U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Bases de Dados

MIEIC

Base de dados para gestão de um restaurante

Turma 2

Grupo 203

Alunos:

Amadeu Prazeres Pereira - up201605646

Nuno Tiago Tavares Lopes – up201605337

Tomás Nuno Fernandes Novo – up201604503

Índice

Descrição sucinta do contexto

Neste projeto optámos pela elaboração de uma base de dados que facilite a gestão de um restaurante, objetivando uma confortável administração dos recursos humanos e espaços disponíveis do respetivo local de restauração.

As informações manipuladas incidem principalmente sobre os principais componentes necessários ao funcionamento eficaz do restaurante, nomeadamente dados relativamente aos funcionários e à quantidade de mesas disponíveis em cada sala e seu correspondente número de lugares. Também são geridas informações acerca do género de evento de cada sala, das ementas existentes e, como não podia faltar, da organização dos clientes no restaurante.

Na base de dados temos as classes **Cliente** e **Funcionario**, ambos herdam atributos da classe **Pessoa** (nome, NIF, nacionalidade). Os clientes têm como atributos a data de nascimento e um review (0 a 5 estrelas) em relação ao serviço desempenhado pelo restaurante. Estes podem se sentar numa **Mesa** de 2, 4 ou 10 pessoas e podem efetuar uma **Reserva** destas mesmas, indicando a hora, a data e o número de pessoas. A classe **Mesa** possui um número que a identifica, o número de lugares e a despesa total dos clientes que pertencem à mesa.

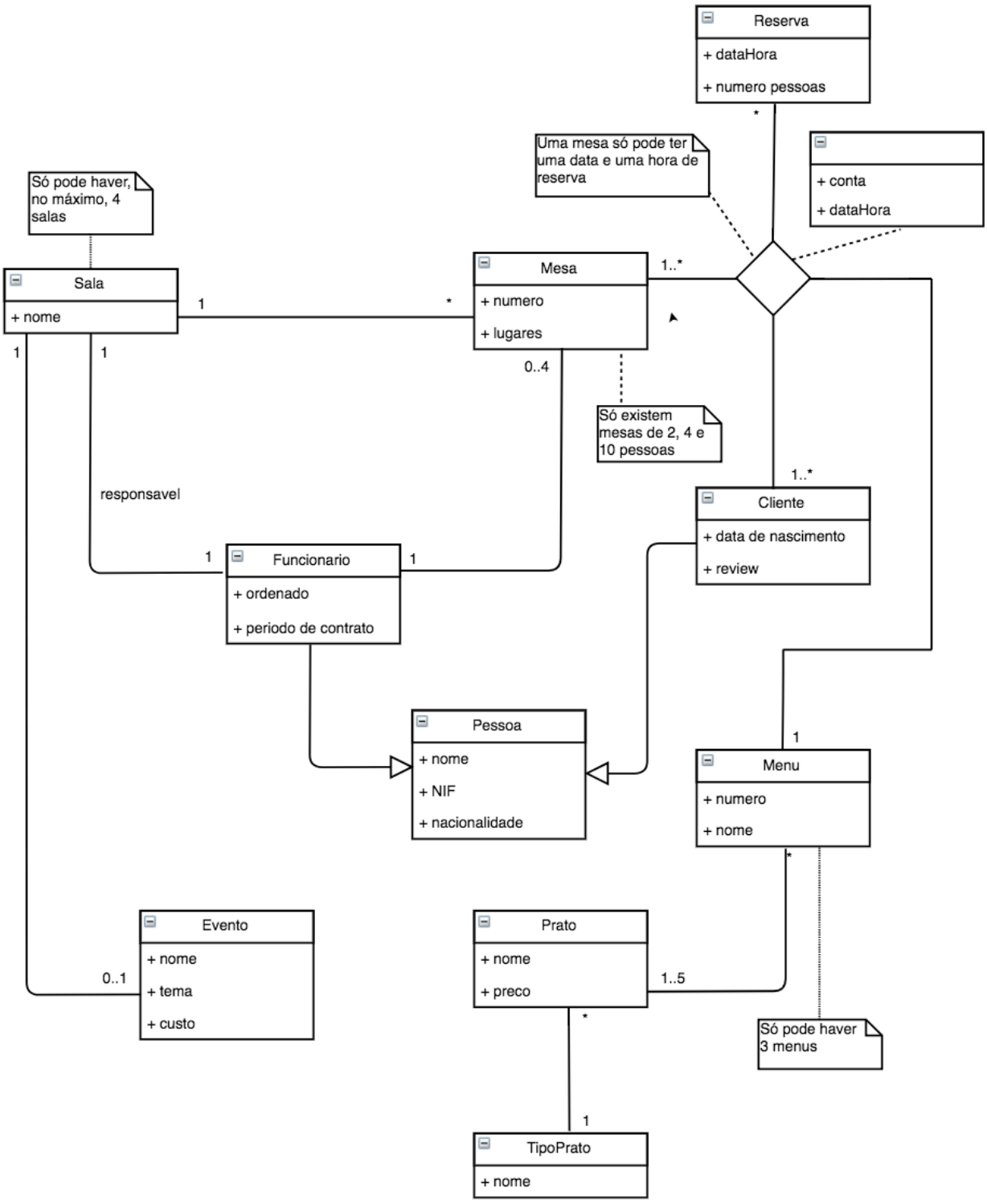
Os funcionários por sua vez são identificados por um ID único. Apenas um funcionário é responsável por cada **Sala** e cada um está encarregado, no máximo, de 4 mesas. No restaurante só podem existir até 4 salas.

