Projeto Final da Disciplina de Banco de Dados
Sexto Andar

Guilherme Silva Souza - 19/0014059
Maria Eduarda Holanda - 19/0043725

[1] Dep. Ciência da Computação – Universidade de Brasília (UnB)
Banco de Dados

1. Introdução

O gerenciamento de dados sempre esteve presente no cotidiano do ser humano, até hoje temos o hábito de anotar em uma lista produtos a serem comprados, essa anotação pode acontecer em papel ou no telefone celular, para depois consulta-los e assim fazer a compra desses produtos em algum estabelecimento. Antigamente empresas registravam informações em arquivos físicos, mas com o passar do tempo e aumento no volume de dados ficou inviável de se armazenar e organizar essas informações físicas, assim se tornou mais urgente fazer esse armazenamento e consulta a esses dados. Com o advento da tecnologia surgiu o banco de dados, ele é uma coleção organizada de informações - ou dados - estruturadas, normalmente armazenadas eletronicamente em um sistema de computador. Um banco de dados é geralmente controlado por um sistema de gerenciamento de banco de dados (SGBD). [1]

Há vários tipos de banco de dados, cada um com seu proposito. Entre eles podemos destacar o banco de dados relacional. Os bancos de dados relacionais se tornaram dominantes na década de 1980. Os itens em um banco de dados relacional são organizados como um conjunto de tabelas com colunas e linhas. [1] Como citado acima, o banco de dados relacional é gerenciado por um SGBD, uma funcionalidade básica de um SGBD é a implementação de funcionalidades do tipo CRUD (do inglês *Create*, *Read*, *Update* e *Delete*) o que possibilita os procedimentos essenciais de leitura, escrita, atualização e remoção em um banco de dados. Para realizar tais operações o modelo relacional suporta a linguagem de programação SQL.

Atualmente o mercado de imobiliária movimenta um grande volume de dados, desde cadastro de clientes e imóveis até gerenciamento de contratos. O projeto apresentado nesse relatório busca desenvolver um banco de dados que suporta o fluxo de uma imobiliária digital. Nosso trabalho se propõe a disponibilizar uma ferramenta de manipulação de dados, sejam ele cadastro de cliente, cadastro de imóveis, armazenamento de fotos, etc que possa corresponder as expectativas esperadas, que são um CRUD funcional, uma camada de persistência e utilização de *views* e *procedures*.

Dividimos esse relatório em 6 partes. Mostraremos o diagrama de entidade relacionamento e logo depois o modelo relacional, em seguida apresentamos 5 consultas em álgebra relacional, avaliação das formas normais em cinco tabelas e por último exibimos o diagrama da camada de mapeamento.

