

**Guilherme de Sena Lima Guimarães**

**Matheus Henrique Marques Canuto**

**SITUAÇÃO DE APRENDIZAGEM BANCO DE DADOS**

**BELO HORIZONTE 2021**

**Guilherme de Sena Lima Guimarães**

**Matheus Henrique Marques Canuto**

**SITUAÇÃO DE APRENDIZAGEM BANCO DE DADOS**

Trabalho para Conclusão do 3º Módulo do Curso de Técnico de Informática para Internet do CTTI.

**BELO HORIZONTE 2021**

**Sumário**

**Tipos de Arquitetura de Bancos de Dados**

**Arquitetura Centralizada**

Na arquitetura centralizada, existe um computador com grande capacidade de processamento, o qual é o hospedeiro do SGBD e emuladores para os vários aplicativos. Esta arquitetura tem como principal vantagem a de permitir que muitos usuários manipulem grande volume de dados. Sua principal desvantagem está no seu alto custo, pois exige ambiente especial para mainframes e soluções centralizadas.

**Descrição do Banco de Dados:**

**Nível externo:**

Especificação da organização

Conceitual do BD, vista por um grupo de usuários;

**Nível conceitual:**

Especificação da organização conceitual do BD, ou seja, o quê o BD armazena;

**Nível físico ou interno:**

Especificação das estruturas de armazenamento do BD, ou seja, como o BD está armazenado;



**Arquitetura Cliente-Servidor**

Na arquitetura Cliente-Servidor, o cliente (front\_end) executa as tarefas do aplicativo, ou seja, fornece a interface do usuário (tela, e processamento de entrada e saída). O servidor (back\_end) executa as consultas no DBMS e retorna os resultados ao cliente. Apesar de ser uma arquitetura bastante popular, são necessárias soluções sofisticadas de software que possibilitem: o tratamento de transações, as confirmações de transações (commits), desfazer transações (rollbacks), linguagens de consultas (stored procedures) e gatilhos (triggers). A principal vantagem desta arquitetura é a divisão do processamento entre dois sistemas, o que reduz o tráfego de dados na rede.

**Servidor de Objetos:**

Servidor transfere objetos para a object cache area do cliente. Cliente é responsável por:

* Executar os métodos comandados pela aplicação.
* Obter do servidor os objetos necessários.
* Gerenciar os objetos armazenados na object cache area local, incluindo a criação, atualização e remoção de objetos.

Servidor é responsável pelo:

* Processamento de consultas, incluindo a recolocação dos objetos atualizados no banco.
* Gerência de transações, incluindo a sincronização da object cache area dos cliente.
* Servidor pode fazer pre-fetch de objetos e enviá-los ao cliente.

**Servidor de Páginas:**

Servidor transfere páginas para a buffer pool area do cliente. Cliente é responsável por:

* Executar os métodos comandados pela aplicação.
* Implementar toda a semântica associada a objetos.

Servidor é responsável pelo:

* Armazenamento físico dos objetos apenas.



**Arquitetura Distribuída**

Nesta arquitetura, a informação está distribuída em diversos servidores. Cada servidor atua como no sistema cliente-servidor, porém as consultas oriundas dos aplicativos são feitas para qualquer servidor indistintamente. Caso a informação solicitada seja mantida por outro servidor ou servidores, o sistema encarrega-se de obter a informação

necessária, de maneira transparente para o aplicativo, que passa a atuar consultando a rede, independente de conhecer seus servidores.

Exemplos típicos são as bases de dados corporativas, em que o volume de informação é muito grande e, por isso, deve ser distribuído em diversos servidores. Porém, não é dependente de aspectos lógicos de carga de acesso aos dados, ou base de dados fracamente acopladas, em que uma informação solicitada vai sendo coletada numa propagação da consulta numa cadeia de servidores.

A característica básica é a existência de diversos programas aplicativos consultando a rede para acessar os dados necessários, porém, sem o conhecimento explícito de quais servidores dispõem desses dados.

