PL/php - Feature #4979

Add support for IN/OUT parameters

12/05/2005 07:37 AM - Álvaro Herrera

Status:	In Progress	**Start date:**	
Priority:	Normal	**Due date:**	09/13/2011
Assignee:	Alexey Klyukin	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:	Release 1.5		
Resolution:			

Description

Add support for OUT parameters so that the user can skip specifying a return type explicitely.

History

#1 - 12/05/2005 06:01 PM - anonymous -

I think this can push to 1.3 or even 1.4 as long as we get solid SRF.

#2 - 01/16/2008 08:16 AM - Álvaro Herrera

- Status changed from New to In Progress

#3 - 03/12/2010 08:38 AM - Álvaro Herrera

- Assignee deleted (Álvaro Herrera)

#4 - 06/22/2010 02:43 PM - Álvaro Herrera

- Target version changed from Release 1.3 to Release 1.4

#5 - 07/12/2010 04:46 PM - Álvaro Herrera

- Status changed from In Progress to New

- Target version changed from Release 1.4 to Release 1.5

#6 - 07/27/2010 12:24 PM - Joshua Drake

- Assignee set to Álvaro Herrera

#7 - 08/12/2010 10:35 AM - Álvaro Herrera

1. at compile time, scan &argmodes (from get_func_arg_info) and find out what params are INOUT or INOUT (plpgsql does this in do_compile)
2. if more than one, build a rowtype that holds all the out params (follow out_param_varno in plpgsql code); this is cached in the compiled-function struct.
3. in plphp_spi.c, ZEND_FUNCTION(return_next) needs fixed to work with the cached rowtype
4. plphp_srf_handler probably needs fixed as well, but perhaps this just works, because return_next stuffs the result tuples in the tuple store already.

Note: get_func_arg_info doesn't exist in PostgreSQL 8.1. We'll probably drop support for that release when implementing this.

Note 2: some of the code we use for handling parameter names and stuff predates get_func_arg_info. Since the way we're doing it duplicates the new function, we'll need to rework that as well.

#8 - 05/21/2011 04:50 PM - Álvaro Herrera

- Priority changed from Low to Normal

#9 - 05/21/2011 04:50 PM - Álvaro Herrera

- Assignee changed from Álvaro Herrera to Alexey Klyukin

#10 - 09/06/2011 01:14 PM - Alexey Klyukin

- Due date set to 09/07/2011

- Status changed from New to In Progress

So, long time no updates, but we are pretty close to complete the implementation. My original idea was to model it after PL/pgSQL, as Alvaro suggested earlier, but I found out that building a row type for the result is not necessary, since OUT arguments are represented as a RECORD return type internally, and we already have support for returning records. Thus, adding OUT arguments appeared to be quite straightforward. For a single OUT, it's sufficient to add a 'return $argument_name' statement at the very end of the function during the compilation stage. For multiple OUTs, the approach is the same, but instead of returning a scalar value, an array of references to OUT arguments is created and returned. For instance, here's the function declaration as supplied by a user, and the resulting code just before the compilation:

```
CREATE OR REPLACE FUNCTION plphp_multiple_out(OUT integer, OUT name text, INOUT date) AS
$$
  $args[0] = 1; $args[1] = 'jdoe';
$$ LANGUAGE plphp;
SELECT * FROM plphp_multiple_out(current_date);
```

```
function plphp_proc_24616($args, $argc){ $name = &$args[1];  $_plphp_ret_plphp_proc_24616 = array(&$args[0],&$
args[1],&$args[2]);
      $args[0] = 1; $args[1] = 'jdoe';
    ; return $_plphp_ret_plphp_proc_24616;}
```

(A minor improvement I've just discovered is to return the array(...) statement itself, without creating a name for it).

Initially I have decided to skip the OUT arguments altogether when building an argument list for the function, but this appeared to be incompatible with unnamed OUT arguments (we need to refer to them somehow), so I've used the idea from PL/pgSQL to initialize them with NULL values initially.

I also had to modify the code that builds a postgres tuple from the PHP array to support the normal non-assosiative arrays (i.e. array(1,2,3) instead of array('one'=>1, 'two'=>2, 'three'=>3), so it's not necessary to specify argument names when returning multiple values a function.

