UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA MECÂNICA
GRADUAÇÃO EM ENGENHARIA MECATRÔNICA
DISCIPLINA BANCO DE DADOS
PROFESSOR(A) MARIA CAMILA NARDINI BARIONI

PROJETO FINAL: PIZZA DELIVERY COM ENTRETENIMENTO

DISCENTES:	**NÚMERO DE MATRÍCULA:**
Eduardo Marques da Silva	11721EMT018
Guilherme Salomão Agostini	11721EMT003
Luiz Renato Rodrigues Carneiro	11721EMT004

02/06/2021

Sumário

1- ESPECIFICAÇÃO DO PROBLEMA:

A seguir, é apresentado a especificação do problema, sendo as partes coloridas, indicações semânticas de entidade, atributo e relacionamento. Os requisitos adicionais são informados pela letra "t" e "u":

a) O funcionamento básico do app de Pizza Delivery com Entretenimento é o seguinte: consumidores podem fazer pedidos de pizzas para restaurantes para serem entregues em um endereço específico, e se eles quiserem, eles podem escolher um "pedido de entretenimento" especial. Quando um pedido é um pedido de entretenimento, o entregador permanece com o consumidor após entregar a pizza e diverte os consumidores (por exemplo, cantando, contando piadas, fazendo truques de mágica, etc.) por certo período de tempo.

b) Quando as pessoas criam uma conta para o app e tornam-se usuários do app, cada uma delas tem que indicar sua data de nascimento e informar seu nome e endereço. Cada usuário deve também ser identificado univocamente.

c) Após a criação da conta, devem ser oferecidas três opções para os usuários: a primeira opção no app é selecionar "dono de negócio". Desses donos de negócio, também é necessário solicitar suas contas no LinkedIn para que seja possível adicioná-los na rede profissional dos donos do app.

d) Todo dono de negócio pode possuir várias pizzarias. Uma pizzaria tem que pertencer a um único dono.

e) De cada uma dessas pizzarias, é necessário registrar o CEP, endereço, número de telefone, web site, e os horários de funcionamento (horário de abertura e horário de fechamento).

f) Cada pizzaria pode oferecer diversas pizzas. Para cada uma das pizzas é preciso registrar o nome (marguerita, quatro queijos, etc.) e o preço. Embora duas pizzas de pizzarias diferentes possam ter o mesmo nome, elas não serão exatamente iguais uma vez que o sabor será diferente, e por isso devem ser consideradas únicas. Além disso, as pizzas devem ser distinguíveis mesmo que elas tenham o mesmo preço, por exemplo, uma pizza marguerita da Pizzaria Pronto de Uberlândia que custa R$48,00 deve ser distinguível da pizza marguerita da Pizzaria Papa Léguas de Araguari, que também custa R$48,00.

g) As pizzas oferecidas pelas pizzarias são categorizadas com base em uma hierarquia de categoria fixa (por exemplo, a pizza marguerita pode ser categorizada como PIZZASALGADA -> TRADICIONAL e a pizza quatro queijos pode ser categorizada com PIZZA SALGADA -> ESPECIAL). Cada categoria é identificada por um código numérico e possui uma descrição da categoria.

h) Cada pizzaria pode oferecer também a entrega de acompanhamentos para as pizzas, como, bebidas, saladas e sobremesas. Para cada acompanhamento é necessário armazenar nome, descrição, tipo de acompanhamento (bebida, salada ou sobremesa), e preço. Cada acompanhamento recebe um código que o distingue apenas entre os acompanhamentos de uma mesma pizzaria.

i) A segunda opção no app é selecionar "consumidor faminto". Para esses consumidores famintos, é necessário saber qual o endereço de entrega.

j) Consumidores famintos podem fazer pedidos de pizzas com ou sem acompanhamentos. Cada pedido recebe um ID, e é necessário que o app registre a data e o horário em que o pedido foi feito. Também deve ser permitido que o consumidor faminto indique um horário posterior para a entrega e deve ser perguntado para quantas pessoas é o pedido.

k) Um pedido pode conter uma ou mais pizzas. Uma pizza pode ser incluída em vários pedidos. Para cada pizza pedida deve ser possível escolher opções de preparação. As opções de preparação devem incluir definição da massa (fina, média ou grossa),da borda (normal ou recheada com catupiry) e da quantidade de molho (pouco, normal ou extra).

l) Um pedido pode conter um ou mais acompanhamentos. Para cada acompanhamento solicitado é necessário definir a quantidade desejada do mesmo.

m) Além disso, também deve ser permitido que o consumidor faminto adicione ingredientes extras para cada pizza incluída em seu pedido. É importante notar que a adição de ingredientes extras na inclusão de uma pizza em um pedido é **opcional**. Um mesmo ingrediente extra pode ser adicionado em várias inclusões de pizzas em pedidos.

n) Para cada ingrediente extra é necessário saber seu código identificador, nome e preço.

o) Além disso, um tipo especial de pedido pode ser feito: o pedido de entretenimento. Nem todo pedido tem que ser um pedido de entretenimento. Mas quando um consumidor faminto indica que ele ou ela querem ser entretidos enquanto comem a pizza, será necessário registrar as informações de um pedido padrão e também o tipo de entretenimento que o usuário quer e por quanto tempo (a duração).

p) O custo total do pedido deve ser **calculado** com base nas pizzas escolhidas (levando em consideração também os ingredientes extras adicionados) e nos acompanhamentos e entretenimentos selecionados.

q) A terceira opção no app é selecionar "animador". Quando um usuário seleciona animador, ele tem que fornecer um nome artístico, escrever uma biografia resumida sobre ele mesmo, e indicar o preço para 30 minutos de entretenimento.

r) Todo pedido de entretenimento é atendido por exatamente um animador.

s) Cada animador pode escolher para cada pizzaria(s) ele/ela quer trabalhar. Para cada pizzaria que um animador quiser trabalhar, ele/ela deve indicar sua disponibilidade por dia (Segunda, Terça, Quarta, etc.).

t) O aplicativo deve ser capaz de registrar os entregadores, sendo animador um tipo de entregador. O entregador deve registrar a placa do veículo para que o cliente possa identifica-lo em maior detalhe. O entregador é responsável pela entrega de pedidos convencionais. Os pedidos também podem ser retirados na loja, e nesse caso não é necessário o entregador.

u) O pedido deve registrar o método de pagamento, sendo este, identificado pelo nome (cartão, dinheiro, etc). Caso o método dinheiro seja escolhido, é necessário que o troco seja indicado. Da mesma forma caso o método cartão seja escolhido, é necessário indicar a bandeira do cartão. Não existem pedidos ou pedidos animados que não sejam pagos.

2- ESQUEMA CONCEITUAL:

Será exibido a seguir o esquema conceitual completo da especificação do problema e logo após, o detalhamento dos porquês que nos levaram a esquematizar da maneira apresentada na Figura 1.

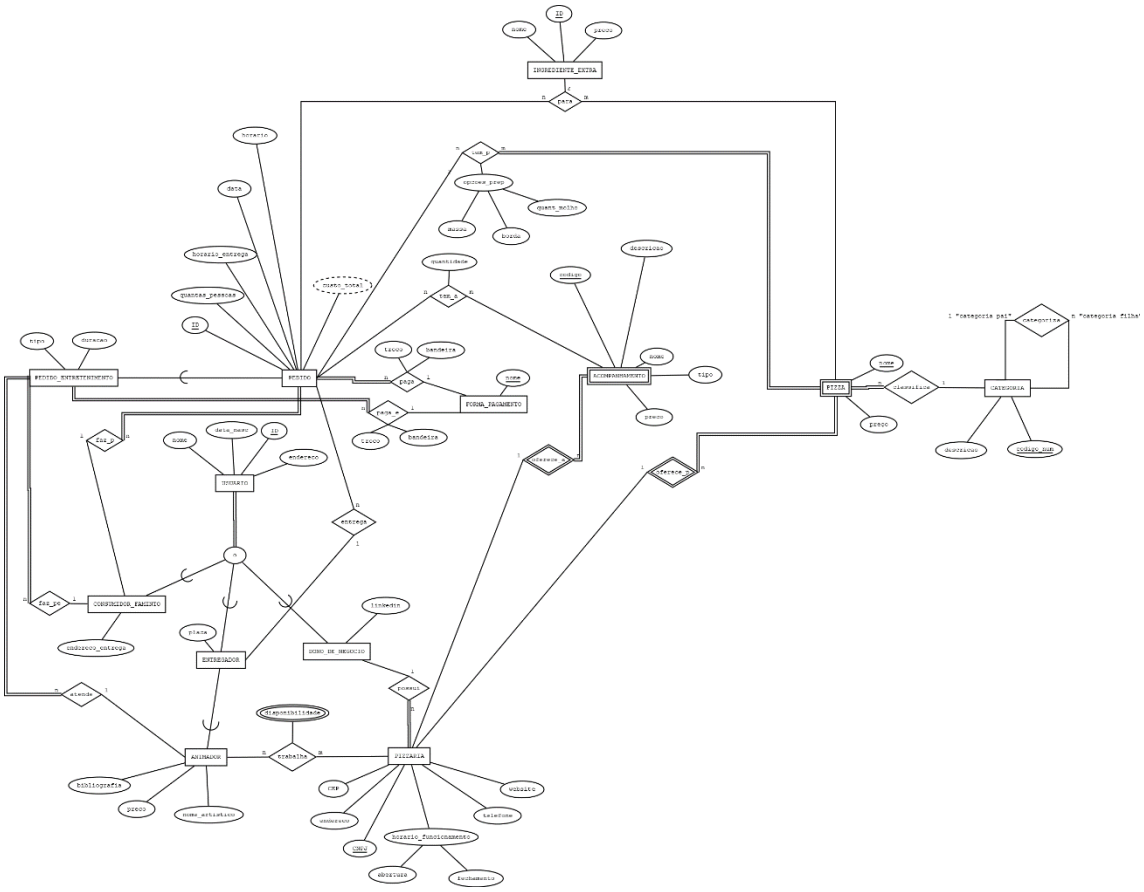

Figura 1 - Esquema conceitual do aplicativo especificado. Para maiores detalhes consulte o arquivo "MER-X.png"