**BD Distribuído**

Banco de dados armazenado em vários servidores, conectados por uma rede de comunicação de dados;

**SGBD Distribuido**

Estende as técnicas de armazenamento de dados, processamento de consultas e gerência de transações para permitir a implementação de BDDs;

**Bancos de dados distribuídos homogêneos**

Mesmo software/esquema em todos os sites, os dados podem estar particionados entre os sites;

Objetivo: oferecer uma visão de um único banco de dados, ocultando os detalhes da distribuição;

**Bancos de dados distribuídos heterogêneos**

Software/esquema diferente em sites diferentes;

Objetivo: integrar bancos de dados existentes para oferecer funcionalidade útil;



**Arquitetura Paralela**

**Introdução:**

* Combinam técnicas de gerência de dados e processamento paralelo para aumentar desempenho e confiabilidade:

Particionamento do BD em discos controlados por multiprocessadores resulta em aumento da taxa de transferência de dados da memória secundária para memória principal (I/O bandwidth);

Paralelização do processamento interno de consultas resulta em diminuição do tempo de resposta;

Paralelização do processamento de transações resulta em aumento da capacidade do sistema (throughput);

**Memória compartilhada**

Processadores e discos têm acesso a uma memória comum, normalmente por meio de um barramento ou por uma rede de interconexão.

Comunicação extremamente eficiente entre os processadores - os dados na memória compartilhada podem ser acessados por qualquer processador sem ter que movê-los usando software.

Desvantagem - A arquitetura não é expansível além de 32 ou 64 processadores, pois o barramento ou a rede de interconexão se torna um gargalo.

Muito usada para graus de paralelismo menores (4 a 8).

**Disco compartilhado**

Processadores compartilham um disco comum;

Todos os processadores podem acessar diretamente todos os discos por meio de uma rede de interconexão, mas os processadores têm memórias privadas.

O barramento da memória não é um gargalo;

A arquitetura oferece um grau de tolerância a falhas - se um processador falhar, os outros processadores podem assumir suas tarefas, pois o banco de dados é residente nos discos que são acessíveis por todos os processadores.

Exemplos: clusters IBM Sysplex e DEC (agora parte da Compaq) rodando Rdb (agora Oracle Rdb) foram os primeiros usuários comerciais.

Desvantagem: o gargalo agora ocorre na interconexão com o subsistema de disco.

Sistemas de disco compartilhado podem se expandir até um número um pouco maior de processadores, mas a comunicação entre os processadores é mais lenta.

**Nada compartilhado**

Processadores não compartilham memória comum nem disco comum; O nó consiste em um processador, memória e um ou mais discos.

Os processadores em um nó se comunicam com outro processador em outro nó, usando uma rede de interconexão. Um nó funciona como servidor para os dados no disco ou discos que o nó possui.

**Exemplos: Teradata, Tandem, Oracle-n CUBE**

Os dados acessados a partir de discos locais (e acessos a memória locais) não passam pela rede de interconexão, minimizando assim a interferência do compartilhamento de recursos.

Os multiprocessadores de nada compartilhado podem ser expandidos até milhares de processadores sem interferência.

Desvantagem principal: custo de comunicação e acesso a disco não local, o envio de dados envolve a interação do software nas duas pontas.

**Hierárquica**

Híbrido das arquiteturas acima.

Combina características de arquiteturas de memória compartilhada, disco compartilhado e nada compartilhado.

Nível superior é uma arquitetura nada compartilhada. Os nós conectados por uma rede de interconexão e não compartilha discos ou memória entre si.

Cada nó do sistema poderia ser um sistema de memória compartilhada com alguns processadores.

Como alternativa, cada nó poderia ser um sistema de disco compartilhado, e cada um dos sistemas compartilhando um conjunto de discos poderia ser um sistema de memória compartilhada.

Reduz a complexidade de programar tais sistemas por arquiteturas de memória virtual distribuída.

