Tarefa Orientada S16

Cursores

Objectivos:

- Declarar cursores
- Utilizar cursores
- Utilizar funções do sistema para trabalhar com cursores
- Alterar dados através de cursores

Um cursor é um objecto da base de dados que permite obter dados de um registo de cada vez, entre um conjunto de registos.

As *APIs* (*Application Programming Interfaces*) mais comuns, tais como a *ADO .NET* e *ODBC*, utilizam os cursores para aceder aos dados no servidor, a partir de aplicações cliente. Dado que as *APIs* gerem os cursores no cliente, não é necessário escrever qualquer código T-SQL no servidor.

Existem, contudo, situações em que é necessário/útil utilizar cursores no servidor, apesar de ser um processo mais lento e consumidor de mais recursos do que outras técnicas de acesso aos dados remotos. Por exemplo, os cursores podem ser úteis para criar procedimentos administrativos ou para gerar SQL dinâmico.

Duas das principais características de um cursor são a sua capacidade, ou ausência dela, de movimentação, para trás e para a frente, entre as várias linhas de um conjunto de registos e a sua sensibilidade, ou ausência dela, às alterações efectuadas na base de dados.

Se um cursor permitir o acesso aos registos anterior e posterior do registo seleccionado, designa-se o cursor de *scrollable*. Por outro lado, se o cursor apenas permitir o acesso ao registo seguinte do registo seleccionado, designa-se o cursor de *forward-only*.

Se um cursor for sensível às alterações efectuadas na base de dados, significa que está ciente das alterações que podem ocorrer depois de ter sido criado. Deste modo, é possível visualizar as alterações dos registos actualizados. Por exemplo, suponha que outro utilizador actualiza o registo número 10 enquanto o cursor está posicionado no registo 9. Quando se procurar o registo seguinte, um cursor sensível a alterações iria devolver o registo actualizado. Inversamente, um cursor não sensível a alterações devolveria o registo original, isto é, ainda sem as alterações efectuadas pelo outro utilizador.

A tabela seguinte resume quatro, dos sete, tipos de cursores disponíveis no SQL-Server.

Tipo padrão	Movimentação pelas linhas não restringida (*scroll*)	Sensibilidade a alterações (de outros utilizadores) (*sensitive*)
Dynamic	✓	✓
Keyset-driven	✓	Apenas alterações e remoções de linhas (não inserções)
Static	✓	X
Forward-only	Apenas para a próxima linha	X

Um cursor *dynamic* é sensível a todas as alterações efectuadas na base de dados. Para tal, este tipo de cursores basicamente executam a consulta a eles subjacente da cada vez que se busca um novo registo. Deste modo, quaisquer dados que tenham sofrido alterações desde que o último registo foi capturado, vão ser incluídos no resultado (conjunto de registos alcançáveis pelo cursor). Todavia, este tipo de cursores requer mais recursos do sistema e resultam em perdas de performance. Por omissão, os cursores *dynamic* são *scrollable*, isto é, permitem a navegação de registos nos dois sentidos.

Os cursores *Keyset-Driven* são sensíveis a operações de *UPDATE* e de *DELETE* sobre os registos fonte. Contudo, não são sensíveis a operações de *INSERT*. Por omissão, os cursores *Keyset-Driven* também são *scrollable*. Quando se abre um cursor *Keyset-Driven*, o sistema guarda uma cópia, na base de dados *tempdb*, dos valores chave únicos (*Keyset*) do conjunto de registos de que dispõe. Depois, quando se busca um registo, o cursor utiliza os valores chave únicos (*Keyset*) para consultar a tabela original.

Um cursor *Static* é insensível a qualquer alteração efectuada sobre os dados originais. Por omissão, os cursores *Static* também são *scrollable*. Quando se abre um cursor *Static*, o sistema cria uma cópia do conjunto de registos de que dispõe na base de dados *tempdb*. Depois, quando se busca um registo, ele é devolvido a partir dessa cópia, e não da tabela original. Dado que as alterações efectuadas na tabela original não afectam a cópia estática da tabela, o cursor não toma conhecimento delas. Dado que a busca das colunas não pertencentes ao *Keyset* é efectuada na tabela original, o cursor é sensível a alterações e remoções. Por outro lado, uma vez que o *Keyset* é definido apenas uma vez quando o cursor é aberto, este tipo de cursores não consegue detectar as novas inserções de registos.

