Estudo de caso PIZZARIA

1. LEVANTAMENTO DE REQUISITOS

Uma pizzaria de tele-entrega apresenta um cardápio composto por diversos tipos de pizza, cujos dados são: número do item, nome da pizza, ingredientes e preços das pizzas pequena, média e grande.

Na pizzaria trabalham funcionários que emitem pedidos de pizzas. Cada pedido possui um número e uma data de emissão, além do nome, telefone e endereço do cliente que solicitou o pedido.

Um pedido é emitido por apenas um funcionário. Um pedido solicita uma ou mais pizzas do cardápio, informando tamanhos (pequena, média ou grande) e quantidades desejados de cada pizza. O cliente pode desejar retirar um ou mais ingredientes de alguma pizza solicitada.

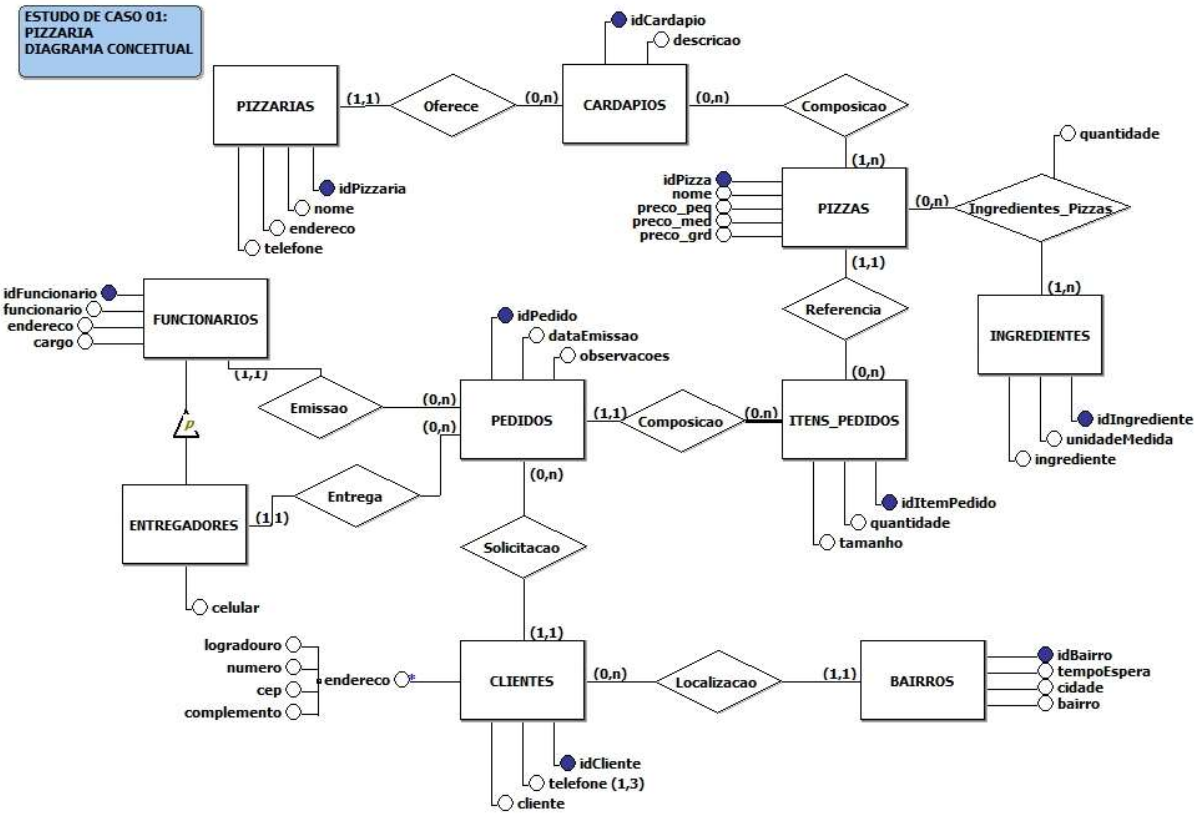
Existem funcionários que são entregadores, ou seja, são responsáveis pela entrega de um ou mais pedidos. Deve-se saber o número do telefone celular destes funcionários para um eventual contato durante uma entrega.

