universidade de aveiro estga escola superior de tecnologia e gestão de águeda

Imobiliária “VendeLares”

Modelação e Implementação da Base de Dados

Rui Duarte nº29979

2017/11/28

UC de Sistemas de Bases de Dados

Licenciatura em Tecnologias da Informação

Edição 2017/2018

UA-ESTGA

ÍNDICE

INTRODUÇÃO

No âmbito da disciplina de Sistemas de Bases de Dados foi-me proposta a criação e modelação de uma base de dados à qual me foram colocados à disposição vários temas. O tema da minha preferência encontra-se abaixo indicado e descrito.

A empresa “VendeLares” necessita de uma ferramenta informática que lhe permita fazer uma gestão dos seus clientes e manter um histórico dos imóveis vendidos, bem como ter uma perceção dos agentes imobiliários com melhores resultados. Um dos objetivos desta empresa é fazer a gestão das pessoas que venderam ou pretendem vender imóveis e das pessoas que compraram casas recorrendo à “VendeLares”.

O sistema a implementar deverá permitir efetuar as seguintes macrooperações:

- permitir registar e visualizar toda a informação respeitante aos imóveis, inclusive dos que já foram vendidos;
- registo de informação sobre os agentes, incluindo visitas a imóveis bem como as vendas efetuadas e lucros obtidos;
- manter um histórico de compras e/ou vendas de cada cliente;
  Para além das operações acima indicadas, a par da modelação, a base de dados deve assegurar a integridade dos dados e o cruzamento de informação eficiente, de forma a não haver redundância e inconsistência dos dados.

