STUDIA INFORMATICA 2010
Volume 31 Number 2B (90)

Małgorzata BACH, Adam DUSZEŃKO, Aleksandra WERNER
Politechnika Śląska, Instytut Informatyki

KONCEPCJA PAMIĘCIOWYCH BAZ DANYCH ORAZ WERYFIKACJA PODSTAWOWYCH ZAŁOŻEŃ TYCH STRUKTUR

Streszczenie. Pomysł na bazy danych przetwarzane i przechowywane w całości w pamięci operacyjnej pojawił się dawno i choć dawał dobre wyniki, nie był możliwy do szerokiego zastosowania ze względu na ograniczenia sprzętowe. Aktualnie, choć nie w całości, możliwe jest wykorzystanie, cały czas rosnącej, przestrzeni pamięci operacyjnej na potrzeby baz danych w postaci baz pamięciowych. Niniejszy rozdział przedstawi, aktualnie oferowane przez wiodących producentów, możliwości realizacji baz pamięciowych.

Słowa kluczowe: pamięciowe bazy danych, TimesTen, SolidDB

THE CONCEPT AND VERIFICATION OF THE BASIC ASSUMPTIONS OF IN-MEMORY DATABASE'S STRUCTURES

Summary. The conception of storing and managing data directly in RAM memory appeared some time ago and in spite of very good efficiency, it was impossible to massive implementation because of very important hardware limitations. Currently it's still not possible to store whole databases in memory but there are some tools to organize pieces of data as In-memory databases. This chapter presents strong and weak elements of In-memory data management.

Keywords: In-memory databases, TimesTen, SolidDB

1. Wstęp

Tworzenie bazy danych w klasycznym podejściu sprowadza się do wyboru modelu organizacji danych, a następnie – w oparciu o jego reguły – do utworzenia konkretnych struktur. Poza zestawem reguł stanowiących podstawę poprawności stworzonej struktury (na przykład w formie tzw. postaci normalnych modelu danych) istnieją reguły decydujące o jakości roz-

wiązania pod względem wydajności. Przez wydajność rozumie się tu, oczywiście, możliwość wykonania zadania wyszukiwania i aktualizacji danych w jak najkrótszym czasie. Pojawienie się języka XML w świecie baz danych rozszerzyło znacząco możliwości modelowania danych w relacyjnych bazach danych [13] (aktualnie często nazywanych bazami relacyjno-hierarchicznymi). Jednak wybór pomiędzy modelowaniem w czysto relacyjnej lub w hierarchicznej przestrzeni nie jest jedynym wyborem jakiego można dokonać, by poprawić wydajność. Po ustaleniu struktury modelu można mianowicie zdecydować, czy całą strukturę lub jej części umieścić w bazie danych dyskowej czy też pamięciowej. Wykorzystanie baz pamięciowych, ich wpływ na wydajność oraz podstawowe konfiguracje rozwiązania przedstawione zostaną w dalszych podpunktach.

2. Zarządzanie pamięcią

Większość systemów informatycznych rozróżnia jawnie dwie przestrzenie pamięci. Podstawowa, na której operuje procesor, to pamięć operacyjna – czyli pamięć tzw. ulotna. Jej podstawowymi parametrami są: duża szybkość odczytu i zapisu, ale również wysoka cena i nadal ograniczona pojemność. Uzupełnieniem tej pamięci są pamięci trwałe, aktualnie w dominującej mierze pamięci dyskowe, a więc mechaniczno-magnetyczne urządzenia realizujące zapis na wirujących dyskach magnetycznych. Ich pojemność i cena są znacznie korzystniejsze od pamięci operacyjnej, jednak prędkość podstawowych operacji zapisu i odczytu jest wielokrotnie mniejsza. Pamięć dyskowa wykorzystywana jest zarówno jako magazyn danych wczytywanych w miarę potrzeb do pamięci i przechowujący aktualnie niepotrzebne dane, ale również jako pamięć wirtualna, w którym to rozwiązaniu pamięć ta emuluje brakującą pamięć operacyjną. Oczywiście, drugie rozwiązanie jest przykrą koniecznością i obarczone jest dużym spadkiem wydajności systemu, jednak pozwala w krytycznych momentach utrzymać zdolność systemu do dalszego działania.

Pewnym unikalnym rozwiązaniem w tym zakresie są systemy klasy IBM AS/400 (aktualnie iSeries, i5), w których stworzona została płaska przestrzeń pamięci [11, 12] (ang. *Single Level Storage*), w której niskopoziomowe mechanizmy sprzętowe zapewniają wirtualizację całej przestrzeni pamięci – zarówno operacyjnej, jak i dyskowej – w postaci jednej ciągłej pamięci. Jest to jednak rozwiązanie unikalne, a w większości spotykanych systemów istnieje rozgraniczenie pomiędzy przestrzenią pamięci dyskowej a operacyjnej.

