ORACLE® *Oracle Press*™

ORACLE 8*i*
PL/SQLプログラミング

ORACLE 8*i*

Advanced PL/SQL Programming

Scott Urman／著
SE編集部／訳
日本オラクル株式会社／監修

SE
SHOEISHA

本書内容に関するお問い合わせについて

このたびは翔泳社の書籍をお買い上げいただき、まことにありがとうございます。弊社では、読者の皆様からのお問い合わせに適切に対応させていただくため、以下のガイドラインへのご協力をお願いしております。下記項目をお読みいただき、手順に従ってお問い合わせください。

●ご質問される前に

弊社Webサイトの「Q&Aコーナー」（http://www.shoeisha.com/info/help.asp）をご参照ください。これまで受けたご質問への回答（FAQ）や、的確なご質問方法に関する情報を掲示しています。

●ご質問方法

弊社Webサイトの質問専用フォーム（http://www.shoeisha.com/book/qa/）をご利用ください。記載漏れや独自の用紙等によるご質問、お電話や電子メールによるお問い合わせ、本書にはさみ込まれたアンケートはがきに記入されたご質問等は、お受けしておりません。

※質問専用シートのお取り寄せについて

Webサイトにアクセスする手段をお持ちでない方は、ご氏名、ご送付先（ご住所／郵便番号／電話番号またはFAX番号／電子メールアドレス）および「質問専用シート送付希望」と明記のうえ、電子メール（qaform@shoeisha.com）、FAX、郵便（80円切手をご同封願います）のいずれかにて“編集部読者サポート係”までお申し込みください。申し込まれた手段によって、折り返し質問シートをお送りいたします。シートに必要事項を漏れなく記入し、“編集部読者サポート係”までFAXまたは郵便にてご返送ください。

●回答について

回答は、ご質問いただいた手段によってご返事申し上げます。ご質問の内容によっては、回答に数日ないしはそれ以上の期間を要する場合があります。

●ご質問に際してのご注意

本書の対象を越えるもの、記述箇所を特定されていないもの、また読者固有の環境に起因するご質問等にはお答えできませんので、予めご了承ください。

●郵便物送付先およびFAX番号

送付先住所　：〒160-0006 東京都新宿区舟町5
FAX番号　：03-5362-3818
宛先　：（株）翔泳社出版局 編集部読者サポート係

CONTENTS

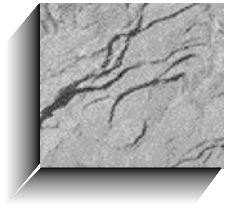
謝辞

このような本を仕上げるのはとても時間のかかる仕事です。特に今回の場合はたいへんでした。一年前にこの仕事に取りかかったときには、こんなに長くかかるとは思いもしませんでした。しかし、時間をかけただけの甲斐があって、よい本になったと思います。

この仕事でお世話になった大勢の人たちにお礼を申し上げなければなりません。まず誰よりも、技術的な校閲をしてくれた人たちに感謝します。今回は、Oracleのいろいろな分野から参加してくれた4人の人物が、この本の技術的な誤りを見つけて正してくれました。Sharon Castledine、Bruce W. Chang、Ali Shehadeh、Simon Slackの4氏です。そのような誤りはもう残っていないと思いますが、もしあったとすれば私の責任です。Osborne社のMonica Faltiss、Jeremy Judson、Janet Waldenの各氏にも感謝します。特に最後の2、3カ月間、何とか私がこの仕事をやり遂げられるように叱咤激励してくれたからです。そして、いろいろな感想や忠告を述べてくれた家族や友人(名前をあげなくてもわかってくれると思う)にももちろん感謝しています。

この本の執筆中には、Oracleのマニュアルも含め、さまざまなマニュアルを参照しました。それらのマニュアルには、『PL/SQL ユーザー・ガイドおよびリファレンス』、『Oracle8*i* アプリケーション開発者ガイド』、『Oracle8*i* 管理者ガイド』、『Oracle8*i* SQL リファレンス』、『Oracle8*i* 概要』、『Oracle Pro*C/C++ プリコンパイラ・プログラマーズ・ガイド』などがあります。

本書に関するご意見は、**surman@us.oracle.com**まで、電子メールでお寄せください。本書の前身『ORACLE PL/SQLプログラミング』および『ORACLE8 PL/SQLプログラミング』については、いろいろなご意見ご要望をいただきました。この場を借りてお礼申し上げます。



はじめに

Oracleは、非常にパワフルで柔軟性のあるリレーショナルデータベースシステムです。しかし、そのパワーと柔軟性の背後には、複雑さもあります。便利なアプリケーションをOracleで設計するには、システム内のデータをOracleがどのように操作するか、理解しておく必要があります。PL/SQLは、内部（Oracleの内部）と外部（ユーザーアプリケーションの内部）の両方の世界において、データ操作用に設計された重要なツールです。PL/SQLは、さまざまな環境で使用できますが、その利点は環境ごとに異なっています。

PL/SQLについて私が書いた最初の本は、*Oracle PL/SQL Programming*（日本語版は『ORACLE PL/SQLプログラミング』小社刊）でした。そこでは、PL/SQL 2.3とOracle7.xを扱っていました。当時はそれが最新バージョンだったからです。それを改訂したのが*Oracle8 PL/SQL Programming*（日本語版は『ORACLE8 PL/SQLプログラミング』小社刊）で、より内容を充実させ、Oracle8についての情報ももちろん盛り込んでいました。そして、3番目がこの本です。

本書でも、内容の拡充と、最新バージョンであるOracle8*i*についての解説を意図しています。ただし、ボリュームの関係で以前の本の内容を全部残すことはできなかったので、今回はPL/SQLのより高度な使い方に比重を置いて記述を進めました。それゆえ、初めてPL/SQLを勉強しようという人は、以前の本から読み始めたほうがよいかもしれません。本書には、以前の本の内容を下敷きにした部分もあれば、まったく新たに書きおこした部分もあります。Oracle8*i*の新しい機能に興味をお持ちの方は、ぜひ本書をお読みください。

新しくなったところ

では今回新しくなったところはどこでしょうか？　最大の違いは、言うまでもなく、Oracle8*i*に関する記述を全編にわたって盛り込んだことです。たとえばChapter 8では、Oracle7.1から使えるようになったDBMS_SQLパッケージと、Oracle8*i*の新機能であるネイティブな動的SQLの両方を含めて、動的SQLについて解説しています。また、PL/SQLの開発&デバッグ環境についての説明も大幅に拡張されています。

PL/SQLの実例

みなさんが本書を読むのは、実際のアプリケーションにPL/SQLを使用することを考えておられるからでしょう。PL/SQLはそのような目的で設計された言語ですから、それは当然のことです。こうした目的に役立つように、私は本書全般に「PL/SQLの実例」を数多く含めるようにしました。これらのサンプルコードは、そのままアプリケーションに応用しても、またはたたき台として利用されても構いません。実はこれらのサンプルコードのアイデアは、みなさん、つまり旧版の読者の方々からいただいたものです。ですから、 PL/SQLが実際に使われる現場にきわめて適した例だと思っています。

サンプルソースコード

本書中で使用しているソースコードは、Webサイトからダウンロードすることができます。完全な形のソースコードが、原書の出版元であるOsborne/McGraw-Hill社のWebサイト（**http://www.osborne.com/oracle/**）からダウンロードできます。ただし、こちらのソースコードはコメントなどが日本語化されていません。

本書中のソースコードの冒頭に、「`--Available online as XXX.sql`」と書かれている場合、「`XXX.sql`」は、Osborne/McGraw-Hill社のWebサイトからダウンロードできるソースコードファイルの名前を示します。

