STUDIA INFORMATICA 2010
Volume 31 Number 2B (90)

Krzysztof CZAJKOWSKI
Politechnika Krakowska, Katedra Teleinformatyki, Wydział Fizyki, Matematyki i Informatyki Stosowanej
Jakub PAŹDZIORKO
Politechnika Krakowska, Wydział Inżynierii Elektrycznej i Komputerowej

BAZY DANYCH W PAMIĘCI OPERACYJNEJ W ŚRODOWISKU ORACLE

Streszczenie. Artykuł dotyczy baz danych funkcjonujących w pamięci operacyjnej (In-Memory Database). Omówione zostały zagadnienia związane z działaniem takich rozwiązań w środowisku Oracle, które udostępnia zarówno niezależne bazy tego typu (Oracle TimesTen), jak również rozszerzenie klasycznego serwera baz danych (Oracle IMDB Cache). W pracy przedstawiono implementację przykładowego systemu wykorzystującego omawiane technologie.

Słowa kluczowe: bazy danych w pamięci operacyjnej, pamięć podręczna w bazie danych, replikacja danych, Oracle IMDB

IN-MEMORY DATABASES IN ORACLE ENVIRONMENT

Summary. This paper concerns in-memory databases. Issues connected with such types of solutions in Oracle environment were discussed, in which independent databases of such a type (Oracle TimesTen) as well as extension for standard database server (Oracle IMDB Cache) are available. In this work implementation of exemplary system using discussed technologies was presented.

Keywords: in-memory database, database cache, data replication, Oracle IMDB

1. Wstęp

W sytuacjach, w których tradycyjne systemy baz danych napotykają ograniczenia spowodowane zbyt dużą liczbą połączeń oraz idącym za tym wzrostem wykonywanych transakcji, zastosowanie znajdują bazy funkcjonujące w pamięci operacyjnej. Wychodząc naprzeciw

zapotrzebowaniom użytkowników pojawiają się w ofercie producentów systemów bazodanowych rozwiązania, które przeciwnie do tradycyjnych serwerów, nie wykorzystują dysków twardych w typowy sposób (składując na nich dane i odwołując się do nich w przypadku zapytań i modyfikacji). Całość bazy danych podczas jej pracy znajduje się w pamięci operacyjnej. Do takich systemów zalicza się Oracle TimesTen, który świetnie radzi sobie z problemami powstającymi w sytuacjach, gdy konieczne jest jednoczesne obsługiwanie bardzo wielu operacji. Znajduje on zastosowanie w specjalistycznych branżach, takich jak np. telekomunikacja, systemy pre-paidowe, bilingowe, giełdowe, transportowe, logistyczne, wojskowe itp. Dodatkowo może też zostać wykorzystany w szeroko rozumianej branży usługowej, zwłaszcza jeśli weźmiemy pod uwagę możliwość łączenia bazy TimesTen ze standardową bazą Oracle. Firma Oracle udostępniła również opcję TimesTen In-Memory Database Cache, rozszerzającą możliwości standardowego serwera o funkcjonalności zaczerpnięte z systemu TimesTen. Dzięki takiemu połączeniu systemy budowane w oparciu o te technologie posiadają zalety zarówno szybkiej bazy danych, zdolnej do obsłużenia ogromnej liczby operacji, jak i zalety bazy pozwalającej na przechowywanie wielkich ilości danych. Warto też wspomnieć, że takie rozwiązanie jest w pełni skalowalne, a także bezpieczne, gdyż całość może zostać wzbogacona o proces replikacji poszczególnych baz tworzących system. W pracy zaprezentowano przykładowe zastosowanie bazy Oracle TimesTen w systemie branży usługowej.

2. Oracle TimesTen In-Memory Database

Oracle TimesTen In-Memory Database to relacyjna baza danych, której konstrukcja została zoptymalizowana pod kątem funkcjonowania całej bazy w pamięci operacyjnej. Jej przeznaczeniem jest działanie w warstwie aplikacji i obsługa bardzo dużej liczby operacji, stąd podstawowymi kryteriami przy projektowaniu tej bazy były możliwie jak najkrótszy czas odpowiedzi przy jednoczesnej możliwie jak największej przepustowości [1]. Poza firmą Oracle również wielu innych producentów oferuje rozwiązania typu IMDB, m.in.: Polyhedra (*Polyhedra*), McObject (*eXtremeDB*), Entitydatabase (*ERDB*), rozwijane są również systemy typu OpenSource, jak FastDB czy MonetDB [12, 13]. Technologia *In-Memory DataBase* (IMDB) zakłada implementację relacyjnej bazy danych, w której wszystkie dane od momentu uruchomienia są przechowywane w pamięci RAM, a struktury danych oraz algorytmy dostępu wykorzystują ten fakt w celu osiągania jak największej wydajności. W porównaniu do tradycyjnych RDBMS wykorzystujących pamięć podręczną, IMDB wymagają dużo mniejszych mocy obliczeniowych, ponieważ eliminowany jest narzut związany z zarządzaniem buforami pamięci dla wielu lokalizacji danych (na dysku oraz w pamięci). W systemach

IMDB dyski są wykorzystywane nie jako podstawowe obszary składowania danych, ale tylko w celu zachowania ich trwałości i przywracania w przypadku wystąpienia awarii [6]. Trwałość danych jest uzyskiwana poprzez zapisy zmian (obejmujących zatwierdzone transakcje) na dysk oraz okresowe aktualizowanie obrazu bazy danych na dysku, określane jako checkpoint [4]. Odstępy czasu, w jakich realizowane są zapisy dzienników, są określane przez aplikację i mogą być zsynchronizowane z zakończeniem transakcji lub odroczone w czasie. W wielu sytuacjach ważniejsza od synchronicznego zapisu jest przepustowość, szczególnie w przypadkach, gdy czas trwania transakcji jest niewielki (np. w telefonii komórkowej).

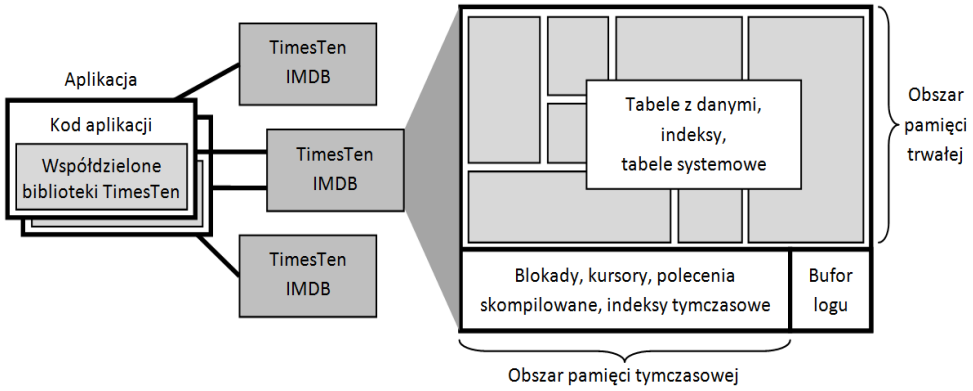
TimesTen umożliwia wykorzystanie języków SQL i PL/SQL [10] za pomocą ODBC, JDBC [11], Oracle Call Interface (OCI), TTClass i Pro*C. TTClass (TimesTen C++ Interface Classes) to biblioteka klas C++ dostarczająca opakowania dla funkcjonalności ODBC [5].

