Relatório

Tabelas

A base de dados do hotel é constituída pelas seguintes tabelas

```
create table Pessoa(
id varchar(10) not null primary key(id),
nome varchar(50) not null,
sexo char(1) check(sexo in ('M', 'F')), --só pode digitar M ou F
morada varchar(100) not null,
codPostal varchar(8) not null,
localidade varchar(20) not null,
telefone varchar(9) not null unique(telefone) --um numero só pode estar associado a uma pessoa
email varchar(100) not null unique(email), --um email só pode estar associado uma pessoa
dataNasc date not null);

create table Hospede(
id varchar(10) not null references Pessoa(id),
codHospede int not null identity(1,1) primary key(codHospede),
descontos decimal default 20); --fica com um valor predefinido 20

create table Hotel(
id int not null references Hospede(codHospede),
classificacao decimal(2,1) not null);

create table Departamento(nome varchar(20) not null primary key(nome));

create table Funcionarios(
id varchar(10) not null references Pessoa(id),
codFun int not null identity(1,1) primary key(codFun),
depart varchar(20) not null references Departamento(nome),
cargo varchar(20) not null,
dataAdmissao date not null,
salario money not null);

create table TipoApartamento(
tipoApart varchar(20) not null primary key(tipoApart));

create table Apartamento(
codApart int not null identity(1,1) primary key(codApart),
localizacao char(2) not null unique(localizacao),
disponivel bit not null,
tipoApart varchar(20) not null references TipoApartamento(tipoApart),
quantQuartos smallint not null,
precoDia money not null,
limpo bit not null);

create table Reserva(
codReserva int not null primary key(codReserva) identity(1,1),
hospede int not null references Hospede(codHospede),
Apart int not null references Apartamento(codApart),
dataCheck_int datetime not null,
datacheck_out datetime not null,
statusReserva char(1) not null check (statusReserva in('R', 'O', 'C')),
transferidos bit default 0,
obsHospedagem varchar(100),
numerosHospedes smallint not null,
pago money not null default 0,
credito money default 0);
```

```
create table Hospedagem(
codReserva int not null primary key(codReserva),
dataEntrada datetime not null,
dataSaida datetime,
multa decimal default 0,
foreign key (codReserva) references Reserva(codReserva));

create table Transferidos(
codigo int not null, primary key(codigo),
codApart int not null,
data datetime not null,
foreign key (codigo) references Reserva(codReserva));

create table Frigobar(
codigo int not null references Apartamento(codApart),
tipo varchar(30) not null,
preco money not null,
stock smallint not null);

create table Consumos(
codReserva int references Hospedagem(codReserva),
produto varchar(30) not null,
preco money not null,
consumidos int);

create table InsertedCod(cod int not null);

create table InsertedApart(apart int not null);
```

Procedimentos e Triggers

Este procedimento permite registar hospede na base dados, recebe os dados do hospede e verifica se a pessoa existe, se existir não adiciona, mas se não existir vai registar a pessoa, isto porque se um funcionário quiser se hospedar no hotel, vai ter que registar-se como cliente e a pessoa já vai estar registada. E depois já adiciona o hospede.

```
create procedure AddHospedes @id varchar(10), @nome varchar(50), @sexo char(1), @morada
varchar(100), @codPostal varchar(8), @local varchar(20),
@telefone varchar(9), @email varchar(100), @dataNasc date as
if not exists(select * from Pessoa where Pessoa.id = @id)
begin
	insert into Pessoa values(@id, @nome, @sexo, @morada, @codPostal,@local, @telefone,
	@email, @dataNasc)
end
insert into Hospede values(@id);
```

