Relatório Final

Bases de Dados - BDAD

Cláudio Monteiro
Filipa Ivars Silva
José Luís Magro

1 de Junho de 2014

Descrição de contexto

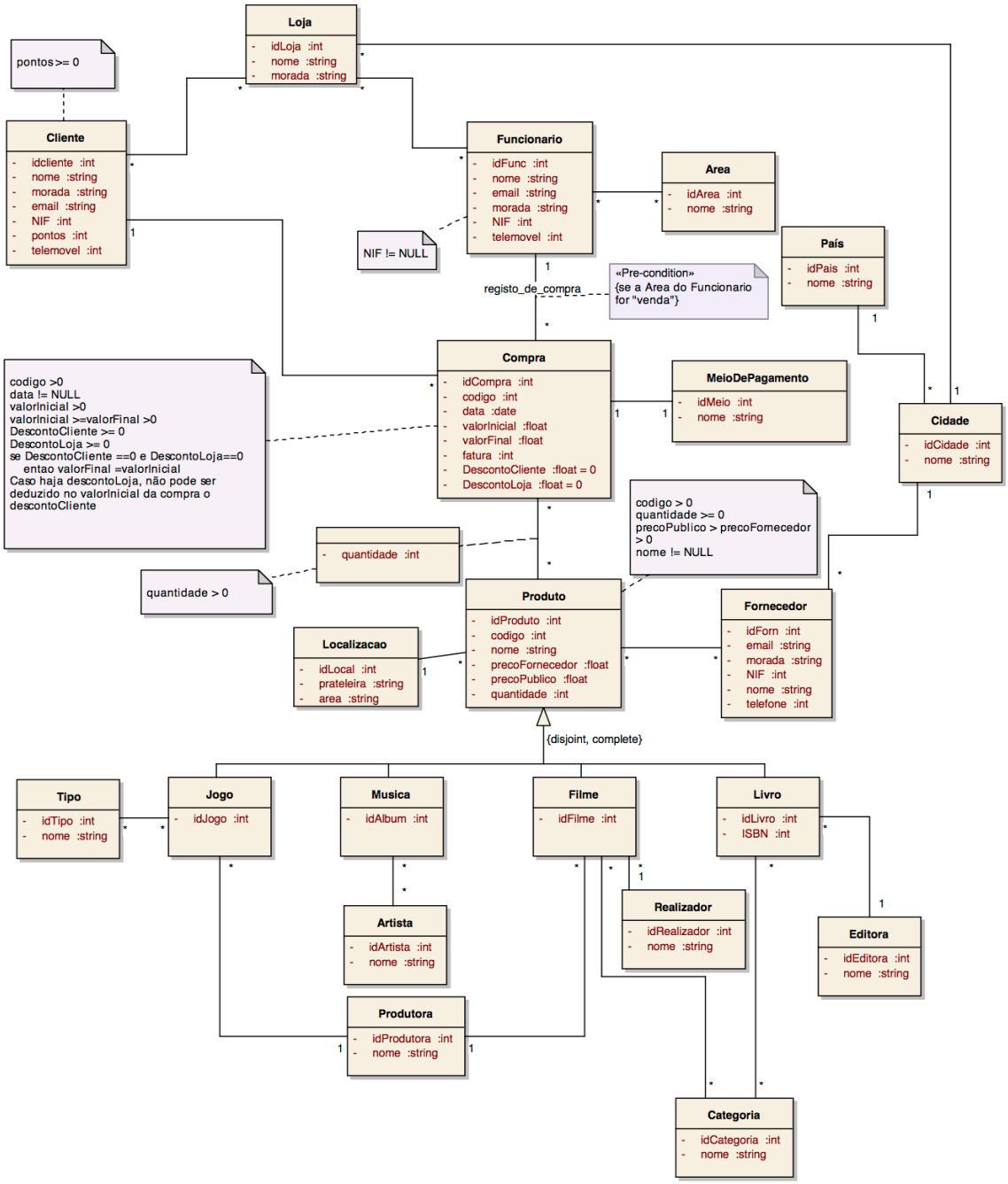
Este trabalho pretende dar uma estrutura e organização a uma base de dados para uma loja de entretenimento, i.e., que vende produtos tais como livros, filmes, jogos e música. O tema proposto é baseado nas chamadas "lojas de entretenimento" e como tal tem como objectivo estruturar a base de dados de uma loja desse género. Esta base de dados irá juntar todos os elementos que fazem parte de uma dessas lojas, tais como, informação sobre a loja e seus clientes, fornecedores e todo o tipo de produtos tais como filmes ou jogos incluindo informação detalhada sobre estes. O conceito do nosso trabalho passa por criar uma da base de dados que:

