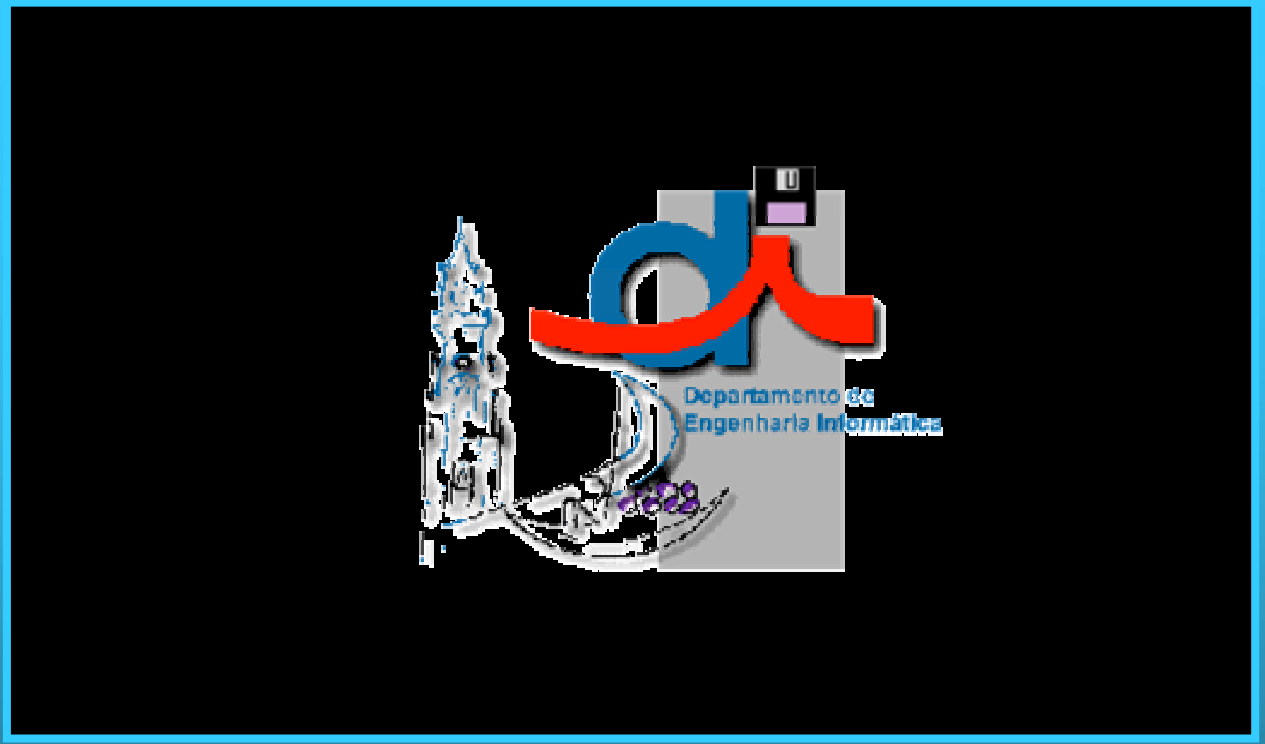


Open Database Connectivity

Nuno Castro Ferreira
nacf@dei.isep.ipp.pt

Sumário

- ODBC
- Arquitectura ODBC
- Estrutura de uma aplicação
- API do ODBC

