

IDŹ DO

PRZYKŁADOWY ROZDZIAŁ



SPIS TREŚCI

KATALOG KSIĄŻEK

KATALOG ONLINE

ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK

DODAJ DO KOSZYKA

CENNIK I INFORMACJE

ZAMÓW INFORMACJE
O NOWOŚCIACH

ZAMÓW CENNIK

CZYTELNIA

FRAGMENTY KSIĄŻEK ONLINE

Oracle9i. Przewodnik dla początkujących

Autorzy: Michael Abbey, Ian Abramson, Michael Corey
Tłumaczenie: Przemysław Szeremiota (rozdziały 11 – 17, dodatek A), Piotr Świerczyński (rozdziały 1 – 10)
ISBN: 83-7197-996-7
Tytuł oryginału: [Oracle9i: A Beginner's Guide](#)
Format: B5, stron: 456



Obejmuje wydania Oracle 7.x, 8i, 9i

Systemy baz danych Oracle, będące podstawą współczesnego e-biznesu, to złożone i skomplikowane rozwiązania. Jeśli chcesz poznać ich ogromny potencjał, trzymasz w ręku właściwy podręcznik, autoryzowany przez firmę Oracle.

Książkę „Oracle9i. Przewodnik dla początkujących”, wypełnioną do ostatniej strony użytecznymi wskazówkami i przykładami. Poznasz język proceduralny PL/SQL i podstawowe zagadnienia związane z administracją bazy danych, po czym następuje omówienie zaawansowanych technik zarządzania i manipulowania danymi. To kompletny podręcznik - prowadzi Cię od konfiguracji bazy danych, tworzenia tabel i wykonywania zapytań, przez tworzenie formularzy i raportów, aż po partycjonowanie danych i zarządzanie uprawnieniami użytkowników. Dodatkowo książka zawiera omówienie narzędzi SQL*Plus, Oracle Enterprise Manager i Oracle Summary Machine.

- Poznaj bazy danych Oracle9i i serwer aplikacji Oracle9iAS
- Korzystaj z pomocy Oracle Support Services i z zasobów Oracle Technology Network, twórz zgłoszenia TAR, iTAR, korzystaj z serwisu Metalink
- Twórz tabele i raporty, projektuj i uruchamiaj aplikacje korzystające z języków SQL i PL/SQL
- Poznaj przeznaczenie plików danych, dzienników odtwarzania i plików kontrolnych
- Połącz komputery w sieć Oracle Net i korzystaj z możliwości oferowanych przez przetwarzanie rozproszone
- Zarządzaj olbrzymimi tabelami, gromadź dane w hurtowni danych i przeprowadzaj zaawansowane analizy zbiorcze za pomocą narzędzi i funkcji Oracle Summary Engine
- Buduj bazy danych za pomocą zapytań DDL i DML



Spis treści

Informacje o Autorach	13
Wprowadzenie	15
Część I Pierwsze kroki.....	17
Rozdział 1. Oracle: firma i produkty	19
Terminologia	19
Firma Oracle Corporation: historia.....	19
1977: początek	20
1978: powstaje Relational Software Inc.....	21
1979: pierwsza komercyjna baza danych trafia na rynek	21
1980: powstaje firma Oracle Systems	21
1981 – 1983: pierwszy system zarządzania relacyjną bazą danych, działający na komputerach typu mainframe i minikomputerach	22
1984: wersja 4. systemu Oracle — spójność odczytu.....	22
1985: Oracle wchodzi w sektor aplikacji.....	23
1986: pierwszy system zarządzania bazą danych z funkcjami bazy rozproszonej.....	23
1987: intensywny rozwój Oracle.....	24
1988: Oracle Financials/Oracle CASE.....	24
1989: powstanie Oracle 6.2	24
1990 – 1991: przekroczona granica 1 mld dolarów	25
1992: Ray Lane w Oracle.....	25
1993: rosnąca rola działu aplikacji	26
1994 – 1995: 2 miliardy sprzedaży i komputer sieciowy.....	26
1996: Oracle wchodzi na rynek detaliczny	27
1997: pojawia się pakiet Oracle8.....	27
1998: obsługa systemu Linux.....	28
1999: pojawia się pakiet Oracle8i.....	28
2000: numer jeden.....	28
Aktualna oferta	29
Pytania do rozdziału 1.	30
Rozdział 2. Mechanizmy uzyskiwania pomocy	31
Terminologia	31
Oracle Support Services.....	33
Zgłaszanie wniosków TAR do działu OSS (starszy sposób).....	35
Przekazywanie dokumentacji towarzyszącej zgłaszaniu wniosków TAR	36

Serwis MetaLink.....	38
Biblioteki techniczne	39
Fora	41
Wnioski iTAR.....	44
Informacje reliktowe klienta.....	44
Krótki opis wniosku TAR	45
Sieć Oracle Technology Network	47
Serwis Oracle AppsNet.....	49
Grupy dyskusyjne i serwery list dyskusyjnych.....	51
Grupy dyskusyjne	52
Serwery list dyskusyjnych.....	55
Dokumentacja on-line.....	59
Inne witryny	60
Wyszukiwarki internetowe	61
Witryny godne szczególnego polecenia	63
Pytania do rozdziału 2.	65
Rozdział 3. Serwer Oracle.....	67
Terminologia	67
Architektura serwera	69
Pomocnicze procesy drugoplanowe.....	71
Proces zapisujący do plików danych (dbw0).....	71
Monitor procesów (pmon).....	72
Monitor systemu (smon).....	72
Proces zapisujący do plików dziennika powtórzeń (lgwr).....	73
Proces punktu kontrolnego (ckpt).....	73
Odtwarzacz (reco).....	73
Archiwizator (arc0)	73
Plik INIT.ora	74
Wpisy dotyczące położenia plików.....	75
Wpisy dotyczące ograniczeń.....	76
Wpisy dotyczące funkcji	76
Zmiana wartości parametrów.....	77
Plik kontrolny.....	79
Dzienniki powtórzeń	79
Pliki danych bazy danych	82
Segmenty wycofania/ przestrzeni tabel cofania	83
Najważniejsze struktury pamięci.....	84
Bufor danych.....	84
Bufor bibliotek.....	85
Blokady i zatraski.....	86
Blokady.....	86
Zatraski.....	87
Tworzenie nowej bazy danych Oracle9i	88
Pytania do rozdziału 3.	90
Część II Czas na konkrety	91
Rozdział 4. Obiekty baz danych	93
Terminologia	93
Tabele — miejsce przechowywania danych.....	95
Polecenie create table — przykład.....	95
Polecenie create table as — przykład.....	96
Perspektywy — niestandardowy wybór jednej lub większej liczby tabel.....	97
Polecenie create view — przykład.....	97

Perspektywy materializowane — perspektywy, w których są przechowywane dane	98
Polecenie create materialized view (dawniej snapshot) — przykład	98
Modyfikowanie zapytań	99
Indeksy — szybki sposób korzystania z danych	99
Zalety wstępnego sortowania	100
Indeksy unikatowe i nieunikatowe	100
Reguła 95/5	101
Indeks bitmapowy — indeks dla grup wierszy o niewielkim zróżnicowaniu	101
Wyzwalacze — programy inicjowane przez zdarzenia	101
Polecenie create trigger — przykład	102
Synonimy — pseudonimy obiektów	104
Polecenie create synonym — przykład	104
Sekwencja — szybki sposób uzyskania unikatowej liczby	105
Polecenie create sequence — przykład	105
Polecenie create role — sposób zarządzania uprawnieniami	105
Polecenie create role — przykład	106
Funkcje, procedury i pakiety	106
Polecenie create function	106
Polecenie create procedure	108
Polecenie create package	108
Inne obiekty baz danych	109
Polecenie create operator	109
Polecenie create directory	109
Polecenie create library	110
Powiązania bazodanowe	110
Polecenie create cluster	112
Pytania do rozdziału 4.	113
Rozdział 5. SQL*Plus 101.....	115
Jak uzyskać dostęp do programu SQL*Plus	115
Dostęp za pomocą wiersza polecenia	116
Dostęp przy użyciu ikon	117
Kończenie sesji programu SQL*Plus	117
Instrukcje Data Definition Language (DDL)	117
Instrukcja create/drop	118
Typy danych	119
Komenda describe	119
Wartość not null	120
Instrukcje Data Manipulation Language (DML)	120
Instrukcja insert	121
Instrukcja select	122
Środowisko programu SQL*Plus	127
Łączenie tabel	131
Klucze główne i klucze obce	131
Klauzula break on	132
Klauzula break on z opcją skip	133
Obliczanie wartości w kolumnach w punktach łamania	134
Komenda break on report	135
Pytania do rozdziału 5.	136
Rozdział 6. PL/SQL 101	139
Terminologia	140
PL/SQL: język programowania opracowany przez firmę Oracle	141
Zestaw znaków języka PL/SQL	143
Obsługiwane znaki	143
Operatory arytmetyczne i relacyjne	143