1.1. DIAGRAMA DE CLASSES CONCEPTUAL

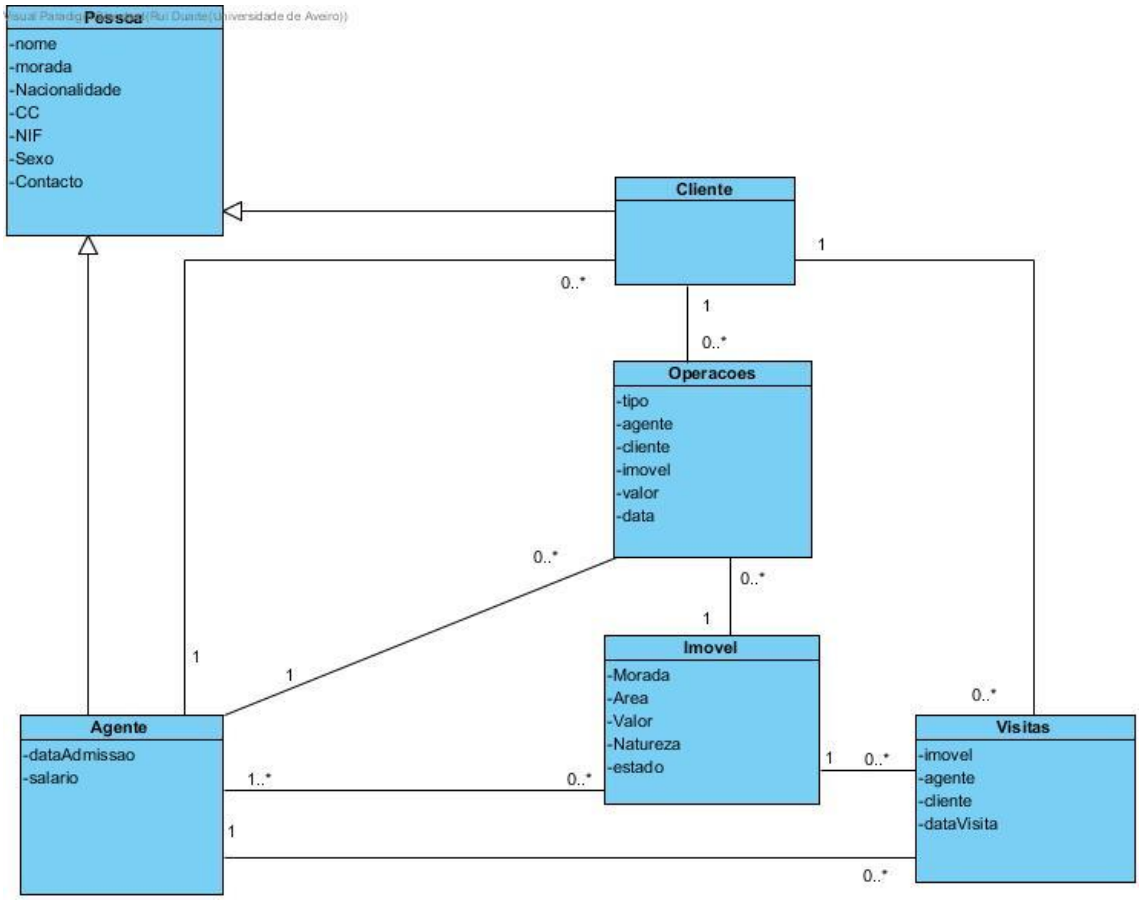

Figura 1 - Diagrama de classes

Associações e multiplicidade entre conceitos:

Agente -> Pessoa : agente é uma generalização da classe pessoa o que significa que herda todos os seus atributos.

Cliente -> Pessoa : cliente é uma generalização da classe pessoa o que significa que herda todos os seus atributos.

Agente -> Cliente {1 0..*} : Um agente pode ter vários clientes associados mas um cliente apenas pode estar associado a um único agente.

Agente -> Operacoes {1 0..*} : Um agente pode estar envolvido em várias operações de compra e venda mas uma operação só se efetua com um único agente

Agente -> Imovel {1 0..*} : Um agente pode ter vários imóveis a seu encargo mas um imóvel só pode ter um agente envolvido.

Agente -> Visitas {1 0..*} : Um agente pode efetuar várias visitas a imóveis mas numa visita só pode existir um agente.

Cliente -> Operacoes {1 0..*} : Um cliente pode comprar e vender vários imóveis mas cada operação é efetuada com um cliente.

Cliente -> Imovel {1 0..*} : Um cliente pode ter vários imóveis mas o mesmo imóvel não pode pertencer a várias pessoas ao mesmo tempo.

Cliente -> Visitas {1 0..*} : Um cliente pode fazer várias visitas mas cada visita só é feita com um cliente.

Imovel -> Operacoes {1 0..*} : Um imóvel pode ter várias operações mas uma operação só envolve um único imóvel.

Imovel -> Visitas {1 0..*} : Um imóvel pode sofrer de várias visitas mas cada visita só é efectuada a um imóvel.

1.2. DESCRIÇÃO DOS CONCEITOS

Conceito	Descrição	Sinónimos / Observações
Pessoa	Registar dados de todas as pessoas	
Cliente	Registar todos os dados do cliente	Generalização de pessoa
Vendedor	Registar todos os dados do vendedor	Generalização de pessoa
Imovel	Registar todos os dados do imóvel	
Operacoes	Registar todas as operações de compra e venda efetuadas	
Visitas	Registar todas as visitas efectuadas	

II – MODELAÇÃO LÓGICA DA BASE DE DADOS

2.1. DIAGRAMA LÓGICO DA BASE DE DADOS

Esquema relacional:

Cliente (idcliente (PK), nome, morada (FK), nacionalidade, cc, nif, sexo, contacto, mail)

Agente (idagente (PK), nome, morada (FK), nacionalidade, cc, nif, sexo, contacto, mail, dataadmissao, salario)

Imovel (idimovel (PK), morada (FK), area, valor, natureza, estado, numquartos, piscina, varanda, dispensa, garagem, aquecimento)

Operacoes (idoper (PK), tipo, agente (FK), cliente (FK), imovel (FK), valor, data)

Visitas (idvisita (PK), imovel (FK), agente (FK), cliente (FK), data)

Morada (idmorada (PK), rua, porta, andar, fracao, codpostal, freguesia, concelho, distrito)

2.2. NORMALIZAÇÃO

Primeira forma normal (1FN) – Requer que todos os atributos sejam atómicos. A normalização para 1FN elimina grupos repetitivos.