a) O parágrafo nos diz que existem entidades que representam "consumidor", "pedido", "pedido entretenimento", "entregador" e "restaurantes". Os pedidos serão entregues à um endereço específico, provavelmente atributo do consumidor. São apresentados dois relacionamentos: fazer e entregar.

b) Existe uma entidade "usuário" possuindo atributo data de nascimento, nome, endereço, e ID. O ID é a chave primaria da entidade pois é único.

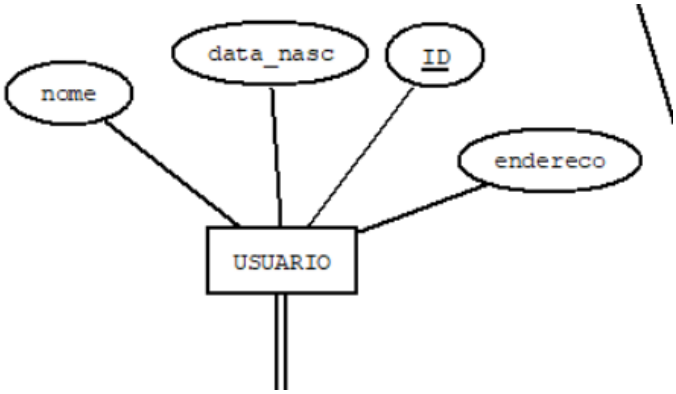

Figura 2 - Entidade usuário vinda de b).

c) O parágrafo informa que existem três subclasses originadas de usuário, sendo uma delas: dono de negócio. Esta subclasse, que é uma entidade filho, possui um atributo extra: Linkedin. Além disso foi escolhido a restrição *overlap* "sobreposição" pois um usuário pode pertencer a várias classes simultaneamente, por exemplo um dono pode também ser um consumidor faminto. Além disso a participação é total pois não existe um usuário que não pertence a pelo menos uma dessas subclasses.

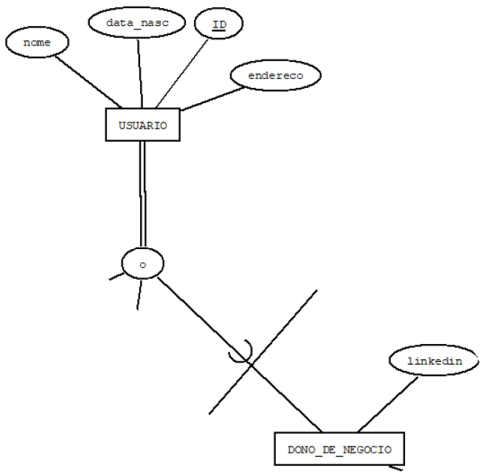

Figura 3 - Sub classe dono de negócio de c)

d) Existe uma entidade chamada pizzaria que equivale a entidade restaurante que por sua vez está relacionada com a entidade dono por meio do relacionamento "possui". A cardinalidade desse relacionamento é de 1 para n, pois uma pizzaria pode possui um único dono e um dono pode possuir várias pizzarias, além disso toda pizzaria deve possuir um dono assim exigindo dependência de participação total por parte da pizzaria. A pizzaria não é uma entidade fraca pois ela pode ser vendida para um novo dono.

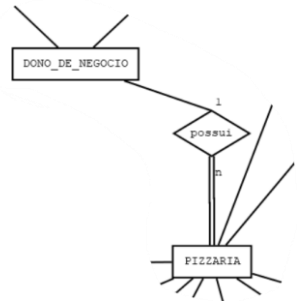

Figura 4 – Relacionamento entre pizzaria e dono de negócio

e) Essa entidade pizzaria possui vários atributos entre eles CEP, endereço, número de telefone, web site, e os horários de funcionamento (horário de abertura e horário de fechamento). Os horários de funcionamento serão representados por um atributo composto pois só existem dois tipos de horário e eles quase sempre estão presentes. Foi adicionado em consenso com a equipe que seria necessário um atributo extra para ser a chave primária, assim foi adicionado a chave primária CNPJ.

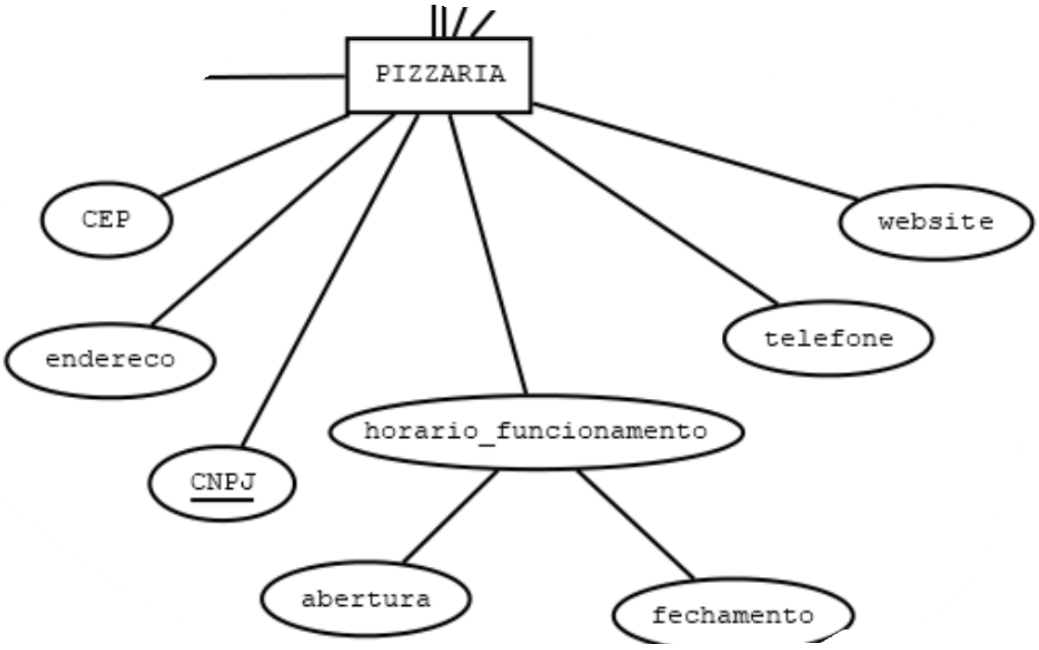

Figura 5 – Atributos da entidade pizzaria

f) A entidade pizza é uma entidade fraca da entidade pizzaria, dessa forma, seu nome é uma chave primária fraca, dessa forma sendo definido por um relacionamento com cardinalidade 1 para n e dependência de participação total por parte da pizza. A chave primária da pizza será composta pela chave primária fraca e pela chave primária da

pizzaria, distinguindo as pizzas de mesmo nome, porém em pizzarias distintas. Além disso a pizza possui um atributo chamado preço. Assim uma pizza pode pertencer a apenas uma pizzaria e uma pizzaria pode ter várias pizzas.

