ISEL
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

2º Trabalho Prático

Sistemas de informação II

Licenciatura Engenharia Informática e de Computadores

Docente: Afonso Remédios

Inês Nunes – 43590

Diogo Baptista – 41481

Ricardo Rodrigues – 43594

Conteúdo

Introdução 3
Problema Proposto da Primeira E Segunfa Fase 4
Modelo Entidade-Associação 5
Avaliação Experimental 6
Exercício 1 6
Exercício 2 6
Exercício 3 7
Exercício 4 7
Exercício 5 7
Conclusão 8
Webgrafia 8

Introdução

Nesta segunda fase do trabalho prático, iremos pôr em prática os conhecimentos lecionados em aula, como C#, ADO.NET, Entity Framework e SQL, de forma a construir uma aplicação em C#, que nos permita implementar e testar todas as funcionalidades da primeira fase deste trabalho.

No final deste trabalho, teríamos de ser capazes de:

- Desenvolver uma aplicação em C# usando diferentes implementações de acesso a dados.
- Utilizar corretamente o processamento transacional, concretizado através de mecanismos disponíveis na plataforma .NET.
- Utilizar corretamente ADO.NET em modo "conectado", para acessos a dados.
- Utilizar corretamente (ADO.NET) Entity Framework para acessos a dados.
- Garantir a correta libertação de ligações a recursos, quando estes não estejam a ser utilizados.
- Garantir a correta implementação das restrições de integridade e/ou lógica de negócio.
- Organizar o código de acesso a dados usando padrões de desenho, nomeadamente, *Data Mapper*, *Lazy Load*, *Data Transfer Object* e *Virtual Proxy*.
- Organizar o código de acesso a dados para permitir o processamento transacional, iniciado nesse código ou existente no contexto de chamada.

Problema Proposto da Primeira E Segunfa Fase

Primeira fase:

"A empresa FSolv pretende desenvolver um sistema de informação para a gestão de faturas não simplicadas. O sistema tem de ser aditável. Uma fatura é identificada por um código que segue o formato: **FTyyyy-xxxxx**, onde yyyy representa o ano e xxxxx representa o número da fatura
emitida num ano. Mesmo que uma fatura seja anulada, o valor de xxxxx é monótono crescente.
Cada fatura é caracterizada por uma data de criação, uma data de emissão (eventualmente nula), um estado, um valor total (sem IVA) e o valor de IVA a pagar. Ambas as datas têm, pelo menos, resolução ao segundo. O valor total e o valor de iva são resultados dos valores associados dos itens existentes na fatura. Cada item é identificado por um número dentro da fatura e tem uma descrição. É possível definir para cada item da fatura um desconto e o número de unidades presentes na fatura. Considere que uma fatura pode ter os seguintes estados:

- Emitida, impossibilitando posteriores alterações à fatura;
- Em atualização, quando ainda não está finalizada (e.g. falta adicionar itens);
- Proforma, impossibilitando qualquer alteração à fatura, exceto a passagem para os estados Emitida ou Anulada;
- Anulada, impossibilitando posteriores alterações à fatura;

Quando é necessário devolver itens presentes numa fatura, deve ser criada uma nota de crédito. As notas de crédito são semelhantes a faturas, exceto nos itens que a constituem, que podem ser um subconjunto não vazio dos itens da fatura anulada (e referenciada na nota de crédito).
Além disso, a identificação da uma nota de crédito é dada por **NCyyyy-xxxxx**, seguindo a mesma lógica da identificação das faturas. Uma nota de crédito só pode ter os estados "Em atualização e Emitida". Neste último caso são impossíveis alterações _a nota de crédito.
Os itens de uma fatura estão associados a produtos. Cada produto _e identificado por um SKU, tem uma descrição, uma percentagem de iva (eventualmente nula), e o preço de venda unitário (sem iva). Deve ser possível registar o contribuinte associado a uma fatura. Um contribuinte tem um número de identificação fiscal, um nome (eventualmente nulo) e uma morada (eventualmente nula). Sempre que é feita uma alteração a uma fatura, por exemplo, uma mudança de estado, deve ficar registado no sistema a data em que a alteração foi feita, qual o estado antes da alteração (eventualmente o mesmo) "

Segunda fase:

"Nesta fase do trabalho pretende-se que os alunos criem uma aplicação que use diferentes frameworks de acesso a dados. A aplicação deve ter como meta a reutilização de código, fácil manutenção e eficiência, e ser independente do modo de acesso a dados."

