

Boletim de Notas Técnicas

1 Introdução

Neste Boletim Informativo Técnico (BIT) ou Boletim de **NOTAS TÉCNICAS** você encontra atualizações, aprimoramentos, acertos e novos recursos introduzidos nos produtos **OpenBASE**.

Em se tratando de Notas Técnicas, não se encontram aqui todas as informações de maneira completa e integrada, sendo apenas abordados os tópicos que sofreram modificações ou que precisam de maiores esclarecimentos.

Os assuntos apresentados neste BIT são renovados com frequência, incorporados periodicamente aos manuais OpenBASE, que você poderá consultar para obter informações mais completas e integradas sobre todos os produtos da família OpenBASE. Esta documentação completa pode ser acessada e/ou obtida em nossa página [Internet](#).

Os assuntos abordados neste BIT são os seguintes:

- **Administração de Bancos de Dados OpenBASE**
- **Desenvolvimento de aplicativos – Opus, OpusWin e GERAL**
- **Desenvolvimento de aplicativos - Ferramentas RAD**
- **Utilitários do OpenBASE**
- **Dicas e macetes do universo OpenBASE**

2 Administração de Bancos de Dados OpenBASE

A seguir relacionamos as novidades, modificações e acertos implementados pelo projeto **OpenBASE** nesta área.

2.1 *Arquivo com log das transações*

2.1.1 Propósito

Permitir a geração de um arquivo de log que registra todas as transações (inclusão, alteração ou exclusão) ocorridas no banco de dados Openbase

2.1.2 Sintaxe

```
[<< <comentário> >>]
[$CONTROLE <opção>, ... , <opção>]
BANCO [<percurso>] <nome_bd> <código_de_seguranca> ..
    [TRANLOG= arquivo]
... [{ARQRECUP / DIARIO / DIAREC}]...
... [{BLOQARQ / BLOQPAG / BLOQREG}]...
... [ESQUEMA=[<percurso_bd_origem>] <nome_bd_origem>...
... <código_de_seguranca_bd_origem>...
... [<palavra_de_nível_bd_origem>] ]
[NIVEIS: <numero_nivel> <palavra_nivel> <numero_nivel> <palavra_nivel>]
```

2.1.3 Argumentos

Arquivo - Percurso e nome do arquivo de log que será gerado. Se o percurso não for informado, o arquivo de log será gerado no diretório corrente.

2.1.4 Utilização

Este arquivo de log é útil quando não se tem idéia da origem de algum erro possa vir a ocorrer na utilização do Openbase. O arquivo de log guardará as seguintes informações:

- data-hora: 19 bytes
- ação: 1 byte
 - L - início da transação

- M - modify
- I - insert
- D - delete
- U - fim da transação
- nome do arquivo: 10 bytes (brancos se ação for L ou U)
- nome do prog, func ou proc: 20 bytes
- login do usuário: 20 bytes

Será criado um arquivo <nome do banco>.U que conterà o nome do arquivo de log que guardará as transações. Na abertura do banco se não existir o arquivo de log, o mesmo é criado. Deve-se tomar cuidado com o tamanho deste arquivo de log, para que o espaço em disco seja preservado.

2.1.5 Exemplo

```
banco demo 1 TRANLOG=log.txt
```

```
nome: tabela E
```

```
pm_cod (0) u02
```

```
pm_nome u40
```

```
pm_campodt u06
```

```
*****
```

```
*Carrega dados no banco acima
```

```
*****
```

```
prog carga
```

```
database demo 1 a 2
```

```
select a
```

```
use tabela
```

```
i = 1
```

```
for i= 1 to 10
```

```
  replace pm_cod with str(i,2);
```

```
    pm_nome with "Marcos";
```

```
    pm_campodt with "111298"
```

```
  insert
```

```
  if dberr() # 0
```

```
    ? dbmens()
```

```
endif
```

```
next
```

```
return
```

```
*****
```

```
*Altera dados no banco acima
```

```
*****
```

```
prog cargaalt
```

```
database demo 1 a 2
```

```
select a
```

```
use tabela
```

```
locate
```

```
Do While .not. eof()
```

```
  replace pm_nome with "Bruno"
```

```
  modify
```

```
  if dberr() # 0
```

```
    ? dbmens()
```

```
endif
```

```

continue
Enddo
return

```

```

*****
*Exclui dados no banco acima
*****

```

```

prog cargaexc
database demo 1 a 2
select a
use tabela
locate
Do While .not. eof()
    delete
    if dberr() # 0
        ? dbmens()
    endif

    continue
Enddo
return

```

O arquivo log.txt que guarda o log referente as transações feitas nos três programas acima, será:

Data-Hora	Ação/Arquivo	Prog/Func/Proc	Login
18/02/2002-18:52:29	L	carga	root
18/02/2002-18:52:29	I tabela	carga	root
18/02/2002-18:52:29	U	carga	root
18/02/2002-18:52:30	L	carga	root
18/02/2002-18:52:30	I tabela	carga	root
18/02/2002-18:52:30	U	carga	root
18/02/2002-18:52:30	L	carga	root
18/02/2002-18:52:30	I tabela	carga	root
18/02/2002-18:52:30	U	carga	root
18/02/2002-18:52:30	L	carga	root
18/02/2002-18:52:30	I tabela	carga	root
18/02/2002-18:52:30	U	carga	root
18/02/2002-18:52:30	L	carga	root
18/02/2002-18:52:30	I tabela	carga	root
18/02/2002-18:52:30	U	carga	root
18/02/2002-18:52:30	L	carga	root
18/02/2002-18:52:30	I tabela	carga	root
18/02/2002-18:52:30	U	carga	root
18/02/2002-18:52:30	L	Carga	root
18/02/2002-18:52:30	I tabela	Carga	root
18/02/2002-18:52:30	U	Carga	root
18/02/2002-18:52:30	L	Carga	root
18/02/2002-18:52:30	I tabela	Carga	root
18/02/2002-18:52:30	U	Carga	root
18/02/2002-18:52:30	L	Carga	root
18/02/2002-18:52:30	I tabela	Carga	root
18/02/2002-18:52:30	U	Carga	root

Data-Hora	Ação/Arquivo	Prog/Func/Proc	Login
18/02/2002-19:24:00	D tabela	cargaexc	root
18/02/2002-19:24:00	U	cargaexc	root
18/02/2002-19:24:00	L	cargaexc	root
18/02/2002-19:24:00	D tabela	cargaexc	root
18/02/2002-19:24:00	U	cargaexc	root
18/02/2002-19:24:00	L	cargaexc	root
18/02/2002-19:24:00	D tabela	cargaexc	root
18/02/2002-19:24:00	U	cargaexc	root
18/02/2002-19:24:00	L	cargaexc	root
18/02/2002-19:24:00	D tabela	cargaexc	root
18/02/2002-19:24:00	U	cargaexc	root

2.2 Itens com Valor Nulo

2.2.1 Propósito

Permitir que um ítem de dado armazene um valor desconhecido ou não informado. Os tipos de itens de dados que aceitam valor nulo, são:

- U,A(alfabéticos)
- I,B,N,S,C,P<F(numéricos)
- D(Data)

2.2.2 Sintaxe

```
<nome_item> [({<rep>}) / (<ligações>) / (<caminho>)/(0)]...
... <tipo>:<tamanho>[,<num_decimais>] ... ||
... [(<num.nivel_ier>,<num_nivel_grav>)] ...
... [({POS <nome_item_rd>[+ <deslocamento>]) /...
... VIRTUAL (<nome_item_part>, ..., <nome_item_part>)] ...
... [NULO] [ATU = {C/S/R/}] [DEL = {C/R/S}][UNICA]
```

2.2.3 Argumentos

Nenhum

2.2.4 Utilização

Um item nulo, possui um byte a mais do que seu tamanho original, pois o primeiro byte indica se o item é nulo(byte binário é zero) ou não(byte binário é um).

Para se atribuir um valor nulo a um item do tipo U, A ou D(sem \$date), utilize a constante cadeia NULLCHAIN que equiivale a uma cadeia vazia(""). Para se atribuir um valor nulo a um ítem nulo do tipo numérico, utilize a constante numérica NULLNUMBER, que equivale ao número -999999999999999999.

Para se saber se o ítem lido é nulo ou não, basta compará-lo a NULLCHAIN se este item for do tipo U,A,D(sem \$date) ,ou a NULLNUMBER caso este ítem seja numérico.

Um item virtual que possua algum componente nulo, também será nulo.

Ao se utilizar o geral, os valores nulos são exibidos como ????.?.

Se for utilizada a máscara "????....?" um item nulo será exibido como ????.?.

2.2.5 Exemplo

```
banco bdteste 1
nome:arq1 e
cod1(0) n5
nome u21 nulo
salario n10,2 nulo
admiss d3 nulo
*
*****
```

Carga do banco definido acima

```
prog
database bdteste 1 a 2
select a
use arq1
locate
for i = 1 to 10
  if (i%2) = 0

    replace   cod1   with i,;
              nome   with "Teste" ,;
              salario with 2000,;
              admiss with date()
    insert
    if dberr() # 0
      ? dbmens()
    endif

  else
    replace   cod1   with i,;
              nome   with NULLCHAIN,;
              salario with NULLNUMBER,;
              admiss with NULLCHAIN

    insert
    if dberr() # 0
      ? dbmens()
    endif
  endif
next
return
```

Lê os dados carregados pelo programa acima

```
prog
database bdteste 1 a 2
use arq1
locate
Do While .not. eof()
  @01,02 say cod1
  if nome = NULLCHAIN
    @01,10 say " Nome NULO"
  else
    @01,10 say nome
  endif
  if salario = NULLNUMBER
    @03,10 say " Salario NULO"
  else
    @03,10 say salario
  endif
  if admiss = NULLCHAIN

    @05,10 say " Data NULA"
  else
```

```

    @05,10 say admis
endif
wait
continue
Enddo

```

2.3 Dicionário de Dados e Arquivos Locais

2.3.1 Propósito

Replicar parte de um banco de dados de uma máquina servidor para uma máquina cliente, melhorando o tempo de acesso.

2.3.2 Sintaxe

No esquema do banco de dados deve existir:

```

banco <nomebanco> <seg>
[LOCAL=<diretório na máquina cliente onde será copiado o dicionário de dados>]
nome: <nomearquivo> <tipo>
[LOCAL=<diretorio na máquina cliente onde será copiado o arquivo de dados>]

```

2.3.3 Utilização

Esta opção deve ser utilizada quando o tempo de acesso ao banco de dados é fundamental. O arquivo de dados copiado para máquina cliente, não é atualizável. Esta atualização é feita diretamente na máquina servidor. Apenas quando o banco for aberto novamente, o arquivo de dados atualizado será copiado para máquina cliente. O tempo de acesso ao banco de dados é diminuído, quando existem muitas consultas ao mesmo.

Para se acessar esta opção, utilize, na OPUS, os seguintes comandos:

```

$local= "Nome/IP máquina servidor"
set LOCAL on

```

Observação:

Quando utilizamos a DLL dupwin32.dll (Cliente/Servidor), o comando **LigaOpcao("local")** equívale ao comando OPUS **set LOCAL on**.

Quando utilizamos o GERAL, o comando **ligue LOCAL** equívale ao comando OPUS **set LOCAL on**. No caso da opus, devem-se utilizar as bibliotecas **libbddup.a (unix)** ou **libbddup.lib (windows)**.

2.4 Carga de Arquivos em Memória

2.4.1 Propósito

Agilizar o tempo de acesso a um banco de dados Openbase.

2.4.2 Sintaxe

Criar um arquivo texto, com o mesmo nome do dicionário de dados e com extensão .ini. Este arquivo deverá ser preenchido da seguinte forma:

```

UNIX:    shmkey = <numero>
          shared  = <arquivo>
          .....
          shared = <arquivo>

WINNT:
          shared = <arquivo>
          .....

          shared = <arquivo>

```

2.4.3 Argumentos

- **número** - número da memória compartilhada a ser criada.
- **nome** - arquivo do banco a ser carregado em memória

2.4.4 Utilização

Esta opção permite que arquivos de dados do Openbase sejam carregados em memória compartilhada, melhorando consideravelmente o tempo de acesso ao banco de dados.

Deve ser utilizada quando o tempo de acesso ao banco de dados for um fator determinante.

Além dos arquivos de dados são carregados os arquivos de índice também.

A atualização destes arquivos em memória é permitida, entretanto o tempo de acesso aos mesmos é prejudicado.

2.5 *BDVESCLI*

2.5.1 Propósito

Recuperar o esquema de um Banco de Dados Distribuído.

2.5.2 Sintaxe

```
Bdvesqcli -b<banco> [-s<segurança>] -n[<nível>] [-O] [-g<arq_sai>][[-c]
```

2.5.3 Argumentos

-b<banco>	Indica o percurso e o nome do Banco de Dados
-s<segurança>	Informa o código de segurança do Banco de Dados. Se omitido, será considerado o valor 1.
-n<nível>	Indica a palavra de nível do usuário. Se omitido, será assumido o valor "a"(DEFAULT)
-o	Indica que os arquivos não serão verificados na abertura do banco de dados
-c	Determina que as opções REDEFU, COMPCAB E NOMEIGUAL do comando \$CONTROLE serão gravadas no fonte do esquema
-g<arq_sai>	Indica o nome do arquivo de saída que conterá o fonte do esquema do banco de dados. Caso se omita este parâmetro o arquivo será gerado na saída padrão

2.5.4 Utilização

Através do utilitário **bdvescli**, é possível a partir do dicionário de dados gerado pela compilação de um esquema de um bando de dados distribuído com as opções **distrib** e **servidor=<host>**, se recuperar o esquema de banco de dados distribuído.

2.5.5 Consulte

BDVESQ

2.6 *BDSERA*

2.6.1 Sintaxe

```
Nohup bdsera -s <caminho> &
```

2.6.2 Argumentos

-s <caminho>	<caminho> é a localização deste programa, por exemplo: c:\usr1\bdsera.exe. Valor default: c:\usr\tsgbd\bdsera.exe.
--------------	---

2.6.3 Utilização

O programa **bdsera.exe**, deve ser inicializado, como serviço, caso o Windows trabalhe como servidor de banco de dados. No Windows 9X deverá ser incluído no registry a chave:

```
Hkey_Local_Machine\Software\Microsoft\Windows\Current Version\RunServices
```

O valor da chave acima deverá conter o percurso completo do **bdsera**, por exemplo, **c:\usr\tsgbd\bdsera.exe**.

2.7 Banco de dados distribuídos com replicação

Para melhor compreender o exemplo a seguir, imaginemos que os arquivos **arq1**, **arq2** e **arq3** estão presentes nas máquinas **h1**, **h2** e **h3**, respectivamente e o arquivo **arq1** também esta presente no servidor **h2**. O esquema do banco de dados distribuído, **dist.esq**, é descrito a seguir.

```
banco dist 1 arqrecup distrib=h1
nome: arq1 e servidor=h1 replicação=h2
C1 (1) N3
nome:arq2      R      servidor=h2
C2 (arq1)     N3
nome:arq3      E      servidor=h3
C3(0)         U03
```

Tal esquema (**dist.esq**) será compilado, utilizando o seguinte comando:

```
deficli -h h1 dist.esq
```

onde :

h1 é o nome do servidor onde foi especificada a cláusula **distrib**. Feito isto, serão gerados os seguintes arquivos:

Dist.B Arquivo de bloqueio de banco de dados gerado nos servidores h1, h2, h3.
Dist.R Arquivo de recuperação gerado nos servidores h1,h2,h3
Dist.h Dicionário de dados distribuídos, gerado nos servidores h1,h2,h3.
dist.L Arquivo de log que registra as transações ocorridas nos arquivos distribuídos pelos servidores que compõem o banco de dados distribuído.
Presente apenas no servidor h1.

O Banco de Dados distribuído presente no servidor h1, utiliza arquivos do servidor h1 (arq1), h2 (arq1/arq2) , e h3 (arq3), sendo que o arquivo arq1, presente no servidor h1, é replicado, ou seja, qualquer alteração neste arquivo, é refletido no arquivo arq1, presente no servidor h2. Com isso, o arquivo arq1 do servidor h2, funciona como um backup online, do arquivo arq1 do servidor h1, sendo útil, em caso de pane no servidor h1.

Para que um banco de dados distribuído com replicação funcione, é necessário que o arquivo bdserv, esteja carregado em todas as máquinas (host), que participaram do esquema do banco de dados distribuído. O arquivo bdsera, também deve ser carregado no servidor onde será compilado o banco de dados distribuído, caso haja necessidade de se garantir a integridade referencial dos arquivos (presença de chave estrangeira).

2.8 Arquivos transpostos

2.8.1 Definição dos arquivos

Um arquivo **transposto** é definido especificando-se ao lado do nome e tipo a opção transposto ou especificando-se \$ controle transposto quando todos os arquivos serão transpostos.

2.8.2 Definição dos itens

Na definição do item pode ser especificado um "domínio" para definir a compressão dos valores. O domínio pode ser uma lista de valores ou intervalo para itens numéricos.

Se especificada uma lista de valores (para tipos U, I, N, P e D) serão atribuídos bits para representar estes valores: 1-2 valores 1 bit, 3-4 valores 2 bits, etc.

Se especificado um intervalo (para tipos N, I, P e D) será compactado o valor segundo o intervalo, 1-2 1 bit, 3-4 2 bits etc.

2.8.3 Criação dos arquivos

O arquivo de dados será criado com o cabeçalho e 1 byte para indicar registros excluídos. Para cada item será criado um arquivo com nome nnn<arq>. O formato do arquivo será relativo acessando-se um valor na posição (n-1)*t onde n é o número do registro e t o tamanho dos valores em bits.

