報道各位

2019年2月15日
株式会社インプレスR&D
https://nextpublishing.jp/

PostgreSQLを賢く使う！マニュアルの前に読む本！

『わたしとぼくのPL/pgSQL』発行

技術書典シリーズ、2月の新刊

インプレスグループで電子出版事業を手がける株式会社インプレスR&Dは、『わたしとぼくのPL/pgSQL』(著者:目黒 聖)を発行いたしました。

『技術書典シリーズ』とは、今もっとも注目すべき、エンジニアによるアウトプットの場である技術同人誌イベント「技術書典」で、頒布された同人誌を底本として、商業書籍として刊行する書籍シリーズです。

『わたしとぼくのPL/pgSQL』
https://nextpublishing.jp/isbn/9784844398271

著者:目黒 聖
小売希望価格:電子書籍版 1600円(税別)／印刷書籍版 1800円(税別)
電子書籍版フォーマット:EPUB3／Kindle Format8
印刷書籍版仕様:B5判／カラー／本文64ページ
ISBN:978-4-8443-9827-1
発行:インプレスR&D

＜＜発行主旨・内容紹介＞＞

【マニュアルを読む前に読む、PL/pgSQL基本解説書】

本書は、PostgreSQLで一連のデータベース操作をユーザー関数として扱うことができる「ストアド・ファンクション」を作成するための言語、「PL/pgSQL」の基本的な文法を解説したものです。

OracleのPL/SQLで書かれたストアド・ファンクションをPostgreSQLのPL/pgSQLに書き換えるための注意点を説明しています。これからPL/pgSQLを使ってみようという人が、「マニュアルを読む前に読んでみる本」です。

〈本書の対象読者〉

・Linux で PostgreSQL の操作がある程度できる

・プログラミングの経験がある

・SQL を書いた経験がある

・PL/pgSQL はあまり詳しくない

（本書は、次世代出版メソッド「NextPublishing」を使用し、出版されています。）

PL/pgSQL の基礎を丁寧に解説

第2章　PL/pgSQLの基礎

本章ではPL/pgSQLの利点や基本的な文法について解説します。PL/pgSQLで記述するメリットは何か、PL/pgSQLはどのような特徴があるのかを理解しましょう。

2.1　PL/pgSQLの利点

まずPL/pgSQLの利点を簡単に説明します。通常のSQLとは何が異なるのか、PL/pgSQLを使用することでどのようなメリットが存在するのかを理解することで、PL/pgSQLを使用したほうがいい場合とそうでない場合の判断がつきやすくなります。

2.1.1　手続き処理が実行できる

第1章「はじめてのPL/pgSQL」でも少し記述しましたが、PL/pgSQLでは単純なSQLでは不可能、または記述が煩雑になってしまう複雑な処理を実行できます。例えば「あるデータを検索し、そのデータをもとにレコードを作成し、別テーブルに追加する」というような一連の処理が可能です

2.1.2　移植性に優れている

違うOSでも、PostgreSQLが動く環境であればPL/pgSQLのプログラムを修正することなく実行できます[1]。

2.1.3　パフォーマンスに優れている場合がある

例えば、「あるデータを検索し、そのデータをもとにレコードを作成し、別テーブルに追加する」というような一連の処理をアプリケーションで実行しようとした場合を考えます。

上手くSQLが書けなければ、検索と追加の2回、SQLがアプリケーションからPostgreSQLに送信され、結果も2回返ることになります。これが2回程度では問題ないかもしれませんが、10回、20回とSQLを実行する場合、送受信で使用するネットワークのリソースやDBとの接続のコストが無視できなくなってくるでしょう。

PL/pgSQLの場合、手続き的な処理はすべてPostgreSQL内部で実行され、呼び出すときにはSQLをひとつ実行するだけです。途中の結果を受信する必要もなく、負荷は下がることが予想できます[2]。