対象読者

本書はPL/SQLに関して、ユーザーガイドとしてもリファレンスマニュアルとしてもご利用いただけます。主要な対象読者は、すでにPL/SQLを扱った経験があって、さらに高度な使い方やOracle8*i*で追加された機能に興味をお持ちの方々です。初めてPL/SQLに接するという方には、本書の前身『ORACLE8 PL/SQLプログラミング』と併せてお読みになることをお勧めします。2つの本には重複する部分もありますが、PL/SQLの全体像を理解するにはそれがベストです。

本書の内容

本書は16の章(Chapter)と3つの付録(Appendix)から構成されています。16の章は大きく3つの部(Part)に分かれます。すなわち、「概要と環境」、「リレーショナル機能」、「オブジェクト機能とラージオブジェクト」の3部構成になっています。

Part I　PL/SQLの基本と開発環境

ここでは、PL/SQLの概要とその実行環境および開発環境について解説します。

Chapter 1　PL/SQLの概要

この章ではPL/SQLという言語の主要な特長を紹介します。PL/SQLのバージョンと、それに対応するOracleデータベースのバージョンについても説明します。また、本書を通じてサンプルとして使用するデータベーススキーマもここで紹介します。

Chapter 2　PL/SQLの開発環境と実行環境

PL/SQLは、クライアントやサーバーなど、いろいろな種類の環境で実行することができます。この章では、PL/SQLエンジンの置き場所や、各種エンジン間の通信について説明します。

Chapter 3　トレースとデバッグ

この章では、アプリケーションのデバッグを行うためのいろいろなテクニックを紹介します。PL/SQLコードにありがちな問題を見つけて解決する方法や、既存のアプリケーションのトレースを行う方法も説明します。また、Oracle8*i*で利用できるようになった新しいプロファイラを紹介します。

Prat II　リレーショナル機能

ここでは、PL/SQLのリレーショナル機能について解説します。取り上げる内容は、プロシージャ、トリガー、データベースジョブとファイルI/O、動的SQL、セッション間通信、外部ルーチンなどです。また、PL/SQLに関してOracle8*i*で新たに追加された要素についても、ここで紹介します。

Chapter 4　サブプログラムとパッケージの作成

サブプログラム(プロシージャとファンクション)は、PL/SQLコードを、名前の付いたコール可能なブロックとして構成したものです。サブプログラムは、データベース内に格納していつでも使用することができます。パッケージは、関連するサブプログラムと宣言を、1つのユニットとして格納できるようグループ化したものです。この章では、サブプログラムとパッケージを作成するための構文を説明します。

Chapter 5　サブプログラムとパッケージの使用

前章の続きとして、サブプログラムとパッケージの使い方を説明します。SQL文を使ってサブプログラムやパッケージをコールする方法、サブプログラムやパッケージ同士の依存性、パッケージと共有プールの対話関係、などを取り上げます。また、起動者権限プロシージャなどのOracle8*i*の新機能も紹介します。

Chapter 6　トリガー

トリガーは、引き金となるイベントが生起したときに自動的に実行される特別なPL/SQLブロックです。トリガーイベントになりうるのは、INSERT文などのDML操作、DDL文、あるいはシステムイベントです。トリガーは所定のDML文の代わりに実行することもできます。この章では、あらゆる種類のトリガーについて説明します。

Chapter 7　データベースジョブとファイルI/O

DBMS_JOBパッケージを使うと、PL/SQLジョブを(ストアドプロシージャの形式で)所定の時間に自動的に実行されるようにすることができます。UTL_FILEパッケージは、PL/SQLがOSファイルを読み込んだり書き込んだりできるようにします。この章では、この2つのパッケージについて例を挙げて詳しく説明します。

Chapter 8　動的SQL

動的SQLは非常にパワフルなプログラミング技法で、柔軟性に富んだプログラムを書くことができます。この章では、DBMS_SQLパッケージとネイティブな動的SQLについて説明します。これらを使うと、実行時に作成・発行されるSQL文およびPL/SQLブロックを記述することができます。この方法は、DML文しか使用できないPL/SQLの制約を乗り越えたいときに使います。

Chapter 9　セッション間通信

データベースセッション間で直接通信することを可能にする組み込みパッケージが2つあります。データベースパイプ(DBMS_PIPE)とデータベースアラート(DBMS_ALERT)です。この章では、これら2つのパッケージの比較と、実用例を見ていきます。

Chapter 10　外部ルーチン

外部ルーチンでは、CやJavaで書かれたファンクションをPL/SQLから直接コールすることができます(CはOracle8以降、JavaはOracle8*i*以降で対応可能)。Cルーチンは別のプロセスで実行されますが、Javaルーチンは、Oracle8*i*に含まれるJava仮想マシン(JServer)によって、同じ1つのプロセスの中でPL/SQLとしてデータベースに読み込まれ実行されます。この章では、これら2種類の外部ルーチンについて詳しく説明します。

Chapter 11　Oracle8*i* の新機能

この章では、PL/SQLに関してOracle8*i*で新たに追加されたすべての要素について簡単に紹介します。他の章では取り上げることができなかった事柄については詳しく説明し、Oracle8とのパフォーマンスの比較結果も示します。

Part III　オブジェクト機能とLOB

ここでは、Oracle8およびOracle8*i*のオブジェクト機能について解説します。また、ラージオブジェクト(LOB)についても説明します。LOBとは、データベースの1つの列に4ギガバイトまでのデータを格納するものです。

Chapter 12　オブジェクトの概要

この章では、Oracle8のオブジェクト型を宣言するための構文と方法を、オブジェクトインスタンスやコールメソッドを初期化する方法も含めて説明します。また、Oracle8*i*から利用できるようになった静的メソッドについても説明します。

Chapter 13　データベース内のオブジェクト

この章では、データベースにオブジェクトを格納する方法と、SQLやPL/SQLを使ってそれらのオブジェクトを操作する方法について説明します。また、オブジェクト参照とその使い方についても説明します。

Chapter 14　コレクション

コレクションはPL/SQLオブジェクトをグループ化したもので、索引付き表、NESTED TABLE、VARRAYも含まれます。これらのデータ型は、他の言語でいう配列に似ています。この章では、NESTED TABLEやVARRAYをデータベースに格納する方法を、コレクションメソッドと併せて説明します。

Chapter 15　ラージオブジェクト (LOB) の概要

Oracle8のLOBは文字データ(それもいろいろな文字セットの文字データ)とバイナリデータの両方を格納できます。この章では、さまざまなLOBの型と、LOBにアクセスするためのインタフェースについて説明します。SQLのLOBインタフェースについては特に詳しく説明します。

Chapter 16　LOBの操作とDBMS_LOB

前章の続きとして、DBMS_LOBパッケージついて詳しく説明します。このパッケージは、LOBをPL/SQLから操作するためのものです。PL/SQLパッケージなので、PL/SQLブロックを発行できる環境でならどこでも使用できます。

