MoreData

INTRODUÇÃO AO SQL EM INFORMIX

1. INTRODUÇÃO

A linguagem RDSQL é a linguagem de acesso à base de dados fornecida pela informix.

O utilitário dbaccess é fornecido com os motores de dases de dados, e permite a execução do RDSQL interactivo e gestão básica da estrutura da base de dados.

1.1. Noções de Bases de Dados Relacionais: Um exemplo

Para uma melhor apreensão global e integrada dos comandos do RDSQL 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.

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.

1.1.1. 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. DBACCESS

DBACCESS é um utilitário fornecido com o Sistema de Gestão de Bases de Dados Relacionais Informix e que executa as seguintes tarefas:

```
Criar Bases de Dados e Tabelas.
Carregar dados de ficheiros do sistema operativo.
Fazer buscas através de SQL (System Query Language) interactivo.
```

Para aceder ao DBACCESS digite **dbaccess** e aparecerá no ecran o Main Menu do dbaccess, se este estiver bem instalado.

1.2.1. Os ecrans do **DBACCESS**

O DBACCESS 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 DBACCESS permite executar são sempre iniciadas a partir de selecção duma opção do Main Menu.

As opções do Main Menu são as seguintes :

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

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 DBACCESS e voltar ao sistema operativo.

1.2.3. Ambiente do INFORMIX

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
```

INFORMIX TERM Existem no UNIX duas bases de dados de capacidades dos terminais termcap e terminfo. Para escolher a que deseja utilizar deverá activar a variável INFORMIXTERM com um dos valores atrás descritos, como por exempllo:

```
INFORMIXTERM=termcap
export INFORMIXTERM
```

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 o motor standard (SE) e

```
SQLEXEC=sqlturbo
export SQLEXEC
```

para a versão on-line

2. RDSQL

2.1.Acedendo ao RDSQL

Seleccionando a opção Query-language do Main Menu pode inserir e correr instruções de RDSQL, isoladas ou em sequência sendo neste caso separadas pelo ";".

Depois de seleccionar a opção Query-language aparecerá o ecran CHOOSE DATABASE indicando que, neste momento, está disponível a base de dados STORES.

Para seleccionar a base de dados STORES digite o nome respectivo e pressione RETURN, ou então selecione a base de dados, deslocando o cursor ao longo da lista das bases de dados e pressione RETURN. Aparecerá então o ecran de menu RDSQL.

2.2. As Opções do RDSQL

O menu do RDSQL tem as seguintes opções :

New Permite introduzir um novo conjunto de instruções, através do editor do RDSQL. Ao escolher esta opção o conjunto anterior de instruções perder-se-á, se não for salvaguardado pela opção SAVE.

Run Executa o conjunto actual de instruções.

Modify Mostra o editor do RDSQL com o conjunto actual de instruções e permite que se corrijam erros de sintaxe, façam alterações e se acrescentem instruções.

Use-editor Permite editar as instruções num editor do sistema em vez do editor do RDSQL.

Output Envia os resultados da execução do conjunto actual de instruções para uma impressora, para um ficheiro do sistema ou para outro programa.

Choose Mostra uma lista dos ficheiros com comandos de RDSQL e permite selecionar um deles e torná-lo no conjunto actual de instruções.

Save Salvaguarda o conjunto actual de instruções num ficheiro de comandos. O RDSQL acrescenta a extensão ".sql" ao nome dado ao ficheiro.

Info Mostra informação sobre uma tabela da base de dados actual.

Drop Apaga um ficheiro de comandos RDSQL.

Exit Volta ao Main Menu.

2.3. Regras para escrever instruções de RDSQL.

O RDSQL intrepreta qualquer sucessão de espaços tabs ou NEWLINES como um único espaço. O encadeamento das instruções e/ou de parte das instruções é arbitrário. Assim pode dar ás instruções a forma que achar mais legível. Pode incluir comentários que serão ignorados pelo RDSQL desde que estejam entre chavetas "{...}".

2.4. Como Usar Bases de Dados e Tabelas.

As acções sobre bases de dados e tabelas podem ser executadas de 2 formas diferentes:

Utilizando instruções SQL, na opção Query-language do main Menu.

Utilizando as opções Database e Table para trabalhar bases de dados e tabelas, respectivamente.

Neste capítulo iremos usar a primeira forma (instruções SQL). As opções Database e Table executam algumas das funções disponíveis no RDSQL e o seu uso é directo.

2.5. Criação duma Base de Dados - CREATE DATABASE.

Como exercício vamos criar a base de dados **TESTE**.

Selecione a opção Query-language do Main Menu. -> Aparece-lhe o ecran CHOOSE DATABASE.

Saia do ecran CHOOSE DATABASE pressionando a tecla DEL. -> Aparece-lhe o ecran de menu RDSQL.

Selecione a opção New. Aparece-lhe o ecran de entrada de texto NEW.

Digite a instrução : **create database teste**

Pressione ESC -> Volta ao menu RDSQL.

Selecione a opção Run -> Criou a base de dados teste e tornou-a na base de dados actual, escrevendo o seu nome na terceira linha do ecran de menu RDSQL.

2.5.1. Selecção da Base de Dados Actual - DATABASE.

Vamos selecionar STORES como a base de dados actual.
Em vez da opção New selecione a opção User-edit para introduzir a instrução RDSQL através do editor com que costuma trabalhar habitualmente.

Insira a instrução: **database stores**

Grave o ficheiro e saia do editor. -> Aparece-lhe o menu RDSQL com o comando gravado pelo editor na secção de texto do ecran.

Selecione a opção Run -> Tornou STORES na base de dados actual escrevendo o seu nome na terceira linha do ecran de menu RDSQL.

2.5.2. Criação duma Tabela - CREATE TABLE

Para criar uma tabela é necessário utilizar a instrução **create database** e especificar o nome da tabela e o nome e o tipo de dados de cada coluna. Os tipos de dados permitidos pelo RDSQL são os seguintes:

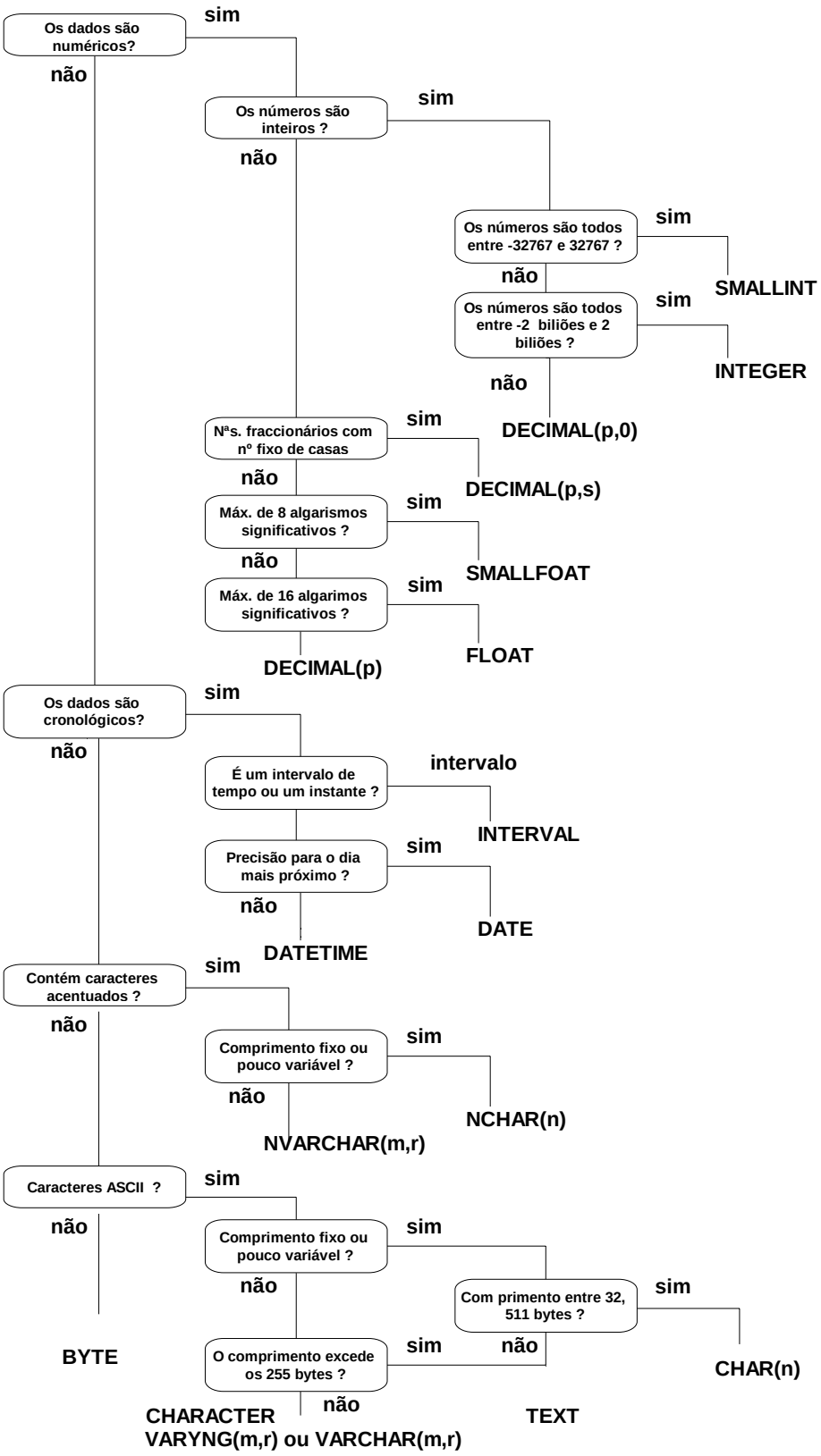
CHAR(n)	Alfanumérico de comprimento n.
SMALLINT	Numérico inteiro (binário) de -32767 a +32767.

INTEGER	Numérico inteiro (binário) de -2 147 483 647 a +2 147 486 647.
DECIMAL(m,n)	Numérico decimal de comprimento m e n casa decimais.
SMALLFLOAT	Numérico com vírgula flutuante
FLOAT	Numérico com vírgula flutuante de dupla palavra
MONEY(m,n)	Numérico decimal com cifrão ($) no início, de comprimento m e com n casas decimais
SERIAL(n)	Numérico inteiro sequêncial gerado automaticamente pelo RDSQL. A numeração começa no inteiro n.
DATE	Formato de data. Existe a possibilidade de seleccionar vários formatos de datas. O formato que iremos utilizar será: MM-DD-AAAA.
DATETIME	Ponto temporal com precisão definida.
INTERVAL	Intervalo de tempo com precisão definida.
VARCHAR (m,n)	Alfanumérico de tamanho variável **TEXT** Texto de comprimento indefinido *
BYTE	Qualquer formato binário. *

Ao definir as colunas é ainda necessário especificar se a coluna é de preenchimento obrigatório acrescentando NOT NULL a seguir à definição do tipo de dados da coluna.

* Só INFORMIX On Line

DIAGRAMA DE DECISÃO PARA ESCOLHA DOS TIPO DE DADOS



Noção de NULL. Quando não existe nenhum valor associado a uma coluna diz-se que essa coluna contém o valor NULL. Note que NULL é diferente de zero ou de espaço.

Vamos agora criar a tabela CLIENTES

Digite a seguinte instrução:

```
create table clientes ( num_clie    serial(101),
                        nome  char(20),
                        morada      char(20),
                        concelho    char(10),
                        cod_post    smallint,
                        telef       integer,
                        credito     decimal(15,2),
                        data_adm    date  )
```

e execute-a -> Criou a tabela CLIENTES.

2.5.3. Informações sobre uma tabela - INFO

Para obter informações sobre uma tabela pode-se utilizar a instrução INFO ou seleccionar a opção INFO do menu RDSQL. Esta última é a mais prática e será a que iremos usar.

Selecione a opção Info. -> Aparece o ecran INFO FOR TABLE.

Selecione a tabela clientes. -> Aparece o menu INFO - clientes.

Selecione a opção columns. -> Aparece uma lista, com uma linha por cada coluna da tabela CLIENTES, contendo o nome da coluna, o tipo de dados e a indicação da aceitação de NULL por essa coluna.

2.5.4. Alterações ao Formato das Tabelas - ALTER TABLE, RENAME TABLE, RENAME COLUMN

Imagine que pretendia alterar, na tabela **clientes**, a coluna NOME para NOT NULL, a coluna COD_POST para tipo de dados alfanumérico, suprimir a coluna CREDITO e acrescentar uma coluna SITUACAO com tipo de dados alfanumérico digite o seguinte query:

```
alter table clientes modify (nome char(20) not null, cod_post char(6));
alter table clientes drop (credito);
alter table cliente add (situacao char(10) before data_adm)
```

e execute-a, verificando se as alterações foram executadas utilizando a opção Info.

Apesar da tabela clientes já possuir as colunas correctamente descritas pretende-se alterar o nome da coluna CONCELHO para LOCALIDADE e mudar o nome da tabela para CLIENT-A. Digite as instruções :

```
rename column clientes.concelho to localidade;
rename table clientes to client-a
```

e execute-as, verificando se as alterações foram executadas utilizando a opção Info.

2.5.5. Supressão duma Tabela - DROP TABLE.

Vamos suprimir a tabela CLIENT-A. Digite o query :

```
drop table client-a
```

e execute-o confirmando com a opção Info.

2.5.6. Supressão duma Base de Dados - CLOSE DATABASE, DROP DATABASE

Antes de suprimir a base de dados TESTE tem de se fechar os seus ficheiros com a instrução CLOSE DATABASE. Digite o query:

```
close database;
drop database teste
```

e execute-o. -> Suprimiu a base de dados teste.

2.6. Manipulação de Dados

As instruções de RDSQL mais utilizadas na exploração duma base de dados são aquelas que manipulam os dados, quer dizer as que permitem consultar, inserir, apagar ou actualizar as linhas das tabelas.

2.6.1. Como Consultar uma Tabela - SELECT

A instrução SELECT é a mais comum, pois é utilizada para selecionar uma ou mais linhas de uma ou mais tabelas, sendo usada na execução de qualquer relatório. O formato da instrução SELECT com as suas cláusulas mais comuns é o seguinte:

```
SELECT      lista de colunas
FROM        lista de tabelas
WHERE       condições
ORDER BY    lista de colunas
```

Um exemplo:

```
SELECT      order_num, stock_num, quantity
FROM        items
WHERE       item_num=3
ORDER BY    manu_code
```

Iremos ver à medida que executarmos os exercícios seguintes qual o significado e quais as possibilidades da instrução e das suas cláusulas.

2.6.1.1. Gravação e Recuperação dum query.

Às vezes pode pretender iniciar um novo query mas salvaguardar o query actual para usar mais tarde. O RDSQL permite gravar o query actual num ficheiro com o nome à sua escolha ao qual o RDSQL acrescentará a extensão ".sql". Vejamos o seguinte exemplo:

Selecione o menu RDSQL e digite o query do exemplo anterior.

Selecione a opção Save. -> Aparece o ecran SAVE.

Introduza o nome que pretende dar ao ficheiro onde ficará guardado o query. -> Grava o ficheiro e retorna ao menu RDSQL.

Para recuperar o query:

Selecione a opção Choose do menu RDSQL. -> Aparece o menu CHOOSE com a lista dos ficheiros de comandos disponiveis.

Selecione o nome do ficheiro que gravou no exercício anterior. -> Retorna ao menu RDSQL com o query guardado no ficheiro selecionado pronto para ser executado.

Para apagar o ficheiro de comando:

Selecione a opção Drop do menu RDSQL. -> Aparece o menu DROP COMMAND FILE com a lista dos ficheiros de comandos disponiveis.

Selecione o nome do ficheiro que pretende apagar -> Apaga o ficheiro de comandos selecionado e retorna ao menu RDSQL.

2.6.1.2. Selecção de Algumas Colunas duma Tabela

Para selecionar algumas colunas, use SELECT, seguido dos nomes das colunas que pretende, ordenadas da esquerda para a direita, separadas por vírgulas. para especificar a tabela use FROM seguido do nome da tabela. Introduza a seguinte instrução:

```
SELECT      description, stock_num, manu_code
FROM        stock
```

Obtendo este relatório:

description	stock_num	num manu_code
baseball gloves	1	HRO
baseball gloves	1	HSK
baseball gloves	1	SMT
baseball	2	HRO
baseball bat	3	HSK
football	4	HSK
football	4	HRO
tennis racquet	5	NRG
tennis racquet	5	SMT
tennis racquet	5	ANZ
tennis ball	6	SMT
tennis ball	6	ANZ
basketball	7	HRO
volleyball	8	ANZ
volleyball net	9	ANZ

2.6.1.3. Selecção da Totalidade das Colunas duma Tabela

Para obter a totalidade das colunas, use SELECT seguido de * (asterisco), depois FROM e o nome da tabela. As colunas aparecerão, da esquerda para a direita, na mesma ordem com que se definiu a tabela. Introduza a seguinte instrução:

```
SELECT      *
FROM        stock
```

Obtendo este relatório:

stock_num	manu_code	description	unit_price	unit	unit_descr
1	HRO	baseball gloves	$250.00	case	10 gloves/case
1	HSK	baseball gloves	$800.00	case	10 gloves/cas
1	SMT	baseball gloves	$450.00	case	10 gloves/cas
2	HRO	baseball	$126.00	case	24/case
3	HSK	baseball bat	$240.00	case	12/case
4	HSK	football	$960.00	case	24/case
4	HRO	football	$480.00	case	24/case
5	NRG	tennis racquet	$28.00	each	each
5	SMT	tennis racquet	$25.00	each	each
5	ANZ	tennis racquet	$19.80	each	each
6	SMT	tennis ball	$36.00	case	24 cans/case
6	ANZ	tennis ball	$48.00	case	24 cans/case
7	HRO	basketball	$600.00	case	24/case
8	ANZ	volleyball	$840.00	case	24/case
9	ANZ	volleyball net	$20.00	each	each

2.6.1.4. Eliminação de linhas duplicadas

Se escrever DISTINCT logo a seguir à palavra SELECT elimina as linhas duplicadas do resultado. Introduza a seguinte instrução:

```
SELECT DISTINCT   unit_descr
FROM              stock
WHERE             unit='case
```

Obtendo este relatório:

unit_descr
10 gloves/case
24/case
12/case
24 cans/case

resultado mostra-nos as várias embalagens do tipo "case". Ainda que a mesma embalagem seja usada para mais do que um artigo só aparece uma vez no resultado.

2.6.1.5. Selecções por Condições.

Muitas vezes não se pretende obter todas as linhas duma tabela. Por exemplo pode querer saber quais são os artigos que tenham stock_num = 5 (raquetes de ténis). Introduza a seguinte instrução:

```
SELECT      *
FROM        stock
WHERE       stock_num=5
```

Obtendo este relatório:

stock_num	manu_code	description	unit_price	unit	unit_descr
5	ANZ	tennis racquet	$19.80	each	each
5	NRG	tennis racquet	$28.00	each	each
5	SMT	tennis racquet	$25.00	each	each

O resultado anterior mostra todos os artigos da tabela **stock** que tenham stock_num igual a 5. Se existisse só uma linha nestas condições só apareceria essa linha. Se nenhuma das condições da cláusula WHERE fosse satisfeita pelas linhas da tabela obter-se-ia no ecran um resultado vazio.

2.6.1.6. Selecção de Linhas Usando Dados Alfanuméricos.

Para selecionar linhas duma tabela usando condições sobre dados alfanuméricos, basta colocar esses dados entre " " " (plicas). Introduza e execute o query:

```
SELECT      customer_num, company, address1, address2
FROM        customer
WHERE       city="Redwood City"
```

Obtendo o resultado:

customer_num	company	address1	address2
104	Play Ball!	East Shopping Cntr.	422 Bay Road
108	Quinn's Sports	587 Alvarado	
110	AA Athletics	520 Topaz Way	
114	Sporting Place	947 Waverly Place	
117	Kids Korner	850 Lytton Court	

Repare na coluna address2. Esta coluna não possui valores em 4 linhas. Isto quer dizer que os valores da coluna nessas linhas é NULL (e não espaços como veremos a seguir).

2.6.1.7. Selecção de Linhas com Nulls.

Suponhamos que pretendemos a mesma informação do query anterior, mas neste caso para a lista dos clientes (customer) que tenham address2 com valor NULL. Introduza e execute o query:

```
SELECT      customer_num, company, address1, address2
FROM        customer
WHERE       adress2 is nulls
```

Obtendo o resultado:

customer_num	company	address1	address2
101	All Sports Supplies	213 Erstwild Court	
102	Sports Spot	785 Geary St	
105	Los Altos Sports	1899 La Loma Drive	
106	Watson & Son	1143 Carver Place	
107	Athletic Supplies	41 Jordan Avenue	
108	Quinn's Sports	587 Alvarado	
110	AA Athletics	520 Topaz Way	
111	Sports Center	3199 Sterling Court	
112	Runners & Others	234 Wyandotte Way	
113	Sportstown	654 Oak Grove	
114	Sporting Place	947 Waverly Place	
115	Gold Medal Sports	776 Gary Avenue	
116	Olympic City	1104 Spinosa Drive	
117	Kids Korner	850 Lytton Court	
118	Blue Ribbon Sports	5427 College	

Observe que na cláusula WHERE escreve-se " is null " e não " = null " que é incorrecto. Para selecionar linhas sem Nulls escreva " is not null".

2.6.1.8. Selecção de Linhas Usando Desigualdades

Até agora só selecionámos linhas usando igualdades na cláusula WHERE. Também podemos fazê-lo usando qualquer tipo de comparação, usando os seguintes simbolos:

```
!= ou <> diferente
>     maior
>=    maior ou igual
<     menor
<=    menor ou igual
```

Vamos selecionar os artigos que têm um valor unitário inferior a 100.

```
SELECT      *
FROM        stock
WHERE       unit_price < 100
```

Obtem-se o resultado:

stock_num	menu_code	description	unit_price	unit	unit_descr
5	NRG	tennis racquet	$28.00	each	each
5	SMT	tennis racquet	$25.00	each	each

5	ANZ	tennis racquet	$19.80	each	each
6	SMT	tennis ball	$36.00	case	24 cans/case
6	ANZ	tennis ball	$48.00	case	24 cans/case
9	ANZ	volleyball net	$20.00	each	each

Se quiser saber quais os artigos que têm preço unitário entre 100 e 200 inclusive use o operador BETWEEN.

```
SELECT      *
FROM  stock
WHERE unit_price BETEWEN 100 AND 200
```

Obtendo o resultado:

stock_num	manu_code	description	unit_price	unit	unit_descr
1	HRO	baseball gloves	$250.00	case	10 gloves/case
2	HRO	baseball	$126.00	case	24/case
3	HSK	baseball bat	$240.00	case	12/case

2.6.1.9. Selecção de Linhas usando condições pela Negativa

Pode usar a negação de qualquer condição escrevendo NOT antes desta desta. Para clarificar a que condição o NOT se aplica, use parêntesis. Por exemplo:

NOT A AND B OR C significa ((NOT A) AND B) OR C

e se pretender condições diferentes tem então de usar parêntises obrigatóriamente.

Com os operadores maior que, menor que ou igual, NOT deve preceder a condição. Por exemplo deve usar:

WHERE NOT UNIT_PRICE < 100

e não

WHERE UNIT-PRICE NOT < 100

Só nos casos que se enumeram a seguir é que NOT não precede a condição:

NOT NULL; NOT LIKE; NOT IN; NOT BETWEEN.

Exemplos:

Este query seleciona todos os artigos cujo preço unitário não esteja compreendido entre 100 e 300. select * from stock where unit_price not between 100 and 300 obtendo-se o relatório:

stock_num	manu_cod	description	unit_price	unit	unit_descr
1	HSK	baseball gloves	$800.00	case	10 gloves/case
1	SMT	baseball gloves	$450.00	case	10 gloves/case
4	HSK	football	$960.00	case	24/case
4	HRO	football	$480.00	case	24/case
5	NRG	tennis racquet	$28.00	each	each
5	SMT	tennis racquet	$25.00	each	each
5	ANZ	tennis racquet	$19.80	each	each
6	SMT	tennis ball	$36.00	case	24 cans/case
6	ANZ	tennis ball	$48.00	case	24 cans/case
7	HRO	basketball	$600.00	case	24/case
8	ANZ	volleyball	$840.00	case	24/case
9	ANZ	volleyball net	$20.00	each	each

O query seguinte seleciona os artigos cujo preço unitário não seja inferior a 400 e que não pertençam à categoria de futebol.

```
select      *
from        stock
where       not unit_price <  400
            and   not description = 'football'
```

Obtendo-se o relatório:

stock_num	manu_code	description	unit_price	unit	unit_descr
1	HSK	basebal gloves	$800.00	case	10 gloves/case
1	SMT	basebal gloves	$450.00	case	10 gloves/case
7	HRO	basketball	$600.00	case	24/case
8	ANZ	volleyball	$840.00	case	24/case

2.6.1.10. Selecção com linhas Ordenadas

Para ordenar o resultado de um query por determinadas colunas, use ORDER BY, seguido dos nomes das colunas que controlam a ordenação.

As linhas do resultado aparecerão ordenadas por ordem ascendente pela coluna que especificou com ORDER BY (por defeito assume por ordem ascendente). Se pretender que as linhas do resultado sejam ordenadas por ordem descendente escreva ORDER BY -nome da coluna- DESC.

A primeira coluna a seguir a ORDER BY será a primeira a ser ordenada. A segunda coluna da cláusula ORDER BY será ordenada dentro da primeira e assim por diante. Por exemplo:

```
select      order_date,  customer_num,  order_num
from        orders
where       paid_date is not null
order by order_date desc,customer_num
```

obtendo-se o relatório:

order_date	customer_num	order_num
06/06/1984	104	1013
06/06/1984	110	1015
06/05/1984	104	1011
06/05/1984	106	1014
06/05/1984	115	1010
06/04/1984	104	1003
06/04/1984	110	1008
06/04/1984	111	1009
06/04/1984	116	1005
06/01/1984	101	1002
06/01/1984	104	1001

Note que a coluna ORDER_DATE está ordenada por ordem descendente e que dentro desta CUSTOMER_NUM está ordenada por ordem ascendente.

2.6.1.11. Selecção de Linhas por Partes de Valores

Para selecionar dados de que só se conheça parte do seu valor, use LIKE numa cláusula WHERE com um símbolo para os dados desconhecidos.

"**%**" Significa "conjunto de zero ou mais caracteres"

"**_**" Significa "qualquer caracter".

Use mais do que um destes símbolos em sequência para representar o número exacto de caracteres que faltam. As instruções que se seguem querem dizer, "Mostre todos os artigos cuja descrição inclua baseball".

```
select      stock_num, manu_code, description
from        stock
where       description like 'baseball%'
```

obtendo-se o relatório:

stock_num	manu_code	description
1	HRO	baseball gloves
1	HSK	baseball gloves
1	SMT	baseball gloves
2	HRO	baseball

3	HSK	baseball bat

AS instruções que se seguem querem dizer, "Mostre todos os artigos cuja descrição tenha exactamente 10 caracteres e termine em ball".

```
select      stock_num, manu_code, description
from        stock
where       description like '_____ball'
```

obtendo-se o relatório:

stock_num	manu_code	description
7	HRO	basketball
8	ANZ	volleyball

Pode usar "%" mais do que uma vez numa expressão. Por exemplo :

```
select      stock_num, manu_code, description
from        stock
where description like '%o%b%'
```

Encontrará todos os artigos cuja descrição inclua as letras "o" e "b" e por esta ordem.

stock_num	manu_code	description
4	HSK	football
4	HRO	football
8	ANZ	volleyball
9	ANZ	volleyball net

Pode usar "%" e "_" juntos numa expressão. Por exemplo:

```
select      stock_num, manu_code, description
from        stock
where description like '___k%'
```

Encontrará todos os artigos cuja descrição inclua a letra "k" na quarta posição.

stock_num	manu_code	description
7	HRO	basketball

Pode usar NOT antes de LIKE para especificar os valores que quer excluir. Por exemplo :

```
select      stock_num, manu_code, description
from        stock
```

```
where description not like '%ball%'
```

Encontrará todos os artigos cuja descrição não inclua "ball".

stock_num	manu_code	description
5	NRG	tennis racquet
5	SMT	tennis racquet
5	ANZ	tennis racquet

2.6.1.12. Selecção de Linhas com Condições Compostas

Pode utilizar condições compostas ligando as expressões com AND, OR e IN.

Usando AND

O query seguinte selecciona as encomendas e respectivos clientes do primeiro semestre de 1984 e que ainda não foram pagas.

