Relatório de Base de Dados:

-Sistema de Gestão de uma Empresa de Vestuário Laboral

**Bruno Silva, nº mec: 80210**

**Fábio Santos, nº mec: 80317**

**Curso: Engenharia Informática**

**Disciplina: Base de Dados**

**Data: 09/06/2017**

Introdução

A aplicação a desenvolver para o trabalho de base de dados trata-se de uma aplicação capaz de fazer a gestão de uma empresa de vestuário têxtil laboral, a qual tem associadas várias fábricas filiais, cada uma com empregados associados e uma certa quantidade de diferentes materiais (panos, linhas, botões) no seu armazém. Esse material será usado para a produção de produtos têxteis laborais (como botas de agricultura, coletes, batas). Teremos também clientes que irão efetuar encomendas de produtos á empresa.

Este projeto foi feito em conjunto com a cadeira de Interação Humano-Computadores.

Análise de requisitos e contextualização da aplicação

Para a análise de requisitos, estivemos em contacto com pessoas que trabalham nesta área, de modo a obter feedback para que possamos assim perceber que funcionalidades o sistema deveria ser capaz de fazer, e que componentes devem existir para a especificação da criação da base de dados. Os requisitos funcionais da aplicação, bem como uma contextualização mais aprofundada dos componentes da aplicação seguem-se a seguir:

1. O utilizador deste sistema será um gestor que trabalha numa fábrica filial da empresa, precisando este de gerir clientes, as encomendas, a lista de fornecedores, a lista de filiais da empresa, os produtos a fabricar, os desenhos para produzir esses produtos e os materiais têxteis disponíveis no armazém da fabrica, os quais estão associados a um fornecedor.
2. Os produtos disponibilizados pela empresa são do tipo vestuário laboral, exceto peças que envolvem mais do que apenas materiais têxteis pois saem do âmbito da empresa (ex: capacete da construção civil pois envolve uma camada protetora de plástico/metal).
3. O utilizador terá em mãos a capacidade de gerir todo o processo de produção e venda de produtos elaborados pelas fábricas. Tal implica que o processo inicie com um pedido de um cliente, passe na aquisição dos materiais têxteis a fornecedores (se não existirem já em quantidade suficiente no armazém da fábrica), na produção do produto solicitado pelo cliente, e termine na entrega de encomendas a clientes.
4. As várias filiais da empresa terão a sua informação associada na base de dados. Essa informação inclui informação fiscal, contactual e de localização. Tal informação poderá, por exemplo ser usada na faturação para os clientes, consultada ou ainda alterada.
5. Os materiais têxteis que o utilizador poderá introduzir no sistema são: pano, linha e acessórios como botões, molas, fechos, elásticos, fitas de velcro (peças mais usuais) entre outros. Com todos estes materiais deverá ser possível criar qualquer tipo de vestuário laboral (produto personalizado) que esteja dentro do âmbito de uma fábrica de vestuário laboral.
6. O utilizador do sistema terá a liberdade de adicionar produtos base ao sistema, que irão indicar as instruções de produção necessárias para produzir um produto final, denominado neste trabalho por produto personalizado. A partir deste produto base é possível ter produtos personalizado que estão de acordo com as especificações estabelecidas pelo produto base, mas com cores e tamanhos diferentes. Informações relativas ao autor e data de criação/atualização do desenho também estão presentes no produto base. Imagens conceptuais do produto base também poderão ser introduzidas no sistema.
7. O utilizador poderá adicionar produtos personalizados ao sistema, produtos esses que serão as peças de vestuários laboral produzidos e vendidos a clientes. No entanto, cada produto personalizado deverá ter a ele associado um produto base estabelecido, e deverá, como a lei indica, possuir uma etiqueta com informações de composição, normas, e país de fabrico do produto. Além disso, produtos personalizados deverão possuir informação sobre o preço de venda ao cliente, cor, tamanho do produto e unidades em armazém.
8. Qualquer filial não pode produzir um dado produto sem ter os materiais necessários para a produção do mesmo em armazém. Quando se produzir um produto personalizado, deverá remover-se do stock a quantidade de cada material utilizado na produção do mesmo.
9. Um empregado da empresa pode ter diferentes permissões sobre o sistema. Se for um gestor de stock apenas poderá gerir as encomendas e os clientes. Um gestor de produção pode gerir materiais, produtos base e personalizado. Um gestor de recursos humanos gere os funcionários da empresa, e um gestor de empresa gere as fábricas filiais e os fornecedores associados á empresa. Um utilizador pode ter mais do que um destes tipos de aplicação.
10. Deverá ser possível que o utilizador do sistema tenha acesso a informação de clientes como a sua informação pessoal, contactual, bancária e de localização.
11. O utilizador poderá apenas vender produtos que estejam em stock. Tal significa que a entrega de uma dada encomenda só é efetuada depois de os itens da encomenda estarem inteiramente produzidos e prontos para venda.
12. Deverá ser possível observar o histórico de compras de um dado cliente.
13. Os estados da encomenda podem variar entre “aceite”, “entregue” e “cancelado”.

