I-SQL

MoreData

Índice

1. INTRODUÇÃO

1.1. Noções de Bases de Dados Relacionais: Um exemplo

Para uma melhor apreensão global e integrada dos comandos do INFORMIX-SQL que irão ser focados ao longo deste manual, todos os exercícios de aplicação serão executados no contexto da base de dados de aprendizagem STORES que se descreve a seguir. Durante a apresentação da base de dados STORES irão sendo introduzidos e explicados conceitos e termos utilizados no Modelo Relacional, que normalmente se utiliza para descrever uma base de dados relacional.

STORES

A base de dados STORES pretende ser o suporte da informação necessária á gestão duma firma que comercializa artigos de desporto.

1.1.1. Modelo de Dados

O modelo de dados é constituído pelos seguintes elementos: (ver apêndice 1)

ENTIDADES

São representadas pelos quadrados.

Definem-se como:

'Classes de objectos com relevância para o negócio/instituições inequivocamente identificadas'
ou

'Classes de objectos sobre os quais o negócio/instituição necessita de manter informação actualizada'.

A cada entidade estará ligada uma tabela ou um ficheiro, conforme o sistema de informação descrito pelo modelo de dados for implementado num Sistema de Gestão de Bases de Dados Relacionais, ou num sistema clássico de ficheiros indexados ou sequenciais.

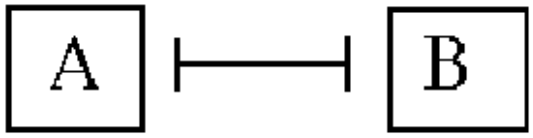
RELAÇÕES

São representadas pelas linhas que unem os quadrados do modelo de dados e definem as relações existentes entre as entidades (tabelas) .

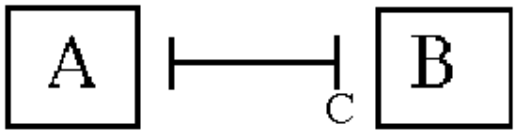
TIPOS DE RELAÇÕES

Um a um

Fortes: A 1 elemento de A corresponde 1 e só 1 elemento de B e vice-versa.

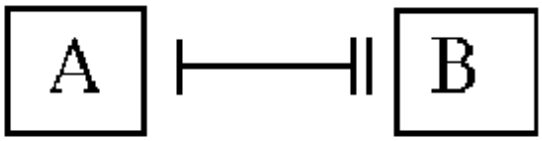


Fracas (condicional): A 1 elemento de A corresponde 1 e só 1 elemento de B e vice-versa, mas pode existir A sem B.

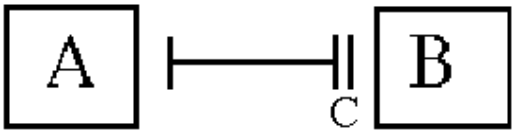


Um a N (várias)

Fortes: A 1 elemento de A corresponde pelo menos 1 elemento de B e a 1 elemento de B nunca corresponde mais do que 1 elemento de A.



Fracas (condicional): A 1 elemento de A corresponde pelo menos 1 elemento de B e a 1 elemento de B nunca corresponde mais do que 1 elemento de A, mas pode existir A sem B.



1.1.2. Tabelas da Base de Dados STORES

Na base de dados STORES existem as seguintes tabelas: (consulte os documentos 'Modelo de Dados' e 'Detalhe das entidades do apêndice 1).

1 customer

CLIENTES: Contém informações sobre 18 distribuidores dos artigos comercializados. Inclui o número de identificação, nome, morada e número de telefone do cliente.

2. orders

ENCOMENDAS: Contém encomendas pedidas pelos clientes registados na tabela customer. Inclui o número de identificação da encomenda, identificação do cliente, datas e outras caracteristicas da encomenda.

3. itens

LINHAS DE ENCOMENDA: Contém informações sobre o artigo encomendado. Inclui o número do artigo, a identificação do fabricante, a quantidade do artigo encomendada e o valor da linha de encomenda.

4. stocks

ARTIGOS: Contém as características de cada artigo. Inclui o número de identificação, fabricante, descrição e preço de cada artigo.

5. manufact

FABRICANTE: Contém a identificação e o nome do fabricante.

1.1.3. Estrutura duma tabela

A estrutura duma tabela relacional é constituída por um determinado número de colunas ,tendo todos os elementos de cada coluna as mesmas características, e por um número indeterminado de linhas constituídas por um elemento de cada uma das colunas (ver fig 1).

	col1	col2	col3	col4
lin 1	1	azul	metal	500
lin 2	2	azul	madeira	200
lin 3	3	verde	metal	500
.				
.				
.				
lin n				

Fig. 1

Como se pode ver existe uma correspondência entre uma tabela e um ficheiro sequêncial.

Assim:

TABELA	FICHEIRO SEQUÊNCIAL
linha	registo
coluna	campo

CHAVES PRIMÁRIAS

Existem dois tipos de colunas numa tabela:

1- A(s) coluna(s) que identificam cada uma das linhas da tabela às quais se chama CHAVE PRIMÁRIA da tabela ou entidade, pois é o identificador da entidade.

2- As restantes colunas da tabela e que têm cada uma delas uma relação binária com a chave primária.

CHAVES ESTRANGEIRAS

A ligação entre ENTIDADES (tabelas) é feita através de determinadas colunas da primeira entidade çujos valores seleccionam na segunda entidade a(s) linha(s) que tenham chaves primariás correspondentes. Às colunas da primeira tabela que são utilizadas desta forma e não fazem parte da chave primária chamam-se CHAVES ESTRANGEIRAS ou SECUNDÁRIAS.

No apêndice 1 apresentam-se os seguintes 4 documentos que descrevem a base de dados STORES e se evidenciam os conceitos descritos nos parágrafos anteriores.

Modelo de Dados	descreve a base de dados
Detalhe das Entidades	descreve as tabelas.
Lista de Entidades	lista as tabelas com as respectivas chaves primárias e estrangeiras.

Esquema gráfico representando as 5 tabelas da base de dados e a forma como se ligam entre si.

1.2. INFORMIX -SQL

INFORMIX-SQL é um Sistema de Gestão de Bases de Dados Relacionais e que executa as seguintes tarefas:

```
Criar Bases de Dados e Tabelas.
Definir ecrans (forms).
Inserir e alterar dados usando ecrans (forms).
Carregar dados de ficheiros do sistema operativo.
Fazer buscas quer através de forms quer através de SQL (System
Query Language) interactivo.
Produzir e formatar relatórios (reports).
Definir um Menu que inclua chamadas ao INFORMIX-SQL, a utilitários e
a outros programas.
```

Para aceder ao INFORMIX-SQL digite **isql** e aparecerá no ecran o Main Menu do INFORMIX-SQL, se este estiver bem instalado.

1.2.1. Os ecrans do INFORMIX-SQL

O INFORMIX-SQL usa 2 tipos de ecrans; o ecran de Menu e o ecran para entrada de texto.

Os Ecrans de Menu

Os ecrans de Menu são do tipo do Main Menu. A primeira linha do ecran contém as opções do ecran de menu. A linha do ecran contém a descrição da opção actualmente seleccionada.

Como seleccionar as opções do ecran de Menu

Para seleccionar as várias opções do ecran de Menu pode usar um dos métodos seguintes :

Seleccionar a opção pretendida, percorrendo as opções disponíveis com a tecla SPACEBAR ou com as teclas de deslocacão do cursor "->" e "<-", e premir a tecla RETURN.

Premir a primeira letra da opção que pretenda seleccionar. Tanto faz digitar quer a letra maiúscula quer a letra minúscula.

Como Abandonar o Menu

Em todos os ecrans de menu existe a opção EXIT utilizada para abandonar o menu.

Como Chamar pelo Help

Sempre que tenha dificuldades em interpretar as opções do menu digite CTRL-W e aparecerão no ecran uma série de mensagens descrevendo a opção seleccionada. Para retornar ao ecran de menu pressione RETURN

Ecrans com Linhas de Pergunta

Estes ecrans não têm opções e permitem introduzir os novos nomes de bases de dados, tabelas, ficheiros com instruções RDSQL, etc. Depois de introduzido o texto pressione RETURN e o Informix-SQL processará a(s) tarefa(s) seleccionada(s).

Como nestes ecrans não existe opção EXIT para abandonar o ecran de entrada de texto e voltar ao menu anterior deve pressionar a tecla DEL.

O procedimento para chamar pelo HELP é idêntico ao do ecran de menu, ou seja deve digitar CTRL-W.

1.2.2. As opções do Main Menu

As tarefas que o INFORMIX-SQL permite executar são sempre iniciadas a partir de selecção duma opção do Main Menu.

As opções do Main Menu são as seguintes :

Form Permite criar, retirar, modificar, compilar e correr um ecran formatado (screen form).

Report Permite criar, retirar, modificar, compilar e correr um relatório (report).

Query - Language Permite a utilização do RDSQL, que é uma linguagem do INFORMIX-SQL que utiliza o standard SQL (System Query Language).

User - Menu Permite criar, modificar e correr um menu para uso geral.

Database Permite criar e retirar uma base de dados e torná-la na base de dados corrente.

Table Permite criar, retirar e modificar uma tabela e consultar informações sobre a tabela.

Exit Permite sair do INFORMIX-SQL e voltar ao sistema operativo.

1.2.3. Ambiente do INFORMIX - SQL

Há três variáveis de ambiente que têm de estar correctamente definidas para que o INFORMIX-SQL possa funcionar. Duas destas variáveis (PATH, TERM) são variáveis gerais do UNIX, a outra INFORMIXDIR, é específica dos produtos da Informix Inc. Há um conjunto de variáveis de ambiente específicas do INFORMIX que lhe afectam o comportamento, cujo nome começa sempre pelas letras "DB", algumas variáveis do S.O. UNIX afectam também o comportamento do INFORMIX.

INFORMIXDIR Num primeiro passo para a utilização do INFORMIX-SQL, é necessário saber onde foram instalados os programas. Todos os programas distribuidos com o INFORMIX-SQL estão armazenados num conjunto de subdirectórios abaixo de um directório principal (Ex: "/usr/informix" nas máquinas UNIX, "informix" nas máquinas DOS). Se o produto foi instalado noutro directório (por ex: "/u0/informix"), então é necessário dizê-lo ao INFORMIX-SQL, definindo a variável INFORMIXDIR. Se estiver a

```
INFORMIXDIR=/u0/informix
export INFORMIXDIR
```

Usando o C shell (csh) a sintaxe é:

```
setenv INFORMIXDIR \u0\informix
```

No caso de se estar num sistema DOS (MS-DOS ou PC-DOS) então a sintaxe é:

```
set INFORMIXDIR=C:\u0\informix
```

Para que não seja necessário proceder a esta operação em cada sessão de trabalho pode modificar-se o ficheiro .profile (sh ou ksh) ou .login (csh) ou AUTOEXEC.BAT (DOS). Esta técnica aplica-se a todas as variáveis de ambiente.

NOTA: Em algumas máquinas se o INFORMIX-SQL for instalado no seu local por defeito (/usr/informix em UNIX), então não é necessário definir a variável INFORMIXDIR. No entanto aconselha-se a que se defina sempre esta variável.

PATH A segunda variável a definir serve para ter a certeza de que o interpretador dos comandos ou o shell saibam onde estão os comandos do INFORMIX, esta é uma variável do S.O. UNIX, para mais pormenores deve consultar-se a documentação do Sistema Operativo. Todos os comandos estão instalados em "$INFORMIXDIR/bin" (isto é, "/usr/informix/bin"). Então a forma de inicializar a variável PATH é, por exemplo:

```
PATH=$INFORMIXDIR/bin:$PATH
export PATH
```

TERM e TERMCAP Tal como a maioria dos produtos para o sistema UNIX, o INFORMIX-SQL trabalha com quase todo o tipo de terminais. Os produtos Informix usam o ficheiro de descrição de terminais termcap normal do sistema UNIX. No caso de o ficheiro "/etc/termcap" não existir ou de se pretender usar características do terminal não definidas na descrição standard do UNIX, é necessário inicializar convenientemente a variável de ambiente TERMCAP, do seguinte modo:

```
TERMCAP=$INFORMIXDIR/etc/termcap
export TERMCAP
```

pois vem com o INFORMIX-SQL um ficheiro onde estão definidos a maioria dos terminais. Este encontra-se no directório "etc" abaixo do directório onde foi instalado o INFORMIX-SQL.

DBDATE Consideremos o seguinte formato de data "06/09/88", para os americanos lê-se 9 de Junho de 1988; para os europeus lê-se 6 de Setembro de 1988. Pode-se definir como é se quer que o INFORMIX-SQL interprete esta data, para isso é necessário inicializar correctamente a variável DBDATE. O formato por defeito é:

```
DBDATE=mdy2/
export DBDATE
```

Esta variável deve conter uma sequência das letras "m", "d" e "y" por qualquer ordem para indicar qual a ordem em que aparecem respectivamente o mês, o dia e o ano, um caracter separador dos dias meses e anos (usalmente "/" ou "-"), e o número de digitos do ano (2 ou 4).

O exemplo acima indica que os elementos introduzidos numa variável tipo data estão com a ordem mês, dia, ano, que se tem 2 digitos para o ano e que o caracter separador é "/". Pode-se inicializar a variável DBDATE de qualquer uma das formas apresentadas a seguir:

DBDATE	Ex: para escrever 1 de Fevereiro de 2003
mdy2/	02/01/03
dmy2/	01/02/03
dmy4/	01/02/2003
mdy4/	02/01/2003
y4md-	2003-02-01
y2md/	03/02/01

DBMONEY O INFORMIX-SQL é um produto americano, como tal os valores monetários são representados como dolares "$", e o indicador de casa decimal por defeito é ".". Para usar outro formato nas colunas MONEY é necessário incializar a variável DBMONEY, especificando o separador das partes inteira e decimal, e sufixo ou prefixo indicador de unidade. Por exemplo na alemanha onde se usa "Deutsch Marks", fariamos:

```
DBMONEY=,DM
export DBMONEY
```

assim teriamos para escrever doze mil Deutsch Marks apareceria "123.000,00DM".

DBPATH Esta variável de ambiente serve para indicar ao INFORMIX-SQL onde vai procurar bases de dados, forms (écrans de entrada de dados), reports (relatórios), ou scripts de RDSQL.

