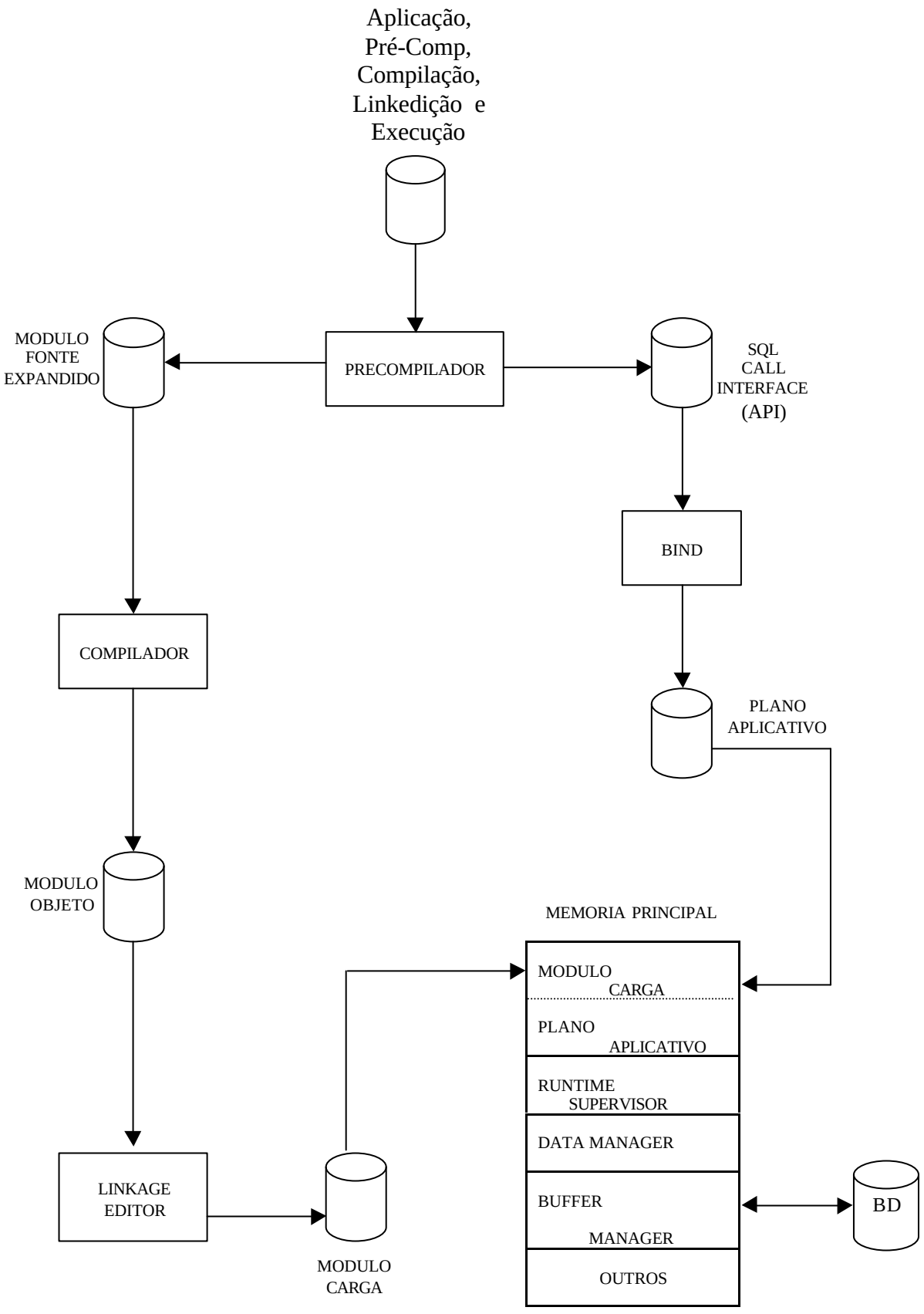
Capítulo 10

APLICAÇÃO/CONEXÃO COM LINGUAGENS PROCEDURAIS


Aplicação,
Pré-Comp,
Compilação,
Linkedição e
Execução
MODULO
FONTE
EXPANDIDO
PRECOMPILADOR
SQL
CALL
INTERFACE
(API)
BIND
COMPILADOR
PLANO
APLICATIVO
MODULO
OBJETO
MEMORIA PRINCIPAL
MODULO
CARGA
PLANO
APLICATIVO
RUNTIME
SUPERVISOR
DATA MANAGER
BUFFER
MANAGER
OUTROS
LINKAGE
EDITOR
MODULO
CARGA
BD