Powyższe rozgraniczenie ma ścisłe przełożenie na bazy danych, jednak warto podkreślić, że bazy dyskowe nie wykonują wszystkich operacji bezpośrednio na nośnikach trwałych. W celu zminimalizowania opóźnień związanych z operacjami dyskowymi stosowane są bufory pamięciowe, pośredniczące w operacjach odczytu i zapisu. Tak jak operacje odczytu nie

sprowadzają się do odczytania pojedynczych wierszy tabeli, ale całej strony zawierającej wiele wierszy, tak operacje modyfikacji czy dodawania wierszy w pierwszym kroku wykonywane są na buforach przechowywanych w pamięci operacyjnej, a dopiero w drugim kroku zapisywane są na dysk. Trwałość operacji bazodanowych w tym rozwiązaniu realizowana jest głównie na poziomie dziennika transakcji odnotowującego, które operacje modyfikacji zostały już rozpropagowane na pliki tabel na dyskach twardych, a które są nadal w buforach.

Operowanie stronami przy odczycie i zapisie znacząco poprawia wydajność baz dyskowych, jednak czasem intensywność operacji wejścia-wyjścia sprawia, iż mechanizm ten staje się niewystarczający. Oczywiście, stosowanie jako przestrzeni dyskowych rozwiązań macierzowych, pozwalających na zrównoleglenie operacji zapisu i odczytu lub definiowanie struktur pamięciowych w postaci tabel tymczasowych poprawia wydajność, jednak nie w takim stopniu, w jakim jest to potrzebne.

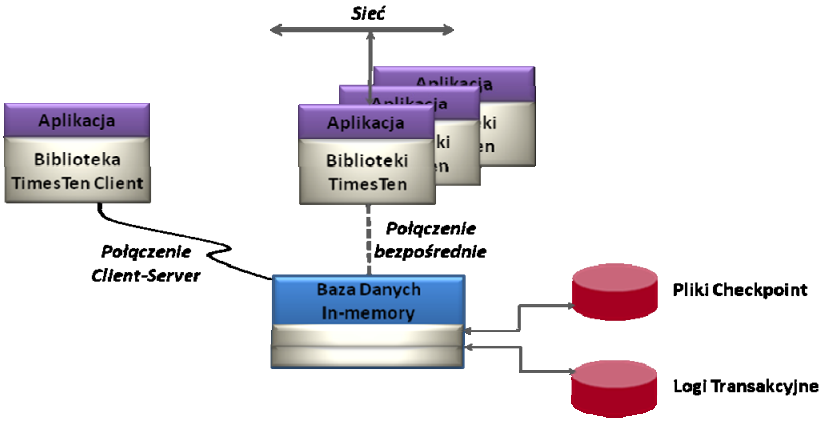
W początkach baz danych pojawiła się koncepcja, aby systemy zarządzające bazami danych opierały się na pamięci operacyjnej jako głównej i wręcz jedynej przestrzeni gromadzenia danych. Jednak intensywny wzrost wielkości baz danych spowodował, że bardzo szybko ich rozmiar znacznie przekroczył skromne, zwłaszcza w tamtych czasach, wielkości pamięci operacyjnej. W tej sytuacji praktycznie odstąpiono zupełnie od tego pomysłu na organizację SZBD. Aktualnie, gdy cena pamięci znacznie zmalała, a uzyskiwane pojemności są całkiem spore, ponownie powrócono do koncepcji baz pamięciowych.

Na pierwszy rzut oka korzyści wynikające z użycia baz pamięciowych są oczywiste i trudno się spodziewać, aby mogły z nimi pod względem wydajnościowym konkurować bazy dyskowe. Należy jednak pamiętać, iż bazy te również posiadają pewne elementy, które mogą zaważyć na ich wydajności i te elementy zostaną w tym opracowaniu zweryfikowane. Dla przykładu, wspomniana wcześniej trwałość danych, która nie może być poniechana w bazach pamięciowych, wymaga zastosowania dodatkowych mechanizmów zabezpieczających spójność danych i trwałość transakcji w przypadku niekontrolowanego wstrzymania pracy systemu. Uzyskuje się to przez prowadzenie, tak jak w bazach dyskowych, dziennika transakcji zapisywanego w miarę wykonywania operacji na nośniku trwałym. Dzięki temu w razie niekontrolowanej przerwy w pracy i utraty zawartości pamięci operacyjnej istnieje możliwość odtworzenia jej stanu ponownie poprzez wykonanie wszystkich operacji zarejestrowanych w dzienniku transakcji. Drugim rozwiązaniem, znacznie szybszym, jest zastosowanie struktur klastra niezawodnościowego. Zdublowanie serwera bazy pamięciowej i równoległe operowanie na dwóch systemach pozwala w przypadku awarii jednego z nich przenieść działanie systemu na węzeł rezerwowy lub chociaż bezpiecznie złożyć zawartość pamięci operacyjnej na nośniki trwałe [2, 3, 9].

2.1. TimesTen i SolidDB jako przykłady systemów pamięciowych baz danych

Obecnie praktycznie każdy liczący się dostawca systemów baz danych oferuje możliwość realizacji części lub całej bazy danych w postaci pamięciowej. W niniejszym opracowaniu bardziej szczegółowej analizie zostaną poddane rozwiązania firm Oracle i IBM, które proponują dwie konfiguracje baz In-memory [1, 2, 8, 9].