DBPRINT Quando se envia um report para uma impressora, este é enviado para um programa que se encarrega das tarefas de gestão de impressões (por ex: lpstat no UNIX), o nome do tal programa pode ser indicado na variável DBPRINT. Em qualquer altura se pode mudar a impressora de destino por defeito, mudando a variável DPRINT. .P Se quizermos por exemplo enviar para a impressora "beta" com as opções "-onb", fazemos:

```
DPRINT="lp -s -dbeta -onb"
export DBPRINT
```

DBEDIT Esta variável de ambiente serve para definir o editor de texto que é chamado por defeito. Por exemplo:

```
DBEDIT=vi
export DBEDIT
```

DBTEMP Por defeito o UNIX cria os ficheiros temporários no directório "/tmp". Para indicar outro directório deve colocar-se o nome deste na variável DBTEMP.

DBDELIMETER Esta variável serve para definir os delimitadores, dos campos usados pelos comandos LOAD e UNLOAD. No caso desta variável não estar definida o INFORMIX assume o caracter "|".

DBMENU Esta variável define qual é o menu que é executado quando se escolhe a opção "USER-MENU". Se não for inicializada é executado o menu MAIN.

```
DBMENU=main
export DBMENU
```

SQLEXEC Esta variável indica qual o programa de acesso à base de dados (database engine ou database mechanism) usado. Esta variável só pode ser usada quando se utilize o INFORMIX-TURBO ou ON-LINE. Se na mesma máquina coexistirem os dois métodos de acesso à base de dados deve inicializar-se esta variável com:

```
SQLEXEC=sqlexec
export SQLEXEC
```

para a versão standard e

```
SQLEXEC=sqlturbo
export SQLEXEC
```

para a versão turbo.

1.2.3.1.FORMS e PERFORM

Desenho e processamento de ecrans.

O INFORMIX-SQL permite que se desenhem e compilem ecrans e que estes comuniquem interactivamente com a base de dados através dos programas FORMBUILD e PERFORM respectivamente.

O PERFORM permite as seguintes operações sobre a base de dados, sem ser necessário conhecer a linguagem RDSQL:

- Inserir, actualizar e suprimir dados nas tabelas da base de dados.
- Validar, dentro de certos limites, os valores inseridos nos campos definidos no FORM.
- Consultar a base de dados, seleccionando os respectivos dados segundo vários critérios.
- Combinar os dados elementares da base de dados (as colunas das tabelas), nomeadamente através de operações algébricas, e apresentá-los em campos específicos do ecran.

Uma form é escrita num ficheiro, por exemplo através dum editor de texto, e depois compilada pelo programa FORMBUILD. A versão compilada pode então ser executada sempre que necessário pelo programa PERFORM.

Assim a form tem como suporte físico dois ficheiros: o programa fonte com a extensão ".per" e o programa compilado com a extensão ".frm".

1.3. Os Menus Associados ao Form.

Como introdução aos Forms vamos primeiro descrever a sua apresentação e as suas funções básicas, ou seja apresentando e explicando os dois principais menus associados aos Forms: o menu FORM e o menu RUN.

1.3.1. O Menu FORM

Entre no menu INFORMIX-SQL e escolha a opção Form --> entrou no menu Form. O menu Form apresenta as seguintes 7 opções:

RUN Esta opção executa uma form já compilada. Quando a escolher aparecerá um ecran com a lista das forms existentes. Pode seleccionar a form que pretender iniciando assim a execução pelo PERFORM dessa form, aparecendo a seguir o menu de PERFORM.

MODIFY Esta opção permite fazer alterações a forms já existentes. Tal como na opção RUN aparece um ecran com a lista das forms existentes para seleccionar uma delas. Ao escolher uma form aparece-lhe o desenho da form, num editor do sistema, pronto a ser alterado. Depois de concluidas as alterações e de ter saído do editor, aparece o menu MODIFY FORM com as seguintes opções:

COMPILE: Compila a form. Se a form tiver erros estes são colocados num ficheiro juntamente com a form, podendo então corrigi-la e compilá-la de novo. Estes procedimentos podem ser executados ciclicamente até obter uma compilação da form sem erros.

SAVE-AND-EXIT: Grava as alterações à form no ficheiro fonte, sem executar compilação.

DISCARD-AND-EXIT: Não grava as alterações à form no ficheiro fonte, mantendo neste a versão anterior ao inicio das alterações. Também não executa compilação.

GENERATE Cria automáticamente uma form com um formato por defeito (default). Quando esta opção é seleccionada aparecem ecrans para selecção da base de dados, se ainda não foi seleccionada, e para escolha das tabelas que constituirão a form. Veremos mais em pormenor esta opção na secção seguinte "Criação e Compilação duma Form Default".

NEW Permite criar uma form a partir dum ficheiro vazio e através dum editor. Quando terminar a edição da form o INFORMIX-SQL comportar-se-á de forma igual à da opção MODIFY.

COMPILE Compila uma form já existente. Apresenta um ecran para seleccionar a form a compilar. Se a compilação falhar comportar-se-á de forma igual à da opção MODIFY.

DROP Remove uma form da base de dados, tanto a versão fonte como a versão objecto (compilada). Apresenta um ecran para escolher a form a suprimir e pede confirmação para fazê-lo.

EXIT Retorna ao menu INFORMIX-SQL.

1.3.2. O Menu Perform.

Viu-se anteriormente que quando se selecciona no menu de FORM a opção RUN se chega ao ecran de PERFORM com os respectivos menu e desenho de ecran propriamente dito (o screen). Neste capítulo vamos descrever e exemplificar o uso das opções deste menu.

Primeiro seleccione a opção RUN --> Obtem-se o ecran com a lista das forms existentes.

A seguir escolha a form CUSTOMER --> Obtem-se o ecran de PERFORM para a tabela CUSTOMER com a seguinte forma:

```
PERFORM:   Query  Next  Previous  Add  Update  Remove  Table  Screen...
Searches the active database table.              ** 1: customer table**
_____________________________________________________________________
_____________________________________________________________________

                                CUSTOMERS

Customer Number:  [           ]

Company:  [                    ]
First Name:  [               ]       Last Name:  [               ]

Address:  [                    ]
          [                    ]

City:  [               ]       State:  [  ]    Zip:  [     ]

Telephone:  [                  ]

_____________________________________________________________________
_____________________________________________________________________
```

com as opções do menu de PERFORM alinhadas nas duas primeiras linhas e imediatamente a seguir o desenho da form para a tabela CUSTOMER.

1.3.2.1. A Opção QUERY.

A opção QUERY permite consultar a base de dados duma forma muito simples. Obtêm-se resultados idênticos aos conseguidos com a instrução RDSQL SELECT, introduzida no capítulo 2.3.1, embora a forma de apresentação dos dados seja diferente e possa ser desenhada ao nosso gosto.

Para especificar quais as linhas a seleccionar existem algumas regras que são introduzidas a seguir:

Se nenhum dos campos da form for preenchido serão seleccionadas todas as linhas (corresponde a um SELECT sem cláusula WHERE).

Para indicar quais os critérios de selecção preencha os campos sobre os quais pretende formular condições de selecção.

Os campos que não forem preenchidos não fazem parte das condições de selecção.

Para especificar quais as condições de selecção utilize os seguintes caracteres de controlo:

```
= ou " "    igual a
<>          diferente de
>           maior que
<           menor que
 >=         maior ou igual
<=          menor ou igual
:           intervalo de valores
```

```
*           zero ou mais caracteres
<<          o valor mínimo
>>          o valor máximo
```

Estes caracteres de controlo variam segundo a versão do INFORMIX-SQL. Vejam os manuais da vossa versão. Para transitar pelos vários campos do ecran utilize:

A tecla **RETURN** ou **TAB** para avançar para o campo seguinte.

A tecla **BACKSPACE** recua um espaço ou para o campo anterior se o cursor estiver colocado na primeira coluna do campo.

As teclas **CTRL B** para retornar ao campo anterior.

Depois do ecran estar preenchido use:

ESC para executar o query.

DEL para anular e retornar à situação anterior.

CTRL C para pôr a espaços todos os campos do ecran.

CTRL W para chamar uma listagem de help onde se explicam os vários comandos de edição para os campos do ecran e o significado dos caracteres de controlo.

Vejamos um exemplo para a form da tabela CUSTOMER seleccionada anteriormente. Seleccione a opção QUERY e introduza "101:110" no campo "Customer Number". Pressione a tecla ESC para executar o query obtendo o resultado seguinte:

```
PERFORM:   Query  Next  Previous  Add  Update  Remove  Table  Screen...
Searches the active database table.              ** 1: customer table**
_____________________________________________________________________
_____________________________________________________________________

                              CUSTOMERS

Customer Number:  [101        ]

        Company:  [All Sports Supplies ]
     First Name:  [Ludwig         ]       Last Name:  [Pauli          ]

        Address:  [213 Erstwild Court  ]
                  [                    ]

           City:  [Sunnyvale      ]       State:  [CA]    Zip:  [94086]

      Telephone:  [408-789-8075      ]


_____________________________________________________________________
_____________________________________________________________________

9 row(s) found
```

Exercite-se executando mais queries sobre a tabela CUSTOMER utilizando outros parâmetros de controlo.

1.3.2.2. As Opções NEXT e PREVIOUS.

No exemplo anterior o resultado tem várias linhas. No entanto no screen apresentado só visualizamos uma linha da tabela CUSTOMER de cada vez. Para visualizarmos as restantes linhas do resultado usam-se as opções NEXT e PREVIOUS, para passar para as linhas seguintes e anteriores do resultado respectivamente. Percorra o resultado do exemplo anterior usando estas opções.

NOTA: Se digitar um número antes da letra correspondente à opção (N ou P) avançará ou recuará esse número de linhas do resultado.

1.3.2.3. A Opção ADD

A opção ADD permite inserir dados na base de dados. para tal basta inserir nos campos do ecran os dados correspondentes e pressionar ESC. Algumas validações básicas são feitas automàticamente, tais como:

-um campo numérico só aceitará dados numéricos.

-rejeitará datas com formato incorrecto.

-se um campo do ecran, corresponder a uma coluna que não aceite valores NULL, tem de ser preenchido.

Como exemplo vamos seleccionar a opção ADD do menu de PERFORM da tabela CUSTOMER e introduzir a seguinte linha:

```
ADD: ESCAPE adds new data. INTERRUPT discards it. ARROW keys move cursor
Adds new data to the active database table       ** 1: customer table**
_____________________________________________________________________________
_____________________________________________________________________________

                                CUSTOMERS

Customer Number:  [0         ]

        Company:  [SAMAR               ]
     First Name:  [Jose           ]       Last Name:  [Liberto        ]

        Address:  [Rua Morais Soares   ]
                  [N. 16               ]

           City:  [LISBOA         ]       State:  [CA]    Zip:  [1500 ]

      Telephone:  [100-001-564515    ]


_____________________________________________________________________________
_____________________________________________________________________________
```

Note que o campo "Customer Number" está protegido e não permite sequer que o cursor entre nele pois corresponde a uma coluna com tipo de dados SERIAL, sendo preenchido automáticamente.

Note também que se não preencher o campo "State" o PERFORM não processa a operação de ADD e dá uma mensagem de erro alertando para o facto do campo corresponder a uma coluna que não aceita valores NULL. Note ainda que este campo não aceita um valor qualquer, sendo isto resultado duma validação que foi especificada pelo programador. Veremos adiante como. Como resultado da operação ADD obtem-se o seguinte ecran:

```
PERFORM:   Query  Next  Previous  Add  Update  Remove  Table  Screen...
Adds a row to the active database table.          ** 1: customer table**
_____________________________________________________________________
_____________________________________________________________________

                                CUSTOMERS

Customer Number:  [119        ]

        Company:  [SAMAR               ]
     First Name:  [Jose           ]       Last Name:  [Liberto        ]

        Address:  [Rua Morais Soares   ]
                  [N. 16               ]

           City:  [LISBOA         ]       State:  [CA]    Zip:  [1500 ]

      Telephone:  [100-001-564515    ]


_____________________________________________________________________
_____________________________________________________________________

Row added
```

Experimente introduzir os mais variados tipos de dados nos campos do ecran e observe a reacção do PERFORM.

1.3.2.4. A opção UPDATE

A opção UPDATE permite alterar o conteúdo dos campos dos dados actualmente no ecran.

O funcionamento é idêntico ao da opção ADD, excepto o facto de que a linha da tabela já existe.

Como exercício tente alterar no ecran anterior o campo "Zip" de "1500" para "1600" e verifique se a alteração foi efectuada executando a opção query para o "Customer Number" igual a "119".

NOTA: A opção UPDATE é indicada para o tratamento ecran a ecran, ou seja linha a linha. Para alterar de forma sistemática várias linhas utilize a instrução RDSQL UPDATE.

1.3.2.5. A Opção REMOVE

A opção REMOVE suprime a linha actualmente no ecran da base de dados.

A sua utilização é muito simples bastando seleccionar a opção e confirmar se pretende ou não suprimir a linha exposta no ecran.

Como exercício suprima a linha actualmente no ecran (deverá ter "Customer Number" igual a "119").

NOTA: Tal como para a opção UPDATE, a opção DELETE também é indicada para o tratamento ecran a ecran. Para suprimir de forma sistemática várias linhas da tabela utilize a instrução RDSQL DELETE.

1.3.2.6. A Opção TABLE.

Esta opção permite, quando num ecran coexistem mais de uma tabela, seleccionar a tabela pretendida

1.3.2.7. A Opção OUTPUT

Esta opção permite gravar o ecran actual ou todos os ecrans, constituintes do resultado dum query, num ficheiro do sistema.

A sua utilização é simples bastando responder às questões que lhe são postas pelos diálogos desta opção, nomeadamente:

-Qual o nome do ficheiro onde se pretende guardar o output.

-Se se trata dum ficheiro novo ou se pretende acrescentar este output a um ficheiro já existente.

-Se pretende gravar todo o resultado ou apenas uma página.

1.3.2.8. A Opção EXIT

Esta opção interrompe a sessão do PERFORM e retorna ao menu anterior (o menu FORM).

