



UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE  
INSTITUTO METRÓPOLE DIGITAL  
TECNOLOGIA DA INFORMAÇÃO

JASON WILLYAN CASTRO DO NASCIMENTO

## **Banco de dados para o gerenciamento de uma biblioteca**

Natal  
2022

# Índice

<b>Introdução</b>	<b>3</b>
<b>Diagrama Entidade Relacionamento</b>	<b>4</b>
<b>Criando Tabelas</b>	<b>4</b>
<b>Povoando Tabelas</b>	<b>8</b>
<b>Criando Trigger</b>	<b>11</b>
<b>Criando Índices</b>	<b>18</b>

## **Introdução**

Foi desenvolvido uma base de dados para a gestão das informações de uma biblioteca. Com um estudo preliminar, apurou-se que a base de dados necessita de algumas entidades relevantes.

A entidade Livro é caracterizado por alguns atributos como título, o status que pode ser: "Disponível" ou "Indisponível", indicando se o livro está disponível para ser alugado ou não, o ISBN (International Standard Book Number), um sistema internacional de identificação de livros e softwares que utiliza números para classificá-los por título, autor, país, editora e edição e por fim a data de aquisição do livro.

Na base de dados, atribui-se a entidade Exemplar, o qual armazena dados dos livros como a quantidade de exemplares de um livro, o idioma, o ano, o gênero e a edição. Tem a entidade livro como chave estrangeira.

Sendo assim, imputa-se que cada livro tenha um autor e editora, assim como em uma biblioteca que necessita de funcionários, clientes e um fornecedor. Então, criamos uma entidade para cada uma delas.

Faz-se necessário que a biblioteca possa fazer alguns procedimentos para seu funcionamento, de modo que os clientes e/ou funcionários possam fazer empréstimos, por consequência, temos a entidade empréstimo. Se um empréstimo estiver com atraso, ou seja, um cliente ou funcionário não entregar o livro na data de devolução, será cobrado uma multa. Assim, originou-se a entidade multa, que armazena a data, motivo e valor da multa a ser cobrado.

Com a finalidade de que a biblioteca possa fazer requisições de novos livros ou aumentar seu estoque, a base de dados tem a entidade requisição no qual um funcionário pode fazer o pedido de compra de um livro, esta entidade armazena a data de requisição e o status que pode ser “Aberta”, que significa que o pedido de compra não foi finalizado e os livros não foram entregue ou “Fechada” que define que a compra foi finalizada e os livros foram entregue e já estão na biblioteca. A entidade requisição é chave estrangeira da entidade compra que armazena os dados da compra como a quantidade de livros, o preço e o título do livro.

Nesse projeto entidades que armazenam dados de clientes, funcionários e fornecedores serão consideradas como subtabelas, tais como: telefone, endereço, etc.

## **Diagrama Entidade Relacionamento**

Para dar início ao projeto, foi criado um DER (Diagrama Entidade Relacionamento) utilizando as regras de normalização e a ferramenta DB Designer.

Resultado do processo de modelagem



### Endereço

```
CREATE TABLE Endereco (  
  idEndereco SERIAL NOT NULL,  
  logradouro VARCHAR(20),  
  numero INTEGER,  
  complemento VARCHAR(20),  
  bairro VARCHAR(20),  
  cidade VARCHAR(20),  
  UF VARCHAR(2),  
  CEP VARCHAR(8),  
  PRIMARY KEY(idEndereco));
```

### Editora

```
CREATE TABLE Editora (  
  idEditora SERIAL NOT NULL,  
  nomeEditora VARCHAR(20),  
  PRIMARY KEY(idEditora));
```

### Autor

```
CREATE TABLE Autor (  
  idAutor SERIAL NOT NULL,  
  nomeAutor VARCHAR(25),  
  PRIMARY KEY(idAutor));
```

### Exemplar

```
CREATE TABLE Exemplar (  
  idExemplar SERIAL NOT NULL,  
  Livro_idLivro INTEGER NOT NULL,  
  quantidadeExemplar INTEGER NOT NULL,  
  idiomaExemplar VARCHAR(20),  
  anoExemplar INTEGER,  
  generoExemplar VARCHAR(100),  
  edicaoExemplar INTEGER,  
  PRIMARY KEY(idExemplar),  
  FOREIGN KEY(Livro_idLivro)  
    REFERENCES Livro(idLivro));
```

