup nadrzędnych, ich stany są niezależne (jedne grupy mogą być wyłączone, podczas gdy inne działają normalnie).

Przechodzenie pomiędzy poszczególnymi stanami replikacji realizowane jest za pomocą dwóch mechanizmów. Mechanizm **zawieszenia** (ang. suspend) replikacji to sekwencja operacji, realizujących przejście grupy nadrzędnej ze stanu normalnego do wyłączenia. Zawieszenie może zostać wykonane tylko dla grupy nadrzędnej, będącej aktualnie w stanie normalnym. Natomiast operacja odwrotna, czyli przejście grupy nadrzędnej ze stanu wyłączenia do stanu normalnego realizowane jest przez mechanizm **przywrócenia** (ang. resume) replikacji.

Żądania administracyjne

Konfiguracja i zarządzanie środowiskiem replikacji odbywa się przez wywoływanie odpowiednich procedur i funkcji z pakietów, tworzących *Replication Management API*. Przykładem może być utworzenie w węźle nadrzędnej grupy replikacji – to zadanie realizuje procedura `CREATE_MASTER_REPGROUP` z pakietu `DBMS_REPCAT`. Każde takie uruchomienie procedury czy funkcji nosi nazwę **żądania administracyjnego** (ang. administrative request).

Żądania administracyjne, uruchamiane w węźle definicyjnym grupy replikacji, muszą zostać przesłane do tych wszystkich pozostałych węzłów środowiska, które biorą udział w replikacji obiektów z grupy (np. utworzenie grupy replikacji w węźle definicyjnym powoduje przesłanie żądań utworzenia odpowiadających grup w pozostałych węzłach środowiska). Przesyłanie żądań pomiędzy

węzłami jest realizowane **synchronicznie** przy pomocy wewnętrznych mechanizmów *SZBD Oracle*. Dzięki zastosowaniu synchronicznej propagacji żądań *SZBD Oracle* zapewnia, że wszystkie węzły zostaną powiadomione o wprowadzonej zmianie w konfiguracji. Jeśli jednak przesłanie nie zakończy się powodzeniem, wówczas zwracany jest komunikat o błędzie, a transakcja, w ramach której żądanie zostało wywołane, jest wycofywana.

Wszystkie żądania administracyjne, jakie zostały odebrane przez węzeł, są dostępne w perspektywie *DBA_REPCATLOG*. Perspektywa ta nazywana jest **kolejką żądań administracyjnych** (ang. administrative request queue). Tam też jest zapisywany stan (status) realizacji żądania. Jeśli żądaniu towarzyszyło polecenie DDL, zostaje ono zapisane w perspektywie *DBA_REPDDL*. Każdy węzeł przesyła status żądań administracyjnych do węzła definicyjnego grupy replikacji. Możliwe są następujące stany żądań administracyjnych:

- **READY**. Żądanie jest gotowe do wykonania. Jeśli stan żądania w kolejce pozostaje przez długi czas **READY**, może to oznaczać dwie sytuacje. Po pierwsze, żądanie może czekać na realizację wcześniejszych żądań z kolejki. Po drugie, żądanie może czekać na wolny proces, który je uruchomi. W drugim przypadku administrator może ręcznie uruchomić żądanie wykonując procedurę `DO_DEFFERED_REPCAT_ADMIN` z pakietu `DBMS_REPCAT`.
- **AWAIT_CALLBACK**. Żądanie oczekuje na potwierdzenie pomyślnego zakończenia wykonania innego żądania, realizowanego w innym węźle środowiska. Gdy żądanie otrzyma potwierdzenie, jego status zostaje zmieniony lub zostaje usunięte z kolejki żądań. Stan **AWAIT_CALLBACK** jest możliwy wyłącznie w węźle definicyjnym grupy replikacji.
- **ERROR**. Określa, że żądanie nie zostało wykonane z powodzeniem. Numer błędu określa wówczas kolumna *errnum*, a jego tekst kolumna *message* w *DBA_REPCATLOG*.
- **DO_CALLBACK**. Stan ten określa, że żądanie musi poinformować inne żądanie w węźle definicyjnym o swoim pomyślnym zakończeniu lub niepowodzeniu. Ten stan jest możliwy tylko dla żądań w węzłach innych niż węzeł definicyjny grupy replikacji.

Kolejka żądań administracyjnych w każdym z węzłów zawiera tylko te żądania, które mają być zrealizowane właśnie w tym węźle. Natomiast kolejka żądań w węźle definicyjnym zawiera wszystkie żądania, które mają zostać wykonane w każdym z węzłów środowiska. Dzięki temu administrator może monitorować stany żądań we wszystkich węzłach środowiska replikacji.

Żądanie administracyjne, wykonane w kontekście grupy replikacji w jej węźle definicyjnym, zostaje synchronicznie przesłane do węzłów, biorących udział replikacji obiektów z grupy. Oczywiście żądania zostają zapisane w lokalnych kolejkach żądań. Uruchomienie żądań z kolejki następuje asynchronicznie. Żądanie, które zostało wykonane w węźle definicyjnym, może czekać na pomyślne zakończenie żądań w pozostałych węzłach (czyli być w stanie **AWAIT_CALLBACK**). Jeśli wykonanie żądania zakończy się z powodzeniem we wszystkich węzłach, jest ono usuwane z kolejek żądań zarówno w węźle definicyjnym, jak i w pozostałych węzłach środowiska. Jeśli jednak wykonanie żądania w węźle nie będącym węzłem definicyjnym nie zakończy się sukcesem, wówczas *SZBD Oracle* usuwa to żądanie z kolejki żądań węzła i zmienia stan odpowiadającego zdarzenia w węźle definicyjnym z **AWAIT_CALLBACK** na **ERROR**.

Proces replikacji multimaster

W rozdziale tym zostanie omówiony sposób, w jaki zmiany obiektów replikacji na jednym z węzłów środowiska są przesyłane i aplikowane w pozostałych węzłach. W zależności od momentu przesłania zmian wyróżniamy replikację asynchroniczną i replikację synchroniczną.

Replikacja asynchroniczna

Główną cechą asynchronicznej replikacji multimaster jest rozdzielenie transakcji, na którą składają się operacje modyfikacji lokalnych replik obiektów (lokalne operacje DML), od transakcji przesyłającej te modyfikacje do replik obiektów w zdalnych węzłach środowiska. Operacje z lokalnej transakcji węzła zostają zapamiętane lokalnie, a następnie, co pewien okres czasu, zostają przesłane do węzłów zdalnych. Stąd wynika ważna cecha replikacji asynchronicznej – replikowane schematy we wszystkich węzłach środowiska uzyskują jednakowy stan (zbieżność) dopiero po pewnym czasie.

Rozpatrzymy przykładowy proces propagacji operacji, składających się na lokalną transakcję w węźle **BAZA01.WORLD** z rysunku 1. Załóżmy, że w skład transakcji wchodzą następujące operacje:

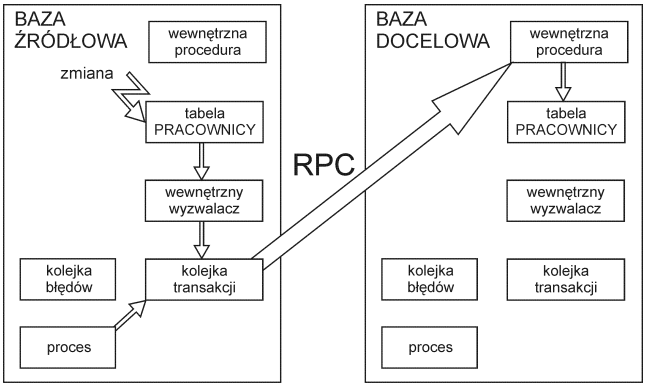
1. wstawienie rekordu do tabeli PRACOWNICY (operacja *insert*),
2. modyfikacja pięciu rekordów z tabeli PRACOWNICY (operacja *update*),
3. usunięcie trzech rekordów z tabeli ETATY (operacja *delete*).

