12) Conectividade

12.1) Integração com o PHP
12.2) Integração com Java (jdbc)
12.3) Integração com MS Access via ODBC
12.4) Integração com Visual C++
12.5) Integração com .NET

O PostgreSQL oferece conectividade com os principais linguagens e ferramentas que utilizam SGBDs.

Existem somente duas interfaces clientes incluídas na distribuição do PostgreSQL:

- libpq – é a interface primária para a linguagem C. Muitas outras interfaces são consruídas sobre esta (http://www.postgresql.org/docs/8.3/interactive/libpq.html).
- ecpq – SQL incorporado ao PostgreSQL.

Artigo sobre ecpq - http://www.vivaolinux.com.br/artigos/impressora.php?codigo=4652

Todas as outras interfaces são projetos externos e são distribuídos separadamente.

Interfaces Clientes Mantidos Externos

Name	Language	Comments	Website
DBD::Pg	Perl	Perl DBI driver	http://search.cpan.org/dist/DBD-Pg/
JDBC	JDBC	Type 4 JDBC driver	http://jdbc.postgresql.org/
libpqxx	C++	New-style C++ interface	http://pqxx.org/
Npgsql	.NET	.NET data provider	http://npgsql.projects.postgresql.org/
ODBCng	ODBC	An alternative ODBC driver	http://projects.commandprompt.com/public/odbcng/
pgtclng	Tcl		http://pgfoundry.org/projects/pgtclng/
psqlODBC	ODBC	The most commonly-used ODBC driver	http://psqlodbc.projects.postgresql.org/
psycopg	Python	DB API 2.0-compliant	http://www.initd.org/

12.1) Integração com o PHP

Para a integração do PHP com o PostgreSQL não há necessidade de se instalar nenhum driver externo, ela acontece usando a libpq do PostgreSQL. A não ser que estejamos usando no Windows poderá acontecer da conexão não estar habilitada, então basta descomentar no php.ini a linha:

extension=php_pgsql.dll

A conexão da linguagem PHP com o SGBD é efetuada através da função:

pg_connect() - http://www.php.net/manual/pt_BR/function.pg-connect.php

Exemplo:

```
$con = pg_connect("host=127.0.0.1 dbname=teste user=postgres password=postgres port=5432");
if (!$con){
   echo "Falha na conexão com o banco. Veja detalhes técnicos: " . pg_last_error($con);
}
```

Obs.: Lembrando que uma conexão ao PostgreSQL é realizada com um único banco.

Após a coexão podemos manipular a codificação de caracteres com as funções:

```
pg_set_client_encoding("UNICODE");
echo pg_client_encoding();
echo pg_set_client_encoding();
```

http://www.php.net/manual/pt_BR/function.pg-set-client-encoding.php
http://www.php.net/manual/pt_BR/function.pg-client-encoding.php

Instalar Xampplite e postgresql_phpgenerator para fazer demonstração com banco dba_projeto.

12.2) Integração com Java (jdbc)

Configuração do Java em Windows

No prompt:

doskey.com /insert

No painel de controle

Adicionar ao path:

c:\jsdk\bin;c:\jsdk\jre\bin;%PATH%

Criar a variável chasspath

.;c:\jsdk\lib\tools.jar;c:\jsdk\lib\dt.jar;c:\jsdk\lib\htmlconverter.jar;c:\jsdk\jre\lib;c:\jsdk\jre\lib\rt.jar;
c:\jsdk\jre\lib\ext\postgresql-8.3-603.jdbc3

No Linux (Ubuntu)

Para saber qual versão está usando, qual seu path e alterar, se for o caso:

sudo update-alternatives --config java

Então copie o driver jdbc para o diretório, por exemplo (postgresql-8.3-603.jdbc3.jar):

/usr/lib/jvm/java-6-sun/jre/lib/ext

O driver para integrar aplicações em Java ao PostgreSQL é o JDBC.

Para selecionar a versão correta do JDBC devemos ter em mãos a versão do PostgreSQL e a versão do Java a usar e visitar o site http://jdbc.postgresql.org/download.html#jdbcselection

A versão for Windows já traz o respectivo JDBC. Na dúvida:

- JDK 1.1 - JDBC 1. Note que com a versão 8.0 o suporte ao JDBC 1 foi removido
- JDK 1.2, 1.3 - JDBC 2.
- JDK 1.3 + J2EE - JDBC 2 EE. Contém adicional suporte para classes javax.sql.
- JDK 1.4, 1.5 - JDBC 3. Contém suporte para SSL e javax.sql, mas não requer J2EE
- JDK 1.6 - JDBC4. Suporte para métodos JDBC4 é limitado.