## Livro

```
CREATE TABLE Livro (  
  idLivro SERIAL NOT NULL,  
  Autor_idAutor INTEGER NOT NULL,  
  Editora_idEditora INTEGER NOT NULL,  
  tituloLivro VARCHAR(45),  
  statusLivro VARCHAR(20) CHECK (statusLivro = 'Disponível' or statusLivro = 'Indisponível'),  
  ISBN VARCHAR(13),  
  dataAquisicaoLivro DATE,  
  PRIMARY KEY(idLivro),  
  FOREIGN KEY(Editora_idEditora)  
    REFERENCES Editora(idEditora),  
  FOREIGN KEY(Autor_idAutor)  
    REFERENCES Autor(idAutor));
```

## Cliente

```
CREATE TABLE Cliente (  
  idCliente SERIAL NOT NULL,  
  Email_idEmail INTEGER NOT NULL,  
  Telefone_idTelefone INTEGER NOT NULL,  
  Endereco_idEndereco INTEGER NOT NULL,  
  nomeCliente VARCHAR(50),  
  cpfCliente VARCHAR(11),  
  dataNascimentoCliente DATE,  
  PRIMARY KEY(idCliente),  
  FOREIGN KEY(Endereco_idEndereco)  
    REFERENCES Endereco(idEndereco),  
  FOREIGN KEY(Telefone_idTelefone)  
    REFERENCES Telefone(idTelefone),  
  FOREIGN KEY(Email_idEmail)  
    REFERENCES Email(idEmail));
```

## Funcionário

```
CREATE TABLE Funcionario (  
  idFuncionario SERIAL NOT NULL,  
  Email_idEmail INTEGER NOT NULL,  
  Telefone_idTelefone INTEGER NOT NULL,  
  Endereco_idEndereco INTEGER NOT NULL,  
  nomeFuncionario VARCHAR(50),  
  cpfFuncionario VARCHAR(11),  
  dataNascimentoFuncionario DATE,  
  generoFuncionario VARCHAR(20),  
  numeroPisFuncionario VARCHAR(11),  
  cargoFuncionario VARCHAR(20),  
  dataAdmissaoFuncionario DATE,  
  salarioFuncionario REAL,  
  PRIMARY KEY(idFuncionario),  
  FOREIGN KEY(Endereco_idEndereco)  
    REFERENCES Endereco(idEndereco),  
  FOREIGN KEY(Telefone_idTelefone)  
    REFERENCES Telefone(idTelefone),  
  FOREIGN KEY(Email_idEmail)  
    REFERENCES Email(idEmail));
```

## Fornecedor

```
CREATE TABLE Fornecedor (  
  idFornecedor SERIAL NOT NULL,  
  Email_idEmail INTEGER NOT NULL,  
  Telefone_idTelefone INTEGER NOT NULL,  
  Endereco_idEndereco INTEGER NOT NULL,  
  razaoSocial VARCHAR(20),  
  cnpjFornecedor VARCHAR(14),  
  PRIMARY KEY(idFornecedor),  
  FOREIGN KEY(Endereco_idEndereco)  
    REFERENCES Endereco(idEndereco),  
  FOREIGN KEY(Telefone_idTelefone)  
    REFERENCES Telefone(idTelefone),  
  FOREIGN KEY(Email_idEmail)  
    REFERENCES Email(idEmail));
```

## Requisição

```
CREATE TABLE Requisicao (  
  idRequisicao SERIAL NOT NULL,  
  Funcionario_idFuncionario INTEGER NOT NULL,  
  dataRequisicao DATE CHECK (dataRequisicao >= CURRENT_DATE),  
  statusRequisicao VARCHAR(12) CHECK (statusRequisicao = 'Aberta' or statusRequisicao = 'Fechada'),  
  PRIMARY KEY(idRequisicao),  
  FOREIGN KEY(Funcionario_idFuncionario)  
    REFERENCES Funcionario(idFuncionario));
```

