



PostgreSQL<sup>®</sup>  
Database Management System

Introduction to  
Programming with the  
PostgreSQL<sup>®</sup> C API (**libpq**)

by Zlatan Klebic



# The PostgreSQL® DBMS

- Licensed under the BSD license terms.
- Has features not found in other commercial DBMS's.
- Well supported among the community.
- Support for clustering.



# The PostgreSQL® C API

- Referred to as 'libpq'
- Cross-Platform C library  
(Supported in Windows, Linux, FreeBSD, etc.)
- Transparent operation across a network.

# The process of utilizing the `libpq` API

- Create a session between the server and the client.
- Test the session between the server and the client.
- Sending a query command from the client to the server.
- Retrieving the result of the query command from the server to the client.
- Closing the connection session between the client and the server.

# The `libpq` C API insight

- Main Data Types:
  - `PGconn` - structure holding the connection info.
  - `PGresult` - structure holding the query result.
  - `ExecStatusType` – holding the query status.
  - `ConnStatusType` – holding the connection status.

# The libpq C API insight

- Most common API functions:
  - `PGconn *PQConnectdb(char *connection)`
  - `void PQfinish(PGconn *connection)`
  - `char *PQerrorMessage(const PGconn *connection)`
  - `ExecStatusType PQresultStatus(const PGresult *result)`
    - `PGRES_EMPTY_QUERY`
    - `PGRES_COMMAND_OK`
    - `PGRES_TUPLES_OK`
    - `PGRES_BAD_RESPONSE`
    - `PGRES_NONFATAL_ERROR`
    - `PGRES_FATAL_ERROR`
  - `ConnStatusType PQstatus(const PGconn *connection)`
    - `CONNECTION_OK`
    - `CONNECTION_BAD`
  - `void PQclear(PGresult *result)`
  - `PGresult *PQexec(PGconn *connection, const char *sql_query)`
  - `void PQfinish(PGconn *connection)`
  - `int PQntuples(PGresult *result)`
  - `int PQnfields(PGresult *result)`
  - `char *PQgetvalue(PGresult *result, int row, int column)`



# Useful tool accompanying PostgreSQL® DBMS

- pgAdmin III
- Great GUI front-end for managing the PostgreSQL DBMS
- Fully cross-platform to Windows, Linux and FreeBSD

# PostgreSQL® C API Sample Code

## Snippet

## Inserting Data

```
#include <libpq-fe.h>
...
...
int main(void)
{
    const char *connectionInfo;
    PGconn *connection;
    PGresult *result;
    connectionInfo = "host=localhost dbname=test, user=postgres, password='postgres'";
    connection = PQconnectdb(connectionInfo);
    if(PQstatus(connection) != CONNECTION_OK)
    {
        printf("Connection not established\n");
    }
    else
    {
        printf("Connection is established\n");
        result = PQexec(connection, "INSERT INTO table1 VALUES('data1', 'data2', 33434)");
    }
    return 0;
}
```

# PostgreSQL® C API Sample Code

## Snippet

## Retrieving Data

```
#include <libpq-fe.h>
...
...
int main(void)
{
    const char *connectionInfo;
    PGconn *connection;
    PGresult *result;
    connectionInfo = "host=localhost dbname=test, user=postgres password='postgres'";
    connection = PQconnectdb(connectionInfo);
    if(PQstatus(connection) != CONNECTION_OK)
    {
        printf("Connection not established\n");
    }
    else
    {
        printf("Connection is established\n");
        result = PQexec(connection, "SELECT * FROM table1 WHERE userid = 9");
        firstname = PQgetvalue(result, 0, 0);
        lastname = PQgetvalue(result, 0, 1);
        printf("firstname= %s, lastname = %s ", firstname, lastname);
    }
    return 0;
}
```

# PostgreSQL® C API Sample Code

## Snippet

## Updating Data

```
#include <libpq-fe.h>
...
...
int main(void)
{
    const char *connectionInfo;
    PGconn *connection;
    PGresult *result;
    connectionInfo = "host=localhost dbname=test user=postgres password='postgres'";
    connection = PQconnectdb(connectionInfo);
    if(PQstatus(connection) != CONNECTION_OK)
    {
        printf("Connection not established\n");
    }
    else
    {
        printf("Connection is established\n");
        result = PQexec(connection, "UPDATE table1 SET lastname = 'anothername' WHERE userid = 9");
    }
    return 0;
}
```

# PostgreSQL® C API Sample Code

## Snippet

## Deleting Data

```
#include <libpq-fe.h>
...
...
int main(void)
{
    const char *connectionInfo;
    PGconn *connection;
    PGresult *result;
    connectionInfo = "hostaddr = 127.0.0.1 dbname = test, user = postgres password = postgres";
    connection = PQconnectdb(connectionInfo);
    if(PQstatus(connection) != CONNECTION_OK)
    {
        printf("Connection not established\n");
        return 1;
    }
    else
    {
        printf("Connection is established\n");
        result = PQexec(connection, "DELETE FROM table1 WHERE userid = 8");
    }
}
```

# The ATM Machine Tutorial

- The tutorial is as simple as possible.
- Explaining the layer of abstraction considering the porting to other DBMSs.
- Thoroughly tested in Windows environment, using Visual C++ 6.0/2003, Metrowerks Codewarrior IDE and Dev-C++ IDE with MinGW.
- The code is fully compatible with Linux or FreeBSD implementations.
- Download it from:  
[www.openinformatics.net](http://www.openinformatics.net)



# Other good projects as an example for your practice

- System/User Activity Logging
- Game Development
- Scientific Research
- Data Management
- And countless other applications...



Questions?