Uma imagem contendo Interface gráfica do usuário

Descrição gerada automaticamente **GRUPO 5**

**Projeto Rota da Cerveja**

Prof.º Maria Angélica Calixto de Andrade Cardieri

Curso: Análise e Desenvolvimento de Sistemas

Disciplina: Laboratório de Banco de Dados

Ana Carolina Costa Marcondes RA: 0030482013003

Cintia Felix Mendonça RA: 0030482121025

Ryan Barbosa RA: 0030482011038

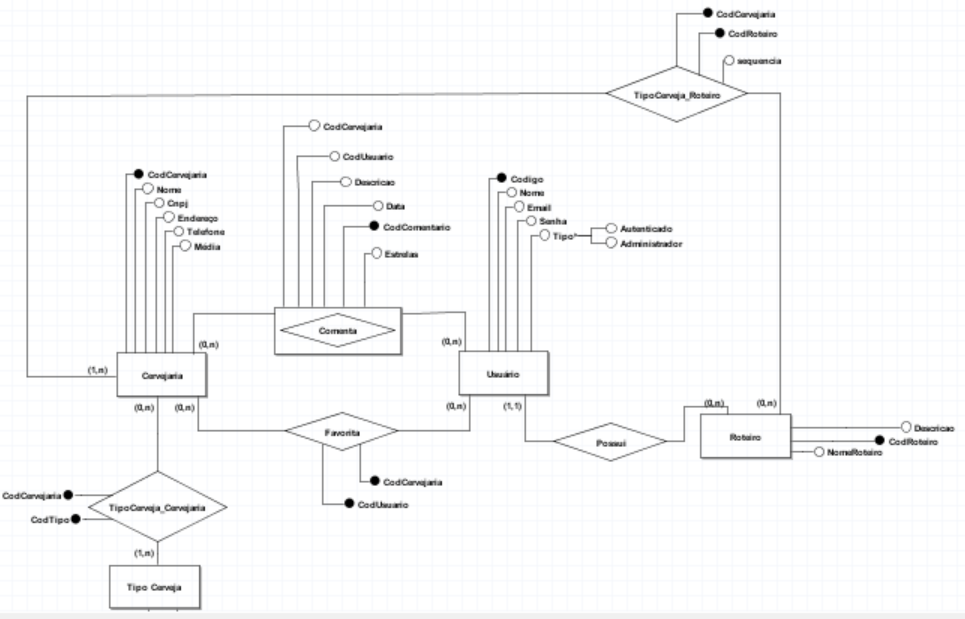
Sorocaba

Dezembro/2022

1. **Descrição do contexto**

Este projeto descreve a arquitetura do esquema de banco de dados do projeto de aplicação mobile Rota Cervejeira, que permitirá aos seus usuários a composição de um guia interativo, onde será possível criar roteiros turísticos para visitação das cervejarias artesanais existentes na Região Metropolitana de Sorocaba, associadas à Associação Cerveja Livre.

**2. Preparação da base de dados**

**a. Elaborar o DER das tabelas envolvidas.**

**b. Criar os scripts das tabelas, definindo as cONstraints após a criação usando o comando ALTER Table.**

* **CRIAÇÃO DAS TABELAS**

**CREATE TABLE** tb\_usuario(

cod number(5) not null,

nome varchar2(30) not null,

email varchar2(50) not null,

senha varchar2(12) not null,

tipo varchar2(15) not null

);

**CREATE TABLE** tb\_cervejaria(

codCervejaria number(5) not null,

nome varchar2(30) not null,

cnpj varchar2(14) not null,

media number(4, 2),

endereco varchar2(50) not null,

telefone varchar2(12) not null

);

**CREATE TABLE** tb\_roteiro(

codRoteiro number(5) not null,

nomeRoteiro varchar2(20) not null,

descricao varchar2(100),

codUsuario number(5) not null

);

**CREATE TABLE** tb\_tipoCerveja(

codTipo number(5) not null,

nomeTipo varchar(50) not null

);

**CREATE TABLE** tb\_favoritar(

codCervejaria number(5) not null,

codUsuario number(5) not null,

contador number(1)

