



Oracle Database 12c

Programowanie w języku PL/SQL

Twój przewodnik po PL/SQL!

Michael McLaughlin

Tytuł oryginału: Oracle Database 12c PL/SQL Programming

Tłumaczenie: Tomasz Walczak

ISBN: 978-83-246-9923-0

Original edition copyright © 2014 by McGraw-Hill Education (Publisher).
All rights reserved.

Polish edition copyright © 2015 by HELION S.A.
All rights reserved.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/or12ps>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:

<ftp://ftp.helion.pl/przyklady/or12ps.zip>

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

O autorze	15
Podziękowania	17
Wprowadzenie	19

CZĘŚĆ I Podstawy języka PL/SQL

1 Przegląd programowania w języku Oracle PL/SQL	33
Tło historyczne języka PL/SQL	33
Architektura używana przy programowaniu baz Oracle	35
Baza danych	36
Język PL/SQL	37
Architektura przetwarzania instrukcji w Oracle	40
Model dwuwarstwowy	41
Model n-warstwowy	41
Podsumowanie	43
Test wiedzy	43
2 Nowe funkcje	45
Nowe funkcje SQL-a	45
Tworzenie obiektów LIBRARY za pomocą katalogu wirtualnego	46
Definiowanie tabel z obsługą wymiaru VT	46
Wzbogacona składnia instrukcji LEFT OUTER JOIN w bazach Oracle	47
Domyślne wartości kolumn oparte na sekwencjach	47
Wartości domyślne przy bezpośrednim wstawianiu null	49
Kolumny IDENTITY	49
Większa pojemność typów łańcuchowych i typu RAW	50
Przekazywanie wartości z instrukcji SQL-a do zewnętrznych programów	50
Określanie liczby wyników i zwracanych przedziałów wyników w natywnym SQL-u	52
Sterownik bazy Oracle dla aplikacji dla baz MySQL	55
Instrukcje CROSS APPLY, OUTER APPLY i LATERAL w SQL-u	55
Widoki z ustawieniem BEQUEATH CURRENT_USER	56
Nowe funkcje języka PL/SQL	57
Zapisywanie wyników funkcji z uprawnieniami jednostki wywołującej	57
Tworzenie białych list jednostek wywołujących w programach języka PL/SQL	57
Natywna obsługa klienckich interfejsów API w zakresie typów języka PL/SQL	58
Nowy pakiet utl_call_stack języka PL/SQL	59
Nowa procedura expand_sql_text w pakiecie dbms_utility	59
Nowy formalny schemat procedury parse w pakiecie dbms_sql	60
Funkcje języka PL/SQL w klauzuli WITH SQL-a	60

6 Programowanie w języku PL/SQL

Stosowanie w SQL-u typów danych specyficznych dla języka PL/SQL	61
Niejawne wiązanie parametrów REF CURSOR	63
Skrypty pomocnicze	64
Podsumowanie	64
Test wiedzy	64
3 Podstawy języka PL/SQL	67
Struktura bloków	67
Sekcja wykonawcza	68
Podstawowa struktura bloku	68
Sekcja deklaracji	71
Sekcja obsługi wyjątków	71
Działanie zmiennych w blokach	72
Bloki anonimowe	72
Zagnieżdżone bloki anonimowe	76
Lokalne bloki nazwane	78
Składowane bloki nazwane	80
Podstawowe skalarne i złożone typy danych	82
Skalarne typy danych	82
Kotwiczenie atrybutów i tabel	84
Złożone typy danych	86
Struktury sterujące	97
Struktury warunkowe	97
Struktury iteracyjne	99
Wyjątki	106
Wyjątki zdefiniowane przez użytkownika	107
Dynamiczne wyjątki zdefiniowane przez użytkownika	108
Operacje masowe	108
Funkcje, procedury i pakiety	110
Funkcje	110
Procedury	112
Pakiety	113
Zasięg transakcji	117
Pojedynczy zasięg transakcji	117
Wiele zasięgów transakcji	118
Wyzwalacze bazodanowe	119
Podsumowanie	120
Test wiedzy	120
4 Podstawowe elementy języka	123
Jednostki leksykalne	123
Ograniczniki	123
Identyfikatory	127
Literały	129
Komentarze	130
Zmienne i typy danych	131
Typy zmiennych	132
Skalarne typy danych	134
Duże obiekty (typy LOB)	147
Złożone typy danych	149
Systemowe kursory referencyjne	152
Podsumowanie	154
Test wiedzy	154

5	Struktury sterujące	157
	Instrukcje warunkowe	157
	Instrukcje IF	163
	Instrukcje CASE	166
	Instrukcje kompilacji warunkowej	168
	Instrukcje iteracyjne	171
	Pętle proste	171
	Pętle FOR	177
	Pętle WHILE	178
	Kursory	181
	Kursory niejawne	182
	Kursory jawne	186
	Instrukcje masowe	196
	Instrukcje BULK COLLECT INTO	196
	Instrukcje FORALL	200
	Pomocnicze skrypty	204
	Podsumowanie	204
	Test wiedzy	205
6	Kolekcje	207
	Wprowadzenie do kolekcji	207
	Typy obiektowe: tablice VARRAY i tabele zagnieżdżone	210
	Tablice VARRAY	210
	Tabele zagnieżdżone	213
	Tablice asocjacyjne	225
	Definiowanie i stosowanie tablic asocjacyjnych	226
	API Collection	231
	Metoda COUNT	233
	Metoda DELETE	234
	Metoda EXISTS	235
	Metoda EXTEND	236
	Metoda FIRST	236
	Metoda LAST	237
	Metoda LIMIT	237
	Metoda NEXT	238
	Metoda PRIOR	238
	Metoda TRIM	239
	Pomocnicze skrypty	240
	Podsumowanie	240
	Test wiedzy	240
7	Obsługa błędów	243
	Typy i zasięg wyjątków	243
	Błędy kompilacji	245
	Błędy czasu wykonania	247
	Wbudowane funkcje do zarządzania wyjątkami	252
	Wyjątki zdefiniowane przez użytkownika	253
	Deklarowanie wyjątków zdefiniowanych przez użytkownika	253
	Dynamiczne wyjątki zdefiniowane przez użytkownika	256
	Funkcje do zarządzania stosem błędów	258
	Pomocnicze skrypty	263
	Podsumowanie	263
	Test wiedzy	263

CZĘŚĆ II

Programowanie w języku PL/SQL

8	Funkcje i procedury	267
	Architektura funkcji i procedur	268
	Zasięg transakcji	273
	Wywoływanie podprogramów	275
	Notacja oparta na pozycji	275
	Notacja oparta na nazwie	276
	Notacja mieszana	276
	Notacja z pominięciem	276
	Notacja w wywołaniach w języku SQL	276
	Funkcje	277
	Wybór rodzaju funkcji	278
	Opcje używane przy tworzeniu funkcji	279
	Funkcje o parametrach przekazywanych przez wartość	290
	Funkcje o parametrach przekazywanych przez referencję	301
	Procedury	303
	Procedury o parametrach przekazywanych przez wartość	304
	Procedury o parametrach przekazywanych przez referencję	307
	Pomocnicze skrypty	309
	Podsumowanie	309
	Test wiedzy	309
9	Pakiety	311
	Architektura pakietu	311
	Specyfikacja pakietu	316
	Elementy prototypu	317
	Dyrektywa prekompilatora <code>SERIALLY_REUSABLE</code>	319
	Zmienne	320
	Typy danych	322
	Komponenty — funkcje i procedury	324
	Ciało pakietu	325
	Elementy prototypu	325
	Zmienne	327
	Typy	329
	Komponenty — funkcje i procedury	330
	Uprawnienia jednostki definiującej i jednostki wywołującej	332
	Zarządzanie pakietami w katalogu bazy danych	335
	Wyszukiwanie, walidacja i opisywanie pakietów	336
	Sprawdzanie zależności	337
	Metody sprawdzania poprawności — znaczniki czasu i sygnatury	338
	Podsumowanie	339
	Test wiedzy	339
10	Duże obiekty	341
	Praca z wewnętrznie składowanymi dużymi obiektami	342
	Przypisywanie do dużych obiektów danych o wielkości poniżej 32 kilobajtów	342
	Przypisywanie do dużych obiektów danych o wielkości powyżej 32 kilobajtów	344
	Wczytywanie plików do wewnętrznie przechowywanych kolumn	352
	Wczytywanie lokalnych plików do kolumn typu <code>CLOB</code> lub <code>NCLOB</code>	352
	Wczytywanie plików lokalnych do kolumn typu <code>BLOB</code>	355
	Używanie dużych obiektów za pomocą stron WWW	357

Praca z plikami binarnymi (typ BFILE)	363
Tworzenie i używanie katalogów wirtualnych	363
Wczytywanie ścieżek kanonicznych i nazw plików	368
Pakiet DBMS_LOB	374
Stałe pakietu	375
Wyjątki pakietu	376
Metody do otwierania i zamykania	376
Metody do manipulowania dużymi obiektami	377
Metody do introspekcji	382
Metody do obsługi obiektów typu BFILE	385
Metody do obsługi tymczasowych dużych obiektów	387
Metody do obsługi bezpiecznych odnośników	388
Skrypty pomocnicze	390
Skrypt przekształcający dane typu LONG na wartości typu CLOB	390
Zarządzanie dużymi obiektami w systemie plików	391
Zarządzanie obiektami typów CLOB i BLOB z poziomu stron WWW	391
Zarządzanie obiektami typu BFILE z poziomu stron WWW	391
Podsumowanie	391
Test wiedzy	391
11 Typy obiektowe	393
Wprowadzenie do obiektów	395
Deklarowanie typów obiektowych	396
Implementacja ciała typów obiektowych	398
Tworzenie białych list dla typów obiektowych	402
Gettery i settery	403
Statyczne metody składowe	405
Porównywanie obiektów	406
Dziedziczenie i polimorfizm	412
Deklarowanie klas pochodnych	414
Implementowanie klas pochodnych	415
Ewolucja typu	418
Kolekcje obiektów	419
Deklarowanie kolekcji obiektów	419
Implementowanie kolekcji obiektów	420
Skrypty pomocnicze	422
Podsumowanie	423
Test wiedzy	423
12 Wyzwalacze	425
Wprowadzenie do wyzwalaczy	425
Architektura wyzwalaczy w bazie danych	428
Wyzwalacze DDL	431
Funkcje-atrybuty zdarzeń	433
Tworzenie wyzwalaczy DDL	442
Wyzwalacze DML	444
Wyzwalacze z poziomu instrukcji	445
Wyzwalacze z poziomu wierszy	447
Wyzwalacze złożone	453
Wyzwalacze zastępujące	457
Wyzwalacze systemowe (bazy danych)	461
Ograniczenia związane z wyzwalaczami	462
Maksymalny rozmiar wyzwalaczy	462
Instrukcje języka SQL	463

10 Programowanie w języku PL/SQL

Typy danych LONG i LONG RAW	463
Tabele mutujące	463
Wyzwalacze systemowe	464
Skrypty pomocnicze	465
Podsumowanie	465
Test wiedzy	465
13 Dynamiczny SQL	467
Architektura dynamicznego SQL-a	468
Wbudowany dynamiczny język SQL (NDS)	468
Instrukcje dynamiczne	469
Instrukcje dynamiczne z danymi wejściowymi	471
Instrukcje dynamiczne z danymi wejściowymi i wyjściowymi	473
Instrukcje dynamiczne o nieznanym liczbie danych wejściowych	477
Pakiet DBMS_SQL	478
Instrukcje dynamiczne	479
Instrukcje dynamiczne o zmiennych wejściowych	482
Instrukcje dynamiczne o zmiennej liczbie danych wejściowych i stałej liczbie danych wyjściowych	484
Dynamiczne instrukcje o zmiennej liczbie danych wejściowych i wyjściowych	487
Definicja pakietu DBMS_SQL	492
Skrypty pomocnicze	503
Podsumowanie	503
Test wiedzy	503

CZĘŚĆ III Dodatki i słowniczek

A Wprowadzenie do bazy Oracle	507
Architektura bazy danych Oracle	508
Uruchamianie i zatrzymywanie serwera bazy Oracle Database 12c	513
Operacje w systemach Unix i Linux	514
Operacje w systemie Microsoft Windows	518
Uruchamianie i zatrzymywanie odbiornika Oracle	519
Architektura MVCC	523
Transakcje na danych	524
Kontrola blokowania i izolacji w instrukcjach DML	527
Uprawnienia jednostki definiującej i uprawnienia jednostki wywołującej	528
Uprawnienia jednostki definiującej	528
Uprawnienia jednostki wywołującej	529
Interaktywne i wsadowe przetwarzanie instrukcji SQL-a	530
Interfejs SQL*Plus uruchamiany z wiersza poleceń	530
Narzędzie Oracle SQL Developer	547
Administrowanie bazą danych	553
Dodawanie kont użytkowników	553
Stosowanie ograniczeń w bazach danych	560
Wzmacnianie zabezpieczeń	569
Zarządzanie danymi	577
Dostrajanie SQL-a	580
Instrukcja EXPLAIN PLAN	581
Pakiet DBMS_XPLAN	582

Śledzenie instrukcji SQL-a	585
Instrukcje związane z sesją śledzenia	586
Przekształcanie nieprzetworzonych plików śladu na czytelny format	588
Podsumowanie	589
B Wprowadzenie do języka SQL	591
Typy danych środowiska SQL*Plus w Oracle	594
Język definicji danych (DDL)	597
Instrukcja CREATE	598
Instrukcja ALTER	654
Instrukcja RENAME	669
Instrukcja DROP	670
Instrukcja TRUNCATE	672
Instrukcja COMMENT	672
Instrukcje DML	672
Transakcje zgodne z modelem ACID	673
Instrukcja INSERT	676
Instrukcja UPDATE	688
Instrukcja DELETE	699
Instrukcja MERGE	703
Język kontroli transakcji (TCL)	709
Zapytania — instrukcje SELECT	711
Zapytania zwracające kolumny lub wyniki z kolumn	712
Zapytania agregujące	725
Zapytania selektywnie zwracające kolumny lub wyniki	729
Wyniki złączeń	736
Złączenia dotyczące wierszy	738
Złączenia łączące kolekcje	746
Podsumowanie	749
C Funkcje wbudowane języka SQL	751
Funkcje znakowe	751
Funkcja ASCII	751
Funkcja ASCIISTR	752
Funkcja CHR	752
Funkcja CONCAT	753
Funkcja INITCAP	753
Funkcja INSTR	753
Funkcja LENGTH	754
Funkcja LOWER	754
Funkcja LPAD	755
Funkcja LTRIM	755
Funkcja REPLACE	756
Funkcja REVERSE	756
Funkcja RPAD	756
Funkcja RTRIM	757
Funkcja UPPER	757
Funkcje do konwersji typów danych	758
Funkcja CAST	758
Funkcja CONVERT	760
Funkcja TO_CHAR	760
Funkcja TO_CLOB	762

Funkcja TO_DATE	762
Funkcja TO_LOB	763
Funkcja TO_NCHAR	764
Funkcja TO_NCLOB	764
Funkcja TO_NUMBER	764
Funkcje do konwersji dat i czasu	765
Funkcja ADD_MONTHS	765
Funkcja CURRENT_DATE	765
Funkcja CURRENT_TIMESTAMP	766
Funkcja DBTIMEZONE	766
Funkcja EXTRACT	766
Funkcja FROM_TZ	767
Funkcja LAST_DAY	767
Funkcja LOCALTIMESTAMP	767
Funkcja MONTHS_BETWEEN	768
Funkcja NEW_TIME	768
Funkcja ROUND	769
Funkcja SYSDATE	769
Funkcja SYSTIMESTAMP	769
Funkcja TO_CHAR(data)	770
Funkcja TO_DSINTERVAL	771
Funkcja TO_TIMESTAMP	771
Funkcja TO_TIMESTAMP_TZ	771
Funkcja TO_YMINTERVAL	772
Funkcja TRUNC(data)	772
Funkcja TZ_OFFSET	772
Funkcje do zarządzania kolekcjami	773
Funkcja CARDINALITY	773
Funkcja COLLECT	773
Funkcja POWERMULTISET	776
Funkcja POWERMULTISET_BY_CARDINALITY	777
Funkcja SET	777
Operatory zbiorów działające dla kolekcji	777
Operator CARDINALITY	779
Operator EMPTY	779
Operator MULTISET	779
Operator MULTISET EXCEPT	780
Operator MULTISET INTERSECT	780
Operator MULTISET UNION	781
Operator SET	781
Operator SUBMULTISET OF	782
Funkcje liczbowe	783
Funkcja CEIL	783
Funkcja FLOOR	783
Funkcja MOD	783
Funkcja POWER	785
Funkcja REMAINDER	785
Funkcja ROUND	786
Funkcje do zarządzania błędami	786
Funkcja SQLCODE	787
Funkcja SQLERRM	787
Funkcje różne	789
Funkcja BFILENAME	789
Funkcja COALESCE	791

Funkcja DECODE	791
Funkcja DUMP	792
Funkcja EMPTY_BLOB	792
Funkcja EMPTY_CLOB	795
Funkcja GREATEST	795
Funkcja LEAST	797
Funkcja NANVL	798
Funkcja NULLIF	798
Funkcja NVL	799
Funkcja SYS_CONTEXT	799
Funkcja TABLE	802
Funkcja TREAT	804
Funkcja USERENV	805
Funkcja VSIZE	806
Podsumowanie	806
D Wbudowane pakiety i typy języka PL/SQL	807
Nowe pakiety w wersjach Oracle Database 11g i Oracle Database 12c	807
Przykłady zastosowania pakietów	814
Przykład ilustrujący pakiet DBMS_APPLICATION_INFO	814
DBMS_COMPARISON	818
DBMS_CRYPTO	823
DBMS_FGA	825
Przypadek użycia — narzędzie do analizy zapytań	826
Podsumowanie	831
E Wprowadzenie do wyrażeń regularnych	833
Wprowadzenie do wyrażeń regularnych	833
Klasy znaków	833
Klasy porządkowania	836
Metaznaki	836
Metasekwencje	838
Literały	838
Stosowanie wyrażeń regularnych	839
Funkcja REGEXP_COUNT	839
Funkcja REGEXP_INSTR	842
Funkcja REGEXP_LIKE	843
Funkcja REGEXP_REPLACE	844
Funkcja REGEXP_SUBSTR	846
Podsumowanie	847
F Wprowadzenie do opakowywania kodu w języku PL/SQL	849
Ograniczenia w opakowywaniu kodu w języku PL/SQL	850
Ograniczenia związane z narzędziem wrap języka PL/SQL	850
Ograniczenia funkcji DBMS_DDL.WRAP	850
Stosowanie narzędzia wrap	850
Opakowywanie kodu za pomocą pakietu DBMS_DDL	851
Funkcja WRAP	851
Procedura CREATE_WRAPPED	855
Podsumowanie	856
G Wprowadzenie do hierarchicznego programu profilującego języka PL/SQL	857
Konfigurowanie schematu	857
Zbieranie danych	859

14 Programowanie w języku PL/SQL

Odczytywanie danych wyjściowych programu profilującego	861
Odczyt surowych danych wyjściowych	862
Definiowanie tabel na potrzeby programu profilującego języka PL/SQL	863
Zapytania o przetworzone dane	863
Używanie narzędzia plshprof	865
Podsumowanie	867
H Słowa zarezerwowane i kluczowe języka PL/SQL	869
Podsumowanie	875
I Odpowiedzi do testów wiedzy	877
Rozdział 1.	877
Rozdział 2.	879
Rozdział 3.	881
Rozdział 4.	883
Rozdział 5.	885
Rozdział 6.	887
Rozdział 7.	889
Rozdział 8.	891
Rozdział 9.	893
Rozdział 10.	895
Rozdział 11.	897
Rozdział 12.	899
Rozdział 13.	901
Słowniczek	905
Skorowidz	917

Nowe funkcje

W tym rozdziale opisane są nowe funkcje języków SQL i PL/SQL, które bezpośrednio wpływają na sposób pisania programów w języku Oracle PL/SQL i zarządzania nimi. W czasie gdy powstawała ta książka, możliwe było jeszcze pojawienie się dodatkowych funkcji w tym wydaniu bazy lub w jej następnej wersji. Informacje o najnowszych zmianach znajdziesz w przewodniku *Oracle Database New Features Guide*.