## Compra

```
CREATE TABLE Compra (  
  idCompra SERIAL NOT NULL,  
  Fornecedor_idFornecedor INTEGER NOT NULL,  
  Requisicao_idRequisicao INTEGER NOT NULL,  
  quantidadeCompra INTEGER,  
  precoCompra FLOAT,  
  tituloLivroCompra VARCHAR(45),  
  PRIMARY KEY(idCompra),  
  FOREIGN KEY(Requisicao_idRequisicao)  
    REFERENCES Requisicao(idRequisicao),  
  FOREIGN KEY(Fornecedor_idFornecedor)  
    REFERENCES Fornecedor(idFornecedor));
```

## Multa

```
CREATE TABLE Multa (  
  idMulta SERIAL NOT NULL,  
  Emprestimo_idEmprestimo INTEGER NOT NULL,  
  dataMulta DATE,  
  motivoMulta VARCHAR(30),  
  valorMulta REAL,  
  PRIMARY KEY(idMulta),  
  FOREIGN KEY(Emprestimo_idEmprestimo)  
    REFERENCES Emprestimo(idEmprestimo));
```



## Empréstimo

```
CREATE TABLE Emprestimo (  
  idEmprestimo SERIAL NOT NULL,  
  Livro_idLivro INTEGER NOT NULL,  
  Funcionario_idFuncionario INTEGER NOT NULL,  
  Cliente_idCliente INTEGER NOT NULL,  
  dataEmprestimo DATE,  
  dataDevolucao DATE,  
  PRIMARY KEY(idEmprestimo),  
  FOREIGN KEY(Cliente_idCliente)  
    REFERENCES Cliente(idCliente),  
  FOREIGN KEY(Funcionario_idFuncionario)  
    REFERENCES Funcionario(idFuncionario),  
  FOREIGN KEY(Livro_idLivro)  
    REFERENCES Livro(idLivro));
```

## Povoando Tabelas

Com as tabelas criadas, podemos preencher.

### Email

```
INSERT INTO Email(email) VALUES  
( 'samantha@hotmail.com'), -- id = 1  
( 'clover@hotmail.com'), -- id = 2  
( 'alexandra@hotmail.com'), -- id = 3  
( 'jonsnow@hotmail.com'), -- id = 4  
( 'tyrion@hotmail.com'), -- id = 5  
( 'jmormont@gmail.com'), -- id = 6  
( 'ciadolivro@gmail.com'); -- id = 7
```

### Telefone

```
INSERT INTO Telefone(telefone, DDD) VALUES  
( '923456789', '84'), -- id = 1  
( '934567892', '84'), -- id = 2  
( '924567893', '84'), -- id = 3  
( '915423698', '84'), -- id = 4  
( '910254893', '84'), -- id = 5  
( '999451389', '84'), -- id = 6  
( '997290012', '84'); -- id = 7
```

## Endereço

```
INSERT INTO Endereco(logradouro, numero, complemento, bairro, cidade, UF, CEP) VALUES
('Rua Jardim', 12, 'Casa', 'Rosa dos Ventos', 'Parnamirim', 'RN', '59145888'), -- id = 1
('Rua Sarinho', 501, 'Apartamento 12', 'Neópolis', 'Natal', 'RN', '05987222'), -- id = 2
('Rua Domingo', 25, 'Casa', 'Lagoa Nova', 'Natal', 'RN', '55987223'), -- id = 3
('Rua dos Monges', 366, 'Casa', 'Petropolis', 'Natal', 'RN', '45236784'), -- id = 4
('Rua Persa', 90, 'Apartamento 45', 'Bom Senhor', 'Macaíba', 'RN', '85194568'), -- id = 5
('Rua Lago Seco', 55, 'Casa B', 'Nelis', 'Touros', 'RN', '78456032'), -- id = 6
('Rua da saudade', 1593, 'Casa', 'Pitimbu', 'Natal', 'RN', '89726011');
```

## Editora

```
INSERT INTO Editora(nomeEditora) VALUES
('Principis'), -- id = 1
('Faro Editorial'), -- id = 2
('Casa dos Livros'), -- id = 3
('Rocco'), -- id = 4
('Editora Arqueiro');
```

