PostgreSQLからPerlを使う

樋口千洋

最初に

- 講演者はPL/Perlの熟練者ではありません
  - これから使おうと志している者です
  - 粗相があるかと思いますが一緒に勉強していければ幸いです

PerlからPostgreSQLを使う

- PostgreSQL
  - BSDライセンスのリレーショナルデータベース
  - 現在の最新版は8.1.3
- PerlからPostgreSQLを使う(またはその逆)
  - 双方の長所を活かして柔軟性が向上する
  - 呼び出し方
    - PostgreSQLのPgライブラリ
    - DBD/DBI
    - PL/Perl

PL/Perlとストアードプロシジャ

- PL/Perl
  - PostgreSQLで提供されるストアードプロシジャ
- ストアードプロシジャ
  - データベースへの処理手順をまとめたもの
  - オーバヘッドの低減
- 標準で提供されるストアードプロシジャ
  - PL/PgSQL、PL/Tcl、PL/Perl、PL/Python
- 第三者が提供するストアードプロシジャ
  - PL/Ruby、PL/R

PL/Perlを使う理由

- SQLでの記述力の限界を補う
  - 選択値をリスト処理したい
- 効率性の追求
  - バックエンドとのオーバヘッド回避
- 保守性の向上
  - 見通しがよくなる

PL/Perlのインストール

- Fedora Core 4
  - Perlが入っている(必須？)
  - PostgreSQLが入っている(当たり前)
  - postgresql-pl-8.0.7-1.FC4.1
- ソースから
  - % ./configure --with-perl …
  - % sudo make install
- シェルプロンプトから
  - % create language plperl(u) <データベース名>

PL/Perlの文法

- CREATE FUNCTION *funcname* (*argument-types*) RETURNS *return-type*
  AS $$
    # PL/Perl関数本体
  $$ LANGUAGE plperl(u);

変数の受け渡し

- 引数は@_に格納される
- 返り値はスカラ
  - リストを返すことはできない
    - 8.1以前
    - 8.1以降

例1

```
CREATE FUNCTION perl_max (integer, integer) RETURNS integer AS $$
   if ($_[0] > $_[1]) { return $_[0]; }
      return $_[1];
$$ LANGUAGE plperl;
```

```
CREATE FUNCTION perl_max (integer, integer) RETURNS integer AS $$
    my ($x,$y) = @_;
    if (! defined $x) {
       if (! defined $y) { return undef; }
    return $y; }
    if (! defined $y) {return $x; }
    if ($x > $y) { return $x; }
    return $y;
$$ LANGUAGE plperl;
```

例2

```
CREATE OR REPLACE FUNCTION perl_set_int(int) RETURNS SETOF INTEGER
AS $$
    return [0..$_[0]];
$$ LANGUAGE plperl;

SELECT * FROM perl_set_int(5);

CREATE OR REPLACE FUNCTION perl_set() RETURNS SETOF testrowperl
AS $$
    return [
        { f1 => 1, f2 => 'Hello', f3 => 'World' },
        { f1 => 2, f2 => 'Hello', f3 => 'PostgreSQL' },
        { f1 => 3, f2 => 'Hello', f3 => 'PL/Perl' }
    ];
$$ LANGUAGE plperl;

SELECT * FROM perl_set();
```

例3

```
CREATE TABLE test (i int, v varchar);
INSERT INTO test (i, v) VALUES (1, 'first line');
INSERT INTO test (i, v) VALUES (2, 'second line');
INSERT INTO test (i, v) VALUES (3, 'third line');
INSERT INTO test (i, v) VALUES (4, 'immortal');
CREATE OR REPLACE FUNCTION test_munge() RETURNS SETOF test AS $$
    my $rv = spi_exec_query('select i, v from test;');
    my $status = $rv->{status};
    my $nrows = $rv->{processed};
    foreach my $rn (0 .. $nrows - 1) {
        my $row = $rv->{rows}[$rn];
        $row->{i} += 200 if defined($row->{i});
        $row->{v} =~ tr/A-Za-z/a-zA-Z/ if (defined($row->{v}));
        return_next($row);
    }
    return undef;
$$ LANGUAGE plperl
SELECT * FROM test_munge();;
```

例4

```
CREATE TYPE foo_type AS (the_num INTEGER, the_text TEXT);
CREATE OR REPLACE FUNCTION lotsa_md5 (INTEGER) RETURNS SETOF foo_type AS $$
    use Digest::MD5 qw(md5_hex);
    my $file = '/usr/share/dict/words';
    my $t = localtime;
    elog(NOTICE, "opening file $file at $t" );
    open my $fh, '<', $file # ooh, it's a file access!
        or elog(ERROR, "Can't open $file for reading: $!");
    my @words = <$fh>;
    close $fh;
    $t = localtime;
    elog(NOTICE, "closed file $file at $t");
    chomp(@words);
    my $row;
    my $sth = spi_query("SELECT * FROM generate_series(1,$_[0]) AS b(a)");
    while (defined ($row = spi_fetchrow($sth))) {
        return_next({
            the_num => $row->{a},
            the_text => md5_hex($words[rand @words])
        });
    }
    return;
$$ LANGUAGE plperlu;
SELECT * from lotsa_md5(500);
```

謝辞

- 日本PostgreSQLユーザ会
- PostgreSQL文書・書籍関連分科会