Em cada sala pode existir um **Evento**, que é identificado pelo seu nome, pelo tipo (Concerto, Magia, etc.) e pelo custo da organização do evento.

À disposição dos clientes, o restaurante possui 3 **Menus**, sendo que cada um pode possuir entre 1 a 5 **Pratos**, sendo estes distinguidos entre si através do nome, **Tipo** e preço.

Desta forma, a solução proposta visa organizar eficientemente a realização das atividades de restauração.

Diagrama UML



Esquema Relacional

O texto abaixo representa o nosso modelo relacional, sendo que os atributos a sublinhado são as chaves primárias de cada relação.

Sala (idSala, nome, responsavel->Funcionario)

Mesa (numero, lugares, idFuncionario->Funcionario, idSala->Sala)

Reserva (idReserva, dataHora, numeroPessoas)

Pessoa (idPessoa, nome, NIF, nacionalidade)

Cliente (idCliente->Pessoa, data de nascimento, review)

Funcionario (idFuncionario->Pessoa, ordenado, periodoContrato)

Menu (numero, nome)

Evento (idEvento, nome, tema, custo, idSala->Sala)

Prato (idPrato, nome, preco, idTipoPrato->TipoPrato)

TipoPrato (idTipoPrato, nome)

ClienteMesaReservaMenu (idReserva->Reserva, idMesa->Mesa, idCliente->Cliente, idMenu->Menu, conta, dataHora)

MenuPrato (idMenu->Menu, idPrato->Prato)

Análise dependências funcionais e formas normais

Nesta secção será feita a análise das dependências funcionais do nosso modelo, assim como a verificação da não violação à Forma Normal Boyce-Codd e à 3ª Forma Normal.

Sala

idSala -> nome, responsável

Mesa

numero -> lugares, idFuncionario, idSala

Reserva

idReserva -> dataHora, numeroPessoas

Cliente

idCliente -> data de nascimento, review

Funcionario

idFuncionario -> ordenado, periodoContrato

Pessoa

idPessoa -> nome, NIF, nacionalidade

NIF -> idPessoa, nome, nacionalidade

Evento

idEvento -> nome, tema, custo, idSala

Menu

numero -> nome

nome -> numero

Prato

idPrato -> nome, preco, idTipoPrato

nome -> idPrato, preco, idTipoPrato

TipoPrato

idTipoPrato -> nome

nome -> idTipoPrato

ClienteMesaReservaMenu

idReserva, idMesa, idCliente, idMenu -> conta, dataHora

MenuPrato

Não possui dependências funcionais.