Por omissão, um cursor *forward-only* é sensível às alterações registadas na base de dados, mas não é navegável nos dois sentidos (*scrollable*).

Dos três tipos de cursores que permitem a navegação nos dois sentidos, os cursores *dynamic* requerem mais recursos do sistema, logo são os mais lentos. Os cursores *static* são os mais rápidos. Por outro lado, os cursores *forward-only* ocupam menos recursos do sistema do que os outros três tipos de cursores, tornando-se, assim, mais rápidos.

Para declarar um cursor, no SQL Server, utilize a seguinte sintaxe:

```
DECLARE nome_do_cursor CURSOR
        [LOCAL | GLOBAL]
        [FORWARD_ONLY | SCROLL]
        [FAST_FORWARD| STATIC | KEYSET | DYNAMIC]
        [READ_ONLY | SCROLL_LOCKS | OPTIMISTIC]
FOR instrução_SQL
        [FOR UPDATE [OF nome_da_coluna [, …]]]
```

Utilizam-se as palavras-chave *LOCAL* e *GLOBAL* para definir o âmbito do cursor. Um cursor *LOCAL* apenas pode ser utilizado dentro do procedimento em que foi declarado. Um cursor *GLOBAL* pode ser usado por qualquer *script* ou procedimento na ligação actual. Todavia, não pode ser utilizado em duas ou mais conexões. Se omitir estas palavras-chave, o modo predefinido vai depender da definição da opção da base de dados *CURSOR-DEFAULT* que, por omissão, tem o valor *GLOBAL*.

Utilizam-se as palavras-chave *FORWARD_ONLY* e *SCROLL* para determinar se o cursor é navegável, ou não, nos dois sentidos (para trás e para a frente do registo actual).

Utilizam-se as palavras-chave *FAST_FORWARD*, *STATIC*, *KEYSET* e *DYNAMIC* para identificar o tipo de cursor.

Utilizam-se as palavras-chave *READ_ONLY*, *SCROLL_LOCKS* e *OPTIMISTIC* para identificar o modo como a concorrência vai ser tratada pelo cursor, isto é, o tipo de bloqueio que vai ser efectuado sobre um registo quando os dados são actualizados através do cursor. Como iremos ver mais afrente, se pretender actualizar ou eliminar dados através do cursor, pode usar as opções *OPTIMISTIC* ou *SCROLL_LOCKS*, como diferentes implicações ao nível do bloqueio de registos.

A cláusula *FOR UPDATE OF* também é utilizada quando se pretende actualizar dados através do cursor. Esta cláusula pode ser usada para restringir as colunas que podem ser actualizadas. Se for omitida, qualquer coluna do resultado da instrução *SELECT* definida na instrução *DECLARE* pode ser actualizada. Por outro lado, se incluir esta cláusula na declaração do cursor, então uma coluna pode ser actualizada apenas se estiver incluída na lista especificada a seguir a *FOR UPDATE OF*.

No entanto, dado que as vistas proporcionam um modo mais flexível de identificar colunas específicas, normalmente baseia-se o cursor numa vista que devolva apenas as colunas que podem ser actualizadas, em vez de utilizar a cláusula *FOR UPDATE OF*.

A instrução SQL especificada na cláusula *FOR* define o conjunto de registos associados ao cursor. Pode ser utilizada qualquer tipo de instrução *SELECT*, incluindo junções, subconsultas, funções de agregação ou consultas de sumário. Todavia, não pode ser utilizada a instrução *SELECT INTO*.

Apenas tem que se garantir que o tipo de cursor é compatível com o tipo de instrução *SELECT* utilizado na sua declaração. Por exemplo, se utilizar uma cláusula *GROUP BY* na definição da consulta subjacente ao cursor, não pode declarar o cursor como sendo do tipo *dynamic*. Neste caso, o SQL Server iria converter implicitamente o tipo de cursor para *static*.

1 Formule, analise e execute as instruções a seguir apresentadas.

1.1 Declaração de um cursor que utiliza as opções predefinidas.

```
DECLARE Cursor_Facturas CURSOR
FOR
      SELECT * FROM Facturas
```

Dado que não foram especificadas quaisquer palavras-chave opcionais, o cursor foi criado usando as definições predefinidas: *GLOBAL*, *DYNAMIC*, *FORWARD_ONLY* e *OPTIMISTIC*.

