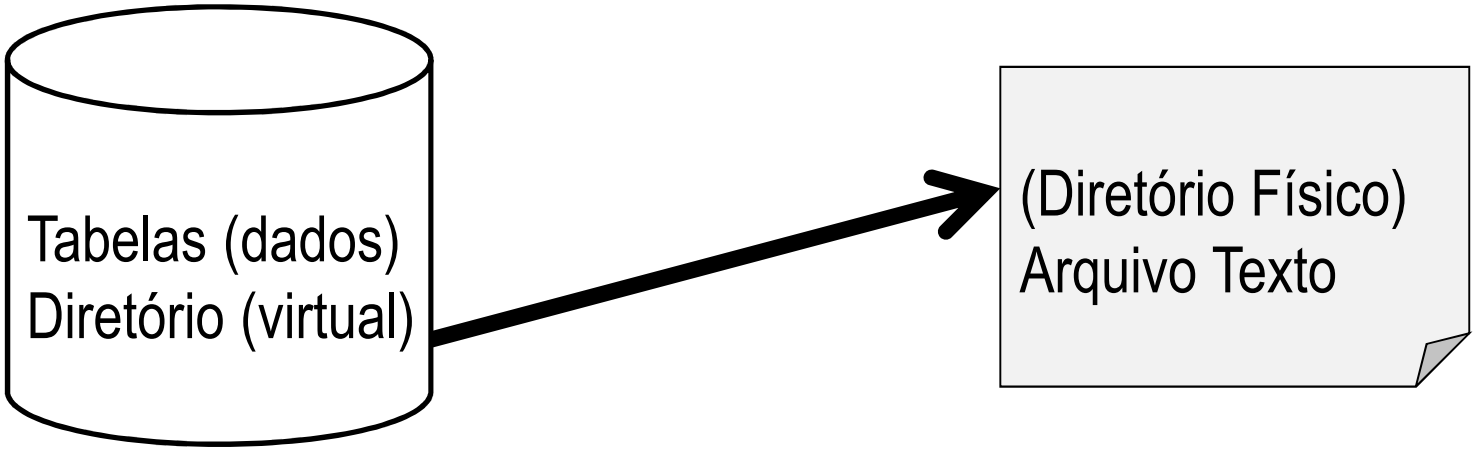
Pacotes para Dados Externos

Dados Externos



Cursores – Pacotes – Textos Externos

Conectar com administrador (conn / as sysdba)
Ou conn system/abc as sysdba

Concessão de execução para o pacote.
Grant execute on utl_file to exercicios;

Criação do diretório virtual
Create directory DESTINO as 'c:\oraclexe';

Concessão de privilégios para leitura e gravação
Grant read,write on directory DESTINO to exercicios;

Cursores – Pacotes – Textos Externos

```
create or replace procedure Exp_Estados(uf IN varchar2) is
   CURSOR CursosEstados
   IS SELECT c.nomecidade,c.estado FROM cidades c
    WHERE c.estado=uf;
Logico utl_file.file_type;
BEGIN
 Logico := utl_file.fopen('DESTINO','Cidades.txt','W');
   FOR Dados IN CursosEstados
   LOOP
      utl_file.PUT_LINE(Logico,Dados.nomecidade);
   END LOOP;
 utl_file.fclose(Logico);
end;
```

Cursores – Exportação de Dados

```
create or replace procedure Exp_Estados_SQL is
   CURSOR CursosEstados
   IS SELECT 'insert into exercicios3.cidades(idcidade,nomecidade,estado) values ('||
      idcidade||','||Chr(39)||nomecidade||Chr(39)||','||Chr(39)||
      estado||Chr(39)||');' Resultado FROM cidades e;

Logico utl_file.file_type;
BEGIN
 Logico := utl_file.fopen('DESTINO','Estados.sql','W');
   FOR Dados IN CursosEstados
   LOOP
      utl_file.PUT_LINE(Logico,Dados.Resultado);
   END LOOP;
 utl_file.fclose(Logico);
end;
```

