Anna Kowalczyk-Niewiadomy
Wyższa Szkoła Informatyki w Łodzi
Wydział Informatyki i Zarządzania
email: anna_kowalczyk@wsinf.edu.pl

FUNCTION RESULT CACHE - POPRAWA WYDAJNOŚCI APLIKACJI PL/SQL

Streszczenie – Na przestrzeni ostatnich kilkudziesięciu lat obserwujemy intensywny rozwój w dziedzinie gromadzenia i przechowywania danych. Systemy zarządzania bazami danych stały się zasobem krytycznym wielu firm i przedsiębiorstw, które każdego dnia przetwarzają ogromne ilości danych nagromadzonych przez ostatnie lata. Wraz ze wzrostem magazynowanych danych, przeszukiwanie ich stało się niezwykle czasochłonne. Ponadto, powielanie tego samego zapytania prowadziło do wielokrotnego przeszukiwania bazy danych. Niniejszy artykuł ma na celu przedstawienie mechanizmu „Function Result Cache" zastosowanego w Oracle 11g, który umożliwi sprawną i szybką pracę z bazą danych omijając powielanie tych samych operacji.

1 Wstęp

Intensywny rozwój w dziedzinie informatyki, a co za tym idzie również dynamiczny rozwój systemów zarządzania bazami danych takich jak Oracle RDBMS spowodował ogromny postęp praktycznie w każdej dziedzinie życia. Obecnie, zaczynając od firm i instytucji związanych z bankowością, finansami, ubezpieczeniami poprzez świetnie rozwijający się handel elektroniczny, można by wymieniać kolejne dziedziny życia, w których systemy baz danych stanowią niezbędny element podstawowej infrastruktury informatycznej. Pamiętajmy jednak, iż dynamiczny rozwój tych systemów oraz gwałtowny wzrost ilości danych gromadzonych w postaci cyfrowej pociąga za sobą wysokie wymagania im stawiane. Obecnie, zaplecze sprzętowe jest problemem marginalnym, którego rozwiązanie zależy właściwie tylko od możliwości finansowych inwestora. Pozostają jednak problematyczne kwestie, jakimi są bezpieczeństwo i efektywność przeszukiwania danych. Jak wiemy, przeszukiwanie obszernych kolekcji danych jest niezwykle czasochłonne. Dotarcie do jednej informacji może pochłonąć dużo czasu i zasobów sprzętowych. Co więcej, ponawianie tego samego zapytania skutkuje wielokrotnym przeszukiwaniem całej bazy w poszukiwaniu jednej i tej samej informacji. Oracle 11g udostępnia nam nowe

możliwości dotyczące przyspieszenia pracy z danymi poprzez wprowadzenie Function Result Cache [1][2]. Niniejszy artykuł ma na celu zaprezentować efektywny sposób wykorzystania tego mechanizmu (FRC), polegającego najogólniej mówiąc na cache'owaniu wyników funkcji składowanych. Funkcjonalność ta umożliwia nam sprawną i szybką pracę z bazą danych i ominięcie powielania operacji. FRC udostępnia prosty mechanizm pozwalający na przechowywanie w pamięci wyników wykonanych wcześniej funkcji PL/SQL-owych. W rezultacie ponowne wywołanie funkcji nie wykonuje jej samej, a jedynie odwołuje się do obszaru pamięci, w którym przechowywany jest wynik pod odpowiednim kluczem haszującym, opartym o parametry wywołania. Nietrudno zauważyć, iż czas pracy znacznie się skraca, gdyż nie powielamy tych samych operacji i zapytań do bazy danych z każdym wywołaniem funkcji. W rezultacie kod odpowiednio dostosowanych funkcji wywołuje się relatywnie rzadziej, opierając się głównie na danych rezydujących w buforze i operując na danych historycznych. Niniejszy artykuł przedstawi praktyczne zastosowania mechanizmu FRC poparte odpowiednim kodem PLSQL a także wskaże ryzyka związane z użyciem tego mechanizmu.