Po zatwierdzeniu zmian rozpoczyna się proces propagacji zmian.

1. Każdy rekord, który zostaje wstawiony, zmodyfikowany czy też usunięty z replikowanej tabeli przez lokalną operację DML, powoduje uruchomienie wewnętrznego wyzwalacza związanego z replikowaną tabelą. 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 (na liście są wszystkie węzły środowiska oprócz węzła, dla którego transakcja jest transakcją lokalną). W rozpatrywanym przykładzie kolejka odroczonych transakcji będzie zawierała jedno RPC dla operacji wstawienia rekordu, pięć RPC dla operacji modyfikacji oraz trzy dla operacji usunięcia rekordów. Na liście węzłów docelowych dla transakcji będą znajdowały się węzły **BAZA02.WORLD** i **BAZA03.WORLD**.
2. Co określony czas zostaje uruchomiony 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. Przesłanie może również zostać wykonane na żądanie administratora.
3. Odroczona transakcja dociera do węzła docelowego. RPC, składające się na transakcję, zostają odebrane przez wewnętrzny pakiet i wykonane lokalnie. Skutkuje to zaaplikowaniem zdalnych operacji (czyli operacji z kroku 1) w lokalnych kopiach replikowanych obiektów. Jeśli któraś z operacji składających się na odroczoną transakcję nie może być z powodzeniem wykonana lokalnie, trafia do lokalnej kolejki błędów. Niepowodzenie może być spowodowane wystąpieniem konfliktu (np. próbą wstawienia do repliki tabeli rekordu, który narusza ograniczenia integralnościowe). W takim przypadku zostaje wykonana procedura rozwiązywania konfliktów. Wykrywanie i rozwiązywanie konfliktów podczas replikacji zostanie szczegółowo opisane w dalszej części artykułu. . Może zajść również sytuacja, w której jeden z docelowych węzłów odroczonej transakcji nie jest dostępny (np. na skutek awarii sieci). W takim wypadku oczywiście transakcja nie może zostać zastosowana. Pozostaje ona w lokalnej kolejce odroczonych transakcji (w węźle wysyłającym), a jej przesłanie jest ponawiane w następnych momentach uruchomienia procesu przesyłania. Jest to kolejna ważna cecha asynchronicznej replikacji multimaster – niedostępność jednego z węzłów nie wpływa na pracę pozostałych węzłów środowiska.
4. Pomyślne przesłanie i zaaplikowanie odroczonej transakcji we wszystkich zdalnych węzłach docelowych nie powoduje natychmiastowego usunięcia transakcji z kolejki odroczonych transakcji węzła, w którym cały proces się rozpoczął. Specjalny proces, uruchamiający się co zdefi-

niowany przez administratora replikacji czas, przegląda kolejkę odroczonych transakcji węzła i usuwa z niej transakcje, których propagacja zakończyła się sukcesem.

Na rysunku 3 przedstawiono mechanizm asynchronicznej replikacji multimaster dla operacji DML na replikowanej tabeli.



Rysunek 3. Mechanizm asynchronicznej replikacji multimaster.

Replikacja asynchroniczna to domyślny sposób działania replikacji multimaster. Nie powoduje zmniejszenia wydajności realizacji zarówno operacji odczytu danych replikowanych (są to oczywiście odczyty lokalne) jak i ich modyfikacji (dzięki rozdzieleniu procesu propagacji zmian od lokalnej transakcji). Nie stawia dużych wymagań co do infrastruktury sieciowej i wydajności sprzętu komputerowego. Jednak dzieje się to kosztem opóźnienia w uzyskaniu spójnego stanu replik obiektów we wszystkich węzłach środowiska.

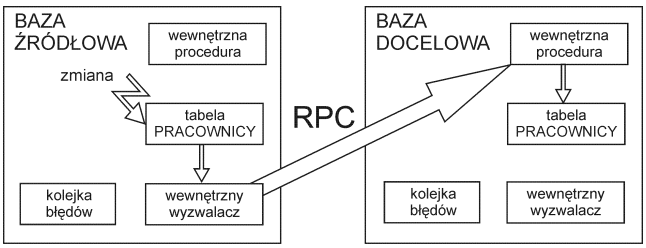
Replikacja synchroniczna

W replikacji synchronicznej w ramy transakcji, modyfikującej lokalne repliki obiektów, zostają włączone operacje przesłania i zaaplikowania tych zmian do replik zdalnych (we wszystkich pozostałych węzłach środowiska). Jeśli chociaż jeden z węzłów środowiska nie jest dostępny i przesłanie zmian do tego węzła nie jest możliwe, cała transakcja zostaje wycofana (czyli również operacje modyfikacji lokalnych replik obiektów oraz operacje przesłania zmian do innych, aktualnie dostępnych węzłów środowiska). Jest to podstawowa wada replikacji synchronicznej. Włączenie propagacji zmian do lokalnej transakcji powoduje, że replikowane schematy wszystkich węzłów środowiska są zawsze identyczne – nie ma, jak to było przy replikacji asynchronicznej, okresu czasu, w którym występowałyby jakiekolwiek różnice. Co więcej, przy replikacji synchronicznej nie występują żadne konflikty – operacja lokalna, która narusza spójność schematu węzła zdalnego, nie może zostać wykonana, a więc powoduje wycofanie całej transakcji. Tak więc nie są potrzebne żadne metody usuwania konfliktów.

Opisane teraz zostaną poszczególne kroki procesu replikacji synchronicznej.

1. Podobnie, jak przy replikacji asynchronicznej, każda operacja DML, wykonana lokalnie w węźle na replikowanej tabeli, powoduje uruchomienie wewnętrznego wyzwalacza, który generuje zdalne wywołania procedur (RPC).
2. Zdalne wywołania procedur (RPC) zostaje natychmiast przesłane do węzłów docelowych.
3. Wewnętrzne procedury w węzłach zdalnych odbierają RPC i próbują zaaplikować zmiany lokalnie. Jeśli w jakimkolwiek węźle proces ten zakończy się niepowodzeniem, wówczas cała transakcja zostaje wycofana.

Mechanizm replikacji synchronicznej został przedstawiony na rysunku 4.



Rysunek 4. Mechanizm synchronicznej replikacji multimaster.

Jak już wspomniano, podczas replikacji synchronicznej nie występują konflikty. Każda operacja modyfikacji lokalnej repliki tabeli powoduje założenie lokalnej blokady na modyfikowanych rekordach, a następnie generowane jest żądanie założenia blokady na odpowiednich rekordach w zdalnej replice tabeli (zauważmy, że propagacja modyfikacji do węzłów zdalnych odbywa się **bezpośrednio** po jej wykonaniu na węźle lokalnym, a nie po pewnym okresie czasu po zatwierdzeniu transakcji, jak to było w replikacji asynchronicznej). Oczywiście żadna inna transakcja nie może wykonać operacji modyfikacji zablokowanych rekordów, a przez to nie może dojść do konfliktów. Blokady zostają zwolnione w momencie zatwierdzenia transakcji. Wykorzystuje się tu **protokół zatwierdzania dwufazowego** (ang. two-phase commit protocol).

