ecpg - PostgreSQL Embedded SQL/C Precompiler

General

- PostgreSQL Version 7.4.8 includes ecpg version 3.1.1
- PostgreSQL Version 8.2.5 includes ecpg version 4.2.1
- supports full ANSI-SQL standard
- ecpg does some SQL syntax checking - some statements which show no syntax problems on precompile/compile, may yield a syntax error when executed
- “ecpg –version” command displays version info
- ecpg is case sensitive
- uses “EXEC SQL ...” syntax
- character strings in SQL statements are denoted by single quotes (‘)
    - Infx allowed single or double quotes
    - character strings in C statements still use double quotes (“)
- normal file extension for ecpg file is .pgc
- Momjian, Douglas and Schonig books all have short chapters on ecpg

Informix Compatibility Mode

(" -C INFORMIX" option for ecpg)

- allows many of the datetime related functions and data types to be used
    - note that function dttofmtasc does not work and must be replaced by dttoasc
- currently used by HSEB software
- in the future, will not use Informix Compatibility Mode

AUTO_COMMIT Option

(" -t " option for ecpg)

- if "-t" option (auto_commit = on) is not specified and a SELECT statement fails, all subsequent SELECT statements will fail
- EXEC SQL INSERT example below needs to be followed by COMMIT
    - if the A-t@ option is specified, then COMMIT is not needed
    - same for UPDATE and DELETE

Opening and Closing a Database

In ecpg, to open a connection to a database

EXEC SQL CONNECT TO dbname;

To close the connection

EXEC SQL DISCONNECT dbname;

If the application exits without first closing the database connection, the following message will be written to the postgres log:

unexpected EOF on client connection

Error Codes

- ecpg has sqlca structure containing error return code
- error codes in sqlca.sqlcode are different bet Infx and psql
    - ecpg returns error code = 0 if there is no error and error code = -400 for most errors
    - ecpg does not return error code -239 for violation of unique constraint
    - ecpg doc recommends dropping use of slqca.sqlcode field and replacing it with sqlca.sqlstate (5 char string)
        - sqlca.sqlstate='00000' signifies no error
- sqlca structure does not contain an ISAM error - ecpg has no concept of an ISAM error
- in Infx, sqlca.sqlerrd[1] contains the table defined serial primary key, psql does not return it
    - in psql, you can get the current serial value, run it through the nextval function and then insert a new record

Cursors

Infx has EXEC SQL CLOSE and EXEC SQL FREE statements to close a cursor and free its resources. psql has the EXEC SQL CLOSE statement which according to the doc closes the cursor and frees all resources related to it. psql has no FREE statement.

Informix allows a cursor to be declared and opened in one routine and fetched in a different routine. PostgreSQL does not allow this. A "cursor undefined" error is generated by the precompiler in PostgreSQL.

Infx allows reuse of a cursor within a loop without closing it. Executing the EXEC SQL OPEN, closes the previous use of the cursor AND then reopens it. In psql, the cursor must be closed before reusing in a loop.

Text Field

The variable to fetch (or select) into must be of type char * and the variable must be set to NULL initially so that memory for the contents can be allocated. If you do not set the variable to NULL, you will overwrite random areas of memory.

Descriptor Area

- Infx: EXEC SQL DESCRIBE sid INTO sqlda;
    - sid is the prepared statement "SELECT * FROM tablename;"
    - the statement is never executed
    - the descriptor area is used to get at the field descriptions

- psql has not implemented a DESCRIBE statement
    - need to execute a FETCH for a record into the descriptor area and then access the individual fields of the record
    - does not work for empty tables
    - there is email chatter about implementing a DESCRIBE statement in the

future

PostgreSQL Key Words

The following is a list of column and table names from the IHFS db which are postgres or SQL standard "key words":

action
admin
date
level
location
name
notify
number
owner
source
state
storage
type
value

Using these key words in SQL statements may cause problems as noted below.

Informix esql statements using the key word "value" such as

```
EXEC SQL SELECT   value    FROM …
```

will generate a syntax error and must be rewritten as

```
EXEC SQL SELECT  "value"  FROM …
```

Note that if the above statement is written using single quotes as in

```
EXEC SQL SELECT 'value' FROM …
```

there is no syntax error. However, a runtime error of -206 is generated.

Note also that if "value" is used in a prepared statement, the double quotes (") are not necessary.

The statement

```
EXEC SQL SELECT  county  INTO  :county  FROM  Location …
```

compiles and executes properly without quoting the key word "location".

The statement

EXEC SQL SELECT state INTO :state FROM Location …

compiles and executes properly without quoting the key words "state" and "location".

The statement

EXEC SQL SELECT focalpoint INTO :fp FROM Admin …

compiles and executes properly without quoting the key word "admin".

The statement

EXEC SQL SELECT level INTO :level FROM RiverStat …

compiles and executes properly without quoting the key word "level".