W pierwszym z rozwiązań organizacji baz pamięciowych (zastosowanym w produktach Oracle TimesTen oraz IBM SolidDB) struktury danych, indeksy czy algorytmy optymalizacji zapytań zostały zaprojektowane przy założeniu, że wszystkie wymagane dane są przechowywane w całości w pamięci ulotnej. Pamięć dyskowa jest tu stosowana tylko do tworzenia kopii zapasowych i odzyskiwania danych, przy czym okresowa synchronizacja z obrazem dyskowym odbywa się w sposób nieblokujący. Ogólna architektura tego rozwiązania, na przykładzie bazy danych TimesTen, została przedstawiona na rysunku 1.

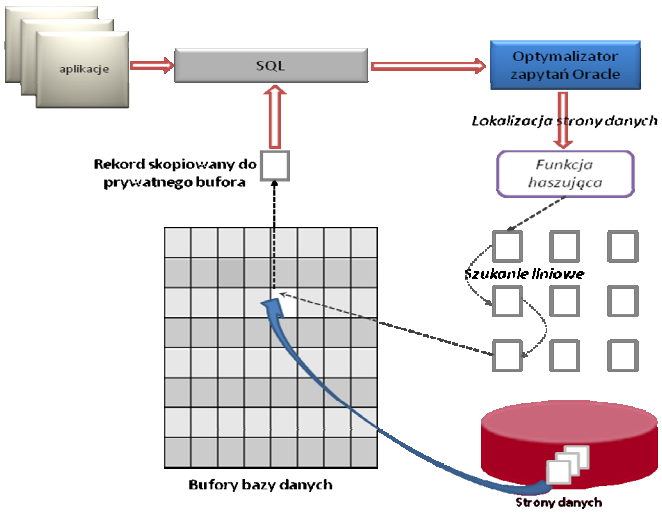


Rys. 1. Architektura bazy TimesTen
Fig. 1. TimesTen database architecture

Z kolei produkty Oracle In-Memory Database Cache oraz IBM SolidDB Universal Cache, to rozwiązania hybrydowe, umożliwiające wykorzystanie bazy In-memory jako pamięci podręcznej dla tradycyjnej (dyskowej) bazy relacyjnej. W tym przypadku baza pamięciowa zawiera kopię wybranych, krytycznych wydajnościowo podzbiorów danych i oferuje ich płynną synchronizację z relacyjną bazą danych, zapewniając efektywny dostęp do danych przechowywanych w tak skonfigurowanej pamięci podręcznej. Proces synchronizacji jest dla użytkownika w pełni transparentny.

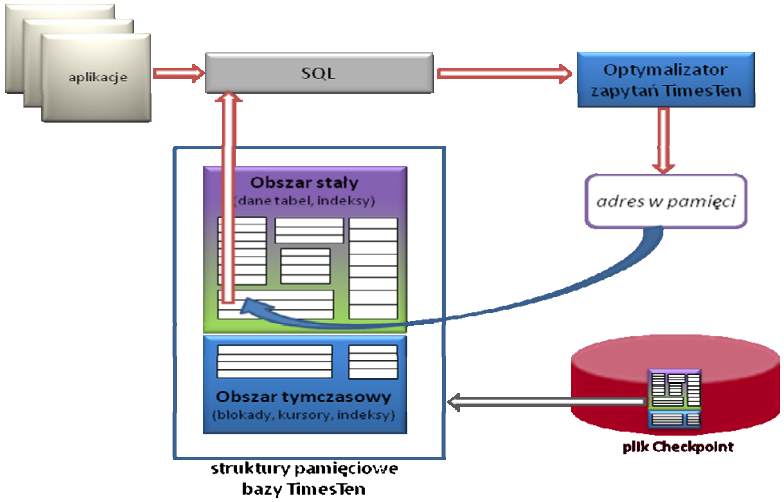
Funkcjonalność bazy danych In-memory TimesTen jest oparta na zbiorze bibliotek współdzielonych (ang. *shared libraries*), które mogą być konsolidowane z aplikacją i wywoływane bezpośrednio z kodu aplikacji, współdzieląc przestrzeń adresową z aplikacją. Dzięki takiemu rozwiązaniu eliminowana jest komunikacja międzyprocesowa czy komunikacja w trybie klient-serwer, co usprawnia proces przetwarzania zapytań kierowanych do bazy In-memory.

Jest to jedna z głównych różnic między bazą danych In-memory a tradycyjną bazą danych, która najczęściej jest implementowana jako kolekcja programów uruchamianych niezależnie od aplikacji jako osobne procesy.



Rys. 2. Przetwarzanie zapytań w konwencjonalnej bazie Oracle
Fig. 2. Query processing in Oracle conventional database

Na rysunku 2 zaprezentowano ideę komunikacji aplikacji klienckich z konwencjonalną bazą danych Oracle, natomiast na rys. 3 – z bazą TimesTen.