```
select      order_num,order_date,customer_num,
            ship_instruct
from        orders
where       order_date  between '01/01/1984' and '06/30/1984'
            and paid_date is null
```

obtendo-se o relatório:

order_num	order_date	customer_num	ship_instruction
1004	06/04/1984	106	ring bell twice
1006	06/04/1984	112	after 10 am
1007	06/04/1984	117	
1012	06/05/1984	117	

Usando OR

O query seguinte selecciona as encomendas e respectivos clientes que tenham peso superior a 20 ou montante de expedição superior a 15.

```
select      order_num,  order_date,  customer_num,
            ship_weight, ship_charge
from        orders
where       ship_weight > 20 or ship_charge > 15
```

obtendo-se o relatório:

order_num	order_date	customer_num	ship_weight	ship_charge
1001	06/01/1984	104	20.40	$10.00
1002	06/01/1984	101	50.60	$15.30
1003	06/04/1984	104	35.60	$10.80
1004	06/04/1984	106	95.80	$19.20
1005	06/04/1984	116	80.80	$16.20
1006	06/04/1984	112	70.80	$14.20
1007	06/04/1984	117	125.90	$25.20
1008	06/04/1984	110	45.60	$13.80
1009	06/04/1984	111	20.40	$10.00
1010	06/05/1984	115	40.60	$12.30
1012	06/05/1984	117	70.80	$14.20
1013	06/06/1984	104	60.80	$12.20
1014	06/05/1984	106	40.60	$12.30
1015	06/06/1984	110	20.60	$6.30

Usando IN

As vezes é mais prático usar a instrução IN em vez de múltiplas instruções OR para procurar vários valores duma coluna. Por exemplo :

em vez de:

```
select      order_num,  order_date,  customer_num,
            ship_instruct
from        orders
where customer_num = 104 or customer_num = 106
            or customer_num = 110
```

use:

```
select      order_num,  order_date,  customer_num,
            ship_instruct
from        orders
where customer_num in (104, 106, 110)
```

obtendo-se o relatório:

order_num	order_date	customer_num	ship_instruct
1001	06/01/1984	104	ups
1003	06/04/1984	104	via ups
1004	06/04/1984	106	ring bell twice
1008	06/04/1984	110	closed Monday
1011	06/05/1984	104	ups
1013	06/06/1984	104	via ups

1014	06/05/1984	106	ring bell, kick door loudly
1015	06/06/1984	110	closed Mon

Quando IN for utilizado devem ser especificados, pelo menos, dois valores dentro dos parêntesis. Por exemplo, WHERE CUSTOMER_NUM IN (104) é incorrecto. Neste caso terá de escrever WHERE CUSTOMER_NUM = 104.

Também pode especificar os valores que pretender excluir usando IN com NOT. Por exemplo WHERE CUSTOMER_NUM NOT IN (104, 106,10).

Mais Alguns Exemplos

Se usar AND e OR utilize parentesis para claro o agrupamento das suas comparações. As condições dentro dos parentesis serão executadas primeiro. Caso não use parentesis lembre-se que todas as condições ligadas por AND serão executadas primeiro, e só depois serão executadas as condições ligadas por OR. Por exemplo o query:

```
select      order_num,  order_date,  customer_num,
            ship_weight,  ship_charge
from        orders
where (customer_num > 104  and ship_weight > 20)
            or ship_charge > 15
```

produzirá o mesmo resultado com ou sem parentesis porque AND é calculado antes de OR:

order_num	order_date	customer_num	ship_weight	ship_charge
1002	06/01/1984	101	50.60	$15.30
1004	06/04/1984	106	95.80	$19.20
1005	06/04/1984	116	80.80	$16.20
1006	06/04/1984	112	70.80	$14.20
1007	06/04/1984	117	125.90	$25.20
1008	06/04/1984	110	45.60	$13.80
1009	06/04/1984	111	20.40	$10.00
1010	06/05/1984	115	40.60	$12.30
1012	06/05/1984	117	70.80	$14.20
1014	06/05/1984	106	40.60	$12.30
1015	06/06/1984	110	20.60	$6.30

Mas se movermos os parentesis:

```
select      order_num,  order_date,  customer_num,
            ship_weight, ship_charge
from        orders
```

```
where       customer_num > 104  and
            (ship_weight > 20  or ship_charge > 15)
```

Obtendo-se o resultado:

order_num	order_date	customer_num	ship_weight	ship_charge
1004	06/04/1984	106	95.80	$19.20
1014	06/05/1984	106	40.60	$12.30
1008	06/04/1984	110	45.60	$13.80
1015	06/06/1984	110	20.60	$6.30
1009	06/04/1984	111	20.40	$10.00
1006	06/04/1984	112	70.80	$14.20
1010	06/05/1984	115	40.60	$12.30
1005	06/04/1984	116	80.80	$16.20
1007	06/04/1984	117	125.90	$25.20
1012	06/05/1984	117	70.80	$14.20

Se for necessário também se podem usar vários níveis de parentesis.

2.6.1.13. Usando Valores Calculados

Pode escrever operações aritméticas com qualquer um dos seguintes operadores:

```
            +     adição
            -     subtracção
            *     multiplicação
            /     divisão
```

Este query

```
select      stock_num,  manu_code,  (total_price/quantity)
from        items
where order_num = 1005
```

produz o seguinte relatório

stock_num	manu_code	(expression)
5	NRG	$28.00
5	ANZ	$19.80
6	SMT	$36.00
6	ANZ	$48.00

A expressão TOTAL_PRICE/QUANTITY calcula o "preço unitário".

Algumas Notas.

EXPRESSION: O RDSQL não sabe qual o nome a dar à coluna calculada. Por isso chama-lhe EXPRESSION.

NULLS: O resultado de qualquer operação aritmética sobre uma coluna com uma ou várias linhas com o valor NULL será NULL para essa(s) linha(s). (Lembre-se que NULL não é zero. NULL significa que RDSQL não conhece nenhum valor para essa coluna e essa linha. Logo o resultado duma operação sobre um valor desconhecido é ainda um valor desconhecido.)

2.6.1.14. Selecção Por valores Calculados

Para selecionar dados através de valores calculados use expressões aritméticas na cláusula WHERE. Por exemplo:

```
select      stock_num,  manu_code,  (total_price/quantity)
from        items
where       order_num = 1005 and total_price/quantity > 30
```

produz o seguinte relatório

stock_num	manu_code	(expression)
6	SMT	$36.00
6	ANZ	$48.00

Neste caso selecionámos os preços unitários superiores a 30 da encomenda número 1005.

2.6.1.15. Ordenação de Linhas por Valores Calculados

Para selecionar linhas ordenadas por valores calculados refira-se à coluna do valor calculado pelo seu número de coluna. Por exemplo:

```
select      stock_num,  manu_code,  (total_price/quantity)
from        items
where       order_num = 1005 order by 3
```

obtendo o seguinte relatório

stock_num	manu_code	(expression)
5	ANZ	$19.80

5	NRG	$28.00
6	SMT	$36.00
6	ANZ	$48.00

ORDER BY 3 significa "ordenado pela terceira coluna specificada na cláusula SELECT".(Não escreva ORDER BY -espressão aritmética- pois não funciona).

NOTA: Os valores NULL aparecerão nas últimas ou primeiras linhas conforme a ordenação seja ascendente ou descendente, respectivamente.

2.6.2. Inserção de Linhas num Tabela - INSERT

Até ao fim deste capítulo a maior parte dos exercícios serão feitos sobre a tabela FABRICANTE que é uma réplica da tabela MANUFACT.

Verifique se a tabela já está criada. Se não estiver crie-a.

2.6.2.1. Inserção de uma Linha numa Tabela

Para inserir uma Linha na Tabela FABRICANTE introduza a seguinte instrução:

```
insert into fabricante (manu_code, manu_name)
            values ('ZZZ', 'Zacarias')
```

Para confirmar que esta linha foi inserida selecione a totalidade das linhas e colunas da tabela, devendo obter um resultado semelhante ao que se segue:

manu_code	manu_name
ZZZ	Zacarias

NOTA: Num caso como este em que estamos a inserir valores para todas as colunas da tabela não é obrigatório preencher os nomes das colunas a seguir ao nome da tabela. Experimente inserir mais uma linha na tabela, sem escrever os nomes das colunas.

2.6.2.2. Inserção de Valores numa Coluna duma Tabela

Se pretender inserir valores nalgumas colunas da tabela especifique quais as colunas e os respectivos valores que pretende inserir. Por exemplo:

```
insert into fabricante (manu_code) values ('ZAP')
```

Para confirmar que esta linha foi inserida selecione a totalidade das linhas e colunas da tabela, devendo obter um resultado semelhante ao que se segue:

manu_code	manu_name
ZZZ	Zacarias
ZAP	

NOTA: As colunas que não foram especificadas receberão o valor NULL.

2.6.2.3. Como Copiar Linhas duma Tabela para Outra

Suponha que pretende inserir na tabela FABRICANTE algumas das linhas de MANUFACT. A instrução seguinte copia de MANUFACT as linhas cujo MANU_CODE é diferente de ANZ e insere-as em FABRICANTE.