2 Function result cache

Załóżmy, iż jesteśmy pracownikami instytucji, której zadaniem jest praca z danymi dotyczącymi tysięcy podatników. Niemal równolegle pracownicy pobierają te same informacje, wykonując funkcje oparte o zapytania na tabeli podatnicy. Jako przykład prześledźmy prostą funkcję zwracającą informacje na temat podatnika o konkretnym znanym nam identyfikatorze (Fun.1).

```
CREATE OR REPLACE FUNCTION frc_fun1(p_id podatnicy.id_podatnika%TYPE)
RETURN podatnicy%ROWTYPE
IS
l_podatnik podatnicy%ROWTYPE;
BEGIN

  SELECT * INTO l_podatnik FROM podatnicy WHERE id_podatnika = p_id;
  DBMS_OUTPUT.PUT_LINE('Wykonano frc_fun1 dla podatnika o id:'||p_id);
  RETURN l_podatnik;

EXCEPTION
   WHEN NO_DATA_FOUND THEN
     RETURN l_podatnik;
    END;
```

Fun.1. Funkcja zwracająca dane podatnika.

Dane w tabeli podatnicy nie ulegają częstym modyfikacjom, a nam zależy głównie na szybkim przeszukiwaniu tej tabeli. Funkcjonalność Function Result Cache (FRC) umożliwia sprawną i szybką pracę z bazą danych. FRC udostępnia prosty mechanizm pozwalający na przechowywanie w pamięci wyników wykonanych wcześniej funkcji PL/SQLowych. W rezultacie ponowne wywołanie funkcji nie wykonuje samej funkcji, a jedynie odwołuje się do obszaru pamięci, w którym przechowywany jest wynik pod odpowiednim kluczem haszującym opartym o parametry wywołania. Nietrudno zauważyć, iż czas pracy znacznie się skraca, gdyż nie powielamy tych samych operacji i zapytań do bazy danych z każdym wywołaniem funkcji. W rezultacie kod aplikacji wywołuje się relatywnie rzadko opierając się głównie na danych w buforze i operując na danych statycznych. W praktyce wygląda to następująco (Fun.2).

```
CREATE OR REPLACE FUNCTION frc_fun1(p_id podatnicy.id_podatnika%TYPE)
RETURN podatnicy%ROWTYPE
RESULT_CACHE RELIES_ON (podatnicy)
IS
l_podatnik podatnicy%ROWTYPE;
BEGIN

  SELECT * INTO l_podatnik FROM podatnicy WHERE id_podatnika = p_id;
  DBMS_OUTPUT.PUT_LINE('Wykonano frc_fun1 dla podatnika o id:'||p_id);
  RETURN l_podatnik;

EXCEPTION
   WHEN NO_DATA_FOUND THEN
     RETURN l_podatnik;
    END;
```

Fun.2. Zastosowanie klauzuli RESULT_CACHE.

Klauzula RESULT_CACHE daje kompilatorowi wskazanie, aby przechowywał w wewnętrznej pamięci, każdego zwróconego przez funkcję rekordu charakteryzowanego poprzez ID podatnika. Jeśli w trakcie pracy bazy funkcja zostanie ponownie wykonana dla identycznego ID silnik nie wykona poleceń wewnątrz całej funkcji zawierającego zapytanie wybierające, lecz odwoła się do przechowywanego w pamięci wyniku. Ponieważ działanie z cachem, który nie jest oparty o dane statyczne wymaga odświeżania baza Oracle 11g Release 1 wymusza na nas wyspecyfikowanie nazwy tabel zależnych poleceniem RELIES_ON. Poprzez wskazanie RELIES_ON(podatnicy) informujemy system, iż każda zmiana na tabeli podatnicy musi skutkować odświeżeniem cache'u. Wersja Oracle 11g Release 2 już na etapie kompilacji sprawdza zależności i sama

automatycznie dobiera powiązane tabele, co zwalnia nas z obowiązku ustawiania wcześniej wspomnianej klauzuli. Nasuwa się od razu prosty wniosek, że cache warto zakładać tylko na funkcjach odwołujących się do rzadko zmienianych tabel. Kontynuując, następne wywołanie funkcji frc_fun1 wykona zapytanie wybierające zwracając najnowsze dane z tabeli. Warto zauważyć fakt, iż cache jest częścią obszaru systemowego SGA (System Global Area) zatem jest on współdzielony. Jest to wielka zaleta tego rozwiązania, gdyż jego zawartość jest dostępna dla wszystkich otwartych sesji dla danej instancji serwera. Przed wprowadzeniem FRC w Oracle 11g możliwe było cache'owanie na poziomie pakietów. Jednak takie podejście pozwalało na korzystanie z cache tylko na poziomie pojedynczej sesji, a dane były przechowywane w PGA (Process Global Area). W momencie gdy liczba równolegle podłączonych użytkowników rosła obciążenie pamięci znacząco się zwiększało. Powracając do FRC spróbujmy kilkukrotnie wywołać funkcję w poszukiwaniu podatnika o zadanym identyfikatorze (Fun.3).

```
DECLARE
l_podatnik podatnicy%ROWTYPE;
BEGIN
  l_podatnik:= frc_fun1(722);
DBMS_OUTPUT.PUT_LINE('Podatnik :'||l_podatnik.nazwisko_podatnika||'
'||l_podatnik.imie_podatnika||' NIP:'||l_podatnik.nip);
    END;
```

Fun.3. Wywołanie funkcji cach'ującej.