1.3.2.9. A Opção CURRENT

Esta opção permite visualizar a linha da tabela activa mais actual. Esta opção é útil num ambiente multiutilizador aonde as linhas da mesma tabela podem ser alteradas por utilizadores diferentes.

1.3.2.10. As Opções MASTER e DETAIL

Estas opções permitem tornar activas as tabelas da relação MASTER/DETAIL definida no capítulo 3.3.4.

1.4. O PERFORM Default

Uma das caracteristicas mais simpáticas do PERFORM é a que permite gerar automaticamente forms , associadas a uma ou mais tabelas.

Neste capitulo vamos gerar uma destas forms, associada à tabela ORDERS, visualizar e analisar as suas componentes.

1.4.1. Criação dum PERFORM Default.

Primeiro vamos posicionar-nos no menu FORM e seleccionar a opção GENERATE. --> Aparece o ecran CREATE FORM, para a introdução do nome da form a gerar.

Digite "orders" e faça RETURN --> Aparece o ecran CHOOSE TABLE com a lista das tabelas da base de dados STORES, para que possa seleccionar as tabelas que farão parte da form.

Seleccione a tabela ORDERS --> Aparece o menu CREATE FORM, que pede para indicar se pretende ou não seleccionar mais tabelas.

Como a form pretendida é só sobre a tabela orders, seleccione a opção TABLE-SELECTION-COMPLETE --> Cria e compila a form "orders" e regressa ao menu FORM.

Agora que a nossa form está criada vamos visualizá-la.

Seleccione a opção RUN e a form "orders" --> Aparece no ecran o desenho gerado automaticamente para a form "orders".

```
PERFORM:   Query  Next  Previous  Add  Update  Remove  Table  Screen...
Searches the active database table.              ** 1: orders table**
order_num          [           ]
order_date         [          ]
customer_num       [           ]
ship_instruct      [                                        ]
backlog            [ ]
po_num             [          ]
ship_date          [          ]
ship_weight        [          ]
ship_charge        [$        ]
paid_date          [          ]
```

Trabalhe um pouco sobre esta form, usando as opções do menu PERFORM que aprendeu no capítulo anterior, começando por visualizar as linhas da tabela ORDERS (opção QUERY).

1.4.2. Descrição dum PERFORM Default

Seleccione a opção MODIFY e a form "orders" --> Aparece um editor, neste caso o vi, com a seguinte listagem de instruções:

```
{ File: orders.per }
database stores
screen
{
order_num          [f000       ]
order_date         [f001      ]
customer_num       [f002       ]
ship_instruct      [f003                                        ]
backlog            [a]
po_num             [f004      ]
ship_date          [f005      ]
ship_weight        [f006      ]
ship_charge        [f007     ]
paid_date          [f008      ]
}
end
tables
orders
attributes
f000 = orders.order_num;
f001 = orders.order_date;
f002 = orders.customer_num;
f003 = orders.ship_instruct;
a = orders.backlog;
f004 = orders.po_num;
f005 = orders.ship_date;
f006 = orders.ship_weitght;
f007 = orders.ship_charge;
f008 = orders.paid_datae;
end
```

Analisando este conjunto de instruções destaca-se o seguinte:

- Existem 4 grupos distintos de instruções, correspondendo a cada um deles uma determinada secção da form.
- A primeira secção é constituída por uma única linha, que identifica a base de dados "database stores".
- A segunda secção contém o desenho do ecran, define o comprimento e identifica os campos constituintes do ecran. Começa pela linha "screen" e termina com a linha "end". As instruções que definem e desenham o ecran estão dentro de chavetas "{...}" e são constituidas por texto livre e variáveis, que são representadas por parentesis rectos "[...]" cujo espaçamento define o comprimento máximo da variável. Dentro destes parentesis está escrito o código que identifica cada uma das variáveis.

- A secção seguinte começa pela linha "tables", seguida pela lista das tabelas incluídas na form (neste caso a tabela ORDERS).
- A última secção começa na linha "attributes" e contém a ligação entre a variável do ecran (representada pelo seu código de identificação) e uma coluna da tabela ORDERS.

Estas secções são o esqueleto de qualquer form. Manipulando e inserindo instruções nestas secções e na secção instructions (a quinta e última secção, descrita no capítulo seguinte) conseguem-se criar e explorar as capacidades do PERFORM.

Como exercício tente alterar a apresentação e a colocação das variáveis e do texto livre da form gerada anteriormente. Para isso altere apenas as instruções da screen section, desenhando um ecran a seu gosto, mantendo os códigos e comprimentos máximos das variáveis. Utilize a opção MODIFY como se explica no capítulo 3.1.

1.5. O PERFORM e as suas Capacidades .

Neste capítulo vamos descrever a maior parte das instruções do PERFORM. Estas irão sendo introduzidas durante as várias fases do processo de desenvolvimento de um ecran complexo, com várias tabelas, validações, mensagens de erro etc. Seleccionaram-se as instruções normalmente utilizadas nos tipos de ecrans que fazem parte de qualquer aplicação.

O ecran que se vai construir começa por ser uma proposta genérica, que vai ser desenvolvida e aperfeiçoada por etapas. A ordem de apresentação destas é a que normalmente se utiliza na prática, salvo nalgumas situações para facilitar a exposição. A proposta de partida e' a seguinte:

- Pretende-se um único ecran para trabalhar com as tabelas ORDERS e ITEMS da base de dados STORES, nomeadamente para poder visualizar simultâneamente a encomenda (ORDERS) e as respectivas linhas de encomenda (ITEMS).
- Além das colunas destas tabelas pretende-se visualizar o preço unitário e nome do fabricante do artigo encomendado e o montante total da encomenda.
- Os diversos campos devem ser validados (tipo de dados correcto, valores admissiveis, etc.). Sempre que exista um erro deve ser assinalado com uma mensagem descrevendo o erro e o cursor deve posicionar-se no campo em questão.

1.5.1. Desenho do Ecran ORDER_ITEM

Como ponto de partida vamos gerar o PERFORM default ORDER_ITEM para as tabelas ORDERS, ITEMS, STOCK (por causa do preço unitário - coluna unit_price) e MANUFACT (por causa

do nome do fabricante - coluna manu_name), obtendo-se o esqueleto do programa com o comprimento e a ligação das colunas às tabelas já definidos. Crie o PERFORM default mencionado no parágrafo anterior. Visualize-o usando a opção MODIFY, obtendo o seguinte programa:

```
{ File: orders_item.per }
database stores
screen
{
order_num          [f000       ]
order_date         [f001      ]
customer_num       [f002       ]
ship_instruct      [f003                                    ]
backlog            [a]
po_num             [f004      ]
ship_date          [f005      ]
ship_weight        [f006      ]
ship_charge        [f007     ]
paid_date          [f008      ]
item_num           [f009  ]
order_num          [f010       ]
stock_num          [f011  ]
manu_code          [a0 ]
quantity           [f012  ]
total_price        [f013       ]
stock_num          [f014  ]
manu_code          [a1 ]
description        [f015           ]
unit_price         [f016     ]
unit               [f017]
unit_descr         [f018           ]
manu_code          [a2 ]
manu_name          [f019           ]
}
end
tables
orders
items
stock
manufact
attributes
f000 = orders.order_num;
f001 = orders.order_date;
f002 = orders.customer_num;
f003 = orders.ship_instruct;
a = orders.backlog;
f004 = orders.po_num;
f005 = orders.ship_date;
f006 = orders.ship_weight;
f007 = orders.ship_charge;
f008 = orders.paid_date;
f009 = items.item_num;
f010 = items.order_num;
f011 = items.stock_num;
a0 = items.manu_code;
f012 = items.quantity;
f013 = items.total_price;
f014 = stock.stock_num;
a1 = stock.manu_code;
f015 = stock.description;
```

```
f016 = stock.unit_price;
f017 = stock.unit;
f018 = stock.unit_descr;
a2 = manufact.manu_code;
f019 = manufact.manu_name;
end
```

1.5.2. Ecran (Screen Section)

A seguir vamos trabalhar sobre a Screen Section para obter o desenho do ecran pretendido, considerando que:

- Se pretende ter num só ecran todos os campos já citados.
- Deve acrescentar mais um campo: o montante total da encomenda, que não é coluna de nenhuma das tabelas da base de dados STORES. Aos campos com estas caracteristicas chamam-se "Display Only Fields" e veremos como se definem quando abordarmos a Attributes Section.
- Deve apagar todos os restantes campos das tabelas STOCK e MANUFACT.

Uma solução para o ecran pretendido poderá ser a que se apresenta na screen section seguinte:

```
screen
{
                                ENCOMENDA
=============================================================================
order_num:         [f000       ]        customer_num:      [f002       ]
order_date:        [f001      ] backlog: [a]  po_num:       [f004      ]
ship_instruct:     [f003                                     ]
ship_date:         [f005      ]          ship_weight:       [f006      ]
ship_charge:       [f007     ]
paid_date:         [f008      ]                total:       [d1        ]

                           LINHAS DE ENCOMENDA
=============================================================================
item_num:    [f009  ]    stock_num: [f011  ]      manu_code: [a0 ]
manu_name: [f019          ]   quantity:[f012  ]  unit_price [f016     ]
total_price        [f013       ]
}

end
```

1.5.3. Especificação das Tabelas (Tables Section)

As tabelas intervenientes são indicadas na tables section do programa.

No nosso caso não é necessário alterar a tables section da form default criada anteriormente, que será:

```
tables orders items stock manufact
```

1.5.4. Definição de Atributos (Attributes Section)

Continuando o nosso exercício vamos agora desenvolver a Attributes Section. Esta define o comportamento e o aspecto de cada um dos campos do ecran definidos na Screen Section. Qualquer campo da Screen Section deve ser descrito na Attributes Section. Na Attributes Section define-se:

- Como o PERFORM apresenta o campo no ecran.
- Um valor default para o campo do ecran.
- Limites para os valores a entrar no campo do ecran.
- A ordem do percurso do cursor pelos campos do ecran, dada pela sequência de apresentação dos campos do ecran na Attributes Section.
- A ordem de activação das tabelas, determinada pela ordem de apresentação das respectivas colunas na Attributes Section.
- Dois tipos de ligações ao campo do ecran: a colunas de tabelas da base de dados e a display only fields.

Vamos iniciar o desenvolvimento da Attributes Section a partir da form default. Assim vamos:

- apagar as linhas referentes às colunas da tabela STOCK excepto a coluna "unit_price".
- apagar a linha referente à coluna "manu_code" da tabela MANUFACT.
- Inserir as linhas correspondentes aos "display only fields".

1.5.4.1. Display Only Fields

Os Display Only Fields não estão associados a colunas de bases de dados e existem apenas no ecran. Os seus valores resultam de cálculos e/ou de decisões lógicas com base nos outros campos.

Existem dois tipos de Display Only Fields; os que admitem e os que não admitem dados de entrada. Nos exemplos seguintes temos a definição de cada um destes tipos de Display Only Fields:

```
d1 = displayonly type money;
b = displayonly allowing input type char;
```

Nesta instrução de definição existem três tipos de palavras chave:

DISPLAYONLY Estabelece que o campo é um Display Only Field.

TYPE É obrigatória e precede a definição do tipo de dados.

ALLOWING INPUT Indica que o campo admite entrada de dados.

O primeiro exemplo define o campo "total" da form ORDER_ITEM. Vejamos a nossa Attributes Section depois destas alterações:

```
attributes
f000 = orders.order_num;
f001 = orders.order_date;
f002 = orders.customer_num;
f003 = orders.ship_instruct;
a = orders.backlog;
f004 = orders.po_num;
f005 = orders.ship_date;
f006 = orders.ship_weight;
f007 = orders.ship_charge;
f008 = orders.paid_date;
d1 = displayonly type money;
f009 = items.item_num;
f010 = items.order_num;
f011 = items.stock_num;
a0 = items.manu_code;
f012 = items.quantity;
f013 = items.total_price;
f016 = stock.unit_price;
f019 = manufact.manu_name;
end
```

A seguir ordene estas instruções pela ordem que pretende que o cursor percorra os respectivos campos, compile a form order_item e corrija os eventuais erros.

1.5.4.2. Join de Tabelas no PERFORM

Nos ecrans com várias tabelas existe sempre a necessidade de as relacionar. Isso faz-se com o Join (equivalente a uma instrução SELECT com várias tabelas na cláusula FROM - veja o capitulo 2.5.6).

No PERFORM os Joins definem-se na Attributes Section (no caso de joins compostos também é necessário utilizar a Instructions Section, como veremos), ligando ao código de identificação do campo do ecran, as colunas das tabelas que usadas pelo join.

Por exemplo as tabelas ORDERS e ITEMS estão relacionadas pela coluna comum "order_num". Como existem no ecran dois campos "order_num" com códigos de identificação diferentes (f000 e f010) vamos suprimir um deles (o da tabela ITEMS) e relacioná-los para o join, escrevendo:

```
f000  = orders.order_num;
      = items.order_num;
```

NOTA: Não é possivel existirem num ecran dois campos diferentes associados à mesma coluna. Daí a necessidade de suprimir um dos campos "order_num" (neste caso o f010).

1.5.4.3. O Verify Join

Uma situação muito frequente é a necessidade de verificar, quando da inserção duma nova linha numa tabela, se o valor a introduzir já existe ou não noutra tabela. O PERFORM resolve este problema com o Verify Join.

Por exemplo quando se inserem novas linhas na tabela ITEMS a coluna "order_num" tem de existir na coluna, com o mesmo nome, da tabela ORDERS. O Verify Join permite testar esta situação e se o valor não existir previamente, interrompe a função ADD e dá a mensagem correspondente.

O Verify Join define-se colocando um asterisco "*" antes da tabela.coluna dominante, ou seja aquela onde residem os valores que confirmam a verificação. Assim:

```
f000  = *orders.order_num;
      = items.order_num;
```

NOTA: O facto de se definir um Join na Attributes Section numa ou em duas linhas pode ser importante. Quando se define numa única linha os atributos (palavras chave que definem o comportamento e o aspecto do campo do ecran) são escritos a seguir `a ultima coluna de Join e referem-se às colunas de ambas as tabelas. No caso de se escrever a instrução em duas linhas pode escrever atributos diferentes para cada coluna de join imediatamente a seguir a esta.