2.2　無名コードブロック

PL/pgSQLの利点がわかったところで、基本的なPL/pgSQLの文法の説明に入る前に、「無名コードブロック」について説明します。

無名コードブロックとは、一時的な「名無しの」関数のことです。構文はリスト2.1の通りです。

リスト2.1: 無名コードブロックの構文

```
DO [ LANGUAGE lang_name ] 関数本文
```

「関数本文」には、これから説明するPL/pgSQLの文法に従った関数を記述します。「1.3 はじめての「Hello PL/pgSQL」」で紹介した例文が、無名コードブロックの例です。

PL/pgSQLを使って関数を書く際は無名コードブロックを使用できますが、無名コードブロックで作成した関数はあくまでもその場限りの名無しの関数です。他のセッションはもちろん、同じセッションでも再利用することができません。

さらに、次のような制限が存在します。

・引数が指定できない

・戻り値がない

引数が指定できませんので、動的に変わるパラメータは指定できません。場合によって変わるパラメータが存在するのであれば、無名コードブロックを記述したときに、その場合に応じたパラメータを直接記述する必要があります。

また、戻り値がないため、この関数の結果を他のSQLで使用することができません。

無名コードブロックはこのような制限があるため、使用する場面が限られます。文法を確認するだけなら無名コードブロックでもいいのですが、PL/pgSQLで書いた関数を他のセッションやSQLで再利用する場合はCREATE FUNCTIONというSQLを使用し、関数を作成します。

2.3　CREATE FUNCTION

次に、PL/pgSQLを使用するためには切っても切り離せないCREATE FUNCTIONについて説明します。

マニュアルから抜粋したCREATE FUNCTIONの構文はリスト2.2の通りです。

リスト2.2: CREATE FUNCTIONの構文

```
CREATE [ OR REPLACE ] FUNCTION
    name ( [ [ argmode ] [ argname ] argtype
        [ { DEFAULT | = } default_expr ] [, ...] ] )
    [ RETURNS rettype
      | RETURNS TABLE ( column_name column_type [, ...] ) ]
  { LANGUAGE lang_name
    | TRANSFORM { FOR TYPE type_name } [, ... ]
    | WINDOW
    | IMMUTABLE | STABLE | VOLATILE | [ NOT ] LEAKPROOF
    | CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
    | [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER
```

1 もちろん、PostgreSQLから他のDBMSに移行する、という場合には移行先の環境に合わせた修正が必要となります。

2 ただし、当然PostgreSQLの処理のオーバーヘッドが増加しますので、すべてをPL/pgSQLにすればよいというものではありません。

変数、定数、制御構造、例外処理などをそれぞれの章で解説

第6章　例外処理

例外とは、簡単に言うとエラーのことです。適切に例外処理を行わない場合、関数はエラーを吐いて終了してしまいます。

6.1　例外の発生

では、適当な例外を発生させてみましょう。第5章「テーブルからデータを取り出す」で使用したテーブルに列を追加して、表6.1を作成しました。これを操作して例外を発生させてみます。

表6.1: 都道府県の面積(平方km)テーブル

code(char型、主キー)	name(text型)	jinko(numeric型)	area(integer型)
13	東京都	13000000	2191
14	神奈川県	9000000	2416
27	大阪府	8000000	1905
23	愛知県	7500000	5172
11	埼玉県	7300000	3798
12	千葉県	6300000	0

リスト6.1: 例外発生

```
CREATE OR REPLACE FUNCTION func6_1()
RETURNS void AS $$
  DECLARE
    mitsudo numeric;
    menseki_cursor CURSOR FOR SELECT * FROM MENSEKI ORDER BY code;
  BEGIN
    FOR var_rec IN menseki_cursor LOOP
      mitsudo := ROUND(var_rec.jinko / var_rec.area, 2);
      RAISE NOTICE '都道府県名=%,人口密度=%', var_rec.name, mitsudo;
    END LOOP;
    RETURN;
  END;
$$ LANGUAGE plpgsql;
```

リスト6.1は面積テーブルからデータを都道府県コード順に取り出し、人口を面積で割って人口密度を標準出力に出力する関数です。しかし、データの誤りで千葉県の面積が0になっていました。このまま実行すると、次のようなエラーメッセージが表示されます。

```
postgres=# select func6_1();
NOTICE: 都道府県名=埼玉県,人口密度=1922.06
ERROR: division by zero
CONTEXT: PL/pgSQL function func6_1() line 7 at assignment
```

