UNIVERSIDADE FEDERAL DE SÃO CARLOS

Bacharelado em Ciência da Computação

BANCO DE DADOS

Profª. Dra. Sahudy Montenegro González

PROJETO PRÁTICO

Grupo 8 - Tema: Mercado

Felipe Ottoni Pereira

Letícia Almeida Paulino de Alencar Ferreira

Ricardo Yugo Suzuki

Sorocaba

3 de Abril de 2023

Fase Final (Correções da Fase Intermediária 2 + Relatório Final)

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

1. DESCRIÇÃO DO PROBLEMA

Um mercado que trabalha com a venda de diversos produtos possui um sistema para organização e gerenciamento de clientes, funcionários, vendas e produtos. Nesse sentido, os produtos são definidos por seu nome (arroz, sabão, suco, bala…), código de barras, preço (R$0,50 - R$2.000,00), marca e peso (gramas, quilogramas, litros e mililitros). O mercado vende apenas para clientes cadastrados, com seu nome e CPF, ademais, cada cliente pode gerar vendas, nas quais produtos podem ser comprados em diferentes quantidades.

As vendas são identificadas pelo código de sua nota fiscal, possuem data, composta por dia, mês e ano de sua realização, e seu preço total, calculado a partir do preço dos produtos contidos nela e de suas respectivas quantidades. Além disso, em uma venda há a compra de produtos e esta é gerada por um cliente.

Existem diferentes tipos de funcionários no mercado, todos devem possuir CPF, nome, salário (R$600,00 a R$4.000,00), cargo (limpeza, repositor, caixa, empacotador), data de nascimento e idade (mínimo: 18 anos), calculada por meio da data de nascimento. Já os funcionários que são caixas também precisam ter seu grau de escolaridade (Ensino Fundamental, Ensino Médio, Ensino Superior) registrado.

Funcionários do tipo caixa possuem uma relação hierárquica em que há um supervisor que gerencia seus subordinados. Ademais, toda venda é realizada por um caixa e um caixa pode realizar várias vendas. Desse modo, o sistema do mercado busca estruturar e administrar seu funcionamento.

Consultas

C1. Ricardo | Qual o montante do preço total das vendas por mês?

C2. Letícia | Quantas vendas são maiores ou iguais a um determinado preço em um determinado mês de um ano?

C3. Felipe | Quais são os produtos mais vendidos, em ordem decrescente em determinado ano?

C4. Ricardo | Quais produtos não foram vendidos no ano Y para o cliente Z?

C5. Felipe | Listar os funcionários que tenham o maior salário de seus cargos

C6. Letícia | Qual ou quais caixas fizeram mais atendimentos?

2. PROJETO CONCEITUAL

A Figura 1 apresenta modelo de entidade-relacionamento, composta pelo diagrama, de acordo com o problema descrito no texto, e suas anotações semânticas, com os trechos do mini-mundo que não puderam ser representados com elementos de modelagem. Além disso, o diagrama segue notações vistas em sala de aula e a ferramenta para sua construção foi o Lucidchart.

Anotações Semânticas

A. O atributo **idade** é calculado por meio da **data de nascimento**.
B. O atributo **preço total** é calculado a partir do **preço dos produto**s e a **quantidade** do produto comprado.
C. Um supervisor não pode ser gerenciado por um subordinado.

