PGECons
PostgreSQL Enterprise Consortium

PostgreSQL エンタープライズ・コンソーシアム 技術部会 WG#2

ストアドプロシージャ移行調査編

製作者
担当企業名 クオリカ株式会社

改訂履歴

版	改訂日	変更内容
1.0	2013/03/25	新規作成

はじめに

■本資料の目的

本資料は、異種 DBMS から PostgreSQL へストアドプロシージャを移行する作業の難易度およびボリュームの事前判断と、実際に書き換えを行う際の参考資料として利用されることを想定しています。

■本資料で記載する範囲

本資料では、移行元の異種 DBMS として Oracle Database を想定し、PostgreSQL へストアドプロシージャを移行する際に書き換えが必要である箇所とその書き換え方針について手続き言語を中心に記載します。スキーマ、SQL、組み込み関数については本資料では取り扱っていません。これらに関しては、それぞれ「スキーマ移行調査編」、「SQL 移行調査編」、「組み込み関数移行調査編」を参照してください。

■本資料で扱う用語の定義

資料で記述する用語について以下に定義します。

表 1: 用語定義

No.	用語	意味
1	DBMS	データベース管理システムを指します。ここでは、PostgreSQL および異種 DBMS の総称として利用します。
2	異種 DBMS	PostgreSQL ではない、データベース管理システムを指します。 本資料では、Oracle Database が該当します。
3	Oracle	データベース管理システムの Oracle Database を指します。

■本資料で扱う DBMS およびツール

本書では以下の DBMS を前提にした調査結果を記載します。

表 2: 本書で扱う DBMS

DBMS 名称	バージョン
PostgreSQL	9.2.1
Oracle Database	11gR2 11.2.0.2.0

目次

1. PostgreSQL のストアドプロシージャについて

データベースに対する一連の処理手順をまとめて RDBMS 内に格納する、「ストアドプロシージャ」について PostgreSQL における特徴を紹介します。

1.1. PostgreSQL におけるストアドプロシージャ

PostgreSQL ではストアドプロシージャはユーザ定義関数（FUNCTION）として定義を行います。
実行方法は、ほかの関数と同様に SQL 文で利用します。
関数として実装するため呼び出し方法も SQL 文の中で関数として利用することになります。
処理ロジックの記述には、C や Perl などでも処理ロジックを組み込むことも可能です。
PostgreSQL 専用の手続き言語として PL/pgSQL が用意されています。
上記以外に、Cや Perl などでも処理ロジックを組み込むことも可能です。

1.2. PL/pgSQL について

PL/pgSQL は、Oracle の PL/SQL と同様に SQL に制御構造（条件分岐や LOOP 処理）などを組み込んだ、PostgreSQL で標準として実装されている手続き言語です。
記述された処理ロジックは、ユーザ定義関数としてデータベースに格納する事が出来ますが、事前にコンパイルはされずに、実行時に解釈され実行されます。

2. Oracle から PostgreSQL への移行（定義関連）

2.1. CREATE FUNCTION 文

CREATE FUNCTION 文の比較

Oracle	PostgreSQL(処理ロジックを PL/pgSQL で記述する場合)

CREATE OR REPLACE FUNCTION
proc_f (p1 IN NUNBER)
RETURN VARCHAR2(10)
IS
変数名　データ型;
BEGIN
処理内容;
END;
	CREATE OR REPLACE FUNCTION
	proc_f (p1 IN INTEGER)
	RETURNS VARCHAR(10)
	AS
	$$
	DECLARE
	変数名　データ型;
	BEGIN
	処理内容;
	END;
	$$
	LANGUAGE plpgsql;

PostgreSQL では処理内容の記述部分（変数宣言と BEGIN から END まで）を文字列定数として作成される必要があります。
そのためにドル引用符付け（$$）を使って処理記述の範囲を囲います。
単一引用符で範囲を囲む方法も可能ですが、この場合には関数の本体部分で使用される単一引用符(')とバックスラッシュ(¥)は二重にする必要があります。

