STUDIA INFORMATICA 2009
Volume 30 Number 2B (84)

Dariusz KARPISZ
Politechnika Krakowska, Instytut Informatyki Stosowanej

IMPLEMENTACJA METOD DO MONITORINGU NIEUŻYWANYCH OBIEKTÓW DLA SERWERA ORACLE

Streszczenie. Zagadnienie utrzymania efektywnej pracy systemów baz danych jest niezwykle rozległe. Jednym z pojawiających się problemów jest identyfikacja nieużywanych obiektów, które w perspektywie dłuższego okresu czasu mogą negatywnie wpływać na wydajność systemu. Proponowane w opracowaniu rozwiązanie bazujące na dedykowanym pakiecie PL/SQL dla systemu Oracle 10g może wspomagać administratorów w długookresowej praktyce konserwacji baz danych przez wskazanie nieużywanych obiektów.

Słowa kluczowe: audyt, monitoring, nieużywane obiekty, Oracle

IMPLEMENTATION OF METHODS FOR MONITORING OF UNUSED OBJECTS FOR ORACLE SERVER

Summary. The problem of maintain of database systems is very complex. One of the most frequent appearing problems is unused object identification, what in long term of time may in negative way affect on system productivity. Proposed in this paper solution based on P/SQL for Oracle 10g may supporting DBA maintains database systems.

Keywords: database audit, monitoring, unused objects, Oracle

1. Wprowadzenie

Zagadnienie zapewnienia efektywnej pracy systemów baz danych może być analizowane dla różnych etapów cyklu życia projektów informatycznych. Zaniedbania w procesie projektowym, a następnie na etapie wdrożenia i ewaluacji systemu skutkują pozostawieniem wielu obiektów, które w docelowym systemie mogą być zbędne. Każdy taki obiekt w negatywny

sposób wpływa na efektywność całego systemu [2, 3]. W dalszej części niniejszego artykułu zostaną szczegółowo omówione niektóre z aspektów monitorowania nieużywanych obiektów.

1.1. Ogólne zagadnienia utrzymania złożonych systemów baz danych

Utrzymanie dużych systemów baz danych, podlegających ciągłej ewaluacji, jest zagadnieniem złożonym, wymagającym stosowania właściwych procedur postępowania. Należy zauważyć, że wiele systemów baz danych, w szczególności o krytycznym znaczeniu dla jednostek je wykorzystujących, jest utrzymywanych przynajmniej w dwóch egzemplarzach. Podstawę systemu stanowi baza produkcyjna. Kolejne egzemplarze baz utrzymywane są w celach testowych i/lub deweloperskich [2]. Optymalnym rozwiązaniem pod względem elastyczności użytkowania i bezpieczeństwa [6] jest utworzenie trzech egzemplarzy:

- bazy produkcyjnej,
- bazy testowej, stanowiącej odzwierciedlenie bazy produkcyjnej (pod względem schematów i częściowo pod względem składowanych danych),
- bazy deweloperskiej, służącej tworzeniu nowych aplikacji i rozwiązań po stronie silnika DBMS (*Database Management System*).

Należy zauważyć, że baza testowa powinna jak najdokładniej odzwierciedlać bazę produkcyjną pod względem schematu logicznego i fizycznego ulokowania obiektów. Podobne wymagania sugeruje się, jeśli chodzi o zgodność danych. Często spełnienie takich wymogów jest jednak niemożliwe, np. w systemach klasy OLTP (*Online Transaction Processing*), gdzie wykorzystywane są zaawansowane rozwiązania partycjonowania i klastrowania ściśle dostosowane do posiadanego systemu sprzętowego. Baza testowa jest wygodnym narzędziem do testowania nowych aplikacji przed wdrożeniem do bazy produkcyjnej. Baza deweloperska jest swoistym poligonem dla nowo tworzonych aplikacji i notowany jest dla niej najwyższy poziom niezgodności z bazą produkcyjną.

Problem utrzymania systemów baz danych jest oczywiście zagadnieniem niezwykle rozległym skupiającym zagadnienia z zakresu strojenia, replikacji, wysokiej dostępności oraz wielu innych dziedzin (np. [3, 4]). W pracy zostanie przedstawione jedno z często pomijanych zagadnień, które wydaje się być niezwykle istotne w systemie dynamicznym, podlegającym ciągłej ewaluacji.