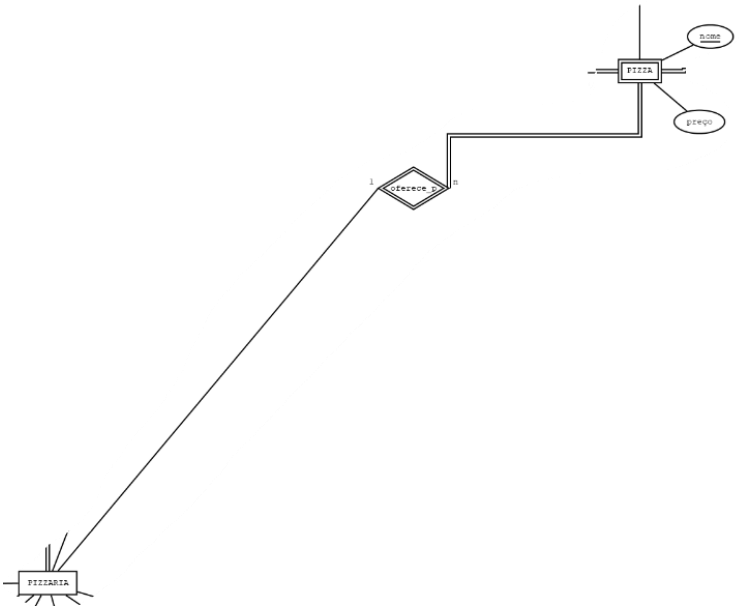

Figura 6 – Relacionamento entre pizzaria e pizza

g) Uma entidade categoria foi criada com o atributo descrição e chave primária código numérico. Esta entidade possui uma auto junção de cardinalidade 1 para n, ou seja, uma categoria pode classificar várias categorias. A pizza é classificada por meio de um relacionamento com a categoria, esse relacionamento é de n para 1 pois, uma pizza pode pertencer a uma categoria e uma categoria pode possuir várias pizzas e apresenta dependência de participação total por parte da pizza, pois toda pizza possui uma categoria.

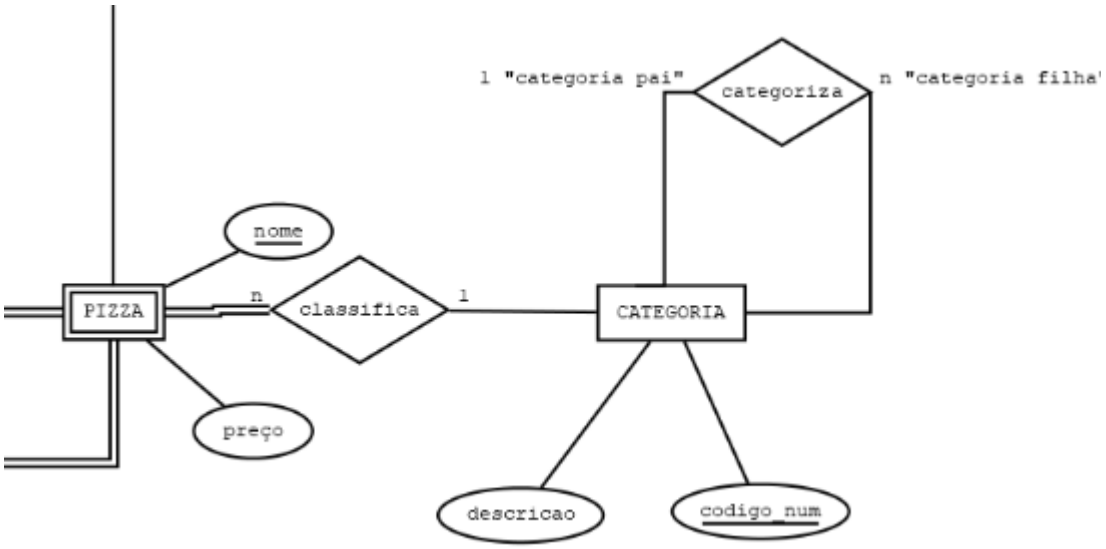

Figura 7 – Categoria de hierarquia fixa para pizzas

h) A entidade acompanhamento é entidade fraca da entidade pizzaria, dessa forma, seu código é uma chave primária fraca, sendo assim, definido por um relacionamento com cardinalidade 1 para n e dependência de participação total por parte do acompanhamento. Além disso essa entidade possui os seguintes atributos: nome, descrição, tipo de acompanhamento, e preço.

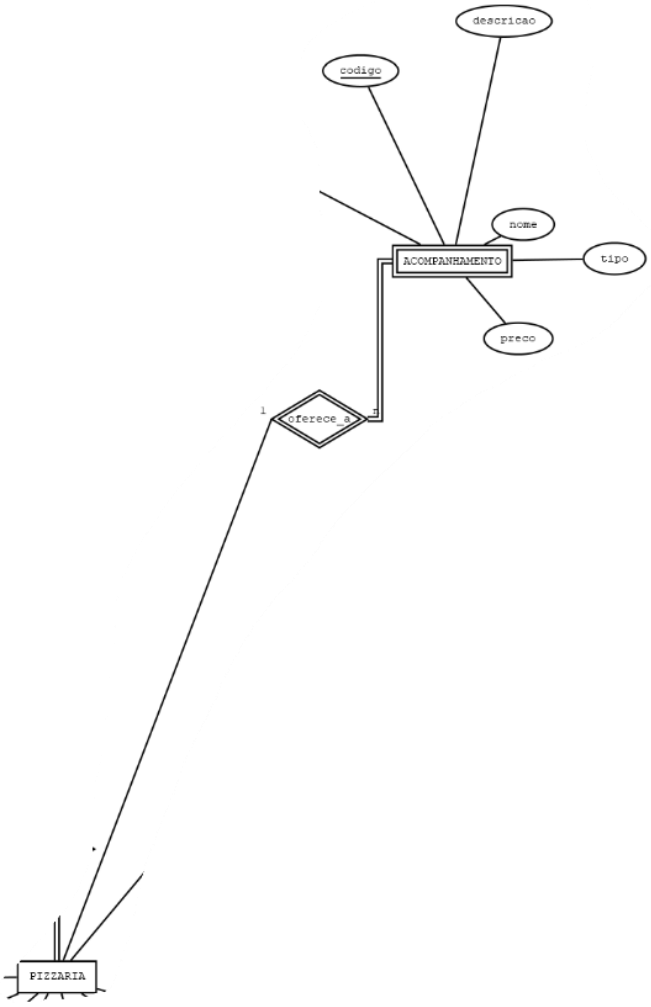

Figura 8 – Acompanhamento é uma entidade fraca de pizzaria

i) A entidade usuário tem uma subclasse consumidor faminto que possui o atributo endereço de entrega.

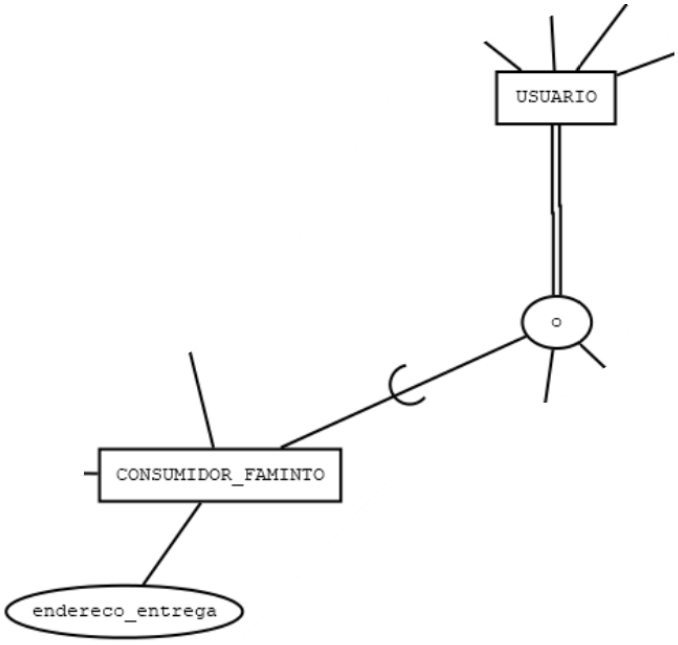

Figura 9 – Consumidor faminto subclasse de usuário

j) A entidade pedido foi criada com chave primária ID, atributos data, horário, horário posterior para a entrega e quantas pessoas. Essa entidade possui um relacionamento “faz_p” com o consumidor, esse relacionamento possui cardinalidade 1 para n, pois um pedido pode ser de apenas um consumidor e um consumidor pode fazer vários pedidos, além disso apresenta participação total por parte do pedido, pois não existem pedidos sem consumidores.