Também chamada **arquitetura de memória não uniforme.**



**Relatório Técnico**

O modelo de banco de dados escolhido para criação do projeto foi o banco de dados relacional – SQL com arquitetura estendível por atender os requisitos de segurança quanto a transação por ser atômica, consistente, isolada e durável, proteção e tratamento dos dados como a integridade tem como objetivo assegurar que a informação seja protegida contra qualquer alteração imprópria ou não autorizada. A alteração inclui qualquer inclusão, alteração ou exclusão feita intencionalmente ou não nos dados por qualquer software, pessoa ou dispositivo não autorizado. Neste caso a integridade dos dados é perdida e o uso continuado dos dados pode resultar em decisões erradas e informações incorretas nos sistemas, a disponibilidade é para assegurar que os dados estarão disponíveis aos usuários e sistemas que estão autorizados a acessá-los no momento em que forem acessados, a confidencialidade tem como objetivo assegurar que os dados não serão publicados à pessoas não autorizadas a acessá-los e a exposição não autorizada de dados que poderia resultar, entre outras coisas, à perda de confiança, processos legais contra a empresa, constrangimento, etc.

A arquitetura selecionada promete resolver os problemas de transação de dados:

**Atomicidade**

Todas as ações que compõem a unidade de trabalho da transação devem ser concluídas com sucesso, para que seja efetivada. Se durante a transação qualquer ação que constitui unidade de trabalho falhar, a transação inteira deve ser desfeita (rollback). Quando todas as ações são efetuadas com sucesso, a transação pode ser efetivada e persistida em banco (commit).

**Consistência**

Todas as regras e restrições definidas no banco de dados devem ser obedecidas. Relacionamentos por chaves estrangeiras, checagem de valores para campos restritos ou únicos devem ser obedecidos para que uma transação possa ser completada com sucesso.

**Isolamento**

Cada transação funciona completamente à parte de outras estações. Todas as operações são parte de uma transação única. O principio é que nenhuma outra transação, operando no mesmo sistema, possa interferir no funcionamento da transação corrente(é um mecanismo de controle). Outras transações não podem visualizar os resultados parciais das operações de uma transação em andamento (ainda em respeito à propriedade da atomicidade).

**Durabilidade**

Significa que os resultados de uma transação são permanentes e podem ser desfeitos somente por uma transação subsequente.Por exemplo: todos os dados e status relativos a uma transação devem ser armazenados num repositório permanente, não sendo passíveis de falha por uma falha de hardware.

Para proteger um Banco de Dados contra os problemas citados e garantir que estes objetivos de segurança sejam cumpridos foram implementadas algumas medidas de controle.

**Controle de Acesso**

Em um sistema de banco de dados multiusuário, o sistema deve oferecer técnicas para permitir que um conjunto de usuários tenha acesso somente a parte selecionada dos dados sem que tenham acesso ao restante do banco de dados, o que é de suma importância quando se trata de um grande banco de dados integrado que precisa ser acessado por vários usuários e sistemas dentro da organização.

Para isso, o SGBD inclui um sistema de autorização de acesso e um subsistema de segurança que é responsável por garantir a segurança do banco de dados contra cesso não autorizado. São usados dois mecanismos para isso:

Mecanismos de Segurança Discricionários - São usados para conceder acesso aos usuários para consulta e atualização às arquivos, registros e campos de um banco de dados.

Mecanismos de Segurança obrigatórios - São usados para conceder acesso ao usuários através da classificação dos usuários com base nas funções que eles desempenham no banco de dados. Para isso é usado o conceito de papéis ( roles ) ou perfis de usuário que são usados para determinar quem pode acessar o que de acordo com o papel que possui no sistema.

