Desenvolvimento de aplicações

- até agora temos dado comandos de SQL num interpretador de comandos: `psql`

- quando se desenvolve aplicações os comandos de SQL aparecem misturados numa linguagem de programação.

- existem várias maneiras do o fazer.

- e até podemos fazer aplicações para a web (disciplina de ADI).

Utilização de SQL em programas

- Embedded SQL

- Utilizando uma API
  (application programming interface)

Embedded SQL

- Utiliza-se um pre-processador que traduz as instruções de SQL em chamadas de funções.

- As instruções de SQL começam com `EXEC SQL`

Embedded SQL (cont.)

- A linguagem de programação e o SQL necessitam de partilhar variáveis.

```
EXEC SQL BEGIN DECLARE SECTION;
  <host-language declarations>
EXEC SQL END DECLARE SECTION;
```

Embedded SQL (cont.)

- Nos comandos SQL as variáveis partilhadas são precedidas por o sinal ':'

- Ver exemplos no vosso livro (cap. 7).

- Em PostgreSQL, o pre-processador para C chama-se *ECPG*.

- Consultar a documentação do PostgreSQL → Programmer's Guide → Client Interfaces → ECPG

APIs para o PostgreSQL

- libpq (C)
- lippq++ (C++)
- pgtcl (Tcl)
- JDBC (Java)
- PyGreSQL (Python)
- ...

libpq

- *libpq* é uma biblioteca de funções que permite que programas clientes escritos em C possam comunicar com o servidor do PostgreSQL.

- *libpq* serve de base para outras APIs.

libpq (cont.)

- Funções principais:
  - PQconnectdb
  - PQexec
  - PQstatus
  - PQntuples, PQnfields, PQgetvalue
  - PQclear
  - PQfinish

- Consultar a documentação do PostgreSQL → Programmer's Guide → Client Interfaces → libpq

libpq (cont.)

- Programas em C têm de incluir o cabeçalho `libpq-fe.h` e têm de ser *linkados* com a biblioteca `libpq`

- Aparentemente, não está instalado nas salas de aulas.
  (é necessário pacote *postgresql-devel*).

- No meu computador (Suse 9.0 Linux),

```
gcc -I/usr/include/pgsql/ -c -o prog.o prog.c
gcc prog.o -lpq -o prog
```

- Exemplo nos próximos acetatos ...

```
/*
 * Test program using libpq to access a PostgreSQL database.
 */
#include <stdio.h>
#include <libpq-fe.h>

void exit_nicely(PGconn *connection)
{
  PQfinish(connection); exit(1);
}

int main()
{
  int         row, col;
  int         nTuples, nColumns;
  PGconn      *connection;
  PGresult    *result;

  /* make a connection to the database */
  connection = PQconnectdb("host=diana dbname=filmes");

  /*
   * check that the connection was successfully made
   */
  if (PQstatus(connection) == CONNECTION_BAD)
  {
    fprintf(stderr, "Connection to database failed.\n");
    fprintf(stderr, "%s", PQerrorMessage(connection));
    exit_nicely(connection);
  }
```

```
  /*
   * execute an SQL query
   */
  result = PQexec(connection,
              "SELECT nome,datanascimento FROM actores");
  if (!result || PQresultStatus(result) != PGRES_TUPLES_OK)
  {
    fprintf(stderr, "SELECT command failed\n");
    fprintf(stderr, "%s", PQerrorMessage(connection));
    PQclear(result);
    exit_nicely(connection);
  }

  /* print the result */
  nTuples = PQntuples(result);
  nColumns = PQnfields(result);
  for (row = 0; row < nTuples; row++)
  {
    for (col = 0; col < nColumns; col++)
      printf("%-30s", PQgetvalue(result, row, col));
    printf("\n");
  }

  /* cleanup */
  PQclear(result);

  /* close the connection to the database */
  PQfinish(connection);
  return 0;
}
```