Para a Versão Atual (8.3) do PostgreSQL

JDBC3 Postgresql Driver, Version 8.3-603 (preferir este, mais maduro)

JDBC4 Postgresql Driver, Version 8.3-603

Pequenos exemplos em Java acessando banco PostreSQL através do JDBC:

Para rodar os exemplos abaixo precisamos ter o J2SE (JDK) instalado.

Comando para compilar:

javac nome.java

Executando:

java nome.class

Este exemplo apenas testa a conexão com o servidor e o banco de dados.

Apenas copie o driver JDBC para o diretório \jre\lib\ext do Java, crie um arquivo chamado jdbc_teste.java com este conteúdo:

```
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.SQLException;

public class jdbc_teste2 {
  public static void main(String[] argv) {
  System.out.println("Checando se o Driver está registrado com DriverManager.");

  try {
    Class.forName("org.postgresql.Driver");
  } catch (ClassNotFoundException cnfe) {
    System.out.println("Driver não encontrado!");
    System.out.println("Deixe mostrar o relatório do que encontrei e sair.");
    cnfe.printStackTrace();
    System.exit(1);
  }

  System.out.println("Driver Registrado corretamente, tentarei uma connection.");

  Connection c = null;

  try {
```

```
    // The second and third arguments are the username and password,
    // respectively. They should be whatever is necessary to connect
    // to the database.
    c = DriverManager.getConnection("jdbc:postgresql://localhost:5433/dba_projeto", "postgres",
"postgres");
  } catch (SQLException se) {
    System.out.println("Erro ao conectar: imprimindo o resultado e saindo.");
    se.printStackTrace();
    System.exit(1);
  }

  if (c != null)
    System.out.println("Conexão bem sucedida ao banco!");
  else
    System.out.println("Nunca deve ver isso.");
  }
}
```

Agora outro arquivo que mostra os registros de uma tabela

Crie um arquivo jdbc_teste2.java com o conteúdo abaixo:

```
import java.sql.*;

public class jdbc_teste  {
    public static void main(String args[]) {
        String url = "jdbc:postgresql://localhost:5433/dba_projeto";
        Connection con;
        String query = "select * from clientes";
        Statement stmt;
        try {
            Class.forName("org.postgresql.Driver");
        } catch(java.lang.ClassNotFoundException e) {
            System.err.print("ClassNotFoundException: ");
            System.err.println(e.getMessage());
```

```
        }
        try {
            con = DriverManager.getConnection(url,"postgres", "postgres");
            stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery(query);
            ResultSetMetaData rsmd = rs.getMetaData();
            int numberOfColumns = rsmd.getColumnCount();
            int rowCount = 1;
            while (rs.next()) {
                System.out.println("Cliente " + rowCount + ":  ");
                for (int i = 1; i <= numberOfColumns; i++) {
                    System.out.print("    Campo " + i + ":  ");
                    System.out.println(rs.getString(i));
                }
                System.out.println("");
                rowCount++;
            }
            stmt.close();
            con.close();

        } catch(SQLException ex) {
            System.err.print("SQLException: ");
            System.err.println(ex.getMessage());
        }
    }
}
```

Mais um pequeno exemplo, consultando uma tabela:

Crie um arquivo com nome jdbc_teste3.java, com o conteúdo das duas páginas seguintes:

```
import java.sql.*;
public class jdbc_teste3 {
public static void main(String args[])
{
```

```
		String url = "jdbc:postgresql://localhost:5433/dba_projeto";
		System.out.println("------------------------------------------------------------");
		System.out.println("Esta é a URL: " + url);
		Connection db;
		ResultSet rs;
		//Statement sq_stmt;
		try
		{
			Class.forName( "org.postgresql.Driver" );
		}
		catch ( java.lang.ClassNotFoundException e )
		{
			System.err.print( "ClassNotFoundException: " );
			System.err.println( e.getMessage () );
		}

		System.out.println("Driver do PostgreSQL selecionado. ");

try
{
		db = DriverManager.getConnection( url, "postgres", "postgres" );
}
catch ( SQLException ex )
{
		System.err.println( "SQLException: " + ex.getMessage() );
}

System.out.println("Conexão aberta. ");
try
{
		db = DriverManager.getConnection( url, "postgres", "postgres" );
		Statement sq_stmt = db.createStatement();
		String sql_str = "SELECT * FROM clientes";
```

```
            rs = sq_stmt.executeQuery(sql_str);
            while (rs.next())
            {
                        System.out.println("----------------------------------------");
                        String cpf = rs.getString("cpf");
                        String nome = rs.getString("nome");
                        String email = rs.getString("email");
                        System.out.println("CPF: " + cpf);
                        System.out.println("Nome: " + nome);
                        System.out.println("Email: " + email + " ");
            }
            System.out.println("----------------------------------------");
            System.out.println("----------------------------------------");
}
catch ( SQLException ex )
{
            System.err.println( "SQLException: " + ex.getMessage() );
}

            System.out.println("Consulta efetuada. ");

            System.out.println("Conexão fechada. ");
            System.out.println("----------------------------------------");
            }
}
```

