Khoa Công Nghệ Thông Tin
Trường Đại Học Cần Thơ

Giao diện lập trình PostgreSQL

Đỗ Thanh Nghị
dtnghi@cit.ctu.edu.vn

Cần Thơ
24-04-2005

Nội dung

- Ngôn ngữ PL/pgSQL
- Giao diện lập trình C
- Giao diện lập trình C++
- Giao diện lập trình JAVA
- Giao diện lập trình PHP

PostgreSQL, lập trình cả Frontend và Backend

- Frontend:
  - Những ngôn ngữ truy cập dữ liệu từ bên ngoài
  - Các scripts và những ứng dụng
  - Libpq, Ecpg (C), Libpq++, Libpqxx (C++), Jdbc (Java), Odbc, Perl, Python, Pgtclsh (Tcl/Tk), Php
  - .NET ?
  - Qt (C++)
- Backend:
  - Những ngôn ngữ giúp mở rộng tính năng của server
  - Pl/pgSQL, Pl/Perl, Pl/Tcl, Pl/Python, C

- **Ngôn ngữ PL/pgSQL**
- Giao diện lập trình C
- Giao diện lập trình C++
- Giao diện lập trình JAVA
- Giao diện lập trình PHP

Giới thiệu về PL/pgSQL

- PL/pgSQL
  - So sánh được với PL/SQL của Oracle
  - Hỗ trợ định nghĩa: hàm, triggers
  - Đưa cấu trúc điều khiển đến với ngôn ngữ SQL
  - Hỗ trợ cho các tính toán phức tạp
  - Sử dụng được tất cả kiểu, phép toán, hàm sẵn có của PostgreSQL
  - Nhóm dãy lệnh SQL trong một lời gọi thủ tục sẽ giảm được chi phí nối kết client-server
  - Dễ sử dụng, portable trên PostgreSQL

Cài đặt ngôn ngữ PL/pgSQL cho cơ sở dữ liệu

- Sử dụng **createlang**
  - Cài đặt ngôn ngữ PL/pgSQL cho cơ sở dữ liệu **test**
  - **/usr/bin/createlang plpgsql test**
- Sử dụng **psql**
  - **/usr/bin/psql test**

```
test=# CREATE FUNCTION plpgsql_call_handler()
test-# RETURNS OPAQUE AS '/usr/lib/pgsql/plpgsql.so'
test-# LANGUAGE 'C';

test=#  CREATE LANGUAGE 'plpgsql' HANDLER
   plpgsql_call_handler
test-#  LANCOMPILER 'PL/pgSQL';
```

Cấu trúc khối của PL/pgSQL

```
[ <<label>> ]
[ DECLARE
    declarations ]
BEGIN
    statements
END;
```

- Chú thích chương trình
  - -- chú thích 1 dòng
  - /* chú thích 1 đoạn */

```
CREATE FUNCTION somefunc()
RETURNS integer AS
 ’
DECLARE
    quantity integer;
BEGIN
    -- foo example
    quantity := 50*50;
RETURN quantity;
END;
‘ LANGUAGE ‘plpgsql’;
```

Khai báo trong PL/pgSQL

- Khai báo biến
  - Sử dụng tất cả các kiểu dữ liệu của SQL
  - name [CONSTANT] type [NOT NULL]
    [{ DEFAULT | := } expression];
  - Ví dụ:

```
quantity integer DEFAULT 32;
url varchar := 'http://mysite.com';
user_id CONSTANT integer := 10;
myrow table2name%ROWTYPE; -- tablename%ROWTYPE
myfield users.user_id%TYPE; -- tablename.columnname%TYPE
arow RECORD;
```

Khai báo trong PL/pgSQL

- Alias của những tham số hàm
  - Hàm sẽ có những tham số với những định danh : $1, $2, … $n
  - Khai báo Alias cho những định danh: name ALIAS FOR $n;
  - Hoặc CREATE FUNCTION fun_name(var_i type_i, …)
  - Ví dụ:

```
CREATE FUNCTION instr(varchar, integer) RETURNS integer AS
‘
DECLARE
    v_string ALIAS FOR $1;
    index ALIAS FOR $2;
BEGIN
    …
END;
‘ LANGUAGE ‘plpgsql’;
```

Lệnh cơ bản trong PL/pgSQL

- Gán, identifier := expression;
  - Ví dụ: user_id := 1072; tax := subtotal*0.01;
- SELECT INTO target select_expressions FROM ...;
  - target có thể là biến record, row hoặc biến đơn
  - Ví dụ: SELECT INTO users_rec * FROM users
    WHERE user_id=3;
- Thực hiện phép truy vấn nhưng bỏ đi kết quả trả về
  - Chỉ cần thay thế SELECT bằng PERFORM trong câu truy vấn
- Lệnh NULL
  - Không làm gì