## Autor

```
INSERT INTO Autor(nomeAutor) VALUES
('Sun Tzu'), -- id = 1
('Charlie Donlea'), -- id = 2
('Anne Frank'), -- id = 3
('Antoine de Saint-Exupéry'), -- id = 4
('Carolina Kepnes'), -- id = 5
('John Grisham');
```

## Exemplar

```
INSERT INTO Exemplar(Livro_idLivro, quantidadeExemplar, idiomaExemplar, anoExemplar, generoExemplar, edicaoExemplar) VALUES
(7, 2, 'Português', 2019, 'Tratado, Não ficção', 3), -- id = 1
(8, 6, 'Português', 2017, 'Suspense, Ficção, Mistério, Policial processual, Suspense psicológico', 1), -- id = 2
(9, 12, 'Português', 2020, 'Biografia, Autobiografia, Narrativa pessoal, Literatura judaica', 6), -- id = 3
(10, 4, 'Português', 2013, 'Literatura infantil, Fábula, Novela, Ficção especulativa', 18), -- id = 4
(11, 7, 'Português', 2018, 'Ficção, Suspense', 1), -- id = 5
(12, 2, 'Português', 2020, 'Ação. Thriller político, Drama, Drama judicial, Suspense, Crime, Mistério, Adaptação, Aventura', 1); -- id = 6
```

## Livro

```
INSERT INTO Livro(Autor_idAutor, Editora_idEditora, tituloLivro, statusLivro, ISBN, dataAquisicaoLivro) VALUES
(1, 1, 'A arte da guerra', 'Disponível', '9788594318596', '12/08/2020'), -- id = 7
(2, 2, 'A garota do lago', 'Disponível', '9788562409882', '05/01/2019'), -- id = 8
(3, 1, 'O diário de Anne Frank', 'Indisponível', '9786550970406', '13/09/2021'), -- id = 9
(4, 3, 'O Pequeno Príncipe', 'Disponível', '9788522005239', '02/12/2013'), -- id = 10
(5, 4, 'Você', 'Disponível', '9788532530943', '26/08/2019'), -- id = 11
(6, 5, 'O dossiê pelicano', 'Indisponível', '9788530601232', '09/09/2021');
```

## Cliente

```
INSERT INTO Cliente(Email_idEmail, Telefone_idTelefone, Endereco_idEndereco, nomeCliente, cpfCliente, dataNascimentoCliente) VALUES
(4, 4, 4, 'Jon Snow', '15151212234', '15/09/1990'), -- id = 1
(5, 5, 5, 'Tyrion Lannister', '15498745638', '29/03/1985'), -- id = 2
(6, 6, 6, 'Jorah Mormont', '54896120078', '01/012/1979');
```

## Funcionário

```
INSERT INTO Funcionario(Email_idEmail, Telefone_idTelefone, Endereco_idEndereco, nomeFuncionario, cpffuncionario, dataNascimentoFuncionario,
generoFuncionario, numeroPisFuncionario, cargoFuncionario, dataAdmissaoFuncionario, salarioFuncionario) VALUES
(1, 1, 1, 'Samantha', '01234567899', '20/09/2005', 'Cisgênero', '01234567891', 'Bibliotecária', '02/05/2022', 2500.00), -- id = 1
(2, 2, 2, 'Clover', '12345678990', '01/07/2005', 'Cisgênero', '12345678910', 'Gerente', '15/010/2021', 6000.00), -- id = 2
(3, 3, 3, 'Alexandra', '23456789901', '20/09/2000', 'Cisgênero', '23456789101', 'Recepcionista', '15/03/2022', 1800.50); -- id = 3
```

## Fornecedor

```
INSERT INTO Fornecedor(Email_idEmail, Telefone_idTelefone, Endereco_idEndereco, razaoSocial, cnnpjFornecedor) VALUES
(7, 7, 7, 'CIA do Livro', '19826388162092'); -- id = 1
```

## Requisição

```
INSERT INTO Requisicao(Funcionario_idFuncionario, dataRequisicao, statusRequisicao) VALUES
(2, '05/07/2022', 'Aberta'), -- id = 2
(3, '10/08/2022', 'Aberta');
```