);

**CREATE TABLE** tb\_comentario(

codComentario number(5) not null,

codCervejaria number(5) not null,

codUsuario number(5) not null,

descricaoComentario varchar(100),

dataComentario date,

estrelas number(1)

);

**CREATE TABLE** tb\_tipoCerveja\_Cervejaria(

codCervejaria number(5) not null,

codTipo number(5) not null

);

**CREATE TABLE** tb\_Cervejaria\_Roteiro(

codCervejaria number(5) not null,

codRoteiro number(5) not null,

-- ordem escolhida pelo usuário das cervejarias do roteiro

sequencia number(5) not null

);

* **ADICIONANDO CONSTRAINTS**
* **TABELA USUARIO**

**ALTER TABLE** tb\_usuario add cONstraint pk\_codUsuario primary key (cod);

**ALTER TABLE** tb\_usuario add cONstraint chk\_tipoUsuario check (tipo in('Autenticado', 'Administrador'));

* **TABELA CERVEJARIA**

**ALTER TABLE** tb\_cervejaria add cONstraint pk\_codCervejaria primary key (codCervejaria);

* **TABELA ROTEIRO**

**ALTER TABLE** tb\_roteiro add cONstraint pk\_codRoteiro primary key (codRoteiro);

**ALTER TABLE** tb\_roteiro add cONstraint fk\_codUsuario foreign key (codUsuario) references tb\_usuario;

* **TABELA TIPO CERVEJA**

**ALTER TABLE** tb\_tipoCerveja add cONstraint pk\_codTipo primary key (codTipo);

* **TABELA FAVORITAR**

**ALTER TABLE** tb\_favoritar add cONstraint fk\_codCervejaria foreign key (codCervejaria) references tb\_cervejaria;

**ALTER TABLE** tb\_favoritar add cONstraint fk\_codUsuario foreign key (codUsuario) references tb\_usuario;

**ALTER TABLE** tb\_favoritar add cONstraint pk\_codCervejaria\_codUsuario primary key (codCervejaria, codUsuario);

* *Garante que o user não favorite 2x uma mesma cervejaria*
* **TABELA COMENTARIO**

**ALTER TABLE** tb\_comentario add cONstraint pk\_codComentario primary key (codComentario);

**ALTER TABLE** tb\_comentario add cONstraint fk\_codCervejaria\_comentario foreign key (codCervejaria) references tb\_cervejaria;

**ALTER TABLE** tb\_comentario add cONstraint fk\_codUsuario\_comentario foreign key (codUsuario) references tb\_usuario;

* **TABELA TIPOCERVEJA - CERVEJARIA**

**ALTER TABLE** tb\_tipoCerveja\_Cervejaria add cONstraint fk\_codCervejaria\_tipo foreign key (codCervejaria) references tb\_cervejaria;

**ALTER TABLE** tb\_tipoCerveja\_Cervejaria add cONstraint fk\_codTipo\_tipo foreign key (codTipo) references tb\_tipoCerveja;

**ALTER TABLE** tb\_tipoCerveja\_Cervejaria add cONstraint pk\_codCervejaria\_codTipo primary key (codCervejaria, codTipo);

* *Garante que o user não adiciONe o mesmo tipo 2x em uma mesma cervejaria*
* **TABELA CERVEJARIA - ROTEIRO**

**ALTER TABLE** tb\_Cervejaria\_Roteiro add cONstraint fk\_codCervejaria foreign key (codCervejaria) references tb\_cervejaria;

**ALTER TABLE** tb\_Cervejaria\_Roteiro add cONstraint fk\_codRoteiro foreign key (codRoteiro) references tb\_roteiro;

**ALTER TABLE** tb\_Cervejaria\_Roteiro add cONstraint pk\_codCervejaria\_codRoteiro (codCervejaria, codRoteiro);

* *Garante que o usuário não adicione 2x a cervejaria no roteiro*

**c. Incluir dados significativos suficientes para se realizar os testes dos comandos e procedimentos que serão criados.**