Lệnh cơ bản trong PL/pgSQL

- Thực thi lệnh động: EXECUTE command_string;
  - Lệnh thao tác trên bảng và kiểu dữ liệu khác nhau ở từng thời điểm thực thi
  - Ví dụ:

```
-- strings concatenation, str1 || str2
EXECUTE 'UPDATE tbl SET '
        || quote_ident(colname)
        || ' = '
        || quote_literal(newvalue)
        || ' WHERE key = '
        || quote_literal(keyvalue);
```

- GET DIAGNOSTICS variable = item [, ...] ;
  - Lấy trạng thái kết quả: ROWCOUNT, RESULT_OID, OID

Lệnh cơ bản trong PL/pgSQL

- Những câu lệnh như: SELECT INTO, PERFORM, UPDATE, INSERT, DELETE, FOR, FETCH
  - Lấy trạng thái kết quả: FOUND
  - Ví dụ:

```
SELECT INTO myrec * FROM emp WHERE empname = myname;
IF NOT FOUND THEN
   RAISE EXCEPTION ‘’employee % not found’’, myname;
END IF;
```

Kết quả trả về

- Return expression;
  - Trả kết quả về cho lời gọi hàm
- Return Next expression;
  - Khi hàm số được khai báo kết quả trả về có dạng: SETOF …
  - Lời gọi hàm của hàm func() trả về kết quả với Return Next
  - SELECT * from func();

Ví dụ 1

```
test=# CREATE FUNCTION sales_tax(real) RETURNS real AS
test-# '
test'# DECLARE
test'#     subtotal ALIAS FOR $1;
test'# BEGIN
test'#     RETURN subtotal * 0.06;
test'# END;
test'# ' LANGUAGE 'plpgsql';

test=# select sales_tax(25.8);
 sales_tax
-----------
     1.548
(1 row)
```

Ví dụ 2

```
test=# CREATE FUNCTION get_tourist_name (integer) RETURNS text AS
test-# '
test'#   DECLARE
test'#     tid ALIAS FOR $1;
test'#     fname text;
test'#     lname text;
test'#   BEGIN
test'#     SELECT INTO fname, lname name, last_name FROM tourists WHERE nt = tid;
test'#     RETURN fname || lname;
test'#   END;
test'# ' LANGUAGE 'plpgsql';

test=# select get_tourist_name(3);
 get_tourist_name
------------------
 DoHiep Thuan
(1 row)
```

Ví dụ 3

```
test=# CREATE FUNCTION merge_fields(id integer) RETURNS text AS
test-# '
test'# DECLARE
test'#     t1_row tourists%ROWTYPE;
test'# BEGIN
test'#     SELECT * INTO t1_row FROM tourists WHERE nt=id;
test'#     RETURN t1_row.name || " " || t1_row.last_name;
test'# END;
test'# ' LANGUAGE 'plpgsql';

test=# select merge_fields(4);
   merge_fields
------------------
 Do Thi Bich Hanh
(1 row)
```

Ví dụ 4

```
test=# CREATE FUNCTION merge_fields_t(id integer) RETURNS text AS
test-# '
test'# DECLARE
test'#     t1_row tourists%ROWTYPE;
test'#     t1_type tourists.type%TYPE;
test'# BEGIN
test'#     SELECT * INTO t1_row FROM tourists WHERE nt=id;
test'#     t1_type := t1_row.type;
test'#     RETURN t1_row.name || " " || t1_row.last_name || " " || t1_type;
test'# END;
test'# ' LANGUAGE 'plpgsql';
CREATE FUNCTION
test=# select merge_fields_t(1);
   merge_fields_t
----------------------
 Do Thanh Nghi sportif
(1 row)
```

Ví dụ 5

```
test=# CREATE FUNCTION concat_selected_fields(in_t cities) RETURNS text AS
test-# '
test'# BEGIN
test'#   RETURN in_t.nc || " " || in_t.name;
test'# END;
test'# ' LANGUAGE 'plpgsql';

test=# select concat_selected_fields(cities) from cities;
 concat_selected_fields
------------------------
 4 McGill
 2 Hull
 3 Laval
 1 Montreal
(4 rows)
```

Cấu trúc « IF » trong PL/pgSQL

- IF-THEN

```
IF boolean-expression THEN
    statements
END IF;
```

- IF-THEN-ELSE

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

- IF-THEN-ELSIF-ELSE

```
IF boolean-expression THEN
    statements
[ ELSIF boolean-expression THEN
    statements
[ ELSIF boolean-expression THEN
    statements
    ...]]
[ ELSE
    statements ]
END IF;
```