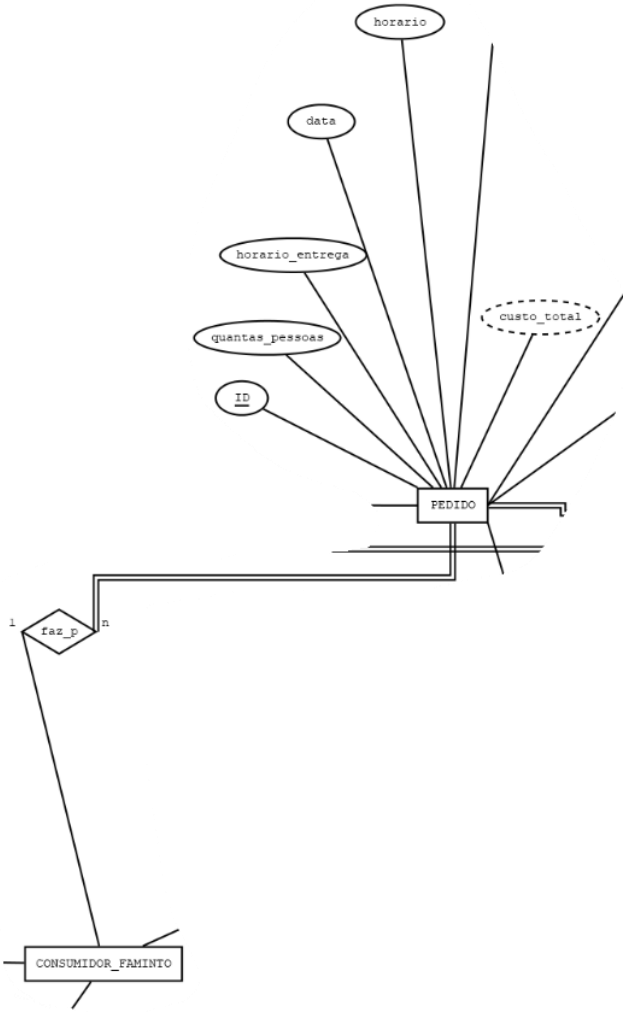

Figura 10 – Entidade pedido e relacionamento entre pedido e consumidor faminto

k) Para englobar várias pizzas dentro de um pedido foi feito o relacionamento "tem_p" com cardinalidade de muitos para muitos, pois um pedido pode possuir várias pizzas e um tipo de pizza pode estar em vários pedidos. Além disso a participação é total por parte da pizza, pois todo pedido deve conter pelo menos uma pizza. Para identificar a opção de preparação foi criado um atributo composto no relacionamento, esse atributo é um atributo composto por massa, borda e quantidade de molho. O motivo pelo qual o atributo está no relacionamento é que cada pedido pode ter uma opção de preparação distinta de uma determinada pizza. Essa estratégia permite vincular várias pizzas de tipo diferente para um determinado pedido, contudo não permite duas pizzas idênticas no mesmo pedido. Essa escolha foi adotada pois é muito improvável a pessoa decidir pedir duas pizzas idênticas. Uma maneira de solucionar esse problema seria implementando uma agregação entre as entidades "pedido" "pizza" e o relacionamento "tem_p", gerando

uma chave primária a mais que raramente seria utilizada. Ou criar um novo atributo quantidade no relacionamento pedido, contudo caso fosse criado esse novo atributo quantidade não seria mais possível apontar para qual pizza dentro daquela quantidade desejaria colocar o ingrediente extra.

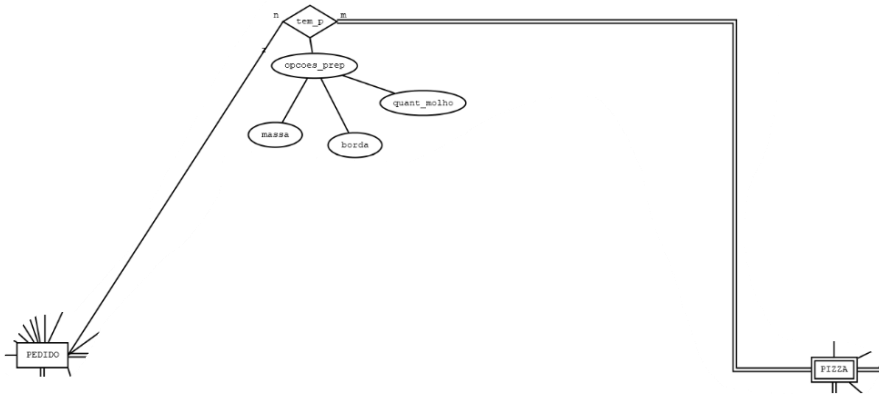

Figura 11 – Relacionamento que inclui pizza em pedido

l) Para incluir os acompanhamentos no pedido foi criado um relacionamento "tem_a" com cardinalidade muita para muitos, pois um pedido pode conter vários acompanhamentos, um tipo de acompanhamento pode estar em vários pedidos. Além disso a quantidade do acompanhamento pode ser informada como um atributo desse relacionamento. O motivo pelo qual o atributo está no relacionamento é que cada pedido pode ter uma quantidade distinta de um determinado acompanhamento.

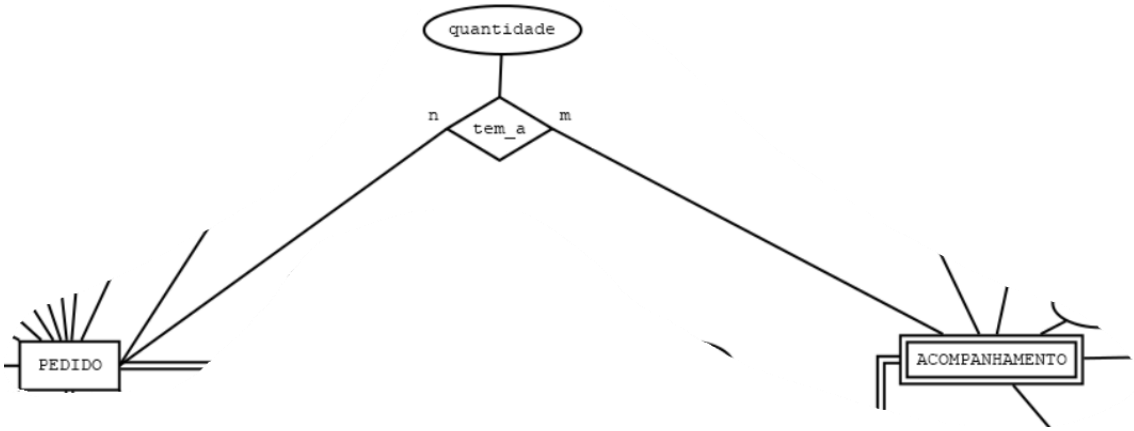

Figura 12: Relacionamento que inclui acompanhamento em pedido

m) Para associar um pedido com um ingrediente extra e com uma pizza, foi criado o relacionamento ternário “para” com cardinalidade muitos para muitos para muitos, pois um pedido com um ingrediente extra pode pertence a muitas pizzas, e uma pizza com um ingrediente extra pode pertencer a vários pedidos e um pedido com uma pizza pode ter vários ingredientes extras. Como é opcional conter ingrediente extra em um pedido a participação é parcial no pedido.

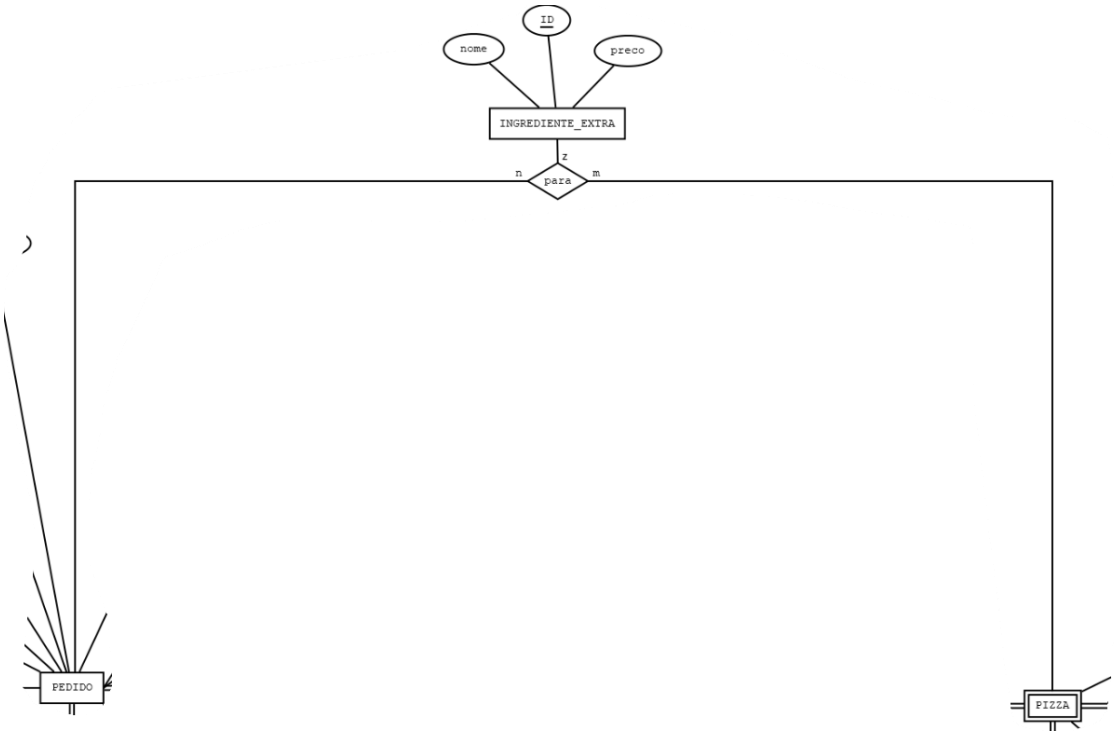

Figura 13 – Relacionamento de pedido com ingrediente extra e de pizza com ingrediente extra.

n) A entidade ingrediente extra possui os atributos nome, preço e chave primária ID.