W skład bazy TimesTen wchodzą następujące elementy [1]:

- Biblioteki współdzielone (*Shared Libraries*) – zestaw bibliotek odpowiedzialnych za funkcjonalności bazy TimesTen. Są one dołączane do aplikacji i funkcjonują jako część jej procesów (w przeciwieństwie do tradycyjnych RDMBS, które funkcjonują jako zbiór programów wykonywalnych, do których łączą się aplikacje – taki sposób połączenia jest również możliwy w TimesTen, jest jednak mniej wydajny). TimesTen wykorzystuje opatentowany algorytm MicroLogging, który zapobiega problemom związanym z nieprawidłowym zakończeniem działania bazy (nawet w takiej sytuacji baza pozostaje spójna). Stanowi to przewagę nad niektórymi, podobnymi rozwiązaniami, w których stosuje się architekturę klient-serwer w celu uniknięcia tego problemu (np. Polyhedra). W tej sytuacji TimesTen stosuje bardzo bezpieczne i jednocześnie dużo wydajniejsze rozwiązanie.
- Struktury danych przechowywane w pamięci (*Memory-Resident Data Structures*) – w ich skład wchodzą wszystkie dane użytkowników, indeksy, katalogi systemu, bufory dziennika, obszar tymczasowy (rysunek 1). Wiele aplikacji może współdzielić jedną bazę TimesTen, a jedna aplikacja może wykorzystywać wiele baz.
- Procesy systemowe (*System Processes*) – odpowiadają za obsługę uruchamiania i zamykania bazy, wykrywanie błędów aplikacji na poziomie systemu oraz zapewniają obsługę ładowania, punktów zachowania oraz zakleszczeń na poziomie bazy danych.
- Programy administracyjne (*Administrative Programs*) – programy użytkowe wywoływane bezpośrednio przez użytkownika, wykorzystywane między innymi do masowego kopiowania (*Bulk Copy*), tworzenia kopii zapasowych i ich przywracania, migracji bazy danych oraz monitorowania systemu.
- Pliki kontrolne oraz pliki dzienników na dysku (*Checkpoint Files, Log Files*).

W przypadku wymagania wysokiej dostępności lub rozłożenia obciążenia wykorzystane może być rozwiązanie replikacji danych pomiędzy dwoma lub więcej serwerami. Funkcjo-

nalność ta z pewnością odróżnia produkt firmy Oracle od podobnych rozwiązań innych producentów. Serwer główny (*Master*) jest konfigurowany do wysyłania zmian danych, a pozostałe serwery (*Subscribers*) konfigurowane są do ich odbierania. Możliwa jest również replikacja dwustronna. W celu zapewnienia maksymalnej wydajności, agent replikacji wykrywa zmiany w bazie poprzez monitorowanie dzienników transakcji i wysyła zbiorczo zmiany do pozostałych serwerów. Replikowane są tylko zatwierdzone transakcje. W węźle odbiorcy agent replikacji uaktualnia stan bazy danych za pośrednictwem niskopoziomowego interfejsu, w celu pominięcia narzutu związanego z warstwą SQL.



Rys. 1. Oracle TimesTen IMDB
Fig. 1. Oracle TimesTen IMDB

3. Oracle In-Memory Database Cache

Oracle In-Memory Database Cache jest rozszerzeniem standardowej bazy Oracle, działającym w oparciu o bazę Oracle TimesTen In-Memory Database [2]. Głównym jej założeniem jest umożliwienie dostępu do danych znajdujących się w istniejącej bazie danych Oracle przy jednoczesnym wykorzystaniu zalet, jakie oferuje baza TimesTen. Poprzez udostępnienie specjalnych mechanizmów, które pozwalają na importowanie danych do bazy TimesTen oraz propagację zmian na niej wykonanych z powrotem do bazy Oracle, zyskujemy przede wszystkim możliwość znacznie wydajniejszego wykonywania operacji. Jednocześnie całe rozwiązanie jest na tyle uniwersalne, że po przeprowadzeniu nieznacznych modyfikacji bazy Oracle, zyskujemy możliwość skalowania całej architektury systemu działającego w oparciu o tą bazę (Oracle). Jest to z pewnością bardzo praktyczne i przydatne rozszerzenie, które stanowi ważną zaletę tej technologii. Podobne rozwiązania innych producentów nie oferują takich mechanizmów, przez co ich wykorzystanie ogranicza się do budowania samodzielnych baz. Oczywiście istnieje możliwość stworzenia podobnego rozwiązanie samemu, jednak przygotowanie takiego mechanizmu bezpośrednio przez producenta zapewnia dużo lepszą integrację obu technologii, a co za tym idzie, znacznie lepszą wydajność.

3.1. Architektura rozwiązania

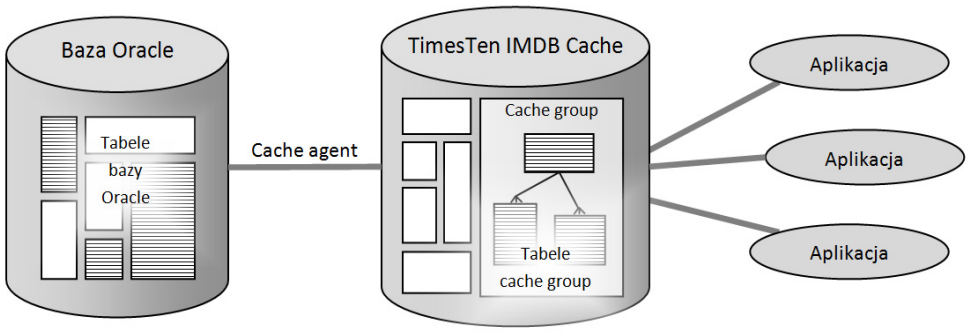
Architektura rozwiązania działającego w oparciu o Oracle IMDB Cache musi składać się z co najmniej dwóch podstawowych elementów: jednej bazy centralnej i co najmniej jednego węzła TimesTen (samodzielna baza lub aktywna baza w przypadku Standby Pair). Połączenie tych dwóch elementów (bazy centralnej i bazy węzła) pozwala na utworzenie tzw. Cache Grid. Funkcję bazy centralnej pełni (standardowa) baza Oracle. Dzięki jej zastosowaniu nadal możliwe jest składowanie znacznych ilości danych, do czego nie jest zbyt dobrze przystosowana baza TimesTen. Dodatkowo baza ta nie jest praktycznie w ogóle modyfikowana, przez co w łatwy sposób można skalować istniejące systemy. Pozwala to na swobodną rozbudowę i ograniczenie rosnącej liczby połączeń do bazy centralnej.