Ví dụ 6

```
test=# CREATE FUNCTION get_hotel_name(id integer) RETURNS text AS
test-# '
test'# DECLARE
test'#     t1_row hotels%ROWTYPE;
test'# BEGIN
test'#     SELECT * INTO t1_row FROM hotels WHERE nh=id;
test'#     IF NOT FOUND THEN
test'#       RAISE EXCEPTION "hotel % not found", id;
test'#     END IF;
test'#     RETURN t1_row.name;
test'# END;
test'# ' LANGUAGE 'plpgsql';
CREATE FUNCTION
test=# select get_hotel_name(3);
 get_hotel_name
----------------
 Onmi
(1 row)
```

Cấu trúc lặp trong PL/pgSQL

- LOOP

```
[<<label>>]
LOOP
    statements
END LOOP;
```

- EXIT

```
EXIT [ label ]
        [ WHEN expression ];
```

Cấu trúc lặp trong PL/pgSQL

- WHILE-LOOP

  [<<label>>]

  WHILE expression LOOP

    statements

  END LOOP;

- FOR (biến lặp là nguyên)

  [<<label>>]

  FOR name IN [REVERSE] expression .. expression LOOP

    statements

  END LOOP;

Cấu trúc lặp trong PL/pgSQL

- FOR (với kết quả truy vấn)

```
[<<label>>]
FOR record_or_row IN query LOOP
    statements
END LOOP;

[<<label>>]
FOR record_or_row IN EXECUTE text_expression LOOP
    statements
END LOOP;
```

Ví dụ 7

```
test=# CREATE FUNCTION get_hotels_name () RETURNS text AS '
test'#   DECLARE
test'#     output text := "\n";
test'#     row_data hotels%ROWTYPE;
test'#   BEGIN
test'#      -- Iterate through the results of a query.
test'#     FOR row_data IN SELECT * FROM hotels ORDER BY name LOOP
test'#       output := output || row_data.name || "\n";
test'#     END LOOP;
test'#     RETURN output;
test'#   END;
test'# ' LANGUAGE 'plpgsql';

test=# select get_hotels_name();
         get_hotels_name
---------------------------------
New World
Onmi
Plaza
Royal
Sami

(1 row)
```

Ví dụ 8

```
test=# CREATE FUNCTION demo(in_text text) RETURNS text AS
test-# '
test'#    DECLARE
test'#      i integer;
test'#      len integer;
test'#      out_text text;
test'#      str text;
test'#    BEGIN
test'#      str := upper(in_text);
test'#      out_text := "";
test'#      i := 1;
test'#      len := length(str);
test'#      WHILE i <= len LOOP
test'#        out_text := out_text || substr(str, i, 1) || " ";
test'#        i := i + 1;
test'#      END LOOP;
test'#      RETURN out_text;
test'#    END;
test'# ' LANGUAGE 'plpgsql';
```

Ví dụ 9

```
test=# CREATE FUNCTION get_hotels_demo () RETURNS text AS
test-# '
test'#    DECLARE
test'#      output text := "\n";
test'#      row_data hotels%ROWTYPE;
test'#      demo_text text;
test'#    BEGIN
test'#      -- Iterate through the results of a query.
test'#      FOR row_data IN SELECT * FROM hotels ORDER BY name LOOP
test'#        select into demo_text demo(row_data.name);
test'#        output := output || demo_text || "\n";
test'#      END LOOP;
test'#      RETURN output;
test'#    END;
test'# ' LANGUAGE 'plpgsql';

test=# select get_hotels_demo();
                  get_hotels_demo
------------------------------------------------------------
N E W   W O R L D
O N M I
P L A Z A
R O Y A L
S A M I
(1 row)
```

Bẫy lỗi

```
[ <<label>> ]
[ DECLARE
    declarations ]
BEGIN
    statements
EXCEPTION
    WHEN condition [ OR condition ... ] THEN
        handler_statements
    [ WHEN condition [ OR condition ... ] THEN
          handler_statements
      ... ]
END;
```

Ví dụ 10

```
test=# CREATE FUNCTION check_div(a real, b real) RETURNS real AS
test-# '
test'#    DECLARE
test'#      ret real;
test'#    BEGIN
test'#      ret := a / b;
test'#      RETURN ret;
test'#      EXCEPTION
test'#         WHEN division_by_zero THEN
test'#            RAISE NOTICE "caught division_by_zero";
test'#            RETURN a;
test'#    END;
test'# ' LANGUAGE 'plpgsql';

test=# select check_div(7, 2);
 check_div
-----------
       3.5
(1 row)
test=# select check_div(7, 0);
NOTICE:  caught division_by_zero
 check_div
-----------
         7
(1 row)
```

Cursor trong PL/pgSQL

- Đọc một vài dòng của kết quả phép truy vấn ở một thời điểm: tránh tiêu tốn bộ nhớ
  - Khai báo: name CURSOR [(arguments)] FOR query ;
  - Mở cursor:

    OPEN unbound_cursor FOR SELECT ...;

    OPEN unbound_cursor FOR EXECUTE query_string;
    OPEN bound_cursor [(argument_values)];
  - Đọc dòng kế tiếp: FETCH cursor INTO target;
  - Đóng cursor: CLOSE cursor;
  - Kết quả trả về là cursor: sử dụng refcursor