Rys. 3. Przetwarzanie zapytań w bazie TimesTen
Fig. 3. Query processing in TimesTen database

W pierwszym przypadku baza danych kopiuje bloki danych do buforów w pamięci, w drugim – aplikacje łączą się bezpośrednio z przestrzenią adresową bazy danych, więc wszystkie operacje są wykonywane wprost na stronach pamięci współdzielonej systemu operacyjnego zawierających właściwe dane [4, 5, 6, 7].

W celu zbadania wydajności systemów pamięciowych jako samodzielnych systemów zarządzania bazami danych, wykorzystane zostały konfiguracje autonomicznych baz pamięciowych, a nie tych pracujących w trybie Cache.

3. Środowisko testowe

Bazy pamięciowe jako priorytet traktują wydajność. Mimo iż oferują one praktycznie analogiczną funkcjonalność do tradycyjnych baz, ich głównym zadaniem nie jest tworzenie skomplikowanych struktur składowania i przetwarzania danych, ale szybki dostęp do źródła danych. W tej sytuacji należało odpowiednio ukierunkować badania wydajnościowe. Podstawową kwestią było wyznaczenie punktu odniesienia dokonywanych pomiarów.

3.1. Przyjęta metoda pomiaru

Przy tradycyjnych (dyskowych) bazach danych częstość operacji dyskowych powoduje, iż w dużej mierze wydajność tych baz uwarunkowana jest wydajnością mechanizmów składowania danych.

W przypadku baz pamięciowych, w których czynnik operacji dyskowych jest znacząco wyeliminowany, zarówno wydajność obliczeniowa, jak i pamięciowa jest kluczowa dla wydajności całego rozwiązania.

Na czas dostępu do danych T_d składa się: nawiązanie połączenia z bazą danych, wykonanie zapytania selekcjonującego poszukiwane rekordy danych oraz ich przesłanie do aplikacji, co można określić zależnością:

$$T_d = T_p + \mathrm{T_{wz}}(n) + T_{po}(n) + T_{pr}(n),$$

gdzie:

T_p – czas nawiązania połączenia,

T_{wz} – czas wykonania zapytania,

T_{po} – czas pobrania danych,

T_{pr} – czas przesłania danych.

W tym zestawieniu tylko czas nawiązania połączenia nie jest funkcją liczby rekordów n zwracanych przez zapytanie i dlatego jako element stały zostanie w pomiarach pominięty.

W związku z dużą popularnością serwerów aplikacyjnych opartych na języku Java, właśnie w tym języku napisano program, który łączy się z bazą danych poprzez interfejs JDBC, wykonuje zapytanie, a następnie pobiera cały zbiór wynikowy. Dane nie są buforowane po stronie aplikacji, aby nie wprowadzać dodatkowego kosztu nieistotnego dla samego pomiaru.

Te same zapytania kierowano do systemów dyskowych i odpowiadających im systemów pamięciowych, przy czym pomiary systemów dyskowych traktowano jako punkt odniesienia dla systemów pamięciowych.

Analizowana baza danych zawierała informacje o osobach zarejestrowanych w urzędzie pracy (m.in. ich danych osobowych i historii ich zatrudnienia).

Do testów przygotowany został zbiór kilkunastu zapytań charakteryzujących się różną selektywnością oraz złożonością operacji selekcji i projekcji, z których ostatecznie na potrzeby niniejszego opracowania wybrano 10 najbardziej – zdaniem autorów – reprezentatywnych. Wśród nich były zapytania:

- proste operujące na pojedynczej tabeli (zapytania o numerach: 1, 2, 3),
- proste ze złączeniami równościowymi i zewnętrznymi oraz (w 2 przypadkach) operatorem LIKE „%<*tekst*>%” w predykacie (zapytanie 4, 5, 6),
- z funkcjami agregującymi i warunkami nakładanymi na wynik agregacji (zapytanie nr 7 i 8),
- podzapytania – w tym podzapytania skorelowane (numery zapytań: 8, 9, 10).

Zarówno systemy dyskowe jak i pamięciowe zainicjowane zostały tym samym zbiorem danych. Początkowa liczba wierszy tabeli OSOBA (dla której zdefiniowano 75 kolumn) wynosiła 20 tys., natomiast tabeli OKRES_ZATRUDNIENIA (15 kolumn) 70 tys. W celu sprawdzenia skalowalności rozwiązań pomiary zostały powtórzone dla jedno-, dwu- i trzykrotnie zwielokrotnionych zbiorów danych.