Cursores – Pacotes – Textos Externos

```
CREATE OR REPLACE PROCEDURE Cidades_Clientes AS
CURSOR C_Cidades IS SELECT DISTINCT cl.idcidade,ci.nomecidade FROM clientes
     cl,cidades ci WHERE cl.idcidade=ci.idcidade;

CURSOR C_Clientes(P_Cidade IN NUMBER) is SELECT cli.nomecliente FROM clientes cli
WHERE cli.idcidade=P_Cidade;
Arquivo utl_file.file_type;
BEGIN
  Arquivo := utl_file.fopen('DESTINO','Cli_Cid.txt','W');
  FOR Dados_Cidades IN C_Cidades
  LOOP
    utl_file.put_line(Arquivo,Dados_Cidades.nomecidade);
    FOR Dados_Clientes IN C_Clientes(Dados_Cidades.idcidade)
    LOOP
      utl_file.put_line(Arquivo,'    '||Dados_Clientes.nomecliente);
    END LOOP;
  END LOOP;
  utl_file.fclose(Arquivo);
END;
```

Cursores – Pacotes – Textos Externos

```
<html>
<body>
<table border="1">
   <tr>
   <td>row 1, cell 1</td>
   <td>row 1, cell 2</td>
   </tr>
   <tr>
   <td>row 2, cell 1</td>
   <td>row 2, cell 2</td>
   </tr>
</table>
</body>
</html>
```

Cursores – Pacotes – Textos Externos

```
create or replace procedure Exp_Estados_html(nome IN VARCHAR2) is
   CURSOR CursosEstados
   IS SELECT c.nomecidade,c.estado FROM cidades c
   WHERE Upper(c.nomecidade) LIKE Upper(nome)||'%';
Logico utl_file.file_type;
BEGIN
 Logico := utl_file.fopen('DESTINO','html_oracle.html','W');
 utl_file.PUT_LINE(Logico,'<html><body>');
 utl_file.PUT_LINE(Logico,'<table border=1>');
   FOR Dados IN CursosEstados
   LOOP
      utl_file.PUT_LINE(Logico,'<tr>');
      utl_file.PUT_LINE(Logico,'<td>'||Dados.nomecidade||'</td>');
      utl_file.PUT_LINE(Logico,'<td>'||Dados.estado||'</td>');
      utl_file.PUT_LINE(Logico,'</tr>');
   END LOOP;
 utl_file.PUT_LINE(Logico,'</table>');
 utl_file.PUT_LINE(Logico,'</body></html>');
 utl_file.fclose(Logico);
end;
```

Verificação de Existência de Arquivo Físico

```
CREATE OR REPLACE FUNCTION Verificar_Arquivo (dir IN VARCHAR2,
       arquivo IN VARCHAR2) RETURN NUMBER is
  existe boolean;   tamanho number;  bloco number;   r NUMBER(1);
BEGIN
  r:=0;
  utl_file.fgetattr (dir,arquivo, existe, tamanho, bloco );
  IF (existe) THEN
    r:=1;
  END IF;
  RETURN r;
END;
```

```
SELECT verificar_arquivo('DESTINO','dados.txt') FROM dual
```

Verificação de Existência de Arquivo Físico

```
CREATE OR REPLACE PROCEDURE Gravar_Arquivo(pasta IN VARCHAR2,
        arquivo IN VARCHAR2,conteudo IN VARCHAR2) AS
a utl_file.file_type;
BEGIN
IF verificar_arquivo (pasta,arquivo)=0 THEN
  a:=utl_file.fopen(pasta,arquivo,'W');
ELSE a:=utl_file.fopen(pasta,arquivo,'A');
END IF;
utl_file.PUT_LINE(a,conteudo);
utl_file.fclose(a);
END;
```

```
EXEC Gravar_Arquivo('DESTINO', 'arquivo.txt','teste')
```