2.8.4 Exemplos

nome:	arqtran e transposto
item1(0)	n3
sexo	u1 valores (m,f)
idade	n2 intervalo (0-99)

2.9 Tipos de Itens

2.9.1 Itens Tipo A

2.9.1.1 Propósito

Tipo de dados, que permite a ordenação de itens caracter acentuados.

2.9.1.2 Sintaxe

<nome item> A <tamanho do item>

2.9.1.3 Utilização

Um item tipo A é idêntico ao item tipo O, com a diferença que as vogais acentuadas são convertidas para vogais sem acento, assim como o cedilha é convertido para C, no arquivo de índice, visando a ordenação. No arquivo de dados as vogais são armazenadas com acento.

2.10 Arquivos Complementares ao Dicionário de Dados

Além do Dicionário de Dados, são gerados os seguintes arquivos, (após a compilação do esquema do Banco de Dados) com o mesmo nome do Dicionário de Dados, mas com diferente extensão:

<dicion>	Nome do dicionário de Dados.
<dicion>.B	Arquivo criado quando se utiliza bloqueio de dicionário (bloqueio de banco).
<dicion>.R	Arquivo de Recuperação (ARQRECUP especificado no esquema).
<dicion>.P	Arquivo criado para gerenciar nomes de percurso (path) assim como nomes de dicionário, nomes de arquivo e nomes de itens com tamanho maior que 56 posições.
<dicion>.C	Indica que está se usando arquivos transposto (domínios para arquivos transpostos).
<dicion>.T	Indica que está se utilizando espelhamento de arquivos de dados (ESPELHO=).
<dicion>.H	Indica que está sendo utilizado banco de dados distribuído (DISTRIB). Dicionário de dados distribuído gerado em cada uma das máquinas que compõem o banco de dados distribuído.
<dicion>.L	Arquivo de transação distribuída. Gerado em bancos de dados distribuído. Armazena a imagem anterior das transações, não completadas, nos arquivos que compõem um banco de dados distribuído.
<dicion>.G	Arquivo gerado na máquina cliente, que contém o percurso local no cliente LOCAL=), do dicionário de dados no servidor, quando se utiliza um banco de dados distribuído.
<dicion>.S	Data-hora para arquivos replicados (REPLICACAO). Este arquivo pode ser editado.
<dicion>.U	Arquivo para controlar o diário de transações (TRANLOG=)
<dicion>.Z	Arquivo pode permitir refazer as transações (ARQREFAZ).

2.11 Arquivo de recuperação (ARQRECUP)

2.11.1 Propósito

Ao se declarar um banco de dados *OpenBASE* em um esquema, a opção ARQRECUP passa a ser default. Caso se deseje que o banco de dados *OpenBASE* não possua arquivo de recuperação (ARQRECUP), deve-se especificar a opção NAORECUP.

2.11.2 Sintaxe

banco [<percurso>] <nome_bd><codigo_de_segurança> [{NAORECUP/DIARIO/DIAREC/AUTOREC}]
[{BLOQARQ/BLOQCHA/BLOQPAG/BLOQREG}]

2.11.3 Utilização

Ao se utilizar o ARQRECUP como default, na declaração de um banco de dados *OpenBASE*, ganha-se em segurança, já que se uma transação não for completa, a mesma pode ser refeita (rollback), a partir da imagem anterior do registro, que é guardada em <NOMEDICIONARIO>.R. Para isto basta executar o utilitário bdrecu.

2.12 Número de ligações no DEFINE

Numa relação entidade (tipo E ou T) o número de ligações pode ser especificado como *, o que será automaticamente substituído pelo número de referências nos arquivos posteriores.

2.12.1 Exemplos

nome: PESSOA e	
NOME1 (*)	u30 <<será substituído por 1>>
IDADE	n2
nome: FILHOS r	
NOME1 (PESSOA)	U30
IDADEF	n2

2.13 Datas tipo D4

No *OpenBASE* um item de dado do tipo D2, só aceita datas compreendidas entre 01/01/1901 e 04/06/2080, inclusive, na conversão de uma variável de memória para o item(D2) do banco. Caso a data esteja fora deste intervalo, é emitida a mensagem OPUS(varite) => Estouro na conversão numérica. Para datas com tipo D4 esta tal conversão pode ser feita para datas dentro ou fora do intervalo mencionado anteriormente..

O item D4 indica uma data no formato aaaa/mm/dd onde aaaa ocupa 2 bytes, mm 1 byte e dd 1 byte. O dia 1 é considerado 01/01/0001. A data 31/12/1900 ainda é considerada como data nula para se manter compatibilidade com o tipo D2.

Para sua utilização na OPUS em conjunto com as datas tipo D2 foram feitas as seguintes alterações:

2.13.1 DTOC

Para itens do banco do tipo D2 retorna uma cadeia de caracteres no formato **dd/mm/aa** se set century off e 1901 <= ano <= 1999. Caso contrário, retorna uma cadeia no formato dd/mm/aaaa.

Para itens do banco do tipo D4, retorna uma cadeia no formato **dd/mm/aaaa**, mesmo para datas menores do que 1901 e datas maiores do que 1999.

2.13.2 CTOD

Considera uma cadeia no formato dd/mm/aa como **dd/mm/19aa**.

2.13.3 Conversão de variável para item do banco

Se tipo D2, 01/01/1901 <= data <= 04/06/2080, caso contrário é emitida a mensagem OPUS (varite) => Estouro na conversão numérica.

2.13.4 Conversão do tipo do item D2 para D4.

Os itens tipo D2 de um banco de dados podem ser descarregados e carregados no mesmo banco de dados com os itens alterados de tipo D2 para tipo D4.

2.14 Especificação de "path" (percurso)

No Windows antes do path (caminho) do dicionário e arquivos de um banco de dados deve haver a definição do drive. Se não houver, automaticamente são colocados da seguinte maneira:

define

banco <nome><seg><caminho>

nome:<nome><tipo><caminho>

Se no <caminho> não houver drive é colocado o drive corrente.

Com estas alterações no dicionário todos os caminhos tem o drive antes. Na abertura de um banco em um drive diferente do corrente, todos os arquivos serão encontrados. Se um banco for mudado de drive deve ser executado o define para colocar o novo drive no dicionário.

2.15 Estrutura de Esquema

2.15.1 Declaração do Banco de Dados (Opção Diário)

O arquivo diário pode ser criado para um banco de dados específico, bastando para isso, que na declaração do dicionário de dados, seja especificado:

banco [<percurso>] <nome_bd> <codigo_de_segurança> [Diario[=<arq>]] [Diarec[=<arq>]]

Onde :

<arq> é o nome do arquivo Diário, criado para o banco de dados nome_bd, que guarda a imagem posterior do registro, refazendo transações completadas (dados do registro após este ser modificado). Se for especificada a cláusula <Diarec>, a imagem anterior do registro (dados do registro antes deste ser modificado) e a imagem posterior do registro (dados do registro após este ser modificado), são guardadas no arquivo diário.

Pode-se também reduzir, de forma significativa, o tamanho do arquivo diário, bastando-se para isso, especificar a cláusula LOGICO, na declaração do dicionário de dados, conforme:

banco[<percurso>]<nome_bd><codigo_de_segurança> [Diario[=<arq>]][LOGICO]

[Diarec [=<arq>]][LOGICO]

Com a cláusula LOGICO, são gravados no arquivo diário, os registros lógicos, isto é, somente a área de dados, sem o cabeçalho com os endereços dos registros anterior e seguinte.

2.15.2 Declaração do Dicionário de Dados

2.15.2.1 Descrição

<percurso> <nome_bd> O percurso do dicionário de dados deve ser especificado entre aspas, caso contenha espaço em branco em seu nome.

2.15.2.2 Exemplos

banco "c:\Arquivo de Programas\banco\teste"

2.16 Tamanho do Registro para item O4

2.16.1 Propósito

Aumenta o tamanho de um registro que armazena texto ou imagem de um item O4.

2.16.2 Sintaxe

banco [percurso] <nome banco> <codigo-segurança> [tamregmemo=<num>]

2.16.3 Argumentos

Tamregmemo é o tamanho em bytes do registro que irá armazenar o texto ou imagem. Pode variar de 500 bytes a 5000 bytes.

2.16.4 Utilização

Útil para aumentar o tamanho do registro em um arquivo OO<número> gerado para armazenar imagens/texto de um item O4. Com isso, o tamanho da imagem/texto armazenado no arquivo OO<número> é maior, evitando estouro na tabela de bloqueio, ao se utilizar bloqueio de registro, já que uma imagem/texto muito grande pode ser representada por vários registros. Quanto maior o tamanho do registro, menos registros são necessários para armazenar a imagem/texto.

2.16.5 Exemplos

banco exemplo 1 tamregmemo=4000

2.17 Opções de Controle

2.17.1 \$Controle Maxicab

2.17.1.1 Propósito

Com esta opção, definida na primeira linha do esquema do banco de dados, (neste caso será aplicada para todos os arquivos do banco de dados), ou definida para um determinado arquivo, os endereços dos registros passam a ter 4 bytes em vez de 3 bytes. Com isso a quantidade de registros em um arquivo aumenta, permitindo-se armazenar uma quantidade de dados muito maior no arquivo, já que a quantidade de endereços nos arquivos passam a ser bem maior.

2.17.2 \$PRIMBIN

2.17.2.1 Propósito

Tal opção é especificada em um arquivo, no esquema do banco de dados, caso o primeiro item do arquivo seja binário (I,B,D ou M). Deve-se ter cuidado, pois caso o número do registro onde o primeiro item for binário, seja múltiplo de 255, o sistema operacional o considera como registro deletado.

2.17.3 \$Controle Uppercase

2.17.3.1 Propósito

Na especificação de um esquema, utiliza-se a opção \$Controle Uppercase, para que os nomes dos arquivos e itens sejam convertidos para letra maiúscula.

2.17.3.2 Sintaxe

\$Controle Uppercase

2.17.3.3 Argumentos

Opção utilizada na estrutura de um esquema de um banco de dados *OpenBASE*, para que os nomes de itens e arquivos, sejam gerados em letra maiúscula. Útil, ao se utilizar aplicativos que tratam banco de dados *OpenBASE*, como: geral e utilitários (bddesc, bdadic, bndinc, bdveri, etc), que são sensíveis a letra minúscula/maiúscula.

2.17.3.4 Exemplos

```
$Controle uppercase
banco teste 1 arqrecup
nome: tabela E
  codigo(0) U02
  nome (0) U20
```

Seria gerado, após a compilação deste esquema:

TABELA	E
CODIGO	U02
NOME	U20

2.17.4 \$Controle Inibin

2.17.4.1 Propósito

Evita que itens binários do tipo D2 ou D4, tenha os bits invertidos, caso o processador seja Intel.

2.17.4.2 Sintaxe

\$ Controle Inibin

2.17.4.3 Utilização

Útil para evitar problemas, na ordenação de datas de campos binários D4, D2, já que tais itens são binários invertidos (bits são agrupados, do fim para o início).

2.17.5 \$controle nomegrande

2.17.5.1 Propósito

Ao se especificar, \$ controle nomegrande, na primeira linha de um esquema de um banco de dados *OpenBASE*, permite-se que os nomes do dicionário de dados, arquivos e itens tenham até 63 bytes de tamanho, e não mais 10 bytes (arquivos) e 12 bytes (itens).

2.17.5.2 Exemplos

```
$controle nomegrande
banco banco_com_nomegrande 1
nome: arquivo_com_nome_muito_grande e
item_com_nome_grande (0) U02
```

2.18 Espelhamento do Bancos de Dados OpenBASE.

2.18.1 Propósito

Permite que uma cópia do banco de dados, seja automaticamente gerada em outro diretório, funcionando como backup dos arquivos de dados.

2.18.2 Sintaxe

```
banco [<percurso>] <nome_banco> <codigo_de_segurança> espelho = <path>
```

2.18.3 Argumentos

Path Diretório ou percurso, onde os arquivos de dados do banco serão copiados.

2.18.4 Utilização

A opção espelho é útil, quando deseja-se obter uma cópia online, dos arquivos de dados de um BANCO *OpenBASE*, à medida em que estes vão sendo atualizados. Os arquivos entidades (tipo E), são espelhados como arquivos do tipo F, enquanto que arquivos Relacionamento (tipo R), são espelhados como arquivos do tipo S. Os arquivos espelhados não podem ser acessados para gravação, apenas consulta. Antes de se inicializar o processo de espelhamento, é necessário que os arquivos originais, sejam copiados para o percurso (path), onde será feito o espelhamento para que os arquivos fiquem atualizados e compatíveis.

2.18.5 Exemplos

```
banco /usr/alm/dic/BDALM 12 espelho = /usr1/alm/dic.
```

2.19 Itens Memo tipo M, O e Q

2.19.1 Propósito

Armazenar um campo texto ou uma imagem bmp, gif ou jpg.

2.19.2 Sintaxe

```
nome: nomearq E
item_chace (0) tipo/tamanho
item_memo M4/O4/Q4
```

2.19.3 Utilização

Em um esquema, no lugar de se definir um item memo M4 pode ser definido um item tipo O4 ou Q4. Neste caso, no lugar de ser criado um diretório com nome do arquivo onde está o item M4(mmnomearq) e dentro deste diretório ser criado outro diretório com o nome do item M4 onde são armazenados os arquivos memo referentes ao itens M4, é criado um arquivo para conter os memos, evitando ocupação desnecessária de espaço em disco, ou problemas de permissão de leitura/execução/gravação deste arquivos.

Na linguagem OPUS pode ser utilizados as funções MEMOPUT, MEMOGET, MEMODEL e MEMOFILE para processar itens tipo O ou Q.

O utilitário bdveri verifica a integridade referencial entre os itens memo (M4, O4 ou Q4) e os arquivos no diretório mm<arq>/<item> ou cadeias nos arquivos oo<número> ou qq<arq><item>.

Para não ser efetuada a verificação especificar a opção -m.

O utilitário bddesc descarrega nas opções -a<arquivo> ou -S (todos) no mesmo formato os itens memo, permitindo a conversão de tipo quando utilizado o utilitário bdadic.

Ao se alterar um item de M4 para O4 ou Q4, ou de O4 para Q4, deve-se descarregar o arquivo em questão(bddesc), alterar o tipo do item no esquema, recompilar(define <nome do esquema>) o banco recriando o arquivo, e por fim recarregar tal arquivo(bdadic)

O item Q4, foi criado, visto que a utilização do item O4, gera para cada item O4, um arquivo com o nome oo+número seqüencial do item O4 no esquema. Com isso, cada vez que for acrescentado ou retirado qualquer item declarado antes do item O4, o arquivo oo+número seqüencial do item O4 é redefinido com outro nome.

2.19.4 Exemplos

```
nome: arqtes e
c1(0)          N3
mem1           O4
mem2           O4
```

Para o tipo O são criados, automaticamente, os arquivos:

```
nome: oo000002 r
chave(0)       I4
valor          U500
```

e

```
nome: oo000003 r
chave(0)       I4
valor          U500
```

onde:

os valores dos itens chave dos arquivos oo000002 e oo000003 são os valores dos itens mem1 e mem2 quando diferentes de zero. Os nomes dos arquivos são formados de oo + número do item tipo O.

Para o tipo Q são criados automaticamente os arquivos:

```
nome: qqarqtmem1 r
chave(0)       I4
valor          U500
```

e

```
nome: qqarqtmem2 r
chave(0)       I4
valor          U500
```

onde: os valores dos itens chave dos arquivos qqarqumem1 e qqarqumem2 são os valores dos itens mem1 e mem2 quando diferentes de zero. Os nomes dos arquivos são formados de **qq + as quatro primeiras letras do nome do arquivo + as quatro primeiras letras do nome do item q4**.

Caso ocorra coincidência na formação do nome do arquivo **qq<nomearq>+<nomeitem>** ao se compilar o esquema(define), ocorre erro.

2.20 Esquema Criptografado

2.20.1 Propósito

Compilar um esquema criptografado, evitando que o esquema de um banco de dados *OpenBASE* possa ser visualizado pelo usuário, proporcionando maior segurança e restrições de acesso aos bancos de dados.

2.20.2 Sintaxe

```
define -x <chave> <nome do esquema criptografado>
```

2.20.3 Argumentos

-x <chave> Chave é senha utilizada na criptografia do esquema.

2.20.4 Utilização

Visando melhorar o controle de acesso a um esquema de um banco de dados *OpenBASE*, foi feita uma opção de compilação de um esquema criptografado, gerado a partir da função ENCRYPT da Opus, conforme exemplo abaixo.

2.20.5 Exemplos

O exemplo a seguir, é um programa em Opus, que faz a criptografia de um esquema. Este programa recebe como parâmetros, o esquema do banco sem ser criptografado, o esquema do banco a ser criptografado e uma senha de criptografia.

```
prog criptografia
parameters esqnor, esqcri, chave
private dynamic var1
private dynamic var2
x = memoread (esqnor, var1)  && Carrega na variável var1 conteúdo de esqnor
l = len (var1)
do while l %8 < > 0  && o tamanho da cadeia criptografada deve ser multiplo de oito
  var1 = var1 + " "
  ++l
enddo
var2= encrypt (var1,chave,l) && função que criptografa a cadeia, var1,utilizando x
* a senha chave, e atribuindo o resultado a variável var2.
x= memowrit (esqcri, var2)  && grava no esquema criptografado esqcri, o
* conteúdo de var2 já criptografado pela função encrypt
```

Após compilar o programa acima, deve-se executar o comando:

```
criptografia esqnor esqcri, abc.
```

Onde:

define -x abc esqcri	Compila um esquema criptografado (esqcri)
Criptografa	Programa executável que faz a criptografia.
Esqnor	Nome do esquema normal, sem ser criptografado.
Esqcri	Nome do esquema a ser criptografado.
Abc	Senha a ser utilizada na criptografia.