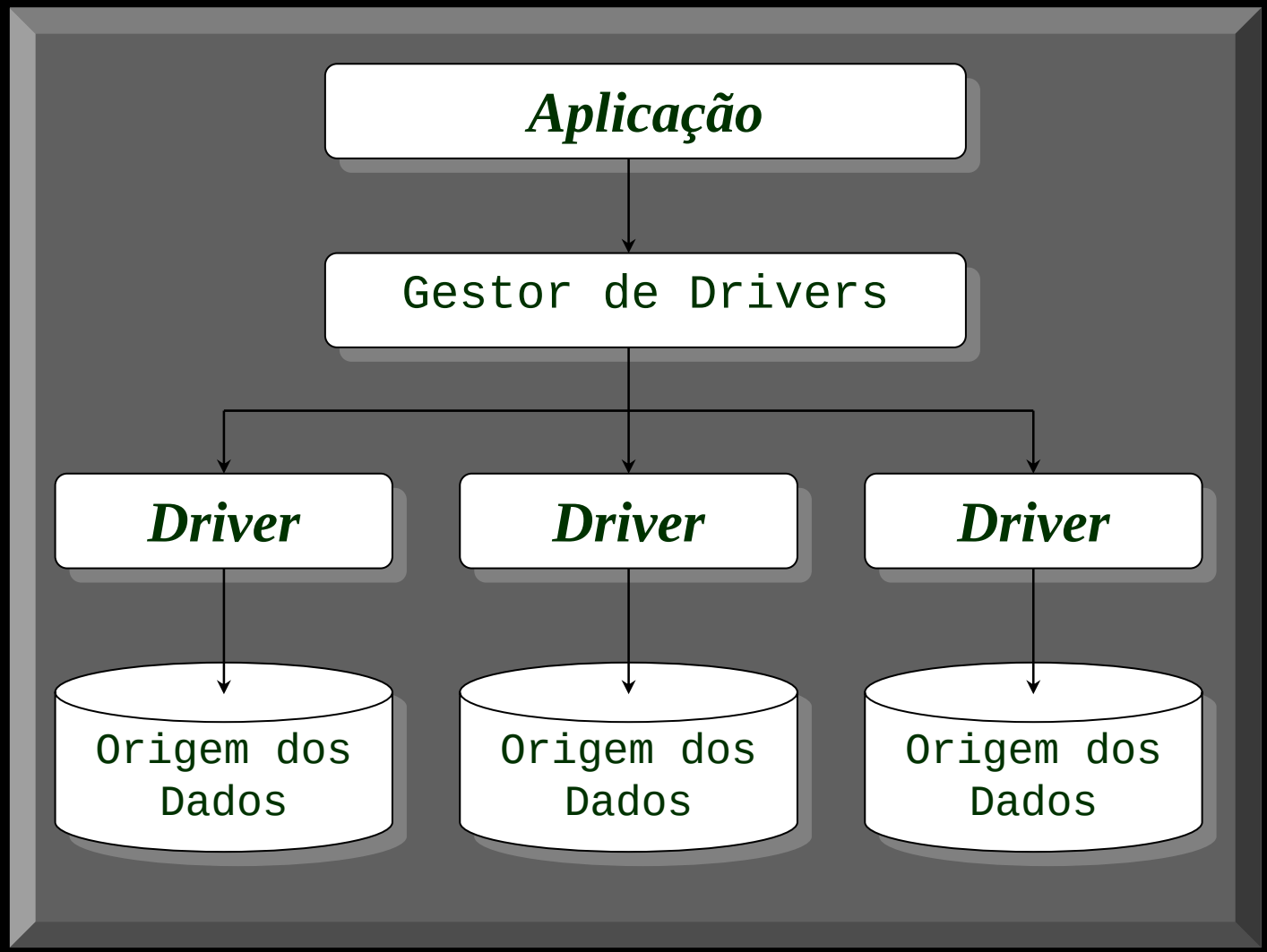
ODBC

- ODBC – Open Database Connectivity:
  - Permite o acesso a vários DBMS (Database Management System) através de uma só API
  - Isolado da aplicação e do DBMS