Appendixes

PL/SQLについての便利なリファレンスです。

Appendix A　組み込みパッケージ

言語の能力を大きく拡張するOracleの組み込みパッケージについて説明します。他の章では取り上げることができなかったパッケージを紹介します。

Appendix B　PL/SQLの予約語

PL/SQLのすべての予約語の一覧です。変数やその他のPL/SQLオブジェクトに、これらの語を使用してはいけません。

Appendix C　データディクショナリ

データディクショナリのビューについて説明します。PL/SQL関連のビューについては特に詳しく説明します。

PART I

PL/SQLの基本と開発環境

CHAPTER 1

PL/SQLの概要

PL/SQLは、各種の環境でOracleデータベースにアクセスする際に使用する高度なプログラミング言語です。PL/SQLは、PL/SQLコードを高速に効率よく処理できるよう、データベースサーバーに搭載されます。PL/SQLはまた、Oracleツールによってはクライアント側でも利用できます。この章では、PL/SQLを使用する理由、開発方法、PL/SQL言語の主要な機能、およびPL/SQLとデータベースのバージョンについて説明します。また、本書全体を通して詳しく学習していく概念についても紹介します。さらに、この章の末尾では、本書に関する表記規則と、例で使用するデータベースの表について説明します。

PL/SQLを使う理由

Oracleは、リレーショナルデータベースです。リレーショナルデータベースへのアクセスに使用する言語がSQL（Structured Query Language：構造化問い合わせ言語）であり、これは「シークェル」とも読まれます。SQLは、柔軟で効率のよい言語であり、リレーショナルデータベースの操作および検索を行うための機能を備えています。たとえば、次のSQL文は、栄養学専攻のすべての学生に関するデータをデータベースから削除します。

```
DELETE FROM students
  WHERE major = 'Nutrition';
```

students表など、本書で使用するデータベースの表については、16ページにある「例で使用する表」で紹介します。

SQLは、「第4世代言語（4GL）」です。つまりSQLでは、処理方法ではなく、処理の目的を記述します。たとえば、上記のDELETE文の場合、データベースが栄養学専攻の学生を実際にどのようにして調べているのか、我々にはわかりません。おそらく、サーバーは何らかの順序ですべての学生をループ処理によって調べ、削除の対象となる適切なエントリを決定しています。しかし、その詳細は、我々には見えません。

一方、CやCOBOLなどの「第3世代言語（3GL）」は、よりプロシージャ型です。つまり、第3世代言語でプログラミングする際は、問題を解決するためのアルゴリズムを、1ステップずつ記述します。たとえば、DELETE処理は、次のように記述します。

```
LOOP over each student record
  IF this record has major = 'Nutrition' THEN
    DELETE this record;
  END IF;
END LOOP;
```

C++やJavaのようなオブジェクト指向言語も第3世代の言語です。これらは、オブジェクト指向の設計の原理は受け入れていますが、アルゴリズムはまだ段階的に設計されているところです。どのタイプの言語にも、長所と短所があります。SQLなどの第4世代言語は、第3世代言語に比べてかなりシンプルであり、コマンドの数も少ないのが一般的です。また、4GLの場合、基本とな

るデータ構造やアルゴリズムをユーザーが知る必要はありません。これらはランタイムシステムによってインプリメントされています。しかし、3GLで使用するプロシージャ型構造体を用いたほうが、プログラムをうまく記述できる場合もあります。そこで登場するのが、PL/SQLなのです。PL/SQLは、SQL(4GL)の能力および柔軟性と、3GLのプロシージャ型構造体の両方を備えています。

PL/SQLは、Procedural Language/SQLを短縮したものです。その名前のとおり、PL/SQLは、他のプロシージャ型言語が持つ次のような構造体を追加することによって、SQLの機能を拡張します。

- 変数と型(定義済みのものとユーザー定義のもの)
- IF-THEN-ELSE文やループなどの制御構造
- プロシージャとファンクション
- オブジェクト型とメソッド(PL/SQL 8以降)

OracleのSQLは、プロシージャ型の構造体がうまく統合されているため、パワフルな構造化言語になっています。たとえば、ある学生の専攻を変更する場合を考えてみましょう。その学生が存在していない場合は、新しいレコードを作成するものとします。PL/SQLでは、この処理を次のようにコーディングできます。

```
-- Available online as 3gl_4gl.sql
SQL> DECLARE
  2    /* SQL文で使用する変数を宣言する。 */
  3    v_NewMajor VARCHAR2(10) := 'History';
  4    v_FirstName VARCHAR2(10) := 'Scott';
  5    v_LastName VARCHAR2(10) := 'Urman';
  6  BEGIN
  7    /* students表を更新する。 */
  8    UPDATE students
  9      SET major = v_NewMajor
 10      WHERE first_name = v_FirstName
 11      AND last_name = v_LastName;
 12    /* レコードがあるか確認する。もしなければ、
         レコードを挿入する。*/
 14    IF SQL%NOTFOUND THEN
 15      INSERT INTO students (ID, first_name, last_name, major)
 16        VALUES (student_sequence.NEXTVAL, v_FirstName, v_LastName,
 17                v_NewMajor);
 18    END IF;
 19  END;
```

この例には、4GL構造体である異なる2つのSQL文(UPDATEとINSERT)と、3LG構造体(変数の宣言とIF条件文)が含まれています。

注意

前述の例を実行するには、まず参照されるデータベースオブジェクト(**student表**と**student_sequence**という順序)を作成する必要があります。これには、**RelTables.sql**スクリプトを使用します。このスクリプトはOsborne McGraw-Hill 社のWebサイト(http://www.osborne.com/oracle)からダウンロードできます。

PL/SQLは、SQLの柔軟性と3GLのパワーおよび小回りの良さを両方とも備えている点で、ユニークな存在です。データベースにアクセスできるだけでなく、必要なプロシージャ型構造体もすべて備えています。そのため、PL/SQLは堅牢でパワフルであり、複雑なアプリケーションの設計に適した言語です。

PL/SQLとネットワークトラフィック

多くのデータベースアプリケーションは、クライアント／サーバーモデルまたは3階層モデルを使用して作成されています。クライアント／サーバーモデルでは、プログラム自体はクライアントマシンに存在し、データベースサーバーに要求を送信して情報を入手します。それらの要求はSQLで実行します。通常、SQL文は1度に1つずつ送信されるため、ネットワーク通信量が増えてしまいます。その様子は、**図1-1**の左側のようになります。これを、右側の図と比べてみてください。複数のSQL文を1つのPL/SQLブロックにまとめ、1つの単位としてサーバーに送信できるのです。そうすることにより、ネットワークトラフィックが減り、アプリケーションも高速になります。

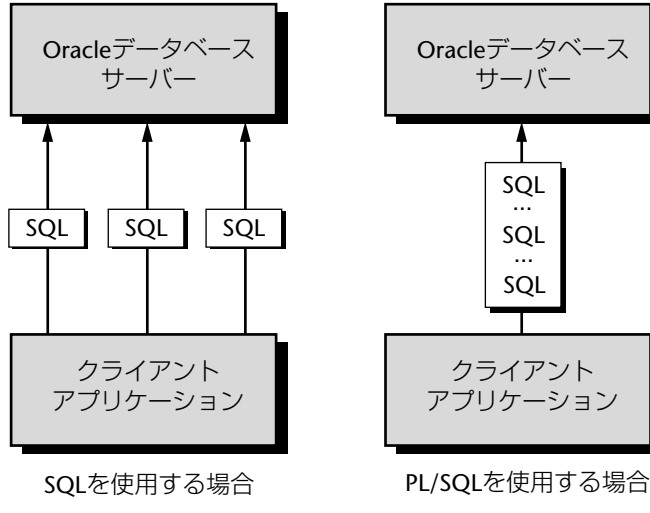


図1-1　クライアント／サーバー環境でのPL/SQL

クライアントとサーバーの両方が同じマシン上で動作している場合でも、パフォーマンスは向上します。この場合は、ネットワークが介在しません。しかし、複数のSQL文をまとめれば、データベースに対するコール回数が減るため、プログラムが簡単になります。