O ile wydajność operacji odczytu replikowanych danych pozostaje wysoka (dane są oczywiście odczytywane lokalnie), o tyle stosowanie protokołu zatwierdzania dwufazowego zmniejsza szybkości wykonywania operacji modyfikacji danych. Stosowanie replikacji synchronicznej nakłada duże wymagania na przepustowość infrastruktury sieciowej, wydajność sprzętu i oprogramowania.

Wydajność propagacji

Operacja przesyłania lokalnych transakcji do węzłów zdalnych (czyli propagacja transakcji) może być realizowana sekwencyjnie lub może zostać zrównoleglona. Przy **propagacji sekwencyjnej** (ang. serial propagation) lokalne transakcje są kolejno przesyłane do węzłów zdalnych. **Propagacja równoległa** (ang. parallel propagation) wykorzystuje zbiór procesów. Każdy z nich przesyła transakcje osobnym strumieniem. Specjalny proces, nazywany koordynatorem, kontroluje działanie procesów przesyłających. Jego zadaniem jest m.in. wykrywanie zależności między transakcjami i jeśli takie istnieją, sterowanie procesami w taki sposób, aby te transakcje dotarły w odpowiedniej kolejności do węzłów zdalnych.

Dla większości zastosowań wystarczające będzie użycie propagacji równoległej, realizowanej przez jeden proces. Zwiększanie liczby procesów ma spowodować zwiększenie wydajności działania środowiska replikacji. Nie ma jednak prostego wzoru definiującego liczbę procesów równolegle propagujących transakcje. Ustawienie właściwej wartości powinno zostać poprzedzone serią testów. Należy również pamiętać, że procesy instancji używane są także do innych operacji równoległych, takich jak np. równoległe zapytania. Dlatego też należy dokładnie zaplanować, jaką pulą procesów będzie dysponowała instancja bazy danych (należy zwrócić szczególną uwagę na parametry `PARALLEL_AUTOMATIC_TUNING`, `PARALLEL_MAX_SERVERS` i `PARALLEL_MIN_SERVERS` – po ich opis odsyłamy do dokumentacji *SZBD Oracle*).

Przy propagacji sekwencyjnej system nie korzysta z puli procesów. Dlatego w tym przypadku nie jest konieczna zmiana liczby procesów instancji. Ten tryb propagacji został zachowany jedynie dla zapewnienia zgodności z poprzednimi wersjami *SZBD Oracle*.

Osobnym zagadnieniem, mającym wpływ na wydajność działania środowiska replikacji, jest rozmiar danych przesyłanych pomiędzy węzłami. Jeśli rekord tabeli został poddany operacji modyfikacji (operacji *update*), węzeł, w którym ta operacja miała miejsce, przesyła do pozostałych węzłów nie tylko cały zmodyfikowany rekord (czyli „nowe" wartości kolumn rekordu), ale również rekord w postaci sprzed wykonania operacji modyfikacji (czyli „stare" wartości kolumn rekordu). Dostarczenie takich informacji ma na celu umożliwienie węzłom odbierającym wykrycie ewentualnego konfliktu, jaki mogła spowodować operacja. W przypadku operacji wstawienia rekordu do węzłów docelowych zostaje oczywiście przesłana tylko jego „nowa" wartość, natomiast dla usunięcia rekordu – tylko wartość „stara". Jeśli jednak wszystkie węzły środowiska replikacji posługują się *SZBD Oracle* w wersji 8.0 lub późniejszej, istnieje możliwość zmniejszenia rozmiaru danych przesyłanych pomiędzy węzłami. W takim przypadku przesyłane są jedynie następujące dane:

- wartości kolumn wchodzących w skład klucza głównego,
- „stare" wartości wszystkich kolumn z tych grup kolumn, które zawierają kolumny zmodyfikowane przez operację,
- „nowe" wartości wszystkich zmodyfikowanych kolumn.

To rozwiązanie nosi nazwę **minimalnej komunikacji** (ang. minimal communication) i powinno być stosowane zawsze wtedy, gdy jest to możliwe. Decyzję o wykorzystaniu opcji minimalnej komunikacji podejmuje się w momencie generacji obiektów wspomagających replikację.

Parametry inicjalizacyjne węzła

Instancja bazy danych, która ma brać udział w replikacji, musi zostać odpowiednio skonfigurowana. Istnieje zestaw parametrów inicjalizacyjnych, których poprawne ustawienie jest niezbędne do właściwej pracy węzła. Najważniejsze z nich to `COMPATIBLE`, `GLOBAL_NAMES`, `JOB_QUEUE_PROCESSES`, `OPEN_LINKS` i `PROCESSES`. Po dokładny opis wartości parametrów dla węzła środowiska zaawansowanej replikacji odsyłamy do [O9Rep].

Replikacja proceduralna

Replikacja proceduralna to szczególny przypadek replikacji multimaster. Odmiennie, niż w przypadku zwykłej replikacji multimaster, **nie** jest to replikacja na poziomie rekordu. Pomiędzy węzłami replikowane są **wywołania procedury składowanej**, która modyfikuje zawartość tabeli. Natomiast nie są replikowane bezpośrednie operacje modyfikacji tabeli.

Replikację proceduralną stosuje się w przypadku, gdy w systemie przeważają transakcje modyfikujące duże ilości danych. W takiej sytuacji replikacja na poziomie rekordu jest mało wydajna. Rozważmy sytuację, w której z tabeli PRACOWNICY należy usunąć wszystkie rekordy, opisujące pracowników likwidowanego wydziału X (załóżmy, że jest to pięć tysięcy rekordów). W przypadku replikacji na poziomie rekordu wymagałoby to przesłania do zdalnych węzłów pięciu tysięcy żądań modyfikacji danych. W przypadku replikacji proceduralnej przesyłane jest tylko wywołanie odpowiedniej procedury, która zrealizuje żądaną operację usunięcia danych.

Przy replikacji proceduralnej jednocześnie może działać tylko jedna replikowana procedura (obowiązuje propagacja sekwencyjna). Wywołania procedury zostają natychmiast przesłane do węzłów docelowych w przypadku replikacji synchronicznej lub są składowane w lokalnej kolejce transakcji przy replikacji asynchronicznej.

Jeśli węzeł nadrzędny ma zdefiniowanych kilka nadrzędnych grup replikacji, z których jedna lub więcej są aktualnie wyłączone, nie jest możliwe działanie replikacji proceduralnej dla **żadnej** z grup w węźle. Powodem takiego zachowania jest możliwość modyfikacji przez replikowaną procedurę, należąca do jednej z grup replikacji, danych obiektów z innych grup. Replikacja proceduralna

może działać wyłącznie wtedy, gdy wszystkie nadrzędne grupy replikacji w węźle działają normalnie.

W przypadku stosowania replikacji proceduralnej w środowisku hybrydowym (czyli będącym połączeniem replikacji multimaster z replikacją migawkową), wywołania procedur są przesyłane wyłącznie do węzłów nadrzędnych. Węzły migawkowe nie uczestniczą w propagacji. Replikacja proceduralna może zostać zainicjowana w węźle migawkowym, jednak węzeł migawkowy nigdy nie będzie odbiorcą wywołań replikowanej procedury.