1.2 Declaração de um cursor que permite actualização de dados em colunas específicas da tabela *Fornecedores*.

```
DECLARE Cursor_Forncecedores_UPDATE CURSOR
GLOBAL SCROLL DYNAMIC SCROLL_LOCKS
FOR
      SELECT * FROM Fornecedores
FOR UPDATE OF Nome, Endereço, Localidade, CódigoPostal, Telefone,
              PNoneContacto, ÚNomeContacto
```

Neste exemplo são incluídas todas as colunas da tabela Fornecedores no conjunto de registos associados ao cursor, mas apenas sete destas colunas podem ser actualizadas através do cursor. Note ainda que as palavras-chave GLOBAL e DYNAMIC não são necessárias, pois são consideradas por omissão. Contudo, deste modo, a leitura da declaração do cursor fica mais facilitada.

A seguir, apresentam-se cinco instruções SQL para trabalhar com cursores no SQL Server.

Instrução	Descrição
DECLARE CURSOR	Define um novo cursor
OPEN	Abre um cursor declarado e preenche o seu conteúdo
FETCH	Encontra uma dada linha de um cursor
CLOSE	Fecha um cursor
DEALLOCATE	Remove as estruturas de dados do cursor

Sucintamente, após declarar o cursor através da instrução *DECLARE*, utiliza-se a instrução *OPEN* para abrir e preencher o cursor com os dados provenientes da execução da instrução *SELECT* definida na instrução *DECLARE*. Depois, utiliza-se a instrução *FETCH* para percorrer um registo de cada vez do conjunto de registos que constitui o resultado da instrução

SELECT. Após finalizar a utilização do cursor, utiliza-se a instrução *CLOSE* para o fechar. Como resultado, são libertados os recursos do sistema que o cursor necessita para armazenar e navegar sobre o resultado da instrução *SELECT* definida na instrução *DECLARE*. Note, contudo, que a definição do cursor ainda existe, mesmo depois de ter sido fechado. Deste modo, é possível tornar a abrir e preencher novamente o cursor. Quando pretender eliminar a definição do cursor e libertar os restantes recursos do sistema associados ao cursor, utiliza-se a instrução *DEALLOCATE*.

1.3 *Script* que declara e utiliza um cursor.

```
USE Pagamentos

DECLARE Cursor_Fornecedores CURSOR
STATIC
FOR
    SELECT IDFornecedor, Nome
    FROM Fornecedores
    ORDER BY Nome

OPEN Cursor_Fornecedores

FETCH NEXT FROM Cursor_Fornecedores

WHILE @@FETCH_STATUS = 0
    FETCH NEXT FROM Cursor_Fornecedores

CLOSE Cursor_Fornecedores

DEALLOCATE Cursor_Fornecedores
```

A segunda instrução declara um cursor *STATIC*, *GLOBAL*, *FORWARD_ONLY* e *OPTIMISTIC*. A instrução *SELECT* incluída na cláusula *FOR* define o conjunto de registos que vai ser utilizado pelo cursor. Depois, a instrução *OPEN* abre e preenche o cursor. A instrução *FETCH NEXT* localiza o próximo registo do cursor. Uma vez que esta é a primeira instrução *FETCH* sobre um cursor aberto de novo, vai ser localizado o primeiro registo do conjunto de registos que vai ser utilizado pelo cursor. Depois, dentro do ciclo *WHILE*, é localizado o próximo registo até chegar ao último registo do conjunto de registos associados ao cursor. A função *@@FETCH_STATUS* usada como condição do ciclo *WHILE* para testar se a última instrução *FETCH* foi processada com

sucesso. Mais tarde voltaremos a este assunto. Finalmente, são utilizadas as instruções *CLOSE* e *DEALLOCATE* para libertar todos os recursos utilizados pelo cursor.

A seguir, apresenta-se o resultado óbito com a execução do *script* anterior.

Results | Messages

	IDFornecedor	Nome
1	81	Bell
1	4	Bertrand
1	110	Cartografia do Minho
1	10	Construtora do Liz
1	3	CTT
1	82	FCA
1	34	IBM
1	121	Mc Graw Hill
1	2	ONI Telecom
1	122	Palinter
1	5	Porto Editora
1	7	Portugal Mail
1	1	Portugal Telecom
1	9	Visabeira
1	8	Vispapel
1	6	WebBoom
	IDFornecedor	Nome

A seguir, apresenta-se a sintaxe da instrução *FETCH*.