Um pedido destina-se a um bairro. Para cada bairro existe um tempo máximo de espera para a entrega de um pedido.

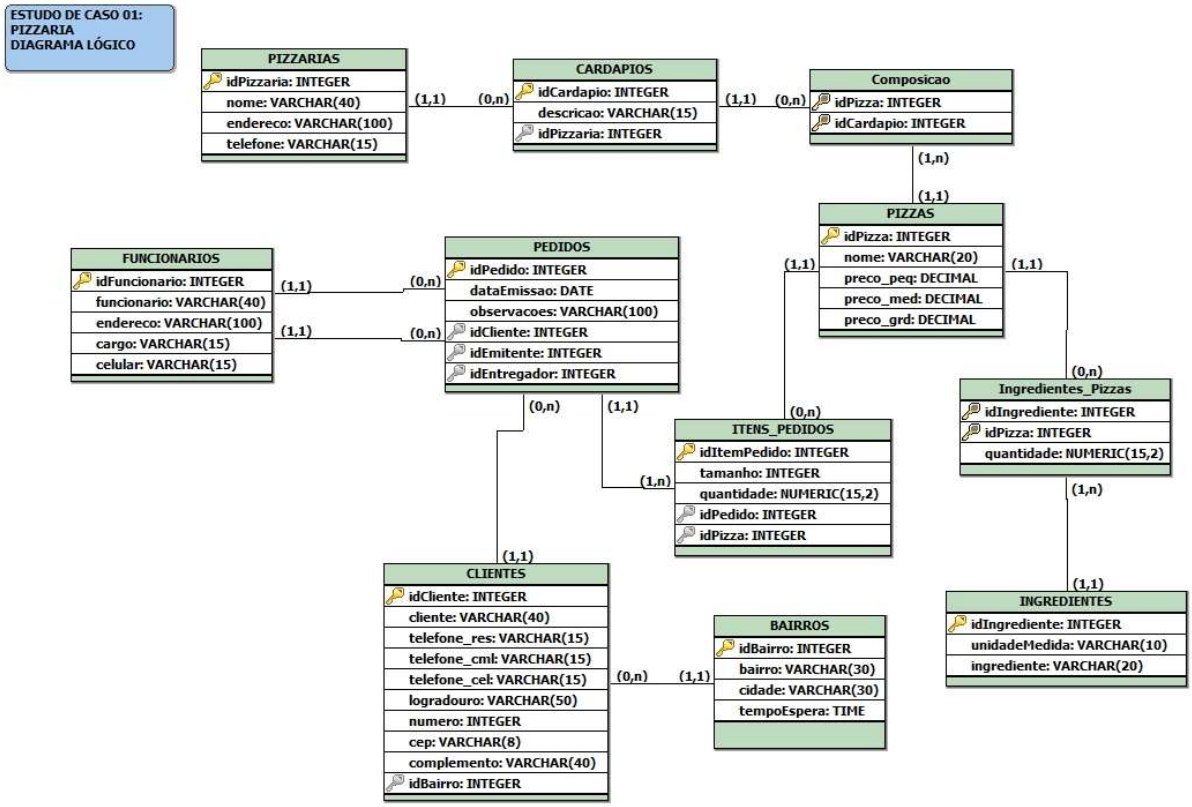
Você é livre para definir outros atributos que julgar relevantes.

Principais consultas/relatórios:
- Listar as Pizzas de cada cardápio
- Mostrar a receita/composição de uma pizza
- Listar os pedidos por data / por bairro / por entregador
- Consultar pedidos por cliente
- Qual o cliente que mais encomendou pizzas?
- Qual o entregador que fez mais entregas no mês?

2. MODELAGEM CONCEITUAL PIZZARIA



3. MODELAGEM LÓGICA PIZZARIA



4. PROJETO FÍSICO PIZZARIA

Exemplos de *Scripts* SQL (para o SGBD *PostgreSQL*)
Os exemplos abaixo apresentam a sintaxe do SQL e o *script* para o BD da Pizzaria. Em alguns casos, há mais de uma forma de escrever o *script*, por isso, preste atenção e procure entender as diferenças.

