PL/pgSQL Example

CREATE FUNCTION add10( int, int ) RETURNS int AS 'BEGIN
  RETURN $1 + $2 + 10;
END;
' LANGUAGE plpgsql;

PL/pgSQL General Syntax

CREATE [OR REPLACE] FUNCTION func( parameter_types )
RETRUN return_type AS 'DECLARE
  declarations
BEGIN
  statements
END;
' LANGUAGE plpgsql

pgNOTES

- No "create procedure"
- create or replace
  - no "drop if exists" though
- Parameter and return types
  - SQL types
  - row and record
  - set
  - void
- Single quotes

Comments

- SQL-style: --
- C-style: /\*

  -- this is an one line comment
  return $1 + $2; -- return sum of the two params

  /\*
  multiple-line
  comment
  */

Errors and Messages

RAISE INFO "Calling cs_create_job(\%\%", v_job_id;
RAISE EXCEPTION "Inexistent ID \%", user_id;
## Declarations

Syntax:

```
name type [ (default | :=) expression ];
name alias for $n
```

Examples:

```
id int default 0;
nname varchar(20) := "cysun";
op1 alias for $1;
op2 alias for $2;
```

## Composite Variable Types

- **Row type:**
  - name table%ROWTYPE;
- **Record type:**
  - name RECORD;

## Composite Return Types

- **RECORD**
- **Table type**
- **User defined type**

```
CREATE TYPE name AS ( attr_name attr_type [...] );
```

## Statement – Assignment

```
identifier := expression ;
```

## Statement – SELECT INTO

```
SELECT INTO target col(s) FROM ... WHERE;
```

- **target** is of row or record type, or a list of variable names
  - takes the value of the first row returned
  - null if no row is returned

## Statement – RETURN

```
RETURN expression;
RETURN NEXT expression;
RETURN;
```

- **RETURN statement is required even for functions that return void**
- **RETURN NEXT builds up the result set, and does not exit the function**
Cursors

name CURSOR [ ( arguments ) ] FOR query;

Examples:
curs1 refcursor;
curs2 CURSOR FOR SELECT * FROM table1;
curs3 CURSOR (k integer) FOR SELECT * FROM table1 WHERE key=k;

Opening and Closing Cursors

◆ Bound and unbound cursors

OPEN curs1 FOR SELECT * FROM table2;
OPEN curs2;
OPEN curs3(47);
CLOSE curs1;
CLOSE curs2;
CLOSE curs3;

Fetching From a Cursor

FETCH cursor INTO target;

Examples:
FETCH curs1 INTO record1;
FETCH curs2 INTO v1, v2, v3;
◆ Special variable FOUND

Statement – Conditionals

IF boolean-expression THEN
  statements
END IF;
IF boolean-expression THEN
  statements
ELSE
  statements
END IF;
IF boolean-expression THEN
  statements
ELSIF boolean-expression THEN
  statements
ELSE
  statements
END IF;

Statement – Simple Loops

LOOP
  statements
END LOOP;
WHILE expression LOOP
  statements
END LOOP;
EXIT [ WHEN expression ];

Statement – FOR (Integer)

FOR name IN [ REVERSE ] expression .. expression LOOP
  statements
END LOOP;

Examples:
FOR i IN 1..10 LOOP
  -- something here
END LOOP;
FOR i IN REVERSE 10..1..1 LOOP
  -- something here
END LOOP;
Statement – FOR (Query)

FOR record_or_rowtype IN query LOOP
  statements
END LOOP;

Create Triggers

CREATE TRIGGER name
  { BEFORE | AFTER } { event [ OR ... ] }
  ON table [ FOR [ EACH ] { ROW | STATEMENT } ]
  EXECUTE PROCEDURE funcname ( arguments )

About Triggers

- Event
  - insert, update, delete
- A trigger is associated with a table
- The trigger can be executed
  - once per statement, or
  - once per row

About Firing Triggers

- Multiple triggers are fired in alphabetic order by trigger name
- No restrictions on cascading triggers

About Trigger Procedures

- Must be defined before the trigger is created
- Must be declared as no parameter and return a trigger type
- Return
  - NULL, or
  - the tuple to be modified

Trigger Procedure in PL/pgSQL

- Special variables
  - NEW, OLD
  - TG_NAME, TG_WHEN, TG_LEVEL, TG_OP
  - TG_RELID, TG_RELNAME
  - TG_NARGS, TG_ARGV[]
Trigger Examples

- `fk_insert_trig`
- `emp_stamp`
  - [http://www.postgresql.org/docs/7.4/static/plpgsql-trigger.html](http://www.postgresql.org/docs/7.4/static/plpgsql-trigger.html)