10.1 - Aplicação /Conexão Cobol

Exemplo 1:

```
*  Exemplo de tratamento de tabelas que  retornam na leitura mais de uma linha  *
*  Comando SQL  CURSOR                                                          *
       ID  DIVISION.
       PROGRAM-ID. COBSGBD.
      ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE  SECTION.
       01    CONT  PIC  S9(8)  COMP  VALUE 0.
             EXEC  SQL  INCLUDE  SQLCA  END-EXEC.
             EXEC  SQL  BEGIN DECLARE SECTION END-EXEC.
             01    NULOS.
                   02  NULO1  PIC S9(4) COMP.
                   02  NULO2  PIC S9(4) COMP.
             01   ATRIBUTOS.
                  02  X1      PIC S9(4) COMP VALUE 0.
                  02  X2      PIC X(20).
             01   USUARIO     PIC  X(10)  VALUE   'SYSTEM'.
             01   PASSWD      PIC  X(10) VALUE   'MANAGER'.
             01   SERVIDOR    PIC  X(8)  VALUE   'TecBD'.
        PROCEDURE  DIVISION.
       DEFINE-AREA-TRABALHO.
             EXEC SQL
             DECLARE  TRABALHO  CURSOR FOR
                   SELECT  matricula, nome  FROM  Empregado  END-EXEC.
       LOGON.
             EXEC SQL CONNECT  :USUARIO IDENTIFIED BY :PASSWD
                   USING :SERVIDOR    END-EXEC.
       INICIO.
             EXEC  SQL  OPEN  TRABALHO  END-EXEC.
              IF  SQLCODE  NOT  =   0  THEN
                  DISPLAY  ' ERRO NO OPEN ' SQLCODE
                  STOP RUN.
             EXEC SQL
             FETCH  TRABALHO INTO :X1:NULO1, :X2:NULO2
             END-EXEC.
       CORPO.
             PERFORM LER-IMPRIME UNTIL SQLCODE NOT = 0.
       FIM.
             DISPLAY 'LINHAS LIDAS '  CONT.
             EXEC SQL CLOSE  TRABALHO  END-EXEC.
             STOP RUN.
       LER-IMPRIME.
              ADD 1  TO  CONT.
              DISPLAY   'DADOS LIDOS ' X1 ' ' X2.
              EXEC  SQL
              FETCH  TRABALHO INTO :X1:NULO1, :X2:NULO2
              END-EXEC.
              DISPLAY 'SQLCODE DA LEITURA' SQLCODE.
       FIM-LER. EXIT.
```

Exemplo 2:

```
*  Exemplo de tratamento de tabelas que        *
*  retornam na leitura mais de uma linha       *
*  Comando SQL  CURSOR                         *
*  LENDO OS DADOS E ATUALIZANDO  *

 ID  DIVISION.
  PROGRAM-ID. COBSGBD.
 ENVIRONMENT DIVISION.
  DATA DIVISION.
  WORKING-STORAGE  SECTION.
  01    CONT  PIC  S9(8)  COMP  VALUE 0.
         EXEC  SQL  INCLUDE  SQLCA  END-EXEC.
         EXEC  SQL  BEGIN DECLARE SECTION END-EXEC.
         01    NULOS.
               02  NULO1  PIC S9(4) COMP.
               02  NULO2  PIC S9(4) COMP.
         01   ATRIBUTOS.
              02  X1     PIC S9(4) COMP VALUE 0.
              02  X2     PIC S9(4) COMP VALUE 0.
         01   USUARIO     PIC  X(10)  VALUE   'SYSTEM'.
         01   PASSWD      PIC  X(10) VALUE   'MANAGER'.
         01   SERVIDOR     PIC  X(8)  VALUE   'TecBD'.
   PROCEDURE  DIVISION.
  DEFINE-AREA-TRABALHO.
         EXEC SQL
         DECLARE  TRABALHO  CURSOR FOR
              SELECT a,b FROM  LIXO  for update
                END-EXEC.
  INICIO.
         DISPLAY ' INICIO  DO  PROGRAMA '.
  LOGON.
         EXEC SQL CONNECT :USUARIO IDENTIFIED BY :PASSWD
               USING :SERVIDOR
         END-EXEC.
  INICIO.
         EXEC SQL  OPEN  TRABALHO  END-EXEC.
          IF  SQLCODE  NOT  =   0  THEN
              DISPLAY  ' ERRO NO OPEN ' SQLCODE
              STOP RUN.
         EXEC SQL
         FETCH  TRABALHO INTO :X1:NULO1, :X2:NULO2
         end-exec.
   CORPO.
         PERFORM LER-IMPRIME UNTIL SQLCODE NOT = 0.
   FIM.
         DISPLAY 'LINHAS LIDAS '  CONT.
         EXEC SQL COMMIT END-EXEC.
         EXEC SQL CLOSE TRABALHO  END-EXEC.
         STOP RUN.
```

```
 LER-IMPRIME.
          ADD 1  TO  CONT.
          DISPLAY   'DADOS LIDOS ' X1 ' ' X2.
          IF X1 = 4
           EXEC SQL
              UPDATE LIXO SET B=0
             WHERE CURRENT OF TRABALHO
          END-EXEC.
          EXEC  SQL
           FETCH  TRABALHO INTO :X1:NULO1, :X2:NULO2
          END-EXEC.
          DISPLAY 'SQLCODE DA LEITURA' SQLCODE.
FIM-LER. EXIT.
```