```
insert      into fabricante
select      *
from        manufact
where       manu_code <> ÁNZ'
```

Para confirmar que estas linhas foram inseridas selecione a totalidade das linhas e colunas da tabela, devendo obter um resultado semelhante ao que se segue:

manu_code	manu_name
ZZZ	Zacarias
ZAP	
SMT	Smith
NRG	Norge
HSK	Husky
HRO	Hero

NOTA: A uma instrução SQL contida noutra instrução de SQL chama-se um SUBQUERY (no nosso caso INSERT contém SELECT). No capítulo 4 "Aprofundamento das possibildades do SELECT" trataremos dos subqueries mais em pormenor.

2.6.3. Alteração dos Valores duma Tabela - UPDATE

O query seguinte modifica o valor da coluna MANU_NAME para 'Zacar', na linha com MANU_CODE igual a ZZZ.

```
update fabricante
      set  manu_name = 'Zacar' where manu_code = 'ZZZ'
```

Para confirmar que estas linhas foram alteradas selecione a totalidade das linhas e colunas da tabela, devendo obter um resultado semelhante ao que se segue:

manu_code	manu_name
ZZZ	Zacar
ZAP	
SMT	Smith
NRG	Norge
HSK	Husky
HRO	Hero

A instrução UPDATE é muito versátil. Suponha que pretende aumentar em 10% os artigos com STOCK_NUM igual a 4, na tabela STOCK. Introduza e execute o query:

```
update stock
      set  unit_price  =  unit_price  *  1.1
      where  stock_num = 4
```

Para confirmar que estas linhas foram alteradas selecione a totalidade das linhas e colunas da tabela, e compare o resultado que se segue com o obtido no capítulo 2.3.

stock_num	manu_code	description	unit_price	unit	unit_descr
1	HRO	baseball gloves	$250.00	case	10 gloves/case
1	HSK	baseball gloves	$800.00	case	10 gloves/case
1	SMT	baseball gloves	$450.00	case	10 gloves/case
2	HRO	baseball	$126.00	case	24/case
3	HSK	baseball bat	$240.00	case	12/case
4	HSK	football	$1056.00	case	24/case
4	HRO	football	$528.00	case	24/case
5	NRG	tennis racquet	$28.00	each	each
5	SMT	tennis racquet	$25.00	each	each
5	ANZ	tennis racquet	$19.80	each	each

6	SMT	tennis ball	$36.00	case	24 cans/case
6	ANZ	tennis ball	$48.00	case	24 cans/case
7	HRO	basketball	$600.00	case	24/case
8	ANZ	volleyball	$840.00	case	24/case
9	ANZ	volleyball net	$20.00	each	each

NOTA: Para alterar várias colunas basta introduzi-las na cláusula SET, separadas por vírgulas.

2.6.4.Supressão de Linhas duma Tabela - DELETE

Para apagar uma ou mais linhas duma tabela use a instrução DELETE. Por exemplo:

```
delete from fabricante
      where manu_code = 'ZZZ' or manu_code = 'ZAP'
```

Para confirmar que estas linhas foram apagadas selecione a totalidade das linhas e colunas da tabela, e compare o resultado que se segue com o obtido no capítulo 2.3.2.

manu_code	manu_name
SMT	Smith
NRG	Norge
HSK	Husky
HRO	Hero

NOTA: Ao usar a instrução DELETE tem de ter muita atenção com a cláusula WHERE, pois pode apagar linhas da tabela que não quisesse ou até pode apagar todas as linhas da tabela se se esquecer de usar a clausula WHERE.

2.7. Indices

Quando nas Bases de Dados as tabelas têm um número de linhas muito elevado e/ou possuem queries muito complexos o tempo de resposta degrada-se muito. A forma de melhorar os tempos de resposta passa pela criação de indices sobre colunas criteriosamente escolhidas.

São dois os grandes objectivos ao criar indices sobre colunas duma tabela:

- Aumentar a velocidade da ordenação das linhas da tabela.
- Optimizar a execução dos queries.

Como quando se alteram as linhas duma tabela também se alteram os indices da tabela, os processos de alteração dum grande número de linhas podem tornar-se muito pesados e portanto demorados. No entanto nos casos em que os dados são alterados e inseridos interactivamente, isto é linha a linha, este problema normalmente não se põe.

2.7.1. Criação dum indice - CREATE INDEX

Suponha que pretende criar na tabela ITEMS, um indice composto sobre as colunas ITEM_NUM e ORDER_NUM e que não pretende entradas duplicadas.

Introduza o query:

```
create distinct index ind3_ite on items (item_num, order_num)
```

e execute-o. Se pretender confirmar que o indice foi criado utilize a opção Info do menu RDSQL, e a seguir selecione a opção Indexes.

2.7.2. Como Retirar um Indice - DROP INDEX

Suponha que o índice criado anteriormente já não é necessário e que deve ser removido. Introduza o query:

```
drop index  ind3_ite
```

e execute-o. Se pretender confirmar que o indice foi retirado utilize a opção Info do menu RDSQL, e a seguir selecione a opção Indexes.

2.7.3. Indices Cluster - ALTER INDEX

Como tanto em UNIX como em DOS os dados são extraídos do disco em blocos, quanto maior for o número de linhas da tabela que estejam fisicamente no mesmo bloco e simultâneamente na mesma ordem do indice, mais rápido será o processo de pesquisa pelo indice.

É possivel, pelo menos temporàriamente, ter uma tabela com ordenação física igual à ordenação dos indices. A este processo chama-se CLUSTERING. Para obter um indice cluster pode criá-lo com a cláusula CLUSTER (CREATE INDEX CLUSTER nome-indice), ou então se o indice já existir, alterá-lo para cluster com a instrução ALTER INDEX TO CLUSTER.

À medida que são alteradas ou inseridas novas linhas na tabela o indice perde o seu estado de cluster. Quando precisar de novo do indice cluster volte a introduzir a instrução ALTER INDEX TO CLUSTER.

Como uma tabela só pode ter uma ordem física só se pode ter um indice cluster de cada vez. Vejamos os exemplos seguintes:

Para tornar cluster o indice ST_MAN_IX já existente da tabela ITEMS introduza o query:

```
alter index st_man_ix to cluster
```

Se se pretender ordenar fisicamente a tabela por outro indice tem de se libertar o cluster do indice ST_MAN_IX da tabela ITEMS, antes de se introduzir o novo cluster. Introduza as instruções seguintes para tornar cluster o indice I_O_NUM_IX:

```
alter index  st_man_ix  to  not  cluster;
alter index i_o_num_ix to cluster
```

2.7.4. Alguma Estratégias de Indexação

Não crie indices para tabelas com menos de 200 linhas.

Não crie indices sobre colunas com um pequeno número de valores diferentes. Nestes casos deve usar um indice cluster.

Se a cláusula WHERE duma instrução SELECT tem condições sobre uma única coluna crie um indice sobre essa coluna. Se existem condições sobre várias colunas crie um indice composto sobre as colunas consideradas.

2.8. Aprofundamento das Possibilidades do SELECT

No ponto 2.3 descreveram-se as características e possibilidades básicas da instrução SELECT. Neste ponto iremos apresentar algumas capacidades adicionais do SELECT, nomeadamente:

Funções agregadas para obter valores para um grupo de linhas, tais como a média dos valores duma coluna ou a soma dos valores duma coluna.

Funções sobre datas que permitem manipular datas completas ou partes de datas.

Especificar cláusulas GROUP BY, HAVING para indicar:

- Como pretende que as linhas do resultado sejam agrupadas (GROUP BY).
- Impor uma condição que as linhas do resultado,como grupo, devem respeitar (HAVING).

Operador UNION que permite combinar dois queries num só, reunindo os resultados de cada um deles.

Obter resultados de duas ou mais tabelas (Join).

Utilizar instruções SELECT dentro de cláusulas WHERE (Subqueries).

Criar uma tabela temporária que contenha o resultado dum query (cláusula INTO TEMP).

2.8.1. Funções Agregadas

Uma função agregada ou de coluna produz um único valor para um grupo de linhas. Por exemplo suponha que pretende saber qual o artigo mais caro da tabela STOCK.