1.2. Problem niezidentyfikowanych obiektów

Niezależnie od wybranego rozwiązania działająca baza danych po pewnym okresie czasu zaczyna zapełniać się nowymi obiektami. Taki trend możemy zaobserwować szczególnie monitorując serwer przez dłuższy okres czasu (kwartał – rok). Wiele z obserwowanych

obiektów odgrywa istotną rolę w procesie składowania i przetwarzania w produkcyjnej bazie danych. W zakresie zgodności z bazą testową/deweloperską wymagana jest dla nich pełna replikacja na poziomie DDL (*Data Definition Language*). Niestety, istnieją również takie obiekty, dla których trudno odszukać miarodajną dokumentację lub informacja o ich znaczeniu w ogóle nie istnieje. Jest to związane ze stosowaniem nie do końca przemyślanych procedur wdrożenia dodatkowych elementów schematu DDL z bazy deweloperskiej do bazy testowej, a później produkcyjnej. Z biegiem czasu liczba takich „nierozpoznanych" obiektów rośnie, a rozmiar danych przez nie zajmowany zaczyna być znaczący z punktu widzenia całości systemu baz danych. W przypadku trójstopniowego modelu rozwoju baz danych „niezidentyfikowane" obiekty można próbować eliminować na zasadzie porównania kolejnych wersji schematu na bazie produkcyjnej, testowej i deweloperskiej. Nie zawsze jest to jednak możliwe, w szczególności jeśli wszystkie działania związane z produkcją oprogramowania prowadzone są na jednym egzemplarzu bazy, takie obiekty mogą niekorzystnie wpływać na wydajność bazy danych i wykorzystanie przestrzeni dyskowej. Taka sytuacja występuje w przypadku mniejszych systemów, kiedy inwestor nie zamierza kupować kosztownych licencji na oprogramowanie DBMS.

Niezidentyfikowane obiekty powstają w różnych okolicznościach i mogą funkcjonować jako:

a) obiekty tymczasowe,
b) obiekty używane przez starsze wersje oprogramowania warstwy aplikacji,
c) obiekty używane przez oprogramowanie wycofane z użycia lub odłączone od aktualnej bazy produkcyjnej,
d) obiekty utworzone przez użytkowników i obiekty automatycznie tworzone przez aplikacje (dynamiczne wywołania DDL).

Usunięcie niezidentyfikowanych obiektów jest procesem trudnym. Jeśli zostanie stwierdzone, że nikt nie jest w stanie określić, do czego dany obiekt służy, nie wynika wprost, że obiekt ten nie jest wykorzystywany i może być usunięty. Istnieje wiele możliwości wykorzystania takiego obiektu, np. obiekt może być wykorzystywany jedynie okresowo w celu buforowania pewnych danych [4, 5]. Być może nie można go usunąć ze względu na konieczność spełnienia więzów integralności z innymi obiektami schematu. Z kolei, brak wprowadzonych ograniczeń integralności dla danego obiektu nie oznacza braku zależności z innymi obiektami (przez relacje sterowane po stronie aplikacji wyższej warstwy).

Dalsze rozważania przedmiotowego zagadnienia zostaną przedstawione na przykładzie mechanizmu audytu dostępnego w serwerze baz danych Oracle 10g.

2. Audyt schematu bazy danych

Audyt schematu bazy danych jest mechanizmem służącym do monitoringu i zapisu operacji wykonywanych na wskazanych obiektach bazy danych oraz operacji wykonywanych przez konkretnego użytkownika [6]. Istnieją mechanizmy audytu wbudowane bezpośrednio w bazę danych (kontrola z poziomu DBMS), aplikacje zewnętrzne, najczęściej stowarzyszone z edytorami-systemami dla administratorów baz danych (np. przy użyciu systemu *TOAD – Tool for Application Developers*) oraz specjalizowane rozbudowane rozwiązania dla dużych systemów (np. *Oracle Audit Vault*).

Przed próbą wykorzystania rozwiązań zewnętrznych warto zastanowić się nad wykorzystaniem możliwości użytkowanego systemu DBMS. Serwer Oracle 10g oferuje mechanizm audytu z możliwością śledzenia obiektów różnego typu. Dostęp do informacji jest możliwy dzięki wykorzystaniu klasycznych perspektyw słownika danych oraz dynamicznych perspektyw wydajnościowych.

