PROGRESS DCA

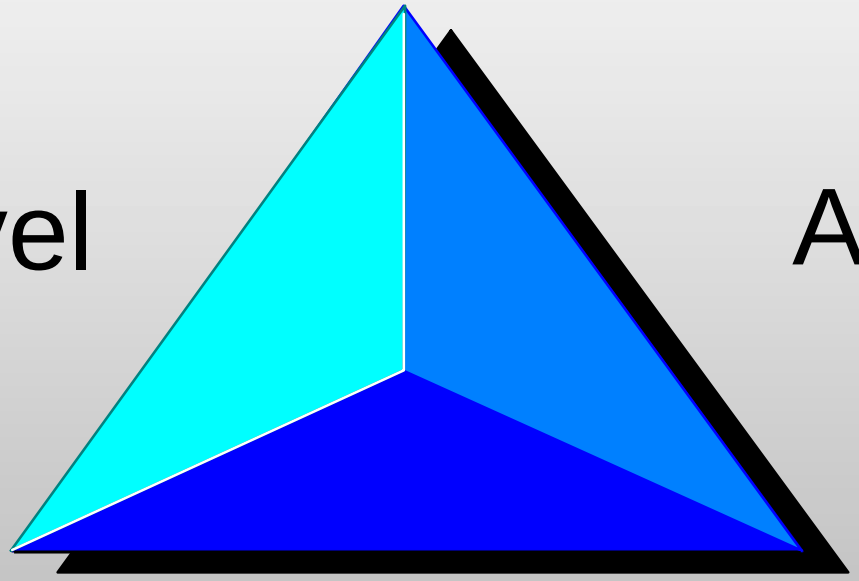
Desenvolvendo Aplicações Caracter

Completo ambiente de desenvolvimento de aplicações, composto por :

- **Sistema Gerenciador de BD Relacional**
- **Linguagem de 4a. Geração**
- **Ferramentas de Programação**

Benefícios

Flexível



Aberto

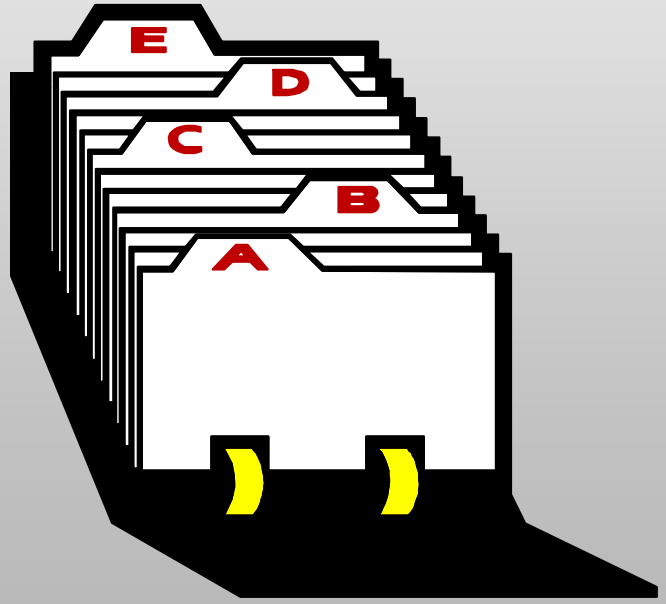
Completo

Benefícios

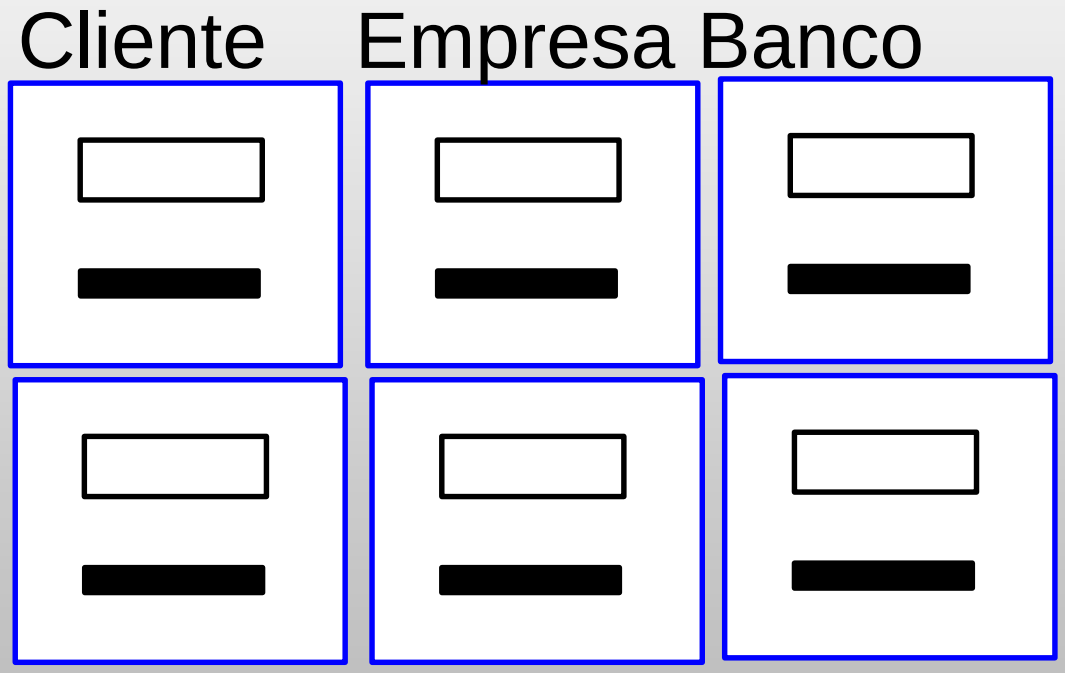
- Flexível
  - Assegura portabilidade das aplicações em outras plataformas
- Aberto
  - Suporta o padrão industrial
  - Assegura opções de desenvolvimento heterogêneo
- Completo
  - Permite construção de grandes aplicações
  - Produz soluções Client/Server

Componentes Lógicos

- Base de Dados
  - Tabela
    - Registro
    - Campo
  - Índice

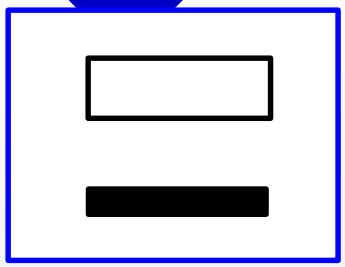


Componentes Lógicos

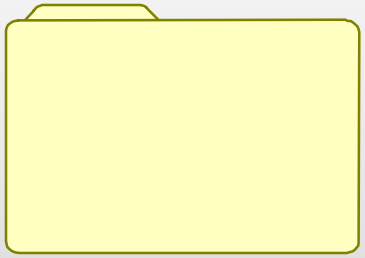


= Base Dados

COMPONENTES LÓGICOS

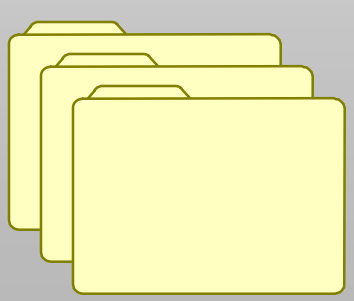


= Tabela



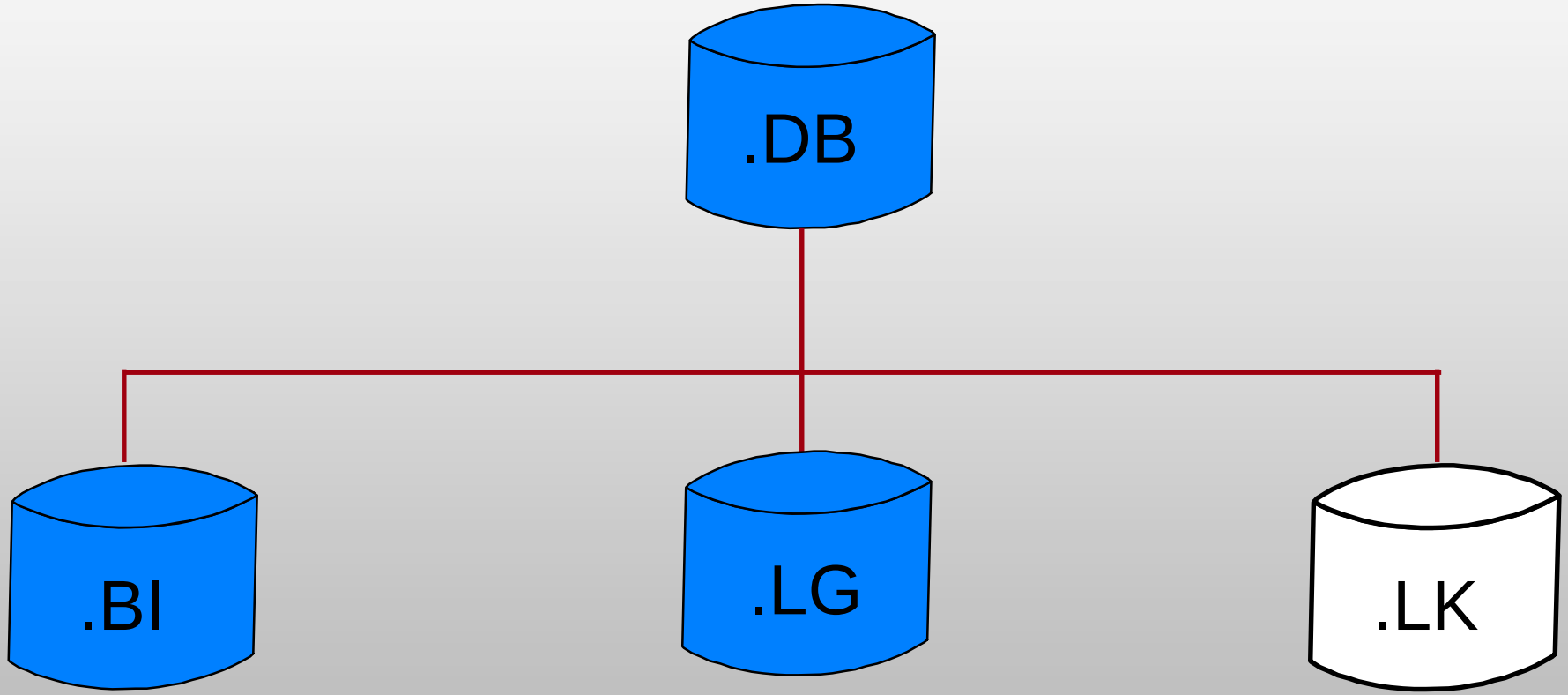
= Registro

= Campo

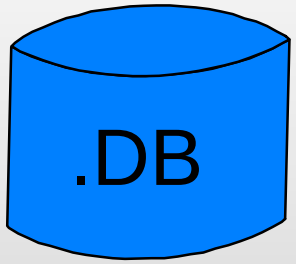


= Índice

Componentes Físicos

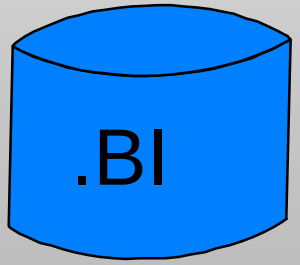


Componentes Físicos



-> **Database**

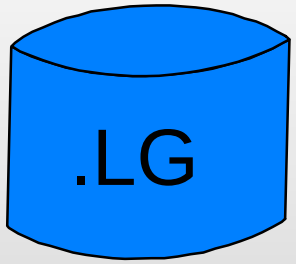
contém os dados atuais e descrições da base



-> **Before-Image**

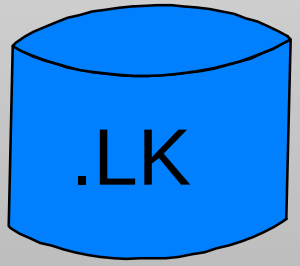
contém a imagem dos dados de uma transação

Componentes Físicos



-> **Log**

contém as referências dos usuários , hora de entrada, saída e programas acessados (histórico).



-> **Lock**

Informa se a base está sendo acessada.

Outros Arquivos Importantes

Progress.INI -> **Configurações do Progress.**

Contém informações importantes sobre a sessão e localização de arquivos do progress.

```
[Startup]
ImmediateDisplay=yes
Use-3D-Size=No
Keep3DFillinBorder=yes
DLC=c:\dlc82
PROCFG=c:\dlc82\PROGRESS.CFG
PROMSGS=c:\dlc82\promsgs

PROPATH=.,c:\dlc82\gui,c:\dlc82,c:\dlc82\bin
```

Outros Arquivos Importantes

.PF -> **Configurações da Sessão e conexões das bases de dados.**

- **Os parâmetros de conexão das base de dados que serão utilizados no aplicativo ficam setados aqui;**

- **Temos também os parâmetros da sessão que será executada.**

Acessando o Progress

- **Criar um diretório de trabalho no winchester;**
- **Criar um atalho com:**

Objeto: **C:\DLC82\bin\prowin32.exe -p _desk.p**

Iniciar em: **C:\<diretório de trabalho>**

Acessando o Progress

Principais parâmetros:

-1	carga do banco de dados monousuário
-d	formato de data (-d dmy)
-E	formato europeu de ponto decimal
-T	diretório dos arquivos temporários
-db	nome físico do banco a ser conectado
-ld	nome lógico do banco
-H	nome da máquina onde está o banco
-S	nome do serviço no servidor para o banco
-N	protocolo de comunicação (-N TCP)
-pf	arquivo de parâmetros
-ininame	nome do arquivo .INI utilizado
-basekey	utiliza .INI ou o Registry do Windows (INI)
-cpstream	mapa de caracteres a ser utilizado (ibm850)
-p	programa a ser executado

Acessando o Progress

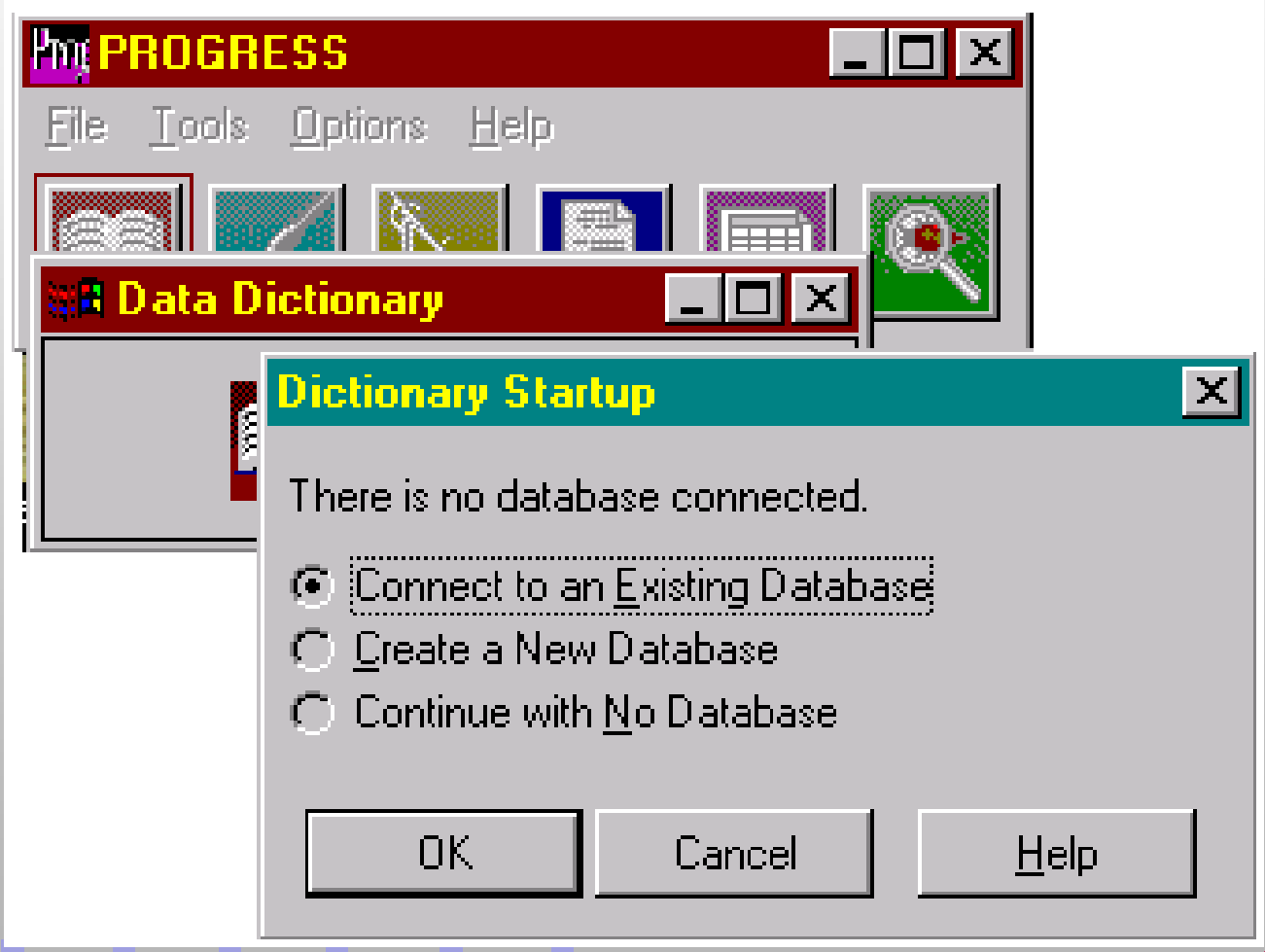
1 2 3 4 5 6

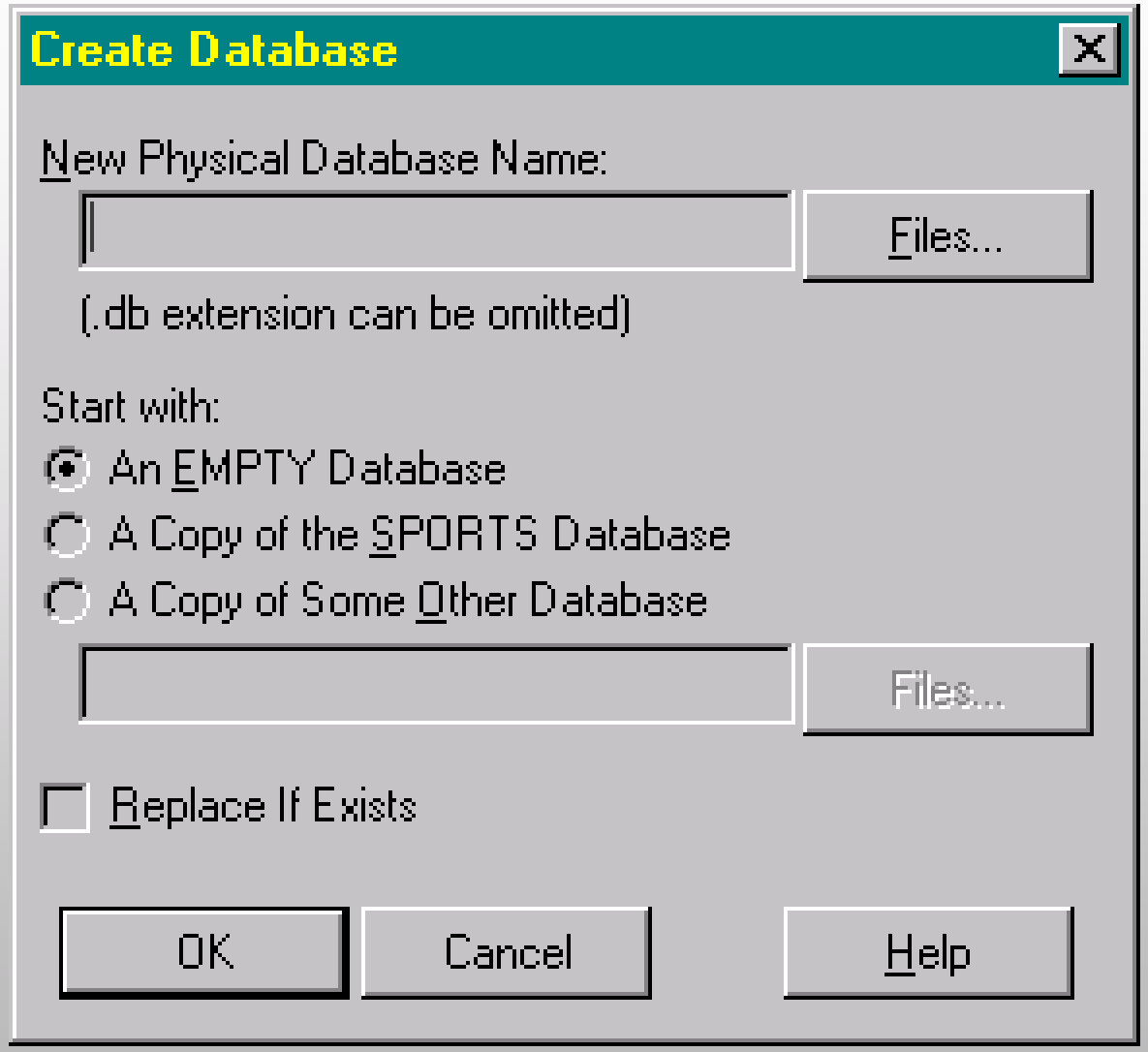
Onde:

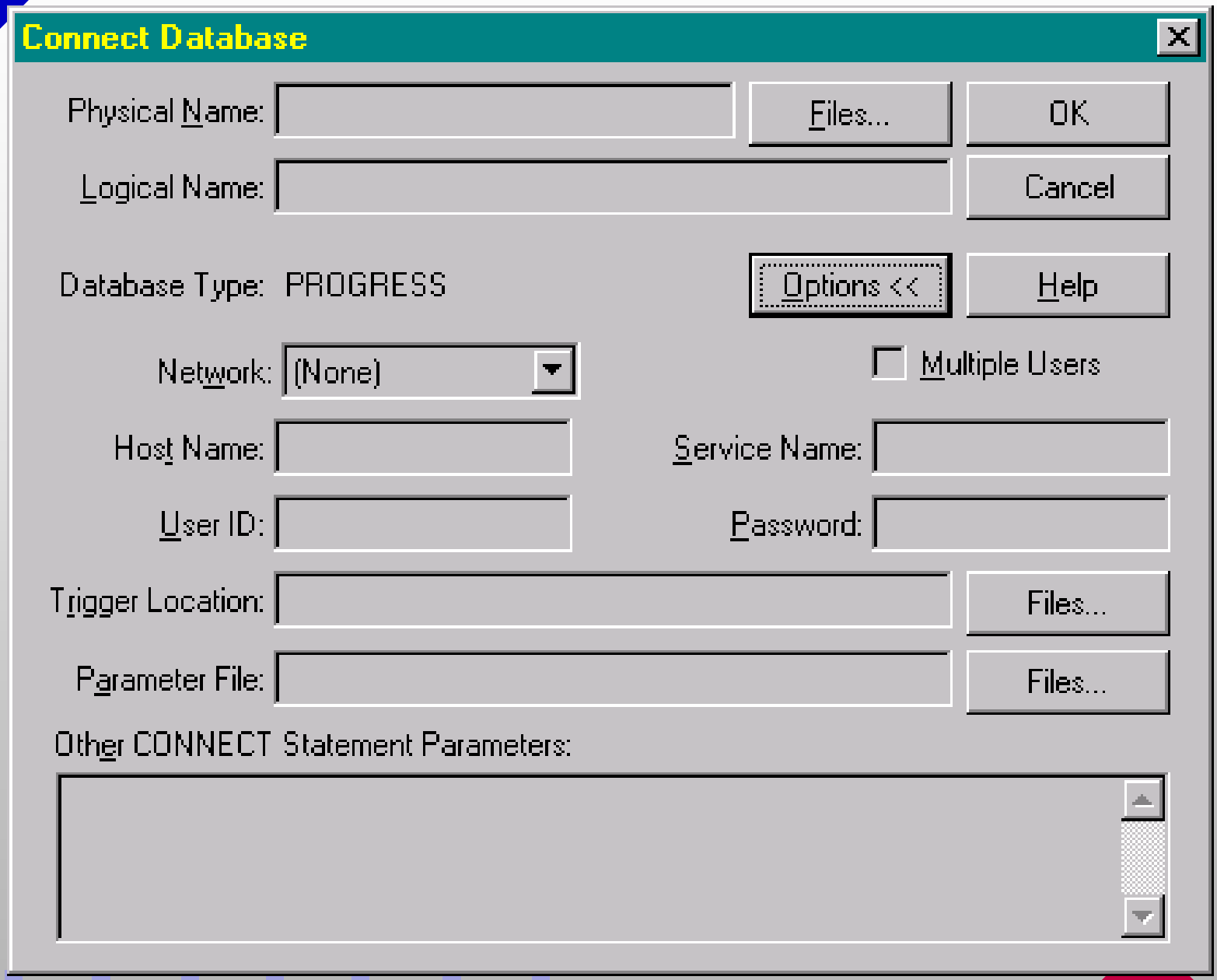
1) Data Dictionary
2) Procedure Editor
3) User Interface Builder (UIB)
4) Results (Não utilizado)
5) Reporter Builder (Não utilizado)
6) Application Debugger

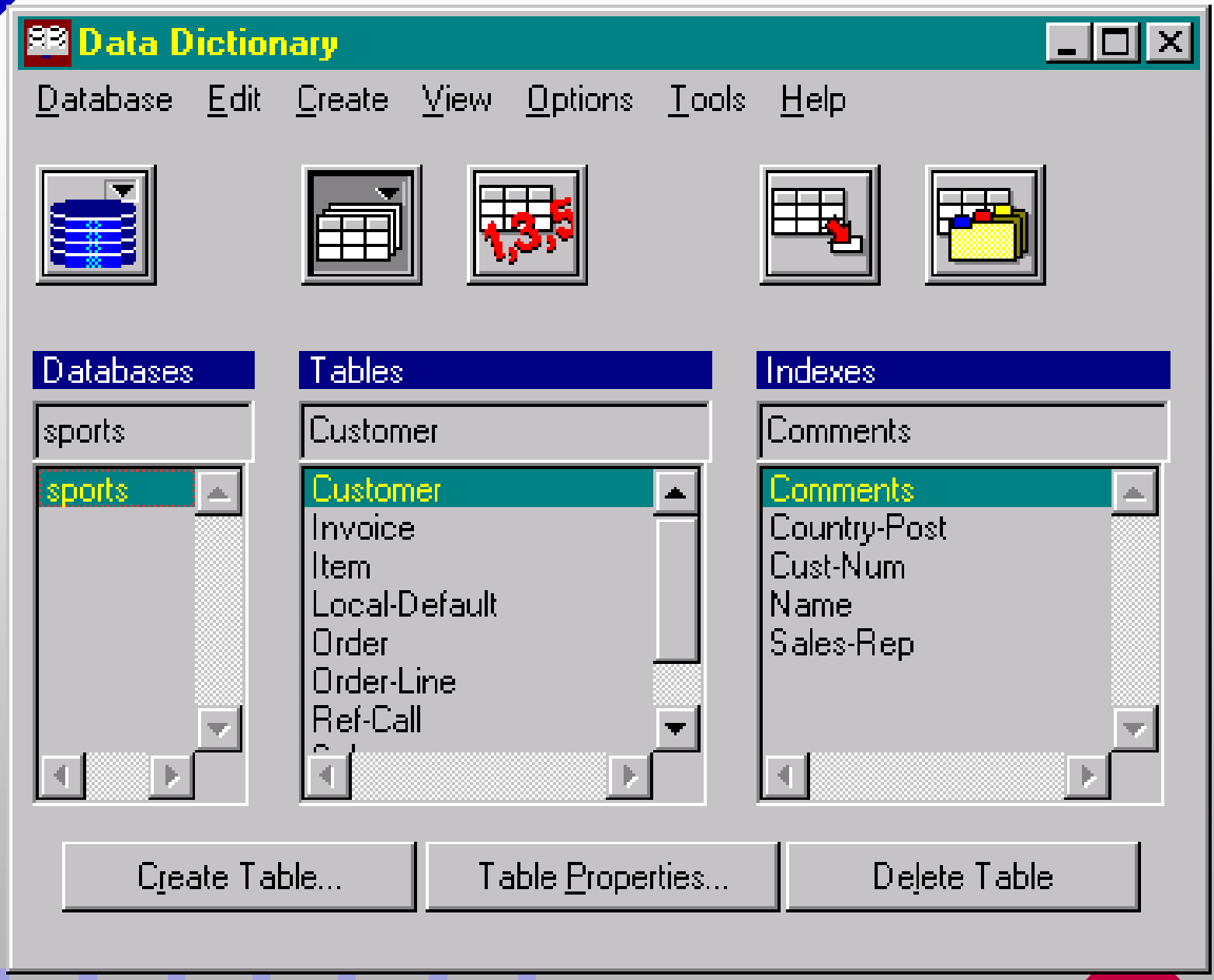
Acessando o Dicionário Progress

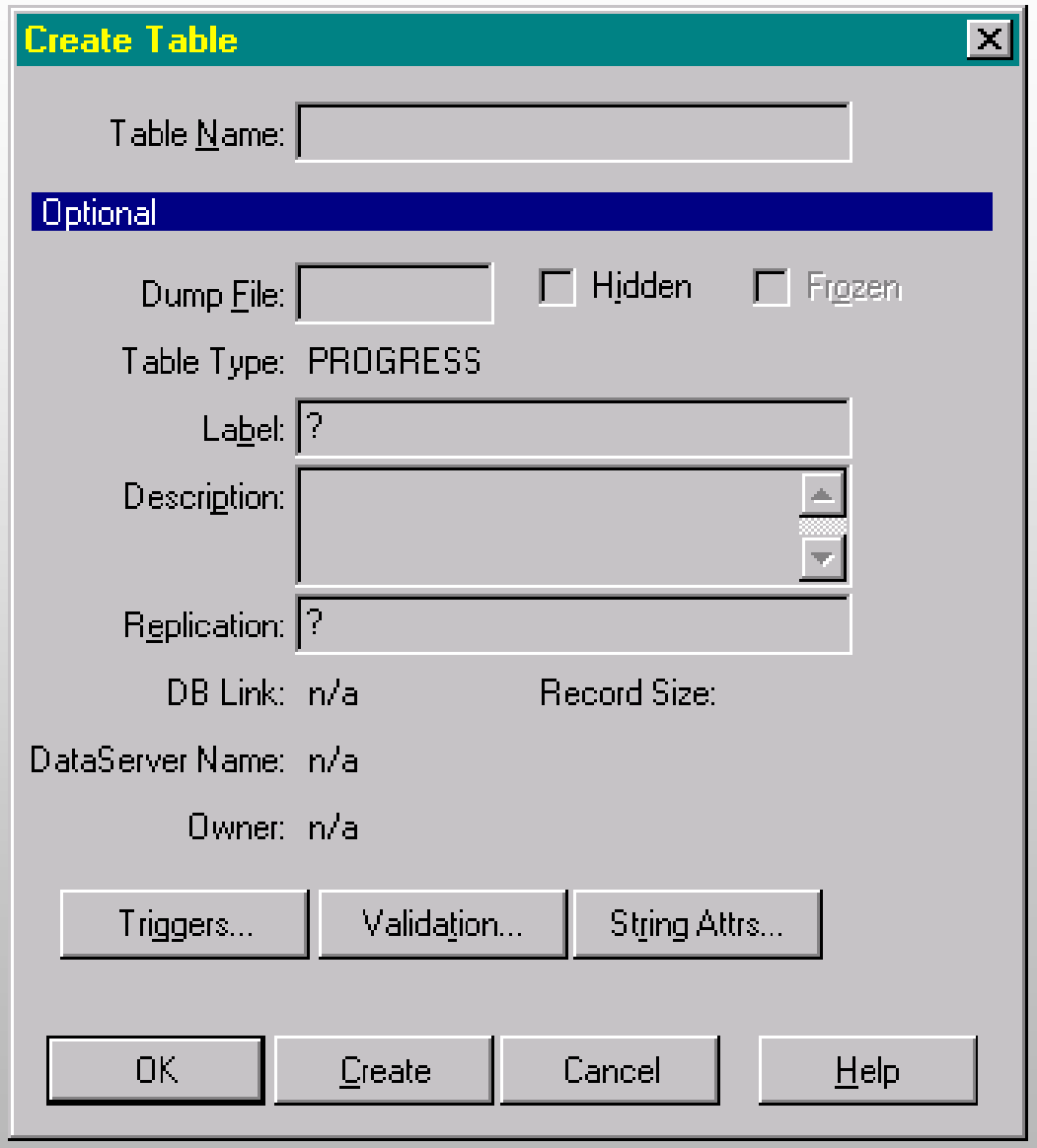
Criando/Conectando Banco de Dados

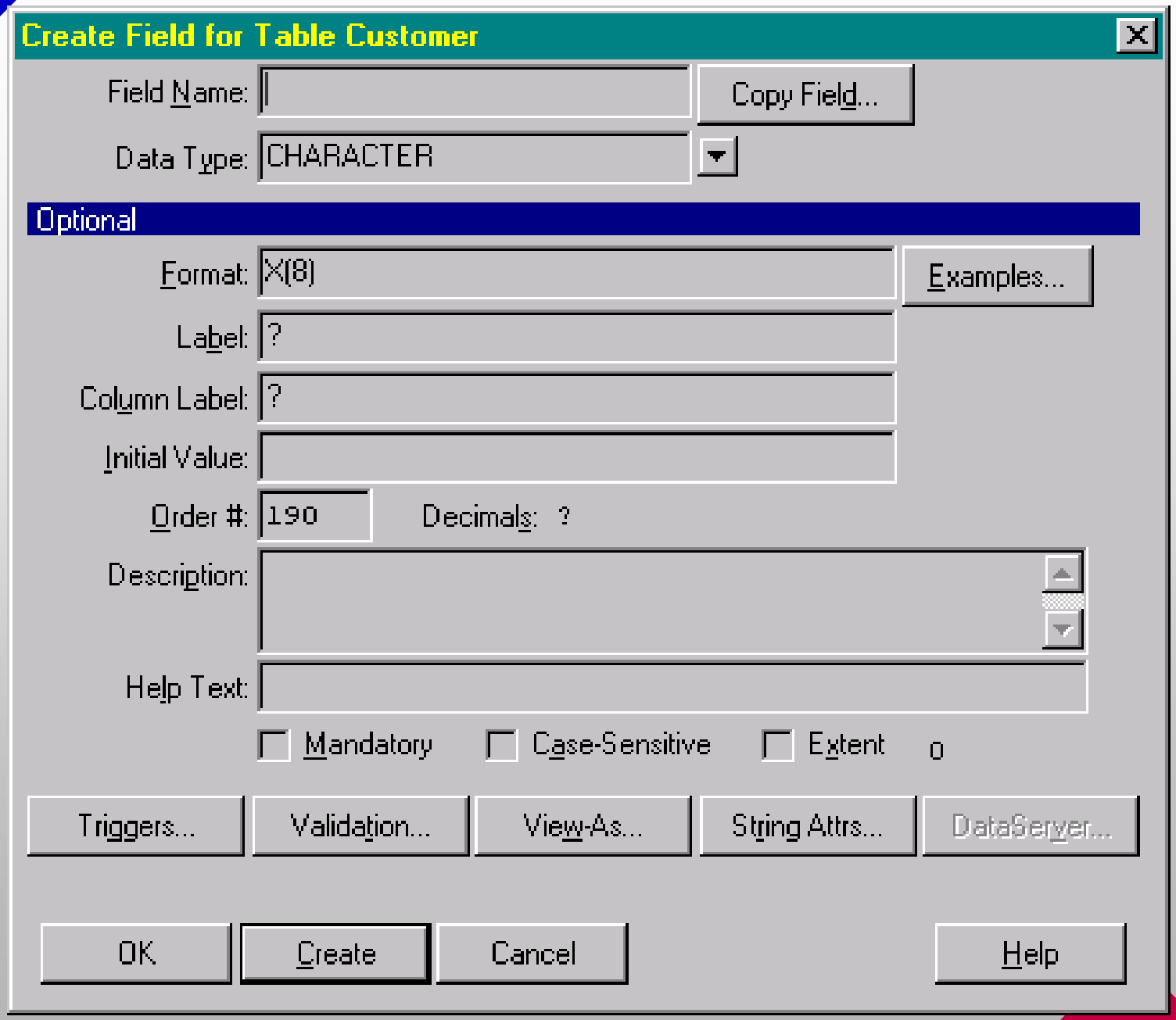



Create Database
New Physical Database Name:
Files...
(.db extension can be omitted)
Start with:
An EMPTY Database
A Copy of the SPORTS Database
A Copy of Some Other Database
Files...
Replace If Exists
OK
Cancel
Help


Connect Database
Physical Name:
Files...
OK
Logical Name:
Cancel
Database Type: PROGRESS
Options <<
Help
Network: (None)
Multiple Users
Host Name:
Service Name:
User ID:
Password:
Trigger Location:
Files...
Parameter File:
Files...
Other CONNECT Statement Parameters:


Data Dictionary
Database Edit Create View Options Tools Help
1,3,5
Databases
Tables
Indexes
sports
Customer
Comments
sports
Customer
Invoice
Item
Local-Default
Order
Order-Line
Ref-Call
Comments
Country-Post
Cust-Num
Name
Sales-Rep
Create Table...
Table Properties...
Delete Table

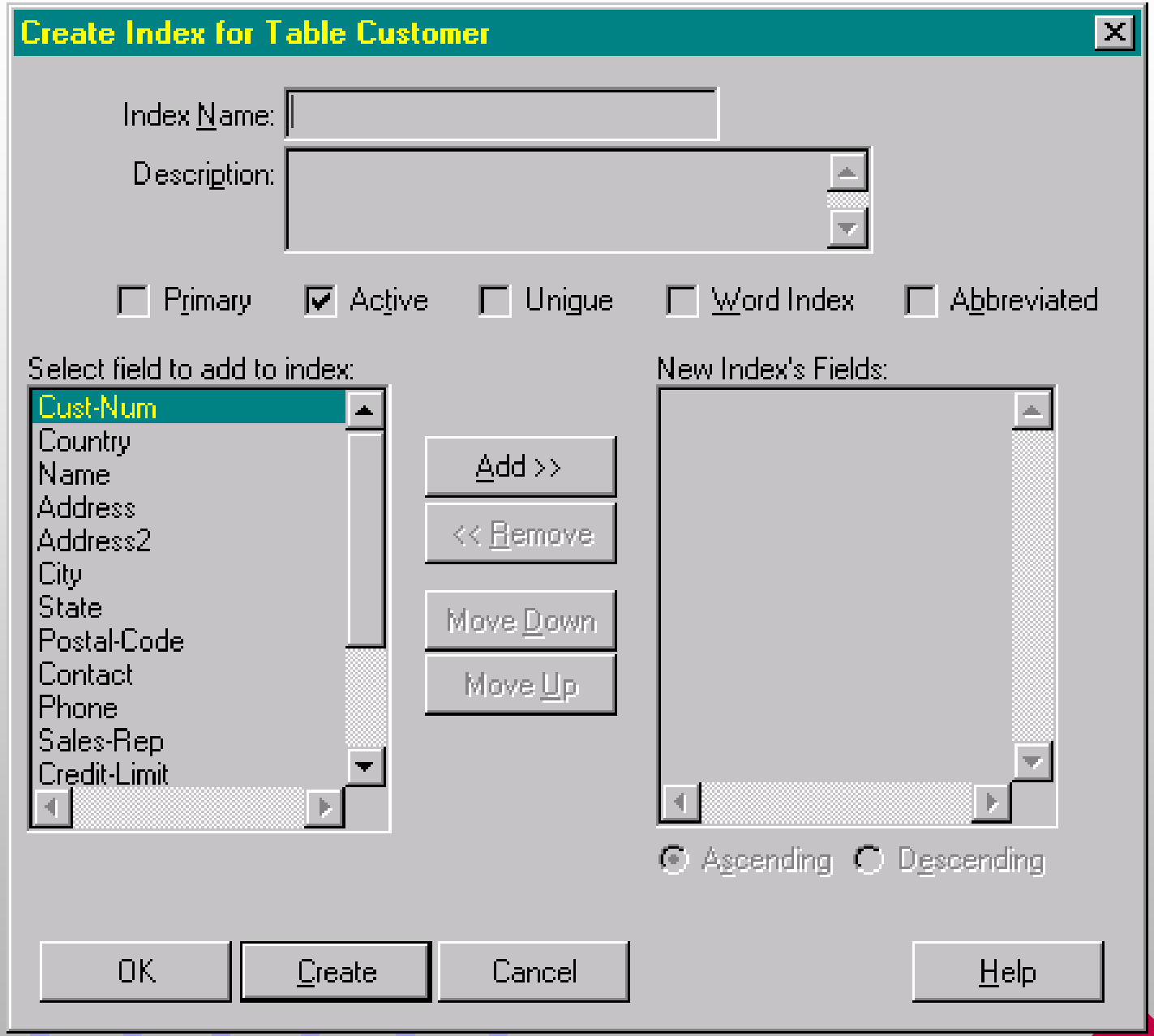

Create Table
Table Name:
Optional
Dump File:
Hidden
Frozen
Table Type: PROGRESS
Label: ?
Description:
Replication: ?
DB Link: n/a
Record Size:
DataServer Name: n/a
Owner: n/a
Triggers...
Validation...
String Attrs...
OK
Create
Cancel
Help


Create Field for Table Customer
Field Name:
Copy Field...
Data Type:
CHARACTER
Optional
Format:
X(8)
Examples...
Label:
?
Column Label:
?
Initial Value:
Order #:
190
Decimals:
?
Description:
Help Text:
Mandatory
Case-Sensitive
Extent
0
Triggers...
Validation...
View-As...
String Attrs...
DataServer...
OK
Create
Cancel
Help

Índices

Razões para definição de índice:

- **Retorno rápido do registro;**
- **Ordenação automática dos registros;**
- **Rápido processamento entre arquivos;**
- **Força a unicidade.**


Create Index for Table Customer
Index Name:
Description:
Primary
Active
Unique
Word Index
Abbreviated
Select field to add to index:
Cust-Num
Country
Name
Address
Address2
City
State
Postal-Code
Contact
Phone
Sales-Rep
Credit-Limit
Add >>
<< Remove
Move Down
Move Up
New Index's Fields:
Ascending
Descending
OK
Create
Cancel
Help