Ví dụ 11

```
test=# CREATE FUNCTION reffunc(refcursor) RETURNS refcursor AS
test-# '
test'# BEGIN
test'#     OPEN $1 FOR SELECT * FROM museums;
test'#     RETURN $1;
test'# END;
test'# ' LANGUAGE 'plpgsql';

test=# BEGIN;
test=# SELECT reffunc('funccursor');
  reffunc
------------
test=# FETCH ALL IN funccursor;
 nm | nc |    name
 ----+----+----------
   1 |  1   | CULTURE
   2 |  1   | ROI
   1 |  2   | LOUVRE
   1 |  4   | HISTOIRE
   2 |  4   | SIECLE
   3 |  1   | CHARLIE
(6 rows)
test=# COMMIT;
```

Ví dụ 12

```
test=# CREATE FUNCTION myfunc(refcursor, refcursor) RETURNS SETOF refcursor AS
test-# '
test'# BEGIN
test'#     OPEN $1 FOR SELECT * FROM cities;
test'#     RETURN NEXT $1;
test'#     OPEN $2 FOR SELECT * FROM museums;
test'#     RETURN NEXT $2;
test'#     RETURN;
test'# END;
test'# ' LANGUAGE 'plpgsql';
```

Ví dụ 13

```
test=# BEGIN;
test=# SELECT * FROM myfunc('cursor_cities', 'cursor_museums');
test=# FETCH ALL FROM cursor_cities;
 nc |   name      |  pop       | nb_museums
----+-------------+------------+------------
  4 | McGill      |   5000     |          2
  2 | Hull        |    900     |          1
  3 | Laval       |   9800     |          0
  1 | Montreal    | 150000     |          3
(4 rows)

test=# FETCH ALL FROM cursor_museums;
 nm | nc |    name
----+----+----------
  1 |  1 | CULTURE
  2 |  1 | ROI
  1 |  2 | LOUVRE
  1 |  4 | HISTOIRE
  2 |  4 | SIECLE
  3 |  1 | CHARLIE
(6 rows)
test=# COMMIT:
```

Thông báo & lỗi trong PL/pgSQL

- RAISE level ‘’format” [, variable [, ...]];
  - Level: DEBUG, LOG, INFO, NOTICE, WARNING, EXCEPTION. EXCEPTION

Ví dụ 14

```
test=# CREATE FUNCTION func() RETURNS integer AS
test-# '
test'# DECLARE
test'#     quantity integer := 30;
test'# BEGIN
test'#     RAISE NOTICE "Quantity here is %", quantity;
test'#     quantity := 50;
test'#     -- Create a subblock
test'#     DECLARE
test'#         quantity integer := 80;
test'#     BEGIN
test'#         RAISE NOTICE "Quantity here is %", quantity;
test'#     END;
test'#     RAISE NOTICE "Quantity here is %", quantity;
test'#     RETURN quantity;
test'# END;
test'# ' LANGUAGE 'plpgsql';
```

```
test=# select func();
NOTICE:  Quantity here is 30
NOTICE:  Quantity here is 80
NOTICE:  Quantity here is 50
 func
------
   50
(1 row)
```

Thủ tục trigger trong PL/pgSQL

- Những biến đặc biệt
  - NEW: giá trị mới của mẩu tin
  - OLD: giá trị củ của mẩu tin
  - TG_NAME: tên trigger
  - TG_WHEN: BEFORE|AFTER khi định nghĩa trigger
  - TG_LEVEL: ROW|STATEMENT khi định nghĩa trigger
  - TG_OP: INSERT|UPDATE|DELETE.
  - TG_RELID: OID của bảng sinh ra lời gọi trigger
  - TG_RELNAME: tên của bảng sinh ra lời gọi trigger
  - TG_NARGS: số lượng tham số của thủ tục trigger
  - TG_ARGV[]: tham số của thủ tục trigger

- Ngôn ngữ PL/pgSQL
- **Giao diện lập trình C**
- Giao diện lập trình C++
- Giao diện lập trình JAVA
- Giao diện lập trình PHP

Giao diện lập trình C

- Chương trình C giao tiếp với server:
  - Sử dụng thư viện: libpq
  - Phải include tập tin header: libpq-fe.h
  - Trình tự của chương trình
  - Tạo kết nối tới cơ sở dữ liệu trên server
  - Gửi câu lệnh truy vấn SQL tới server
  - Server thực hiện câu lệnh truy vấn
  - Lấy kết quả trả về của lệnh truy vấn
  - Hiển thị hoặc xử lý kết quả trả về
  - Đóng kết nối tới cơ sở dữ liệu