Audyt można prowadzić z różnym poziomem śledzenia [5]:

- na poziomie polecenia (ang. Statement auditing),
- na poziomie przywileju (ang. Privilege auditing),
- na poziomie obiektu (ang. Schema object auditing),
- na poziomie zawartości (ang. Fine-grained auditing).

Dane dotyczące audytu mogą być zapisane w tabeli słownika *SYS.AUD$* lub do wskazanego pliku (np. w standardzie *XML*). Z tabeli *SYS.AUD$* na temat audytowanego obiektu/użytkownika można odczytać między innymi takie informacje, jak:

- nazwa użytkownika,
- identyfikator sesji (*SID*),
- rodzaj przeprowadzonej operacji,
- czas przeprowadzenia operacji,
- użyte przywileje systemowe.

Mechanizm audytu na poziomie obiektu może w wyniku zapisywać dane dotyczące dostępu do obiektu przez konkretne polecenia *SQL*. Możliwe jest monitorowanie poleceń DML (*Data Manipulation Language*), poleceń modelu uprawnień *GRANT*/*REVOKE* oraz zapytań. Audyt można przeprowadzić jako *AUDIT BY SESSION*, *AUDIT BY ACCESS* lub *AUDIT BY USER*. Audyt zostaje włączony w zależności od parametru AUDIT TRAIL znajdującego się w pliku startowym *init.ora*. Dla instancji serwera baz danych możliwa jest zmiana tego parametru przez nadpisanie jego wartości w pliku SPFILE (ang. *Server Parameter File*). Wymagane jest wówczas zamknięcie instancji. Weryfikację wartości powyższego parametru umożliwia następujące zapytanie [1]:

```
select NAME, VALUE  from V$PPARAMETER where upper(NAME)='audit_trail';
```

3. Narzędzie audytu i identyfikacji nieużywanych obiektów

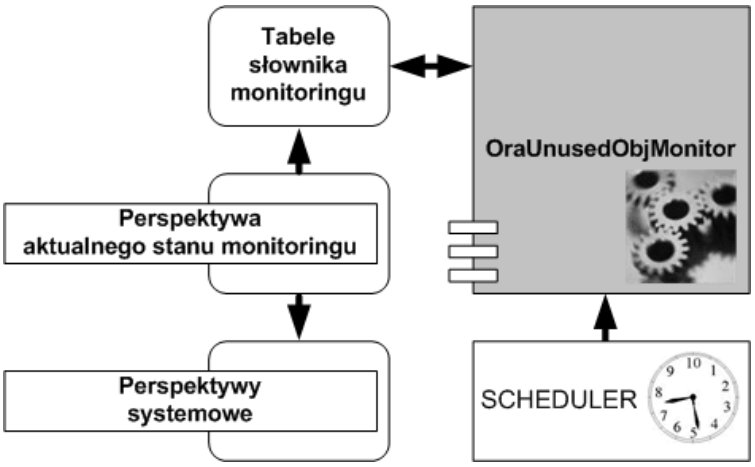
Istnieje wiele narzędzi służących do audytu serwerów baz danych. Najczęściej producenci systemów zarządzania bazami danych dołączają do nich aplikacje do monitoringu serwera i poszczególnych instancji [5]. Niestety, w domenie sprawdzania aktywności wybranych grup obiektów bazy danych konieczne jest korzystanie z samodzielnie przygotowanych skryptów SQL. Dla użytkowników serwerów Oracle udostępniono odpowiednie pakiety i perspektywy słownika danych [7], dzięki którym możliwa była przedstawiona dalej implementacja narzędzia do monitorowania nieużywanych obiektów.

3.1. Koncepcja systemu

Oferowany dla serwera Oracle 10g mechanizm audytu może być wykorzystany do identyfikacji nieużywanych obiektów bazy danych, jednak nie powinno się prowadzić audytu dla wszystkich obiektów, a następnie analizować wyników monitoringu. Taka operacja wymagałaby czasochłonnych czynności raportowych w ramach opracowania wyników, co powodowałoby zauważalne obciążenie systemu baz danych. Również sam mechanizm audytu, ze względu na konieczność sterowania przez DBMS dodatkowymi czynnościami, obciąża system informatyczny. Proponowanym rozwiązaniem powyższych problemów jest początkowe włączenie audytu dla wszystkich obiektów (z wyłączeniem niektórych z nich), a następnie stopniowe wyłączanie audytu dla tych, dla których stwierdzono aktywność.