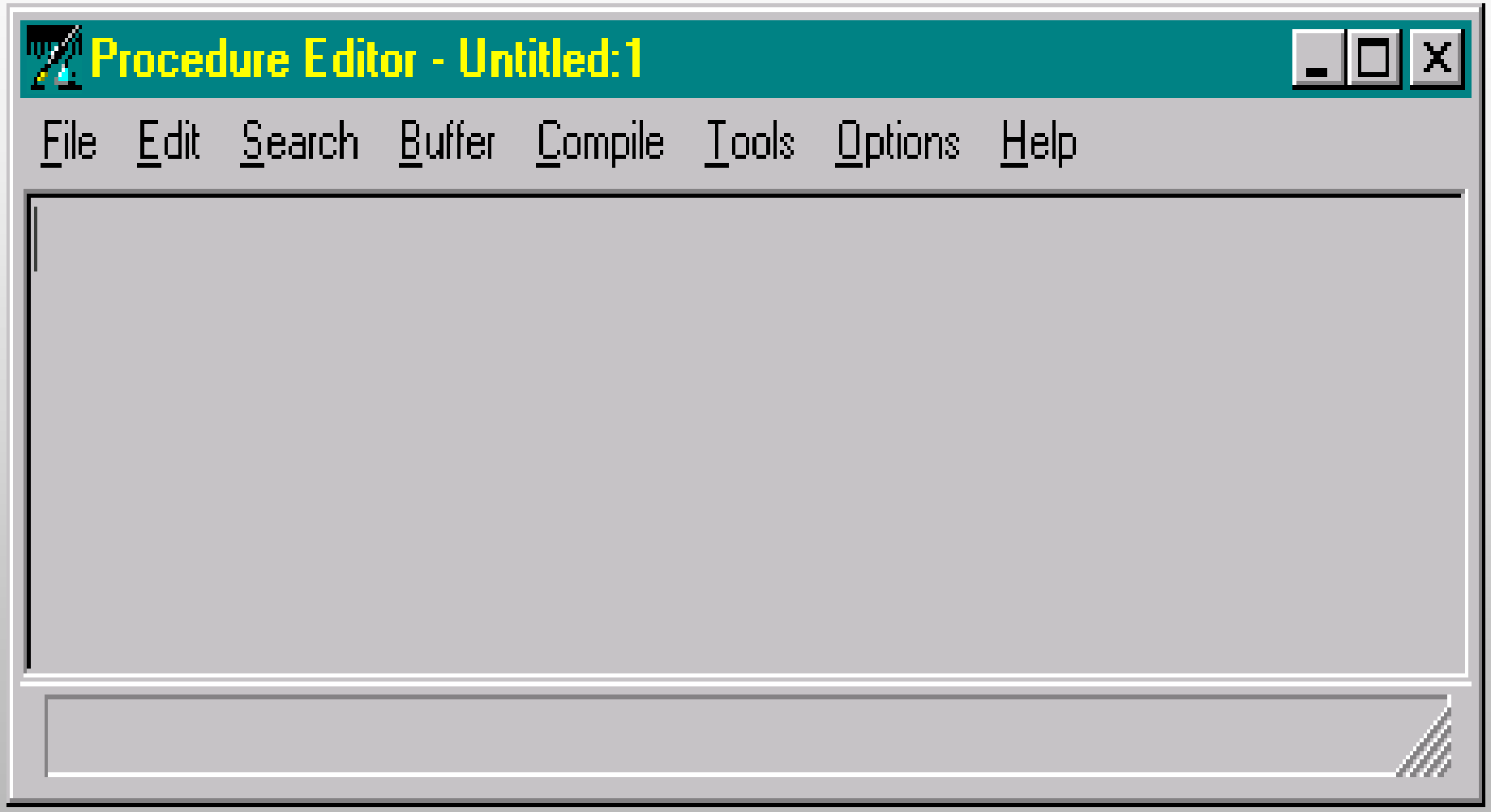
LABORATÓRIO 1

Criar a tabela CLIENTES com as seguintes características:

Atributos	Tipo	Formato	Label
Código	integer	>>>9	Código
Nome	character	x(40)	Nome
Endereco	character	x(40)	Endereço
Bairro	character	x(20)	Bairro
Cidade	character	x(20)	Cidade
CEP	character	99.999-999	CEP
Estado	character	x(2)	UF

Índice: Código -> Atributo: Código -> Primário e Único

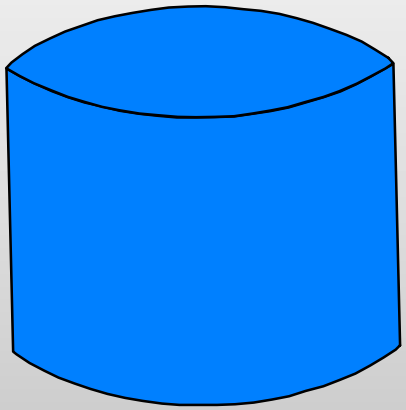
Acessando o Procedure Editor



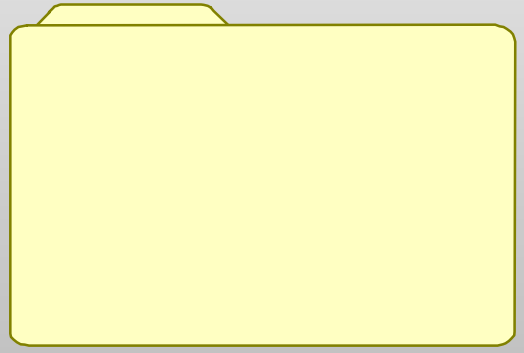
Componentes da Linguagem

- Blocos
- Declarações
- Funções
- Operadores
- Variáveis
- Expressões
- Símbolos especiais
- Elementos da interface com o usuário (atributos e métodos)
- Eventos

Localização dos Dados



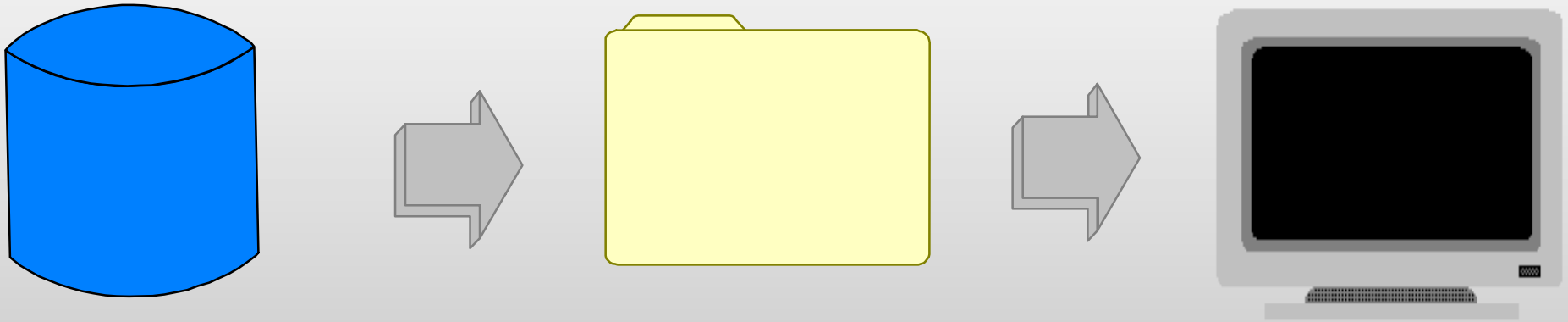
Base de Dados



Buffer Registro

Buffer tela

Movimentando dados
Base de Dados → Tela

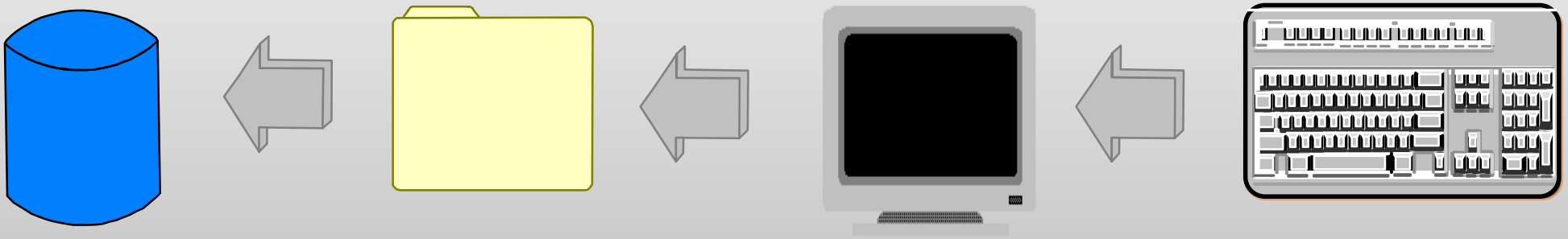


Base de Dados

Buffer Registro

Buffer tela

Movimentando dados
Tela → Base de Dados



Base de Dados

Buffer Registro

Buffer Tela

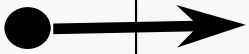
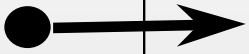
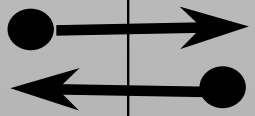
Entrada Usuário

Regras para Movimentação de Dados

- Os programas não podem manipular os dados da base de dados até que eles sejam copiados para o buffer de registro;
- O usuário não pode interagir com os dados até que eles sejam copiados do buffer de registro para o buffer de tela;

Regras para Movimentação de Dados

- As alterações que são feitas no buffer de tela ou buffer de registro não são atualizadas automaticamente em outro buffer. Deve-se programar o controle de movimentação entre os buffers.

Comando	Banco de Dados	Buffer de Registro	Buffer de Tela	Usuário
ASSIGN				
CREATE				
DELETE				
DISPLAY				
ENABLE				
FIND				
FOR EACH				
GET				
INSERT				
PROMPT-FOR				
RELEASE				
SET				
UPDATE				

Criando e Modificando Registros

CREATE - Cria um novo registro no banco, inicializa com os valores defaults e deixa uma cópia do registro no buffer de registro.
Ex.: CREATE customer.

UPDATE - Executa as ações de DISPLAY, PROMPT-FOR e ASSIGN.
Ex.: UPDATE customer.
UPDATE customer EXCEPT cust-num.

Criando e Modificando Registros

INSERT - Executa as ações de CREATE, DISPLAY, PROMPT-FOR e ASSIGN.
Ex.: INSERT customer.

ASSIGN - Copia os dados do buffer de tela para o buffer de registro.
Ex.: ASSIGN customer.cust-num.

Criando e Modificando Registros

SET - Executa as ações de PROMPT-FOR e ASSIGN.
Ex.: SET customer.

DELETE - Elimina um registro do buffer de registro e do banco de dados.
Ex.: DELETE customer.

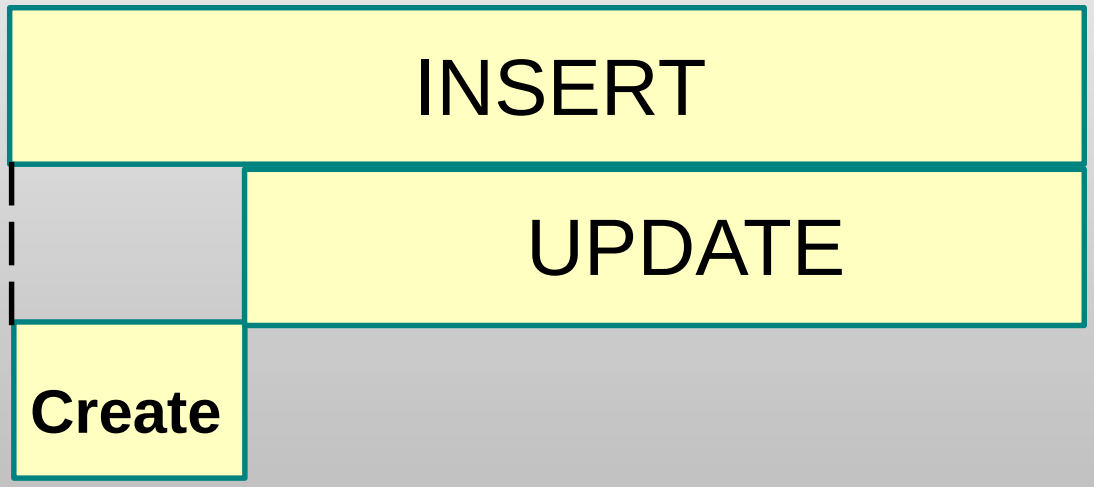
Criando e Modificando Registros

ASSIGN - Copia os dados do buffer de tela para seus respectivos campos e/ou variáveis no buffer de registro.

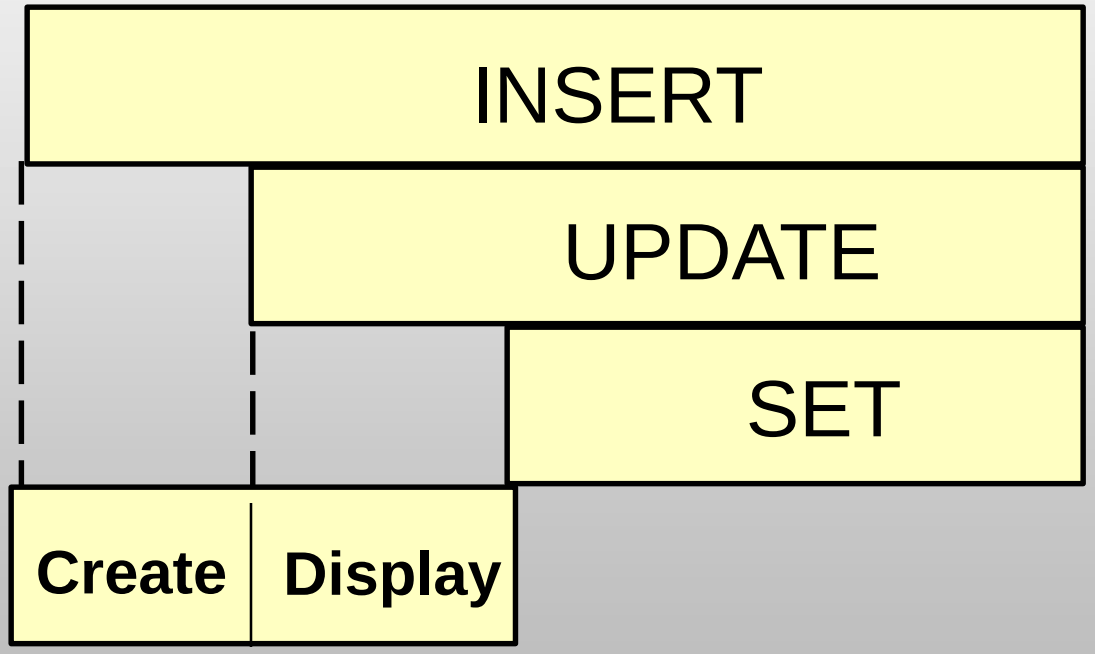
Ex.: 1. ASSIGN customer.cust-num.

2. DEF VAR i-cust-num AS integer NO-UNDO.
   PROMPT-FOR i-cust-num.
   FIND FIRST customer EXCLUSIVE-LOCK.
   ASSIGN customer.cust-num = i-cust-num.

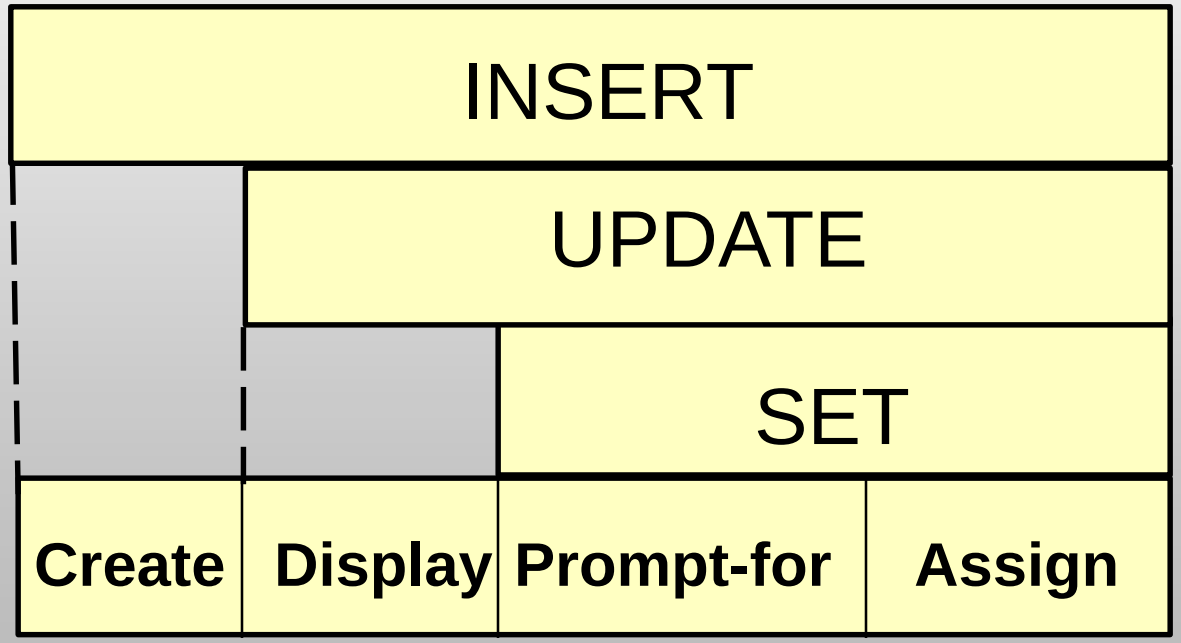
Comandos que substituem Comandos



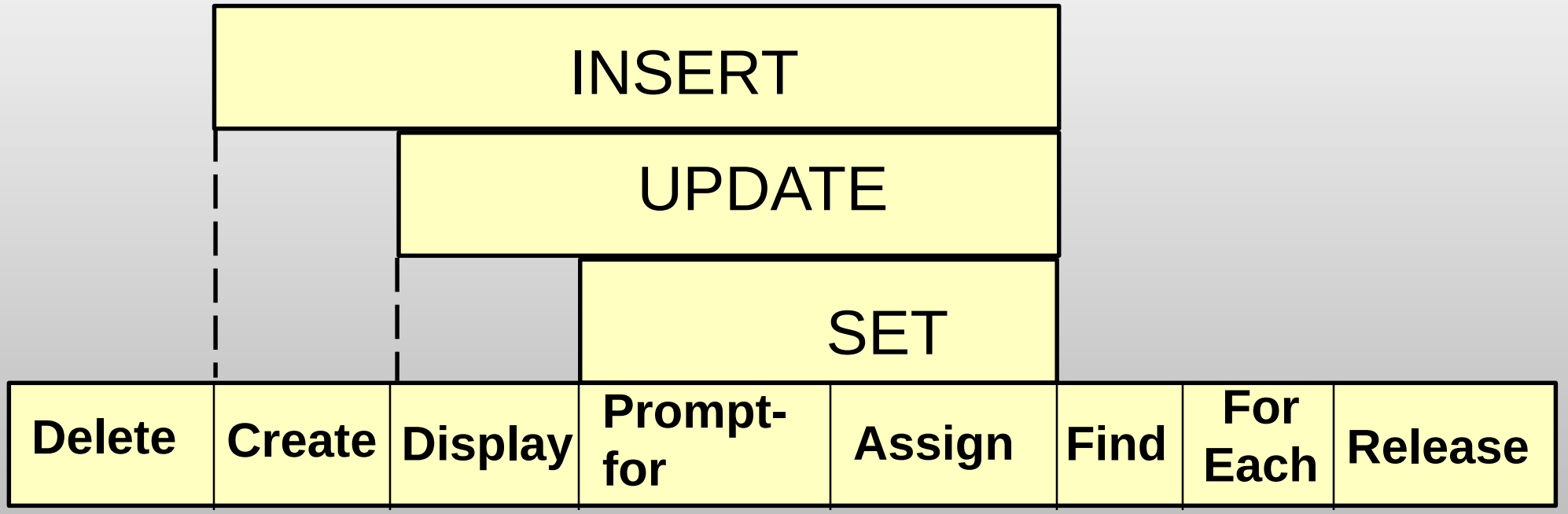
Comandos que substituem Comandos



Comandos que substituem Comandos



Quadro de Comandos



Tipos de Dados do Progress

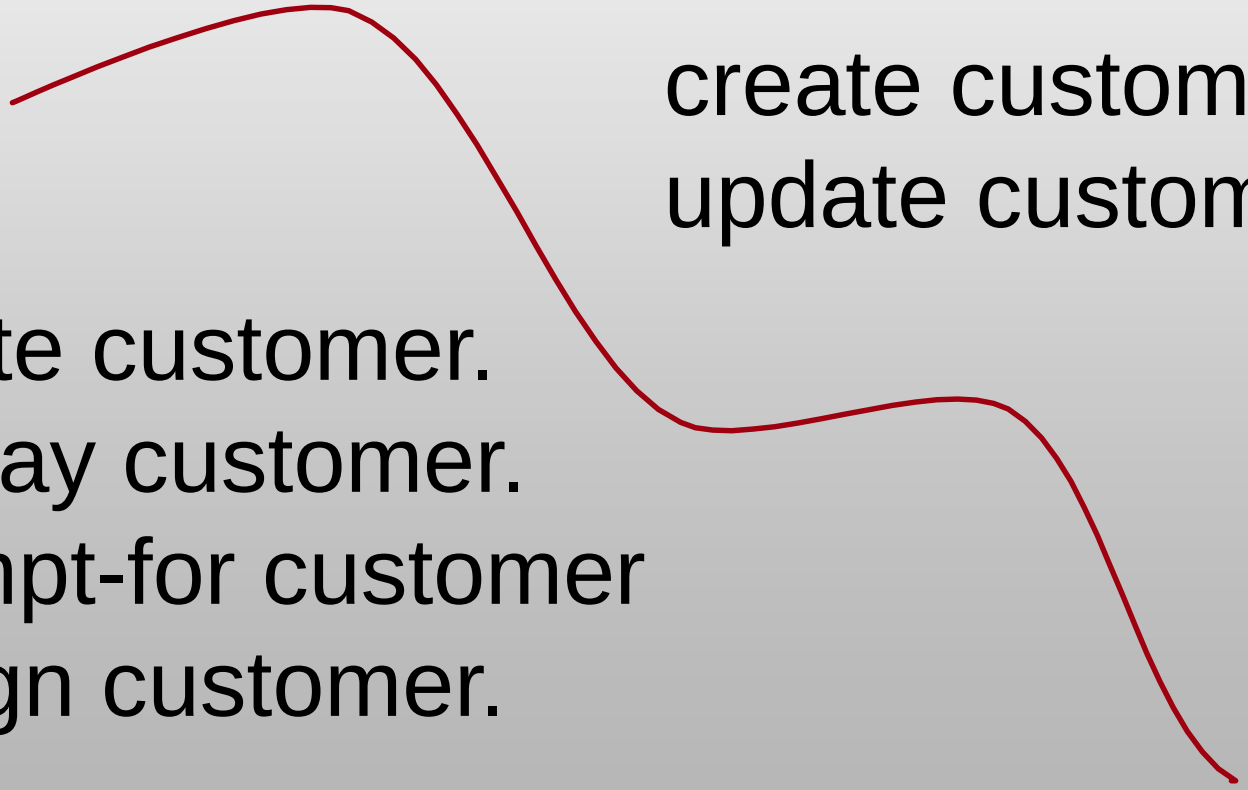
Tipo	Formato	Exemplo
Character	x(8) Aceita: A – Alfabetico ! – Maiusculas	“x(40)” –> string alfanumérica “AAA-9999” –> MCZ-9283 “!x(10)” –> STRING ALFANUMÉRICA
Integer	>>>,>>9 Aceita: z – espaço 9 – mostra “0”	“>>>,>>9” –> 1.500 “999,999” –> 001.500 “zzz,zz9” –> 1.500
Decimal	>>>,>>9.99	“>>>,>>9.99” –> 1.500,45 “999,999.99” –> 001.500,45 “zzz,zz9.99” –> 1.500,45
Logical	Yes/No Aceita: True/False	Yes No
Date	MM/DD/AAA	
Rowid	----	
Recid	>>>>>9	

Observe a movimentação

insert customer.

create customer.
update customer.

create customer.
display customer.
prompt-for customer
assign customer.



