RELATÓRIO DISCENTE DE ACOMPANHAMENTO

Campus Polo Cavalhada – Porto Alegre

DESENVOLVIMENTO FULL STACK

Missão Prática | Nível 2 | Mundo 3

RPG0015 – Vamos manter as informações!

Turma: 9003

Semestre Letivo: 3º - 2024.1

GUILHERME BERNARDES BASTOS

Repositório: https://github.com/guilhermebernardes96/Mundo3-Nivel2

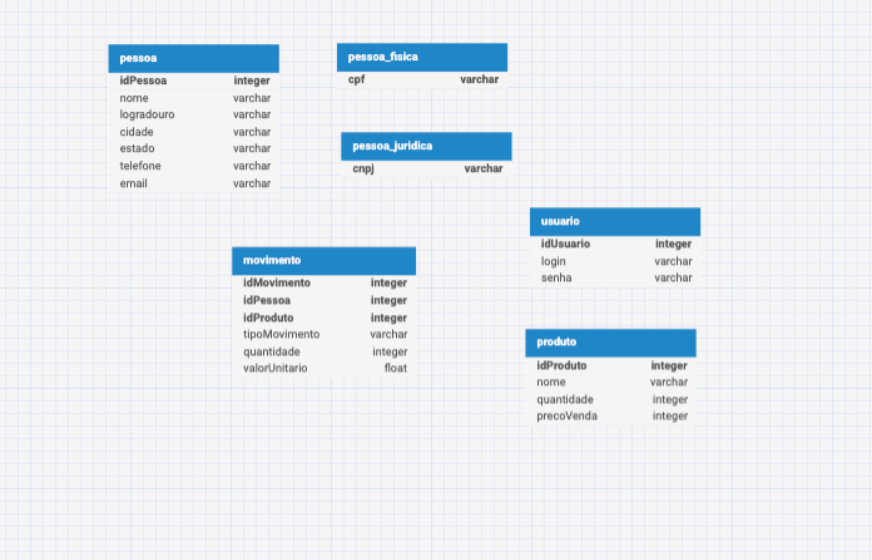
CRIANDO O BANCO DE DADOS

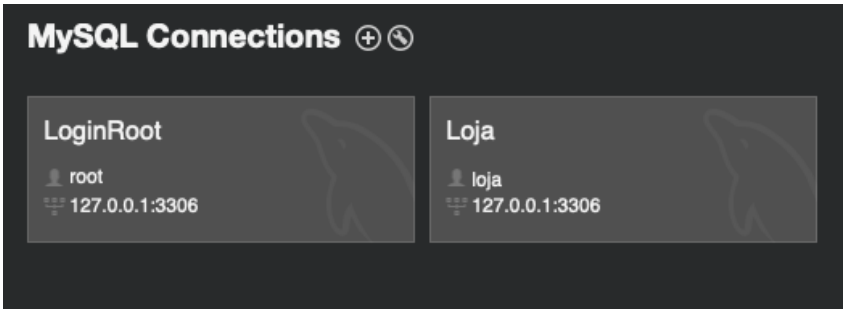
Objetivo da Prática

Identificar requisitos de um sistema e transformar no modelo mais adequado utilizando ferramentas de modelagem para base de dados. Explorar a sintaxe SQL na criação, na consulta e manipulação de dados.

1º Procedimento

CÓDIGOS



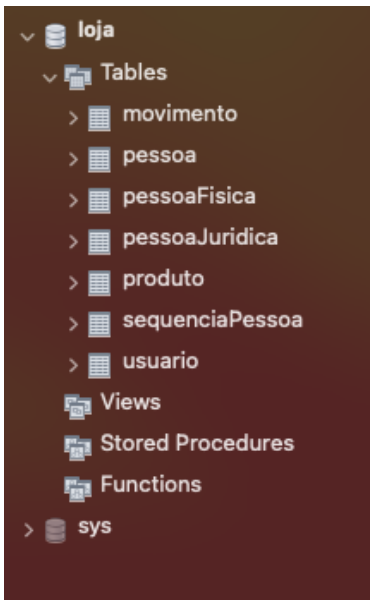



```sql
create table pessoa (
    idPessoa integer not null primary key,
    nome varchar (255) not null,
    logradouro varchar (255) not null,
    cidade varchar (255) not null,
    estado char (2) not null,
    telefone varchar (11) not null,
    email varchar (255) not null
);

create table pessoaFisica (
    idPessoa integer not null primary key,
    cpf varchar(11) not null,
    constraint fk_pessoaFisica_pessoa foreign key (idPessoa) references pessoa (idPessoa)
);

create table pessoaJuridica (
    idPessoa integer not null primary key,
    cnpj varchar(14) not null,
    constraint fk_pessoaJuridica_pessoa foreign key (idPessoa) references pessoa (idPessoa)
);