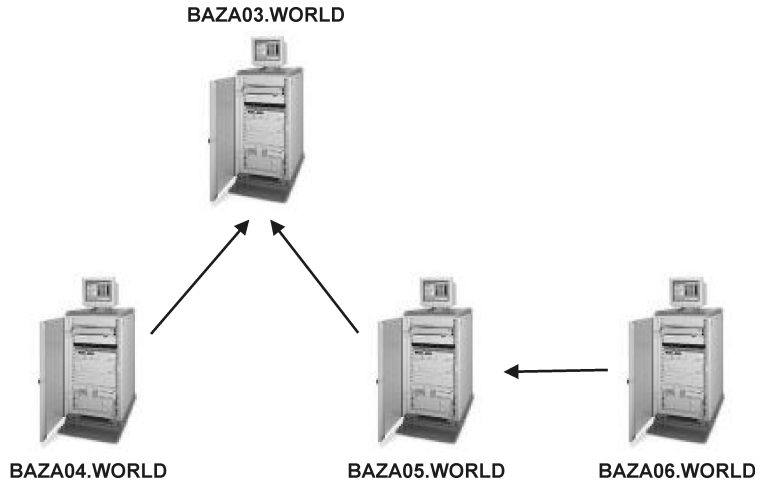
Replikacja proceduralna wymaga utworzenia pakietu, który będzie służył do modyfikacji danych. Pakiet musi zostać skonstruowany przy spełnieniu następujących założeń:

- pakiet może zawierać wyłącznie procedury, niedozwolone jest definiowanie funkcji,
- parametry procedur pakietu mogą być wyłącznie typu `IN`, nie są dopuszczalne parametry `OUT` i `INOUT`,
- w ciałach procedur pakietu, których wywołanie będzie replikowane, musi zostać wyłączona replikacja na poziomie rekordu (patrz opis procedury `REPLICATION_OFF` z pakietu `DBMS_REPUTIL` w [O9RepAPI]),
- procedury nie mogą odwoływać się do funkcji SQL, których wykonanie jest zależne od stanu lokalnego środowiska (np. funkcji `SYSDATE`).

Po szczegółowy opis replikacji proceduralnej odsyłamy do [WrBę03] i [O9Rep].

2.5. Replikacja migawkowa

Środowisko replikacji migawkowej tworzą dwa rodzaje węzłów: **węzły nadrzędne** (ang. master site) oraz **węzły migawkowe** (ang, materialized view site). Każdy węzeł migawkowy jest połączony z **dokładnie jednym** węzłem nadrzędnym, natomiast węzeł nadrzędny może być połączony z wieloma węzłami migawkowymi. Najczęściej spotykaną konfiguracją jest **architektura jednowarstwowa**, w której w środowisku występuje tylko jeden węzeł nadrzędny. Możliwa jest jednak również konfiguracja, w której węzeł migawkowy jest jednocześnie węzłem nadrzędnym dla innych węzłów migawkowych. Taka **architektura wielowarstwowa** została przedstawiona na rysunku 5. Węzłami nadrzędnymi są węzły **BAZA03.WORLD** i **BAZA05.WORLD**, natomiast węzły migawkowe to **BAZA04.WORLD**, **BAZA05.WORLD** i **BAZA06.WORLD**. Węzeł **BAZA05.WORLD** jest zarówno węzłem nadrzędnym, jak i migawkowym.



Rysunek 5. Środowisko replikacji migawkowej.

Replikacja migawkowa jest **replikacją na poziomie rekordu**. Węzły migawkowe zawierają migawki (perspektywy zmaterializowane). Migawka jest połączona z tabelą z węzła nadrzędnego i kopiuje lokalnie jej dane. W takim wypadku tabela nosi nazwę **tabeli nadrzędnej** (ang. master table) Gdy mamy do czynienia z architekturą wielowarstwową, wówczas migawka w węźle migawkowym kopiuje dane z migawki w węźle nadrzędnym – wówczas migawka w węźle nadrzędnym nosi nazwę **nadrzędnej migawki** (ang. master materialized view).

Proces propagacji zmian w replikacji migawkowej różni się znacznie od propagacji w opisanej już wcześniej replikacji multimaster. O ile w replikacji multimaster replikowane tabele węzła są w sposób ciągły uaktualniane przez operacje z innych węzłów, o tyle przy replikacji migawkowej operacje z węzła nadrzędnego są łączone w „paczki" (ang. batch updates) i uaktualniają migawkę w węźle migawkowym. Taki sposób propagacji zmian nosi nazwę **odświeżania** (ang refresh).

Najważniejszą cechą replikacji migawkowej jest możliwość ograniczenia i wstępnego przetworzenia danych, kopiowanych z węzła nadrzędnego do węzła migawkowego. Ograniczenie danych może być zarówno poziome (replikowanie rekordów z tabeli węzła nadrzędnego, spełniających określone warunki), jaki pionowe (replikowanie danych z wybranych kolumn tabel węzła nadrzędnego). Przypominamy, że w replikacji multimaster takie ograniczenia nie mogły być stosowane – replikowana była cała tabela.

Jeśli w węzłach migawkowych zastosowano migawki modyfikowalne, wówczas lokalne zmiany danych w migawkach w węźle migawkowym mogą zostać replikowane do węzła nadrzędnego, modyfikując zawartość tabeli (lub migawki jeśli węzłem nadrzędnym był węzeł migawkowy). Jak widzimy, propagacja zmian w replikacji migawkowej mimo nierównego statusu węzłów może mieć również charakter dwukierunkowy (podobnie jak w replikacji multimaster).

Replikacja migawkowa sprawdza się w zastosowaniach wymagających replikowania tylko określonego podzbioru danych. Bardzo cenną cechą replikacji migawkowej jest możliwość pracy odłączonej. Węzłem migawkowym środowiska może być komputer przenośny, natomiast węzłem nadrzędnym – centralny serwer firmy.

Architektura replikacji migawkowej

Organizacja węzła nadrzędnego

Organizacja węzła nadrzędnego w replikacji migawkowej jest identyczna jak w przypadku replikacji multimaster. Jeśli tabela (migawka) w węźle nadrzędnym ma być źródłem danych dla migawki w węźle migawkowym i odświeżanie ma być realizowane w sposób przyrostowy, wówczas należy zdefiniować dziennik perspektywy zmaterializowanej. Jeśli migawka ma być odświeżana w sposób pełny, założenie dziennika nie jest potrzebne.

Organizacja węzła migawkowego

Łączniki bazodanowe

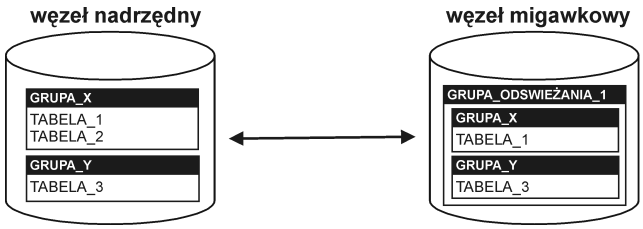
Węzeł migawkowy wymaga utworzenia trzech łączników bazodanowych do węzła nadrzędnego. Pierwszy łącznik jest łącznikiem prywatnym administratora węzła migawkowego i wskazuje na użytkownika – pełnomocnika administratora w węźle nadrzędnym (patrz rozdział „Mechanizmy administracyjne w replikacji migawkowej"). Kolejny łącznik prywatny jest własnością nadawcy odroczonych transakcji w węźle migawkowym i wskazuje na użytkownika – odbiorcę w węźle nadrzędnym. Wreszcie ostatni wymagany łącznik jest własnością użytkownika – właściciela perspektyw zmaterializowanych i wskazuje na pełnomocnika użytkownika odświeżającego migawki z węzła nadrzędnego.