Struktura języka PL/SQL.....	144
Zmienne języka PL/SQL.....	145
Struktury kontrolne	148
Struktury logiczne if	149
Wyrażenia case	152
Pętle.....	152
Język SQL w programach w języku PL/SQL.....	155
Kursory.....	155
Pętla for kursora.....	157
Obsługa wyjątków.....	158
Składowane procedury i funkcje	160
Składowane procedury.....	160
Funkcje	164
Podstawowe mechanizmy usuwania błędów	165
Dalsze kroki.....	166
Pytania do rozdziału 6.	167
Rozdział 7. DBA 101	169
Terminologia.....	169
Co to jest baza danych?	170
Co to jest instancja Oracle?	171
Globalny obszar systemu (SGA).....	171
Procesy drugoplanowe systemu Oracle.....	172
Polecenie startup open	173
Polecenie shutdown.....	175
Przestrzeń tabel w Oracle9i.....	176
Tworzenie przestrzeni tabel — klauzula extent management dictionary	176
Tworzenie przestrzeni tabel — polecenie extent management local autoallocate.....	177
Polecenie create undo tablespace	178
Polecenie alter tablespace add data file	179
Polecenie alter tablespace offline	179
Usuwanie przestrzeni tabel.....	180
Segmenty wycofania.....	180
Polecenie create rollback segment.....	180
Polecenie alter rollback segment online	182
Ręczne zmniejszanie segmentu wycofania	182
Polecenie drop rollback segment.....	183
Dzienniki powtórzeń	183
Zwielokrotnione pliki dziennika powtórzeń	183
Usuwanie dziennika powtórzeń.....	184
Dodawanie dziennika powtórzeń.....	185
Pliki kontrolne — lista kontrolna bazy danych.....	185
Tworzenie plików kontrolnych	186
Tworzenie konta użytkownika.....	186
Polecenie grant connect, resource	187
Pytania do rozdziału 7.	188
Część III Wykraczamy poza podstawy.....	189
Rozdział 8. Więcej o programie SQL*Plus.....	191
Terminologia.....	192
Wdrażanie programu SQL*Plus w środowisku produkcyjnym.....	192
Umieszczanie komentarzy w kodzie przy użyciu instrukcji rem, -- i /*...*/	192
Pliki bazowe programu SQL*Plus.....	193

Operatory zbiorowe union, intersect i minus	196
Operator union	197
Operator union all.....	197
Operator minus	197
Operator intersect.....	198
Nie mieszaj ogórków z dżemem	198
Edytowanie wiersza poleceń w programie SQL*Plus.....	198
Korzystanie z edytora wiersza polecenia.....	199
Gdyby tak można było użyć edytora VI lub Emacs.....	199
Tabela dual.....	200
Standardowe funkcje Oracle.....	200
Funkcje i operatory matematyczne.....	200
Funkcje znakowe	202
Dane typu date.....	203
Funkcje daty w programie SQL*Plus	204
Funkcje grupowe.....	206
Wyszukiwanie powtarzających się danych za pomocą klauzuli group by	209
Usuwanie powtarzających się danych za pomocą funkcji group by	210
Generowanie kodu SQL za pomocą innego fragmentu kodu SQL.....	212
Generowanie plików danych za pomocą kodu SQL.....	213
Zapytanie wewnątrz zapytania.....	214
Instrukcja decode.....	216
Aktualizacja przy użyciu instrukcji update	217
Pytania do rozdziału 8.	218
Rozdział 9. Więcej o języku PL/SQL.....	219
Terminologia	220
Pakiety i przeciążania podprogramu	220
Zaawansowane funkcje obsługi błędów w programach w języku PL/SQL.....	222
Wyjątki definiowane przez użytkownika.....	222
Zmienne przeznaczone do obsługi błędów dostępne w oprogramowaniu Oracle.....	223
Transakcje autonomiczne.....	225
Bezpieczeństwo danych z poziomu PL/SQL.....	226
Pakiety dostarczone przez firmę Oracle.....	226
Pakiet utl_file.....	227
Dynamiczny SQL.....	230
Pytania do rozdziału 9.	233
Rozdział 10. Więcej o administratorze bazy danych.....	235
Terminologia	236
Tworzenie kopii zapasowych i odtwarzanie	236
Eksport	236
Rola mechanizmu eksportu w tworzeniu kopii zapasowych.....	237
Parametry programu eksportu.....	237
Tryby pracy programu eksportu.....	239
Typy eksportu.....	243
Import.....	243
Rola importu w odtwarzaniu.....	243
Parametry programu importu.....	244
Tryby działania programu importu	245
Typy importu.....	247
Funkcje odtwarzania nośników.....	247
Tworzenie kopii zapasowych „na gorąco” i „na zimno”	248
Praca w trybie archiwizacji dzienników powtórzeń.....	249
Zapisywanie kopii zapasowej „na gorąco”	251
Odtwarzanie nośników — przykład.....	252
Pytania do rozdziału 10.....	255

Rozdział 11. Oracle Enterprise Manager.....	257
Terminologia	257
Możliwości OEM — przegląd	258
Uruchamianie	261
Zatrzymanie.....	263
Zarządzanie przestrzenią tabel.....	264
Zmiana rozmiaru pliku danych	266
Dodanie pliku danych	267
Zmniejszanie rozmiaru pliku danych.....	267
Zarządzanie kontami użytkowników	268
Tworzenie konta użytkownika	268
Uprawnienia do zajmowania przestrzeni w bazie danych.....	270
Przyznawanie uprawnień do obiektów bazy danych.....	271
Zarządzanie obiektami	273
Zarządzanie obiektami — arkusz SQL*Plus Worksheet.....	275
Pytania do rozdziału 11.....	277
Rozdział 12. Przetwarzanie rozproszone	279
Terminologia	279
Partycjonowanie aplikacji z wykorzystaniem przetwarzania rozproszonego.....	280
Oracle Net	281
Plik listener.ora	281
Plik tnsnames.ora	283
Program Network Configuration Assistant	284
Rozmieszczenie pliku tnsnames.ora	288
Nawiązywanie połączenia w środowisku Oracle Net.....	289
Pytania do rozdziału 12.....	290
Część IV I jak Internet.....	293
Rozdział 13. I w Oracle9i.....	295
Terminologia	295
Internetowa baza danych Oracle	297
Komunikacja z bazą danych.....	298
Java w bazie danych.....	301
Wysoka dostępność.....	302
Zabezpieczenie przed awarią systemu	303
Zabezpieczenie przed awarią dysku.....	304
Zabezpieczenie przed błędami użytkownika.....	304
Zarządzanie planowanymi przestojami	305
Architektura Real Application Clusters.....	306
Internetowy system plików (IFS)	307
Pytania do rozdziału 13.....	308
Rozdział 14. Wszędzie WWW	309
Terminologia	309
Serwer aplikacji Oracle9i.....	311
Komponenty Communication Services	313
Komponenty Business Logic Services	316
Komponenty Presentation Services	317
Usługi buforowania	321
Usługi zarządzania treścią	323
Usługi Oracle Portal	323
Usługi wspomagania decyzji	325
Java w bazie danych.....	326
Pytania do rozdziału 14.....	329

Część V Oracle9i dla już nie początkujących	331
Rozdział 15. Formularze i raporty	333
Terminologia	333
Przykładowe tabele	334
Oracle Forms i Oracle Reports — wprowadzenie	335
Oracle Forms i Oracle Reports — składniki	335
Inicjowanie połączenia z bazą danych	336
Formularze — Forms Builder	336
Kreator bloku danych	338
Kreator układu	339
Edycja istniejącego formularza	342
Raporty — Raport Builder	348
Kreator Report Wizard	348
Modyfikacja raportu	351
Pytania do rozdziału 15	353
Rozdział 16. Partycjonowanie danych	355
Terminologia	355
Po co partycjonować dane?	356
Rozmiar woluminu danych	357
Łatwość zarządzania partycjonowanymi danymi	358
Zwiększenie wydajności	358
Partycjonowanie zakresowe	359
Wybór klucza partycjonowania	359
Zakresowe partycjonowanie tabeli — kod SQL	362
Indeksowanie tabel partycjonowanych	364
Indeksy partycjonowane lokalnie	364
Indeksy partycjonowane globalnie	367
Indeks lokalny czy globalny?	368
Indeksy partycjonowane prefiksowane i nieprefiksowane	371
Partycjonowanie według listy	371
Partycjonowanie haszowane	373
Jaki zastosować model partycjonowania?	374
Liczba wierszy	374
Metody dostępu	376
Model partycjonowania hybrydowego	377
Pytania do rozdziału 16	378
Rozdział 17. Hurtownia danych. Funkcje analizy zbiorczej	381
Terminologia	382
Czym jest hurtownia danych?	383
Projektowanie hurtowni danych	384
Wymiarowy projekt bazy danych	386
Partycjonowanie hurtowni danych	387
Wykonywanie kopii zapasowej hurtowni danych	390
Wypełnianie hurtowni danych	391
Wczytywanie danych — program SQL*Loader	393
Wczytywanie danych — tabele zewnętrzne	395
Wczytywanie danych — PL/SQL	397
Funkcje związane z przetwarzaniem danych w hurtowniach danych Oracle9i	400
Perspektywy zmaterializowane	400
Rozszerzone operacje agregujące	413
Funkcja rollup	416
Funkcja cube	417