Exceptions

PROGRAM_ERROR
ROWTYPE_MISMATCH
SELF_IS_NULL
STORAGE_ERROR
SUBSCRIPT_BEYOND_COUNT
SUBSCRIPT_OUTSIDE_LIMIT
SYS_INVALID_ROWID
TIMEOUT_ON_RESOURCE
TOO_MANY_ROWS
VALUE_ERROR
ZERO_DIVIDE

ACCESS_INTO_NULL
CASE_NOT_FOUND
COLLECTION_IS_NULL
CURSOR_ALREADY_OPEN
DUP_VAL_ON_INDEX
INVALID_CURSOR
INVALID_NUMBER
LOGIN_DENIED
NO_DATA_FOUND
NOT_LOGGED_ON

Exceptions

```
CREATE OR REPLACE FUNCTION FDivisao(x IN NUMBER,y IN NUMBER) RETURN NUMBER IS
valor   NUMBER;
BEGIN
  valor := x/y;
  RETURN valor;
   EXCEPTION
    WHEN ZERO_DIVIDE THEN
      DBMS_OUTPUT.PUT_LINE('Erro: divisão por ZERO.');
      RETURN NULL;
END;
```

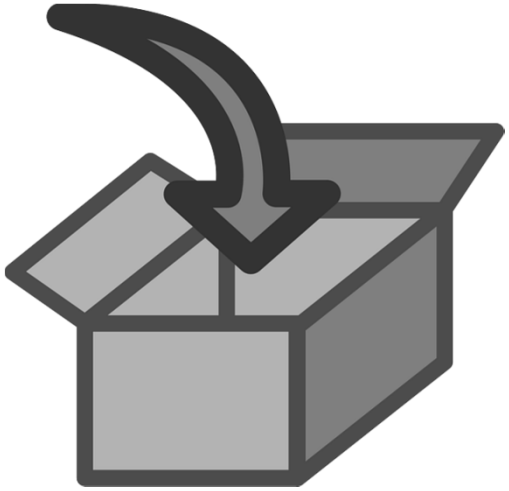
```
CREATE OR REPLACE FUNCTION FInverso(x IN NUMBER) RETURN NUMBER IS
valor   NUMBER;
BEGIN
  valor := 1/x;
  RETURN valor;
   EXCEPTION
    WHEN ZERO_DIVIDE THEN
      DBMS_OUTPUT.PUT_LINE('Erro: divisão por ZERO.');
      RETURN NULL;
END;
```

Exceptions

```
CREATE OR REPLACE PROCEDURE ImportarDados as
  Valor VARCHAR2(80);
  Arquivo UTL_FILE.FILE_TYPE;
BEGIN
  Arquivo:=UTL_FILE.FOPEN('DESTINO','dados.txt','R');
  LOOP
    BEGIN
      UTL_FILE.GET_LINE(Arquivo, Valor);
      Dbms_Output.put_line(Valor);
    EXCEPTION
      WHEN NO_DATA_FOUND THEN
        EXIT;
    END;
  END LOOP;
  UTL_FILE.FCLOSE(Arquivo);
END;
```