Migawka (perspektywa zmaterializowana)

Migawka to jedyny obiekt replikacji w węźle migawkowym. Kopiuje dane z obiektu nadrzędnego (tabeli lub innej migawki w przypadku architektury wielowarstwowej).

Grupy migawek (perspektyw zmaterializowanych)

Grupa migawek (perspektyw zmaterializowanych) (ang. materialized view group) to zbiór logicznie powiązanych migawek z węzła migawkowego. Musi być połączona z dokładnie jedną grupą replikacji z węzła nadrzędnego, nazwy obu grup muszą być identyczne. Rysunek 6 przedstawia dwie grupy migawek w węźle migawkowym, połączonych z grupami replikacji w węźle nadrzędnym.



Rysunek 6. Grupy migawek wraz z grupami nadrzędnymi

Migawki z grupy migawek nie muszą kopiować danych ze wszystkich obiektów z grupy nadrzędnej. Na rysunku 6 widzimy, że migawka z grupy o nazwie GRUPA_X kopiuje tylko dane z tabeli TABELA_1, natomiast nie kopiuje danych z tabeli TABELA_2.

Grupy migawek są niezbędne jedynie wówczas, jeśli w węźle migawkowym zaimplementowane są migawki modyfikowalne. Jeśli migawka nie należy do żadnej grupy, wówczas musi to być migawka tylko do odczytu bądź zapisywalna.

Grupy odświeżania

Jeśli określone migawki powinny być odświeżane jednocześnie w celu zachowania spójności replikowanych przez nie danych, wówczas należy je dodać do **grupy odświeżania** (ang. refresh group). Po odświeżeniu wszystkich migawek, należących do tej samej grupy odświeżania, ich dane pochodzą z tego samego punktu w czasie. Maksymalna liczba migawek w grupie odświeżania to 400. Przykład grupy odświeżania o nazwie GRUPA_ODSWIEZANIA_1 umieszczono na rysunku 6.

Grupa odświeżania może zawierać migawki z różnych grup migawek. Migawki z tej samej grupy migawek mogą należeć do różnych grup odświeżania. Jednak decydując się na takie rozwiązanie należy zdawać sobie sprawę, że może to prowadzić do problemów z zachowaniem spójności danych w węźle migawkowym.

Grupa odświeżania definiuje okres czasu pomiędzy kolejnymi uruchomieniami procesu odświeżenia migawek.

Mechanizmy administracyjne w replikacji migawkowej

Funkcje administracyjne w węźle nadrzędnym

Jeśli węzeł ma pełnić funkcję węzła nadrzędnego w środowisku replikacji migawkowej, oprócz użytkownika – administratora węzła (pełniącego identyczne funkcje jak w replikacji multimaster), należy utworzyć dwóch dodatkowych użytkowników administracyjnych. Pierwszy z nich, **pełnomocnik administratora perspektyw zmaterializowanych** (ang. proxy materialized view

administrator) to użytkownik, wykonujący operacje w imieniu administratora migawek z węzła migawkowego. Otrzymuje jedynie podstawowy zestaw praw, niezbędnych do realizacji swoich funkcji. Drugi użytkownik, **pełnomocnik użytkownika odświeżającego perspektywy zmaterializowane** (ang. proxy refresher) wykonuje operacje w imieniu analogicznego użytkownika z węzła migawkowego.

Funkcje administracyjne w węźle migawkowym

Węzeł, będący węzłem migawkowym w środowisku replikacji migawkowej, powinien posiadać następujący zbiór użytkowników. Pierwszy z nich, **administrator perspektyw zmaterializowanych** (ang. materialized view administrator) to użytkownik odpowiedzialny za konfigurację i zarządzanie węzłem migawkowym. Jest odpowiednikiem administratora replikacji ze środowiska replikacji multimaster. Funkcją użytkownika **nadawcy** (ang. propagator) jest przesyłanie transakcji z lokalnej kolejki odroczonych transakcji węzła migawkowego do węzła nadrzędnego. **Użytkownik odświeżający migawki** (ang refresher) jest odpowiedzialny za aplikowanie zmian, jakie zaszły w tablicach lub migawkach węzła nadrzędnego, w lokalnych migawkach. Funkcję tego użytkownika może pełnić administrator węzła migawkowego. Ostatni z użytkowników, **odbiorca** (ang. receiver) odbiera odroczone transakcje, przesyłane przez nadawców z innych węzłów migawkowych. Użytkownik ten jest niezbędny tylko wtedy, gdy węzeł migawkowy będzie węzłem nadrzędnym dla innych węzłów migawkowych (architektura wielowarstwowa).

Proces odświeżania perspektyw zmaterializowanych

Proces odświeżania w replikacji migawkowej składa się z następujących kroków:

1. Jeśli migawka jest migawką modyfikowalną i wchodzi w skład grupy migawek, wówczas wszystkie transakcje, które dokonały modyfikacji danych migawki, zostają zapisane w kolejce odroczonych transakcji (podobnie jak przy asynchronicznej replikacji multimaster). Pierwszym krokiem procesu odświeżenia migawki jest przesłanie zawartości kolejki odroczonych transakcji do węzła nadrzędnego (lub węzła migawkowego w przypadku architektury wielowarstwowej).
2. Właściwe odświeżenie migawki – migawka kopiuje dane z obiektu z węzła nadrzędnego. Należy pamiętać, że jest to kopia z punktu w czasie. Rozważmy następującą sytuację. Węzeł nadrzędny zawiera tabelę PRACOWNICY, która jest obiektem nadrzędnym dla migawki PRACOWNICY_MV w węźle migawkowym. Pomiędzy kolejnymi procesami odświeżania perspektywy zawartość tabeli nadrzędnej została poddana 15 operacjom modyfikacji danych. Jednak odświeżenie migawki będzie polegało jedynie na doprowadzeniu jej do stanu odpowiadającego bieżącemu stanowi obiektu nadrzędnego.

2.6. Wykrywanie i rozwiązywanie konfliktów

W trakcie normalnej pracy środowiska replikacji często dochodzi o występowania **konfliktów**. Aby czytelnik zrozumiał intuicyjnie ideę konfliktu, rozważmy następujący przykład. W środowisku asynchronicznej replikacji multimaster z rysunku 1 obserwujemy jednoczesną pracę dwóch użytkowników: *X* (**BAZA01.WORLD**) i *Y* (**BAZA02.WORLD**). Obaj w tym samym momencie dokonują operacji modyfikacji tych samych rekordów z replikowanej tabeli PRACOWNICY – patrz tabela 1.

Tabela 1. Przykład sytuacji prowadzącej do wystąpienia konfliktu.

Użytkownik X (**BAZA01.WORLD**)	*Użytkownik Y* (**BAZA02.WORLD**)
SQL> update pracownicy 2 set pr_pensja = pr_pensja*1.1 3 where pr_et_nazwa = 'ANALITYK' 4 / 10 wierszy zostało zmodyfikowanych. SQL> COMMIT; Zatwierdzanie zostało ukończone.	SQL> delete pracownicy 2 where pr_et_nazwa = 'ANALITYK' 3 / 10 wierszy zostało usuniętych. SQL> COMMIT; Zatwierdzanie zostało ukończone.

Transakcje, realizowane przez obu użytkowników zostają zatwierdzone – z lokalnego punktu widzenia są prawidłowe. Jednak co się stanie w momencie, gdy operacja z węzła **BAZA01.WORLD** zostanie przesłana do węzła **BAZA02.WORLD**? Rekordy, które mają zostać zmodyfikowane, już nie istnieją, gdyż zostały usunięte przez lokalną operację węzła **BAZA02.WORLD**. System zgłasza wystąpienie konfliktu.