3.2. Wymiana danych pomiędzy bazami

Każda baza wchodząca w skład Cache Grid może działać w odmienny sposób. Oznacza to, że dla każdej z nich osobno można przygotować zupełnie niezależną konfigurację, w obrębie której definiuje się następujące elementy:

- zbiór tabel, z których należy importować dane (oraz do których należy przesyłać ewentualne zmiany),
- warunki, które muszą spełnić dane, aby mogły one zostać zaimportowane,
- parametry automatycznego odświeżania importowanych danych (częstotliwość itp.),
- sposób przesyłania zmian wykonanych na danych znajdujących się w bazie (np. synchroniczny lub asynchroniczny zapis),
- parametry automatycznego zwalniania miejsca w bazie TimesTen (usuwanie niepotrzebnych dłużej danych).

W celu połączenia powyższych ustawień tworzy się w bazie TimesTen tzw. grupy (*Cache Groups)* [3]. W obrębie jednej bazy danych może funkcjonować wiele różnych grup. Każda z nich składa się z jednej głównej tabeli (*Root Table*) oraz dowolnej liczby tabel potomnych (*Child Tables*). Tabele te stanowią odwzorowanie identycznych obiektów znajdujących się i działających po stronie bazy Oracle, a operacje przenoszenia danych pomiędzy bazami wykonywane są przez mechanizm nazywany agentem (*Cache Agent*) (rysunek 2). Należy wspomnieć o pewnych ograniczeniach, jakie wprowadza każda grupa. Pierwsze z nich nie pozwala na użycie tej samej tabeli w obrębie wielu grup. Każda tabela musi znajdować się w co najwyżej jednej z nich, a więc należy przy tworzeniu struktury bazy TimesTen zaplanować je w taki sposób, aby powiązane ze sobą tabele znalazły się w tych samych grupach. Drugie z ograniczeń wymusza konieczność połączenia tabel znajdujących się w obrębie danej grupy za pomocą relacji (relacja ta nie musi istnieć w obrębie bazy).



Rys. 2. Działanie mechanizmu IMDB Cache
Fig. 2. IMDB Cache mechanism working

Po przyjęciu powyższych założeń, dane znajdujące się w ramach dowolnej grupy możemy podzielić na instancje (*Cache Instance*). Każda instancja składa się z dokładnie jednego wiersza w tabeli głównej oraz jego rekordów potomnych.

3.3. Konfiguracje grup

Z uwagi na kierunek oraz sposób przesyłania danych pomiędzy bazą centralną a węzłem TimesTen wyróżnić można kilka rodzajów konfiguracji [3]. Poniżej zaprezentowane zostały wszystkie możliwe typy wraz z krótką charakterystyką.

3.3.1. Grupa wyłącznie do odczytu (Read-Only Cache Group)

W przypadku tej grupy nie zakłada się możliwości wykonywania zmian na bazie TimesTen. Wyłącznym jej przeznaczeniem jest prezentowanie danych (tabeli lub jej fragmentu) tylko do odczytu na bazie TimesTen. Dane mogą być odświeżane automatycznie lub ręcznie.

3.3.2. Grupa z asynchronicznym zapisem (Asynchronous Writethrough Cache Group)

W grupie tej poza możliwością prezentacji danych z bazy Oracle dopuszcza się możliwość ich modyfikacji w bazie TimesTen. Modyfikacje (w tym także wstawianie i kasowanie rekordów) przesyłane są do bazy centralnej i tam są także wprowadzane. W przypadku grupy z asynchronicznym zapisem przesyłanie tych zmian odbywa się w tle. Transakcje na bazie Oracle TimesTen zatwierdzane są bez konieczności oczekiwania na ich zatwierdzenie po stronie bazy Oracle. Rozwiązanie takie nie gwarantuje więc spójności danych pomiędzy obiema bazami (może się zdarzyć, że transakcja na bazie Oracle nie zostanie zatwierdzona, lub że dane zostaną nadpisane przez innego użytkownika).

3.3.3. Grupa z synchronicznym zapisem (Synchronous Writethrough Cache Group)

W tym przypadku przesłanie zmian do bazy Oracle wykonywane jest na bieżąco dla każdej transakcji tuż przed jej zatwierdzeniem. Tylko pomyślne zatwierdzenie zmian na bazie centralnej umożliwia zatwierdzenie transakcji na bazie TimesTen. W tym rozwiązaniu, kosz-

tem zwiększenia czasu potrzebnego na wykonanie operacji modyfikującej dane, zyskuje się gwarancję spójności danych w obu bazach.

3.3.4. Grupa użytkownika (User Managed Cache Group)

Jeśli żadna z powyższych konfiguracji nie spełnia wymagań użytkownika, istnieje możliwość utworzenia własnej grupy. W jej obrębie sami możemy zdecydować o takich parametrach, jak możliwość modyfikowania poszczególnych tabel czy odświeżania danych znajdujących się w tych tabelach.

3.4. Metody wypełniania grup danymi

Każdy rodzaj grupy może pobierać dane z bazy centralnej. Z uwagi na sposób, w jaki te dane będą wczytywane, wyróżniamy dwa podstawowe typy. Wyboru należy dokonać w momencie tworzenia grupy, dodając odpowiedni wpis w jej definicji.

Pierwszy rodzaj nazywany jest grupą zasilaną bezpośrednio (*Explicitly Loaded Cache Group*). W przypadku tej metody dane wczytywane są wyłącznie za pomocą dedykowanego polecenia. Po jego wykonaniu, wszystkie dane, do których aplikacja będzie mieć dostęp, znajdują się w bazie TimesTen. Jest to domyślny typ dla tworzonej grupy.

Drugi typ stanowi grupa dynamiczna (*Dynamic Cache Group*). W jej przypadku nie występuje konieczność umieszczania wszystkich danych w bazie przed rozpoczęciem pracy. W przypadku gdy aplikacja próbuje uzyskać dostęp do rekordu, który nie został jeszcze wczytany do bazy TimesTen, operacja importu danych zostaje wykonana automatycznie. Mechanizm ten łączy się zazwyczaj z automatycznym usuwaniem niepotrzebnych danych. Dzięki temu powstaje grupa, która zmienia się w sposób dynamiczny.

