Faculdade de Engenharia da Universidade do Porto

Empresa de Comércio Online

Base de Dados

2016/2017 -- Mestrado Integrado em Engenharia Informática e Computação:

Turma 4, Grupo 5:

Afonso Bernardino da Silva Pinto

(up201503316@fe.up.pt)

João Manuel Guimarães Fontes

(up201502863@fe.up.pt)

Tomás Sousa Oliveira

(up201504746@fe.up.pt)

Docentes:

Carla Lopes ctl@fe.up.pt João Moreira jmoreira@fe.up.pt Henrique Cardoso hlc@fe.up.pt

Resumo

Criado no âmbito da disciplina Base de Dados leccionada no 2º ano do Mestrado Integrado em Engenharia Informática e Computação, uma empresa de comércio online é o tema deste projeto que tem como objetivos a criação e interrogação de uma base de dados.

Palavras-Chave

Empresa de Comércio Online; UML; Modelo Conceptual; Esquema Relacional; Formas Normais; SQLite; Base de Dados; FEUP.

Agradecimentos

Este projeto foi resultado de diversas contribuições e colaborações, dada de forma direta e indireta, mas todas elas essenciais à sua realização. Gostaríamos assim de expressar os nossos sinceros agradecimentos a todos os que tornaram possível este trabalho.

Ao professor João Moreira pela orientação dada e valioso acompanhamento constante durante o desenvolvimento do projeto.

Introdução

Este projeto foi realizado no âmbito da unidade curricular Base de Dados, do 2º ano do Mestrado Integrado em Engenharia Informática e Computação da Faculdade de Engenharia da Universidade do Porto.

O objetivo deste projeto é implementar e interrogar uma base de dados. Para tal será necessário criar o modelo conceptual, mapear esse modelo para um esquema relacional, implementar esse esquema numa base de dados SQLite, introduzir dados, e, por fim, interrogar a base de dados.

Optamos por implementar uma empresa de comércio online que não só permitisse aos seus utilizadores efectuar comprar online através de casa mas também através das lojas físicas existentes com suporte nos mesmos meios.

O presente relatório servirá para explorar as funcionalidades da empresa de comércio online e fornecer detalhes sobre a implementação da sua base de dados.

Índice

1. Modelo Conceptual

1.1 Descrição do Contexto

Uma empresa de comércio pretende informatizar o seu serviço de vendas. De cada cliente interessa armazenar o seu nome, nif, email, género, número de telefone, morada e código postal. Sobre os funcionários da empresa pretende-se guardar o nome, morada, código postal, o telefone, a especialidade desse funcionário e o respetivo custo horário.

Quanto a cada loja da companhia é necessário saber a morada e os funcionários lá empregados. Cada loja guarda o registo de todos os produtos em stock. Sobre cada produto sabe-se: o código, a categoria, o nome, o custo unitário e a quantidade disponível.

Para cada compra efetuada é guardada informação do cliente, a data e hora de compra e de entrega, e ainda: (1) a quantidade de cada produto adquirido; (2) o funcionário responsável pela mesma; e (3) o número de quilómetros e portes de entrega se aplicável.