Pierwsze wywołanie skutkuje następującym komunikatem:

```
Wykonano frc_fun1 dla podatnika o id:722
Podatnik :Pol Adam NIP:1122333444
```

Każde kolejne natomiast pokazuje, iż polecenia wewnątrz ciała funkcji frc_fun1 nie zostały uruchomione, a wynik otrzymaliśmy na podstawie wcześniejszego wyszukiwania

```
Podatnik :Pol Adam NIP: 1122333444
```

Wszelkie zmiany na tabeli wyspecyfikowanej w klauzuli RELIES_ON wymagają odświeżenia cache'u, zatem poniższe polecenie modyfikujące wymusi wykonanie poleceń wewnątrz ciała funkcji przy jej ponownym wywołaniu.

```
UPDATE podatnicy SET ulica='Nowa 10' where id_podatnika=722;
```

Podsumowując, Fuction Result Cache znacznie poprawia efektywność otrzymywania rezultatów z funkcji, które wywoływane są wielokrotnie z identycznymi parametrami poprzez pominięcie przetwarzania wewnątrz bloku wykonywalnego. FRC jest znacznie szybszy od wielokrotnych powtórzeń wymagających każdorazowego przeszukiwania całej kolekcji danych. Wyniki przechowywane są w obszarze SGA, co pozwala na współdzielenie między sesjami. Ponadto, wszelkie zmiany na tabeli skutkują odświeżeniem zawartości cache'u. Najprostszym do zrozumienia i najtrafniejszym miejscem zastosowania cache są funkcje, które zwracają np. parametry konfiguracyjne aplikacji. Podsumowując Function Result Cache bez wątpienia jest niezwykle przydatną funkcjonalnością, jeśli chodzi o optymalizację czasów wywołań funkcji. W dalszej części spróbujemy dokładniej przyjrzeć się jak FRC działa w połączeniu z Oracle Virtual Private Database [3][4].

3 Virtual Private Database i function result cache

Oracle Virtual Private Database jest efektywnym narzędziem pozwalającym na definiowanie zasad polityki bezpieczeństwa, dostępu i operacji na danych. System bazodanowy automatycznie ogranicza dostęp do danych, dla konkretnego użytkownika niewidoczną klauzulą WHERE. W skutek tego, dwóch użytkowników wykonujących to samo zapytanie (np. SELECT * FROM podatnicy) otrzyma zupełnie odmienne wyniki. Ograniczenia nałożone na użytkownika dotyczące dostępu do danych w obrębie funkcjonalności VPD są dla niego niewidoczne. Zgodnie z wcześniejszym opisem cache'owanie wyników funkcji, w sposób znaczący zmniejsza czas uzyskania odpowiedzi, poprzez odwoływanie się do rezultatów przechowywanych w pamięci (SGA). Ze względu na fakt, iż pamięć ta jest współdzielona między wszystkie otwarte sesje w obrębie instancji, możemy się domyślać, iż takie rozwiązanie w połączeniu z Oracle VPD może przysporzyć pewnych niespodziewanych kłopotów. Przyjrzyjmy się zatem jakie problemy mogą się pojawić, gdy nieumiejętnie wykorzystamy oba mechanizmy i jak możemy im przeciwdziałać. Właścicielem schematu i danych w tabeli podatnicy jest użytkownik o nazwie admin, który udostępnia dane użytkownikom pracownik1, pracownik2. Tabela podatnicy oprócz standardowych pól informacyjnych zawiera kolumnę *utworzyl*, zawierającą informację na temat pracownika, który dodał rekord. Trigger podatnicy_audit zadba o poprawność informacji w tej kolumnie.

```
--z poziomu ADMIN
CREATE TABLE podatnicy
(
```

```
id_podatnika NUMBER PRIMARY KEY,
imie_podatnika VARCHAR2(50),
nazwisko_podatnika VARCHAR2(50),
pesel_podatnika NUMBER,
nip NUMBER,
ulica VARCHAR2(200),
/*pozostałe kolumny*/
utworzyl   VARCHAR2 (30)
);
/
CREATE OR REPLACE TRIGGER podatnicy_audit BEFORE INSERT ON podatnicy
FOR EACH ROW
DECLARE
BEGIN
   :new.utworzyl:= USER;
END;
/
GRANT ALL ON podatnicy TO pracownik1;
GRANT ALL ON podatnicy TO pracownik2;
/
CREATE OR REPLACE FUNCTION frc_fun1(p_id podatnicy.id_podatnika%TYPE)
RETURN podatnicy%ROWTYPE RESULT_CACHE RELIES_ON (podatnicy)
IS
l_podatnik podatnicy%ROWTYPE;
BEGIN
  SELECT * INTO l_podatnik FROM podatnicy WHERE id_podatnika = p_id;
  DBMS_OUTPUT.PUT_LINE('Wykonano frc_fun1 dla podatnika o id:'||p_id);
  RETURN l_podatnik;
END;
/
GRANT EXECUTE ON frc_fun1 TO pracownik1
    GRANT EXECUTE ON frc_fun1 TO pracownik2
```