Todas as dependências funcionais acima referidas:

- Estão na 1ª Forma Normal, pois todos os atributos têm exclusivamente valores atómicos.
- Estão na 2ª Forma Normal, pois todos os atributos têm dependência funcional da chave primária.
- Estão na 3ª Forma Normal, pois não existe dependência transitiva.
- Estão na Forma Normal de Boyce-Codd, pois não existe redundância em dependências funcionais.

Restrições e respetiva implementação

Os seguintes atributos (identificados com sublinhado) possuem a restrição PRIMARY KEY, na sua relação (identificada a **negrito**), dado que identificam um tuplo na tabela respetiva de forma única:

Reserva: idReserva (não pode haver duas reservas com o mesmo id)

Mesa: numero (não pode haver duas mesas com o mesmo número)

Cliente: idCliente (não pode haver dois clientes com o mesmo id)

Funcionario: idFuncionario (não pode haver dois funcionários com o mesmo id)

Sala: idSala (não pode haver duas salas com o mesmo id)

Pessoa: idPessoa (não pode haver duas pessoas com o mesmo id)

Evento: idEvento (não pode haver dois eventos com o mesmo id)

Prato: idPrato (não pode haver dois pratos com o mesmo id)

Menu: numero (não pode haver dois menus com o mesmo número)

TipoPrato: idTipoPrato (não pode haver dois tipos de prato com o mesmo id)

Os seguintes atributos (identificados com sublinhado) possuem a restrição FOREIGN KEY, na sua relação (identificada a **negrito**), dado que representam entidades pertencentes a outras tabelas da base de dados, tendo de existir na sua relação:

Sala: responsavel (chave estrangeira para Funcionario)

Mesa: idFuncionario (chave estrangeira para Funcionario), idSala (chave estrangeira para Sala)

Cliente: idCliente (chave estrangeira para Pessoa -> generalização)

Funcionario: idFuncionario (chave estrangeira para Pessoa -> generalização)

Evento: idSala (chave estrangeira para Sala)

Prato: idTipoPrato (chave estrangeira para TipoPrato)

ClienteMesaReservaMenu: idReserva (chave estrangeira para Reserva), idMesa (chave estrangeira para Mesa), idCliente (chave estrangeira para Cliente), idMenu (chave estrangeira para Menu)

MenuPrato: idMenu (chave estrangeira para Menu), idPrato (chave estrangeira para Prato)

Os seguintes atributos (identificados com sublinhado) possuem a restrição UNIQUE, na sua relação (identificada a **negrito**), dado que são atributos únicos em cada relação:

Menu: nome (não existem dois menus com o mesmo nome)

Pessoa: NIF (não existem duas pessoas com o mesmo NIF)

Prato: nome (não existem dois pratos com o mesmo nome)

TipoPrato: nome (não existem dois tipos de prato com o mesmo nome)

Os únicos atributos (identificados com sublinhado) que não possuem a restrição NOT NULL, na sua relação (identificada a **negrito**), são os seguintes (todos os outros possuem):

ClienteMesaReservaMenu: conta, dataHora (de momento ambos estes atributos podem ser NULL pois mais tarde vão ter de ser calculados a partir de outros atributos)

Cliente: dataNascimento (o cliente pode não querer dar a sua data de nascimento)

As seguintes restrições são implementadas na respetiva **relação** através de um CHECK para assegurar que qualquer combinação dos atributos envolvidos nas instancias tornem verdade as seguintes condições:

Cliente: A review tem de ser um valor compreendido entre 0 e 5

Funcionario: O ordenado e o período de contrato têm de ser superiores a 0

Mesa: As mesas só podem ter 2, 4 ou 10 lugares

Reserva: Uma reserva tem de ter um número de pessoas superior a 0

Interrogações

Apresenta-se abaixo, uma lista de queries pertinente para a compreensão do correto funcionamento dos vários componentes e respetivas conexões na base de dados.

1. Custo médio dos eventos realizados nas salas: utilização do método AVG para calcular o custo médio dos eventos que foram realizados nas salas do restaurante.

2. Nomes dos funcionários que são responsáveis por uma sala e o respetivo número da sala: verificação da associação das classes **Pessoa**, **Funcionario** e **Sala**.