Como exercício tente identificar os vários Joins existentes no ecran proposto, definir quais são Verify Joins, altere a Attributes Section e compile a form. Como exemplo poder-se-á definir os seguintes joins, todos eles do tipo verify:

```
f000  = *orders.order_num;
      = items.order_num;
```

```
f002  = orders.customer_num;
      = *customer.customer_num;
```

Este join implica a inserção na Tables Section da tabela CUSTOMERS.

```
f011  = items.stock_num;
      = stock.stock_num;
a0    = items.manu_code;
      = stock.manu_code;
```

Estas duas ultimas instruções estão associadas pois representam um join composto sobre a chave primária da tabela STOCK (veja o apêndice 1.2). A definição do join composto será completada na Instructions Section com a instrução COMPOSITES, no capítulo 3.4.

1.5.4.4. Os Atributos

A seguir iremos descrever todos os atributos disponiveis e aplicar, na form ORDER_ITEM, os de uso mais comum.

1.5.4.4.1. AUTONEXT

Este atributo faz com que o cursor, nas opções ADD e UPDATE, salte para o campo seguinte quando o campo actual estiver totalmente preenchido e se introduzir o caracter seguinte. É muito útil e na form ORDER_ITEM iremos usá-lo nas datas e nos campos só com um caracter. Exemplo:

```
f001 = orders.order_date, autonext;
```

1.5.4.4.2. COMMENTS

Este atributo emite a mensagem que pretender sempre que o campo a que está associado estiver a ser processado. Na form ORDER_ITEM iremos aplicá-lo ao campo "a" com a seguinte mensagem:

VALORES ADMISSIVEIS: n ou y.

Exemplo:

```
a= orders.backlog, autonext,comments = "VALORES ADMISSIVEIS: n ou y";
```

1.5.4.4.3. DEFAULT

Insere um valor inicial no campo do ecran a que está associado, quando se selecciona a opção ADD.

Na form ORDER_ITEM iremos aplicá-lo aos campos "a", "b" e "f001". Neste último campo (a data da encomenda) vai-se inserir a data do dia, escrevendo "today" como valor inicial. Exemplo:

```
f001 = orders.order_date, autonext, default = today;
```

1.5.4.4.4. FORMAT

Permite controlar o formato de display dos campos a que está associado, desde que tenham tipo de dados DECIMAL, FLOAT, SMALLFLOAT ou DATE. Os caracteres de controlo do formato são os seguintes:

- **###.##** Para campos numéricos. Indica o número de campos à esquerda e à direita do ponto decimal.
- **mm** Para datas. Representação do mês através de dois dígitos.
- **mmm** Para datas. Representação do nome do mês através de abreviatura com três letras.
- **dd** Para datas. Representação do dia através de dois dígitos.
- **yy** Para datas. Representação do ano através de dois dígitos.
- **yyyy** Para datas. Representação do ano através de quatro dígitos.
- As datas podem ter os separadores "-" ou "/" à escolha.

Na form ORDER_ITEM iremos usar o atributo "format" apenas nas datas e com o formato "yyyy/mm/dd". Exemplo:

```
f001 = orders.order_date, autonext, default = today,
format = "yyyy/mm/dd";
```

1.5.4.4.5. INCLUDE

Permite especificar valores ou gamas de valores admissiveis a introduzir no campo associado a este atributo. .P Pode especificar uma série de valores separados por virgulas ou um valor mínimo e um valor máximo separados pela palavra "to".

Na form ORDER_ITEM vamos usar o atribute include nos campos:

"a" : Toma os valores "n" ou "y" ("n", "y").

f009" : Admitindo que uma encomenda não pode ter mais do que 100 linhas de encomenda (0 to 100). "

Exemplo:

f009 = items.item_num, include = (0 to 100);

1.5.4.4.6. LOOKUP

Este atributo permite executar um join entre a tabela associada ao código de identificação, que inicia a instrução, e uma segunda tabela. Este tem as mesmas potencialidades do Join já estudado, podendo-se inclusive definir um Verify Join. No entanto tem a particularidade de colocar num outro campo do ecran, definido para esse efeito, qualquer outra coluna da segunda tabela, diferente da coluna de join.

Como a descrição deste atributo é complicada, vamos mostrar primeiro um exemplo que será comentado posteriormente. Exemplo:

```
a0 = items.manu_code, autonext,
    lookup f019 = manufact.manu_name
    joining *manufact.manu_code;
```

DESCRIÇÃO:

- O join é definido pela palavra "joining" que relaciona a tabela e a coluna associada ao código do campo que inicia a instrução, com a segunda tabela e a respectiva coluna de join. (a0 = items.manu_code, ... , joining *manufact.manu_code;).
- O asterisco indica que se trata duma tabela dominante dum verify join.
- A palavra "lookup" define qual a coluna da segunda tabela e o campo do ecran onde a primeira deve ser colocada. (..., lookup f019 = manufact.manu_name, ...).
- O campo f019 não admite entrada de dados através do ecran.

1.5.4.4.7. NOENTRY

Permite evitar a inserção de dados na tabela e coluna associadas a este campo do ecran, quando da execução da opção ADD.

Embora a form ORDER_ITEM contenha campos associados `a tabela STOCK, estes são apenas utilizados para mostrar características dos artigos encomendados e não para inserir novas linhas na tabela STOCK. Para conseguir esse efeito será necessário utilizar o atributo "noentry" em todos os campos associados a colunas da tabela STOCK. Exemplo:

```
f016 = stock.unit_price, noentry;
```

1.5.4.4.8. NOUPDATE

Permite evitar a alteração de dados na tabela e coluna associadas a este campo do ecran, quando da execução da opção UPDATE.

Na form ORDER_ITEM vamos impor que não se pode alterar as chaves primárias das linhas das tabelas, pois isso implicaria a perda da integridade existencial. Em termos práticos essa alteração seria equivalente a um REMOVE seguido de um ADD. Assim vamos usar o atributo "noupdate" nas colunas que são chaves primárias da tabela ITEMS.

Como a coluna chave primária da tabela ORDERS (order_num) tem tipo de dados SERIAL é desnecessário usar o atributo "noupdate".

Por motivos idênticos aos considerados no atributo "noentry", não se pode permitir a alteração de linhas da tabela STOCK. Assim será necessário utilizar o atributo "noupdate" em todos os campos associados a colunas da tabela STOCK. Exemplo:

```
f000  = *orders.order_num;
      = items.order_num, noupdate;
.
.
.
f009 = items.item_num, include = (0 to 100), noupdate;
```

1.5.4.4.9. QUERYCLEAR

Já deve ter reparado que quando selecciona a opção QUERY pela segunda vez consecutiva, por exemplo para a tabela ORDERS, todos os campos associados às colunas dessa tabela ficam a espaços, excepto os campos "order_num" e "costumer_num", ou seja os campos que estão associados a colunas de join.

Usando o atributo "queryclear" esses campos também ficam a espaços quando seleccionar a opção QUERY. Exemplo:

```
f000 = *orders.order_num,queryclear;
     = items.order_num, noupdate;

f002 = orders.customer_num,queryclear;
     = *customer.customer_num;
```

NOTA: Quando a tabela ORDERS está activa e se selecciona a opção QUERY os campos "order_num" e "customer_num" ficam a espaços. Quando é a tabela ITEMS que está activa o campo "order_num" mantem o valor anterior.

1.5.4.4.10. REQUIRED

Este atributo torna a inserção obrigatória na coluna a que for associado, durante a execução da opção ADD.

Na form ORDER_ITEM vamos tornar a inserção obrigatória nas seguintes colunas: item.order_num, item.item_num, item.stock_nume item.manu_code. Exemplo:

```
f000 = *orders.order_num, queryclear;
     = items.order_num,noupdate,required;
```

1.5.4.4.11. REVERSE

Este atributo mostra os campos, a que está associado, em reverse video.

Na form ORDER_ITEM vamos usar em reverse video os campos "order_num" e "item_num". Exemplo:

```
f000 = *orders.order_num,queryclear,reverse;
     = items.order_num,noupdate,required;
```

1.5.4.4.12. PICTURE

Permite definir a forma dum campo do ecran, com tipo de dados alfanumérico. A forma do campo define-se, no atributo "picture", combinando com qualquer outro caracter os seguintes caracteres:

A - Representa qualquer letra.

- Representa qualquer dígito.

X - Representa qualquer caracter.

Exemplo:

```
f004=orders.po_num,picture="A-XXXXX-##";
```

NOTA: Antes de introduzir dados no campo "po_num" este tem a forma seguinte: [- -].

1.5.4.4.13. RIGHT

Encosta à direita os dados presentes no campo do ecran associado a este atributo. Exemplo:

```
f006 = orders.ship_weight, right;
```

1.5.4.4.14. ZEROFILL

Encosta um campo numérico à direita e completa-o com zeros à esquerda. Exemplo:

```
f009 = item.item_num, zerofill;
```

1.5.4.4.15. VERIFY

Este atributo usa-se quando se pretende introduzir, num determinado campo, o mesmo valor duas vezes para reduzir a probabilidade de erro na entrada dos dados. Exemplo:

```
f000 = orders.order_num, verify;
```

1.5.4.4.16. DOWNSHIFT

Transforma as letras maiúsculas, dum campo alfanumérico, em letras minúsculas. Exemplo:

```
f003 = orders.ship_instruct,downshift;
```

1.5.4.4.17. UPSHIFT

Transforma as letras minúsculas, dum campo alfanumérico, em letras maiúsculas. Exemplo:

```
f003 = orders.ship_instruct, upshift;
```

1.5.4.4.18. A Attributes Section da form ORDER_ITEM

Apresenta-se a seguir a Attributes Section do ecran proposto como o objectivo de alcançar o que foi sendo descrito ao longo dos últimos capítulos.

```
attributes
f000 = *orders.order_num,queryclear,reverse;
      = items.order_num, noupdate, required;
f002 = orders.customer_num, queryclear;
      = *customer.customer_num;f001
      = orders.order_date, autonext,
            default = today,format = "yyyy/mm/dd";
a = orders.backlog, autonext, default = "n",
       include = ("n", "y"),comments = "VALORES ADMISSIVEIS:  n  ou  y";
f004 = orders.po_num;
f003 = orders.ship_instruct;
f005 = orders.ship_date, autonext, format = "yyyy/mm/dd";
f006 = orders.ship_weight;
f007 = orders.ship_charge;
f008 = orders.paid_date, autonext, format = "yyyy/mm/dd";
d1 = displayonly type money;
f009 = items.item_num, include=(0 to 100), noupdate, required, reverse;
f011 = items.stock_num, queryclear, required;
```

```
      = stock.stock_num, noentry, noupdate, queryclear;
f012 = items.quantity ;

f013 = items.total_price;
f016 = stock.unit_price, noentry, noupdate, queryclear;
end .
```

1.5.5. A Instruction Section

A Instruction Section é a última secção da form e é opcional. Nesta secção poderemos executar as seguintes funções:

- Definir joins compostos.
- Substituir os caracteres de enquadramento dos campos do ecran.
- Criar relações MASTER/DETAIL.
- Definir blocos de controlo.

1.5.5.1. Joins Compostos - COMPOSITES

Quando decrevemos, no capitulo anterior, a forma de definir um join referiu-se que para um join composto era necessário completar essa definição na Instructions Section.

Com o comando "composites" define-se quais são as colunas de join da primeira e segunda tabelas e também quais as colunas que se relacionam entre tabelas.

Na form ORDER_ITEM temos o caso da chave primária da tabela STOCK que é composta pelas colunas "stock_num" e "manu_code". Esta chave é usada para obter o preço unitário do artigo especificado na linha de encomenda, através dum join entre as tabelas ITEMS e STOCKS. Este join é simultâneamente um verify join para validar a existência prévia do artigo encomendado. Isto consegue-se usando o conjunto de instruções da Attributes e Instructions Sections que constituem o exemplo seguinte:

```
f011 = items.stock_num, queryclear, required;
      = stock.stock_num, noentry, noupdate, queryclear;
a0 = items.manu_code, autonext, queryclear, required,
       lookup f019 = manufact.manu_name joining *manufact.manu_code;
      = stock.manu_code, noentry, noupdate, queryclear ;
.
instructions
composites <items.stock_num, items.manu_code>
         * <stock.stock_num, stock.manu_code>;
.
end
```

NOTA: As colunas de join de cada tabela estão incluídas entre os caracteres "<...>" e pela ordem de relacionamento entre tabelas. O asterisco "*" define qual é tabela dominante do verify join.

1.5.5.2. Alteração dos Caracteres de Enquadramento - DELIMITERS

Com o comando "delimiters" pode alterar os caracteres de enquadramento do campo do ecran, quando este é visualizado através da execução da form pelo PERFORM. Os caracteres default são os parêntesis rectos "[...]". Exemplo:

```
delimiters "ab"
```

em que "a" é o caracter de abertura e "b" o caracter de fecho.

NOTA: Pode seleccionar como caracteres de enquadramento o espaço " ", no entanto não e' aconselhável utilizá-lo nos ecrans com várias tabelas, pois torna-se dificil ver quais os campos pertencentes à tabela activa.

1.5.5.3. Relações Master/Detail - MASTER OF

A relação Master/Detail existe entre duas tabelas quando uma linha da primeira tabela (Master) pode estar relacionada com várias linhas da segunda tabela (Detail).

Quando definir esta relação através do comando "master of" pode utilizar as opções DETAIL e MASTER do menu do PERFORM. Se seleccionar a opção DETAIL o PERFORM torna activa a segunda tabela, ficando disponíveis todas as linhas relacionadas com a linha presente na primeira tabela. Se seleccionar a opção MASTER o PERFORM torna activa a primeira tabela, mostrando a linha relacionada com a linha existente na segunda tabela.

Repare que a definição da relação Master/Detail implica que sejam definidos os joins que suportam estas relações.

Na form ORDER_ITEM existem duas relações Master/Detail, definidas assim:

```
customer master of orders; orders master of items;
```

1.5.5.4. Blocos de Controlo

Os blocos de controlo permitem executar as seguintes operações:

- Controlar o movimento do cursor quando se insere ou altera uma linha de uma tabela.
- Validar dados de entrada dependentes de outros dados já existentes.