Modelo Entidade-Associação

Para a execução deste trabalho tivemos de fazer uma alteração no modelo, pois para correr os testes seguidos, tanto do ADO.NET como do EF, e verificar os seus tempos de execução, tivemos de retirar a PK da entidade FATURA_HIST, dt_atualizacao, pois a rapidez com que os testes eram feitos

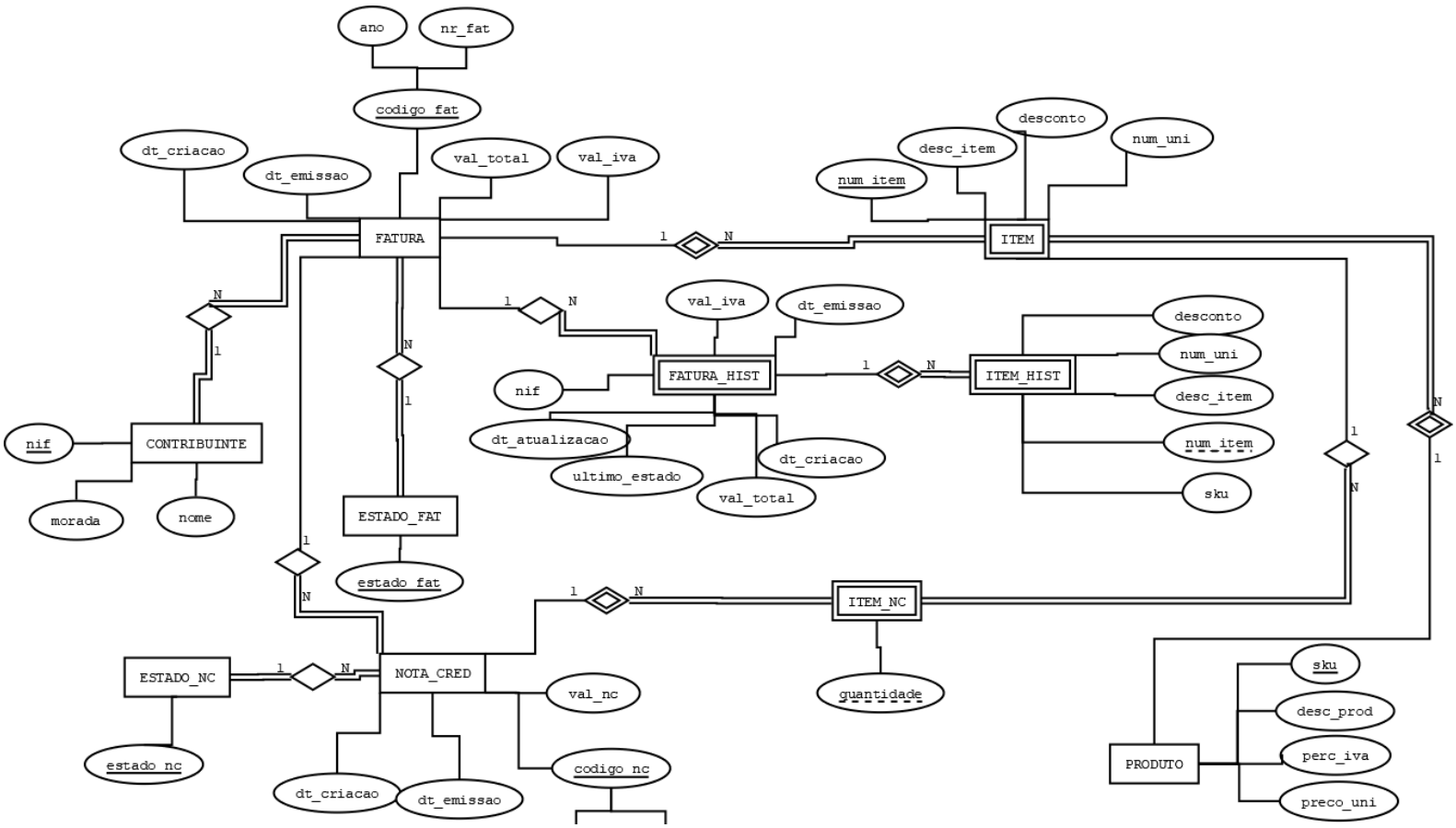

Figura 1- Novo modelo EA corrigido e alterado

provocava a criação de datas iguais inclusive os segundos, originando erro e impedindo a verificação dos tempos de execução.