3.5. Grupy globalne i lokalne

Do każdej bazy centralnej może zostać przyłączona dowolna liczba węzłów – baz TimesTen. W związku z tym może dojść do sytuacji, w której należy rozważyć problem współbieżności pracy. W wielu przypadkach możliwe jest takie zaplanowanie systemu, aby dane były odpowiednio partycjonowane – dzięki temu zapewniamy, że każda baza modyfikująca dane w bazie centralnej nie nadpisze danych modyfikowanych przez inne źródło. Może się jednak zdarzyć, że wprowadzenie takiego partycjonowania nie jest możliwe. Czasem niezbędne jest, aby każda z baz-węzłów mogła modyfikować dowolne dane przy jednoczesnym zachowaniu spójności całego rozwiązania. W takiej sytuacji istnieje możliwość zastosowania jednej z opcji dostępnych w IMDB Cache. Podczas definiowania grupy możemy mianowicie określić, czy ma być ona lokalna czy globalna.

Grupa lokalna jest domyślnym typem funkcjonowania grupy. W tym przypadku nie zakłada się, że te same dane będą modyfikowane w różnych bazach TimesTen, lub też, że wprowadzone zostało odpowiednie partycjonowanie danych.

W przypadku grupy globalnej zakłada się wysoki poziom współbieżności pracy. Jeśli jedna z baz ma modyfikować dane, musi najpierw przejąć nad nimi kontrolę. Wiąże się to, oczywiście, z pewnym dodatkowym narzutem czasowym, jednak dzięki temu modyfikacji danych dokonuje zawsze jedna baza TimesTen. Tylko dynamiczna grupa z asynchronicznym zapisem danych może zostać zdefiniowana jako grupa globalna.

3.6. Przesyłanie danych pomiędzy bazami TimesTen i Oracle

Wprowadzane w poszczególnych bazach zmiany mogą być przesyłane w obrębie całej infrastruktury. Operacja importu zmian z bazy centralnej do bazy TimesTen nosi nazwę odświeżania, natomiast wysyłanie modyfikacji w przeciwnym kierunku nazywane jest propagacją. Obie operacje mogą być wykonywane automatycznie lub ręcznie.

W przypadku odświeżania automatycznego (*Autorefresh*) aktualizacja następuje co określony odstęp czasu, a nanoszenie modyfikacji może odbywać się przyrostowo lub w sposób pełny. Wyboru odpowiedniej metody dokonuje sam użytkownik. Jeśli zmiany występują często i dotyczą niewielkiej grupy rekordów, lepszym sposobem jest aktualizacja przyrostowa. Druga metoda sprawdza się dla rzadszych aktualizacji, obejmujących duże zakresy danych.

Przy odświeżaniu ręcznym (*Manual Refresh*) aktualizacja danych odbywa się na żądanie za pomocą komendy *REFRESH CACHE GROUP* (równoznaczne z wykonaniem *UNLOAD* i *LOAD*). Metoda sprawdza się w sytuacjach, gdy konieczność odświeżenia można przewidzieć.

W trybie automatycznej propagacji (*Propagate*), każda zmiana wykonana w TimesTen jest w sposób automatyczny przesyłana do bazy Oracle. Dla zapisu asynchronicznego przesyłanie następuje w tle, a w zapisie asynchronicznym odbywa się na bieżąco.

Natomiast w przypadku ręcznej propagacji (*Flush*) wszystkie zmiany dokonane na bazie TimesTen przesyłane są dopiero w momencie użycia odpowiedniej komendy przez użytkownika (*FLUSH CACHE GROUP*). Ograniczeniem tego rozwiązania jest niemożność przesyłania informacji o usunięciu rekordu.

3.7. Automatyczne oczyszczanie bazy (Automatic Data Aging)

W sytuacji gdy analiza danych dotyczy wyłącznie najświeższych informacji, pomocne okazuje się zastosowanie automatycznego oczyszczania bazy. Pozwala ono na usunięcie niepotrzebnych rekordów na podstawie dwóch algorytmów. Pierwszy z nich (*Usage-Based Aging*) swoje działanie opiera na statystykach użycia poszczególnych rekordów. Dane nie-

używane przez dłuższy czas podlegają automatycznemu czyszczeniu. Druga z metod umożliwia usuwanie rekordów na podstawie czasu (*Time-Based Aging*). Rekordy starsze od ustalonej wartości są automatycznie kwalifikowane do usunięcia w trakcie oczyszczania bazy [7].

4. Przykład zastosowania

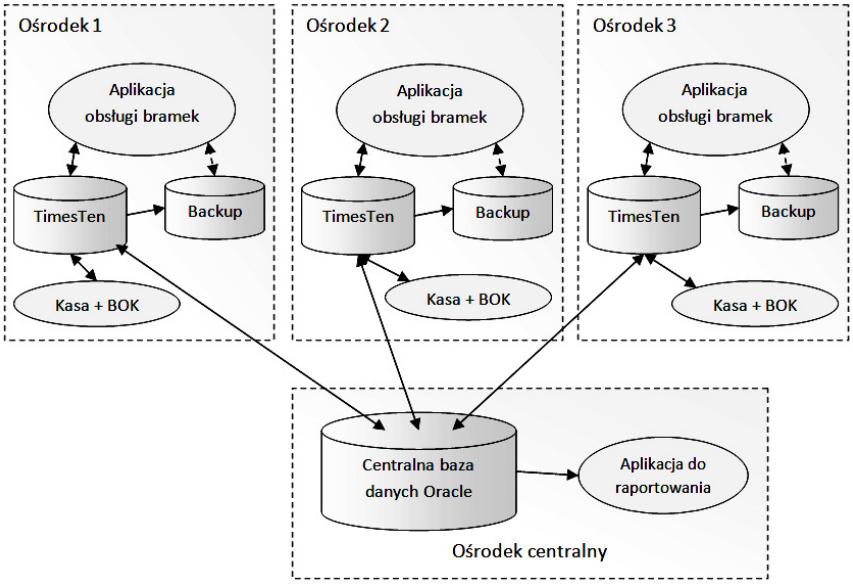
Opisane zalety bazy TimesTen, oraz mechanizmy oferowane przez IMDB Cache można wykorzystać do stworzenia bardzo ciekawych rozwiązań. Jednym z nich jest przedstawiony w tym rozdziale system do obsługi sieci ośrodków narciarskich.

4.1. Założenie

Głównym założeniem było zaimplementowanie rozwiązania zdolnego do obsługi wielu wyciągów rozmieszczonych w odległych od siebie ośrodkach. Celem było stworzenie wydajnego mechanizmu kontroli karnetów oraz systemu służącego do zarządzania nimi. Dodatkowo zaletą tego rozwiązania jest wprowadzenie możliwości przenoszenia informacji o wykupionych usługach pomiędzy różnymi lokalizacjami. Dzięki temu użytkownicy korzystający z jednego wyciągu mogą bez przeszkód wykorzystać zakupione usługi na pozostałych obiektach. Klient posiadający karnet może przemieszczać się i korzystać ze wszystkich wyciągów należących do budowanego systemu. Cała architektura umożliwia skalowanie, dzięki czemu sieć łatwo może być rozbudowywana. Dodatkową niezawodność i bezpieczeństwo całości zapewnia mechanizm replikacji baz TimesTen znajdujących się w poszczególnych lokalizacjach.