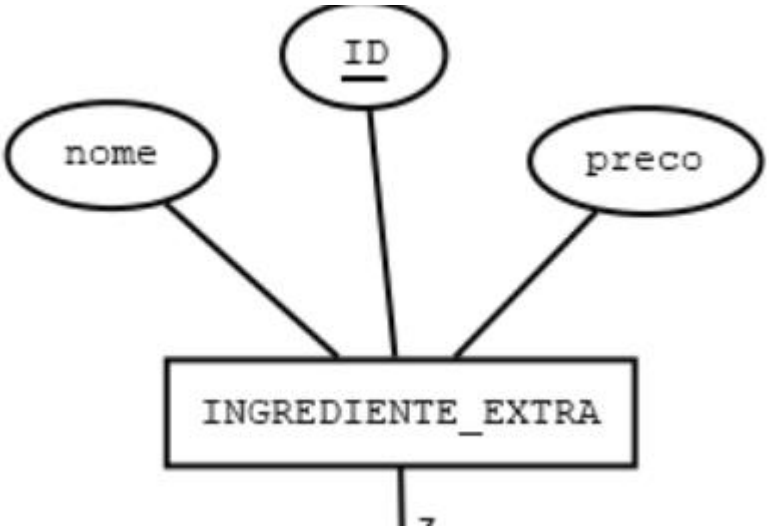

Figura 14 - Entidade ingrediente extra

o) Como o pedido de entretenimento herda atributos do pedido normal esse foi representado como uma subclasse de pedido e contendo os atributos tipo e duração que não estavam presentes no pedido. Além disso existe dois relacionamentos, um entre pedido e consumidor faminto e outro entre pedido entretenimento e consumidor faminto. A cardinalidade de ambos é 1 para n, pois um pedido/pedido entretenimento pertence a um único consumidor e um consumidor pode fazer vários pedido/pedido entretenimento, ademais ambos apresentam participação total em pedido/pedido entretenimento, pois todo pedido/pedido entretenimento deve possuir pelo um consumidor.

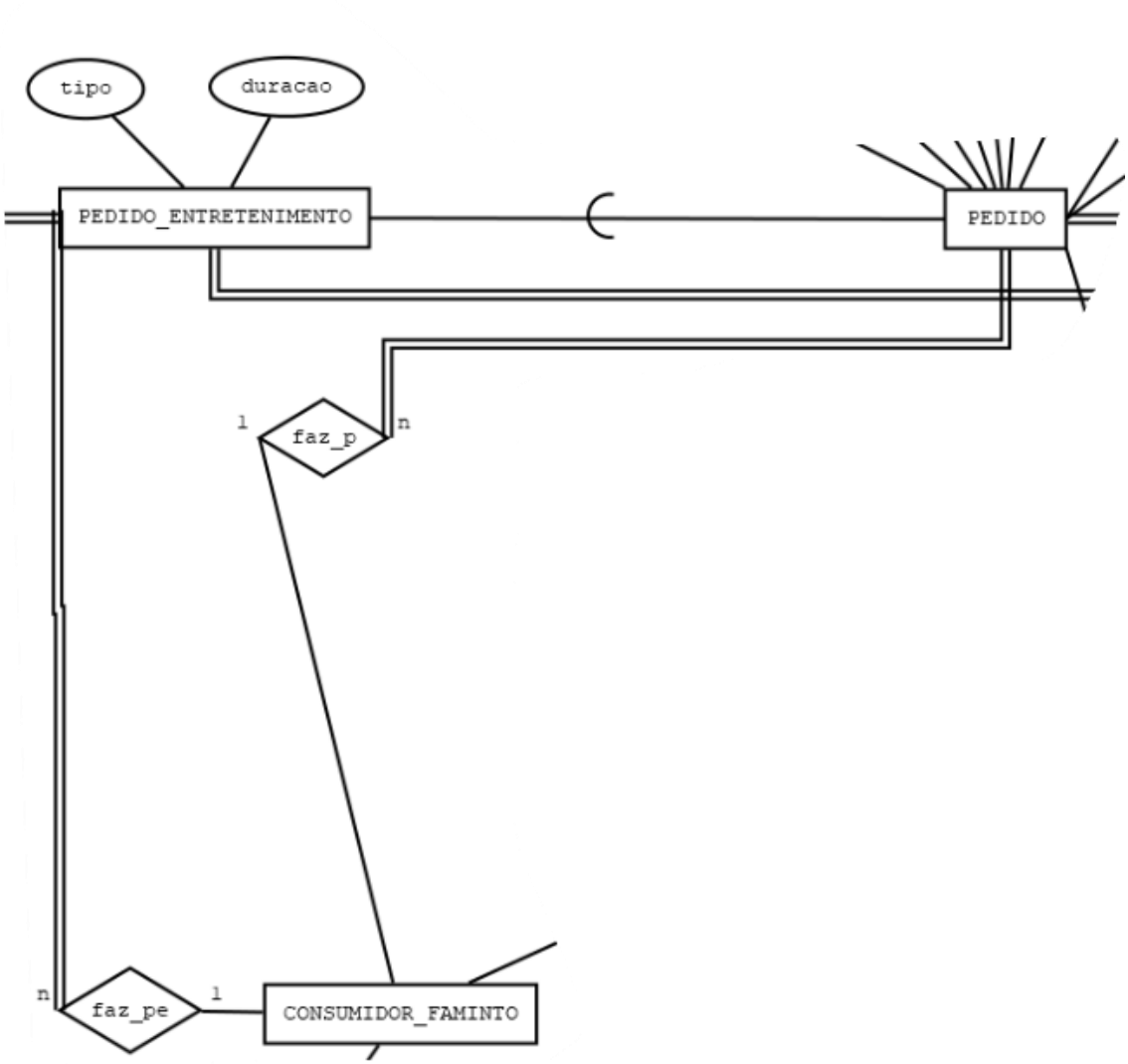

Figura 15 – Entidade pedido entretenimento com relacionamentos entre consumidor e pedido/pedido entretenimento.

p) Foi adicionado um atributo derivativo à entidade pedido, esse valor será calculado a partir da soma dos valores das pizzas escolhidas + ingredientes extras + acompanhamentos + entretenimento/entrega.

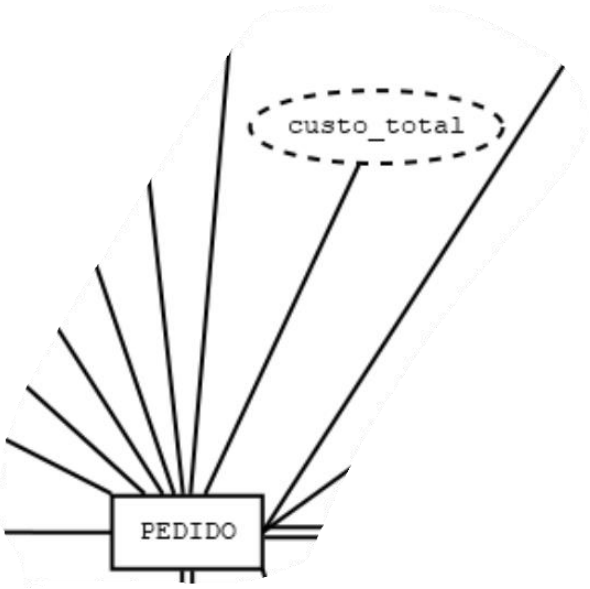

Figura 16 – Atributo derivativo custo_total

q) Foi criada uma entidade animador que é filha da entidade usuário, e foram adicionados três atributos a essa entidade (nome artístico, biografia e preço).

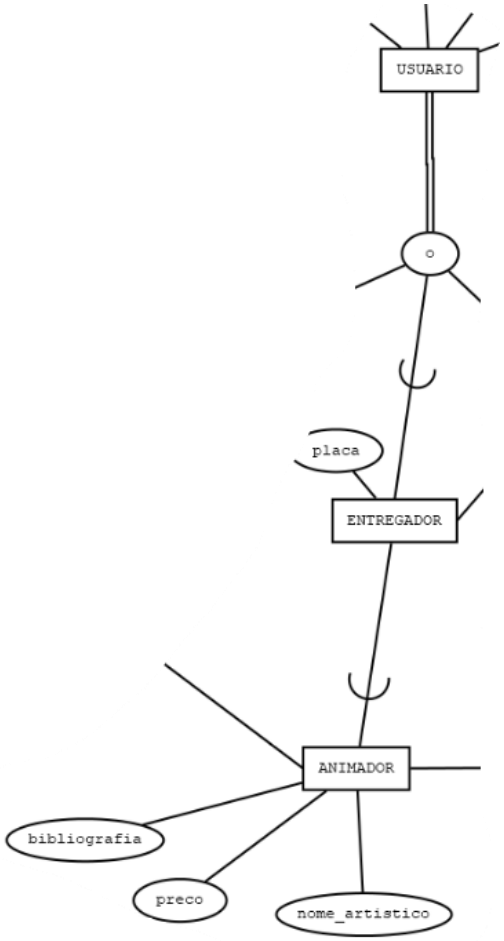

Figura 17 – Animador, entidade filha de usuário

r) Foi adicionado um relacionamento atende, entre animador e pedido entretenimento com participação total pois todo pedido de entretenimento precisa de um animador. Além disso, possui cardinalidade n para 1, pois um pedido entretenimento pode ter apenas um animador e um animador pode atender vários pedidos.