- controle o registo de clientes que possuem uma conta na loja;
- guarde o registo das compras e pagamentos, associados às mesmas, e efetuados pelo cliente (com ou sem ficha cliente agregada) à loja;
- armazene informações acerca dos funcionários;
- trate da catalogação dos produtos disponíveis na superfície, e
- registe as informações dos fornecedores e respetivas encomendas feitas pela loja aos mesmos

Descrição dos principais conceitos

- **Loja** - representação física de uma loja do franchise
- **Cliente** - representação física de uma pessoa registada na área cliente da loja (cartão cliente e pontos associados)
- **Funcionario** - representação física de uma pessoa empregada na loja
- **Area** - área/função de cada empregado na loja (por exemplo venda, armazém)
- **Compra** - armazena os dados relativos ao ato da compra
- **Pagamento** - registo do resultado do ato da compra (fatura a emitir/emitida e meio de pagamento)
- **Produto** - registo dos dados de cada produto e da sua quantidade em stock
- **Localização** - dados de onde cada produto se encontra na loja de forma a facilitar a procura
- **Fornecedor** - representação física de uma entidade que fornece cada produto
- **Jogo**, **Musica**, **Filme** e **Livro** - representação física de um jogo, um álbum de música, um filme e um livro, respetivamente

Diagrama de classes UML



Esquema Relacional

- Cliente (idCliente, nome, morada, NIF, pontos, telemovel, email);
- Funcionario (idFunc, nome, email, morada, NIF, telemovel);
- Area (idArea, nome);
- Pais (idPais, nome);
- MeioDePagamento (idMeio, nome);
- Fornecedor (idForn, nome, email, morada, NIF, telefone);
- Localizacao (idLocal, prateleira, area)
- Tipo (idTipo, nome);
- Musica (idProduto);
- Artista (idArtista, nome);
- Produtora (idProdutora, nome);
- Realizador (idRealizador, nome);
- Editora (idEditora, nome);
- Categoria (idCategoria, nome);
- Loja (idLoja, nome, morada, idCidade->Cidade);
- Cidade (idCidade, nome, idPais->Pais);
- Compra (idCompra, código, data, valorInicial, valorFinal, DescontoCliente, DescontoLoja, fatura, idCliente->Cliente, idMeio->MeioDePagamento, registo de compra->Funcionario)
- Produto (idProduto, código, nome, tipoProduto, precoFornecedor, precoPublico, quantidade, idLocal->Localizacao);
- Jogo (idProduto, idProdutora->Produtora);
- Filme (idProduto, idProdutora->Produtora, idRealizador->Realizador)
- Livro (idProduto, ISBN, idEditora->Editora)
- ClienteLoja (idCliente, idLoja);
- LojaFuncionario (idLoja, idFunc);
- FuncionarioArea (idFunc, idArea);
- CompraProduto(idCompra, idProduto, quantidade);
- ProdutoFornecedor(idProduto, idForn);
- TipoJogo (idTipo, idProduto);
- MusicaArtista (idProduto, idArtista);
- FilmeCategoria (idProduto, idCategoria);
- LivroCategoria (idProduto, idCategoria);

Instruções de LDD-SQL necessárias à criação da Base de Dados

(em anexo: *LojaCreate.sql*)

Instruções de LMD-SQL necessárias ao preenchimento da Base de Dados

(em anexo: *LojaInsert.sql*)

Consultas LMD-SQL

(em anexo: *queries.sql*)