10.2 - Aplicação /Conexão C

Exemplo 1:

```
/*  Linguagem C usando comandos SQL e conneccao com Oracle
    Consulta a tabela de Empregados a partir da matricula
*/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sqlproto.h>

EXEC SQL BEGIN DECLARE SECTION;
        VARCHAR      username[20];
        VARCHAR      password[20];
        VARCHAR      servidor[20];
        int          emp_number;
        VARCHAR      emp_name[15];
        float        salary;
        float        commission;
        short        commission_ind;
EXEC SQL END DECLARE SECTION;

        int          total_consultas;

EXEC SQL INCLUDE sqlca;

void sqlerror(void);       /* tratamento de erros */

void main()
{

        /* coneccao com o ORACLE */

        strcpy((char *)username.arr, "SCOTT");      /* username */
        username.len = (short) strlen((char *)username.arr);
        strcpy((char *)password.arr, "TIGER");      /* password */
        password.len = (short) strlen((char *)password.arr);
        strcpy((char *)servidor.arr, "TecBD");   /* servidor */
        servidor.len = (short) strlen((char *)servidor.arr);

        EXEC SQL WHENEVER SQLERROR DO sqlerror();

        EXEC SQL CONNECT :username IDENTIFIED BY :password
             using :servidor;

        printf("\nConnect com ORACLE usuario: %s\n", username.arr);
        printf("\n                            servidor: %s\n", servidor.arr);

        /* Loop, Selecao dos empregados, a partir da matricula */

        total_consultas = 0;
        while (1) {
```

```
            emp_number = 0;
            printf("\nInforme matricula (0 para terminar): ");
            scanf("%d", &emp_number);
            if (emp_number == 0) break;

            EXEC SQL WHENEVER NOT FOUND GOTO notfound;

            EXEC SQL SELECT ENAME, SAL, COMM
                 INTO :emp_name,
                     :salary,
                     :commission :commission_ind
                 FROM EMP
                 WHERE EMPNO = :emp_number;

            printf("\n\nEmpregado\tSalario\tComissao\n");
            printf("--------\t------\t----------\n");
            emp_name.arr[emp_name.len] = '\0';
            printf("%-8s\t%6.2f\t", emp_name.arr, salary);
            if (commission_ind == -1)
                  printf("NULL\n");
            else
                  printf("%6.2f\n", commission);

            total_consultas = total_consultas + 1;
            continue;

        notfound:

            printf("\n   Empregado nao existe.\n");

        } /*   Fim do loop   */

        printf("\n\nTotal  consultas: %d.\n", total_consultas);
        printf("\nFim de programa.\n");

        EXEC SQL COMMIT WORK RELEASE;
        exit(0);  /* fim de programa /desconeccao com o Oracle */

}
void sqlerror()
{
        EXEC SQL WHENEVER SQLERROR CONTINUE;

        printf("\nORACLE erro :\n");
        printf("\n% .70s \n", sqlca.sqlerrm.sqlerrmc);

        EXEC SQL ROLLBACK RELEASE;

        exit(1);
}
```

Exemplo 2:

Construir uma transação on line para cadastrar uma venda na tabela de VENDA:

VENDA (cod_cli, cod_pro, quantidade, cod_local, valor_total, data_venda)

Descrição da lógica da aplicação:

1. Interface com o usuário. (processado no cliente)
   .Obter código do cliente, código do produto, código do local e quantidade vendida.

