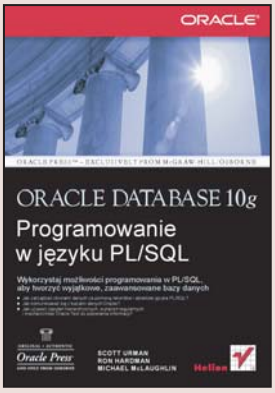
Oracle Database 10g. Programowanie w języku PL/SQL



Autor: Scott Urman, Ron Hardman, Michael McLaughlin
Tłumaczenie: Tomasz Walczak
ISBN: 978-83-246-1207-9
Tytuł oryginału: Oracle Database 10g PL/SQL Programming
Format: B5, stron: 752

Wykorzystaj możliwości programowania w PL/SQL, aby tworzyć wyjątkowe, zaawansowane bazy danych

- Jak zarządzać zbiorami danych za pomocą rekordów i obiektów języka PL/SQL?
- Jak komunikować się z bazami danych Oracle?
- Jak używać zapytań hierarchicznych, wyrażeń regularnych i mechanizmów Oracle Text do pobierania informacji?

Tworzenie baz danych i zarządzanie nimi wymaga nie tylko umiejętności technicznych i dyscypliny, ale również pomysłowości, wyobraźni i – co najważniejsze – odpowiednich narzędzi. Zatem właściwym wyborem będzie Oracle – elastyczny system zarządzania relacyjnymi bazami danych o niezwykle dużych możliwościach. Pozwala na tworzenie i administrowanie zaawansowanymi aplikacjami przy użyciu języka PL/SQL, który służy do pobierania, wstawiania, aktualizowania i usuwania danych, a także tworzenia i przechowywania obiektów, użytkowników oraz kontrolowania praw dostępu do danych.

Książka „Oracle 10g. Programowanie w języku PL/SQL" zawiera materiał zarówno dla początkujących, jak i zaawansowanych użytkowników. Przedstawiono w niej nie tylko podstawowe elementy języka (strukturę bloku, bloki anonimowe i nazwane, komunikaty o błędach i inne), ale także ułożony według wersji spis głównych rozszerzeń. Korzystając z tego podręcznika, nauczysz się, jak system przetwarza transakcje i zobaczysz, jak działa baza danych na zapleczu. Dowiesz się, jak pisać wyzwalacze i używać ich do zarządzania lokalnymi oraz zdalnymi egzemplarzami bazy. Dodatkowo poznasz sztuczki programistyczne i praktyczne przykłady technik stosowanych w pracy z Oracle.

- Używanie języka SQL w PL/SQL
- Konfigurowanie transakcji
- Rodzaje kursorów i ich działanie
- Korzystanie z rekordów i kolekcji
- Używanie narzędzi SQL*Plus i Developer
- Tworzenie procedur, funkcji i pakietów
- Obsługa błędów
- Wyzwalacze w bazach danych
- Komunikacja między sesjami
- Procedury zewnętrzne
- Obiekty w bazach danych
- Szeregowanie zadań

W jednej książce otrzymujesz więc wszystko, czego potrzeba, aby tworzyć zaawansowane bazy danych i profesjonalnie nimi zarządzać!

Wydawnictwo Helion
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

Spis treści

Rozdział 1.

Wprowadzenie do języka PL/SQL

Czasem można natrafić na dobrze napisany kod, który składa się na naprawdę kiepską aplikację. Wystarczy przyjrzeć się doskonale napisanym wirusom krążącym w sieci czy produktom niektórych firm, które zniknęły z rynku, wypuszczających programy pełne gadżetów, ale zupełnie bezużyteczne. Programowanie to coś więcej niż stosowanie składni. Jest to zajęcie, w którym wiedzę należy połączyć z pomysłowością, umiejętnością komunikacji, odpowiednim podejściem i dyscypliną, aby rozwijać się na ścieżce kariery i tworzyć wysokiej klasy aplikacje.

W tej książce skoncentrujemy się przede wszystkim na składni i zasadach. Odpowiemy na szereg pytań typu „dlaczego warto tego używać?", które pojawią się przy wprowadzaniu nowych funkcji. Przedstawione analizy wykraczają poza opis faktu, że Oracle **może** służyć do wykonania danych operacji. Pokazujemy także, **jak** i **dlaczego** ich używać.

W pierwszym rozdziale przedstawiamy podstawy niezbędne do zrozumienia pozostałej części tej książki. Opisujemy tu następujące zagadnienia:

- język SQL i jego związki z relacyjnymi bazami danych;
- w jaki sposób język PL/SQL korzysta z SQL i zwiększa jego możliwości;
- zagadnienia programistyczne, obejmujące porównanie języków proceduralnych do programowania obiektowego;
- historię i funkcje języka PL/SQL;
- zalety (i wady) PL/SQL;
- jak podejść do lektury pozostałych części tej książki i najlepiej wykorzystać ten starannie opracowany tekst.

Wprowadzenie do języków programowania

Java, C++, PL/SQL i Visual Basic to niektóre z najbardziej popularnych współczesnych języków programowania. Każdy z nich różni się od pozostałych i ma własne niepowtarzalne cechy. Choć są to odrębne języki, mają pewne wspólne właściwości. Języki programowania można pogrupować właśnie według tych wspólnych elementów. Wcześniej wymienione języki należą do dwóch odrębnych kategorii — języków proceduralnych i obiektowych.

Języki proceduralne, takie jak PL/SQL i Visual Basic, wykonują operacje liniowo. Programy w tych językach **rozpoczynają** działanie na początku, a **kończą** — na końcu. Jest to uproszczona definicja, jednak to sposób działania stanowi główną różnicę między językami proceduralnymi a obiektowymi. Każda instrukcja musi czekać na zakończenie poprzedniej, zanim rozpocznie działanie. Dla wielu początkujących programistów zapoznanie się z jednym z języków proceduralnych to najlepszy sposób na rozpoczęcie nauki. Program musi wykonać serię operacji i właśnie tak działa kod w takich językach — krok po kroku.