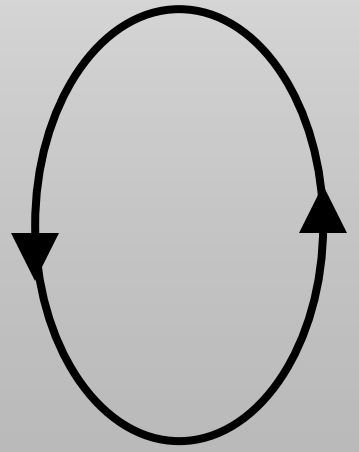
Tipos de Blocos

- REPEAT
- FOR EACH
- DO
- PROCEDURES
- TRIGGERS

REPEAT

Laço (repetição) automático. Continua a interagir até encontrar um END-ERROR ou outra condição de término definida pelo usuário.

Ex: REPEAT WITH 1 COLUMN:
    CREATE customer.
    UPDATE customer.
END.



FOR EACH

Lê um registro a cada interação do bloco e copia os dados do banco para o buffer de registro, e do buffer de registro para o buffer de tela.

Ex:
```
FOR EACH customer:
    DISPLAY cust-num.
END.
```

DO

Individualiza um grupo de comandos dentro de um bloco simples.

```
Ex:  FOR  EACH customer:
          DISPLAY cust-num.
         IF  credit-limit  > 5000 THEN  DO:
             UPDATE  name  address.
             MESSAGE "Registro Alterado".
          END.
     END.
```

FIND

Busca apenas um registro em uma tabela.

FIRST (primeiro) LAST (último)

NEXT (próximo) PREV (anterior)

Ex.: FIND FIRST customer.

DISPLAY customer.

PROMPT-FOR

Solicita uma entrada e disponibiliza no buffer de tela.

Ex.: PROMPT-FOR customer.cust-num.
FIND customer
WHERE customer.cust-num = INPUT
customer.cust-num.
DISPLAY customer WITH 1 COLUMN.

Substituindo "WHERE"

```
PROMPT-FOR customer.cust-num.
FIND customer USING customer.cust-num.
DISPLAY customer WITH 1 COLUMN.
```

```
WHERE customer.cust-num = INPUT customer.cust-num.
```

REPEAT

Laço (repetição) automático. Continua a interagir até encontrar um END-ERROR ou outra condição de término definida pelo usuário.

Ex:
```
REPEAT WITH 1 COLUMN:
    PROMPT-FOR customer.cust-num.
    FIND customer USING customer.cust-num.
    DISPLAY customer.
END.
```

BEGINS

Verifica se a 1. expressão inicia com a 2. expressão.

```
Ex:  FOR  EACH customer
          WHERE customer.name BEGINS “S”:
          DISPLAY customer.cust-num
                  customer.name
                  customer.phone.
     END.
```

MATCHES

Compara uma expressão do tipo caracter com um padrão e retorna o valor verdadeiro se a expressão for atendida.

Ex: FOR EACH customer
WHERE customer.address MATCHES “*st”:
DISPLAY cust-num name address.
END.

CONTAINS

Faz a busca por determinada palavra ou combinação delas, em atributos do tipo word-index.

```
Ex:  FOR  EACH item
          WHERE item-name CONTAINS “ball”:
          DISPLAY item-num item-name price.
     END.
```

QUERY

Busca registros no banco de dados. É utilizada com widgets browses ou declaração GET.

- Definir a query
Ex.: DEFINE QUERY query-name FOR table
- Abrir a query
Ex.: OPEN QUERY query-name FOR EACH table
- Acessar os registros
GET ou BROWSE

GET

Retorna um registro de uma query previamente aberta.

```
GET   FIRST query-name.
      NEXT
      LAST
      PREV
```

BROWSE

Interface para a query.

Ex:

```
DEFINE QUERY query-name FOR customer.
DEFINE BROWSE browse-name QUERY query-name
    DISPLAY customer.cust-num
            customer.name
     WITH 15 DOWN TITLE “Browse de Clientes”.
OPEN QUERY query-name FOR EACH customer.
UPDATE browse-name WITH FRAME f1.
```

BY

Usa-se a opção BY para classificação de registros por um campo não indexado.

Ex:

```
FOR EACH customer BY balance DESCENDING:
        DISPLAY balance name phone.
END.
```

Default : ordenação ascendente

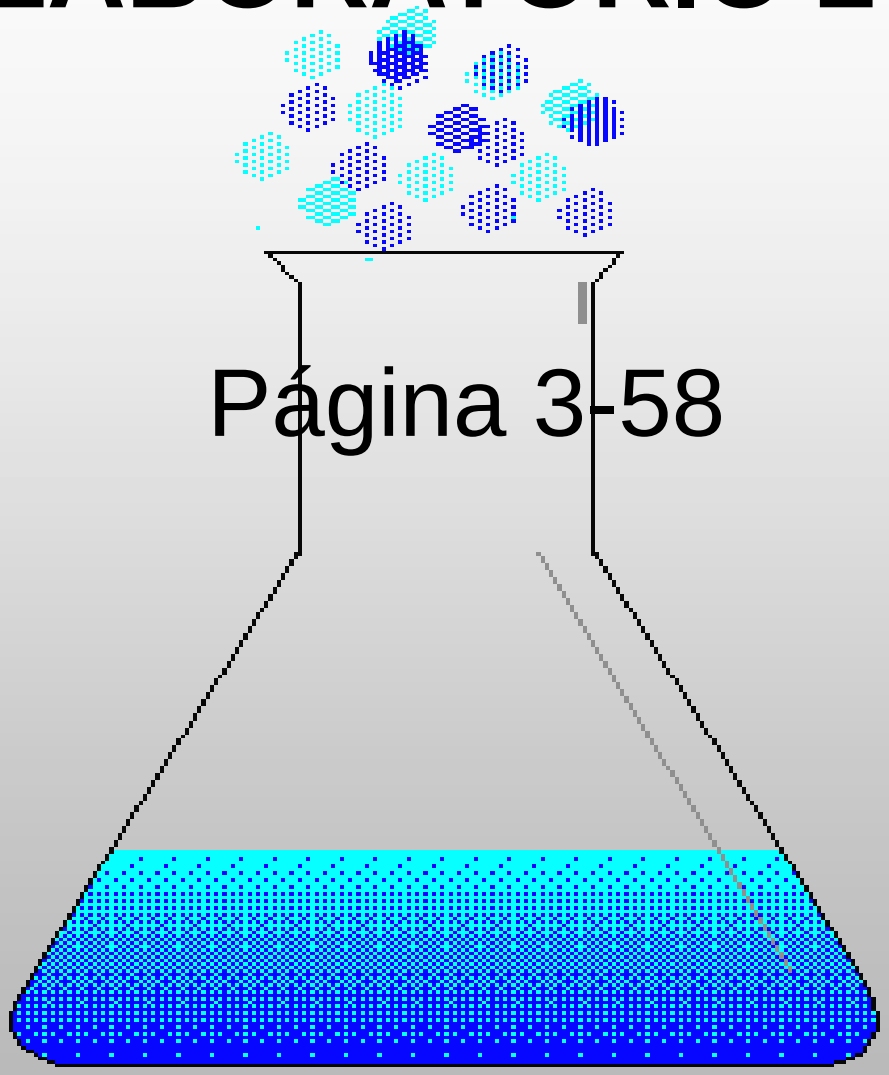
USE-INDEX

A opção USE-INDEX permite você escolher qualquer índice definido, para selecionar registros.

Ex:

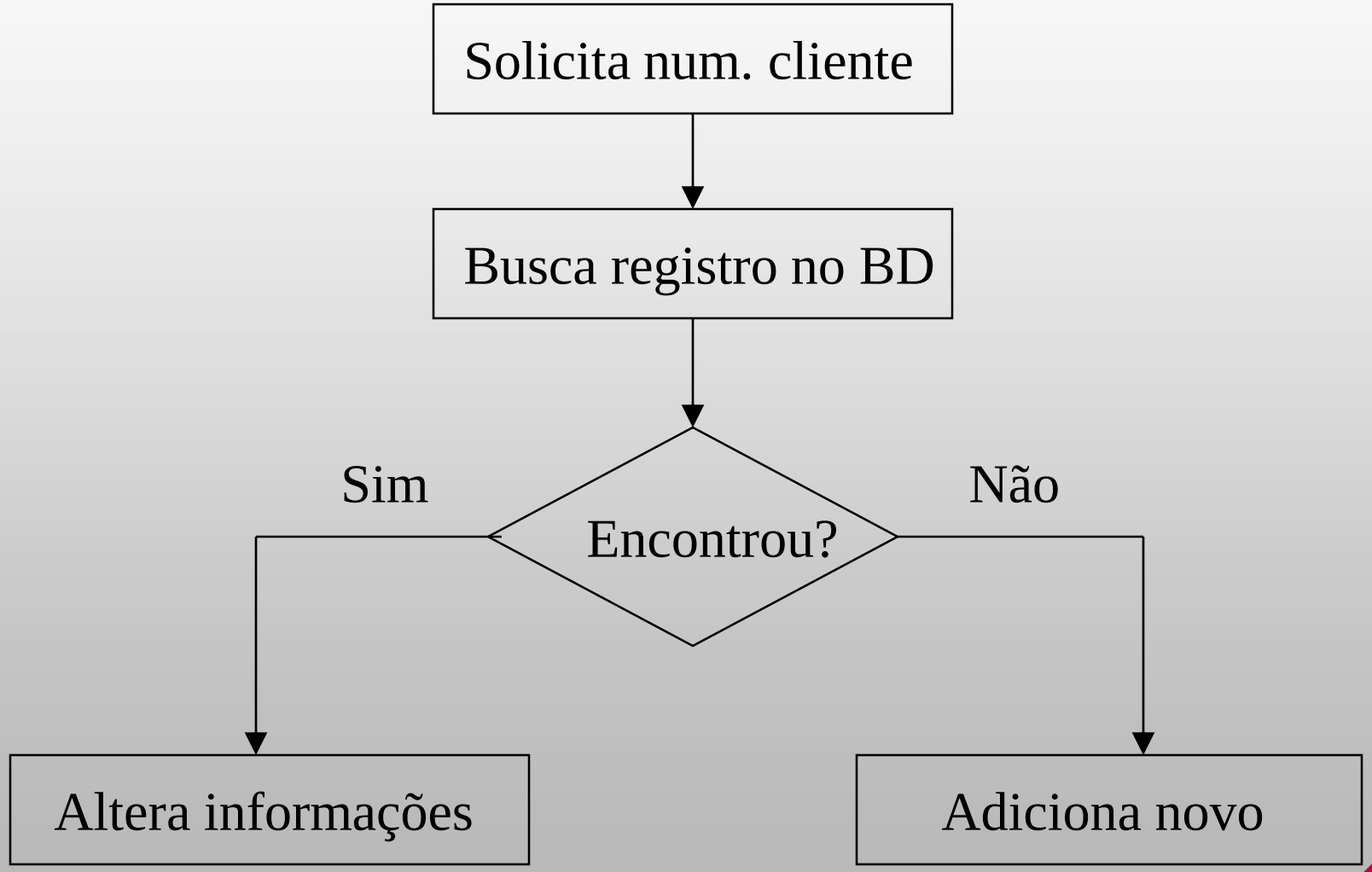
```
FOR EACH customer USE-INDEX name:
        DISPLAY balance name phone.
END.
```

LABORATÓRIO 2



Página 3-58

Processamento Condicional



Processamento Condicional

```
IF expression THEN
    bloco
ELSE DO:
    bloco
END.
```

Processamento Condicional

```
CASE expression:
    WHEN value THEN
        bloco
    WHEN value THEN DO:
        bloco
    END.
    WHEN value THEN
       bloco
    OTHERWISE
END CASE.
```

Eliminando Registros

Elimina o registro da tabela onde o ponteiro estiver posicionado.

Ex:

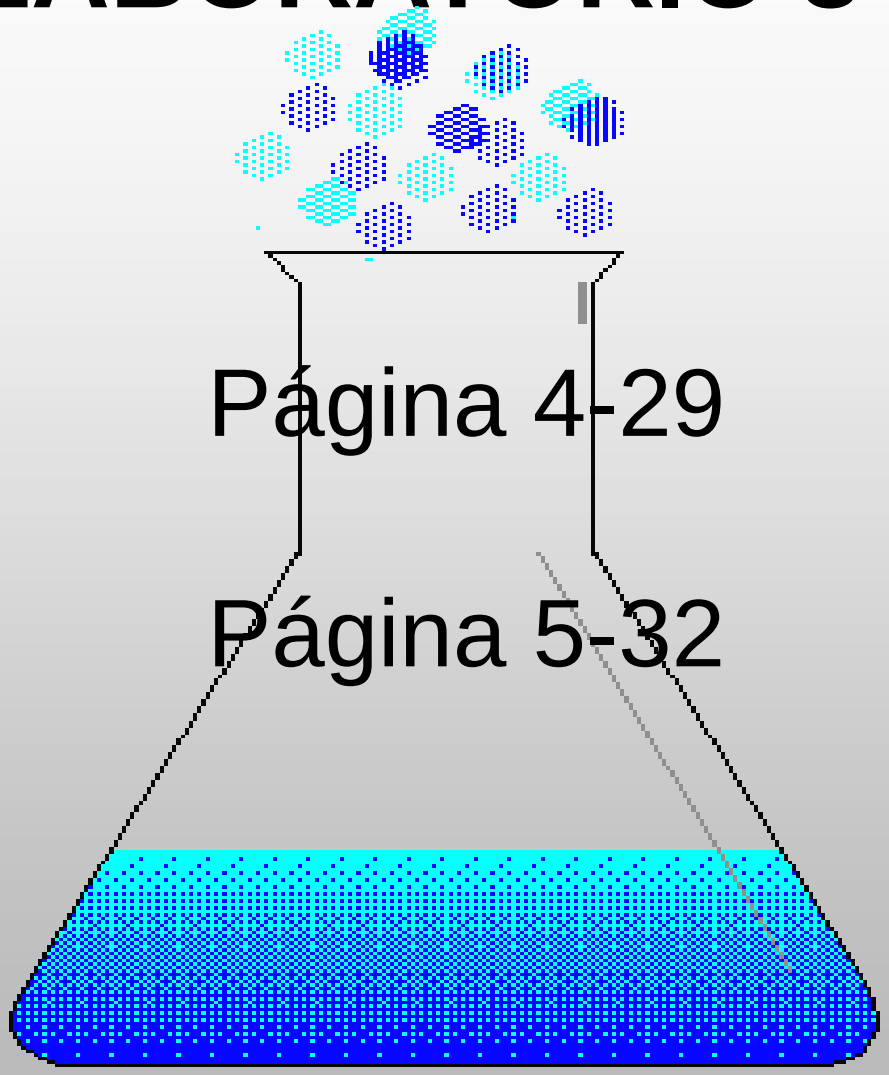
```
REPEAT:
    PROMPT-FOR customer.cust-num.
    FIND customer USING cust-num.
   DELETE customer.
END.
```

Variáveis

DEFINE VARIABLE variable-name
AS Tipo de dado
LIKE Tabela.atributo
INITIAL “xxxx”
NO-UNDO

Ex.: DEFINE VARIABLE c-customer AS char NO-UNDO.
DEF VAR i-cust-num LIKE customer.cust-num
NO-UNDO.

LABORATÓRIO 3



Página 4-29

Página 5-32

FRAMES

É uma área retangular dentro da tela do Progress usada para mostrar informações.

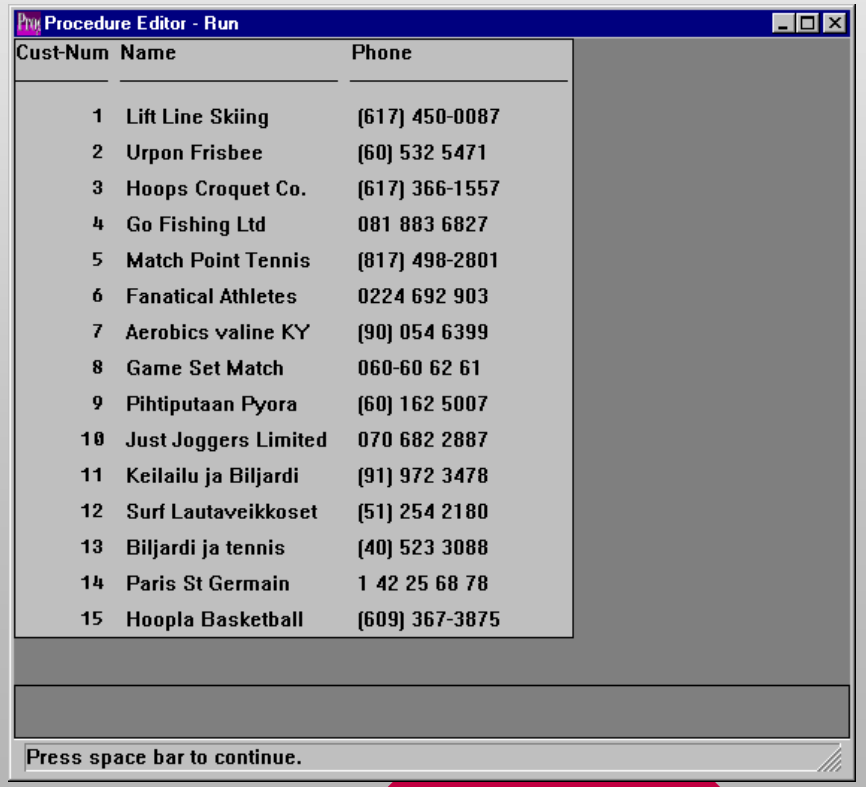
Layout default para seus dados.

Espaço disponível:

Frame (21 linhas)

Mensagens (2 linhas)

Linha de status (1 linha)



FRAMES

As frames default do progress ocorrem nos seguintes blocos :

- Procedure
- Repeat
- For each
- Do with frame

FRAMES

```
DISPLAY “Relatório de Clientes”
                WITH FRAME abc CENTERED.
FOR EACH customer:
   DISPLAY cust-num name
                WITH FRAME xxx DOWN.
END.
```

one-down: dados de uma única interação do bloco

down: dados de múltiplas interações do bloco.

DEFINE FRAME

Define uma frame para uso por um procedimento ou por diversos procedimentos.

Ex.: DEFINE FRAME xxx

```
        customer.cust-num      SKIP(2)
        customer.name     AT 5 SKIP
        customer.address AT 5
        WITH SIDE-LABELS OVERLAY.
```

OPÇÕES DE POSICIONAMENTO

AT - Alinhamento pela direita.

F1: Short
Fld2: Medium
Field3: Long

COLON - Alinhamento pelos dois pontos.

F1: Short
Fld2: Medium
Field3: Long

TO - Alinhamento pela esquerda.

F1: Short
Fld2: Medium
Field3: Long

VIEW - apresenta uma frame.

Ex.: VIEW FRAME frame-name.

PAUSE - suspende a execução até que uma tecla seja pressionada ou um certo tempo passar.