Funkcje szeregujące.....	418
Funkcje ramy.....	422
Funkcje statystyczne.....	424
Pytania do rozdziału 17.....	425

Dodatki.....427

Dodatek A	Odpowiedzi na pytania do rozdziałów.....	429
	Pytania do rozdziału 1.....	429
	Pytania do rozdziału 2.....	429
	Pytania do rozdziału 3.....	430
	Pytania do rozdziału 4.....	430
	Pytania do rozdziału 5.....	431
	Pytania do rozdziału 6.....	431
	Pytania do rozdziału 7.....	432
	Pytania do rozdziału 8.....	432
	Pytania do rozdziału 9.....	432
	Pytania do rozdziału 10.....	433
	Pytania do rozdziału 11.....	433
	Pytania do rozdziału 12.....	434
	Pytania do rozdziału 13.....	434
	Pytania do rozdziału 14.....	435
	Pytania do rozdziału 15.....	435
	Pytania do rozdziału 16.....	436
	Pytania do rozdziału 17.....	436
	Skorowidz.....	439

Rozdział 4.

Obiekty baz danych

W niniejszym rozdziale omówiono definicje i przykłady obiektów baz danych, najczęściej spotykanych podczas pracy z bazami danych Oracle9i. Opisano w nim również pewne obiekty, z których użytkownik zapewne nie będzie korzystał ale o których istnieniu należy wiedzieć.

Przez obiekt bazy danych można rozumieć każdy element bazy danych, który jest tworzony za pomocą instrukcji SQL `create`. Instrukcje `create` i inne instrukcje DDL zostały dokładniej zaprezentowane w rozdziale 5. Pomyślne wykonanie instrukcji `create` powoduje utworzenie nowego obiektu bazy danych. Obiekty te mogą mieć różny rozmiar i format. W tym rozdziale zostaną omówione sposoby tworzenia każdego z nich. W przystępny zostaną również opisane role, jakie te obiekty pełnią w bazie danych.

W niniejszym rozdziale omówiono następujące zagadnienia:

- ◆ tabele;
- ◆ perspektywy, w tym perspektywy materializowane;
- ◆ indeksy;
- ◆ wyzwalacze;
- ◆ synonimy;
- ◆ sekwencje;
- ◆ role;
- ◆ funkcje, procedury i pakiety.

Terminologia

Znajomość poniższej terminologii znacznie ułatwi zrozumienie treści zawartych w tym rozdziale.

- ◆ Baza danych prowadzi *słownik danych*, który zawiera wszystkie informacje na temat sposobu przechowywania danych w bazie, miejsca ich przechowywania oraz możliwości wykorzystania tych danych przez bazę.
- ◆ *DDL* to skrót oznaczający *język definicji danych* (*Data Definition Language*). Są to polecenia SQL rozpoczynające się od instrukcji *create*, *revoke*, *grant* i *drop*. Służą one do tworzenia i usuwania obiektów baz danych, jak również tworzenia zezwoleń dostępu do baz danych i ich obiektów.
- ◆ *DML* — *język przetwarzania danych* (*Data Manipulation Language*) to polecenia SQL rozpoczynające się od instrukcji *select*, *insert*, *delete* lub *update*. Służą one do przetwarzania zawartości bazy danych.
- ◆ *Zezwolenia* to uprawnienia przyznawane przez właścicieli obiektów, na mocy których inni użytkownicy mogą korzystać z danych takiego właściciela. Istnieją dwie kategorie zezwoleń: *zezwoleń na poziomie obiektu* i *zezwoleń na poziomie systemu*. Zezwolenia na poziomie obiektu oznaczają, że użytkownik może wybierać, wstawiać, aktualizować lub usuwać dane z obiektu bazy danych, takiego jak tabela. Przykładowe zezwolenia na poziomie systemu to umożliwienie użytkownikowi bazy danych nawiązywanie połączenia z bazą danych lub tworzenia obiektu w bazie.
- ◆ *Indeks* to niewielka kopia tabeli systemu Oracle, przechowywana w bazie po wstępnym posortowaniu. Indeksy tabel umożliwiają uzyskanie bardzo szybkiego dostępu do danych zawartych w tabelach. Pełnią one rolę podobną do indeksów w książce, ułatwiających znalezienie określonego wyrazu lub wyrażenia.
- ◆ *Więzy integralności* to zasady zapewniające logiczną spójność danych w bazie danych. Przykładowo, aby uprościć identyfikowanie danych klienta w bazie danych, przypisuje się mu unikatowy identyfikator.
- ◆ *Synonim* to alternatywna nazwa obiektu znajdującego się w bazie danych.
- ◆ *Tabela* to obiekt bazy danych, w którym są umieszczane dane użytkownika. W słowniku danych systemu Oracle9i przechowywane są informacje o każdej z tabel. Dzięki tym informacjom baza Oracle umożliwi obsługę danych umieszczonych w tabelach.
- ◆ *Wywalacze* bazy danych to programy składowane w bazie danych, które są uaktywniane przez określone zdarzenia. Przykładem takiego zdarzenia może być wstawienie wiersza danych do tabeli.
- ◆ Dzięki *perspektywom* użytkownicy baz danych mogą wyświetlać zawartość wybranych przez siebie tabel znajdujących się w bazie danych. *Perspektywa* jest tworzona na podstawie instrukcji SQL przechowywanej w tej bazie. Podczas korzystania z tej perspektywy następuje wykonanie instrukcji, a wyniki zapytania są wyświetlane na ekranie.
- ◆ *Rola* to zbiór uprawnień przyznawanych użytkownikom. Użytkownik staje się członkiem danej roli, dziedzicząc w ten sposób należące do niej uprawnienia.

Tabele — miejsce przechowywania danych

Tabela to obiekt bazy danych, w którym przechowywane są wszystkie dane. Każdy element danych załadowany do bazy danych Oracle musi zostać umieszczony w tabeli. W praktyce wszystkie informacje wymagane przez bazę danych Oracle do pracy są przechowywane w szeregu tabel, które są popularnie określane mianem *słownika danych*. Słowniki danych to swego rodzaju tabele tabel. Informują one bazę danych o rodzaju danych przechowywanych w bazie, ich położeniu oraz możliwościach wykorzystania przez bazę danych.

Tabela składa się z kolumn. Każda kolumna musi mieć unikatową nazwę w obrębie tabeli oraz musi mieć przypisany typ danych (np. `varchar2`, `date` lub `number`) wraz z określeniem długości wpisu (który może być wyznaczany typem danych, jak w przypadku danych typu `date`). Każda kolumna tabeli może być również określona parametrem `null` lub `not null`. Parametr `not null` oznacza, że w kolumnie muszą zostać umieszczone dane. Innymi słowy w przypadku wierszy danych, które mają zostać umieszczone w tabeli, wszystkie kolumny opisane parametrem `not null` muszą zawierać prawidłowe wartości danych.

Aby wymusić stosowanie określonych zasad spójności logicznej (zapewniających integralność danych) wobec danych w tabeli, system Oracle⁹ⁱ umożliwia tworzenie dla poszczególnych tabel więzów integralności i wyzwalaczy. Wyzwalacze zostaną dokładniej omówione w dalszej części tego rozdziału.