Zaobserwowano, że wraz ze wzrostem liczby wierszy w tabelach baz danych obu systemów, przyrost rozmiaru (rozumianego jako zajętość pamięci komputera) bazy danych In-memory jest często rzędu 10-12 razy większy od rozmiaru analogicznej bazy w systemie Oracle. Przykładowo, 18 MB danych zapisanych w tabelach klasycznego systemu Oracle pociągało za sobą zajętość nieco ponad 180 MB pamięci w przypadku bazy TimesTen. Warto przy tym podkreślić, że o ile wykorzystywane w testach polecenie:

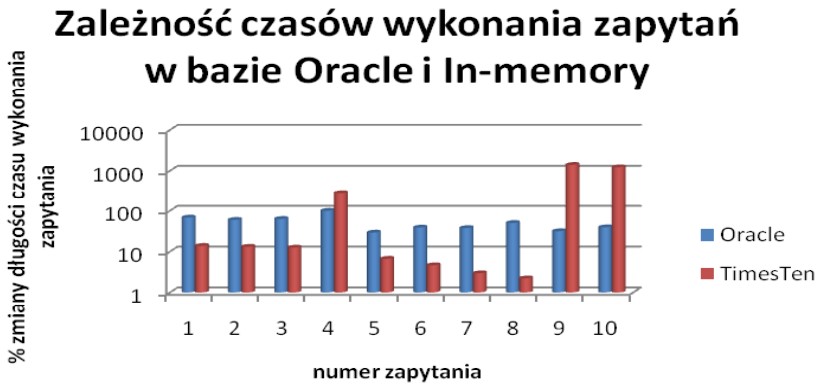
```
SELECT
   segment_name table_name, sum(bytes)/(1024*1024) table_size_meg
FROM
   user_extents
WHERE
   segment_type='TABLE'
AND
   segment_name = nazwaTabeli
GROUP  BY segment_name;
```

zwracało informację o wielkości danych zapisanych w tabelach Oracle, o tyle używana dla TimesTen komenda `DSSIZE` narzędzia `ttisql` (będącego odpowiednikiem SQL PLUS Oracle) określała rozmiar wszystkich struktur pamięciowych bazy TimesTen. Takie zestawienia porównawcze pozwoliły jednak wstępnie szacować, jaka będzie zajętość pamięci komputera po załadowaniu do niej bazy In-memory o ustalonej liczbie wierszy.

W trakcie testów nie starano się unikać pytań nieoptymalnych, „zasobożernych". Wstępnie nie zdefiniowano również żadnych indeksów. Wynikało to z chęci sprawdzenia, jak radzą sobie oba typy baz danych w sytuacji, gdy korzystają z niej użytkownicy, którzy nie posiadają zbyt dużej wiedzy z zakresu optymalizacji zapytań i sposobów podnoszenia wydajności bazy. Tak więc w zestawie tym znalazły się pytania o niskiej selektywności czy odnoszące się do wszystkich kolumn tabeli (np. `select * from …, select count(*) from …`), oraz takie, które w literaturze angielskiej zwane są „no-sargable"[1]. Przykładem tych ostatnich mogą być zapytania wykorzystujące operator `NOT EXISTS` czy `LIKE '%.<znaki>%'` .

3.2. Uzyskane rezultaty

W celu zwiększenia czytelności wykresów, osiągnięte czasy wykonania poszczególnych zapytań przeskalowano, podając odczytaną wielkość jako procent maksymalnego czasu odczytanego dla zapytań testowanych w klasycznym systemie Oracle.



Rys. 4. Porównanie szybkości przetwarzania zapytań w bazie TimesTen i klasycznej Oracle
Fig. 4. Query processing comparison in TimesTen database and Oracle conventional one

Mimo iż – zgodnie z oczekiwaniami – wyraźnie dało się zauważyć tendencję do zmniejszania się długości czasu odpowiedzi w przypadku pracy w środowisku TimesTen (średnio 8-9 razy), jednak zastanawiał bardzo duży narzut czasowy obserwowany w zapytaniach: 4, 9 i 10. Przy czym różnica, w tym przypadku na niekorzyść TimestTen, była tak duża (dla pytania 9 nawet 44 razy dłuższy czas wykonania), że właśnie te trzy zapytania zdecydowały o tym, iż sumaryczny czas wykonania pełnego zestawu pytań był ponad trzykrotnie dłuższy dla bazy pamięciowej.

Sytuacja uległa pewnej poprawie po założeniu indeksów na kolumnach biorących udział w selekcji czy złączeniu. W tym przypadku sumaryczny czas wykonania zestawu testowego był o 13% lepszy dla systemu TimesTen. Nadal jednak pytania 4 i 9 były wykonywane szybciej w tradycyjnym rozwiązaniu Oracle.

[1]Warunki zapewniające należytą wydajność podczas wyszukiwania bywają określane jako „sargable" (ang. *Search ARGument Able*).

Postanowiono dokładniej przyjrzeć się „problematycznym” zapytaniom. Odpowiednio pytanie 4 było sformułowane następująco:

```
SELECT *
FROM osoba o1, osoba o2
WHERE o1.data_ur<o2.data_ur and o2.id_osoby=10000
```

Po dokładniej analizie planów wykonania stwierdzono, iż optymalizator przy standardowych ustawieniach nie wykorzystuje dla tego pytania istniejącego indeksu zdefiniowanego dla atrybutu `id_osoby` (również po zmianie kolejności warunków we frazie `WHERE`). Po wymuszeniu bardziej optymalnego planu wykonania (tak, aby wspomniany indeks został wykorzystany) uzyskano prawie 9-krotne skrócenie czasu wykonania. Inną możliwością skrócenia czasu wykonania tego zapytania okazało się zdefiniowania go w alternatywnej formie, a mianowicie:

```
SELECT *
FROM  osoba o1
WHERE o1.data_ur<(SELECT  o2.data_ur
                  FROM osoba o2
                  WHERE o2.id_osoby=10000)
```

W tym przypadku optymalizator wykorzystał istniejący indeks i uzyskano 15-krotne skrócenie czasu w stosunku do czasu pierwotnego.