2. Diagrama de Entidade Relacionamento

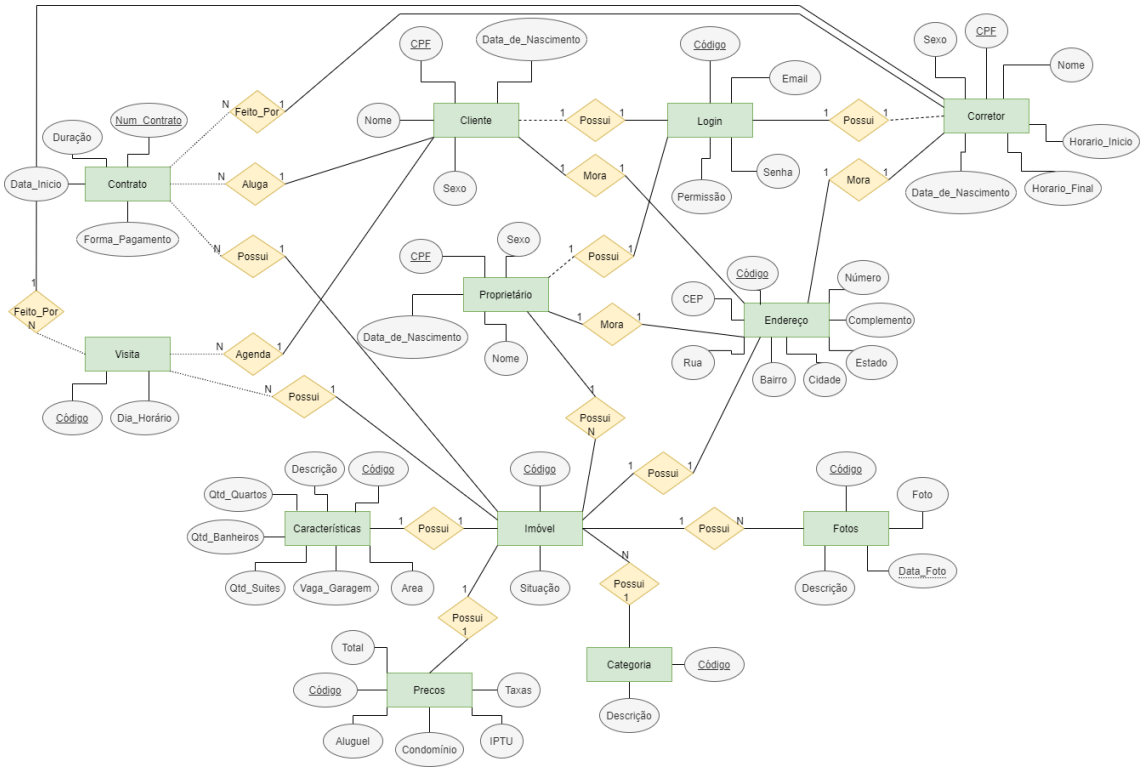
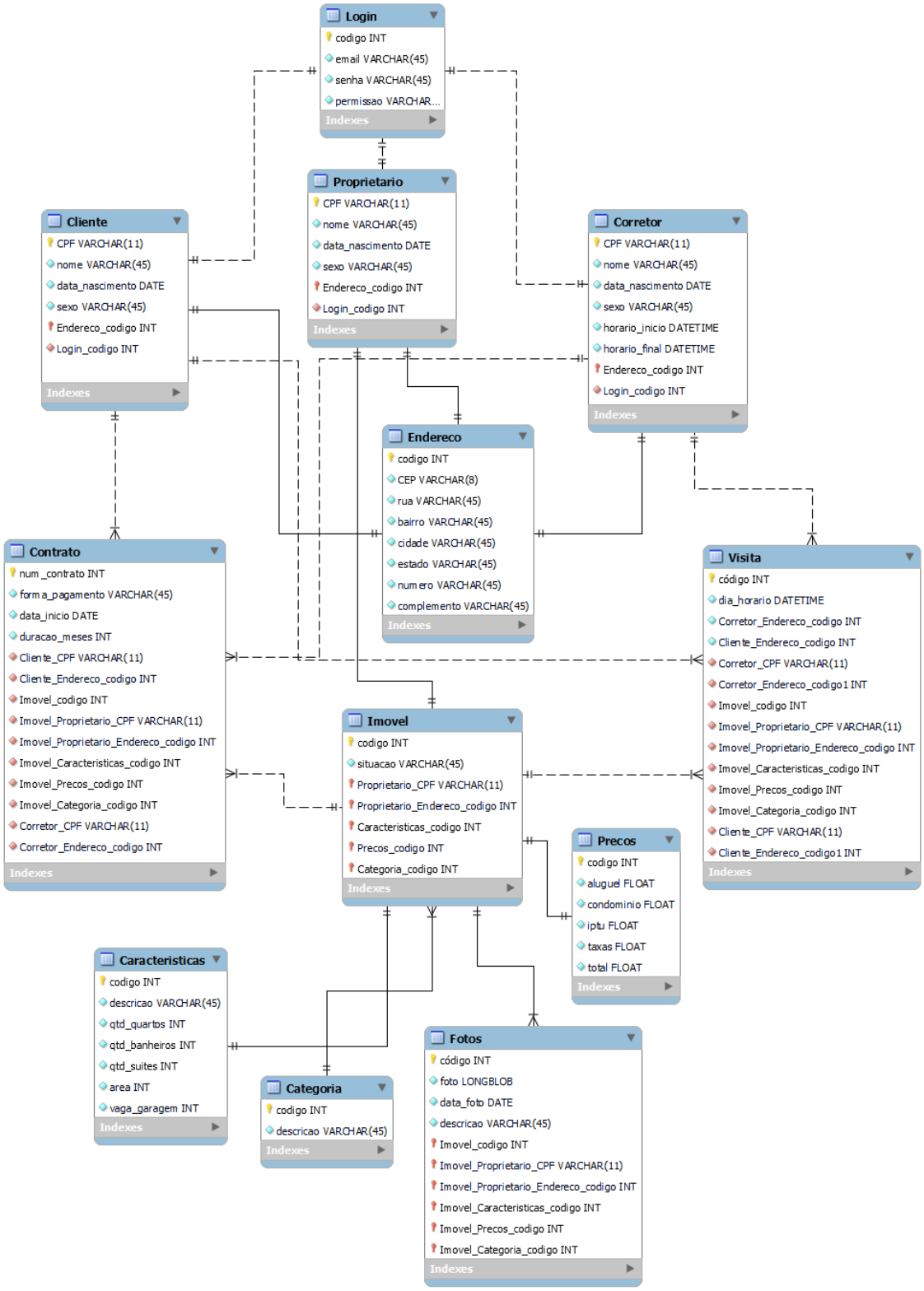


Figura 1. As linhas pontilhadas representam um relacionamento parcial, enquanto as linhas em negrito representam um relacionamento total.