FETCH [**NEXT** | **PRIOR** | **FIRST** | **LAST** | **ABSOLUTE** n | **RELATIVE** n]
FROM [**GLOBAL**] nome_do_cursor
[**INTO** @nome_da_variável [, …]]

A palavra-chave indicada a seguir à instrução *FETCH* especifica a direcção da navegação. Por omissão, é considerada a palavra-chave *NEXT*. Se o cursor for *forward-only*, *NEXT* é a única direcção válida. Na cláusula *FROM* especifica-se o nome do cursor através do qual vamos localizar os registos.

A palavra-chave *ABSOLUTE* seguida do valor *n* permite localizar o registo que está na posição *n* do conjunto de registos associado ao cursor. A numeração começa no valor 1. Dado que o número de registos associados a um cursor *DYNAMIC* pode ser alterado, não pode utilizar a palavra-chave *ABSOLUTE* com este tipo de cursores.

A palavra-chave *RELATIVE* seguida do valor *n* permite localizar o registo que está na posição *n* a partir do último registo localizado. Se *n* for negativo, a localização é feita para trás. Se *n* for positivo, a localização é feita para a frente do último registo localizado. Se *n* for igual a zero, é localizado o mesmo registo.

Se existirem cursores *GLOBAL* ou *LOCAL* ambos com o mesmo nome, o registo irá ser localizado, de modo predefinido, a partir do cursor *LOCAL*. Assim, se pretender localizar um registo a partir do cursor *GLOBAL*, deve utilizar a palavra-chave *GLOBAL* na instrução *FETCH*.

Utilize a palavra-chave *INTO* seguida de uma lista de variáveis para atribuir os valores obtidos através da instrução *FETCH* a essas variáveis. Note que deve listar as variáveis na ordem apropriada. O tipo de dados de cada variável deve ser compatível com o tipo de dados da coluna correspondente e o número de variáveis deve ser igual ao número de colunas obtidas pela instrução *FETCH*. Se omitir a cláusula *INTO*, o registo é devolvido directamente para o cliente. No caso de *Management Studio*, o registo é mostrado no tabulador de resultados.

1.4 A seguir apresentam-se alguns exemplos de instruções *FETCH*.

FETCH FROM Cursor_Fornecedores	Localiza o próximo registo
FETCH NEXT FROM Cursor_Fornecedores	Localiza o próximo registo
FETCH PRIOR FROM Cursor_Fornecedores	Localiza o registo anterior
FETCH FIRST FROM Cursor_Fornecedores	Localiza o primeiro registo
FETCH LAST FROM Cursor_Fornecedores	Localiza o último registo
FETCH ABSOLUTE 3 FROM Cursor_Fornecedores	Localiza o terceiro registo
FETCH RELATIVE 4 FROM Cursor_Fornecedores	Localiza o quarto registo após o registo actual
FETCH RELATIVE -2 FROM Cursor_Fornecedores	Localiza o segundo registo antes do registo actual
FETCH RELATIVE 0 FROM Cursor_Fornecedores	Localiza o registo actual novamente
FETCH FROM Cursor_Fornecedores INTO @VarIDFornecedor, @VarNome	Localiza o próximo registo e atribui os valores a variáveis locais

Pode utilizar a função do sistema *@@FETCH_STATUS* para determinar o estado da última instrução *FETCH* executada. A seguir apresentam-se os valores que podem ser devolvidos por esta função.

Valor devolvido	Significado
0	FETCH com sucesso;
-1	Não devolveu nenhuma linha pois a posição do cursor excedeu os limites do conjunto activo, estando este posicionado: ▪ antes da primeira linha ▪ depois da última linha
-2	A linha devolvida já não é um membro do conjunto activo (e.g. porque foi removida).

Normalmente, esta função é utilizada, depois de ter sido localizado o primeiro registo, na expressão condicional de um ciclo *WHILE* para localizar os restantes registos associados a um cursor.

Para cursores *STATIC* e *DYNAMIC*, esta função pode devolver os valores 0 ou -1.

Para cursores *KEYSET_DRIVEN*, a função pode ainda devolver o valor -2, significando que se tentou localizar um registo que foi eliminado. Por outras palavras, outro utilizador ou processo eliminou o registo que era membro do conjunto de chaves (*keyset*) guardado pelo cursor.