Kolejnym niewydajnym zapytaniem, które poddano dokładniejszej analizie było:

```
SELECT *
FROM okres_zatrudnienia o
WHERE
   NOT EXISTS (SELECT *
               FROM osoba o1
               WHERE o.id_osoby=o1.id_osoby)
```

Dla celów porównawczych zapisano je również w alternatywnej formie – z użyciem frazy `NOT IN`:

```
SELECT *
FROM okres_zatrudnienia o
WHERE o.id_osoby
   NOT IN (SELECT o1.id_osoby
               FROM osoba o1)
```

oraz z użyciem złączenia zewnętrznego:

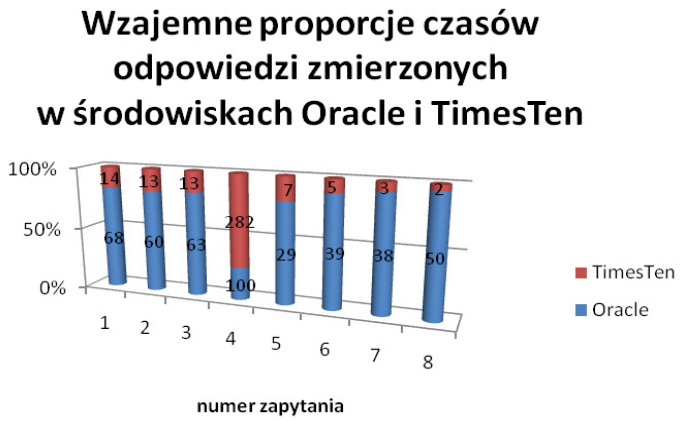
```
SELECT *
FROM okres_zatrudnienia o LEFT OUTER JOIN osoba o1 ON o.id_osoby =o1.id_osoby
WHERE o1.id_osoby IS NULL;
```

Okazało się, że w przypadku zapytania skorelowanego, plan wykonania zadania na serwerze Oracle był wydłużony o etap filtrowania obu połączonych tabel (predykat: `o.id_osoby =o1.id_osoby`), co spowodowało znaczący wzrost kosztu wykonania zapytania (1595 umownych jednostek – w porównaniu z 721 jednostkami dla złączenia zewnętrznego). Zarówno

w przypadku złączenia zewnętrznego, jak i pytania skorelowanego, łączenie odbywało się według algorytmu hash-join – co oznacza, że dla większej z obu tabel (tu: OKRES_ZATRUDNIENIA) tworzona była tabela haszowa, odczytywana zawartość tabeli mniejszej (tu: OSOBA), a w następnym kroku tabela haszowa używana była do odnalezienia wiersza w tabeli OKRES_ZATRUDNIENIA. Biorąc pod uwagę założenie, że tabela haszowa może zostać w całości utworzona w pamięci, oczywiste jest, iż łączenie hash-join będzie najbardziej efektywne.

Poprawę wydajności, spowodowaną zmianą definicji rozważanego zapytania, zaobserwowano również w bazie o konfiguracji In-memory. Użycie w zapisie operatora NOT EXISTS lub NOT IN oznaczało (dla bazy bez indeksów) czas wykonania zapytania rzędu 70 s (dla bazy bez zdefiniowanych indeksów), natomiast wymuszenie realizacji złączenia zewnętrznego wiązało się z około 200-krotnym obniżeniem tego czasu (na odpowiedź należało czekać tylko nieco ponad 0,3 s). Zastosowanie indeksów, oczywiście, symetrycznie zmniejszało oba odczyty.

Podsumowując, przeprowadzone obserwacje potwierdziły, iż w kolejnych pracach należałoby dodatkowo rozważyć konieczność strojenia zapytań SQL.



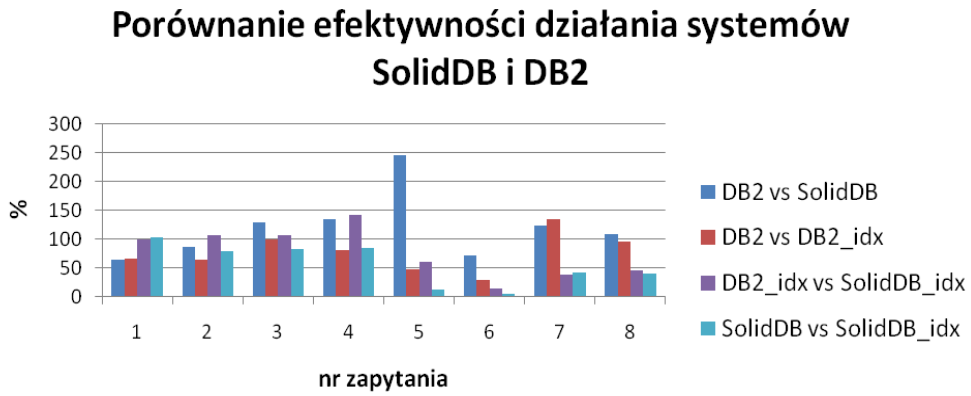
Rys. 5. Procentowy udział długości czasu odpowiedzi na zapytanie w bazach TimesTen i Oracle
Fig. 5. Percentage share of query execution times in TimesTen database and Oracle one