処理内容の記述に使用している言語の指定が必須で、LANGUAGE 句で指定します。

変数宣言部に DECLARE が必須ですので追加する必要があります。

引数を持たない FUNCTION を作成するとき、には Oracle では”()”を省略できますが、PostgreSQL では”()”の記述が必須です。

上記以外では
RETURN → RETURNS
IS → AS
に書き換える必要があります。

2.2. CREATE PROCEDURE 文

PostgreSQL には PROCEDURE は実装されていません。
FUNCTION で代用する事になります。

2.3. CREATE PACKAGE 文

POCEDURE と同様に PACKAGE は実装されていません。
FUNCTION で代用することになります。
PACKAGE レベルで共通使用する定数などは、一時テーブルに保存するなどの方法を検討する必要があります。

2.4. ALTER FUNCTION 文

Oracle と PostgreSQL では互換性がありません。

Oracle では再コンパイルに関する命令になります。
PostgreSQL では関数名の変更、所有者の変更などの FUNCTION が保持している情報を変更する命令になります。

2.5. DROP FUNCTION 文

DROP FUNCTION 文の比較

Oracle	PostgreSQL
DROP FUNCTION proc_f;	DROP FUNCTION proc_f (p1 IN INTEGER);

PostgreSQL では、引き渡しパラメータも含めて指定する必要があります。

3. Oracle から PostgreSQL への移行（標準手続き言語関連）

Oracle と PostgreSQL にそれぞれ実装されている手続き言語である、PL/SQL と PL/pgSQL における記述の相違を中心に書換え方法を記述します。

3.1. 構造

構造のステートメントには相違ありません。

```
DECLARE
  A1  CHAR(10);
BEGIN
  A1 := 'ABC';
```

```
  RETURN A1;
END;
```

「DECLARE 部」で変数の宣言
「BEGIN 部」で処理内容の記述
「END」でブロックの終了

3.2. コメント

コメントの記述には相違ありません。

-- コメント記述　　:行末までをコメントとします。
/* コメント記述 */ :/* から */ までのブロック（複数行でも可）をコメントとします。

3.3. データ型

PostgreSQL で使用可能なデータ型は PL/pgSQL で使用できます。
データ型の変換については別ドキュメント「組み込みデータ型対応表（Oracle-PostgreSQL）」を参照して変換してください。
同様に%ROWTYPE 型や%TYPE はそのまま使用できます。

RECORD 型については注意が必要です。

Oracle	PostgreSQL
type rec_name is RECORD (C1 CHAR(10), C2 CHAR(4));	rec_name RECORD;

PL/pgSQL では RECORD 型の宣言時にはレコードの内容は記述しません。
レコードの内容は直接 SELECT 文を記述したり、カーソルの FETCH で使用されると定義が確定されます。
例1.　SELECT の結果をレコード型にストアする
　　rec_name IN SELECT C1, C2 FROM tb1
例2.　カーソル cu の結果をレコード型にストアする
　　fetch cu into rec_name

3.4. 変数の宣言

プログラム内で使用する変数は必ず宣言部に記述して宣言を行う必要があります。
但し、例外として FOR ループで使用するループ変数はこの限りではありません。

例外の名前の宣言は PL/pgSQL では宣言する事が出来ません。
RAISE 文を使ってエラーを発生させます。

3.5. 制御構造

3.5.1. LOOP 命令

LOOP の記述には相違ありません。

```
LOOP
  A1 := A1 + 1;
  EXIT WHEN A1 > 100;
```

```
END LOOP;
```

「LOOP」と「END LOOP」の間に記述された命令を繰り返し実行します。
LOOP を抜けるためには EXIT を使用します。
EXIT に続けて LOOP を抜ける条件式を記述します。
EXIT のみでは無条件で LOOP から抜けます。

3.5.2. WHILE 命令

WHILE の記述には相違ありません。

```
WHILE A1 < 10 LOOP
   A1 := A1 + 1;
END LOOP;
```

「WHILE」と「LOOP」の間に繰り返しの条件式を記述し、
「END LOOP」の間に繰り返す命令を記述します。
条件式を満たす前に LOOP を抜けるためには EXIT を使用します。