Polecenie `create table` — przykład

Oto kod tworzący przykładową tabelę o nazwie ZWIERZAKI:

```
SQL> CREATE TABLE ZWIERZAKI (
  2   ID_ZWIERZAKA          INTEGER          PRIMARY KEY,
  3   GAT_ZWIERZAKA        VARCHAR2 (20) NOT NULL,
  4   NAZWA_ZWIERZAKA      VARCHAR2 (20),
  5*  PLEC_ZWIERZAKA_MZ    CHAR(1)          CHECK (PLEC_ZWIERZAKA_MZ IN ('M', 'Z'))
SQL> /
Tabela została utworzona.
```

Tabela ZWIERZAKI zawiera cztery kolumny. Pierwsza kolumna nosi nazwę `ID_ZWIERZAKA`, a znajdujące się w niej dane będą typu `integer`. Na kolumnę narzucono więzy integralności `primary key` (klucz główny). W ten sposób baza danych będzie wymuszała zasadę spójności logicznej określającą, że każdy identyfikator zwierzęcia musi być unikatowy w tej tabeli. Innymi słowy, żadne wpisy w tabeli ZWIERZAKI nie mogą korzystać z tego samego identyfikatora. W przypadku próby umieszczenia identyfikatora będącego duplikatem już istniejącego identyfikatora zostanie zwrócony komunikat o błędzie.

Następna kolumna to `GAT_ZWIERZAKA`, której dane są typu `varchar2` o maksymalnej długości 20 znaków. Atrybut `not null` oznacza, że każdy wpis dodany do tabeli ZWIERZAKI musi zawierać prawidłową wartość dla kolumny `GAT_ZWIERZAKA`.

Trzecia kolumna to NAZWA_ZWIERZAKA, której dane również są typu varchar2, a ich maksymalna długość nie może przekraczać 20 znaków. W odróżnieniu od kolumny GAT_ZWIERZAKA wpisy dodawane do tabeli ZWIERZAKI mogą ale nie muszą zawierać wartości dla tej kolumny.

Ostatnia kolumna tabeli ZWIERZAKI to PLEC_ZWIERZAKA_MZ. Dane w tej kolumnie są typu char, a ich długość wynosi dokładnie jeden znak. Podobnie jak w kolumnie ID_ZWIERZAKA, stosowane jest ograniczenie integralności. Opcja check oznacza, że w tej kolumnie dozwolone są jedynie wartości M lub Z. Po wpisaniu wartości w tej kolumnie baza danych sprawdza, czy jest to właśnie jedna z dwóch akceptowanych wartości.

Polecenie create table as — przykład

Bardzo przydatną funkcją w bazie danych Oracle⁹ⁱ jest możliwość tworzenia tabeli na podstawie istniejącej tabeli. Za pomocą tego mechanizmu można szybko utworzyć kopię całej tabeli lub tylko wybranego fragmentu. Jest to również doskonałe narzędzie tworzenia środowiska testowego. Oto bardzo prosty przykład zastosowania tej funkcji:

```
SQL> CREATE TABLE duzy_nowy_nabor
  2   AS select *
  3     from nowy_nabor
  4*   where kod_woj = 'PM'
SQL> /
Tabela została utworzona.
```

Spójrzmy teraz na zawartość tabeli wyjściowej:

```
SQL> select * from nowy_nabor;
```

NAZWISKO	WOJ	DATA_ZATR	WYNAGRODZ
-----	---	-----	-----
Jacek	PM	01-STY-01	20000
Nowak	LD	10-CZE-01	30000
kowalski	CT	15-SIE-01	40000
Ania	MA	12-GRU-01	50000
marek	ZP	12-LIS-01	60000
wiesiek	SL	11-MAJ-01	90000
tomek	PM	15-SIE-01	55000
pawel	LD	02-LUT-01	30000
zbyszek	LD	05-CZE-01	20000
magda	PM	26-CZE-01	75000
basia	PM	08-MAJ-01	80000

11 wierszy zostało wybranych.

Poniżej pokazano nowo utworzoną tabelę DUZY_NOWY_NABOR. Zawiera ona tylko wpisy spełniające warunek kod_woj = 'PM'.

```
SQL> select * from duzy_nowy_nabor;
```

NAZWISKO	WOJ	DATA_ZATR	WYNAGRODZ
-----	---	-----	-----
Jacek	PM	01-STY-01	20000
Tomek	PM	15-SIE-01	55000
Magda	PM	26-CZE-01	75000
Basia	PM	08-MAJ-01	80000

Perspektywy — niestandardowy wybór jednej lub większej liczby tabel

Perspektywa to niestandardowy zbiór danych pochodzących z jednej lub więcej tabel bazowych. Tabele bazowe z kolei to tabele lub również perspektywy. W odróżnieniu od tabeli, perspektywa nie zawiera danych a jedynie składowaną instrukcję SQL. Gdy użytkownik uruchamia zapytanie korzystające z perspektywy, baza danych otwiera słownik danych, odszukuje składowaną instrukcję SQL i wykonuje ją. Dane znalezione w wyniku realizacji zapytania są prezentowane w formie tabeli.

Perspektywy działają w sposób transparentny — użytkownik może odnosić wrażenie, że ma do czynienia z tabelą. Podobnie jak w przypadku tabeli, w perspektywie można wstawiać, aktualizować, usuwać oraz wybierać dane¹. Wszelkie zmiany wprowadzane w perspektywie są uwzględniane w tabelach bazowych.

Perspektywy są wykorzystywane z wielu powodów. Można na przykład pozwolić pracownikowi zajmującemu się wypłatami na uzyskiwanie dostępu w tabeli płac do informacji o warunkach zatrudnienia poszczególnych pracowników ale nie do informacji o ich wynagrodzeniach. Innym razem perspektywa może posłużyć do ukrycia złożoności danych. Można na przykład umożliwić użytkownikom dostęp do perspektywy, podczas gdy instrukcja SQL, za pomocą której utworzono tą perspektywę, wykorzystuje skomplikowane złączenie wielu tabel. Dzięki temu użytkownicy omijają najbardziej złożone elementy relacyjnych baz danych.

Polecenie create view — przykład

W przykładzie poniżej zademonstrowano sposób łączenia dwóch tabel:

```
SQL> CREATE VIEW wid_dzial_prac AS
  2   SELECT e.imie_prac, e.nazw_prac, d.nazwa_dzial
  3     FROM pracownik e,
  4          dzial d
  5*  WHERE e.id_dzial = d.id_dzial
SQL> /
Perspektywa została utworzona.
```

Teraz dla perspektywy zostanie wykonana instrukcja select. Z punktu widzenia użytkownika perspektywa wid_dzial_prac funkcjonuje identycznie jak tabela, w rzeczywistości jednak jest to składowana w bazie danych instrukcja SQL, z którą do momentu wykonania nie są powiązane żadne dane.

```
SQL> SELECT * FROM wid_dzial_prac;
IMIE_PRAC      NAZW_PRAC      NAZWA_DZIAL
-----
AGNIESZKA     NOWAK          Finance
KRZYSZTOF    JACKOWSKI     Sprzedaz
MARIA        NIEDZIELSKI   Kadry
MICHAL       CZARNY        Marketing
```

¹ Z pewnymi ograniczeniami dotyczącymi wstawiania, modyfikacji i usuwania — *przypr. tłum.*

Perspektywy materializowane — perspektywy, w których są przechowywane dane

W odróżnieniu od zwykłej perspektywy, która zawiera jedynie instrukcje SQL, perspektywa materializowana zawiera wiersze danych, będące efektem wykonania zapytania SQL w jednej lub więcej tabel bazowych. Każda zmiana w tabeli bazowej jest odnotowywana w osobnym dzienniku w bazie danych. Perspektywy materializowane można skonfigurować w taki sposób, aby automatycznie dokonywały synchronizacji z tymi tabelami, podlegając aktualizacji w odstępach czasu określonych przez użytkownika. Perspektywy mogą być przechowywane w tej samej bazie danych, jak źródłowe tabele bazy lub też w zupełnie innych zdalnych bazach danych.

Poniżej zostaną pokazane przykłady, w jaki sposób można wykorzystywać perspektywy materializowane w hurtowni danych. Często są używane do wstępnego obliczania i przechowywania danych zbiorczych, takich jak sumy czy średnie. Jeśli perspektywa materializowana dotyczyłaby miesięcznej sprzedaży, gdzie do tabel bazowych byłyby wpisywane dane o sprzedaży w nowym miesiącu, w perspektywie materializowanej byłyby dokonywane automatyczne aktualizacje wartości zbiorczych uwzględniające prosty sposób przyspieszenia wykonywania zapytań w dużych hurtowniach danych.

W środowiskach rozproszonych stosowanie perspektyw materializowanych umożliwia dokonywanie replikacji danych w lokalizacjach rozproszonych oraz synchronizacji aktualizacji między tymi lokalizacjami. W środowisku mobilnym perspektywy tego typu mogą być wykorzystywane do pobierania podzbiorów danych z centralnego serwera do klientów mobilnych oraz dokonywania okresowych aktualizacji między serwerem a klientami.