```

```sql
create table usuario (
    idUsuario integer not null primary key,
    login varchar (50) not null,
    senha varchar (50) not null
);

create table produto (
    idProduto integer not null primary key,
    nome varchar (255) not null,
    quantidade integer not null,
    precoVenda numeric
);

create table movimento (
    idMovimento integer not null primary key,
    idUsuario integer not null,
    idPessoa integer not null,
    idProduto integer not null,
    quantidade integer not null,
    tipoMovimento char (1) not null,
    valorUnitario float not null,
    constraint fk_movimento_usuario foreign key (idUsuario) references usuario (idUsuario),
    constraint fk_movimento_pessoa foreign key (idPessoa) references pessoa (idPessoa),
    constraint fk_movimento_produto foreign key (idProduto) references produto (idProduto)
);

```

```sql
create table sequenciaPessoa (
    id int auto_increment primary key
);

create index pessoaFisica_fkindex1 on pessoaFisica (idPessoa);
create index pessoaJuridica_fkindex2 on pessoaJuridica (idPessoa);
```

RESULTADO



2º Procedimento

CÓDIGOS

```sql
insert into usuario
    (idUsuario, login, senha)
values
    (1, 'op1', 'op1'),
    (2, 'op2', 'op2')
;
```

```sql
insert into produto
    (idProduto, nome, quantidade, precoVenda)
values
    (1, 'Banana', 100, 5.00),
    (2, 'Laranja', 500, 2.00),
    (3, 'Manga', 800, 4.00)
;
```

```sql
insert into pessoa
    (idPessoa, nome, logradouro, cidade, estado, telefone, email)
values
    (7, 'Joao', 'Rua 12, casa 3', 'Riacho do Sul', 'PA', '1111-1111', 'joao@riacho.com'),
    (1, "Ferdinand Vaughan","Malesuada Rd.","Divinópolis","RS","1382-1754","at.sem@aol.com"),
    (2, "Jermaine Faulkner","7971 Nulla Street","Lima","SC","4210-9413","ut.molestie@outlook.net"),
    (3, "Avram Banks","3066 Mi Rd.","Ciudad Santa Catarina","PR","1423-2371","nunc@protonmail.ca"),
    (4, "Allen Flowers","5075 Malesuada Road","Uman","GO","3771-8832","fringilla.purus@yahoo.couk"),
    (5, "India Spencer","4332 Eget St.","Governador Valadares","RN","5253-5530","nisi.mauris@yahoo.ca"),
    (6, "Tanek Roy","3862 Ut, Road","Christchurch","RJ","3242-6301","eget@hotmail.edu"),
    (8, "Deacon Pickett","7634 Curabitur Ave","Alto del Carmen","RS","7511-7288","donec.egestas@protonmail.com"),
    (9, "Meredith Gaines","P.O. Box 867, 5786 Elit. Ave","Melton","SC","6904-7307","orci@yahoo.org"),
    (10, "Damian Sweet","Ap #657-9669 Ut, St.","Nizhny","PR","5981-5724","ac@hotmail.net"),
    (11, "Latifah Conway","Ap #628-8354 Auctor Av.","Sokoto","GO","1224-6424","consectetuer.adipiscing@icloud.com"),
    (12, "Astra Serrano","Ap #210-4769 Vivamus Rd.","Lambayeque","RN","9537-8153","tincidunt.vehicula.risus@google.couk
    (15, 'JJC', 'Rua 11, Centro', 'Riacho do Norte', 'PA', '1212-1212', 'jjc@riacho.com')
;
```

```sql
insert into pessoaFisica
    (idPessoa, cpf)
value
    (7, '11111111111'),
    (1, "00000000000"),
    (3, "99999999999"),
    (5, "77777777777"),
    (8, "66666666666"),
    (10, "55555555555"),
    (12, "44444444444")
;
```

```
45    insert into pessoaJuridica
46        (idPessoa, cnpj)
47    value
48        (15, '22222222222222'),
49        (2, "22222222222222"),
50        (4, "44444444444444"),
51        (6, "66666666666666"),
52        (9, "88888888888888"),
53        (11, "99999999999999")
54    ;
```

```
56    insert into movimento
57        (idMovimento, idUsuario, idPessoa,
58        idProduto, quantidade, tipoMovimento,
59        valorUnitario)
60    value
61        (1, 1, 7, 1, 20, 'S', 4.00),
62        (4, 1, 7, 2, 15, 'S', 2.00),
63        (5, 2, 7, 2, 10, 'S', 3.00),
64        (7, 1, 15, 2, 15, 'E', 5.00),
65        (8, 1, 15, 3, 20, 'E', 4.00)
66    ;
```