Omówienie nowych mechanizmów jest w tym rozdziale podzielone na dwie części:

- nowe funkcje SQL-a,
- nowe funkcje języka PL/SQL.

Niektóre funkcje zasługują na wielostronicowe opisy z przykładowym kodem, natomiast inne wymagają tylko krótkiego przedstawienia, ponieważ są omówione w innych miejscach książki. Wtedy wskazują tylko rozdział z opisem danego mechanizmu. Jeśli dopiero poznajesz język PL/SQL, możesz przejść bezpośrednio do rozdziału 3., aby zapoznać się z podstawami tego języka. Następnie wróć do niniejszego rozdziału.

Nowe funkcje SQL-a

Oracle udostępnia w bazie Oracle Database 12c wiele nowych funkcji. Zmiany w składni SQL-a są stosunkowo liczne, dlatego tu opisano głównie te, które mają wpływ na działanie kodu.

- W Oracle Database 12c można korzystać z katalogów wirtualnych przy tworzeniu obiektów LIBRARY z zewnętrznymi procedurami.
- Technologia Flashback została usprawniona dzięki wprowadzeniu wymiarów rzeczywistego czasu (ang. *valid-time* — VT).
- W Oracle Database 12c rozbudowano składnię instrukcji JOIN ze standardu ANSI 92. Teraz można wywołać polecenie LEFT OUTER JOIN dla dwóch lub więcej tabel podanych po lewej stronie złączenia.
- Wartości domyślne kolumn mogą teraz zawierać referencje do pseudokolumn `.nextval` i `.currval`, co jest wygodnym rozwiązaniem.
- W bazie Oracle Database 12c wprowadzono kolumnę IDENTITY (kolumnę identyfikacyjną), która automatycznie generuje sekwencyjne wartości klucza sztucznego.
- W Oracle Database 12c pojawiła się klauzula ON NULL dla wartości domyślnych, co chroni przed bezpośrednim zastąpieniem takich wartości wartością null.
- W Oracle Database 12c zwiększono pojemność typów danych VARCHAR2, NVARCHAR2 i RAW (wymaga to poprawnego ustawienia parametrów bazy).

- Oracle Database 12c, podobnie jak baza Microsoft SQL Server, umożliwia przekazywanie wyników zapytań bezpośrednio do programów zewnętrznych.
- Oracle Database 12c zapewnia natywną obsługę SQL-a w zakresie ograniczania liczby wyników zapytania i przedziałów zwracanych wyników.
- W Oracle Database 12c pojawił się nowy sterownik zastępujący bibliotekę kliencką MySQL 5.5.
- Do SQL-a dodano polecenia CROSS APPLY, OUTER APPLY i LATERAL, przydatne przy pracy z tabelami zagnieżdżonymi.
- Obecnie można tworzyć widoki za pomocą uprawnień jednostki definiującej (jest to dawny sposób, obecnie używany jako domyślny) lub przy użyciu uprawnień jednostki wywołującej. Jedyną różnicą jest składnia. W modelu z uprawnieniami jednostki definiującej używane jest polecenie DEQUEATH DEFINER, a dla jednostki wywołującej używa się instrukcji DEQUEATH INVOKER.

Tworzenie obiektów LIBRARY za pomocą katalogu wirtualnego

Obiekty LIBRARY to repozytoria z zewnętrznymi bibliotekami. Takie biblioteki są napisane w C lub w językach, które można wywoływać w C. Użycie takiej biblioteki wymaga umieszczenia pliku z nią w katalogu i wskazania nazwy tego katalogu w pliku *listener.ora* i instrukcji CREATE LIBRARY.

W Oracle Database 12c można zastąpić katalog fizyczny katalogiem wirtualnym. Poniższa składnia pokazuje, jak utworzyć bibliotekę za pomocą katalogu fizycznego:

```
SQL> CREATE OR REPLACE LIBRARY demo IS
  2  '<katalog_główny_oracle>/<niestandardowa_biblioteka>/<nazwa_pliku>.<rozszerzenie>';
  3  /
```

Za pomocą nowej składni obiekt LIBRARY można utworzyć na podstawie katalogu wirtualnego:

```
SQL> CREATE OR REPLACE LIBRARY demo IS '<nazwa_biblioteki.so>' IN
  2  nazwa_katalogu_wirtualnego;
```

Drugi argument to katalog wirtualny. Tworzenie takich katalogów opisano w podrozdziale „Katalogi wirtualne” w dodatku B. W Oracle Database 12c proces tworzenia katalogów tego rodzaju się nie zmienił.

Definiowanie tabel z obsługą wymiaru VT

W Oracle Database 12c dostępna jest obecnie obsługa wymiaru VT (ang. *valid time*). Ten wymiar różni się od wymiaru czasu transakcji (ang. *transaction time* — TT). VT odpowiada dacie zdarzenia biznesowego, na przykład zatrudnienia, promocji lub zwolnienia pracownika. TT odpowiada fizycznemu momentowi wstawienia lub zaktualizowania wiersza.

W Oracle Database 11g pojawiły się archiwa Flashback Data Archive, w których używano czasu TT. Technologia Flashback pozwala przyrzeć się danym archiwalnym w celu przeanalizowania trendów w zapytaniach, różnic w raportach lub historii transakcji. Są to wymiary retrospektywne, ponieważ dane są w nich dzielone na podstawie czasu operacji.

W Oracle Database 12c wprowadzono obsługę wymiaru VT w wyniku sformalizowania w definicjach tabel dwóch podejść. Jedno polega na definiowaniu okresów przez jawne przypisanie ich do kolumn. Drugie definiuje okresy w kolumnach niejawnie. Nowa składnia SQL-a związana z wymiarem VT to PERIOD FOR. Jej zastosowanie zobaczysz w przykładowych instrukcjach CREATE TABLE w dalszych punktach.

Warto zauważyć, że to wymiar VT, a nie TT steruje operacjami retrospektywnymi. Za pomocą wymiaru VT można zarządzać procesem ILM (ang. *Information Lifecycle Management*).

Tabele z jawnie dodanymi kolumnami VT

Przyjrzyj się teraz przykładowej tabeli *renta1*. Obejmuje ona kolumny *check_out_date* i *return_date*. W wersjach starszych niż Oracle Database 12c za zarządzanie tymi kolumnami odpowiadał interfejs API aplikacji. Zawierają one ważną logikę biznesową opisującą, jak wypożyczalnia filmów wideo, na przykład Redbox, nalicza płatności. Wymiar VT pozwala teraz jawnie zidentyfikować takie kolumny:

```

SQL> CREATE TABLE rental
 2 ( rental_id      NUMBER GENERATED ALWAYS AS IDENTITY
 3 , customer_id   NUMBER CONSTRAINT nn_rental_01 NOT NULL
 4 , check_out_date DATE CONSTRAINT nn_rental_02 NOT NULL
 5 , return_date    DATE
 6 , created_by    NUMBER CONSTRAINT nn_rental_03 NOT NULL
 7 , creation_date DATE CONSTRAINT nn_rental_04 NOT NULL
 8 , last_updated_by NUMBER CONSTRAINT nn_rental_05 NOT NULL
 9 , last_update_date DATE CONSTRAINT nn_rental_06 NOT NULL
10 , PERIOD FOR rental_event (check_out_date, return_date)
11 , CONSTRAINT pk_rental PRIMARY KEY(rental_id)
12 , CONSTRAINT fk_rental_01 FOREIGN KEY(customer_id)
13 REFERENCES contact(contact_id)
14 , CONSTRAINT fk_rental_02 FOREIGN KEY(created_by)
15 REFERENCES system_user(system_user_id)
16 , CONSTRAINT fk_rental_03 FOREIGN KEY(last_updated_by)
17 REFERENCES system_user(system_user_id));

```

Wiersze 4. i 5. to kolumny VT z logiką biznesową. W wierszu 10. do okresu związanego z regułą biznesową jawnie przypisywany jest identyfikator. Pozwala to na wywoływanie retrospektywnych zapytań dotyczących danego okresu.

Oto przykładowe zapytanie z wykorzystaniem wymiaru VT:

```

SQL> SELECT *
 2 rental AS OF PERIOD FOR rental_event
 3 TO_TIMESTAMP('04-AUG-2013 12:00:00 AM');

```

Ponadto jeśli używasz pakietu `dbms_flashback_archive`, możesz zastosować człon `AS OF` do przedziałów VT. Zachęcam do korzystania z podejścia z jawnym wskazywaniem kolumn VT.

Tabele z niejawnie definiowanymi kolumnami z czasem VT

Gdy Oracle wprowadza nowe funkcje, zawsze udostępnia różne możliwości. Dotyczy to także kolumn z czasem VT. Możesz utworzyć tabelę z niejawnie definiowanymi kolumnami z czasem VT. W tym celu usuń referencje do tych kolumn. W kodzie przykładowej tabeli `rental` oznacza to zmianę wiersza 10. w następujący sposób:

```
10 , PERIOD FOR rental_event
```

W tym wierszu pominięto kolumny z instrukcji `CREATE TABLE`.

Wzbogacona składnia instrukcji LEFT OUTER JOIN w bazach Oracle

Oracle Database 12c obsługuje obecnie polecenie `LEFT OUTER JOIN` umożliwiające podanie po lewej stronie złączenia dwóch i więcej tabel. Wcześniej można było w tym miejscu zastosować tylko jedną tabelę. W bazie Oracle Database 11g próby podania dwóch tabel powodowały błąd `ORA-01417`.

Oto zalety tej nowej funkcji:

- Podawanie tabel po lewej stronie umożliwia zmianę ich kolejności, co z kolei pozwala na generowanie lepszych planów wykonania instrukcji.
- Obsługa wielu widoków upraszcza pracę programistów piszących operacje `OUTER JOIN`.

Wadą rozbudowanej instrukcji `LEFT OUTER JOIN` z baz Oracle jest brak przenośności. Korzystne jest natomiast to, że można tworzyć wydajniejsze operacje `OUTER JOIN`.

Domyślne wartości kolumn oparte na sekwencjach

Oracle Database 12c umożliwia jawne wiązanie sekwencji z tabelami. Dostępne są dwie techniki. Jedna polega na utworzeniu sekwencji i jawnym powiązaniu jej z kolumną tabeli. Druga wymaga wykorzystania kolumn identyfikacyjnych (z modyfikatorem `IDENTITY`; jest to następna nowa funkcja). W pierwszym z tych podejść można wykorzystać domyślne wartości z niezależnych sekwencji.

Przyjrzyj się przykładowej tabeli customer. Dla uproszczenia ma ona tylko dwie kolumny. Pierwsza to kolumna z kluczem sztucznym zawierająca kolejne wartości. Te wartości nie są powiązane z danymi z żadnej tabeli i powinny być powiązane w sposób „jeden do jednego” z kluczem naturalnym tabeli. Klucz naturalny to jedna lub — zazwyczaj — więcej kolumn, które niepowtarzalnie identyfikują każdy wiersz tabeli. W tym przykładzie kluczem naturalnym jest kolumna customer_name. W praktyce jest mało prawdopodobne, by taka kolumna była dobrym kluczem naturalnym, jednak takie założenie pozwala uprościć przykład i skoncentrować się na domyślnych wartościach kolumny.

Przed utworzeniem tabeli należy przygotować sekwencję. Jest to odejście od techniki stosowanej w przeszłości w bazach Oracle, jednak tu omawiane są rozwiązania z nowego wspaniałego świata baz Oracle Database 12c. Utwórz więc sekwencję rozpoczynającą się od wartości 1:

```
SQL> CREATE SEQUENCE customer_s;
```

Tę sekwencję należy utworzyć na początku, ponieważ jest ona używana w czasie tworzenia tabeli o domyślnych wartościach kolumny.

```
SQL> CREATE TABLE customer
 2 ( customer_id NUMBER DEFAULT customer_s.nextval
 3 , customer_name VARCHAR2(20));
```

Ponieważ ten przykład ma ilustrować zarządzanie wartościami klucza głównego i klucza obcego w ramach transakcji, trzeba też utworzyć następną sekwencję i tabelę. Tu będą to sekwencja preference_s i tabela preference.

Zamiast rozdzielać obie instrukcje, poniżej przedstawiono je jedna pod drugą:

```
SQL> CREATE SEQUENCE preference_s;
SQL> CREATE TABLE preference
 2 ( preference_id NUMBER DEFAULT preference_s.nextval
 3 , customer_id NUMBER DEFAULT customer_s.currval
 4 , preference_name VARCHAR2(20));
```

Wartości sekwencji podanej po słowie DEFAULT pozwalają pominąć wyzwalacze ON INSERT. Dzięki nim można też zrezygnować z bezpośredniego stosowania pseudokolumn .nextval i .currval w instrukcjach INSERT związanych z sekwencją. Trzeba jednak pamiętać, że zależność między instrukcjami .nextval i .currval się nie zmieniła. *Trzeba wywołać .nextval dla sekwencji przed wywołaniem dla niej .currval w danej sesji.*



Uwaga

Z powodu zależności między dwoma wspomnianymi pseudokolumnami sekwencji zalecam ostrożność przy stosowaniu tej techniki.

Teraz można wstawiać wiersze do obu tabel za pomocą zmodyfikowanej sygnatury (ang. *override signature*). Takie sygnatury to listy wszystkich wymaganych i używanych opcjonalnych kolumn, których wartości programista chce wstawić do tabeli. Przy wstawianiu danych do dwóch przykładowych tabel trzeba się upewnić, że wartości kolumn customer_id są ze sobą zgodne. Dzięki temu możliwe są złączenia równościowe między tabelami customer i preference.

```
SQL> INSERT INTO customer (customer_name) VALUES ('Pan Nowak');
SQL> INSERT INTO preference (preference_name) VALUES ('Jasne mocne');
```

Oba wiersze są więc wstawiane bez bezpośrednio podanych wartości klucza sztucznego. Zobaczmy, czy baza Oracle Database 12c poprawnie radzi sobie z taką sytuacją. Proste zapytanie ze złączeniem:

```
SQL> SELECT *
 2 FROM customer c INNER JOIN preference p USING(customer_id);
```

powinno dać następujący wynik:

```
CUSTOMER_ID CUSTOMER_NAME PREFERENCE_ID PREFERENCE_NAME
-----
1 Pan Nowak                1 Jasne mocne
```

Wyniki są dowodem na to, że to podejście działa. Z dalszego punktu „Kolumny IDENTITY” dowiesz się, jak używać takich kolumn.

Wartości domyślne przy bezpośrednim wstawianiu null

Baza Oracle Database od dawna umożliwia podawanie wartości domyślnych dla dowolnej kolumny. Taką wartość można było jednak zmienić w wyniku bezpośredniego podania wartości null w instrukcji INSERT. W bazie Oracle Database 12c taką bezpośrednio podaną wartość null można zastępować wartością domyślną.

```
SQL> CREATE TABLE essay
 2 ( essay_name VARCHAR2(30) DEFAULT essay_s.nextval
 3 , essayist VARCHAR2(30)
 4 , published DATE DEFAULT TRUNC(SYSDATE)
 5 , text CLOB
 6 , CONSTRAINT essay_pk
 7 PRIMARY KEY (essay_name, essayist, published));
```

Wiersz 4. gwarantuje, że próby pominięcia danych w kolumnie published spowodują wstawienie bieżącej daty. W dodatku C (poświęconym funkcjom wbudowanym SQL-a) wyjaśniono, że funkcja TRUNC usuwa godziny i minuty z typów danych obejmujących datę i czas. Wszystkie typy danych DATE w bazach Oracle obejmują datę i czas.