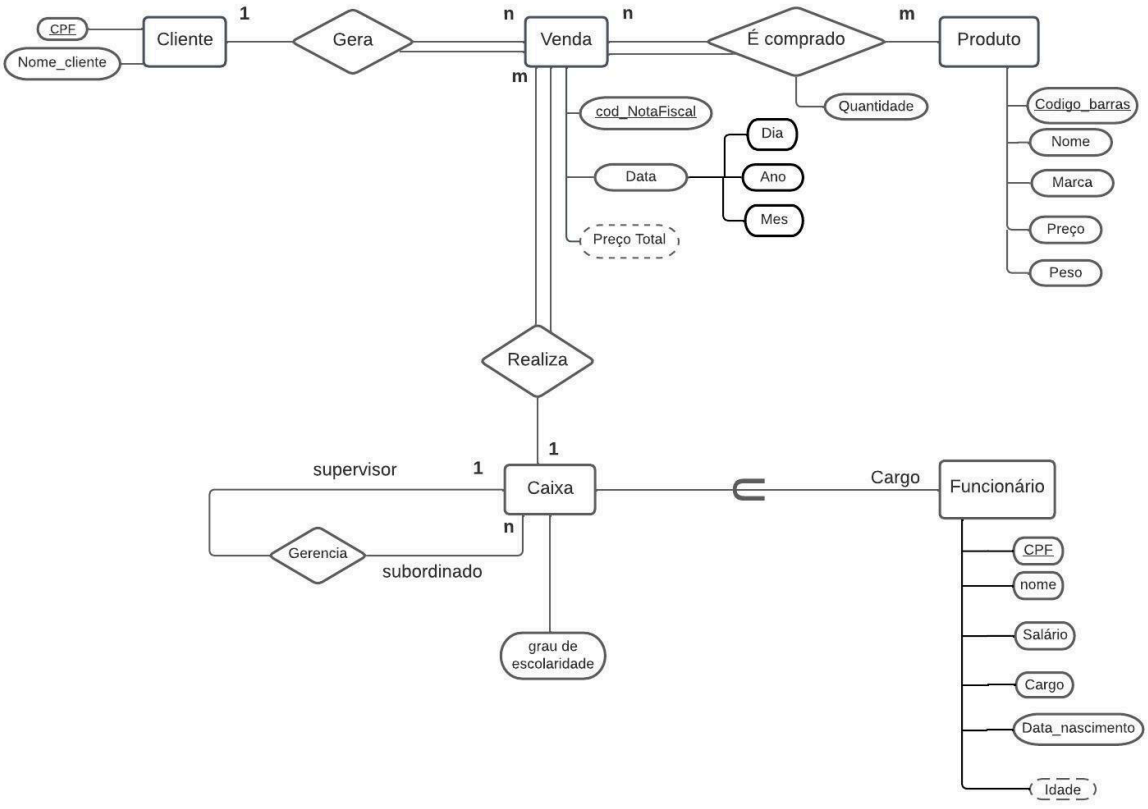

Figura 1: Modelo de Entidade-Relacionamento

A Tabela 1 representa uma tabela de metadados formada por meio das descrições dos tipos de atributo por tipos-entidade e tipos-relacionamento, com suas respectivas restrições. Desse modo, cada linha corresponde a um tipo-entidade ou tipo-relacionamento e cada coluna mostra seus tipos-atributos, o tipo de cada atributo e sua restrição, respectivamente.

Tipo-Entidade	Atributos	Tipo	Restrição
Produto	**Código de barra** **Nome** **Marca** **Preço** **Peso**	Identificador Monovalorado Monovalorado Monovalorado Monovalorado	Obrigatório Obrigatório Obrigatório Obrigatório, entre 0,5 e 2.000 Obrigatório
Cliente	**CPF** **Nome cliente**	Identificador Monovalorado	Obrigatório Obrigatório
Funcionário	**CPF** **Nome** **Salário** **Cargo** **Data de nascimento** **Idade**	Identificador Monovalorado Monovalorado Monovalorado Monovalorado Calculado	Obrigatório Obrigatório Obrigatório, entre 600 e 4.000 Obrigatório, Caixa, Limpeza, Repositor, Empacotador, Padeiro, Acougueiro Obrigatório Obrigatório, >= 18
Caixa	**Grau de escolaridade**	Monovalorado	Obrigatório, Ensino Fundamental, Ensino Médio ou Ensino Superior
Venda	**Código nota fiscal** **Data (dia, mês, ano)** **Preço total**	Identificador Composto Calculado	Obrigatório Obrigatório Obrigatório, >= 0,5
É comprado	**Quantidade**	Monovalorado	Obrigatório, > 0

Tabela 1: Tipos de atributos por tipo-entidade e tipo-relacionamento do DER.

3. PROJETO LÓGICO

A transformação entre modelos ocorreu manualmente por cada membro do grupo e depois houve uma comparação e discussão sobre os resultados. Nesse sentido, primeiro foram mapeados todos os tipo-entidades e seus atributos, o atributo composto `Data` está representado por `dia, mês, ano` em `Venda`.