```
select      max(unit_price)
from  stock
```

Obtendo como resultado:

(max)
$1056.00

A função agregada MAX foi aplicada uma vez a todos os valores de UNIT_PRICE. Como resultado obteu-se uma linha e um único valor.

O RDSQL admite as seguintes funções de coluna:

AVG Calcula a média de todos os valores da coluna especificada que satisfaçam a cláusula WHERE. Só se pode aplicar AVG a colunas numéricas.

MAX Selecciona o valor máximo contido na coluna especificada e que pertençam a linhas que satisfaçam a cláusula WHERE.

MIN Selecciona o valor mínimo contido na coluna especificada e que pertençam a linhas que satisfaçam a cláusula WHERE.

COUNT Calcula o número de linhas que satisfaçam a cláusula WHERE.

SUM Soma todos os valores da coluna que satisfaçam a cláusula WHERE.

DISTINCT Pode utilizar a palavra DISTINCT imediatamente antes do nome da coluna, nas funções AVG, SUM e COUNT para que as linhas duplicadas não sejam consideradas para o resultado.

Por exemplo pode querer saber quantos tipos de artigos diferentes existem na tabela stock, usando o seguinte query:

```
select      count(distinct stock_num)
from        stock
```

Obtendo o resultado:

(count)
9

Pode usar mais do que uma função de coluna numa instrução SELECT. Por exemplo se pretender saber qual o valor médio duma encomenda e ao mesmo tempo qual foi o valor máximo introduza o query seguinte:

```
select      avg(total_price), max (total_price)
from        items
where order_num=1003
```

Obtendo o resultado:

(avg)	(max)
$319.67	$840.00

2.8.2. Funções sobre Datas

As funções sobre datas são:

DATE() Transforma uma expressão num valor de tipo data.

DAY() Extrai duma data a parte correspondente ao dia.

MONTH() Extrai duma data a parte correspondente ao mês.

YEAR() Extrai duma data a parte correspondente ao ano.

WEEKDAY() Extrai duma data um inteiro que representa o dia da semana. Este inteiro varia de 0 (domingo) a 6 (sábado).

Vejamos o exemplo seguinte, onde se aplicam algumas destas funções:

```
select      order_num, order_date, customer_num,
            day(order_date), paid_date
from        orders
where paid_date > date('06/01/1984') + 30
```

Obtendo-se o resultado:

order_num	order_date	customer_num	(expression)	paid_date
1002	06/01/1984	101	1	07/03/1984
1008	06/04/1984	110	4	07/17/1984
1009	06/04/1984	111	4	07/21/1984
1014	06/05/1984	106	5	07/18/1984

2.8.2.1. Operações Aritméticas com Datas

É possível executar adições e subtracções entre datas e períodos de tempo expressos em dias. No exemplo anterior temos na cláusula WHERE a soma duma data com um período de tempo. Note que a adição de duas datas completas dá um resultado incorrecto (experimente). Já a subtração de duas datas completas é possivel e obtem-se como resultado um período de tempo em dias. Por exemplo o query:

```
select      order_num,  order_date, customer_num,
            (paid_date  - order_date),   paid_date
from        orders
where paid_date  > date('06/01/1984') + 30
```

Obtendo_se o resultado:

order_num	order_date	customer_num	(expression)	paid_date
1002	06/01/1984	101	32	07/03/1984
1008	06/04/1984	110	43	07/17/1984
1009	06/04/1984	111	47	07/21/1984
1014	06/05/1984	106	43	07/18/1984

em que temos na coluna "expression" o número de dias que decorreram entre a encomenda e o respectivo pagamento.

2.8.3. A Cláusula GROUP BY

Quando se utilizam funções de coluna (agregadas) obtem-se como resultado um valor. Usando a cláusula GROUP BY a função aplica-se a cada um dos grupos definidos pela cláusula, logo obtêm-se no resultado tantos valores quanto o número de grupos existentes na tabela.

A cláusula GROUP BY permite que se seleccionem as características de grupos de linhas em vez das linhas individualmente. Quando se especifica a cláusula GROUP BY o INFORMIX-SQL divide as linhas selecionadas (isto é as que satisfazem a cláusula WHERE) em grupos que tenham o mesmo valor numa ou mais colunas. A seguir cada grupo é processado para produzir um único resultado (normalmente aplicando uma função de coluna). Uma ou mais colunas podem ser especificadas na cláusula GROUP BY para agrupar as linhas da tabela. As colunas seleccionadas pela instrução SELECT deterão propriedades do grupo de linhas e não propriedades de linhas individuais.

Por exemplo se pretender saber qual o montante total de cada encomenda introduza e execute o seguinte query:

```
select      order_num,  sum(total_price)
from        items
group by order_num
```

Obtendo o resultado:

order_num	(sum)
1001	$250.00
1002	$1200.00
1003	$959.00
1004	$2126.00

1005	$562.00
1006	$498.00
1007	$1696.00
1008	$940.00
1009	$450.00
1010	$84.00
1011	$99.00
1012	$1040.00
1013	$143.80
1014	$1440.00
1015	$450.00

Vejamos agora um exemplo em que se especificam duas colunas na cláusula GROUP BY. Neste exemplo pretende-se agrupar as encomendas por cliente e mês de pagamento da encomenda e obter o peso correspondente das encomendas agrupadas.

```
select      customer_num, month(paid_date), sum(ship_weight)
from        orders
group by customer_num, 2
order by customer_num, 2
```

Obtendo-se o resultado:

customer_num	(expression)	(sum)
101	7	50.60
104	6	127.20
106		95.80
106	7	40.60
110	6	20.60
110	7	45.60
111	7	20.40
112		70.80
115	6	40.60
116	6	80.80
117		125.90
117		70.80

NOTA: Se existirem valores NULLS na coluna especificada na cláusula GROUP BY o INFORMIX-SQL considerará como um grupo diferente cada linha que tenha NULLS nessa coluna.

2.8.4. A Cláusula HAVING

Pode-se usar a cláusula HAVING para especificar condições de pesquisa sobre os grupos selecionados pela cláusula GROUP BY. HAVING significa que se pretende só aqueles grupos que

satisfazem a condição da cláusula HAVING. Assim as condições da cláusula HAVING devem testar propriedades de cada grupo e não propriedades de linhas individuais do grupo.

A cláusula HAVING escreve-se imediatamente a seguir à cláusula GROUP BY e pode conter o mesmo tipo de condições da cláusula WHERE, desde que um dos membros da condição contenha uma unção de grupo.

Por exemplo se quiser saber os montantes totais das encomendas de artigos com STOCK_NUM inferior a 8 e com montantes totais superiores a 500 introduza e execute o seguinte query:

```
select      order_num,  sum(total_price)
from        items
group  by order_num
having max(stock_num) < 8
      and sum(total_price) > 500
```

Obtendo o resultado:

order_num	(sum)
1002	$1200.00
1004	$2126.00
1005	$562.00
1007	$1696.00
1014	$1440.00

2.8.5. Operador UNION

Ao utilizar o operador UNION pode combinar duas ou mais instruções SELECT para constituir uma única instrução SELECT. Quando o INFORMIX-SQL encontra o operador UNION, cria uma "tabela intermédia" que é o resultado de cada uma das instruções SELECT ligadas pelo operador UNION e a seguir combina todas as "tabelas intermédias", eliminando as linhas duplicadas. Pode usar qualquer das cláusulas e técnicas que aprendeu até agora na codificação das instruções SELECT, incluindo ORDER BY.

Por exemplo se pretender listar as encomendas de artigos fabricados pelo fabricante com código 'HRO' e simultâneamente as encomendas com montante superior a 800 introduza execute o seguinte query:

```
select      order_num, manu_code
```

```
from        items
where manu_code  = 'HRO'
union
select      order_num,  manu_code
from        items
where total_price > 800
```

obtendo o resultado:

order_num	manu_code
1001	HRO
1004	HRO
1007	HRO
1014	HRO
1002	HSK
1003	ANZ
1008	ANZ
1012	ANZ
1014	HSK

Repare que, como não se usou a cláusula ORDER BY, as linhas do resultado não estão ordenadas e que aparecem em primeiro lugar as linhas correspondentes à primeira instrução SELECT. Para obter o resultado ordenado utilize o query:

```
select      order_num, manu_code
from        items
where manu_code = "HRO"
union
select      order_num,  manu_code
from        items
where total_price > 800 order by 1
```

NOTA: Qualquer cláusula ORDER BY deve aparecer sempre depois da a instrução SELECT do UNION. Para especificar as colunas pelas quais deve ser ordenado o resultado use o número de ordem da coluna na cláusula SELECT (num UNION não se pode usar o nome das colunas para este fim).

Se pretender manter no resultado as linhas duplicadas escreva ALL a seguir a UNION. No exemplo anterior teria:

```
select      order_num, manu_code
```

```
from        items
where manu_code  = 'HRO'
union all
select      order_num, manu_code
from        items
where total_price > 800 order by 1
```

Obtendo o resultado:

order_num	manu_code
1001	HRO
1002	HSK
1003	ANZ
1004	HRO
1004	HRO
1004	HRO
1007	HRO
1007	HRO
1007	HRO
1007	HRO
1008	ANZ
1012	ANZ
1014	HRO
1014	HSK

No UNION os valores das colunas especificadas num SELECT devem ter o mesmo tipo de dados e o mesmo comprimento dos correspondentes valores das colunas dos outros SELECTS.

2.8.6. JOIN - Selecção de Linhas em mais de uma Tabela

Nos exemplos de instrucções SELECT utilizados até agora, a informação requerida residiu sempre numa única tabela. Muitas vezes a informação pretendida não reside numa única tabela. Ao formar uma "tabela resultado", pode precisar de algumas colunas de uma tabela e doutras colunas de outra tabela. Pode utilizar uma instrução SELECT para selecionar e juntar (Join) valores de colunas de duas ou mais tabelas.

No JOIN os valores das colunas da linha duma tabela são combinados com valores de colunas de outra tabela para formarem uma única linha da "tabela resultado". O INFORMIX-SQL pesquisa ambas

as tabelas especificadas no SELECT de "join" para selecionar valores de todas as linhas que satisfaçam as condições da cláusula WHERE.

Por exemplo para obter a lista das encomendas e respectivos números e nomes dos clientes precisa de colunas da tabela ORDERS (order_num, customer_num) e da tabela CUSTOMER (company). Para executar este JOIN escreva a instrução SELECT com as duas tabelas na claúsula FROM e quando existirem colunas da tabelas diferentes com o mesmo nome devem ser prefixadas com o nome da tabela para se distinguirem. Introduza o query:

```
select       order_num,   orders.customer_num,   company
from        orders, customer
where       orders.paid_date is null and
            orders.customer_num = customer.customer_num
```

Obtendo o resultado:

order_num	customer_num	company
1004	106	Watson & Son
1006	112	Runners & Others
1007	117	Kids Korner
1012	117	Kids Korner

Nos casos em que existam condições de selecção complexas torna-se pouco prático prefixar as colunas com o nome das tabelas. Tem a alternativa de introduzir, imediatamente a seguir ao nome da tabela na cláusula FROM, um prefixo à sua escolha. Por exemplo o query:

```
select      it.order_num, od.customer_num,   cu.company
from        items it, orders od, customer cu
where       it.stock_num = 6
      and   od.order_num = it.order_num
      and   cu.customer_num = od.customer_num
```

obtendo o resultado:

order_num	customer_num	company
1005	116	Olympic City
1006	112	Runners & Others
1010	115	Gold Medal Sports
1013	104	Play Ball!
1005	116	Olympic City
1006	112	Runners & Others
1010	115	Gold Medal Sports
1013	104	Play Ball!

Quando se pretende seleccionar valores de colunas de mais de uma tabela, mas com todas as linhas da tabela, especificada em primeiro lugar na cláusula FROM, presentes no resultado satisfazendo ou não as condições de join diz-se que temos um OUTER JOIN. O INFORMIX-SQL permite executar OUTER JOINS, associando o valor NULL às colunas do resultado correspondentes à tabela em segundo lugar na cláusula FROM, nas linhas que não satisfazem as condições de join. Para indicar um outer join basta escrever a palavra OUTER antes da tabela especificada em segundo lugar na cláusula FROM. Veja o exemplo:

```
select      order_num, orders.customer_num,
            company, zipcode
from        orders, outer customer
where orders.paid_date is null
      and   orders.customer_num = customer.customer_num
      and   customer.zipcode like '9402_'
```

Obtendo o resultado:

order_num	customer_num	company	zipcode
1004	106		
1006	112	Runners & Others	94022
1007	117		
1012	117		

2.8.7. Subqueries

As condições de pesquisa da cláusula WHERE da instrução SELECT possibilitam ainda:

-Comparar uma expressão com o resultado de outra instrução SELECT.

-Determinar se uma expressão está incluída no resultado de outra instrução SELECT.

-Verificar se existem linhas seleccionadas por outra instrução SELECT.

Às instruções SELECT chamadas por cláusulas WHERE de outras instruções SELECT chamam-se SUBQUERIES. Dum subquery pode resultar um valor, nenhum valor ou um conjunto de valores, mas deve conter uma só coluna ou expressão na lista das colunas selecionadas para o resultado e não deve ter cláusula ORDER BY.

Para identificar e testar as várias situações de subqueries citadas introduzem-se os parâmetros, que se descrevem a seguir,imediatamente antes da instrução SELECT do subquery. O subquery deve ser escrito entre parêntesis.

ALL Indica que o subquery pode retornar nenhum, um ou mais valores e que a condição de pesquisa (do SELECT do primeiro nível) é verdadeira se a comparação for verdadeira para todos os valores retornados. Se o subquery não retornar nenhum valor a condição de pesquisa é verdadeira.

ANY Indica que o subquery pode retornar nenhum, um ou mais valores e que a condição de pesquisa (do SELECT do primeiro nível) é verdadeira se a comparação for verdadeira para pelo menos um valor retornado. Se o subquery não retornar nenhum valor a condição de pesquisa é falsa.

SOME Tem a mesma função que ANY.

IN Verifica se a expressão da condição de selecção está contida no resultado do subquery.

EXISTS Verifica se o subquery retorna linhas . A condição de selecção é verdadeira se o subquery retornar pelo menos uma linha.

NOT Aplica-se antes dos parâmetros anteriores e que nega os seus valores lógicos.

Vejamos alguns exemplos:

Exemplo 1:

```
select      order_num
from        items where stock_num = 9
      and   quantity = (select max(quantity)
                        from items
                        where stock_num = 9)
```

Tendo como resultado:

order_num
1012

Este subquery retorna um único valor [MAX(QUANTITY)]. O objectivo deste exemplo é obter a lista das encomendas ou a encomenda do maior número de artigos com código de stock igual a 9.

Exemplo 2:

```
select      distinct order_num
from        items
where       total_price > all (select total_price
                               from items
                               where order_num = 1005)
```

Tendo como resultado:

order_num
1002
1003
1004
1007
1008
1009
1012
1014
1015

Este subquery retorna vários valores. O exemplo lista todas as encomendas que tenham pelo menos uma linha de encomenda que tenha preço total superior a todas as linhas da encomenda 1005.

Exemplo 3:

```
select      distinct customer_num
from        orders
where order_num not in (select order_num
                        from items
                        where stock_num = 1)
```

Tendo como resultado:

customer_num
101
104
116

112
110
115
117
106

Este exemplo lista todos os clientes que não encomendaram artigos com código de stock igual a 1.

Exemplo 4:

```
select      customer_num
from        customer
where customer_num =   any (select customer_num
                            from orders
                            where paid_date is null)
```

Tendo como resultado:

customer_num
106
112
117

Este exemplo lista todos os clientes que fizeram encomendas que ainda não foram pagas.

Exemplo 5:

```
select      o.order_num
from        orders o
where exists (select *
            from customer c
            where phone like '415%' and
            c.customer_num = o.customer_num)
```

Tendo como resultado:

order_num
1001
1003

```
| 1004 |
| 1005 |
| 1006 |
| 1007 |
| 1008 |
| 1010 |
| 1011 |
| 1012 |
| 1013 |
| 1014 |
| 1015 |
```

Este exemplo lista todas as encomendas que foram feitas por clientes que tenham número de telefone começado por 415.

2.8.8. INTO TEMP - Criação dum Resultado numa Tabela Temporária

A cláusula INTO TEMP cria uma tabela temporária que contém o resultado do query. Esta tabela desaparece quando se termina a sessão do INFORMIX-SQL. Os nomes das colunas da tabela temporária são iguais aos das colunas selecionadas pela instrução SELECT e a tabela temporária não tem indices associados.

Vamos guardar o resultado dum query, já utilizado como exemplo anteriormente, na tabela temporária RESULT.

```
select      it.order_num,   od.customer_num,   cu.company
from        items it, orders od, customer cu
where       it.stock_num = 6 and od.order_num = it.order_num
      and   cu.customer_num = od.customer_num
into temp result
```

Para visualizarmos o resultado temos de introduzir a instrução:

```
select *
from result
```

Obtendo o resultado:

```
order_num          customer_num       company
```

1005	116	Olympic City
1006	112	Runners & Others
1010	115	Gold Medal Sports
1013	104	Play Ball!
1005	116	Olympic City
1006	112	Runners & Others
1010	115	Gold Medal Sports
1013	104	Play Ball!

2.9. Views

Quando se define uma VIEW (vista) sobre uma tabela, esta parecerá que contém apenas as colunas e linhas de que precisa. Funciona como se tivesse criado uma nova tabela contendo apenas as colunas e linhas necessárias ao seu trabalho.

Trabalha-se com uma view como se fosse uma tabela, embora a view esteja totalmente dependente dos dados duma ou mais tabelas. A view não possui dados próprios e portanto não necessita de espaço em disco para os guardar. Como uma view deriva duma tabela já existente, quando alterar dados da view, estará na realidade a alterar dados da tabela. Logo os dados das views parecerão que são automàticamente alteradas logo que as tabelas de que dependem forem alteradas.

À medida que os dados duma tabela são alterados, também os dados que são acessíveis através duma view, definida sobre essa tabela, serão alterados. Pode alterar os dados acedidos por uma view desde que a view esteja definida sobre uma única tabela , que nenhuma das suas colunas seja uma expressão ou função das colunas da tabela e tiver autorização para alterar os dados da view.

O administrador da base de dados pode criar várias views diferentes sobre a mesma tabela. Assim os utilizadores terão acesso apenas aos dados que necessitem. As views diminuem a complexidade e ao mesmo tempo restringem o acesso. Quando utiliza uma view (em vez da tabela sobre a qual foi definida a view), não pode aceder a outras linhas ou colunas que não estejam incluídas na view.

Em geral existem duas razões para usar views:

- Uma view pode evitar que utilizadores não autorizados tenham acesso a dados importantes (por exemplo pode aceder a uma view da tabela de pessoal e não aceder à coluna de salários).
- Ao usar uma view as instruções de RDSQL tornam-se mais fáceis de escrever e os resultados mais manejáveis pois só se apercebe das colunas e linhas de que precisa.

Vamos criar uma view sobre as tabelas ITEMS, ORDERS e CUSTOMER de modo a que se possa ter ligado a cada encomenda o número de encomenda, o preço total da encomenda e a identificação do cliente.