## Compra

```
INSERT INTO Compra(Fornecedor_idFornecedor, Requisicao_idRequisicao, quantidadeCompra, precoCompra, tituloLivroCompra) VALUES
(1, 2, 23, 12.50, '1984'), -- id = 1
(1, 3, 14, 25.99, 'Orgulho e preconceito');
```

## Empréstimo

```
1 INSERT INTO Emprestimo(Livro_idLivro, Funcionario_idFuncionario, Cliente_idCliente, dataEmprestimo, dataDevolucao) VALUES
2 (7, 1, 1, '03/07/2022', '25/12/2022'), -- id = 7
3 (11, 1, 1, '03/07/2022', '12/09/2022'), -- id = 8
4 (10, 1, 1, '03/07/2022', '05/08/2022');
```

# Criando Trigger

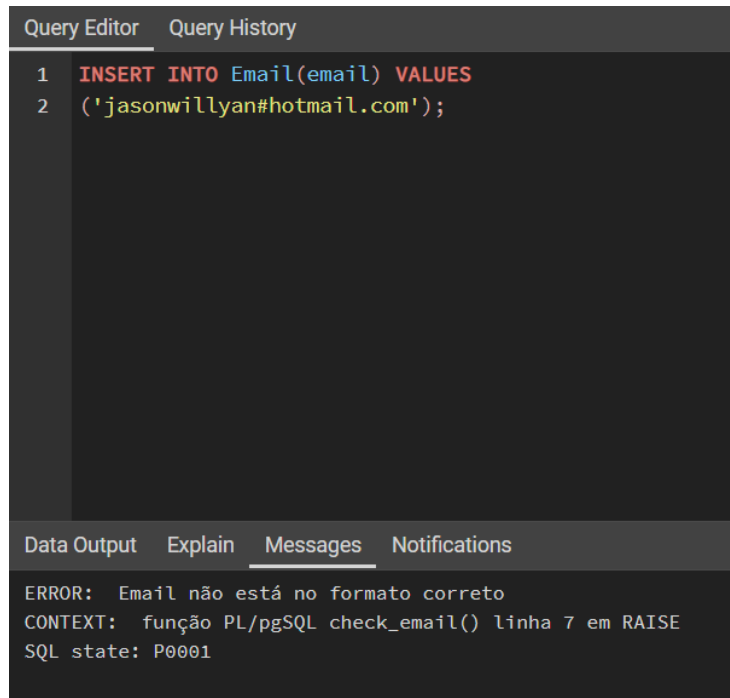
Com as tabelas preenchidas, foram criadas triggers e functions.

## Verifica se o email está no formato correto

```
CREATE FUNCTION check_email() RETURNS
trigger AS $check_email$
BEGIN
IF NEW.email LIKE '%@%' THEN
    RETURN NEW;
ELSE
    RAISE EXCEPTION 'Email não está no formato correto';
    RETURN NEW;
END IF;
END;
$check_email$ LANGUAGE plpgsql;

CREATE TRIGGER check_email BEFORE INSERT OR UPDATE
ON Email
FOR EACH ROW
WHEN (pg_trigger_depth() = 0)
EXECUTE PROCEDURE check_email();
```