Entrada de Dados

Textos
Id
str



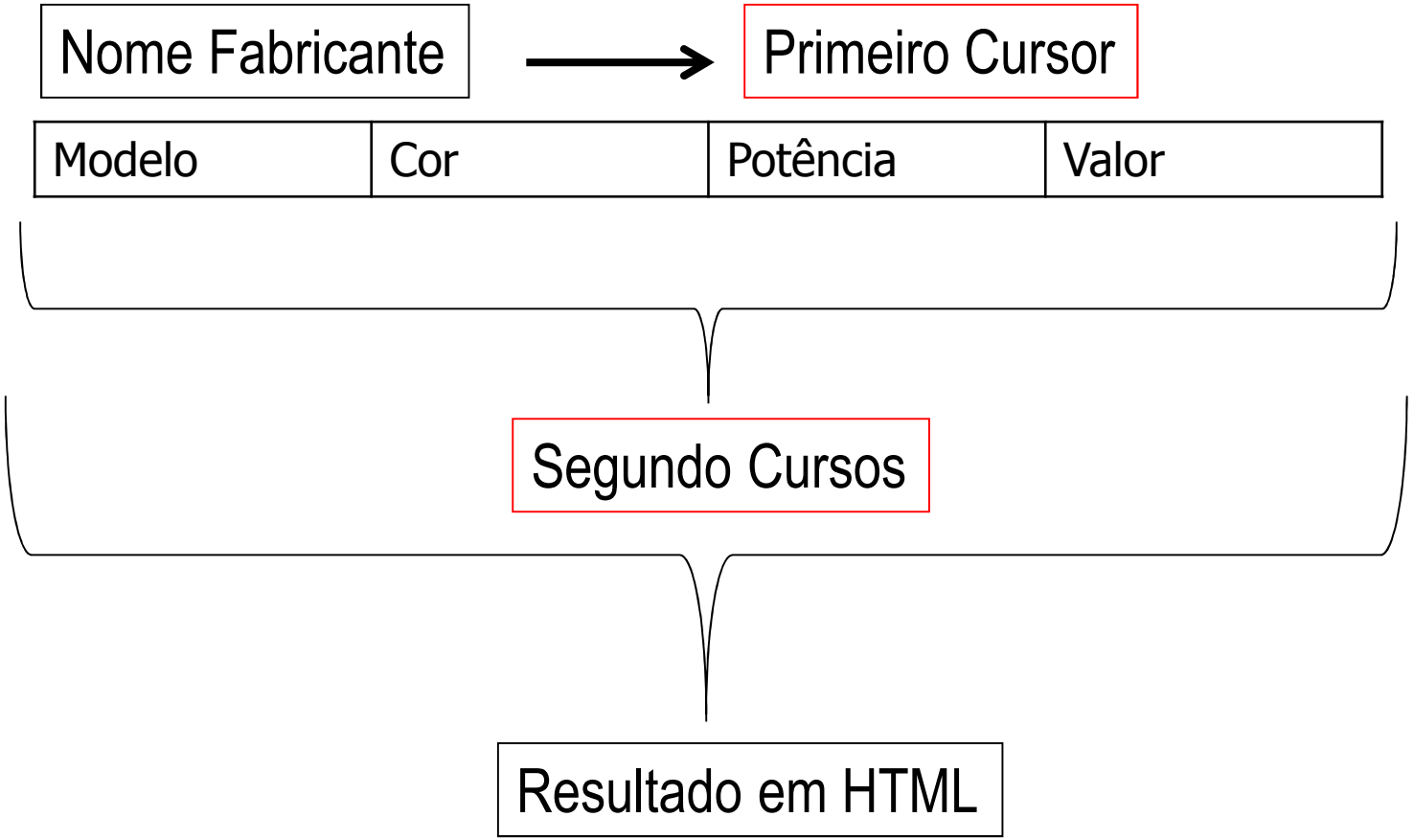
Exceptions

```
CREATE TABLE textos(id NUMBER(4) PRIMARY KEY,str VARCHAR2(100))
```

```
CREATE OR REPLACE PROCEDURE DadosTabela(origem IN VARCHAR2) as
  Valor VARCHAR2(100);
  Arquivo UTL_FILE.FILE_TYPE;
  indice NUMBER(4);
BEGIN
  Arquivo:=UTL_FILE.FOPEN('DESTINO',origem,'R');
  indice:=0;
  SELECT Nvl(Max(t.id),0) INTO indice FROM textos t;
  LOOP
    BEGIN
      UTL_FILE.GET_LINE(Arquivo, Valor);
      indice:=indice+1;
      INSERT INTO textos(id,str) VALUES (indice,Valor);
    EXCEPTION
      WHEN NO_DATA_FOUND THEN
        EXIT;
    END;
  END LOOP;
  UTL_FILE.FCLOSE(Arquivo);
END;
```