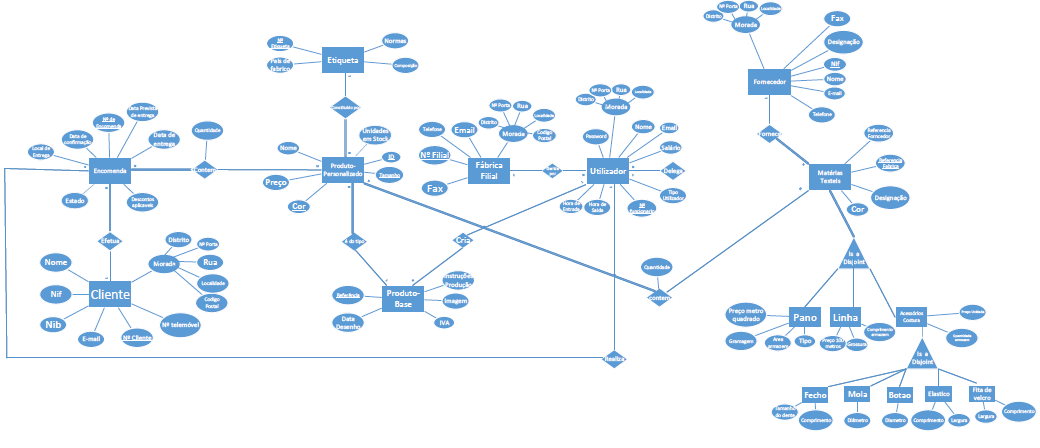
Note-se que, um produto base significa um desenho guia para a produção de um produto, a partir do qual podemos ter um produto com cores e tamanhos diferentes, sendo este o produto que é produzido e vendido ao cliente, que é denominado por produto personalizado na base de dados. No entanto, na interface da aplicação, chamamos “desenho de produto” ao produto base e simplesmente “produto” ao produto personalizado.

Diagrama entidade – relação

Dada a análise de requisitos anteriores, devemos agora mapear todas as entidades e relações do contexto deste problema para uma visão mais abstrata da estrutura do tipo de dados, usando para esse efeito o diagrama entidade-relação.

Em resumo, as entidades base envolvidas no contexto do problema, são:

* Cliente, que tem informação básica e de contacto associada.
* Encomenda, efetuada por um cliente e que contém vários produtos personalizados
* Produto personalizado, constituído por 1 ou mais materiais, e baseado num produto base, que tem também associado a ele uma etiqueta;
* Produto base, criado por um gestor de produção;
* Utilizador, que será um funcionário da empresa, estando este a trabalhar numa dada filial, e que pode ser um gestor de produção, responsável pela produção de produtos, ou gestor de encomendas, que gere as encomendas de clientes, ou um gestor de recursos humanos, responsável por gerir os funcionários da empresa, ou ainda um funcionário chefe da filial, que pode gerir tudo.
* Fábrica Filial da empresa, gerida por um funcionário chefe
* Material Têxtil, usado em produtos personalizados, e fornecido por empresas fornecedoras. O material têxtil divide-se em vários tipos: pano, linha e acessórios de costura, e dentro destes temos mais tipos de materiais. Um material só pode ser de um sub-tipo, isto é, não podemos ter um material que é pano e linha ao mesmo tempo. Mas um Material é obrigatoriamente um subtipo, ou seja, ou é um pano, ou uma linha, ou um acessório de costura;
* Fornecedor, que fornece os materiais têxteis á fábrica filial que os solicitou;