- Modificar os dados nos campos do ecran depois de terem sido executadas as operações de ADD, UPDATE ou QUERY.
- Mostrar nos campos do ecran informação agregada, por exemplo médias e totais de colunas incluídas em resultados de queries, modificados ou não pelas opções do PERFORM.

Os blocos de controlo são constituidos por uma série de acções, que serão executadas antes ou depois das operações do PERFORM terem sido concluídas, sendo designados por blocos BEFORE ou AFTER respectivamente.

Os blocos "before" podem ser utilizados com as opções ADD, UPDATE e REMOVE. .P Os blocos "after" podem ser utilizados com as opções ADD, UPDATE, QUERY e REMOVE.

1.5.5.4.1. Bloco de Controlo BEFORE

O bloco de controlo BEFORE permite executar uma série de acções antes do PERFORM executar uma operação. A estrutura do bloco de controlo BEFORE é a seguinte:

```
BEFORE lista de opções OF lista de tabelas/colunas acção
      acção
      acção
      acção
      acção
         .
      acção
```

Em que:

- BEFORE identifica o tipo de bloco de controlo.
- Opção é uma palavra chave que identifica as operações relacionadas com as acções do bloco de controlo e em que condições estas são executadas. As opções admissíveis neste tipo de bloco são: EDITADD, EDITUPDATE e REMOVE.
- A "lista de tabelas/colunas" Identifica as tabelas e respectivas colunas objecto do bloco de controlo BEFORE.
- A "acção" identifica uma das cinco instruções, que serão definidas posteriormente, e que permitem assignar valores a campos do ecran, mover o cursor, escrever mensagens, voltar ao menu PERFORM e escolher uma acção ou grupo de acções dependendo serem ou não satisfeitas condições sobre valores do campo do ecran.

1.5.5.4.2. Bloco de Controlo AFTER

O bloco de controlo AFTER permite executar uma série de acções depois do PERFORM executar uma operação.

A estrutura do bloco de controlo AFTER é semelhante à do bloco BEFORE:

```
AFTER lista de opções OF lista de tabelas/colunas
      acção
      acção
      acção
      acção
      acção
         .
      acção
```

Em que:

- AFTER identifica o tipo de bloco de controlo.
- As opções admissíveis neste tipo de bloco são: ADD, UPDATE, QUERY, REMOVE, DISPLAY, EDITADD, e EDITUPDATE.
- A "lista de tabelas/colunas Identifica as tabelas e respectivas colunas objecto do bloco de controlo AFTER.
- As acções são idênticas às do bloco BEFORE.

1.5.5.4.3. As Opções EDITADD e EDITUPDATE

EDITADD e EDITTUPDATE permitem executar uma série de acções antes ou depois da entrada de dados num campo durante uma uma operação de ADD ou UPDATE respectivamente. As acções executam-se antes da linha ser gravada na tabela.

NOTAS: Se a lista de tabelas/colunas contém apenas colunas e se se tratar dum bloco BEFORE, o PERFORM executa as acções quando o cursor se move para o campo correspondente e antes de entrar os dados. Se a lista de tabelas/colunas contém apenas colunas e se se tratar dum bloco AFTER, o PERFORM executa as acções quando da entrada dos dados e depois de pressionar a tecla RETURN. Antes da execução das acções o PERFORM faz todas as validações definidas na Attributes Section (INCLUDE,REQUIRED, etc). Quando na lista de tabelas/colunas se especifica o nome duma tabela num bloco BEFORE o PERFORM executa as acções antes da entrada de quaisquer dados na form. Isto permite entrar dados default e mensagens conforme a tabela activa. Quando na lista de tabelas/colunas se especifica o nome duma tabela num bloco AFTER, o PERFORM executa as acções depois da entrada dos dados e de pressionar a tecla ESC, para iniciar a transacção, mas antes da linha ser gravada na tabela.

1.5.5.4.4. A Opção ADD

A palavra chave ADD indica que as acções serão executadas depois da operação ADD e depois da linha ter sido gravada na tabela.

NOTA: Só admite lista de tabelas e só pode ser usada com blocos AFTER.

1.5.5.4.5. A Opção UPDATE

A palavra chave UPDATE indica que as acções serão executadas depois da operação UPDATE e depois da linha ter sido gravada na tabela.

NOTA: Só admite lista de tabelas e só pode ser usada com blocos AFTER.

1.5.5.4.6. A Opção QUERY

A palavra chave QUERY indica que as acções serão executadas depois da operação QUERY.

NOTA: Só admite lista de tabelas e só pode ser usada com blocos AFTER.

1.5.5.4.7. A Opção REMOVE

A palavra chave REMOVE indica que as acções serão executadas antes (só nas versões mais recentes) e depois da operação REMOVE.

NOTA: Só admite lista de tabelas.

1.5.5.4.8. A Opção DISPLAY

A palavra chave DISPLAY indica que as acções serão executadas depois de qualquer operação do PERFORM que implique o display de dados no ecran.

NOTA: Só admite lista de tabelas.

1.5.5.4.9. A Acção ABORT

Esta acção permite interromper a actual operação de ADD, UPDATE ou REMOVE sem alterar a base de dados e voltar ao menu PERFORM.

Só se pode utilizar com as opções ADD, UPDATE ou REMOVE e não está disponível nas versões mais antigas do INFORMIX-SQL. Exemplo:

```
after editadd editupdate of items.quantity
      if f012 < 1 or f012 is null then
      begin
            comments bell " CAMPO DE PRENCHIMENTO OBRIGATORIO"
            abort
      end
```

1.5.5.4.10. A Acção LET

Esta acção permite assignar um valor a um campo do ecran. A sua forma é a seguinte:

```
LET código_campo  =  expressão
```

em que:

- código_campo é o código de identificação do campo da SCREEN SECTION da form.
- "expressão" pode ser:
  1. Outro código de identificação do campo do ecran.
  2. Uma constante.
  3. Qualquer das seguintes funções agregadas: COUNT, TOTAL, AVERAGE, MAX e MIN. Estas funções agregadas são as mesmas que estudámos para a instrução SELECT do RDSQL. Define-se qual é a coluna a que se aplica a função agregada escrevendo a seguir a esta o código de identificação associado `a coluna.
  4. A palavra chave TODAY que associa ao campo a data actual.
  5. Qualquer combinação dos componentes de "expressão" anteriores com os operadores aritméticos "+", "-", "*" e "/". Pode usar parentesis para explicitar a precedencia de execução dos operadores.

Exemplo:

```
after add update query of items
      let d1 = total of f013
```

1.5.5.4.11. A Acção NEXTFIELD

Usa-se conjuntamente com as opções EDITADD e EDITUPDATE e controla o movimento do cursor, permitindo colocá-lo no campo que pretender.

NOTAS:

- Pode usar a palavra chave opcional EXITNOW a seguir ao código de identificação da coluna, tendo o mesmo efeito de pressionar a tecla ESC, ou seja executa a opção seleccionada (ADD, UPDATE ou REMOVE) e retorna ao menu PERFORM.
- O movimento do cursor só se efectiva depois de executadas as opções EDITADD e EDITUPDATE.

Exemplo:

```
after editadd editupdate of items.quantity
      if f012 < 1 or f012 is null then
      begin
            comments bell " CAMPO DE PRENCHIMENTO OBRIGATORIO"
            nextfield = f012
      end
```

1.5.5.4.12. A Acção COMMENTS

Permite escrever uma mensagem na linha de status do ecran. A mensagem tem de ser escrita entre plicas(").

Pode usar as palavras chaves opcionais BELL e/ou REVERSE, antes da mensagem, para actuar o sinal sonoro e/ou apresentar a mensagem em reverse video, respectivamente.

Como nas opções ADD, UPDATE, QUERY ou REMOVE aparece uma mensagem do sistema na linha de status só faz sentido usar a acção COMMENTS com EDITADD e EDITUPDATE.

Repare que o atributo COMMENTS (da ATTRIBUTES SECTION) escreve a mensagem na linha de comentário. Exemplo:

```
after editadd editupdate of items.quantity
      if f012 < 1 or f012 is null  then
      begin
            comments bell " CAMPO DE PRENCHIMENTO OBRIGATORIO"
            nextfield = f012
      end
```

1.5.5.4.13. A Acção IF-THEN-ELSE

Permite executar acções condicionadas pelos valores dos campos do ecran. Esta acção tem a seguinte forma:

```
IF expressão lógica THEN acção ELSE acção
```

em que:

- "expressão lógica" pode ser uma combinação de comparações (=, <>, <, >, <=, >=) e operações lógicas (AND,OR e NOT). Pode ainda usar os operadores IS NULL, IS NOT NULL e MATCHES, neste caso só para campos alfanuméricos.
- Se especificar várias acções consecutivas deverá colocá-las entre as palavras chave BEGIN e END.

Exemplo:

```
after editadd editupdate of items.quantity
      if f012 < 1 or f012 is null then
      begin
            comments bell " CAMPO DE PRENCHIMENTO OBRIGATORIO"
            nextfield = f012
      end
```

1.5.5.4.14. Conclusão da Form ORDER_ITEM.

Neste capítulo vamos concluir a form ORDER_ITEM com as instruções da Instruction Section.

As instruções referentes aos joins compostos e à definição das relações MASTER/DETAIL jà foram explicadas. Falta apenas introduzir os blocos de controlo.

Na form ORDER_ITEM vamos usar os seguintes blocos de controlo.

1. Um bloco para validar a coluna "quantity" da tabela ITEMS, durante as opções ADD e UPDATE. Usou-se o bloco de controlo AFTER EDITADD EDITUPDATE para permitir que os valores inválidos fossem detectados antes da linha ser gravada na tabela. Usou-se ainda a acção NEXTFIELD para colocar o cursor no campo com o valor incorrecto.
2. Para calcular o valor da coluna "total_price" da tabela ITEMS também se usou um bloco de controlo AFTER EDITADD EDITUPDATE porque esse valor necessita de ser calculado antes da gravação.
3. Usou-se um bloco de controlo AFTER ADD UPDATE QUERY para colocar no campo DISPLAY ONLY "total" a soma dos totais de cada linha de encomenda. Neste caso só depois de executadas as operações é que os valores para o cálculo do total estarão disponíveis.
4. Finalmente usou-se um bloco de controlo AFTER DISPLAY para apagar o campo DISPLAY ONLY "total" e assim eliminar o último valor calculado, sempre que a tabela ORDERS seja activada, evitando-se confusões.

Apresenta-se a seguir a INSTRUCTIONS SECTIONS da form ORDER_ITEM:

```
instructions
composites <items.stock_num, items.manu_code>
      *<stock.stock_num, stock.manu_code>;

customer master of orders;
orders master of items;

after editadd editupdate of items.quantity
      if f012 < 1 or f012 is null  then
      begin
            comments bell " CAMPO DE PREENCHIMENTO OBRIGATORIO"
            nextfield = f012
      end
after editadd editupdate of items
      let f013 = f016 * f012
after add update query of items
      let d1 = total of f013
after display of orders
      let d1 = "   "
end
```

1.5.5.4.15. Uso de Forms no Sistema Operativo

É possivel consultar, inserir, alterar dados e criar forms sem entrar nos menus do INFORMIX-SQL.

1.5.6. O comando SPERFORM

Para processar uma form através do sistema operativo basta escrever o comando SPERFORM seguido do nome da form.

Por exemplo se quiser processar a form ORDER_ITEM basta digitar na linha de comando do sistema operativo:

```
1% sperform order_item
```

e pressionar RETURN para lhe aparecer o ecran já conhecido da form ORDER_ITEM e o menu PERFORM.

1.5.7. O Comando SFORMBLD

O processo de criação duma form através do sistema operativo pode ser constituido pelas mesmas etapas que usámos no desenvolvimento da form ORDER_ITEM, ou seja a criação duma form default, através do comando SFORMBLD, e posterior alteração através dum editor de texto.

O comando SFORMBLD tem a estrutura dum comando UNIX, com três opções disponíveis.

1. **-s** Esta opção compila a form fonte existente no ficheiro cujo nome segue a opção no comando. Note que o verdadeiro nome do ficheiro é constituído pelo nome existente no comando seguido pela extensão ".per". No caso de existirem incorrecções, cria um ficheiro com o mesmo nome da form fonte mas com extensão ".err", e que contém as mensagens que indicam os pontos onde se verificam as incorrecções. Pode-se editar este ficheiro e corrigir os erros e, depois de se apagarem as mensagens de erro, gravá-lo no ficheiro com extensão ".per" e compilá-lo de novo.
2. **-v** Esta opção também compila a form fonte mas verifica se o comprimento dos campos definidos na SCREEN SECTION têm o mesmo comprimento das colunas a que são associados na ATTRIBUTES SECTION, e apresenta as respectivas mensagens no ficheiro de erros.
3. **-d** Esta opção cria uma form default tal como a opcão GENERATE do menu PERFORM. Pode especificar o nome da form, a base de dados e as tabelas que compoêm a form

escrevendo-as por esta ordem e a seguir à opção -d do comando. No entanto se escrever apenas o comando seguido da opção -d e pressionar RETURN o comando pedir-lhe-á que introduza esses elementos e aceita-os sempre que pressionar RETURN. Para indicar que não pretende especificar mais tabelas pressione RETURN pela segunda vez.

Exemplos:

```
sformbld -d ref stores orders items
```

Cria e compila a form REF com as tabelas ORDERS e ITEMS da base de dados STORES.

```
sformbld -v ref
```

Compila a form REF.

2. ACE - Geração de mapas

O INFORMIX-SQL permite que se desenhem, compilem e executem mapas sobre os dados existentes na base de dados através dos programas ACEPREP e ACEGO respectivamente.

O ACE permite:

- Seleccionar na base de dados a informação necessária para o mapa utilizando uma ou várias instruções SELECT.
- Desenhar o mapa especificando a localização dos campos, definindo cabeçalhos, footings, dimensões da página e o suporte externo do mapa.
- Combinar os dados elementares da base de dados (as colunas das tabelas), nomeadamente através de operações algébricas, e apresentá-los em campos específicos do mapa.
- Definir parâmetros e variáveis que serão especificados antes ou durante a execução do mapa.