2.21 Opções de Execução do DEFINE

2.21.1 Propósito

Inibe a confirmação de criação de arquivos, após a compilação do Banco de Dados.

2.21.2 Sintaxe

```
define -s <esquema>
```

2.21.3 Utilização

-s Indica que não será perguntado ao usuário, se deseja recriar os arquivos do banco de dados. Substitui a opção **\$controle semarquivo**, utilizada no esquema do banco de dados.

-r Todos os arquivos do banco de dados, serão (re)-criados, após a compilação do esquema, sem confirmação prévia.

-g o dicionário de dados e os arquivos referente ao bloqueio do banco e arquivo de recuperação não serão recriados.

3 Programação de aplicativos

3.1 *SET CONNECTION to <host>*

3.1.1 Propósito

Conectar de forma remota dois ou mais banco de dados **na mesma máquina ou em máquinas distintas**

3.1.2 Sintaxe

SET CONECTION TO <host>

3.1.3 Argumentos:

<host> - nome do host relacionado a um endereço IP na tabela de hosts.

3.1.4 Utilização

Esta opção deve ser utilizada sempre que uma aplicação cliente-servidor desejar utilizar mais de um banco de dados Openbase,(podem ser iguais ou não) localizados em diferentes diretórios no mesmo servidor **ou em servidores distintos**. Deve-se atribuir diferentes nomes ao mesmo endereço IP do servidor que se deseja conectar, na tabela de hosts(c:\windows\hosts).

3.1.5 Exemplo

```
prog
if CONNECT("marte") = .F. && Estabeleço conexão com máquina marte - sco-unix
  ? "ERRO NA CONECAO COM ts8"
endif
database /usr1/clientes/exemplo 1 a 2 && Abro banco exemplo na máquina marte
select a
use pessoa
locate
Do While .not. eof()
  ? nome
  ? str(idade)
  wait
  continue
Enddo

if CONNECT("aix") = .F. && Estabeleço conexão com máquina aix
  ? "ERRO NA COMECAO COM AIX"
endif
database /usr1/clientes/exemplo 1 a 2 && Abro o banco exemplo na máquina aix
select a
use PESSOA
locate
Do While .not. eof()
  ? NOME1
  ? str(IDADE1)
  wait
  continue
Enddo
SET CONNECTION TO "marte" && Altero conexão para máquina sco-unix
database /usr1/clientes/exemplo 1 a 2
```

```

select a
use pessoa
locate
Do While .not. eof()
? nome_p
? str(idade)
wait
continue
Enddo

```

Observação: As máquinas de nome "aix" e "ts8", possuem diferentes endereço IP no arquivo hosts (tabela de hosts). O banco de dados exemplo é o mesmo nas duas máquinas.

3.2 *GetSockName()*

3.2.1 Propósito

Retorna o endereço IP da máquina cliente a partir de um programa opus cliente/servidor .

3.2.2 Sintaxe

```
m = GetSockName()
```

3.2.3 Utilização

Esta função é útil, quando a partir de uma programa Opus em ambiente cliente servidor(send/receive ou \$client) desejamos saber o endereço IP da máquina cliente.

3.2.4 Exemplo

```

prog
a = cliser(6667,"10.1.1.10")
? GetSockName() && retorna o endereço IP da máquina que está acessando o IP 10.1.1.10
x = cliser(0,"")

```

Não confundir com a função cadeia TTYIP () que retorna o endereço IP da estação de trabalho (cliente) que se conecta ao servidor UNIX, via emulador de terminal TELNET.

3.3 *GetPeerName()*

3.3.1 Propósito

Retorna o endereço IP da máquina servidor a partir de um programa opus cliente/servidor .

3.3.2 Sintaxe

```
m = GetPeerName()
```

3.3.3 Utilização

Esta função é útil, quando a partir de uma programa Opus em ambiente cliente servidor(send/receive ou \$client) desejamos saber o endereço IP da máquina servidor.

3.3.4 Exemplo

```

prog
a = cliser(6667,"10.1.1.10")
? GetPeerName() && retorna o endereço IP da máquina servidor , ie., 10.1.1.10
x = cliser(0,"")

```

Não confundir com a função cadeia TTYIP () que retorna o endereço IP da estação de trabalho (cliente) que se conecta ao servidor UNIX, via emulador de terminal TELNET.

3.4 Funções para controle de processos

As funções abaixo, fornecem ao usuário, informações referentes ao número e nome dos processos ativos na máquina. Valem tanto para o ambiente Unix, quanto para o Win9x, e Windows NT.

3.4.1 GetPid

3.4.1.1 Propósito

A função numérica GetPid() retorna o número do processo corrente.

3.4.1.2 Sintaxe

GetPid()

3.4.1.3 Utilização

Esta função é útil, quando desejamos, a partir de um programa Opus, saber o número do processo gerado pelo programa Opus, que está sendo executado.

3.4.2 KillProcess

3.4.2.1 Propósito

A função lógica KillProcess(<num>) termina o processo indicado por <num>.

3.4.2.2 Sintaxe

m = KillProcess (número)

3.4.2.3 Argumento

Número – número do processo que se deseja terminar

3.4.2.4 Utilização

Esta função deve ser utilizada, quando a partir de um programa Opus, desejarmos eliminar um determinado processo. Retorna TRUE se o processo indicado por <número> foi eliminado com sucesso.

3.4.3 Aprocesses

3.4.3.1 Propósito

Retorna a quantidade de processos que estão ativos

3.4.3.2 Sintaxe

tot_processo = APROCESSES ([<vet1> [,<vet2>]])

3.4.3.3 Argumentos

vet1 = Nome dos processos ativos

vet2 = Número dos processos ativos

3.4.3.4 Utilização

A função numérica APROCESSES ([<vet1> [,<vet2>]]) retorna o número de processos em execução. Se <vet1> for especificado os nomes dos processos são retornados e se <vet2> for especificado seus números são obtidos. (No ambiente Windows NT é necessária estar instalada a PSAPI.DLL e o programa ligado com a PSAPI.LIB)

3.4.3.5 Exemplo

```
n=aproceses()
```

```
decl nome[n]=space(20)
```

```
decl nume[n]=0
```

```
n=aproceses(nome,nume) && Seria retornado o nome e os respectivos números  
&& dos processos que estão sendo executados  
&& na máquina.
```

3.5 Zip()

3.5.1 Propósito

Comprimir um arquivo a partir de uma aplicação Opus.

3.5.2 Sintaxe

```
M = zip(<arqzip>,<arqori>)
```

3.5.3 Argumentos

arqzip - nome do arquivo .zip que será criado

arqori - nome do arquivo a ser compactado

3.5.4 Utilização

Esta função é útil, quando a partir de uma programa Opus , deseja-se compactar um arquivo. utilizado-se para isto o WinZip. Com isso, pode-se criar um arquivo zipado, e o mesmo pode ser enviado a outra máquina através das funções ftpsend() , SendFile(<arquivo>).

3.5.5 Exemplo

```
m = zip("teste.zip","test.txt")
```

3.6 UnZip()

3.6.1 Propósito

Descomprimir um arquivo a partir de uma aplicação Opus.

3.6.2 Sintaxe

```
m = unzip(<arqzip>,<arqdes>)
```

3.6.3 Argumentos

arqzip - nome do arquivo .zip que será descompactado

arqori - nome do arquivo a ser criado

3.6.4 Utilização

Esta função é útil, quando a partir de uma programa Opus , deseja-se descompactar um arquivo, utilizado-se para isto o WinZip. Com isso, pode-se criar um arquivo destino , extraído a partir de um arquivo compactado, caso este arquivo destino exista no arquivo compactado(.zip). Se o arquivo destino não for informado, o arquivo que existir no arquivo compactado, será criado. .

3.6.5 Exemplo

```
m = unzip("teste.zip","") && será criado o arquivo que existir dentro de teste.zip
```

3.7 FTPSEND – FTPRECEIVE

3.7.1 Propósito

Permitir o envio ou recebimento de arquivos, via ftp, a partir de um programa Opus.

3.7.2 Sintaxe

```
ftpsend (<ser>, <log>,<pas><ori>,<des>)
```

```
ftpreceive (<ser>, <log>,<pas>,<ori><des>)
```

3.7.3 Argumentos

<ser>	nome do servidor
<log>	login no servidor

<pas>	senha no servidor
<ori>	arquivo origem
<des>	diretório/arquivo destino

3.7.4 Utilização

Util em programas Opus, quando se deseja enviar ou receber arquivos de um servidor remoto, sem ter que se utilizar algum arquivo batch ou shell.

Caso o cliente seja windows 9x, NT, 2000, deve estar instalada a biblioteca WININET.lib

3.7.5 Exemplos

```

If ftpsend ("10.1.1.26", "root", "senha", "\tmp\teste", "/usr/tmp")
? "Arquivo foi enviado com sucesso"
endif

```

3.8 Bibliotecas Shared Objects(.so)

3.8.1 Propósito

Fazer com que um programa Opus acesse as bibliotecas libbd e libfacbib em tempo de execução.

3.8.2 Sintaxe

libbd.so e libfacbib.so

3.8.3 Argumentos

Nenhum

3.8.4 Utilização

Ao utilizar estas bibliotecas, uma aplicação Opus só faz acesso a elas, em tempo de execução, e não mais em tempo de linkedição. Com isso, o tamanho do programa executável diminui em até 95%.

Além disso, caso haja alguma alteração nas bibliotecas libbd e libfacbib do Openbase, não será necessário a recompilação do programa Opus.

PS: Esta opção só está disponível para o Linux

3.9 \$testitem

3.9.1 Propósito

Fazer com que um programa Opus, em tempo de execução, verifique se as características dos itens de dado estão de acordo com o dicionário de dados.

3.9.2 Sintaxe

\$testitem

3.9.3 Utilização

Um programa Opus que abra um banco de dados Openbase, ao ser compilado gera um fonte C que contém as características dos itens de dado(tipo, tamanho,no de casas decimais, se é nulo ou não). Logo sempre que houver uma mudança no esquema do banco de dados e o dicionário de dados for regenerado, o programa Opus deve ser recompilado.

Com a opção \$item se houver alguma diferença entre o dicionário de dados e o programa Opus em relação as características dos itens de dado, será emitida a mensagem de erro:

“OPUS(IteVar)=> Tipo/Tamanho do item(nome_item> Invalido”

“OPUS(VarIte)=> Tipo/Tamanho do item(nome_item> Invalido”

Esta opção é útil, quando ainda se está desenvolvendo o esquema do banco de dados.

3.10 FILENAME()

3.10.1 Propósito

Retornar o nome de um arquivo externo ou arquivo do Openbase.

3.10.2 Sintaxe

```
nome_arq = FILENAME()
```

3.10.3 Argumentos

Nenhum

3.10.4 Utilização

Esta função é útil quando em um programa Opus, necessita-se saber o nome do arquivo do banco de dados ,ou o nome do arquivo externo, que está em uso(aberto).

3.10.5 Exemplo

```
prog
database teste 1 a 2
select a
use TABELA
nome = FILENAME()
? nome && deve exibir o nome do arquivo aberto , ou seja, TABELA.
```

3.11 Arquivos Externos

3.11.1 L - Seqüencial em linha.

Este tipo de arquivo tem tamanho de registro variável e todo registro deve terminar com um caractere de nova linha (NL = \n), Tira brancos a direita.

Não permite a alteração do tamanho do registro num comando CHANGE.

Num comando DELETE o registro é preenchido com -.

A inclusão de um registro é feito no final do Arquivo, com o comando APPEND.

3.11.2 M - Seqüencial em linha.

Este tipo de arquivo tem tamanho de registro variável e todo registro deve terminar com um caractere de nova linha (NL = \n).

Tira brancos a direita. Permite a alteração do registro num comando CHANGE porque cria um novo arquivo. Num comando DELETE o registro é excluído porque cria um novo arquivo. Permite o comando INSERT para inserir um registro no meio do arquivo.

3.12 Compilação de programas

3.12.1 Opção de compilação -l

3.12.1.1 Propósito

Ao se compilar o programa Opus com a opção -l, significa que o programa executável não será gerado. Esta opção de compilação indica também que não será verificada a vigência da cópia *OpenBASE*, em caso de cópia demonstração.

3.12.1.2 Sintaxe

```
Opus -l <nome do programa>.f
```

3.12.1.3 Utilização

Útil quando se deseja apenas verificar a sintaxe de um programa OPUS.

3.12.2 Opção de compilação -r

3.12.2.1 Propósito

Ao se compilar o programa Opus com a opção -r, pode-se passar parâmetros ao compilador C.

3.12.2.2 Sintaxe

```
opus -r <lista parametros> <nome do programa>
```

3.12.2.3 Utilização

Útil quando se utiliza mais de um processador na máquina. Neste caso deve-se passar a opção DSMP (SIMETRICA (MULTIPLE PROCESS)).

3.13 Opções de Controle

3.13.1 Configurando o Ambiente

É possível selecionar as opções para o comando OPUS \$ via as variáveis de ambiente do UNIX. Estas variáveis são:

OPUSOPT

Quando selecionada indica ao compilador quais opções este deve seguir durante a compilação de um fonte OPUS, o que equivale à usar estas opções dentro do próprio fonte com o comando "\$".

DIRLIB

Quando selecionada indica o diretório onde as bibliotecas de sistema da OPUS estão localizadas (libfacbib.a e libbd.a).

DIRINC

Quando selecionada indica o diretório onde os arquivos de inclusão de sistema da OPUS estão localizados (types.h, tobjetos.h, etc.)

Exemplos

Informar ao compilador OPUS que os programas compilados irão usar a versão 5.3 e as bibliotecas do usuário. Para fazer isto, selecione as variáveis seguintes:

```
$ PATH = /V5.3:$PATH
$ DIRLIB = /V5.3/lib
$ DIRINC = /V5.3/include
$ OPUSOPT = "library = mesmalib.a, dirlib, dirinc"
$ export PATH DIRLIB DIRINC OPUSOPT
```

3.13.2 \$CONVNEED

3.13.2.1 Propósito

Evitar conversões de dados desnecessárias.

3.13.2.2 Sintaxe

```
$ CONVNEED
```

3.13.2.3 Utilização

Nos comandos de leitura (FIND, SEEK e LOCATE), o conteúdo de todo o registro lido é convertido para item de memória, além destes são convertidos os itens de outros arquivos lidos por JOIN automático. Esta opção inibe a conversão total e automática, sendo assim, os itens de banco de dados só serão convertidos no momento de sua utilização, agilizando em alguns casos o processamento do programa. A utilização desta opção deve ser criteriosa pois não permite JOINS automáticos, pois isso inviabilizaria a otimização.

3.13.3 \$ORACLE

3.13.3.1 Propósito

Permitir o acesso a uma base de dados ORACLE.

3.13.3.2 Sintaxe

\$ORACLE

3.13.3.3 Utilização

Permite que um programa OPUS acesse bases de dados ORACLE e *OpenBASE* simultaneamente. Este comando instrui ao compilador para gerar um código fonte para o pré-compilador C do ORACLE, o ProC, com extensão ".pc" ao invés de um fonte C com extensão ".c". O fluxo de compilação invoca o compilador ProC.

Os comandos de acesso a base de dados *OpenBASE* continuam ativos e a base ORACLE é acessada através do comando com sintaxe SQL do ProC.

Opções para o pré-compilador ProC podem ser passadas pela opção \$PCC_OPTIONS.

3.13.3.4 Exemplos

O exemplo a seguir mostra um programa com esta característica

```
*
* Teste Opus acessando base de dados ORACLE
* Uso concomitante do ORACLE com OpenBASE.
*
$ORACLE, savec
$pcc_options = "include=/usr/ORACLE/c/lib errors=yes host=c reclen=132"
prog
database TORACLE 1 a 2
use emp
EXEC SQL INCLUDE sqlca;
empno, mgr, sal, comm, deptno = 0
hiredate, ename, job = space(10)
username = "SCOTT"
password = "TIGER"
EXEC SQL WHENEVER SQLERROR GOTO sqlerror;
EXEC SQL CONNECT :username IDENTIFIED BY : password;
0,0 say "Conectado ORACLE ao usuario: "+username
EXEC SQL DECLARE salespeople CURSOR FOR
SELECT EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO
FROM EMP;
EXEC SQL OPEN salespeople;
EXEC SQL WHENEVER NOT FOUND GOTO end_of_fetch;
?
nlin = 0
do while .t.
EXEC SQL FETCH salespeople INTO :empno, :ename, :job,
:mgr, :hiredate, :sal, :comm, :deptno;
incr nlin
replace empno, ename, job, mgr, hiredate, sal, comm, deptno
insert
enddo

LABEL SQL end_of_fetch
EXEC SQL CLOSE salespeople;
? "Transferidas ", nlin, " linha", iif(nlin = 1, ".", "s.")
EXEC SQL COMMIT WORK RELEASE;
?
? "Tenha um ", "BOM DIA " bold
quit (0)

LABEL SQL sqlerror
```

```

clear
? get_string (sqlca.sqlerrm.sqlerrmc, get_short(sqlca.sqlerrm.sqlerrml))
EXEC SQL WHENEVER SQLERROR CONTINUE;
EXEC SQL ROLLBACK WORK RELEASE;
quit (1)

```

3.13.4 \$ODBC

3.13.4.1 Propósito

Permite que um programa OPUS acesse um banco de dados SQL através do driver ODBC. O banco de dados SQL é derivado a partir do banco de dados *OpenBASE*, através do utilitário BDTSQL. Os dados podem ser descarregados do banco *OpenBASE* no formato <INSERT INTO <arq> values (<val1>,...<valn>) através da opção -q no *bddesc* e incluídos no banco SQL.

3.13.4.2 Sintaxe

```

$odbc
set odbc on

```

3.13.4.3 Utilização

Permite que alguns comandos e funções da OPUS gerem código para ODBC, podendo acessar bancos de dados SQL.