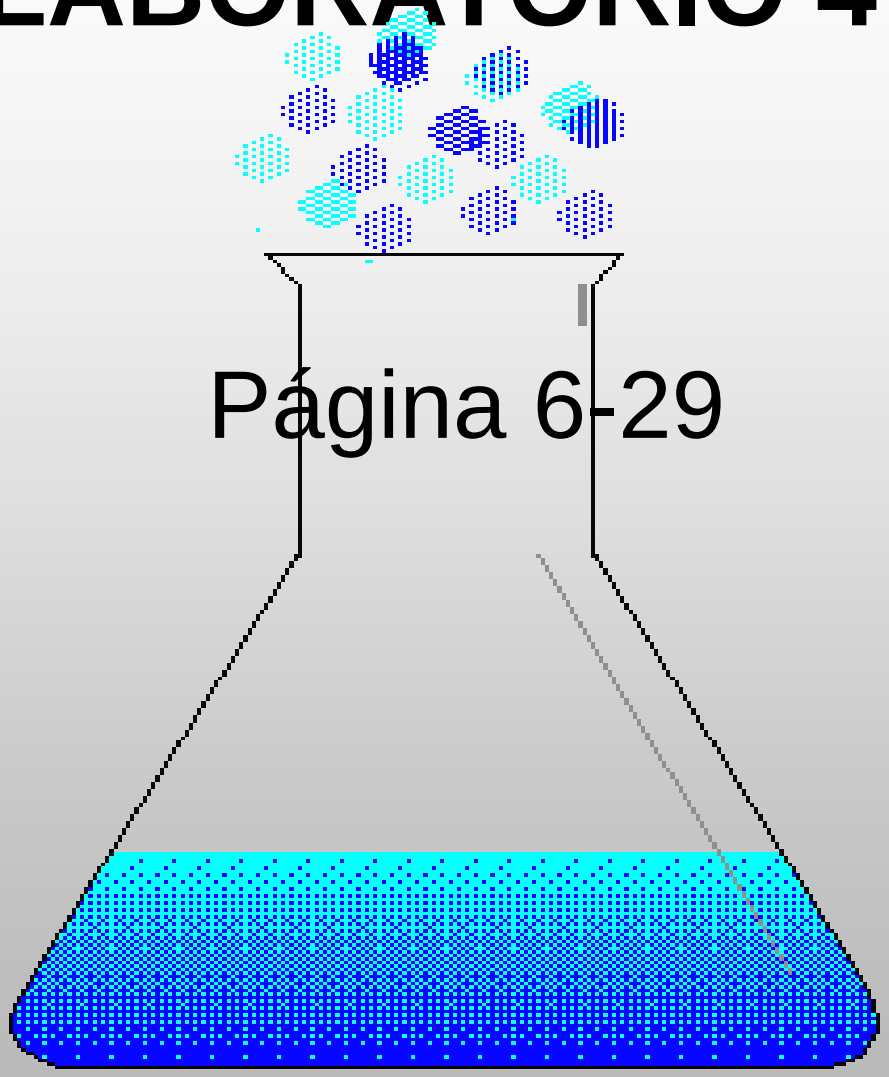
Ex.: PAUSE.

PAUSE 5.

HIDE - apaga uma frame.

Ex.: HIDE FRAME frame-name NO-PAUSE.

LABORATÓRIO 4



Página 6-29

INCLUDES

Em tempo de compilação PROGRESS lê estes arquivos include como sendo parte do programa.

Vantagem: reutilização de código.

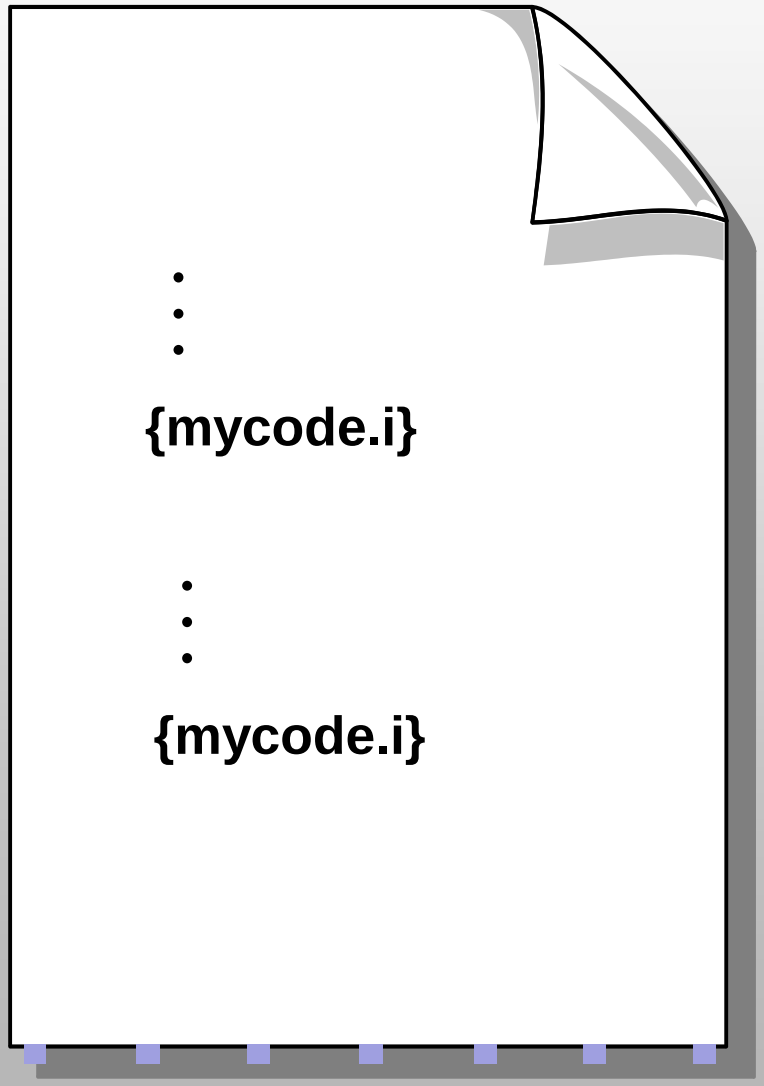
Ex.: {i-valida.i}

{i-valida.i “Código Inválido”}

```
/* i-valida.i */
MESSAGE “{1}” VIEW-AS ALERT-BOX.
```

Código Modular com Arquivos de Includes

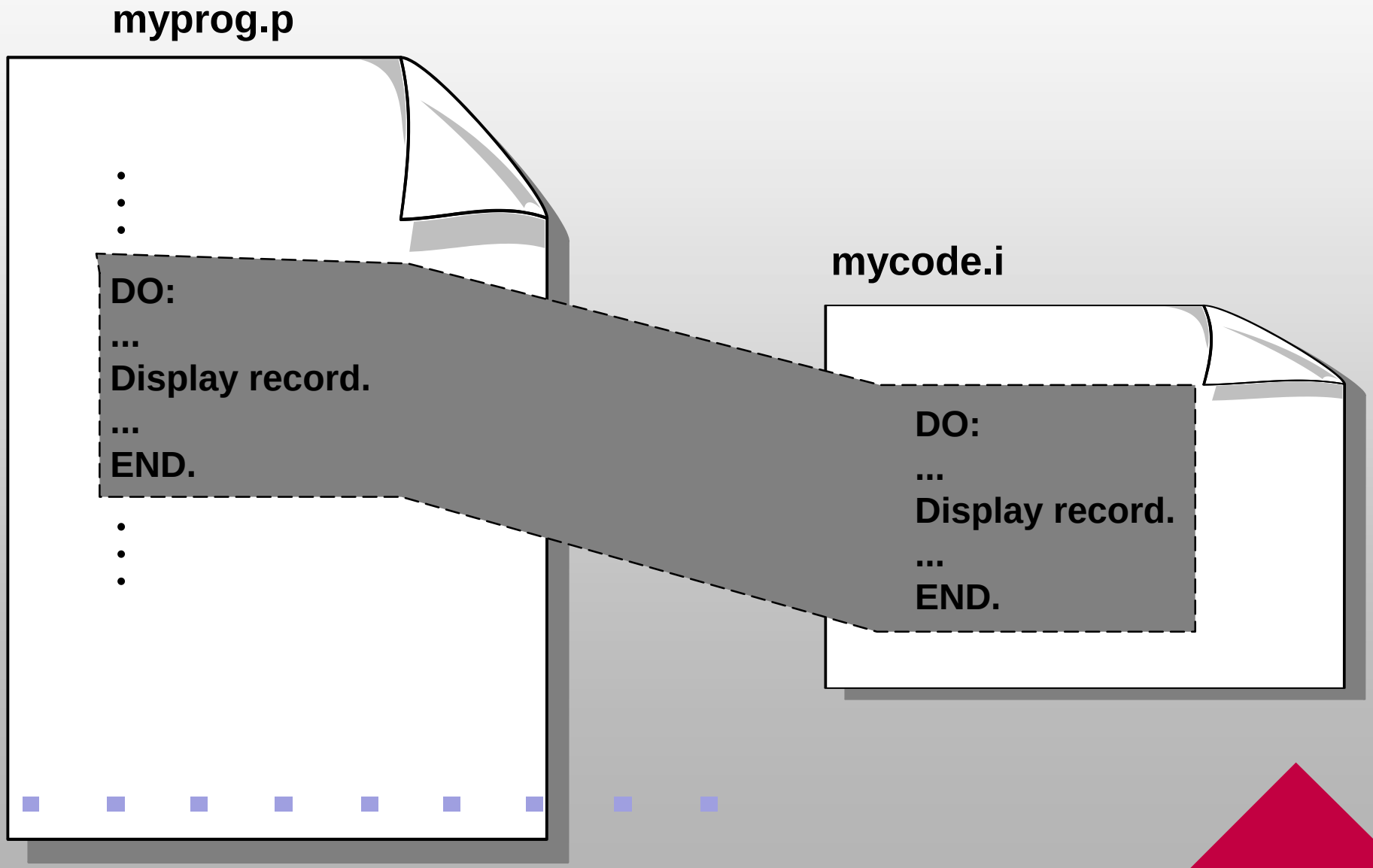
myprog.p



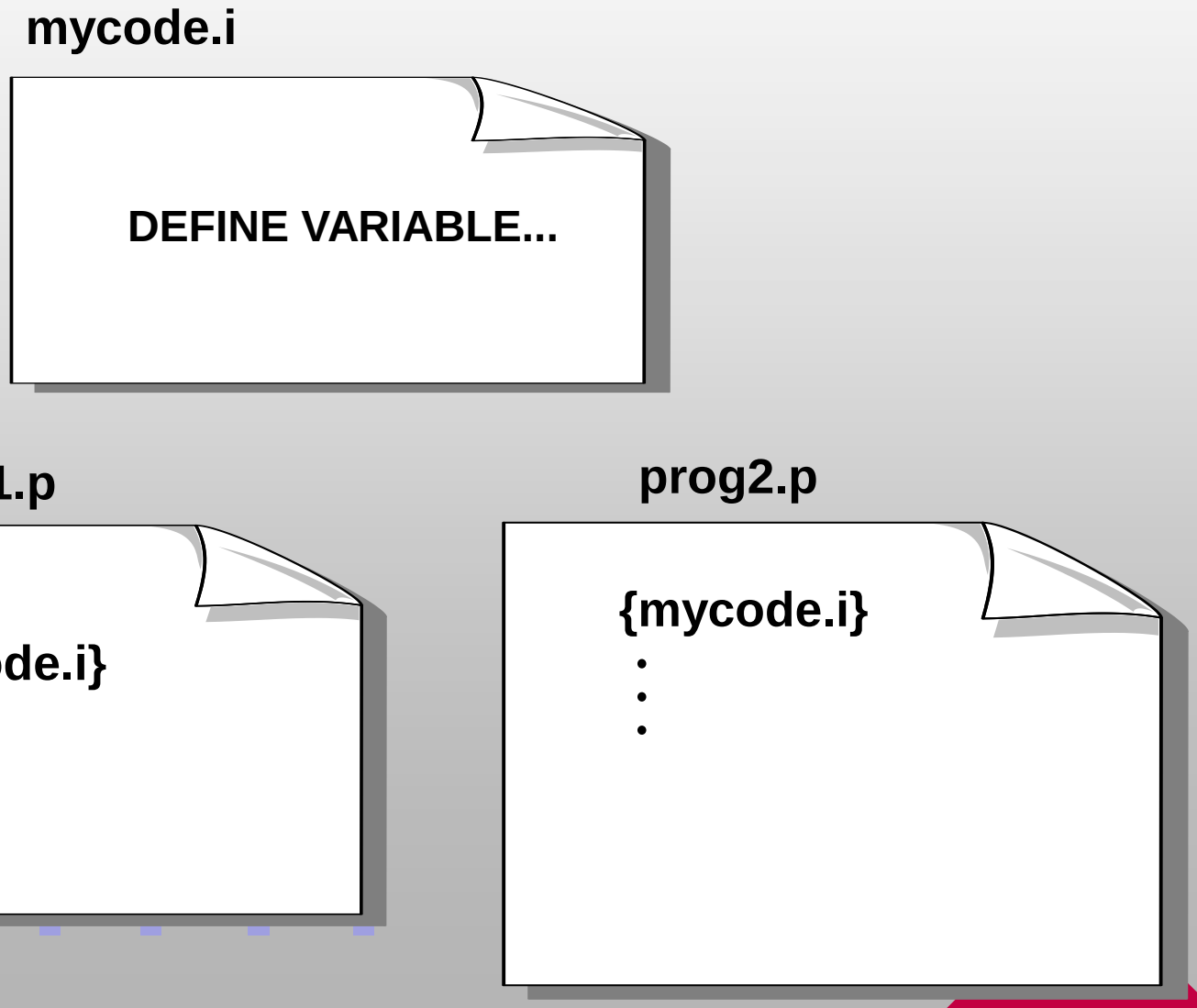
mycode.i

```
DO:
    ...
    Display record.
    ...
END.
```

Includes São Expansões em Tempo de Compilação



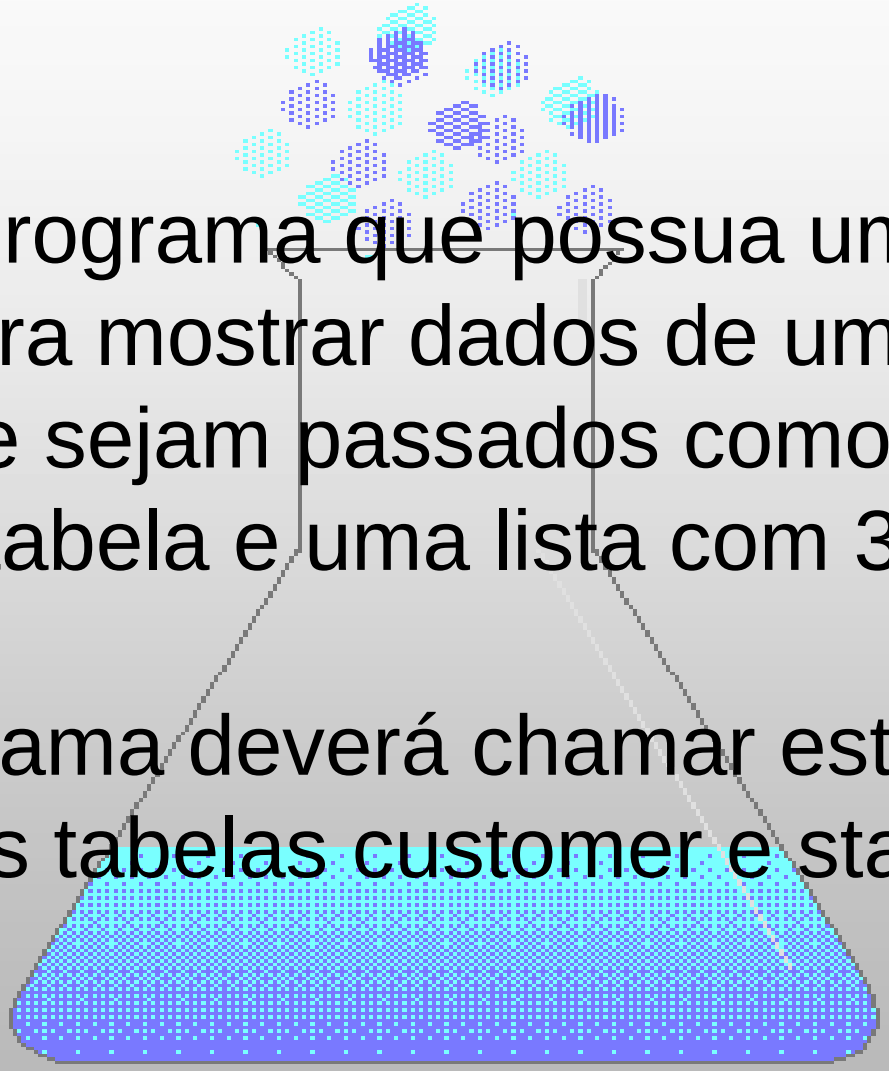
Usando Arquivos de Include para Definir Variáveis



Exemplo de Includes

```
{i-valida.i &table="customer"
            &field="cust-num}

/* i-valida.i */
FOR EACH {&table}:
    DISPLAY {&field}.
END.
```

LABORATÓRIO 5

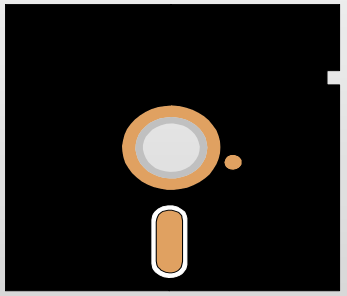


• Criar um programa que possua uma include genérica para mostrar dados de uma determinada tabela, onde sejam passados como parâmetros o nome da tabela e uma lista com 3 atributos.

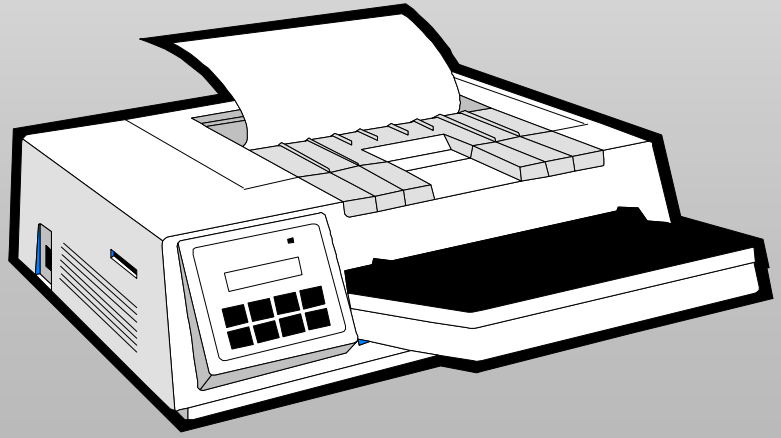
• Este programa deverá chamar esta include para as seguintes tabelas customer e state.

REDIRECIONANDO A SAÍDA

OUTPUT TO nome-arq



PRINTER



TERM

STREAM - Define canais adicionais de saída

Ex.: DEFINE STREAM s1.

OUTPUT [STREAM xxx] TO arquivo.txt
VALUE(c-arquivo)
[APPEND].

PUT [STREAM xxx]
PUT [STREAM xxx] UNFORMATTED } + SKIP

FUNÇÕES

TOTAL

COUNT

MAXIMUM

MINIMUM

AVERAGE

Os labels são mostrados na ordem especificada acima.

TOTALIZANDO VALORES

ACCUMULATE - Calcula uma ou mais expressões de totalização dentro de um bloco de interação.

ACCUM - Acessa os valores acumulados pela função accumulate.

BREAK

Define categorias de quebra.

Represent.	Num Cliente	Limite Crédito	
aaa	1	60.000	
aaa	3	3.000	
		63.000	TOTAL
bbb	4	5.000	
bbb	8	50.000	
		55.000	TOTAL
		118.000	TOTAL

FUNÇÕES

FIRST-OF - Retorna verdadeiro se a iteração atual for a primeira do grupo de quebra.

LAST-OF - Retorna verdadeiro se a iteração atual for a última do grupo de quebra.

Ex.:
```
FOR EACH customer
        BREAK BY customer.sales-rep:
        IF FIRST-OF(customer.sales-rep) THEN
           DISPLAY customer.sales-rep
                   customer.name.
END.
```

CABEÇALHOS E RODAPÉS

Opção HEADER

PAGE-TOP

PAGE-BOTTOM

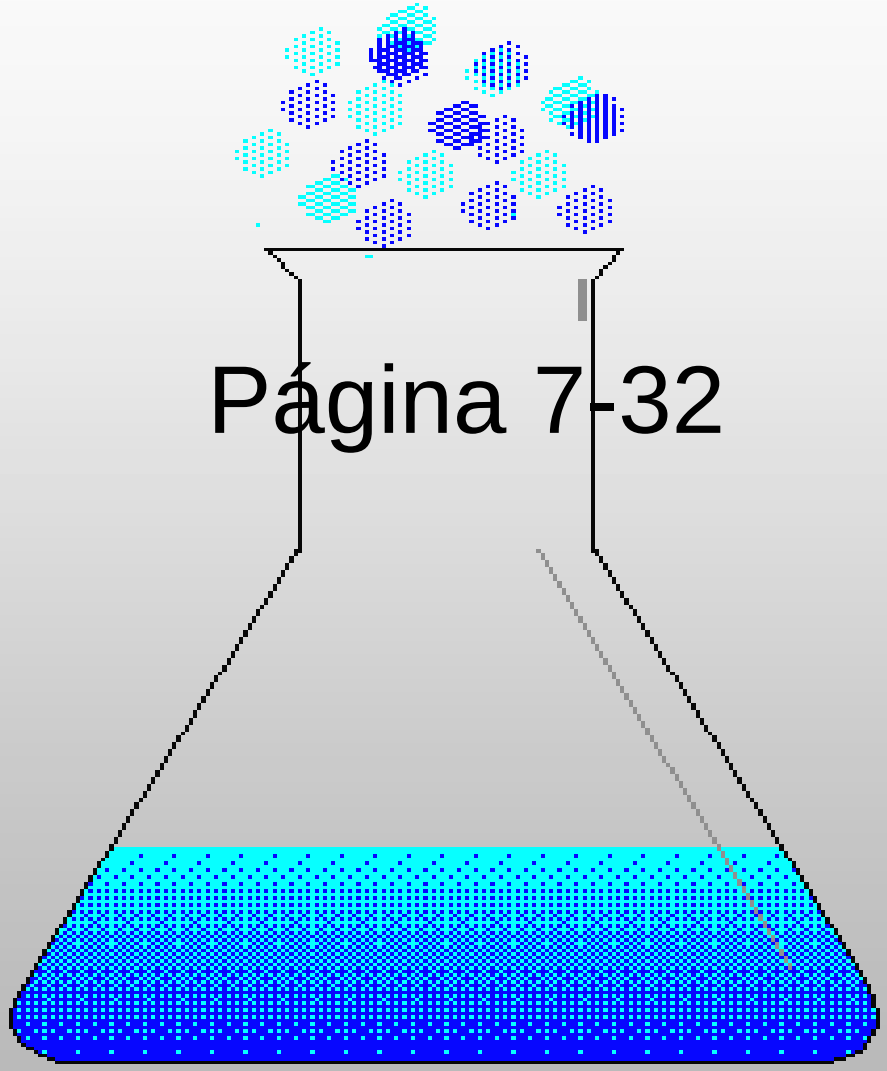
na definição da frame.