Tipo-Relacionamento `Cliente - Venda`

O tipo-relacionamento `Gera` entre `Cliente` e `Venda` está representado pela adição da coluna `CPF_cliente, que referencia cliente,` especificamente na tabela `Venda` para evitar atributos multivalorados, pois é uma relação 1:N. Não escolhemos representar por meio de tabela própria para evitar quebrar os princípios de repetição de chave e mais junções, além disso, há a vantagem que não deve haver campo nulo pois é uma relação de participação total, assim, em toda venda há um cliente ligado à ela.

Tipo-Relacionamento `Caixa - Venda`

O tipo-relacionamento `Realiza` entre `Caixa` e `Venda` também foi representado por adição de coluna `CPF_caixa, que referencia caixa,` especificamente na tabela `Venda` para evitar atributos multivalorados, pois é uma relação 1:N. Não escolhemos tabela própria a fim de evitar quebrar os princípios de repetição de chave e mais junções, além disso, há a vantagem que não deve haver campo nulo pois é uma relação de participação total, assim, em toda venda há um caixa ligado à ela.

Tipo-Relacionamento `Produto - Venda`

O tipo-relacionamento `EhComprado` entre `Produto` e `Venda` precisa ser representado por uma tabela própria por se tratar de um relacionamento N:N. Sua chave primária é composta pelas chaves primárias dos tipo-entidades que fazem parte do tipo-relacionamento.

`Funcionario - Caixa`

Para representar a herança entre `Funcionario` e `Caixa` foi utilizado uma tabela para cada tipo-entidade, pois há a vantagem de não haver campos nulos, mas há repetição de chave e criamos junção. Não optamos pelas outras alternativas como: uma só tabela para toda hierarquia, pois assim teríamos campos nulos e precisaríamos verificar via programação que estamos utilizando um caixa no tipo-Relacionamento `Realiza`; uma tabela para cada tipo-entidade especializada, porque há apenas uma especialização e há especializações parciais como funcionários de limpeza que não possuem atributos próprios; ou uma tabela com atributos discriminadores, pois teríamos campos nulos e `Caixa` se relaciona com outros tipo-entidades, então teríamos que verificar via programação.

Tipo-Relacionamento Unário em `Caixa`

O auto-relacionamento que ocorre em `Caixa`, uma relação 1:N, foi representado por meio de adição de coluna. Embora essa representação admita campos nulos no caso dos caixas que são gerentes, já que estes não devem ser gerenciados, evitamos repetição de chaves e junções.

Mapeamento do esquema conceitual para o esquema relacional

```
cliente(CPF, nome)

venda(cod_nota_fiscal, dia, mês, ano, CPF_cliente, CPF_caixa)
     CPF_cliente referencia cliente.
     CPF_caixa referencia caixa.
produto(codigo_barras, nome, marca, preco, peso)

funcionario(CPF, nome, salario, cargo, data_nascimento)

caixa(CPF, grau_escolaridade, CPF_supervisor)
     CPF referencia funcionario.
     CPF_supervisor referencia caixa.

eh_comprado(cod_nota_fiscal, codigo_barras, quantidade)
     cod_NotaFiscal referencia Venda.
     cod_Barras referencia Produto.
```

O modelo obtido atende a 3° forma normal, porque está na 1° forma normal, pois não têm atributos multivalorados, também está na 2° Forma Normal, pois não há dependência funcional parcial, e não tem dependência funcional transitiva. Além disso, as Dependências Funcionais do modelo relacional são:

`Cliente`: `CPF → Nome`
(`nome` é dependente funcional de `CPF`)

`Venda`:`cod_nota_fiscal → dia, mês, ano, CPF_cliente, CPF_caixa`
(`dia, mês, ano, CPF_cliente` e `CPF_caixa` são dependentes funcionais de `cod_nota_fiscal`)

Produto: codigo_barras → nome, marca, preco, peso
(nome, marca, preco e peso são dependentes funcionais de codigo_barras)
Funcionario: CPF → nome, salario, cargo, data_nascimento
(nome, salario, cargo e data_nascimento são dependentes funcionais de CPF)

Caixa: CPF → grau_escolaridade, CPF_supervisor
(grau_escolaridade e CPF_supervisor são dependentes funcionais de CPF)