Do wykonania takiego zadania potrzebny jest mechanizm, który co pewien czas sprawdzałby, które obiekty zostały dotychczas użyte i wyłączał dla nich audyt. Takie podejście nie powoduje odczuwalnego spadku wydajności, gdyż obiekty intensywnie używane będą wyłączone z mechanizmu audytu jako pierwsze. Koncepcję kompletnego systemu do realizacji powyższych zadań przedstawiono na rys. 1. System *OraUnusedObjMonitor* zostanie omówiony w dalszej części niniejszej pracy.

System Oracle umożliwia definiowanie zadań wykonywanych cyklicznie dzięki funkcjonalności dostępnej w pakiecie *DBMS_SCHEDULER* [7].



Rys. 1. Koncepcja systemu identyfikacji nieużywanych obiektów
Fig. 1. Idea of unused objects identification system

3.2. Cykliczne zadanie monitorujące

W ramach systemu audytu i identyfikacji nieużywanych obiektów wykorzystano pakiet *DBMS_SCHEDULER*. Jest to nowa propozycja dostępna w wersji Oracle 10g, zastępująca pakiet *DBMS_JOB* [7]. Pakiet *DBMS_SCHEDULER* ma bardzo duże możliwości, jednak w pracy zostanie wykorzystana jedynie podstawowa jego funkcjonalność dotycząca zadań cyklicznych.

DBMS_SCHEDULER jest pakietem wywoływanym z poziomu języka *PL/SQL* i pozwala na określenie procedury prostego kodu lub obiektu programu utworzonego na poziomie pakietu. Uruchamianie zadania następuje z częstotliwością określoną w wywołaniu procedury *DBMS_SCHEDULER.CREATE_JOB* lub z częstotliwością utworzonego wcześniej obiektu schedulera. Utworzenie zadania i późniejsze jego uruchomienie powoduje, że procedura, program dla *DBMS_SCHEDULER* lub kod *inline* jest wykonywany zgodnie z określonym interwałem czasowym. Do monitorowania wykonania zadań cyklicznych można użyć dostępnych perspektyw słownika danych [5]. Przykładem takiej perspektywy jest *DBA_SCHEDULER_JOB_RUN_DETAILS*.

Definicję zadania cyklicznego dla proponowanego systemu przedstawia poniższy kod.

```
dbms_scheduler.create_job(
          job_name                => 'orauom_1',
          job_type                => 'stored_procedure',
          job_action              => 'OraUnusedObjMonitor.pUpdate_unused',
          start_date              => sysdate,
          repeat_interval    => 'freq=minutely;interval=' || tinterval,
          enabled                       => false
      );
```

Dla przedstawionego kodu *DBMS_SCHEDULER* wywołuje co *tinterval* minut procedurę *pUpdate_unused* z pakietu *OraUnusedObjMonitor*. Utworzone zadanie *orauom_1* zostanie w zadanym momencie czasu uaktywnione za pomocą procedury *DBMS_SCHEDULER.ENA-*

BLE(). Po zakończeniu procesu monitoringu rozpoczęte zadanie można usunąć z wykorzystaniem procedury *DBMS_SCHEDULER.DROP_JOB()*.

Utworzone zadanie monitorujące będzie wykonywać następujące czynności:

- sprawdzenie, które z monitorowanych obiektów zostały użyte między wywołaniami zadania i zapisanie informacji o takich przypadkach,
- wyłączenie monitoringu dla już użytych obiektów.

Przedstawioną funkcjonalność będzie udostępniał zadaniu cyklicznemu pakiet *OraUnusedObjMonitor*. Ponadto, w tym pakiecie znajdą się procedury odpowiadające za inicjalizację, zakończenie zadania oraz obsługę błędów. Pakiet musi być utworzony w schemacie *SYS*, gdyż operacje manipulacji audytem obiektów i monitoringiem indeksów wymagają uprawnień systemowych. Pakiet ten do swojego działania będzie wykorzystywał dedykowane słowniki danych audytowych.