RESULTADO

```sql
select
    pessoa.idPessoa,
    pessoa.nome,
    pessoa.logradouro,
    pessoa.cidade,
    pessoa.estado,
    pessoa.telefone,
    pessoa.email,
    pessoaFisica.cpf
from
    pessoa
join
    pessoaFisica on pessoa.idPessoa = pessoaFisica.idPessoa
;
```

100% 2:14

Result Grid Filter Rows: Search Export:

idPessoa	nome	logradouro	cidade	estado	telefone	email	cpf
1	Ferdinand Vaughan	Malesuada Rd.	Divinópolis	RS	1382-1754	at.sem@aol.com	00000000000
3	Avram Banks	3066 Mi Rd.	Ciudad Santa Catarina	PR	1423-2371	nunc@protonmail.ca	99999999999
5	India Spencer	4332 Eget St.	Governador Valadares	RN	5253-5530	nisi.mauris@yahoo.ca	77777777777
7	Joao	Rua 12, casa 3	Riacho do Sul	PA	1111-1111	joao@riacho.com	11111111111
8	Deacon Pickett	7634 Curabitur Ave	Alto del Carmen	RS	7511-7288	donec.egestas@protonmail.com	66666666666
10	Damian Sweet	Ap #657-9669 Ut, St.	Nizhny	PR	5981-5724	ac@hotmail.net	55555555555
12	Astra Serrano	Ap #210-4769 Vivamus Rd.	Lambayeque	RN	9537-8153	tincidunt.vehicula.risus@google.couk	44444444444

Result 4

```sql
select
    pessoa.idPessoa,
    pessoa.nome,
    pessoa.logradouro,
    pessoa.cidade,
    pessoa.estado,
    pessoa.telefone,
    pessoa.email,
    pessoaJuridica.cnpj
from
    pessoa
join
    pessoaJuridica ON pessoa.idPessoa = pessoaJuridica.idPessoa
;
```

100% 1:15

Result Grid Filter Rows: Search Export:

idPessoa	nome	logradouro	cidade	estado	telefone	email	cnpj
2	Jermaine Faulkner	7971 Nulla Street	Lima	SC	4210-9413	ut.molestie@outlook.net	22222222222222
4	Allen Flowers	5075 Malesuada Road	Uman	GO	3771-8832	fringilla.purus@yahoo.couk	44444444444444
6	Tanek Roy	3862 Ut, Road	Christchurch	RJ	3242-6301	eget@hotmail.edu	66666666666666
9	Meredith Gaines	P.O. Box 867, 5786 Elit. Ave	Melton	SC	6904-7307	orci@yahoo.org	88888888888888
11	Latifah Conway	Ap #628-8354 Auctor Av.	Sokoto	GO	1224-6424	consectetuer.adipiscing@icloud.com	99999999999999
15	JJC	Rua 11, Centro	Riacho do Norte	PA	1212-1212	jjc@riacho.com	2222222222222

```sql
select
    movimento.idMovimento,
    produto.nome as produto,
    pessoa.nome as fornecedor,
    movimento.quantidade,
    movimento.valorUnitario,
    (movimento.quantidade * movimento.valorUnitario) as valorTotal
from
    movimento
join
    produto on movimento.idProduto = produto.idProduto
join
    pessoa on movimento.idPessoa = pessoa.idPessoa
where
    movimento.tipoMovimento = 'E'
;
```

100% 2:47

Result Grid Filter Rows: Search Export:

idMovimento	produto	fornecedor	quantidade	valorUnitario	valorTotal
7	Laranja	JJC	15	5	75
8	Manga	JJC	20	4	80

```sql
select
    movimento.idMovimento,
    produto.nome as produto,
    pessoa.nome as fornecedor,
    movimento.quantidade,
    movimento.valorUnitario,
    (movimento.quantidade * movimento.valorUnitario) as valorTotal
from
    movimento
join
    produto on movimento.idProduto = produto.idProduto
join
    pessoa on movimento.idPessoa = pessoa.idPessoa
where
    movimento.tipoMovimento = 'S'
;
```

100% 2:64

Result Grid Filter Rows: Search Export:

idMovimento	produto	fornecedor	quantidade	valorUnitario	valorTotal
1	Banana	Joao	20	4	80
4	Laranja	Joao	15	2	30
5	Laranja	Joao	10	3	30

```sql
select
    produto.nome as produto,
    sum(movimento.quantidade * movimento.valorUnitario) as valorTotalEntradas
from
    movimento
join
    produto on movimento.idProduto = produto.idProduto
where
    movimento.tipoMovimento = 'E'
group by
    produto.nome
;
```