Poniższa instrukcja INSERT wstawia wiersz do tabeli essay. Ta instrukcja działa tak samo w wersjach 11g i 12c bazy Oracle Database — wstawia bieżącą datę (bez godzin, minut i sekund) do kolumny published. Dzieje się tak, ponieważ kolumna published nie znajduje się na liście kolumn w zmodyfikowanej sygnaturze.

```
INSERT INTO essay
( essay_name
, essayist
, text )
VALUES
('Dlaczego chcę zostać Supermanem?'
,'21-SEP-2011'
,'W pewnym okresie swojego życia każdy chciałby stać się kimś innym...');
```

Jeśli do zmodyfikowanej sygnatury dodasz kolumnę published, będziesz mógł bezpośrednio wstawić wartość null. Powoduje ona zastąpienie standardowej wartości domyślnej. W wersjach starszych niż Oracle Database 12c nie można było zapobiec takiemu zastępowaniu wartości. W Oracle Database 12c dostępna jest instrukcja ON NULL, która pozwala uniemożliwić bezpośrednio wstawianie wartości null do kolumny.

Aby uzyskać ten efekt, należy wprowadzić następującą zmianę w wierszu 4. wcześniejszej instrukcji CREATE TABLE:

```
4 , published DATE DEFAULT ON NULL TRUNC(SYSDATE)
```

Instrukcja ON NULL powoduje, że w bazie Oracle Database 12c w kolumnie published nie można wstawić wartości null.

Kolumny IDENTITY

Spółeczność związana z bazami danych (czyli konkurencja) wskazywała na słabość baz Oracle, ponieważ nie obejmowały one kolumn IDENTITY (kolumn identyfikacyjnych). Takie kolumny obsługują automatyczne numerowanie wierszy i zwykle pełnią funkcję klucza sztucznego, tworzonego na podstawie sekwencji kolejnych wartości.

W bazie Oracle Database 12c dostępny jest operator tożsamościowy. Jeszcze lepszą wiadomością jest to, że ta wersja bazy umożliwia generowanie wartości identyfikacyjnych. Podstawowa kolumna IDENTITY zwykle ma nazwę id, a bazy Oracle obsługują tę konwencję (chyba że programista zmieni nazwę takiej kolumny).



W nazwach kolumn IDENTITY lepiej stosować przyrostek _id zamiast id.

Poniższy kod tworzy tabelę o dwóch kolumnach — identyfikacyjnej i tekstowej:

```
SQL> CREATE TABLE identity
  2 ( id NUMBER GENERATED ALWAYS AS IDENTITY
  3 , text VARCHAR2(10));
```

Ta przykładowa tabela pozwala pominąć kolumnę `id` w instrukcjach `INSERT`. Gdyby tabela obejmowała tylko jedną kolumnę, podanie wartości kolumny `id` byłoby konieczne. Łatwiej napisać zmodyfikowaną sygnaturę, która jest jednym z przykładów stosowania *notacji opartej na nazwach*. W takich sygnaturach należy podać *listę kolumn* między nazwą tabeli a klauzulą `VALUES` lub podzapytaniem.

Przykładowa tabela `identity` jest bardzo prosta, ponieważ domyślne ustawienie kolumny identyfikacyjnej to `ALWAYS`, co oznacza, że nie można ręcznie wprowadzać wartości kolumny `id`. Dlatego jeśli tabela nie obejmuje innych kolumn, nie można w niej wstawiać wierszy. Wstawianie wierszy do tabeli z kolumną identyfikacyjną jest możliwe tylko wtedy, gdy dana tabela ma więcej niż jedną kolumnę (tak jak w przykładzie).

Poprawny sposób wywoływania instrukcji `INSERT` polega na pominięciu kolumny `id` na liście kolumn:

```
SQL> INSERT INTO identity (text) VALUES ('Jeden');
```

Dlaczego firma Oracle wybrała ustawienie ALWAYS jako domyślne? W dokumentacji baz Oracle nie jest to wyjaśnione, jednak podejrzewam, że chodziło o następującą kwestię: jeśli programista użyje opcji `BY DEFAULT` i poda liczbę wyższą niż obecnie wygenerowana wartość sekwencji, może zduplikować wartość kolumny bez klucza głównego lub ograniczenia niepowtarzalności i spowodować nieudane wstawianie, jeśli tabela ma klucz główny lub ograniczenie niepowtarzalności.

W dodatku B znajduje się punkt „Kolumny IDENTITY”, w którym opisano, jak korzystać z takich kolumn. Tu warto wspomnieć, że należy zapoznać się z klauzulą `RETURNING INTO` instrukcji `INSERT`, ponieważ sekwencja wartości dla kolumn `IDENTITY` jest generowana przez system i trudno dostępna. Szczegółowe informacje znajdziesz w ramce „Wiązanie kolumn `IDENTITY` z sekwencjami” w dodatku B.

Kolumny `IDENTITY` zmieniają sposób pracy w bazach Oracle — przynajmniej wtedy, gdy nie musisz zapewniać obsługi starszego kodu, na przykład opartego na pakiecie Oracle E-Business Suite. Podejście zastosowane w bazie Oracle Database 12c sprawia, że można zaprzestać korzystania z pseudokolumn `sequence.nextval` i `sequence.currval`. Dzięki temu można zarządzać wartościami sztucznych kluczy głównych i zewnętrznych w zasięgu transakcji.

Kolumny `IDENTITY` wymagają stosowania klauzuli `RETURNING INTO` w instrukcjach `INSERT`. Pozwala to zapisać w zmiennej lokalnej ostatnią wartość sekwencji z instrukcji `INSERT`. Następnie można ponownie wykorzystać tę zmienną i przypisać ją jako klucz obcy do zależnej tabeli. Oczywiście oparte jest to na założeniu, że wstawianie wartości do tabel odbywa się w jednostce transakcyjnej w bloku języka PL/SQL.

Większa pojemność typów łańcuchowych i typu RAW

Maksymalną pojemność typów `VARCHAR2`, `NVARCHAR2` i `RAW` można teraz skonfigurować w SQL-u. Aby pozostawić limit 4000 bajtów, ustaw parametr `max_string_size` na wartość `STANDARD`. Inna możliwość to ustawienie tego parametru na wartość `EXTENDED`. Wtedy maksymalna pojemność tych typów to 32 767 bajtów.

Zaleta większego limitu pojemności tych typów powinna być oczywista dla programistów używających wcześniej bazy Oracle Database 11g. W owej wersji w języku PL/SQL typy `VARCHAR2`, `NVARCHAR2` i `RAW` miały pojemność 32 767 bajtów, jednak tak dużych wartości nie można było zapisać w kolumnach tych samych typów. Obecnie jest to możliwe.

Przekazywanie wartości z instrukcji SQL-a do zewnętrznych programów

W wersjach starszych niż Oracle Database 12c wyniki instrukcji `SELECT` trzeba było zapisywać za pomocą typu danych języka SQL lub PL/SQL. To wymagało dodatkowych kroków przy zagnieżdżaniu

zapytań w programach w języku PL/SQL. Aby w programach zewnętrznych uzyskać dostęp do wyników, trzeba było stosować pasujące skalarne lub złożone typy danych. Typami złożonymi były zwykle tabele SQL-a, systemowe kursory referencyjne języka PL/SQL lub zbiory wyników z potokowych funkcji SQL-a.

W Oracle Database 12c dostępna jest nowa procedura `return_result` z pakietu `dbms_sql`. Z tego punktu dowiesz się, jak korzystać z tej procedury i z tego pakietu.

Nowy mechanizm działa podobnie jak technologia Shared Source Common Language Infrastructure (CLI) Microsoftu. Według artykułu „The Microsoft Shared Source CLI Implementation” Davida Stutza (ten tekst został opublikowany w marcu 2002 roku w sieci MSDN) „Microsoft opracował technologię Shared Source CLI, aby naukowcy, studenci, profesorowie i inni zainteresowani programiści mogli poznawać zaawansowaną infrastrukturę języka komputerowego, uczyć jej innych i eksperymentować z nią”. Z tego samego artykułu wynika, że Microsoft udostępnia licencję na implementację technologii Shared Source CLI każdemu, kto obieca, że będzie modyfikował kod CLI tylko na potrzeby użytku niekomercyjnego. Jednak w 2009 roku Microsoft dodał język C# i technologię CLI do listy specyfikacji objętych programem Community Promise. To powinno oznaczać (choć przyznaję, że nie jestem prawnikiem), iż każdy może bezpiecznie implementować te technologie bez obaw o pozew patentowy ze strony Microsoftu.

W Wikipedii, pod poniższym adresem, znajduje się ciekawy artykuł na temat CLI:

http://en.wikipedia.org/wiki/Common_Language_Infrastructure

Ponieważ w dokumentacji baz Oracle nie ma żadnych informacji o kwestiach licencyjnych, wygląda na to, że firma Oracle musi ufać zapewnieniom Microsoftu z programu Community Promise lub w inny sposób rozwiązała kwestie korzystania z CLI. W CLI parametry funkcji podaje się tak:

```
CREATE FUNCTION mydb.getConquistador
(@nationality AS VARCHAR(30))
RETURNS TABLE
RETURN SELECT * FROM mydb.conquistador WHERE nationality = @nationality;
```

Funkcja zgodna z Shared Source CLI przekazuje referencję do zbioru wyników jako wartość zwracaną funkcji. W bazach Oracle stosuje się inne podejście. Tu używane są procedury `get_next_result` i `return_result` z pakietu `dbms_sql`. Przyjmują one parametry przekazywane przez referencję. Specyfikacje tych procedur znajdziesz w tabeli 2.1.

Tabela 2.1. Procedury z pakietu `dbms_sql` przekazujące niejawnie zbiory wyników

Procedura	Opis
<code>get_next_result</code>	Procedura <code>get_next_result</code> przyjmuje dwa parametry. Pierwszy to parametr typu IN przekazywany przez wartość. Jest nim referencja do kursora <code>dbms_sql</code> . Drugi to przeciążony parametr typu OUT przekazywany przez referencję. Przyjmuje albo jeden systemowy kursor referencyjny języka PL/SQL, albo referencję do takiego kursora. Programista nie może bezpośrednio używać parametru <code>rc</code> w trybie OUT. Oto prototypy tej procedury: <pre>GET_NEXT_RESULT(c IN INTEGER, rc OUT SYS_REFCURSOR) GET_NEXT_RESULT(c IN INTEGER, rc OUT INTEGER)</pre>
<code>return_result</code>	Procedura <code>return_result</code> przyjmuje dwa parametry. Pierwszy to przeciążony parametr typu IN OUT przekazywany przez referencję. Parametr <code>rc</code> to albo jeden systemowy kursor referencyjny języka PL/SQL, kolekcja takich kursorów, referencja do takiego kursora, albo kolekcja referencji do takich kursorów. Drugi to przekazywany przez wartość parametr logiczny o wartości domyślnej TRUE. Oto prototypy tej procedury: <pre>RETURN_RESULT(rc IN OUT SYS_REFCURSOR, to_client IN BOOLEAN [DEFAULT TRUE]) RETURN_RESULT(rc IN OUT INTEGER, to_client IN BOOLEAN [DEFAULT TRUE])</pre>

Poniżej przedstawiony jest program w postaci bloku anonimowego, ilustrujący zwracanie wyniku z kursora niejawnego:

```

SQL> COLUMN item_title      FORMAT A30
SQL> COLUMN item_subtitle  FORMAT A40
SQL> DECLARE
 2   /* Deklarowanie kursora. */
 3   lv_cursor SYS_REFCURSOR;
 4 BEGIN
 5   /* Otwieranie statycznego kursora. */
 6   OPEN lv_cursor FOR
 7     SELECT i.item_title
 8           , i.item_subtitle
 9     FROM item i
10    WHERE REGEXP_LIKE(i.item_title,'^Star.*');
11
12   /* Wywołanie procedury dbms_sql.return_result. */
13   dbms_sql.return_result(lv_cursor);
14 END;
15 /

```

W wierszu 3. zadeklarowany jest systemowy kursor referencyjny języka PL/SQL. W wierszach od 6. do 10. tworzone jest statyczne zapytanie powiązane z tym kursorem. W wierszu 13. ten lokalny kursor jest zwracany do klienta.

Przedstawiony blok anonimowy wyświetla poniższe informacje, ponieważ wyniki z kursora są niejawnie przekazywane przez referencję z powrotem do zasięgu wywołania:

ITEM_TITLE	ITEM_SUBTITLE
Star Wars I	Phantom Menace
Star Wars II	Attack of the Clones
Star Wars III	Revenge of the Sith

Gdy anonimowy blok obejmuje dwa systemowe kursory referencyjne (lub większą ich liczbę) i dwa wywołania procedury `return_result` z pakietu `dbms_sql` (lub większą ich liczbę), zwracane są przynajmniej dwa zbiory wyników. Procedura `get_next_result` zwraca jeden zbiór wyników.

Pojawiły się też nowe funkcje bibliotek zewnętrznych przeznaczone do pracy z niejawnymi zbiorami wyników. Na przykład w bibliotece OCI8 2.0 pojawiła się funkcja `oci_get_implicit_resultset()`. Można z niej korzystać razem z wszystkimi funkcjami z rodziny `oci_fetch_*`.

Dzięki temu pojawiają się ciekawe alternatywy do stosowania systemowych kursorów referencyjnych oraz potokowych i obiektowych funkcji tabelowych. Od wersji Oracle Database 10g można tworzyć obiektowe funkcje tabelowe oraz używać funkcji `TABLE` do zwracania kolekcji elementów typów skalarnych i złożonych z baz Oracle jako relacyjnych zbiorów wyników.

Określanie liczby wyników i zwracanych przedziałów wyników w natywnym SQL-u

W wersjach starszych od Oracle Database 12c można było tylko ograniczyć liczbę zwracanych wierszy za pomocą operacji `ROWNUM`. Nowe klauzule `FETCH FIRST` i `OFFSET` dają większe możliwości. W Oracle Database 12c dostępny jest rozbudowany zbiór opcji do pobierania pierwszych *n* wierszy.

Aby ograniczyć liczbę wyników zapytania do jednego wiersza, zastosuj następujący kod:

```

SQL> SELECT i.item_title
 2 FROM   item i
 3 FETCH FIRST 1 ROWS ONLY;

```

Wiersz 3. pokazuje, jak za pomocą klauzuli `FETCH FIRST` pobrać tylko jeden wiersz. Najzabawniejsze jest to, że trzeba tu zastosować słowa kluczowe w liczbie mnogiej — `ROWS ONLY` (czyli tylko wiersze).

Łatwo można się domyślić, że w celu zwrócenia pierwszych pięciu wierszy należy zmodyfikować wiersz 3. w następujący sposób:

```

3 FETCH FIRST 5 ROWS ONLY;

```

Załóżmy, że nie wiesz, ile wierszy zwróci zapytanie, a nie chcesz ograniczać liczby zwracanych wyników do 20, 50, 100 lub 500 (są to najczęściej stosowane wartości). Oracle udostępnią składnię, która

pozwała pobrać określoną część wszystkich wierszy. W tym celu należy dodać do klauzuli `FETCH FIRST` słowo kluczowe `PERCENT`, tak jak w poniższej wersji wiersza 3.:

```
3 FETCH FIRST 20 PERCENT ROWS ONLY;
```

Baza Oracle Database 12c umożliwia też pominięcie zbioru wierszy przed wczytaniem ich określonej liczby. W ten sposób pobieranych jest *n* wierszy od określonego miejsca zbioru wyników. Potrzebną składnię przedstawia zmodyfikowana wersja wiersza 3.:

```
3 OFFSET 20 ROWS FETCH FIRST 20 ROWS ONLY;
```

Minimalna wartość klauzuli `OFFSET` to 0. Warto o tym pamiętać przy podawaniu parametrów dla takich zapytań.

Do zastosowania parametrów w takiej instrukcji można wykorzystać zmienne związane:

```
3 OFFSET :bv_offset ROWS FETCH FIRST :bv_rows ROWS ONLY;
```

Jeśli chcesz zastosować parametry w zapytaniu pobierającym określoną liczbę wierszy, zawsze powinniś używać klauzuli `OFFSET`, ponieważ pozwala napisać jedną instrukcję mającą dwie funkcje. Jedną polega na odczycie danych od początku zbioru rekordów (wtedy klauzula `OFFSET` ma wartość 0). Druga pozwala wczytać dane od dowolnego miejsca innego niż początek zbioru. Dane są wczytywane do końca tylko wtedy, gdy wartość `:bv_rows` jest większa od liczby pozostałych rekordów.

Klauzule `FETCH FIRST` i `OFFSET` są też dostępne w blokach PL/SQL. Te klauzule można stosować w instrukcjach `SELECT INTO` i jako definicje kursorów statycznych. W anonimowych blokach PL/SQL można też korzystać ze zmiennych związanych.

Poniższy fragment pokazuje, jak zastosować zapytanie `SELECT INTO`.

```
SQL> DECLARE
 2   /* Deklarowanie zmiennej lokalnej. */
 3   lv_item_title VARCHAR2(60);
 4 BEGIN
 5   /* Pobieranie danych do zmiennej lokalnej. */
 6   SELECT i.item_title
 7   INTO lv_item_title
 8   FROM item i
 9   FETCH FIRST 1 ROWS ONLY;
10   dbms_output.put_line('||lv_item_title||');
11 END;
12 /
```

Wiersz 9. pobiera tylko pierwszy wiersz z zapytania. W wierszu 9. można też zastosować klauzulę `OFFSET`:

```
9   OFFSET 5 ROWS FETCH FIRST 1 ROWS ONLY;
```

Jak wspomniano, w anonimowych blokach PL/SQL można stosować zmienne związane. Jeśli wartość zmiennej `:bv_size` to 1, można wywołać następujący kod:

```
9 OFFSET :bv_offset ROWS FETCH FIRST :bv_size ROWS ONLY;
```

Zmienna `:bv_size` musi mieć wartość 1, ponieważ instrukcja `SELECT INTO` może zwrócić tylko jeden wiersz. Dlatego przy większej wartości wystąpi wyjątek `ORA-01422`, informujący, że zapytanie zwróciło zbyt wiele wierszy.

Za pomocą dynamicznego zapytania umieszczonego w zewnętrznym programie możesz wyeliminować ryzyko zwrócenia zbyt wielu wierszy. W tym celu należy wykorzystać biblioteki `ODBC` lub `JDBC`.