Aplicando estas relações, os atributos de cada entidade (alguns deles já mencionados anteriormente), e tendo em atenção as multiplicidades, as entidades fortes e fracas, e a obrigatoriedade de participação na relação, obtemos o seguinte DER (também é disponibilizado um pdf onde é possível ver com mais facilidade o DER):

Modelo Relacional e Normalização

Partindo do DER anteriormente apresentado, foram aplicados os passos apresentados nos slides da aula teórica da cadeira que falam do modelo relacional. Começou-se, portanto, por criar relações para cada entidade regular no der, incluindo os seus atributos.

Como não existem entidades fracas, nem relações de 1 para 1, avançou-se para o passo 4, onde para cada entidade do lado N, se adiciona uma chave estrangeira que referencia a chave primária da entidade do lado N. A maior parte das relações do DER são de 1:N, por isso isto foi aplicado á grande maioria das relações, permitindo assim a que, por exemplo, a encomenda tenha uma chave estrangeira que indique o nº de cliente a quem essa encomenda pertence.

Nas relações N:N, onde se criou uma nova tabela do relacionamento, que incluiu as chaves estrangeires que referenciam as chaves primárias de ambas as relações, e que combinadas formam a chave primária da nova tabela. Por exemplo, para a relação N:N encomenda que contém produtos personalizados, foi criada uma tabela de nome conteúdo\_encomenda, que contém como chave estrangeira a chave primária de encomenda (nº encomenda) e a chave primária do produto personalizado (cor, tamanho, referência do produto personalizado, ID), e o atributo “quantidade” que faz parte do relacionamento.

Não existem atributos multivalor, nem relacionamentos n-ários.

Finalmente, para os materiais têxteis, criou-se essa tabela do modelo relacional, e criaram-se 3 novas tabelas, cujas chaves primárias são chaves estrangeiras que referenciam a chave primária da tabela de Materiais Têxteis (referência fábrica), formando assim as tabelas pano, linha e acessórios de costura, que têm também atributos específicos a cada uma. Para a relação Acessórios de costura é aplicado o mesmo método, criando assim mais 5 tabelas para os diferentes sub-tipos de acessórios.

Passamos á normalização do modelo relacional criado anteriormente, de modo a reduzir a redundância que possa aí existir.

Primeiramente verificou-se em cada relação se existe algum atributo multi-valor que pudesse ser decomposto noutra relação, mas não existe nenhum atributo multi-valor em nenhuma relação, por isso, o modelo relacional já se encontra na 1ª forma normalizada.

Para a 2ª forma normalizada, verificou-se se existem dependências parciais nas relações, mas todas os atributos de todas as relações dependem apenas da chave primária da relação. Por exemplo no caso do produto personalizado separamos os seus detalhes como a cor e a etiqueta para uma tabela chamada produto-personalizado-detalhes uma vez que a cor e a etiqueta de um produto estão apenas dependentes da referencia e do modelo de um dado produto e não do seu tamanho. Por outro lado coisas como preço e unidades em armazém estão dependentes dos 3 atributos chave (referencia, modelo e tamanho).

Para atingir 3ª forma normalizada, verificou-se a existência de dependências transitivas nas relações. Existe uma dependência transitiva nos atributos de localidade e distrito, que dependem da chave não primária código postal, nas relações cliente, utilizador, filial e fornecedor, portanto foi criada uma nova relação “zona” com a chave primária código postal, e os atributos concelho e distrito, sendo que código postal passa a ser uma chave estrangeira que referencia esta chave primária nas relações referidas anteriormente.

