Guilherme Augusto de Macedo, Matheus Liberato Domingues da Silva, Victor Hugo Carlquist da Silva

Modelo de Banco de Dados para Gerenciamento de Pizzaria: Modelagem e Implementação

Campos do Jordão

2013

Guilherme Augusto de Macedo, Matheus Liberato Domingues da Silva, Victor Hugo Carlquist da Silva

Modelo de Banco de Dados para Gerenciamento de Pizzaria: Modelagem e Implementação

Trabalho final apresentado na disciplina de Banco de Dados II no quarto módulo do curso de Tecnologia em Análise e Desenvolvimento de Sistemas do IFSP-CJO.

Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - *campus* Campos do Jordão

Orientador: Paulo Giovani de Faria Zeferino

Campos do Jordão

2013

Guilherme Augusto de Macedo, Matheus Liberato Domingues da Silva, Victor Hugo Carlquist da Silva

Modelo de Banco de Dados para Gerenciamento de Pizzaria: Modelagem e Implementação

Trabalho final apresentado na disciplina de Banco de Dados II no quarto módulo do curso de Tecnologia em Análise e Desenvolvimento de Sistemas do IFSP-CJO.

Banca Examinadora
03 de dezembro de 2013

Prof. Paulo Giovani de Faria Zeferino
Orientador

Prof. Convidado 1
Convidado 1

Prof. Convidado 2
Convidado 2

Campos do Jordão
2013

RESUMO

Este trabalho tem como objetivo a criação de um projeto de banco de dados para gerenciar uma pizzaria. Seguindo algumas regras de negócio, foi elaborado um modelo conceitual, um modelo lógico e um modelo físico. Depois de implementado, o banco de dados foi alimentado com valores para teste. Esses valores de teste são a base para as consultas, visões, procedimentos armazenados e para os gatilhos. As ferramentas utilizadas para a construção do banco de dados foram: *br-Modelo* para o modelo conceitual, *Mysql Workbench* para a modelagem lógica, e o *SQL Server* em conjunto com o *SQL Server Management Studio* para a implementação do modelo físico.

Palavras-chaves: Banco de Dados. Modelagem Lógica. Modelagem Conceitual.

ABSTRACT

This work aims at creating a project database ăăăăto manage a pizzeria. Following a few rules of business, was prepared ăăăăa conceptual model, logical model and a physical model. Once deployed, ăăăăthe database was fed values for testing. These test values ăăăăare the basis for the queries, views, stored procedures and triggers. ăăăăThe tools used to build the database were: *br-Modelo* ăăăăto the conceptual model, MySQL Workbench for logic modeling, and ăăăăSQL Server in conjunction with SQL Server Management Studio to ăăăăimplementing the physical model. ăăăă ăăăăăăăăă ăăăă **Keywords**: Database. Logic modeling. Conceptual Modeling.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

INTRODUÇÃO

A solução proposta tem por objetivo a modelagem conceitual, lógica e física de um projeto de Banco de Dados para gerenciamento/automatização de uma pizzaria.

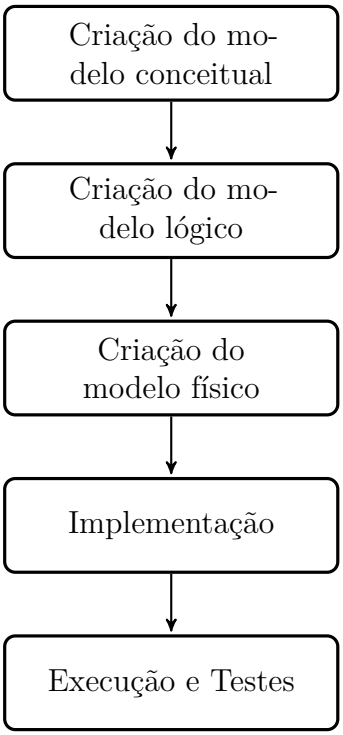
Depois de gerado o modelo físico, implementou-se a solução utilizando o *SQL Server Management Studio*. Com base nessa implementação, consultas, *views*, *triggers*, entre outras rotinas, foram criadas para fins de execução e testes.

Os capítulos seguintes estão divididos em Metodologia Proposta e as Regras de Negócio onde estão detalhadas as metodologias utilizadas para a execução o projeto, seguidos de explicações a respeito do modelo conceitual, lógico e físico. Posteriormente, as consultas realizadas são explicadas, assim como o restante das rotinas elaboradas.