Języki obiektowe, takie jak Java czy C++, są bardziej abstrakcyjne. W programach napisanych w takich językach działają struktury nazywane **obiektami**. Na przykład, zamiast pisać kod zbierający informacje o książce bezpośrednio ze struktur danych, można utworzyć **obiekt** o nazwie `KSIĄŻKA`. Każdy **obiekt** ma **atrybuty** — liczbę stron, cenę, tytuł itd. Atrybuty opisują obiekt. **Metody** służą do wykonywania zadań. Manipulują danymi, pobierając lub modyfikując je. Jeśli programista chce na przykład zmienić cenę książki, powinien wywołać metodę odpowiedzialną za to zadanie. Jest to inne podejście niż w przypadku języków proceduralnych, gdzie należy wykonać serię operacji, aby osiągnąć ten sam efekt.

Do rzadkich przypadków języków należących do obu kategorii należy najnowsza wersja PL/SQL, którą można traktować zarówno jako język proceduralny, jak i obiektowy. Obiekty wprowadzono w Oracle 8, choć w początkowych wersjach brakowało obsługi zaawansowanych funkcji, takich jak dziedziczenie, ewolucja typów czy dynamiczne wiązanie metod. W Oracle 9*i*R1 rozpoczęto wprowadzanie zmian w kierunku umożliwienia w pełni obiektowego programowania za pomocą języka PL/SQL. Oracle 10*g* obsługuje już większość podstawowych możliwości programowania obiektowego.


Uwaga

Wspomniane wcześniej funkcje programowania obiektowego są szczegółowo opisane w rozdziałach 14. i 15.

Wskazówki dla początkujących programistów

Podobnie jak wielu innych programistów rozpoczynałem przygodę z programowaniem od języka Basic. Miał on prostą składnię, a dotyczące go programistyczne „prawidła" można było zastosować także do wielu innych języków. Uważam, że to samo można powiedzieć o języku PL/SQL.

Moją ulubioną cechą języka PL/SQL nie jest ścisła integracja z bazą danych (choć jest ona obecna), zaawansowane funkcje i możliwości programistyczne (w trakcie lektury tej książki przekonasz się, jak wiele można zrobić za pomocą tego języka), ani żaden inny rodzaj funkcjonalności. Najbardziej cenię ustrukturyzowane podejście do programowania. Każdej instrukcji `BEGIN` odpowiada `END`, a każdej instrukcji `IF` — `END IF`.

Jako instruktor uczący języka PL/SQL wiele osób rozpoczynających przygodę nie tylko z PL/SQL, ale z programowaniem w ogóle, wiem, że każdy może się go nauczyć. Jest ustrukturyzowany, liniowy i nie pozwala na zbyt duże odstępstwa od składni. To dobre cechy. Dzięki temu można nauczyć się stosowania struktury i zasad. Jeśli programista nie będzie ich przestrzegał, otrzyma natychmiastową informację zwrotną przy próbie uruchomienia kodu.

```
Warning: Procedure created with compilation errors.
```

> Oczywiście struktura nie gwarantuje dobrego kodu, a jedynie ułatwia naukę języka. Zwracaj uwagę na zachowanie eleganckiego wyglądu kodu, stosuj odpowiednie konwencje nazewnicze, dokumentuj kod i przede wszystkim ćwicz. Nie korzystaj ze sztuczek, które powodują, że kod staje się niewydajny i trudny w pielęgnacji. Podobnie jak w każdym innym języku możliwe jest napisanie okropnego kodu, który skompiluje się poprawnie.

Warto pamiętać, że najlepsi programiści nie muszą być specjalnie uzdolnieni technicznie. Są to osoby z dużymi zdolnościami komunikacyjnymi, które potrafią wejść w rolę użytkowników i klientów. Jest to szczególnie istotne na etapie projektowania. Programista spotyka się z menedżerami projektu, innymi programistami, administratorami baz danych, użytkownikami, inżynierami z działu oceny jakości i zarządem. Osoby z każdej z tych grup mają inne cele w cyklu rozwoju systemów i stawiają programistom odmienne wymagania. Sukces lub porażka projektu zależą od nastawienia oraz umiejętności komunikacji, a cechy te w ostatecznym rozrachunku określają sukces programisty.

PL/Co?

Czym więc jest PL/SQL? Jest to proceduralne (a czasem obiektowe) rozszerzenie programistyczne języka SQL udostępniane przez Oracle i przeznaczone wyłącznie do obsługi produktów tej firmy. Jeśli programista zna język Ada, odkryje, że PL/SQL jest niezwykle do niego podobny. Podobieństwo obu tych języków wynika z tego, że PL/SQL wywodzi się z Ady, a twórcy PL/SQL zapożyczyli wiele pomysłów właśnie z tego języka.

PL w nazwie PL/SQL to skrót od angielskiego **procedural language**, czyli **język proceduralny**. PL/SQL to język chroniony prawami autorskimi niedostępny poza bazą danych Oracle. Jest to język trzeciej generacji (3GL), który udostępnia konstrukty programistyczne podobne do innych języków klasy 3GL, włączając w to deklaracje zmiennych, pętle, obsługę błędów itd. Początkowo PL/SQL był językiem czysto proceduralnym. Jednak, jak opisuje to poprzedni punkt, obecnie PL/SQL można traktować także jako język obiektowy. Czy nie należy w związku z tym zmienić jego nazwy na PL/OO/SQL?

Język SQL

SQL w nazwie PL/SQL to skrót od angielskiego ***structured query language***, czyli **ustrukturyzowany język zapytań**. Język SQL służy do pobierania (instrukcja `SELECT`), wstawiania (`INSERT`), aktualizowania (`UPDATE`) i usuwania (`DELETE`) danych. Można go używać do tworzenia i przechowywania obiektów oraz użytkowników, a także do kontrolowania praw dostępu do danych.

SQL (wymawiaj *sikłel* lub *es-kju-el*) to punkt wejścia lub okno do bazy danych. Jest to język czwartej generacji (4GL), który ma być łatwy w użyciu oraz nauce. Podstawowa składnia języka SQL nie została utworzona przez korporację Oracle. Ten język powstał na podstawie prac doktora E. F. Codda oraz firmy IBM we wczesnych latach 70. Instytut ANSI (ang. *American National Standards Institute*) uznaje język SQL i publikuje jego standardy.