* **INSERÇÃO DE DADOS NAS TABELAS**
* **TABELA USUÁRIO**

**INSERT INTO** tb\_usuario values (1, 'João da Silva', 'joao.silva@gmail.com', 'senha', 'Administrador');

**INSERT INTO** tb\_usuario values (2, 'Maria das Dores', 'maria1997@gmail.com', 'senha', 'Autenticado');

**INSERT INTO** tb\_usuario values (3, 'Beatriz Bonaparte', 'bea.bona@gmail.com', 'senha', 'Autenticado');

**INSERT INTO** tb\_usuario values (4, 'Luiz Soares', 'l.soares23@gmail.com', 'senha', 'Autenticado');

**INSERT INTO** tb\_usuario values (5,'Julia Lima', 'julialima98@gmail.com', 'senha', 'Autenticado');

* **TABELA CERVEJARIA**

**INSERT INTO** tb\_cervejaria values (1, 'Burgman', 'XXXXXX0001XX', '', 'Av. Eng. Carlos Reinaldo Mendes, 5025', '15985621965');

**INSERT INTO** tb\_cervejaria values (2, 'Wayne190', 'XXXXXXXX0123', '', 'R. Bom Pastor, 2043', '1598543900');

**INSERT INTO** tb\_cervejaria values (3, 'Synergy', 'XXXXXX000132', '', 'Rua Aparecida José Nunes de Campos, 53', '15956324000');

**INSERT INTO** tb\_cervejaria values (4, 'HappyBrew', 'XXXXXX000658', '', 'Rua José, 44', '1589564000');

* **TABELA ROTEIRO**

**INSERT INTO** tb\_roteiro values (25, 'Passeio','Foi incrível!', 1 );

**INSERT INTO** tb\_roteiro values (26, 'FDS', 'Muito divertido', 3);

**INSERT INTO** tb\_roteiro values (32, 'conhecendo a Rota', 'Bem legal, recomendo a todos', 5);

**INSERT INTO** tb\_roteiro values (40, 'Rota Cerveja', '', 4);

**INSERT INTO** tb\_roteiro values (63, 'Rota de Sabado', '', 2);

* **TABELA TIPO CERVEJA**

**INSERT INTO** tb\_tipoCerveja values (1, 'Pilsen');

**INSERT INTO** tb\_tipoCerveja values (2, 'Lager');

**INSERT INTO** tb\_tipoCerveja values (3, 'Pale Ale');

**INSERT INTO** tb\_tipoCerveja values(4, 'Tripel');

**INSERT INTO** tb\_tipoCerveja values(5, 'Stout');

* **TABELA FAVORITAR**

**INSERT INTO** tb\_favoritar values (1, 3);

**INSERT INTO** tb\_favoritar values (2, 3);

**INSERT INTO** tb\_favoritar values (4, 1);

**INSERT INTO** tb\_favoritar values (4, 2);

**INSERT INTO** tb\_favoritar values (1, 2);

* **TABELA COMENTÁRIO**

**INSERT INTO** tb\_comentario values (10, 1, 5, 'Adorei o passeio, uma das melhoras cervejarias que visitamos!', to\_date('10/11/2022', 'DD/MM/YYYY'),5);

**INSERT INTO** tb\_comentario values (15, 2, 4, 'Gostei muito dos estilos e do atendimento, voltarei em breve', to\_date('01/12/2022', 'DD/MM/YYYY'),3);

**INSERT INTO** tb\_comentario values (20, 3, 1, 'Minha familia adorou, com certeza voltaremos', to\_date('15/11/2022', 'DD/MM/YYYY'),1);