1 METODOLOGIA PROPOSTA

Para a execução deste trabalho a metodologia foi dividida em três etapas: *Criação do modelo conceitual*, *Criação do modelo lógico*, *Criação do modelo físico*, *Implementação* e *Execução e Testes*. A figura 1 ilustra a sequência de execução destas etapas.

Figura 1 – Epatas da metodologia



Fonte: Autor

2 REGRAS DE NEGÓCIO

A modelagem foi realizada tomando por base as seguintes regras de negócio requisitos:

1. Opção de realização de pedidos online;
2. Pizzaria delivery;
3. Após cadastro, opção do cliente cadastrar dependentes;
4. Registro de admissão e demissão de funcionários;
5. Log automático das atividades dos funcionários;
6. Controle de estoque com base nos fornecedores e nos ingredientes das pizzas;
7. Esquema de backup automático da base de dados.

3 MODELO CONCEITUAL

O modelo conceitual foi elaborado no programa *BrModelo*. A Figura 2 mostra como foi feita essa modelagem para que os pedidos realizados pelos funcionários fossem armazenados na tabela *Log*. Isso é feito através de triggers.

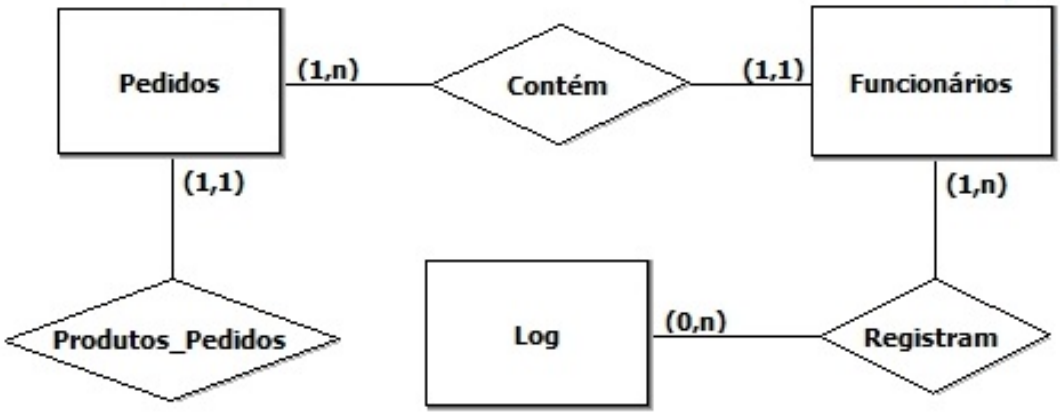


Figura 2 – Utilização de triggers para alimentar a tabela *Log*

.

Na Figura 3 é possível notar que cada funcionário pode ter nenhum ou vários dependentes. Também é possível observar que os clientes podem realizar nenhum ou vários pedidos, mas cada pedido pertence a um único cliente.

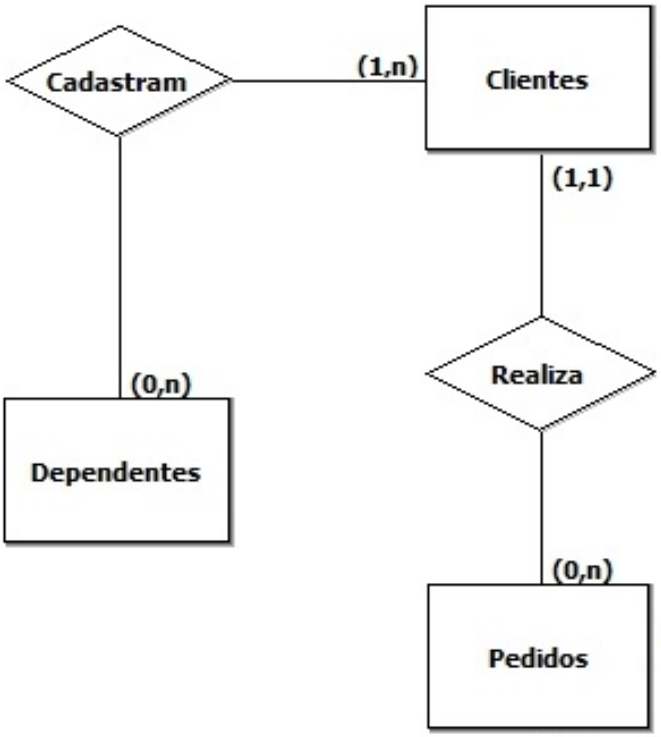


Figura 3 – Entidades: Dependentes, Clientes e Pedidos.