Oracle obsługuje język SQL zgodny ze standardami ANSI, jednak udostępnia także własną wersję — SQL*Plus. Dzięki SQL*Plus Oracle obsługuje dodatkowe polecenia i funkcje, których nie obejmują standardy. SQL*Plus to narzędzie dostępne w kilku postaciach:

- **Wiersz poleceń** uruchamiany z wiersza poleceń systemów Unix czy DOS.
- **Graficzny interfejs użytkownika** w programach SQL*Plus Client, SQL Worksheet, Enterprise Manager.
- **Strona internetowa** w aplikacjach iSQL*Plus, Enterprise Manager w wersji 10*g*.

Po zainstalowaniu samego klienta można skonfigurować połączenie sieciowe ze zdalnymi bazami danych. W Oracle 10*g* wykonanie tej operacji jest jeszcze łatwiejsze niż wcześniej, a służą do tego bazujące na przeglądarce programy Enterprise Manager i iSQL*Plus, które są konfigurowane w trakcie instalacji.

Przegląd relacyjnych baz danych

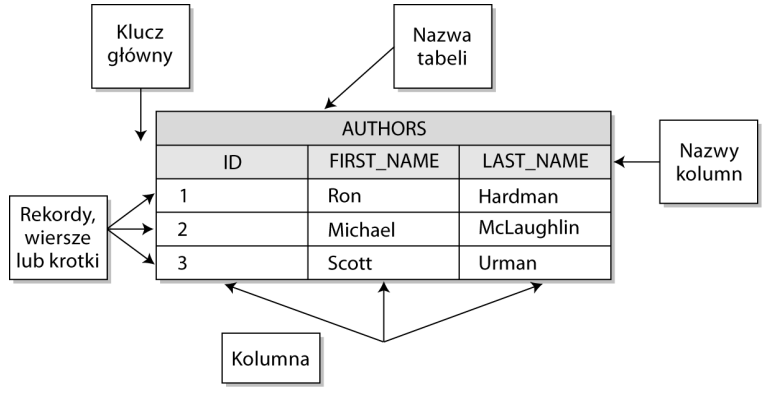
SQL to okno do bazy danych, jednak czym jest taka baza? **Baza danych** to, mówiąc ogólnie, cokolwiek, co przechowuje dane. Elektroniczną bazą danych może być coś tak prostego jak arkusz kalkulacyjny czy dokument utworzony za pomocą edytora tekstu.

Jak łatwo sobie wyobrazić, przechowywanie dużych ilości danych w arkuszu kalkulacyjnym lub dokumencie edytora tekstu może szybko stać się bardzo trudne. Takie jednowymiarowe bazy danych nie umożliwiają szybkiego odfiltrowywania niepotrzebnych danych, zapewniania spójności wpisywanych informacji czy obsługi pobierania danych.

Oracle to **system zarządzania relacyjnymi bazami danych** (ang. *relational database management system* — RDBMS). **Tabele** w takim systemie składają się z kolumn, które określają typ danych, jakie można w nich przechowywać (znaki, liczby itd.). Tabela musi mieć przynajmniej jedną **kolumnę**. Dane umieszczane w tabeli są zapisywane w **wierszach**. Taka struktura obowiązuje w produktach wszystkich producentów relacyjnych baz danych (rysunek 1.1).

W Oracle tabele są własnością użytkownika lub schematu. Schemat to kolekcja obiektów, takich jak tabele, które należą do określonego użytkownika bazy danych. W jednej bazie danych mogą znajdować się dwie tabele o takiej samej nazwie, o ile mają różnych właścicieli. Inni producenci

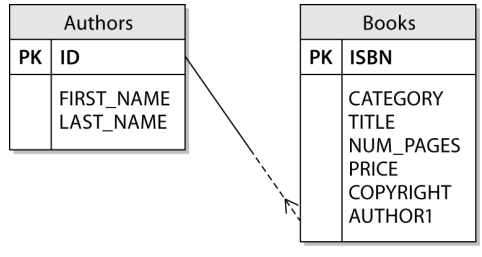
Rysunek 1.1.
Struktura tabeli



systemów bazodanowych nie zawsze stosują takie podejście. Na przykład w SQL Server obowiązuje inna terminologia. Bazy danych w SQL Server bardziej przypominają schematy Oracle, a **serwer** SQL Server funkcjonuje podobnie jak baza danych w Oracle. Efekt jest jednak taki sam. Obiekty, takie jak tabele, zawsze mają swego właściciela.

Możliwe jest przechowywanie wszystkich danych w pojedynczej tabeli, podobnie jak w arkuszu kalkulacyjnym, jednak nie pozwala to wykorzystać możliwości, jakie dają relacyjne funkcje bazy Oracle. Na przykład tabela zawierająca dane o książkach wydawnictwa Helion jest niepełna bez informacji o autorze. Możliwe, że dany autor napisał wiele pozycji. W modelu bazującym na zwykłym pliku tekstowym z danymi lub na pojedynczej tabeli autor będzie wymieniony wielokrotnie. Można uniknąć tej nadmiarowości, rozbijając dane na dwie tabele z kolumną, która wiąże powiązane dane. Rysunek 1.2 przedstawia, jak można rozbić dane na dwie tabele.

Rysunek 1.2.
Diagram ERD z tabelami BOOKS i AUTHORS



Na rysunku 1.2 widoczne są dwie tabele — `AUTHORS` i `BOOKS`. Informacje o autorze obejmują tylko jedno wystąpienie jego imienia i nazwiska. Każdy wiersz z danymi ma identyfikator `ID`, który jest niepowtarzalny i nie może przyjmować wartości `NULL` (wartość `NULL` oznacza puste pole, dlatego wartość `NOT NULL` oznacza pole niepuste).

Ponieważ dostępna jest tabela `AUTHORS`, nie trzeba powtarzać informacji o danym autorze przy każdej jego książce. Wystarczy dodać pojedynczą kolumnę `AUTHOR1` do tabeli `BOOKS` i wstawić odpowiednią wartość `ID` z tabeli `AUTHORS` dla każdego tytułu w tabeli `BOOKS`. Używając tabeli `BOOKS.AUTHOR1` jako klucza zewnętrznego (`FOREIGN KEY`), za pomocą języka SQL można powiązać ze sobą obie tabele. Poniżej znajduje się przykładowy kod.