Głównym elementem systemu jest baza centralna gromadząca informacje niezbędne do funkcjonowania całej sieci ośrodków. Każdy z ośrodków posiada osobną bazę TimesTen wykorzystywaną przez wszystkie systemy w obrębie danej lokalizacji. Rozwiązanie takie, dzięki odpowiedniemu wykorzystaniu mechanizmu IMDB Cache, pozwala na importowanie danych znajdujących się w bazie centralnej, dzięki czemu każdy z węzłów – baz TimesTen – będzie miał dostęp do dowolnych informacji. Dodatkowo operacje związane z obsługą karnetów i korzystaniem z poszczególnych usług przeprowadzane są lokalnie. Dzięki temu wszystkie podstawowe procesy wykonywane podczas funkcjonowania ośrodka realizowane są możliwie szybko. Poza wymianą informacji o wykupionych karnetach i wykorzystanych usługach, bazy TimesTen ograniczają przechowywanie informacji zbędnych w danym momencie. Odpowiednie skonfigurowanie mechanizmu automatycznego oczyszczania bazy TimesTen (*Automatic Data Aging*) sprawia, że nieaktualne dane dotyczące historii, które zostały już umieszczone w bazie centralnej, mogą zostać bezpiecznie usunięte z baz lokalnych.

Aby zapewnić niezawodność całego rozwiązania, zastosowany został mechanizm replikacji bazy TimesTen w celu utworzenia baz zapasowych [9]. Dzięki temu w momencie awarii można przełączyć wszystkie systemy na bazę zapasową, a praca całego ośrodka nie jest wstrzymywana. Ogólny schemat przedstawiono na rysunku 3.



Rys. 3. Schemat systemu
Fig. 3. System schema

4.2. Instalacja i konfiguracja

Pierwszym etapem tworzenia opisywanego systemu jest przygotowanie (instalacja oraz konfiguracja) bazy centralnej oraz baz TimesTen stanowiących poszczególne węzły sieci.

4.2.1. Baza centralna

Instalacja serwera dla centralnej bazy danych, z punktu widzenia planowanej współpracy z bazą TimesTen, nie wymaga podejmowania żadnych dodatkowych czynności. Należy jedynie zwrócić uwagę na dwie kwestie: domyślne kodowanie znaków (kodowanie w bazie Oracle oraz TimesTen musi być identyczne – należy wybrać takie, które jest wspierane przez obie bazy [7]) oraz kolejność instalowania systemów (jeżeli obie bazy, baza TimesTen oraz baza Oracle, mają znaleźć się na jednej maszynie, to TimesTen musi zostać zainstalowana jako druga w kolejności).

Kluczowe dla funkcjonowania Oracle TimesTen IMDB Cache jest utworzenie odpowiednich użytkowników na bazie centralnej. Standardowa konfiguracja zakłada utworzenie co najmniej trzech użytkowników:

- właściciela wszystkich obiektów związanych z prawidłowym funkcjonowaniem mechanizmu IMDB Cache,
- administratora mechanizmu Cache (*Cache Administrator*),

- właściciela schematu i wszystkich obiektów utworzonych w jego obrębie.

W celu usprawnienia tego procesu Oracle dostarcza odpowiedni skrypt initCacheGlobalSchema.sql, który umożliwia założenie użytkownika o nazwie timesten (właściciela obiektów mechanizmu IMDB Cache), utworzenie obiektów zawierających informacje niezbędne do funkcjonowania mechanizmu IMDB Cache (należy wcześniej utworzyć przestrzeń tabel przeznaczoną dla tych obiektów) oraz utworzenie roli TT_CACHE_ADMIN_ROLE, definiującej zbiór niezbędnych uprawnień dotyczących utworzonych obiektów [8].

4.2.2. Baza TimesTen

Podczas instalacji Oracle TimesTen oferowane są następujące komponenty:

- TimesTen Data Manager – zawiera główny moduł serwera bazy danych. Instalacja wyłącznie tego komponentu jest wystarczająca do stworzenia lokalnie działającej bazy.
- TimesTen Data Manager Debug Libraries – instaluje biblioteki, które ułatwiają wykrywanie problemów w trakcie tworzenia oprogramowania (domyślnie nieinstalowany).
- TimesTen Server – przeznaczeniem tego komponentu jest umożliwienie łączenia się do bazy TimesTen ze zdalnych maszyn. Omawiany system zakłada funkcjonowanie różnych modułów, jak np. aplikacja bramek, moduł kasowy czy aplikacja do obsługi Biura Obsługi Klienta, dlatego też instalacja tego komponentu jest niezbędna.
- TimesTen Client – umożliwia połączenie do zdalnej bazy TimesTen. W momencie instalacji bazy centralnej instalacja tego komponentu nie jest konieczna. Musi on jednak zostać zainstalowany na komputerach, z których uruchamiane będą aplikacje klienckie.

Stworzenie nowego repozytorium danych polega na utworzeniu tzw. DSN (*Data Source Name*). W przypadku systemu Windows czynność te można wykonać tworząc nowe źródło danych ODBC.

Podobnie jak w przypadku bazy centralnej, również dla TimesTen pierwszą czynnością niezbędną do wykonania na bazie danych jest utworzenie odpowiednich użytkowników. W przypadku bazy stanowiącej węzeł całego rozwiązania wystarczy utworzyć co najmniej dwóch użytkowników:

- Użytkownika zarządzającego grupami oraz całą siecią Cache (*Cache Manager User*). Nazwa tego użytkownika powinna być taka sama, jak nazwa użytkownika mogącego uzyskać dostęp do tabel w bazie centralnej, które wejdą w skład grup. Może to być administrator Cache, właściciel obiektów lub też inny użytkownik posiadający stosowne uprawnienia.
- Co najmniej jednego użytkownika będącego właścicielem tabel tworzących grupy (*Cache Table User*). Każda tabela zaimportowana za pomocą mechanizmu Cache posiada swojego niezmienionego właściciela. Utworzenie użytkownika TimesTen o takiej samej nazwie sprawi, że stanie się on automatycznie właścicielem tych obiektów.