Poniższy fragment ilustruje technikę dynamicznego pobierania *n* wierszy w języku PHP:

```
15 // Deklaracja instrukcji SQL-a.
16 $sql = "SELECT i.item_title "
17       . "FROM item i "
18       . "OFFSET :bv_offset ROWS FETCH FIRST :bv_rows ROWS ONLY";
19
20 // Przygotowanie instrukcji i powiązanie dwóch łańcuchów znaków.
21 $stmt = oci_parse($c,$sql);
22
23 // Powiązanie zmiennych lokalnych z instrukcją języka PHP.
```

```

24 oci_bind_by_name($stmt, ":bv_offset", $offset);
25 oci_bind_by_name($stmt, ":bv_rows", $rows);
26
27 // Wykonanie instrukcji języka PL/SQL.
28 if (oci_execute($stmt)) {

```

Następny przykład pokazuje, jak pobrać *n* wierszy ze środka zbioru w statycznym kursorze:

```

SQL> DECLARE
  2  /* Deklaracja zmiennej lokalnej. */
  3  lv_item_title VARCHAR2(60);
  4  /* Deklaracja statycznego kursora. */
  5  CURSOR c IS
  6  SELECT i.item_title
  7  FROM item i
  8  OFFSET 10 ROWS FETCH FIRST 1 ROWS ONLY;
  9  BEGIN
 10  /* Otwieranie, pobieranie, wyświetlanie i zamykanie kursora. */
 11  OPEN c;
 12  FETCH c INTO lv_item_title;
 13  dbms_output.put_line('|||lv_item_title|||');
 14  CLOSE c;
 15  END;
 16 /

```

W wierszu 8. użyty jest literał do ustawienia wartości klauzuli `OFFSET` i liczby zwracanych wierszy. Nie można podstawiać zmiennych za literały (a przynajmniej nie w produkcyjnej wersji bazy Oracle Database 12c Release 1).

Oto próba użycia kursora dynamicznego:

```

SQL> DECLARE
  2  /* Deklaracja zmiennej lokalnej. */
  3  lv_item_title VARCHAR2(60);
  4  /* Deklaracja statycznego kursora. */
  5  CURSOR c
  6  ( cv_offset NUMBER
  7  , cv_size NUMBER ) IS
  8  SELECT i.item_title
  9  FROM item i
 10  OFFSET cv_offset ROWS FETCH FIRST cv_size ROWS ONLY;
 11  BEGIN
 12  NULL;
 13  END;
 14 /

```

W wierszu 10. ustawiane są ograniczenia zapytania zwracającego wybraną liczbę wierszy. Używane są do tego parametry kursora: `cv_offset` i `cv_size`. Wiersz 12. zapobiega błędowi parsowania. W bloku wykonania musi znajdować się instrukcja; w przeciwnym razie parsowanie zakończy się niepowodzeniem. Ten blok jednak i tak powoduje błąd parsowania. Występuje wyjątek i odłączenie programu od aktywnej sesji. Oto wyświetlany stos błędów:

```

ERROR:
ORA-03114: not connected to ORACLE

DECLARE
*
ERROR at line 1:
ORA-03113: end-of-file on communication channel
Process ID: 4148
Session ID: 31 Serial number: 3187

```

Jest to nieprzechwycony wyjątek. Takie wyjątki zwykle są szybko eliminowane. Z powodu takiego, a nie innego typu błędów podejrzewam, że Oracle w przyszłości rozwiąże ten problem. Możliwe też, że ten błąd zostanie udokumentowany jako ograniczenie. Do czasu opublikowania tej książki kwestia powinna zostać rozwiązana.

Sterownik bazy Oracle dla aplikacji dla baz MySQL

Baza Oracle Database 12c udostępnia sterownik dla aplikacji dla baz MySQL. Zastępuje on bibliotekę kliencką MySQL 5.5. Dzięki niemu można korzystać z aplikacji i narzędzi opartych na językach, które wykorzystują interfejs API w języku C bazy MySQL. Są to na przykład języki: PHP, Ruby, Perl i Python. Zaletą tego podejścia jest to, że użytkownicy mogą korzystać z aplikacji dla baz MySQL zarówno w takich bazach, jak i w bazach Oracle. Zwiększa to przenośność rozwiązań opartych na wymienionych językach skryptowych.

Instrukcje CROSS APPLY, OUTER APPLY i LATERAL w SQL-u

Instrukcja APPLY SQL-a umożliwia wywoływanie funkcji tabelowych (zwracających kolekcje, których można używać jak tabel) dla każdego wiersza zwróconego przez zewnętrzne wyrażenie tabelowe zapytania. Złączenie traktuje wtedy funkcję tabelową jako prawy operand, a zewnętrzne wyrażenie tabelowe jako lewy operand. Złączenie analizuje każdy wiersz z prawej strony dla każdego wiersza z lewej strony i łączy wyniki w ostateczny zbiór wyników.

Istnieją dwie odmiany tej operacji. Instrukcja CROSS APPLY to wersja złączenia wewnętrznego. Zwraca wiersze z tabeli (lub ze zbioru tabel) podanej po lewej stronie instrukcji CROSS APPLY pasujące do klauzuli WHERE zapisanej wewnątrz wierszowo po stronie prawej.

Poniżej pokazano przykładowy kod ze złączeniem CROSS APPLY:

```
SQL> SELECT i.item_title
2 FROM item i CROSS APPLY
3 (SELECT *
4 FROM rental_item ri
5 WHERE i.item_id = ri.item_id
6 OFFSET 0 ROWS FETCH FIRST 1 ROWS ONLY);
```

Instrukcja OUTER APPLY to odmiana złączenia lewostronnego. Tworzy ona złączenie zewnętrzne między tabelą lub zbiorem złączanych tabel a zapisanym wewnątrz wierszowo widokiem. Ten widok musi obejmować klauzulę WHERE, która określa relację między zbiorem wyników z lewej strony a podanym po prawej widokiem. Zwracane są wszystkie wiersze z tabeli podanej po lewej stronie złączenia i pasujące wyniki z kolekcji lub wartości null.

Oto przykładowy kod, w którym zastosowano złączenie OUTER APPLY:

```
SQL> SELECT i.item_title
2 FROM item i OUTER APPLY
3 (SELECT *
4 FROM rental_item ri
5 WHERE i.item_id = ri.item_id
6 OFFSET 0 ROWS FETCH FIRST 1 ROWS ONLY);
```

Klauzula LATERAL służy do podawania podzapytań w lateralnych widokach wewnątrz wierszowych. W klauzuli FROM zapytania można podać tabele, z którymi korelowany jest dany lateralny widok wewnątrz wierszowy. Przy stosowaniu takich widoków obowiązują pewne ograniczenia:

- Nie można stosować klauzul PIVOT i UNPIVOT ani klauzul z referencjami tabel.
- Nie można stosować lewostronnej korelacji, gdy lateralny widok wewnątrz wierszowy zawiera klauzulę PARTITION w zapytaniu i pojawia się po prawej stronie złączenia.
- W widoku lateralnym nie można stosować lewostronnej korelacji dla pierwszej tabeli ze złączenia prawostronnego lub pełnego złączenia zewnętrznego.

Klauzula LATERAL jest częścią standardu ANSI SQL i stanowi rozszerzenie składni widoków wewnątrz wierszowych baz Oracle. Choć poniższe zapytanie można łatwo przekształcić na złączenie INNER JOIN, ta wersja pozwala zademonstrować ograniczenia wyeliminowane dzięki klauzuli LATERAL z bazy Oracle Database 12c.

```
SQL> SELECT *
2 FROM contact c CROSS JOIN
3 (SELECT *
```



```

4      FROM member m
5      WHERE c.member_id = m.member_id);

```

To zapytanie próbuje zapisać widok wewnątrzwerszowy obejmujący skorelowane podzapytanie. W efekcie generowany jest następujący komunikat o błędzie:

```

WHERE c.member_id = m.member_id)
*
```

```

ERROR at line 5:
ORA-00904: "C"."MEMBER_ID": invalid identifier

```

Ten błąd oznacza, że nie można znaleźć aliasu c tabeli contact. Widok wewnątrzwerszowy nie potrafi znaleźć tego aliasu, ponieważ staje się on dostępny dopiero po zakończeniu parsowania klauzuli FROM. To dlatego Oracle zgłasza błąd nieprawidłowego identyfikatora (szczegółowe omówienie identyfikatorów znajdziesz w rozdziale 4.). Błąd tego samego rodzaju występuje w złączeniach CROSS APPLY i OUTER APPLY.

Klauzula LATERAL umożliwia korzystanie w widoku wewnątrzwerszowym z tabel podanych po lewej stronie instrukcji CROSS JOIN. Jest tak, ponieważ cały kod do słowa kluczowego LATERAL jest parsowany osobno. Rozdzielenie parsowania na dwa etapy pozwala widokowi wewnątrzwerszowemu umieszczonemu z prawej strony słowa kluczowego LATERAL na poprawne zinterpretowanie identyfikatora. To oznacza, że teraz w widokach wewnątrzwerszowych można stosować korelacje:

```

SQL> SELECT *
2 FROM contact c CROSS JOIN
3     LATERAL(SELECT *
4              FROM member m
5              WHERE c.member_id = m.member_id);

```

Słowo kluczowe LATERAL z wiersza 3. pozwala podzapytaniu znaleźć tabele podane po lewej stronie instrukcji CROSS JOIN. Operacje są tu wykonywane od lewej do prawej, dlatego ta technika nie zadziała, jeśli niezrozumiały identyfikator znajduje się po prawej stronie.

Widoki z ustawieniem BEQUEATH CURRENT_USER

W wersjach starszych niż Oracle Database 12c widoki zawsze działały z uprawnieniami jednostki definiującej. Takie uprawnienia są stosowane domyślnie dla funkcji, procedur, pakietów i typów. Przy definiowaniu jednostek programów składowanych można zastosować klauzulę AUTHID DEFINER, przy czym nie jest to konieczne, ponieważ to ustawienie jest domyślne.

W Oracle Database 12c pojawiła się możliwość ustawiania uprawnień widoków. Domyślne ustawienie to BEQUEATH DEFINER, które działa jak opcja AUTHID DEFINER stosowana do jednostek programów składowanych. Aby zmienić domyślne uprawnienia, należy zastosować ustawienie BEQUEATH CURRENT_USER.

Powtórzenie materiału

W tym podrozdziale przedstawiono następujące informacje na temat nowych funkcji SQL-a dostępnych w bazie Oracle Database 12c:

- W Oracle Database 12c oprócz wskazywania zmiennych środowiskowych w instrukcjach LIBRARY można też stosować katalogi wirtualne.
- Oracle Database 12c umożliwia bezpośrednie i pośrednie definiowanie wymiarów VT, co zwiększa możliwości administratorów w zakresie analiz retrospektywnych.
- W Oracle Database 12c wzbogacono możliwości złączeń LEFT OUTER JOIN. Obecnie mogą one obejmować wiele tabel po lewej stronie.
- W Oracle Database 12c wprowadzono instrukcje CROSS APPLY, OUTER APPLY i LATERAL, przeznaczone do pracy z zagnieżdżonymi tabelami.
- Oracle Database 12c udostępnia domyślne kolumny przechowujące wartości pseudokolumn .nextval i .currval dla nazwanych sekwencji.

- W Oracle Database 12c wprowadzono kolumny IDENTITY, zawierające automatycznie zwiększane sekwencje liczb. Takie kolumny pełnią funkcję klucza sztucznego.
- W Oracle Database 12c do wartości domyślnych można dodać klauzulę ON NULL, co uniemożliwia bezpośrednie ustawianie wartości null przy wstawianiu lub aktualizowaniu wierszy tabeli.
- Oracle Database 12c pozwala ustawić parametr w celu zwiększenia pojemności typów VARCHAR2, NVARCHAR2 i RAW do 32 767 bajtów, co odpowiada ich pojemności w języku PL/SQL.
- Oracle Database 12c udostępnia słowo kluczowe BEQUEATH, służące do ustawiania dla widoków uprawnień jednostki definiującej lub wywołującej.

Nowe funkcje języka PL/SQL

W Oracle Database 12c do języka PL/SQL dodano liczne nowe funkcje:

- Można zapisywać w pamięci podręcznej wyniki funkcji z uprawnieniami jednostki wywołującej.
- Ważne usprawnienie pozwala tworzyć białe listy jednostek wywołujących dla funkcji składowanych, procedur, pakietów i typów.
- Dostępna jest natywna obsługa wiązania pakietów PL/SQL i typów logicznych jako parametrów. Pojawiła się też natywna obsługa klienckich interfejsów API w zakresie typów danych języka PL/SQL.
- Dodany został pakiet `utl_call_stack`.
- W pakiecie `dbms_utility` znalazła się nowa procedura `expand_sql_text`.
- Procedura `parse` ma nowy formalny schemat określania niepełnych nazw obiektów.
- W klauzuli `WITH SQL-a` można stosować funkcje języka PL/SQL.
- Można definiować lokalne typy języka PL/SQL i stosować je w zagnieżdżonych instrukcjach SQL-a.
- Dostawca ODP.NET (ang. *Oracle Data Provider for .NET*) może teraz wiązać parametry typu `REF CURSOR` z procedurami składowanymi.

W dalszych punktach znajdziesz omówienie wszystkich tych nowych funkcji języka PL/SQL.

Zapisywanie wyników funkcji z uprawnieniami jednostki wywołującej

Oracle Database 12c umożliwia zapisanie w pamięci podręcznej wyników funkcji z uprawnieniami jednostki wywołującej. Do zapisywanych wyników dołączana jest tożsamość bieżącego użytkownika. To pozwala zapisać różne wyniki dla jednego programu z uprawnieniami jednostki wywołującej. To oznacza, że można zapisywać w pamięci podręcznej wyniki deterministycznych funkcji z uprawnieniami jednostki wywołującej, czyli funkcji korzystających z wartości `CURRENT_USER`.

Wprowadzenie funkcji z uprawnieniami jednostki wywołującej zmienia sposób rozwiązywania problemów. W środowisku rozproszonym (na przykład z bazami PDB) takie funkcje pozwalają uzyskać wyższą przepustowość.

Tworzenie białych list jednostek wywołujących w programach języka PL/SQL

Oracle Database 12c umożliwia przechowywanie białych list użytkowników, którzy mogą wywoływać funkcje, procedury, pakiety i typy obiektowe. Umieszczenie użytkownika na białej liście umożliwi mu wywoływanie procedur składowanych. Jest to uzupełnienie innych zabezpieczeń. Użytkownik, który ma uprawnienia do wykonywania procedury składowanej ze schematu, musi też znajdować się na liście uprawnionych użytkowników.

W dokumentacji bazy Oracle Database 12c przedstawiono nowy sposób opisywania procedur składowanych. Ogólne określenie rodzaj_jednostki oznacza tu funkcje, procedury, pakiety i typy obiektowe. Klauzula ACCESSIBLE BY służy do określania białych list dla tworzonych lub zastępowanych programów.

W dokumentacji bazy Oracle przedstawiony jest następujący prototyp:

```
[ACCESSIBLE BY (rodzaj_jednostki [schema.]nazwa_jednostki
[, rodzaj_jednostki [schema.]nazwa_jednostki]
[,... ])]
```

Jest to bezpośredni i krótki zapis, jednak bardziej zrozumiały może okazać się rozszerzony prototyp, ponieważ słowo kluczowe zastępujące określenie rodzaj_jednostki poprzedza w nim nazwę programu składowanego:

```
[ACCESSIBLE BY
( [{FUNCTION | PROCEDURE | PACKAGE | TYPE}] [schema.]nazwa_jednostki)
[, [{FUNCTION | PROCEDURE | PACKAGE | TYPE}] [schema.]nazwa_jednostki)]
[,... ]]
```

Poniższy krótki przykład ilustruje, jak napisać funkcję dodającą jednostki do białej listy. Ta funkcja umieszcza na białej liście funkcję, procedurę, pakiet i typ, co pozwala przedstawić kompletny opis tego mechanizmu.

```
SQL> CREATE OR REPLACE FUNCTION library
  2 ( pv_message VARCHAR2 ) RETURN VARCHAR2
  3 ACCESSIBLE BY
  4 ( FUNCTION video.gateway
  5 , PROCEDURE video.backdoor
  6 , PACKAGE video.api
  7 , TYPE video.hobbit ) IS
  8 lv_message VARCHAR2(20) := 'Witaj, ';
  9 BEGIN
 10 lv_message := lv_message || pv_message || '!';
 11 RETURN lv_message;
 12 END;
 13 /
```

W wierszach od 3. do 7. zadeklarowana jest lista uprawnionych jednostek wywołujących. Każda z nich może z powodzeniem wywołać funkcję biblioteczną, natomiast nie mogą tego zrobić żadne inne funkcje, procedury, pakiety ani typy. Próba utworzenia nowej funkcji wywołującej funkcję z ustawioną białą listą, na przykład:

```
SQL> CREATE OR REPLACE FUNCTION black_knight
  2 ( pv_message VARCHAR2 ) RETURN VARCHAR2 IS
  3 BEGIN
  4 RETURN library(pv_message);
  5 END;
  6 /
```

spowoduje błąd kompilacji. Można wtedy wyświetlić informacje o błędzie:

```
SQL> show errors
Errors for FUNCTION BLACK_KNIGHT:
LINE/COL ERROR
-----
4/3      PL/SQL: Statement ignored
4/10     PLS-00904: insufficient privilege to access object LIBRARY
```

Tworzenie białych list jednostek wywołujących to przydatne i od dawna oczekiwane usprawnienie, niedostępne na razie w żadnej innej bazie danych.

Natywna obsługa klienckich interfejsów API w zakresie typów języka PL/SQL

Ta funkcja umożliwia klienckim interfejsom API baz Oracle opisywanie i wiązanie typów z pakietów języka PL/SQL i typów logicznych. Używane są do tego interfejsy API OCI i JDBC. Do wiązania i wykonywania funkcji oraz procedur języka PL/SQL można też używać aplikacji opartych na języku C.