Avaliação Experimental

Neste trabalho foi instalada o package "*Console Tables*" que garante a apresentação dos dados na consola em modo tabela, facilitando assim a leitura dos dados.

Exercício 1

a) Nesta alínea, criámos um projeto no *visual studio*, simples e com a ajuda das implementações realizadas em aula pelo professor, realizamos uma aplicação que implementa e testa cada umas das funcionalidades requeridas na primeira fase do trabalho da alínea f) a j), mapeando os *stored procedures* e funções criadas nessa mesma fase e chamando-as no nosso código.
b) Nesta alínea não foi chamado/utilizado nenhum stored procedure, como era pedido, por isso fizemos código "manualmente" auxiliado pelos *mappers* e *services* que criámos.
c) Nesta alínea, realizamos num só teste a chamada de todos os stored procedures necessários, realizados anteriormente, e ainda criámos mais um, nomeadamente, a alínea K da primeira fase do trabalho, para concluirmos este exercício.

Exercício 2

Criámos outro projeto no *visual studio*, utilizando uma abordagem *database first*, uma vez que já tínhamos a base de dados criada da fase 1, recorrendo a ADO.NET Entity Model que utiliza a tecnologia do Entity Framework para mapear a nossa base de dados em objetos modelos que nos facilita na realização das *queries*. Basicamente criou uma estrutura baseada no ADO.NET automaticamente. Após a criação do projeto EF fizemos o Exercício 1 nesta abordagem.

No final deste exercício juntamos os dois projetos num só para se tornar numa só aplicação, que interaja com o *user*.

Exercício 3

Uma vez que o Entity Framework foi contruído sobre a infraestrutura do ADO.NET, este não pode ser mais rápido que o mesmo. Inicialmente até é mais lento ao carregar os dados pela primeira vez. Verificamos que os testes do EF demoram cerca de 5 segundos a mais que os do ADO.NET.

No entanto, o EF não só nos facilita a manutenção do código como reduz a quantidade de código que teríamos de escrever para obter as mesmas funcionalidades da nossa aplicação caso estivéssemos só a usar ADO.NET.

Exercício 4

Neste exercício começamos por ir ao ficheiro Model1.idmx e começamos por colocar o atributo nome da entidade Contribuinte com o concurrency mode fixed. Uma vez que o nif e código_fat são chaves primárias não podemos fazê-lo aplicar este método, por isso, iremos usar o atributo "nome" para realizar o exercício.

De seguida fizemos uma classe "Exercicio4_UI" em que usa dois contextos e pede duas vezes ao utilizador para colocar o nome. Ao fazê-lo vai dar o seguinte erro:

```
O Contribuinte a alterar o que tem o nif 111222333
Insira o seu nome: ines
Insira o seu novo nome: ines nunes
Store update, insert, or delete statement affected an unexpected number of rows (0). Entities may have been modified or deleted since entities were
loaded. See http://go.microsoft.com/fwlink/?LinkId=472540 for information on understanding and handling optimistic concurrency exceptions.
Press any key to return.
```

O erro que podemos verificar é de optimistic concurrency que envolve a tentativa de guardar novos dados em uma entidade, mas esses dados foram alterados anteriormente, ou seja já não são os mesmo que foram inicialmente carregados na entidade logo como existe um conflito nos dados e é lançada uma exceção.

Exercício 5

No que diz respeito à consistência de dados, tanto Entity Framework como ADO.NET são semelhantes, uma vez que a consistência do EF uma vez que é criada através do ADO.NET também se mantém na EF.

Conclusão

Dado o problema proposto, conseguimos cumprir os objetivos requeridos, embora as ligeiras alterações na primeira fase do trabalho.

Após a realização deste trabalho, os nossos conhecimentos de ADO.NET e EF, C# e SQL foram bastante aprofundadas.

Webgrafia

- PowerPoint Presentation (isel.pt)
- c# - Entity Framework Refresh context? - Stack Overflow
- c# - How to pass table value parameters to stored procedure from .net code - Stack Overflow