1.2 Diagrama UML

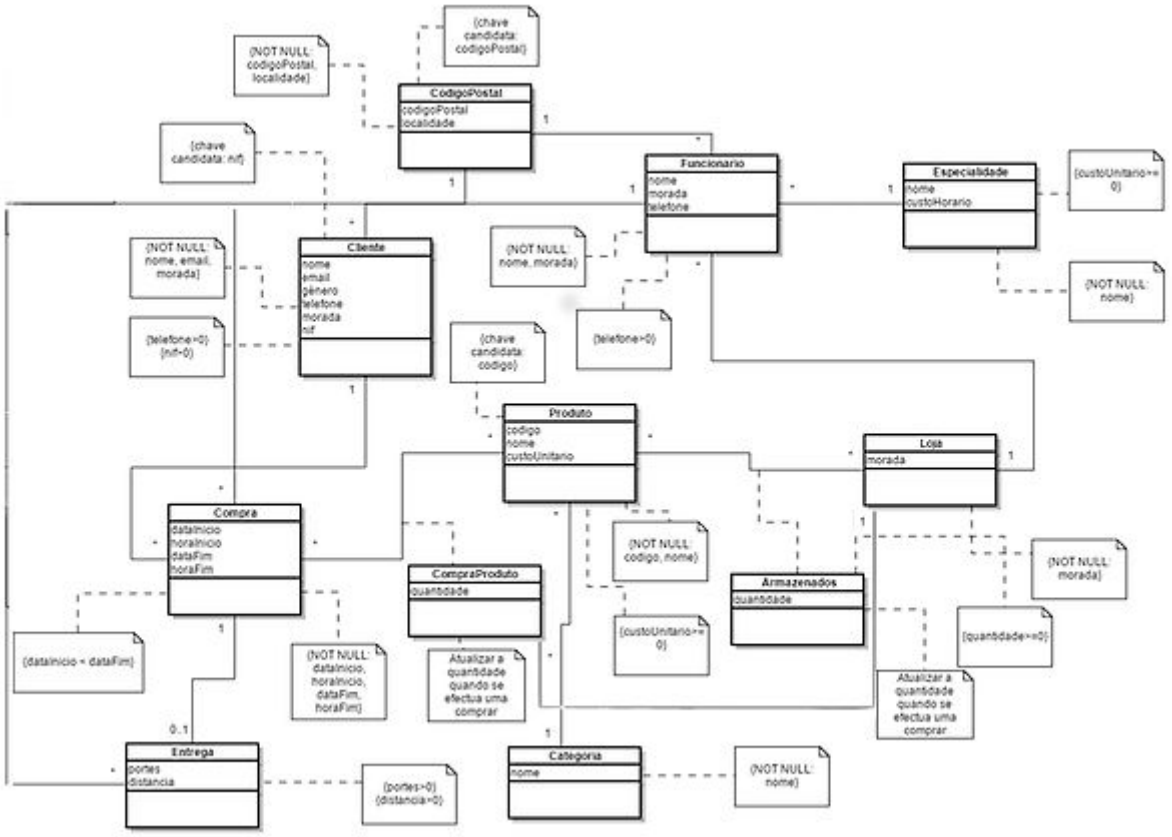

Figura 1. Diagrama UML, disponível em goo.gl/UZUQGs (versão inicial)

2. Esquema Relacional

Categoria(idCategoria, nome)

Produto(codigo, nome, custoUnitário, idCategoria->Categoria)

CodigoPostal(codigoPostal, localidade)

Cliente(nif, nome, email, género, telefone, morada, codigoPostal->CodigoPostal)

Loja(idLoja, morada)

Armazenados(codigo->Produto, idLoja->Loja, quantidade)

Especialidade(idEspecialidade, nome, custoHorario)

Funcionario(idFuncionario, nome, morada, telefone, idEspecialidade->Especialidade, codigoPostal->CodigoPostal)

Compra(idCompra, dataInicio, horaInicio, dataFim, horaFim, idFuncionario->Funcionario, nif->Cliente)

CompraProduto(idCompra->Compra, codigo->Produto, quantidade, idLoja->Loja)

Entrega(idEntrega, portes, distância, idCompra->Compra (unique), idFuncionario->Funcionario)

3. Formas Normais e Análise de Dependências Funcionais

3.1 Dependências Funcionais

Categoria:

idCategoria -> nome

Produto:

codigo -> nome, custoUnitario, idCategoria

Codigo Postal:

codigoPostal -> localidade

Cliente:

nif -> nome, email, genero, telefone, morada, codigoPostal

morada -> codigoPostal

Loja:

idLoja -> morada

Armazenados:

codigo, idLoja -> quantidade

Especialidade:

idEspecialidade -> nome, custoHorario

Funcionario:

idFuncionario -> nome, morada, telefone, codigoPostal, idEspecialidade

morada -> codigoPostal

Compra:

idCompra -> dataInicio, horaInicio, dataFim, horaFim, nif, idFuncionario

CompraProduto:

idCompra, codigo -> quantidade, idLoja

Entrega:

idEntrega -> portes, distancia, idCompra, idFuncionario

3.2 Violações à Terceira Forma Normal

Cliente(<u>nif</u>, nome, email, género, telefone, morada, codigoPostal->CodigoPostal)

Não se encontra na 3ªForma Normal.