3.3. Słownik monitoringu

Dzięki wykorzystaniu przez pakiet *OraUnusedObjMonitor* własnych tabel słownikowych, jego wykorzystanie jest elastyczne, gdyż zmiana funkcjonalności w zakresie audytowanych/monitorowanych obiektów wymaga jedynie zmiany wpisów w tabelach słownika pakietu. Należy jednak pamiętać o utworzeniu tabel słownika monitoringu poza przestrzenią tabel *SYSTEM*, aby nie powodować jej zbędnej fragmentacji. Proponuje się dodanie przestrzeni tabel *SYSTEM_WORK* udostępnionej standardowo dla schematu *SYSTEM*. Polecenia przypisujące odpowiednie przestrzenie tabel przedstawia następujący kod [1]:

```
alter user SYSTEM default tablespace SYSTEM_WORK temporary tablespace TEMP;
alter user SYS temporary tablespace TEMP;
```

W skład słownika monitoringu wchodzą następujące tabele:

- ORAUOM_UNUSED_OBJ – tabela przechowująca dane monitorowanych obiektów. Po uruchomieniu zadania cyklicznego jest inicjalizowana wpisami dla wszystkimi obiektów, które powinny być audytowane.
- ORAUOM_SYSTEM_SCHEMAS – tabela z informacjami o schematach, które nie powinny być monitorowane. W większości są to znane schematy systemowe. Ich monitoring jest zbędny, a obiekty systemowe nie powinny podlegać kasowaniu ze względu na możliwość spowodowania awarii całej instancji.
- ORAUOM_EXCLUDED_SCHEMAS – tabela z informacjami o schematach innych niż systemowe, które użytkownik pakietu OraUnusedObjMonitor chce wyłączyć z procedury audytu.

- ORAUOM_NO_AUDIT_TYPES – tabela z informacją o typach obiektów, które nie powinny podlegać monitoringowi. Ze względu na odmienny sposób analizy z poziomu DBMS, takiemu ograniczeniu podlegają indeksy. Dodanie wpisów to tej tabeli umożliwia dynamiczną zmianę zakresu monitoringu.
- ORAUOM_LOGS – tabela komunikatów o realizacji zadania cyklicznego.

Ponadto, możliwe jest wykorzystanie perspektywy będącej unią zapytania do tabeli *ORAUOM_UNUSED_OBJ* oraz perspektyw słownika systemowego *V$OBJECT_USAGE*, *DBA_SEGMENTS*.

3.4. Pakiet OraUnusedObjMonitor

Mechanizmem integrującym zadania audytowe systemu są procedury utworzone w ramach pakietu *OraUnusedObjMonitor*. Z wcześniej wymienionych przyczyn właścicielem pakietu musi być użytkownik *SYS*.

```
create or replace package SYS.OraUnusedObjMonitor
is
   procedure pCreate_jobs(tinterval in natural);
   procedure pStart_cleaning(tinterval in natural);
   procedure pStop_cleaning;
   procedure pInit_unused;
   procedure pTurn_audits_on;
   procedure pUpdate_unused;
   procedure pTurn_audits_off;
   procedure pTurn_monitoring_on;
   procedure pUpdate_monitornig;
   procedure pTurn_monitoring_off;
   procedure pAdd_log(log_msg in varchar2);
END;
```

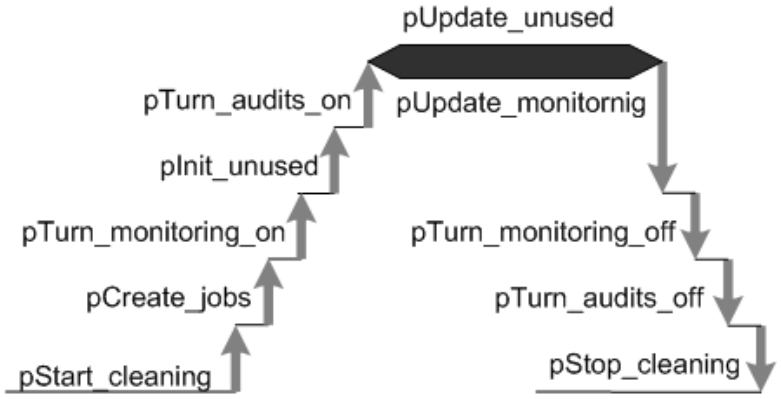
Opis procedur pakietu zostanie przeprowadzony zgodnie z chronologią ich wywoływania podczas procedury monitoringu nieużywanych obiektów.