Uwaga

Warto użyć skryptu *CreateUser.sql* z katalogu z kodem źródłowym przykładów z tego rozdziału. Ten skrypt tworzy użytkownika o nazwie `plsql` i nadaje mu potrzebne uprawnienia.

```
-- Kod dostępny jako część pliku PlsqlBlock.sql
CREATE TABLE authors (
  id         NUMBER PRIMARY KEY,
  first_name VARCHAR2(50),
  last_name  VARCHAR2(50)
);

CREATE TABLE books (
  isbn       CHAR(10) PRIMARY KEY,
  category   VARCHAR2(20),
  title      VARCHAR2(100),
  num_pages  NUMBER,
  price      NUMBER,
  copyright  NUMBER(4),
  author1    NUMBER CONSTRAINT books_author1
             REFERENCES authors(id)
);
```

Po wstawieniu kilku rekordów do tabeli można wywołać instrukcję SELECT złączającą obie tabele zgodnie z ich powiązaniami.

```
SELECT b.title, a.first_name, a.last_name
FROM authors a, books b
WHERE b.author1 = a.id;
```

Ta instrukcja złącza dwie tabele i pobiera dane w takim formacie, w jakim byłyby widoczne w zwykłym pliku tekstowym. Różnica polega na mniejszej nadmiarowości, mniejszej liczbie okazji do popełnienia błędu i większej elastyczności. Aby dodać informacje o wydawcy, wystarczy utworzyć tabelę o nazwie PUBLISHER o identyfikatorze ID, a następnie dodać do tabeli BOOKS kolumnę działającą jako klucz zewnętrzny wskazującą na kolumnę PUBLISHER.ID.


Uwaga

Rozszerzony opis języka SQL znajduje się w dokumentacji elektronicznej na stronie *http://otn.oracle.com*.

PL/SQL a SQL

SQL zapewnia kompletny dostęp do danych. Oznacza to, że można pobrać dowolne dane — po pewnym czasie i w wielu przypadkach w niedoskonały sposób. Nie ma gwarancji co do wydajności i dostępnych jest niewiele funkcji programistycznych znanych z większości języków. SQL między innymi nie umożliwia:

- przechodzenia w pętli po rekordach i manipulowania każdym z nich po kolei;
- zabezpieczenia kodu za pomocą szyfrowania i przechowywania kodu całkowicie po stronie serwera, a nie po stronie klienta;
- obsługi wyjątków;
- korzystania ze zmiennych, parametrów, kolekcji, rekordów, tablic, obiektów, kursorów, wyjątków, kolumn BFILE itd.

Choć SQL daje dużo możliwości, a SQL*Plus (interfejs języka SQL autorstwa firmy Oracle) obejmuje polecenia i wbudowane funkcje spoza standardów ANSI, SQL pozostaje bardziej sposobem dostępu do bazy danych niż językiem programowania. PL/SQL rozpoczyna się w miejscu, w którym kończy się SQL, i oferuje wspomniane wyżej funkcje, a także wiele innych możliwości.

Uwaga

Nie martw się, jeśli nie znasz wszystkich wymienionych wcześniej funkcji programistycznych — w końcu dlatego czytasz tę książkę. Są one szczegółowo opisane w dalszych rozdziałach.

Prawie wszystkie operacje języka SQL można wykonać za pomocą PL/SQL. W Oracle 9*i*R1 parser języka PL/SQL jest taki sam jak parser SQL, co gwarantuje, że polecenia są traktowane tak samo niezależnie od tego, gdzie zostały wywołane. W starszych wersjach Oracle zdarzały się sytuacje, w których instrukcje języka SQL były interpretowane zupełnie inaczej. Teraz się to zmieniło.

W poniższym przykładzie utworzone wcześniej tabele BOOKS i AUTHORS są wykorzystane w kodzie w języku PL/SQL.

```
-- Kod dostępny jako część pliku PlsqlBlock.sql
SET SERVEROUTPUT ON
DECLARE
   v_title books.title%TYPE;
   v_first_name authors.first_name%TYPE;
   v_last_name authors.last_name%TYPE;

   CURSOR book_cur IS
      SELECT b.title, a.first_name, a.last_name
      FROM authors a, books b
      WHERE a.id = b.author1;
BEGIN
   DBMS_OUTPUT.ENABLE(1000000);
   OPEN book_cur;
   LOOP
      FETCH book_cur INTO v_title, v_first_name, v_last_name;
      EXIT WHEN book_cur%NOTFOUND;

      IF v_last_name = 'Hardman'
      THEN
         DBMS_OUTPUT.PUT_LINE('Ron Hardman jest współautorem '||v_title);
      ELSE
         DBMS_OUTPUT.PUT_LINE('Ron Hardman nie napisał '||v_title);
      END IF;
   END LOOP;

   CLOSE book_cur;

EXCEPTION
   WHEN OTHERS
      THEN
      DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/
```

Ten przykład obejmuje instrukcję SELECT użytą już wcześniej, ale tym razem kod przechodzi w pętli po wszystkich wynikach zapytania, sprawdzając, czy nazwisko autora to 'Hardman' i odpowiednio formatując dane wyjściowe. Na tym polega wartość języka SQL 4GL w połączeniu z funkcjami proceduralnego języka 3GL.


Uwaga

Zwróć uwagę na strukturę ostatniego bloku. Każdej instrukcji BEGIN odpowiada instrukcja END.

PL/SQL a Java

Bazy danych Oracle 8*i* obejmują obsługę języka Java oraz procedur składowanych tego języka. Dlaczego więc nie używać Javy?

PL/SQL jest i zawsze był ściśle zintegrowany z bazą danych Oracle. Oracle nieustannie usprawnia wydajność tego języka, dodając zwiększające tę integrację funkcje, takie jak wbudowana kompilacja kodu PL/SQL. Oznacza to, że w trakcie kompilacji kod jest przekształcany na język C (w tym języku jest napisana baza Oracle). W trakcie wykonywania kodu system nie musi przekształcać kodu między językami PL/SQL i C. Powoduje to znaczny wzrost wydajności — aż do 30% w porównaniu z trybem domyślnym obejmującym przekształcanie kodu.