Arquitectura ODBC

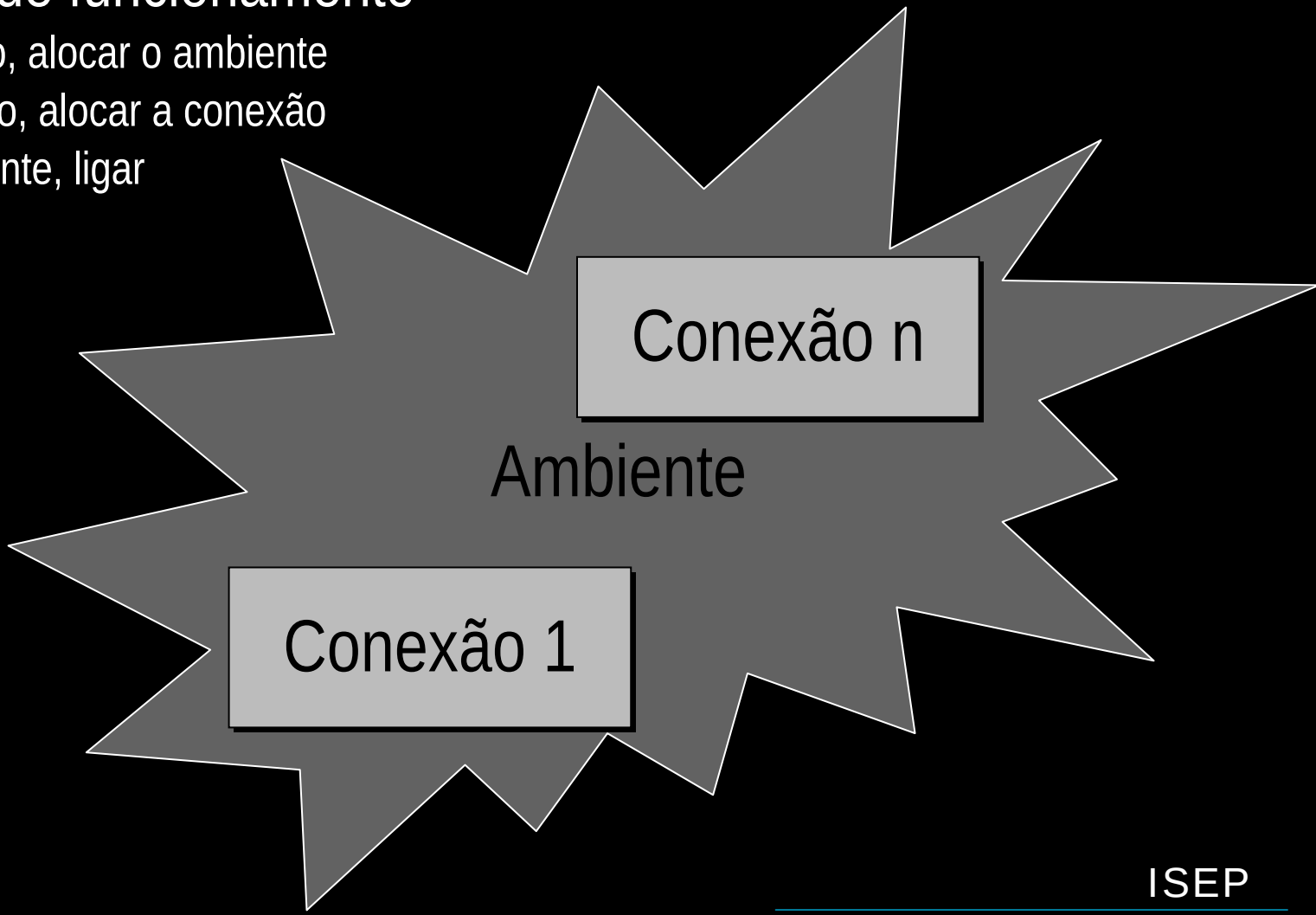
- Aplicação
  - Executa processamento e chama o ODBC
- Gestor de Drivers
  - Passa as funções de ODBC para o driver
- Drivers
  - Processa as funções do ODBC
- Origem dos Dados
  - Dados a que pretendemos aceder

Arquitectura ODBC



Estrutura de uma Aplicação

- 1- Estabelecer a Conexão
  - Lógica de funcionamento
    - Primeiro, alocar o ambiente
    - Segundo, alocar a conexão
    - Finalmente, ligar



Estrutura de uma Aplicação

- 1- Estabelecer a Conexão – Sem Transações
  - SQLAllocHandle(...Ambiente...)
  - SQLAllocHandle(...Conexão...)
  - SQLConnect

Alocar Handles

```
rc = SQLAllocHandle(SQL_HANDLE_ENV,
  SQL_NULL_HANDLE, &hAmbiente);
rc = SQLAllocHandle(SQL_HANDLE_DBC,
  hAmbiente, &hConexao);
rc = SQLConnect(hConexao, “SERVIDOR”,
  SQL_NTS, “UTILIZADOR”, SQL_NTS,
  “PASSWORD”, SQL_NTS);
```

Estrutura de uma Aplicação

- O que é uma transacção?

Uma transacção é uma sequência de operações executadas numas só unidade lógica de trabalho. Uma unidade lógica de trabalho tem que ter quatro propriedades, chamadas propriedades ‘ACID’ (Atómica, Consistente, Isolada e Durável) de modo a ser considerada uma transacção.