- *pStart_cleaning* – procedura rozpoczynająca zadanie monitoringu, stanowiąca interfejs pakietu. Wszystkie procedury poza *pStart_cleaning* i *pStop_cleaning* są prywatnymi elementami pakietu. Przyjmuje jeden parametr określający częstość aktualizowania stanu obiektów. Jej działanie polega na uruchomieniu kolejno: procedury *pCreate_jobs*, uruchomieniu zadań schedulera i inicjalizacji danych dotyczących audytu i monitoringu.
- *pCreate_jobs* – procedura tworząca zadanie cykliczne dla *DBMS_SCHEDULER*. Pierwsze zadanie odpowiedzialne jest za aktualizację danych dotyczących audytu obiektów, drugie za aktualizację danych dotyczących monitoringu indeksów. Zadania zostają uruchomione dopiero z poziomu procedury wywołującej – jeśli w procedurze tworzącej zadania nie wystąpi błąd. Taki sposób inicjalizacji daje pewność, że zadania zostaną uruchomione tylko, jeśli uda się poprawnie utworzyć oba z nich.
- *pTurn_monitoring_on* – procedura włączająca monitoring indeksów. Użycie indeksów jest badane za pomocą mechanizmu monitoringu, a nie za pomocą audytu. Monitoring

powinien być włączany dla wszystkich indeksów, z wyjątkiem indeksów schematów systemowych i indeksów znajdujących się obecnie w pamięci podręcznej (gdyż są używane).

- *pInit_unused* – procedura uzupełniająca dane (inicjalizująca) tabeli *ORAUOM_UNUSED_OBJ*. Ładowanie danych odbywa się przez wybór wszystkich obiektów ze słownika *DBA_OBJECTS* z wyłączeniem obiektów znajdujących się w pamięci podręcznej (gdyż są używane), obiektów ze schematów systemowych (informacja z *ORAUOM_SYSTEM_SCHEMAS*), obiektów z innych wyłączonych schematów (informacja z *ORAUOM_EXCLUDED_SCHEMAS*), obiektów typów niemonitorowanych (informacja z *ORAUOM_NO_AUDIT_TYPES*) oraz obiektów tymczasowych.
- *pTurn_audits_on* – procedura odpowiadająca za włączenie audytu dowolnych operacji przeprowadzonych przez dowolnego użytkownika na wszystkich obiektach znajdujących się w słowniku *ORAUOM_UNUSED_OBJ*. Dzięki odpowiedniemu zapisowi ewentualne błędy nie powodują wycofania wszystkich dotąd dokonanych operacji, ale są logowane i funkcjonowanie procedury jest kontynuowane.
- *pUpdate_monitornig* – procedura sprawdzająca postęp monitoringu indeksów na podstawie wpisów z *V$OBJECT_USAGE*. Analizuje, które indeksy zostały wykorzystane pomiędzy dwoma ostatnimi wywołaniami zadania, i wyłącza dla nich monitoring.
- *pUpdate_unused* – analogicznie jak *pUpdate_monitornig*, sprawdzany jest postęp w monitoringu obiektów na podstawie wpisów z *SYS.AUD$*. Pobierane są dane obiektów, które zostały ostatnio użyte, a następnie dla każdego z tych obiektów ustawiana jest data ostatniego dostępu oraz wyłączany jest audyt. Ostatecznie kasowane są wszystkie dane z tabeli *SYS.AUD$*.
- *pTurn_monitoring_off* oraz *pTurn_audits_off* – wyłączają monitoring indeksów i audyt obiektów. Służą do zakończeniu zadania monitoringu.
- *pStop_cleaning* – procedura zakończenie procesu monitoringu. Wykonywane są kolejno: usunięcie zadań schedulera, wyłączenie audytu, wyłączenie monitoringu, wyczyszczenie tabeli monitorowanych obiektów.

Kolejność wywoływania procedur podczas procesu monitoringu przedstawiona została na rys. 2.