Kolejną zaletą PL/SQL jest jego zwięzłość. Aby przekształcić instrukcję SQL na blok języka PL/SQL (bloki są opisane w rozdziale 3.), wystarczy dodać instrukcję BEGIN na początku i instrukcję END na końcu. Nie można tego samego powiedzieć o języku Java. Poniżej przedstawiony jest najprostszy blok w języku PL/SQL:

```
BEGIN
   NULL;
END;
/
```

Wypróbuj go — działa. Nie wykonuje żadnych operacji, ale uruchomi się.

PL/SQL ma jeszcze kilka innych charakterystycznych cech:

- Obecnie PL/SQL korzysta z tego samego parsera co SQL, dlatego zapewniona jest zgodność między interfejsami obu tych języków.
- Instrukcje PL/SQL można wykonywać z poziomu SQL.
- Do PL/SQL wciąż dodawane są funkcje obiektowe, co powoduje, że wiele przyczyn przechodzenia na język Java staje się nieaktualnych.

Nie jest prawdą, że **zawsze** należy używać języka PL/SQL, a **nigdy** Javy. Java udostępnia wiele funkcji niedostępnych w PL/SQL, jednak nie należy jej traktować jako języka zastępującego PL/SQL, a jedynie jako alternatywę.


Uwaga

Od czasu dodania do Javy obsługi baz danych wielokrotnie słyszałem opinie, że oznacza to koniec języka PL/SQL, który zostanie wyparty właśnie przez Javę — nie jest to prawda.

Historia i możliwości języka PL/SQL

Bogaty zestaw funkcji w najnowszych wersjach PL/SQL to wynik 13 lat (w czasie pisania książki) nieustannego rozwoju i usprawnienia tego języka przez Oracle. PL/SQL to język utworzony w odpowiedzi na określone potrzeby osób z firmy Oracle oraz spoza niej. Choć liczne funkcje zostały dodane w celu zaspokojenia wymagań programistów baz danych ze społeczności użytkowników, wiele innych wynika z potrzeb związanych z rozwojem aplikacji i szkoleniami prowadzonymi przez firmę Oracle. Jako programista uważam za dobry znak to, że osoby z korporacji Oracle wykorzystują w dużym stopniu te same technologie, których sam używam w pracy.

Trudno jest wyobrazić sobie bazę danych Oracle bez PL/SQL, jednak język ten został wprowadzony stosunkowo niedawno.

Wersje 1.x

Język PL/SQL 1.0 został przedstawiony w 1991 roku wraz z wersją 6.0 serwera bazodanowego. Jak można tego było oczekiwać po nowym języku programowania, brakowało w nim wielu funkcji, które powinny znaleźć się w bardziej dojrzałych wersjach. Jednak społeczność programistów używających Oracle doceniła go, ponieważ oferował możliwości — na przykład logikę struktury `IF-THEN` — niedostępne w owym czasie w języku SQL.

Wersje 2.x

Do czasu udostępnienia wersji 2.3 (wprowadzonej wraz z wersją 7.3 bazy danych) Oracle dodał obsługę procedur i funkcji składowanych, a także wiele wbudowanych pakietów. PL/SQL okazał się kluczem do sukcesu, jaki przyniosły narzędzia programistyczne autorstwa Oracle, a aplikacje wykorzystujące Oracle bazowały w dużym stopniu na ścisłej integracji języka PL/SQL z serwerem bazodanowym.

Wersja 8.0

W Oracle 8.0 dodano obsługę obiektów. Choć w początkowych wersjach nie obejmowała ona zbyt wielu funkcji, pozwalała się domyślić, w jakim kierunku Oracle planuje rozwijać język PL/SQL. Obiektowe rozszerzenia były dodawane w każdej następnej wersji, w tym także w najnowszej — 10*g*R1. Kolejną zmianą w wersji 8.0 była modyfikacja numeracji wersji PL/SQL oraz serwera bazodanowego. Od tego czasu numery kolejnych wydań PL/SQL odpowiadały wersjom serwera bazodanowego, z którym język ten jest zintegrowany.

Wersja 8.1

Wraz z udostępnieniem Oracle 8*i* rozwój funkcji języka PL/SQL zszedł na dalszy plan z powodu integrowania języka Java z „bazą danych dla internetu". Nie oznaczało to jednak, że w PL/SQL nie znalazło się nic nowego. Jedno z moich ulubionych rozszerzeń — macierzysty dynamiczny SQL (ang. *Native Dynamic SQL* — NDS) — obejmowało instrukcję `EXECUTE IMMEDIATE`. Uwielbiam to polecenie. Pojawia się ono w niemal każdym skrypcie do tworzenia schematów przedstawionym w przykładowym kodzie w tej książce.

Wersja 9.0

Oracle 9*i*R1 to istotna wersja w rozwoju PL/SQL. Poniższa lista przedstawia niektóre z najważniejszych usprawnień tego języka wprowadzonych w tej edycji:

- SQL i PL/SQL korzystają teraz z tego samego parsera, co gwarantuje spójność. Wcześniej instrukcje, które można było wywołać w oknie SQL*Plus, nie zawsze działały w PL/SQL.
- W 9*i*R1 dodano semantykę znaków, która umożliwia definiowanie precyzji zmiennych i kolumn zarówno w znakach, jak i w bajtach. Znaki Unicode nie są traktowane w taki sam sposób i mogą różnić się liczbą bajtów. Precyzja w Oracle jest określana w bajtach, a nie w znakach. Deklaracja zmiennej `VARCHAR2(2)` oznacza, że zmienna może przechowywać dwa bajty, a nie dwa znaki. Niektóre znaki azjatyckie zajmują trzy bajty, co oznacza, że przypisanie pojedynczego chińskiego znaku do zmiennej o precyzji 2 może się nie powieść. Może to być bardzo irytujące.
- Obsługa obiektów obejmuje dziedziczenie i ewolucję typów. Brak tych elementów był wyraźną słabością obsługi obiektowości w PL/SQL.
- Wbudowana kompilacja umożliwia kompilację kodu PL/SQL do kodu w języku C (system Oracle jest napisany w tym języku), co skraca czas wykonywania instrukcji, ponieważ w trakcie wykonywania programu nie jest konieczne przekształcanie kodu między językami.