Estrutura de uma Aplicação

- 1- Estabelecer a Conexão – Com Transacções!
  - Alocar Handles ...
  - SQLSetConnectAttr(...SQL_ATTR_AUTOCOMMIT, SQL_AUTOCOMMIT_OFF...) ⟸ Desligar o tratamento automático de transacções
  - SQLConnect

```
rc = SQLSetConnectAttr(hConexao,
  SQL_ATTR_AUTOCOMMIT,
  (void*)SQL_AUTOCOMMIT_OFF, 0);


rc = SQLConnect(hConexao, “SERVIDOR”,
  SQL_NTS, “UTILIZADOR”, SQL_NTS,
  “PASSWORD”, SQL_NTS);
```

Estrutura de uma Aplicação

- Transacções...

Num ambiente transaccional, as transacções, iniciam quando é executado um comando com as características ACID e terminam quando é chamado o SQLEndTran, fazendo um pedido de COMMIT ou ROLLBACK para todos os comandos associados a uma conexão ou a um ambiente.

Quando o SQLEndTran é invocado, é iniciada uma nova transacção.

Estrutura de uma Aplicação

- 2- Iniciar
  - SQLGetInfo
  - SQLAllocHandle(...Comando...)
  - SQLSetStmtOption

```
rc = SQLAllocHandle(SQL_HANDLE_STMT,
  hConexao,
  &hComando);
```

Estrutura de uma Aplicação

- 3- Executar
  - SQLBindParameter
  - SQLExecute

```
rc = SQLBindParameter(hComando, 1,
  SQL_PARAM_INPUT, SQL_CHAR, SQL_VARCHAR,
  30, 0, strColuna1, 30, &iTamCol1);


rc = SQLExecDirect(hComando, “SELECT *
  FROM CLIENTES WHERE NOME = ?”, SQL_NTS);
```

Estrutura de uma Aplicação

- Vantagens do uso dos parâmetros
  - Simplificação do código
  - Aumento de desempenho em chamadas consecutivas às rotinas de execução (necessita de um Prepare, BindParameter e depois, em ciclo, o Execute)
  - Preparar uma só vez os comandos e executar as vezes necessárias
  - ...

```
CALL spClientes(?, ?, ?)
INSERT INTO CLIENTES VALUES (?, ?)
SELECT * FROM STOCK WHERE PRODUTO=?
```

Estrutura de uma Aplicação

- 4a- Se for um SELECT - Recuperar os Resultados
  - SQLNumResultCols
  - SQLDescribeCol
  - SQLBindCol
  - SQLFetch
  - SQLGetData

```
rc = SQLBindCol(hComando, 1, SQL_C_CHAR,
  strCampo1, 30, &lTamCampo1);
while(SQL_Fetch(hComando)==SQL_SUCCESS)
{...
}
```

Estrutura de uma Aplicação

- 4b-Se for um Update, Delete ou Insert
  - SQLRowCount

```
rc = SQLRowCount(hComando, &iNumeroRegistos);
```

Estrutura de uma Aplicação

- 5- Terminar a Transacção
  - SQLEndTran

```
rc = SQLEndTran(SQL_HANDLE_ENV,
          hAmbiente,
          SQL_COMMIT);
```

Estrutura de uma Aplicação

- 6- Desligar
  - SQLFreeHandle(...Comando...)
  - SQLDisconnect
  - SQLFreeHandle(...Conexão...)
  - SQLFreeHandle(...Ambiente...)

```
rc = SQLFreeHandle(SQL_HANDLE_STMT, hComando);
rc = SQLDisconnect(hConexao);
rc = SQLFreeHandle(SQL_HANDLE_DBC, hConexao);
rc = SQLFreeHandle(SQL_HANDLE_ENV, hAmbiente);
```

API do ODBC

- Includes necessários
- Estabelecer a Conexão e Iniciar
- Executar um comando
- Recuperar os resultados
- Desligar

API do ODBC

- Includes necessários
  - `#include <sql.h>`
  - `#include <sqlext.h>`
  - `#include <sqltypes.h>`