Tạo nối kết đến cơ sở dữ liệu

- Sử dụng hàm PQsetdbLogin

```
PGconn *PQsetdbLogin (
                const char *pghost,
                const char *pgport,
                const char *pgoptions,
                const char *pgtty,
                const char *dbName,
                const char *login,
                const char *pwd );
```

  - Các hàm khác: PQsetdb, PQconnectdb, etc.
  - Trạng thái kết nối: PQstatus

Gởi câu truy vấn tới server

- Sử dụng hàm PQexec

PGresult *PQexec(PGconn *conn, const char *command);

  - Các hàm khác: PQexecParams, PQprepare, PQexecPrepared, etc.
  - Trạng thái kết quả: PQresultStatus
  - Lấy kết quả trả về: PQntuples, PQnfields, PQgetvalue, etc.

int PQntuples(const PGresult *res);

int PQnfields(const PGresult *res);

char *PQgetvalue(const PGresult *res, int row_no, int column_no);

Kết thúc kết nối tới server

- Trước hết nên giải phóng kết quả và sau đó sẽ đóng nối kết đến server, sử dụng hàm PQclear và PQfinish

  void PQclear(PGresult *res);

  void PQfinish(PGconn *conn);

Chương trình ví dụ, testlibpq.c

```
/*
 *  libpq sample program
 */

#include <stdio.h>
#include <stdlib.h>
#include <pgsql/libpq-fe.h>   /* libpq header file */

int main() {

 char       query_string[256];
 PGconn      *conn;
 PGresult    *res;
 int        i;
```

Chương trình ví dụ, testlibpq.c

```
 /* connect to the database */
conn = PQconnectdb("dbname=test host=m-nghi2 port=5432 user=nghi password=xxxxxx");

/* did the database connection fail? */
if (PQstatus(conn) == CONNECTION_BAD) {
        fprintf(stderr, "Connection to database failed.\n");
        fprintf(stderr, "%s", PQerrorMessage(conn));
        exit(1);
}

/* create an SQL query string */
sprintf(query_string, "SELECT name FROM cities");

 /* send the query */
   res = PQexec(conn, query_string);
```

Chương trình ví dụ, testlibpq.c

```
    if (PQresultStatus(res) != PGRES_TUPLES_OK) {          /* did the query fail? */
        fprintf(stderr, "SELECT query failed.\n");
        PQclear(res);
        PQfinish(conn);
        exit(1);
    }

    /* loop through all rows returned */
    for (i = 0; i < PQntuples(res); i++)
        printf("%s\n", PQgetvalue(res, i, 0));          /* print the value returned */

    PQclear(res);                /* free result */

    PQfinish(conn);              /* disconnect from the database */

    return 0;
}
```

Dịch chương trình

- Chú ý đến vị trí thư viện libpq
- Dịch: **gcc –o testlibpq testlibpq.cc -lpq**

  => sẽ có chương trình thực thi **testlibpq**

- Ngôn ngữ PL/pgSQL
- Giao diện lập trình C
- **Giao diện lập trình C++**
- Giao diện lập trình JAVA
- Giao diện lập trình PHP

Giao diện lập trình C++

- Chương trình C++ giao tiếp với server:
  - Sử dụng thư viện: libpq++
  - Phải include tập tin header: libpq++.h
  - Trình tự của chương trình cũng gồm các bước
  - Tạo kết nối tới cơ sở dữ liệu trên server
  - Gửi câu lệnh truy vấn SQL tới server
  - Server thực hiện câu lệnh truy vấn
  - Lấy kết quả trả về của lệnh truy vấn
  - Hiển thị hoặc xử lý kết quả trả về
  - Đóng kết nối tới cơ sở dữ liệu

Tạo nối kết đến cơ sở dữ liệu

- Sử dụng lớp PgConnection hay PgDatabase kế thừa từ lớp PgConnection

  PgConnection::PgConnection(const char *conninfo)

  PgDatabase::PgDatabase(const char *conninfo)

  - Trạng thái kết nối

  bool PgConnection::ConnectionBad() const

  ConnStatusType PgConnection::Status()

Gởi câu truy vấn tới server

- Sử dụng các phương thức

ExecStatusType PgConnection::Exec(const char* query)

int PgConnection::ExecCommandOk(const char *query)

int PgConnection::ExecTuplesOk(const char *query)

  - Lấy thông tin lỗi:

const char *PgConnection::ErrorMessage()

  - Lấy kết quả trả về: PgDatabase::Tuples(), PgDatabase::Fields(), PgDatabase::GetValue(…), etc.

int PgDatabase::Tuples() const

int PgDatabase::Fields()

const char *PgDatabase::GetValue(int tup_no, int field_no)

Chương trình ví dụ, testlibpq++.cc

```
//
//  libpq++ sample program
//

#include <iostream>
#include <pgsql/libpq++.h>   // libpq++ header file
using namespace std;
int main() {

char       query_string[256];
int        i;
```