As especificações do mapa são escritas num ficheiro, por exemplo através dum editor de texto, e depois compiladas pelo programa ACEPREP. A versão compilada pode então ser executada sempre que necessário pelo programa ACEGO. Assim o mapa tem como suporte físico dois ficheiros: o programa fonte com a extensão ".ace" e o programa compilado com a extensão ".arc".

2.1. Os menus Associados ao ACE

Como introdução ao ACE vamos primeiro descrever as suas funções básicas, apresentando e explicando os menus associados ao ACE.

2.1.1. O menu REPORT

Entre no menu INFORMIX-SQL e escolha a opção Report --> entrou no menu REPORT. O menu Report apresenta as seguintes 7 opções:

RUN Esta opção executa um mapa já compilado. Quando a escolher aparecerá um ecran com a lista dos mapas existentes. Pode seleccionar o mapa que pretender iniciando assim a sua execução pelo ACE.

MODIFY Esta opção permite fazer alterações a mapas já existentes. Tal como na opção RUN aparece um ecran com a lista dos mapas existentes para seleccionar um deles. Ao escolher um aparece-lhe o desenho do mapa,num editor do sistema, pronto a ser alterado. Depois de concluidas as alterações e de ter saído do editor, aparece o menu MODIFY REPORT com as seguintes opções:

COMPILE: Compila o mapa. Se o mapa tiver erros estes são registados num ficheiro juntamente com as especificações do mapa, podendo então corrigi-lo e compilá-lo de novo. Estes procedimentos podem ser executados ciclicamente até obter uma compilação do mapa sem erros.

SAVE-AND-EXIT: Grava as alterações ao mapa no ficheiro fonte, sem executar compilação.

DISCARD-AND-EXIT: Não grava as alterações ao mapa no ficheiro fonte, mantendo neste a versão anterior ao inicio das alterações. Também não executa compilação.

GENERATE Cria automàticamente um mapa com um formato por defeito (default). Quando esta opção é seleccionada aparecem ecrans para selecção da base de dados, se ainda não foi seleccionada, e para escolha das tabelas que contêem os dados necessários ao mapa. Veremos mais em pormenor esta opção na secção seguinte "Criação e Compilação dum report Default".

NEW Permite criar um mapa a partir dum ficheiro vazio e através dum editor. Quando terminar a edição do mapa o INFORMIX-SQL comportar-se-á de forma igual à da opção MODIFY.

COMPILE Compila um mapa já existente. Apresenta um ecran para seleccionar o mapa a compilar. Se a compilação falhar comportar-se-á de forma igual à da opção MODIFY.

DROP Remove um mapa da base de dados, tanto a versão fonte como a versão objecto (compilada). Apresenta um ecran para escolher o mapa a suprimir e pede confirmação para fazê-lo.

EXIT Retorna ao menu INFORMIX-SQL.

2.2. O Mapa Default

Uma das caracteristicas do ACE é a que permite gerar automàticamente mapas associados a uma tabela.

Neste capitulo vamos gerar um destes mapas, associado à tabela ORDERS, e visualizar e analisar as suas componentes.

2.2.1. Criacao dum Mapa Default

Primeiro vamos posicionar-nos no menu REPORT e seleccionar a opção GENERATE. --> Aparece o ecran CREATE REPORT, para a introdução do nome do mapa a gerar.

Digite "ordcust" e faça RETURN. --> Aparece o ecran CHOOSE TABLE com a lista das tabelas da base de dados STORES, para que possa seleccionar as tabelas que o mapa necessita.

Seleccione a tabela ORDERS. --> Reaparece o menu REPORT.

Agora que o nosso report está criado vamos

visualizá-lo.

Seleccione a opção RUN e o mapa "ordcust". --> Aparece no ecran o mapa gerado automàticamente, vendo-se que este percorre o ecran e fixa-se nas últimas linhas, pois o comprimento da página dum mapa default é definido para a impressora.

No seu ecran terá a seguinte parte do mapa:

```
order_num               1014
order_date              06/05/1984
customer_num            106
ship_instruct           ring bell, kick door loudly
backlog                 n
po_num                  8052
ship_date               06/09/1984
ship_weight             40.60
ship_charge             $12.30
paid_date               07/18/1984

order_num               1015
order_date              06/06/1984
customer_num            110
ship_instruct           closed Mon
backlog                 n
po_num                  MA003
ship_date               06/11/1984
ship_weight             20.60
ship_charge             $6.30
paid_date               06/28/1984
```

Como se vê o mapa default é bastante grosseiro mostrando a totalidade das linhas da tabela seleccionada e apresentadas de forma identica à dum resultado duma instrução SELECT executada pelo menu RDSQL.

2.2.2. Descrição dum Mapa Default

Seleccione a opção MODIFY e o mapa "ordcust". --> Aparece um editor, neste caso o vi, com a seguinte listagem de instruções:

```
database stores end
select
      order_num,
      order_date,
      customer_num,
      ship_instruct,
      backlog,
      po_num,
      ship_date,
      ship_weight,
      ship_charge,
      paid_date
from orders end
format every row end
```

Analisando este conjunto de instruções destaca-se o seguinte:

- Existem 3 grupos distintos de instruções,correspondendo a cada um deles uma determinada secção do mapa, terminada pela palavra END.
- A primeira secção é constituída por uma única linha, que

identifica a base de dados "database stores end". É a DATABASE SECTION.

- A segunda secção contém a instrução SELECT que selecciona as colunas e linhas da tabela que irão constituir o mapa.
- É a SELECT SECTION. A última secção é constituida pelas palavras EVERY ROW que identificam o formato default. É a FORMAT SECTION.

Estas secções e outras 3 opcionais (DEFINE, INPUT e OUTPUT), que serão explicadas no capítulo seguinte,constituem a estrutura de qualquer mapa.

2.3. O ACE e as suas Capacidades

Neste capítulo vamos descrever a maior parte das instruções do ACE. Estas irão sendo introduzidas durante as várias fases do processo de desenvolvimento e um mapa que inclua as instruções normalmente utilizadas nos tipos de mapas que fazem parte de qualquer aplicação.

O mapa a construir começa por ser uma proposta genérica, que vai ser desenvolvida e aperfeiçoada por etapas. A ordem de apresentação destas é a que normalmente se utiliza na prática, salvo nalgumas situações para facilitar a exposição.

A proposta de partida e' a seguinte:

- Considerando a base de dados STORES pretende-se elaborar um relatório que contenha as encomendas e respectivos montantes efectuadas por cada cliente, num determinado periodo de tempo.
- Pretende-se apresentar no relatório o valor e o periodo de tempo que decorreu entre o envio e o pagamento de cada encomenda.
- As encomendas serão agrupadas por cliente , com o respectivo total.
- No final do relatório apresenta-se o total das encomendas enviadas nesse período.
- O periodo de tempo deverá ser definido por uma data de começo e por uma data de fim, devendo estas ser introduzidas a pedido do programa.

Como ponto de partida vamos aproveitar o relatório default ORDCUST, gerado no capítulo anterior. Vamos actualizá-lo progressivamente, através da opção MODIFY do menu REPORT, para,`a medida que formos introduzindo as potencialidades do ACE obter o relatório proposto.

2.3.1. DATABASE SECTION

Esta secção do mapa é obrigatória e é a primeira secção do relatório. Identifica a base de dados onde reside a informação necessária ao relatório. Exemplo:

```
database
      stores end
```

NOTA: Pode inserir comentários em qualquer ponto do programa desde que os escreva entre chavetas "{" ... "}" .

2.3.2. DEFINE SECTION

Esta secção é opcional e, quando existir, deve situar-se depois da DATABASE SECTION. Define as variáveis e parâmetros (nome da variável ou parâmetro e respectivo tipo de dados).

Os tipos de dados possiveis são os mesmos que se apresentaram no RDSQL com excepção do tipo SERIAL que tem de ser substituido por INTEGER.

Neste capítulo, dedicado à obtenção de reports através do ACE e do menu INFORMIX-SQL, só iremos abordar a definição e utilização de variáveis, pois os parâmetros só podem ser utilizados quando o ACE é chamado pelo sistema operativo.

Exemplo:

```
define
      variable totcli decimal(10,2) { acumulador do total p/
         cliente }
      variable totger decimal(10,2) { acumulador do total geral }
      variable tot1 decimal(10,2)   { total encomenda }
      variable pg1 char(3)    { prazo de  pagamento }
      variable pgc1 char(5)   { dias  do  prazo  de pagamento}
      variable inic  date     {  data  inicio  do periodo}
      variable  fim  date     {  data   fim   do periodo}
end
```

NOTAS:

- Este exemplo corresponde à DEFINE SECTION do mapa ORDCUST. Neste distinguem-se 3 tipos de variáveis:
  1. as que se destinam a cálculos de valores parciais e finais (totcli, totger).
  2. as que se destinam a guardar e controlar os valores a imprimir (pg1,pgc1).
  3. finalmente as que se destinam a receber as datas do periodo a consultar (inic e fim). Estas variáveis poderão ser utilizadas em qualquer parte da FORMAT SECTION e na SELECT SECTION, nesta última desde que sejam precedidas do caracter "$".
- As variáveis podem ser inicializadas com valores introduzidos pelo ecran, desde que se defina na INPUT SECTION uma instrução PROMPT para essas variáveis, como veremos a seguir.

2.3.3. INPUT SECTION

Esta secção é opcional e, quando existir, deve situar-se depois da DEFINE SECTION. Indica quais as variáveis que são inicializadas exteriormente ao programa e define a mensagem que aparecerá no ecran, para indicação de qual a variável a inicializar.

Esta secção é constituida só por instruções PROMPT.

Exemplo:

```
input
      prompt for inic using
            " Introduza a  data  de  inicio (mm/dd/aaaa):"
      prompt for fim  using
            " Introduza  a  data  de  fim (mm/dd/aaaa):"
end
```

NOTA: Não se pode aceitar através da instrução PROMPT nem nomes de bases de dados nem nomes de ficheiros externos para onde se envie o relatório (ver OUTPUT SECTION).

2.3.4. OUTPUT SECTION

Esta secção é opcional e, quando existir, deve situar-se entre a INPUT SECTION e a SELECT SECTION. Define as dimensões da página do relatório e especifica qual o seu suporte de saída (ecran,ficheiro, impressora, etc).

Esta secção tem o formato que se apresenta a seguir:

```
output
      report to  <instrução>
      left margin  <instrução>
      right margin  <instrução>
      top margin  <instrução>
      bottom margin  <instrução>
      page lengh  <instrução>
end
```

NOTA: Qualquer dos comandos da OUTPUT section é opcional. Nos casos em que não sejam utilizados o ACE assume as especificações default.

2.3.4.1. REPORT TO

Este comando envia o output do ACE (o relatório) para o ecran, um ficheiro, uma impressora ou, nos sistemas unix, para uma pipe.

Exemplos:

1. report to "nome de ficheiro"
2. report to printer
3. report to pipe "nome de programa"

NOTAS:

- Quando não se especificar este comando o relatório é enviado para o ecran.
- Não se pode usar REPORT TO mais do que uma vez no programa de especificação do relatório.

2.3.4.2. LEFT MARGIN

Este comando define o comprimento da margem esquerda do relatório. No exemplo que se segue define-se uma margem esquerda com 2 espaços.

Exemplo:

```
left margin 2
```

NOTAS: O comprimento da margem direita é de 5 espaços.

2.3.4.3. RIGHT MARGIN

Este comando define a margem direita do relatório. Esta margem não está relacionada com a margem esquerda, definindo apenas o comprimento da página em caracteres, contados a partir da margem esquerda da página (espaço 0).

Exemplo:

```
right margin 80
```

NOTAS:

- O comprimento da margem esquerda default é de 132 caracteres.
- Este comando só se pode utilizar com o bloco de controlo EVERY ROW (utilizado no relatório default).

2.3.4.4. TOP MARGIN

Este comando define o número de linhas que são deixadas em branco antes de se escrever a primeira linha da página do relatório.

Exemplo:

```
top margin 2
```

NOTAS: O comprimento da margem inicial default é de 3 linhas.

2.3.4.5. BOTTOM MARGIN

Este comando define o número de linhas que são deixadas em branco depois de se escrever a última linha da página do relatório.

Exemplo:

```
bottom margin 2
```

NOTAS: O comprimento da margem fim de página default é de 3 linhas.

2.3.4.6. PAGE LENGHT

Este comando define o número de linhas de cada página do relatório.

Exemplo:

```
page length 60
```

NOTAS:

- O comprimento da página default é de 66 linhas.
- O comprimento da página, definido pelo PAGE LENGTH inclui as margens inicial e final definidas pelos comandos TOP e BOTTOM MARGIN.

Como exemplo de sintese vamos escrever a OUTPUT SECTION do relatório ORDCUST.

```
output
      page length 24
      top margin 0
      bottom margin 0
      left margin 0
end
```

NOTAS:

- Analisando este exemplo vê-se que se trata dum mapa a enviar para o ecran, com comprimento de 24 linhas e em que se aproveita o espaço ao máximo, pois especificaram-se margens nulas (verticais e horizontais).
- A margem direita, ou seja o comprimento da linha será definido pela lógica do programa. Como não se trata dum mapa default não se pode usar o comando RIGHT MARGIN.

2.3.5. SELECT SECTION

Nesta secção seleciona-se a informação da base de dados que constitui o relatório.

Esta secção é obrigatória e é constuida por uma ou várias instruções SELECT, mantendo-se as regras já aprendidas no RDSQL com as seguintes pequenas alterações.

- Quando se usarem várias instruções SELECT encadeadas têm de possuir, excepto na última, a cláusula INTO TEMP e guardar na tabela temporária o resultado intermédio que servirá de input para o SELECT seguinte.
- Só se pode usar a cláusula ORDER BY no último SELECT.

- Na cláusula ORDER BY não se podem usar nem inteiros nem o nome de coluna com o nome da tabela como prefixo (tabela.coluna).

  No caso de não ser possivel distinguir duas ou mais colunas de tabelas diferentes, pelo nome da coluna, devem-se usar labels para identificar distintamente todas as colunas.
- A SELECT SECTION deve terminar com END.