Comandos:

database, find, locate [start], seek, insert, delete, modify, empty, save, restore, set order to, sort on, unlock e undo.

Funções

chain(), reccount(), indreccountt(). A função automática também gera código para ODBC.

O comando **database** tem a seguinte sintaxe:

```
database <banco><seg><nivel><modo abertura>
```

onde:

<banco> é o nome do DSN utilizado no ODBC.

<seg> é ignorado no ODBC.

<nivel> deve ser: "UID = <usuário>; PWD = <senha>;"

<modo> é o modo de abertura do banco.

Deve existir na máquina o diretório \USR\TSGBD\ODBC\, para que ao ser acessado durante a compilação de um programa Opus, o banco SQL, cria um dicionário *OpenBASE* neste diretório. Se este dicionário existir o banco SQL não é acessado. Se o banco SQL for alterado, o dicionário *OpenBASE* deve ser removido do diretório \usr\tsgbd\odbc, para que possa ser recriado.

Com esta opção, um programa Opus pode acessar um banco de dados SQL, sem que o código fonte do programa seja alterado.

Basta apenas existir o driver ODBC instalado e configurado na máquina.

3.13.5 \$PCC_OPTIONS

3.13.5.1 Propósito

Passar parâmetros para o compilador ProC.

3.13.5.2 Sintaxe

```
$ PCC_OPTIONS = <expC>
```

3.13.5.3 Argumentos

<expC> Representam as opções para o pré-compilador ProC.

3.13.5.4 Utilização

Utiliza-se esta opção em conjunto da opção \$ORACLE, para passar parâmetros ao pré-compilador ProC da ORACLE.

3.13.5.5 Exemplos

O exemplo a seguir mostra a utilização deste comando.

```
*
* Teste Opus acessando base de dados ORACLE
* Uso concomitante do ORACLE com OpenBASE.
*
$oracle, savec
$PCC_OPTIONS = "include=/usr/ORACLE/c/lib errors=yes host=c ireclen=132"
prog
database TORACLE 1 a 2
use emp
EXEC SQL INCLUDE sqlca;
...
...
```

3.14 Comandos

3.14.1 &

3.14.1.1 Propósito

Máscara de Edição para & valores numéricos.

Recebendo somente números, preenchendo com zeros à esquerda ou deixando brancos se o campo estiver vazio.

3.14.1.2 Exemplos

```
prog
tel = space (10)
@ 10,20 say "telefone..." get tel pic (&&&) &&& - &&&&
read
```

3.14.2 CANCEL

3.14.2.1 Propósito

Encerrar um programa retornando o controle ao processo chamador.

3.14.2.2 Sintaxe

```
CANCEL
QUIT [(<numero>)]
```

3.14.2.3 Argumentos

<numero> status do término do programa.

3.14.2.4 Utilização

Na sua execução fecha todos os arquivos abertos e o banco de dados utilizado.

Através dos comando QUIT é possível retornar um código (para o sistema operacional) que representará o status de fim do programa.

Se o programa foi invocado por um outro programa escrito em OPUS, esse código pode ser capturado pela função DBERR(). Sob o sistema operacional UNIX, se um processo escrito em outra linguagem houver invocado um programa desenvolvido em OPUS, o status do comando QUIT estará armazenado na variável ambiental \$?.

O comando CANCEL sempre retorna 0 (zero).

3.14.2.5 Exemplos

O exemplo a seguir mostra a utilização dos comandos CANCEL e QUIT retornando um código de erro.

```
IF tudobem && tudo bem é uma variável lógica
CANCEL
ELSE
```

```
QUIT(1)
ENDIF
```

3.14.2.6 **Consulte**

EXIT, FUNCTION, PROCEDURE, PROGRAM, RETURN, RUN, SLEEP e DBERR().

3.14.3 **DECLARE LONG**

Declare Long <Vetor>[<numero>]=<expressão>

A palavra LONG indica que a dimensão do vetor <numero> pode ser maior que 65535.

Vetor pode ser acessado ou modificado utilizando os comandos STORE, GET, ACEITE, INCR, DECR, mas as rotinas AFILL, ASORT, etc... não podem indicar este vetor.

3.14.3.1 **Exemplos**

```
declare long vetopt[70000]=space (70).
```

3.14.3.2 **Consulte**

DECLARE, STORE, GET, INCR, DECR, AFILL(), ASORT().

3.14.4 **Cláusula DYNAMIC**

3.14.4.1 **Propósito**

Serve para alocar uma variável dinamicamente, sem tamanho pré-definido.

A variável pode ser usada nos comandos MEMOGET, MEMOPUT, MEMOREAD, MEMOWRIT, MLCOUNT, MEMOLINE, MEMOEDIT, MEMOSAVE, MEMOSEEK, LEN, ENCRYPT, ?, + (Concatenação), = (Atribuição), TRIM, EMPTY, UPPER, LOWER, WORD, STRTRAN, SUBSTR, LEFT, RIGHT, LTRIM, ALLTRIM, HTML_GET, HTML_PUT, REPLACE.

As variáveis dinâmicas podem ser passadas como parâmetros.

A cláusula DYNAMIC pode ser associada com os comandos Static, Public e Private.

3.14.4.2 **Sintaxe**

```
STATIC DYNAMIC <VARIÁVEL> [(C)]
```

3.14.4.3 **Argumentos**

Variável – Nome da variável de trabalho.

3.14.4.4 **Exemplos**

```
private dynamic stor(c)
public dynamic stor(c)
static dynamic stor(c)
```

3.14.4.5 **Consulte**

PRIVATE, PUBLIC, STATIC.

3.14.5 **EMPTY**

3.14.5.1 **Propósito**

Esvaziar todo um arquivo.

3.14.5.2 **Sintaxe**

```
EMPTY
ZAP
```

3.14.5.3 **Utilização**

Esvaziar todo o conteúdo do arquivo selecionado.

3.14.5.4 **Exemplos**

```
prog
database EXEMPLO 1 a 2
use PESSOA
empty          && Irá esvaziar o arquivo PESSOA.
```

3.14.5.5 *Consulte*

ZAP.

3.14.6 GO [BYTE]

3.14.6.1 *Observações*

A opção BYTE foi criada para posicionar em arquivos externos, mesmo tipo L ou M. <expN> indica o byte a posicionar.

No caso de arquivos externos tipo S ou R, a <expN> corresponde ao número do registro menos 1 vezes o tamanho do registro.

No caso de arquivos externos tipo L ou M, <expN> corresponde a soma dos tamanhos dos registros mais 1 (UNIX) ou mais 2 (DOS) anteriores ao registro.

3.14.6.2 *Exemplos*

```
use*
  abc s len(20)
  a u20
enduse
go 2          && equivale a go byte 20
use*
  abc 1 len(20)
  a u20
enduse
decl posi[3]=0
locate
p=0
for i=1 to 3
  posi[i]=p
  p=p+len(a)+2
  continue
next
go byte posi[2]  && posiciona no segundo registro
```

3.14.7 REINDEX

3.14.7.1 *Propósito*

Re-indexar um arquivo externo.

3.14.7.2 *Sintaxe*

```
REINDEX
```

3.14.7.3 *Argumentos*

É utilizado para reorganizar o índice de um arquivo externo que está sendo usado. Deve ser utilizado quando a função BADINDEX() retornar verdadeiro (.T.).

3.14.7.4 *Exemplos*

```
PROG
USE *
  diary I LEN (60)
  phone_num n10KEY
  name          U20
  addr          U30
ENDUSE
IF BADINDEX ( )
  ? "reorganiza o arquivo diário"
  REINDEX
ENDIF
```

3.14.8 Comandos de Tabulação

3.14.8.1 Sintaxe

```
CATEGORIZE <var> <valor1> ... <valorn>
BREAK <var1> ... <varn>
RECODE <var1> <var2> [ ... <varn>]
    <val1> <val2>[ ... <valn>]
    ...
    <val1> <val2> [ ... <valn>]
GROUP <var1> <var2>
    <val1> <val2>
    ...
    <val1> <val2>
SET FORMAT TO [<proc>]
SET LEVEL TO <num>
SET INVALID TO <num>
CROSS <ident> <varc1> <varc2> [QT = <varq>] [WT = <varw>]
    [BY=<varb1>...<varbn>]
TABULATE <ident> <varc1><varq1> [<varq2>] ... [<varqn>] [WT = <varw>]
    [BY=<varb1>...<varbn>]
TERMINATE
CONVERT <varqd1> [...<varqdn>] <varc1> [...<varen>]
    <varqo1> [...<varqon1>] <valc1> [...<valcn1>]
    ...
    <varqo1n> [...<varqonn>] <valcn> [...<valcnn>]
```

Onde :

<ident> literal identificador da matriz (tamanho = 24)

<varc.> variável numérica categorizada

<varq.> variável numérica quantitativa

<val..> valor numérico

3.14.9 Semântica

O comando CATEGORIZE indica as categorias de uma variável.

O comando BREAK indica as variáveis de quebra, isto é, se houver mudança de valor de qualquer variável as matrizes em memória deverão ser gravadas, contendo os campos de controle o valor destas variáveis.

O comando RECODE indica a criação de uma variável categorizada a partir dos valores de uma ou mais variáveis.

O comando GROUP indica a criação de uma variável categorizada a partir de uma variável quantitativa.

O comando SET FORMAT TO indica a subrotina a gravar os valores das matrizes.

Se o nome da subrotina não for informado, as matrizes são impressas.

Os valores são passados para a subrotina num parâmetro cadeia no formato:

<ident>:<varq1>;<varq2>; : <varb1>;<varb2>;<varb3>;<varb4>; :<val1>;...<valn>

identificação : valores das variáveis de quebra : valores das variáveis por : valores das células da matriz Os grupos de valores são separados por : (dois pontos) e os valores são separados por ; (ponto e vírgula).

Todos os campos são em formato texto (ASCII).

O comando SET LEVEL TO indica o nível a gravar no campo de controle da matriz de saída.

O comando SET INVALID TO indica o valor da categoria inválida para as variáveis categorizadas.

O comando CROSS indica uma matriz a ser calculada com número de linhas igual ao número de categorias da primeira variável e número de colunas igual ao número de categoria da segunda variável. Se a variável QT não for especificada soma um às células da matriz, caso contrário, soma o valor desta variável às células. Se a variável WT for especificada, multiplica-se este valor pelo valor a ser somado nas células. Se as variáveis BY forem especificadas, calcula-se uma matriz para cada combinação de valores destas variáveis, gravando-se estes valores em campo de controle.

O comando TABULATE indica uma matriz a ser calculada com número de linhas igual ao número de categorias da variável categorizada e número de colunas igual ao número de variáveis quantitativas especificadas. Para cada registro selecionado, soma-se o valor de cada variável quantitativa nas células da linha correspondente a categoria da variável categorizada. Se a variável WT for especificada, multiplica-se este valor pelo valor a ser somado nas células. Se as variáveis BY forem especificadas, calcula-se uma matriz para cada combinação de valores destas variáveis, gravando-se estes valores em campo de controle. O comando TERMINATE indica a gravação final das tabelas em memória.

O comando CONVERT indica a conversão de valores de uma ou mais variáveis quantitativas origem para uma ou mais variáveis quantitativas destino associadas a categorias de uma ou mais variáveis categorizadas, causando no comando CROSS ou TABULATE que utilizem uma ou mais variáveis quantitativas destino a soma dos valores das variáveis quantitativas origem nas células das matrizes indicadas pelas categorias especificadas.

3.15 Cláusulas SET

3.15.1 SET ALARM TO

3.15.1.1 Propósito

Determina o tempo em segundo, que o bdserv inicializado por uma aplicação cliente, deva permanecer ativo, sem que haja comunicação da aplicação cliente.

3.15.1.2 Sintaxe

SET ALARM TO [<tempo>]

3.15.1.3 Argumentos

tempo – tempo em segundos, que o bdserv filho, inicializado pela aplicação cliente, permanecerá ativo, sem que haja, comunicação da aplicação cliente.

3.15.1.4 Utilização

Evitar que um processo filho bdserv, permaneça ativo infinitamente, caso a aplicação cliente seja finalizada de forma anormal, sem finalizar o bdserv (disconnect).

3.15.1.5 Exemplos

SET ALARM TO 3.600 Indica que após 3.600 segundos (1 hora) sem comunicação, o bdserv sairá do ar.

SET ALARM TO Não determina nenhum tempo (tempo infinito), para que o bdserv saia do ar, caso a aplicação cliente não se comunique com o servidor (bdserv).

3.15.2 SET CENTURYWINDOW TO < num >

Para valores menores que num, o ano passa a ser admitido como 20num ou seja dois mil + o ano setado em num.

Para valores maiores do que num, o ano passa a ser admitido como 19num ou seja mil novecentos + ano setado em num.

3.15.2.1 Exemplos

Set centurywindow to 20.

Para ano menor do que 20, por exemplo 19, é admitido o ano 2019.

Para anos maiores ou igual a 20, por exemplo 21, é admitido o ano 1921.

3.15.3 SET CURSORPOS ON|OFF

3.15.3.1 Propósito

Desabilita o posicionamento do cursor.

3.15.3.2 Sintaxe

SET CURSORPOS ON|OFF

3.15.3.3 Utilização

Útil, quando se deseja exibir alguma expressão ou valor, através do comando ?, a partir da posição corrente do cursor, onde o programa foi executado.

3.15.3.4 Argumentos

OFF	Retira o posicionamento do cursor, fazendo com que a posição corrente do cursor onde o programa foi executado, seja assumida como posição inicial.
ON	Faz com que o posicionamento do cursor, seja controlado pelo programa (linha, coluna).

3.15.3.5 Exemplos

```
$noclear
prog
xind = 0
set cursorpos off
Do while xind < 200
  ?xind
  sleep (1)
  ++xind
Enddo
Return
```

3.15.4 SET DEMO on/off

3.15.4.1 Propósito

Permite a abertura de um banco de dados e seus respectivos arquivos de dados e de índice, em um CDROM. Os arquivos são lidos, e nada é gravado.

3.15.4.2 Sintaxe

```
set demo on
```

3.15.4.3 Utilização

Útil quando se deseja executar um programa a partir de um CDROM, estando o banco de dados no CDROM. Com esta opção ligada, o banco é aberto no diretório corrente onde foi executado o programa. É necessário que o percurso onde está o banco de dados e seus arquivos (dados e índices), sejam iguais, tanto no CDROM quanto no disco rígido. Caso contrário deve-se executar o comando NAME = "X:<Percurso BD> <Nome BD>". Neste caso, o banco será aberto de acordo com o conteúdo da variável NAME. Os comandos de gravação, tais como INSERT, DELETE, MODIFY, são ignorados.

3.15.5 SET EXACT

3.15.5.1 Propósito

Determina que ao se comparar expressões cadeias, estas sejam exatamente iguais. A opção set exact on, passa a ser default na Opus, ou seja, se no programa principal não for dado o comando, set exact off, assume-se que set exact on está habilitado.

3.15.5.2 Sintaxe

```
Set exact on
```

3.15.5.3 Utilização

Ao se tornar default, o set exact on, evita que ao se comparar uma expressão cadeia não vazia com uma expressão cadeia vazia, o resultado seja verdadeiro.

3.15.6 SET MESSLEN TO <num>

3.15.6.1 Propósito

Indica o tamanho máximo da mensagem que um programa cliente, em Opus, utiliza para se comunicar com o servidor (bdserv).

3.15.6.2 Sintaxe

SET MESSLEN TO <NUM>

3.15.6.3 Argumentos

<num> Tamanho, em bytes, da mensagem a ser enviada pelo programa cliente, na conexão com o servidor.

3.15.6.4 Utilização

Permite que um cliente, envie uma mensagem de tamanho variável, na comunicação com o servidor. Caso não seja utilizado, o valor default da mensagem é de 1000 bytes. Útil na comunicação de redes WAN e LAN, onde se utiliza um MTU (Maximum Transfer Unit) de 296 bytes, já que neste caso, o tamanho máximo da mensagem que um programa cliente em Opus enviaria ao servidor, na abertura do banco, seria de 296 bytes (Set Messlen to 296), e não 1000 bytes.

3.15.7 SET NOTIMESTAMP ON

3.15.7.1 Propósito

Faz com que um registro que possua como primeiro item, um item T8, não tenham o valor deste item automaticamente atualizado, quando tal registro for incluído ou alterado.

3.15.7.2 Sintaxe

SET NOTIMESTAMP ON

3.15.7.3 Utilização

Útil em programas feitos para correções no banco de dados, onde não se deseja que registros que possuam itens do tipo T8, tenham o valor destes itens atualizado com a data e hora do sistema operacional, quando os registros forem incluídos ou alterados.

3.15.8 SET PHONEND ON

3.15.8.1 Propósito

Indica que na rotina PHONETIC, não será eliminado o gênero (o e a finais) e o número (S e outras terminações de plural).

3.15.8.2 Exemplos

```
? PHONETIC ("MENESES") se exhibirá MENE
? PHONETIC ("MENEZES") se exhibirá MENEZ
SET PHONEND ON
? PHONETIC ("MENESES") se exhibirá MENEZES
? PHONETIC ("MENEZES") se exhibirá MENEZES
```

3.15.8.3 Consulte

PHONAME() PHONETS(), PHONETIC (), PHONOPT E WORD () .

3.15.9 SET PPID ON|OFF

3.15.9.1 Propósito

Indica que de tempos em tempos, será verificado se o número do processo pai, do processo de um determinado programa, é 1 (INIT). Se for o programa será terminado.

3.15.9.2 Sintaxe

SET PPID ON <número de segundos>

3.15.9.3 Utilização

Útil em casos que na emulação de terminais (Telnet, Netterm) ou conexão via "Terminal burro" (por exemplo, terminal server), ao se desfazer a conexão com o servidor, estando um programa no ar, tal programa é terminado, mas o programa "chamado" por este programa permanece no ar.