3. Modelo Relacional



4. Álgebra relacional

$$\sigma_{(fk\_login = login.codigo)\ AND\ (fk\_endereco = endereco.codigo)}(Cliente\ x\ Login\ x\ Endereco)$$

1) A equação acima realiza uma operação de Produto Cartesiano entre as tabelas *Cliente, Login* e *Endereco*. Ela resulta na combinação das tuplas das três relações envolvidas. Isto é, resultando em todas as colunas da tabela *Cliente*, seguidas pelas colunas das tabelas *Login* e *Endereco*.

$$\pi_{(cpf,\ nome,\ CEP,\ email,\ permissao)}(\sigma_{(fk\_login = login.codigo)\ AND\ (fk\_endereco = endereco.codigo)}(Cliente\ x\ Login\ x\ Endereco))$$

2) A equação acima é muito parecida com a primeira, pois realiza uma operação de Produto Cartesiano com as mesmas tabelas. A diferença é que trata-se de uma projeção dos atributos `cpf`, `nome`, `CEP`, `email` e `permissao`, ou seja, a equação retorna apenas os atributos mencionados.

$$\pi_{(imovel.codigo,\ situacao,\ cidade,\ estado,\ cpf,\ nome)}(\sigma_{(fk\_proprietario = proprietario.cpf)\ AND\ (fk\_endereco = endereco.codigo\ )}\ (Imovel\ x\ Endereco\ x\ Proprietario))$$

3) A terceira equação também trata-se de uma projeção. Nesse caso, são retornados os atributos `imovel.codigo`, `situacao`, `cidade`, `estado`, `cpf` e `nome`. Essa projeção é realizada sobre o produto cartesiano das tabelas *Imovel*, *Endereco* e *Proprietario*.

$$\sigma_{(fk\_preco = precos.codigo)\ AND\ (fk\_proprietario = proprietario.cpf)\ AND\ (precos.total > 3000)}(Imovel\ x\ Proprietario\ x\ Precos)$$

4) A equação acima trata-se de uma operação de Produto Cartesiano entre as tabelas *Imovel, Proprietario* e *Precos*. Nesse caso, a operação retornará todas as colunas das tabelas citadas, no entanto, só retornará as fileiras que possuem o atributo $precos.total > 3000$.

$$\pi_{(cliente.cpf,\ cliente.nome,\ imovel.codigo,\ dia\_horario)}(\sigma_{(fk\_cliente = cliente.cpf)\ AND\ (fk\_imovel = imovel.codigo)}(Visita\ x\ Imovel\ x\ Cliente))$$

5) A última equação realiza também uma operação de projeção, agora sobre o produto cartesiano das tabelas *Visita, Imovel* e *Cliente*, retornando, assim, os atributos `cliente.cpf`, `cliente.nome`, `imovel.codigo` e `dia_horario`.

5. Avaliação das formas normais

Normalização de banco de dados é um conjunto de regras que visa, principalmente, a organização de um projeto de banco de dados para reduzir a redundância de dados, aumentar a integridade de dados e o desempenho. Para normalizar o banco de dados, deve-se examinar as colunas (atributos) de uma entidade e as relações entre entidades (tabelas), com o objetivo de se evitar anomalias observadas na inclusão, exclusão e alteração de registros. Faremos aqui a análise de 5 tabelas de acordo com as formas normais a seguir: [2]

1ª forma normal: Para a tabela estar na 1ª forma normal, todos os seus atributos devem ser atômicos.

2ª forma normal: Para estar na 2ª forma normal, a tabela deve estar, primeiramente, na 1ª forma normal. Além disso, todo atributo do complemento de uma chave candidata deve ser totalmente funcionalmente dependente daquela chave.

3ª forma normal: Para estar na 3ª forma normal, é preciso estar na 2ª forma normal. Ademais, os atributos não-chaves devem ser dependentes não-transitivos da chave primária, isto é, devem ser mutuamente independentes e dependentes unicamente e exclusivamente da chave primária.