De acordo com a Figura 4, é possível observar que o Fornecedor alimenta o estoque e os produtos são feitos com ingredientes retirados do estoque.

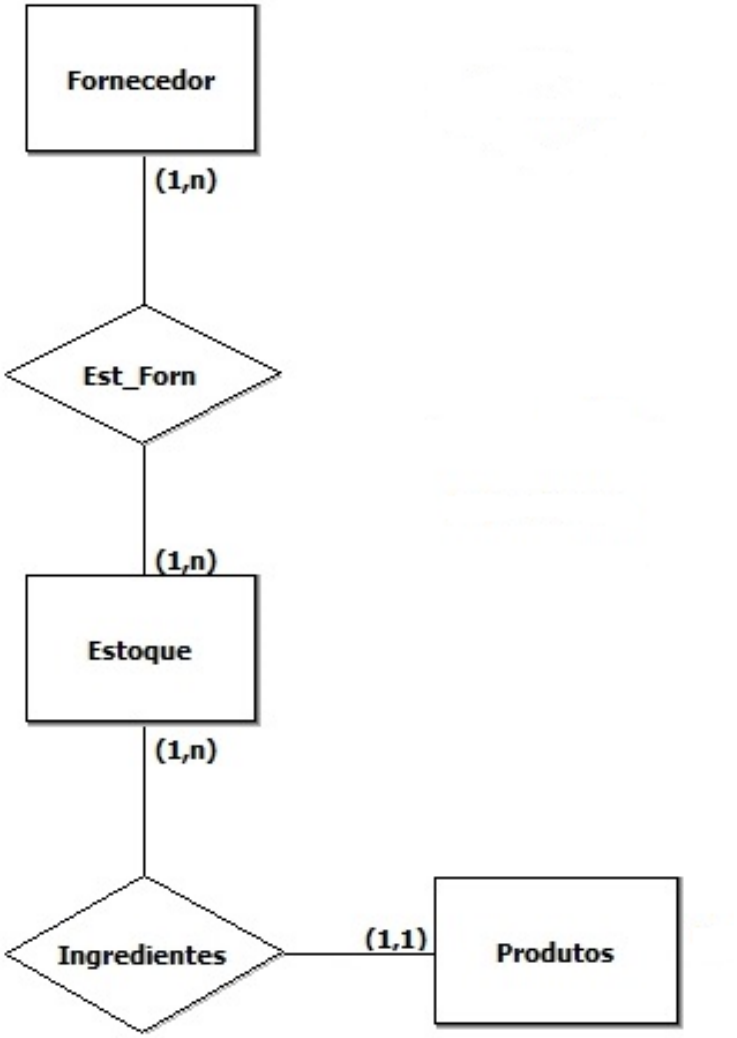


Figura 4 – Entidades: Fornecedor, Estoque e Produtos

Na Figura 5 é possível observar como ficou a modelagem completa do sistema.