O atributo **morada** sofreu essa normalização, sendo criada uma tabela nova (**Morada**) com os atributos **idmorada, rua, porta, andar, fracao, codpostal, freguesia, concelho** e **distrito**, garantindo assim a consistência de dados entre as várias tabelas com o atributo morada (neste caso **Cliente**, **Agente** e **Imóvel**).

Segunda forma normal (2FN) – requer 1FN e refere que os atributos que não pertencem a uma chave candidata devem depender da mesma na totalidade e não parcialmente.

Terceira forma normal (3FN) – requer 2FN e requer que não haja nenhuma dependência funcional entre atributos não-chave.

No caso das formas 2FN e 3FN não foi necessário nenhum tipo de ajuste, devido à simplicidade da base de dados em causa e, principalmente pelo facto de ter sido criada a tabela **Operacoes**, que resolveu as possíveis dependências funcionais entre atributos não-chave que pudessem existir.

III - MODELAÇÃO FÍSICA DA BASE DE DADOS

3.1. SGBD ALVO

O Sistema de gestão de base de dados (SGBD) fornece a interface entre os dados que são armazenados na base de dados (BD) e os seus utilizadores. Além disso permite definir, gerir e aceder aos dados existentes na BD.

O SGBD escolhido para este projeto é o PostgreSQL que tem como principais características o facto de ser open-source, baseado no modelo orientado a objetos, a compatibilidade com múltiplas plataformas, a eficiência e robustez.