Lista crescente dos funcionários que atenderam mais clientes

```
SELECT *
FROM (
        SELECT f.nome, f.idFunc, COUNT(1) n
        FROM Funcionario as f, Compra as c
        WHERE f.idFunc = registo_de_compra
        GROUP by registo_de_compra
        ) as t
ORDER BY n ASC;
```

Lista crescente dos funcionarios que venderam mais produtos

```
SELECT *
FROM (
        SELECT f.nome, f.idFunc,
        SUM(quantidade) n
        FROM Funcionario as f, Compra as c,CompraProduto as cp
        WHERE f.idFunc = registo_de_compra
        AND cp.idCompra = c.idCompra
        GROUP BY f.idFunc
        ) as t
ORDER BY n DESC;
```

Lista crescente dos funcionarios que fizeram vendas com maior lucro

```
SELECT *
FROM (
        SELECT f.nome, f.idFunc,
        SUM(valorFinal) n
        FROM Funcionario as f, Compra as c
        WHERE f.idFunc = registo_de_compra
        GROUP BY registo_de_compra
        ) as t
ORDER BY n ASC;
```

Lista crescente dos funcionários que mais facturaram em 2013

```
SELECT *
FROM (
        SELECT f.nome, f.idFunc,
        SUM(valorFinal) n
        FROM Funcionario as f, Compra as c
        WHERE f.idFunc = registo_de_compra
        AND data LIKE "%2013%"
        GROUP BY registo_de_compra
        ) as t
ORDER BY n ASC;
```

Lista decrescente de lojas por lucro no ano de 2013

```
SELECT l.nome,l.idLoja,sum(valorFinal) n
FROM Loja as l,Compra as c, LojaFuncionario as lf, Funcionario as f
WHERE f.idFunc = registo_de_compra
AND l.idLoja = lf.idLoja
AND f.idFunc = lf.idFunc
GROUP BY l.nome ;
```

Lista decrescente dos meios de pagamento mais usados

```
SELECT *
FROM (
        SELECT pa.nome, c.idMeio,
        COUNT(1) n
        FROM MeioDePagamento as pa, Compra as c
        WHERE pa.idMeio = c.idMeio
        GROUP BY c.idMeio
        ) as t
ORDER BY n desc;
```

Lista decrescente dos meios de pagamento mais rentáveis

```
SELECT *
FROM (
        SELECT pa.nome, c.idMeio,
        SUM(valorFinal) n
        FROM MeioDePagamento as pa, Compra as c
```

```
        WHERE pa.idMeio = c.idMeio
        GROUP BY c.idMeio
        ) as t
ORDER BY n desc;
```

Lista os filmes e livros com a mesma categoria ordenados por nome

```
SELECT idCategoria,idProduto,nome
FROM LivroCat
WHERE idCategoria = (
        SELECT idCategoria
        FROM LivroCat
        INTERSECT
        SELECT idCategoria
        FROM FilmeCat
        )
UNION select idCategoria,idProduto,nome
FROM FilmeCat
WHERE idCategoria = (
        SELECT idCategoria
        FROM LivroCat
        INTERSECT
        SELECT idCategoria
        FROM FilmeCat
        )
ORDER BY nome;
```

Lista ordenada alfabeticamente dos filmes por realizador

```
SELECT r.nome ,p.nome, p.idProduto
FROM Produto as p, Realizador as R,Filme as f
WHERE p.idProduto = f.idProduto
AND f.idRealizador = r.idRealizador
ORDER BY r.nome;
```

Triggers

No ato de compra, ao inserir no CompraProduto a quantidade do produto a vender, atualiza o stock dos produtos na tabela Produto.

```
CREATE TRIGGER update_produto
        AFTER INSERT ON CompraProduto
        BEGIN
                UPDATE Produto set quantidade = quantidade - NEW.quantidade
                WHERE idProduto = NEW.idProduto;
        END;
```

Quando um cliente faz uma compra atualiza os pontos do cliente consoante o valor final (valorFinal) da compra (tabela Compra)

```
CREATE TRIGGER update_pontos
        AFTER insert on Compra
        BEGIN
                UPDATE Cliente set pontos = pontos + New.valorFinal*2
                WHERE idCliente = NEW.idCliente;
        END;
```