都道府県コード順に処理するので、一番コードが小さい埼玉県は実行されますが、次の千葉県の処理で0除算エラーが発生します。そして、それ以降のレコードは処理されません。

実際に運用して使用する場合、例外が発生してしまうことで全てが停止してしまうのは避ける必要があります。適切に例外を捕捉し、例外が発生した場合は例外が発生した時のための処理を行いましょう。

6.2　例外の捕捉

「2.5.1 PL/pgSQLのブロック構造」で、「EXCEPTIONからEND;までは例外処理部」という説明をしました。ここではより詳しく、例外処理部の記述方法を説明します。

まずは、実際にリスト6.1を修正し、例外を捕捉してみましょう。

リスト6.2: 例外の捕捉

```
 1: CREATE OR REPLACE FUNCTION func6_2()
 2: RETURNS void AS $$
 3:   DECLARE
 4:     mitsudo numeric;
 5:     menseki_cursor CURSOR FOR SELECT * FROM MENSEKI ORDER BY code;
 6:   BEGIN
 7:     FOR var_rec IN menseki_cursor LOOP
 8:       BEGIN -- 副ブロック開始
 9:         mitsudo := ROUND(var_rec.jinko / var_rec.area, 2);
10:         RAISE NOTICE '都道府県名=%,人口密度=%', var_rec.name, mitsudo;
11:       EXCEPTION
12:         WHEN division_by_zero THEN
13:           RAISE NOTICE '都道府県名[%]の処理中に0除算が行われました。', var_rec.name;
14:         WHEN others THEN
15:           RAISE NOTICE 'その他の例外が発生しました。';
16:       END; -- 副ブロック終了
17:     END LOOP;
18:     RETURN;
19:   END;
20: $$ LANGUAGE plpgsql;
```

リスト6.2では、リスト6.1にはないループ内の副ブロックが存在します。そして副ブロックには、「EXCEPTION」ブロックが存在します。「EXCEPTION」から「END;」までは例外処理部であり、「BEGIN」

新バージョンの PostgreSQL 11 で実装されたストアド・プロシージャを付録で解説

付録A　ストアド・プロシージャ

本書で対象としていたPostgreSQLのバージョンは9.6でした。その次のメジャーバージョンアップであるPostgreSQL 10では、PL/pgSQLとストアド・ファンクションにはあまり変化はありませんでした。

しかし、2018年10月にリリースされたPostgreSQL 11では、ストアド・プロシージャが実装され、そのためにPL/pgSQLの機能も追加されました。本章は付録として、PostgreSQL 11で強化された点を紹介します。

A.1　ストアド・プロシージャの作り方

PostgreSQL 10までは、プロシージャではなくファンクションしかありませんでした。両者の違いを端的に表すと、「戻り値があるかないか」です。プロシージャには戻り値がなく、ファンクションには戻り値があります。

そのため、これまでPostgreSQLでプロシージャのようなものを実装するには、CREATE FUNCTIONでRETURNSをvoidで作成し、戻り値を返さないファンクションを作る必要がありました。

リストA.1: 戻り値がないファンクション

```
CREATE OR REPLACE FUNCTION func()
RETURNS void AS $$
  BEGIN
    RAISE NOTICE 'Hello PL/pgSQL';
    RETURN;
  END;
$$ LANGUAGE plpgsql;
```

リストA.1のような関数をストアド・プロシージャとして作成する場合、リストA.2のように記述します。

リストA.2: プロシージャの例

```
CREATE OR REPLACE PROCEDURE proc()
AS $$
  BEGIN
    RAISE NOTICE 'Hello PL/pgSQL';
  END;
$$ LANGUAGE plpgsql;
```

構文はCREATE FUNCTIONと非常に似ています。リストA.2では指定していませんが、引数もCREATE FUNCTIONと同様の書き方で指定可能です[1]。

異なる点は、まずCREATE PROCEDUREであることです。また、そもそも戻り値が存在しないため、RETURNSは存在しません。関数本文中にRETURNも不要です。