**Modelo Físico**

create database SA;

use SA;

#drop database SA;

#lista tabelas

#Todos as entidades estÃ£o com chave estrageira (NÃ£o)

#endereco\_agente ok chaves estrangeiras ok

#comissao ok chaves estrangeiras ok

#agente ok chaves estrangeiras ok

#emprestimo ok chaves estrangeiras ok

#parcela ok chaves estrangeiras ok

#cliente (completar chaves estrangeiras) chaves estrangeiras ok

#informacao\_comercial ok chaves estrangeiras ok

#conta\_cliente ok chaves estrangeiras ok

#informacoes\_complementares ok chaves estrangeiras ok

#taxa\_administrativa ok chaves estrangeiras ok

#remessa ok chaves estrangeiras ok

#item\_remessa ok chaves estrangeiras ok

#lista\_banco\_permitidos ok chaves estrangeiras ok

#empresa ok

#conta\_empresa ok

create table endereco\_agente

(

id\_endereco int auto\_increment,

cep\_agente int,

rua\_agente varchar(45),

cidade\_agente varchar(45),

bairro\_agente varchar(45),

numero\_agente varchar(45),

primary key (id\_endereco)

);

insert into endereco\_agente

values (null, '30840010', 'Contadores', 'Belo Horizonte', 'AlipÃ­o de Melo', '420'),

(null, '30561781', 'Alameda das Acacias', 'Belo Horizonte', 'SÃ£o Joaquim', '65'),

(null, '35489734', 'Coronel JosÃ© Dias Bicalho', 'Belo Horizonte', 'SÃ£o Joaquim', '250');

create table parcela(

id\_parcela int auto\_increment,

situacao bit,

#0 para inativo e 1 para inativo

data\_vencimento date,

valor float,

valor\_apos\_vencimento float,

primary key (id\_parcela),

data\_pagamento date);

insert into parcela

values ('1', 1, '2021-10-05', '510','1000','2021-10-03'),

('2', 1, '2021-10-05', '300','1000','2021-10-03'),

('3', 1, '2021-10-05', '486','1000','2021-10-03');

CREATE TABLE emprestimo (

id\_emprestimo INT AUTO\_INCREMENT,

parcela\_id\_parcela INT,

numero\_contrato FLOAT,

situacao\_contrato BIT,

status\_contrato BIT,

data\_emprestimo DATE,

valor\_desconto FLOAT,

quantidade\_parcelas INT,

vencimento\_emprestimo DATE,

tipo\_vencimento\_emprestimo VARCHAR(45),

data\_primeira\_parcela DATE,

valor\_bruto\_emprestimo FLOAT,

categoria\_emprestimo VARCHAR(45),

data\_contrato\_emprestimo DATE,

usuario\_cadastro\_emprestimo VARCHAR(45),

atendente\_emprestimo INT,

valor\_liquido\_emprestimo INT,

PRIMARY KEY (id\_emprestimo),

FOREIGN KEY (parcela\_id\_parcela) REFERENCES parcela (id\_parcela)

);

insert into emprestimo

value (null, '1', '36987', 1, 1, '2021-08-07', '230', '3', '2021-05-11', 'juros', '2021-09-07', '1530', 'negativado', '2021-08-07', 'marcos ferreira','1', '1530'),

(null, '2', '98987', 1, 1, '2021-08-07', '123', '4', '2021-05-11', 'juros', '2021-09-07', '1200', 'negativado', '2021-08-07', 'luiz carlos', '1', '1200'),

(null, '3', '97898', 1, 1, '2021-08-07', '216', '15', '2021-05-11', 'juros', '2021-09-07', '7290', 'negativado', '2021-08-07', 'lucas claro', '1', '7290');

create table comissao(

id\_comissao int auto\_increment,

id\_emprestimo int ,

nome\_titular varchar(45),

banco varchar(45),

agencia int,

tipo\_conta varchar(45),

documento\_titular int,

percentual\_comissao float,

numero\_conta int,

primary key (id\_comissao),

foreign key (id\_emprestimo) references emprestimo (id\_emprestimo)

);

insert into comissao

values (null, '1','Guilherme de Sena','nubank','0001','corrente', '967532578', '10','983576218'),

(null, '2','Thiago Gomes','inter','0001','corrente', '787421547', '8','897544648'),