2. Regras de negócio:
   2.1. Verificar se existe estoque para atender a venda, caso contrário, não aceitar a venda.(processado no cliente)

   2.2. Cálculo do valor total.
   quantidade x preco_unitario (processado no cliente)

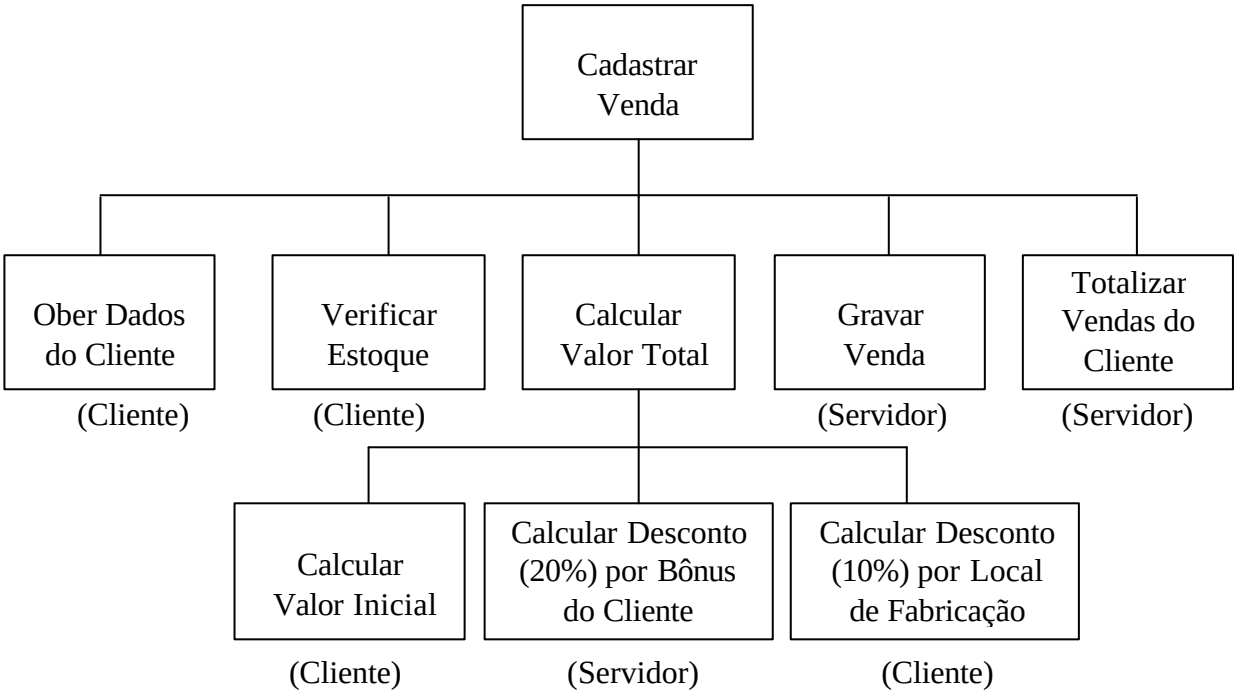
   .Se o cliente tem bonus dar um desconto de 20% e somar 10 pontos ao bonus do cliente. (processado no servidor)

   .Se o local de venda é o mesmo de fabricação do produto dar um desconto de 10%. (processado no cliente)

   2.3. Totalizar as vendas de um cliente em R$(reais) em uma tabela consolidada, onde esta é consultada por aplicações gerenciais intensivamente. (processado no servidor)

   CONSOLIDADA(cod_cli, cod_pro, total_vendas)