## Exemplo

The image shows a screenshot of a database query editor interface. At the top, there are two tabs: 'Query Editor' and 'Query History'. The 'Query Editor' tab is active, showing two lines of SQL code: '1 INSERT INTO Email(email) VALUES' and '2 ('jasonwillyan@hotmail.com');'. Below the query editor, there are four tabs: 'Data Output', 'Explain', 'Messages', and 'Notifications'. The 'Messages' tab is active, displaying an error message: 'ERROR: Email não está no formato correto', followed by 'CONTEXT: função PL/pgSQL check\_email() linha 7 em RAISE' and 'SQL state: P0001'.

```
Query Editor  Query History

1  INSERT INTO Email(email) VALUES
2  ('jasonwillyan@hotmail.com');

Data Output  Explain  Messages  Notifications

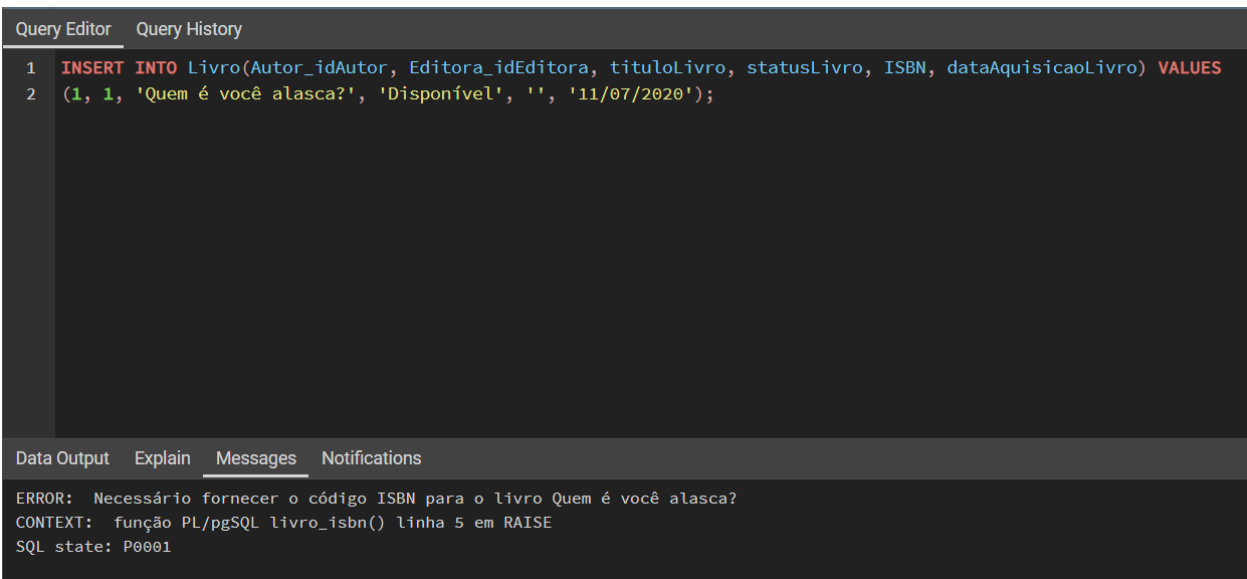
ERROR:  Email não está no formato correto
CONTEXT:  função PL/pgSQL check_email() linha 7 em RAISE
SQL state: P0001
```

## Verifica se o código ISBN é fornecido

```
CREATE FUNCTION livro_isbn() RETURNS
trigger AS $livro_isbn$
BEGIN
IF NEW.ISBN IS NULL OR NEW.ISBN = '' THEN
    RAISE EXCEPTION 'Necessário fornecer o código ISBN para o livro %', NEW.tituloLivro;
END IF;
RETURN NEW;
END;
$livro_isbn$ LANGUAGE plpgsql;

CREATE TRIGGER livro_isbn BEFORE INSERT
ON Livro
FOR EACH ROW
EXECUTE PROCEDURE livro_isbn();
```

## Exemplo



The screenshot shows a database query editor with two tabs: "Query Editor" and "Query History". The "Query Editor" tab is active and contains the following SQL statement:

```
1 INSERT INTO Livro(Autor_idAutor, Editora_idEditora, tituloLivro, statusLivro, ISBN, dataAquisicaoLivro) VALUES
2 (1, 1, 'Quem é você alasca?', 'Disponível', '', '11/07/2020');
```

Below the query editor, there are four tabs: "Data Output", "Explain", "Messages", and "Notifications". The "Messages" tab is active and displays an error message:








```
ERROR: Necessário fornecer o código ISBN para o livro Quem é você alasca?
CONTEXT: função PL/pgSQL livro_isbn() linha 5 em RAISE
SQL state: P0001
```

## Atualiza a quantidade do exemplar









```
CREATE FUNCTION quantidade_update() RETURNS
trigger AS $quantidade_update$
BEGIN
    UPDATE Exemplar
    SET quantidadeExemplar = quantidadeExemplar - 1
    WHERE Livro_idLivro = NEW.Livro_idLivro;
RETURN NULL;
END;
$quantidade_update$ LANGUAGE plpgsql;

CREATE TRIGGER quantidade_update AFTER INSERT
ON Emprestimo
FOR EACH ROW
EXECUTE PROCEDURE quantidade_update();
```