(null, '3','Pedro Silva','nubank','0001','corrente', '748976181', '7','897486424');

create table agente(

id\_agente int auto\_increment,

tipo\_agente varchar(2),

nome\_agente varchar(45),

celular int(13),

telefone\_fixo int(13),

email varchar(575),

status\_agente bit not null,

id\_endereco int ,

comissao\_status bit,

id\_comissao int ,

primary key (id\_agente),

foreign key (id\_endereco) references endereco\_agente (id\_endereco),

foreign key (id\_comissao) references comissao (id\_comissao)

);

insert into agente

values (null, 'pf', 'Guilherme de Sena','319956578', '31387524', 'guilherme.sena@gmail.com', 1, '1', 1, '1'),

(null, 'pj', 'Thiago Gomes','319879324', '31375963', 'Thiago.gomes@gmail.com', 1, '2', 1, '2'),

(null, 'pf', 'Pedro Silva','319936587', '31398956', 'Pedro.silva@gmail.com', 1, '3', 1, '3');

create table informacao\_complementar

(

id\_informacao\_complementar int auto\_increment,

nome\_mae varchar(45),

nome\_pai varchar(45),

cartorio varchar(45),

primary key (id\_informacao\_complementar)

);

insert into informacao\_complementar

values (null, 'Maria ventura', 'Marcos Ventura', 'Venda Nova'),

(null, 'Luana Padilha', 'Lucas Padilha', 'Centro'),

(null, 'Marina Lins', 'Luiz Lins', 'Venda Nova');

create table conta\_cliente(

id\_conta\_cliente int auto\_increment,

tipo\_conta varchar(45),

conta\_cliente int,

agencia\_cliente int,

digito\_conta\_cliente int,

operacao\_cliente varchar(45),

banco\_cliente varchar(45),

digito\_agencia int,

primary key (id\_conta\_cliente)

);

insert into conta\_cliente

values (null, 'corrente', '983567241', '0001', '9', 'corrente', 'Nubank', '1'),

(null, 'poupanÃ§a', '653817620', '0005', '8', 'poupanÃ§a', 'inter', '9'),

(null, 'poupanÃ§a', '848972375', '0009', '5', 'poupanÃ§a', 'inter', '5');

create table taxa\_administrativa(

#TA = taxa administrativa

id\_TA int auto\_increment,

situacao bit,

tipo\_vencimento varchar(45),

valor float,

vencimento date,

data\_inicio\_tax date,

primary key (id\_TA)

);

insert into taxa\_administrativa

values (null, 1, 'mensal', '51', '2021-10-05', '21-08-07'),

(null, 1, 'mensal', '30', '2021-10-05', '21-08-07'),

(null, 1, 'mensal', '48', '2021-10-05', '21-08-07');

create table lista\_bancos\_permitidos(

codigo\_banco int auto\_increment,

nome\_banco varchar(45),

primary key (codigo\_banco)

);

insert into lista\_bancos\_permitidos

values (null, 'nubank'),

(null, 'inter'),

(null, 'Banco do Brasil'),

(null, 'Bradesco');

create table informacao\_comercial(

id\_info\_comer int auto\_increment,

local\_de\_trabalho varchar(45),

tel\_comercial int,

beneficio varchar(45),

primary key (id\_info\_comer)

);

insert into informacao\_comercial

values (null,'INSS', '319885213', 'MASP'),

(null,'Bombeiro', '319744645', 'SIAPE'),

(null,'Prefeitura', '319537478', 'MASP');

create table cliente

(

#chave primaria

id\_cliente int auto\_increment,

#chave estrangeira

id\_emprestimo int,

id\_conta\_cliente int,

id\_informacao\_complementar int,

id\_taxa\_adimistrativa int,

id\_informacao\_comercial int,

id\_agente int,

#atributos

nome varchar (45),

sexo varchar(10),

telefone\_fixo int,

data\_nascimento date,

id\_documento int,

celular\_cliente int,

email\_cliente varchar (575),

data\_cadastro date,

cpf int,

primary key (id\_cliente),

foreign key (id\_emprestimo) references emprestimo (id\_emprestimo),

foreign key (id\_conta\_cliente) references conta\_cliente (id\_conta\_cliente),

foreign key (id\_informacao\_complementar) references informacao\_complementar (id\_informacao\_complementar),

foreign key (id\_taxa\_adimistrativa) references taxa\_administrativa (id\_TA),

foreign key (id\_informacao\_comercial) references informacao\_comercial (id\_info\_comer),

foreign key (id\_agente) references agente (id\_agente)