3.5.3. FOR 命令

FOR の記述には相違ありません。

```
FOR A1 IN 1 .. 10 LOOP
   D1 := D1 + 1;
END LOOP;
```

IN の後に記述した最小値から最大値までの間、「LOOP」から「END LOOP」に記述された命令を繰り返し実行します。

但し、「REVERSE」を使って値を最大値から最小値までを行う場合には書換えが必要です。

Oracle	PostgreSQL

FOR A1 IN REVERSE 1 .. 10 LOOP
  D1 := D1 + 1;
END LOOP;
	FOR A1 IN REVERSE 10 .. 1 LOOP
	  D1 := D1 + 1;
	END LOOP;

最大値と最小値の値の指定が逆になります。

3.5.4. EXIT 命令

EXIT の記述には相違ありません。

```
EXIT;
EXIT [ label ] ;
EXIT WHEN A1 > 10;
```

Label が指定されない場合には最も内側の LOOP を終わらせます。
Label の指定がある場合には指定されたラベルのループを抜けます。
WHEN が指定された場合には、条件式を満たしていれば EXIT を実行します。

3.5.5. CONTINUE 命令

CONTINUE の記述には相違ありません。

```
CONTINUE;
CONTINUE [ label ] ;
CONTINUE  WHEN A1 > 10;
```

Label が指定されない場合には実行している LOOP の先頭に戻り次の反復に制御を移します。
Label の指定がある場合には指定されたラベルの先頭に戻り次の反復に制御を移します。
WHEN が指定された場合には、条件式を満たしていれば CONTINUE を実行します。

3.5.6. IF 命令

IF については注意が必要です。

Oracle	PostgreSQL
IF A1 = 0　THEN A1 := 10 　ELSIF A1 > 0 THEN A1 := A1 + 1 　ELSE A1 := 1 END IF;	IF A1 = 0　THEN A1 := 10 　ELSEIF A1 > 0 THEN A1 := A1 + 1 　ELSE A1 := 1 END IF;

IF のあとの比較条件式に対して真もしくは偽を判断して THEN もしくは ELSE の後に記述された命令が実行されます。
Oracle では「ELSIF」の記述が PostgreSQL では「ELSEIF」に変わります。
この部分以外での相違はありません。

3.5.7. CASE 命令

CASE の記述には相違ありません。

```
CASE A1
 WHEN  1, 2 THEN
      A1 :=  A1 + 10
 ELSE
      A1 :=  A1 + 1
END CASE;
```

WHEN 句内の値と比較を行い一致すれば指定された命令が実行されます。
全ての WHEN を順番に評価した後一致するものがない場合、ELSE の命令を実行します。
一致する WHEN がなく ELSE の記述が無い場合には、CASE_NOT_FOUND 例外が発生します

3.5.8. GOTO 命令

PostgreSQL には GOTO 命令がありません。

Oracle	PostgreSQL
GOTO label;	対応する命令なし

置換える命令がありません。
無条件に指定したラベルに制御を移すことは出来ません。

3.6. カーソル

3.6.1. カーソルの宣言

カーソルの宣言については注意が必要です。

Oracle	PostgreSQL
CURSOR C1 IS SELECT C1 FROM foo;	C1 CURSOR FOR SELECT C1 FROM foo;

どちらも宣言は DECLARE 部で行いますが、文法が違います。
FOR の部分は IS で記述されていても文法エラーにはなりません。

3.6.2. カーソルの OPEN

カーソルの OPEN の記述には相違ありません。

```
OPEN C1;
```

宣言をおこなったカーソルから行を取り出すために、OPEN によりカーソルを開きます。

3.6.3. カーソルのFETCH

カーソルの FETCH の記述には相違ありません。

```
FETCH C1 INTO wk_c1;
```

カーソルから行を 1 行づつ取り出して変数に格納します。

3.6.4. カーソルの終了判定

カーソルをすべて FETCH したときの判定方法は注意が必要です。

Oracle	PostgreSQL
C1%NOTFOUND;	NOTFOUND;

Oracle では、カーソル名を明示して終了判定（NOTFOUND）しますが、PostgeSQL ではカーソル名の指定はできません。

3.6.5. カーソルの更新

カーソルのカレント行に対する更新の記述には相違ありません。

```
UPDATE foo SET C1 = '0' WHERE CURRENT OF C1;
DELETE FROM foo WHERE CURRENT OF C1;
```