1.5 Ciclo *WHILE* que localiza, navegando para a frente, os vários registos de um cursor.

```
FETCH FIRST FROM Cursor_Fornecedores

WHILE @@FETCH_STATUS = 0
      BEGIN
            …
            FETCH NEXT FROM Cursor_Fornecedores
      END
```

1.6 Ciclo *WHILE* que localiza, navegando para trás, os vários registos de um cursor.

```
FETCH LAST FROM Cursor_Fornecedores

WHILE @@FETCH_STATUS = 0
      BEGIN
            …
            FETCH PRIOR FROM Cursor_Fornecedores
      END
```

1.7 Ciclo *WHILE* que localiza os vários registos de um cursor *KEYSET-DRIVEN*.

```
FETCH FIRST FROM Cursor_Fornecedores_KEYSET
WHILE @@FETCH_STATUS <> -1
      BEGIN
            IF @@FETCH_STATUS = -2
                  PRINT 'Registo eliminado'
            …
            FETCH NEXT FROM Cursor_Fornecedores_KEYSET
      END
```

Note que a função *@@FETCH_STATUS* abrange todos os cursores abertos da ligação actual. Deste modo, se estiver a utilizar vários cursores, não use

outras instruções entre a instrução *FETCH* e o teste à função *@@FETCH_STATUS*.

Pode utilizar a função do sistema *@@CURSOR_ROWS* para obter o número de registos associados ao último cursor aberto. A seguir apresentam-se os valores que podem ser devolvidos por esta função.

Valor devolvido	Significado
-m	Cursor ainda não completamente preenchido, sendo m o número de colunas existentes no conjunto activo
n	Cursor já preenchido completamente com n linhas
0	Nenhum cursor foi aberto ou o último cursor aberto já foi fechado ou destruído
-1	Nº de linhas variável pelo facto de se tratar de um cursor dinâmico

Se o valor devolvido pela função for igual a zero, significa que o cursor não tem registos associados. Se for um número positivo, representa o número de registos associados ao cursor. Se for iguala -1, indica que o último cursor a ser aberto é do tipo *DYNAMIC*. Dado que o número de registos associados a um cursor deste tipo pode mudar com as eliminações e inserções de registos efectuadas por outros utilizadores, o número total de registos é desconhecido. Por outro lado, o número de registos associados a cursores dos tipos *STATIC* e *KEYSET-DRIVEN* é fixado quando se abre o cursor.

Quando dois ou mais utilizadores tentam modificar os mesmos dados simultaneamente, o SQL Server gera um erro de concorrência. Uma das formas que o SQL Server tem para gerir a concorrência passa por bloquear um registo enquanto um utilizador estiver a utilizá-lo. Enquanto o registo estiver bloqueado, o sistema impede que os outros utilizadores ou processos modifiquem ou eliminem esse registo. Basicamente, podem ser utilizados

dois tipos distintos de bloqueio quando trabalha com dados através de um cursor: optimista e pessimista.

Se utilizar um bloqueio pessimista (*pessimistic*), o sistema assume que dois ou mais utilizadores ou processos irão tentar modificar simultaneamente o mesmo registo. Deste modo, o sistema bloqueia um registo quando ele é localizado. O bloqueio é mantido até que seja localizado outro registo ou até que o cursor seja fechado. Assim, garante-se que nenhum outro utilizador ou processo possa modificar ou eliminar esse registo enquanto estiver bloqueado.

Note, contudo, que os outros utilizadores ou processos, tentarem actualizar ou eliminar um registo bloqueado, não vão receber nenhuma mensagem a informar que o registo está bloqueado. Em vez disso, a instrução de *UPDATE* ou de *DELETE* irá ficar suspensa até que o registo em causa seja desbloqueado. Assim, dado que o bloqueio permanece até se localizar outro registo ao até o cursor ser fechado, não deve manter o cursor na mesma posição durante muito tempo.

Se utilizar um bloqueio optimista (*optimistic*), o sistema assume que nenhum outro utilizador ou processo irá tentar modificar simultaneamente o mesmo registo. Para sistemas com poucos utilizadores, esta assunção é usualmente verdadeira. Portanto, o sistema não bloqueia o registo. O sistema verifica se o registo foi actualizado desde que foi localizado. Se tentar actualizar o registo, mas ele já foi actualizado ou eliminado por outro utilizador ou processo, a actualização provoca um erro com o número 16934. Depois, pode usar código para tratar esse erro de forma adequada. Por exemplo, se o registo tinha sido actualizado por outro utilizador, pode localizá-lo novamente e tentar submeter a actualização.