3階層モデルの場合もPL/SQLを用いることで利点が得られます。この場合、(通常はHTMLブラウザを実行する)クライアントがアプリケーションサーバーと対話し、アプリケーションサーバーはデータベースと対話します。PL/SQLのメリットが生かせるのはアプリケーションサーバーとデータベースの対話の部分です。このタイプの環境についてはChapter 2で詳しく説明します。

規格

Oracleでは、SQL言語のANSI(American National Standards Institute：米国規格協会)規格をサポートしています。このANSI規格は、ANSIのX3.135-1992『Database Language SQL』で定義されています。この規格は、一般にはSQL92(またはSQL2)として知られていますが、SQL言語しか定義していません。PL/SQLでSQLに提供している3GL拡張機能は、定義されていません。SQL92には、Entry、Intermediate、Fullという3つの準拠レベルがあります。Oracle7 R7.2以降(Oracle8、Oracle8*i*を含む)は、National Institute for Standards and Technology(NIST)によって公認されているように、EntryレベルのSQL92規格に従ってコンパイルを実行します。Oracleでは、将来のバージョンのOracleとPL/SQLがFullの規格レベルに準拠するよう、ANSIと作業を進めています。

PL/SQLの特長

PL/SQLが備えている各種の特長と機能は、例で説明するのがよいでしょう。本節では、PL/SQL言語の主な特長をいくつか説明します。

基本的な特長

本書では主にPL/SQLの高度な機能を取り上げますが、以下の項ではPL/SQLの基本機能について簡単に説明しておきます。詳しくは、製品マニュアル『Oracle 8i PL/SQLユーザーズ・ガイドおよびリファレンス』または『ORACLE8 PL/SQLプログラミング』(小社刊)を参照してください。

ブロック構造

PL/SQLの基本単位は、「ブロック」です。PL/SQLで作成するプログラムはすべてブロックで構成され、それらのブロックは互いにネストさせることができます。各ブロックでは通常、プログラム内における論理的な処理単位を実行します。そのため、タスクごとに別々のブロックになります。各ブロックの構造は、次のとおりです。

```
DECLARE
/* 宣言部 － ここには、PL/SQLの変数、型、カーソル、およびローカルなサブプログラムを記述します。*/
BEGIN
/* 実行部 － ここには、PL/SQL文およびSQL文を記述します。これがブロックのメインセクションであり、唯一の必須セクションです。*/
EXCEPTION
/* 例外処理部 － ここには、エラー処理用の文を記述します。*/
END;
```

必須のセクションは、実行部だけです。実行部は少くとも1つの実行可能文を含まなければなりません。宣言部および例外処理部は、オプションです。PL/SQLのプログラムでは、ブロックのセクションごとに機能が異なっています。

PL/SQLは、Adaという第3世代の言語がベースになっています。Adaで使用可能な構造体の多くは、PL/SQLにも見ることができます。ブロック構造も、その1つです。PL/SQLで見ることができるAdaのその他の特徴には、例外処理、プロシージャおよびファンクションを宣言する際の構文、パッケージなどがあります。

エラー処理

ブロック内の例外処理部は、ユーザープログラムで発生したランタイムエラーに対処するために使用します。エラー処理用のコードをプログラムの本体から分けることにより、プログラム自体の構造が簡単になります。たとえば、次のPL/SQLブロック内の例外処理部では、受け取ったエラーだけでなく、現在の時刻とエラーが発生したユーザーもロギングしています。

```
-- Available online as Error.sql
SQL> DECLARE
  2    v_ErrorCode NUMBER;           -- エラーコード
  3    v_ErrorMsg  VARCHAR2(200);    -- エラーのメッセージテキスト
  4    v_CurrentUser VARCHAR2(8);    -- 現在のデータベースユーザー
  5    v_Information VARCHAR2(100); -- エラーに関する情報
  6  BEGIN
  7    /* データ処理を行うコード */
  8  EXCEPTION
  9    WHEN OTHERS THEN
 10    -- 組み込みファンクションを使用して、
 11    -- ログ変数に値を指定する。
 12    v_ErrorCode := SQLCODE;
 13    v_ErrorMsg := SQLERRM;
 14    v_CurrentUser := USER;
 15    v_Information := 'Error encountered on ' ||
 16      TO_CHAR(SYSDATE) || ' by database user ' || v_CurrentUser;
 17    -- log_table表にログメッセージを挿入する。
 18    INSERT INTO log_table (code, message, info)
 19      VALUES (v_ErrorCode, v_ErrorMsg, v_Information);
 20  END;
```

注意

前述のサンプルコードをはじめ、本書で紹介しているコードはOsborne社のWebサイトで提供されています。詳細については、「はじめに」の「サンプルソースコード」を参照してください。

変数と型

PL/SQLとデータベースの間では、「変数」によって情報をやり取りします。変数とは、データの格納場所のことです。プログラムでは、変数からデータを読み取ったり、変数にデータを代入したりできます。今示した例では、**v_CurrentUser**、**v_ErrorCode**、**v_Information**がすべて変数です。変数は、ブロック内の宣言部で宣言します。

どの変数にも、特定の「型」を対応付けます。型によって、該当する変数に格納できるデータの種類を定義します。PL/SQLの変数は、次のように、データベースの列と同じ型にすることができます。

```
DECLARE
  v_StudentName  VARCHAR2(20);
  v_CurrentDate  DATE;
  v_NumberCredits NUMBER(3);
```

あるいは、次のように、別の型にすることもできます。

```
DECLARE
  v_LoopCounter BINARY_INTEGER;
  v_CurrentlyRegistered BOOLEAN;
```

PL/SQLでは、ユーザー定義の型（PL/SQL表およびレコード）もサポートしています。ユーザー定義の型を使用すれば、ユーザープログラムで操作するデータの構造を、次のようにカスタマイズできます。

```
DECLARE
  TYPE t_StudentRecord IS RECORD (
    FirstName  VARCHAR2(10),
    LastName   VARCHAR2(10),
    CurrentCredits NUMBER(3)
  );
  v_Student t_StudentRecord;
```

ループ

PL/SQLでは、各種のループをサポートしています。ループを使用すれば、一連の同じ文を繰り返して実行できます。たとえば、次のブロックでは「単純ループ」を使用して、1～50の数値を**temp_table**に挿入しています。

```
-- Available online as SimpleLoop.sql
SQL> DECLARE
  2    v_LoopCounter BINARY_INTEGER := 1;
  3  BEGIN
  4    LOOP
  5      INSERT INTO temp_table (num_col)
  6        VALUES (v_LoopCounter);
  7      v_LoopCounter := v_LoopCounter + 1;
  8      EXIT WHEN v_LoopCounter > 50;
  9    END LOOP;
 10  END;
```

別の「数値型FORループ」というループも使用できます。このループは、構文がより簡単です。このループを使用すれば、今示した例と同じ処理を次のように記述できます。

```
-- Available online as NumericLoop.sql
SQL> BEGIN
  2    FOR v_LoopCounter IN 1..50 LOOP
  3      INSERT INTO temp_table (num_col)
  4        VALUES (v_LoopCounter);
  5    END LOOP;
  6  END;
```