Ativar/Desativar frames - VIEW / HIDE

PAGE-SIZE - número de linhas por página.

PAGE-NUMBER - retorna o número da página.

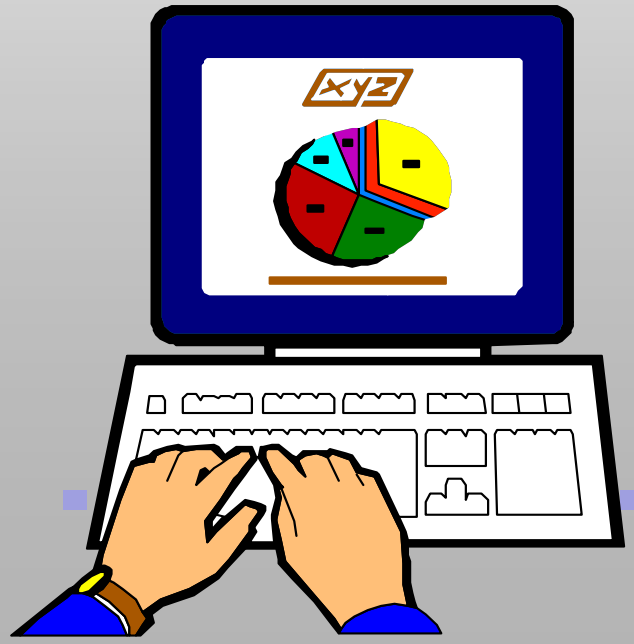
LABORATÓRIO 6



Página 7-32

PROGRAMAÇÃO ORIENTADA A EVENTOS

Usuário dirige a execução através do acionamento dos componentes da interface.



A interface é composta por widgets, que geram diferentes eventos.

COMPORTAMENTO DOS PROGRAMAS

Procedural

Comandos

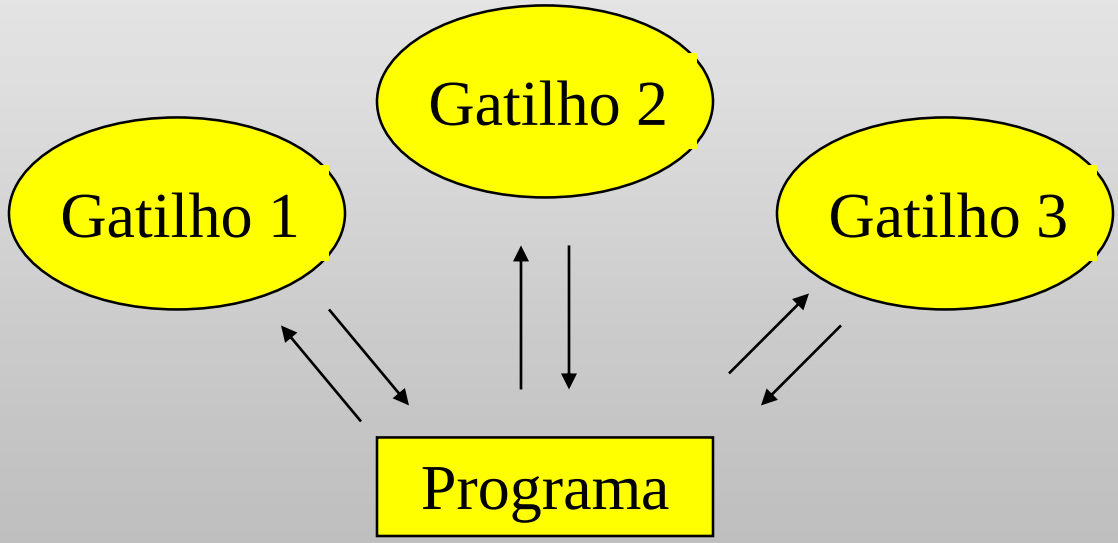
.

.

.

Comandos

Eventos



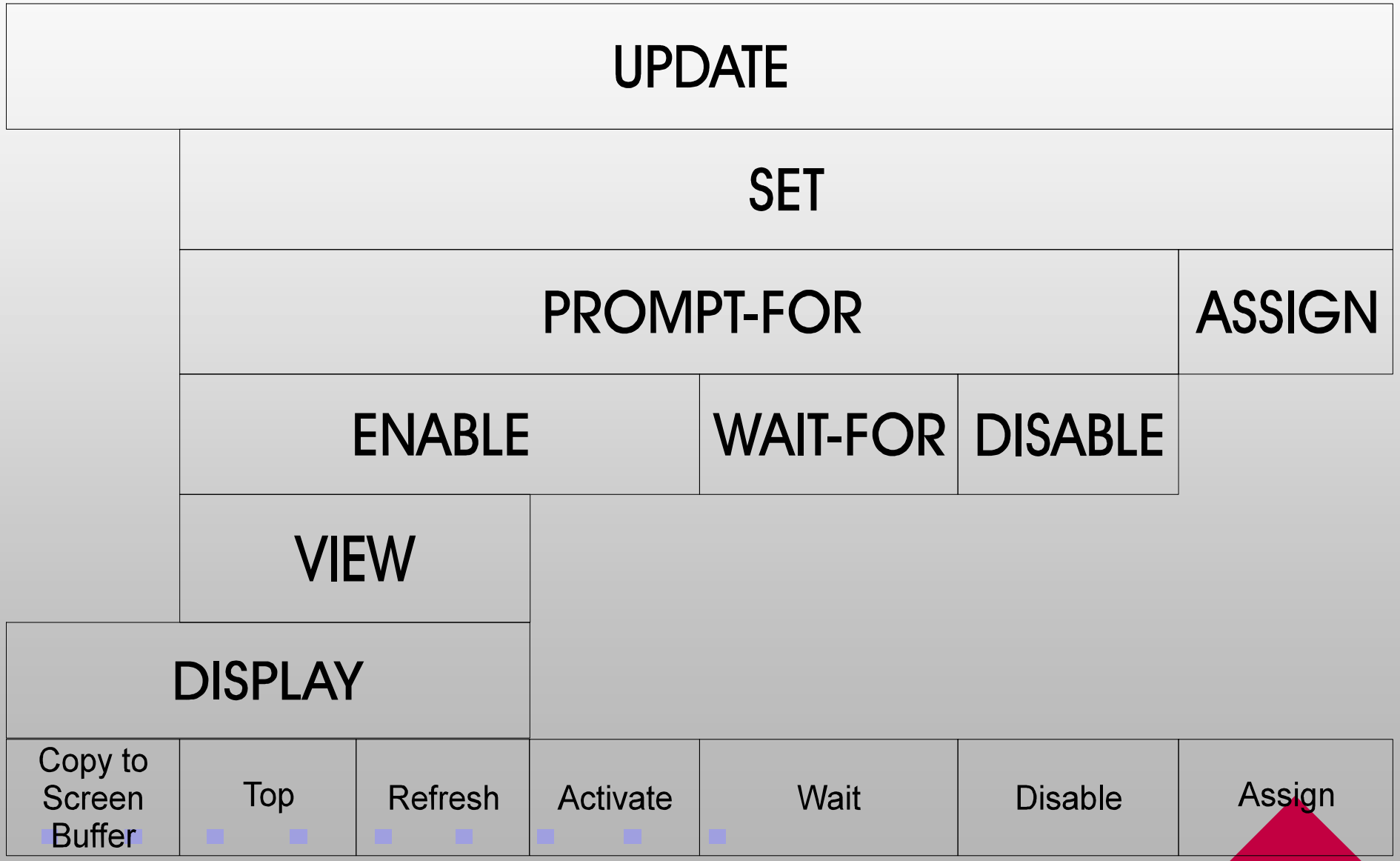
EVENTOS

Ações do usuário ou do Progress que ocorrem durante a execução de uma aplicação.

Tipos de Eventos:

- Eventos causados pelo usuário (acesso a elementos da interface)
- Eventos relacionados ao banco de dados

Comandos de Construção de Blocos



Passos para Utilizar um Objeto

Passo	Sintaxe
Definir o objeto e mostrá-lo	DEFINE...
Definir os triggers para o objeto	ON event OF object DO: ... END.
Mostrar o objeto na frame	DISPLAY...WITH FRAME...
Executar o bloco principal do programa	WAIT-FOR event OF object

WIDGETS

Existem 4 tipos básicos :

- Window Widget: espaço de trabalho da aplicação. Criada quando a aplicação for iniciada.
- Frame Widget: recipiente para outros widgets.
- Data Widget: forma de representação dos dados. São eles: fill-in, text, editor, combo-box, etc.
- Action Widget: forma de representar os comandos. Botões e menus compõem os widgets de ação. Esses widgets permitem que o usuário dirija a operação de sua aplicação.

WIDGETS

ATRIBUTOS

Apresentam/modificam as características dos WIDGETS.

Ex.: ASSIGN wgh_retang1:bgcolor IN FRAME f_dados = 8.

MÉTODOS

Retornam/adicionam informações/dados dos WIDGETS. OBS: Normalmente retorna um expressão lógica.

Ex.: ASSIGN v_log_stat = v_cod_estab:load-mouse-pointer(“cross.ico”) IN FRAME f_dados = 8.

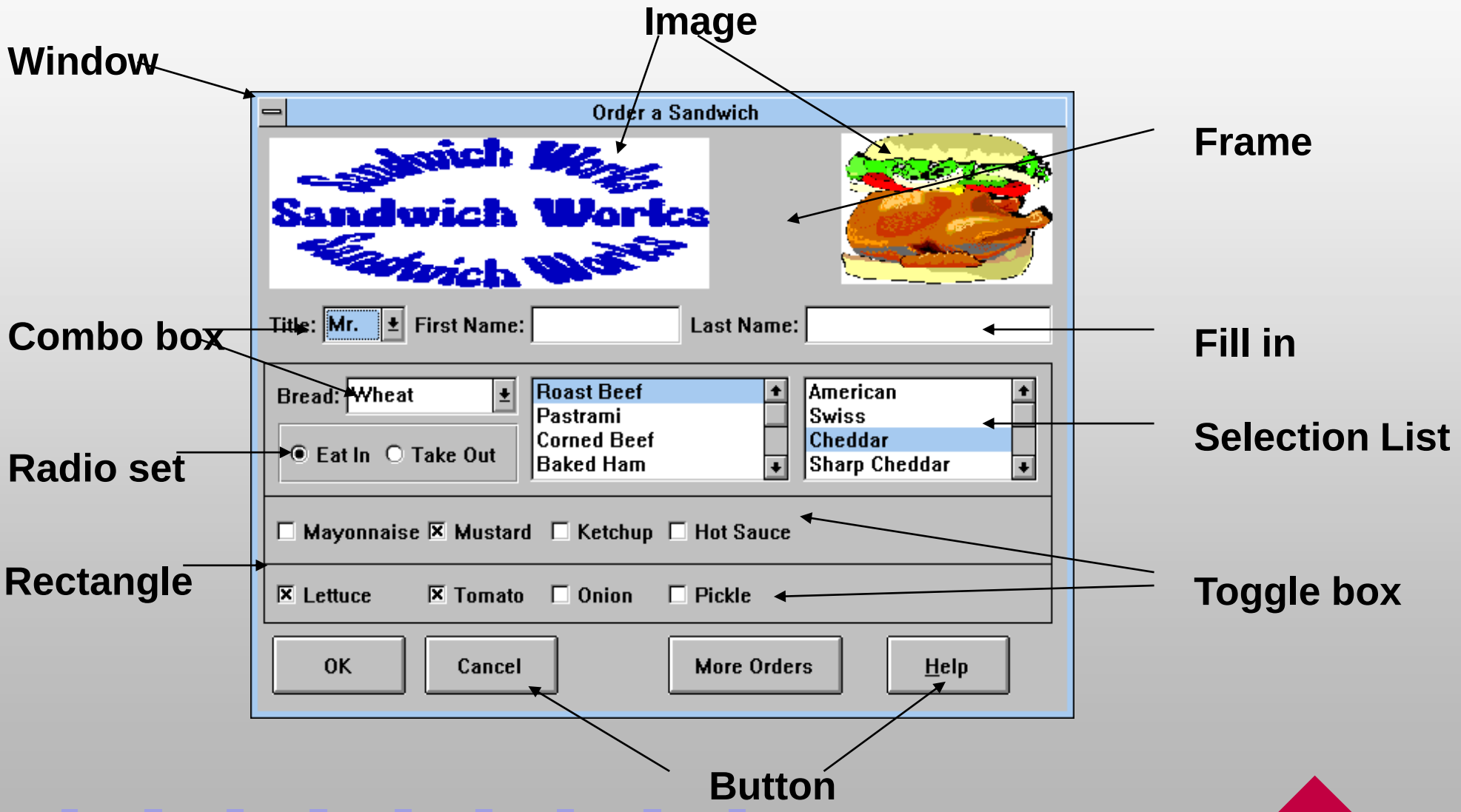
TRIGGERS

Bloco de código executado sempre que um evento ocorre.

ON event-list OF widget-list
    trigger-block

Ex.: ON choose OF bt-cancela DO:
        MESSAGE “Botão de Cancela” VIEW-AS
            ALERT-BOX INFORMATION.
        QUIT.
    END.

Objetos Gráficos



VIEW-AS

Use VIEW-AS para especificar os widgets de representação dos dados.

Ex. DEFINE VARIABLE l-estudante AS logical
VIEW-AS TOGGLE-BOX LABEL
"Estudante?".

FILL-IN

```
DEFINE VARIABLE c-name AS character
       VIEW-AS FILL-IN LABEL "Nome"
       FORMAT "x(30)".
```

Nome:

Atributos
- Format
- Label
- Screen-value
- Sensitive
- Visible
- Tooltip

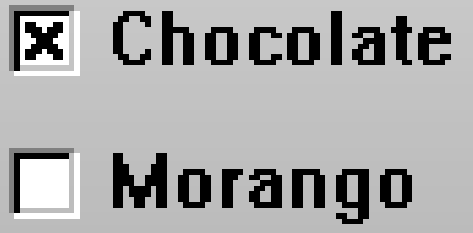
Métodos
- Load-mouse-pointer

Eventos
- Entry
- Leave

TOGGLE-BOX

DEFINE VAR l-cobertura1 AS logical LABEL
"Chocolate" VIEW-AS *TOGGLE-BOX*.

DEFINE VAR l-cobertura2 AS logical LABEL
"Morango" VIEW-AS *TOGGLE-BOX*.



Ao mudar o valor marcado o Progress executa a trigger value-changed

TOGGLE-BOX

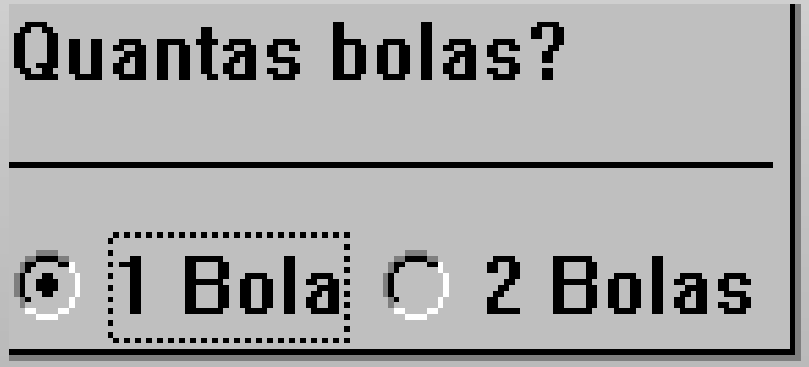
Atributos
- Label
- Screen-value
- Sensitive
- Visible
- Tooltip

Métodos
- Load-mouse-pointer

Eventos
- Entry
- Leave
- Value-changed

RADIO-SET

```
DEF VAR i-num-bolas  AS integer
    LABEL "Quantas bolas?"
    VIEW-AS RADIO-SET HORIZONTAL
    RADIO-BUTTONS  "1 Bola", 1 , "2 Bolas", 2.
```



Ao mudar o valor marcado o Progress executa a trigger value-changed

RADIO-SET

Atributos
- Label
- Screen-value
- Sensitive
- Visible
- Tooltip
- Horizontal
- List-items

Métodos
- Load-mouse-pointer
- Add-last
- Delete
- Disable
- Enable
- Replace

Eventos
- Entry
- Leave
- Value-changed

SELECTION-LIST

```
DEF VAR c-coberturas  AS character
    LABEL "Cobertura"
    VIEW-AS SELECTION-LIST MULTIPLE
    LIST-ITEMS "Chantily    ",
                "Creme      ",
                "Mel        ",
                "Morango    "
    SCROLLBAR-VERTICAL
    INNER-CHARS 12
    INNER-LINES 4 SORT.
```

SELECTION-LIST

Atributos
- Label
- Screen-value
- Sensitive
- Visible
- Tooltip
- Multiple
- Num-items
- List-items

Métodos
- Load-mouse-pointer
- Add-last
- Add-first
- Delete
- Entry
- Replace
- Lookup

Eventos
- Entry
- Leave
- Value-changed
- Default-action

EDITOR

```
DEF VAR c-editor AS char FORMAT "x(3000)"
    LABEL "Editor de Textos"
    VIEW-AS EDITOR INNER-LINES 10
                   INNER-CHARS 20
           SCROLLBAR-VERTICAL
           SCROLLBAR-HORIZONTAL.
```

Editor de Textos:

Teste de Editor de Textos com capacidade de 3000 caracteres sendo que serão apresentadas páginas de 10 linhas de altura por 20 colunas de largura

EDITOR

Atributos
- Label
- Screen-value
- Sensitive
- Visible
- Tooltip
- Num-lines
- Read-only
- Scrollbar-vertical
- Scrollbar-horizontal

Métodos
- Load-mouse-pointer
- Read-file
- Save-file

Eventos
- Entry
- Leave

COMBO-BOX

```
DEF VAR c-coberturas AS character
    LABEL "Cobertura"  VIEW-AS COMBO-BOX
    LIST-ITEMS "Chantily   ",
                "Creme     ",
                "Mel       ",
                "Morango   ".
```

COMBO-BOX

Atributos
- Format
- Label
- Screen-value
- Sensitive
- Visible
- Tooltip
- List-items
- num-items

Métodos
- Load-mouse-pointer
- Add-first
- Add-last
- Delete
- Entry
- Lookup
- Replace

Eventos
- Entry
- Leave
- Value-changed

BOTÕES

```
DEF BUTTON btn-sair    LABEL "Sair” AUTO-ENDKEY.
DEF BUTTON btn-ok      LABEL ”Ok”   AUTO-GO.

DEF BUTTON btn-atualizar  LABEL "Atualizar".
DEF BUTTON btn-desfazer   LABEL "Desfazer”
                          IMAGE FILE “im-undo”.
```

BOTÕES

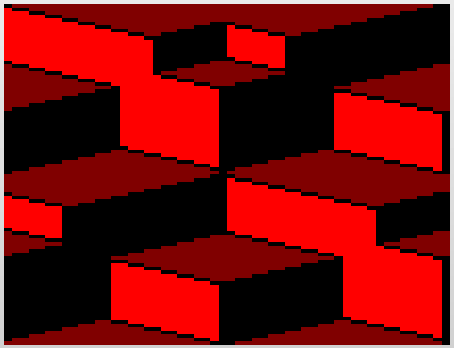
Atributos
- Label
- Sensitive
- Visible
- Tooltip

Métodos
- Load-mouse-pointer
- Load-image
- Load-image-down
- Load-image-up
- Load-image-insensitive

Eventos
- Entry
- Leave
- Choose

IMAGE

DEF IMAGE im-logo **FILE** ”datasul.bmp” **SIZE** 10 **BY** 1.