3.15.10 SET SORT

3.15.10.1 Propósito

Gravar em um arquivo os endereços dos registros ordenados.

3.15.10.2 Sintaxe

SET SORT TO [<expC>]

3.15.10.3 Argumentos

<expC> Representa um nome de arquivo.

3.15.10.4 Utilização

Quando se informar , através deste comando, a <exp.C> especificando o nome do arquivo, sempre que for executado um comando SORT ON os endereços dos registros ordenados serão gravados neste arquivo. Para se desligar esta característica basta invocar o comando SET SORT TO omitindo a expressão caractere.

Da mesma forma ocorre no uso do LOCATE, pois sempre que se informar um arquivo em SET SORT TO, os registros lidos pelo comando LOCATE e pelo comando CONTINUE serão lidos na ordem especificada por este arquivo.

3.15.11 SET STRLERR ON

3.15.11.1 Sintaxe

OFF (default)

Se esta opção estiver ligada, as variáveis cadeias ao serem convertidas para itens tipo U tem seu tamanho verificado. Se maiores que o tamanho do item é emitida a mensagem:

OPUS (varite) = estouro em conversão de cadeia para item tipo U.

3.15.12 SET TIMEOUT

3.15.12.1 Propósito

Indica o tempo (Timeout) que um programa cliente espera pela conexão ao servidor.

3.15.12.2 Sintaxe

SET TIMEOUT TO <num>.

3.15.12.3 Argumentos

<num> Representa um número em segundos.

3.15.12.4 Utilização

Permite o aumento do número default (atualmente 10), para um número maior, quando sua rede estiver lenta.

3.15.13 SET TRUNCATE

3.15.13.1 Propósito

Truncar valores numéricos maiores que suas máscaras.

3.15.13.2 Sintaxe

SET TRUNCATE ON / off

3.15.13.3 Utilização

Quando ligado (ON), os valores numéricos que tenham mais dígitos que os definidos em suas máscaras de edição (PICTURE E TRANSFORM), são truncados perdendo os dígitos mais a esquerda. Para que isto não ocorra, basta desligar (OFF) este comando. Neste caso, serão exibidos asteriscos no lugar dos valores truncados.

3.15.13.4 Exemplos

O exemplo a seguir mostra como utilizar este comando.

PROG

SET DECIMALS TO 2 ON var_num

```

var_num = 327654.88
@ 10,10 SAY "Valor Truncado (TRUNCATE ON) = "
@ 10,45 SAY var_num PIC "99,99" && Exibira 54,88
@ 11,45 SAY TRANSFORM ("99,99", var_num) && Exibira 54,88
SET TRUNCATE OFF
@ 13,10 SAY "Valor Asteriscos (TRUNCATE OFF) = "
@ 13,45 SAY var_num PIC "99,99" && Exibira **,**
@ 14,45 SAY TRANSFORM ("99,99", var_num) && Exibira **,**

```

3.16 FUNÇÕES

3.16.1 ADIR()

<varn>=ADIR(<expc> [,<vetn>[,<vett>[,<veth>[,<vetd>[,<veta>]]]]))

Retorna o número de arquivos encontrados em um diretório e opcionalmente características destes arquivos em vetores.

A expressão cadeia <expc> contém o nome do diretório e nome de arquivos a pesquisar.

O nome do diretório é opcional. O nome do arquivo pode conter "*" (asterisco) para indicar quaisquer caracteres.

O vetor cadeia <vetn> deve ter elementos de tamanho no mínimo 14 e conterá os nomes dos arquivos encontrados.

O vetor numérico <vett> conterá os tamanhos dos arquivos encontrados.

O vetor <veth> deve ter elementos de tamanho no mínimo 8 e conterá as horas de atualização dos arquivos encontrados.

O vetor <vetd> deve ter elementos de tamanho no mínimo 8 e conterá as datas de alteração dos arquivos encontrados.

O vetor <veta> deve ter elementos de tamanho no mínimo 6 e conterá os atributos dos arquivos encontrados.

Em DOS os atributos são:

para arquivo RDONLY	R	só leitura
para arquivo HIDDEN	H	escondido
para arquivo SYSTEM	S	do sistema
para arquivo VOLID	V	
para arquivo SUBDIR	D	diretório
para arquivo ARCH	A	arquivo normal

Em UNIX os atributos são:

para arquivos IFREG	A	arquivo normal;
para arquivos IFBLK	B	arquivo bloco
para arquivos IFDIR	D	diretório
para arquivos IFCHR	C	arquivo caracter
para arquivos IFIFO	F	arquivo fifo

3.16.1.1 Exemplos

```

n=ADIR("*.f")
if n>0
  decl v1[n]=space(14)
  decl v2[n]=0
  decl v3[n]=space(8)
  decl v4[n]=space(8)
  decl v5[n]=space(6)
  n=ADIR("*.f",v1,v2,v3,v4,v5)
  for i=1 to n
    ?"nome=",v1[i],"tamanho=",v2[i],"hora=",v3[i],"data=",v4[i],"atributos=",v5[i]
  next
endif

```

3.16.2 ABASE()

3.16.2.1 Propósito

Retornar o número de itens básicos (não redefinidos) ou preencher com a definição do arquivo selecionado.

3.16.2.2 Argumentos

- <vetC1> representa o nome do vetor que será preenchido com o nome dos itens básicos do arquivo. Deve ter no mínimo 12 caracteres.
- <vetC2> representa o nome do vetor que será preenchido com o tipo dos itens básicos do arquivo. Deve ter no mínimo 1 caracter.
- <vetN1> representa o nome do vetor que será preenchido com o tamanho dos itens básicos do arquivo.
- <vetN2> representa o nome do vetor que será preenchido com o número de casas decimais dos itens básicos numéricos do arquivo.

3.16.2.3 Utilização

Retorna a quantidade de itens básicos (não redefinidos) do arquivo selecionado preenchendo, opcionalmente, vetores com a característica do registro do arquivo de banco de dados. Quando utilizada sem argumentos, retorna o número de itens básicos do arquivo. Os vetores devem ser declarados com o número de elementos igual ao número de itens básicos do arquivo.

3.16.2.4 Exemplos

O exemplo a seguir mostra-nos como declarar vetores com o número de elementos igual ao número de itens básicos do registro.

```
USE tab_func
qtd_item = ABASE()  && Retorna o número de itens básicos do registro
DECLARE vet_nom[qtd_item] = SPACE(12)
DECLARE vet_tip[qtd_item] = " "
* Exibe a quantidade de itens básicos (qtd_item) e preenche os
* vetores
? ABASE(vet_nom, vet_tip)
```

3.16.3 BADINDEX()

3.16.3.1 Propósito

Verificar a consistência de um arquivo externo indexado.

3.16.3.2 Sintaxe

```
BADINDEX ( )
```

3.16.3.3 Utilização

Retorna verdade (.T.), quando não existir coerência entre o arquivo de dados e seu arquivo de índice, de um arquivo tipo I e X.

Caso contrário, retorna falso (.F.), indicando a exata relação entre dados e índice.

Quando BADINDEX() retornar verdadeiro (.T.), indicando que o arquivo de índice está incorreto, é aconselhável utilizar o comando REINDEX para corrigi-lo.

3.16.3.4 Exemplos

O exemplo a seguir demonstra a utilização desta função.

```
PROG
USE *
    agenda I LEN(60)
    tel_num  n10 KEY
    nome     u20
    endereco u30
ENDUSE
IF BADINDEX ( )
```

```
? "Reindexando arquivo agenda"  
REINDEX  
ENDIF
```

3.16.3.5 *Consulte*

REINDEX().

3.16.4 **CENTSECONDS ()**

3.16.4.1 *Propósito*

Retorna o número de centésimos de segundos da hora atual.

3.16.5 **CHAIN ()**

3.16.5.1 *Propósito*

Retorna o número de registros inicializados com a cadeia utilizada na pesquisa `seek <item> $=cadeia.`

3.16.5.2 *Sintaxe*

```
Seek <item do banco>$= <string>  
? chain()
```

3.16.5.3 *Argumentos*

<item do banco> Campo do arquivo utilizado na pesquisa.
<cadeia de caracter> Cadeia de caracter a ser pesquisada.

3.16.5.4 *Exemplos*

```
select a  
use pessoa  
Seek nome$="Ana"  
?chain && Exibe a quantidade de nomes inicializados com Ana.
```

3.16.6 **CHARTOEM**

3.16.6.1 *Propósito*

Converter uma cadeia escrita em um aplicativo (editor texto) Windows, para o formato ASCII, utilizada em um aplicativo não Windows (console application).

3.16.6.2 *Sintaxe*

```
CharToOem ("texto")
```

3.16.6.3 *Argumentos*

"texto" Representa a expressão cadeia a ser convertida para o formato ASCII.

3.16.6.4 *Utilização*

Esta função é útil, quando se deseja exibir vogais acentuadas e o cedilha, em um texto escrito em um editor de textos do Windows (Notepad, Wordpad), em um aplicativo que rode em console application.

3.16.6.5 *Exemplos*

O programa abaixo foi escrito no wordpad e compilado na OPUS (console application):

```
prog  
var=CharToOem ("Atenção")  
010,10 say var pec "xxxxxxx"  
wait  
return
```

3.16.7 **CONNECT ()**

3.16.7.1 *Propósito*

Inicializar a conexão com um servidor.

3.16.7.2 *Sintaxe*

Connect (<host>)

3.16.7.3 *Argumentos*

<host> Nome ou endereço IP do servidor a se conectar.

3.16.7.4 *Utilização*

Função lógica que estabelece uma conexão com uma máquina servidora. Se já houver uma conexão anterior esta é mantida.

A cada conexão, um arquivo bdserv é carregado no servidor, indicando que tal conexão foi feita.

Inicialmente, antes que qualquer conexão cliente-servidor seja estabelecida, é necessário que pelo menos um bdserv esteja ativo no servidor.

3.16.8 DISCONNECT ()

3.16.8.1 *Propósito*

Finaliza a conexão com um servidor.

3.16.8.2 *Sintaxe*

Disconnect (<host>)

3.16.8.3 *Argumentos*

<host> Nome ou endereço IP do servidor a se conectar.

3.16.8.4 *Utilização*

Função lógica que encerra a conexão estabelecida com um servidor, através da função connect.

Retira do ar o bdserv, ativado pela função connect, evitando com isso que haja vários processos (bdserv) ativos, sem necessidade.

3.16.9 ENCRYPT ()

3.16.9.1 *Propósito*

Criptografar ou De-criptografar uma cadeia de caracter.

3.16.9.2 *Sintaxe*

<varc> = encrypt (<expc1>, <exp2>, <expn>)

3.16.9.3 *Argumentos*

varc	Recebe o valor criptografado da cadeia expc 1.
Expc1	Cadeia de caracter a ser criptografada.
Expec2	Cadeia de caracter, que funciona como senha, na de-criptografia de expc 1. Deve Ter tamanho máximo de 8 posições.
Expn	Expressão numérica, podendo ser: 0 – Criptografa 1 – Des-criptografia

3.16.9.4 *Utilização*

Criptografa uma cadeia de caracteres cujo tamanho deve ser múltiplo de 8, a partir de uma senha (cadeia caracter). O valor criptografado da cadeia de caracter é gerado como retorno da função. Tal valor é de-criptografado a partir da senha (cadeia de caracter) utilizado na criptografia.

3.16.9.5 *Exemplos*

```
res = encrypt ("12345678", "abc", 1)
? encrypt (mes, "abc", 0 "      && retorna 12345678
```

3.16.10 FACUM ()

3.16.10.1 *Propósito*

Acumular somatório de itens numéricos.

3.16.10.2 **Sintaxe**

FACUM (<exp.C>, <vet.C>, <vet.N>)

3.16.10.3 **Argumentos**

- <exp.C> Representa o nome de um arquivo do banco de dados.
- <vet.C> Representa um vetor caractere com nome dos itens que serão acumulados.
- <vet.N> Representa um vetor numérico onde serão armazenados os resultados das totalizações.

3.16.10.4 **Utilização**

Esta função lê sequencialmente o arquivo <exp.C> acumulando em <vet.N> os valores dos itens definidos em <vet.C>.

Os itens definidos em <vet.C> devem ser de tipo numérico (tipos N, P, S, C, B, I ou F).

Para cada elemento do vetor <vetC> corresponde um elemento do vetor <vetN> com o resultado da acumulação do item correspondente.

Esta função quando bem sucedida retorna 0 (zero). Caso contrário retorna -1, neste caso verificar se algum dos itens passados em <vetC> não numérico, a quantidade de elemento de <vetN> corresponde a quantidade de elementos de <vetC> ou se ocorreu erro durante o acesso ao banco de dados (ver se DBERR() # 0).

3.16.11 **FAFIELDS()**

3.16.11.1 **Propósito**

Retornar o número de itens de um arquivo.

3.16.11.2 **Sintaxe**

FAFIELDS (<expC>[, <vetC1>[, <vetC2>[, <vetN1>[, <vetN2>]]]])

3.16.11.3 **Argumentos**

- <expC> Representa o nome do arquivo do banco de dados.
- <vetC1> Representa o nome do vetor que será preenchido com o nome dos itens do arquivo. Deve ter no mínimo 12 caracteres.
- <vetC2> Representa o nome do vetor que será preenchido com o tipo dos itens. Deve ter no mínimo 1 caractere.
- <vetN1> Representa o nome do vetor que será preenchido com o tamanho dos itens.
- <vetN2> Representa o nome do vetor que será preenchido com o número de casas decimais dos itens numéricos.

3.16.11.4 **Utilização**

Retorna a quantidade de itens do arquivo <exp.C> preenchendo, opcionalmente, vetores com a característica do registro do arquivo de banco de dados.

Os vetores devem ser declarados com o número de elementos igual ao número de itens do arquivo, para isso basta invocar FAFIELDS() antes de declarar os vetores.

3.16.11.5 **Exemplos**

O exemplo a seguir declara vetores com o número de elementos igual ao número de itens do registro.

```
? FOPEN ("bdemp1", "a", 33, 2) && Abre o banco de dados
qtd_item = FAFIELDS("tab_func")
DECLARE vet_nom[qtd_item] = SPACE(12)
DECLARE vet_tip[qtd_item] = " "
* Exibe a quantidade de itens do arquivo
```

3.16.12 **FAJOINS()**

3.16.12.1 **Propósito**

Retornar o número de itens que podem ser lido através de JOINS.

3.16.12.2 **Sintaxe**

FAJOINS (<expC>[, <vetC1>[, <vetC2>[, <vetN1>[, <vetN2>]]]])

3.16.12.3 Argumentos

- <expC>** Representa o nome do arquivo do banco de dados.
- <vetC1>** Representa o nome do vetor que será preenchido com o nome dos itens.
Deve ter no mínimo 12 caracteres.
- <vetC2>** Representa o nome do vetor que será preenchido com o tipo dos itens.
Deve ter no mínimo 1 caractere.
- <vetN1>** Representa o nome do vetor que será preenchido com o tamanho dos itens.
- <vetN2>** Representa o nome do vetor que será preenchido com o número de casas decimais dos itens numéricos.

3.16.12.4 Utilização

Retorna a quantidade de itens que podem ser lidos automaticamente a partir do arquivo definido por <exp.C>, preenchendo, opcionalmente, vetores com as características destes itens.

Os itens retornados pertencem aos arquivos que mantenham relação com o arquivo <exp.C>. Os vetores devem ser declarados com o número de elementos igual ao número de itens que permitam junções com o arquivo selecionado, para isso basta invocar a FAJOINS() antes de declarar os vetores.ens (qtd_item) e preenche os vetores

3.16.12.5 Exemplos

O exemplo a seguir declara vetores com o número de elementos igual ao número de itens lidos a partir do arquivo desejado.

```
? FOPEN ("bdemp1", "a", 33, 2) && Abre o banco de dados.  
qtd_item = FAJOINS ("tab_func") && Retorna o numero de itens  
DECLARE vet_nom[qtd_item] = SPACE(12)  
DECLARE vet_tip[qtd_item] = " "  
* Exibe a quantidade de itens (qtd_item) e preenche os vetores  
* com a características dos itens  
? FAJOINS ("tab_func", vet_nom, vet_tip)  
? FAFIELDS("tab_func", vet_nom, vet_tip)
```

3.16.13 FAKEYS()

3.16.13.1 Propósito

Retornar o número de chaves de um arquivo.

3.16.13.2 Sintaxe

```
FAKEYS (<expC>[, <vetC1>[, <vetC2>[, <vetN1>[, <vetN2>]]]])
```

3.16.13.3 Argumentos

- <expC>** Representa o nome do arquivo do banco de dados.
- <vetC1>** Representa o nome do vetor que será preenchido com o nome das chaves.
Deve ter no mínimo 12 caracteres.
- <vetC2>** Representa o nome do vetor que será preenchido com o tipo das chaves.
Deve ter no mínimo 1 caractere.
- <vetN1>** Representa o nome do vetor que será preenchido com o tamanho das chaves.
- <vetN2>** Representa o nome do vetor que será preenchido com o número de casas decimais das chaves numéricas.

3.16.13.4 Utilização

Retorna a quantidade de chaves do arquivo definido por <exp.C>, preenchendo, opcionalmente, vetores com as características destas chaves.

Os vetores devem ser declarados com o número de elementos igual ao número de chaves do arquivo, para isso basta invocar FAKEYS() antes de se declarar os vetores.

3.16.13.5 Exemplos

O exemplo a seguir declara vetores com o número de elementos igual ao número de chaves do registro.

```

? FOPEN ("bdemp1", "a", 33, 2)
qtd_item = FAKEYS ("tab_func")
DECLARE vet_nom[qtd_item] = SPACE(12)
DECLARE vet_tip[qtd_item] = " "
* Exibe a quantidade de chaves (qtd_item) e preenche vetores
? FAKEYS ("tab_func", vet_nom, vet_tip)

```

3.16.14 FCLOSE ()

3.16.14.1 Propósito

Fechar o banco de dados em uso.