Polecenie `create materialized view` (dawniej `snapshot`) — przykład

Najpierw wszystkie tabele, na których jest oparta perspektywa materializowana, muszą być odnotowane w dzienniku. Oto jeden z możliwych sposobów realizacji tego zadania:

```
SQL> CREATE MATERIALIZED VIEW LOG ON dzial
  2   WITH PRIMARY KEY,
  3   ROWID (nazwa_dzial)
  4*  INCLUDING NEW VALUES
SQL> /
Dziennik perspektywy materializowanej został utworzony.
```

```
SQL> CREATE MATERIALIZED VIEW LOG ON nowy_nabor
  2   WITH PRIMARY KEY,
  3   ROWID (id_dzial, wynagrodz)
  4*  INCLUDING NEW VALUES
SQL> /
Dziennik perspektywy materializowanej został utworzony.
```

Teraz zostanie utworzona sama perspektywa materializowana:

```
SQL> CREATE MATERIALIZED VIEW mtw_dzial_wynagrodz
2     REFRESH FAST ON COMMIT
3     ENABLE QUERY REWRITE
4 AS select d.nazwa_dzial,
5           sum(n.wynagrodz) as wynagrodz_lacz
6     from dzial d,
7          nowy_nabor n
8     where d.id_dzial = n.id_dzial
9     group by d.nazwa_dzial;
```

Perspektywa materializowana została utworzona.

Modyfikowanie zapytań

Jeśli perspektywy materializowane są przechowywane w tej samej bazie danych co tabele podstawowe, optymalizator zapytań może korzystać z funkcji modyfikowania zapytań. Oznacza to, że gdy optymalizator stwierdzi, że może uzyskać wymagane dane szybciej dzięki wykorzystaniu perspektywy materializowanej a nie tabel źródłowych, wskazywanych przez zapytanie, zapytanie to (instrukcja SQL) zostanie zmodyfikowane w taki sposób, że będzie korzystało z materializowanej perspektywy a nie z oryginalnej tabeli źródłowej.

Możliwość modyfikowania oryginalnego zapytania znacznie zwiększa sprawność działania bazy danych. Funkcji tej można używać do tworzenia sprawozdawczej bazy danych (perspektyw materializowanych) opartej na produkcyjnej bazie danych. W takim przypadku zmiana tabel produkcyjnych nie powodowałaby modyfikacji perspektyw materializowanych.

Indeksy — szybki sposób korzystania z danych

Podobnie jak w przypadku indeksów pomagających w szybszym przeszukiwaniu książki, indeks tabeli pomaga szybciej wyszukiwać dane. Indeks może działać znacznie szybciej niż tabela źródłowa, ponieważ jest znacznie mniejszą kopią podzbioru danych takiej tabeli.

Na przykład, może istnieć tabela o nazwie ALFABET, zawierająca 26 kolumn (od A do Z). Dla pierwszych trzech kolumn utworzono indeks, nazwany na przykład ABC. Tabela ma milion wierszy, a zadanie polega na wyszukaniu wszystkich wystąpień słowa Zupa znajdujących się w kolumnie A. Aby przejrzeć zawartość kolumny A, standardowe wyszukiwanie musi obejmować wszystkie kolumny tabeli.

Indeks jest kompletną kopią zawartości kolumn A, B i C, zatem przejrzanie zawartości kolumny A wymaga zanalizowania jeszcze tylko dwóch innych kolumn — B i C. Można sobie wyobrazić, o ile mniej pracy będzie musiała wykonać baza danych w związku z przeszukiwaniem miliona wierszy w kolumnach od A do C w porównaniu z przeszukiwaniem kolumn od A do Z.

Zalety wstępnego sortowania

W relacyjnych bazach danych zawartość tabeli nie jest poddawana żadnemu wstępnemu sortowaniu. Być może Czytelnik zetknął się z terminem księgowym *FIFO* — *First In is First Out* (pierwsze na wejściu jest pierwsze na wyjściu). W środowisku baz danych stosowane jest pojęcie *FIFL* — *First In is First Loaded* (pierwsze na wejściu jest pierwsze ładowane). Indeks jest zawsze wstępnie posortowany. W przykładzie powyżej próba znalezienia słowa Zupa w kolumnie A tabeli źródłowej oznacza, że trzeba przejrzeć cały milion wierszy. Jednocześnie wiadomo, że indeks jest wstępnie posortowany. Wykorzystanie tej informacji pozwoli na znaczne przyspieszenie pracy bazy danych.

Utworzenie indeksu kolumn A, B i C w rzeczywistości powoduje utworzenie następujących indeksów:

```
A  
AB  
ABC
```

Oznacza to, że powstaje indeks kolumny A, indeks połączonych kolumn A i B oraz indeks połączonych kolumn A, B i C.

Nie powstają następujące indeksy:

```
B  
BC  
C
```

Należy również stale pamiętać, że indeks jest niewielką kopią tabeli. W przypadku indeksu ABC każda zmiana zawartości którejkolwiek z kolumn powoduje konieczność aktualizacji indeksu. Zawsze podczas tworzenia indeksu baza danych musi uwzględniać wszystkie zmiany wprowadzone w tabeli źródłowej, na której jest oparty indeks.

Indeksy unikatowe i nieunikatowe

Indeks standardowy może mieć formę indeksu unikatowego lub indeksu nieunikatowego. Indeks unikatowy nie może powtarzać się dla różnych wpisów. Indeks nieunikatowy może być określony dla wielu różnych wpisów w tej samej tabeli.

```
SQL> CREATE UNIQUE INDEX ind_nazwa_dzial  
2* ON dzial (nazwa_dzial)  
SQL> /
```

Indeks został utworzony.

Klucz główny (primary key)

Inny sposób uniknięcia duplikowania wpisów polega na zastosowaniu podczas tworzenia tabeli klucza głównego. Użycie tego ograniczenia stanowi dla bazy danych informację, iż wskazane kolumny nie mogą zawierać zduplikowanych wpisów. W przykładzie poniżej nie będzie tworzona nowa tabela a jedynie zostanie dodany klucz główny dla istniejącej tabeli.

```
SQL> ALTER TABLE dzial
      2* ADD CONSTRAINT kg_nazwa_dzial PRIMARY KEY (nazwa_dzial)
SQL> /
Tabela została zmieniona.
```

Gdy kolumny zostaną określone jako klucz główny, usunięcie tego ograniczenia jest bardzo trudne.

```
SQL> DROP INDEX kg_nazwa_dzial
SQL> /
DROP INDEX kg_nazwa_dzial
      *
BLAD w linii 1:
```

ORA-02429: nie można usunąć indeksu odpowiedzialnego za klucz unikatowy/główny.

Reguła 95/5

W pewnych sytuacjach zastosowanie indeksu może przynieść spowolnienie pracy bazy danych. Reguła 95/5 umożliwia dokonywanie pomiaru skuteczności działania indeksów. Jeśli wynikiem wykonania zapytania będzie zwrócenie nie więcej niż 5% wierszy tabeli, indeks jest właściwie zawsze najszybszym sposobem wyszukiwania danych. Jeśli zaś wynikiem będzie wyszukanie ponad 5% wszystkich danych, lepiej nie korzystać z indeksu.

Indeks bitmapowy — indeks dla grup wierszy o niewielkim zróżnicowaniu

Indeks bitmapowy utworzono dla celów obsługi funkcji hurtowni danych. Czasami zbiór danych charakteryzuje się bardzo niewielkim zróżnicowaniem pewnych wierszy. Przykładem może być zbiór, w którym jednym z kluczowych kryteriów jest płeć. Zwykły indeks jest tutaj nieprzydatny. W wyniku każdego zapytania liczba zwracanych wierszy zawsze byłaby większa niż 5% całości. Poniżej zostanie pokazany efekt zastosowania indeksu bitmapowego:

```
SQL> L
1 CREATE BITMAP INDEX bi_plec_zwierzaka_mz
2* ON zwierzaki(plec_zwierzaka_mz)
SQL> /
```

Indeks został utworzony.

Wyzwalacze — programy inicjowane przez zdarzenia

Wyzwalacze to programy przechowywane w bazie danych, które są wykonywane po zaistnieniu określonego zdarzenia. Mogą one być napisane w języku PL/SQL, Java lub C. Wyzwalacze są definiowane w systemie Oracle, a ich uruchomienie odbywa się w momencie wykonania instrukcji insert, update lub delete wobec powiązanej z nimi tabeli lub perspektywy oraz przy zdarzeniu związanym z bazą danych.

Wyzwalacze mogą być wykorzystywane do wymuszania zabezpieczeń bazy danych, zapobiegania realizacji nieprawidłowych transakcji, wymuszania więzów integralności, wykonywania czynności obserwacyjnych czy nawet utrzymywania repliki tabeli.