Konflikt można zdefiniować jako sytuację, w której lokalne zaaplikowanie operacji z węzła zdalnego nie jest możliwe, gdyż mogłoby bądź spowodować utratę spójności lokalnego schematu węzła, bądź nie istnieją dane w węźle, których dotyczy zdalna operacja (zostały zmodyfikowane lub usunięte przez lokalną transakcję węzła). Wystąpienie konfliktu powoduje niemożność uzgodnienia stanu replikowanych obiektów pomiędzy węzłami, co w konsekwencji może doprowadzić do utraty zgodności schematów węzłów środowiska. Taka sytuacja musi zostać natychmiast wykryta i w miarę możliwości rozwiązana. *SZBD Oracle* próbuje automatycznie rozwiązywać konflikty określonego rodzaju, jednak nie zawsze odnosi sukces. W takim przypadku powiadamia administratora środowiska replikacji o wystąpieniu sytuacji konfliktowej - administrator musi ręcznie spróbować usunąć konflikt.

Konflikty występują tylko przy **asynchronicznej propagacji zmian**. Przy **propagacji synchronicznej** wystąpienie konfliktu **nie jest możliwe**, gdyż sama operacja przesłania zmian do węzłów zdalnych jest częścią lokalnej transakcji, która te zmiany wprowadza. W tym przypadku **moduł zarządzania współbieżnością** systemu zarządzania bazą danych nie dopuszcza do wystąpienia sytuacji błędnych (czyli niepożądanego wpływu jednoczesnych operacji z innych transakcji). Odbywa się to m.in. przez zakładanie blokad na modyfikowanych rekordach. Dlatego w dalszej części tego rozdziału będziemy zakładać, że wszystkie przykłady odnoszą się do środowiska z asynchroniczną propagacją zmian.

Przy stosowaniu replikacji proceduralnej procedura, która replikuje dane pomiędzy węzłami, jest odpowiedzialna za utrzymywanie spójności danych. Powinna zostać tak zaprojektowana, aby unikała konfliktów lub je wykrywała i usuwała.

Zanim przejdziemy do omawiania **mechanizmów wykrywania i rozwiązywania konfliktów**, przyjrzyjmy się **rodzajom konfliktów**, jakie możemy zaobserwować w replikacji.

Rodzaje konfliktów

W środowisku zaawansowanej replikacji mogą występować następujące rodzaje konfliktów:

- **konflikt aktualizacji** – występuje wtedy, gdy transakcje, pochodzące z różnych węzłów środowiska, dokonują w tym samym czasie operacji aktualizacji tego samego rekordu,
- **konflikt unikalności** – występuje wtedy, gdy replikacja rekordu powoduje naruszenie ograniczeń typu klucz podstawowy lub klucz unikalny w węzłach docelowych,

- **konflikt usunięcia** – występuje w przypadku, gdy jedna transakcja usuwa rekordy z replikowanej tabeli, które są w tym samym momencie modyfikowane lub usuwane przez transakcję z innego węzła. Sytuacja, przedstawiona w tabeli 1 jest konfliktem usunięcia.

Możemy wyróżnić jeszcze jeden rodzaj konfliktu, nazywany **konfliktem kolejnościowym** (ang. ordering conflict), powstającym w wyniku różnej kolejności propagacji transakcji pomiędzy węzłami. Występuje on tylko w środowisku, które tworzy trzy lub więcej węzłów. Odnieśmy się znowu do przykładu (środowisko asynchronicznej replikacji multimaster z węzłami **BAZA01.WORLD**, **BAZA02.WORLD** i **BAZA03.WORLD**). Załóżmy, że w wyniku zajścia pewnych okoliczności węzeł **BAZA01.WORLD** nie jest dostępny, czyli uaktualnienia (tworzące transakcje), które w międzyczasie zostały wykonane w replikowanych obiektach w węzłach **BAZA02.WORLD** i **BAZA03.WORLD** nie mogą zostać zaaplikowane w tym węźle. Jak wiemy, odroczone transakcje w przypadku niedostępności węzła docelowego są przechowywane w węzłach wysyłających. Kiedy węzeł **BAZA01.WORLD** stanie się znowu dostępny, odroczone transakcje zostaną przesłane do tego węzła, jednak ich kolejność może być różna od kolejności, w jakiej zostały zaaplikowane na pozostałych węzłach środowiska. Przez to uaktualnienia, które tworzą te transakcje mogą spowodować konflikty.

Wykrywanie konfliktów

Każdy węzeł posiada wbudowane mechanizmy, umożliwiające automatyczne wykrycie sytuacji konfliktowej w momencie aplikowania odroczonej transakcji, przesłanej z węzła zdalnego.

SZBD Oracle, przesyłając pomiędzy węzłami informacje o operacji modyfikacji rekordu, przesyła nie tylko „nowe" wartości kolumn z replikowanego rekordu (czyli postać rekordu po wykonaniu modyfikacji), ale także „stare" wartości kolumn (czyli sprzed wykonania modyfikacji) (patrz rozdział „Wydajność propagacji"). Jeśli węzeł docelowy wykryje, że istnieje różnica pomiędzy starą wartością replikowanego rekordu, przesłaną wraz z operacją z węzła zdalnego, a wartością aktualną odpowiadającego mu lokalnego rekordu, zgłasza wystąpienie konfliktu aktualizacji.

Konflikt unikalności zostaje zgłoszony w węźle docelowym w sytuacji, gdy próba lokalnego wykonania operacji *update* lub *insert* przesłanych z węzła zdalnego powoduje naruszenie ograniczeń integralnościowych typu klucz podstawowy lub klucz unikalny. Z kolei konflikt usunięcia zostaje wykryty w węźle docelowym wówczas, gdy dla operacji *update* lub *delete*, przesłanych z węzła zdalnego, *SZBD Oracle* nie może znaleźć lokalnie rekordów z odpowiadającymi wartościami klucza podstawowego.

Bardzo ważnym zagadnieniem dla możliwości wykrywania konfliktów jest sposób identyfikacji przez *SZBD Oracle* rekordów z replikowanych tabel. *SZBD Oracle* musi mieć możliwość stwierdzenia, że dwa rekordy z replik tej samej tabeli w dwóch różnych węzłach środowiska to faktycznie ten sam rekord (lub właściwie repliki tego samego rekordu). Dlatego każda replikowana tabela powinna mieć zdefiniowany klucz podstawowy. W przypadku braku takiego klucza podczas dodawania tabeli do zbioru obiektów replikacji należy wskazać, które kolumny *SZBD Oracle* ma używać unikalnego identyfikowania rekordów podczas replikacji – będą one tworzyć klucz alternatywny.

Wykrycie konfliktu powoduje przystąpienie do realizacji **procedury rozwiązywania konfliktów**.

Rozwiązywanie konfliktów

Wykrycie konfliktu powoduje uruchomienie odpowiednich procedur, mających na celu rozwiązanie konfliktu i zapewnienie zgodności replikowanych schematów węzłów środowiska. *SZBD Oracle* dostarcza zestaw predefiniowanych metod, próbujących rozwiązać konflikty aktualizacji i unikalności. Nie ma żadnych wbudowanych metod rozwiązywania konfliktów usunięcia i kolejnościowych. Administrator może zaimplementować swoje własne metody rozwiązywania konfliktów

i używać ich zamiast metod predefiniowanych (w przypadku konfliktów aktualizacji i usunięcia) lub w miejsce ich braku (dla konfliktów usunięcia i kolejnościowych).