);

insert into cliente

values (null,'1','1','1','1','1','1','Thiago Ventura', 'Masculino', '3596','1989-10-21','97895882','985224156','thiago.ventura@gmail.com', '2021-05-10', '669861276'),

(null,'2','2','2','2','2','2','Afonso Padilha', 'Masculino', '8214','1995-05-13','87185445','978996553','afonso.padilha@gmail.com', '2020-07-22', '789781415'),

(null,'3','3','3','3','3','3','Christina Lins', 'Femenino', '4763','1997-01-02','95214578','977454145','thiago.ventura@gmail.com', '2021-06-08', '789804156');

create table item\_remessa(

id\_item\_remessa int auto\_increment,

valor int,

data\_retorno date,

codigo\_retorno int,

data\_vencimento\_item date,

numero\_agendamento int,

primary key (id\_item\_remessa)

);

insert into item\_remessa

values (null, '1530', '2021-10-05','1', '2021-10-05', '1'),

(null, '1200', '2021-10-05','2', '2021-10-05', '2'),

(null, '7290', '2021-10-05','3', '2021-10-05', '3');

create table remessa(

id\_remessa int auto\_increment,

#Chave estrangeira

taxa\_administrativa\_id int,

id\_item\_remessa int,

id\_emprestimo int,

#Atributos

usuario\_registro varchar(45),

nsa int,

registro\_envio int,

data\_envio date,

primary key (id\_remessa),

foreign key (taxa\_administrativa\_id) references taxa\_administrativa (id\_TA),

foreign key (id\_item\_remessa) references item\_remessa (id\_item\_remessa),

foreign key (id\_emprestimo) references emprestimo (id\_emprestimo)

);

insert into remessa

values (null, '1', '1','1', 'Guilherme de Sena', '06896', '5645' ,'2021-10-03'),

(null, '2', '2','2', 'Thiago Gomes', '06852', '6888' ,'2021-10-03'),

(null, '3', '3','3', 'Pedro Silva', '06863', '5278' ,'2021-10-03');

create table conta\_empresa(

id\_CE int auto\_increment,

#CE = Conta empresa

apelido\_conta varchar(45),

agencia int,

digito\_agencia int,

digito\_conta int,

nsa\_empresa int,

codigo\_agencia int,

conta\_empresa int,

operacao\_empresa varchar(45),

banco\_empresa varchar(45),

primary key (id\_CE)

);

insert into conta\_empresa

values (null, 'principal', '0001', '1', '2', '046876', '121' ,'569878676', 'Corrente', 'Nubank'),

(null, 'principal', '0003', '4', '8', '846847', '545' ,'787487495', 'PoupanÃ§a', 'Banco do Brasil'),

(null, 'principal', '0008', '9', '5', '987769', '544' ,'457489754', 'PoupanÃ§a', 'inter');

create table empresa(

id\_empresa int auto\_increment,

id\_conta\_empresa int,

nome varchar(45),

celular varchar(45),

email varchar(575),

bairro varchar(100),

cidade varchar(400),

cep int,

numero int,

rua varchar(45),

cnpj int,

primary key (id\_empresa),

foreign key (id\_conta\_empresa) references conta\_empresa (id\_CE)

);

insert into empresa

values (null, '1', 'infopampulha', '31987563275', 'infopampulha@gmail.com', 'SÃ£o Joaquim', 'Belo Horizonte', '324895660', '578', 'jose dias bicalho', '5646857'),

(null, '2', 'Uai de Minas', '31986489454', 'uaideminas@gmail.com', 'alipio de melo', 'Belo Horizonte', '648545421', '320', 'avenida dos engenheiros', '6545445'),

(null, '3', 'Araujo', '31979820212', 'araujo@gmail.com', 'PaquetÃ¡', 'Belo Horizonte', '55686565', '2500', 'avenida tancredo neves', '6856868');