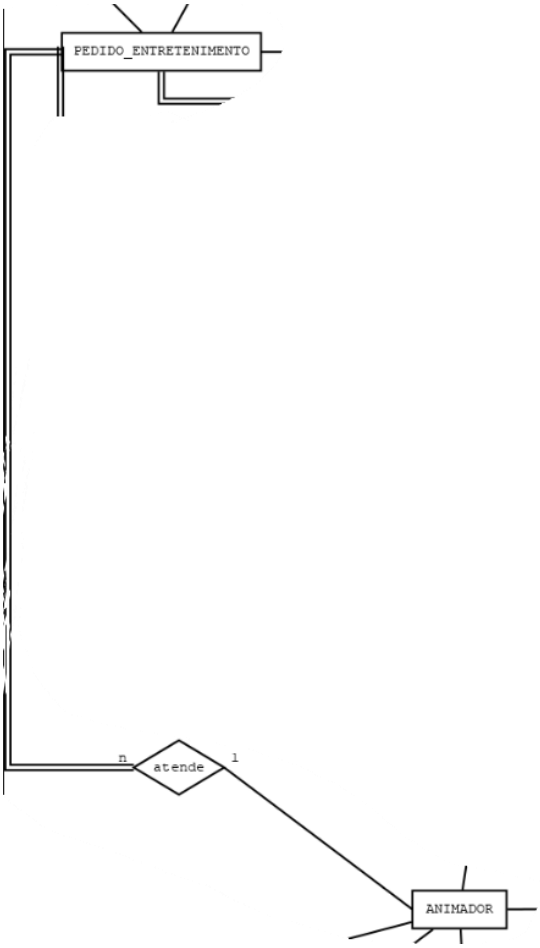

Figura 18 – Relacionamento entre pedido entretenimento e animador

s) O relacionamento trabalha entre animador e pizzaria apresenta cardinalidade muitos para muitos, pois um animador pode trabalhar para várias pizzarias e uma pizzaria pode ter vários entregadores que trabalham nela. Além disso as participações são parciais pois é possível ter um animador cadastrado no sistema e não está trabalhando, e além disso é possível ter uma pizzaria sem animadores. Ademais, nesse relacionamento há um atributo multivalorado para armazenar os dias da semana que o animador tem disponibilidade.

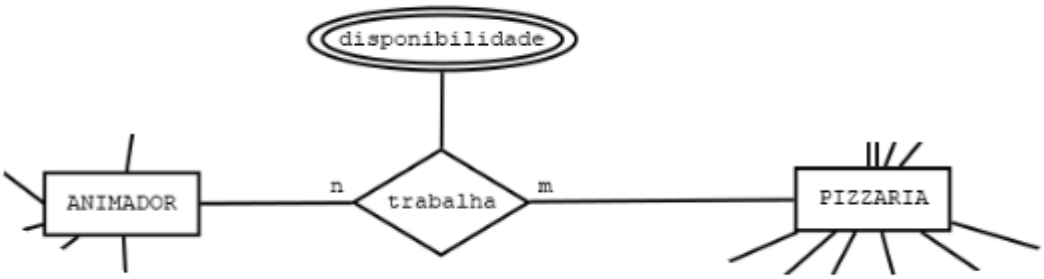

Figura 19 – Relacionamento trabalho entre animador e pizzaria

t) Uma nova entidade foi criada sendo filha da entidade usuário e pai da entidade animador, essa entidade deve chamada de entregador deve registrar a placa do veículo em um atributo. Além disso existe um relacionamento de entrega entre o entregador e o pedido convencional, esse relacionamento é de cardinalidade 1 para n, pois um pedido pode ser entregue por apenas um entregador e um entregador pode entregar vários pedidos. Além disso a participação é parcial para as duas entidades, pois nem todo pedido precisa ser entregue e nem todo entregador precisa estar entregando pedidos.

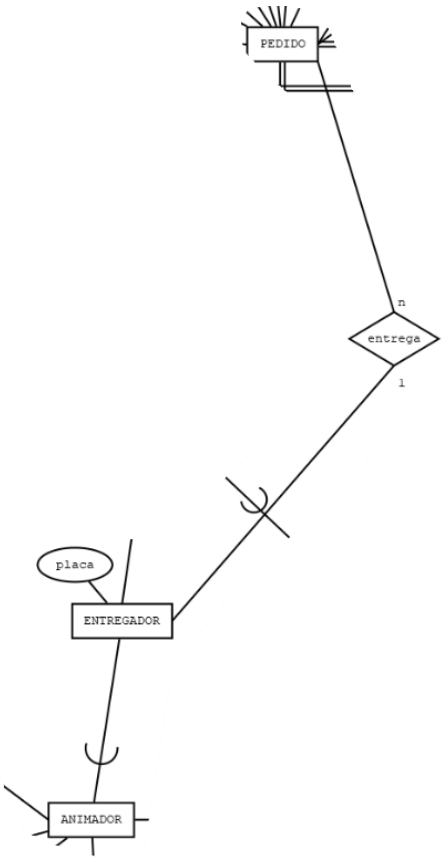

Figura 20 – Nova entidade entregador e novo relacionamento entrega

u) Foi criado uma nova entidade chamada de forma_pagamento que irá registrar o nome da forma de pagamento, identificado pela chave primária nome(Exemplos: “cheque”, “PIX”, “VALES”). O troco (caso seja escolhido a forma de pagamento dinheiro) e a bandeira do cartão (caso seja escolhida a forma de pagamento cartão) foram registrados nos relacionamentos paga e paga_e a partir dos atributos troco e bandeira. Esses atributos estão no relacionamento pois eles dependem da instância do relacionamento entre as duas entidades.

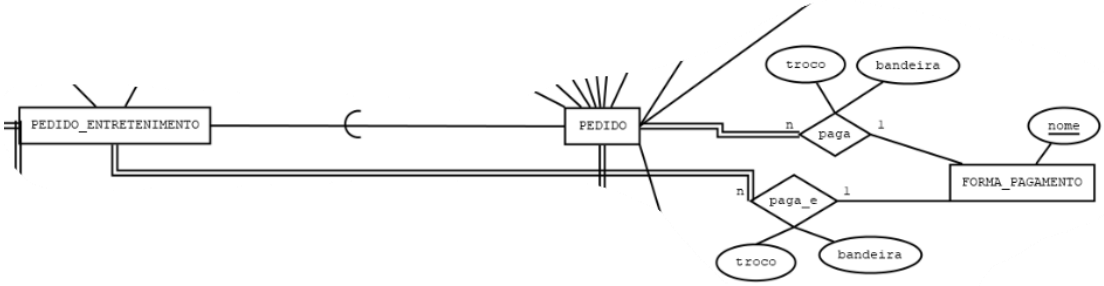

Figura 21 – Entidade forma de pagamento com relacionamentos

3- ESQUEMA RELACIONAL:

ENTREGADOR(ID, data_nasc, nome, endereco, placa);

DONO_DE_NEGOCIO(ID, data_nasc, nome, endereco, linkedin);

CONSUMIDOR_FAMINTO(ID, data_nasc, nome, endereco, endereco_entrega);

ANIMADOR(ID(ENTREGADOR.ID), bibliografia, preco, nome_artistico);

FORMA_PAGAMENTO(nome);

PEDIDO(ID, troco, bandeira, quantas_pessoas, horario_entrega, horario, data, custo_total, ID_F(CONSUMIDOR_FAMINTO.ID), nome_fp(FORMA_PAGAMENTO.nome));

PEDIDO_ENTRETENIMENTO(ID(PEDIDO.ID), tipo, duracao, ID_E(ANIMADOR.ID), ID_F(CONSUMIDOR_FAMINTO.ID));

ENTREGA(ID_P(PEDIDO.ID), ID_E(ENTREGADOR.ID));

PIZZARIA(CNPJ, CEP, endereco, website, telefone, horario_abertura, horario_fechamento, ID(DONO_DE_NEGOCIO.ID));

TRABALHA(ID_E(ANIMADOR.ID), CNPJ(PIZZARIA.CNPJ));

DISPONIBILIDADE(dia, ID_E(TRABALHA.ID), CNPJ(TRABALHA.CNPJ));

CATEGORIA(codigo_num_filho, descricao, codigo_num_pai(CATEGORIA.codigo_num_filho));