Eliminując z dalszych rozważań zapytania powodujące największy narzut czasowy, porównano procentowy udział odpowiadających sobie czasów wykonania kolejnych zapytań w bazie Oracle i bazie TimesTen (rys. 5), stwierdzając występowanie istotnego (bo od 4 do 22-krotnego w przypadku zapytania 8.) zysku czasowego (czyli przyspieszenia) na rzecz środowiska In-memory. Liczby umieszczone na słupkach wskazują na krotność czasu wykonania zapytania w stosunku do czasu wykonania identycznego zapytania na analogicznych tabelach o wskazanej liczbie wierszy pamięciowej bazy danych.

Takie same eksperymenty przeprowadzono dla drugiego ze wspomnianych rozwiązań – bazy DB2 i SolidDB. Część wcześniejszych spostrzeżeń potwierdziła się również i dla tej implementacji bazy In-memory (np. nieoptymalność zapisu zapytań 4, 9 i 10). Wieloaspek-

towe porównanie bazy DB2 i jej pamięciowego odpowiednika – SolidDB przedstawiono na rys. 6. Użyte na wykresie oznaczenia należy interpretować w następujący sposób:

- *DB2 vs SolidDB* wskazuje, jaki procent czasu wykonania zapytania na niepoindeksowanych danych bazy DB2 stanowi czas odczytany dla testu przeprowadzonego na danych bazy SolidDB (również bez zdefiniowanych indeksów),
- *DB2 vs DB2_idx* wskazuje, jaki procent czasu wykonania zapytania na niepoindeksowanych danych bazy DB2 stanowi czas odczytany dla testu przeprowadzonego na danych bazy z założonymi indeksami,
- *DB2_idx vs SolidDB_idx* wskazuje, jaki procent czasu wykonania zapytania na poindeksowanych danych bazy DB2 stanowi czas odczytany dla testu przeprowadzonego na poindeksowanych danych bazy SolidDB,
- *SolidDB vs SolidDB_idx* wskazuje, jaki procent czasu wykonania zapytania na niepoindeksowanych danych bazy SolidDB stanowi czas odczytany dla testu przeprowadzonego na danych bazy SolidDB z założonymi indeksami.



Rys. 6. Wieloaspektowe porównanie efektywności działania systemów SolidDB i DB2
Fig. 6. Multi-dimensional comparison between SolidDB and DB2 systems

Zgodnie z przypuszczeniami, również i w tej implementacji bazy In-memory (tj. w SolidDB) nie uzyskano jednoznacznego przyspieszenia czasu realizacji całego zestawu zapytań. W tym przypadku zarówno niezałączone na rysunku zapytania 9 i 10 (z frazą NOT EXISTS), jak i inne zapytania z klasy „no-sargable" (o numerach 4 i 5), okazały się niewydajne z punktu widzenia optymalizatora zapytań sytemu SolidDB, czego wynikiem było wyraźne wydłużenie czasu wykonania zapytania dla bazy pamięciowej. Założenie potrzebnych indeksów na odpytywanych danych praktycznie niwelowało obserwowany efekt (por. zapytanie 5), choć również i wtedy zdarzały się sytuacje, w których dyskowa konfiguracja danych była lepsza od konfiguracji pamięciowej (zapytanie 4).

W celu oceny skalowalności baz tradycyjnych i pamięciowych zbadano, o ile procent zwiększył się czas odpowiedzi na zapytanie, jeżeli dwu- i trzykrotnie zwiększono liczbę rekordów w odpytywanych tabelach (rys. 7). Zaobserwowana tendencja wyraźnie wskazała na

przewagę systemu In-memory nad bazą danych zarządzaną dyskowo (w 8 przypadkach na 10 przyrost czasu potrzebnego na uzyskanie odpowiedzi był dla bazy SolidDB znacząco niższy niż dla bazy DB2). Ta reguła nie była spełniona jedynie dla wcześniej już opisywanych jako problematyczne zapytań 4 i 9, choć procent wzrostu czasu w SolidDB był tu nieistotnie wyższy niż w DB2 (w zapytaniu 9 procent przyrostu wynosił ok. 106% w stosunku do 81% w DB2, natomiast w zapytaniu 4. różnica ta wynosiła już tylko ok. 3% na korzyść bazy DB2).