#Parte 2

#QuestÃ£o 1

select

\*

from

agente;

#QuestÃ£o 2

select c.nome, e.numero\_contrato, e.situacao\_contrato, e.status\_contrato, e.data\_contrato\_emprestimo, e.valor\_liquido\_emprestimo from emprestimo as e

join cliente c

on e.id\_emprestimo = c.id\_cliente

where c.id\_cliente = '1'

order by nome;

#QuestÃ£o 3

select e.numero\_contrato, e.data\_contrato\_emprestimo, e.valor\_bruto\_emprestimo, e.valor\_liquido\_emprestimo, e.quantidade\_parcelas, c.id\_agente, e.situacao\_contrato, p.id\_parcela, p.data\_vencimento, p.valor,p.situacao

from emprestimo e

join cliente c

on e.id\_emprestimo = c.id\_cliente

join parcela p

on p.id\_parcela = e.parcela\_id\_parcela

order by nome;

#QuestÃ£o 4

create view lista\_taxa\_adm

as select taxa\_administrativa.id\_TA, cliente.id\_cliente, cliente.nome,

agente.id\_agente, agente.nome\_agente, taxa\_administrativa.situacao,

taxa\_administrativa.data\_inicio\_tax, taxa\_administrativa.tipo\_vencimento, taxa\_administrativa.valor,

conta\_cliente.id\_conta\_cliente

from taxa\_administrativa

inner join cliente on taxa\_administrativa.id\_TA = cliente.id\_taxa\_adimistrativa

inner join agente on cliente.id\_agente = agente.id\_agente

inner join conta\_cliente on cliente.id\_conta\_cliente = conta\_cliente.id\_conta\_cliente;

#QuestÃ£o 5

select \* from lista\_taxa\_adm

where id\_cliente = 1;

#QuestÃ£o 6

SET GLOBAL log\_bin\_trust\_function\_creators = 1;

delimiter $

create function retorn\_quantidade\_parcela(numero\_contrato int)

returns int

begin

declare resultado int;

select emprestimo.numero\_contrato, parcela.quantidade\_parcelas into resultado

from emprestimo

inner join parcela on emprestimo.parcela\_id\_parcela = parcela.id\_parcela;

return resultado;

end

$

#QuestÃ£o 7

delimiter $

create procedure listagem (in num\_nsa int)

begin

select remessa.nsa, cliente.id\_cliente, cliente.nome,

parcela.data\_vencimento, parcela.valor, parcela.valor\_apos\_vencimento,

conta\_cliente.agencia\_cliente, conta\_cliente.conta\_cliente

from remessa

inner join cliente on cliente.id\_cliente = remessa.id\_remessa

inner join parcela on parcela.id\_parcela = remessa.id\_remessa

inner join conta\_cliente on conta\_cliente.id\_conta\_cliente = remessa.id\_remessa

where remessa.nsa=num\_nsa;

end

$

call listagem(6896);

#QuestÃ£o 8

DELIMITER $$

create trigger atualizaNSA

before insert on remessa

for each row

begin

set NEW.nsa = (select nsa from remessa order by nsa desc limit 1) + 1;

end

$$

DELIMITER ;

**Referência Bibliográficas**

BARROS JUNIOR, A. A. (18 de agosto de 2021). *Arquitetura de SGBD*. Fonte: Arquitetura de SGBD: <http://files.antoniojr.webnode.com.br/200000121>- ea7d2eb774/Aula03-Arquiteturas%20para%20SGBDs.pdf

Niteroi, A. (18 de agosto de 2021). *UNIPLI*. Fonte: UNIPLI: https://sites.google.com/site/uniplitopicosavancbancodedados/aulas/aula-4--- gerenciamento-de-transacoes

Wikipedia. (29 de Setembro de 2019). *Wikipédia*. Fonte: Wikipédia: <https://pt.wikipedia.org/wiki/Transa%C3%A7%C3%A3o_(banco_de_dados>)