Fun.4. Przykład wykorzystania VPD i FRC.

Następnie każdy z użytkowników dodają po dwa rekordy do tabeli podatnicy (Fun.5).

```
--z poziomu PRACOWNIK1
BEGIN
INSERT INTO admin.podatnicy(id_podatnika,nazwisko_podatnika,imie_podatnika)
VALUES(1,'Jan_1','Kowalski_1');
  INSERT INTO admin.podatnicy(id_podatnika,nazwisko_podatnika,imie_podatnika)
VALUES(2,'Maria_1','Nowacka_1');
  COMMIT;
END;
--z poziomu PRACOWNIK2
```

```
BEGIN
  INSERT INTO admin.podatnicy(id_podatnika,nazwisko_podatnika,imie_podatnika)
VALUES(3,'Jan_2','Kowalski_2');
  INSERT INTO admin.podatnicy(id_podatnika,nazwisko_podatnika,imie_podatnika)
VALUES(4,'Maria_2','Nowacka_2');
  COMMIT;
    END;
```

Fun.5. Przykład dodania rekordów przez użytkowników.

Aktualnie, każdy z pracowników ma dostęp do wszystkich rekordów w tabeli podatnicy. Ponadto uruchomienie poniżej zamieszczonej procedury anonimowej przez pracownika1 umieści wynik funkcji w pamięci, dzięki czemu pracownik2 natychmiast otrzymuje odpowiedź bez uruchamiania funkcji frc_fun1.

```
DECLARE
l_podatnik admin.podatnicy%ROWTYPE;
BEGIN
  l_podatnik:= admin.frc_fun1(1);
  DBMS_OUTPUT.PUT_LINE('Podatnik :'||l_podatnik.nazwisko_podatnika);
  l_podatnik:= admin.frc_fun1(4);
  DBMS_OUTPUT.PUT_LINE('Podatnik :'||l_podatnik.nazwisko_podatnika);
    END;
```

Fun.6. Niepożądane skutki wykorzystania FRC i VPD.

Następny krok to zdefiniowanie reguł Oracle VPD. Zależy nam na tym, aby każdy z pracowników widział rekordy dodane wyłącznie przez siebie. W tym celu, tworzymy funkcję zarządzającą oracle_vpd w której definiujemy warunki dodane w klauzuli WHERE ograniczające dostęp do danych.

```
CREATE OR REPLACE FUNCTION fun_vpd_plicy (schema_in VARCHAR2,
name_in VARCHAR2)
RETURN VARCHAR2 IS
BEGIN
    RETURN 'utworzyl = '' || USER||'';
    END;
```

Fun.7. Funkcja zwracająca informacje na temat USER.

Następnie wywołujemy procedurę ADD_POLICY (Fun.8) z pakietu DBMS_RLS (czyli Row Level Security), aby zastosować regułę dane_indywidualne do tabeli podatnicy.

```
BEGIN
```

```
DBMS_RLS.ADD_POLICY (object_schema     => 'ADMIN'
          , object_name       => 'podatnicy'
          , policy_name       => 'dane_indywidualne'
          , function_schema   => 'ADMIN'
          , policy_function   =>   'fun_vpd_plicy'
          , statement_types   => 'SELECT'
          , update_check      => TRUE);
   END;
```

Fun.8. Wywołanie procedury ADD_POLICY.