カーソル

「カーソル」は、データベースから(SELECT文を使って)取り出した複数の行を処理する際に使用します。カーソルを使用すれば、返された一連の行をユーザープログラムで1つずつ調べ、処理することができます。たとえば、次のブロックでは、データベース内のすべての学生に関して、名前と姓を取り出しています。

```
-- Available online as CursorLoop.sql
SQL> DECLARE
  2    v_FirstName VARCHAR2(20);
  3    v_LastName  VARCHAR2(20);
  4    -- カーソル宣言。
  5    -- 行を返すSQL文を定義する。
  6    CURSOR c_Students IS
  7      SELECT first_name, last_name
  8        FROM students;
  9  BEGIN
 10    -- カーソル処理を開始する。
 11    OPEN c_Students;
 12    LOOP
 13      -- 1行を取り出す。
 14      FETCH c_Students INTO v_FirstName, v_LastName;
 15      -- すべての行が取り出された後、ループを抜ける。
 16      EXIT WHEN c_Students%NOTFOUND;
 17      /* ここで、データを処理する。 */
```

```
 18    END LOOP;
 19    -- 処理の終了。
 20    CLOSE c_Students;
 21  END;
```

高度な機能

以下の項では、以降の章で詳しく説明する高度な機能の例をいくつか紹介します。これらの機能は、PL/SQLの基本機能に基づくものです。

プロシージャとファンクション

プロシージャとファンクション(サブプログラムと総称される)は、コンパイルしてデータベースに格納し、後続のブロックからコールすることができる特殊なタイプのPL/SQLブロックです。たとえば次の文は、DBMS_OUTPUTパッケージを使用して、指定する専攻学部の全学生の姓名をスクリーンにエコーするプロシージャ、**PrintStudents**を作成します。

```
-- Available online as part of PrintStudents.sql
SQL> CREATE OR REPLACE PROCEDURE PrintStudents(
  2    p_Major IN students.major%TYPE) AS
  3
  4    CURSOR c_Students IS
  5      SELECT first_name, last_name
  6        FROM students
  7        WHERE major = p_Major;
  8  BEGIN
  9    FOR v_StudentRec IN c_Students LOOP
 10      DBMS_OUTPUT.PUT_LINE(v_StudentRec.first_name || ' ' ||
 11                           v_StudentRec.last_name);
 12    END LOOP;
 13  END;
```

このプロシージャを作成してデータベースに格納したら、次のようなブロックを使用してこのプロシージャをコールすることができます。

```
-- Available online as part of PrintStudents.sql
SQL> BEGIN
  2    PrintStudents('Computer Science');
  3  END;
  4  /
Scott Smith
Joanne Junebug
Shay Shariatpanahy
```

注意

DBMS_OUTPUTを使用して出力を行うには、SQL＊PlusでSET SERVEROUTPUT ONコマンドを使用します(Chapter 2参照)。

パッケージ

変数や型と同様に、サブプログラムもパッケージにまとめることができます。パッケージは、仕様部と本体という2つの部分から構成されます。これら2つの部分を使って、関連性のあるオブジェクト同士をまとめてデータベースに格納することができます。たとえば、**RoomsPkg**パッケージには、**rooms**表に新しい部屋を挿入するプロシージャ、**rooms**表から部屋を削除するプロシージャが含まれています。

```
-- Available online as RoomsPkg.sql
SQL> CREATE OR REPLACE PACKAGE RoomsPkg AS
  2    PROCEDURE NewRoom(p_Building rooms.building%TYPE,
  3                      p_RoomNum rooms.room_number%TYPE,
  4                      p_NumSeats rooms.number_seats%TYPE,
  5                      p_Description rooms.description%TYPE);
  6
  7    PROCEDURE DeleteRoom(p_RoomID IN rooms.room_id%TYPE);
  8  END RoomsPkg;

SQL> CREATE OR REPLACE PACKAGE BODY RoomsPkg AS
  2    PROCEDURE NewRoom(p_Building rooms.building%TYPE,
  3                      p_RoomNum rooms.room_number%TYPE,
  4                      p_NumSeats rooms.number_seats%TYPE,
  5                      p_Description rooms.description%TYPE) IS
  6    BEGIN
  7    INSERT INTO rooms
  8      (room_id, building, room_number, number_seats, description)
  9      VALUES
 10      (room_sequence.NEXTVAL, p_Building, p_RoomNum, p_NumSeats,
 11       p_Description);
 12    END NewRoom;
 13
 14    PROCEDURE DeleteRoom(p_RoomID IN rooms.room_id%TYPE) IS
 15    BEGIN
 16      DELETE FROM rooms
 17        WHERE room_id = p_RoomID;
 18    END DeleteRoom;
 19  END RoomsPkg;
```

動的SQL

動的SQLによって、PL/SQLアプリケーションは実行時にSQL文を作成し、実行することができます。動的SQLメソッドには、PL/SQL 2.1以降で使用できるDBMS_SQLパッケージと、Oracle8*i*以降で使用可能なネイティブな動的SQLという2つのメソッドがあります。指定された表を削除する**DropTable**プロシージャは、DBMS_SQLパッケージを使用して作成します。

```
-- Available online as part of DropTable.sql
SQL> CREATE OR REPLACE PROCEDURE DropTable(p_Table IN VARCHAR2) AS
  2    v_SQLString VARCHAR2(100);
```

```
 3    v_Cursor BINARY_INTEGER;
 4    v_ReturnCode BINARY_INTEGER;
 5  BEGIN
 6    -- 入力パラメータに基づいて文字列を作成する。
 7    v_SQLString := 'DROP TABLE ' || p_Table;
 8
 9    -- カーソルをオープンする。
10    v_Cursor := DBMS_SQL.OPEN_CURSOR;
11
12    -- 文を解析して実行する。
13    DBMS_SQL.PARSE(v_Cursor, v_SQLString, DBMS_SQL.NATIVE);
14    v_ReturnCode := DBMS_SQL.EXECUTE(v_Cursor);
15
16    -- カーソルをクローズする。
17    DBMS_SQL.CLOSE_CURSOR(v_Cursor);
18  END DropTable;
```

Oracle8*i*以上では、ネイティブな動的SQLを使用して、**DropTable**を次のように記述できます。

```
-- Available online as part of DropTable.sql
CREATE OR REPLACE PROCEDURE DropTable(p_Table IN VARCHAR2) AS
  v_SQLString VARCHAR2(100);
BEGIN
  -- 入力パラメータに基づいて文字列を作成する。
  v_SQLString := 'DROP TABLE ' || p_Table;

  EXECUTE IMMEDIATE v_SQLString;
END DropTable;
```