カーソルの宣言時に FOR UPDATE を使って作成したカーソルの現在行に対して項目の値の変更およびレコードの削除を行います。

3.6.6. カーソルの CLOSE

カーソルの CLOSE の記述には相違ありません。

```
CLOSE C1%;
```

OPEN したカーソルを閉じます。

3.7. エラーハンドリング

3.7.1. EXCEPTION 文

EXCEPTION の記述には相違ありません。

```
EXCEPTION
   WHEN condition1 THEN  handler_statements1
   WHEN condition2 THEN  handler_statements2
   WHEN OTHERS     THEN  handler_statements3
END;
```

WHEN の後に記述された例外の内容と合致したときに THEN の後に記述された処理を行います。
指定された例外以外が発生したときは、呼び出し元にエラー情報が伝搬します。

例外に設定されている名前に相違があるものは個別に書換えが必要です。
以下は例外の一部についての対比をまとめましたので、参考にしてください。

Oracle の例外名	PostgreSQL の例外名	相違
CASE_NOT_FOUND	CASE_NOT_FOUND	同じ
INVALID_CURSOR	INVALID_CURSOR_STATE	書換え必要
NO_DATA_FOUND	NO_DATA_FOUND	同じ
STORAGE_ERROR	OUT_OF_MEMORY	書換え必要
TOO_MANY_ROWS	TOO_MANY_ROWS	同じ
ZERO_DIVIDE	DIVISION_BY_ZERO	書換え必要

なお、PostgreSQL のエラーコードに対する例外名はマニュアルの付録に記載があるので参考にしてください。
http://www.postgresql.jp/document/9.2/html/errcodes-appendix.html#ERRCODES-TABLE

3.7.2. RAISE 文

RAISE の記述には相違ありません。

```
RAISE exception;
```

事前定義の例外を明示的に呼び出します。
但し、Oracle では宣言部で例外の名前を宣言して、RAISE で例外を呼び出せますが、PostgreSQL では宣言部での名前の宣言が出来ないので、RAISE 文で例外を記述する事になります。

4. Oracle から PostgreSQL への移行（その他）

4.1. 起動方法

実行方法については注意が必要です。

Oracle	PostgreSQL

BEGIN
  EXECUTE stoad_name
END;
	SELECT func_namae();

PostgreSQL では、ストアドファンクション（関数）として登録していますので SELECT 文を使って呼び出します。

4.2. トランザクション制御

PostgreSQLno ストアドファンクションは、外部トランザクションの一部として実行されますので、処理中に COMMIT を実行できません。
Oracle では「PRAGMA AUTONOMOUS_TRANSACTION」を使って呼び出し元とトランザクションを分離する事が出来ますが、PostgreSQL にはこのような機能はありません。

EXCEPTION で例外の発生が判断された時は、BIGIN 以降のすべてのデータベースに対する更新処理が自動的にロールバックします。

5. Oracle から PostgreSQL への移行に関するまとめ

SQL レベルであったり手続き言語の構文については、ある程度単純な置換え作業は可能と思われます。
しかし PostgreSQL ではファンクション（関数）としてのみ登録が可能で、Oracle のパッケージの考え方がなく、複雑なバッチ処理に必要なトランザクション制御も実装できない状況を考えると、単純に移行が出来るストアドプロシージャは限られてくるものと思われます。
処理の内容によっては、ストアドプロシージャは移行するよりも、他の言語で実装する方が容易になる可能性もあります。

Oracle のストアドプロシージャでは、ユーティリティパッケージ（DBMS_OUTPUT や UTL_FILE）が、よく使用されていますが、これらは Oracle が提供しているものなので当然 PostgreSQL には実装されていないのでつかえません。
DBMS_OUTPUT は同様の機能として RAISE NOTICE で代用できますが構文が違うので個別での対応が必要と思われます。

参考ですが Orafce ではユーティリティーパッケージの一部の実装を実現しています。
但し、仕様的に Oracle との違いがありますので注意が必要です。
例）DBMS_OUTPU の通知のタイミング
Oracle　トランザクションの終了時
Orafce　送信都度

著者

版	所属企業・団体名	部署名	氏名
ストアドプロシージャ移行調査編 第 1 版 （2012 年度 WG2）	クオリカ株式会社	開発センター	坂本　浩行