Chương trình ví dụ, testlibpq++.cc

```
// connect to the database
  PgDatabase conn("dbname=test host=m-nghi2 port=5432 \
                   user=nghi password=xxxxxxx");

  // did the database connection fail?
  if (conn.ConnectionBad()) {
    cerr << "Connection to database failed." << endl
        << "Error returned: " << conn.ErrorMessage() << endl;
    exit(1);
  }
```

Chương trình ví dụ, testlibpq++.cc

```
// create an SQL query string
  sprintf(query_string, "SELECT name FROM cities");

  // send the query
  if (!conn.ExecTuplesOk(query_string)) {
    cerr << "SELECT query failed." << endl;
    exit(1);
  }

  // loop through all rows returned
  for (int i=0; i < conn.Tuples(); i++)
     // print the value returned
     cout << conn.GetValue(i,0) << endl;

  return 0;
}
```

Dịch chương trình

- Chú ý đến vị trí thư viện libpq++
- Dịch: **g++ -DHAVE_NAMESPACE_STD**

-DHAVE_CXX_STRING_HEADER -DDLLIMPORT=“”

-I/usr/include/pgsql

-o testlibpq++ testlibpq++.cc

-L/usr/lib -lpq++

=> sẽ có chương trình thực thi **testlibpq++**

Giao diện lập trình C++

- Chương trình C++ giao tiếp với server:
  - Sử dụng thư viện: libpqxx
  - Phải include tập tin header:

```
#include <pqxx/pqxx>
using namespace pqxx;
```

  - Trình tự của chương trình cũng gồm các bước
  - Tạo kết nối tới cơ sở dữ liệu trên server
  - Gửi câu lệnh truy vấn SQL tới server
  - Server thực hiện câu lệnh truy vấn
  - Lấy kết quả trả về của lệnh truy vấn
  - Hiển thị hoặc xử lý kết quả trả về
  - Đóng kết nối tới cơ sở dữ liệu

Tạo nối kết đến cơ sở dữ liệu

- Sử dụng lớp connection

pqxx::connection C(const std::string & dbstring)

dbstring có dạng:

"dbname=… user=… password=… hostaddr=… port=…“

C.is_open() cho phép mở kết nối, phương thức trả về true nếu mở kết nối thành công, trả về false nếu thất bại

C.disconnect() cho phép đóng kết nối

Gởi câu truy vấn tới server

- Sử dụng transaction

pqxx::work W(C)

W.exec(const std::string & sql)

W.commit()

W.abort()

- Sử dụng nontransaction

pqxx::nontransaction N(C)

N.exec(const std::string & sql)

Hiển thị kết quả

- Ví dụ hiển thị kết quả

```
result R(N.exec(sql));
// List down all the records
for (result::const_iterator c = R.begin(); c != R.end(); ++c) {
   cout << "ID = " << c[0].as<int>() << endl;
   cout << "Name = " << c[1].as<string>() << endl;
   cout << "Age = " << c[2].as<int>() << endl;
   cout << "Address = " << c[3].as<string>() << endl;
   cout << "Salary = " << c[4].as<float>() << endl;
}
```

Chương trình ví dụ, testlibpqxx.cc

```
//
//  libpqxx sample program
//

#include <iostream>
#include <pqxx/pqxx>

using namespace std;
using namespace pqxx;

int main() {
char  query_string[256];
```

Chương trình ví dụ, testlibpqxx.cc

```
try{
// connect to the database
connection C("dbname=test hostaddr=m-nghi2 port=5432 \
                    user=nghi password=xxxxxxx");
if (C.is_open()) {
      cout << "Opened database successfully: " << C.dbname() << endl;
} else {
     cout << "Can't open database" << endl;  return 1;
}

// Create SQL statement
sprintf(query_string, "SELECT name FROM cities");
// Create a non-transactional object.
nontransaction N(C);
// Execute SQL query
result R(N.exec(query_string));
```

Chương trình ví dụ, testlibpqxx.cc

```
        // List down all the records
        for(result::const_iterator c = R.begin(); c != R.end(); c++)
                cout << c[0].as<string>() << endl;

        cout << "Operation done successfully" << endl;
        // Close connection
        C.disconnect();
} // try
catch (const std::exception &e) {
        cerr << e.what() << std::endl;  return 1;
}

return 0;

} // main
```

Dịch chương trình

- Chú ý đến vị trí thư viện libpqxx
- Dịch: **g++ -o testlibpqxx testlibpqxx.cc -lpqxx**

  => sẽ có chương trình thực thi **testlibpqxx**

- Ngôn ngữ PL/pgSQL
- Giao diện lập trình C
- Giao diện lập trình C++
- **Giao diện lập trình JAVA**
- Giao diện lập trình PHP