100% 2:77

Result Grid Filter Rows: Search Export:

produto	valorTotalEntradas
Laranja	75
Manga	80

```sql
select
    produto.nome as produto,
    sum(movimento.quantidade * movimento.valorUnitario) as valorTotalSaidas
from
    movimento
join
    produto on movimento.idProduto = produto.idProduto
where
    movimento.tipoMovimento = 'S'
group by
    produto.nome
;
```

100% 2:90

Result Grid Filter Rows: Search Export:

produto	valorTotalSaidas
Banana	80
Laranja	60

```sql
-- OPERADOR SEM MOVIMENTACAO DE ENTRADA
select
    usuario.idUsuario,
    usuario.login
from
    usuario
left join
    movimento on usuario.idUsuario = movimento.idUsuario and movimento.tipoMovimento = 'E'
where
    movimento.idMovimento is null
;
```

100% 2:11

Result Grid Filter Rows: Search Export:

idUsuario	login
2	op2

```sql
select
    usuario.idUsuario,
    usuario.login,
    sum(movimento.quantidade * movimento.valorUnitario) as valorTotalEntrada
from
    usuario
join
    movimento on usuario.idUsuario = movimento.idUsuario and movimento.tipoMovimento = 'E'
group by
    usuario.idUsuario, usuario.login
;
```

100% 6:12

Result Grid Filter Rows: Search Export:

idUsuario	login	valorTotalEntra...
1	op1	155

```sql
select
    usuario.idUsuario,
    usuario.login,
    sum(movimento.quantidade * movimento.valorUnitario) as valorTotalSaida
from
    usuario
join
    movimento ON usuario.idUsuario = movimento.idUsuario and movimento.tipoMovimento = 'S'
group by
    usuario.idUsuario, usuario.login
;
```

100% 1:26

Result Grid Filter Rows: Search Export:

idUsuario	login	valorTotalSaida
1	op1	110
2	op2	30

```sql
select
    produto.idProduto,
    produto.nome,
    sum(movimento.quantidade * movimento.valorUnitario) / sum(movimento.quantidade) as valorMedioVenda
from
    produto
join
    movimento ON produto.idProduto = movimento.idProduto
group by
    produto.idProduto, produto.nome
;
```

100% 1:39

Result Grid Filter Rows: Search Export:

idProduto	nome	valorMedioVenda
1	Banana	4
2	Laranja	3.375
3	Manga	4

ANÁLISE E CONCLUSÃO

Procedimento 1

a. Como são implementadas as diferentes cardinalidades, basicamente 1X1, 1XN ou NxN, em um banco de dados relacional?

Basicamente são implementadas por meio do uso de uma chave estrangeira (foreing kay) e das relações entre as tabelas. O 1X1 é quando está associado a uma linha de outra tabela. O 1XN está relacionado com várias linhas de outra tabela. Já o NxN fica associado em linhas de tabelas além também de outras tabelas.

b. Que tipo de relacionamento deve ser utilizado para representar o uso de herança em banco de dados relacionais?

O relacionamento que deve se usar é o 1X1.

c. Como o SQL Server Management Studio permite a melhoria da produtividade nas tarefas relacionadas ao gerenciamento do banco de dados?

Na verdade, devido ao meu notebook ser um Mac eu não pude utilizar o esse programa, então não sei no que ele poderia melhorar na produtividade das tarefas. Porém, utilizei o MySQL Workbench, que supriu bastante toda minha necessidade apesar da faculdade não me auxiliar e ajudar no que eu poderia fazer.

Procedimento 2

a. Quais as diferenças no uso de sequence e identity?

Os dois são mecanismos para gerar valores automáticos, mas com algumas diferenças. O sequence gera números geralmente usados para preencher colunas de chave primaria. O identity permite que uma coluna da tabela seja automaticamente preenchida com valores ao inserir novas linhas. Cada um pode ser usando dependendo do sistema de gerenciamento do banco de dados, pois cada um funciona um pouco diferente em cada um.

b. Qual a importância das chaves estrangeiras para consistência do banco?

É importante porque com ela conseguimos relacionar uma linha de outra tabela diretamente com a outra. Assim garantindo a integridade do sistema e registros relacionais em ambas tabelas.

c. Quais operadores do SQL pertencem à álgebra relacional e quais são definidos no cálculo relacional?

Em álgebra relacional são eles: Seleção, Projeção, União, Interseção, Diferença, Produto Cartesiano e Junção.

Já os definidos no cálculo relacional são: Seleção, Projeção, Junção, União e Interseção.

d. Como é feito o agrupamento em consultas, e qual requisito é obrigatório?

É feito através do GROUP BY e o requisito obrigatório é de estar tanto no SELECT quando no GROUP BY.