Atributos	Métodos	Eventos
•Sensitive	•Load-mouse-pointer	•Load-image
•Visible		
•Tooltip		

RECTANGLE

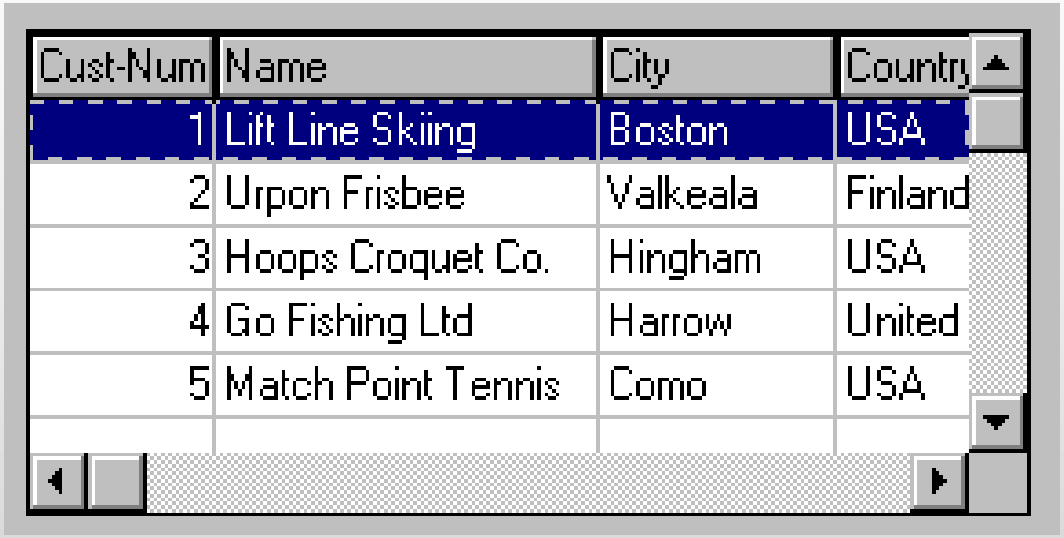
```
DEF RECTANGLE rt-key SIZE 70 BY 5
                     EDGE-PIXELS 3
                     BGCOLOR 3 FGCOLOR 15.
```

Atributos
- Filled
- Sensitive
- Visible
- Tooltip

Métodos
- Load-mouse-pointer

Eventos
- Entry
- Leave
- Choose

BROWSE



Atributos

- Multiple
- Sensitive
- Visible
- Separators
- Title
- Num-selected-rows

Métodos

- Fetch-selected-row
- Is-row-selected
- Select-row
- Load-mouse-pointer

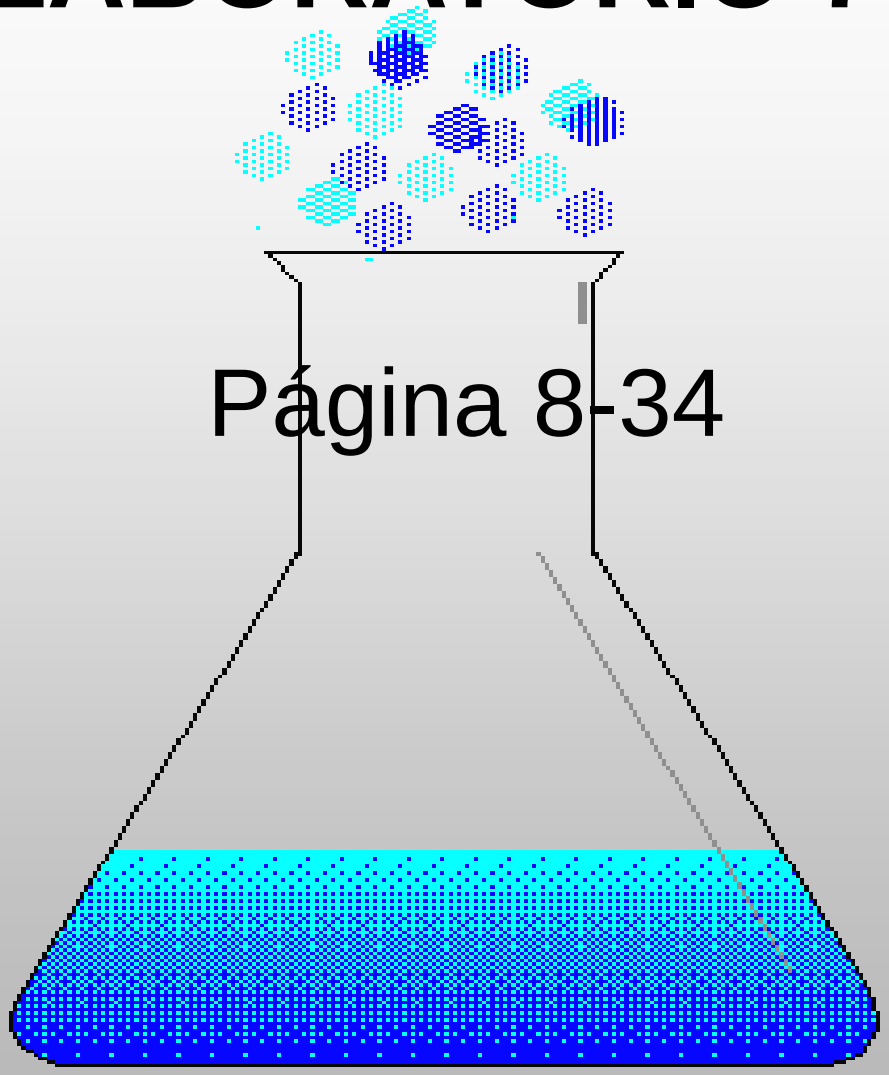
Eventos

- Entry
- Leave
- Default-action
- Value-changed

Representação de Campos

Widget	Tipo de dados		
	Caracter	Numerico	Logico
Combo-Box	x	x	x
Editor	x		
Fill-in	x	x	x
Radio-set	x	x	x
Selection-list	x		
Toggle-box			x
Text	x	x	x

LABORATÓRIO 7



Página 8-34

PRINCIPAIS EVENTOS

- CHOOSE (buttons, menus)
- ENTRY
- LEAVE
- VALUE-CHANGED (toggle-box, radio-set, selection-list, browse)

MANIPULANDO WIDGETS

ENABLE - ativa o widget na tela.
Ex.: ENABLE bt-inclui WITH FRAME f-cliente.

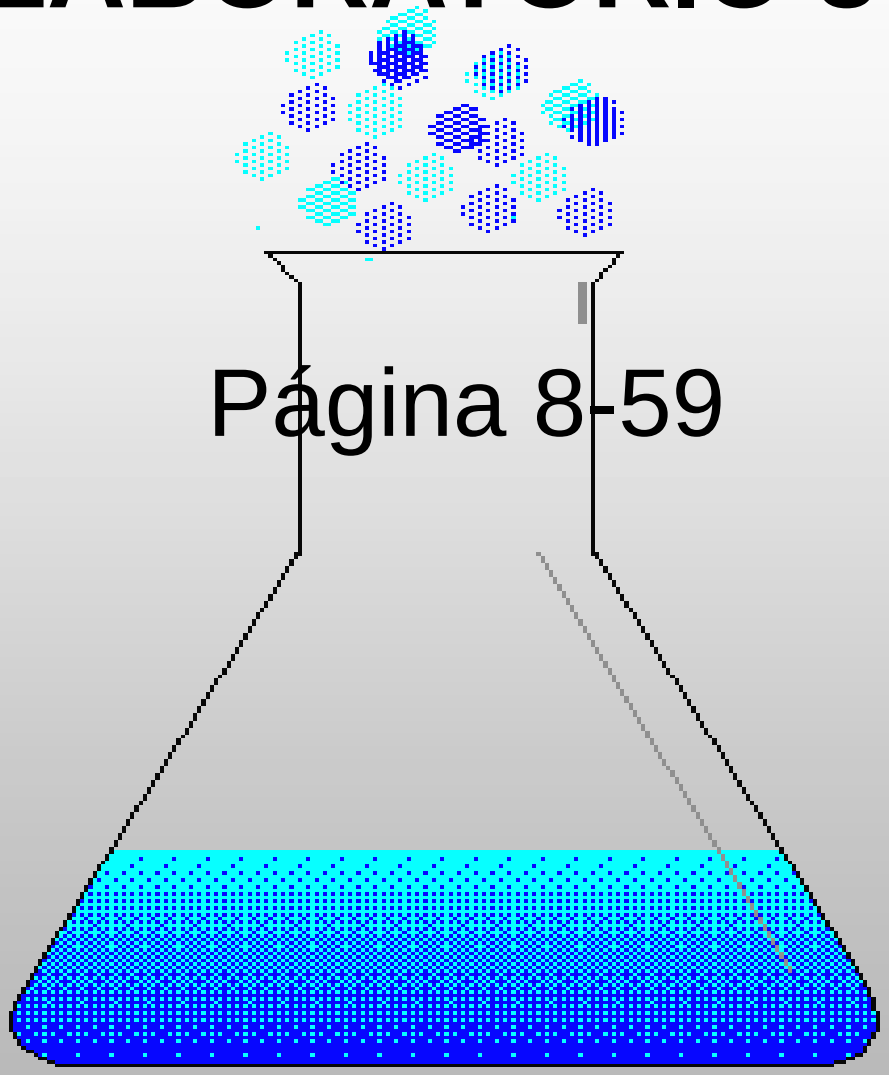
DISABLE - desativa o widget na tela.
Ex.: DISABLE bt-mod WITH FRAME f-cliente.

ATRIBUTOS DOS WIDGETS

widget-name:attribute-name
Ex.: DISP c-name:label WITH FRAME f-ª

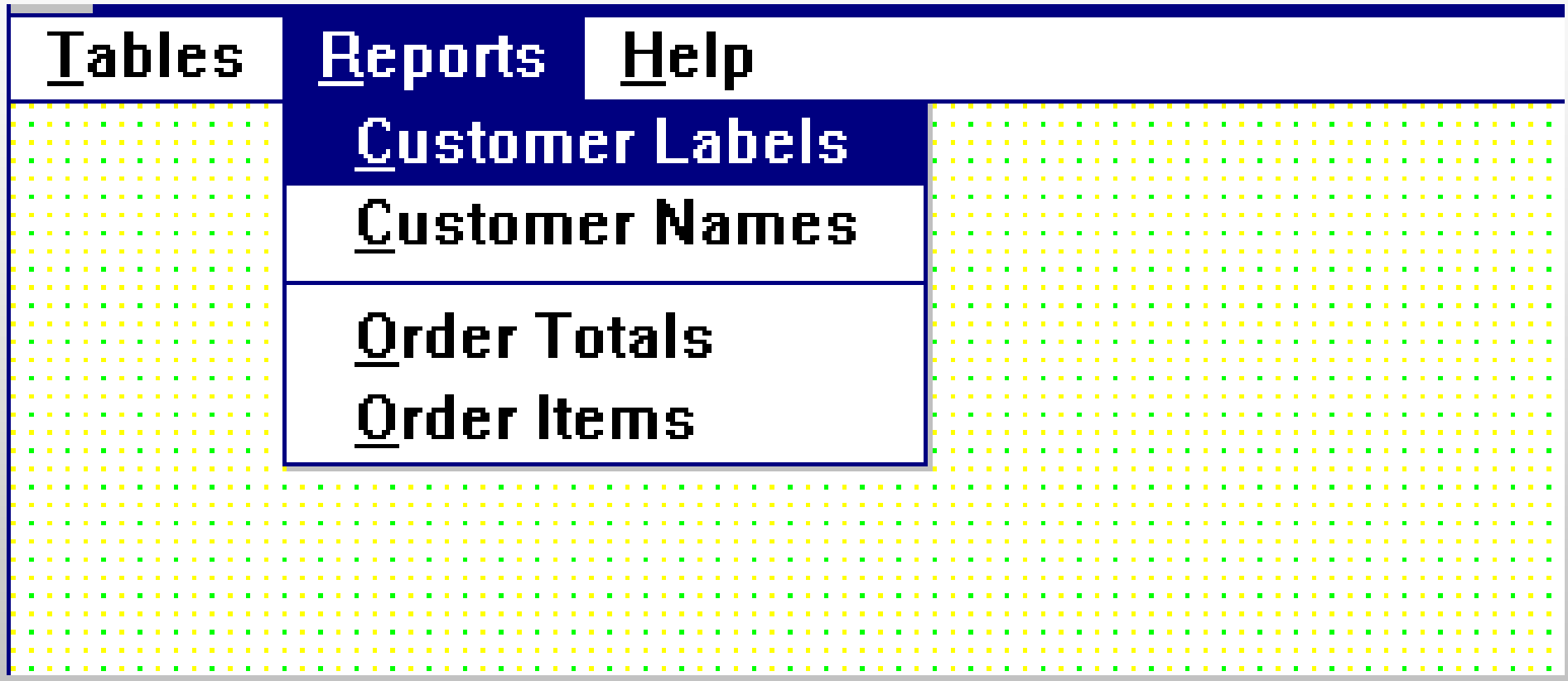
widget-name:attribute-name IN FRAME
frame-name = value
Ex. ASSIGN c-name:help IN FRAME f-a =
“Informe aqui o nome do cliente”.

LABORATÓRIO 8



Página 8-59

MENUS



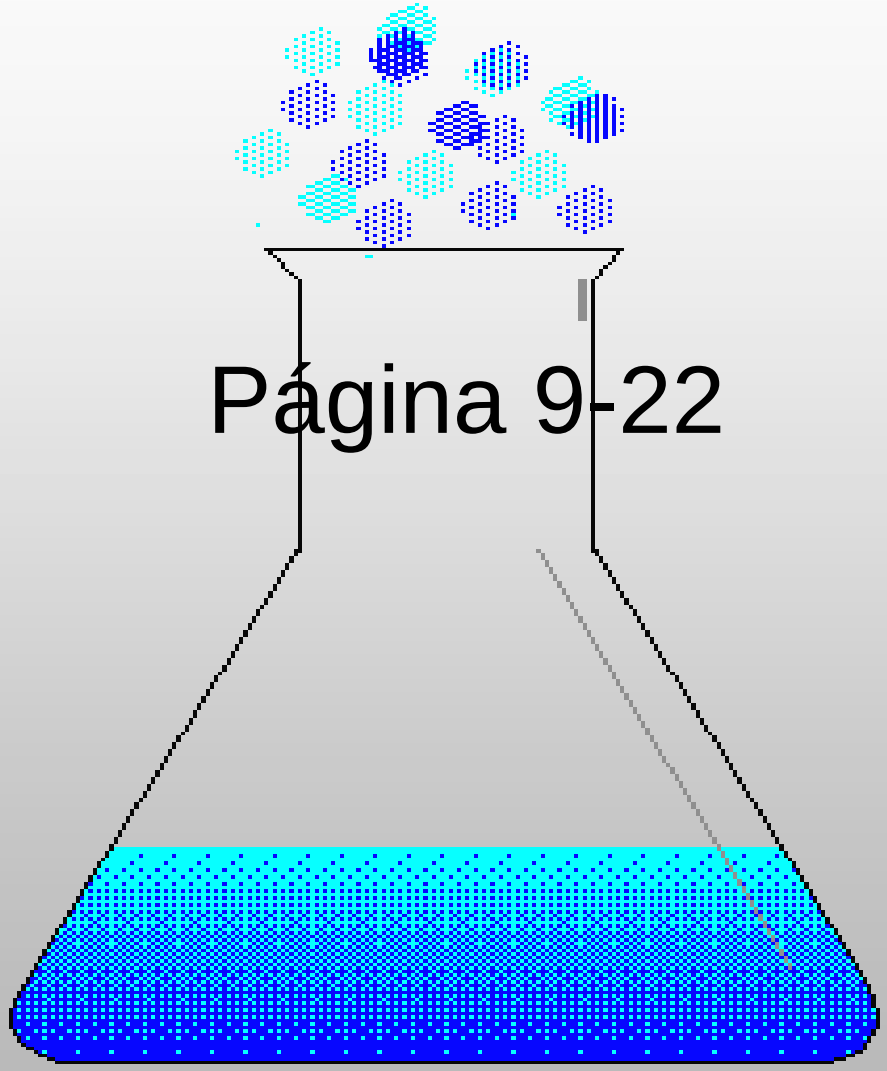

```
ASSIGN DEFAULT-WINDOW:MENUBAR =
       MENU mbar:HANDLE.
```

```
DEFINE SUB-MENU sm-Reports
    MENU-ITEM mi-Labels      LABEL "Customer Labels"
    MENU-ITEM mi-Names       LABEL "Customer Names"
    RULE
    MENU-ITEM mi-Balances    LABEL "Order Totals"
    MENU-ITEM mi-Today       LABEL "Order Items".

DEFINE MENU mbar MENUBAR
    SUB-MENU  sm-Table       LABEL "Tables"
    SUB-MENU  sm-Reports     LABEL "Reports"
    SUB-MENU  sm-Help        LABEL "Help".
```

TRIGGERS PARA MENU

```
ON CHOOSE OF MENU-ITEM mi-item IN MENU
menu-name DO:
      bloco
END.

DEF SUB-MENU sm-sair
     MENU-ITEM mi-sair  LABEL "Sai&r"
     TRIGGERS:
          ON CHOOSE APPLY "window-close" TO
               current-window.
     END TRIGGERS.
```

LABORATÓRIO 9



Página 9-22

COMPARTILHANDO DADOS

DEFINE VARIABLE
NEW
NEW GLOBAL
SHARED
DEFINE SHARED FRAME
DEFINE BUFFER
DEFINE STREAM

BUFFER

É o espelho de uma tabela.
Ex. DEFINE BUFFER bf-cliente FOR customer.

```
define buffer b-aux for customer.

find first customer.
update customer.cust-num
            customer.name.
find b-aux
    where b-aux.cust-num = customer.cust-num.
display b-aux.name
             b-aux.phone.
update customer.address.
```

PARÂMETROS

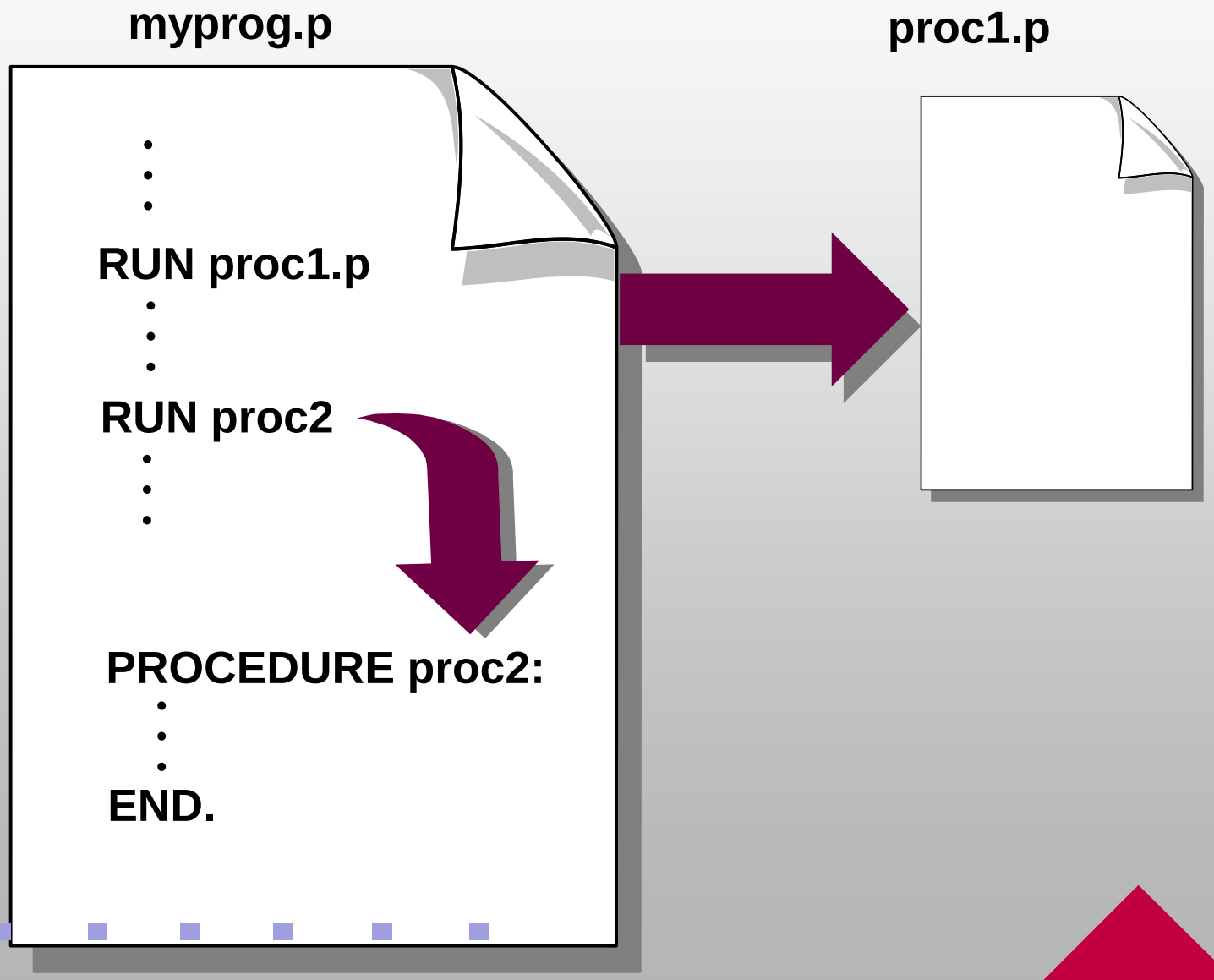
INPUT: recebe um valor do procedimento que o chamou. Ex.: run prog2.p (c-name).
OUTPUT: retorna um valor para o procedimento que o chamou. Ex.: run prog2.p (output i-cont).
INPUT-OUTPUT: recebe e retorna um valor para o procedimento que o chamou. Ex. run (input-output c-name).
RETURN: armazena o valor retornado de uma rotina DLL (somente é válido no ambiente MS-Windows)

PROCEDURES INTERNAS

•Bloco de código que você define dentro de um procedimento PROGRESS.