Na ten moment pracownik2 nie powinien widzieć danych wstawionych przez pracownika1, co w prosty sposób możemy sprawdzić za pomocą polecenia wybierającego:

```
SELECT * FROM admin.podatnicy;
```

Jeśli jednak zdecydujemy się na wyszukanie informacji za pomocą funkcji cache'ującej frc_fun1, okazuje się, że możemy w prosty sposób dostać się do danych, do których nie powinniśmy mieć dostępu. Dlaczego tak się dzieje? Otóż, zauważmy iż podczas wywoływania po raz kolejny funkcji frc_fun1 z tym samym argumentem, nie wyświetla się tekst: Wykonano frc_fun1 dla podatnika o id, gdyż rezultat znajduje się w cache'u. Zatem zapytanie wybierające, którego dotyczy reguła dane_indywidualne nie jest uruchamiane. W efekcie otrzymujemy informacje, które powinny być dla nas w niewidoczny sposób zablokowane.

4 Jak zapewnić bezpieczeństwo korzystając z Oracle VPD i FRC?

Na pierwszy rzut oka można pokusić się o stwierdzenie, że użycie tych dwóch funkcjonalności jednocześnie nie ma sensu. Rzeczywiście, naruszenie reguł Oracle VPD w tym przypadku okazało się bardzo niepożądane w skutkach i mogło w skrajnym przypadku doprowadzić do wycieku istotnych informacji. Niemniej jednak, takie podejście jest przesadzone, gdyż istnieje możliwość pogodzenia dwóch technologii i uczynienia ich kompatybilnymi. Musimy jednak dokonać pewnych zmian w naszym kodzie. W tym konkretnym przypadku problem tkwi w funkcji cache'ującej, która wykonuje polecenie niezależnie od użytkownika, który ją uruchamia. Ze względu na fakt, iż pamięć jest współdzielona między sesjami, dostęp do danych nie jest ograniczony. Spróbujmy zatem wydzielić osobny cache dla każdego użytkownika, dodając do listy parametrów funkcji cache'ującej jego nazwę (Fun.9).

```
CREATE OR REPLACE FUNCTION priv_frc_fun1
                    (p_id podatnicy.id_podatnika%TYPE,
                     p_user VARCHAR2 := USER)
RETURN podatnicy%ROWTYPE RESULT_CACHE RELIES_ON (podatnicy)
IS
  l_podatnik podatnicy%ROWTYPE;
BEGIN
  SELECT * INTO l_podatnik FROM podatnicy WHERE id_podatnika = p_id;
  DBMS_OUTPUT.PUT_LINE('Wykonano frc_fun1 dla podatnika o id:'||p_id);
  RETURN l_podatnik ;

EXCEPTION
WHEN NO_DATA_FOUND THEN
  DBMS_OUTPUT.PUT_LINE('Brak danych dla podatnika o id:'||p_id);
  RETURN l_podatnik;
END;
/
CREATE OR REPLACE FUNCTION frc_fun1 (p_id podatnicy.id_podatnika%TYPE)
   RETURN podatnicy%ROWTYPE
IS
BEGIN
   return priv_frc_fun1(p_id, USER);
    END;
```

Fun.9. Wydzielenie cache dla każdego użytkownika.

Powyższy kod pozwala na zachowanie reguł bezpieczeństwa Oracle VPD przy jednoczesnym wykorzystaniu funkcji cache'ującej, zapewniającej szybki dostęp do danych. Dzięki dodaniu do listy argumentów funkcji frc_fun1 nazwy użytkownika, zapewniliśmy sobie osobny cache dla każdego użytkownika, dzięki czemu pracownik2 nie ma dostępu do danych pracownika1. Należy zwrócić szczególną uwagę na dostęp do funkcji. Optymalnym rozwiązaniem jest stworzenie pakietu, w którym funkcja priv_frc_fun1 jest funkcją prywatną przez co nie ma możliwości ręcznego dodania nazwy użytkownika. Innym sposobem na obejście tego problemu jest stworzenie funkcji priv_frc_fun1 w schemacie admin i nie nadanie uprawnień do niej użytkownikom. Jest to jednak niebezpieczne i trudniejsze w utrzymaniu, gdyż można pogubić się w uprawnieniach i omyłkowo dodać dostęp do niepożądanej funkcji.

```
DECLARE
l_podatnik admin.podatnicy%ROWTYPE;
BEGIN
  l_podatnik:= admin.frc_fun1(1);
  DBMS_OUTPUT.PUT_LINE('Podatnik :'||l_podatnik.nazwisko_podatnika);
  l_podatnik:= admin.frc_fun1(4);
```

```
 DBMS_OUTPUT.PUT_LINE('Podatnik :'||l_podatnik.nazwisko_podatnika);
END;

ODP.:
Brak danych dla podatnika o id:1
Wykonano frc_fun1 dla podatnika o id:4
    Podatnik :Maria_2
```