**INSERT INTO** tb\_comentario values (21, 1, 1, 'Muito legal!', to\_date('15/11/2022', 'DD/MM/YYYY'),2);

**INSERT INTO** tb\_comentario values (22, 1, 1, 'Adorei', to\_date('15/11/2022', 'DD/MM/YYYY'),1);

* **TABELA TIPO CERVEJA - CERVEJARIA**

**INSERT INTO** tb\_tipoCerveja\_Cervejaria values (1, 2);

**INSERT INTO** tb\_tipoCerveja\_Cervejaria values (2, 2);

**INSERT INTO** tb\_tipoCerveja\_Cervejaria values (3, 1);

**INSERT INTO** tb\_tipoCerveja\_Cervejaria values (1, 1);

**INSERT INTO** tb\_tipoCerveja\_Cervejaria values (1, 3);

**INSERT INTO** tb\_tipoCerveja\_Cervejaria values (2, 1);

**INSERT INTO** tb\_tipoCerveja\_Cervejaria values (3, 2);

**INSERT INTO** tb\_tipoCerveja\_Cervejaria values (3, 3);

**INSERT INTO** tb\_tipoCerveja\_Cervejaria values (4, 2);

* **TABELA CERVEJARIA ROTEIRO**

**INSERT INTO** tb\_Cervejaria\_Roteiro values (1, 26, 1);

**INSERT INTO** tb\_Cervejaria\_Roteiro values (1, 40, 1);

**INSERT INTO** tb\_Cervejaria\_Roteiro values (1, 63, 1);

**INSERT INTO** tb\_Cervejaria\_Roteiro values (2, 26, 2);

**INSERT INTO** tb\_Cervejaria\_Roteiro values (3, 32, 1);

**d. Criar índices para as colunas com muitas consultas.**

As tabelas mais consultadas serão Cervejaria e Tipo Cerveja, e as pessoas vão pesquisar tanto as cervejarias quanto tipo pelo Nome, então faz-se necessário criar um índice para o campo nome em ambas as tabelas

**CREATE INDEX** dx\_nomeCervejaria\_tbCervejarias

**ON** tb\_cervejaria(nome);

**CREATE INDEX** dx\_nomeTipoCerveja\_tbTipoCerveja

**ON** tb\_tipoCerveja(nomeTipo);

**3.** **Consultas: Criar consultas significativas para a aplicação**

**a. Uma consulta usando junção de mais de 2 tabelas.**

* Lista os nomes de usuários que incluiram uma determinada cervejaria em seus roteiros. Ex: listar Burgman

**SELECT DISTINCT** u.cod, u.nome FROM tb\_usuario U

**INNER JOIN** tb\_roteiro r

**ON** u.cod = r.codUsuario

**INNER JOIN** tb\_Cervejaria\_Roteiro cr

**ON** r.codRoteiro = cr.codRoteiro

**WHERE** cr.codCervejaria = 1;

**b. Uma consulta que seja útil para a lógica de negócios usando totalização e uma função de data.**

* Lista a quantidade de comentários feitos para uma determinada cervejaria em uma determinada data

**SELECT COUNT**(\*) FROM tb\_comentario

**WHERE** dataComentario = to\_date('15/11/2022', 'DD/MM/YYYY')

**AND** codCervejaria = 1;

OU

* Lista a quantidade de comentário de cada cervejaria que possui comentários em uma determinada data;