O codigoPostal depende da morada e por sua vez a morada depende do nif (chave primária).

Funcionario(<u>idFuncionario</u>, nome, morada, telefone, idEspecialidade->Especialidade, codigoPostal->CodigoPostal)

Não se encontra na 3ªForma Normal.

O codigoPostal depende da morada e por sua vez a morada depende do idFuncionario (chave primária).

3.3 Violações à Forma Normal Boyce-Codd

Cliente(<u>nif</u>, nome, email, género, telefone, morada, codigoPostal->CodigoPostal)

Não se encontra na BCNF.

morada -> codigoPostal, mas morada não é uma chave.

Funcionario(<u>idFuncionario</u>, nome, morada, telefone, idEspecialidade->Especialidade, codigoPostal->CodigoPostal)

Não se encontra na BCNF.

morada -> codigoPostal, mas morada não é uma chave.

4. SQLite

4.1 criar.sql

4.2 povoar.sql

5. Interrogações da Base de dados

1. Listagem de todos os clientes e número de compras: int1.sql

```
.mode    columns
.headers     on
.nullvalue   NULL

-- Listagem de todos os clientes e número de compras
select cliente.nome, count(compra.nif) as nCompras
from cliente inner join compra on cliente.nif = compra.nif
group by cliente.nome
order by nCompras desc;
```

2. Listagem dos três produtos mais requisitados: int2.sql

```
.mode    columns
.headers     on
.nullvalue   NULL

-- Listagem dos três produtos mais requisitados

Select Produto.codigo, nome, SUM(quantidade)
FROM CompraProduto, Produto
WHERE(Produto.codigo = CompraProduto.codigo)
GROUP BY nome
ORDER BY SUM(quantidade) desc limit 3;
```

3. Quais as Lojas onde não vendem barbies? int3.sql

```
.mode    columns
.headers     on
.nullvalue   NULL

-- Quais as Lojas onde não vendem barbies?

Select idLoja
FROM Loja
WHERE idLoja NOT IN
(SELECT idLoja
FROM Armazenados
WHERE codigo = 1);
```

4. Listagem de todas as compras efetuadas depois do dia 20 de Abril de 2017: int4.sql

```
.mode   columns
.headers    on
.nullvalue  NULL

-- Listagem de todas as compras efetuadas depois do dia 20 de Abril de 2017

Select *
FROM Compra
WHERE  (dataFim > '2017-03-20');
```

5. Qual o custo unitário médio, o valor total, o número de unidades distintas de produtos, e o valor do produto mais caro e do mais barato?: int5.sql

```
.mode   columns
.headers    on
.nullvalue  NULL

-- Qual o custo unitário médio, o valor total,
-- o número de unidades distintas de produtos,
-- e o valor do produto mais caro e do mais barato?

SELECT AVG(custoUnitario) "Média", SUM(custoUnitario*quantidade) "Valor total",
(SELECT COUNT(*) FROM Produto) "N de produtos", MIN(custoUnitario) "Poduto mais barato",
MAX(custoUnitario) "Produto mais caro"
FROM Produto, Armazenados
WHERE(Produto.codigo = Armazenados.codigo);
```

6. Que computadores estão disponíveis e em que loja? int6.sql

```
.mode   columns
.headers    on
.nullvalue  NULL

-- Que computadores estão disponíveis e em que loja?

select distinct produto.nome, produto.custoUnitario, loja.idLoja, loja.morada
from produto, loja
inner join armazenados on (produto.codigo = armazenados.codigo)
where produto.nome like 'computador%';
```

7. Cliente responsável pela compra mais cara: int7.sql

```
.mode   columns
.headers    on
.nullvalue  NULL

-- Cliente responsável pela compra mais cara

Select Cliente.nome, CompraProduto.codigo, custoUnitario*quantidade
FROM Cliente, Produto, Compra, CompraProduto
WHERE (Produto.codigo = CompraProduto.codigo
    AND Compra.idCompra = CompraProduto.idCompra
    AND Cliente.nif = Compra.nif)
ORDER BY custoUnitario*quantidade DESC limit 1;
```