それ以外は本書で扱ったCREATE FUNCTIONの構文と同じです。

PL/pgSQLに関して言えば、戻り値がないこと以外は基本的にPL/pgSQLの文法はそのまま使用できます。

A.2　CALL

ストアド・プロシージャの実行方法は、ファンクションとは大きく異なります。CREATE FUNCTIONで作成するユーザー定義関数は、あくまでもSQLの関数として作成されたものなので、呼び出すときはSELECTなどのSQLから呼び出す必要がありました。

しかし、プロシージャはSQLの関数ではありません。これも新しくPostgreSQL 11で追加された、CALLで実行します。

リストA.3: プロシージャの実行

```
CALL proc();
```

リストA.2を呼び出すための記述は、リストA.3だけです。

A.3　トランザクション管理

そして、トランザクション管理が可能になった点が、ストアド・プロシージャの大きな強化点です。

今までストアド・ファンクションではトランザクションの開始・終了ができませんでした[2]。しかし、ストアド・プロシージャではコミットもロールバックもできるようになりました[3]。

リストA.4: プロシージャ内でコミットする例

```
CREATE OR REPLACE PROCEDURE proc_trun(integer)
AS $$
  DECLARE
    max_num ALIAS FOR $1;
  BEGIN
    FOR i IN 1 .. max_num LOOP
      INSERT INTO test (num) VALUES (i);
      IF i % 100 = 0 THEN
        COMMIT; --100回に1回コミット
        RAISE NOTICE 'コミットしました';
      END IF;
```

1. ただし、戻り値はないため引数のモードにOUTを指定することはできません。しかし、INOUTは指定可能です。
2. SQL の関数というものから考えればは当然なのですが、Oracle からの移行という点ではお客様に不便でした。
3. ちなみに、無名コードブロックでもトランザクション制御は可能です。

＜＜目次＞＞

＜＜著者紹介＞＞

目黒 聖(めぐろ たかし)
某都内SI勤務のエンジニア。もともと開発で必要だったためにDBを学び始めたはずなのに、いつのまにか開発から離れDBAに。趣味で居合をやりながら、PostgreSQLで面白いことができないか、日々考えています。
WEB:http://www.maguronomizo.net/
Twitter:@tameguro

＜＜販売ストア＞＞

電子書籍:
Amazon Kindleストア、楽天koboイーブックストア、Apple Books、紀伊國屋書店 Kinoppy、Google Play Store、honto電子書籍ストア、Sony Reader Store、BookLive!、BOOK☆WALKER
印刷書籍:
Amazon.co.jp、三省堂書店オンデマンド、hontoネットストア、楽天ブックス
※ 各ストアでの販売は準備が整いしだい開始されます。
※ 全国の一般書店からもご注文いただけます。

【株式会社インプレスR&D】 https://nextpublishing.jp/
株式会社インプレスR&D（本社：東京都千代田区、代表取締役社長：井芹昌信）は、デジタルファーストの次世代型電子出版プラットフォーム「NextPublishing」を運営する企業です。また自らも、NextPublishingを使った「インターネット白書」の出版などIT関連メディア事業を展開しています。
※NextPublishingは、インプレスR&Dが開発した電子出版プラットフォーム(またはメソッド)の名称です。電子書籍と印刷書籍の同時制作、プリント・オンデマンド(POD)による品切れ解消などの伝統的出版の課題を解決しています。これにより、伝統的出版では経済的に困難な多品種少部数の出版を可能にし、優秀な個人や組織が持つ多様な知の流通を目指しています。

【インプレスグループ】 https://www.impressholdings.com/
株式会社インプレスホールディングス(本社:東京都千代田区、代表取締役:唐島夏生、証券コード:東証1部9479)を持株会社とするメディアグループ。「IT」「音楽」「デザイン」「山岳・自然」「旅・鉄道」「学術・理工学」を主要テーマに専門性の高いメデイア&サービスおよびソリューション事業を展開しています。さらに、コンテンツビジネスのプラットフォーム開発・運営も手がけています。

【お問い合わせ先】
株式会社インプレス R&D NextPublishing センター
TEL 03-6837-4820
電子メール: np-info@impress.co.jp