## Exemplo - Antes do empréstimo

Data Output		Explain	Messages	Notifications			
 idexemplar [PK] integer	 livro_idlivro integer	 quantidadeexemplar integer	 idiomaexemplar character varying (2)	 anoexemplar integer	 generoexemplar character varying (100)	 edicaoexemplar integer	
1	1	7	2 Português	2019	Tratado, Não ficção	3	
2	2	8	6 Português	2017	Suspense, Ficção, Mistério, Policial processual, Suspense psicológico	1	
3	3	9	12 Português	2020	Biografia, Autobiografia, Narrativa pessoal, Literatura judaica	6	
4	4	10	4 Português	2013	Literatura infantil, Fábula, Novela, Ficção especulativa	18	
5	5	11	7 Português	2018	Ficção, Suspense	1	
6	6	12	2 Português	2020	Ação. Thriller político, Drama, Drama judicial, Suspense, Crime, Mistério, Adaptação, Aventura	1	

## Exemplo - Depois do empréstimo

Data Output		Explain	Messages	Notifications			
  idexemplar [PK] integer	 livro_idlivro integer	 quantidadeexemplar integer	 idiomaexemplar character varying (20)	 anoexemplar integer	 generoexemplar character varying (100)	 edicaoexemplar integer	
1	1	7	1 Português	2019	Tratado, Não ficção	3	
2	2	8	6 Português	2017	Suspense, Ficção, Mistério, Policial processual, Suspense psicológico	1	
3	3	9	12 Português	2020	Biografia, Autobiografia, Narrativa pessoal, Literatura judaica	6	
4	4	10	3 Português	2013	Literatura infantil, Fábula, Novela, Ficção especulativa	18	
5	5	11	6 Português	2018	Ficção, Suspense	1	
6	6	12	2 Português	2020	Ação. Thriller político, Drama, Drama judicial, Suspense, Crime, Mistério, Adaptação, Aventu...	1	

## Cria uma trigger de auditoria para empréstimo

### Script para criar tabela de auditoria

```
CREATE TABLE funcionario_log(
    idFuncionarioLog SERIAL NOT NULL,
    operacaoLog VARCHAR(6) NOT NULL,
    dataLog TIMESTAMP NOT NULL,
    emprestimoId INTEGER,
    funcionarioId INTEGER NOT NULL,
    PRIMARY KEY(idFuncionarioLog),
    FOREIGN KEY(emprestimoId)
        REFERENCES Empréstimo(idEmpréstimo),
    FOREIGN KEY(funcionarioId)
        REFERENCES Funcionario(idFuncionario)
);
```

## Insert e Update

```
CREATE FUNCTION funcionario_auditoria() RETURNS
trigger AS $funcionario_auditoria$
BEGIN
    INSERT INTO funcionario_log (operaçãoLog, dataLog, emprestimoId, funcionarioId) SELECT
    TG_OP, NOW(), a.idEmprestimo, a.Funcionario_idFuncionario
    FROM Emprestimo a WHERE a.idEmprestimo = NEW.idEmprestimo;
RETURN NULL;
END;
$funcionario_auditoria$ LANGUAGE plpgsql;

CREATE TRIGGER funcionario_auditoria AFTER INSERT OR UPDATE
ON Emprestimo
FOR EACH ROW
EXECUTE PROCEDURE funcionario_auditoria();
```

## Delete

```
CREATE OR REPLACE FUNCTION funcionario_auditoria_delete() RETURNS
trigger AS $funcionario_auditoria_delete$
BEGIN
    INSERT INTO funcionario_log (operaçãoLog, dataLog, funcionarioId) SELECT
    TG_OP, NOW(), a.Funcionario_idFuncionario
    FROM Emprestimo a WHERE a.idEmprestimo = OLD.idEmprestimo;
RETURN OLD;
END;
$funcionario_auditoria_delete$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER funcionario_auditoria_delete BEFORE DELETE
ON Emprestimo
FOR EACH ROW
EXECUTE PROCEDURE funcionario_auditoria_delete();
```