Rys. 2. Kolejność wywołania procedur podczas procesu monitoringu z wykorzystaniem pakietu OraUnusedObjMonitor

Fig. 2. Sequence of steps of monitoring system for OraUnusedObjMonitor package

Przykład implementacji procedury *pUpdate_unused* przedstawia następujący kod PL/SQL:

```
procedure pUpdate_unused
  is
      cursor auds is
          select obj_name as name, owner from dba_audit_trail;
  begin

      for obj in auds
      loop
          begin
              update ORAUOM_UNUSED_OBJ set UNU_LAST_ACCESS = sysdate
                  where
                      UNU_NAME = obj.name
                  and
                      UNU_OWNER = obj.owner;
              execute immediate 'noaudit all on ' || obj.owner || '."' ||
obj.name || '"';
          exception
              when others then
                  pAdd_log('Nie udało się wyłączyć audytu na obiekcie ' ||
                      obj.owner || '.' || obj.name || '.' || sqlerrm);
          end;
      end loop;
      --  wyczyszczenie tabeli zawierającej dane wynikowe audytu
      execute immediate 'truncate table sys.aud$';
  exception
      when others then
          pAdd_log(sqlerrm);
          raise;
  end pUpdate_unused;
```

4. Podsumowanie

Rozległy system baz danych, podlegający ciągłej ewaluacji na poziomie schematu logicznego i fizycznego, jest trudny do utrzymania pod względem jednoznacznego opisu obiektów. Rosnąca liczba obiektów nieużywanych może w konsekwencji spowodować zmniejszenie wydajności bazy danych. Monitorowanie w przedstawiony sposób zawartości bazy danych pod względem nieużywanych obiektów może rozwiązać wskazany problem. Powinno ono

być przeprowadzane w określanych odpowiednimi procedurami wewnętrznymi odstępach czasu w celu eliminacji niepożądanych obiektów, zwłaszcza z bazy produkcyjnej. Na podstawie wygenerowanych wyników należy określić, czy obiekty, które nie wykazały żadnej aktywności, są konieczne w bazie, jeśli nie, to początkowo proponuje się zmianę ich nazwy. Wyniki należy skonsultować z osobami odpowiedzialnymi za monitorowane schematy, aby nie usunąć obiektów wymaganych do poprawnej pracy systemu. W zależności od zastosowania danego systemu baz danych należy w odpowiedni sposób ustalać terminy prowadzenia monitoringu. W niektórych wypadkach najlepszym okresem może być uruchamianie monitoringu pod koniec roku kalendarzowego, w sytuacji gdy uruchamiane są miesięczne i roczne procesy przetwarzania danych. Możliwe jest wówczas osiągnięcie najbardziej prawdopodobnych wyników.

Przedstawione rozwiązanie zostało przetestowane w środowisku produkcyjnym, dlatego może zostać uznane za skuteczny sposób rozwiązania przedmiotowego problemu.

BIBLIOGRAFIA

1. Gutta R.: Oracle skrypty administracyjne. Mikom, Warszawa 2002.
2. Loney K.: Oracle Database 10g Kompendium administratora. Helion, Gliwice 2005.
3. Whalen E.: Oracle Database 10g Administracja bazy danych w Linuksie. Helion, Gliwice 2007.
4. Theriault M., Carmichael R., Viscusi J.: Oracle DBA – administrowanie bazą danych. RM, Warszawa 2001.
5. Oracle: Oracle Database Administrator's Guide 10g Release 2. B14231-02, 2006.
6. Oracle: Oracle Database Security Guide 10g Release 2. B14266-04, 2008.
7. Oracle: Oracle Database PL/SQL Packages and Types Reference 10g Release 2. B14258-02, 2007.

Recenzent: Dr inż. Łukasz Wyciślik

Wpłynęło do Redakcji 20 stycznia 2009 r.

Abstract

The problem of maintain of database systems is very complex. One of the most frequent appearing problems is unused object identification, what in long term of time may in negative way affect on system productivity.

This kind of problem appears in simple database system and complex systems with master database, test and develop database as well.

Proposed in this paper solution based on P/SQL for Oracle 10g may supporting DBA maintains database systems.

OraUnusedObjMonitor package can support DBMS scheduler for automate information gathering.

Adres

Dariusz KARPISZ: Politechnika Krakowska, Instytut Informatyki Stosowanej,
al. Jana Pawła II 37, 31-864 Kraków, Polska, drejku@poczta.onet.pl.