3.16.14.2 Sintaxe

```
FCLOSE ( )
```

3.16.14.3 Utilização

Fecha o banco de dados em uso, retornando um valor numérico correspondente ao código de erro desta operação (DBERR).

Quando o banco é fechado com sucesso o valor retornado é igual a zero. Caso seja diferente de 0, este valor pode ser utilizado pela função FDBMESS (), para que esta informe o erro ocorrido.

3.16.14.4 Exemplos

O exemplo a seguir mostra como utilizar esta função.

```

PROG
ret = FOPEN ("bdemp1", "a", 1, 2)
IF ret # 0
MESSAGE "Erro " + str(ret,2) + " na abertura do banco"
QUIT
ENDIF
MESSAGE "Banco aberto" bold
WAIT "Fecha o banco ? s/n " TO k
IF UPPER (LEFT (k, 1)) = "S"
ret = FCLOSE ( )
IF ret # 0
MESSAGE "Nao foi possivel fechar o banco" BLINK
ENDIF
ENDIF

```

3.16.15 FGACUM ()

3.16.15.1 Propósito

Acumular somatório de itens numéricos de uma cadeia.

3.16.15.2 Sintaxe

```
FGACUM (<expC1>, <expC2>, <expC3>, <vetC>, <vetN>).
```

3.16.15.3 Argumentos

- <expC1> Representa o nome de um arquivo do banco de dados.
- <expC2> Representa o nome de uma chave que determina a cadeia.
- <expC3> Representa o valor da cadeia.
- <vetC> Representa um vetor caractere com nome dos itens que serão acumulados.
- <vetN> Representa um vetor numérico onde será armazenados os resultados das totalizações.

3.16.15.4 Utilização

Esta função lê os registros cujo valor da chave <expC2> forem iguais a expressões <expC2> acumulando em <vetN> os valores dos itens definidos em <vetC>. Os itens definidos em <vetC> devem ser de tipo numérico (tipos N, P, S, C, B, I ou F).

Para cada elemento de do vetor <vetC> corresponde um elemento do vetor <vetN> com o resultado da acumula F/ D/o do item correspondente.

Esta função quando bem sucedida retorna 0 (zero). Caso contrário retorna -1, neste caso verificar se algum dos itens passados em <vetC> não é numérico, a quantidade de elemento de <vetN> corresponde a quantidade de elementos de <vetC> ou se ocorreu erro durante o acesso ao banco de dados (ver se DBERR() # 0).

3.16.16 FGPACUM ()

3.16.16.1 Propósito

Acumular somatório de itens numéricos de uma cadeia de prefixo.

3.16.16.2 Sintaxe

FGPACUM (<expC1>, <expC2>, <expC3>, <vetC>, <vetN>)

3.16.16.3 Argumentos

- <expC1> Representa o nome de um arquivo do banco de dados.
- <expC2> Representa o nome de uma chave que determina a cadeia.
- <expC3> Representa o valor para pesquisa de <exp.C2>.
- <vetC> Representa um vetor caractere com nome dos itens que serão acumulados.
- <vetN> Representa um vetor numérico onde serão armazenados os resultados das totalizações.

3.16.16.4 Utilização

Esta função lê os registros cujo valor do prefixo da chave <expC2> forem iguais a expressão <expC2> acumulando em <vetN> os valores dos itens definidos em <vetC>. Os itens definidos em <vetC> devem ser de tipo numérico (tipos N, P, S, C, B, I ou F).

Para cada elemento de do vetor <vetC> corresponde um elemento do vetor <vetN> com o resultado da acumulação do item correspondente.

Esta função quando bem sucedida retorna 0 (zero). Caso contrário retorna -1, neste caso verificar se algum dos itens passados em <vetC> não é numérico, a quantidade de elemento de <vetN> corresponde a quantidade de elementos de <vetC> ou se ocorreu erro durante o acesso ao banco de dados (ver se DBERR() # 0).

3.16.17 FILE ()

3.16.17.1 Propósito

Serve para verificar a existência de um arquivo externo.

3.16.17.2 Sintaxe

FILE(<expC>)

3.16.17.3 Argumentos

- <expC> representa o nome do arquivo.

3.16.17.4 Utilização

Retorna verdadeiro (.T.) se o arquivo for encontrado, falso em caso contrário.

3.16.17.5 Exemplos

? file ("config.sys") && retorna .T. se tiver o arquivo

3.16.18 FILETOMEMO()

3.16.18.1 Propósito

Lê de um arquivo em Disco e grava em um campo memo.

3.16.18.2 *Sintaxe*

FILETOMEMO(Campo Memo, Nome do Arquivo).

3.16.18.3 *Utilização*

É utilizado como forma de atualização do campo memo, sem o limite de 64k, associa o campo ao ultimo registro lido.

3.16.18.4 *Vantagens:*

Não precisa passar pela memória para ler ou gravar, ele lê direto.

3.16.18.5 *Exemplo :*

```
prog
database memopol 1 a 2
use unico
find 001
a= filetomemo(desc,"arqt")
...
```

3.16.18.6 *Consulte*

MEMOTOFILE().

3.16.19 FILETOSCREEN()

3.16.19.1 *Propósito*

Restaurar uma tela antes salva pela função SCREENTOFILE().

3.16.19.2 *Sintaxe*

FILETOSCREEN (<expC>)

3.16.19.3 *Argumentos*

<expC> Representa o nome do arquivo de onde será lida a tela.

3.16.19.4 *Utilização*

Exibe na tela do terminal o conteúdo do arquivo especificado pela expressão caractere, este arquivo obrigatoriamente deve conter uma tela anteriormente salva pela função **SCREENTOFILE()**. Retorna falso (.F.) caso não seja possível ler o arquivo especificado. Caso contrário retornará verdade (.T.). A utilização desta função implica na utilização da opção \$SCREEN ou \$SSCREEN.

3.16.19.5 *Exemplos*

O exemplo a seguir demonstra como utilizar esta função.

```
$SCREEN
prog
if .not. FILETOSCREEN ("/usr/telas/telprog1")
  beep
  message "Nao foi possivel restaurar tela"
  sleep 5
  quit
endif
clear
```

3.16.19.6 *Consulte*

SCREENTOFILE().

3.16.20 FJUDI

FJUDI (<banco>, <nivel>, <seguranca>, <modos>)

Indica que outro banco será aberto e seu dicionário será incorporado ao (s) dicionário do (s) banco (s) já aberto (s).

3.16.20.1 *Consulte*

DATAJOIN.

3.16.21 FKEY()

3.16.21.1 Propósito

Retornar características de item chave.

3.16.21.2 Sintaxe

FKEY (<expC1>, <expC2>[, <vetC>])

3.16.21.3 Argumentos

<expC1> Representa o nome do arquivo do banco de dados.
<expC2> Representa o nome de um item.
<vetC> Representa um vetor caractere.

3.16.21.4 Utilização

Retorna um valor numérico correspondente ao tipo do item passado como argumento. Caso seja declarado <vet.C> como argumento, este vetor é preenchido com informações adicionais sobre a chave <exp.C2>. O valor retornado corresponde a soma, ou a um, dos valores descritos a seguir:

1	Para chave primária,
2	Para chave estrangeira
3	Para chave de busca
10	Para chave virtual
100	Para chave única,
1000	Para chave nula
10000	Exclui em cascata,
20000	Valor nulo em cascata na exclusão
100000	Altera em cascata
200000	Valor nulo em cascata na alteração

O vetor <vet.C> é preenchido com informações adicionais. No caso de chave estrangeira, somente o primeiro elemento do vetor é utilizado, sendo preenchido com o nome do arquivo entidade associado a chave.

No caso de chave virtual, os elementos do vetor serão preenchidos com o nome dos itens que compõem a chave virtual. No caso de chave virtual estrangeira, o primeiro elemento é preenchido com o nome da entidade associada e os demais são preenchidos com o nome dos itens que compõem a chave virtual.

Esta função retorna -1 (menos um) quando:

<exp.C1> não for arquivo do banco de dados,
<exp.C2> não for chave no arquivo <exp.C1>,
ou qualquer outra incoerência.

Caso FKEY retorne -1, o vetor <vet.C> retornará com seus elementos preenchidos de brancos.

3.16.21.5 Exemplos

O exemplo a seguir mostra como utilizar esta função.

```
PROG
DECLARE vet_cad[3] = space (12)
ret = FOPEN ("bdemp1", "a", 33, 2)
IF ret # 0
  MESSAGE "Erro na abertura do banco" blink
  QUIT (2)
ENDIF
* Exibira 102 (estrangeira e unica)
? FKEY ("tab_vend", "id1_func", vet_cad)
* Exibira tab_func
? vet_cad[1]
?* Exibira 1 (chave primaria)
```

```
? FKEY ("tab_func", "id0_func", vet_cad)
* Exibira 000000003
? vet_cad[1]
```

3.16.22 FLENGTH ()

3.16.22.1 Propósito

Retornar o tamanho de um item do banco de dados.

3.16.22.2 Sintaxe

```
FLENGTH (<expC1>, <expC2>)
```

3.16.22.3 Argumentos

<expC1> Representa o nome de um arquivo do banco de dados em uso.

<expC2> Representa o nome de um item do arquivo.

3.16.22.4 Utilização

Retorna um valor numérico correspondente ao tamanho do item <expC2> no arquivo <exp.C1> ou o valor numérico -1 (menos um), quando:

<expC2> não for um item de <expC1>,

<expC1> não for um arquivo do banco de dados em uso.

O valor retornado corresponde ao total em bytes ocupado pelo item no disco.

3.16.22.5 Exemplos

O exemplo a seguir mostra como utilizar esta função levando em consideração o arquivo tab_nfis.

```
* nome: tab_nfis L
* registro: id0_nfis (1) n03
* cli_nfis (tab_clie) n03
* ven_nfis (tab_func) n03
* tot_nfis p10,2
* dat_nfis d02

PROG
ret = FOPEN ("bdemp1", "a", 33, 2)
IF ret # 0
? "Erro na abertura do banco bdemp1"
? FDBMESS (ret)
ENDIF
? "Tamanho do item tot_nfis = "
?? FLENGTH ("tab_nfis", "tot_nfis") && Exibira 5
?
? "Tamanho do item id0_nfis = "
?? FLENGTH ("tab_nfis", "id0_nfis") && Exibira 3
?
? "Tamanho do item inexiste = "
?? FLENGTH ("tab_nfis", "inexiste") && Exibira -1
```

3.16.23 FOFFSET ()

3.16.23.1 Propósito

Retornar o deslocamento de um item dentro do registro.

3.16.23.2 Sintaxe

```
FOFFSET (<exp.C1>, <exp.C2>)
```

3.16.23.3 Argumentos

<expC1> Representa o nome de um arquivo do banco de dados em uso.

<expC2> Representa o nome de um item do arquivo.

3.16.23.4 Utilização

Retorna um valor numérico correspondente ao deslocamento do item <exp.C2> dentro do registro no arquivo <exp.C1> ou o valor -1 (menos um), quando:

<exp.C2> não for um item de <exp.C1>,

<exp.C1> não for um arquivo do banco de dados em uso.

O valor retornado corresponde ao total em bytes deslocados a partir do primeiro item do registro. Em registro inicia na posição zero (0).

3.16.23.5 Exemplos

O exemplo a seguir mostra como utilizar esta função levando em consideração o arquivo **tab_nfis**.

```
* nome: tab_nfis L
* registro: id0_nfis (1)    n03
*   cli_nfis (tab_clie) n03
*   ven_nfis (tab_func) n03
*   tot_nfis      p10,2
*   dat_nfis      d02

PROG
ret = FOPEN ("bdemp1", "a", 33, 2)
IF ret # 0
  ? "Erro na abertura do banco bdemp1"
  ? FDBMESS (ret)
ENDIF
? FOFSET ("tab_nfis", "tot_nfis") && Exibira 9
? FOFSET ("tab_nfis", "id0_nfis") && Exibira 0
? FOFSET ("tab_nfis", "inexiste") && Exibira -1
```

3.16.23.6 Consulte

FOPEN()

3.16.24 FONETV

3.16.24.1 Propósito

Devolve o número de códigos fonéticos encontrados na cadeia <nome> e guardados no vetor cadeia <vetor> que deve ter um número de elementos suficientes para contê-los e tamanho de elementos >= 8.

3.16.24.2 Sintaxe

Fonetv (<nome>,<vetor>)

3.16.24.3 Exemplos

```
declare fons (10) = space (8)
?=fonetv ("JOSE DA SILVA", tons)
for j=1 ton
  ? fons [j]
next
```

3.16.25 FOPEN()

3.16.25.1 Propósito

Abrir um banco de dados *OpenBASE*.

3.16.25.2 Sintaxe

FOPEN (<exp.C1>, <exp.C2>, <exp.N1>, <exp.N2>)

3.16.25.3 Argumentos

<exp.C1>	Representa o nome do banco de dados.
<exp.C2>	Representa o nível de acesso às informações dos itens do banco de dados.

<expN1> Representa o código de segurança definido para o banco de dados.
 expN2> Representa o modo de abertura do banco de dados.

3.16.25.4 Utilização

Abre o banco de dados definido por <exp.C1>, retornando um valor numérico correspondente ao código de erro desta operação (DBERR).

Quando o banco de dados é aberto com sucesso o valor retornado é igual a zero. Caso seja diferente de 0, este valor pode ser utilizado pela função FDBMESS(), para que esta informe o erro corrido.

Permite criar programas independentes do banco de dados, pois só verifica a existência do banco durante a execução do programa. Deste modo, as informações para a abertura do banco de dados podem ser passadas como argumentos para o programa.

3.16.25.5 Exemplos

O exemplo a seguir mostra como utilizar esta função.

```

PROG
ACCEPT "Informe o nome do banco ..... " TO nom_ban
ACCEPT "Informe a palavra de nivel ..... " TO pal_niv
INPUT "Informe o código de segurança ..... " TO cod_seg
INPUT "Informe o modo de abertura ..... " TO mod_abe
cod_err = FOPEN (nom_ban, pal_niv, cod_seg, mod_abe)
?
IF cod_err # 0
? "Erro na abertura do banco de dados"
? FDBMESS (cod_err)
ELSE
? "Banco aberto"
ENDIF
  
```

3.16.25.6 Consulte

DATABASE.

3.16.26 FSEEK, FSTART, FFIND

3.16.26.1 Propósito

Nas funções descritas acima, o parâmetro referente ao valor do campo pesquisado, pode se referir a um item virtual.

3.16.26.2 Sintaxe

```
Ret = FSEEK ("Nome_arq","Nome_item","virtual (Valor1, Valor2, ... , Valorn),
Vet1, Vet2)
```

3.16.26.3 Argumentos

Nome_arq	Nome do arquivo onde será feita a pesquisa.
Nome_item	Nome do item virtual onde será feita a pesquisa
Valor1, valor2, ..., valorn	Valor do item virtual que se deseja pesquisar.
Vet1	Vetor cadeia com o nome dos itens que serão lidos
Vet2	Vetor cadeia que receberá o valor dos itens lidos

3.16.26.4 Utilização

Útil para se pesquisar valores de itens virtuais, utilizando-se as funções de pesquisa FSEEK, FSTART e FFIND.

3.16.27 FTPSEND – FTPRECEIVE

3.16.27.1 Propósito

Permitir o envio ou recebimento de arquivos, via ftp, a partir de um programa Opus.

3.16.27.2 Sintaxe

ftpsend (<ser>, <log>, <pas><ori>, <des>)

ftpreceive (<ser>, <log>, <pas>, <ori><des>)

3.16.27.3 Argumentos

<ser>	nome do servidor
<log>	login no servidor
<pas>	senha no servidor
<ori>	arquivo origem
<des>	diretório/arquivo destino

3.16.27.4 Utilização

Útil em programas Opus, quando se deseja enviar ou receber arquivos de um servidor remoto, sem ter que se utilizar algum arquivo batch ou shell.

Caso o cliente seja Windows 9x, NT, 2000, deve estar instalada a biblioteca WININET.LIB

3.16.27.5 Exemplos

```
if ftpsend ("10.1.1.26", "root", "senha", "\tmp\teste", "/usr/tmp")
? "Arquivo foi enviado com sucesso"
endif
```

3.16.28 GetDiskUsage()

3.16.28.1 Propósito

Retorna a percentagem do espaço ocupado do disco solicitado.

3.16.28.2 Sintaxe

GetDiskUsage (<device>).

3.16.28.3 Argumentos

<device> Indica o nome da participação do disco rígido, onde se verificará a percentagem ocupada.

3.16.28.4 Utilização

Esta função é útil para avisar ao usuário, a percentagem ocupada, de uma determinada partição do disco rígido, evitando que o mesmo fique cheio.

3.16.28.5 Exemplos

```
prog
database dbbanco 1 a 2
if GetDiskUsage ("/dev/hd2")>95
valor = GetDiskUsage ("/dev/hd2")
@10,10 say "Atenção seu disco rígido está cheio" + str (valor) blink
wait
endif
```

3.16.29 GETCLIENT()

3.16.29.1 Propósito

Esta função cadeia obtém o nome do cliente, armazenado no Registry do Windows e/ou no arquivo "FacPrint".

3.16.29.2 Exemplos

```
? GetClient()
```

3.16.30 GETDIRECTORY()

3.16.30.1 Propósito

Esta função cadeia obtém o nome do diretório (ou pasta) default onde se encontram localizados os Bancos de dados *OpenBASE*. Esta informação é obtida a partir do Registry do Windows (ou do velho arquivo "FacPrint").