8. Qual a compra realizada há mais tempo, contabilizando a data actual?: int8.sql

```
.mode   columns
.headers    on
.nullvalue  NULL

-- Qual a compra realizada há mais tempo, contabilizando a data actual?

SELECT idCompra, julianday('now')-julianday(dataFim) 'Days'
FROM Compra limit 1;
```

9. Qual o funcionário com o segundo custo Horário mais alto? : int9.sql

```
.mode   columns
.headers    on
.nullvalue  NULL

-- Qual o funcionário com o segundo custoHorário mais alto?

select funcionario.nome, custoHorario
from funcionario
join especialidade on (especialidade.idespecialidade = funcionario.idespecialidade)
where custoHorario = (select max(custohorario)
from especialidade
where custoHorario
not in (select max(custohorario)
from especialidade));
```

10. Quais as localidades onde mora alguém, seja ele cliente ou funcionário? int10.sql

```
.mode   columns
.headers    on
.nullvalue  NULL

-- Quais as localidades onde mora alguém, seja ele cliente ou funcionário?

SELECT localidade
FROM CodigoPostal, Cliente
WHERE CodigoPostal.codigoPostal=Cliente.codigoPostal
UNION
SELECT localidade
FROM CodigoPostal, Funcionario
WHERE CodigoPostal.codigoPostal=Funcionario.codigoPostal;
```

6. Adição de gatilhos à base de dados

1. Gatilho que retira da Loja o número de produtos adquiridos logo após a uma compra

```
-- Gatilho que retira da Loja o número de produtos adquiridos logo após a uma compra

CREATE TRIGGER atualizaStock
AFTER INSERT ON CompraProduto
FOR EACH ROW
BEGIN
    UPDATE Armazenados SET quantidade = quantidade - NEW.quantidade
    WHERE Armazenados.codigo = NEW.codigo
    AND Armazenados.idLoja = NEW.idLoja;
END;
```

- gatilho1_adiciona.sql
- gatilho1_remove.sql
- gatilho1_verifica.sql

2. Gatilho que atualiza a quantidade de um produto que esteja quase a esgotar na Loja (adiciona 5 unidades ao produto, caso tenha menos de 2 unidades):

- gatilho2_adiciona.sql
- gatilho2_remove.sql
- gatilho2_verifica.sql

```
-- Gatilho que atualiza a quantidade de um produto que esteja quase a esgotar na Loja

CREATE TRIGGER atualizaProduto
BEFORE INSERT ON CompraProduto
FOR EACH ROW
WHEN ((SELECT COUNT(*) FROM
        Armazenados
        WHERE Armazenados.codigo = NEW.codigo
        AND Armazenados.idLoja = NEW.idLoja
        AND Armazenados.quantidade - NEW.quantidade <= 1) >= 1)
BEGIN
    UPDATE Armazenados SET quantidade = quantidade + 5
    WHERE Armazenados.codigo = NEW.codigo
    AND Armazenados.idLoja = NEW.idLoja;
END;
```

3. Gatilho que impede criar uma Especialidade caso o seu custo horário seja inferior ao salário mínimo:

```
-- Gatilho que impede criar uma Especialidade caso o
-- seu custoHorário seja inferior ao salário mínimo

CREATE TRIGGER minimumWage
BEFORE INSERT ON Especialidade
FOR EACH ROW
BEGIN
    Select CASE
        WHEN New.custoHorario <= 2
    THEN RAISE(ABORT, 'Salario invalido')
END;
END;
```

- gatilho3_adiciona.sql
- gatilho3_remove.sql
- gatilho3_verifica.sql

7. Avaliação da Unidade Curricular

Através desta unidade curricular, conseguimos familiarizar-mos-nos com a biblioteca SQLite e adquirir/cimentar conceitos sobre a implementação e interrogação de base de dados.

8. Autoavaliação

Afonso Pinto

Contribuição: 47.5%

João Fontes

Contribuição: 5%

Tomás Oliveira

Contribuição: 47.5%