4.1. Criação do BANCO DE DADOS

```
CREATE DATABASE nome_do_banco;

CREATE DATABASE pizzarias;
```

4.2. Criação de TABELAS PRIMÁRIAS [SOMENTE COM CHAVE PRIMÁRIA]

```
CREATE TABLE nome_da_tabela (
      nome_campo      tipo_de_dado NOT NULL,
      outros_campos   tipo_de_dado NOT NULL,
      CONSTRAINT pk_nome_chave PRIMARY KEY (nome_campo)
);

CREATE TABLE pizzarias (
      idPizzaria SERIAL NOT NULL,
      nome       VARCHAR (40) NOT NULL,
      endereco   VARCHAR (100) NOT NULL,
      telefone   VARCHAR (15) NOT NULL,
      CONSTRAINT pk_pizzaria PRIMARY KEY (idPizzaria)
);
```

Ou também dessa forma:

```
CREATE TABLE pizzarias (
      idPizzaria SERIAL PRIMARY KEY NOT NULL,
      nome       VARCHAR (40) NOT NULL,
      endereco   VARCHAR (100) NOT NULL,
      telefone   VARCHAR (15) NOT NULL
);

CREATE TABLE pizzas (
      idPizza      SERIAL PRIMARY KEY NOT NULL,
      nome         VARCHAR (20) NOT NULL,
      preco_peq    DECIMAL NOT NULL,
      preco_med    DECIMAL NOT NULL,
      preco_grd    DECIMAL NOT NULL
);

CREATE TABLE ingredientes (
      idIngrediente   SERIAL PRIMARY KEY NOT NULL,
      ingrediente     VARCHAR (20) NOT NULL,
      unidadeMedida   VARCHAR (10) NOT NULL
);
```

```
CREATE TABLE bairros (
     idBairro    SERIAL PRIMARY KEY NOT NULL,
     bairro      VARCHAR (30) NOT NULL,
     cidade      VARCHAR (30) NOT NULL,
     tempoEspera TIME
);

CREATE TABLE funcionarios (
      idFuncionario   SERIAL PRIMARY KEY NOT NULL,
      funcionario     VARCHAR (40) NOT NULL,
      endereco        VARCHAR (100) NOT NULL,
      cargo           VARCHAR (15) NOT NULL,
      celular         VARCHAR (15)
);
```

4.3. Criação de TABELAS SECUNDÁRIAS [COM CHAVE ESTRANGEIRA]

```
CREATE TABLE nome_da_tabela (
      nome_campo      tipo_de_dado NOT NULL,
      outros_campos   tipo_de_dado NOT NULL,
      CONSTRAINT pk_nome_chave PRIMARY KEY (nome_campo),
      CONSTRAINT fk_nome_chave FOREIGN KEY (nome_campo)
                                    REFERENCES nome_tabela
);

CREATE TABLE cardapios (
      idCardapio   SERIAL NOT NULL,
      descricao    VARCHAR (15) NOT NULL,
      idPizzaria   INTEGER NOT NULL,
      CONSTRAINT pk_cardapio PRIMARY KEY (idCardapio),
      CONSTRAINT fk_cardapio_pizzaria FOREIGN KEY (idPizzaria)
                                        REFERENCES pizzarias
);
```