PL/SQL – Exercícios

Procedimento → RelatorioFabricantes (Nome Físico do Arquivo)



Pacotes

PL/SQL - Pacotes

```
CREATE [OR REPLACE] PACKAGE package_name
   IS | AS
   [variable_declaration ...]
   [constant_declaration ...]
   [exception_declaration ...]
   [cursor_specification ...]
         [PROCEDURE [Schema..] procedure_name
                  [ (parameter {IN,OUT,IN OUT} datatype [,parameter]) ]
         ]
         [FUNCTION [Schema..] function_name
                  [ (parameter {IN,OUT,IN OUT} datatype [,parameter]) ]
                  RETURN return_datatype
         ]
END [package_name];
```

PL/SQL - Pacotes

```
CREATE OR REPLACE PACKAGE DBMS_Calculadora
 AS

FUNCTION Somar(v1 IN NUMBER,v2 IN NUMBER) RETURN number;
FUNCTION Subtrair(v1 IN NUMBER,v2 IN NUMBER) RETURN number;
FUNCTION Multiplicar(v1 IN NUMBER,v2 IN NUMBER) RETURN number;
FUNCTION Dividir(v1 IN NUMBER,v2 IN NUMBER) RETURN number;

END;
```

PL/SQL - Pacotes

```
CREATE OR REPLACE PACKAGE BODY DBMS_Calculadora
 AS

FUNCTION Somar(v1 IN NUMBER,v2 IN NUMBER) RETURN NUMBER is
BEGIN
 RETURN v1+v2;
END;

FUNCTION Subtrair(v1 IN NUMBER,v2 IN NUMBER) RETURN NUMBER is
BEGIN
 RETURN v1-v2;
END;

FUNCTION Multiplicar(v1 IN NUMBER,v2 IN NUMBER) RETURN NUMBER is
BEGIN
 RETURN v1*v2;
END;

FUNCTION Dividir(v1 IN NUMBER,v2 IN NUMBER) RETURN NUMBER is
BEGIN
 RETURN v1/v2;
END;

END;
```

PL/SQL – Pacotes - Sobrecarga

```
TTextos=class
   procedure caracteres(arquivo1,arquivo2:string); overload;
   procedure caracteres(arquivo1:string); overload;
   procedure caracteres(arquivo1,arquivo2:string;
                                          var total:string); overload;
 end;
```

```
class TImpressao
  { public:
        void Imprimi r(int i)
              { cout << “Imprimindo int: " << i << endl; }
        void Imprimir (double f)
              { cout << " Imprimindo float: " << f << endl; }
        void Imprimir (char* c)
              { cout << " Imprimindo character: " << c << endl; }
};
```

PL/SQL – Pacotes - Sobrecarga

Pacote DBMS_Vendas

PROCEDURE consulta(datai IN DATE,dataf IN DATE)

PROCEDURE consulta(cliente IN number)

PL/SQL - Pacotes

```
CREATE OR REPLACE PACKAGE BODY DBMS_Vendas IS

PROCEDURE consulta(datai IN DATE,dataf IN DATE) as
CURSOR DadosVenda IS SELECT pr.descricaoproduto,Count(*) Total FROM vendas ve,produtos pr
WHERE pr.idproduto=ve.idproduto AND ve.datavenda>=datai AND ve.datavenda<=dataf
GROUP BY pr.descricaoproduto;
BEGIN
 FOR l IN DadosVenda
 LOOP
   Dbms_Output.put_line(l.descricaoproduto||' - '||l.Total);
 END LOOP;

END;

PROCEDURE consulta(cliente IN number) as
CURSOR DadosVenda IS SELECT pr.descricaoproduto,Count(*) Total FROM vendas ve,produtos pr
WHERE pr.idproduto=ve.idproduto AND ve.idcliente=cliente GROUP BY pr.descricaoproduto;
BEGIN
 FOR l IN DadosVenda
 LOOP
   Dbms_Output.put_line(l.descricaoproduto||' - '||l.Total);
 END LOOP;
END;
END;
```