5.1. Tabela Endereço

```
create table endereco (
    codigo int primary key not null auto_increment,
    CEP varchar(8) not null,
    rua varchar(50) not null,
    bairro varchar(50) not null,
    cidade varchar(50) not null,
    estado varchar(50) not null,
    numero int not null,
    complemento varchar(50) not null
);
```

1FN: Analisando a tabela endereço, podemos perceber que todos os seus atributos são atômicos, uma vez que nenhum pode ter mais de um valor.

2FN: No caso da tabela endereço, ela cumpre os requisitos mínimos, pois está na 1ª forma normal. Além disso, a chave primária `codigo` funcionalmente determina todos os outros atributos, uma vez que todos os atributos dependem da chave primária.

3FN: Como vimos, a tabela endereço está na 2ª forma normal. No entanto, a tabela não está normalizada na 3ª forma, pois o atributo `CEP` determina a `rua` e `bairro` e também o atributo `cidade` determina o `estado`. Uma forma de normalizar seria criar mais duas tabelas, uma com `CEP`, `rua` e `bairro`, e outra com `cidade` e `estado`, adicionando à tabela Endereço, somente as chaves primárias, como chaves estrangeiras.

```sql
create table precos(
    codigo int not null primary key auto_increment,
    aluguel float not null,
    condominio float not null,
    iptu float not null,
    taxas float not null,
    total float not null
);
```

5.2. Tabela Precos

1FN: Examinando a entidade `precos` podemos observar que ela cumpre os requisitos para estar na 1ª forma normal pois todos seus atributos são atômicos.

2FN: A tabela `precos` também esta na 2ª forma normal pois, primeiramente ela perfaz o requisito de estar na 1ª forma normal. Ademais, sua chave primária `codigo` consegue definir todos os seus atributos assim não é possível definir um preço partindo de qualquer outro atributo.

3FN: A entidade `precos` não está na 3ª forma normal pois, apesar de ela estar na 2ª forma normal, o atributo `total` depende dos atributos `aluguel`, `condominio`, `iptu` e `taxas`. Para passar a tabela para a 3ª forma normal poderíamos excluir o atributo `total` da tabela, que pode ser movido para outra tabela referenciando `preços`.

5.3. Tabela Cliente

```sql
create table cliente (
    cpf varchar(11) primary key not null,
    nome varchar(50) not null,
    data_de_nascimento date not null,
    sexo varchar(10) not null,
    fk_endereco int not null,
    foreign key (fk_endereco) references endereco(codigo)
    on delete cascade
    on update cascade,
    fk_login int not null,
    foreign key (fk_login) references login(codigo)
    on delete cascade
    on update cascade
);
```

1FN: Analisando a tabela `cliente` vemos que ela cumpre os requisitos para estar na 1ª forma normal pois todos seus atributos são atômicos.

2FN: A tabela `cliente` também esta na 2ª forma normal pois, primeiramente ela cumpre a condição de estar na 1ª forma normal. Além disso, seus atributos dependem exclusivamente do `CPF` pois só ele consegue precisar um cliente.

3FN: A entidade `cliente` está na 3ª forma normal pois, além de esta na 2ª forma normal, todos seus atributos são independentes uns dos outros e ao mesmo tempo

são dependentes exclusivamente da chave primária da tabela.

5.4. Tabela Imóvel

```
create table imovel(
    codigo int primary key not null auto_increment,
    fk_endereco int not null,
    foreign key (fk_endereco) references endereco(codigo)
    on delete cascade
    on update cascade,
    fk_proprietario varchar(11) not null,
    foreign key (fk_proprietario) references proprietario(cpf)
    on delete cascade
    on update cascade,
    fk_categoria int not null,
    foreign key (fk_categoria) references categoria(codigo)
    on delete cascade
    on update cascade,
    fk_preco int not null,
    foreign key (fk_preco) references precos(codigo)
    on delete cascade
    on update cascade,
    situacao varchar(20) not null,
    fk_caracteristicas int not null,
    foreign key (fk_caracteristicas) references caracteristicas(codigo)
    on delete cascade
    on update cascade
);
```

1FN: Ao analisar a tabela, podemos ver que imóvel possui uma *primary key*, que deve ser atômica, cinco *foreign keys*, que também são atômicas e somente um atributo `situacao`, que também só assume um valor. Portanto, está na 1ª forma normal.