•Pode-se chamá-lo de qualquer ponto do programa. Acessam os mesmos dados do procedimento, sem a necessidade de compartilhamento.

Programa com Procedures Internas



```
FOR EACH customer:
    DISPLAY customer.cust-num
                customer.name.
    RUN pi-mostra-pedidos.
END.

PROCEDURE pi-mostra-pedidos:
    FOR EACH order OF customer:
        DISPLAY order.order-num.
    END.
END.
```

EXECUTANDO PROCEDURES INTERNAS/PROGRAMAS

RUN {*extern-procedure*}
[PERSISTENT [SET *handle*]]
[(*parameter* [, *parameter*] . . .)]
[*argument*] . . . [NO-ERROR]

Ex.:

```
RUN pi-calculo (INPUT 45,
                OUTPUT v_perc).

RUN w-proces.w PERSISTENT SET hproces.

RUN pi-digito IN hproces (INPUT 382,
                          OUTPUT v_digit).
```

INPUT FROM

INPUT [STREAM xxx] FROM arquivo.txt.

•Se o arquivo de entrada conter mais campos que o procedimento usa, PROGRESS ignora os campos excedentes.

•Se o arquivo de entrada conter menos campos que o esperado, PROGRESS processa a linha do arquivo, mas não seta os campos que ficaram sem entrada.

INPUT FROM

```
INPUT FROM cust.txt.
REPEAT:
    CREATE customer.
    SET customer.cust-num
        customer.name
        customer.sales-rep.
END.
```

IMPORT

```
DEF VAR i-cust    AS integer      NO-UNDO.
DEF VAR c-name AS character NO-UNDO.
DEF VAR c-rep     AS character NO-UNDO.

INPUT FROM cust.txt.
REPEAT:
      IMPORT i-cust c-name c-rep.
      MESSAGE i-cust    SKIP
                        c-name SKIP
                        c-rep VIEW-AS ALERT-BOX.
END.
```

EXPORT

```
OUTPUT TO c:/tmp/state.d.
FOR EACH state:
    EXPORT state.
END.
```

```
"AK" "Alaska" "West"
"AL" "Alabama" "South"
"AR" "Arkansas" "Central"
"AZ" "Arizona" "West"
"CA" "California" "West"
"CO" "Colorado" "West"
```

TRATAMENTO DE ERROS

- Erro de sistema

- Erro gerado pelo procedimento - a busca de um registro falha ou o procedimento tenta criar um registro com chave duplicada.

- Erro de interrupção de processamento - quando o usuário pressiona as teclas de ERROR, ENDKEY ou END-ERROR.

TRANSAÇÕES

- Unidade completa de trabalho.
- A transação é um trabalho que se for terminada a sua execução de forma anormal, será completamente desfeito, ou seja, todo realizado ou nada feito.

Procedimento
REPEAT
FOR EACH

} **Blocos que tem processamento de transação**

TRANSAÇÕES

Uma transação é uma interação do bloco mais externo que contiver declarações que alterem diretamente o banco de dados, ou a leitura de registros com a opção EXCLUSIVE-LOCK.

Escopo de uma transação determina a quantidade de trabalho que PROGRESS desfaz em eventos de falha do sistema.

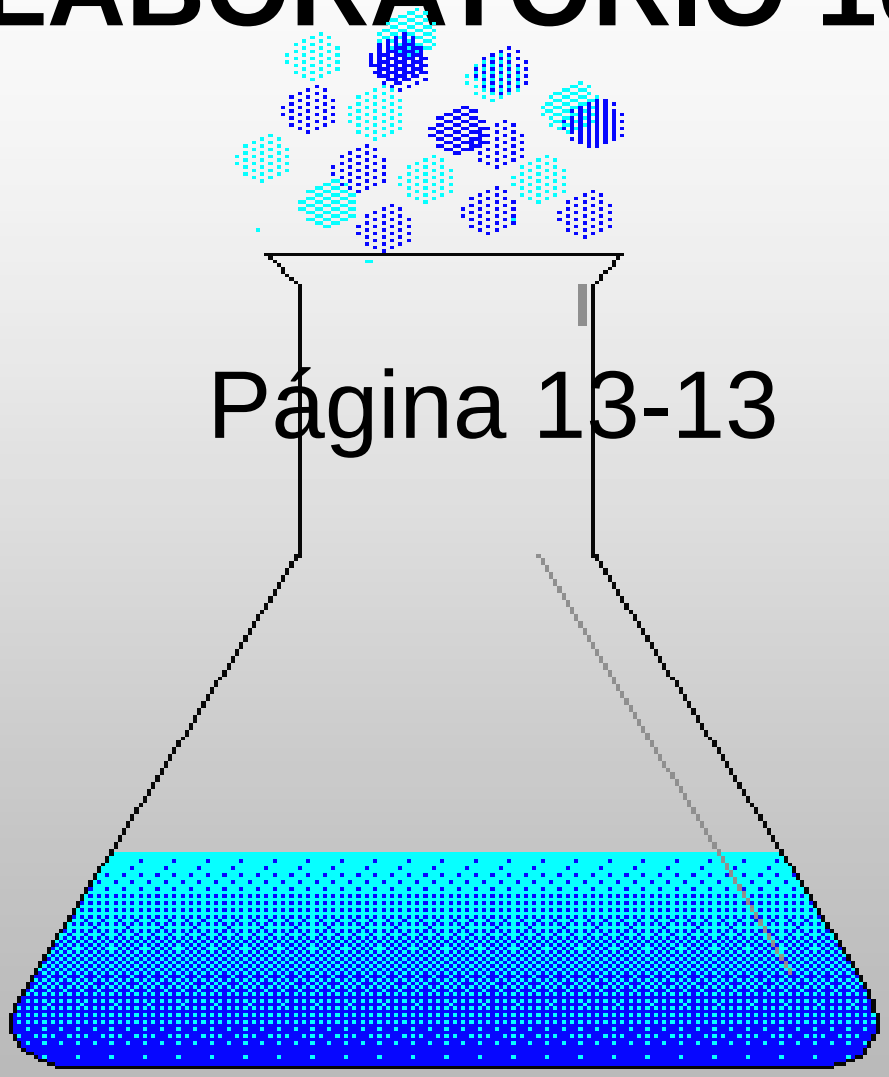
TRANSACTION

```
REPEAT:
    CREATE customer.
    UPDATE cust-num name phone.
END.

DO TRANSACTION:
    REPEAT:
        CREATE customer.
        UPDATE cust-num name phone.
    END.
END.
```

BLOQUEIO DE REGISTROS

NO-LOCK
SHARE-LOCK (Não pode ser utilizado)
EXCLUSIVE-LOCK

LABORATÓRIO 10



Página 13-13

FUNÇÕES

- CAN-DO: verifica uma string dentro de uma lista separada por vírgulas.
IF NOT CAN-DO(lista,string) THEN ...

- CAN-FIND: retorna verdadeiro se um registro foi encontrado.
Ex.: IF CAN-FIND(customer
WHERE customer.cust-num =
i-cust-num) THEN ...

FUNÇÕES

LOOKUP: retorna um valor inteiro indicando a posição da string dentro de uma lista. Separador default: “,”.
LOOKUP(string,lista,[separador]).
Ex.: LOOKUP(“sc”,”am,pr,sc”)

ENTRY: retorna uma string caracter baseada em uma lista, a partir de uma posição inteira. Separador default: “,”.
ENTRY(posição,lista,[separador]).
Ex.: ENTRY(2,c-lista,“/”).

FUNÇÕES

- NUM-ENTRIES: retorna o número de itens de uma lista. Separador default: “,”.
NUM-ENTRIES(lista,[separador])
Ex.: NUM-ENTRIES(c-lista,“.”).

- SUBSTRING: extrai uma porção de caracteres de uma string.
SUBSTR(string,posição-inicial,posição-final)
Ex.: SUBSTR(c-nome,2,10).
SUBSTR(c-address,10).

FUNÇÕES

- STRING: converte dados não caracter para o tipo de dado caracter.

STRING(source,[format]).

Ex.: STRING (customer.cust-num).

STRING (customer.postal-code,”99999”).

FUNÇÕES

- INDEX: retorna um inteiro, indicando a posição de uma expressão dentro de uma string.

INDEX(string,expressão,posição-inicial p/procura).
Ex.: INDEX(“curso de progress”, “de”).
INDEX(“curso de progress”, “pr”, 10).

- RECID: endereço do registro.

Ex.: FIND customer
WHERE RECID(customer) = 3848 NO-ERROR.

TABELAS TEMPORÁRIAS

- São iguais as tabelas;
- Possuem índice;
- Podem denegrir a performance, caso tenham muitos registros e não possuam índice adequado;

```
DEF TEMP-TABLE tt_name   [NO-UNDO]
        [LIKE table-name
        [USE-INDEX index-name [AS PRIMARY]] ... ]
        [FIELD field-name  {AS data-type | LIKE field}] ...
        [INDEX index-name
              [IS [UNIQUE] [PRIMARY] [WORD-INDEX ]]
              {index-field [ASCENDING|DESCENDING]}...]...
```

TABELAS TEMPORÁRIAS

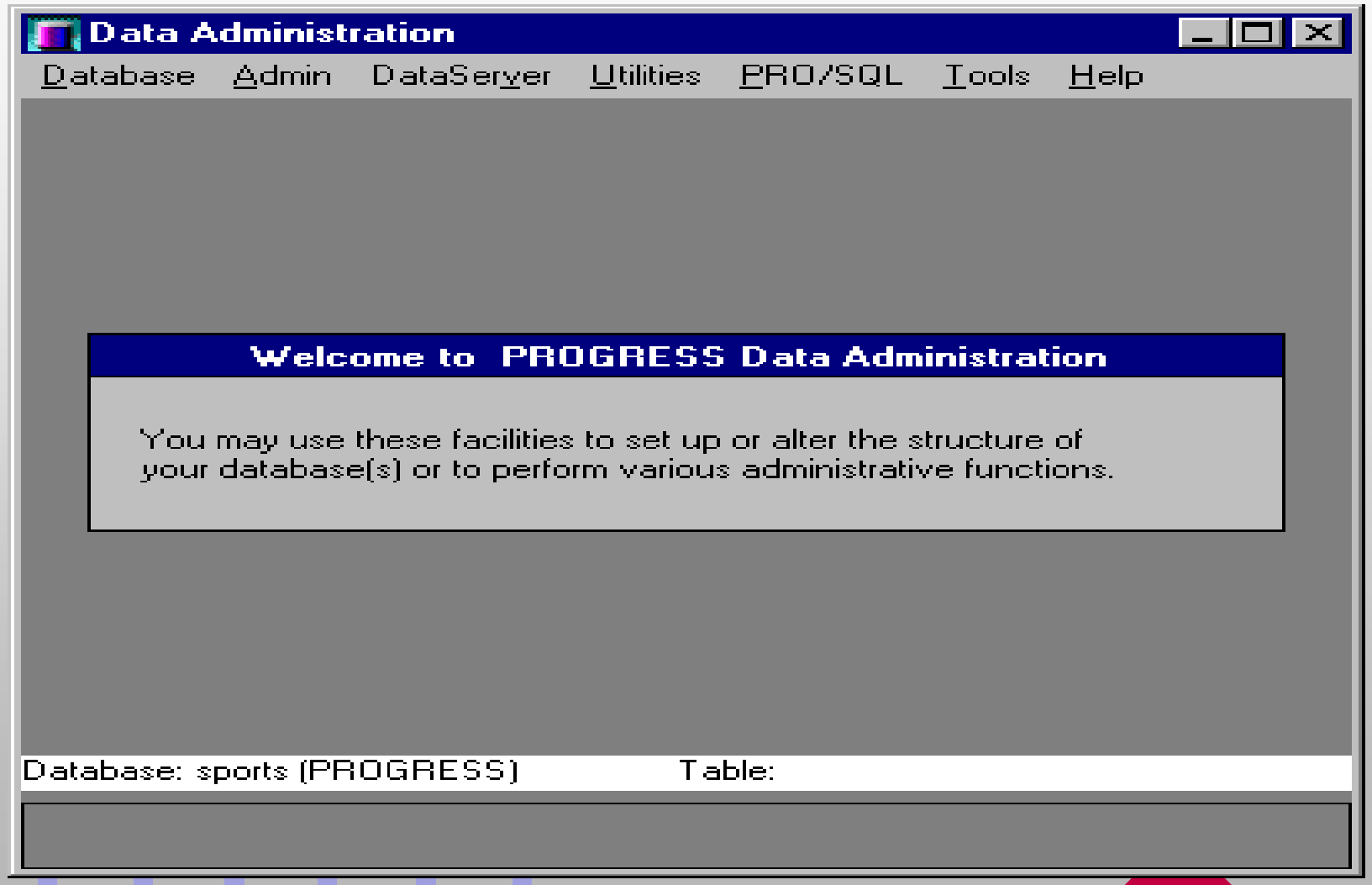
Ex.:

```
DEF TEMP-TABLE tt_cliente NO-UNDO
        LIKE customer
        FIELD cod_fornec AS character FORMAT “x(5)”
                                      LABEL “Fornecedor”
        FIELD region          LIKE state.region
        INDEX fornec IS PRIMARY cod_fornec.

INSERT tt_cliente EXCEPT comments
            WITH 1 COLUMN.

FOR EACH tt_cliente NO-LOCK:
        DISP tt_cliente EXCEPT comments
                WITH SIDE-LABELS.
END.
```

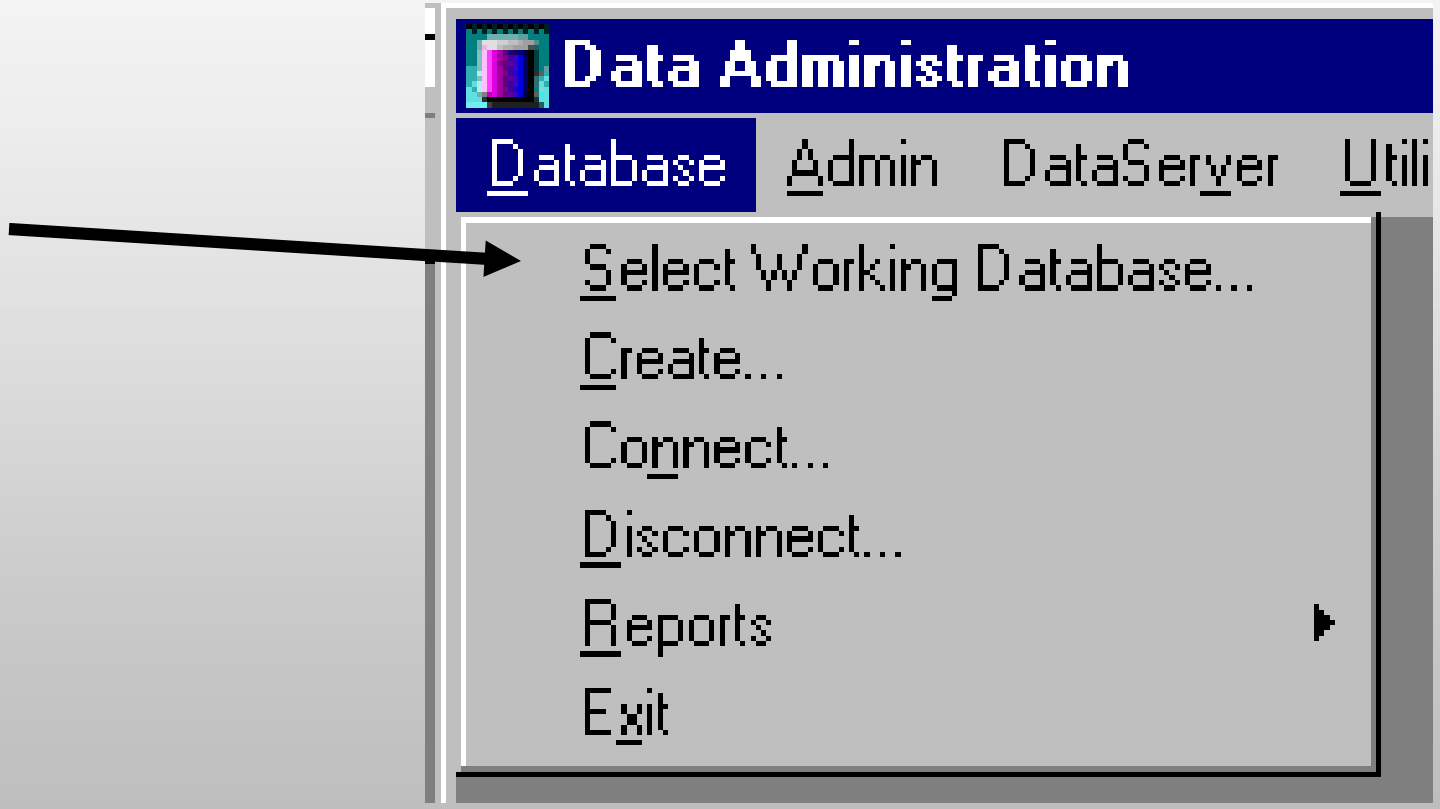
LABORATÓRIO 11

- Criar a tabela temporária que seja igual a tabela ORDER;
- Adicionar o atributo ***name***, este deverá ser igual ao atributo da tabela CUSTOMER;
- O programa deverá ler todos os CUSTOMERS, todas os ORDERS dos mesmos a atualizar a temp-table, alimentando os atributos: ***cust-num, name, order-date, order-num*** e ***sales-rep***.
- Mostrar uma listagem que mostre os atributos acima e todos os registros da temp-table.

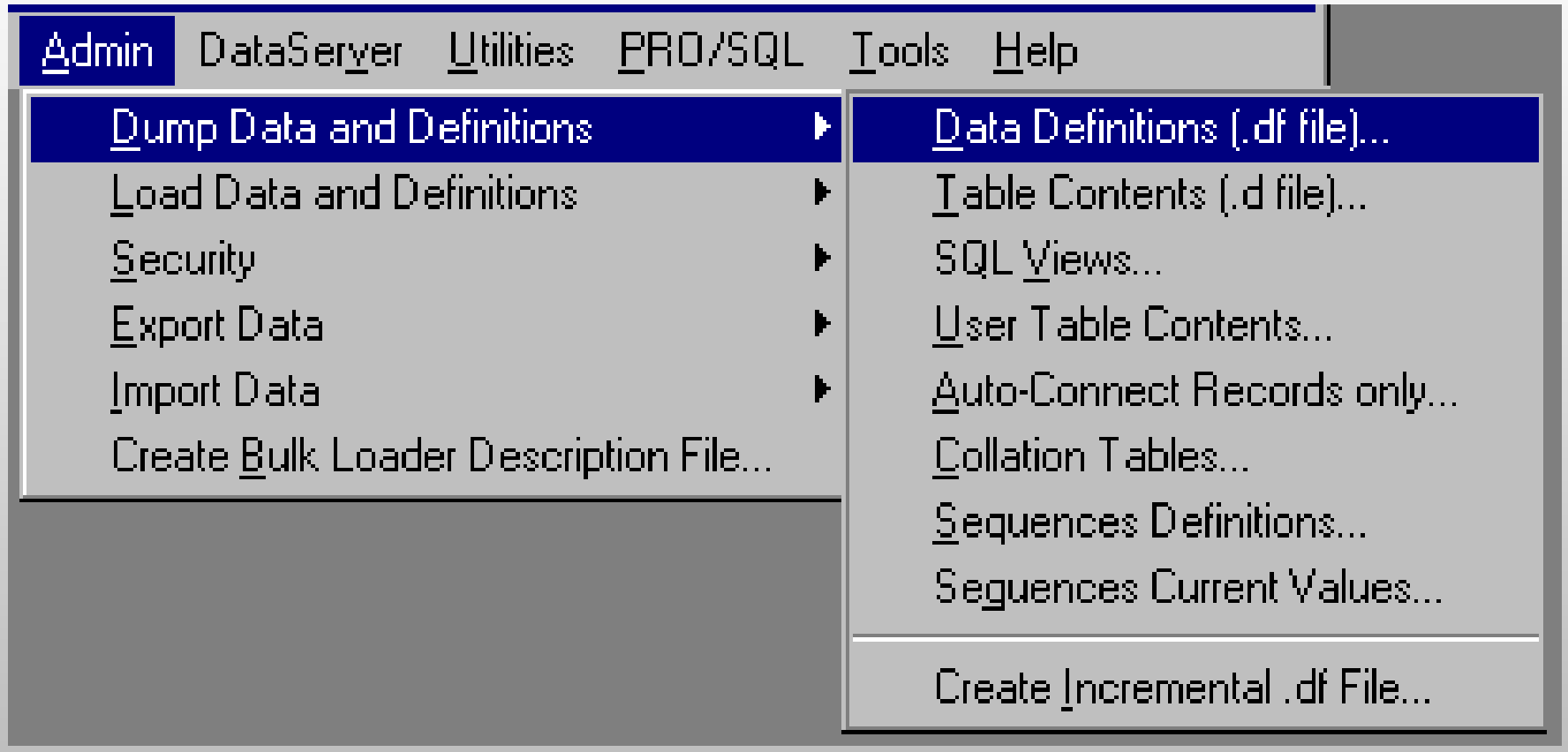
Dump/Load da Base de Dados



Dump/Load da Base de Dados



Dump/Load da Base de Dados



LABORATÓRIO 12

- Criar uma base de dados com o nome de vocês;
- Fazer o DUMP da base de dados SPORTS;
- Fazer um LOAD dos dados na base de dados de vocês.

DÚVIDAS ?