```
create view resorder (order_num, customer_num, company,price)
as    select      i.order_num, o.customer_num, c.company,
                  sum(i.total_price)
      from        items i, orders o, customer c
      where o.order_num = i.order_num
      and   c.customer_num = o.customer_num
      group by 1,2,3
```

Para visualizar a view criada entre o query:

```
select *
from resorder
```

Obtendo o resultado:

order_num	customer_num	company	price
1001	104	Play Ball!	$250.00
1002	101	All Sports Supplies	$1200.00
1003	104	Play Ball!	$959.00
1004	106	Watson & Son	$2126.00
1005	116	Olympic City	$562.00
1006	112	Runners & Others	$498.00
1007	117	Kids Korner	$1696.00
1008	110	AA Athletics	$940.00
1009	111	Sports Center	$450.00
1010	115	Gold Medal Sports	$84.00
1011	104	Play Ball!	$99.00
1012	117	Kids Korner	$1040.00
1013	104	Play Ball!	$143.80
1014	106	Watson & Son	$1440.00
1015	110	AA Athletics	$450.00

Note que os dados desta view não podem ser actualizados.

NOTA: Na instrução SELECT associada a CREATE VIEW não pode utilizar a cláusula ORDER BY nem o operador UNION. Não pode utilizar com uma view as seguintes instruções do RDSQL: ALTER TABLE, ALTER INDEX, CREATE INDEX e DROP INDEX.

2.10. Transacções

Chama-se TRANSACÇÃO a uma séria de operações (instrucções de RDSQL) sobre a base de dados que constituem um bloco coerente de tratamento da informação e, só depois de todas as operações do bloco terem sido completadas com sucesso ou não, é que a base de dados é actualizada ou reposta na situação imediatamente anterior ao início da transacção.

Para poder usar transacções tem de criar um ficheiro que registe todas as modificações da base de dados. É o "transation log file". Pode criá-lo quando cria a base de dados com a cláusula WITH LOG IN da instrução CREATE DATABASE ou, no caso da base de dados já ter sido criada sem log file, com a instrução START DATABASE.

Inicia-se uma transacção com a instrução BEGIN WORK e conclui-se com as instruções COMMIT WORK, para actualizar a base de dados, ou ROLLBACK WORK, para repor a situação anterior à transacção.

Vejamos alguns exemplos:

1- Vamos primeiro criar o log file da base de dados STORES.

```
close  database;
start  database   stores   with   log   in "/usr/login/stlog"
```

2- A seguir vamos iniciar uma transacção e inserir algumas linhas na tabela FABRICANTE:

```
begin work;
insert into fabricante values ('MNM', 'manom');
insert into fabricante values ('DIP', 'dipar'); rollback work
```

Como se terminou a transacção com ROLLBACK WORK as inserções não foram efectuadas, como verificarão ao executar o query:

```
select *
from fabricante
```

Obtendo o resultado:

manu_code	manu_name
SMT	Smith
NRG	Norge
HSK	Husky

```
HRO                 Hero
```

3- A seguir vamos executar a mesma transacção, mas termiando-a com COMMIT WORK:

```
begin work;
insert into fabricante values ('MNM', 'manom');
insert into fabricante values ('DIP', 'dipar'); commit work
```

Neste exemplo as inserções foram efectuadas, como verificarão o executar o query:

```
select *
from fabricante
```

Obtendo o resultado:

manu_code	manu_name
DIP	dipar
MNM	manom
SMT	Smith
NRG	Norge
HSK	Husky
HRO	Hero

Depois de iniciada a transacção e enquanto não for terminada, quer por COMMIT WORK quer por ROLLBACK WORK, todas as linhas que forem alteradas pela transacção ficam protegidas (locked) e mais nenhum utilizador pode aceder a essas linhas para alteração (embora possa consultá-las).

Existe um número máximo de linhas que podem estar protegidas simultânemente por todos os utilizadores. Esse número depende do sistema operativo. Nos casos em que este facto possa causar problemas pode superá-lo protegendo toda a tabela até terminar a transacção.

Para proteger a tabela use a instrução LOCK TABLE IN [SHARE ou EXCLUSIVE] MODE. Se usar SHARE os restantes utilizadores podem consultar a tabela. Se usar EXCLUSIVE mais nenhum utilizador pode aceder à tabela quer em consulta quer em alteração. A instrução LOCK TABLE deve ser escrita imediatamente a seguir a BEGIN WORK, no caso de ser utilizada numa transacção.

Vejamos como se protegeria a tabela FABRICANTE num dos exemplos de transacção anteriores:

```
begin work;
lock table fabricante   in  share  mode;
insert into fabricante values ('MNM', 'manom');
insert into fabricante values ('DIP', 'dipar'); commit work
```

Sempre que se tenha protegido uma tabela, esta não pode ser protegida por mais nenhum utilizador, enquanto a transacção não tiver terminado ou executar o comando UNLOCK TABLE (no caso de não ter iniciado uma transacção).

2.11. Segurança

2.11.1. Autorizações

Se criar uma base de dados fica automáticamente como o administrador dessa base de dados. Nesta altura é a única pessoa que tem acesso à base de dados. Para que outros utilizadores tenham acesso à base de dados tem de lhes dar autorização.

Existem os seguintes três níveis de acesso a bases de dados:

CONNECT	Permite o acesso às tabelas da base de dados sem permissão de criar tabelas permanentes e indices.
RESOURCE	Permite o acesso às tabelas da base de dados com permissão de criar tabelas permanentes e indices.
DBA	Permite os mesmos privilégios do administrador da base de dados.

Além das autorizações concedidas sobre a base de dados o criador duma tabela também pode conceder autorizações sobre essa tabela a outros utilizadores.

Existem as seguintes autorizações de acesso a tabelas:

ALTER	Autoriza a adicionar e subtrair colunas ou modificar o tipo de dados da tabela.
DELETE	Autoriza a apagar linhas da tabela.
INDEX	Autoriza a criar indices.
INSERT	Autoriza a inserir linhas na tabela.
SELECT [(cols)]	Autoriza a consultar dados da tabela ou das suas colunas especificadas entre parentesis.
UPDATE [(cols)]	Autoriza a alterar dados da tabela ou das suas colunas especificadas entre parentesis.
ALL	Concede todas as autorizações acima descritas.

É a instrução GRANT que permite conceder autorizações quer a bases de dados quer a tabelas.

Pode conceder autorização para que o utilizador também conceda autorizações sobre a tabela se usar a cláusula WITH GRANT OPTION.

As autorizações (para bases de dados e para tabelas) são concedidas a utilizadores (descritos pelos respectivos logins) ou então a todos os utilizadores escrevendo a cláusula PUBLIC.

Em caso de conflito entre autorizações é sempre a autorização mais restritiva que tem preferência. Por exemplo se um utilizador tem nível RESOURCE para uma base de dados, mas não tem autorização para criar indices numa tabela é esta última situação que prevalece.

Como exercícios propõe-se que se concedam autorizações sobre as tabelas da base de dados STORES existente na sua área a outros colegas de curso.

Vejamos alguns exemplos hipotéticos de utilização da instrução GRANT:

```
grant  stores connect  to  public;
grant  stores dba  to  joao;
grant  all  on  fabricante  to  luis, sergio;
grant  select  on  fabricante  to  public
```

Para retirar autorizações a utilizadores usa-se a instrução REVOKE. Por exemplo

```
revoke  delete  on  fabricante  from  luis
```

2.11.2. Recuperação de Dados

Este ponto deve ser abordado num curaso dirigido para a admnistração de dados e não num curso de introdução como este. No entanto convém realçar alguns aspectos que podem interessar a quem programa.

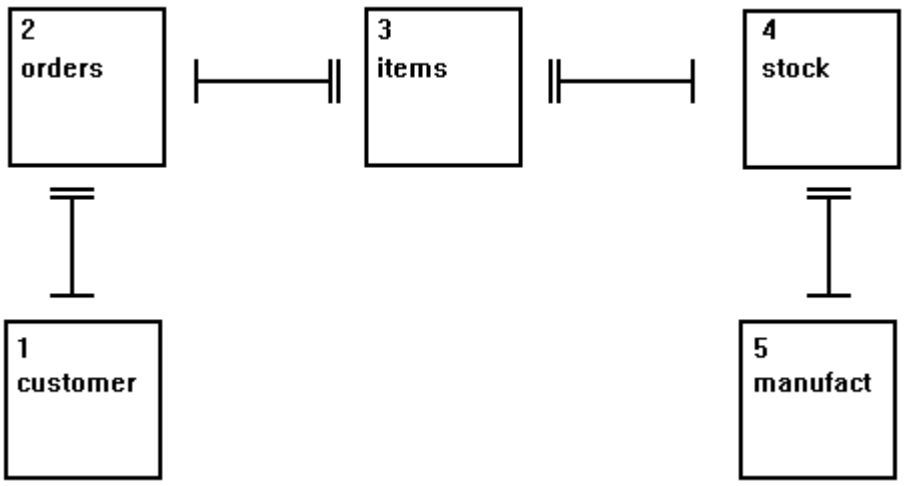
O instrumento mais potente de recuperação de dados é o já nosso conhecido "transaction log file". Este ficheiro contém as operações sobre a base de dados desde o instante imediatamente após ter sido gravado um ficheiro de backup. Com estes dois ficheiros e usando a instrução ROLLFORWARD DATABASE é possivel recuperar a base de dados até ao COMMIT WORK da último transacção executada.

Existe também a possibilidade de registar as operações executadas sobre uma tabela num ficheiro a que se chama AUDIT TRAIL. Pode-se recuperar a tabela através da instrução RECOVER AUDIT. Este processo pode ser usado com o "transaction log file" ou como alternativa a este.

3. Apêndice 1

DESCRICAO DA BASE DE DADOS STORES.

DESPODATA/DE1 * MODELO DE DADOS * 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		

DESPODATA/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

Esquema de ligação entre as tabelas da base de dados STORES