A seguir apresentam-se as três opções de gestão da concorrência para cursores criados com T-SQL.

Opção	Descrição
OPTIMISTIC	Não bloqueia o registo. Significa que o registo pode ser modificado por outro utilizador ou processo depois de o registo ter sido localizado.
SCROLL_LOCKS	Cada registo é bloqueado quando é localizado. Isto significa que nenhum outro processo ou utilizador pode modificar o registo até que o bloqueio seja libertado por se te localizado outro registo ou por se ter fechado o cursor
READ_ONLY	não bloqueia o registo, pois não se pode actualizar dados através do cursor

Note que a opção *SCROLL_LOCKS* implementa o tipo de bloqueio pessimista (*pessimistic*). Assim, para utilizar este tipo de bloqueio, utilize a palavra-chave *SCROLL_LOCKS* na declaração do cursor.

Por omissão, os cursores do tipo *STATIC* são definidos como *READ_ONLY* e os cursores *KEYSET-DRIVEN* e *DYNAMIC* implementam um bloqueio *OPTIMISTIC*.

Se o sistema tiver muitos utilizadores for necessário actualizar várias vezes dados através do cursor, pode utilizar o bloqueio pessimista. Em particular, se as actualizações afectarem dados críticos, que não podem ser corrigidos simplesmente através de uma nova submissão da instrução de UPDATE, então pode utilizar este tipo de bloqueio.

Contudo, dado que o bloqueio pessimista consome mais recursos do sistema e atrasa o acesso aos dados, deve ser evitada a utilização deste tipo de bloqueio sempre que possível.

Pode actualizar ou eliminar o último registo localizado por um cursor utilizando a cláusula *WHERE CURRENT OF* em vez de usar uma condição de pesquisa nas instruções *UPDATE* e *DELETE*.

A sintaxe da cláusula *WHERE CURRENT OF* é a seguinte.

WHERE CURRENT OF nome_do_cursor

Note, todavia, que, para poder utilizar um cursor na cláusula *WHERE CURRENT OF*, o cursor tem que ser definido de modo a implementar o controlo da concorrência optimista ou pessimista. Isto é, não pode ser *READ_ONLY*.

1.8 *Script* que declara um cursor e localiza o primeiro registo.

```
USE Pagamentos

DECLARE Cursor_Fornecedores_DYNAMIC CURSOR
DYNAMIC SCROLL_LOCKS
FOR
    SELECT * FROM Fornecedores ORDER BY Nome

OPEN Cursor_Fornecedores_DYNAMIC
FETCH Cursor_Fornecedores_DYNAMIC
CLOSE Cursor_Fornecedores_DYNAMIC
DEALLOCATE Cursor_Fornecedores_DYNAMIC
```

Dado que foi especificada a opção *SCROLL_LOCKS* na declaração do cursor, o registo é bloqueado, de modo que não possa ser modificado, excepto através do cursor.

A seguir, apresenta-se o resultado obtido através da execução do *script* do passo anterior.

Results | Messages

	IDFornecedor	Nome	Endereço	Localidade	CódigoPostal	Telefone	PNomeContacto	ÚNomeContacto	CondiçãoPagamentoPredefinida	ContaBalançoPredefinida
1	81	Bell	Rua do Jardim das Palas, nº 1	Setubal	1500-111	212987666	Alexandre	Farinha	2	160

1.9 Instrução *UPDATE* que actualiza o registo na posição actual do cursor.

```
UPDATE Fornecedores
SET Nome = 'Microsoft'
WHERE CURRENT OF Cursor_Fornecedores_DYNAMIC
```

1.10 Para esta instrução funcionar, declare e abra novamente o cursor com o nome *Cursor_Fornecedores_DYNAMIC*.

Note que é utilizada a cláusula *WHERE CURRENT OF* com o nome do cursor que contém o registo a actualizar. Por outro lado, a instrução *UPDATE* é aplicada sobre a tabela base que contém o registo que vai ser actualizado. Isto faz sentido, pois os dados a serem actualizados residem na tabela e não no conjunto de registos definido pelo cursor. Neste caso, está-se a usar o cursor para localizar o registo a actualizar.

1.11 Para verificar que o registo foi alterado, na tabela *Fornecedores*, formule a seguinte instrução *SELECT*.

```
SELECT *
FROM Fornecedores
WHERE IDFornecedor = 81
```

O resultado é o seguinte.