Nowy pakiet utl_call_stack języka PL/SQL

W Oracle Database 12c pojawił się pakiet `utl_call_stack`. Udostępnia on liczne funkcje usprawniające obsługę stosu błędów. *Stos błędów* to sekwencja zgłoszonych wyjątków przekazanych w górę łańcucha wywołań. Zawartość tego pakietu i sposób korzystania z niego opisano w rozdziale 6.

Nowa procedura `expand_sql_text` w pakiecie `dbms_utility`

W Oracle Database 12c w pakiecie `dbms_utility` dostępna jest nowa procedura `expand_sql_text`. Ta procedura umożliwia przekształcenie widoku zależnego od innych widoków w jedno zapytanie. Jest to bardzo przydatne, gdy chcesz zobaczyć kompletny obraz działania kodu.

Procedurę `expand_sql_text` można wykorzystać do ustalenia, jak widoki oparte na innych widokach są powiązane z tabelami. Jest to najprostsze rozwiązanie oprócz ręcznego refaktoryzowania kodu widok po widoku. Problem z procedurą `expand_sql_text` w Oracle polega na tym, że przyjmuje ona obiekt CLOB i zwraca taki obiekt, natomiast widoki są przechowywane w kolumnach typu LONG. Przekształcenie typu LONG na CLOB nie jest łatwe. Dlatego napisałem funkcję, która zrobi to za Ciebie. Tę funkcję, `long_to_clob`, znajdziesz w rozdziale 10.

Nawet z funkcją `long_to_clob` używanie procedury `expand_sql_text` wymaga kilku kroków, co ilustruje poniższa funkcja:

```
SQL> CREATE OR REPLACE FUNCTION expand_view
 2 ( pv_view_name    VARCHAR2 ) RETURN CLOB IS
 3
 4   /* Deklaracje kontenerów na widoki. */
 5   lv_input_view   CLOB;
 6   lv_output_view  CLOB;
 7
 8   /* Deklaracja docelowej zmiennej (z powodu ograniczeń instrukcji SELECT INTO). */
 9   lv_long_view    LONG;
10
11  /* Deklaracja dynamicznego kursora. */
12  CURSOR c (cv_view_name VARCHAR2) IS
13    SELECT text
14    FROM   user_views
15    WHERE  view_name = cv_view_name;
16
17  BEGIN
18    /* Otwieranie, pobieranie i zamykanie kursora w celu pobrania tekstu widoku. */
19    OPEN c(pv_view_name);
20    FETCH c INTO lv_long_view;
21    CLOSE c;
22
23    /* Przekształcanie typu LONG na CLOB. */
24    lv_input_view := long_to_clob(pv_view_name, LENGTH(lv_long_view));
25
26    /* Przesyłanie tekstu widoku i pobieranie kompletnego tekstu. */
27    dbms_utility.expand_sql_text(lv_input_view, lv_output_view);
28
29    /* Zwracanie wyjściowej wartości CLOB. */
30    RETURN lv_output_view;
31  END;
32 /
```

Choć nie lubię stosować typu danych LONG (jest to prawdziwy „dinozaur” w bazie Oracle), jest on potrzebny do zilustrowania omawianej tu ciekawej funkcji. W wierszu 9. znajduje się deklaracja zmiennej `lv_long_view` typu LONG. Choć używanie kursora z parametrami jest tu pewną przesadą, warto konsekwentnie stosować się do dobrych praktyk. Nie można wykorzystać tu instrukcji `SELECT INTO`, ponieważ nie współdziała ona z typem LONG. Klauzula `FETCH INTO` pozwala na zastosowanie typu LONG, dlatego wykorzystano ją w przypisaniu w wierszu 20.

Dalej znajduje się wywołanie funkcji `long_to_clob` z parametrem `pv_view_name` i długością kolumny tekstowej widoku. Ten kod zgłasza dwa zapytania do katalogu (ponieważ funkcja `long_to_clob` ponawia takie zapytanie), jest to jednak konieczne, aby uniknąć przepisywania danych znak po znaku z typu `LONG` do typu `CLOB`. Oracle nie udostępnia wielu możliwości w zakresie pracy z typem `LONG`. Na przykład funkcja `to_clob` nie przyjmuje parametrów typu `LONG`.

Pełne omówienie funkcji `long_to_clob` znajdziesz w rozdziale 10. Tu wystarczy zauważyć, że wykorzystuje ona pakiety `dbms_sql` i `dbms_lob` do przekształcenia danych typu `LONG` na typ `CLOB`. Więcej informacji o pakiecie `dbms_sql` zawiera rozdział 13. W rozdziale 10. opisano pakiet `dbms_lob` i pracę z dużymi obiektami.

Wiersz 27. to wywołanie procedury `expand_sql_text`, a w wierszu 30. zwracany jest wyjściowy obiekt `CLOB` z procedury `expand_sql_text`. Ostatecznie funkcja zwraca obiekt `CLOB` z kompletnym zapytaniem opartym na tabelach. Po uzyskaniu tego zapytania można przeanalizować jego wydajność.

Nowy formalny schemat procedury parse w pakiecie dbms_sql

Pakiet `dbms_sql` obejmuje nowy formalny schemat procedury `parse`. Ta procedura obecnie interpretuje niepełne nazwy obiektów. Dzięki temu jednostka programu z uprawnieniami jednostki definiującej może sterować określeniem nazw w uruchamianych dynamicznych instrukcjach SQL-a. Za pomocą procedury `dbms_sql.parse` możesz na przykład wywołać polecenie `DROP TABLE` w procedurze składawanej.

Funkcje języka PL/SQL w klauzuli WITH SQL-a

W Oracle Database 12c można wywoływać funkcje języka PL/SQL w klauzuli `WITH`. Jedyny problem związany jest z uruchamianiem takich funkcji, ponieważ obejmują średniki. Założmy, że chcesz wywołać instrukcję w środowisku SQL*Plus. Najpierw trzeba wyłączyć domyślny symbol kończący SQL-a (średnik — ;). Służy do tego następujące polecenie tego środowiska:

```
SET SQLTERMINATOR OFF
```

Następnie możesz utworzyć funkcję lokalną w instrukcji `WITH`:

```
SQL> COLUMN person FORMAT A18
SQL> WITH
 2  FUNCTION glue
 3  ( pv_first_name VARCHAR2
 4  , pv_last_name  VARCHAR2) RETURN VARCHAR2 IS
 5  lv_full_name   VARCHAR2(100);
 6  BEGIN
 7  lv_full_name := pv_first_name || ' ' || pv_last_name;
 8  RETURN lv_full_name;
 9  END;
10  SELECT glue(a.first_name,a.last_name) AS person
11  FROM actor a
12 /
```

Funkcja z wierszy od 2. do 9. łączy dwa łańcuchy znaków i wstawia między nie jedną spację. Średniki są traktowane w zapytaniu jak zwykle znaki, ponieważ wyłączono domyślny symbol kończący SQL-a. Warto też zauważyć, że w środowisku SQL*Plus instrukcje SQL-a są uruchamiane za pomocą ukośnika i że kompletna instrukcja nie ma kończącego średnika w wierszu 11.

W tym prostym przykładzie tabela `actor` zawiera nazwiska dwóch aktorów (z filmu *Iron Man*), dlatego zapytanie zwraca następujące dane:

```
PERSON
-----
Robert Downey
Gwyneth Paltrow
```

Gdy otworzysz takie zapytanie w narzędziu Oracle SQL Developer lub podobnym, mogą wystąpić pewne problemy z parsowaniem przy uruchamianiu zapytania. Najłatwiejsze rozwiązanie polega na umieszczeniu zapytania w widoku, ponieważ widok eliminuje konieczność zmiany ustawienia SQLTERMINATOR w czasie wykonywania programu. Poniższy kod tworzy widok z funkcją języka PL/SQL zagnieżdżoną w instrukcji WITH:

```
SQL> CREATE OR REPLACE VIEW actor_v AS
2 WITH
3   FUNCTION glue
4   ( pv_first_name VARCHAR2
5   , pv_last_name VARCHAR2) RETURN VARCHAR2 IS
6   BEGIN
7     RETURN pv_first_name || ' ' || pv_last_name;
8   END;
9   SELECT glue(a.first_name,a.last_name) AS person
10  FROM actor a
11 /
```

Widok to nic innego jak składowane zapytanie. Widok actor_v pozwala skrócić funkcję glue o dwa wiersze. Można pominąć deklarację lv_full_name i zastąpić przypisywanie złączanych wartości do zmiennej instrukcją, która bezpośrednio je zwraca (wiersz 7.).

Jeśli zechcesz uruchomić zwykłe polecenia SQL-a, z domyślnym średnikiem, ponownie włącz domyślny symbol kończący SQL-a:

```
SET SQLTERMINATOR ON
```

Oczywistą zaletą klauzuli WITH jest to, że jest uruchamiana raz, po czym można ją wielokrotnie wykorzystać w zasięgu zapytania. Można też zagnieżdżyć funkcje o zasięgu lokalnym ograniczonym do jednego zapytania. Po co stosować klauzulę WITH, skoro można wykorzystać globalną tabelę tymczasową? Tom Kyte odpowiedział na to pytanie w dziale *Ask Tom* (<http://asktom.oracle.com>), gdzie napisał, że optymalizator potrafi scalić klauzulę WITH z resztą instrukcji, natomiast w przypadku globalnej tabeli tymczasowej jest to niemożliwe.

Stosowanie w SQL-u typów danych specyficznych dla języka PL/SQL

Możliwość przekazywania specyficznych dla baz Oracle typów danych z języka PL/SQL to cenna funkcja bazy Oracle Database 12c. Aby to zadziałało, potrzebna jest pewna sztuczka — należy zadeklarować zmienną lokalną w programie składowanym, a następnie wykorzystać tę zmienną w zagnieżdżonej instrukcji SQL-a.

Przyjrzyj się działaniu tej techniki na przykładzie kolekcji języka PL/SQL oraz nazwanego i anonimowego bloku tego języka. Zademonstrowanie tego podejścia wymaga pięciu kroków. W szóstym kroku zobaczysz, w jakiej sytuacji pojawiają się problemy, i jednocześnie zrozumiesz, dlaczego w Oracle Database 12c potokowe funkcje tabelowe nadal są potrzebne.

Pierwszy krok to utworzenie pozbawionego ciała pakietu type_defs. Specyfikacja *pakietu pozbawionego ciała* obejmuje tylko definicje typu i kursora. Takie pakiety stosuje się w celu współużytkownia typów i kursorów między różnymi jednostkami programu.

W poniższej specyfikacji pakietu tworzona jest tylko tablica asocjacyjna specyficzna dla języka PL/SQL (jest to kolekcja z rzadkimi indeksami):

```
SQL> CREATE OR REPLACE PACKAGE type_defs IS
2   TYPE psql_table IS TABLE OF VARCHAR2(20)
3   INDEX BY BINARY_INTEGER;
4 END type_defs;
5 /
```

W drugim kroku tworzona jest tabela honeymooner, z kolumną identyfikacyjną i kolumną person. Kolumna person zawiera dane typu używanego we wspomnianej wcześniej tablicy asocjacyjnej. Dane z tej tabeli posłużą do zapełnienia tablicy asocjacyjnej z języka PL/SQL.

Oto definicja tej tabeli:

```
SQL> CREATE TABLE honeymooner
  2 ( honeymooner_id NUMBER GENERATED ALWAYS AS IDENTITY
  3 , person VARCHAR2(20));
```

Trzeci krok wymaga wstawienia do tabeli honeymooner czterech wierszy:

```
SQL> INSERT INTO honeymooner (person) VALUES ('Ralph Kramden');
SQL> INSERT INTO honeymooner (person) VALUES ('Alice Kramden');
SQL> INSERT INTO honeymooner (person) VALUES ('Edward Norton');
SQL> INSERT INTO honeymooner (person) VALUES ('TheIma Norton');
```

Trzy pierwsze kroki tworzą pozbawiony ciała pakiet `plsql_table` i tabelę `honeymooner` oraz zapełniają tę tabelę czterema wierszami. Czwarty krok wymaga utworzenia funkcji `implicit_convert`, która wczytuje cztery dodane wcześniej wiersze z tabeli i zapisuje je w tablicy asocjacyjnej języka PL/SQL, a następnie zwraca tę tablicę:

```
SQL> CREATE OR REPLACE FUNCTION implicit_convert
  2 RETURN type_defs.plsql_table IS
  3 lv_index NUMBER := 1; -- Zmienna licznika.
  4 lv_list TYPE_DEFS.PLSQL_TABLE; -- Zmienna kolekcji.
  5 CURSOR c IS SELECT person FROM honeymooners;
  6 BEGIN
  7 FOR i IN c LOOP
  8 lv_list(lv_index) := i.person;
  9 lv_index := lv_index + 1;
 10 END LOOP;
 11 RETURN lv_list; -- Zwracanie kolekcji języka PL/SQL o zasięgu lokalnym.
 12 END;
 13 /
```

W wierszu 2. jako typ zwracanej wartości ustawiona jest tablica asocjacyjna języka PL/SQL. W wierszu 4. znajduje się deklaracja zmiennej lokalnej tego samego typu. Pętla zapełnia tę tablicę, a w wierszu 11. zwracana jest zmienna lokalna zawierająca tę tablicę.

W piątym kroku należy zaimplementować blok anonimowy wywołujący funkcję `implicit_convert`. W instrukcjach tego bloku *lokalna* tablica asocjacyjna języka PL/SQL jest przekazywana do instrukcji SQL-a, która wczytuje tę tablicę za pomocą funkcji `TABLE`.

Oto ten blok anonimowy:

```
SQL> DECLARE
  2 list TYPE_DEFS.PLSQL_TABLE;
  3 BEGIN
  4 list := implicit_convert;
  5 FOR i IN (SELECT column_value
  6 FROM TABLE(list)) LOOP
  7 dbms_output.put_line(i.column_value);
  8 END LOOP;
  9 END;
 10 /
```

W wierszu 2. zadeklarowana jest zmienna typu `plsql_table` z pakietu `type_defs`. Wiersz 4. zawiera wywołanie funkcji `implicit_convert` i przypisanie zwróconej tablicy asocjacyjnej języka PL/SQL do zmiennej lokalnej. Wiersze 5. i 6. obejmują instrukcję `SELECT`, która wykorzystuje lokalnie zadeklarowaną zmienną języka PL/SQL w funkcji `TABLE`.

W wersjach starszych niż Oracle Database 12c funkcja `TABLE` potrafiła tylko przekształcać *tablice* `VARRAY` lub *tabele zagnieżdżone* na zbiór wyników SQL-a. Obecnie funkcja `TABLE` potrafi przekształcić lokalną zmienną z tablicą asocjacyjną języka PL/SQL w zasięgu SQL-a.

Jeśli umieszysz wiersz 4. (ze zmienną *lokalną*) w komentarzu i zastąpisz zmienną lokalną wywołaniem funkcji `implicit_convert` w wierszu 6., wystąpi błąd. Oto te zmiany:

```
4 -- list := implicit_convert;
5 FOR i IN (SELECT column_value
6 FROM TABLE(implicit_convert)) LOOP
```

Te zmiany prowadzą do następującego stosu błędów:

```
FROM TABLE(implicit_convert)) LOOP
      *
ERROR at line 6:
ORA-06550: line 6, column 28:
PLS-00382: expression is of wrong type
ORA-06550: line 6, column 22:
PL/SQL: ORA-22905: cannot access rows from a non-nested table item
ORA-06550: line 5, column 13:
PL/SQL: SQL Statement ignored
ORA-06550: line 7, column 26:
PLS-00364: loop index variable 'I' use is invalid
ORA-06550: line 7, column 5:
PL/SQL: Statement ignored
```

Jest też dobra wiadomość — można przekształcić tablicę asocjacyjną języka PL/SQL przez opakowanie jej w potokową funkcję tabelową. Takie funkcje nadal są potrzebne w języku PL/SQL. Założmy, że chcesz usunąć pozbawiony ciała pakiet, w którym zdefiniowałeś tablicę asocjacyjną języka PL/SQL. Wymaga to zmodyfikowania kodu w bloku anonimowym:

```
SQL> DECLARE
2  TYPE local_table IS TABLE OF VARCHAR2(20)
3  INDEX BY BINARY_INTEGER;
4  lv_index NUMBER := 1; -- Counter variable.
5  lv_list LOCAL TABLE; -- Local PL/SQL collection.
6  CURSOR c IS SELECT person FROM honeymooners;
7  BEGIN
8  FOR i IN c LOOP
9  lv_list(lv_index) := i.person;
10 lv_index := lv_index + 1;
11 END LOOP;
12 FOR i IN (SELECT column_value
13 FROM TABLE(lv_list)) LOOP
14 dbms_output.put_line(i.column_value);
15 END LOOP;
16 END;
17 /
```

Ten blok także nie zadziała, ale nie, jak miało to miejsce wcześniej, z powodu próby przetworzenia tablicy asocjacyjnej zwróconej przez funkcję języka PL/SQL (choć na podstawie stosu błędów można uznać, że w obu sytuacjach przyczyną problemów jest taka sama). Tym razem źródłem błędów jest to, że typ języka PL/SQL nie jest zdefiniowany w katalogu bazy danych i baza Oracle nie potrafi go znaleźć. To oznacza, że baza nie wie, co ma przekształcić na analogiczny typ SQL-a.

Choć firma Oracle nie wyjaśnia szczegółów procesu przekształcania typów, można spróbować domyślić się, jak on przebiega. Zgaduję, że baza Oracle odwzorowuje niejawną kolekcję z języka PL/SQL na jawną tabelę zagnieżdżoną SQL-a. Jeśli na tym etapie książki ten opis wydaje Ci się zbyt techniczny, nie martw się. W rozdziale 6. znajdziesz szczegółowe omówienie złożonych typów danych (kolekcji).

Oto krótkie podsumowanie: można przypisać *lokalną* zmienną języka PL/SQL do lokalnego kontekstu SQL-a. Jednak obecnie nie można przypisać w ten sposób *nie-lokalnej* tablicy asocjacyjnej języka PL/SQL zwróconej przez funkcję.