オブジェクト型

Oracle 8 and higher

PL/SQL 8が付属したOracle8は、オブジェクト型をサポートします。オブジェクト型には属性とメソッドがあり、データベース表に格納できます。以下は、1つのオブジェクト型を作成する例です。

```
-- Available online as part of ch12/objTypes.sql
CREATE OR REPLACE TYPE Student AS OBJECT (
  ID               NUMBER(5),
  first_name       VARCHAR2(20),
  last_name        VARCHAR2(20),
  major            VARCHAR2(30),
  current_credits  NUMBER(3),

  -- スペースで区切って姓名を返す。
  MEMBER FUNCTION FormattedName
    RETURN VARCHAR2,
  PRAGMA RESTRICT_REFERENCES(FormattedName, RNDS, WNDS, RNPS, WNPS),

  -- 専攻（major）をp_NewMajorで指定された値に更新する。
```

```
  MEMBER PROCEDURE ChangeMajor(p_NewMajor IN VARCHAR2),
  PRAGMA RESTRICT_REFERENCES(ChangeMajor, RNDS, WNDS, RNPS, WNPS),

  -- 現在の値にp_CompletedClassの単位数を追加して、
  -- current_creditsを更新する。
  MEMBER PROCEDURE UpdateCredits(p_CompletedClass IN Class),
  PRAGMA RESTRICT_REFERENCES(UpdateCredits, RNDS, WNDS, RNPS, WNPS),

  -- 学生のソートに使用されるORDERファンクション。
  ORDER MEMBER FUNCTION CompareStudent(p_Student IN Student)
    RETURN NUMBER
);

CREATE OR REPLACE TYPE BODY Student AS
  MEMBER FUNCTION FormattedName
    RETURN VARCHAR2 IS
  BEGIN
    RETURN first_name || ' ' || last_name;
  END FormattedName;

  MEMBER PROCEDURE ChangeMajor(p_NewMajor IN VARCHAR2) IS
  BEGIN
    major := p_NewMajor;
  END ChangeMajor;

  MEMBER PROCEDURE UpdateCredits(p_CompletedClass IN Class) IS
  BEGIN
    current_credits := current_credits +
    p_CompletedClass.num_credits;
  END UpdateCredits;

  ORDER MEMBER FUNCTION CompareStudent(p_Student IN Student)
    RETURN NUMBER IS
  BEGIN
    -- まず姓で比較する。
    IF p_Student.last_name = SELF.last_name THEN
      -- 姓が同じ場合は、名前で比較する。
      IF p_Student.first_name < SELF.first_name THEN
        RETURN 1;
      ELSIF p_Student.first_name > SELF.first_name THEN
        RETURN -1;
      ELSE
        RETURN 0;
      END IF;
    ELSE
      IF p_Student.last_name < SELF.last_name THEN
        RETURN 1;
      ELSE
        RETURN -1;
      END IF;
```

```
    END IF;
  END CompareStudent;
END;
```

コレクション

PL/SQLコレクションは、他の3GLの配列に類似しています。PL/SQLには、索引付き表(2.0以降)、NESTED TABLE(8.0以降)、そしてVARRAY(8.0以降)という3種類のコレクションがあります。以下は、さまざまなタイプのコレクションの例です。

```
-- Available online as Collections.sql
SQL> DECLARE
  2    TYPE t_IndexBy IS TABLE OF NUMBER
  3    INDEX BY BINARY_INTEGER;
  4    TYPE t_Nested IS TABLE OF NUMBER;
  5    TYPE t_Varray IS VARRAY(10) OF NUMBER;
  6
  7    v_IndexBy t_IndexBy;
  8    v_Nested t_Nested;
  9    v_Varray t_Varray;
 10  BEGIN
 11    v_IndexBy(1) := 1;
 12    v_IndexBy(2) := 2;
 13    v_Nested := t_Nested(1, 2, 3, 4, 5);
 14    v_Varray := t_Varray(1, 2);
 15  END;
```

組み込みパッケージ

PL/SQL言語で提供される機能に加えて、Oracleではその他の機能を提供する多数の組み込みパッケージを提供しています。本書ではこれらの組み込みパッケージについて詳しく説明していきます。本書で取り上げるのは、次のような主要な組み込みパッケージです。組み込みパッケージ全般についてはAppendix Aにまとめてあります。

パッケージ名	説明
DBMS_ALERT	データベースアラート。セッション間の通信を可能にします。
DBMS_JOB	ジョブスケジューリングサービス。
DBMS_LOB	ラージオブジェクトの操作。
DBMS_PIPE	データベースパイプ。セッション間の通信を可能にします。
DBMS_SQL	動的SQL。
UTL_FILE	テキストファイル入力および出力。

一般に、DBMS_*パッケージはサーバーでのみ使用できるのに対して、UTL_*パッケージはサーバーサイドとクライアントサイドの両方のPL/SQLで使用できます。(Oracle Formsなどの一部のクライアント環境では、その他のパッケージも使用できます。)

本書の表記規則

本書では以降、いくつかの表記規則を使用します。これには、PL/SQLのバージョン間での違いを区別するためのアイコンや、Oracleのマニュアル名、そしてオンラインコードの場所などが含まれます。

PL/SQLとOracleのバージョン

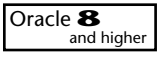
PL/SQLは、Oracleサーバーに入っています。PL/SQLの最初のバージョン(1.0)は、Oracleバージョン6と一緒にリリースされました。Oracle7には、PL/SQL 2.xが入っています。そしてOracle8とともに、PL/SQLのバージョンも8になりました。したがって、Oracle8*i*(8.1に対応)にはPL/SQLバージョン8.1が入っています。Oracleデータベースのその後の各リリースにも、対応するバージョンのPL/SQLが入っています。この状況を、各リリースで新しくサポートされた主要な機能とともに、**表1-1**に示します。

本書では、バージョン2.0～8.1のPL/SQLについて説明します。特定のリリースでしか使用できない機能については、次のようにアイコンで示します。

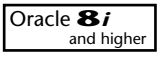
このアイコンで示す段落では、PL/SQL 2.1以降で使用可能な機能(DBMS_SQLパッケージなど)について説明します。

このアイコンで示す段落では、PL/SQL 2.2以降で使用可能な機能(カーソル変数など)について説明します。

このアイコンで示す段落では、PL/SQL 2.3以降で使用可能な機能(UTL_FILEパッケージなど)について説明します。



このアイコンで示す段落では、PL/SQL 8.0以降で使用可能な機能(オブジェクト型など)について説明します。



このアイコンで示す段落では、PL/SQL 8.1以降で使用可能な機能(ネイティブな動的SQLなど)について説明します。

適切な機能を利用できるよう、PL/SQLのリリース番号には注意してください。データベースに接続すると、最初の文字列にデータベースのバージョン番号が表示されます。次に例を示します。


接続:
```
Oracle8 Enterprise Edition Release 8.0.6.0.0 - Production
With the Objects option
PL/SQL Release 8.0.6.0.0 - Production
```

Oracleのリリース番号	PL/SQLのリリース番号	追加または変更された機能
6	1.0	（最初のバージョン）
7.0	2.0	・CHARデータ型を固定長に変更
		・ストアドサブプログラム（プロシージャ、ファンクション、パッケージ、およびトリガー）
		・ユーザー定義の複合型（表およびレコード）
		・DBMS_PIPEおよびDBMS_ALERTパッケージでのセッション間通信
		・DBMS_OUTPUTパッケージを使用したSQL*Plusまたはサーバー・マネージャでの出力
7.1	2.1	・ユーザー定義のサブタイプ
		・SQL文でのユーザー定義ファンクションの使用
		・DBMS_SQLパッケージでの動的PL/SQL
7.2	2.2	・カーソル変数
		・制約をユーザー定義できるサブタイプ
		・DBMS_JOBパッケージによるPL/SQLバッチ処理のスケジューリング
7.3	2.3	・カーソル変数の機能拡張（サーバー上でのフェッチ）
		・UTL_FILEパッケージでのファイルI/O
		・PL/SQLの表の属性と表へのレコードの格納
		・コンパイル済みトリガーの格納
8.0	8.0	・オブジェクト型とメソッド
		・コレクション型（nested tableとvarray）
		・アドバンストキューイングオプション
		・外部プロシージャ
		・LOB機能強化
8.1	8.1	・ネイティブな動的SQL
		・Java外部ルーチン
		・起動者権限
		・NOCOPYパラメータ
		・自律的トランザクション
		・一括操作

表1-1　OracleとPL/SQLのバージョンの対応状況

あるいは、次のように表示されます。

```
SQL*Plus: Release 8.1.7.0.0 - Production on 日 Apr 29 14:52:50 2001
(c) Copyright 2000 Oracle Corporation.  All rights reserved.

Oracle8i Enterprise Edition Release 8.1.7.0.0 - Production
With the Partitioning option
JServer Release 8.1.7.0.0 - Production
に接続されました。
```