Vamos considerar como exemplo o SELECT para o relatório ORDCUST:

```
select
      orders.customer_num cust,  orders.order_num  ord,
      orders.ship_date,orders.paid_date, items.total_price,
      customer.company
from
      orders, items, customer
where
      orders.order_date between $inic and $fim
      and items.order_num = orders.order_num
      and customer.customer_num = orders.customer_num
order by cust, ord
end
```

NOTA:

- Neste exemplo, embora não fossem necessárias, usaram-se as labels "cust" e "ord" para exemplificar a sua utilizacão.
- Repare na primeira linha da cláusula WHERE, na utilização das variáveis com as datas que definem o periodo de pesquisa, inicializadas exteriormente ao programa ($inic e $fim).

2.3.6. FORMAT SECTION

Esta secção é obrigatória e segue-se à SELECT SECTION. Define, quer o tratamento a dar à informação resultante do último ou único SELECT, quer a forma do relatório.

Existem dois tipos de FORMAT SECTION. O primeiro é o que foi gerado no mapa default e tem a seguinte estrutura:

```
format
      every row
end
```

O segundo tipo é mais complexo e tem de conter obrigatoriamente um ou mais blocos de controlo. Estes blocos têm de incluir, pelo menos, uma instrução, por exemplo PRINT ou SKIP. Este tipo de FORMAT, que não permite a inclusão da instrução EVERY ROW, tem a seguinte estrutura:

```
format
      page header {1 bloco de controlo}
      page trailer      {   "          "    }
      first page header {   "          "    }
      on every row      {   "          "    }
      on last row {   "          "    }
      before group off  {n blocos de controlo}
      after group off   {n blocos de controlo}
```

Na FORMAT SECTION não se pode referir a uma coluna pelo prefixo da tabela mais nome da coluna (tabela.coluna) utilizando-se, caso seja necessário, as labels que se definiram na SELECT SECTION.

A ordem de escrita dos blocos de controlo é arbitrária.

O número de blocos BEFORE (ou AFTER) que se podem utilizar é igual ao número de colunas existentes na cláusula ORDER BY do SELECT.

2.3.6.1. EVERY ROW

Esta instrução usa-se quando se pretende gerar um relatório rapidamente, sem nenhumas preocupações com a sua forma de apresentação. Aparece sempre nos mapas default.

Quando usar EVERY ROW não pode especificar nenhum bloco de controlo.

2.3.6.2. AFTER GROUP OF

Este bloco de controlo define as acções (instruções) que se devem executar depois de ser processado um determinado grupo de linhas. Um grupo de linhas define-se como um conjunto de linhas que têm o mesmo valor para uma determinada coluna que figure na cláusula ORDER BY do SELECT.

Para melhor compreender este bloco de controlo podemos compará-lo com as acções a executar quando se detecta uma quebra imediatamente antes de se começar a processar um novo grupo, nos processamentos de quebra de ficheiros sequenciais clássicos.

Exemplo:

```
after group of cust
      let totcli = group total of total_price
      let totger = totger + totcli
```

```
      print column 01, "TOTAL CLIENTE:",
            column 17, totcli using "$$$$$####&.&&"
```

NOTAS:

- Repare-se que o bloco em questão inclui uma série de instruções variadas (testes, operacões aritméticas, assignamentos, funções agregadas, impressão, etc).
- A instrução SELECT tem de ter cláusula ORDER BY.
- No mapa ORDCUST vamos ter 2 blocos AFTER GROUP OF. Um para cada nivel de quebra, ou seja um para o grupo definido pelos items da mesma encomenda (com ORD igual) e outro para o grupo definido pelas encomendas do mesmo cliente (com CUST igual).

2.3.6.3. BEFORE GROUP OF

Este bloco de controlo define as acções (instruções) que se devem executar antes de ser processado um determinado grupo de linhas.

Exemplo:

```
before group of cust
      skip 1 line
      need 2 lines
      print column 01, "CLIENTE:",
            column 10, cust using "<<<<",
            column 16, company clipped
```

NOTAS:

- Estes blocos poderão ser utilizados para limpar variáveis de acumulaçao de totais ou de controlo.
- A instrução SELECT tem de ter cláusula ORDER BY.
- No mapa ORDCUST vamos ter 1 bloco BEFORE GROUP OF, associado à coluna cust.

2.3.6.4. ON EVERY ROW

Este bloco de controlo define as acções (instruções) que se devem executar para cada linha do resultado do SELECT.

Repare na diferença entre este bloco e EVERY ROW.

Exemplo:

```
on every row
      print itens.quantitity
```

NOTAS: No mapa ORDCUST não vamos ter blocos ON EVERY ROW porque as acções sobre cada linha do resultado (acumulação do TOTAL_PRICE para cada encomenda) serão especificadas no bloco AFTER GROUP OF ORD pela função agregada GROUP TOTAL.

2.3.6.5. ON LAST ROW

Este bloco de controlo define as acções (instruções) que se devem executar depois de processada a última linha do resultado do SELECT.

Exemplo:

```
on last row
      skip 3 line
      print column 01, "TOTAL GERAL:",
            column 17, totger using "$$$$$####&.&&"
```

NOTAS: No mapa ORDCUST vamos ter no bloco ON LAST ROW as instruções para imprimir os totais gerais.

Os 4 blocos de controlo descritos anteriormente têm uma ordem de processamento bem definida, representada graficamente pela figura que se segue, em que a ordem de processamento é representada pela ordem de escrita dos blocos de controlo.

```
before group of cust
before group of ord
on every row
after proup of ord
after group of cust
on last row
```

NOTAS:

- A ordem de execução dos BEFORE GROUP é igual à ordem das colunas na cláusula ORDER BY do SELECT.
- A ordem de execução dos AFTER GROUP é inversa da ordem das colunas na cláusula ORDER BY do SELECT.

2.3.6.6. FIRST PAGE HEADER

Este bloco de controlo inclui as acções (instruções) que se devem executar para definir o cabeçalho da primeira página.

Exemplo:

```
first page header
      print "RESUMO DAS ENCOMENDAS POR ",
            "CLIENTE DE ", inic , " A ", fim,
            column 70, "pagina ", pageno using "<<<"
```

NOTAS:

- O ACE executa este bloco de controlo antes de gerar qualquer output. Por isso também pode ser utilizado para inicializar variáveis.
- No mapa ORDCUST usou-se o mesmo cabeçalho para todas as páginas. Portanto não se utilizou este bloco.

2.3.6.7. PAGE HEADER

Este bloco de controlo inclui as acções (instruções) que se devem executar para definir o cabeçalho de cada página.

Exemplo:

```
page header
      print  "RESUMO DAS ENCOMENDAS POR ",
            "CLIENTE DE ", inic , " A ", fim,
             column 70, "pagina ", pageno using "<<<"
```

NOTAS:

- Na primeira página aparece o cabeçalho definido no bloco FIRST PAGE HEADER, se existir.
- Os cabeçalhos definidos no FIRST PAGE HEADER e no PAGE HEADER têm de ter o mesmo número de linhas.

2.3.6.8. PAGE TRAILER

Este bloco de controlo inclui as acções (instruções) que se devem executar para definir o rodapé de cada página.

Exemplo:

```
page trailer
      pause "pressione RETURN para continuar"
```

NOTAS: Não se pode incluir a instrução SKIP TO TOP OF PAGE.

2.3.6.9. INSTRUÇÕES

Os blocos de controlo da FORMAT SECTION determinam quando uma ou várias acções ocorrem, enquanto que as INSTRUÇÕES determinam qual a acção a efectuar.

Pode considerar-se dois tipos de instruções:

- Simples: constituida por uma única instrução.
- Compostas: constituidas por um grupo de instruções encabeçadas pela palavra BEGIN e finalizadas pela palavra END.

2.3.6.9.1. FOR

A instrução FOR define um ciclo (loop), executando consecutivamente uma instrução simples ou composta, incrementando o indice e controlando a condição de saída, e quando esta se verifica passa o controlo do programa para a instrução seguinte ao fim do ciclo.

Exemplo:

```
for i=1 to 10 step 2 do
      begin
      let a = a * a + 2 let b = a * 2
      end
```

NOTAS:

- Não se pode usar incrementos negativos.
- Se se tiver várias instruções a seguir ao DO têm de estar entre BEGIN e END.

2.3.6.9.2. IF THEN ELSE

Esta instrução executa um teste a uma condição e processa, em alternativa, 2 instruções (simples ou compostas) conforme a condição se verifica ou não.

Exemplo:

```
if paid_date is null then
      begin
            let pg1 = "fal"
            let pgc1 = "ta   "
      end
else
      begin
            let pg1 = paid_date - ship_date
            let pgc1 = " dias"
      end
```

NOTAS:

- Pode-se usar até 128 IF THEN ELSE embebidos.
- Se se tiver várias instruções a seguir a THEN ou ELSE têm de estar entre BEGIN e END.

2.3.6.9.3. LET

A instrução LET assigna um valor a uma variável.

Exemplo:

```
begin
      let pg1 = paid_date - ship_date
      let pgc1 = " dias"
end
```

2.3.6.9.4. NEED

Esta instrução faz com que a próxima linha seja impressa na página seguinte, se não estiver disponível na página corrente, o número de linhas especificado na instrução NEED.

Exemplo:

```
before group of cust
      skip 1 line
      need 2 lines
      print column 01, "CLIENTE:",
            column 10, cust using "<<<<",
            column 16, company clipped
```

2.3.6.9.5. PAUSE

Esta instrução faz com que um relatório destinado a um ecran possa ser fixado para ser visualizado. Pressione RETURN para avançar pelo relatório.

Exemplo 1:

```
page trailer
      pause "pressione RETURN para continuar"
```

Exemplo 2:

```
before group of cust
      skip 1 line
      need 2 lines
      print column 01, "CLIENTE:",
            column 10, cust using "<<<<",
            column 16, company clipped
```

NOTA: Esta instrução não tem nenhum efeito se o relatório for dirigido para uma impressora, um ficheiro ou uma pipe.

2.3.6.9.6. PRINT

Esta instrução imprime informação no ecran, numa impressora ou num ficheiro, conforme for especificado na OUTPUT SECTION.

Exemplo:

```
print column 10, "Encomenda:",
      column 21, ord using "<<<<",
      column 29, "Total:",
      column 36, tot1 using "$$$$$####&.&&",
      column 53, "Pagamento em:",
      column 67, pg1, pgc1 clipped
```

NOTA:

- Cada instrução PRINT imprime o seu output numa linha. Pode-se, com um único PRINT escrever várias linhas, separando-as com ";".
- Conforme o tipo de dados, no PRINT, as colunas ocupam espaços previamente fixados. Para controlar o comprimento dos campos no relatório utilize as palavras CLIPPED e USING.

2.3.6.9.7. SKIP

Esta instrução salta o número de linhas em branco especificadas a seguir a SKIP.

2.3.6.9.8. SKIP TO TOP OF PAGE

Esta instrução faz com que a próxima linha seja escrita no início da próxima página.

2.3.6.9.9. WHILE

A instrução WHILE define um ciclo, executando consecutivamente uma instrução simples ou composta, enquanto for verdadeira a condição que segue a palavra WHILE. Quando esta se verifica passa, o controlo do programa para a instrução seguinte ao fim do ciclo.

Exemplo:

```
while pageno = 1 do
      begin
            let a = a * a + 2
            let b = a * 2
      end
```

NOTAS: Se se tiver várias instruções a seguir ao DO têm de estar entre BEGIN e END.

2.3.6.9.10. FUNÇÕES AGREGADAS

As funções agregadas possibilitam a execução de operações sobre colunas das linhas de todo resultado do SELECT, podendo-se ainda escolher entre estas as que satisfaçam determinado critério.

Existe ainda a hipótese de definir funções agregadas só para grupos de quebra, desde que se utilize a palavra GROUP.

A estrutura da instrução para definir uma função agregada é a seguinte:

```
[group]
      <função agregada> of
       <coluna ou expressão aritmética envolvendo coluna>
[where  <expressão lógica>]
```

NOTAS:

- GROUP e WHERE são opcionais.
- GROUP só se pode usar num bloco AFTER.

Estão disponiveis as seguintes funções:

COUNT Conta o número total de linhas do resultado ou do grupo e que satisfaçam a cláusula WHERE, se existir.

PERCENT Executa um função de contagem semelhante a **COUNT** mas dá o resultado em percentagem do número total de linhas seleccionadas a considerar.

TOTAL Acumula o valor da coluna especificada para todas as linhas do resultado ou do grupo e que satisfaçam a cláusula WHERE, se existir.

AVERAGE ou AVG Calcula o valor médio da coluna especificada para todas as linhas do resultado ou do grupo e que satisfaçam a cláusula WHERE, se existir.

MIN Calcula o valor mínimo da coluna especificada para todas as linhas do resultado ou do grupo e que satisfaçam a cláusula WHERE, se existir.

MAX Calcula o valor máximo da coluna especificada para todas as linhas do resultado ou do grupo e que satisfaçam a cláusula WHERE, se existir.

Exemplos:

```
after group of cust
      let totcli = group total of total_price
      let totger = totger + totcli
      print column 01, "TOTAL CLIENTE:",
            column 17, totcli using "$$$$$####&.&&"
      print group total of total_price where  total_price  > 500
            using "$$$$$####&.&&"
```

2.3.6.9.11. CLIPPED

Esta instrução suprime nos campos alfanuméricos,todos os espaços à direita do último caracter diferente de espaço.

Exemplo:

```
before group of cust
      skip 1 line
      need 2 lines
      print column 01, "CLIENTE:",
            column 10, cust using "<<<<",
            column 16, company clipped
```

2.3.6.9.12. COLUMN

Esta instrução permite imprimir, o campo que se segue, a partir da coluna especificada.

Exemplo:

```
before group of cust
      skip 1 line
      need 2 lines
      print column 01, "CLIENTE:",
            column 10, cust using "<<<<",
            column 16, company clipped
```

2.3.6.9.13. LINENO

Esta expressão contém o número da linha que o ACE está a imprimir actualmente.

Exemplo:

```
if lineno = 24 then skip to top of page
```

NOTA: Não utilize LINENO em blocos page header ou trailer porque não funcionará correctamente.

2.3.6.9.14. PAGENO

Esta expressão contém o número da página que o ACE está a imprimir actualmente.