Niejawne wiązanie parametrów REF CURSOR

Dostawca ODP.NET potrafi teraz niejawnie wiązać parametry REF CURSOR procedur składowanych. Dzieje się tak, gdy programista poda metadane w plikach konfiguracyjnych platformy .NET.

Powtórzenie materiału

W tym podrozdziale przedstawiono następujące zagadnienia związane z nowymi funkcjami języka PL/SQL w bazie Oracle Database 12c:

- Oracle Database 12c umożliwia zapisywanie w pamięci podręcznej wyników funkcji o uprawnieniach jednostki wywołującej.
- Oracle Database 12c umożliwia tworzenie białych list z jednostkami wywołującymi funkcje składowane, procedury, pakiety i typy obiektowe.
- Oracle Database 12c udostępnia natywną obsługę klienckich interfejsów API w zakresie typów danych języka PL/SQL.
- Oracle Database 12c udostępnia nowy mechanizm zarządzania stosem błędów — pakiet `utl_call_stack`.
- Oracle Database 12c umożliwia rozwinięcie kodu widoków zależnych od innych widoków. Służy do tego procedura `expand_sql_text` z pakietu `dbms_utility`.
- Pakiet `dbms_sql` obejmuje nowy formalny schemat, który pozwala określać niepełne nazwy obiektów.
- Oracle Database 12c obsługuje zagnieżdżanie funkcji języka PL/SQL w instrukcjach w klauzuli `WITH SQL-a`.
- Oracle Database 12c umożliwia stosowanie w lokalnych instrukcjach SQL-a lokalnych zmiennych o typach danych z języka PL/SQL.
- Oracle Database 12c obsługuje w dostawcy ODP.NET niejawne wiązanie typu danych `REF CURSOR` z języka PL/SQL.

Skrypty pomocnicze

W tym podrozdziale opisane są powiązane z tą książką programy dostępne w witrynach wydawnictw McGraw-Hill Professional i Helion.

- Plik `dynamic_topnquery.php` zawiera kompletny przykładowy program, którego fragmenty przedstawiono w tym rozdziale.
- Plik `white_list.sql` obejmuje wszystkie funkcje, procedury, pakiety i typy potrzebne w pokazanych w tym rozdziale przykładach ilustrujących białe listy.
- Plik `expanding_view.sql` zawiera funkcje potrzebne do przekształcania typu `LONG` na typ `CLOB` i wywoływania omówionej w tym rozdziale procedury `dbms_utility.expand_sql_text`.

Podsumowanie

W tym rozdziale zapoznałeś się z nowymi funkcjami bazy Oracle Database 12c. W dalszych rozdziałach także znajdziesz informacje o różnicach między starszymi wersjami baz Oracle a bazą Oracle Database 12c.

Test wiedzy

Test wiedzy to zestaw pytań typu „prawda czy fałsz” i wielokrotnego wyboru, dzięki którym sprawdzisz, jak dobrze opanowałeś materiał z poszczególnych rozdziałów. Odpowiedzi na pytania znajdziesz w dodatku I.

Prawda czy fałsz?

1. `__` Wymiar VT określa moment zatwierdzenia transakcji.
2. `__` Można zdefiniować kolumnę z wartościami domyślnymi, w której do generowania sekwencji używana jest pseudokolumna `.nextval`.
3. `__` Można zdefiniować kolumnę z wartościami domyślnymi, w której do generowania sekwencji używana jest pseudokolumna `.currval`.
4. `__` Pseudokolumna `.currval` nie jest już zależna od wcześniejszego wywołania pseudokolumny `.nextval` w sesji.

5. __Oracle Database 12c nie umożliwia zapobiegania bezpośredniemu ustawieniu wartości null w instrukcji INSERT, co oznacza, że można zastąpić wartość domyślną kolumny.
6. __Kolumny identyfikacyjne umożliwiają automatyczne generowanie wartości kolumny z kluczem sztucznym.
7. __W bazie Oracle Database 12c typy danych VARCHAR2, NVARCHAR2 i RAW zawsze mają 32 767 bajtów.
8. __Dzięki zmianom wprowadzonym w bazie Oracle Database 12c funkcja języka PL/SQL może zwrócić tablicę asocjacyjną tego języka bezpośrednio do instrukcji SQL-a.
9. __Oracle Database 12c obsługuje pobieranie n pierwszych wyników zapytania bez konieczności określania przedziału pobieranych danych.
10. __Można umieścić funkcję języka PL/SQL w klauzuli WITH zapytania i wywoływać ją w programach zewnętrznych.

Pytania wielokrotnego wyboru

11. Które z poniższych słów kluczowych można stosować przy definiowaniu widoku? Poprawnych może być kilka odpowiedzi.
 - A. Słowa kluczowe AUTHID DEFINER.
 - B. Słowa kluczowe BEQUEATH INVOKER.
 - C. Słowa kluczowe AUTHID CURRENT_USER.
 - D. Słowa kluczowe BEQUEATH DEFINER.
 - E. Wszystkie z powyższych.
12. Które z poniższych stwierdzeń dotyczących zapisywania w pamięci podręcznej wyników funkcji z uprawnieniami jednostki wywołującej są prawdziwe? Poprawnych może być kilka odpowiedzi.
 - A. Dla poszczególnych jednostek wywołujących istnieją różne zbiory wyników.
 - B. Dla każdej jednostki wywołującej istnieje ten sam zbiór wyników.
 - C. Funkcja z uprawnieniami jednostki wywołującej, której wyniki są zapisywane w pamięci podręcznej, musi być deterministyczna.
 - D. Funkcja z uprawnieniami jednostki wywołującej, której wyniki są zapisywane w pamięci podręcznej, może być niedeterministyczna.
 - E. Wszystkie z powyższych.
13. Które z poniższych stwierdzeń dotyczących przekształcania tekstu rozwijanych instrukcji SQL-a z typu LONG na typ CLOB są prawdziwe w kontekście pracy z widokami CDB_, DBA_, ALL_ i USER_VIEWS w bazie Oracle Database 12c? Poprawnych może być kilka odpowiedzi.
 - A. Możesz używać funkcji wbudowanej to_lob do przekształcania typu LONG na typ CLOB.
 - B. Możesz używać funkcji wbudowanej to_clob do przekształcania typu LONG na typ CLOB.
 - C. Możesz używać pakietu dbms_sql do przekształcania typu LONG na typ VARCHAR2.
 - D. Możesz używać funkcji wbudowanej length do określania długości danych typu LONG.
 - E. Możesz używać pakietu dbms_lob do tworzenia tymczasowych danych typu CLOB.
14. Które z poniższych stwierdzeń dotyczących typów danych języka PL/SQL dostępnych w zagnieżdżonych instrukcjach SQL-a są prawdziwe? Poprawnych może być kilka odpowiedzi.
 - A. Typ danych języka PL/SQL musi być zadeklarowany w pakiecie.
 - B. Instrukcja SQL-a musi być zagnieżdżona w bloku języka PL/SQL, w którym zdefiniowany jest określony typ.
 - C. Typ danych języka PL/SQL musi być zdefiniowany lokalnie.
 - D. Typ danych języka PL/SQL może być zwracany przez funkcję tego języka.
 - E. Wszystkie z powyższych.

15. Która z poniższych instrukcji umożliwia dostęp do sztucznego klucza głównego z kolumny identyfikacyjnej w celu użycia wartości tego klucza w dalszej instrukcji INSERT jako wartości klucza obcego?
- A. Instrukcja RETURN INTO.
 - B. Instrukcja RETURNING INTO.
 - C. Instrukcja .nextval.
 - D. Instrukcja .currval.
 - E. Żadne z powyższych.

Skorowidz

A

ACID, 523
administrowanie bazą danych, 553
agregacja selektywna, 736
agregowanie, 725
aktualizacje, 689
aktualizowanie

- elementów typu ADT, 696
- masowe, 817
- na podstawie zapytań skorelowanych, 699
- tabel zagnieżdżonych, 692

aliasy, 713
analiza zapytań, 826
anulowanie instrukcji, 539
APEX, Oracle Application Express, 42
API, Application Programming Interface, 58, 393
API Collection, 231
apostrof, 125, 538
architektura

- bazy danych Oracle, 508
- dynamicznego SQL-a, 468
- egzemplarza bazy danych, 510
- funkcji i procedur, 268
- MVCC, 523
- odbiornika Oracle, 512
- pakietu, 311
- przetwarzania instrukcji, 40
- wyzwalaczy, 428

arytmetyka modularna, 784
asocjacja, 124
asymetryczne złożone kolekcje, 223
atak, 569
atomowość, 673
atrybut %BULK_EXCEPTIONS, 202

atrybuty kursorów

- jawnych, 186
- masowych, 196
- niejawnych, 182

audytor EDP, 566

B

baza danych, 36

- CDB, 517, 555
- Oracle, 507, 559
- PDB, 513, 533, 557
- VPD, 640

bezpieczeństwo, 558, 810
białe listy, 402
białe listy jednostek, 57
biblioteka

- OCI, 511
- OCI8, 153

biblioteki Javy, 295, 651
biznesowy przypadek użycia, 642, 646
bloki

- anonimowe, 72, 204, 544
- zagnieżdżone, 76
- kompilacji warunkowej, 169
- nazwane, 545
- lokalne, 78
- składowane, 80

blokowanie, 528
błąd, 243, 482, 623
ORA-01436, 721
ORA-06502, 106
ORA-22337, 667
ORA-25015, 695
PLS-00382, 861
błędy

- czasu kompilacji, 245, 252
- czasu wykonania, 247, 252
- w sekcji deklaracji, 251

brakujący indeks, 215
bufor, 134

C

CBO, cost-based optimization, 581
certyfikat

- LDAP, 655
- SSL, 655

ciało

- obiektu, 394, 399
- pakietu, 115, 325, 330
- wyzwalacza, 462

CLI, 51
CTE, Common Table Expressions, 718
cudzysłów, 127
cykl życia

- operacji, 592
- transakcji, 525

czarna skrzynka, 268, 315
czas do złamania hasła, 571

D

dane lokalne, 273
daty, 83, 140, 733
DCL, Data Control Language, 40
DDL, Data Definition Language, 40
debugowanie, 809
definicja pakietu DBMS_SQL, 492
definiowanie kolumn

- jawne, 46
- niejawne, 47

tabel, 46, 657
tablic asocjacyjnych, 226
defragmentacja indeksów, 665

deklarowanie
 klas pochodnych, 414
 kolekcji obiektów, 419
 typów obiektowych, 396
 wyjątków, 253
 wyzwalacza, 428
 zmiennej, 137

diagram klasy, 395

DML, Data Manipulation Language, 40, 182

docelowe kolekcje rekordów, 199
 równoległe, 199

dodawanie
 kolumn, 659
 kolumn i ograniczeń, 660
 kolumn VT, 46
 kont użytkowników, 553

domyślne wartości kolumn, 47

dopasowywanie wartości, 700

dostęp do
 tabeli, 582
 zasięgu SQL-a, 853

dostępność pseudorekordów, 453

dostrajanie SQL-a, 580

drzewo, 721
 dziedziczenia dla złączeń, 737

duże obiekty, 147, 341, 680, 698

dynamiczne instrukcje, 487
 instrukcje DDL, 469, 480
 instrukcje DML, 470, 481
 kursory jawne, 190
 kursory niejawne, 185
 pętle proste, 174
 wyjątki, 108
 wyjątki użytkownika, 256

dynamiczny SQL, 467

dyrektywa
 \$\$PLSQL_LINE, 169
 \$\$PLSQL_OWNER, 169
 \$\$PLSQL_TYPE, 169
 \$\$PLSQL_UNIT, 169
 SERIALLY_REUSABLE, 319

dziedziczenie, 412

E

EDI, Electronic Data Interchange, 704

EDP, Electronic Data Processing, 566

edycja instrukcji, 538

egzemplarz
 bazy danych, 510, 512
 klasy, 394

elementy prototypu, 317, 325

etykiety, 126

ewolucja typu, 418

F

fabryka, 420

fabrykacja tabel, 759

FILO, First-In, Last-Out, 258

format
 ANSI, 714
 IEEE 754, 145

formatowanie danych, 541

funkcja 110, 267–309, 645
 ADD_MONTHS, 734, 765
 ASCII, 751
 ASCIISTR, 752
 BFILENAME, 789
 CARDINALITY, 773
 CAST, 140, 689, 734, 758
 CEIL, 783
 character_set, 794
 CHR, 752
 COALESCE, 791
 COLLECT, 773
 dla kolumn, 775
 dla tabel, 773
 COMPARE, 383
 CONCAT, 753
 CONVERT, 760
 COUNT, 726
 CREATETEMPORARY, 387
 CURRENT_DATE, 734, 765
 CURRENT_TIMESTAMP, 766
 DBFS_LINK_GENERATE_PATHTH, 389
 DBMS_DDL.WRAP, 850
 dbms_lob.fileexists, 365
 DBTIMEZONE, 766
 DECODE, 729, 791
 DUMP, 792
 EMPTY_BLOB, 792
 EMPTY_CLOB, 347, 795
 EXECUTE, 499
 EXECUTE_AND_FETCH, 499
 EXTRACT, 141, 734, 766
 FETCH_ROWS, 499
 FILEEXISTS, 386
 FILEISOPEN, 386
 FLOOR, 783

FROM_TZ, 767

get_canonical_bfilename, 372

GET_DBFS_LINK, 389

get_directory_path, 370

get_salary, 824

GET_STORAGE_LIMIT, 384

GETCHUNKSIZE, 383

GETCONTENTTYPE, 389

GETLENGTH, 384

GETOPTIONS, 384

GREATEST, 734, 795

INITCAP, 753

INSTR, 384, 753

ISOPEN, 377, 499

ISSECUREFILE, 380

ISTEMPORARY, 387

LAST_DAY, 734, 767

LAST_ERROR_POSITION, 499

LAST_ROW_COUNT, 500

LAST_ROW_ID, 500

LAST_SQL_FUNCTION_CODE, 500

LEAST, 734, 797

LENGTH, 147, 754

LOCALTIMESTAMP, 767

LOWER, 754

LPAD, 755

LTRIM, 755

MAP, 407

MOD, 783

MONTHS_BETWEEN, 734, 768

NANVL, 798

NEW_TIME, 768

NEXT_DAY, 734

NULLIF, 798

NVL, 97, 799

OPEN_CURSOR, 500

ORA_DATABASE_NAME, 435

ORA_DES_ENCRYPTED_PASWORD, 435

ORA_DICT_OBJ_NAME, 435

ORA_DICT_OBJ_NAME_LIST, 435

ORA_DICT_OBJ_OWNER, 436

ORA_DICT_OBJ_OWNER_LIST, 436

ORA_DICT_OBJ_TYPE, 436

ORA GRANTEE, 436

ORA_INSTANCE_NUM, 437

ORA_IS ALTER_COLUMN, 437

ORA_IS_CREATING_NESTED_TABLE, 438

ORA_IS_DROP_COLUMN,
438
ORA_IS_SERVERERROR, 439
ORA_LOGIN_USER, 439
ORA_PARTITION_POS, 439
ORA_PRIVILEGE_LIST, 439
ORA_REVOKEE, 440
ORA_SERVER_ERROR, 440
ORA_SERVER_ERROR_DEPTH,
440
ORA_SERVER_ERROR_MSG,
440
ORA_SERVER_ERROR_NUM_
PARAMS, 441
ORA_SERVER_ERROR_
PARAM, 441
ORA_SQL_TXT, 441
ORA_SYSEVENT, 441
ORA_WITH_GRANT_
OPTION, 441
ORDER, 408, 411
order_comp, 410
POWER, 785
POWERMULTISET, 776
POWERMULTISET_BY_
CARDINALITY, 777
REGEXP_COUNT, 839, 840, 845
REGEXP_INSTR, 842
REGEXP_LIKE, 834, 843, 845
REGEXP_REPLACE, 844
REGEXP_SUBSTR, 846
REMAINDER, 785
REPLACE, 756
REVERSE, 756
ROUND, 141, 734, 786
ROUND(data), 769
ROUND(liczba), 769
RPAD, 756
RTRIM, 757
salutation, 114
SET, 777
SPACE_ERROR_INFO, 441
SQLCODE, 106, 248, 787
SQLERRM, 106, 248, 787
SUBSTR, 385, 717
SYS_CONTEXT, 799
SYSDATE, 734, 769
SYSTIMESTAMP, 769
TABLE, 219, 802
TO_CHAR, 714, 735, 760
TO_CHAR(data), 770
TO_CLOB, 762
TO_CURSOR_NUMBER, 501
TO_DATE, 140, 735, 762

TO_DSINTERVAL, 771
TO_LOB, 763
TO_NCHAR, 764
TO_NCLOB, 764
TO_NUMBER, 764
TO_REFCURSOR, 501
TO_TIMESTAMP, 771
TO_TIMESTAMP_TZ, 771
TO_YMINTERVAL, 772
TREAT, 417, 669, 804
TRUNC, 141, 772
TZ_OFFSET, 772
UPPER, 757
USERENV, 805
VSIZE, 806
WRAP, 851
funkcje
funkcje-atrybuty zdarzeń, 433
deterministyczne, 294
do konwersji dat i czasu, 765
do konwersji typów danych, 758
do zarządzania błędami, 786
do zarządzania kolekcjami, 773
jako wyrażenia, 164
liczbowe, 783
niedeterministyczne, 296
pakietu DBMS_SQL, 493
pakietu utl_call_stack, 259
pesymistyczne, 272
PL/SQL, 57
potokowe, 285
rekurencyjne, 299
różne, 789
składowane, 320
SQL-a, 45
wbudowane SQL-a, 751
znakowe, 751
zwracające tabele obiektowe,
287, 642