EhComprado: cod_nota_fiscal, codigo_barras → quantidade
(a quantidade comprada é dependente funcional de cod_nota_fiscal e codigo_barras)

4. PROJETO FÍSICO DE BANCO DE DADOS

A criação do projeto físico do banco de dados, de acordo com o esquema lógico, está no arquivo "*ddl.sql*". Nesse sentido, no *script* documentado estão os comandos SQL para criação do *database* mercado e para as criações das tabelas cliente, produto, funcionário, caixa, venda e eh_comprado, respectivamente. Dessa maneira, as tabelas cliente, produto e funcionário, foram criadas primeiro, pois não possuem chaves estrangeiras, assim, caixa foi criada, pois referencia funcionário, e venda criada em seguida, já que referencia caixa e cliente. Então, a tabela eh_comprado foi criada por último porque referencia produto e venda.

Ademais, por meio das restrições presentes na Tabela 1, cláusulas CHECK foram definidas para esses atributos e para os atributos que não fazem parte da chave primária das tabelas acrescentamos a cláusula NOT NULL. Por fim, nas tabelas caixa, venda e eh_comprado, por causa das chaves estrangeiras presentes nas tabelas, adicionamos as cláusulas de chave estrangeira com CASCADE para UPDATE e DELETE.

A alimentação do banco de dados mercado está no arquivo "*dados.sql*", onde estão os *scripts* de inserção de tuplas das tabelas criadas. Dessa maneira, as inserções ocorreram na mesma ordem que as criações de suas respectivas tabelas devido as dependências existentes e ao final de cada conjunto de inserção de uma tabela há exemplos comentados de cada caso de

exceção seja por *trigger*, tamanho do tipo de dado, cláusula `CHECK` , tipo de dado ou cláusula `FOREIGN KEY`. Portanto, para os dados inseridos procurou-se simular dados reais, para isso utilizou-se geradores online para o `CPF` e `codigo_barras`, já o `cod_nota_fiscal` são sequenciais por data, logo o menor número identifica a primeira compra e o maior a última. Além disso, ao final do documento há comandos de consulta SQL para exibir as tabelas completas.

5. ESPECIFICAÇÃO DE CONSULTAS EM ÁLGEBRA RELACIONAL E NA SQL

As consultas especificadas podem ser feitas por Álgebra Relacional e por SQL, nesse sentido, cada consulta foi descrita e realizada por ambos métodos. Para a escrita da AR utilizou-se as notações ensinadas em sala de aula, por meio das ferramentas de equação do *Google Docs*. Já para a escrita da SQL, nas consultas com valores variáveis esses estão especificados com "**<>**".

Consulta 1 - Qual o montante do preço total das vendas por mês?

Felipe

Essa consulta tem como objetivo informar ao usuário o montante das vendas de cada mês em seus respectivos anos. Seria de extrema importante para o mercado conseguir visualizar o lucro que tem em determinado mês, além disso é bom para efeitos comparativos, podendo comparar em quais meses eles são melhores ou piores que em outros meses por exemplo.

AR

$$t1 \leftarrow Produto \bowtie eh\_comprado$$

$$t2 \leftarrow {}_{cod\_nota\_fiscal}F_{sum(\pi_{(preco * quantidade)}\,(t1))}\,(t1)$$

$$t3 \leftarrow \rho_{(cod\_nota\_fiscal,\ preco\_total)}\,(t2)$$

$$t4 \leftarrow \pi_{(ano,mes,cod\_nota\_fiscal)}\,(Venda)$$

$$t5 \leftarrow t3 \bowtie t4$$

$$t6_{(ano,mes,montante)} \leftarrow {}_{ano,mes}F_{sum\,(preco\_total)}\,(t5)$$

$$\pi_{(ano,\ mes,\ montante)}\,(t6)$$

SQL

```
SELECT mes, ano, SUM(preco_total) as lucro
FROM (SELECT cod_nota_fiscal , SUM(preco_itens) AS preco_total, mes, ano
      FROM (SELECT * , pr.preco * comp.quantidade AS preco_itens
            FROM eh_comprado comp NATURAL JOIN produto pr NATURAL JOIN
      venda v)t1
      GROUP BY cod_nota_fiscal, ano, mes
      ORDER BY cod_nota_fiscal, ano, mes)t2
GROUP BY ano, mes
ORDER BY ano, mes;
```

Consulta 2 - Quantas vendas são maiores ou iguais a um determinado preço em um determinado mês de um ano?