2FN: Sabendo que a tabela está na 1ª forma, podemos analisar se ela está na 2ª forma. Como o único atributo do complemento da chave primária `situacao` é totalmente funcionalmente dependente da chave `codigo`, então a tabela também está na 2ª forma normal.

3FN: Como vimos, a tabela está na 2ª forma normal. Além disso, é possível perceber que todos os atributos não-chave são dependentes não-transitivos da chave primária, pois todos são mutuamente independentes e dependentes exclusivamente do imóvel. Um proprietário, por exemplo, possui um imóvel, mas não tem relação nenhuma com as características do imóvel, uma vez que as características se referem unicamente ao imóvel.

5.5. Tabela Contrato

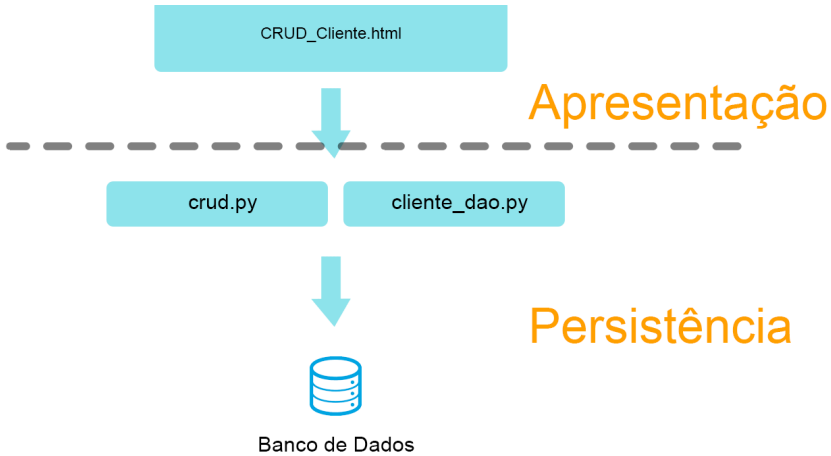
1FN: Ao olhar para tabela `contrato` percebemos que seus atributos são atômicos, assim cada um só pode ser expressado por um único valor. Diante disso, a tabela está na 1ª forma normal.

2FN: A entidade `contrato` está na 1ª forma normal, analisamos agora se ela está na segunda. Percebemos que os atributos não chave, `forma_pagamento`, `duracao_meses` e `data_inicio` depende unicamente do atributo chave `numero_contrato`. Caracterizando assim a entidade na 2ª forma normal.

```sql
create table contrato(
    numero_contrato int not null primary key auto_increment,
    fk_imovel int not null,
    foreign key (fk_imovel) references imovel(codigo)
    on delete cascade
    on update cascade,
    fk_corretor varchar(11) not null,
    foreign key (fk_corretor) references corretor(cpf)
    on delete cascade
    on update cascade,
    fk_cliente varchar(11) not null,
    foreign key (fk_cliente) references cliente(cpf)
    on delete cascade
    on update cascade,
    forma_de_pagamento varchar(20) not null,
    duracao_meses int not null,
    data_inicio date not null
);
```

3FN: Para estar na 3ª forma normal, é preciso estar na 2ª forma normal, como vimos a tabela *contrato* está. Verificamos que os atributos não chave da tabela devem ser mutuamente independentes e dependentes unicamente e exclusivamente da chave primária, isto é, os atributos não chave da tabela unicamente do atributo chave `numero_contrato`. Assim fazendo a tabela está na 3ª forma normal.

6. Camada de mapeamento para a tabela Cliente



Acima é mostrado o fluxo de informações do sistema proposto. Na camada de apresentação temos o html responsável pela interface com usuário, quando o usuário faz a requisição de alguma informação que está armazenado no banco de dados, como por exemplo dados da tabela cliente, essa requisição passará pela camada de persistência, primeiramente pelo arquivo crud.py que é responsável em pegar a requisição e validar, e depois pelo arquivo ciente_dao.py que é responsável pela conexão com o banco de dados.

7. Repositório Github

Para acessar todos os códigos utilizados no projeto, assim como o script que gerou o banco de dados basta clicar no link: Link do repositório no Github.

Referências

[1] Oracle. O que é um banco de dados. https://www.oracle.com/br/database/what-is-database, 2021. [Online; accessed 25-Octuber-2021].

[2] Wikipedia. Normalização de dados. https://pt.wikipedia.org/wiki/Normaliza%C3%A7%C3%A3o_de_dados, 2021. [Online; accessed 26-Octuber-2021].