G

generowanie haseł, 572, 573
getter, 403
godziny, 140
gwiazdka, 69, 725

H

hasło, 532, 570–574, 600, 655
hierarchiczny program profilujący,
857–867

I

IDE, integrated development
environment, 33
identyfikator SID, 513
identyfikatory, 126, 127
w cudzysłowach, 128
wbudowane, 128
IDL, Interface Definition
Language, 619
IEEE 754, 145
ILM, Information Lifecycle
Management, 46
iloczyn kartezjański, 740
implementacja ciała, 398
implementowanie
klas pochodnych, 415
kolekcji obiektów, 420
indeks_kursora, 184
indeksy, 637, 664
klastrowe, 638
niewidoczne, 665
oparte na B-drzewach, 638
oparte na funkcjach, 639
oparte na mapach bitowych, 639
rosnące i malejące, 638
specyficzne dla domeny
aplikacji, 639
widoczne, 665
z odwróconymi kluczami, 638
informacje o kluczach
zewnątrznych, 567
inicjowanie obiektów, 344
inlining, 308
instrukcja
ALTER, 654, 662
ALTER TABLE, 660
ALTER USER, 655
APPLY, 55
BULK COLLECT INTO, 196
CASE, 98, 166, 625
CASE z wyszukiwaniem, 167
COMMENT, 672
COMMIT, 117, 710
CONTINUE, 103, 179
CREATE, 598
CTAS, 815, 818
DECODE, 729
DELETE, 201, 675, 699
DESCRIBE, 612, 693
DROP, 670, 671
elsif, 97
EXIT, 102, 171

instrukcja
 EXPLAIN PLAN, 581
 FORALL, 109, 200
 GOTO, 103
 GRANT, 479
 if, 97, 163
 if-then-else, 163
 if-then-elsif-then-else, 165
 INSERT, 200, 346, 673
 INSERT ALL, 686, 687
 LEFT OUTER JOIN, 47
 LIMIT, 198
 MERGE, 703, 708
 OUTER APPLY, 55
 RENAME, 669
 ROLLBACK, 117, 710
 SAVEPOINT, 117, 710
 SELECT, 622, 711
 SELECT INTO, 220
 shutdown immediate, 516
 startup, 517
 TRUNCATE, 672
 UPDATE, 201, 346, 674, 688

instrukcje
 DCL, 40, 710
 DDL, 40
 DML, 40, 182, 297, 672
 dynamiczne, 469, 477, 484
 z danymi wejściowymi, 471, 482
 z danymi wyjściowymi, 473
 iteracyjne, 171, 181
 kompilacji warunkowej, 168
 masowe, 196
 SQL-a w SQL*Plus, 537
 TCL, 40
 warunkowe, 157

interfejs
 API, 58
 API Collection, 231
 OCI, 197
 SQL*Plus, 530

interfejsy programowe, 619
 interwałowe typy pochodne, 141
 interwały, 140
 iterowanie, 172
 izolacja, 673
 izolacja READ UNCOMMITTED, 527

J

jawne identyfikowanie kolumn, 46
 JDBC, Java Database
 Connectivity, 526

JDK, Java Software Development
 Kit, 548
 jednostka
 definiująca, 332, 528
 wywołująca, 332, 529
 jednostki leksykalne, 123
 jednowymiarowe tablice znaków,
 843
 język
 DCL, 40
 DDL, 40, 597
 DML, 40
 PL/SQL, 33, 37
 SQL, 591
 TCL, 40, 709
 języki imperatywne, 591
 JVM, Java Virtual Machine, 320

K

kardynalność, 562
 katalog wirtualny, 46, 363, 648, 705
 kategorie napastników, 569
 klasa, 394
 pochodne, 414
 porządkowania, 836
 znaków, 833
 klauzula
 ACCESSIBLE BY, 114, 318
 BULK COLLECT, 108
 CASCADE CONSTRAINTS, 670
 DEFAULT ON NULL, 612
 DETERMINISTIC, 280
 FOR UPDATE, 527, 674
 FROM, 712
 GROUP BY, 712, 725
 HAVING, 712
 LATERAL, 55
 NOT FINAL, 666
 NOT NULL, 140
 OR REPLACE, 640
 ORDER BY, 712
 PARALLEL_ENABLE, 281
 PIPELINED, 282
 RESULT_CACHE, 288
 RETURNING INTO, 691
 RETURNING INTO, 345
 VALUES, 673, 677
 WHEN, 98, 685
 WHERE, 712
 WHERE CURRENT OF, 101
 WITH, 60
 WITH CHECK OPTION, 645

klauzule instrukcji SELECT, 712
 klucz
 główny, 565
 naturalny, 609
 sztuczny, 678
 zewewnętrzny, 566
 kod ograniczenia, 658
 kolejność operacji, 162
 kolekcje, 149, 151, 207–240
 ADT, 90, 94
 asymetrycznych typów
 złożonych, 223
 docelowe ograniczone, 198
 języka PL/SQL, 87, 94
 języka SQL, 87, 90
 obiektów, 419
 rekordów, 198, 199
 równoległe, 199
 typów złożonych, 218
 UDT, 92, 95
 zagnieżdżone, 635
 złożone asymetryczne, 223
 kolumny
 IDENTITY, 49
 identyfikacyjne, 629, 643
 modyfikowalne w widoku, 644
 niewidoczne, 626
 niezagregowane, 727
 typu CLOB, 693
 wirtualne, 624
 zagregowane, 727
 komentarze, 126, 130
 kompilacja warunkowa, 169
 konfigurowanie
 precyzyjnej kontroli, 825
 programu profilującego, 857
 sieci, 519
 SQL Developer, 548
 środowiska SQL*Plus, 533
 konto użytkownika, 512, 531, 555,
 599, 655
 kontrola blokowania, 527
 konwersja
 dat i czasu, 765
 jawna, 133
 niejawna, 133
 typów danych, 682, 758
 korelowanie, 688
 kotwiczenie atrybutów i tabel, 84
 kropka, 125
 kursory, 181
 jawne, 186
 jawne dynamiczne, 190
 jawne statyczne, 187

masowe, 196
 niejawne, 182
 dynamiczne, 185
 jednowierszowe, 182
 statyczne, 184
 wielowierszowe, 183
 referencyjne systemowe, 292
 zagnieżdżone, 191

L

leniwa kompilacja, 669
 liczba, 84, 144
 danych wejściowych, 477, 484,
 487
 danych wyjściowych, 484, 487
 wyników, 52
 licznik, 179
 Linux, 514
 lista, 633
 literały, 129, 838
 liczbowe, 129
 logiczne, 129
 w postaci łańcuchów znaków,
 129
 z datą i czasem, 130
 znakowe, 129
 LOB, Large Objects, 341
 logiczne typy danych, 135
 logika trójwartościowa, 97
 lokalne bloki nazwane, 78

Ł

łamanie hasła, 571
 łańcuchy znaków, 83, 136, 615
 łańcuchy znaków Unicode, 143
 łączenie
 kolekcji, 746
 kolumn z sekwencjami, 631

M

mechanizm
 CBO, 581
 zapisu danych, 510
 zapisu dziennika, 510
 metadane sesji, 456
 metasekwencje, 838
 metasekwencje standardu POSIX,
 838

metaznak, 836
 daszek, 834
 myślnik, 834
 znak dolara, 834
 metaznaki standardu POSIX, 836,
 837

metoda

COUNT, 231, 233
 DELETE, 231, 234
 EXISTS, 232, 235
 EXTEND, 232, 236
 FIRST, 232, 236
 LAST, 232, 237
 LIMIT, 232, 237
 NEXT, 232, 238
 PRIOR, 232, 238
 TRIM, 232, 239

metody

do introspekcji, 382
 do manipulowania dużymi
 obiektami, 377
 do obsługi obiektów typu
 BFILE, 385
 do otwierania, 376
 do zamykania, 376
 statyczne, 405
 migracja, 606, 659, 763
 model
 ACID, 523, 673, 675
 dwuwarstwowy, 41
 n-warstwowy, 41
 oparty na sygnaturach, 338
 oparty na znacznikach czasu, 338
 ORM, 635
 OSI, 511

modele przetwarzania, 41
 modularyzacja, 393
 modyfikowanie kolumn, 659
 i ograniczeń, 662

monitor

procesów, 509
 systemu, 509
 MVCC, Multiversion Concurrency
 Control, 523
 myślnik, 834

N

NaN, Not a Number, 798
 narzędzie, 811
 Audit Vault and Database
 Firewall, 570
 do analizy zapytań, 826

Net8, 521
 Oracle Enterprise Manager, 515
 Oracle SQL Developer, 547, 551
 plshprof, 865
 SQL*Loader, 648, 649
 SQL*Net, 575
 tkprof, 588
 wrap, 850
 nawiasy klamrowe, 67
 nazwa tabeli, 610, 657
 NDS, Native Dynamic SQL, 467, 484
 niejawne
 definiowanie kolumn, 47
 wiązanie parametrów, 63
 niewidoczne kolumny, 607
 notacja
 mieszana, 276
 oparta na nazwie, 276
 oparta na pozycji, 275
 w wywołaniach, 276
 z pominięciem, 276
 nowe funkcje, 45

O

obiekt, 150, 394
 obiektowe typy danych, 619
 obiekty
 typu BFILE, 391
 typu LOB, 624
 obliczanie interwału, 142
 obliczenia matematyczne, 127
 obsługa
 atrybutu
 %BULK_EXCEPTIONS, 202
 bezpiecznych odnośników, 388
 błędów, 243
 dat, 734
 dużych obiektów, 387, 680
 hurtowni danych, 809
 klienckich interfejsów API, 58
 obiektów typu BFILE, 385
 wyjątków, 248, 249
 wymiaru VT, 46
 obszar PGA, 213
 OCI, Oracle Call Interface, 197,
 343, 511
 OCI8, Oracle Call Interface 8, 153
 odbiornik, 519, 574
 odczytywanie danych
 wyjściowych, 861
 odpowiedzi do testów, 877–903

OEM, Oracle Enterprise Manager, 42
 ograniczanie głębokości przeszukiwania, 723
 ograniczenia, 560, 561
 bazy danych, 605
 funkcji DBMS_DDL.WRAP, 850
 klucza głównego, 565
 tabeli, 636
 w opakowywaniu kodu, 850
 ograniczenie
 CHECK, 568, 605, 636, 661
 FOREIGN KEY, 566, 605, 637, 662
 NOT NULL, 561, 605, 663
 PRIMARY KEY, 605, 637, 661
 UNIQUE, 563, 605, 636, 663
 ograniczniki, 123–127
 ograniczniki etykiet, 126
 okno
 Open, 552
 Select Database Connection, 549
 Usługi, 518
 Worksheet, 550
 opakowywanie kodu, 849–856
 opcja
 INVALIDATE, 669
 RNDS, 280
 RNPS, 280
 TRUST, 280
 WNDS, 280
 opcje programu plshprof, 865
 operacje
 CRUD, 592
 masowe, 108
 na danych, 733
 w systemie
 Linux, 514
 Microsoft Windows, 518
 Unix, 514
 operator
 AND, 158
 asocjacji, 125
 BETWEEN, 158
 CARDINALITY, 779
 CASE, 732
 dodawania, 127
 dzielenia, 127
 EMPTY, 779
 IN, 159
 INTERSECT, 748
 IS A SET, 160
 IS EMPTY, 159
 IS NULL, 159

 komentarza, 126
 LIKE, 160
 łączenia, 125
 MEMBER OF, 160
 MINUS, 749
 mnożenia, 127
 MULTISET, 779
 MULTISET EXCEPT, 780
 MULTISET INTERSECT, 780
 MULTISET UNION, 781
 NOT, 161, 689
 OR, 161, 689
 potęgowania, 127, 611
 przypisania, 70, 76, 124
 równości, 702
 SET, 781
 SUBMULTISET, 161
 SUBMULTISET OF, 782
 UNION, 747
 UNION ALL, 748
 operatory
 porównywania, 158–161
 zbiorów działające dla kolekcji, 777
 Oracle Streams, 810
 ORDBMS, 34, 633
 ORM, Object Relational Model, 635
 OSI, 511
 oś czasu, 35

P

pakiet
 DBMS_APPLICATION_INFO, 814
 DBMS_COMPARISON, 818
 dbms_crypto, 823
 DBMS_DDL, 851
 dbms_fga, 825
 DBMS_LOB, 352, 375, 376
 dbms_random, 572
 DBMS_SQL, 51, 469, 478
 DBMS_UTILITY, 59, 108
 DBMS_XPLAN, 582
 JDK, 548
 sql_stats, 829
 utl_call_stack, 59, 259
 pakiety, 113, 311, 645, 807
 architektura, 311
 ciało, 325
 elementy prototypu, 317, 325
 do obsługi hurtowni danych, 809
 do zarządzania zadaniami, 809

funkcje, 324, 330
 opisywanie, 336
 narzędziowe, 811–813
 procedury, 324, 330
 specyfikacja, 316
 stałe, 375
 typy danych, 322
 walidacja, 336
 wyjątki, 376
 zmienne, 320, 327
 związane z
 bezpieczeństwem, 810
 debugowaniem, 809
 Oracle Streams, 810
 wydajnością, 810
 XML-em, 813
 parametry
 REF CURSOR, 63
 SYS_CONTEXT, 800, 802
 USERENV, 806
 partycje złożone, 653
 partycjonowanie oparte na listach, 651
 przedziałach, 652
 skrótach, 652
 pętla
 FOR, 99
 oparta na kursorze, 184
 oparta na przedziale, 177, 181
 FORALL, 200
 WHILE, 102, 178
 pętle
 proste, 103, 171
 dynamiczne, 174
 statyczne, 173
 z warunkiem na wejściu, 178
 z warunkiem na wyjściu, 172
 PGA, Process Global Area, 184
 PGA, Program Global Area, 213
 PL/SQL, 33
 plik
 associative_array.sql, 240
 asymmetrical_composites.sql, 240
 basic_objects.sql, 422
 bulk_processing_logic.sql, 204
 collection_api.sql, 240
 compound_triggers.sql, 465
 conditional_logic.sql, 204
 ConvertBlobToImage.php, 391
 ConvertFileToImage.php, 366, 391

- create_web_blob_loading.sql, 391
- create_web_clob_loading.sql, 391
- deterministic.sql, 309
- ddl_triggers.sql, 465
- dml_triggers.sql, 465
- dynamic_topnquery.php, 64
- exception_handling.sql, 263
- expanding_view.sql, 64, 390
- get_bfilename.sql, 391
- get_canonical_bfilename.sql, 391
- get_directory_path.sql, 391
- glogin.sql, 534
- iterative_logic.sql, 204
- listener.ora, 520
- java_library.sql, 309
- load_blob_from_file.sql, 391
- load_clob_from_file.sql, 391
- magic.html, 866
- map_compare.sql, 423
- merging.sql, 309
- order_compare.sql, 423
- pass_by_reference.sql, 309
- pipelined.sql, 309
- QueryItemBFILE.php, 372
- QueryRelativeBFILE.php, 367, 391
- ReadCanonicalFileToImage.php, 374, 391
- recursive.sql, 309
- reserved_key_word.sql, 874
- result_cache.sql, 309
- sfile.ora, 338
- sql_collection.sql, 240
- stack_trace_management.sql, 263
- static_methods.sql, 423
- symmetrical_composites.sql, 240
- system_triggers.sql, 465
- UploadItemBlob.php, 391
- UploadItemBlobForm.htm, 359, 391
- UploadItemDescription.php, 391
- UploadItemDescription
 - ↳ Form.htm, 359, 391
- white_list.sql, 64
- pliki
 - binarne, 363
 - CSV, 648, 705
 - lokalne, 355
 - narzędzia SQL*Loader, 649
 - Oracle Data Pump, 650
 - śladu, 588
 - TSV, 648
 - wsadowe, 539
- pobieranie ostatniej wartości sekwencji, 681
- podkursory, 191
- podprogramy, 128, 269
- podstawowa struktura bloku, 68
- podzapytania
 - jednowierszowe, 715, 716
 - skalarne, 715
 - skorelowane, 715, 717
 - wielowierszowe, 715, 716
- podział na partycje, 652
- pojemność typów łańcuchowych, 50
- poła struktury, 376
- polecenie
 - @, 540
 - COMMIT, 643
 - DESCRIBE, 620, 668, 820
 - HOST, 537
 - netmgr, 575
 - QUIT, 537
 - su, 514
- polimorfizm, 412
- połączenie
 - z bazami danych, 819
 - z CDB, 517
 - z SQL*Plus, 530
- pomijanie iteracji, 176
- porównywanie, 125, 717
 - obiektów, 406–408
 - pakietów, 808
- poziomy
 - dostępu, 821
 - izolacji, 527
- procedura, 112, 303, 645
 - APPEND, 378
 - BIND_ARRAY, 493
 - BIND_VARIABLE, 494
 - BIND_VARIABLE_CHAR, 494
 - BIND_VARIABLE_RAW, 494
 - BIND_VARIABLE_ROWID, 495
 - change_unprotected, 321
 - CLOSE, 377
 - CLOSE_CURSOR, 495
 - COLUMN_VALUE, 495
 - COLUMN_VALUE_CHAR, 496
 - COLUMN_VALUE_LONG, 496
 - COLUMN_VALUE_RAW, 496
- COLUMN_VALUE_ROWID, 496
- CONVERTTOBLOB, 378
- COPY, 378
- COPY_DBFS_LINK, 388
- COPY_FROM_DBFS_LINK, 388
- CREATE_WRAPPED, 855
- DEFINE_ARRAY, 497
- DEFINE_COLUMN, 497
- DEFINE_COLUMN_CHAR, 497
- DEFINE_COLUMN_LONG, 497
- DEFINE_COLUMN_RAW, 497
- DEFINE_COLUMN_ROWID, 498
- DESCRIBE_COLUMNS, 498
- ERASE, 379
- expand_sql_text, 59
- FILECLOSE, 385
- FILECLOSEALL, 386
- FILEGETNAME, 386
- FILEOPEN, 386
- FRAGMENT_DELETE, 379
- FRAGMENT_INSERT, 379
- FRAGMENT_MOVE, 379
- FRAGMENT_REPLACE, 380
- FREETEMPORARY, 387
- GET_DBFS_LINK_STATE, 390
- GET_DEDUPLICATED_REGIONS, 383
- LOADBLOBFROMFILE, 381
- LOADCLOBFROMFILE, 354, 381
- LOADFROMFILE, 381
- OPEN, 377
- PARSE, 60, 500
- printt, 829
- quantity_onhand, 861
- READ, 385
- SET_DBFS_LINK, 390
- set_session_longops, 816
- SETCONTENTTYPE, 390
- SETOPTIONS, 381
- tandem, 710
- TRIM, 382
- VARIABLE_VALUE, 501
- VARIABLE_VALUE_CHAR, 502
- VARIABLE_VALUE_RAW, 502
- VARIABLE_VALUE_ROWID, 502
- WRITEAPPEND, 382