Polecenie create trigger — przykład

Poniżej zostanie zaprezentowany przykład użycia wyzwalacza do zapisu obserwacji wszystkich działań wykonywanych na tabeli ZWIERZAKI. Działania te będą rejestrowane w tabeli DZIENNIK.

```
SQL> descr dziennik;
```

Nazwa	Null?	Typ
ID_DZIEN	NOT NULL	NUMBER
TABELA_DZIEN	NOT NULL	VARCHAR2(20)
DML_DZIEN		VARCHAR2(20)
ID_KLUCZA_DZIEN		NUMBER(38)
DATA_DZIEN		DATE
NAZWA_UZYTK_DZIEN		VARCHAR2(20)

Teraz zostanie utworzony wyzwalacz, który będzie rejestrował wszystkie zdarzenia insert, update lub delete w tabeli DZIENNIK:

```
CREATE OR REPLACE TRIGGER wyzw_zwierzaki_gat_zwierzaka
  BEFORE
  DELETE OR INSERT OR UPDATE
  ON zwierzaki
  FOR EACH ROW
BEGIN
  if INSERTING then
    insert into dziennik
      (id_dzien, tabela_dzien, dml_dzien,
       id_klucza_dzien, nazwa_uzytk_dzien , data_dzien)
    values
      (kol_id_dzien.nextval, 'DZIENNIK', 'INSERT',
       :nowe.id_zwierzaka, user , sysdate);
  elsif DELETING then
    insert into dziennik
      (id_dziennika, tabela_dzien, dml_dzien,
       id_klucza_dzien, nazwa_uzytk_dzien , data_dzien)
    values
      (kol_id_dzien.nextval, 'DZIENNIK', 'DELETE',
       :stare.id_zwierzaka, user , sysdate);
  else
    insert into dziennik
      (id_dzien, tabela_dzien, dml_dzien,
       id_klucza_dzien, nazwa_uzytk_dzien , data_dzien)
    values
      (kol_id_dzien.nextval, 'DZIENNIK', 'UPDATE',
       :stare.id_zwierzaka, user , sysdate);
  end if;
```

```

EXCEPTION
  WHEN others THEN
    raise_application_error(-20000, 'BLAD wyzw_zwierzaki_gat_zwierzaka: '
      ||SQLERRM);

END wyzw_zwierzaki_gat_zwierzaka;
/

```

Gdyby użytkownik teraz zajął do tabeli **DZIENNIK**, stwierdziłby, że jest ona pusta. Przed wykonaniem komend `update`, `insert` i `delete` warto również sprawdzić zawartość tabeli **ZWIERZAKI**.

```

SQL> select * from dziennik
SQL> /

```

nie wybrano żadnych wierszy

```

SQL> select * from zwierzaki;

```

ID_ZWIERZAKA	GAT_ZWIERZAKA	NAZWA_ZWIERZAKA	PLEC_ZWIERZAKA_MZ
1	PIES	FAFIK	M
2	KOT	MRUCZEK	M

Teraz można już wykonać komendy `insert`, `update` i `delete` na tabeli **ZWIERZAKI**. Jak zapewne Czytelnik pamięta, są to komendy, które spowodują uaktywnienie wyzwalacza bazy danych.

```

SQL> insert into zwierzaki (id_zwierzaki, gat_zwierzaka, nazwa_zwierzaka, plec_zwierzaka_mz)
2*      values (3, 'KON', 'BLYSKAWICA', 'Z')
SQL> /

```

1 wiersz został utworzony.

```

SQL> update zwierzaki
2      set nazwa_zwierzaka = 'SWIETA',
3          plec_zwierzaka_mz = 'Z'
4*  where id_zwierzaka = 1
SQL> /

```

1 wiersz został zmodyfikowany.

```

SQL> delete from zwierzaki
2*  where id_zwierzaka = 2
SQL> /

```

1 wiersz został usunięty.

Oto aktualny stan tabeli **ZWIERZAKI**:

```

SQL> select * from zwierzaki;

```

ID_ZWIERZAKA	GAT_ZWIERZAKA	NAZWA_ZWIERZAKA	PLEC_ZWIERZAKA_MZ
1	PIES	SWIETA	Z
3	KON	BLYSKAWICA	Z

A oto wynik działania wyzwalacza i wpisy utworzone przez niego w tabeli DZIENNIK:

```
SQL> select * from dziennik;
  ID_DZIEK  TABELA_DZIEK  DML_DZIEK  ID_KLUCZA_DZIEK  DATA_DZIEK  NAZWA_UZYTK_DZIEK
-----
          1  DZIENNIK      INSERT      3  09-SIE-01  KNOWAK
          2  DZIENNIK      UPDATE      1  09-SIE-01  KNOWAK
          3  DZIENNIK      DELETE      2  09-SIE-01  KNOWAK
```

Synonimy — pseudonimy obiektów

Synonim to alternatywna nazwa tabeli, perspektywy, sekwencji lub jednostki programu. Synonimy są używane z różnych powodów:

- ♦ ukrycie prawdziwej nazwy właściciela obiektu bazy danych;
- ♦ ukrycie prawdziwej lokalizacji obiektu bazy danych;
- ♦ nadanie obiektowi nazwy, która jest mniej skomplikowana lub łatwiejsza w użyciu.

Synonim może być *prywatny* lub *publiczny*. Z synonimu prywatnego może korzystać jedynie użytkownik, który go utworzył, natomiast synonim publiczny jest dostępny w całej bazie danych.

Polecenie create synonim — przykład

Oto przykład tworzenia synonimu prywatnego:

```
SQL> CREATE SYNONYM jackiewicz
2*   for marek.jackiewicz
SQL> /
Synonim został utworzony.
```

Teraz zostanie podjęta próba wykonania zapytania na tabeli ZWIERZAKI:

```
SQL> SELECT * FROM zwierzaki;
SELECT * FROM zwierzaki
      *
BLAD w linii 1:
ORA-00942: tabela lub perspektywa nie istnieje.
```

W następnym kroku zostanie utworzony synonim:

```
SQL> CREATE SYNONYM zwierzaki
2*   for zwierzaki
SQL> /
Synonim został utworzony.
```

Ponowna próba wyświetlenia zawartości tabeli ZWIERZAKI przyniesie następujący rezultat:

```
SQL> SELECT * FROM zwierzaki;
```

GAT_ZWIERZAKA	NAZWA_ZWIERZAKA
PIES	FAFIK
KOT	MRUCZEK
KROLIK	CIAPEK
KON	KROL

Sekwencja — szybki sposób uzyskania unikatowej liczby

Sekwencje to bardzo wydajny sposób generowania szeregu kolejnych liczb. Często w relacyjnych bazach danych pojawia się potrzeba utworzenia unikatowej liczby, która będzie pełniła rolę klucza głównego. Przykładowo, użytkownik dostaje informację na temat możliwości bardzo atrakcyjnego zakupu jakiejś spółki „.com”. Dzwoni wtedy do maklera i składa zlecenie zakupu. Z tą transakcją będzie powiązany unikatowy numer zlecenia, który zostanie utworzony właśnie przy użyciu sekwencji. Sekwencje są niezależne od jakichkolwiek tabel i są przechowywane w pamięci w oczekiwaniu na żądanie użycia.

W przeszłości, gdy nie istniały jeszcze obiekty bazy danych o nazwie sekwencje, użytkownik tworzył tabele z kolumną zawierającą liczby porządkowe. Tabele te stanowiły główne *wąskie gardło* w wielu zaawansowanych aplikacjach. Zanim użytkownik zablokował tabelę, sprawdził bieżącą wartość w kolumnie, zwiększył ją o 1 a następnie zwolnił blokadę, akcje były już sprzedane. Obiekt sekwencji jest składowany w pamięci i jest udostępniany natychmiast na żądanie użytkownika.

Polecenie create sequence — przykład

W tym przykładzie instrukcja `cache 1000` informuje bazę danych, że po wyczerpaniu wartości sekwencji umieszczonych w jej buforze musi ona utworzyć kolejnych 1 000 numerów. Należy pamiętać, że zamknięcie bazy danych powoduje usunięcie zawartości bufora sekwencji. Efektem będą luki w sekwencji liczb.

```
SQL> CREATE SEQUENCE sek_zes_proj
2     START WITHIN 1000 INCREMENT BY 1
3     MINVALUE 1
4*    CACHE 1000 NOCYCLE NOORDER
SQL> /
Sekwencja została utworzona.
```

Polecenie create role — sposób zarządzania uprawnieniami

W przeszłości przyznawanie innym użytkownikom uprawnień dostępu do aplikacji musiało być dokonywane kolejno dla poszczególnych tabel. W każdej aplikacji dostępny był określony zbiór uprawnień, zależnych od tego, kim był główny użytkownik bazy.