According to the documentation, quoting a key word makes it case-sensitive. Unquoted key words in postgres are folded into lower case. Note that folding key words into lower case is against the SQL standard. Appendix C of the postgres online documentation contains the full list of key words.

Other differences between esqlc and ecpg

- esqlc is a script which does both precompile and compile
- ecpg is an executable which does a precompile only
    - must explicitly execute gcc to compile code
    - see Momjian p 195 for sample statements for compiling code
    - in AWIPS OB6, the executable is located in the /usr/local/pgsql/.bin dir
    - NOTE: AWIPS will be using gcc version 3.2.3 in OB6 and version 3.4.3 in OB7

- esql allows smallint in DECLARE section
- ecpg does not allow use of smallint in DECLARE section - must use short

- ecpg 4.2.1 (postgres V8.2.x) has "WHERE CURRENT OF … " clause for cursors

- esql has rsetnull/risnull routines
- ecpg (V7.4) has ECPGset_informix_null/ECPGis_informix_null routines
- ecpg (V8.2.x) replaced ECPGset_informix_null/ECPGis_informix_null routines with rsetnull/risnull

- esql has a datatype called "string" which automatically strips off trailing blanks and adds a NULL terminator - used by generated code for char and varchar
- ecpg has no such datatype

- ecpg: if an application exits without first closing the database connection, the following

message will be written to the postgres log:

unexpected EOF on client connection

- esql: if an application exits without first closing the database connection, no error is generated

esql Function Name	ecpg Function Name
dtcurrent	PGTYPEStimestamp_current
dtcvasc	PGTYPES_from_asc

Code Fragments

The following code fragments illustrate other differences between ecpg and esql/c:

The following code snippet illustrates a bug in ecpg 3.1.1. In the first snippet, an SQL statement is declared for a cursor. The subsequent OPEN statement causes the app to core dump. In the second snippet, a PREPARE/DECLARE/OPEN using the same SQL statement succeeds.

```
/--- DECLARE/OPEN - fails ---------------------------/

EXEC SQL DECLARE pp24_cursor CURSOR WITH HOLD FOR
      select lid, ts, dur, obstime, "value", pe, extremum
      from rawpp
      where (pe = 'PP' and
      (obstime between :datetime_in and :datetime_out) and
      "value" >= 0.0 and
      (dur = :dur1 or dur = :dur2) and
      ts like 'R%');

EXEC SQL OPEN pp24_cursor; /* core dump at this line */


  / --- PREPARE/DECLARE/OPEN - works ------------/

strcpy(ts_str,"R%");

sprintf(query_stmt,"select lid, ts, dur, obstime, value, pe, extremum
from rawpp");

sprintf(query_stmt,"%s where (value >= 0.0 and (obstime between '%s'
and '%s')",query_stmt, datetime_in,datetime_out);

sprintf(query_stmt,"%s and (dur=%s or dur=%s) and ts like '%s')",
query_stmt,dur1,dur2,ts_str);

EXEC SQL PREPARE pp_q from :query_stmt;
EXEC SQL DECLARE pp24_cursor CURSOR WITH HOLD FOR pp_q;
EXEC SQL OPEN pp24_cursor;
```

---\

The following code

```
EXEC SQL UPDATE Lightning
          set ( x_hgrid,     y_hgrid,     obstime,     no_of_strike )
          = (:db_x_hgrid, :db_y_hgrid, :db_obstime, :db_no_strike)
          where ................... ;
```

precompiles, compiles and links without error or warning. However, when an attempt is made to execute, a syntax error is written to the postgres log. The code should be rewritten as

```
EXEC SQL UPDATE Lightning
  SET x_hgrid = :db_x_hgrid,
        y_hgrid = :db_y_hgrid,
              .
              .
              .
```

```
The following code caused compiler errors related to the use of
variable names "add" and "user":

$char   rrfc[9];
$char   user[32];
$char   add[150];
$char   fname[30];

The above was fixed by using the "BEGIN DECLARE" and "END DECLARE"
notation:

EXEC SQL BEGIN DECLARE SECTION;
char rrfc[9];
char user[32];
char add[150];
char fname[30];
EXEC SQL END DECLARE SECTION;

"user" and "add" appear in the extensive list of PostgreSQL reserved
words found in Appendix C of the documentation.
```

Following code worked properly in Informix/esql

```
sprintf(sql,"DELETE FROM …");    -- string variable with name "sql"
EXEC  SQL  EXECUTE  IMMEDIATE  :sql;
```

In PostgreSQL, above code results in a sqlstate = 02000 error. This is because "sql" is a

reserved word. Changing the above to

```
sprintf(sql_stmt,"DELETE FROM …");
EXEC  SQL  EXECUTE  IMMEDIATE  :sql_stmt;
```

results in statements executing without error.