4.2.3. Zakładanie Cache Grid

Po przeprowadzeniu procesu instalacji pierwszej z baz TimesTen oraz utworzeniu odpowiednich użytkowników można przystąpić do operacji tworzenia Cache Grid. Operację tę wykonać można z poziomu dowolnej bazy TimesTen. W tym celu logujemy się jako użytkownik cacheuser i wywołujemy poniższą procedurę:

```
Command>  call ttCacheUidPwdSet('cacheuser','oracle');
```

Parametrami tej procedury są kolejno: nazwa użytkownika będącego administratorem Cache na bazie centralnej, oraz jego hasło. Jest to pierwsze polecenie, które należy wykonać na dowolnej bazie dołączanej do naszej infrastruktury. Wprowadzenie tych informacji umożliwia wykonanie właściwego polecenia, tworzącego Cache Grid:

```
Command>  call ttGridCreate('SkiResorts');
```

Parametrem powyższej procedury jest nazwa tworzonej sieci. Operację tę przeprowadza się tylko raz dla całej infrastruktury połączonej za pomocą mechanizmu IMDB Cache. Po pomyślnym wykonaniu polecenia wystarczy już tylko podłączyć aktualną bazę TimesTen do nowo utworzonej sieci Cache Grid:

```
Command>  call ttGridNameSet('SkiResorts');
Command>  call ttGridAttach(1,'SkiResort1','SRV1',5002);
```

Te dwa polecenia wykonuje się kolejno dla każdej utworzonej bazy TimesTen, która powinna zostać przypisana do wspólnej sieci. Pierwszy z parametrów drugiego polecenia określa, że mamy do czynienia z samodzielną bazą. Drugi stanowi uniwersalny dla danej bazy identyfikator. Dwa ostatnie parametry to nazwa maszyny, na której zainstalowana została baza TimesTen oraz port służący do łączenia się z serwerem Oracle. W tym momencie dysponujemy podstawowym środowiskiem, w którym możemy rozpocząć tworzenie grup.

4.2.4. Struktura bazy danych

Tabele w omawianym systemie podzielić można ze względu na ich przeznaczenie, na trzy grupy: tabele zawierające informacje o strukturze sieci ośrodków narciarskich, tabele zawierające informacje o karnetach oraz tabele zawierające dane dotyczące historii.

Ponieważ dane dotyczące struktury sieci ośrodków narciarskich muszą być dostępne dla każdej lokalizacji, utworzona zostanie odpowiednia grupa, za pomocą której będzie można zaimportować dane do poszczególnych baz TimesTen. Najlepiej do tego celu będzie się nadawała grupa „tylko do odczytu”. Dane w tych tabelach nie będą ulegały częstym zmianom, dlatego można ustalić interwał odświeżania na jedną godzinę. W przypadku gdyby zaszła potrzeba zaktualizowania całej grupy, czynność tę można przeprowadzić ręcznie.

```
Command> CREATE READONLY CACHE GROUP SRCG_SR_STRUCTURE
          AUTOREFRESH INTERVAL 60 MINUTES STATE ON
```

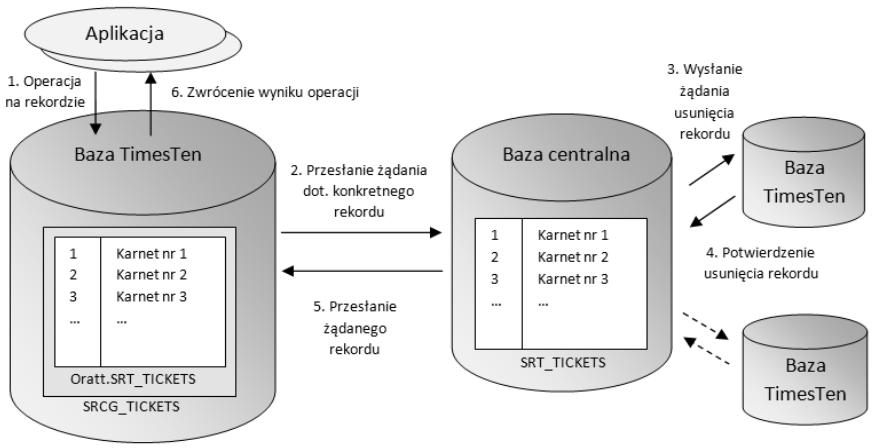
```
FROM oratt.SRT_SKI_RESORTS (
SKI_RESORT_ID NUMBER NOT NULL,
... kolejne kolumny ...
PRIMARY KEY (SKI_RESORT_ID)
),
oratt.SRT_SKI_LIFTS (
SKI_LIFT_ID   NUMBER NOT NULL,
... kolejne kolumny ...
PRIMARY KEY (SKI_LIFT_ID),
FOREIGN KEY (SKI_RESORT_ID) REFERENCES oratt.SRT_SKI_RESORTS
) WHERE (oratt.SRT_SKI_LIFTS.STATUS = 'AC');
```

4.2.5. Przechowywanie informacji o karnetach

Dane karnetów przechowywane są w bazie centralnej, skąd za pomocą mechanizmu IMDB Cache udostępniane są na poszczególne bazy TimesTen. Z uwagi na fakt, że w przeważającej większości przypadków bilet funkcjonuje w obrębie poszczególnych ośrodków, warto stworzyć infrastrukturę, która będzie przechowywała w danym momencie wyłącznie informacje o karnetach wykorzystywanych w konkretnym ośrodku. Wczytanie danych z innego ośrodka narciarskiego (karnetu zarejestrowanego w innym obiekcie) powinno następować dopiero na żądanie podczas pierwszej kontroli karnetu w nowym ośrodku. Do tego celu najkorzystniejsze wydaje się skorzystanie z globalnej, dynamicznej grupy z zapisem asynchronicznym, dla której uruchomiony zostanie mechanizm automatycznego oczyszczania danych [7]. Jej działanie zaprezentowano na rysunku 4.

Omawiana grupa tworzona jest z użyciem poniższego polecenia.

```
Command>  CREATE DYNAMIC ASYNCHRONOUS WRITETHROUGH GLOBAL CACHE GROUP
          SRCG_SR_TICKETS FROM oratt.SRT_TICKETS (
             TICKET_ID NUMBER NOT NULL,
          ... kolejne kolumny ...
             PRIMARY KEY (TICKET_ID)
          ) AGING LRU ON;
```



Rys. 4. Działanie globalnej, dynamicznej grupy z zapisem asynchronicznym
Fig. 4. Dynamic, global, asynchronous writethrough cache group working

Podczas tworzenia grupy nie określa się od razu parametrów oczyszczania. Operacja ta wykonywana jest osobno przy użyciu wewnętrznej procedury: ttAgingLRUConfig. Pierw-

szy parametr określa stopień wypełnienia bazy danych, który należy osiągnąć po przeprowadzeniu oczyszczania, drugi określa stopień zajętości powodujący rozpoczęcie oczyszczania, trzeci natomiast odstęp (w minutach) pomiędzy kolejnymi kontrolami zajętości bazy. Stworzona w ten sposób konfiguracja dotyczy całej bazy.

```
Command>  CALL ttAgingLRUConfig(.75, .95, 15);
```