3.2. REPRESENTAÇÃO FÍSICA DA BASE DE DADOS

3.2.1. ESTRUTURA DA BASE DE DADOS

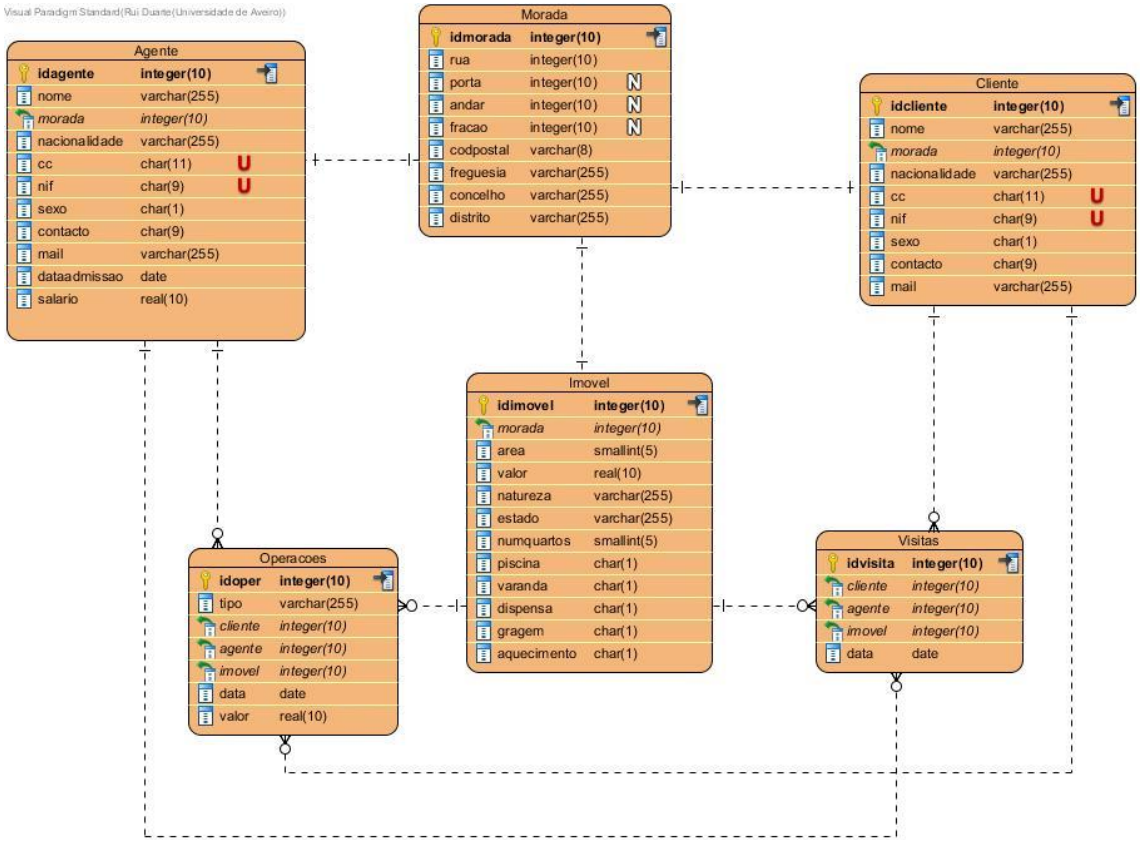

Figura 2 - Diagrama da base de dados

3.2.2. RESTRIÇÕES

Constraints CHECKS no Domínio (globais):

CREATE DOMAIN GENERO AS CHAR(1) NOT NULL CHECK (VALUE IN('M','F'));

Este constraint permite que qualquer tabela que use este tipo de dados use esta regra (só poder ser introduzida a letra M ou F).

CREATE DOMAIN SIMNAO AS CHAR(1) DEFAULT ('N') NOT NULL CHECK (VALUE IN('S','N'));
Decidi criar este constraint ao invés de usar o tipo de dados BOOLEAN para tornar o programa mais "português" e o campo em causa poder ser preenchido com S ou N ao invés do T ou F.

Contraints CHECK nas Tabelas (locais):

salario REAL NOT NULL CHECK(salario>0)

area INT NOT NULL CHECK(area>0),

valor REAL NOT NULL CHECK(valor>0),

Os constraints acima indicados obrigam à introdução de valores positivos (não existem salários, áreas ou imóveis com valores negativos)

natureza VARCHAR NOT NULL CHECK (natureza IN ('apartamento','moradia','garagem','loja')),

estado VARCHAR NOT NULL CHECK (estado IN ('compra','venda','comprado','vendido')),

tipo VARCHAR NOT NULL CHECK(tipo IN ('compra','venda')),

Os constraints acima indicados obrigam a que os atributos **natureza, estado** e **tipo** apenas aceitem os valores definidos entre os parênteses e que evita que sejam escritos de diferentes formas, o que assegura a consistência na pesquisa dos dados.

Constraints NOT NULL

Os constraints NOT NULL permitem tornar um campo de preenchimento obrigatório. Foi usado em todos os atributos com exceção dos atributos **andar, porta** e **fração** (existem imóveis sem este tipo de atributos).

Constraints UNIQUE

Os constraints UNIQUE permitem somente a introdução de valores únicos. Considerando que as chaves primárias e estrangeiras são sempre únicas apenas tive necessidade de aplicar este conceito aos campos referentes ao cartão de cidadão e número de contribuinte.

3.2.4. PROCEDIMENTOS DE EXECUÇÃO AUTOMÁTICA (*TRIGGERS*)

O trigger no ficheiro em anexo muda o estado do imóvel dependendo do tipo de operação introduzida. Se a operação for de compra o estado do imóvel em causa é alterado para "comprado". No caso de ser de venda, é alterado para "vendido".

3.3. MECANISMOS DE SEGURANÇA

3.3.1. REGRAS DE ACESSO

Criei 3 grupos de acesso com permissões distintas.