Letícia

Essa consulta procura mostrar o número de vendas que foram maiores ou de mesmo valor que um valor estimado, em um mês e ano específico. Seria importante verificar com rapidez quantas compras bateram a meta de gasto que o mercado espera dos clientes por compra.

AR

$t1 \leftarrow Produto \bowtie eh\_comprado$

$t2 \leftarrow {}_{cod\_nota\_fiscal}F_{sum(\pi_{(preco * quantidade)}(t1))}(t1)$

$t3 \leftarrow \rho_{(cod\_nota\_fiscal,\ preco\_total)}(t2)$

$t4 \leftarrow \pi_{(ano,mes,cod\_nota\_fiscal)}(Venda)$

$t5 \leftarrow t3 \bowtie t4$

$t6 \leftarrow \sigma_{mes = <mes> \wedge ano = <ano> \wedge preco\_total >= <valor>}(t5)$

$\rho_{(mes,\ ano,\ quantidade)}({}_{mes,\ ano}F_{count\ (*)}(t6))$

SQL

```
SELECT mes, ano, COUNT(*)
FROM (SELECT v.mes, v.ano, preco_total
      FROM (SELECT cod_nota_fiscal , SUM(preco_itens) AS preco_total
```

```
        FROM (SELECT * , pr.preco * comp.quantidade AS preco_itens
              FROM eh_comprado comp NATURAL JOIN produto pr) t1
              GROUP BY cod_nota_fiscal) t2 NATURAL JOIN venda v
WHERE v.mes = <mes> AND v.ano = <ano> AND preco_total >= <valor>) t3
GROUP BY mes, ano;
```

Consulta 3 - Quais são os produtos mais vendidos, em ordem decrescente em determinado ano?

Felipe

Liste o nome e a marca dos produtos mais vendidos no mercado em ordem decrescente em determinado ano, além de sua respectiva quantidade vendida. Seria de importância para o mercado descobrir em quais produtos ele deve investir mais em estoque, por exemplo.

AR

$$t1 \leftarrow {}_{codigo\_barras}F_{SUM(quantidade)}(\sigma_{ano = <ano>}(eh\_comprado))$$

$$t2 \leftarrow \rho_{(codigo\_barras,\, quantidade)}(t1)$$

$$\pi_{nome,\, marca,\, quantidade}(t2 \bowtie produto)$$

SQL

```
SELECT nome, marca ,SUM(quantidade) AS quantidade_total_vendida
FROM produto NATURAL JOIN eh_comprado comp NATURAL JOIN venda v
WHERE ano = <ano>
GROUP BY codigo_barras
ORDER BY quantidade_total_vendida DESC;
```

Consulta 4 - Quais produtos não foram vendidos no ano Y para o cliente Z?

Ricardo

Essa consulta tem o objetivo de mostrar ao usuário produtos que não foram vendidos para um determinado cliente, assim, aqueles que não foram comprados no ano determinado. Seria importante para descobrir as preferências de seus clientes.

AR

$$t1 \leftarrow \sigma_{CPF\_cliente = <CPF> \ ^\wedge \ ano = <ano>} \ (produto \bowtie eh\_comprado \bowtie venda \ )$$

$$t2 \leftarrow \pi_{nome, marca, cod\_barras} \ (t1)$$

$$t3 \leftarrow \pi_{nome, marca, cod\_barras} (produto) \ - \ t2$$

$$\pi_{nome, marca} (t3)$$

SQL

```
SELECT nome, marca FROM produto
WHERE codigo_barras NOT IN
      (SELECT codigo_barras
       FROM eh_comprado NATURAL JOIN venda
       WHERE cpf_cliente = <CPF> AND ano = <ano>
       );
```

Consulta 5 - Listar os funcionários que tenham o maior salário de seus cargos

Ricardo

Essa consulta procura exibir os maiores salários de cada cargo e os seus respectivos funcionários. É de extrema importância administrativa, gerencial e de controle para mercado, assim ele pode verificar o teto salarial de cada cargo.