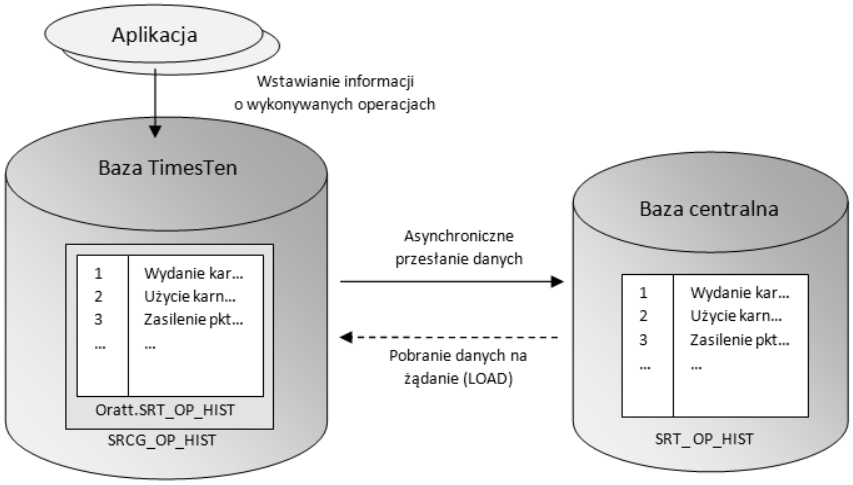
4.2.6. Przechowywanie danych historycznych

W przypadku grup składających się z tabel zawierających wpisy historyczne, również nie zachodzi konieczność importowania wszystkich danych. W większości przypadków ich analiza będzie dotyczyła tylko najświeższych wpisów (dodatkowe dane można pobrać na żądanie). Wystarczy zatem utworzyć dynamiczną grupę z asynchronicznym zapisem (rys. 5). Nie istnieje konieczność zakładania grupy globalnej, gdyż dane wstawiane w poszczególnych ośrodkach nie będą modyfikowane. Wykorzystano mechanizm automatycznego oczyszczania bazy oparty na czasie, który co godzinę będzie usuwał wpisy starsze niż jeden dzień.

Polecenie tworzące przykładową grupę zamieszczone jest poniżej.

```
Command>  CREATE DYNAMIC ASYNCHRONOUS WRITETHROUGH CACHE GROUP
          SRCG_SR_USAGE_HISTORY FROM ORATT.SRT_USAGE_HISTORY (
             ENTRY_ID NUMBER NOT NULL,
             ... kolejne kolumny ...
             PRIMARY KEY (ENTRY_ID)
          ) AGING USE ENTRY_DATE LIFETIME 1 DAYS CYCLE 60 MINUTES ON;
```

W sytuacji, gdy wiele baz jednocześnie wprowadza informacje do systemu, warto zastanowić się nad problemem unikalności kluczy głównych przesyłanych do bazy centralnej. Jeżeli każda baza będzie generowała swoje klucze z sekwencji o tym samym zakresie, prędzej lub później dojdzie do konfliktu. Oczywiście, można by zastosować w tej sytuacji różne zakresy, jednak rozwiązanie to jest trochę mało elastyczne. Dodanie kolejnej bazy do całej infrastruktury wymagałoby zmiany zakresów pozostałych baz. Dużo lepszym rozwiązaniem, znanym z innych systemów bazodanowych, jest wykorzystanie globalnych identyfikatorów UUID. Jednak niestety, baza TimesTen nie udostępnia własnych procedur zdolnych do ich generowania [10, 11]. W sytuacji gdy część logiki aplikacji ma zostać zaszyta w postaci pakietów PL/SQL rozwiązanie to staje się bezużyteczne.



Rys. 5. Działanie dynamicznej grupy z asynchronicznym zapisem
Fig. 5. Dynamic asynchronous writethrough cache group working

W przypadku prezentowanego systemu zdecydowano się na zastosowanie własnego rozwiązania. Polega ono na połączeniu numerów generowanym przez sekwencje, z pewnym dodatkowym parametrem każdej bazy, ustalanym unikalnie w obrębie całej infrastruktury. Połączenie to może zostać uzyskane dzięki zastosowaniu prostego przekształcenia matematycznego. Przykładowo, jeśli zakładamy, że nasza sieć nie przekroczy 100 węzłów, wprowadzamy dla każdej z baz unikalny parametr – dwucyfrową liczbę. Następnie każdą wartość wygenerowaną z sekwencji wystarczy przemnożyć przez 100 i dodać do niej wspomnianą wartość parametru. Jeśli np. parametrem danej bazy jest liczba 14, a sekwencja zwróci kolejny numer 12331, kolejnym kluczem głównym będzie liczba 1233114.

4.3. Implementacja końcowych aplikacji

Najnowsze wydanie bazy Oracle TimesTen (wersja 11) oddaje w ręce swoich użytkowników doskonałe narzędzie, jakim jest wsparcie dla języka PL/SQL [10]. Jest to ten sam język skryptowy, który funkcjonuje na serwerze bazodanowym Oracle. W związku z tym większa część warstwy logicznej systemu może być tworzona w bazie w postaci pakietów i znajdujących się w nich procedur oraz funkcji.

Do zaimplementowania aplikacji wykorzystany został język C#. Co prawda Oracle nie udostępnia dedykowanych bibliotek, jak np. w przypadku Javy, jednak nadal można wykorzystać standardowy interfejs ODBC [1]. Sposób wykonywania operacji na bazie TimesTen wygląda niemal tak samo jak w przypadku standardowej bazy Oracle. Największą różnicę stanowi wywoływanie funkcji i procedur napisanych w PL/SQL. Żadna procedura ani funkcja napisana przez użytkownika nie może zostać użyta w zapytaniu SELECT, a jej wywołanie musi odbyć się wewnątrz bloku anonimowego [10]. Jednak ADO.NET umożliwia wyko-

nywanie całych bloków anonimowych, więc powyższy sposób jest możliwy do zastosowania. Przykładowe użycie jednej z procedur zaprezentowano na poniższym listingu.

```
OdbcConnection c =
    new OdbcConnection("DSN=SR1;UID=oratt;PWD=tpwd;OraclePWD=opwd");
OdbcCommand cmd =
    new OdbcCommand("BEGIN SRP_TICKETS.REG_TICKET(:TCODE); END;", c);
cmdReg.Parameters.Add(":TCODE", OdbcType.VarChar, 8).Value = txtTicketCode.Text;
try {
  cmdReg.Transaction = connection.BeginTransaction();
  cmdReg.ExecuteNonQuery();
  cmdReg.Transaction.Commit();
}
catch (Exception ex) {
  ...
  cmdReg.Transaction.Rollback();
}
finally {
  connection.Close();
}
```

Dodatkowym problem wynikłym w trakcie korzystania z języka C# jest fakt, że w .NET wszystkie zmienne typu String używają domyślnie kodowania Unicode (dokładnie UTF-16). Początkowo wybrane kodowanie UTF-8 (AL32UTF8) powodowało powstawanie błędów podczas automatycznej konwersji. Wyjściem z tej sytuacji okazało się zastosowanie na bazie standardowego kodowania systemu Windows (Windows-1250), które w TimesTen nosi nazwę EE8MSWIN1250. W tym przypadku automatyczna konwersja nie powoduje błędów.