Wersja 9.2

Wiele funkcji wprowadzonych w Oracle 9*i*R2 to usprawnienia rozszerzeń z 9*i*R1. Funkcje obiektowe zostały ulepszone poprzez dodanie wbudowanych funkcji oraz obsługi konstruktorów definiowanych przez użytkowników. W Oracle Text wprowadzono indeks `CTXXPATH`, który zapewnia usprawniony dostęp do dokumentów XML przechowywanych w zmiennych typu `XMLTYPE`.

Wersja 10.0

W PL/SQL 10.0 wprowadzono wiele nowych funkcji:

- Prawdopodobnie najważniejszym dodatkiem w PL/SQL 10*g*R1 jest obsługa wyrażeń regularnych. Przez długi czas były one symbolem skryptów uniksowych i perlowych, a obecnie są dostępne także w Oracle i obsługiwane w PL/SQL. Krótka definicja — wyrażenia regularne pozwalają wyszukiwać i pobierać wzorce tekstowe oraz manipulować nimi.
- Kolejna ciekawa nowa funkcja w 10*g*R1 to ostrzeżenia w trakcie kompilacji kodu. Nie chodzi tu o informowanie o błędach — ta funkcja jest dostępna od dawna. Obecnie można generować ostrzeżenia przy użyciu parametru `plsql_warnings` lub pakietu `DBMS_WARNING`. Te ostrzeżenia informują o potencjalnych problemach z wydajnością i mniejszych kłopotach, które nie powodują błędów w czasie kompilacji.
- Nowe typy danych — `BINARY_FLOAT` i `BINARY_DOUBLE` — to wbudowane zmiennoprzecinkowe typy danych, których można używać jako alternatywy dla typu `NUMBER`.

- Pakiet DBMS_LOB zapewnia obsługę dużych obiektów typu LOB — od 8 do 128 terabajtów (w zależności od rozmiaru bloku). Więcej informacji na ten temat znajduje się w rozdziale 16.
- Dostosowywanie symboli w literałach znakowych. Jeśli programista ma dosyć umieszczania dwóch apostrofów w literałach znakowych, może użyć konstrukcji q'!...!' i umieszczać łańcuchy znaków w obrębie wykrzykników. Pozwala to użyć jednego apostrofu w łańcuchu znaków zamiast dwóch. Poniżej znajduje się przykład zastosowania tej techniki w anonimowym bloku:

```
SET SERVEROUTPUT ON
BEGIN
   DBMS_OUTPUT.PUT_LINE('Ron's');
END;
/
```

Ten blok spowoduje wyświetlenie następującego błędu:

```
ORA-01756: quoted string not properly terminated
```

Aby rozwiązać ten problem, wcześniej trzeba było użyć dwóch symboli apostrofu — 'Ron''s'. W 10*g*R1 można zastosować inną technikę:

```
BEGIN
   DBMS_OUTPUT.PUT_LINE(q'!Ron's!');
END;
/
```

Ten kod wykona się poprawnie i wyświetli zgodnie z oczekiwaniami napis Ron's.

Podstawy języka

Ten punkt przedstawia pewne podstawowe cechy języka PL/SQL, takie jak możliwość wykonywania kodu bez zapisywania go, zachowywania kodu w celu późniejszego użycia oraz różnice między poszczególnymi typami przechowywanych obiektów. W tym miejscu zagadnienia te są opisane ogólnie, w celu przedstawienia pewnych funkcji. Ich bardziej szczegółowy opis znajduje się w rozdziałach 3., 4., 8. i 9.

Bloki anonimowe

Bloki anonimowe to fragmenty kodu, które nie są zachowywane i nie mają nazwy. System wykonuje je w obrębie sesji i nie można ich wywoływać z poziomu innych sesji. Aby ponownie wykonać ten sam kod, trzeba zapisać anonimowy blok w pliku systemu operacyjnego i uruchomić go, ponownie napisać instrukcje lub dołączyć dany blok do programu, który w odpowiednim momencie wywoła kod bloku.

W przykładach przedstawionych w tej książce bloki anonimowe są wykorzystywane wielokrotnie. Takie bloki doskonale nadają się do pisania skryptów lub operacji, których programista nie zamierza wykonywać zbyt często. Poniższy kod zawiera przykładowy blok anonimowy:

```
SET SERVEROUTPUT ON
DECLARE
   v_Date TIMESTAMP;
BEGIN
   SELECT systimestamp - 1/24
   INTO v_Date
   FROM dual;
   DBMS_OUTPUT.PUT_LINE('Godzinę temu: '||v_Date);
END;
/
```

Bloki anonimowe rozpoczynają się od instrukcji `DECLARE` lub `BEGIN` i nie są zachowywane po wykonaniu.


Uwaga

Blok w języku PL/SQL to kompletny fragment kodu tego języka. Program PL/SQL składa się z jednego bloku lub z kilku bloków, między które programista logicznie podzieli operacje. Bloki można także umieszczać wewnątrz innych bloków. Kompletna analiza struktury bloków znajduje się w rozdziale 3.

Procedury

Procedury mają nazwy i są przechowywane. Mogą zwracać **wartość** po wykonaniu, ale **nie muszą** tego robić. Jedyne, co trzeba zwrócić, to informację o sukcesie lub niepowodzeniu wykonania operacji.

Procedury składowane (lub nazwane) otrzymują niepowtarzalną nazwę w momencie ich tworzenia. Należą do użytkownika, który je utworzył, chyba że w tworzącym je skrypcie określono inaczej.

Procedury można wywoływać z wiersza poleceń SQL*Plus, w skryptach języka SQL oraz w innych blokach kodu PL/SQL.

Funkcje

Funkcje różnią się od procedur tym, że **muszą** zwracać wartość. Ich struktura jest bardzo podobna do procedur, a największą różnicą jest wymagana klauzula `RETURN`. Funkcje mają nazwy i można je wywoływać z wiersza poleceń SQL*Plus, w skryptach języka SQL oraz w innych blokach kodu PL/SQL. Przy wykonywaniu funkcji trzeba jednak zapewnić obsługę zwróconej wartości.

Pakiety

Pakiety to logicznie pogrupowane zbiory procedur i funkcji. Składają się z dwóch części: specyfikacji i ciała.