AR

$$t1_{(cargoA,\ max\_salario)} \leftarrow {}_{cargo}F_{MAX(salario)}(funcionario)$$

$$t2 \leftarrow t1 \bowtie_{cargo = cargoA\ \hat{}\ salario = max\_salario} funcionario$$

$$\pi_{cpf,\ nome,\ salario,\ cargo}(t2)$$

SQL

```
SELECT f.CPF, f.nome, f.salario, f.cargo
FROM funcionario f INNER JOIN (
               SELECT cargo, MAX(salario) AS max_salario
               FROM funcionario
               GROUP BY cargo ) t ON f.cargo = t.cargo AND
          f.salario = t.max_salario;
```

Consulta 6 - Qual ou quais caixas fizeram mais atendimentos?

Letícia

Essa consulta mostra o(s) caixa(s) que fizeram/atenderam mais vendas, acaba sendo importante para que o mercado faça um comparativo de venda por caixa, e , principalmente, para que possa desenvolver a prática de ambiente de trabalho saudável e valorização de seus empregados, parabenizando os envolvidos, por exemplo.

AR

$$t1 \leftarrow caixa \bowtie venda$$

$$t2_{(CPF\_caixa,\ quantidade\_vendas)} \leftarrow {}_{CPF\_caixa}F_{count(*)}(t1)$$

$$t3 \leftarrow t2 \bowtie_{CPF = CPF\_caixa} funcionario$$

$$t4_{(quantidade\_vendas\_max)} \leftarrow F_{max(quantidade\_vendas)}(t2)$$

$$t5 \leftarrow \sigma_{quantidade\_vendas = quantidade\_vendas\_max}(t3\ X\ t4)$$

$$\pi_{cpf\_caixa, nome, quantidade\_vendas}(t5)$$

SQL

```
SELECT cpf_caixa, f.nome, t.quantidade_vendas
FROM (SELECT cpf_caixa, COUNT(*) AS quantidade_vendas
      FROM caixa ca, venda v
      WHERE cpf_caixa = cpf
      GROUP BY cpf_caixa) t, funcionario f
WHERE cpf_caixa = cpf AND quantidade_vendas IN (
                    SELECT MAX(quantidade_vendas)
FROM (SELECT cpf_caixa, COUNT(*) AS quantidade_vendas
      FROM caixa ca, venda v
      WHERE cpf_caixa = cpf
      GROUP BY cpf_caixa) t1);
```

6. *TRIGGER*

Os *triggers* propostos e implementados no projeto estão no arquivo "*trigger.sql*", onde estão os *scripts* em SQL para esse recurso. Eles foram adicionados nos cenários descritos a seguir para garantir a consistência e integridade dos dados no banco de dados.

`valida_CPF_cliente e valida_CPF_funcionario`

O objetivo desse *trigger* é fazer com que a função `valida_CPF()` seja executada quando um novo registro for inserido nas tabelas `cliente` ou `funcionario`, uma vez que `CPF` foi definido como *`char(11)`*. Então, é responsável por garantir que o valor do `CPF` inserido atenda à regra de formatação que o `CPF` é composto apenas por número. Assim, se o `CPF` não estiver no formato correto, por exemplo `'123abc78952'`, a função impedirá a inserção do registro. Porém, caso passe na validação, com valores como `'12345678933'` ou `'02345678912'`, a função atualizará o valor do `CPF` na tabela . Além disso, não tem esse gatilho para a tabela `caixa` pois essa tem como chave estrangeira o `CPF` do `funcionário`, logo, a verificação já é realizada.

valida_cod_barras_trigger

O objetivo desse *trigger* é fazer com que a função `valida_cod_barras()` seja executada toda vez que um novo registro for inserido na tabela `produto`, já que o atributo `codigo_barras` foi definido como `char(13)`. Dessa forma, a função verifica se o atributo `codigo_barras` é composto apenas por números. Caso algum caractere não seja um número, a função gera uma exceção com a mensagem "O codigo de barras deve apenas conter números". Já se o atributo contiver apenas números o valor dessa variável será atribuída ao `codigo_barras` do novo registro.