5. Wady i zalety technologii TimesTen

Najważniejszą zaletą omawianej technologii jest z pewnością jej duża wydajność. Jest ona osiągana nie tylko dzięki samemu sposobowi działania (wyłączne wykorzystanie pamięci operacyjnej), ale też dzięki zastosowaniu wielu dodatkowych rozwiązań (np. rezygnacja z architektury klient-serwer na rzecz wykorzystania bibliotek współdzielonych i algorytmu MicroLogging). Warto jednocześnie wspomnieć, że osiągnięcie tej wydajności nie komplikuje w żaden sposób użytkowania bazy. Rozwiązanie wyposażono we wsparcie dla języka SQL oraz wiele użytecznych interfejsów dostępu do bazy danych. Ich wykorzystanie pozwala na integrację z większością dostępnych języków programowania, a przykładem tego jest wykorzystanie platformy .NET w tym artykule.

Cechą, która odróżnia rozwiązanie TimesTen od podobnych produktów, jest natywne wsparcie dla mechanizmów replikacji oraz integracji ze standardową bazą Oracle. Ich zastosowanie nie ma znaczącego wpływu na ogólną wydajność technologii, pozwala jednak na budowanie bardziej odpornych na awarie, w pełni skalowalnych rozwiązań. Dodatkowo niweluje to jedną z poważniejszych wad rozwiązań opartych na pamięci operacyjnej, tj. ograniczenie rozmiaru przechowywanych danych. Integracja z serwerem Oracle pozwala na prze-

chowywanie w bazie TimesTen tylko i wyłącznie niezbędnych wycinków większej ilości danych.

Trudno natomiast znaleźć poważniejsze wady omawianego rozwiązania. Z pewnością można by tutaj wymienić wysoki koszt i ograniczenia rozmiaru związane z wykorzystaniem pamięci RAM, jednak problem ten dotyczy raczej całej grupy produktów niż konkretnie technologii TimesTen. TimesTen, dzięki integracji z serwerem Oracle, radzi sobie stosunkowo dobrze z tymi problemami. Można by jeszcze tylko oczekiwać możliwości integracji technologii TimesTen z pozostałymi bazami danych, na wzór mechanizmu IMDB Cache.

6. Podsumowanie

W artykule zaprezentowano zastosowania baz danych funkcjonujących w pamięci operacyjnej na przykładzie rozwiązań firmy Oracle. Omówiono funkcjonalności niezależnej bazy danych Oracle TimesTen In-Memory Database, jak również rozszerzenia dostępnego dla tradycyjnych systemów bazodanowych firmy Oracle – Oracle TimesTen In-Memory Database Cache. Rozwiązania te mogą znaleźć szerokie zastosowanie w wielu branżach i stanowią alternatywę dla klasycznych systemów, które w trakcie pracy przechowują dane w pamięciach dyskowych. W artykule omówiono zagadnienia związane z konfiguracją takich środowisk, przedstawiono przykładowe zastosowanie oraz zestawiono wady i zalety tej technologii.

BIBLIOGRAFIA

1. Extreme Performance Using Oracle TimesTen In-Memory Database. An Oracle Technical White Paper, July 2009.
2. Using Oracle In-Memory Database Cache to Accelerate the Oracle Database. An Oracle White Paper, July 2009.
3. TimesTen Cache Connect to Oracle Guide Release 7.0, April 2007.
4. Oracle TimesTen In-Memory Database 7.0 Good Practices Guide. An Oracle White Paper, July 2007.
5. Oracle TimesTen In-Memory Database TTClasses Guide Release 7.0, September 2007.
6. The TimesTen Team. Mid-tier caching: the TimesTen approach. ACM SIGMOD international conference on Management of data, NY, 2002, s. 588÷593.
7. Oracle TimesTen In-Memory Database. Operations Guide, Release 11.2.1.
8. Oracle TimesTen In-Memory Database. Cache User's Guide, Release 11.2.1.

9. Oracle TimesTen In-Memory Database. TimesTen to TimesTen Replication Guide. Release 11.2.1.
10. Oracle TimesTen In-Memory Database. PL/SQL Developer's Guide, Release 11.2.1.
11. Oracle TimesTen In-Memory Database. Java Developer's Guide, Release 11.2.1.
12. Narayanamurthy N.: How real is Real Time Database. Torry Harris Business Solutions, www.thbs.com/pdfs/how_real_is_real_time_database.pdf .
13. http://in-memory_database.totallyexplained.com/ .

Recenzent: Dr inż. Bożena Małysiak-Mrozek

Wpłynęło do Redakcji 16 stycznia 2010 r.

Abstract

The growth of requirements that have been put on actual database systems owing to number of operations executed simultaneously causes working out new solutions in this area. One of the possible solutions is databases functioning in random access memory. In-Memory Database technology (IMDB) assumes relational database implementation, in which all data are stored in RAM from the moment of database start and data structures as well as access algorithms use this fact in order to achieve maximum efficiency.

The Oracle Company provides users with two products of such a type: Oracle TimesTen In-Memory Database and Oracle In-Memory Database Cache. Oracle TimesTen In-Memory Database is independent database, which functionality is first of all based on random access memory. Hard drive is used only to keep durability as well as recovery in the case of system failures. Oracle In-Memory Database Cache is an option available for standard Oracle database, functioning on the basis of Oracle TimesTen In-Memory Database. The main assumption of this option is to make possible the access to data located in existing Oracle database, using at the same time advantages offered by TimesTen database. By making special mechanisms available, which allow importing data from TimesTen database as well as propagation of changes made on it back to Oracle database, we gain the possibility to execute operations significantly more efficiently. In this work implementation of exemplary system using discussed technologies was presented.

Adresy

Krzysztof CZAJKOWSKI: Politechnika Krakowska, Wydział Fizyki, Matematyki i Informatyki Stosowanej, Katedra Teleinformatyki, ul. Warszawska 24, 31-155 Kraków, Polska, kc@pk.edu.pl .

Jakub PAŹDZIORKO: Politechnika Krakowska, Wydział Inżynierii Elektrycznej i Komputerowej, ul. Warszawska 24, 31-155 Kraków, Polska, jpazdziorko@gmail.com .