Results | Messages

	IDFornecedor	Nome	Endereço	Localidade	CódigoPostal	Telefone	PNomeContacto	ÚNomeContacto	CondiçãoPagamentoPredefinida	ContaBalançoPredefinida
1	81	Microsoft	Rua do Jardim das Patas, nº1	Setubal	1500-111	212987666	Alexandre	Farinha	2	160

1.12 De modo a colocar a base de dados no seu estado original, formule as próximas instruções.

```
UPDATE Fornecedores
SET Nome = 'Bell'
WHERE IDFornecedor = 81


SELECT *
FROM Fornecedores
WHERE IDFornecedor = 81
```

1.13 Instrução *DELETE* para eliminar o registo na posição actual do cursor.

```
DELETE Fornecedores
WHERE CURRENT OF Cursor_Fornecedores_DYNAMIC
```

Neste caso, vai ser gerado um erro, pois a tabela *Facturas* contém facturas do fornecedor identificado pela posição actual do cursor.

1.14 Liberte todos os recursos do sistema consumidos pelo cursor.

QUESTÕES

2 Implemente as seguintes instruções.

2.1 Crie um *script* que utilize um cursor *STATIC* para mostrar, no separador de resultados do *Management Studio*, o identificador e o nome dos fornecedores registados na tabela Fornecedores, de acordo com a imagem seguinte.

Resultado

```
Messages
1, Portugal Telecom
2, ONI Telecom
3, CTT
4, Bertrand
5, Porto Editora
6, WebBoom
7, Portugal Mail
8, Vispapel
9, Visabeira
10, Construtora do Liz
34, IBM
81, Bell
82, FCA
110, Cartografia do Minho
121, Mc Graw Hill
122, Patinter
```

2.2 Altere o cursor criado no *script* anterior (2.1) de modo a que este seja apenas de leitura e mostre a informação saltando um registo, isto é, mostre o 1º registo, o 3º registo, o 5º registo, etc.

Resultado

```
Messages
1, Portugal Telecom
3, CTT
5, Porto Editora
7, Portugal Mail
9, Visabeira
34, IBM
82, FCA
121, Mc Graw Hill
```

2.3 Crie um *script* que implemente um cursor declarado de acordo com as seguintes regras: apenas de leitura, com deslocação num único sentido (próximo registo) e que utilize o mínimo de recursos possíveis. O *script* deverá mostrar o nome de todos os fornecedores com dívidas, a sua localidade, entre parêntesis, e o montante em dívida de acordo com o resultado abaixo apresentado.

Nota: Este *script* deve:

- Utilizar apenas uma consulta simples à tabela "fornecedores";
- Recorrer à função "fn_DividaFornecedor" (implementada na Tarefa Orientada S15 na questão 2.2.1) que calcula o montante em dívida de um fornecedor passado por parâmetro da UDF, para determinar se um fornecedor possui dívidas, e apresentar o respectivo montante da dívida.

Resultado

```
Messages
IBM (Lisboa)
Dívida: 617.34

FCA (Lisboa)
Dívida: 3705.75

Mc Graw Hill (Lisboa)
Dívida: 2379.82

Patinter (Viseu)
Dívida: 3839.74
```

2.4 Neste exercício, pretende-se listar todos os despartamentos por ordem alfabética. Para cada departamento deverá ser apresentado o nome dos colaboradores pertencentes e os projectos a que cada colaborador está atribuído. Caso não existam colaboradores num determinado departamento ou um colaborador não esteja associado a nenhum projecto, devem ser apresentadas, respectivamente, as seguintes mensagens: "- sem colaborador -" e "- sem projecto -".

Resultado

```
Messages
Departamento | Nome Colaborador | Projeto
----------------------------------------
Contabilidade | Joana Larocas | Gestão Salários 2007
Informática | Jorge Barros | BD Gestão Stocks
Informática | António Carrasqueiro | BD Gestão Pagamentos
Informática | Lucas Abebe | LAN Empresa 2007
Manutenção | Pedro Antunes | - sem projeto -
Operacional | - sem colaborador - | - sem projeto -
Pagamentos | Pedro Soares | Gestão Pagamentos 2007
Pagamentos | Rita Branco | Gestão Pagamentos 2006
Pagamentos | Manuel Carrilho | Gestão Facturas 2006
Recursos Humanos | Paulo SIlva | - sem projeto -
Recursos Humanos | Pedro Silvério | Férias 2007
```