What's left is the support for TABLE arguments (aka SRF OUT arguments) and test cases. I'd think that it needs an extra day of work.

#11 - 09/08/2011 02:15 PM - Alexey Klyukin

- Due date changed from 09/07/2011 to 09/09/2011

So, I've got the SRF functions with OUT arguments working, but came over one problem when dealing when TABLE arguments (which are basically a syntax sugar over the SRF with OUTs). The pl/pgSQL implementation of TABLE arguments allows for doing RETURN NEXT w/o specifying the row to return (in fact, it explicitly forbids putting the row there) and automatically fills the return value with a row constructed from the TABLE arguments. I'd like to do this in PHP as well, but since we don't have such intimate relationship with a PHP parser as pl/pgSQL enjoys, we use a separate function called return_next. This function, naturally, doesn't have access to the variables passed into the original function, but we need to get their values somehow to construct the resulting tuple. The approach I'm currently exploring is to save the symbol table of the pl/PHP function into a global variable right before passing control to the PHP parser, and accessing this table from return_next.

#12 - 09/12/2011 09:48 AM - Alexey Klyukin

- Due date changed from 09/09/2011 to 09/13/2011

Alexey Klyukin wrote:

> This function, naturally, doesn't have access to the variables passed into the original function, but we need to get their values somehow to construct the resulting tuple. The approach I'm currently exploring is to save the symbol table of the pl/PHP function into a global variable right before passing control to the PHP parser, and accessing this table from return_next.

This idea was not working, because we never passed the symbol table prepared just before the actual function call to the PHP function itself. Instead, that symbol table represented an outer block for the function, and we passed the argc and argv parameters via the function signature (by assigning values to argc and argv in our outer block (C function) and then calling plphp_proc_$oid(argc, argv)). By examining the PHP sources I've discovered that we can pass our prepared symbol table directly to the function via the last parameter of call_user_function_ex. I've checked this and it appears to be working. So the last problem seems to be solved, but I need a bit more testing (and some refactoring) to finish with this.

#13 - 09/14/2011 07:16 AM - Alexey Klyukin

Note that the changes will affect argument numbers in functions with OUT arguments. For example (this is actually a test case that failed with the new code due to assumption about argument numbers no longer correct):

```
create function retset5(out int, out int, int, int)
language plphp as $$
    for ($i = 1; $i <= $args[0]; $i++) {
        return_next($args[1] * $i);
    }
$$;
select * from retset5(5, 2);
```

The code assumes that args [0] is the 3rd argument, while it's actually the 1st one. We don't skip OUT arguments in args, doing this would make

impossible to assign values to unnamed OUT arguments. This is really a problem of this test only, since there were no support for OUT arguments in the past. Nevertheless, I think it's worth mentioning here as a reminder to put a notice into the documentation.

#14 - 09/14/2011 11:21 AM - Álvaro Herrera

Excerpts from plphp-tickets's message of mié sep 14 11:16:51 -0300 2011:

> Issue #4979 has been updated by Alexey Klyukin.
>
> Note that the changes will affect argument numbers in functions with OUT arguments. For example (this is actually a test case that failed with the new code due to assumption about argument numbers no longer correct):
>
> ```
> > create function retset5(out int, out int, int, int)
> > language plphp as $$
> >     for ($i = 1; $i <= $args[0]; $i++) {
> >         return_next($args[1] * $i);
> >     }
> > $$;
> > select * from retset5(5, 2);
> >
> ```
>
> The code assumes that args [0] is the 3rd argument, while it's actually the 1st one. We don't skip OUT arguments in args, doing this would make impossible to assign values to unnamed OUT arguments. This is really a problem of this test only, since there were no support for OUT arguments in the past. Nevertheless, I think it's worth mentioning here as a reminder to put a notice into the documentation.

Keep in mind that the new OUT handling code has never been released. I
don't think it's important to mention this caveat.

--
Álvaro Herrera <alvherre@commandprompt.com>
The PostgreSQL Company - Command Prompt, Inc.
PostgreSQL Replication, Consulting, Custom Development, 24x7 support