PIZZA(nome, CNPJ(PIZZARIA.CNPJ), preco, codigo_num_filho(CATEGORIA.codigo_num_filho));

TEM_P(ID(PEDIDO.ID), CNPJ(PIZZA.CNPJ), nome_p(PIZZA.nome), quant_molho, massa, borda);

ACOMPANHAMENTO(codigo, CNPJ(PIZZARIA.CNPJ), descricao, nome, tipo, preco);

TEM_A (ID(PEDIDO.ID), codigo(ACOMPANHAMENTO.codigo), CNPJ(ACOMPANHAMENTO.CNPJ), quantidade);

INGREDIENTE_EXTRA(ID, nome, preco);

PARA (ID_I(INGREDIENTE_EXTRA.ID), CNPJ(PIZZA.CNPJ), ID_P(PEDIDO.ID), nome_p(PIZZA.nome));

JUSTIFICATIVAS:

ABSTRAÇÃO DE GENERALIZAÇÃO E ESPECIALIZAÇÃO DE USUÁRIO PARA CONSUMIDOR FAMINTO, ENTREGADOR E DONO DE NEGÓCIO:

Foram construídas três tabelas, sendo que cada tabela carrega os atributos do USUARIO do modelo conceitual(ID, data_nasc, nome, endereco). O motivo por trás desta escolha está: facilidade para consultar a tabela "CONSUMIDOR_FAMINTO", à qual estará sempre sendo consultada para novos pedidos (serviço principal do aplicativo). Desta forma não é necessário consultar toda a tabela "USUARIO" (caso ela existisse), além de não precisar realizar uma consulta dupla para identificar o nome do consumidor. A desvantagem por trás deste método: será necessário a construção de um gatilho que fiscalize edições em atributos do usuário para garantir a integridade dos usuários que pertençam a mais de uma dessas subclasses.

ABSTRAÇÃO DE GENERALIZAÇÃO E ESPECIALIZAÇÃO DE ENTREGADOR PARA ANIMADOR:

Para animador, foi construído uma nova tabela "ANIMADOR", e esta tabela aponta para a tabela "ENTREGADOR", herdando sua chave primária (*Primary Foreing Key*), ao herdar a chave primária da outra tabela, estamos herdando os atributos por efeito indireto. Vantagens: evitar valores nulos, facilidade de manutenção das tabelas ANIMADOR e ENTREGADOR. Desvantagens: para maiores informações é necessário realizar duas consultas.

ABSTRAÇÃO DE GENERALIZAÇÃO E ESPECIALIZAÇÃO DE PEDIDO PARA PEDIDO ENTRETENIMENTO:

Para pedido entretenimento, foi construído uma nova tabela “PEDIDO_ENTRETENIMENTO”, e esta tabela aponta para a tabela “PEDIDO”, herdando sua chave primária (*PFK*) , ao herdar a chave primária da outra tabela, estamos herdando os atributos por efeito indireto. Vantagens: evitar valores nulos, facilidade de manutenção das tabelas PEDIDO e PEDIDO_ENTRETENIMENTO. Desvantagens: para maiores informações é necessário realizar duas consultas.

RELACIONAMENTOS N:1 COM PARTICIPAÇÃO TOTALITÁRIA EM N:

A chave estrangeira (FK) estará localizada na entidade N pois não existe risco de possuir valores nulos em suas tuplas.

Exemplos no modelo conceitual:

- OFERECE_A;

-OFERECE_P;

-CLASSIFICA;

-PAGA;

-PAGA_E;

-FAZ_P;

-FAZ_PE;

-ATENDE;

-POSSUI.

Nos casos onde o relacionamento possui atributo, estes atributos passaram a fazer parte dos atributos da entidade de cardinalidade N.

RELACIONAMENTOS N:1 COM PARTICIPAÇÃO PARCIAL:

Para evitar valores nulos em tuplas, uma nova tabela foi criada, sendo sua chave primária o valor da chave primária da entidade do lado N. O motivo pelo qual não é necessário considerar a outra chave estrangeira como chave primária, dá-se à sua cardinalidade.

Exemplos de ocorrências no modelo conceitual:

-ENTREGA.

RELACIONAMENTOS N:1 COM PARTICIPAÇÃO PARCIAL EM AUTO JUNÇÃO:

Neste caso, o atributo filho referência o atributo pai a partir de uma chave estrangeira.

RELACIONAMENTOS TERNÁRIO MUITOS PARA MUITOS PARA MUITOS:

Nesse caso como o relacionamento foi ternário e de muitos para muitos para muitos foi criada uma nova tabela contendo a chave primária das três entidades que se relacionam. Desta forma conseguimos garantir unicidade evitando valores nulos.

RELACIONAMENTOS N:M:

Neste caso criamos uma nova tabela sendo sua chave primária formada pelas chaves primárias estrangeiras. Caso tenham atributos, eles entram nessa tabela.

ATRIBUTOS MULTIVALORADOS:

Neste caso criamos uma nova tabela, sendo sua chave primária formada pela chave primária estrangeira e o atributo diferenciador. A escolha deve-se à alta variabilidade na quantidade de atributos evitando nulos nas tuplas.

Exemplo de ocorrência no modelo conceitual:

-Disponibilidade no relacionamento “trabalha”.

OBSERVAÇÕES FINAIS:

- No esquema conceitual nos relacionamentos “PAGA” e “PAGA_E”, ao converter para o mapeamento, esquema relacional, os atributos do relacionamento passou a pertencer à entidade pedido devido à participação total.

- TEM_P: opções de preparação foi descrita por três parametros, ao invés de ser multivalorado, por existir somente três possibilidades.

4- CRIAÇÃO DO BANCO DE DADOS

Cada uma das relações/tabelas presentes na secção anterior do esquema relacional será implementada como uma tabela utilizando o comando CREATE TABLE. A implementação da criação dessas tabelas está no arquivo “SQL_DDL.txt”.

5- ESPECIFICAÇÃO DE CONSULTAS EM SQL

OPERAÇÕES DE INSERÇÃO

A inserção de cada uma das tabelas foi feita de tal forma que as entidades que não possuíam chave estrangeira foram implementadas primeiro. E só após adicionarmos essas

primeiras relações que não possuem chave estrangeira é possível criar as próximas relações que utilizam essas chaves primários dessas tabelas já criadas como chave estrangeira. Todas essas inserções podem ser visualizadas no arquivo “SQL_DML_INSERTS.txt”.

CONSULTAS

Todas as seguintes consultas podem ser visualizadas no arquivo “SQL_DML_CONSULTAS.txt”.

1) Titulo:

Caso o aplicativo queira checar os dados pessoais de uma pessoa pelo nome:

Justificativa:

Quando o cliente quer editar os dados pessoais, ou o aplicativo quer consultar os dados pessoais de um cliente, ele faz uma consulta.

Resolução:

```
--Caso o cliente escolha CONSUMIDOR FAMINTO:
select CF.nome, CF.data_nasc, CF.endereco , CF.endereco_entrega
   from CONSUMIDOR_FAMINTO CF
   where CF.nome LIKE '%Apolinario%';

--Caso o cliente escolha DONO_DE_NEGOCIO:
select DN.nome, DN.data_nasc, DN.endereco , DN.linkedin
   from DONO_DE_NEGOCIO DN
   where DN.nome LIKE '%Apolinario%';
```

```
--Caso o cliente escolha ENTREGADOR:
select EN.nome, EN.data_nasc, EN.endereco, EN.placa
   from ENTREGADOR EN
   where (EN.nome LIKE 'Rafael Costa%');


--Caso o cliente escolha ANIMADOR:
select EN.nome, AN.nome_artistico, AN.bibliografia , AN.preco
   from ENTREGADOR EN, ANIMADOR AN
   where     (EN.ID = AN.ID) AND
         (EN.nome LIKE 'Rafael Coelho%');
```

2) Titulo:

Qual a bandeira do cartão mais utilizada no ano de 2015?

Justificativa:

Bandeiras distintas significam percentuais de comissão distintos. É uma ideia legal, que o aplicativo busque parcerias com bandeiras mais utilizadas. O ano em específico seria para colocar validade na informação coletada. Neste caso, o aplicativo usaria a informação para minimizar custos em 2016, por exemplo.

Resolução:

```
select pe.bandeira, count(pe.bandeira)
         from PEDIDO PE
         where (pe.bandeira is not null) AND
```

```
                (pe.data > '2014-12-30') AND
                (pe.data < '2016-01-01')
    group by pe.bandeira
    order by pe.bandeira desc
    limit 1;
```

3) Titulo:

Quais os quatro sabores de pizza mais pedido nas pizzarias do dono que começa por "Jose da Sil..."?

Justificativa:

Essa consulta permite o dono da pizzaria consultar suas pizzas mais vendidas apenas digitando seu nome ou parte dele. Essencial para saber qual pizza é responsável por a maior parte das vendas.