API do ODBC

Estabelecer a Conexão e Iniciar

```
BOOL InicializaODBC(SQLHENV *hEnv, SQLHDBC *hDBC)
{
SQLRETURN RetCode=SQL_SUCCESS;

// Alocar o Ambiente
RetCode=SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE,
   hEnv);
if(RetCode!=SQL_SUCCESS)
   MostraErroSQL(RetCode, SQL_NULL_HANDLE,
   SQL_NULL_HANDLE, SQL_NULL_HANDLE);

...
```

API do ODBC

Estabelecer a Conexão e Iniciar - Continuação

```
...
/* Indicar qual a versão do ODBC que vai ser usada */
RetCode=SQLSetEnvAttr(*hEnv, SQL_ATTR_ODBC_VERSION,
   (void*)SQL_OV_ODBC3, 0);
if(RetCode!=SQL_SUCCESS)
   MostraErroSQL(RetCode, *hEnv, SQL_NULL_HANDLE,
   SQL_NULL_HANDLE);

// Alocar a Conexão
RetCode=SQLAllocHandle(SQL_HANDLE_DBC, *hEnv, hDBC);
if(RetCode!=SQL_SUCCESS)
   MostraErroSQL(RetCode, *hEnv, SQL_NULL_HANDLE,
   SQL_NULL_HANDLE);
...
```

API do ODBC

Estabelecer a Conexão e Iniciar - Fim

```
...
// Configurar o tipo de commit
RetCode=SQLSetConnectAttr(*hDBC, SQL_ATTR_AUTOCOMMIT,
   (void*)SQL_AUTOCOMMIT_OFF, 0);
if(RetCode!=SQL_SUCCESS)
   MostraErroSQL(RetCode, *hEnv, *hDBC, SQL_NULL_HANDLE);

// Ligar
RetCode=SQLConnect(*hDBC, (unsigned char*)"LojaInform",
   SQL_NTS, NULL, SQL_NTS, NULL, SQL_NTS);
if(RetCode!=SQL_SUCCESS)
   MostraErroSQL(RetCode, *hEnv, *hDBC, SQL_NULL_HANDLE);

return TRUE;
}
```

API do ODBC

- Executar um comando

```
// Alocar o comando
RetCode=SQLAllocHandle(SQL_HANDLE_STMT, hDBC, &hStmt);
if(RetCode!=SQL_SUCCESS)
   MostraErroSQL(RetCode, hEnv, hDBC, SQL_NULL_HANDLE);

// Executar o Comando
RetCode=SQLExecDirect(hStmt, (unsigned char*)”SELECT *
   FROM PRODUTOS”, SQL_NTS);
if(RetCode!=SQL_SUCCESS)
   MostraErroSQL(RetCode, hEnv, hDBC, hStmt);
```

API do ODBC

- Recuperar os resultados

```
RetCode=SQLBindCol(hStmt, 1, SQL_INTEGER, &iProduto,
   sizeof(int), &TamProduto);
if(RetCode!=SQL_SUCCESS)
   MostraErroSQL(RetCode, hEnv, hDBC, hStmt);

RetCode=SQLBindCol(hStmt, 2, SQL_CHAR, cDescricao, 50,
   &TamDescricao);
if(RetCode!=SQL_SUCCESS)
   MostraErroSQL(RetCode, hEnv, hDBC, hStmt);

...
```

API do ODBC

- Recuperar os resultados - Fim

```
...
do
{
   RetCode=SQLFetch(hStmt);
   if(RetCode!=SQL_SUCCESS)
      MostraErroSQL(RetCode, hEnv, hDBC, hStmt);

   if(RetCode==SQL_SUCCESS && RetCode!=SQL_NO_DATA)
   {
      printf("Produto: [%d]\n", iProduto);
      printf("Descricao: [%s]\n", cDescricao);
   }
}
while(RetCode==SQL_SUCCESS || RetCode!=SQL_NO_DATA);
```

API do ODBC

- Fechar o Handle de comando

```
SQLFreeHandle(SQL_HANDLE_STMT, hStmt);
```

API do ODBC

- Desligar

```
void FinalizaODBC(SQLHENV hEnv, SQLHDBC hDBC)
{
   // Desligar da Base de Dados
   SQLDisconnect(hDBC);

   // Libertar o handle de Conexão
   SQLFreeHandle(SQL_HANDLE_DBC, hDBC);

   // Libertar o handle de Ambiente
   SQLFreeHandle(SQL_HANDLE_ENV, hEnv);
}
```

Fim

Obrigado pela atenção!