Exemplo:

```
page header
      print  "RESUMO DAS ENCOMENDAS POR ",
            CLIENTE DE  ", inic , "  A  ", fim,
            column 70, "pagina ", pageno using "<<<"
```

Esta expressão permite definir o formato de impressão dos campos tipo numérico e data, a que está associada.

Exemplos:

```
print  "RESUMO DAS ENCOMENDAS POR ",
      "CLIENTE DE  ", inic , "  A  ", fim,
      column 70, "pagina ", pageno using "<<<"
print column 10, "Encomenda:",
      column 21, ord using "<<<<",
```

```
      column 29, "Total:",
      column 36, tot1 using "$$$$$####&.&&",
      column 53, "Pagamento em:",
      column 67, pg1, pgc1 clipped
```

NOTAS: O conjunto de caracteres que definem o formato têm de estar entre " ". USING pode ser usado nas instruções PRINT e LET. Se, por acaso um campo tiver comprimento superior ao espaço definido para o imprimir, este será preenchido a asteriscos.

A seguir apresentam-se e descrevem-se os principais caracteres para definição do formato:

Numéricos

"*****" Imprime asteriscos em vez de zeros não significativos.

"**&**" Mantém os zeros não significativos.

"**#**" Imprime espaços em vez de zeros não significativos.

"**<**" Imprime um número encostado à esquerda.

"**,**" Este caracter é imprimido na posição em que for escrito. Se não existirem números à sua esquerda não é imprimido.

"**.**" Este caracter é imprimido na posição em que for escrito. Só pode existir um ponto por campo.

"**-**" Este caracter é imprimido quando o número for negativo.

"**+**" Este caracter imprime "+" quando o número for positivo e "-" quando for negativo.

"**$**" Este caracter é imprimido antes do número.

Datas:

"**dd**" Imprime o dia do mês como um número (01 a 31).

"**ddd**" Imprime o dia da semana usando uma abreviatura com letras.

"**mm**" Imprime o mês como um número (01 a 12).

"**mmm**" Imprime o mês usando uma abreviatura com 3 letras.

"**yy**" Imprime o ano como um número de 2 digitos (01 a 99).

"**yyyy**" Imprime o ano como um número de 4 digitos (0001 a 9999).

2.3.7. O relatorio ORDCUST completo

Neste capítulo vamos apresentar a totalidade do programa para produzir o relatório ORDCUST e a seguir o próprio relatório.

Programa:

```
database
      stores
end
define
      variable totcli decimal(10,2)    { acumulador do total p/
                                           cliente }
      variable totger decimal(10,2)    { acumulador do total geral }
      variable tot1 decimal(10,2)      { total encomenda }
      variable pg1 char(3)             { prazo de  pagamento }
      variable pgc1 char(5)            { dias  do  prazo  de pagamento}
      variable inic  date              {  data  inicio  do periodo}
      variable  fim  date              {  data   fim   do periodo}
end
input
      prompt for inic using " Introduza a  data  de  inicio
            (mm/dd/aaaa): "
      prompt for fim  using  "  Introduza  a  data  de  fim
            (mm/dd/aaaa):  "
end
output
      page length 24
      top margin 0
      bottom margin 0
      left margin 0
end
select orders.customer_num cust,  orders.order_num  ord,
      orders.ship_date,orders.paid_date, items.total_price,
      customer.company
from orders, items, customer
where orders.order_date between $inic and $fim
      and items.order_num = orders.order_num
      and customer.customer_num = orders.customer_num
order by cust, ord
end
format
      page header
            print  "RESUMO DAS ENCOMENDAS POR ",
                  "CLIENTE DE  ", inic , "  A  ", fim,
                  column 70, "pagina ", pageno using "<<<"
      page trailer
            pause "pressione RETURN para continuar"
      after group of ord
            let tot1 =  group total of total_price
            if paid_date is null then
                  begin
                        let pg1 = "fal"
                        let pgc1 = "ta   "
                  end
            else
                  begin
                        let pg1 = paid_date - ship_date
                        let pgc1 = " dias"
                  end
            print column 10, "Encomenda:",
```

```
                  column 21, ord using "<<<<",
                  column 29, "Total:",
                  column 36, tot1 using "$$$$$####&.&&",
                  column 53, "Pagamento em:",
                  column 67, pg1, pgc1 clipped
      before group of cust
            skip 1 line
            need 2 lines
            print column 01, "CLIENTE:",
                  column 10, cust using "<<<<",
                  column 16, company clipped
      after group of cust
            let totcli = group total of total_price
            let totger = totger + totcli
            print column 01, "TOTAL CLIENTE:",
                  column 17, totcli using "$$$$$####&.&&"
      on last row
            skip 3 line
            print column 01, "TOTAL GERAL:",
                  column 17, totger using "$$$$$####&.&&"
end
```

RELATÓRIO:

```
RESUMO DAS ENCOMENDAS POR CLIENTE DE  01/01/1984  A  01/01/1991 pagina 1

CLIENTE: 101   All Sports Supplies
         Encomenda: 1002   Total:   $ 1200.00    Pagamento em:  27 dias
TOTAL CLIENTE:      $ 1200.00

CLIENTE: 104   Play Ball!
         Encomenda: 1001   Total:   $  250.00    Pagamento em:  49 dias
         Encomenda: 1003   Total:   $  959.00    Pagamento em:  22 dias
         Encomenda: 1011   Total:   $   99.00    Pagamento em:  49 dias
         Encomenda: 1013   Total:   $  143.80    Pagamento em:  22 dias
TOTAL CLIENTE:      $ 1451.80

CLIENTE: 106   Watson & Son
         Encomenda: 1004   Total:   $ 2126.00    Pagamento em: falta
         Encomenda: 1014   Total:   $ 1440.00    Pagamento em:  69 dias
TOTAL CLIENTE:      $ 3566.00

CLIENTE: 110   AA Athletics
         Encomenda: 1008   Total:   $  940.00    Pagamento em:  15 dias
         Encomenda: 1015   Total:   $  450.00    Pagamento em:  30 dias
TOTAL CLIENTE:      $ 1390.00


RESUMO DAS ENCOMENDAS POR CLIENTE DE  01/01/1984  A  01/01/1991 pagina 2
CLIENTE: 111   Sports Center
         Encomenda: 1009   Total:   $  450.00    Pagamento em:  48 dias
TOTAL CLIENTE:      $  450.00

CLIENTE: 112   Runners & Others
         Encomenda: 1006   Total:   $  498.00    Pagamento em: falta
TOTAL CLIENTE:      $  498.00
```

```
CLIENTE: 115   Gold Medal Sports
         Encomenda: 1010   Total:   $   84.00    Pagamento em:  44 dias
TOTAL CLIENTE:      $   84.00

CLIENTE: 116   Olympic City
         Encomenda: 1005   Total:   $  562.00    Pagamento em:  11 dias
TOTAL CLIENTE:      $  562.00

CLIENTE: 117   Kids Korner
         Encomenda: 1007   Total:   $ 1696.00    Pagamento em: falta
         Encomenda: 1012   Total:   $ 1040.00    Pagamento em: falta
TOTAL CLIENTE:      $ 2736.00



RESUMO DAS ENCOMENDAS POR CLIENTE DE  01/01/1984  A  01/01/1991  agina 3
TOTAL GERAL:        $11937.80
```

2.4. Uso de Reports no Sistema Operativo

É possivel criar e executar relatórios sem entrar nos menus do INFORMIX-SQL.

2.4.1. O comando SACEGO

Para processar um relatório através do sistema operativo basta escrever o comando SACEGO seguido do nome do report. Por exemplo se quiser processar o report ORDCUST basta digitar na linha de comando do sistema operativo: 1% sacego ordcust e pressionar RETURN para lhe aparecer no ecran o já conhecido report ORDCUST.

2.4.1.1. Parâmetros

Para exemplificar o uso de parâmetros na DEFINE SECTION vamos alterar o processo de introdução das datas que limitam o periodo de consulta no mapa ORDCUST. Altere a DEFINE SECTION de modo a obter:

```
define
      variable totcli decimal(10,2)   { acumulador do total p/ }
                                      {cliente }
      variable totger decimal(10,2)   { acumulador do total geral }
      variable tot1 decimal(10,2)     { total encomenda }
      variable pg1 char(3)            { prazo de  pagamento }
      variable pgc1 char(5)           { dias do prazo  para pagamento}
      param[1] inic  date             {  data  inicio  do periodo}
      param[2]  fim  date             {  data   fim   do periodo}
end
```

e apague a INPUT SECTION.

Agora lance o relatório ORDCUST através do sistema operativo com o comando:

```
1% sacego ordcust 01/01/1984 12/01/1984
```

NOTA: Os numeros entre [ ] na definição dos parâmetros na DEFINE SECTION indicam a ordem da apresentação dos valores desses parâmetros no comando SACEGO.

2.4.1.2. O comando SACEPREP

O processo de criação dum report através do sistema operativo é o seguinte: Cria-se um ficheiro com as instruções do programa do relatório, a se que dá a extensão ".ace". Compila-se esse programa através do comando SACEPREP.

O comando SACEPREP tem a estrutura dum comando UNIX, com duas opções disponíveis.

-s Esta opção compila o report fonte existente no ficheiro cujo nome segue a opção no comando. Note que o verdadeiro nome do ficheiro é constituído pelo nome existente no comando seguido pela extensão ".ace". No caso de existirem incorrecções, cria um ficheiro com o mesmo nome do report fonte mas com extensão ".err", e que contém as mensagens que indicam os pontos onde se verificam as incorrecções. Pode-se editar este ficheiro e corrigir os erros e, depois de se apagarem as mensagens de erro, gravá-lo no ficheiro com extensão ".ace" e compilá-lo de novo.

-o <**nome de directório**> Esta opção dirige o output da compilação, quer se trate do programa objecto quer do ficheiro .err, para o directório especificado.

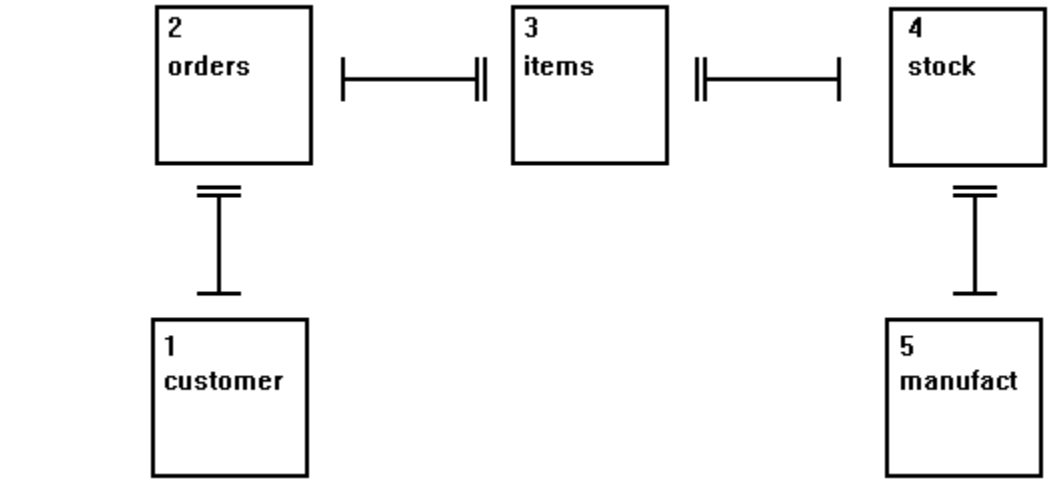
Exemplo:

```
sfceprep -s ordcust
```

3. Apêndice 1

DESCRICAO DA BASE DE DADOS STORES.

MOREDATA/DE1 * MODELO DE DADOS * REF:STODE1
PROJECTO: Informix-sql BASE DADOS: Stores



MOREDATA/DE1 * LISTA DE ENTIDADES * REF:STODE1
PROJECTO: Informix-sql BASE DADOS: Stores

IDENTIFICAÇÃO	DESIGNAÇÃO	CHAVES PRIMÁRIAS	CHAVES ESTRANGEIRAS	O/R
1	customer	customer_num		
2	orders	order_num	customer_num	
3	items	item_num + order_num	stock_num + manu_code	
4	stock	stock_num + manu_code		
5	manufact	manu_code		

MOREDATA/DE1 * DETALHE DE ENTIDADES * REF:STODE1
PROJECTO: Informix-sql BASE DADOS: Stores

IDENTIFICAÇÃO	m	EXPLICAÇÕES / OBSERVAÇÕES	TIPO	
1 CUSTOMER		CLIENTES		
customer_num	s	Número de identificação do cliente	serial	101
fname		Primeiro nome do cliente	char	15
lname		Ultimo nome do cliente	char	15
company		Empresa	char	20
address1		Linha 1 da morada	char	20
address2		Linha 2 da morada	char	20
city		Cidade	char	15
state		Estado	char	2
zipcode		Código postal	char	5
phone		Telefone	char	13
2 ORDERS		ENCOMENDAS		
order_num	s	Número identificador da encomenda	serial	1001
order_date		Data da encomenda	date	
customer_num		Número do cliente que fez a encomenda	integer	
ship_instruct		Instruções para a entrega	char	40
backlog			char	1
po_num		Número de referência da encomenda	char	10
ship_date		Data de expedição	date	
ship_weight		Peso da encomenda	decimal	8,2
ship_charge		Despesas com a expedição	money	6
paid_date		Data de pagamento	date	
3 ITEMS		LINHAS DE ENCOMENDA		
item_num		Número da linha de encomenda	smallint	
order_num		Número identificador da encomenda	integer	
stock_num		Número do artigo	smallint	
manu_code		Código identificador do fabricante	char	3
quantity		Quantidade do artigo encomendado	smallint	
total_price		Preço total da linha de encomenda	smallint	
4 STOCK		ARTIGOS		
stock_num		Número do artigo	smallint	
manu_code		Código identificador do fabricante	char	3
description		Descrição do artigo	char	15
unit_price		Preço unitário	money	6
unit		Unidade	char	4
unit-descr		Descrição da unidade	char	15
5 MANUFACT		FABRICANTE		
manu_code		Código do fabricante	char	3
manu_name		Nome do fabricante	char	15

MOREDATA/DE1 * ESQUEMA * REF:STODE1
PROJECTO: Informix-sql BASE DADOS: Stores

Esquema de ligação entre as tabelas da base de dados STORES