Efektem było ogromne zamieszanie. Potrzebna była osobna baza danych, która pozwoliłaby utrzymać porządek w samych uprawnieniach. Dostrzegając problemy wiążące się ze stosowaniem tego modelu, firma Oracle opracowała obiekt roli.

Wykorzystując to rozwiązanie administrator tworzy rolę w bazie danych, przydziela uprawnienia do tej roli a następnie przypisuje ją wymaganemu użytkownikowi. Na przykład, w warunkach szpitala można utworzyć rolę lekarza oraz rolę pielęgniarki. Lekarz może mieć możliwość tworzenia zlecenia badań laboratoryjnych, podczas gdy pielęgniarka może jedynie odczytywać wyniki tych badań.

Polecenie `create role` — przykład

Oto bardzo prosty obiekt roli, jaki został użyty w przykładzie dotyczącym szpitala:

```
SQL> CREATE ROLE pielęgniarka
  2*   NOT IDENTIFIED
SQL> /
Rola została utworzona.
```

W przypadku roli pielęgniarki parametr `not identified` oznacza, że nie jest potrzebny żaden dodatkowy poziom zabezpieczeń ponad początkowe logowanie do bazy danych. Istnieją również inne aspekty tej instrukcji, które wymagałyby dodatkowych działań ze strony użytkownika, zanim rola zostałaby uaktywniona.



W bazie danych Oracle można przyznawać użytkownikom uprawnienia `select`, `insert`, `update` i `delete` na dowolnych tabelach. Takie prawo jest znane jako *zezwoenie na poziomie obiektu* i stanowi tylko jeden z poziomów zabezpieczeń bazy danych, jakie może tworzyć administrator. Można więc wyobrazić sobie typowe środowisko szpitalne, z ponad 100 tabelami, gdzie w jednej tabeli pielęgniarka ma uprawnienia do wybierania i wstawiania, a lekarz ma uprawnienia do wybierania, wstawiania, aktualizowania i usuwania. Sytuacja ta powtarza się dla każdej z tabel. Zamiast przyznawania uprawnień indywidualnie można utworzyć rolę, przedzielić do niej określone uprawnienia, po czym przypisywać rolę wszystkim wymagającym użytkownikom.

Funkcje, procedury i pakiety

Funkcje, procedury i pakiety zostaną omówione w jednym podrozdziale. Są to obiekty bazy danych, które zawierają kod w języku PL/SQL. Za pomocą tego kodu można dostosowywać programy pod kątem wymagań określonych aplikacji.

Polecenie `create function`

Polecenie `create function` pozwala tworzyć obiekty bazy danych, które rozszerzają możliwości standardowych funkcji dostępnych w bazie danych. Na przykład system Oracle zawiera funkcję o nazwie `sqrt()`. Jest to funkcja, która zwraca pierwiastek kwadratowy z danej liczby.

Funkcja umożliwia wywoływanie programu w języku PL/SQL według jego nazwy. Ważną cechą wyróżniającą funkcje jest to, że wynikiem ich działania musi być zawsze jakaś wartość.

Polecenie create function — przykład

Oto tabela NOWY_NABOR.

```
SQL> select * from nowy_nabor;
NAZWISKO          WOJ   DATA_ZATR          WYNAGRODZ
-----
Jacek             PM    01-STY-01           20000
Nowak            LD    10-CZE-01           30000
Kowalski         CT    15-SIE-01           40000
Ania             MA    12-GRU-01           50000
Marek            ZP    12-LIS-01           60000
Wiesiek          SL    11-MAJ-01           90000
Tomek            PM    15-SIE-01           55000
Pawel            LD    02-LUT-01           30000
zbyszek          LD    05-CZE-01           20000
magda            PM    26-CZE-01           75000
basia            PM    08-MAJ-01           80000
11 wierszy zostało wybranych.
```

Teraz zostanie utworzony program, który będzie sumował informacje o wynagrodzeniach dla danego województwa i zwracał łączną wielkość wynagrodzenia.

```
SQL> CREATE FUNCTION dodaj_wynagr_dla_woj(we_kod_woj IN VARCHAR2)
2   RETURN NUMBER
3   IS wynagr_woj NUMBER(11,2);
4   BEGIN
5       SELECT sum(wynagrodz)
6         INTO wynagr_woj
7         FROM nowy_nabor
8         WHERE woj = we_kod_woj;
9       RETURN(wynagr_woj);
10  END;
11  /
```

Funkcja została utworzona.

Teraz za pomocą tej funkcji można wyszukać łączne wielkości wynagrodzenia dla województwa określonego parametrem kod_woj = 'PM'. Warto zwrócić uwagę na użycie nazwy funkcji — dodaj_wynagr_dla_woj oraz sposób przekazywania do niej skrótu PM.

```
SQL> var x number;
SQL> exec :x := dodaj_wynagr_dla_woj('PM');
Procedura PL/SQL została wykonana pomyślnie.
SQL> print x;
      X
-----
230000
```


Polecenie create procedure

Procedura to zbiór programów w języku PL/SQL, które są wywoływane według nazw. W odróżnieniu od funkcji wynikiem działania procedury nie musi być zwrócenie jakiegś wartości. Procedura może ale nie musi zawierać argumenty wejściowe i wyjściowe.

Polecenie create procedure — przykład

W poniższym przykładzie procedura zawiera parametr wejściowy i wyjściowy.

```
SQL> CREATE OR REPLACE PROCEDURE USTAL_LICZBE_Prac_WG_WOJ (
  2     we_kod_woj      IN varchar2,
  3     wy_liczba_prac OUT integer
  4 )
  5 AS
  6     wyjscie varchar2(6);
  7
  8 BEGIN
  9
 10     select count(e.id_prac)
 11     into wy_liczba_prac
 12     from pracownik e,
 13     woj s
 14     where e.kod_woj = s.kod_woj
 15           and s.kod_woj = we_kod_woj;
 16
 17
 18 EXCEPTION
 19     when others then
 20         dbms_output.put_line ('BLAD USTAL_LICZBE_Prac_WG_WOJ'||SQLCODE||SQLERRM);
 21* END;
SQL> /
```

Procedura została utworzona.

Polecenie create package

Pakiet to zbiór zawierający zarówno procedury, jak i funkcje. Są one grupowane głównie ze względu na podobieństwo realizowanych zadań. Wszystkie wewnętrzne procedury i funkcje są zapisane w słowniku danych jako pojedynczy pakiet składowany.

Polecenie create package — przykład

Pokazany poniżej kod pakietu zawiera kilka funkcji i jedną procedurę.

```
SQL> CREATE OR REPLACE PACKAGE   pkt_prac_woj AS
  2     PROCEDURE ustal_liczbe_prac_wg_woj (we_kod_woj VARCHAR2, wy_liczba_prac INTEGER);
  3     FUNCTION dodaj_wynagr_dla_woj (we_kod_woj VARCHAR2)
  4         RETURN NUMBER;
  5     FUNCTION ustal_imie_i_nazwisko (we_imie VARCHAR2, we_nazwisko VARCHAR2)
  6     RETURN VARCHAR2;
  7* END pkt_prac_woj;
SQL> /
```

Pakiet został utworzony.

Inne obiekty baz danych

W poniższych sekcjach omówiono pewne dodatkowe obiekty baz danych, z którymi może zetknąć się użytkownik. Niektóre z nich podkreślają możliwości funkcjonalne bazy danych Oracle⁹ⁱ, inne zaś Czytelnik zapewne napotka podczas pracy z bazą. Zaprezentowane przykłady pomogą zrozumieć zasady posługiwania się tymi obiektami.

Polecenie create operator

Polecenie `create operator` służy do tworzenia nowego operatora w bazie danych. W rozdziale 8. dokładniej omówiono operatory `union`, `intersect` i `minus`. Komenda `create operator` umożliwia tworzenie nowych operatorów i ich powiązań. Operator może z kolei zawierać odwołania do funkcji, pakietów i innych elementów.

Polecenie create operator — przykład

Warto zwrócić uwagę, w jaki sposób operator odwołuje się do procedury `ustal_imie_i_nazwisko`.

```
SQL> CREATE OPERATOR podaj_imie_i_nazwisko
 2   BINDING
 3   (varchar2, varchar2)      RETURN varchar2
 4   USING ustal_imie_i_nazwisko;
```

Operator utworzony.