Por exemplo, o comando `?GetDirectory()` poderá retornar a cadeia `D:\USR\TSGBD\TSDIC` ou qualquer outro percurso que tenha sido estabelecido como default pelo usuário através do programa de configuração WinCNFG.

3.16.31 GETVERSION()

Esta função cadeia obtém a versão do *OpenBASE*, armazenado no Registry do Windows e/ou no arquivo "FacPrint". Por exemplo, o comando `?GetVersion()` poderá retornar: **V8.1**.

3.16.32 INDRECCOUNT()

3.16.32.1 Propósito

Retorna o número de registros de um arquivo de índice.

3.16.32.2 Sintaxe

```
m=indreccount("<nome arquivo índice>").
```

3.16.32.3 Argumentos

<Nome arquivo índice> Arquivo de índice do qual se deseja obter o número de registros.

3.16.32.4 Exemplos

```
select a
use PESSOA
? indreccount ("NomeP1)
```

3.16.33 MAX()

3.16.33.1 Propósito

Comparar expressões numéricas e retornar a de maior valor.

3.16.33.2 Sintaxe

```
MAX (<exp.N1>,<exp.N2>)
```

3.16.33.3 Argumentos

<expN1> e <expN2> Representam as expressões numéricas a serem comparadas.

3.16.33.4 Utilização

Compara o resultado das expressões numéricas 1 e 2 retornando o valor numérico da que for maior.

3.16.33.5 Exemplos

O exemplo a seguir exemplifica a utilização desta função.

```
prog
var_num = -78
? MAX (var_num, var_num * -1)    && Exibira 78
```

3.16.33.6 Consulte

MIN()

3.16.34 MEMOTOFILE()

3.16.34.1 Propósito

Lê de um campo memo e grava em um arquivo em Disco.

3.16.34.2 Sintaxe

```
MEMOTOFILE( Campo Memo, Nome do Arquivo).
```

3.16.34.3 Utilização

É utilizado como forma de atualização do campo memo, sem o limite de 64k, associa o campo ao ultimo registro lido.

3.16.34.4 Vantagens

Não precisa passar pela memória para ler ou gravar, ele lê direto.

3.16.34.5 Exemplos

```
prog
database memopol 1 a 2
use unico
find 001
x= memotofile(desc,"arqc")
.
```

3.16.34.6 Consulte

FILETOMEMO()

3.16.35 MIN()

3.16.35.1 Propósito

Comparar expressões numérica e retornar a de menor valor.

3.16.35.2 Sintaxe

MIN (<exp.N1>,<exp.N2>)

3.16.35.3 Argumentos

<expN1> e <expN2> Representam as expressões numéricas a serem comparadas.

3.16.35.4 Utilização

Compara o resultado das expressões numéricas 1 e 2 retornando o valor numérico da que for menor.

3.16.35.5 Exemplos

O exemplo a seguir exemplifica a utilização desta função.

```
prog
var_num = -78
? MIN (var_num, var_num * -1) && Exibira -78
```

3.16.35.6 Consulte

MAX()

3.16.36 OemToChar

3.16.36.1 Propósito

Converter uma cadeia, no formato ASCII para o Windows.

3.16.36.2 Sintaxe

OemToChar ("texto")

3.16.36.3 Argumentos

"texto" Representa a expressão cadeia no formato ASCII a ser convertida para o formato Windows.

3.16.36.4 Utilização

Esta função é útil quando se deseja exibir vogais acentuadas e o cedilha de um texto escrito no formato ASCII (console application), em um aplicativo Windows.

3.16.36.5 Exemplos

O programa abaixo foi escrito no edit.com do DOS, e compilado na OpusWin:

```
prog
bt1=1
```

```

var= OemToChar ("Atenção")
Dialog 0,0,79,23
  vartext Null var 02,02,10,01
  defpushbutton "sair"
  bt1 02,10,10,01 IDOK
enddialog

```

3.16.37 PUT_ENV

3.16.37.1 Propósito

Carregar uma variável de ambiente.

3.16.37.2 Sintaxe

```
Putenv ("<exp.c>=<conteúdo>")
```

3.16.37.3 Argumentos

<expc> Representa o nome da variável de ambiente.
 <conteúdo> Cadeia a ser enviada para variável de ambiente.

3.16.37.4 Utilização

Atribui um conteúdo a uma variável de ambiente. Este conteúdo é uma cadeia de caracteres.

3.16.37.5 Exemplo

```
Putenv ("ABC=def")
? Getenv ("ABC") && retorna "def"
```

3.16.38 RESTORES KEYS [<area>]

Restaura os endereços das rotinas especificadas nos comandos SET KEY <num> to <proc>. Salvos na <area> especificada.

3.16.38.1 Exemplos

```

$noLib
prog
private a (100)
set key 28 to p1
wait "1" to x
save keys
set key 28 to p2
wait "2" to x
restore keys
wait "3" to x
proc p1
? "p1"
return
proc p2
? "p2"
return

```

3.16.39 SAVE KEYS [<area>]

Salva os endereços das rotinas especificadas nos comandos SET KEY <num> TO <proc> na <area> especificada que deve Ter no mínimo 100 bytes. Caso não seja especificada uma área é criada automaticamente.

3.16.40 SCREENTOFIL ()

3.16.40.1 Propósito

Gravar conteúdo de tela em um arquivo.

3.16.40.2 *Sintaxe*

SCREENTOFILE (<expC>)

3.16.40.3 *Argumentos*

<expC> Representa o nome do arquivo para onde será copiado o conteúdo da tela.

3.16.40.4 *Utilização*

Copia o conteúdo da tela do terminal para o arquivo indicado pelo resultado da expressão caractere. Retorna falso (.F.) caso não seja possível gravar o arquivo especificado. Caso contrário retornará verdade (.T.). Para se utilizar o conteúdo dos arquivos criados por esta função usar a função FILETOSCREEN (.). A utilização desta função implica na utilização da opção \$SCREEN ou \$SSCREEN.

3.16.40.5 *Exemplos*

O exemplo a seguir demonstra como utilizar esta função.

```
$SCREEN
prog
for linha = 5 to 15
  @ linha, 20 say rep("***",40)
next
if .not. SCREENTOFILE ("/usr/telas/telprog1")
  beep
  message "Nao foi possivel salvar tela"
  sleep 5
  quit
endif
clear
quit
```

3.16.40.6 *Consulte*

\$NOSCREEN, ?, ??, @ ... SAY, SET DEVICE e SET PRINTER, FILETOSCREEN.

3.16.41 **SELECTED()**

3.16.41.1 *Propósito*

Retornar a quantidade de registros selecionados pelo comando QUERY.

3.16.41.2 *Sintaxe*

SELECTED()

3.16.41.3 *Utilização*

Retorna um valor numérico correspondente a quantidade de registros que atenderam a pesquisa feita pelo comando QUERY.

Pode ser utilizada como alternativa na verificação do sucesso da leitura, em substituição a função EOF(), por que o comando QUERY não acusa fim de leitura. Sendo assim só seria possível verificar o fim de leitura após a execução do comando LOCATE.

3.16.41.4 *Exemplos*

O exemplo a seguir mostra como utilizar esta função para verificar o fim da leitura.

```
PROG
DATABASE bdemp1 1 a 3
USE tab_nfis
QUERY id0_clie <> 543 .or. id0_clie <> 647
IF SELECTED () = 0
  ? "Nenhum registro selecionado"
  QUIT (1)
ENDIF
? "Foram selecionados ", SELECTED (), " registros"
LOCATE
```

```

DO WHILE FOUND()
? "Codigo Nf ", id0_nfis
? "Cliente  ", nom_clie
? "Vendedor ", nom_func
CONTINUE
ENDDO

```

3.16.41.5 *Consulte*

QUERY.

3.16.42 **SERVER ()**

3.16.42.1 *Propósito*

Permite que um programa Opus seja um servidor de aplicativo.

3.16.42.2 *Sintaxe*

```

server <PORTA>
UNIX:          nohup <programa> -t <timeout> -n<porta> &
Windows:      <programa> -s <executavel> -n<porta>

```

3.16.42.3 *Argumentos*

<porta> indica o serviço TCPIP disponível, para atender os pedidos de mensagens enviadas ao servidor. (ver arquivo services)

<timeout> indica o tempo em segundos que o programa servidor pode permanecer no ar sem comunicação.

<executável> indica o percurso na máquina servidor onde se encontra o programa executável.

3.16.42.4 *Utilização*

Carrega um programa Opus, deixando-o no ar, até que seja executada a função server (0). Permite ao usuário determinar o número de segundos (time-out) que o programa servidor permanece no ar sem receber comunicação do cliente, assim como o número do serviço TCPIP disponível para a comunicação (porta). Utilizado no protocolo de comunicação send/receive, onde o cliente (programa Opus) envia e recebe mensagens do servidor (programa Opus).

3.16.43 **ServerEecute**

3.16.43.1 *Propósito*

Permite a execução de um comando em uma máquina remota (servidor), a partir de um programa Opus.

3.16.43.2 *Sintaxe*

```

ServerExecute("[Percurso] <comando>")

```

3.16.43.3 *Utilização*

Útil quando se deseja executar um comando em uma determinada máquina, a partir de um programa Opus que está sendo executado em outra máquina.

3.16.43.4 *Exemplos*

```

$client = "10.1.1.26"
prog
if serverexecute ( "/usr/tmp/sh backup.l") = 0
? "Backup realizado"
endif

```

3.16.44 **TTYIP**

3.16.44.1 *Propósito*

A função cadeia TTYIP () retorna o endereço IP da estação de trabalho (cliente) que se conecta ao servidor UNIX, via emulador de terminal TELNET.

3.16.44.2 *Sintaxe*

Var= ttyip ()

3.16.44.3 *Argumentos*

Esta função é útil, quando se deseja conhecer qual a máquina cliente que esta conectada a um servidor UNIX. É necessário que a máquina cliente esteja configurada na rede local.

3.16.44.4 *Exemplos*

endereço = ttyip()

? endereço

Seria exibido o endereço IP da máquina conectada ao servidor UNIX.

3.16.45 TTYNAME ()

3.16.45.1 *Propósito*

Retornar o nome da máquina do cliente conectado por um emulador de terminal a uma máquina UNIX.

3.16.45.2 *Sintaxe*

m = ttyname ()

3.16.45.3 *Utilização*

Útil quando se deseja saber o nome da máquina que está ligada a um servidor UNIX, através de um emulador de terminal. Equivale ao "computer name" obtido no comando "**Who am I**" do UNIX.

4 Desenvolvimento de aplicativos - RAD

O *OpenBASE* disponibiliza as funções e rotinas de manipulação de Bancos de Dados para serem utilizadas pelas linguagens Visuais Basic, Delphi ou outras ferramentas RAD, podendo, assim, ser elaborados aplicativos para acessar Bases de Dados *OpenBASE* utilizando variadas ferramentas de desenvolvimento. O acesso aos Bancos de Dados *OpenBASE* pode ser feito através de bibliotecas dinâmicas (DLLs do *OpenBASE*), através dos recursos ODBC (Driver ODBC do *OpenBASE*) ou utilizando interfaces COM e DCOM para acessar objetos e métodos *OpenBASE*. Os componentes *OpenBASE* podem ser disponibilizados para aplicativos desenvolvidos em diversos ambientes de programação, por exemplo:

- Visual Basic (VB)
- Visual Basic for Applications (VBA)
- Delphi
- PowerBuilder
- Windows Scripting Host (WSH)
- Active Server Pages (ASP)
- PHP: Hypertext Preprocessor (PHP)
- JAVA
- TCL/TK
- Perl
- Python

As ferramentas RAD (Rapid Application Development) possuem, geralmente, as seguintes características:

- São linguagens de quarta geração (4GLs)
- Utilizam modernas interfaces gráficas (GUIs)
- Estão integradas com arquiteturas OOP (Object-Oriented Programming)
- Implementam funções de acesso a Bancos de Dados
- Suportam, em plataformas Windows, acesso a Bancos de Dados utilizando ODBC, padrão Microsoft para Open DataBase Connectivity
- Invocam funções exportadas a partir de DLLs (Dynamic Link Libraries)
- Usam objetos padrão COM (Common Object Model), da Microsoft

As funções e objetos *OpenBASE* estão agrupados em duas bibliotecas de ligação dinâmica (DLLs), conforme a seguir:

ROTWIN32.DL Biblioteca principal com as rotinas básicas de acesso ao Banco de Dados *OpenBASE*.
CLIWIN32.DLL Similar a biblioteca ROTWIN32.DLL, porém as rotinas são voltadas para aplicações cliente dentro de uma arquitetura CLIENTE/ SERVIDOR em redes TCP-IP.

O módulo CLIWIN32.DLL inclui as funções para arquitetura cliente/servidor e permite desenvolver aplicativos cliente cujos servidores podem residir em qualquer outra plataforma, local ou remota, conforme veremos nos exemplos apresentados. As funções (ou rotinas) e métodos incluídos nessas bibliotecas dinâmicas permitem acessar e controlar Bancos de Dados *OpenBASE*. Apresentamos a seguir as novidades do *OpenBASE* nesta área.

4.1 ExecutaPrograma

O comando ExecutaPrograma serve para executar um determinado programa em uma máquina remota (servidor) a partir de um programa que esta sendo executado numa máquina local (cliente).

4.1.1 Sintaxe (VB e Delphi)

```
Declare function ExecutaPrograma ("cliwin32.dll")(byval c as string) as long  
Function ExecutaPrograma ( Comando: Pointer): integer; stdcall; external cliwin32.dll;
```

4.1.2 Exemplos

```
ret = ExecutaPrograma ("/usr/tmp/p.exe");  
if ret = 0  
then msg "EXECUCAO OK"  
endif
```

4.2 ObtemRegistrosComPrefixo

4.2.1 Propósito

Retorna o número de registros selecionados pela rotina **IniciaPorPrefixo**.

4.2.2 Sintaxe

```
IniciaporPrefixo (Arquivo, Item, CadeiaCaracter)  
Obtem RegistrosComPrefixo (Arquivo)
```

4.2.3 Argumentos

Arquivo	Arquivo do Banco de Dados aberto.
Item	Campo do arquivo em cima do qual será feita a pesquisa.
Cadeia Caracter	Texto que será pesquisado.

4.2.4 Exemplos

```
IniciaPorPrefixo ("pessoa", "nome", "Ana")  
ObtemRegistrosComPrefixo ("pessoa")
```

4.3 Defcom32.dll

Permite a execução do define chamado do VB, Delphi ou Opus. Essa DLL possui apenas uma função, chamada **defcom**, que recebe como parâmetro uma string na forma: "**define -d<arqerr><esquema>**"

4.3.1 Exemplo em VB:

```
Declare function defcom lib "defcom32.dll"(byval a as string) as integer  
Ret = defcom ("define -desq.err esq.e").
```

4.3.2 Exemplo em Opus

```
Rundll ("defcom32.dll","defcom", "define -dp.err p.f")
```

4.4 *Opucom32.dll*

Permite a execução da Opus chamada do VB, Delphi ou Opus. Esta DLL possui apenas uma função, chamada **opucom**, que recebe como parâmetro uma string na forma: "**opus -a<arqerr><fonte>**"

4.4.1 Exemplo em VB

```
Declare function opucom lib "opucom32.dll" (byval a as string) as integer
Ret = opucom ("opus -dp.err p.f")
```

4.4.2 Exemplo em Opus

```
Rundll ("opucom32.dll", "opucom", "opus -dp.err p.f")
```

4.5 *LigaOpção e Desliga Opção*

Nas funções LigaOpção/Desliga Opção para se ligar/desligar a opção de abrir todos os arquivos de um banco foi incluída a opção "open".

4.5.1 Sintaxe

```
DesligaOpcao(char *opc)
```

4.5.2 Exemplos

```
ret=DesligaOpção("open")
```

Para se ligar/desligar a opção de século foi incluída a opção "century".

```
ret=LigaOpção("century")
```

Deve ser utilizada após o uso das rotinas que requerem a rotina **LigaOpcao**.

```
Declare Function LeProximoRegistroSubcadeia Lib "cliwin32.dll" (ByVal IpArquivo as String, ByVal IpItem as String, ByVal IpSub as String, ByVal IpBuffer as awy) as long.
```

4.6 *LeProximoSequencial*

4.6.1 Propósito

Lê o próximo registro em um arquivo do banco de dados. Os itens a serem lidos, podem ser especificados com uma string iniciada com %% e seguida dos nomes dos itens separados por vírgulas.

4.6.2 Sintaxe

```
Ret = LeProximoSequencial ("Arquivo", "%%item1, itemn", buffer).
```

4.6.3 Argumentos

Arquivo Nome do arquivo do banco de dados, que será lido.

Item1-n Nome dos itens que serão lidos.

Área Área para onde serão lidos os itens listados no parâmetro anterior. Deve ter pelo menos o tamanho igual a soma dos tamanhos dos itens listados no parâmetro anterior.

4.6.4 Utilização

Possibilitar a leitura sequencial dos itens de um arquivo ou dos itens de outro arquivo relacionado ao arquivo selecionado, através de chaves primária e estrangeira.

4.6.5 Exemplos

```
nome: arq1 E
  codigo(1)      U01
  texto          U20
nome: arq2 R
  codigo2 (arq1)U01
LeProximoSequencial ("arq2", "%% codigo2, texto", buffer).
```

4.7 *Alarme Servidor (<Tempo>)*

4.7.1 **Propósito**

Determina o tempo em segundo, que o bdserv inicializado por uma aplicação cliente, deva permanecer ativo, sem que haja comunicação da aplicação cliente.

4.7.2 **Sintaxe**

`AlarmeServidor (<tempo>)`

4.7.3 **Argumentos**

`<tempo>` indica o tempo em segundos, que o bdserv filho, inicializado pela aplicação cliente, permanecerá ativo, sem que haja, comunicação da aplicação cliente.