## Exemplo

Data Output							Explain	Messages	Notifications
	idfuncionario_log [PK] integer		operação_log character varying (6)		data_log timestamp without time zone		emprestimoid integer		funcionarioid integer
1		1	INSERT		2022-07-12 00:27:59.373469		28		3
2		2	DELETE		2022-07-12 00:28:01.754875		[null]		3
3		3	DELETE		2022-07-12 00:31:14.593173		[null]		3
4		4	DELETE		2022-07-12 00:31:19.08523		[null]		3

## Criando Views

Retorna os funcionários que tem salário maior que 2350.00

```
CREATE VIEW salario_funcionario AS
SELECT FN.nomeFuncionario AS Nome,
       FN.cargoFuncionario AS Cargo,
       FN.salarioFuncionario AS Salario,
       FN.salarioFuncionario * 12 AS Salario_anual
FROM Funcionario FN
WHERE salarioFuncionario > 2350;
ORDER BY Salario_anual DESC;

SELECT * FROM salario_funcionario;
```

### Exemplo

	Data Output	Explain	Messages	Notifications
	nome character varying (50)	cargo character varying (20)	salario real	salario_anual double precision
1	Clover	Gerente	6000	72000
2	Samantha	Bibliotecária	2500	30000

Retorna os livros que tem gênero suspense

```
CREATE MATERIALIZED VIEW livro_genero_suspense AS
SELECT LV.tituloLivro AS Titulo,
       LV.statusLivro AS Status,
       EX.generoExemplar AS Genero
FROM Livro LV
INNER JOIN Exemplar EX
  ON LV.idLivro = EX.Livro_idLivro
WHERE EX.generoExemplar LIKE '%Suspense%'
WITH NO DATA;

REFRESH MATERIALIZED VIEW livro_genero_suspense;

SELECT * FROM livro_genero_suspense;
```

### Exemplo

	titulo character varying (45)	status character varying (20)	genero character varying (100)
1	A garota do lago	Disponível	Suspense, Ficção, Mistério, Policial processual, Suspense psicológico
2	O dossiê pelicano	Indisponível	Ação. Thriller político, Drama, Drama judicial, Suspense, Crime, Mistério, Adaptação, Aventura
3	Você	Disponível	Ficção, Suspense



### Retorna os dados das compras

```
CREATE VIEW compras AS
SELECT FN.nomeFuncionario AS Funcionario, CP.tituloLivroCompra AS Titulo,
       CP.quantidadeCompra AS Quantidade, CP.precoCompra AS Preço,
       CP.quantidadeCompra * CP.precoCompra AS Total
FROM Compra CP
INNER JOIN Requisicao RQ
    ON RQ.idRequisicao = CP.Requisicao_idRequisicao
INNER JOIN Funcionario FN
    ON RQ.Funcionario_idFuncionario = FN.idFuncionario
ORDER BY Total DESC;

SELECT * FROM compras;
```

### Exemplo

	funcionario character varying (50)	titulo character varying (45)	quantidade integer	preço double precision	total double precision
1	Alexandra	Orgulho e preconceito	14	25.99	363.85999999999996
2	Clover	1984	23	12.5	287.5

### Retorna os clientes que tem emprestimo

```
CREATE MATERIALIZED VIEW emprestimos AS
SELECT CT.nomeCliente AS Nome,
       CT.cpfCliente AS CPF,
       LV.tituloLivro AS Livro,
       EP.dataEmprestimo AS Data_Emprestimo,
       EP.dataDevolucao AS Data_Devolução
FROM Cliente CT
INNER JOIN Emprestimo EP
    ON CT.idCliente = EP.Cliente_idCliente
INNER JOIN Livro LV
    ON LV.idLivro = EP.Livro_idLivro
WITH DATA;

SELECT * FROM emprestimos;
```

### Exemplo

	nome character varying (50)	cpf character varying (11)	livro character varying (45)	data_emprestimo date	data_devolução date
1	Jon Snow	15151212234	A arte da guerra	2022-07-03	2022-12-25
2	Jon Snow	15151212234	Você	2022-07-03	2022-09-12

## **Criando Índices**

Não foram usados índices no projeto.