PL/SQL – Pacotes - Exercícios

DBMS_PesquisasCar

- Função p_espacos(str IN VARCHAR2) Retorno=NUMBER
- Função p_inverter(str IN VARCHAR2) Retorno=VARCHAR2
- Função p_pesquisa(str IN VARCHAR2,car IN CHAR)
  Retorno=NUMBER
- Função p_substituir(str IN VARCHAR2,car1 IN CHAR,car2 IN char)
  Retorno=VARCHAR2

PL/SQL – Pacotes - Exercícios

Tabela:Valores_Banco
IDCliente
Saldo

DBMS_Banco

Depositar(Cliente, Valor)
Sacar(Cliente, Valor)
VerificarSaldo(Cliente)

PL/SQL - Pacotes

```
CREATE TABLE Contas(IDCliente NUMBER(10), Saldo number(13,3),
PRIMARY KEY(IDCliente))

CREATE OR REPLACE PACKAGE DBMS_Banco AS

PROCEDURE Depositar(Cliente IN NUMBER, Valor IN NUMBER);
PROCEDURE Sacar(Cliente IN NUMBER, Valor IN NUMBER);
PROCEDURE VerificarSaldo(Cliente IN NUMBER);

END;
```

PL/SQL - Pacotes

```
CREATE OR REPLACE PACKAGE BODY DBMS_Banco AS

PROCEDURE Depositar(Cliente IN NUMBER, Valor IN NUMBER) as
aux NUMBER(13,3);
BEGIN
  SELECT Count(*) INTO aux FROM Contas C WHERE C.IDCLiente=Cliente;
  IF (aux>0) THEN
  BEGIN
    UPDATE Contas SET Saldo=Saldo+Valor WHERE IDCliente=Cliente;
    Dbms_Output.put_line('Depósito Efetuado com sucesso');
  END;
  ELSE
    Dbms_Output.put_line('Conta não encontrada');
  END IF;
END;

PROCEDURE Sacar(Cliente IN NUMBER, Valor IN NUMBER) as
aux NUMBER(13,3);
aux_saldo NUMBER(13,3);

BEGIN
  SELECT Count(*) INTO aux FROM Contas C WHERE C.IDCLiente=Cliente;
  IF (aux>0) THEN
  BEGIN
    SELECT saldo INTO aux_saldo FROM Contas C WHERE C.IDCLiente=Cliente;
    IF aux_saldo>=Valor THEN
     BEGIN
       UPDATE Contas SET Saldo=(Saldo-Valor) WHERE IDCliente=Cliente;
       Dbms_Output.put_line('Saque Efetuado com sucesso');
     END;
      ELSE Dbms_Output.put_line('Saldo Insificiente');
    END IF;
  END;
  ELSE
     Dbms_Output.put_line('Conta não encontrada');
  END IF;
END;
```

```
PROCEDURE VerificarSaldo(Cliente IN NUMBER) as
aux NUMBER(13,3);
aux_saldo NUMBER(13,3);
BEGIN
  SELECT Count(*) INTO aux FROM Contas C
            WHERE C.IDCLiente=Cliente;
  IF (aux>0) THEN
  BEGIN
    SELECT Saldo INTO aux_saldo FROM Contas C
            WHERE C.IDCLiente=Cliente;
    Dbms_Output.put_line('Saldo Atual:'||aux_saldo);
  END;
    ELSE
     Dbms_Output.put_line('Conta não encontrada');
  END IF;
END;

END;
```