Resolução:

```
SELECT DISTINCT(TP.nome_p), COUNT(TP.nome_p)
    FROM TEM_P TP, PIZZARIA PIZ, DONO_DE_NEGOCIO DN, PIZZA PI
    WHERE DN.nome LIKE 'Jose da Sil%'
    AND DN.ID=PIZ.ID
    AND PIZ.CNPJ=PI.CNPJ
    AND TP.CNPJ=PI.CNPJ
    AND TP.nome_p=PI.nome
    GROUP BY TP.nome_p
    ORDER BY COUNT(TP.nome_p) DESC
    LIMIT 4;
```

4) Titulo:

Quais os cinco clientes mais rentáveis?

Justificativa:

O aplicativo pode buscar manter uma lista de clientes “rentaveis”. Essa lista é importante pois ajuda na manutenção dos clientes: os clientes mais rentáveis podem ganhar cupons e vales. Cupons e vales ajudam a manter a fidelidade do cliente.

Resolução:

```
SELECT CF.nome, CF.ID, SUM(custo_total)
    FROM PEDIDO PE, CONSUMIDOR_FAMINTO CF
    WHERE CF.ID = PE.ID_F
    GROUP BY CF.nome, CF.ID
    ORDER BY SUM(custo_total) DESC
    LIMIT 3;
```

5) Titulo:

Quais as pizzas (com sua categoria) e os preços que são servidas na pizzaria de CNPJ x?

Justificativa:

Essa consulta é o cardápio da pizzaria de CNPJ X, que será mostrada ao usuário ao consultar a pizzaria.

No exemplo, X é o CNPJ 15.345.658/0001-42

Resolução:

```
CREATE VIEW CARDAPIO_PIZZAS_15345658000142 AS
select PI.nome, CAT.descricao, PI.preco
```

```
    from PIZZA PI, CATEGORIA CAT
    where (PI.CNPJ = '15.345.658/0001-42') AND
                (PI.codigo_num_filho = CAT.codigo_num_filho);

select * from CARDAPIO_PIZZAS_15345658000142;
```

6) Titulo:

Qual pizzaria apresenta maior faturamento?

Justificativa:

Essa informação é importante para que o aplicativo faça manutenção dos usuários donos. As pizzarias que vendem bastante pelo aplicativo, merecem aparecer primeiro na busca por pizzarias.

Resolução:

```
SELECT TP.CNPJ, SUM(custo_total)
    FROM PEDIDO PE, TEM_P TP
    WHERE (PE.ID = TP.ID)
    GROUP BY TP.CNPJ
    ORDER BY SUM(custo_total) DESC
    LIMIT 1;
```

7) Titulo:

Qual a média do tempo de espera nos estabelecimentos?

Justificativa:

O cliente pode saber a média de tempo que o estabelecimento demora para entregar o pedido antes de fazê-lo.

Resolução:

```
SELECT TP.CNPJ, AVG(PE.horario_entrega - PE.horario) AS tempo_de_espera
    FROM PEDIDO PE, TEM_P TP
    WHERE PE.ID = TP.ID
    GROUP BY TP.CNPJ;
```

8) Titulo:

Listar pizza, acompanhamento, ingredientes e animador com seu determinado custo para um pedido. Gerando um recibo.

OBS: É necessário que a SP - custo total tenha rodado anterior à esta consulta

OBS2: Neste caso, o recibo foi feito para um exemplar pedido de ID = 8.

Justificativa:

Recibo: resumo de todos os itens com seus determinados preços

Resolução:

```
select PI.nome as nome, PI.preco as preco
    from PIZZA PI, TEM_P TP, PEDIDO PE
    where  (PI.CNPJ = TP.CNPJ) AND--juncao PIZZA com tem_p
                (PI.nome = TP.nome_p) AND
                (PE.ID = TP.ID) AND--juncao tem_p com pedido
                (PE.ID = 8)
```

```
union
select AC.nome ||'(x '|| TA.quantidade ||')' as nome, TA.quantidade*AC.preco as preco
        from ACOMPANHAMENTO AC, TEM_A TA, PEDIDO PE
        where (AC.CNPJ = TA.CNPJ) AND--juncao ACOMPANHAMENTO com tem_A
                        (AC.codigo = TA.codigo) AND
                        (PE.ID = TA.ID) AND--juncao tem_A com pedido
                        (PE.ID = 8)
union
select IE.nome ||'>>'|| PI.nome as nome, IE.preco as preco
        from INGREDIENTE_EXTRA IE, PARA PA, PEDIDO PE, PIZZA PI
        where  (IE.ID = PA.ID_I) AND--juncao INGREDIENTE_EXTRA
                        (PE.ID = PA.ID_P) AND--juncao tem_A com pedido
                        (PI.nome = PA.nome_p) AND
                        (PI.CNPJ = PA.CNPJ) AND
                        (PE.ID = 8)
union
select 'Animador -'|| AN.nome_artistico as nome, AN.preco*PEE.duracao as preco
        from PEDIDO PE, PEDIDO_ENTRETENIMENTO PEE, ANIMADOR AN
        where  (PEE.ID = PE.ID) AND
                        (PEE.ID_E = AN.ID) AND
                        (PE.ID = 8)
union
select 'Total' as nome, PE.custo_total as preco
        from PEDIDO PE
```

```
        where (PE.ID = 8)
order by preco asc;
```

9) Titulo:

O ranking das formas de pagamento que geraram maior fluxo de caixa no primeiro semestre de 2015.

Justificativa:

Um indicativo para o aplicativo saber qual a tendência de pagamento que gera maior faturamento, e desta forma, facilitar ou melhorar o serviço que mais agrega resultado.

Resolução:

```
SELECT PE.nome_fp, SUM(PE.custo_total)
FROM PEDIDO PE
WHERE (PE.nome_fp IS NOT NULL) AND
                (PE.data >= '2015-01-01') AND
                (PE.data < '2015-07-01')
GROUP BY PE.nome_fp
ORDER BY SUM(PE.custo_total) DESC;
```

10) Titulo:

Quais os nomes dos entregadores que atendem aos finais de semana?

Justificativa:

As pizzarias podem entrar em contato com o entregador para atender a demanda das entregas.

Resolução:

```
SELECT DISTINCT(EN.nome)
```

FROM DISPONIBILIDADE DI, ENTREGADOR EN, TRABALHA TR, ANIMADOR AN

WHERE (DI.ID_E = TR.ID_E) AND (TR.ID_E = AN.ID) AND (AN.ID = EN.ID) AND (

(DI.dia = 'sabado') OR (DI.dia = 'domingo'));

GATILHO E PROCEDIMENTO ARMAZENADO

PROCEDIMENTO ARMAZENADO (SP)

Como na relação PEDIDO existe um atributo derivativa chamado “custo_total” que é calculado a partir da soma dos valores das pizzas, acompanhamentos, ingredientes extras e entretenimento selecionado, de um determinado pedido. Assim foi feito a escolha de implementar um procedimento armazenado para fazer uma automatização desse cálculo que iria requerer várias consultas, dessa forma, esse SP faz o acesso de todas as demais tabelas necessárias para fazer esse cálculo para um determinado pedido e depois faz um UPDATE no atributo “custo_total” da tabela PEDIDO. A implementação desse SP denominado de CALCULAR_CUSTO_TOTAL está no arquivo “SQL_DML_SP.txt”.

Como ele foi implementado com o parâmetro de entrada ID do pedido é possível gerar o cálculo de um pedido isolado, passando apenas esse ID como parâmetro do SP, ou atualizar toda a tabela de uma vez só, passando todos os IDs da tabela PEDIDO a partir da escrita do comando da seguinte maneira “SELECT CALCULAR_CUSTO_TOTAL (PEDIDO.ID) FROM PEDIDO;”.

Dessa forma o SP implementado recebe como a entrada o ID de um pedido, retorna o custo_total desse pedido e o atribui na tabela pedido para o ID solicitado.

GATILHO

Para contornar o obstáculo gerado pela maneira como a hierarquia dos usuários foi implementa (sem gerar uma relação para a entidade mãe), foi feito a escolha de implementar três gatilhos para sincronizar a atualização dos atributos da entidade mãe.

Por exemplo, um consumidor faminto que é também dono de negócio e/ou animador, caso ele troque de endereço que é um dos atributos do usuário esse valor deve ser atualizado nas demais tabelas.

Um gatilho para CONSUMIDOR_FAMINTO, outro para ENTREGADOR e outro para DONO_DE_NEGOCIO , cada um desses gatilho são disparados caso haja um UPDATE no nome, data_nasc ou endereço e atualiza para as demais relações, assim mantendo a integridade de dados da entidade mãe que não apresenta uma tabela. Esse código pode ser visualizado no arquivo “SQL_DML_TRIGGER.txt”.

Para evitar um loop eterno de um gatilho acionando outro, a função disparada pelo gatilho primeiro checa se é necessário fazer a atualização, e só caso o novo valor for diferente do atual ele faz a alteração.