Este procedimento permite registar funcionário na base dados, recebe os dados do funcionário e verifica se a pessoa existe, se existir não adiciona, mas se não existir vai registar a pessoa, isto porque se um hospede quiser trabalhar no hotel, vai ter que registar-se como uma pessoa. E depois já adiciona o funcionário, mas para adicionar um funcionário é preciso ter departamento, que vai estar ligada com o funcionário.

```
create procedure AddFuncionarios
@id varchar(10), @nome varchar(50), @sexo char(1), @morada varchar(100), @codPostal
varchar(20), @local varchar(20),
@telefone varchar(9), @email varchar(100), @dataNasc date, @depart varchar(20), @cargo
varchar(20), @salario money as
if not exists(select * from Pessoa where Pessoa.id = @id)
begin
```

```
	insert into Pessoa values(@id, @nome, @sexo, @morada, @codPostal,@local, @telefone,
@email, @dataNasc)
end
insert into Funcionarios values(@id, @depart, @cargo, getdate(), @salario);

create procedure AddDepartamento @nome varchar(20) as
insert into Departamento values(@nome);
```

Estes procedimentos server para adicionar os tipos de apartamento do hotel, registar apartamento, sempre que registar um apartamento vai ser disparada uma trigger que vai adicionar frigobar no quarto e está trigger chama o procedimento adicionar produtos dentro da frigobar

```
create procedure AddFrigobar @cod int, @tipo varchar(30), @stock smallint, @preco
decimal(4,2) as
insert into Frigobar values(@cod, @tipo, @stock, @preco);

create procedure AddApartamento @localizacao char(2), @tipoApart varchar(20), @quant
smallint, @preco money as
insert into InsertedApart(apart) values(IDENT_CURRENT('Apartamento'))
insert into Apartamento values(@localizacao, 1, @tipoApart, @quant, @preco, 1);

create trigger AddFrigobarQuarto on Apartamento after insert as
declare @apart int
set @apart = (select apart from InsertedApart)
exec AddFrigobar @apart, 'Sumo', 30, 1.80
exec AddFrigobar @apart, 'Cerveja', 30, 1.80
exec AddFrigobar @apart, 'Biscoito', 30, 1.80
exec AddFrigobar @apart, 'Agua', 30, 1.80
delete from InsertedApart

create procedure AddTipoApart @tipo varchar(20) as
insert into TipoApartamento values(@tipo);
```

Procedimento que permite registar uma reserva, para registar uma registar é preciso verificar se o apartamento está livre e se está limpo, se estiver calcula o valor da estadia, múltipla o preço dia do apartamento por o total de dias que vai decorrer a estadia (usando uma função do sql datediff para calcular o total de dias) e aplica a este preço um desconto diretamente, caso o hospede tiver desconto na sua conta, com o preço já reserva já calculado, basta adicionar a reserva na tabela e coloca o estado com reservado('R'), ai chama o procedimento para muda o desconto do hospede e atualizar a disponibilidade do apartamento, colocar o apartamento como indisponível(com o valor 0, falso), foi desenvolvimento um procedimento ao envés de uma trigger, pelo fato de este procedimento vai ser usado no para transferir um hospede do apartamento.

```
create procedure AddReserva @hospede int, @apart int, @check_in datetime, @check_out
datetime,
@obs varchar(100), @num int as
if exists (select disponivel from Apartamento where codApart = @apart and
Apartamento.disponivel = 1 and Apartamento.limpo = 1)
begin
	declare @total money
	declare @desc int = (select descontos from Hospede where codHospede = @hospede)
	set @total = (select precoDia*datediff(day, @check_in, @check_in)-
descontos/100*precoDia*datediff(day, @check_in, @check_in)
			from Apartamento, Hospede
			where Apartamento.codApart = @apart and Hospede.codHospede =
@hospede)
	insert into Reserva values(@hospede, @apart, @check_in, @check_out, 'R', 0, @obs,
@num, @total, 0)
	exec UpdateDisponivel @apart
```

```
	exec AlterarDescontos @hospede;
end

create procedure AlterarDescontos @hospede int as
update Hospede
set descontos = 0
where codHospede = @hospede

create procedure UpdateDisponivel @apart int as
update Apartamento
set disponivel = 0
where codApart = @apart
```