Powyższy kod jest bardzo uproszczonym przykładem wykorzystania polityk VPD. W rzeczywistych systemach wprowadzane polityki dostępu do danych oparte o VPD, są zdecydowanie bardziej skomplikowane i w znacznym stopniu zależne od dynamicznych uprawnień użytkownika oraz parametrów systemowych. Najczęściej te uprawnienia trzymane są w bazie danych i co jest ważne także w celu poprawienia wydajności zwracane poprzez funkcje cache'owane.

Dodatkowo aktualnie niewiele aplikacji korzystających z tak zaawansowanych mechanizmów jak VPD i FRC wykorzystuje tryb autentykacji i autoryzacji oparty o wielu użytkowników bazodanowych. Najczęściej aplikacje korzystają z connection poolingu. Otwarcie połączenia do bazy wykonywane jest np. poprzez datasource serwera aplikacyjnego, w którym wskazany jest jeden użytkownik bazodanowy. W takim przypadku korzystanie z USER zawsze zwróci nam takiego samego użytkownika. Istnieje proste i skuteczne rozwiązanie tego problemu. Baza danych Oracle udostępnia mechanizm obsługi zmiennych wskazujących na kontekst sesji, w której pracuje użytkownik. Najczęściej kontekst taki jest ustawiany w trigerze logowania lub poprzez funkcję/procedurę składowaną uruchamianą podczas operacji autentykacji/autoryzacji użytkownika. Dostęp do kontekstu sesji można uzyskać poprzez stosowanie funkcji SYS_CONTEXT z odpowiednim parametrem mówiącym o sprawdzanej zmiennej. Jeśli identyfikator klienta jest niewystarczający można również skorzystać z innych parametrów kontekstu takich jak chociażby host ('HOST'), adres IP ('IP_ADDRESS') lub identyfikator instancji ('INSTANCE').

5 Podsumowanie

Podsumowując mechanizm cache'owania wyników danych na podstawie parametrów wejściowych funkcji, czyli function result cache jest doskonałym mechanizmem, który przy niewielkim nakładzie pracy może zostać wykorzystany do poprawy wydajności aplikacji. Mechanizm ten jest wyjątkowo prosty w implementacji szczególnie w Oracle Database 11g Release 2, gdyż nie wymaga wskazywania tabel wyzwalających odświeżanie cache. Oracle Virtual Private Database czyli mechanizm transparentnego zarządzania dostępem do danych na

poziomie wierszy, w prosty sposób pozwala na filtrowanie danych udostępnianych użytkownikom. Dzięki elastycznemu mechanizmowi definiowania polityk dostępu poprzez pakiet DBMS_RLS w prosty sposób możemy „w przybliżeniu" z tabel zrobić modyfikowalne dynamiczne widoki. Oba te mechanizmy implementowane niezależnie działają bez zarzutów. Pewne ryzyka pojawiają się w momencie równoczesnego stosowania VPD oraz FRC. Jednak mimo kilku potencjalnych zagrożeń, wykorzystanie obu mechanizmów jednocześnie, jest możliwe i jak najbardziej wskazane. Co więcej rozsądne korzystanie z VPD i FRC pozwala zwiększyć wydajność i bezpieczeństwo aplikacji bazodanowej.

Literatura

[1] http://docs.oracle.com/cd/B28359_01/appdev.111/b28370/subprograms.htm

[2] http://www.oracle-developer.net/display.php?id=504

[3] http://docs.oracle.com/cd/B28359_01/network.111/b28531/vpd.htm#DBSEG007

[4] http://www.adp-gmbh.ch/ora/security/vpd/

FUNCTION RESULT CACHE - PL/SQL PERFORMANCE IMPROVEMENT

Summary - This paper presents a novel idea of using Function Result Cache together with Virtual Private Database (VPD) to optimize performance of data retrieval from Oracle database. The Oracle VPD defines default table access policies which transparently for users restrict data on table WHERE clause level. Such functionality implies additional restrictions to caching functionality, what can be overlooked by PL/SQL developers. Reading this paper give you important instructions how to avoid such issues.