Ou também dessa forma:

```
CREATE TABLE cardapios (
      idCardapio SERIAL PRIMARY KEY NOT NULL,
      descricao  VARCHAR (15) NOT NULL,
      idPizzaria INTEGER NOT NULL REFERENCES pizzarias
                                              (idPizzaria)
);

CREATE TABLE clientes (
      idCliente     SERIAL PRIMARY KEY NOT NULL,
      cliente       VARCHAR (40) NOT NULL,
      telefone_res  VARCHAR (15),
      telefone_cml  VARCHAR (15),
      telefone_cel  VARCHAR (15),
      logradouro    VARCHAR (50),
      numero        INTEGER,
      cep           VARCHAR (8),
      complemento   VARCHAR (40),
      idBairro      INTEGER NOT NULL REFERENCES bairros
                                              (idBairro)
);
```

```
CREATE TABLE pedidos (
     idPedido     SERIAL PRIMARY KEY NOT NULL,
     dataEmissao  VARCHAR (15) NOT NULL,
     observacoes  VARCHAR (100),
     idCliente    INTEGER NOT NULL REFERENCES clientes
                                              (idCliente),
     idEmitente   INTEGER NOT NULL REFERENCES funcionarios
                                             (idFuncionario),
     idEntregador INTEGER NOT NULL REFERENCES funcionarios
                                              (idFuncionario)
);

CREATE TABLE itens_pedidos (
     idItemPedido  SERIAL PRIMARY KEY NOT NULL,
     tamanho       INTEGER NOT NULL,
     quantidade    NUMERIC (15,2),
     idPedido      INTEGER NOT NULL REFERENCES pedidos
                                              (idPedido),
     idPizza       INTEGER NOT NULL REFERENCES pizzas
                                              (idPizza)
);
```

4.4. Criação de TABELAS DE RELACIONAMENTO (N:M)

```
CREATE TABLE nome_da_tabela (
      campo_1    tipo_de_dado NOT NULL,
      campo_2    tipo_de_dado NOT NULL,
      CONSTRAINT pk_nome_chave PRIMARY KEY (campo1, campo_2),
      CONSTRAINT fk_nome_chave_1 FOREIGN KEY (campo1)
                                 REFERENCES nome_tabela_1
      CONSTRAINT fk_nome_chave_2 FOREIGN KEY (campo_2)
                                 REFERENCES nome_tabela_2
);

CREATE TABLE composicao (
      idPizza     INTEGER NOT NULL,
      idCardapio  INTEGER NOT NULL,
      CONSTRAINT pk_composicao
                 PRIMARY KEY (idPizza, idCardapio),
      CONSTRAINT fk_pizza FOREIGN KEY (idPizza)
                          REFERENCES pizzas,
      CONSTRAINT fk_cardapio FOREIGN KEY (idCardapio)
                                 REFERENCES cardapios
);
```

Ou também dessa forma:

```
CREATE TABLE composicao (
      idPizza     INTEGER NOT NULL
                        REFERENCES pizzas (idPizza),
      idCardapio  INTEGER NOT NULL
                        REFERENCES cardapios (idCardapio),
      CONSTRAINT pk_composicao
                 PRIMARY KEY (idPizza, idCardapio)
);

CREATE TABLE ingredientes_pizzas (
      idPizza        INTEGER NOT NULL
                       REFERENCES pizzas (idPizza),
      idIngrediente  INTEGER NOT NULL
                       REFERENCES ingredientes(idIngrediente),
      quantidade     NUMERIC (15,2) NOT NULL,
      CONSTRAINT pk_ingrediente_pizza
                 PRIMARY KEY (idPizza, idIngrediente)
);
```

5. MANIPULAÇÃO DE DADOS

A manipulação de dados é feita por meio de operações CRUD (*Create, Read, Update, Delete*) ou, simplesmente: *Inclusão*, *Consulta*, *Alteração* e *Exclusão*.

DML – Data Manipulation Language

5.1. Inclusão de dados - INSERT

```
INSERT INTO tabela (campo1, campo2 ...)
     VALUES (valor1, valor2...);
```

```
INSERT INTO pizzarias (nome, endereco, telefone)
     VALUES ('Minha Pizzaria', 'Rua da Pizza', '04712345678');
```

OBS.: O campo **id** não foi adicionado pois ele é gerado automaticamente pelo BD (tipo de dado SERIAL).
OBS.: Campos do tipo *string* ou *date* devem vir entre apóstrofo.

```
INSERT INTO pizzas (nome, preco_peq, preco_med, preco_grd)
     VALUES ('Calabresa', 10.00, 15.00, 20.00);
```

OBS.: Campos numéricos aparecem SEM ASPAS.

```
INSERT INTO cardapios (descricao, idPizzaria)
     VALUES ('Primavera', 1);

INSERT INTO composicao (idPizza, idCardapio)
     VALUES (1, 1);
```

OBS.: Os valores de chave-estrangeira devem ser iguais a valores existentes na tabela referenciada.