この両方が、有効な初期文字列です。PL/SQLのリリース番号がデータベースのリリース番号に対応している点に注意してください。

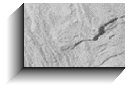
本書で示す例のほとんどは、Solarisシステム上で動作するOracle 8.0.6で作成しました。Oracle8*i*の例は、同様にSolaris上で動作するOracle8*i*バージョン8.1.5で作成しました。また、画面はすべて、サーバーのデータベースに接続したWindows 95またはWindows NT上で採取しました。

Oracleの製品マニュアル

本書中でOracleのマニュアルを参照する個所がありますが、マニュアルの名称はバージョンごとに変わっているので、本書では原則としてOracle8*i*のマニュアル名を使用しています。たとえば、『Oracle8*i* SQLリファレンス』は、Oracle 8では『Oracle8 Server SQLリファレンス』、Oracle7では『Oracle7 Server SQL言語リファレンスマニュアル』に相当します。

例で使用する表

本書で使用する例では、一連の共通のデータベースの表に対して処理を実行します。それらの表は、大学の学生管理システムを実装するためのものです。主要な表としては、**students**、**classes**、**rooms**の3つがあります。これらの表には、システムに必要なメインのエンティティが格納されています。これらの主要な表以外にも、**registered_students**という表には、受講の登録をした学生に関する情報が格納されています。では、表の詳細な構造と、表の作成に必要なSQLについて説明しましょう。



注意

これらの表はすべて、オンラインコードに含まれている**relTables.sql**スクリプトを使用して作成できます。Oracle8表は、**objTables.sql**に入っています。いずれのスクリプトもOsborne McGraw-Hill 社のWebサイト(http://www.osborne.com/oracle)からダウンロードして入手できます。ここではリレーショナル表のみを扱います。オブジェクトのバージョンについては、**objTables.sql**を参照してください。

順序

student_sequenceという順序は、studentsの主キーの一意な値を生成するために使用します。room_sequenceという順序は、roomsの主キーの一意な値を生成するために使用します。

```
CREATE SEQUENCE student_sequence
  START WITH 10000
  INCREMENT BY 1;

CREATE SEQUENCE room_sequence
  START WITH 20000
  INCREMENT BY 1;
```

students

studentsという表には、大学に在籍している学生の情報が格納されています。

```
CREATE TABLE students (
  id                NUMBER(5) PRIMARY KEY,
  first_name        VARCHAR2(20),
  last_name         VARCHAR2(20),
  major             VARCHAR2(30),
  current_credits   NUMBER(3)
  );

INSERT INTO students (id, first_name, last_name, major,
                      current_credits)
  VALUES (student_sequence.NEXTVAL, 'Scott', 'Smith',
          'Computer Science', 11);

INSERT INTO students (id, first_name, last_name, major,
                      current_credits)
  VALUES (student_sequence.NEXTVAL, 'Margaret', 'Mason',
          'History', 4);

INSERT INTO students (id, first_name, last_name, major,
                      current_credits)
  VALUES (student_sequence.NEXTVAL, 'Joanne', 'Junebug',
         'Computer Science', 8);

INSERT INTO students (id, first_name, last_name, major,
                      current_credits)
  VALUES (student_sequence.NEXTVAL, 'Manish', 'Murgratroid',
          'Economics', 8);

INSERT INTO students(id, first_name, last_name, major,
                     current_credits)
  VALUES(student_sequence.NEXTVAL, 'Patrick', 'Poll',
         'History', 4);
```

```
INSERT INTO students(id, first_name, last_name, major,
                     current_credits)
  VALUES (student_sequence.NEXTVAL, 'Timothy', 'Taller',
          'History', 4);

INSERT INTO students(id, first_name, last_name, major,
                     current_credits)
  VALUES (student_sequence.NEXTVAL, 'Barbara', 'Blues',
          'Economics', 7);

INSERT INTO students(id, first_name, last_name, major,
                     current_credits)
  VALUES (student_sequence.NEXTVAL, 'David', 'Dinsmore',
          'Music', 4);

INSERT INTO students(id, first_name, last_name, major,
                     current_credits)
  VALUES (student_sequence.NEXTVAL, 'Ester', 'Elegant',
          'Nutrition', 8);

INSERT INTO students(id, first_name, last_name, major,
                     current_credits)
  VALUES (student_sequence.NEXTVAL, 'Rose', 'Riznit',
          'Music', 7);

INSERT INTO STUDENTS(id, first_name, last_name, major,
                     current_credits)

  VALUES (student_sequence.NEXTVAL, 'Rita', 'Razmataz',
          'Nutrition', 8);

INSERT INTO students(id, first_name, last_name, major,
                     current_credits)

  VALUES (student_sequence.NEXTVAL, 'Shay', 'Shariatpanahy',
          'Computer Science', 3);
```

major_stats

major_statsという表には、各種の専攻先に関して生成した統計情報が格納されています。

```
CREATE TABLE major_stats (
  major          VARCHAR2(30),
  total_credits  NUMBER,
  total_students NUMBER);

INSERT INTO major_stats (major, total_credits, total_students)
  VALUES ('Computer Science', 22, 3);
```

```
INSERT INTO major_stats (major, total_credits, total_students)
  VALUES ('History', 12, 3);

INSERT INTO major_stats (major, total_credits, total_students)
  VALUES ('Economics', 15, 2);

INSERT INTO major_stats (major, total_credits, total_students)
  VALUES ('Music', 11, 2);

INSERT INTO major_stats (major, total_credits, total_students)
  VALUES ('Nutrition', 16, 2);
```

rooms

roomsという表には、使用可能な教室の情報が格納されています。

```
CREATE TABLE rooms (
  room_id          NUMBER(5) PRIMARY KEY,
  building         VARCHAR2(15),
  room_number      NUMBER(4),
  number_seats     NUMBER(4),
  description      VARCHAR2(50)
  );

INSERT INTO rooms (room_id, building, room_number, number_seats,
                   description)
  VALUES (room_sequence.NEXTVAL, 'Building 7', 201, 1000,
          'Large Lecture Hall');

INSERT INTO rooms (room_id, building, room_number, number_seats,
                   description)
  VALUES (room_sequence.NEXTVAL, 'Building 6', 101, 500,
          'Small Lecture Hall');

INSERT INTO rooms (room_id, building, room_number, number_seats,
                   description)
  VALUES (room_sequence.NEXTVAL, 'Building 6', 150, 50,
          'Discussion Room A');

INSERT INTO rooms (room_id, building, room_number, number_seats,
                   description)
  VALUES (room_sequence.NEXTVAL, 'Building 6', 160, 50,
          'Discussion Room B');

INSERT INTO rooms (room_id, building, room_number, number_seats,
                   description)
  VALUES (room_sequence.NEXTVAL, 'Building 6', 170, 50,
         'Discussion Room C');
```

```
INSERT INTO rooms (room_id, building, room_number, number_seats,
                   description)
  VALUES (room_sequence.NEXTVAL, 'Music Building', 100, 10,
         'Music Practice Room');

INSERT INTO rooms (room_id, building, room_number, number_seats,
                   description)
  VALUES (room_sequence.NEXTVAL, 'Music Building', 200, 1000,
          'Concert Room');

INSERT INTO rooms (room_id, building, room_number, number_seats,
                   description)
  VALUES (room_sequence.NEXTVAL, 'Building 7', 300, 75,
          'Discussion Room D');

INSERT INTO rooms (room_id, building, room_number, number_seats,
                   description)
  VALUES (room_sequence.NEXTVAL, 'Building 7', 310, 50,
          'Discussion Room E');
```