Gerador de Relatórios

Agora vamos configurar um cliente Java para conectar ao PostgreSQL através do JDBC. Este cliente é o **gerador de relatórios** através do JDBC.

Instalar

- Instalar o iReport (http://ireport.sf.net)
- Abrir o iReport

Configurar Fonte de Dados

- Menu Data – Conexões/Fonte de Dados
- Novo
- Conexão de Banco de Dados JDBC – Próximo
- Nome (nome da conexão: conTeste)
- Driver (selecione org.postgresql.Driver)
- Caminho do JDBC (ajuste o necessário)

  jdbc:postgresql://localhost:5432/MYDATABASE (original)

  jdbc:postgresql://localhost:5432/dba_projeto
- Endereço do servidor (localhost)
- Banco de dados (dba_projeto)
- Usuário (postgres)
- Senha (postgres)

Antes de conectar ou de testar devemos copiar o driver JDBC para a pasta:

C:\Program Files\JasperSoft\iReport-2.0.4\lib

- Clique em Salvar

Criar Relatório

Clicar no botão New report (varinha mágica) para criar um novo relatório com o assistente

- Em Conexões selecionar a conexão criada (conTeste)
- Clique na caixa Consulta SQL e digite:

  select codigo, valor from pedidos;
- E clique em Próximo
- Clique na seta dupla para a direita >> e Próximo
- Próximo em Agrupar...

- Próximo em layout
- Encerrar

Compilar

- Clique no menu Criar – Compilar e Salve

Executar

- Criar – Executar relatório (usar conexão ativa)

Caso não seja exibido clique em Criar e deixe selecionados: Visualizar JRViewer e Utilizar o virtualizador de Relatórios.

Um outro gerador de relatórios muito bom é o BIRT do grupo Eclipse:

http://www.eclipse.org/birt

12.3) Integração com MS Access

Para aplicações usadas por um único usuário, o MS Access atende geralmente. Ele tem problemas para atender diversos usuários simultâneos.

A conexão do MS Access e de muitos outros aplicativos for Windows com o PostgreSQL se dá através do psqlODBC, que pode ser encontrado em:

http://www.postgresql.org/ftp/odbc/versions/msi/

Baixar a última versão, descompactar e instalar usando o arquivo psqlodbc.msi ou atualizar uma versão existente com o arquivo upgrade.bat.

Após a instalação teremos um novo driver no ODBC do Windows.

Vamos criar uma Conexão ODBC

Adicionar um banco do PostgreSQL no ODBC

Iniciar – Painel de Controle – Ferramentas Administrativas – ODBC

Adicionar – PostgreSQL ANSI ou UNICODE (depende da codificação do banco)

Database – dba_pojeto

Server – localhost

Username – postgres

Password – postgres

Então clique em Test

Observe que em Options existem diversas opções extra.

Clique em Save para guardar e OK.

Criar o Banco

Agora abra o Accecss e crie um banco de dados em branco chamado dba_projeto.

Clicar com o botão direito na janela bancos de dados (área livre)

Importar

Arquivos do tipo (Selecionar o último - Bancos de dados ODBC())

Clique na Aba acima - Fontes de dados de máquina

Selecione a PostgreSQL30W ou outro nome dado, a que criamos e clique em OK

Selecione uma das tabelas para importar e clique em OK

caso apenas vinculemos, as tabelas continuam no PostgreSQL e podemos usá-la e alterá-las, mas seus registros continuarão na tabela que está no PostgreSQL.

Já importando estamos trazendo uma cópia para o Access, uma cópia estática.

12.4) Integração com Visual C++

E também .NET e outros for Windows na parte III do Livro "PostgreSQL 8 for Windows".