5.2. Consulta simples - SELECT

```
SELECT * FROM tabela;

SELECT campo1, campo2 ... FROM tabela;

SELECT *  FROM tabela
     WHERE campo = valor;

SELECT * FROM pizzarias;

SELECT nome, telefone
     FROM pizzarias;

SELECT idPizzaria, nome, telefone
     FROM pizzarias;

SELECT nome, telefone
     FROM pizzarias
     WHERE idPizzaria = 3;
```

5.3. Alteração de dados – UPDATE

```
UPDATE tabela  SET campo1 = valor1, campo2 = valor2 ...
          WHERE campo = valor;

UPDATE pizzarias SET nome = 'Pizza da Nona'
          WHERE idPizzaria = 1;
```

OBS.: Na cláusula WHERE são colocados um ou mais campos para seleção dos registros a serem alterados. Normalmente se usa a chave primária nessa condição.

5.4. Exclusão de registros – DELETE

```
DELETE FROM tabela
     WHERE campo = valor;

DELETE FROM pizzarias
     WHERE idPizzaria = 20;
```

OBS.: Cuidado ao usar o DELETE. Não esquecer de colocar a cláusula WHERE, indicando os campos de seleção do registro a ser excluído.

5.5. Consulta com mais de um campo seletor – SELECT, UPDATE e DELETE

```
SELECT *  FROM tabela
     WHERE campo1 = valor1
           AND campo2 = valor2;  // Pode usar AND ou OR

SELECT *  FROM pizzas
     WHERE preco_peq > 10.00
           AND preco_grd < 30.00;

SELECT *  FROM pizzas
     WHERE preco_peq >= 10.00
       AND preco_grd <= 30.00;

SELECT *  FROM pizzas
     WHERE preco_peq >= 10.00
        OR preco_grd <= 30.00;
```

5.6. Seleção envolvendo mais de uma tabela

use da seguinte forma:

```
SELECT campos
     FROM tabela1, tabela2 ...
     WHERE pk_tabela1 = fk_tabela2;

-- Seleciona todos os cardápios de todas as pizzarias
SELECT *
     FROM pizzarias, cardapios
     WHERE pizzarias.idPizzaria = cardapios.idPizzaria; --RI

-- Seleciona todas as pizzas, de todos os cardápios, de todas
as pizzarias
SELECT *
     FROM pizzarias, cardapios, composicao, pizzas
     WHERE pizzarias.idPizzaria = cardapios.idPizzaria
       AND cardapios.idCardapio = composicao.idCardapio
       AND pizzas.idPizza = composicao.idPizza;
```

Ou mais fácil,

```
SELECT *
     FROM pizzarias p, -- usando Alias
          cardapios ca,
          composicao co,
          pizzas pi
     WHERE p.idPizzaria = ca.idPizzaria
       AND ca.idCardapio = co.idCardapio
       AND pi.idPizza = co.idPizza;
```

Ou pesquise o uso de **JOIN**

```
-- Seleciona todos os cardápios de todas as pizzarias
SELECT *
     FROM pizzarias p
     INNER JOIN cardapios c ON p.idPizzaria = c.idPizzaria;

-- Seleciona todas as pizzas, de todos os cardápios, de todas
as pizzarias
SELECT *
     FROM pizzarias p
     INNER JOIN cardapios c ON p.idPizzaria = c.idPizzaria
     INNER JOIN composicao co ON c.idCardapio = co.idCardapio
     INNER JOIN pizzas pi ON pi.idPizza = co.idPizza;
```

5.7. Consulta com ORDENAÇÃO

```
SELECT *  FROM tabela
          ORDER BY campo1, campo2;

SELECT *
     FROM pizzarias pi,
          cardapios ca
     WHERE pi.idPizzaria = ca.idPizzaria
     ORDER BY pi.nome;
```