valida_nota_fiscal_trigger

O objetivo desse *trigger* é fazer com que a função `valida_nota_fiscal()` seja executada quando um novo registro for inserido na tabela `venda`, porque o atributo `cod_nota_fiscal` foi definido como `char(9)`. Dessa maneira, a função analisa se o atributo `cod_nota_fiscal` é formado apenas por números. Caso encontre algum caractere que não seja um número, a função gera uma exceção, já se o atributo possuir apenas números o valor dessa variável é atribuído a `codigo_barras` do novo registro.

verifica_gerenciamento_trigger

Esse *trigger* é responsável por garantir a integridade referencial da chave estrangeira `CPF_supervisor` na tabela `caixa`. Dessa forma, ele garante que um `caixa` não possa ser seu próprio supervisor, assim, evitando inconsistências nos dados. A função `verifica_gerenciamento()` é executada pelo *trigger* antes da inserção de um registro em `caixa`, nesse sentido, analisa se o `CPF` do supervisor é igual ao `CPF` do novo `caixa`. Desse modo, se for igual a função gera uma exceção com a mensagem "`O CPF do supervisor não pode ser o mesmo que o CPF do caixa`", todavia, se for diferente, ela retorna o novo registro através do comando `RETURN NEW`.

idade_funcionario_trigger

Esse gatilho e a função `valida_idade_funcionario()` garantem que o sistema não permitirá a inserção de registros de funcionários com idade inferior a 18 anos na tabela `funcionario`. Nesse sentido, a função verifica se a idade do novo funcionário é

menor do que 18 anos usando a função `AGE()`, que calcula a idade com base na data de nascimento inserida do novo funcionário. Se a idade for menor do que 18 anos, a função gera uma exceção com a mensagem "`O funcionário deve ter pelo menos 18 anos de idade`", caso o funcionário tenha dezoito ou mais anos, a função retorna o novo registro.

`venda_verifica_data_trigger`

Esse *trigger* e a função `verifica_venda()` tem o objetivo de não permitir a inserção de registros de novas vendas com datas inválidas na tabela `venda`. Isso porque ao inserir uma nova vendo o registro da data é formado pelos atributos `dia`, `mes` e `ano`. Por isso, a função `verifica_data()` analisa se o valor de cada atributo em conjunto formam uma data válida. Se a data não for válida, o gatilho levanta uma exceção e o registro não é inserido na tabela, caso contrário a função retorna o novo registro.

7. CONSIDERAÇÕES FINAIS

O objetivo principal do projeto é desenvolver um protótipo dos componentes que integram o tema escolhido pelo grupo, assim, procurou-se abordar os principais elementos de um mercado. Nesse sentido, os segmentos das etapas intermediárias foram feitos e corrigidos ao longo de todo o desenvolvimento.

Para a etapa final identificamos nossas principais dificuldades na parte de construção das consultas em Álgebra Relacional e em SQL, como também a produção dos *triggers*. Dessa maneira, consultamos a documentação SQL para parte de codificação, entretanto dúvidas de como resolver certas funcionalidades surgiram e aquelas que não conseguimos solucionar não foram implementadas ao projeto. Portanto, seguem as principais limitações do sistema:

- Cliente pode se atender
- Um cliente e um caixa terem o mesmo CPF mas nomes diferentes (temos uma ideia de como fazer a trigger, mas não conseguimos implementar a tempo, mas temos triggers suficientes)
- Garantir a cardinalidade 1 do tipo relacionamento unário de gerência (garantir que apenas uma pessoa seja gerente e que apenas o seu cpf_supervisor seja NULL)

- Ao tentar colocar um novo caixa que tenha outro cargo diferente de caixa, isso funcionaria?

8. FONTES E REFERÊNCIAS

Gerador de Senha - 4Devs. Disponível em: <https://www.4devs.com.br/gerador_de_senha>. Acesso em: 1 abr. 2023.

Chapter 4. SQL Syntax. Disponível em: <https://www.postgresql.org/docs/15/sql-syntax.html>. Acesso em: 1 abr. 2023.

Chapter 7. Queries. Disponível em: <https://www.postgresql.org/docs/15/queries.html>. Acesso em: 2 abr. 2023.

9.9. Date/Time Functions and Operators. Disponível em: <https://www.postgresql.org/docs/15/functions-datetime.html>. Acesso em: 2 abr. 2023.