Diagrama de Estrutura Modular da Aplicação



```
/*************************************************************/
/*    option no precompile   SQLCHEK=FULL e USERID=.....  */
/*                                                           */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sqlproto.h>
EXEC SQL BEGIN DECLARE SECTION;
   char     username[20];
   char     password[20];
   char     servidor[8];
   int      codcli;
   int      codpro;
   int      codlocal;
   int      quantidade;
   int      lnprod;
   int      estoque;
   int      preco;
   int      total;

EXEC SQL END DECLARE SECTION;

EXEC SQL INCLUDE sqlca;

void sqlerror(void);   /* manipulacao de erros */

void main()
{
   /*************conexao com Oracle ***********/

   strcpy((char *)username, "???");
   strcpy((char *)password, "???");
   strcpy((char *)servidor, "???");

   /*EXEC SQL WHENEVER SQLERROR DO sqlerror(); */

   EXEC SQL CONNECT :username IDENTIFIED BY :password
        using :servidor;
   printf("\nConnected com  ORACLE usuario: %s\n", username);

  if (sqlca.sqlcode != 0)
          {
         printf("\nErro de Conexao\n");
         return;
        }
```

```
/**************Cadastra Vendas ************************/
  while(1) {
 codcli=0;
 preco=0;
 estoque=0;
 total=0;
 printf("\ncodigo cliente comprador\n");
 scanf("%d", &codcli);
 if (codcli==0)  break;
 printf("\ncodigo produto vendido\n");
 scanf("%d", &codpro);
 printf("\ncod local de venda\n");
 scanf("%d", &codlocal);
 printf("\nquantidade vendida\n");
 scanf("%d", &quantidade);
 /*EXEC SQL WHENEVER NOT FOUND GOTO retornar; */

 /*************** Tratar Estoque **********************/
 EXEC SQL SELECT local_fab, qtde_estoque, preco_unitario
     INTO :lnprod, :estoque, :preco
        FROM  PRODUTO WHERE PN = :codpro  For Update ;
 if (sqlca.sqlcode != 0)
 {
 printf("\nsqlcode =\n", sqlca.sqlcode);
 printf("\n% .70s \n",sqlca.sqlerrm.sqlerrmc);
 printf("\nProduto nao existe\n");
 EXEC SQL ROLLBACK;
 continue;
 }

 if (estoque < quantidade)
 {
 printf("\nProduto sem estoque\n");
EXEC SQL ROLLBACK;
 continue;
 }

 EXEC SQL UPDATE PRODUTO
  set qtde_estoque = qtde_estoque - :quantidade
      Where pn = :codpro;

  /********** Calcula Preco **********************/

 total = quantidade * preco;
 printf("\nPreco Calculado %d.\n", total);

/***************Tratar Bonus ************************/
 EXEC SQL EXECUTE
         BEGIN
           TRATA_CLI (:codcli, :total, :total);
         END;
  END-EXEC;
```

```
/************ Tratar Local ***********************/

 if (codlocal == lnprod)
 total = total * 90/100;

 printf("\nValor Calculado da Venda = %d. \n", total);

 /************ Gravar Vendas ***********************/

 EXEC SQL INSERT INTO VENDA
  VALUES(:codcli,:codpro,:codlocal,:quantidade,:total, sysdate);

 if (sqlca.sqlcode == 0)
  { printf("\nVenda Incluida %d. \n", total);
   EXEC SQL COMMIT;
  }
  else
   EXEC SQL ROLLBACK;
  continue;
exit(0);
 }
}
 /*************** Tratar Erro ***********************/
 void sqlerror(void)
{
 printf("\nOracle Erro\n", sqlca.sqlcode);
 printf("\nOracle Erro\n", sqlca.sqlerrm);
 EXEC SQL ROLLBACK;
 printf("\nFim de programa c/erro.\n");
 exit(1);
}
```

Atenção: Esta instrução ativa o Trigger.