4.7.4 **Utilização**

Evitar que um processo filho bdserv, permaneça ativo infinitamente, caso a aplicação cliente seja finalizada de forma anormal, sem finalizar o bdserv (`FinalizaServidor`).

4.7.5 **Exemplos**

`Alarme Servidor (3.600)` após 3.600 segundos sem comunicação o bdserv sairá do ar.

`Alarme Servidor (-1)` o bdserv não sai do ar

4.8 *Rotinas contidas na CLIWIN32.DLL*

4.8.1 **PegaData e PegaHora**

4.8.1.1 **Propósito**

Estas rotinas contidas na `CLIWIN32.DLL` tem a função de retornar a data e hora da máquina servidor, onde foi feita a conexão.

4.8.1.2 **Sintaxe (VB):**

```
declare function PegaData lib ("cliwin32.dll") (byval cad as string) as long  
declare function PegaHora lib ("cliwin32.dll") (byval cad as string) as long
```

4.8.1.3 **Exemplo (VB):**

```
dim cad as string*12  
PegaData (cad)  
MsgBox cad
```

4.8.2 **TestaServidor**

4.8.2.1 **Propósito**

Estabelece, de tempos em tempos, a comunicação com o servidor (bdserv) para que o mesmo não saia do ar por timeout.

4.8.2.2 **Sintaxe (VB):**

```
Declare function TestaServidor ("cliwin32.dll") () as long.
```

4.8.2.3 **Exemplos**

```
ret=TestaServidor()
```

5 *Utilitários OpenBASE*

5.1.1 **BDADIC**

5.1.1.1 **Argumentos**

<code>-g</code>	Indica que os registros com tipo T8, serão gravados com a data e hora do sistema operacional.
-----------------	---

5.1.2 BDLIDI

5.1.2.1 Sintaxe

bdlidi [-S] -b<banco> -s<seg> -n<nível> {-d<data-hora> | -t<numero>} [-a<diario>] [-y]

5.1.2.2 Argumentos

O utilitário **bdlidi**, aceita as opções **-n<nível>** e **-s<seg>**, na abertura de banco de dados, onde:
<nível> é a palavra de nível do usuário, especificado no esquema. Se omitido, será considerado o valor "a".
<seg> é o código de segurança do banco, especificado no esquema. Se omitido, será assumido o valor 1.

5.1.3 BDRED

5.1.3.1 Sintaxe

bdredi -b<banco> -s <seg> -n <nível> [-r] [-a<diário>] [-y<numero> -f<arquivo>]
{-d<data-hora> | -t<numero1>} [-e] [-O]

5.1.3.2 Argumentos

O utilitário **bdredi**, aceita as opções **-n<nível>** e **-s<seg>**, na abertura de banco de dados, onde:
<nível> é a palavra de nível do usuário, especificado no esquema. Se omitido, será considerado o valor "a".
<seg> é o código de segurança do banco, especificado no esquema. Se omitido, será assumido o valor 1.

5.1.4 BDSGBD

5.1.4.1 Propósito

Novos parâmetros foram adicionados na inicialização do programa **bdsghd**, como a definição do tamanho da memória compartilhada, e inibição da mensagem que o **bdsghd** já está carregado. O **bdserv**, pode ter sua localização (percurso), informada assim como o tamanho máximo da mensagem recebida.

5.1.4.2 Argumentos

O **bdsghd** pode ter os parâmetros:

-d	Indica que será gravado o diário.
-a<diario>	Onde diario é a localização do diário. Por exemplo: c:\usr1\diario Valor default: c:\usr\tsgbd\tsdic\diario
-e	Indica para esvaziar o diário
-m<numero>	Indica o tamanho da memória compartilhada
-s	Indica no Windows que a mensagem de erro " OpenBASE Server already running" não deve ser emitida

5.1.5 Carregar BDSGBD em AIX 4.x

- Não incluir chamada ao BDSGBD em /etc/rc
- Criar arquivo /etc/rc.local com a seguinte instrução: **nohup bdsghd &**
- Copiar arquivo /etc/inittab para arquivo com outro nome
- Executar comando: **mkitab rclocal:2:once:/etc/rc. Local**

5.1.5.1 Utilização

Para que o **OpenBASE** seja acessado no Windows 9X / NT / 2000, o programa **bdsghd.exe** deve estar inicializado. O arquivo **bdserv.exe** deve ser inicializado caso o Windows trabalhe como servidor de banco de dados.

Os programas **bdsghd** e **bdserv** devem ser iniciados como serviços no Windows.

No Windows 9X deve ser incluída no registry a chave:

Hkey_LM\Software\Microsoft\Windows\Current\Version\RunServices

O **nome** do valor será: **bdsghd** e o **valor** será: **c:\usr\tsgbd\bdsghd.exe**

O **nome** do valor será: **bdserv** e o **valor** será: **c:\usr\tsgbd\bdserv.exe**

5.1.6 BDTSQL

5.1.6.1 Propósito

Criar um esquema SQL, a partir do dicionário de dados de um banco *OpenBASE*.

5.1.6.2 Sintaxe

```
bdtsql -b <banco> [-s <seg>] [-n <nível>] -g<saida>
```

5.1.6.3 Argumentos

<banco> é o nome do dicionário de dados *OpenBASE*.

<seg> é o código de segurança - Default = 1

<nível> é a palavra de nível - Default = a

<saida> é o arquivo onde é gravado o esquema do banco SQL.

<saída1> é o arquivo onde são gravados os comandos CREATE INDEX do banco SQL.

5.1.6.4 Utilização

Utilitário necessário para se gerar um esquema de um banco SQL, a partir de um dicionário de dados *OpenBASE*. Com isso, tal banco SQL pode ser acessado pela OPUS via comandos ODBC.

5.1.7 BDVERI

```
bdveri -b <banco> [-s <segurança>] [-n <nível>] [-a <arq_ban>] [-O] [-e] [-z]  
[-v <valor>] [-i {<nome>|<num2>}I-f] [-c] [-l]
```

```
bdveri -b[<percurso>] <nome_banco>.-t
```

5.1.7.1 Argumentos

-l	A opção -l indica que o banco será aberto em modo 2 (compartilhado) ao invés de modo 3 (exclusivo). Com isso, é possível se executar o bdveri mesmo que o banco de dados esteja em uso por outro processo.
-t	A opção -t indica que para um banco com a opção de espelhamento, serão verificados os arquivos originais e seus espelhos. As duas bases de dados devem estar atualizadas e compatíveis.

5.1.8 BDINDC

```
bdindc ... -b <banco> [-s <segurança>] [-n <nível>] [-a <arq_ban>  
[-S] [-i {<nome>|<número1>}] [-c] [-v <valor>] -r [-m <numero>]] [-O | -o] -l
```

5.1.8.1 Argumentos

-l	Indica que o banco será aberto em modo 2 (compartilhado) no lugar de modo 3 (exclusivo), assim permitindo que o e bdindc seja executado mesmo que o banco esteja em uso por outro processo
-----------	--

5.1.9 BDSERV

5.1.9.1 Propósito

Novos parâmetros foram adicionados na inicialização do programa BDSERV, como a definição do tamanho da memória compartilhada, e inibição da mensagem que o bdserv já está carregado. O bdserv, pode ter sua localização (percurso), informada assim como o tamanho máximo da mensagem recebida.

O bdserv pode ter os parâmetros:

-s<caminho>	Onde caminho é a localização do programa. Exemplo: c:\usr1\bdserv.exe Valor default: c:\usr\tsgbd\bdserv.exe Se o bdserv estiver no diretório c:\usr\tsgbd, esta opção não é necessária para carregá-lo
-b<num>	Onde num é o tamanho máximo da mensagem. Valor (default = 1000). Esta opção do BDSERV, só terá efeito, se o bdserv no servidor e as bibliotecas no

cliente (libfacbib.lib ou cliwin32.lib), estiverem na mesma versão.

5.1.9.2 Utilização

Para que o *OpenBASE* seja acessado no Windows 95. Ou Windows NT, o arquivo bdsgrbd.exe deve estar inicializado. O arquivo bdserv.exe, deve ser inicializado, caso o Windows (95 ou NT) trabalhe como servidor de banco de dados.

Os programas bdsgrbd e bdserv devem ser iniciados como serviços no Windows.

No Windows 9X, incluir no registry com o regedit a chave:

Hkey_LM\Software\Microsoft\Windows\Current\Version\RunServices

O **nome** do valor será: **bdsgrbd** e o **valor** será: **c:\usr\tsgbd\bdsgrbd.exe**

O **nome** do valor será: **bdserv** e o **valor** será: **c:\usr\tsgbd\bdserv.exe**

5.1.10 TimeOut no BDSERV

5.1.10.1 Propósito

Retira o processo bdserv filho (fork) do ar quando uma aplicação cliente se conecta ao bdserv e permanece um determinado tempo sem se comunicar.

5.1.10.2 Sintaxe

```
nohup bdserv -t<tempo> &
```

5.1.10.3 Argumentos

<tempo> indica o tempo em segundos, que o bdserv inicializado por uma aplicação cliente, deva permanecer no ar, sem haver comunicação.

5.1.10.4 Utilização

Esta opção é útil, caso uma aplicação "cliente", se conecta ao bdserv, criando um processo filho (fork) do bdserv, e tal conexão não é encerrada (disconnect), devida saída anormal da aplicação cliente.

5.1.10.5 Exemplos

```
nohup bdserv -t 3600
```

O processo bdserv filho, terminará após 3.600 segundos (1 hora) sem comunicação

5.1.10.6 Observação

Para que esta opção do bdserv funcione, é necessário que no programa cliente seja especificado:

set alarm to [<tempo>] na OPUS ou

AlarmeServidor(<tempo>) na Cliwin32.dll

onde:

<Tempo> é o número de segundos que o bdserv filho permanece ativo sem comunicação.

Na OPUS, "set alarm to" e na Cliwin32.dll, **AlarmeServidor(-1)** retornam aos defaults (tempo infinito).

5.1.11 BDCDBF

5.1.11.1 Propósito

Converte arquivos com extensão ".dbf" (Clipper, Dbase, Dialog ou FoxBase) para um arquivo *OpenBASE* ou arquivo externo. Novas opções foram adicionadas a este utilitário, visando agilizar tal conversão.

5.1.11.2 Sintaxe

```
bdcdbf -a<arq_dbf> [-b <banco>]
```

```
[-c <numero>][-t N | S] -f -d [F | B] -I <esquema> -S -w<aa>
```

5.1.11.3 Argumentos

-a <arq_dbf> nome do arquivo .dbf.

-b <banco> nome do Banco de Dados que será criado. Caso se omita, assume o valor xxxxxx.

-c <número> indica o número do item que corresponderá a chave. Caso se omita, assume o valor 1.

-I inverte os números binários lidos.

-t [N | S] indica o tipo para dados numéricos gerados (N ou S). Caso se omita, assume o valor N.

- f** indica a conversão do arquivo para um arquivo externo tipo L da OPUS.
- d** determina a conversão de dados tipo DATA, sendo F para FRENCH e B para BRITISH. Caso se omita, assume o valor F.
- l** Indica que o nome dos itens serão convertidos para letras minúsculas, no esquema a ser gerado.
- s <esquema>** Indica o nome do esquema a ser gerado, que não será esvaziado, a cada execução do utilitário bdcdbf.
- S** Indica que todos os arquivos com extensão DBF (arquivo.bdf), serão convertidos para <arquivo> _.S, para que sejam recarregados de uma só vez pelo utilitário bddad (-z -S).
- w <aa>** Indica que na conversão de uma data com 2 dígitos no ano, que os valores do ano (aa) lidos e menores ou igual a <aa>, serão convertidos para 19aa. Já os valores do ano (aa) lidos e maiores que <aa> serão convertidos para 20aa.

5.1.11.4 Exemplos

```
bdcdbf -b<banco> -a produtos.dbf -l -s esquema.esq -S
```

5.1.12 BDESC (Opção -q)

5.1.12.1 Propósito

Descarrega um arquivo em um banco *OpenBASE*, de forma que os dados são gerados na forma de comandos SQL.

5.1.12.2 Sintaxe

```
bddesc -b <banco OpenBASE> [-s<codigosegurança>][-n<pal.nivel>]
-a [nome do arquivo do banco OpenBASE]
-g [arquivo com dados descarregados] -q
```

5.1.12.3 Utilização

A opção **-q** é útil, quando se necessita descarregar dados de arquivo (s) *OpenBASE* (ou de todo o banco *OpenBASE*), já que os dados são gerados na forma de comando SQL:

```
INSERT INTO <arq> VALUES (<val1>, ... ,<valn>)
```

5.1.12.4 Exemplos

```
bddesc -b exemplo -atabela -gsaida -q && descarrega o arquivo tabela.
bddesc -b exemplo -S -q && descarrega todos arquivos do banco.
```

6 Dicas e macetes

6.1 Backspace

Segue abaixo uma dica importante de como se deve configurar o Backspace no Linux, para que o sistema **GERAL** ou um programa **OPUS** não abortem ao se teclar o Backspace.

1- Posicione-se no diretório: /usr/lib/kbd/keymaps/i386/qwerty

2- Se seu teclado for padrão ABNT (com ç) descompacte (gunzip) o arquivo **br-abnt2.kmap.gz**, senão descompacte(gunzip) o arquivo

us-acentos.kmap.gz.

3) Abra o arquivo (br-abnt2.kmap ou us-acentos.kmap) de acordo com tipo do seu teclado.

4) Na linha 65:

```
keycode 14 = Delete Delete
```

altere a palavra Delete para Backspace.

5) Comprima o arquivo(gunzip)

6) Execute o comando loadkeys de acordo com tipo do seu teclado.

```
loadkeys br-abnt2.kmap.gz
```

```
loadkeys us-acentos.kmap.gz
```

Obs.: O mesmo pode ser feito em relação a tecla

<control>+\
us_acentos.kmap (br_abnt2.kmap)

6.2 JOBSLEFT

6.2.1 Propósito

Esta função retorna o número de Jobs na fila da impressão (PrintSpooler).

6.2.2 Sintaxe

```
<varn> = JobsLeft()
```

6.2.3 Utilização

Caso o programa queira saber quantos Jobs estão na fila de impressão no momento, e se queira confirmar alguma impressão com o usuário, para que ele saiba se vai demorar ou não.

6.2.4 Exemplos

```
Prog  
For i = 1 to 5
```

6.3 SET PRINT ON

```
? "alo"  
set print off  
n=JobsLeft()  
x=MessageBox (str(n))  
Next
```

6.4 QUIT

6.4.1 Propósito

Forçar término de um programa.

6.4.2 Sintaxe

```
Quit [(varn.>)]
```

6.4.3 Argumentos

<varn> Código de retorno do programa.

6.4.4 Exemplos

```
Quit (1)
```

6.5 Replicação assíncrona automática

6.5.1 Propósito

Permitir que um Banco de Dados *OpenBASE* seja espelhado (copiado) para uma outra máquina, funcionando como um backup incremental.

6.5.2 Sintaxe

No esquema, deve-se definir o arquivo, segundo a Sintaxe

nome: arquivo E replicação

6.5.3 Utilização

Ao se definir um esquema com a cláusula REPLICAÇÃO ao lado do nome do arquivo, serão criados automaticamente três novos itens no arquivo:

ITEM	TIPO
------	------

DAT	T8
DEL	L1
CHA (0)	D7 Pos (DAT+1)

É criado também um arquivo <dicionário>. S que guarda a data e hora da última replicação de cada arquivo replicado.

O item DAT armazena a data e hora que o registro foi incluído, alterado ou excluído.

O item DEL, armazena o valor 1, quando o registro for excluído (exclusão lógica)

Quando um registro é incluído ou alterado, o item DEL recebe o valor 0.

Os registros com DEL=1 não são lidos ou excluídos, a menos que a opção DELETED (SET DELETED ON) esteja ligada.

Caso seja incluído um registro com o mesmo valor de chave do registro deletado logicamente (DEL=1), o item DEL é alterado de 1 para 0.

Daí a necessidade de os arquivos replicáveis serem do tipo entidades (chave primária) ou do tipo relacionamento com pelo menos um item chave única.

6.5.4 Funcionamento

No servidor (máquina para onde serão replicados os dados) ocorre:

nohup bdsrep Inicialização do servidor de replicação

É recebido o nome do banco e o arquivo a replicar.

servidor de replicação (bdsrep), lê a chave primária de cada registro recebido e:

se não existir e DEL = 0, inclui

se existir e DEL=1, exclui

se existir e DEL=0, altera

No cliente (máquina de onde vem os dados replicados) deve-se executar o comando:

```
bdcrep -h<host servidor> -b [percurso bd origem] <banco bd origem> -r [percurso bd destino] <banco bd destino> [-s<seg>] [-n<nivel>]
```

O serviço bdsrep é inicializado no host servidor. (é carregado automaticamente)

É enviado o nome do banco no servidor e o arquivo replicar.

Serão pesquisado no banco cliente, todos os registros que possuam o item CHA>=data_hora da última replicação (contida no arquivo <dicionario.S>) e CHA<=data_hora do sistema operacional. Tais registros são enviados para o servidor.

Todos os registros enviados e que possuam DEL=1 serão excluídos fisicamente no cliente.

arquivo bdsrep na máquina do servidor é finalizado. (sai do ar)

Obs.1: Para o bom funcionamento da replicação assíncrona, os dois bancos devem ser iguais, tanto no cliente quanto no servidor.

Obs.2: Caso se deseje replicar um arquivo que já possua dados gravados, deve-se:

descarregar o arquivo com bddesc.

acrescentar a clausula REPLICACAO no esquema, ao lado do arquivo que se deseja replicar.

compilar o banco de dados (define) recriando o arquivo.

carregar o arquivo com bdadict, utilizando-se a opção -g.