Método que faça o check-in de uma determinada reserva, primeiramente verifica se já pode efetuar o check-in(se hoje for a data de entrada no apartamento), adiciona na tabela Hospedagem que hoje o hospede entrou no apartamento, ai a trigger para atualizar o estado da reserva é dispara(porque houve uma inserção na tabela Hospedagem), dizendo que o hospede já entrou no apartamento('O'), em seguida adiciona os consumos na hospedagem, isto é, adiciona todos os produtos do frigobar, mas fica registado que não houve ainda nenhum consumo(consumidos fica com zero).

```
create procedure AddCkechIn @cod int as
if exists (select * from Reserva where dataCheck_int >= getdate() and codReserva = @cod)
begin
	insert into InsertedCod(cod) values (@cod)
	insert into Hospedagem(codReserva, dataEntrada) values (@cod, getdate());
	exec AddConsumos @cod
end

create trigger UpdateStatus on Hospedagem after insert as
update Reserva
set statusReserva = 'O'
where codReserva = (select cod from InsertedCod);
delete from InsertedCod;

create procedure AddConsumos @cod int as
insert into Consumos values(@cod, 'Sumo', 1.80, 0)
insert into Consumos values(@cod, 'Cerveja', 1.80, 0)
insert into Consumos values(@cod, 'Biscoito', 1.80, 0)
insert into Consumos values(@cod, 'Agua', 1.80, 0)

create procedure UpdateConsumos @agua int, @sumo int, @cerveja int, @biscoito int as
exec UpdateConsumosAgua @agua
exec UpdateConsumosSumo @sumo
exec UpdateConsumosoCerveja @cerveja
exec UpdateConsumosBiscoito @biscoito;

create procedure UpdateConsumosAgua @quant int as
update Consumos
set consumidos += @quant
where produto = 'Agua'

create procedure UpdateConsumosSumo @quant int as
update Consumos
set consumidos += @quant
where produto = 'Sumo'

create procedure UpdateConsumosoCerveja @quant int as
update Consumos
set consumidos += @quant
where produto = 'Cerveja'

create procedure UpdateConsumosBiscoito @quant int as
```

```
update Consumos
set consumidos += @quant
where produto = 'Biscoito'
```

Metodo que faz a transferência de um hospede para um determinado apartamento, coloca o apartamento que o hospede estava para ser limpado(isto é chama o procedimento LimparQuarto, para colocar o quarto para ser feito a limpeza e continua indisponível), acerta as conta com o hospede, verificando se o hospede tem que pagar mais ou menos(se o hospede for transferido para um apartamento mais caro, deve pagar os dias que vai permanecer no quarto mais caro e vise versa), coloca o apartamento que o hospede vai ficar como indisponível para isso chama o método UpdateDisponivel que coloca o apartamento como indisponível, e depois adiciona na tabela transferidos o apartamento onde o hospede estava, qual foi a reserva(codReserva) e quando realizou a transferência, quando inserir na tabela Transferido, é disparado a trigger que muda o campo transferido para true(ou seja, dizer que houve uma transferência na reserva).

```
create procedure AddTransferencia @cod int, @apart int, @hos int as
declare @ap int = (select Apart from Reserva where codReserva = @cod)
insert into InsertedCod(cod) values (@cod)
insert into InsertedApart(apart) values (@apart)
exec LimparQuarto @ap
exec AcertarContar @cod, @ap, @apart
exec UpdateDisponivel @ap
insert into Transferidos values(@cod, @ap, getdate());

create trigger UpdateEstado on Transferidos after insert as
update Reserva
set transferidos = 1,
Apart = (select apart from InsertedApart)
where codReserva = (select cod from InsertedCod)
delete from InsertedApart
delete from InsertedCod;

create procedure AcertarContar @cod int, @ant int, @novo int as
Update Reserva
set credito = pago - ((select precoDia*datediff(day, dataCheck_int, getdate()) from
Apartamento, Reserva
where codApart = @ant and Reserva.Apart = Apartamento.codApart)+(select
precoDia*datediff(day, getdate(), datacheck_out) from Apartamento, Reserva
where codApart = @novo and Reserva.codReserva = @cod))
where codReserva = @cod;

create procedure LimparQuarto @apart int as
update Apartamento
set limpo = 0
where codApart = @apart;
```