Também o Capítulo 13 do PostgreSQL Prático:

http://pt.wikibooks.org/wiki/PostgreSQL_Prático/Conectividade

Conectando com Visual BASIC

```
CurrentProject.Connection.Execute StrSql2
If not linked tables then use something like
  Dim cnn as new ADODB.Connection
  cnn.Open "DSN=my_dbs_dsn_name" 'or a full PostgreSQL connection string
  cnn.Execute StrSql2
```

Outro exemplo:

Criar um DSN ODBC "pgresearch" via ADO e use:

```
Dim gcnResearch As ADODB.Connection
Dim rsUId As ADODB.Recordset

' open the database
Set gcnResearch =3D New ADODB.Connection
With gcnResearch
  .ConnectionString =3D "dsn=3Dpgresearch"
  .Properties("User ID") =3D txtUsername
  .Properties("Password") =3D txtPassword
  .Open
End With
```

Conexão com Visual Studio

http://www.linhadecodigo.com.br/ArtigoImpressao.aspx?id=1687

Using C#

```
using CoreLab.PostgreSql;
...
PgSqlConnection oPgSqlConn = new PgSqlConnection();
oPgSqlConn.ConnectionString =
    "User ID=myUsername;" +
    "Password=myPassword;" +
    "Host=localhost;" +
    "Port=5432;" +
    "Database=myDatabaseName;" +
    "Pooling=true;" +
    "Min Pool Size=0;" +
    "Max Pool Size=100;" +
    "Connection Lifetime=0";
oPgSqlConn.Open();
```

Using VB.NET

```
Imports CoreLab.PostgreSql
...
Dim oPgSqlConn As PgSqlConnection = New PgSqlConnection()
oPgSqlConn.ConnectionString =
    "User ID=myUsername;" & _
    "Password=myPassword;" & _
    "Host=localhost;" & _
    "Port=5432;" & _
    "Database=myDatabaseName;" & _
    "Pooling=true;" & _
    "Min Pool Size=0;" & _
    "Max Pool Size=100;" & _
    "Connection Lifetime=0"
oPgSqlConn.Open()
```

For more information, see: PostgreSQLDirect .NET Data Provider. Download here. Support forms here.

Conectando ao .NET com PostgreSQL

Vamos agora instalar o **.NET Data Provider para PostgreSQL** chamada **Npsgsql**. O download do provedor pode ser feito no endereço:

http://pgfoundry.org/frs/download.php/1408/Npgsql2-MS-Net-bin.zip

Descompacte o arquivo zipado e abra a pasta principal , dentro dela você verá duas pastas , abra a pasta **bin** e copie os arquivos **Npgsql.dll e Mono.Security.dll** para o diretório do projeto de sua aplicação VB .NET.

Após copiar estes arquivos clique com o botão direito do mouse sobre o nome do projeto e selecione a opção **Add Reference**, navegue até o local onde a **Npgsql.dll** foi copiada selecione-a e clique em **OK**.

Agora já temos tudo pronto para usar o **PostgreSQL** no VB 2005 Express, e é isso que vou fazer no próximo artigo: VB 2005 - Acessando o PostGreSQL II

Veja o original para as capturas e mais detalhes:
http://www.macoratti.net/07/11/vbn5_pg1.htm
http://www.macoratti.net/07/11/vbn5_pg2.htm

Openoffice2 Base

Usando o OpenOffice para abrir, editar bancos de dados PostgreSQL, como também criar consultas, formulários e relatórios.

Uma das formas de conectar o OpenOffice ao PostgreSQL é usando um driver JDBC do PostgreSQL.

- Antes devemos ter instalado o OpenOffice com suporte a Java

- Baixe daqui:

http://jdbc.postgresql.org/download.html#jars

Para o PostgreSQL 8.1 podemos pegar o JDBC3 -

http://jdbc.postgresql.org/download/postgresql-8.1-405.jdbc3.jar

A outra é o driver do próprio OO:

http://dba.openoffice.org/drivers/postgresql/index.html

- Abrir o OpenOffice, pode ser até o Writer – Ferramentas – Opções – Java – Class Path – Adicionar Arquivo (indicar o arquivo postgresql-8.0-313.jdbc2.jar baixado) e OK.

- Abrir o OOBase

- Conectar a um banco de dados existente

- Selecionar JDBC - Próximo

- URL da fonte de dados:

jdbc:postgresql://127.0.0.1:5432/bdteste

Classe do driver JDBC:

org.postgresql.Driver

Nome do usuário - postgres

password required (marque, caso use senha)

Concluir

Digitar um nome para o banco do OOBase

Pronto. Agora todas as tabelas do banco bdteste estão disponíveis no banco criado no OOBase.

Também podemos agora criar consulta com assistentes, criar formulários e relatórios com facilidade.

Fonte: http://pt.wikibooks.org/wiki/PostgreSQL_Pr%C3%A1tico/Ferramentas/OpenOffice_Base