Nie wszystkie wbudowane metody rozwiązywania konfliktów doprowadzają replikowane schematy do zbieżności. Przykładem jest metoda nadpisania – bieżąca wartość w węźle docelowym zostaje zastąpiona wartością z węzła zdalnego (metoda zostanie szerzej opisana w następnym rozdziale). W środowisku, składającym się z więcej niż dwóch węzłów nadrzędnych doprowadza to oczywiście do utraty zgodności schematów węzłów. W przypadku metod nie gwarantujących zbieżności węzłów musi zostać zaimplementowany dodatkowy mechanizm, mający za zadanie poinformowanie administratora środowiska o wystąpieniu konfliktu i „połowicznym" jego rozwiązaniu – na administratora spada teraz zadanie doprowadzenia schematów do stanu zgodnego.

Każda metoda rozwiązywania konfliktów musi zostać związana z **grupą kolumn** (ang. column group). Jest to nazwany zbiór logicznie powiązanych ze sobą kolumn replikowanej tabeli. Kolumna może należeć tylko do jednej grupy, grupa może zawierać jedną lub więcej kolumn. Jeśli podczas aplikowania operacji ze zdalnej transakcji, modyfikującej kolumnę tabeli, zostanie wykryty konflikt, do jego rozwiązania zostanie użyta metoda przypisana do grupy kolumn, do której należy modyfikowana kolumna. Metoda może się posługiwać pozostałymi kolumnami z grupy, dlatego też dobór odpowiednich kolumn do grupy jest sprawą bardzo ważną. Do tej samej grupy należy przypisać te kolumny, których wartości są ze sobą związane. Np. kolumna określająca kod pocztowy miasta powinna zostać przydzielona do tej samej grupy, co kolumna zawierająca nazwę miasta.

W przypadku, gdy administrator nie dołączy kolumny replikowanej tabeli do żadnej grupy kolumn, jest ona dołączana do domyślnej grupy o nazwie **shadow**. Jednak do takiej grupy nie można przypisać żadnej metody rozwiązywania konfliktów.

Z jedną grupą kolumn może zostać związanych kilka metod rozwiązywania konfliktów. Jeśli jedna metoda zawiedzie i nie będzie w stanie rozwiązać konfliktu, zostanie użyta następna metoda. Kolejność użycia metod zostaje zdefiniowana w momencie przydzielania metody do grupy.

Informacja o konflikcie, którego rozwiązanie się nie powiodło (zawiodły wszystkie metody) lub też nie zdefiniowano żadnych metod rozwiązywania konfliktów, trafia do **kolejki błędów** węzła, gdzie konflikt wystąpił. Kolejka błędów jest implementowana przez perspektywę *DEFERROR*. Znajdują się tam informacje o transakcjach, których zaimplementowanie w węźle nie było możliwe (czyli spowodowały konflikt). W perspektywie *DEFERROR* nie ma informacji o konfliktach, których rozwiązanie zakończyło się powodzeniem w wyniku zastosowania zdefiniowanych metod. Stąd tak ważną sprawą jest zaimplementowanie mechanizmu powiadamiania administratora o konfliktach rozwiązanych przez metody, które nie gwarantują zbieżności.

Przegląd predefiniowanych metod rozwiązywania konfliktów

Zamieścimy teraz opis kilku przykładowych predefiniowanych metod rozwiązywania konfliktów.

Konflikty aktualizacji

Metoda najpóźniejszego znacznika (ang. latest timestamp) bazuje na znaczniku czasowym, który dla każdej operacji aktualizacji danych określa moment jej wystąpienia. W przypadku konfliktu dwóch operacji *update* wykonana zostaje ta, której znacznik czasowy jest późniejszy. Dobrym rozwiązaniem jest połączenie metody najpóźniejszego znacznika z inną metodą, która przejmie rozwiązanie konfliktu w przypadku, gdy konfliktowe operacje mają identyczne znaczniki czasowe. Metoda wymaga zaimplementowania w każdym z węzłów mechanizmu oznaczania czasu wykonania każdej operacji *update* danych replikowanej tabeli (np. odpowiedniego wyzwalacza).

Istotą **metody nadpisania** (ang. overwrite) jest zastąpienie, w przypadku wystąpienia konfliktu, bieżącej wartości kolumny w węźle docelowym nową wartością kolumny z węzła wysyłające-

go. Metoda gwarantuje zbieżność w środowiskach z jednym węzłem nadrzędnym i wieloma węzłami migawkowymi. Można ją również zastosować w środowisku z wieloma węzłami nadrzędnymi, jednak zbieżność nie jest wtedy zagwarantowana.

Metoda addytywna (ang. additive) służy do rozwiązywania konfliktów w grupach kolumn, zawierających pojedyncze kolumny numeryczne. W przypadku wystąpienia konfliktu nową wartością kolumny jest suma wartości bieżącej i różnicy pomiędzy wartością nową a wartością poprzednią. Metoda zapewnia zbieżność dla środowisk z dowolną liczbą węzłów nadrzędnych i migawkowych. Przeznaczona jest szczególnie dla systemów finansowych, z dużą liczbą operacji składania i wycofywania depozytów.

Po opis pozostałych metod rozwiązywania konfliktów aktualizacji odsyłamy Czytelnika do [WrBę03], [O9Rep] oraz [O9RepAPI].

Konflikty unikalności

Predefiniowane metody rozwiązywania konfliktów unikalności nie gwarantują zbieżności schematów węzłów. Ich działanie sprowadza się tylko do usunięcia sytuacji naruszającej klucz podstawowy lub unikalny w replikowanej tabeli, powinien jednak zostać zaimplementowany dodatkowy mechanizm, który spowoduje zawiadomienie administratora o wystąpieniu konfliktu. Administrator musi ręcznie doprowadzić replikowane schematy do zgodności

Działanie metody **dodania nazwy węzła** (ang. append site name) jest następujące. Jeśli operacja, wchodząca w skład zdalnej transakcji, modyfikując wartość określonej kolumny replikowanej tabeli w węźle docelowym, spowoduje naruszenie jej unikalności (naruszenie klucza podstawowego lub unikalnego), wówczas *SZBD Oracle* do nowej wartości modyfikowanej kolumny dodaje część globalnej nazwy węzła (do pierwszej kropki), z którego pochodzi transakcja. Czyli np. w przypadku konfliktu, spowodowanego przez operację z transakcji z węzła **TEST.WORLD**, zmieniającą wartość kolumny *x* na ‘ABC’, metoda spowoduje zapisanie w kolumnie *x* wartości ‘ABC-TEST’. Oczywiście metoda dodania nazwy węzła może zostać zastosowana wyłącznie dla kolumn tekstowych.

Metoda **dodania sekwencji** (ang. append sequence) działa podobnie jak metoda poprzednia, jednak zamiast nazwy węzła dodaje do wartości kolumny, powodującej konflikt, automatycznie wygenerowany numer sekwencyjny.

Przy metodzie **zignorowania** (ang. discard) jeśli rekord, przesłany z węzła zdalnego, narusza w węźle docelowym klucz podstawowy lub unikalny, wówczas *SZBD Oracle* po prostu ignoruje rekord i nie aplikuje go w węźle docelowym.

2.7. Zarządzanie środowiskiem zaawansowanej replikacji

SZBD Oracle dostarcza kilku mechanizmów, których wykorzystanie umożliwia efektywne zarządzanie środowiskiem zaawansowanej replikacji. Są to: **Replication Management Tool**, **Replication Management API** oraz **katalog replikacji**.