Odwołanie do funkcji `ustal_imie_i_nazwisko` w nowo utworzonym operatorze wygląda następująco.

```
SQL> CREATE OR REPLACE FUNCTION ustal_imie_i_nazwisko
 2   (we_imie IN VARCHAR2,
 3   we_nazwisko IN VARCHAR2)
 4   RETURN VARCHAR2
 5   IS wy_imie_i_nazwisko VARCHAR2(50);
 6
 7   BEGIN
 8
 9   SELECT initcap(ltrim(rtrim(we_imie)))||' '||
10   initcap(ltrim(rtrim(we_nazwisko)))
11   INTO wy_imie_i_nazwisko
12   FROM calosc;
13
14   RETURN(wy_imie_i_nazwisko);
15*  END;
SQL> /
```

Funkcja została utworzona.

Polecenie create directory

Czasami zewnętrzne obiekty bazy danych warto przechowywać w katalogu. Najczęściej rozwiązanie takie stosowane jest w przypadku dużych plików, zwanych często *plikami*

binarnymi. Są to na przykład pliki graficzne. Komenda `create directory` pozwala utworzyć obiekt katalogu, który stanowi alias katalogu zlokalizowanego w systemie plików na serwerze. Następnie w kodzie tworzonego programu zamiast wpisywać pełną ścieżkę do katalogu można umieścić odwołanie do tego obiektu.

Polecenie `create directory` — przykład

Jest to typowe zastosowanie instrukcji `create directory`. Katalog ten będzie zawierał wszystkie pliki binarne użytkownika.

```
SQL> create or replace DIRECTORY moje_pliki_binarne as '/d0/oraclehome/picture';
```

Katalog został utworzony.

Polecenie `create library`

Instrukcja `create library` służy do tworzenia obiektu bazy danych związanego ze współużytkowaną biblioteką systemu operacyjnego. Dzięki temu ze składni języka SQL i PL/SQL można wywoływać funkcje języków (takich jak funkcje języków C czy COBOL) i procedur trzeciej generacji, które następnie mogą kierować wywołania do bibliotek. Zanim wprowadzono ten obiekt, użytkownik musiał samodzielnie wpisywać w kodzie każde odwołanie do biblioteki.

```
SQL> CREATE LIBRARY moja_biblioteka as '/d0/oraclehome/clib';
2 /
```

Biblioteka została utworzona.

Powiązania bazodanowe

Powiązanie bazodanowe (ang. *database link*) pozwala użytkownikowi korzystać z danych umieszczonych w zdalnej bazie danych bez konieczności znajomości dokładnej lokalizacji tych danych. Podczas tworzenia powiązania bazodanowego użytkownik dostarcza informacje dotyczące logowania do odpowiedniej zdalnej bazy danych. Każde użycie powiązania powoduje zainicjowanie sesji sieciowej, której zadaniem jest nawiązanie połączenia ze zdalną tabelą lub perspektywą.

Niezależnie od nazwy użytkownika zalogowanego w lokalnej bazie danych, do logowania w zdalnej bazie danych powiązanie bazodanowe użyje informacji logowania podanych w momencie tworzenia powiązania. Powiązania mogą być tworzone do użytku prywatnego i publicznego.

Polecenie `create database link` — przykład

W tym przykładzie logowanie w zdalnej bazie danych zostanie przeprowadzone dla użytkownika `guru`. Ponadto zostanie wykorzystany serwis internetowy *emile.corp.Ntirety.com*.

```
SQL> CREATE DATABASE LINK powiazanie_bazy_danych_guru
2 CONNECT TO guru IDENTIFIED BY guru
3* USING 'emile.corp.ntirety.com'
SQL> /
```

Powiązanie bazodanowe zostało utworzone.

Teraz nowo utworzone powiązanie zostanie użyte do znalezienia pewnych danych.

```
SQL> SELECT *
  2* FROM kolor@powiazanie_bazy_danych_guru
SQL> /
```

ID_KOLORU	OPIS_KOLORU
CE	Czerwony
ZI	Zielony
NB	Niebieski
CA	Czarny

Powiązanie bazodanowe to swego rodzaju łącze umożliwiające przesyłanie danych w obie strony. Można z niego korzystać na wiele sposobów. W przykładzie poniżej zostanie ono użyte w celu wstawienia danych do zdalnej bazy danych:

```
SQL> INSERT into kolor@powiazanie_bazy_danych_guru(id_koloru, opis_koloru)
  2* values ('ZO', 'Zolty')
SQL> /
```

1 wiersz został utworzony.

Teraz zostaną zaktualizowane dane w zdalnej bazie danych:

```
SQL> UPDATE kolor@powiazanie_bazy_danych_guru
  2 set id_koloru = 'ZY'
  3* where id_koloru = 'ZO'
SQL> /
```

1 wiersz został zmodyfikowany.

Można również usunąć dane ze zdalnej bazy danych:

```
SQL> DELETE FROM kolor@powiazanie_bazy_danych_guru
  2* where id_koloru = 'ZY'
SQL> /
```

1 wiersz został usunięty.

Za pomocą powiązania bazodanowego można również uzyskiwać dostęp do danych innego użytkownika, umieszczonych w zdalnej bazie danych. Jedynym ograniczeniem są uprawnienia konta systemu Oracle⁹ⁱ używanego do zalogowania się w zdalnej bazie danych.

```
SQL> SELECT *
  2* FROM mi cha l .nowak@powiazanie_bazy_danych_guru
SQL> /
```

GRUPA	SPRZEDAZ_MIN	SPRZEDAZ_MAKS
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

Polecenie create cluster

Klasy stanowią alternatywną metodę fizycznego przechowywania danych. Tabele umieszczone w obiekcie klastra mogą być przechowywane z określonymi wstępnymi złączeniami. Rozwiązanie to jest szczególnie przydatne wtedy, gdy kilka tabel jest używanych zawsze razem. Dobrym przykładem jest na przykład połączenie odczytów liczników zużycia wody z informacjami na temat mieszkańców danego mieszkania czy domu. Operator sieci wodociągowej nigdy nie używa danych o zużyciu wody w oderwaniu od informacji o osobach, które przebywają stale pod danym adresem.

Należy pamiętać, że wartość obu tabel jest przechowywana razem. Oznacza to, że gdy użytkownik będzie chciał zapoznać się z informacjami dotyczącymi osób stale zamieszkujących daną lokalizację, baza danych będzie zawsze odczytywała również dane dotyczące zużycia wody.

Polecenie create cluster — przykład

Najpierw należy utworzyć klucz klastra:

```
SQL> CREATE CLUSTER kls_mieszkaniec
  2   (numer_mieszkanca number(3))
  3   size 512
  4*  storage (initial 100k next 50k)
SQL> /
```

Klaster został utworzony.

```
SQL> CREATE TABLE mieszkancy
  2   (nr_mieszk NUMBER(3),
  3   imie_mieszk VARCHAR2(20),
  4   nazw_mieszk VARCHAR2(30),
  5   data_ur_mieszk date,
  6   miasto_zam_mieszk VARCHAR2(20),
  7   kod_woj VARCHAR2(2))
  8*  CLUSTER kls_mieszkaniec (nr_mieszk)
SQL> /
```

Tabela została utworzona.

Należy zwrócić uwagę na sposób odwołania do obiektu klastra podczas tworzenia tabeli.

```
SQL> CREATE TABLE pracownicy_mieszk
  2   (id_prac integer,
  3   nr_mieszk NUMBER(3),
  4   imie_prac VARCHAR2(20),
  5   nazw_prac VARCHAR2(30),
  6   tyt_prac VARCHAR2(20))
  7*  CLUSTER kls_mieszkaniec (nr_mieszk)
SQL> /
```

Tabela została utworzona.

Jak dotąd, omówiono obiekty baz danych, z którymi zapewne użytkownik będzie miał styczność. Zostały również zaprezentowane niektóre bardziej zaawansowane obiekty, takie jak indeksy bitmapowe, powiązania bazodanowe czy operatory. W następnym rozdziale nastąpi omówienie programowania SQL*Plus oraz sposobów jego wykorzystywania.

Pytania do rozdziału 4.

Odpowiedzi na pytania można znaleźć w dodatku A.

1. _____ to obiekt bazy danych zawierający dane użytkownika.
 - A. Perspektywa
 - B. Operator
 - C. Rola
 - D. Tabela
2. Do momentu wykonania _____ nie zawiera żadnych danych, a jedynie składowaną instrukcję SQL.
 - A. perspektywa
 - B. tabela
 - C. indeks
 - D. perspektywa materializowana
3. Z _____ zawsze musi być zwracana wartość.
 - A. pakietu
 - B. procedury
 - C. funkcji
 - D. perspektywy
4. _____ umożliwia korzystanie z danych przechowywanych w zdalnej bazie danych.
 - A. Perspektywa
 - B. Powiązanie bazy danych
 - C. Tabela
 - D. Funkcja
5. _____ to prosty sposób zarządzania zbiorami uprawnień.
 - A. Rola
 - B. Perspektywa
 - C. Powiązanie bazy danych
 - D. Pakiet