Specyfikacja jest publiczna i przedstawia strukturę pakietu. W opisie pakietu w SQL*Plus widoczna jest właśnie specyfikacja. Zawsze powstaje ona lub jest kompilowana przed ciałem. Możliwe jest nawet utworzenie specyfikacji bez ciała.

Typy obiektowe

Typy obiektowe w Oracle umożliwiają pisanie obiektowego kodu w PL/SQL. Typy obiektowe przypominają strukturą pakiety i mają specyfikację oraz ciało. Takie typy zapewniają poziom abstrakcji dla struktur danych, na których bazują.

Typy obiektowe mogą zawierać atrybuty i metody. **Atrybuty** definiują cechy obiektów. Na przykład książka może mieć atrybuty opisujące tytuł, liczbę stron itd.

Metody działają na strukturach danych obiektu. Cała interakcja między aplikacją a danymi obiektu powinna się odbywać za pośrednictwem metod.

Niektóre z zalet stosowania typów obiektowych to:

- **Abstrakcja** — autor aplikacji nie musi zastanawiać się nad relacyjnymi strukturami danych i może myśleć w kategoriach struktur z rzeczywistego świata.
- **Spójność** — jeśli cała interakcja w aplikacji odbywa się poprzez obiekty, a nie bezpośrednio przy użyciu struktur danych, prawdopodobieństwo powstania błędów w danych jest dużo mniejsze.
- **Prostota** — zamiast przekształcać model świata rzeczywistego na kod, można pozostawić go w tym świecie. Jeśli chcę dowiedzieć się czegoś o obiekcie reprezentującym książkę, zaglądam do obiektu **książka**.

Funkcje wprowadzone w wersji Oracle 9*i*R1 obejmują dziedziczenie, dynamiczne wiązanie metod i ewolucję typów. Te cechy sprawiają, że programowanie obiektowe za pomocą PL/SQL daje obecnie dużo więcej możliwości.

Przetwarzanie instrukcji PL/SQL

W trakcie wykonywania bloku PL/SQL kod trafia do silnika przetwarzającego język PL/SQL. Ten silnik może wchodzić w skład samego serwera bazodanowego albo znajdować się w jednym z narzędzi (na przykład w Oracle Raports) obejmujących silnik PL/SQL. Następnie silnik przetwarza kod i przekazuje instrukcje w języku SQL do silnika przetwarzającego ten język lub do narzędzia wykonującego instrukcje SQL. Instrukcje proceduralne trafiają do wykonującego je narzędzia, gdzie są przetwarzane.

Tryb interpretowany

Oracle domyślnie działa w trybie **interpretowanym**. Oznacza to, że składowane procedury, funkcje i pakiety są kompilowane i przechowywane jako kod PL/SQL, a następnie interpretowane przez Oracle (napisany w C) w trakcie wykonywania programu. W trybie interpretowanym kompilacja kodu PL/SQL jest szybsza, jednak wykonywanie instrukcji może być wolniejsze niż przy uprzedniej kompilacji do języka macierzystego.

Kompilacja do języka macierzystego

Kompilacja do języka macierzystego, po raz pierwszy wprowadzona w Oracle 9*i*R1 i usprawniona w wersji 10*g*R1, powoduje przekształcenie kodu PL/SQL na język C w trakcie kompilacji. Przyspiesza to działanie kodu nawet o 30 procent, ponieważ w trakcie wykonywania programu nie jest potrzebna interpretacja.

Jak najlepiej wykorzystać tę książkę?

Książka została starannie sprawdzona i obejmuje zagadnienia dla początkujących, średnio zaawansowanych i zaawansowanych programistów. Do demonstracji poszczególnych funkcji języka służy kod, który w całości można pobrać z internetu. Na witrynie internetowej książki znajduję się pakiet z katalogami odpowiadającymi rozdziałom. W tym pakiecie znajduje się cały kod opisany w niniejszej książce. Kod powiązany z każdym rozdziałem działa niezależnie od kodu z pozostałych części książki — nie występują zależności między kodem z różnych rozdziałów. Dla ułatwienia testów dołączone zostały skrypty do tworzenia schematów. Należy zmodyfikować je tak, aby odpowiadały środowisku i regułom dostępu w używanej bazy danych.

Niektóre zagadnienia wymagają więcej miejsca, niż można im było poświęcić w tej książce. Zamiast ograniczać objętość w celu dopasowania treści do książki, utworzyliśmy dodatkowe materiały, które można pobrać, rozszerzające zagadnienia przedstawione w poszczególnych rozdziałach. Mamy nadzieję, że okażą się one przydatne.

Odbiorcy

Ta książka została napisana z myślą zarówno o początkujących, jak i doświadczonych programistach aplikacji w języku PL/SQL, a także o administratorach baz danych, którzy chcą wykorzystać możliwości tego języka. Rozdziały dla zaawansowanych (od 11. do 17.) wymagają opanowania materiału z rozdziałów od 1. do 10. Także doświadczeni programiści języka PL/SQL powinni przejrzeć kilka pierwszych rozdziałów. Znajduje się tam analiza nowych funkcji i przykładowy kod, który może nasunąć nowe pomysły związane z tworzeniem własnych aplikacji.

Jestem przekonany, że niezależnie od doświadczenia programiści znajdą coś nowego w każdym rozdziale.

Cel

PL/SQL to dojrzały, stabilny język, usprawniany z każdą wersją. Wraz ze wzrostem złożoności nadążanie za zmianami staje się trudnym zadaniem. Ta książka ma pomóc:

- ♦ w nauce PL/SQL programistom, którzy jeszcze nie znają tego języka;
- ♦ w rozwinięciu dobrego stylu i nauce pisania wydajnego kodu;

- w zrozumieniu funkcji, które w innych materiałach są jedynie wspomniane lub w ogóle pominięte;
- w odkryciu, jak wiele możliwości daje ten język.

Zakres

Każda książka ma pewne ograniczenia. W tym przypadku są to:

- Dość ograniczone ujęcie zagadnień związanych z administracją bazami danych. Jeśli chcesz nauczyć się czegoś więcej na ten temat, warto zajrzeć na stronę *http://otn.oracle.com* lub do jednej z doskonałych książek o administracji bazami danych wydawnictwa Oracle Press.
- Opis zagadnień związanych z poprawą wydajności jest ograniczony do efektywnego stosowania PL/SQL. Brak jest technik zwiększania wydajności bazy danych.
- Autor może jedynie udostępnić informacje, porady i przykłady. To od Ciebie zależy, czy wykorzystasz te materiały i zaczniesz pisać dobry kod.