3. Nomes dos 4 funcionários com melhor ordenado: verificação da associação das classes **Pessoa** e **Funcionário**. Ordenação descendente de acordo com o ordenado de cada funcionário. É limitado a 4 funcionários através do parâmetro LIMIT 5.

4. Listagem ordenada dos clientes com menos de 40 anos: verificação das interligações entre a classe **Cliente** e **Pessoa**. Utilização do método DATE('now') para a obtenção da data atual e consequentemente para o cálculo da idade de cada cliente.

5. Listagem dos clientes que pediram um menu em que o prato é do tipo Carne: verificação da associação das classes **Pessoa**, **Cliente**, **TipoPrato**, **Prato**, **ClienteMesaReservaMenu**, **Menu**, **MenuPrato**.

6. Nomes dos clientes e do funcionário responsável por cada um dos clientes e a data do atendimento: verificação das interligações entre **Pessoa**, **Cliente**, **ClienteMesaReservaMenu**, **Reserva**, **Mesa**, **Funcionario**. Utilização do método *strftime* para obter a data do atributo dataHora do tipo DATETIME.

7. Listagem dos clientes, número de reserva e número da mesa tendo como critério uma limitação nas datas das reservas: verificação da associação das classes **Pessoa**, **Cliente**, **Mesa**, **Reserva** e **ClienteMesaReservaMenu**. Ordenação ascendente tendo em conta a data e hora da reserva. A data da reserva está compreendida entre 2018-06-14 e 2018-11-30, dando uso ao parâmetro BETWEEN.

8. Conta total de cada mesa por dia: verificação da associação entre as classes **Mesa**, **Reserva**, **ClienteMesaReservaMenu**, **Menu**, **MenuPrato**, **Prato**. Utilização da diretiva SUM que adiciona sucessivamente o preço do menu de cada cliente de cada mesa.

9. Menu mais pedido pelos clientes: utilização de vistas para obter o tipo de menu mais pedido pelos clientes do restaurante.

10. Nome do segundo funcionário com um ordenado mais elevado: verificação das ligações entre a classe **Funcionario** e **Pessoa**. Utilização do método MIN para obter o ordenado mínimo da seleção dos dois funcionários com ordenados mais elevados.

Gatilhos

Na 3ª parte do projeto, implementámos três gatilhos imprescindíveis na monitorização da nossa base de dados, sendo eles:

- Um gatilho que antes da criação de uma nova **Pessoa** (BEFORE INSERT), valida , ou não, o seu ***id***. Assim, se for inserida na nossa base de dados uma pessoa com um id igual ao id de uma pessoa já existente, este gatilho aborta a atualização da data e comunica ao utilizador que ocorreu um erro.

- Um gatilho que após a inserção de uma nova **Sala** (AFTER INSERT) na nossa base de dados, atualiza a mesma apagando qualquer inserção de uma nova sala, se o número de salas for superior a quatro.

- Um gatilho que após a tentativa de remoção de um **Prato** (BEFORE DELETE) de um menu da base de dados, verifica se o menu ao qual se quer remover o prato fica sem pratos. Se tal acontecer o último prato não é removido do menu, impedindo assim que qual menu fique sem pratos.

Instruções de execução

Para evitar comportamento indesejado por parte da base de dados, os seguintes passos deverão ser seguidos:

- Executar o ficheiro **criar.sql**.
- Executar o ficheiro **povoar.sql**.
- Para correr as interrogações e os gatilhos basta abrir as pastas **Queries** e **Triggers**, respetivamente, que estão no diretório principal da entrega.

Conclusão

Com o finalizar das três partes do projeto, concluímos que a sua realização foi positiva para os 3 elementos constituintes do grupo visto que fomos capazes de colocar em prática os conhecimentos adquiridos tanto nas aulas teóricas como nas práticas e adquirimos novas capacidades de programação após o estudo da linguagem de programação **SQL**.

Criámos uma base de dados coesa, íntegra e que obedece a todas as restrições impostas. As interrogações criadas revelam-se apropriadas para a base de dados. Os gatilhos implementados demonstram-se proveitosos para a manutenção da nossa base de dados.

Sintetizando, a realização deste trabalho apenas trouxe benefícios ao grupo.