**SELECT** c.nome, **COUNT**(\*as Qtde\_comentario

**FROM** tb\_comentario co

**INNER JOIN** tb\_cervejaria c

**ON** co.codCervejaria = c.codCervejaria

**WHERE** dataComentario = to\_date('15/11/2022', 'DD/MM/YYYY') **GROUP BY** c.codCervejaria, c.nome;

**c. Uma consulta usando junção externa left join ou right join**

* Lista a quantidade total de comentários que cada cervejaria possui

**SELECT** c.nome, count(co.codCervejaria) **AS** Qtde\_comentario

**FROM** tb\_cervejaria c

**LEFT** **JOIN** tb\_comentario co

**ON** co.codCervejaria = c.codCervejaria

**GROUP BY** c.codCervejaria, c.nome;

**d. Uma consulta usando o operador de União.**

* Seleciona o endereço de duas cervejarias diferentes

**SELECT** nome, ENDereco

**FROM** tb\_cervejaria

**WHERE** codCervejaria = 1

**UNION**

**SELECT** nome, ENDereco

**FROM** tb\_cervejaria

**WHERE** codCervejaria = 2

**e. Uma consulta usando o operador Minus**

* Retorna as cervejarias que não foram favoritadas por nenhum usuário

**SELECT** codcervejaria, nome

**FROM** tb\_cervejaria minus(

**SELECT** tb\_cervejaria.codcervejaria, tb\_cervejaria.nome

**FROM** tb\_cervejaria

**INNER JOIN** tb\_favoritar

**ON** tb\_cervejaria.codcervejaria = tb\_favoritar.codcervejaria);

**f. Uma consulta usando o operador de intersecção**

* Retorna os tipos de cervejas comercializados pelas cervejarias

**SELECT** codtipo

**FROM** tb\_tipocerveja

**INTERSECT**

**SELECT** codtipo

**FROM** tb\_tipocerveja\_cervejaria;

**4. Subconsultas (sobre o tema escolhido)**

**a. Exemplo de um comando utilizando subconsultas que utilize o operador in.**

* Seleciona código e nome das cervejarias que possuem o tipo de cerveja "Pale Ale"

**SELECT** tb\_cervejaria.codcervejaria, tb\_cervejaria.nome

**FROM** tb\_cervejaria

**INNER JOIN** tb\_tipocerveja\_cervejaria

**ON** tb\_cervejaria.codcervejaria = tb\_tipocerveja\_cervejaria.codcervejaria

**INNER JOIN** tb\_tipocerveja

**ON** tb\_tipocerveja\_cervejaria.codtipo = tb\_tipocerveja.codtipo

**WHERE** tb\_tipocerveja.nometipo **IN** ('Pale Ale')

**b. Exemplo de um comando utilizando subconsultas que utilize o operador not exists**

* Lista as cervejarias que não possuem o tipo de cerveja de código = 3

**SELECT** codcervejaria, nome

**FROM** tb\_cervejaria

**WHERE NOT EXISTS** (**SELECT** \* **FROM** tb\_tipocerveja\_cervejaria

**WHERE** tb\_cervejaria.codcervejaria = tb\_tipocerveja\_cervejaria.codcervejaria

**AND** codtipo = 3);

**c. Exemplo de uma subconsulta utilizada dentro de um comando UPDATE.**

* Atualiza o valor da média de estrelas das cervejarias

**UPDATE** tb\_cervejaria

**SET** media = (**SELECT** avg(tb\_comentario.estrelas)

**FROM** tb\_comentario

**WHERE** tb\_cervejaria.codcervejaria = tb\_comentario.codcervejaria

**AND**

tb\_cervejaria.media **IS NULL**

**GROUP BY** (tb\_comentario.codcervejaria));

**d. Exemplo de uma subconsulta utilizada dentro de um comando DELETE.**

* Excluir da tabela de tb\_tipocerveja os tipos que não são fornecidos por nenhuma cervejaria

**DELETE** FROM tb\_tipocerveja

**WHERE** codtipo

**NOT IN**(**SELECT** codtipo **FROM** tb\_tipocerveja\_cervejaria);

=========================================================

**PL/SQLs**

1. **Escreva uma Stored Procedure que realize algum procedimento que seja útil implementar reuso para a aplicação e indique uma forma de testá-la.**

* Altera a quantidade de estrelas que um usuário atribuiu à uma cervejaria

**CREATE OR REPLACE** procedure Atualiza\_Estrelas(PcodCervejaria number, PcodUsuario number, Pestrelas number)

**AS**

v\_codComentario tb\_comentario.codcomentario%type;

**BEGIN**

**SELECT** codcomentario **INTO** v\_codComentario

**FROM** tb\_comentario

**WHERE** codusuario = PcodUsuario and codcervejaria = PcodCervejaria;

**IF** v\_codcomentario is not null THEN

**UPDATE** tb\_comentario set estrelas = Pestrelas

**WHERE** codcomentario = v\_codComentario;

**END IF**;

**EXCEPTION**

when no\_data\_found THEN

**INSERT INTO** tab\_erro values (sysdate,'Comentário inexistente.');

**END;**

**EXEC** Atualiza\_Estrelas(8, 1, 5);

**SELECT** \* **FROM** tb\_comentario;

1. **Escreva uma função que retorna alguma informação útil para a aplicação e indique uma forma de testá-la.**