O administrador é um superuser com totais poderes, responsável pela administração da base de dados.

O gerente é um utilizador com acesso total à base de dados. É responsável pela introdução das operações de compra e venda na base de dados

O agente é um utilizador que só pode fazer operações de inserção e atualização nas tabelas "cliente", "imovel", "morada" e "visitas", no entanto não poderá fazer qualquer tipo de alteração as tabelas "agente" e "operacoes"

3.3.2. DEFINIÇÃO DE VISTAS (*VIEWS*)

Criei algumas views baseadas nos requisitos definidos anteriormente no capitulo introdutório.

-CREATE VIEW imovendidos

Esta vista permite mostrar os detalhes dos imóveis vendidos. Combina as tabelas "imovel" e "morada".

-CREATE VIEW agente

Esta vista reflete os dados estatísticos dos agentes, mostrando as suas vendas, lucros e número de visitas efetuadas a imóveis ordenadas de forma decrescente por lucro. Combina as tabelas "imóvel", "operações" e "visitas".

-CREATE VIEW clienteHist

Esta vista junta as tabelas "operações" e "cliente" de forma a mostrar o histórico de compras e vendas dos clientes com a respetiva data e valor associados, ordenados de forma ascendente por cliente, seguido do tipo de operação.

IV – CONSULTAS IMPLEMENTADAS (*QUERIES*)

Considerando que grande parte das queries necessitavam de “JOINS” optei por colocá-las em formato de views. No entanto, fiz duas consultas simples, uma para mostrar o resumo da faturação da empresa (compras e vendas em valor e unidades) e outra para mostrar o número de visitas que cada imóvel sofreu desde a sua introdução em sistema.

V – REFLEXÃO CRÍTICA

O planeamento foi de todo o ideal. Entusiasmei-me bastante com o trabalho, sem dúvida que o meu conhecimento, quer teórico, quer prático cresceu substancialmente graças a isso, mas sinto que passei demasiado tempo com os requisitos e com a fase inicial do projeto ao tentar desenhar uma solução completa/complexa demais. Acho que detalhei bastante as tarefas iniciais, mas considerando a altura em que me encontrava (semana de exames, outros trabalhos para terminar, projeto temático para continuar) acabei por não dar o devido detalhe às ultimas fases do projeto, nomeadamente a parte de triggers, segurança por falta de tempo. Até mesmo o teste da base de dados sinto que deveria ter sido feito de forma bastante mais exaustiva.

Para terminar, devo referir que esta base de dados ainda tem muito para ser trabalhada e melhorada. Gostava de ter implementado uma série de procedimentos que, ao pesquisar, acabei por aprender como os tipos de dados compostos e até mesmo as poderosas expressões REGEX para verificar a validade de emails ou números de telefone. Pretendo implementar estas alterações num futuro próximo.

ANEXOS

ANEXO A – CÓDIGO SQL PARA IMPLEMENTAÇÃO DA ESTRUTURA DA BASE DE DADOS

Ficheiro vendelares_estrutura.sql

ANEXO B – CÓDIGO SQL PARA IMPLEMENTAÇÃO DAS RESTRIÇÕES

Incluído no ficheiro vendelares_estrutura.sql

ANEXO C – CÓDIGO SQL PARA IMPLEMENTAÇÃO DOS ÍNDICES

Índices implementados automáticamente pelo tipo de dados SERIAL.

ANEXO D – CÓDIGO SQL PARA IMPLEMENTAÇÃO DOS TRIGGERS

Ficheiro vendelares_triggers.sql

ANEXO E – CÓDIGO SQL PARA IMPLEMENTAÇÃO DAS REGRAS DE SEGURANÇA E VISTAS

Ficheiro vendelares_seguranca.sql

ANEXO F – CÓDIGO SQL PARA IMPLEMENTAÇÃO CONSULTAS (QUERIES)

Incluído no ficheiro vendelares_views.sql