Rys. 7. Procentowy przyrost długości czasu wykonania zapytania dla dwukrotnie powiększonego zbioru danych baz SolidDB i DB2

Fig. 7. Percentage increase of query execution time for double-plus set of data in SolidDB database and DB2 one

4. Podsumowanie

W prezentowanej pracy skoncentrowano się na zagadnieniach związanych ze sposobami organizacji przestrzeni składowania danych w systemach baz danych. W modelach tych analizowano rodzaje pamięci wykorzystywanych do magazynowania danych, które następnie stały się podstawą do zróżnicowania 2 kategorii systemów zarządzania bazami danych: systemów baz pamięciowych (In-memory) i dyskowych.

Platformę testową przeprowadzonych badań stanowiły rozwiązania firm IBM oraz Oracle i ich produkty – odpowiednio – SolidDB i TimesTen, których funkcjonalność rozważano w kontekście ich dyskowych odpowiedników. M. in. przestudiowano sposób zapewnienia w nich trwałości danych, skalowalność i wydajność przetwarzania zapytań należących do różnych klas.

Eksperymenty potwierdziły, iż w większości przypadków bazy pamięciowe znacznie przewyższają pod względem wydajnościowym bazy dyskowe. Nie jest tak jednak zawsze. Bazy In-memory w swoich standardowych konfiguracjach źle radzą sobie z pytaniami nie-

optymalnymi (o małej selektywności czy przykładowo zawierającymi frazy NOT EXISTS). Konieczne jest więc poświęcenie większej uwagi ewentualnemu zapisowi zapytań w alternatywnej formie i sterowaniu pracą optymalizatora. Dopiero odpowiednie dostrojenie bazy In-memory pozwoli w pełni wykorzystać jej możliwości.

Kolejnym krokiem, jaki zamierzają uczynić autorzy w dalszej analizie rozwiązań baz pamięciowych, będzie zbadanie skuteczności rozważanych baz w konfiguracji buforowej.

BIBLIOGRAFIA

1. Dokumentacja Oracle® TimesTen In-Memory Database, adres strony internetowej (stan na rok 2010): http://www.oracle.com/technology/documentation/timesten_doc.html.
2. M. Morzy: Oracle TimesTen – cała baza danych w pamięci operacyjnej, Konferencja PLOUG, 2009.
3. J. Muszyński: Sybase: baza danych "in-memory", Computerworld, 2010.
4. J. LeFevre: TimesTen Taps Physical Memory for Performance, adres strony internetowej (stan na rok 2010): http://findarticles.com/p/articles/mi_m0FOX/is_n6_v3/ai_n27539412/
5. Ogólny zarys architektury TimesTen: Oracle Times Ten data caching, seria Oracle Tips, adres strony internetowej (stan na rok 2010): http://www.dba-oracle.com/ t_times_ten_-data_caching.htm.
6. Using Oracle In-Memory Database Cache to Accelerate the Oracle Database, Oracle Technical White Paper, July 2009.
7. Adres strony internetowej poświęconej rozwiązaniu IBM SolidDB (stan na rok 2010): http://www-01.ibm.com/software/data/soliddb/.
8. Dokumentacja IBM SolidDB In-Memory Database, adres strony internetowej (stan na rok 2010): http://www-01.ibm.com/support/docview.wss?uid=swg27014817.
9. C. Monash: "BM acquires SolidDB to compete with Oracle TimesTen, "DBMS2", 2008.
10. Raul F. Chong, Clara Liu, Sylvia F. Qi, Dwaine R. Snow: Zrozumieć DB2 ®. Nauka na przykładach, PWN, Warszawa 2006.
11. DB2 9.7 Discovery Kit, Wydawnictwo Software Press, 2009.
12. F. G. Soltis: Inside the AS/400: Second Edition, 29th Street Press; 2 Sub edition (October 1997), ISBN-13: 978-1882419661.
13. DB2 9.7 Software Developer's Journal Extra nr 35, Warszawa 2009, Sofware ISSN: 1734-7661.

Recenzent: Dr hab. inż. Tadeusz Wieczorek, prof. Pol. Śląskiej

Wpłynęło do Redakcji 31 stycznia 2010 r.

Abstract

Traditional databases are built to store data on disk. Because of the fact, that disk I/O is very expensive in terms of performance, this often makes traditional databases too slow for different systems that require real-time performance. On the contrary, In-memory databases that reside entirely in operational memory, have emerged specifically to improve this needs. So, taking into account advantages of last solution, the idea of In-memory databases was presented in this article. Especially, the IBM and Oracle systems – SolidDB and TimesTen – were analyzed. In order to estimate In-memory application cost-effectiveness in practice, they were compared to the traditional databases – DB2 and Oracle.

Several experiments were made and results definitely showed, that in most classes of queries, In-memory databases were faster than disk ones (Fig. 4, 5). They were also more scalable (Fig. 7). Unfortunately it wasn't a rule. Examined classes of queries demonstrated the existence of queries not optimal for analyzing In-memory systems (e.g.. queries 4, 9 and 10 – Fig. 4). This researches indicated the need for tuning SQL queries.

Adresy

Małgorzata BACH: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, malgorzata.bach@polsl.pl.
Adam DUSZEŃKO: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, adam.duszenko@polsl.pl.
Aleksandra WERNER: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, aleksandra.werner@polsl.pl.