* **Function:** Fn\_devolve\_nomeTipo **-** Lista o nome das cervejarias e todos os tipos que fornecem

**CREATE OR REPLACE** Function Fn\_devolve\_nomeTipo(PcodTipo tb\_tipocerveja.codtipo%type)

return varchar2

**AS**

Vnome tb\_tipocerveja.nometipo%type;

**BEGIN**

**SELECT** nometipo **INTO** Vnome

**FROM** tb\_tipocerveja

**WHERE** tb\_tipocerveja.codtipo = PcodTipo;

Return (Vnome);

**END** Fn\_devolve\_nomeTipo;

Teste:

**SELECT** tb\_cervejaria.codcervejaria, tb\_cervejaria.nome, Fn\_devolve\_nomeTipo(codtipo) **AS** tipo

**FROM** tb\_cervejaria

**INNER JOIN** tb\_tipocerveja\_cervejaria

**ON** tb\_cervejaria.codcervejaria = tb\_tipocerveja\_cervejaria.codcervejaria;

* **Function:** coment\_maisCurtido - unção que retorna o código do comentário com mais estrelas de uma determinada cervejaria

**CREATE OR REPLACE** function coment\_maisCurtido(codCervejaria tb\_comentario.codCervejaria%type)

return number

**AS**

v\_codComent tb\_comentario.codComentario%type;

**BEGIN**

**SELECT** co.codComentario INTO v\_codComent

**FROM** tb\_comentario co WHERE co.codCervejaria = codCervejaria

**ORDER BY** co.estrelas desc

fetch first 1 rows ONly;

return v\_codComent;

**EXCEPTION**

when no\_data\_found THEN

return 'O código da cervejaria é inexistente' || codCervejaria;

**END** coment\_maisCurtido;

**SELECT** coment\_maisCurtido(codCervejaria)

**FROM** tb\_cervejaria

**SELECT** co.codComentario

**FROM** tb\_comentario co WHERE co.codCervejaria = 3

**ORDER BY** co.estrelas desc

**FETCH FIRST** 1 **ROWS ONLY**;

1. **Escreva um trigger de auditoria para criação e uso de logs sempre que alguma operação crítica for realizada. Deverá usar :NEW e :OLD.**

* **Trigger: TG\_log\_tb\_cervejaria** - Quando a tabela tb\_cervejaria sofrer uma operação de DELETE ou UPDATE é gerado um registro na tabela referente a operação realizada.

**CREATE OR REPLACE** trigger TG\_log\_tb\_cervejaria

**before** **DELETE** **or UPDATE ON** tb\_cervejaria

**FOR EACH ROW**

**BEGIN**

**IF** updating **THEN**

**INSERT INTO** tb\_log\_DELETE **values**(sysdate,user,'tb\_cervejaria', :new.codCervejaria);

**ELSE**

**INSERT INTO** tb\_log\_UPDATE **values**(sysdate,user,'tb\_cervejaria', :old.codCervejaria);

**END IF**;

**END** TG\_ TG\_log\_tb\_cervejaria;

Teste:

**INSERT INTO** tb\_cervejaria values (99, 'Exemplo', 'XXXXXX000999', 0, 'Rua Joâo, 44', '1589564000');

**UPDATE** tb\_cervejaria **set** ENDereco = 'Rua Nova, 48' WHERE codCervejaria = 99;

**DELETE FROM** tb\_cervejaria **WHERE** codcervejaria = 99;

**SELECT** \* **FROM** tb\_log\_DELETE;

**SELECT** \* **FROM** tb\_log\_UPDATE;

* **Trigger: TG\_registro\_comentario** – registra atualização aplicadas a um comentário feito ou excluído por um usuário.