Giao diện lập trình Java

- Chương trình Java giao tiếp với server:
  - Sử dụng thư viện: jdbc (download pg__jdbc.jar)
  - Chỉ đường dẫn đến thư viện, CLASS_PATH
  - Phải import java.sql.*
  - Trình tự của chương trình
  - Tạo kết nối tới cơ sở dữ liệu trên server
  - Gửi câu lệnh truy vấn SQL tới server
  - Server thực hiện câu lệnh truy vấn
  - Lấy kết quả trả về của lệnh truy vấn
  - Hiển thị hoặc xử lý kết quả trả về
  - Đóng kết nối tới cơ sở dữ liệu

Tạo nối kết đến cơ sở dữ liệu

- Loading jdbc driver:

  Class.forName("org.postgresql.Driver");

- Tạo kết nối đến server

  Connection DriverManager::getConnection (
  "jdbc:postgresql://host:port/database_name",
  "username", "password")

  - Lấy thông tin lỗi kết nối: try … catch (SQLException)

Gởi câu truy vấn tới server

- Khởi tạo câu truy vấn và gửi câu truy vấn đến server

```
Statement  Connection::createStatement();
ResultSet  Statement::executeQuery(String query);
int    Statement::executeUpdate(String query);
```

  - Lấy thông tin lỗi trả về: try … catch(SQLException)
  - Lấy kết quả trả về: ResultSet::next(), ResultSet::getString(), ResultSet::getInt(), etc.

```
boolean  ResultSet::next();
String  ResultSet::getString(int column_no);
ResultSetMetaData  ResultSet::getMetaData();
int  ResultSetMetaData.getColumnCount(); …etc
```

Kết thúc kết nối tới server

- Trước hết nên giải phóng kết quả và sau đó sẽ đóng nối kết đến server

  void ResultSet::close();

  void Statement::close();

  void Connection::close();

Chương trình ví dụ, TestJdbc.java

```
/*
 *  Java sample program
 */

import java.io.*;
import java.sql.*;

public class TestJdbc {
Connection  conn; // holds database connection
Statement   stmt; // holds SQL statement

public TestJdbc() throws ClassNotFoundException,  FileNotFoundException, IOException,
    SQLException {

   // load database interface
   Class.forName("org.postgresql.Driver");
```

Chương trình ví dụ, TestJdbc.java

```
// connect to the database
   conn = DriverManager.getConnection("jdbc:postgresql://m-nghi2:5432/test", "nghi", “xxxxxxxx");

   // send the query
   stmt = conn.createStatement();
   ResultSet res = stmt.executeQuery("SELECT name FROM cities");

   if (res != null)
     while(res.next())
        System.out.println(res.getString(1));

   // free the result
   res.close();
   stmt.close();
   // disconnect from the database
   conn.close();
}
```

Chương trình ví dụ, TestJdbc.java

```
public static void main(String args[]) {
    try {
          TestJdbc test = new TestJdbc();
        }
    catch(Exception exc) {
        System.err.println("Exception caught.\n" + exc);
        exc.printStackTrace();
    }

  } // main

} // class TestJdbc
```

Dịch chương trình

- Chú ý đến vị trí thư viện pg__jdbc.jar
- Dịch: **javac TestJdbc.java**

  => chương trình thực thi trên máy ảo java **TestJdbc**

- Ngôn ngữ PL/pgSQL
- Giao diện lập trình C
- Giao diện lập trình C++
- Giao diện lập trình JAVA
- **Giao diện lập trình PHP**

Giao diện lập trình PHP

- Chương trình PHP giao tiếp với server:
  - Trình tự của chương trình
  - Tạo kết nối tới cơ sở dữ liệu trên server
  - Gửi câu lệnh truy vấn SQL tới server
  - Server thực hiện câu lệnh truy vấn
  - Lấy kết quả trả về của lệnh truy vấn
  - Hiển thị hoặc xử lý kết quả trả về trình duyệt dạng HTML
  - Đóng kết nối tới cơ sở dữ liệu

Tạo nối kết đến cơ sở dữ liệu

- Tạo kết nối đến server

  $dbconn = pg_connect("host=… dbname=…
  username=… password=…")

  - Lấy thông tin lỗi: pg_last_error()

Gởi câu truy vấn tới server

- Thực thi câu truy vấn đến server

  $result = pg_query("SQL …")

  - Lấy thông tin lỗi: pg_last_error()
  - Lấy kết quả trả về: pg_num_rows(), pg_num_fields(), pg_fetch_array(), pg_fetch_row(), etc.

  int pg_num_rows(resource result);

  int pg_num_fields(resource result);

  array pg_fetch_array(resource result) ;

  array pg_fetch_row(resource result); …etc

Kết thúc kết nối tới server

- Trước hết nên giải phóng kết quả và sau đó sẽ đóng nối kết đến server

  bool pg_free_result (resource result);

  bool pg_close (resource connection);