Procedimento para realizar o check-out, este procedimento atualiza na tabela Reserva dizendo que já foi feita o check-out (muda o estado da reserva para ‘C’) e atualiza o saldo calcula se o hospede consumiu os produtos do frigobar, depois que atualizar a reserva como ‘C’ é disparada uma trigger para fechar a estadia colocando a data saída com a data atual e calcula se o hospede têm que pagar multa (se o hospede entregou o quarto depois do dia previsto), calcula quanto foi no total, e coloca o quarto como sujo(chama o procedimento LimparQuarto para marcar o campo limpo como falso)

```
create procedure AddCheckOuts @codigo int, @apart int as
insert into InsertedCod values(@codigo)
update Reserva
set statusReserva = 'C',
saldo += (select sum(preco*consumidos) from Consumos where codReserva = @codigo)
where codReserva = @codigo
exec LimparQuarto @apart;
```

```
exec AddCheckOuts 2, 5;

create trigger CloseEstadia on Reserva after update as
declare @cod int = (select cod from InsertedCod)
update Hospedagem
set dataSaida = GETDATE(),
multa = datediff(day, getdate(), (select datacheck_out from Reserva where codReserva =
@cod))
where codReserva = @cod;
```

Método que classifica o hotel, calcula a classificação média de todos os pontos recebimento, faz o uso da função AVG que calcula a média

```
create procedure ClassificarHotel @hospede int, @pontos decimal(2,1) as
insert into Hotel values(@hospede, @pontos)
```

Procedimento que mostra qual é a pontuação do hotel

```
create procedure ShowClassificacao as
select AVG(classificacao) as Classificação from Hotel
```

Procedimento que atualiza no sistema quando um apartamento já estiver limpo e pronto para estar disponível, para outros poder reservar o mesmo.

```
create procedure QuartoLimpo @apart int as
update Apartamento
set limpo = 1,
disponivel = 1
where codApart = @apart;
```

Procedimento que pesquisa um determinado hospede e mostra as informações do mesmo.

```
create procedure PesquisarHospedes @cod varchar(10) as
select nome, Pessoa.id, morada, codPostal, localidade, email, dataNasc, descontos
from Hospede, Pessoa
where Hospede.id = Pessoa.id and Hospede.codHospede = @cod
```

Método que pesquisa um determinado funcionario através do código e mostra as suas informações

```
create procedure PesquisarFuncionarios @cod int as
select codFun, nome, Pessoa.id, morada, codPostal, localidade, telefone, email, dataNasc,
depart, cargo, dataAdmissao, salario
from Funcionarios, Pessoa
where Funcionarios.id = Pessoa.id and @cod = Funcionarios.codFun;
```

Método que pesquisa uma determinada reserva e mostra toda as informações relativamente a essa reserva

```
create procedure PesquisarReserva @cod int as
select * from Reserva where codReserva = @cod;
```

Procedimento que mostra todas as reservas de hoje, todos os hospedes de estão previsto para chegar

```
create procedure ReservasHoje as
select * from Reserva where dataCheck_int = (select convert (date, getdate()));
```

Procedimento que mostra todas os checkouts de hoje, todos os hospedes que estão previstos para sair do hotel hoje

```
create procedure CheckoutHoje as
select * from Reserva where dataCheck_out = getdate();
```

Procedimento que mostra algumas estatísticas, relativamente qual foi a reserva mas bem paga, qual foi a menos paga, qual foi a media gasta pelo utente no hotel, e Total de gasto por todos os hospedes no hotel

```
create procedure RelatoriosEstatisticos as
select MAX(pago) as Maior_Gasto, Min(pago) as Menor_Gasyo, AVG(pago) as Media_Gasto,
Sum(pago) as Gasto_Total
from Reserva
```

Procedimento que mostra quantas transferências que já aconteceu no hotel

```
create procedure TotalTransferencia as
select count(*) as Total_Transferencia from Reserva where transferidos = 1
```

Procedimento que mostra o top 5 de reservas mais bem pagas, entre os hospedes no hotel

```
Create procedure TopReservas as
select Top(5) codReserva, nome, pago
from Reserva, Pessoa, Hospede
where Reserva.hospede = hospede.codHospede and hospede.id = Pessoa.id
order by pago DESC
```