Semelhante ao exercício anterior, agora,
usando Banco de Dados distribuído
e
Processamento disttribuído

```
#ifdef RCSID
static char *RCSid =
   "$Header: CSPGMC3.pc.d,v 1.4.710.1 93/11/24 13:20:04 losborne: Needtomrg_7_1 $
CSPGM3.pc ";
#endif
/*************************************************************/
/*    option no precompile   SQLCHEK=FULL e USERID=.....  */
/*                                                           */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sqlproto.h>
EXEC SQL BEGIN DECLARE SECTION;
   char     username[20];
   char     password[20];
   char     servidor[9];
   char     username2[20];
   char     password2[20];
   char     servidor2[8];
   int      codcli;
   int      codpro;
   int      codlocal;
   int      quantidade;
   int      lnprod;
   int      estoque;
   int      preco;
   int      total;

EXEC SQL END DECLARE SECTION;

EXEC SQL INCLUDE sqlca;

void sqlerror(void);   /* manipulacao de erros */

void main()
{
   /*************conexao com Oracle ***********/

   strcpy((char *)username, "???");
   strcpy((char *)password, "???");
   strcpy((char *)servidor, "TecBD");
   strcpy((char *)username2, "???");
   strcpy((char *)password2, "???");
   strcpy((char *)servidor2, "RDC");
```

```
/*EXEC SQL WHENEVER SQLERROR DO sqlerror(); */
/*********************************************/
/*   DBNOME1 E DBNOME2 VARIAVEIS DE CONTROLE DO
     PROCESSAMENTO DISTRIBUIDO  */

EXEC SQL DECLARE DBNOME1 DATABASE;
EXEC SQL DECLARE DBNOME2 DATABASE;

/****   LOGON NO SERVIDOR 1  ****/
EXEC SQL CONNECT :username IDENTIFIED BY :password
     AT DBNOME1      using :servidor;

printf("\nConnected com  SERVIDOR 1 : %s\n", servidor);

if (sqlca.sqlcode != 0)
         {
        printf("\nErro de Conexao servidor 1 \n");
        return;
       }
/****  LOGON NO SERVIDOR  2  ****/

EXEC SQL CONNECT :username2 IDENTIFIED BY :password2
   AT DBNOME2  using :servidor2;

printf("\nConnected com  SERVIDOR 2 : %s\n", servidor2);

if (sqlca.sqlcode != 0)
         {
        printf("\nErro de Conexao servidor 2 \n");
        return;
       }
/**************Cadastra Vendas ************************/

while(1) {
codcli=0;
preco=0;
estoque=0;
total=0;
printf("\ncodigo cliente comprador\n");
scanf("%d", &codcli);
if (codcli==0)  break;
printf("\ncodigo produto vendido\n");
scanf("%d", &codpro);
printf("\ncod local de venda\n");
scanf("%d", &codlocal);
printf("\nquantidade vendida\n");
scanf("%d", &quantidade);
```

```
/************** Tratar Estoque **********************/
EXEC SQL AT DBNOME1
   SELECT local_fab, qtde_estoque, preco_unitario
   INTO :lnprod, :estoque, :preco
      FROM  PRODUTO WHERE PN = :codpro for update;
if (sqlca.sqlcode != 0)
{
printf("\nsqlcode =\n", sqlca.sqlcode);
printf("\n% .70s \n",sqlca.sqlerrm.sqlerrmc);
printf("\nProduto nao existe\n");
continue;
}

if (estoque < quantidade)
{
printf("\nProduto sem estoque\n");
EXEC SQL ROLLBACK;
continue;
}

EXEC SQL AT DBNOME1
   UPDATE PRODUTO
   set qtde_estoque = qtde_estoque - :quantidade
     Where pn = :codpro;

/**************** Calcula Preco *********************/

total = quantidade * preco;
printf("\nPreco Calculado %d.\n", total);

/**************Tratar Bonus ************************/
/************  PROCESSADO EM SERVIDOR DIFERENTE ***/
EXEC SQL AT DBNOME2
        EXECUTE
         BEGIN
           TRATA_CLI (:cod_cli, :total, :total);
         END;
 END-EXEC;
/************ Tratar Local ************************/

if (codlocal == lnprod)
total = total * 90/100;

printf("\nValor Calculado da Venda = %d. \n", total);
```

```
   /************ Gravar Vendas **********************/

   EXEC SQL AT DBNOME1
    INSERT INTO VENDA
    VALUES(:codcli,:codpro,:codlocal,:quantidade,:total, sysdate);

   if (sqlca.sqlcode == 0)
    { printf("\nVenda Incluida %d. \n", total);
     EXEC SQL AT DBNOME1 COMMIT;
     EXEC SQL AT DBNOME2 COMMIT;
    }
    else{
     EXEC SQL AT DBNOME1 ROLLBACK;
     EXEC SQL AT DBNOME2 ROLLBACK;}
    continue;
exit(0);
  }
 }
  /*************** Tratar Erro **********************/
  void sqlerror(void)
 {
  printf("\nOracle Erro\n", sqlca.sqlcode);
  printf("\nOracle Erro\n", sqlca.sqlerrm);
  EXEC SQL ROLLBACK;
  printf("\nFim de programa c/erro.\n");
  exit(1);
 }
```