Chương trình ví dụ, Test.php

```
<html>
<body>
<?php

$conn = pg_pconnect("host=localhost port=5432
                     dbname=mydb user=user1 password=1234")
        or die ("connect failure". pg_last_error());

$result = pg_query($conn, "select * from cities");


echo "<TABLE BORDER=1>";
echo "<TR><TH> NC </TH> <TH> NAME </TH> <TH> POP </TH>
          <TH> NB_MUSEUMS </TH> </TR>";
```

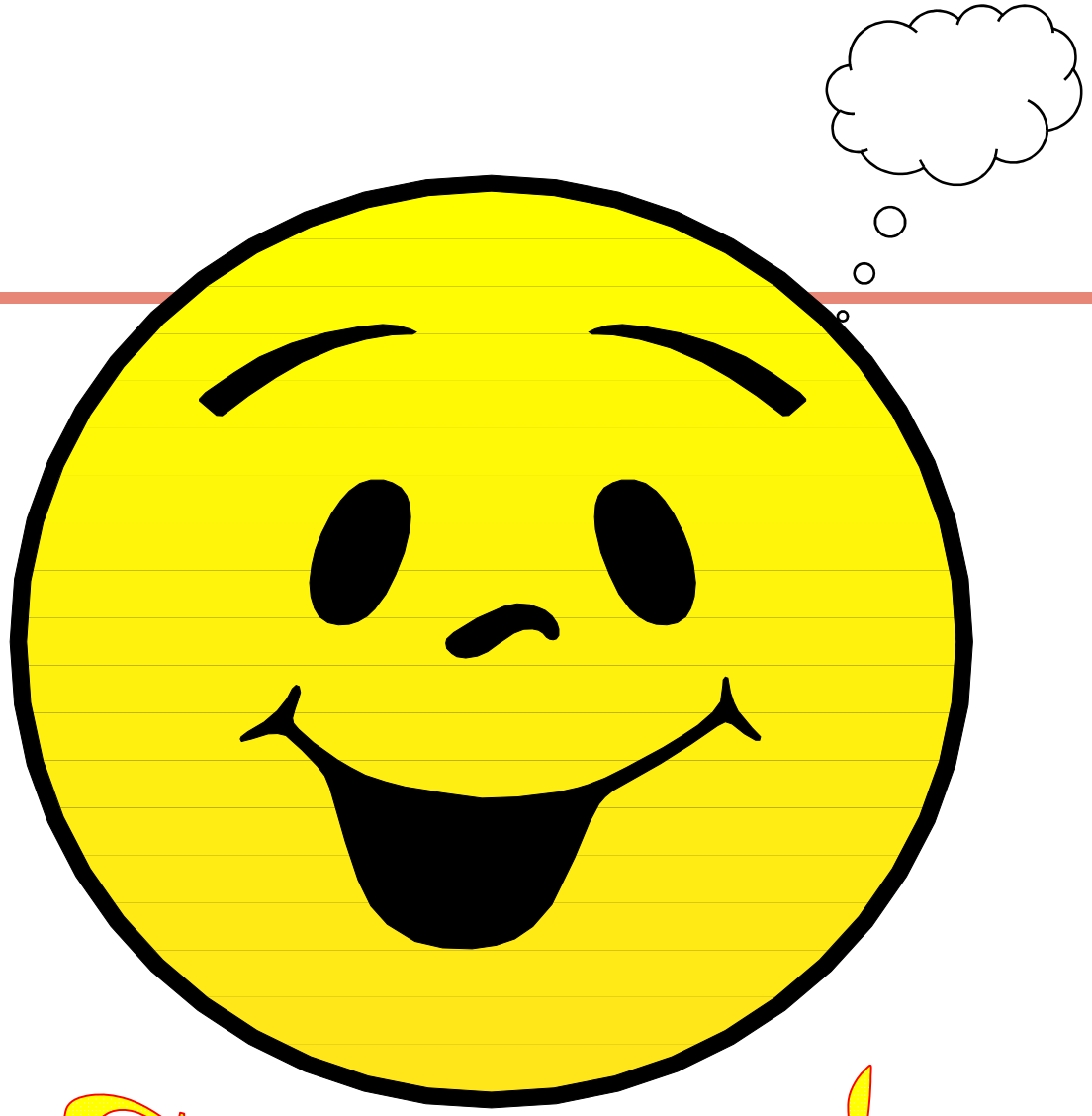
Chương trình ví dụ, Test.php

```
while ($row = pg_fetch_array($result)) {
    echo "<TR>";
    echo "<TD> " . $row["nc"]. " </TD>";
    echo "<TD> " . $row["name"]. " </TD>";
    echo "<TD> " . $row["pop"]. " </TD>";
    echo "<TD> " . $row["nb_museums"]. " </TD>";
    echo "</TR>";
}
echo "</TABLE>";

pg_free_result ($result);
pg_close ($conn);

?>

</body>
</html>
```



Cám ơn !