A BCNF foi bastante simples de se alcançar. Bastou-nos para isso eliminar os atributos repetidos entre 2 relações que tenham sido produto de uma normalização da 3fn. Por exemplo para o caso da “zona” bastou nos eliminar os atributos localidade e distrito da tabela “cliente” pois estes já estavam contidos na “zona”.

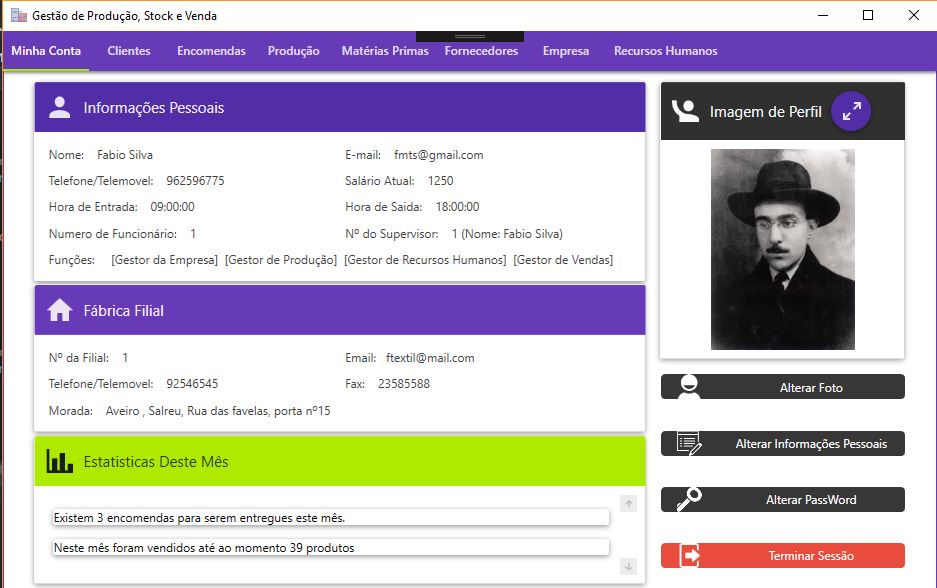
SQL DDL

As tabelas sql foram feitas a partir do modelo relacional apresentado anteriormente. O código-postal foi separado em 2 campos, um para a 1ª parte, e outro campo para a 2ª parte. Para os atributos número de cliente, número de filial, e atributos semelhantes foi usada a propriedade indentity. Em certos atributos foram feitas também algumas verificações (check), tais como verificar que a quantidade de um material nunca é menor que 0, ou que a hora de entrada é menor que a hora de saída.

Após as tabelas serem criadas adiciona-se na tabela “ZONA” todos os códigos postais a nível nacional, que foram obtidos no sito dos CTT, permitindo assim obter o distrito e localidade de um cliente (por exemplo) a partir do seu código-postal. Colocamos também na tabela “ESTADO” todos os estados possíveis que uma encomenda pode tomar (entregue, cancelada, aceite) e todos os locais de entrega de uma encomenda (ao domicílio ou na fábrica) na tabela “[LOCAIS-ENTREGA-ENCOMENDA]”. Na tabela “[TIPO-UTILIZADOR]” colocamos todos os tipos de utilizadores possíveis que podem existir na plataforma. Estes dados introduzidos são fixos e não podem ser modificados na aplicação.

Para que seja possível iniciar a aplicação, adicionamos também um utilizador chefe com permissões para tudo e uma fábrica filial.

SQL DML e descrição da app

Na página inicial da plataforma é apresentada informação pessoal relativa ao utilizador que fez o login, e informação da fábrica filial onde o utilizador trabalha, bem como estatísticas importantes da empresa, tal como o número de produtos vendidos.

Ao navegar para um outro menu da tab de cima, vemos uma listagem cujo conteúdo varia consoante o menu, e em todas essas páginas de listagem é possível pesquisar por determinados atributos. Por exemplo, na página de listagem de clientes é possível pesquisar por NIF de cliente, nome de cliente ou número de cliente, e limpar os resultados dessa procura clicando na lupa que se encontra no lado esquerdo da caixa de texto. Na listagem normal, é utilizado uma query do tipo ‘select… from…’, sendo aplicado joins com outras tabelas sempre que é necessário e pertinente obter informação extra que não se encontra na primeira tabela. Por exemplo, um join na tabela “CLIENTE” e a tabela “ZONA” para obter o distrito e localidade do cliente, ou join entre um utilizador e os dados da fábrica filial de onde ele trabalha.