Replication Management API

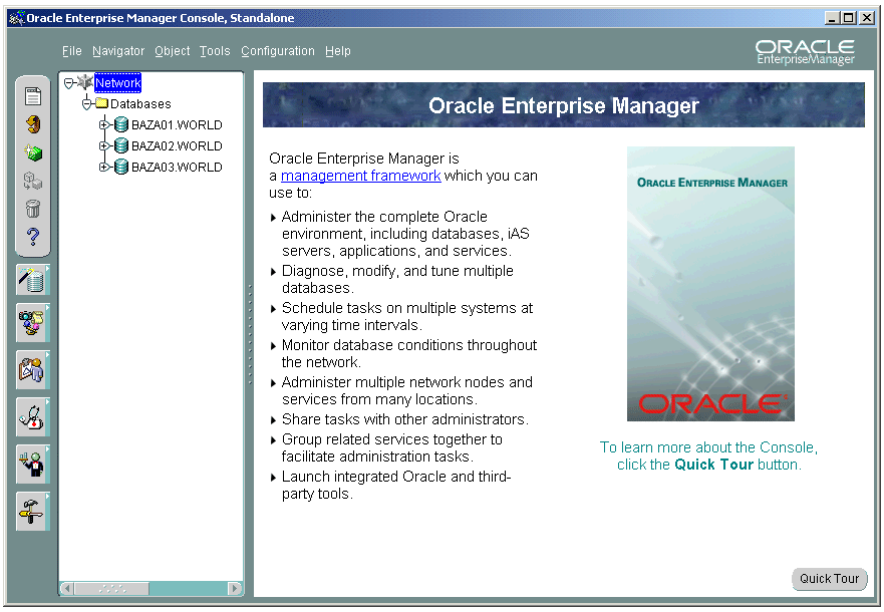
Replication Management API to zestaw pakietów PL/SQL, zawierających procedury i funkcje, umożliwiające konfigurowanie środowiska replikacji. Najważniejsze pakiety to: `DBMS_DEFER`, `DBMS_DEFER_SYS`, `DBMS_MVIEW`, `DBMS_REPCAT`, `DBMS_REPCAT_ADMIN` oraz `DBMS_REPUTIL`. Po szczegółowy opis pakietów odsyłamy do [O9RepAPI].

Katalog replikacji

Katalog replikacji to zestaw tablic i perspektyw słownika danych, udostępniających informacji o konfiguracji i aktualnym stanie środowiska replikacji. Podczas omawiania poszczególnych elementów środowisk zaawansowanej replikacji wskazywaliśmy na najważniejsze obiekty katalogu. Po pełen opis odsyłamy do [O9RepAPI].

Replication Management Tool

Replication Management Tool stanowi część pakietu narzędzi administracyjnych **Oracle Enterprise Mananger**, Jest to środowisko graficzne, automatyzujące proces tworzenia środowiska replikacji i ułatwiające późniejsze nim zarządzanie. Narzędzie w swojej pracy wykorzystuje procedury z **Replication Management API**. Ekran startowy narzędzia przedstawiono na rysunku 7.

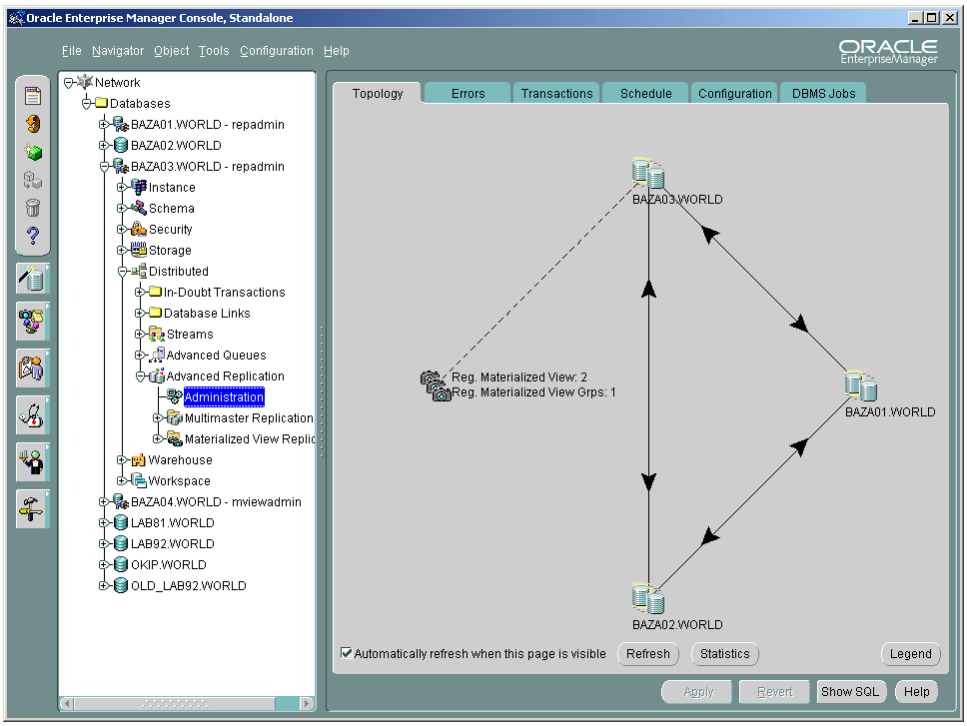


Rysunek 7. Ekran startowy pakietu Oracle Enterprise Manager.

Z narzędzia mogą korzystać użytkownicy SYSTEM, SYS oraz inni, posiadający przywilej `SELECT ANY DICTIONARY`. Pozwala na wykonywanie następujących funkcji:

- konfigurowanie instancji bazy danych pod kątem uczestnictwa w środowisku replikacji jako węzeł nadrzędny lub migawkowy,
- wizualizacja środowiska replikacji (patrz rysunek 8),
- monitorowanie pracy węzła środowiska replikacji,
- tworzenie i zarządzanie grupami replikacji, grupami migawek, dziennikami migawek, samymi migawkami, grupami odświeżania,
- konfigurowanie metod rozwiązywania konfliktów,
- monitorowanie kolejek odroczonych transakcji, żądań, błędów.

Proces konfiguracji środowiska replikacji oraz tworzenie obiektów znacznie ułatwia zestaw kreatorów, realizujących najważniejsze funkcje. Pozwala to na znaczne skrócenie czasu wymaganego do realizacji nawet skomplikowanych czynności.



Rysunek 8. Wizualizacja środowiska replikacji.

Bibliografia

[Bę02] Bębel B.: Zaawansowana replikacja w Oracle9i. Tutorial - VIII Konferencja Użytkowników i Deweloperów Oracle – PLOUG'02, Zakopane, październik 2002

[Dye99] Dye C.: Oracle Distributed Systems. O'Reilly & Associates, 1999, ISBN 1-56592-432-0

[LoKo02] Loney K., Koch G.: Oracle9i: The Complete Reference. McGraw-Hill/Osborne, 2002, ISBN 0-07-222521-1

[LoTh02] Loney K., Theriault M.: Oracle9i DBA Handbook. McGraw-Hill/Osborne, 2002, ISBN 0-07-219374-3

[O9Rep] Dokumentacja techniczna. Oracle9i Advanced Replication, rel. 9.2

[O9RepAPI] Dokumentacja techniczna. Oracle9i Replication Management API Reference, rel. 9.2

[O9SQL] Dokumentacja techniczna. Oracle9i SQL Reference, rel. 9.2

[WrBę03] Wrembel R., Bębel B.: Projektowanie rozproszonych baz danych opartych o Oracle. Wydawnictwo HELION, 2003 (w przygotowaniu)