**CREATE OR REPLACE** TRIGGER TR\_REGISTRO\_COMENTARIO

**BEFORE UPDATE OR DELETE ON TB\_COMENTARIO**

**FOR EACH ROW**

**DECLARE**

Vcodcomentario tb\_comentario.codcomentario%type;

Vcodcervejaria tb\_comentario.codcervejaria%type;

Vcodusuario tb\_comentario.codusuario%type;

Vdescricaocomentario\_alterada tb\_comentario.descricaocomentario%type;

Vdescricaocomentario\_new tb\_comentario.descricaocomentario%type;

Vdatacomentario tb\_comentario.datacomentario%type;

**BEGIN**

Vcodcomentario := :old.codcervejaria;

Vcodcervejaria := :old.codcervejaria;

Vcodusuario := :old.codusuario;

Vdescricaocomentario\_alterada := :old.descricaocomentario;

Vdescricaocomentario\_new := :new.descricaocomentario;

Vdatacomentario := :old.datacomentario;

**IF** updating **THEN**

**INSERT INTO** tb\_log\_comentario values(cod\_log\_comentario.nextval, 'UPDATE', Vcodcomentario, Vcodcervejaria, Vcodusuario, Vdescricaocomentario\_alterada,Vdescricaocomentario\_new,Vdatacomentario);

**END IF;**

**IF** deleting **THEN**

**INSERT INTO** tb\_log\_comentario values(cod\_log\_comentario.nextval, 'DELETE', Vcodcomentario, Vcodcervejaria, Vcodusuario, Vdescricaocomentario\_alterada,Vdescricaocomentario\_new,Vdatacomentario);

**END IF;**

**END;**

Teste:

**UPDATE** tb\_comentario **SET** descricaocomentario = 'Voltaremos sempre' **WHERE** codcomentario = 15;

**DELETE FROM** tb\_comentario **WHERE** codcomentario = 10;

**d. Escreva** **um trigger para impor uma integridade de dados mais complexa do que uma restrição Check(por exemplo).**

* Criação da view para evitar table 'mutating'

**CREATE OR REPLACE** view comentarios **AS**

**SELECT** \* **FROM** tb\_comentario;

* **Trigger: TG\_AtualizaMedia** - Atualiza a média quando um novo comentário é adicionado.

**CREATE OR REPLACE** trigger TG\_AtualizaMedia

instead of insert **ON** comentarios

**FOR EACH ROW**

declare vMedia tb\_cervejaria.media%type;

**BEGIN**

**SELECT** **avg**(tb\_comentario.estrelas) **INTO** vMedia

**FROM** tb\_comentario

**INNER JOIN** tb\_cervejaria **ON** tb\_cervejaria.codcervejaria = tb\_comentario.codcervejaria

**WHERE** tb\_cervejaria.codcervejaria = :new.codcervejaria;

**UPDATE** tb\_cervejaria **set** media = vMedia

**WHERE** codcervejaria = :new.codcervejaria;

**END** TG\_AtualizaMedia;

* **Trigger: Verifica\_Cervejaria\_Duplicada\_Roteiro** – Impede que uma cervejaria já existente em um roteiro seja adicionada novamente

**CREATE OR REPLACE TRIGGER** Verifica\_Cervejaria\_Duplicada\_Roteiro

**BEFORE INSERT OR UPDATE ON** tb\_roteiro

**FOR EACH ROW**

**DECLARE**

v\_old\_cervejaria number := :old.sequencia;

v\_old\_codroteiro number := :old.codroteiro;

**BEGIN**

**IF** updating **THEN**

**if** v\_old\_cervejaria = :new.sequencia **and** v\_old\_codroteiro = :new.codroteiro **then**

raise\_application\_error(-20001,'Cervejaria já adicionada a este roteiro!');

**end if**;

**END IF**;

**IF** inserting **THEN**

**if** v\_old\_cervejaria = :new.sequencia **and** v\_old\_codroteiro = :new.codroteiro **then**

raise\_application\_error(-20001,'Cervejaria já adicionada a este roteiro!');

**end if**;

**END IF;**

**COMMIT;**

**END;**

Teste:

**SELECT** \* **FROM** tb\_roteiro;

**UPDATE** tb\_roteiro **SET** cervejaria\_1 = 1

**WHERE** codroteiro = 26;

**INSERT INTO** tb\_roteiro **VALUES** (26, 3, 'Roteiro Burgman', 1);