Na listagem de uma pesquisa por atributos é usado o query utilizado na listagem normal, mas com um ‘where = x’ quando o atributo pelo qual pretendemos procurar (x) é um número, ou um ‘where like %x%’ quando pretendemos procurar por uma string, tal como um nome para obter todos os atributos pelo qual estamos a pesquisar que contém ‘x’.

Nas páginas de registos a informação registada/selecionada pelo utilizador é inserida na tabela respetiva através da query ‘insert into x ‘, tendo o cuidado de inserir um ‘NULL’ nos casos em que o utilizador optar por não preencher os campos opcionais. Na página de registo de um produto é necessário escolher criar uma nova etiqueta ou usar uma etiqueta já existente. No caso de criar uma nova etiqueta esta será inserida primeiro na base de dados, e só e depois o produto, ao qual será associado o número da etiqueta criada anteriormente. Como o número de etiqueta é incrementado por um identity, é usado o comando IDENT\_CURRENT para obter o valor atual do identity na coluna especificada, neste caso, a etiqueta.

Nas páginas de edição as caixas de texto são automaticamente preenchidas com a informação atual daquilo que foi selecionado, e a partir daí é possível editar essa informação ou preencher informação opcional que não tinha sido preenchida. É usado o query ‘update x set ….’ com todos os dados do formulário, quer estes tenham sido editados por clientes ou não. A validação do input do utilizador é feita ao nível da aplicação, de modo a garantir coerência na base de dados, tais como garantir que o utilizador não deixa um campo em branco de um atributo NOT NULL, ou que o input não tem um tamanho maior do que o tamanho máximo definido na base de dados no caso dos dados do tipo varchar.

Apenas podem ser removidos clientes, pelo que se usa para o efeito o query ‘delete CLIENTE WHERE …..’ .

User Defined Functions

Foram usadas UDF’s sempre que foi necessário obter um único valor de uma tabela, tais como o número de produtos vendidos e dinheiro gerado por mês por exemplo (informações estas que são apresentadas nas estatísticas da página inicial), ou obter um valor derivado da soma de vários dados de tabelas, tal como o número de encomendas para o mês atual. Nestes casos exemplificados foram usadas udf’s escalares, pois como já foi dito apenas precisamos de um único valor de retorno. Alguns exemplos de udf’s escalares usadas são:

* getTipoMaterial, que recebe uma referência de um material têxtil e verifica nas tabelas dos diferentes sub-tipos de materiais se essa referência lá existe. Quando a referência é encontrada, é retornado um varchar com o nome do tipo de material, portanto se a referência estivar na tabela “PANO”, a udf retorna “Pano”;
* getEncomendasMes, que retorna o número de encomendas cuja data de entrega prevista aponta para o mês atual. Isto é usado nas estatísticas da página principal;
* getQuantidadeMaterial, que dado a referência de um material retorna um int com a quantidade em stock do material.

Existe um caso em que pretendemos obter uma tabela e não um único valor, sendo para o efeito usada uma udf inline table-valued com o nome getProductMaterials, que dado um produto, retorna uma tabela com os materiais necessários para o produzir.

Stored Procedures

Em certas funcionalidades mais complexas da aplicação, foram utilizadas stored procedures, pois assim é possível ter uma lógica mais complexa e ter várias querys para ser possível realizar essas funcionalidades pretendidas. Além disso, por vezes, queremos modificar certas informações em tabelas, algo que não é possível com ufd’s daí a escolha de stored procedures. Exemplos de utilização de stored procedures na aplicação seguem-se em baixo:

* usarMaterial, que dado um material, e uma quantidade, subtrai ao valor atual da quantidade em armazém do material a quantidade passada como argumento. Para saber que tipo de material é que foi passada como argumento usamos a função getTipoMaterial. Esta stored procedure é usada na produção de um produto.
* produzirProduto, que recebe a chave primária de um produto e a quantidade de produto que se quer produzir. Primeiro começa-se por verificar se existe uma quantidade de material suficiente em stock (usando como auxilio uma udf já explicada anteriormente, getQuantidadeMaterial). Caso exista, para cada material necessário para a produção, será subtraída a quantidade de material em stock com a quantidade de material necessário para a produção usando a stored procedure referida no ponto anterior. Por fim, usando uma query ‘update’ é atualizado a quantidade de produto em armazém. É usado um cursor para iterar sobre as rows das tabelas dos materiais, e esse cursor é aberto 2 vezes, uma para a validação, e outra para a atualização da quantidade de material em stock. Se foi possível produzir o produto, teremos um output de 1, caso contrário o output é 0.
* entregarEncomenda, que dado uma encomenda, verifica para cada produto associado se existe quantidade de produto em armazém suficiente para a quantidade de produto especificado na encomenda (esta validação é feita através de um cursor). Em caso afirmativo e caso a encomenda não tenha o estado ‘entregue’ ou ‘cancelado’, entregamos o produto, mudando o seu estado para ‘entregue’ e retirando do armazém a quantidade de produto que foi usada na encomenda.

Indíces

Indíces são estruturas de dados que oferecem uma forma mais rápida de aceder aos dados, no entanto, uma má escolha de índices pode fazer com que se demore mais tempo a aceder aos dados do que um acesso sem índices. Assim foi necessário escolher com critério onde eram necessários aplicar índices. Por isso decidimos aplica-los em tabelas onde não se fazem inserções ou atualizações, tal como a tabela ZONA, que contém todos os códigos-postais a nível nacional, e para cada código, o distrito e a localidade desse código. Esta tabela é apenas consultada, por isso foi adicionado um multi-level índex que contém os dois atributos que fazem a chave primária desta tabela.

Existem algumas tabelas que irão receber poucas inserções, tais como as tabelas “UTILIZADOR“, FORNECEDOR” e “[FABRICA-FILIAL]”, pois não se espera que os utilizadores desta plataforma façam muitas inserções com muita frequência nestas tabelas, ao contrários das tabelas de clientes, produtos personalizado e base, materiais e encomendas, onde se espera que sejam feitas inserções e updates com mais frequência, e por isso optámos por não adicionar índices a estas tabelas. Nas tabelas “FORNECEDOR” e “UTILIZADOR” adicionámos um índice primário. Na tabela “[FABRICA-FILIAL]” a chave primária desta já tem por defeito uma unique clustered índex, e esta não sofreu alterações pois esta é a melhor escolha para um índice, visto a chave primária desta tabela ser um identity que incrementa por 1 sempre que se faz uma nova inserção, o que resulta num bom desempenho com este tipo de índices. Visto neste caso termos inserções ordenadas, colocámos o fill factor próximo de 100, tendo assim cada página quase preenchida de modo a não desperdiçar tanto espaço.

Transações

Foram usadas transações em algumas stored procedures mais complexas, que tinham um conjunto de operações e querys nelas associadas que liam e modificavam dados, daí a necessidade das transações de modo a garantir que todos os princípios ACID se verifiquem, isto é, estas operações correm de forma atómica, mantendo a integridade da BD, evitando que outros processos de escrita interfiram nestas operações e que as mudanças efetuadas sejam visíveis para outras transações na BD.

Tendo isto em conta, adicionou-se transações nas stored procedures de entregar encomenda e cancelar encomenda, produzir produto, adicionar material, pois apenas queremos que estas operações efetuam as alterações na BD caso não ocorra nenhum erro, isto é, as operações são todas feitas, ou nenhuma é feita. Também são usadas transações para a stored procedure de registo de um produto personalizado, pois inserimos em 2 tabelas ([PRODUTO-PERSONALIZADO e [PRODUTO-PERSONALIZADO-DETALHES]) a informação relativa ao produto, e queremos ter a certeza de que a informação é adicionada em ambas as tabelas.