Założenia

Główną wersją, dla której została napisana ta książka, jest Oracle 8.1.7.4 — ostateczna wersja bazy danych Oracle 8*i*. Omawiane zagadnienia obejmują wersje 8.1.7, 9*i*R1, 9*i*R2 i 10*g*R1. Aby jak najlepiej skorzystać z tej książki oraz nowych funkcji Oracle, zalecamy pobranie i instalację bazy Oracle 10*g*R1 ze strony OTN (*http://otn.oracle.com*). Po bezpłatnej rejestracji można pobrać serwer bazodanowy.

10*g*R1 to pakiet instalacyjny zajmujący jedną płytę, dlatego pobranie tej edycji będzie dużo szybsze niż jakiejkolwiek wersji 9*i*.

Minimalnym wymaganiem jest dostęp do bazy Oracle i posiadanie odpowiednich uprawnień do utworzenia użytkownika oraz potrzebnych obiektów. Ważne jest, aby zwrócić uwagę na używaną wersję PL/SQL, ponieważ od tego zależy, jakie funkcje będą dostępne.

Począwszy od Oracle 8 numery wersji PL/SQL są zgodne z wersjami bazy danych. W materiałach dotyczących wersji starszych niż Oracle 8 występują oznaczenia takie jak PL/SQL 1.1, 2.X itd. Aby określić, jakiej wersji używasz, sprawdź zawartość widoku `V$VERSION`.

```
SELECT banner
FROM v$version;
```

Ta instrukcja `SELECT` w aktualnie używanym przez mnie środowisku zwróciła następujące informacje:

```
BANNER
----------------------------------
Oracle Database 10g Enterprise Edition Release 10.1.0.2.0 - Prod
PL/SQL Release 10.1.0.2.0 - Production
```

```
CORE    10.1.0.2.0      Production
TNS form 32-bit Windows: Version 10.1.0.2.0 - Production
NLSRTL Version 10.1.0.2.0 - Production
```

Wynika z tego, że używam wersji 10.1.0.2.0 bazy danych, a także wersji 10.1.0.2.0 języka PL/SQL.

Konwencje

W książce używam różnych czcionek do zaznaczenia i wyróżnienia poszczególnych fragmentów tekstu. Przykładowy kod oraz zewnętrzne odwołania do obiektów bazodanowych w tekście są pisane czcionką COURIER. Odwołania do zmiennych w tekście także są pisane tą czcionką. Szczególnie istotne fragmenty w przykładowym kodzie są wyróżnione **pogrubieniem**. Specjalną uwagę zwróć także na Uwagi i Wskazówki pojawiające się w książce.

Przykłady

Do każdego rozdziału dołączone są skrypty do tworzenia użytkownika, które nadają uprawnienia potrzebne do uruchomienia kodu z danego rozdziału. Nie należy korzystać ze skryptów do tworzenia schematów z jednego rozdziału do uruchamiania kodu z innych rozdziałów.

Kod w większości rozdziałów używa obiektów powiązanych z księgarnią. Podstawowe tabele w większości przykładów to BOOKS i AUTHORS, przedstawione już we wcześniejszej części rozdziału na rysunku 1.1. W poszczególnych rozdziałach projekt schematu może być nieco inny, co umożliwia przedstawienie różnych funkcji.

Struktura tabeli KSIAZKI wygląda następująco:

```
DESC BOOKS
 Name                                  Null?    Type
 ------------------------------------- -------- ----------
 ISBN                                  NOT NULL CHAR(10)
 CATEGORY                                       VARCHAR2(20)
 TITLE                                          VARCHAR2(100)
 NUM_PAGES                                      NUMBER
 PRICE                                          NUMBER
 COPYRIGHT                                      NUMBER(4)
 AUTHOR1                                        NUMBER
 AUTHOR2                                        NUMBER
 AUTHOR3                                        NUMBER
```

Poniżej przedstawiona jest struktura tabeli AUTHORS:

```
DESC autorzy
 Name                                  Null?    Type
 ------------------------------------- -------- ----------
 ID                                    NOT NULL NUMBER
 FIRST_NAME                                     VARCHAR2(50)
 LAST_NAME                                      VARCHAR2(50)
```

Skrypt do tworzenia schematu z rozdziału 16., *CreateUser.sql*, używa instrukcji `create table` do utworzenia dwóch przestrzeni tabel przeznaczonych na parametr magazynowania. Nazwy przestrzeni tabel i nazwy plików z danymi oraz lokalizacje można zmodyfikować tak, aby dopasować je do używanego środowiska.

Warto poświęcić chwilę czasu na przegląd różnych metod tworzenia schematów i przykładów. Staraliśmy się stosować w książce różne techniki, aby pokazać, że to samo zadanie można wykonać na kilka sposobów. Przeglądając przykładowy kod, warto pomyśleć o tym, jak zastosować niektóre z przedstawionych technik, konwencji nazewniczych i strategii w projektowaniu własnych aplikacji. Pomysłowość, zdolności komunikacyjne, nastawienie, dyscyplina i wiedza pozwolą Ci rozwinąć karierę programisty Oracle PL/SQL oraz pomogą we wszystkich przedsięwzięciach, jakich się podejmiesz.

Podsumowanie

W tym rozdziale omówiliśmy zagadnienia programistyczne, opisaliśmy, jak PL/SQL wpasowuje się zarówno w obiektowe, jak i proceduralne programowanie, a także przedstawiliśmy pokrótce niektóre funkcje języka PL/SQL, które są opisane w dalszej części książki. Rozdział obejmuje także podstawy działania relacyjnych baz danych i języka SQL oraz porównanie PL/SQL z językami SQL i Java. Na końcu rozdziału znajduje się omówienie książki wraz z poradami, które pomogą jak najlepiej skorzystać z lektury.

Mamy nadzieję, że niniejsza książka okaże się przydatna i odkryjesz dzięki niej, że za pomocą PL/SQL można wykonywać operacje, o których nawet nie myślałeś.