classes

classesという表には、学生が受講できる講座の情報が記述されています。

```
CREATE TABLE classes (
  department       CHAR(3),
  course           NUMBER(3),
  description      VARCHAR2(2000),
  max_students     NUMBER(3),
  current_students NUMBER(3),
  num_credits      NUMBER(1),
  room_id          NUMBER(5),
  CONSTRAINT classes_department_course
    PRIMARY KEY (department, course),
  CONSTRAINT classes_room_id
    FOREIGN KEY (room_id) REFERENCES rooms (room_id)
  );

INSERT INTO classes(department, course, description, max_students,
                    current_students, num_credits, room_id)
  VALUES ('HIS', 101, 'History 101', 30, 11, 4, 20000);

INSERT INTO classes(department, course, description, max_students,
                    current_students, num_credits, room_id)
  VALUES ('HIS', 301, 'History 301', 30, 0, 4, 20004);

INSERT INTO classes(department, course, description, max_students,
                    current_students, num_credits, room_id)
  VALUES ('CS', 101, 'Computer Science 101', 50, 0, 4, 20001);
```

```
INSERT INTO classes(department, course, description, max_students,
                    current_students, num_credits, room_id)
  VALUES ('ECN', 203, 'Economics 203', 15, 0, 3, 20002);

INSERT INTO classes(department, course, description, max_students,
                    current_students, num_credits, room_id)
  VALUES ('CS', 102, 'Computer Science 102', 35, 3, 4, 20003);

INSERT INTO classes(department, course, description, max_students,
                    current_students, num_credits, room_id)
  VALUES ('MUS', 410, 'Music 410', 5, 4, 3, 20005);

INSERT INTO classes(department, course, description, max_students,
                    current_students, num_credits, room_id)
  VALUES ('ECN', 101, 'Economics 101', 50, 0, 4, 20007);

INSERT INTO classes(department, course, description, max_students,
                    current_students, num_credits, room_id)
  VALUES ('NUT', 307, 'Nutrition 307', 20, 2, 4, 20008);

INSERT INTO classes(department, course, description, max_students,
                    current_students, num_credits, room_id)
  VALUES ('MUS', 100, 'Music 100', 100, 0, 3, NULL);
```

registered_students

registered_studentsという表には、学生が現在受講している講座に関する情報が格納されています。

```
CREATE TABLE registered_students (
  student_id NUMBER(5) NOT NULL,
  department CHAR(3)   NOT NULL,
  course     NUMBER(3) NOT NULL,
  grade      CHAR(1),
  CONSTRAINT rs_grade
    CHECK (grade IN ('A', 'B', 'C', 'D', 'E')),
  CONSTRAINT rs_student_id
    FOREIGN KEY (student_id) REFERENCES students (id),
  CONSTRAINT rs_department_course
    FOREIGN KEY (department, course)
    REFERENCES classes (department, course)
  );

INSERT INTO registered_students (student_id, department, course, grade)
  VALUES (10000, 'CS', 102, 'A');

INSERT INTO registered_students (student_id, department, course, grade)
  VALUES (10002, 'CS', 102, 'B');
```

```
INSERT INTO registered_students (student_id, department, course, grade)
  VALUES (10003, 'CS', 102, 'C');

INSERT INTO registered_students (student_id, department, course, grade)
  VALUES (10000, 'HIS', 101, 'A');

INSERT INTO registered_students (student_id, department, course, grade)
  VALUES (10001, 'HIS', 101, 'B');

INSERT INTO registered_students (student_id, department, course, grade)
  VALUES (10002, 'HIS', 101, 'B');

INSERT INTO registered_students (student_id, department, course, grade)
  VALUES (10003, 'HIS', 101, 'A');

INSERT INTO registered_students (student_id, department, course, grade)
  VALUES (10004, 'HIS', 101, 'C');

INSERT INTO registered_students (student_id, department, course, grade)
  VALUES (10005, 'HIS', 101, 'C');

INSERT INTO registered_students (student_id, department, course, grade)
  VALUES (10006, 'HIS', 101, 'E');

INSERT INTO registered_students (student_id, department, course, grade)
  VALUES (10007, 'HIS', 101, 'B');

INSERT INTO registered_students (student_id, department, course, grade)
  VALUES (10008, 'HIS', 101, 'A');

INSERT INTO registered_students (student_id, department, course, grade)
  VALUES (10009, 'HIS', 101, 'D');

INSERT INTO registered_students (student_id, department, course, grade)
  VALUES (10010, 'HIS', 101, 'A');

INSERT INTO registered_students (student_id, department, course, grade)
  VALUES (10008, 'NUT', 307, 'A');

INSERT INTO registered_students (student_id, department, course, grade)
  VALUES (10010, 'NUT', 307, 'A');

INSERT INTO registered_students (student_id, department, course, grade)
  VALUES (10009, 'MUS', 410, 'B');

INSERT INTO registered_students (student_id, department, course, grade)
  VALUES (10006, 'MUS', 410, 'E');

INSERT INTO registered_students (student_id, department, course,
                                 grade)
```

```
  VALUES (10011, 'MUS', 410, 'B');

INSERT INTO registered_students (student_id, department, course,
                                 grade)
  VALUES (10000, 'MUS', 410, 'B');
```

RS_audit

RS_auditという表は、**registered_students**表への変更を記録するために使用します。

```
CREATE TABLE RS_audit (
  change_type    CHAR(1)      NOT NULL,
  changed_by     VARCHAR2(8) NOT NULL,
  timestamp      DATE         NOT NULL,
  old_student_id NUMBER(5),
  old_department CHAR(3),
  old_course     NUMBER(3),
  old_grade      CHAR(1),
  new_student_id NUMBER(5),
  new_department CHAR(3),
  new_course     NUMBER(3),
  new_grade      CHAR(1)
  );
```

log_table

log_tableという表は、Oracleのエラーを記録するために使用します。

```
CREATE TABLE log_table (
  code             NUMBER,
  message          VARCHAR2(200),
  info             VARCHAR2(100)
  );
```

temp_table

temp_tableという表は、他の情報に特に関係のない一時的なデータの格納に使用します。

```
CREATE TABLE temp_table (
  num_col    NUMBER,
  char_col   VARCHAR2(60)
  );
```

connect_audit

connect_auditという表は、Chapter 6の例で、データベースとの接続と切断を記録するのに使用します。

```
CREATE TABLE connect_audit (
  user_name  VARCHAR2(30),
  operation  VARCHAR2(30),
  timestamp  DATE);
```

debug_table

debug_tableという表は、デバッグパッケージでPL/SQLのデバッグ情報を格納するのに使用します(デバッグパッケージは、Chapter 3で作成します)。

```
CREATE TABLE debug_table (
  linecount  NUMBER,
  debug_str  VARCHAR2(100)
  );
```

sourceとdestination

sourceと**destination**という表は、Chapter 3のデバッグの例を示すときに使用します。

```
CREATE TABLE source (
  key NUMBER(5),
  value VARCHAR2(50)  );

CREATE TABLE destination (
  key NUMBER(5),
  value NUMBER);
```

まとめ

この章では、PL/SQL言語の目的や主な機能も含め、PL/SQLの概要について幅広く説明しました。また、PL/SQLとデータベースのバージョンの重要性および対応状況についても説明しました。そして、章の末尾では、本書の例で使用する表を紹介しました。次に続く2つの章では、PL/SQLの開発、デバッグ、実行環境について説明します。