procedury
 pakietu DBMS_SQL, 51, 493
 wczytujące dane, 357

proces
 dwuetapowego zatwierdzenia, 425
 ILM, 46
 zarządzania danymi, 579

profile użytkowników, 571

program, 509
 plshprof, 866
 ReadCanonicalFileToImage.php, 373
 SQL*Plus, 33
 System-Gnome-Monitor, 815
 tnsping, 523
 Wireshark, 575

programowanie, 265

programowanie baz Oracle, 35

protokół
 2PC, 119, 673
 LDAP, 656

prototyp
 ciała pakietu, 115
 instrukcji
 ALTER, 664
 ALTER TABLE, 660
 ALTER USER, 655
 INSERT, 677
 SELECT, 711

przebieg transakcji, 274

przechodzenie
 w dół drzewa, 721
 w górę drzewa, 723

przechwytywanie danych, 575

przeciążanie, 314, 676

przekazywanie parametrów
 do plików, 540
 przez referencję, 269, 301, 307
 przez wartość, 269, 290, 294–297, 304
 w trybie interaktywnym, 540
 w trybie wsadowym, 542

przekazywanie wartości
 z instrukcji, 50

przekształcanie
 dat, 761
 liczb, 761
 łańcuchów, 760
 wartości, 348

przenoszenie
 danych obiektowych, 606
 zbioru danych, 606

przeñośne klasy znaków, 834

przesłanie, 676

przestrzeń nazw, 512, 657

przetwarzanie instrukcji, 486, 712
 masowych, 487
 interaktywne, 530
 wsadowe, 530

przypisywanie, 124
 wartości, 343, 344, 351
 wyniku funkcji, 270

przywracanie danych, 810

przyznawanie uprawnień, 334, 558–601

pseudokolumna ROWNUM, 715

pseudotabela, 545

punkt kontrolny, 510

R

RDBMS, 33

referencja, 620

referencje uprzedzające, 313

rejestrwanie operacji, 862

rekordy, 149, 150

rekurencja, 299

repozytoria danych, 508

RMI, Remote Method Invocation, 619

rodzaje
 ograniczeń, 658
 podprogramów, 269
 zapytań, 711

role użytkowników, 571

rozbudowa typów, 666

rozmiar wyzwalaczy, 462

rozwijanie podprogramów w miejscu wywołania, 308

RPC, Remote Procedure Calls, 619

rzutowanie, 650

S

scalanie
 danych, 704
 źródła importu, 707

sekcja
 deklaracji, 71
 obsługi wyjątków, 71
 wykonawcza, 68

sekwencje, 47, 628

selektor komponentów, 125, 314

separator elementów, 126

serwer bazy Oracle Database 12c, 513

settery, 403

singleton, 329

skalarne typy danych, 82, 134

skanowanie
 indeksów, 583
 map bitowych, 583

składowane bloki nazwane, 80

skorelowane podzapytanie, 703

skrypty, 503, 539

słowa
 kluczowe, 128, 869–874
 zarezerwowane, 128, 182, 869–874

słowo kluczowe
 ALL, 685
 AS, 713
 BEGIN, 153
 DEFAULT, 114, 615
 DEFINE, 534
 EXCEPTION, 154
 INVISIBLE, 608
 OTHERS, 106
 PRIOR, 723

sortowanie, 588
 bąbelkowe, 797
 drzewa, 722

specyfikacja
 pakietu, 316
 typu obiektowego, 394, 400

spójność, 673

sprawdzanie
 haseł, 574
 poprawności, 338
 zależności, 337

SQL, Structured Query Language, 591

stałe
 pakietu DBMS_LOB, 375
 pakietu DBMS_SQL, 492
 systemu DBFS, 388

standard
 ANSI, 527, 593
 IEEE 754, 145
 ISO, 527

statyczne
 kursory jawne, 187
 kursory niejawne, 184
 metody składowe, 405
 pętle proste, 173

statystyki, 831

sterownik bazy Oracle, 55

stos wyjątków, 258

stosowanie
 dużych obiektów, 357
 katalogów wirtualnych, 363
 kolekcji, 209
 narzędzia plshprof, 865
 narzędzia wrap, 850
 obiektów trwałych, 409
 ograniczeń, 560
 pakietów, 814
 tablic asocjacyjnych, 226
 wyrażeń regularnych, 839
 strony WWW, 357
 struktura bloków, 67
 sekcja deklaracji, 71
 sekcja obsługi wyjątków, 71
 sekcja wykonawcza, 68
 struktura podstawowa, 68
 struktura SEQUENCE, 629
 struktury
 iteracyjne, 99
 rekordowe
 BLOB_DEDUPLICATE_
 REGION, 375
 CLOB_DEDUPLICATE_
 REGION, 375
 sterujące, 97, 157
 warunkowe, 97
 symulowanie typu logicznego, 618
 synonimy, 334, 653
 system
 OLTP, 581
 pomocy w SQL*Plus, 536
 systemowe kursory referencyjne,
 149, 152, 292
 systemy
 pokazowe, 431
 produkcyjne, 432
 testowe, 431
 używane do programowania, 431
 szyfrowanie danych, 575

Ś

ścieżka do pakietu JDK, 548
 ścieżki kanoniczne, 368
 śledzenie
 danych w sesji, 586
 instrukcji SQL-a, 585
 operacji w sieci, 556
 środowisko
 IDE, 33
 SQL*Plus, 530, 532

T

tabela
 logging wyzwalaczy, 434
 uprawnień, 561
 tabele, 46, 604, 656
 języka PL/SQL, 208
 mutujące, 463
 obiektowe, 607
 oparte na indeksach, 638
 partycjonowane, 651
 programu profilującego, 858, 863
 tymczasowe, 608
 zagnieżdżone, 213, 635, 683
 zagnieżdżone elementów
 skalarnych, 213
 zewnętrzne, 648
 tablica VARRAY, 210
 tablice, 633
 asocjacyjne, 94, 96, 208, 225
 elementów skalarnych, 226
 elementów złożonych, 229
 prawdziwości, 162
 tęczowe, 570
 TCL, Transaction Control
 Language, 40, 709
 technologia
 IDL, 619
 Oracle Data Pump, 650
 Shared Source CLI, 51
 testowanie
 funkcji DECODE, 730
 konfiguracji, 706
 transakcje, 117, 273
 na danych, 524
 zgodne z modelem ACID, 673
 tryb
 IN, 271
 IN OUT, 271
 OUT, 271
 SERIALIZABLE, 527
 tryby
 blokowania, 528
 dostępu do tabel, 582
 parametrów podprogramów, 271
 skanowania indeksów, 583
 skanowania map bitowych, 583
 TT, transaction time, 46
 tworzenie
 białych list, 402
 białych list jednostek, 57
 funkcji, 279
 haseł, 600

indeksu, 637
 katalogów wirtualnych, 363, 705
 kont użytkowników, 555, 557, 599
 obiektów LIBRARY, 46
 ogólnego użytkownika, 533
 synonimów, 654
 tabel, 706
 wyzwalaczy DDL, 442
 typ
 ADT, 646, 665
 base_t, 620, 668
 BFILE, 148, 363, 385, 391, 596
 BINARY_DOUBLE, 595
 BINARY_FLOAT, 145, 595
 BINARY_INTEGER, 144
 BLOB, 148, 342, 347, 358, 361,
 596, 698
 BOOLEAN, 135
 CHAR, 136, 594
 CHARACTER, 136
 CLOB, 148, 342, 348, 351, 358,
 361, 595, 698
 DATE, 140, 595, 614
 DEC, 146
 DECIMAL, 146
 DOUBLE PRECISION, 147
 FLOAT, 147, 595
 INTERVAL DAY, 596
 INTERVAL YEAR, 596
 LONG, 137, 348, 595
 LONG RAW, 137, 463, 596
 NCHAR, 594
 NCLOB, 149, 342, 351, 595
 NUMBER, 146, 595
 NUMERIC, 146
 NVARCHAR2, 144, 595
 PLS_INTEGER, 145, 147
 RAW, 50, 596
 ROWID, 138, 597
 SIMPLE_INTEGER, 144
 STRING, 138, 594
 suit_object, 227
 TIMESTAMP, 142, 596
 TIMESTAMP WITH TIME
 ZONE, 596
 UDT, 646, 665
 UROWID, 138, 597
 VARCHAR, 138, 594
 VARCHAR2, 138, 594
 XMLTYPE, 597, 624
 typy
 danych, 61, 131, 322
 DBMS_SQL, 492
 logiczne, 135

- przestrzenne, 597
- skalarne, 82, 134
- SQL*Plus, 594
- zdefiniowane przez użyt-
kownika, 86, 87, 128, 597
- złożone, 86, 149
- kolekcji zagnieżdżonych, 633
- liczbowe, 611
- LOB, 147
- łańcuchowe, 50
- obiektowe, 149, 209, 393, 633,
646, 665
- tabele zagnieżdżone, 213
- tablice VARRAY, 210
- pochodne, 417
- rekordowe, 86, 89
- wyjątków, 243
- złożone spoza języka, 208
- zmiennoprzecinkowe, 613

U

- unikatowe indeksy, 564
- Unix, 514
- uprawnienia, 558–561
 - jednostki definiującej, 272, 332,
528
 - jednostki wywołującej, 273, 332,
528
- obiektowe, 558, 601
- przyznawane lokalnie, 602
- systemowe, 601
- uprawnienie
 - ALTER ANY TABLE, 427
 - CREATE ANY TRIGGER, 427
- uruchamianie
 - bazy danych, 516
 - instrukcji z bufora, 538
 - odbiornika Oracle, 519
 - serwera, 513
- ustawianie zmiennych sesji, 545
- ustawienie BEQUEATH
- CURRENT_USER, 56
- usuwanie
 - dużych obiektów, 616
 - elementów z tabel
zagnieżdżonych, 702
 - kolumn, 659
 - kolumn i ograniczeń, 663
 - konstruktora, 668
 - ograniczeń, 663
 - za pomocą zapytań
skorelowanych, 703

- uwierzytelnianie LDAP, 656
- użytkownicy, 598

V

- VPD, virtual private databases, 640
- VT, valid time, 46

W

- wartości domyślne, 49
- wartość
 - NaN, 798
 - null, 49
- wartownik, 171
- warunkowe zwracanie wartości, 189
- wbudowane
 - pakiety i typy PL/SQL, 807
 - typy danych, 611
- wczytywanie
 - danych, 358
 - plików, 352, 355
 - ścieżek kanonicznych, 368
- wdrażanie zarządzania danymi,
579
- węzeł główny, 720
- węzły podrzędne, 720
- widok
 - ALL_DEPENDENCIES, 337
 - ALL_OBJECTS, 336
 - CDB_DEPENDENCIES, 337
 - CDB_OBJECTS, 336
 - DBA_DEPENDENCIES, 337
 - DBA_OBJECTS, 336
 - USER_CONS_COLUMNS, 658
 - USER_CONSTRAINTS,
658USER_DEPENDENCIES,
337
 - USER_OBJECTS, 336
- widoki, 56, 639
 - administracyjne, 567
 - do odczytu i zapisu, 640
 - niemodyfikowalne, 460
 - oparte na funkcjach, 642
 - podzielone na segmenty, 598
 - pojęciowe, 598
 - tylko do odczytu, 641
 - utrwalone, 281
 - wewnętrzne, 718
 - wewnętrzne, 598
 - zewnętrzne, 599
- wiersz poleceń, 515, 530
- Windows, 518

- wirtualne aliasy, 365
- włączanie
 - bufora, 544
 - indeksów, 664
 - śledzenia, 586, 587
- wskaźnik
 - atributu, 124
 - dostępu zdalnego, 125
 - podstawiania, 124
 - zmiennej, 124
- wstawianie
 - danych, 677
 - danych do wielu tabel, 685
 - tablic, 683
 - wartości skalarnych, 678
- wstrzyknięcie kodu SQL, 472
- wybór rodzaju funkcji, 278
- wycofywanie uprawnień, 560
- wydajność, 810
- wydajność hierarchii wywołań, 867
- wyjątki, 106, 243
 - dynamiczne, 108, 256
 - pakietu, 376
- wbudowane, 254
- zdefiniowane przez
 - użytkownika, 107, 253
 - związane z kolekcjami, 233
- wykonywanie
 - bloków nazwanych, 545
 - programów, 544
- wykrzyknik, 537
- wyłączanie
 - indeksów, 664
 - ograniczenia klucza, 567
 - śledzenia, 587
- wymiar VT, 46
- wyniki złączeń, 736
- wyrażenia regularne, 714, 833–847
- wyrażenie
 - CTE, 718
 - DEFAULT, 614
- wyszukiwanie pakietów, 336
- wyświetlanie danych, 361
- wywołania zdalne, 335
- wywoływanie
 - podprogramów, 275
 - programów, 543
- wyzwalacze, 425
 - bazodanowe, 119
 - DDL, 119, 426, 431, 442
 - DML, 426, 429, 444
 - ograniczenia, 462
 - systemowe, 119, 427, 461
 - z poziomu instrukcji, 445

z poziomu wierszy, 447
 zastępujące, 119, 426, 457
 złożone, 426, 453
 wzmacnianie zabezpieczeń, 569
 wzorzec
 fabryki, 420
 projektowy singleton, 329

X

XML, eXtensible Markup
 Language, 704, 813

Z

zabezpieczanie
 bazy danych, 569
 haseł, 570
 odbiornika, 574
 zamykanie bazy danych, 515
 zapisywanie
 danych, 693
 instrukcji, 537
 plików dziennika, 546
 zapytania, 711
 agregujące, 725
 hierarchiczne, 720
 o przetworzone dane, 863
 rozwijające, 740
 skorelowane, 699
 zwracające kolumny, 712, 729

zarządzanie
 błędami, 786
 danymi, 577, 580
 dużymi obiektami, 391
 kolekcjami, 773
 kolumnami, 660
 licznikiem, 179
 obiektami, 391
 pakietami, 335
 stosem błędów, 258
 wyjątkami, 106, 248, 252
 zadaniami, 809
 zasięg
 referencji, 622
 SQL-a, 853
 transakcji, 117, 273
 odrębny, 118
 pojedynczy, 117
 wyjątków, 243
 zatrzaski, 829
 zatrzymywanie
 odbiornika Oracle, 519
 serwera, 513
 usługi, 522
 zbieranie
 danych, 859
 statystyk, 831
 zdarzenia, 592
 DDL, 432
 DML, 458

zdarzenie
 AFTER EACH ROW, 430
 AFTER STATEMENT, 430
 BEFORE EACH ROW, 430
 BEFORE STATEMENT, 430
 zdenormalizowane zbiory danych,
 704
 zewnętrzne zbiory danych, 705
 zliczanie słów, 840, 841
 złączenia, 737
 dotyczące wierszy, 738
 łańcuchów znaków, 714
 łączące kolekcje, 746
 naturalne, 737, 743
 skorelowane, 703
 złączenie
 krzyżowe, 737–740
 lewostronne, 738, 744
 pełne, 738, 746
 prawostronne, 738, 745
 wewnętrzne, 737, 742
 zewnętrzne, 738, 743
 złożone typy danych, 86, 149
 zmienianie
 hasła, 655, 656
 nazw indeksów, 665
 zmienna typu EXCEPTION, 107
 zmienne, 131, 320, 327
 podstawiane, 542
 sesji, 545
 skalarne, 94
 w blokach, 72
 złożone, 96
 znaczniki czasu i sygnatury, 338
 znaki, 136, 143
 znaki ucieczki, 538

Ź

źródła ataków, 569
 źródłowy_lokalizator, 345

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA WYDAWNICZA

 **Helion SA**

Obowiązkowa wiedza każdego bazodanowca!

Oracle to jedna z najlepszych baz danych. Najczęściej korzystają z niej firmy i instytucje. W jej tabelach przechowywane są gigantyczne ilości danych, przetwarzane każdego dnia w celu wyłuskania kluczowych informacji. Wykonywanie operacji na danych bezpośrednio w bazie jest możliwe dzięki rozszerzeniu języka SQL, które pozwala na tworzenie konstrukcji znanych z innych języków programowania. To właśnie PL/SQL!

Jeżeli sięgniesz po tę książkę, będziesz mieć niepowtarzalną okazję błyskawicznego poznania jego potencjału. Na kolejnych stronach znajdziesz informacje na temat podstaw pracy z PL/SQL, a następnie przejdziesz do bardziej zaawansowanych zagadnień. Zdobędziesz wiedzę na temat struktur sterujących, kolekcji oraz pracy z dużymi obiektami. Ponadto nauczysz się obsługiwać błędy oraz budować wyzwalacze. Książka ta jest doskonałą lekturą dla osób chcących poznać możliwości języka PL/SQL!

Dzięki tej książce:

- zaznajomisz się ze strukturami sterującymi
- zbudujesz pętle, instrukcje warunkowe oraz kolekcje
- błyskawicznie zlokalizujesz błędy w Twoim kodzie
- opanujesz możliwości języka PL/SQL

Helion

28952 numer katalogowy

księgarnia internetowa

<http://helion.pl>

zamówienia telefoniczne

0 801 339900

0 601 339900

Informatyka w najlepszym wydaniu

Sprawdź najnowsze promocje:
• <http://helion.pl/promocje>
Książki najchętniej czytane:
• <http://helion.pl/bestsellery>
Zamów informacje o nowościach:
• <http://helion.pl/novosci>

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
<http://helion.pl>

sięgnij po WIĘCEJ



KOD KORZYŚCI

ISBN 978-83-246-9923-0



9 788324 699230

cena: 149,00 zł

Oracle
Press