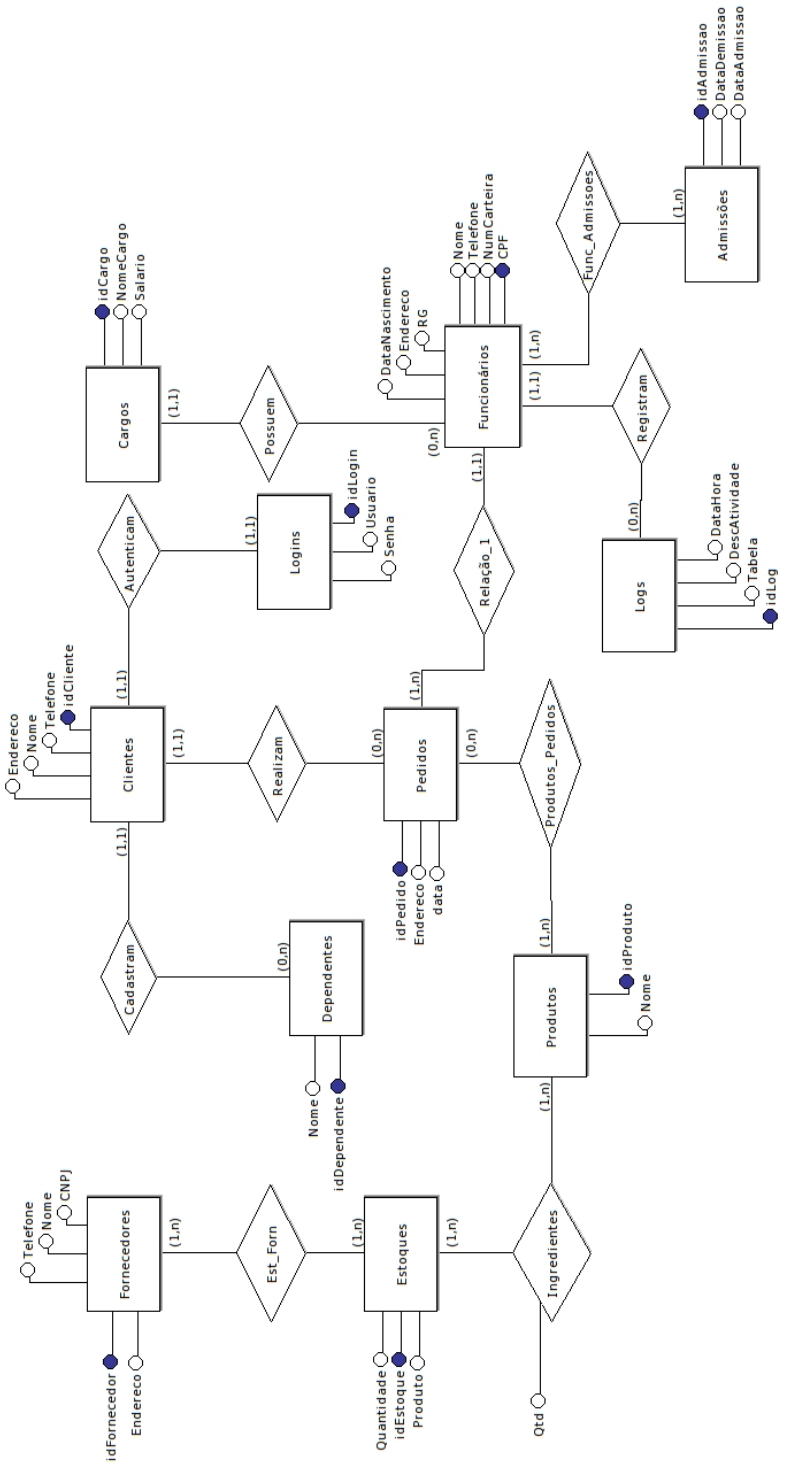


Figura 5 – Modelo Conceitual Completo.

4 MODELO LÓGICO

A Figura 6 representa, conforme o modelo conceitual, a possibilidade do cliente ter ou não login. Isso não impede que o mesmo efetue pedido. Isso aconteceria, por exemplo, no caso do cliente nunca ter feito pedido online.

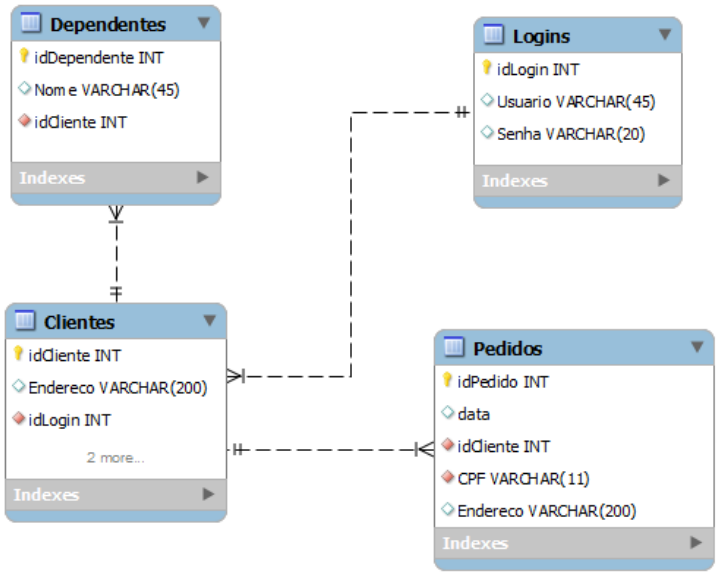


Figura 6 – Modelo Lógico: Dependentes, Clientes, Logins e Pedidos

Na Figura 7 é possível observar os produtos sendo compostos por um ou mais ingredientes; os ingredientes sendo compostos por um ou mais itens do estoque, mas cada item do estoque podendo ser utilizado apenas em uma lista de ingredientes. Também é possível observar a tabela Estoques_Fornecedores, podendo conter vários fornecedores vários itens para o estoque.

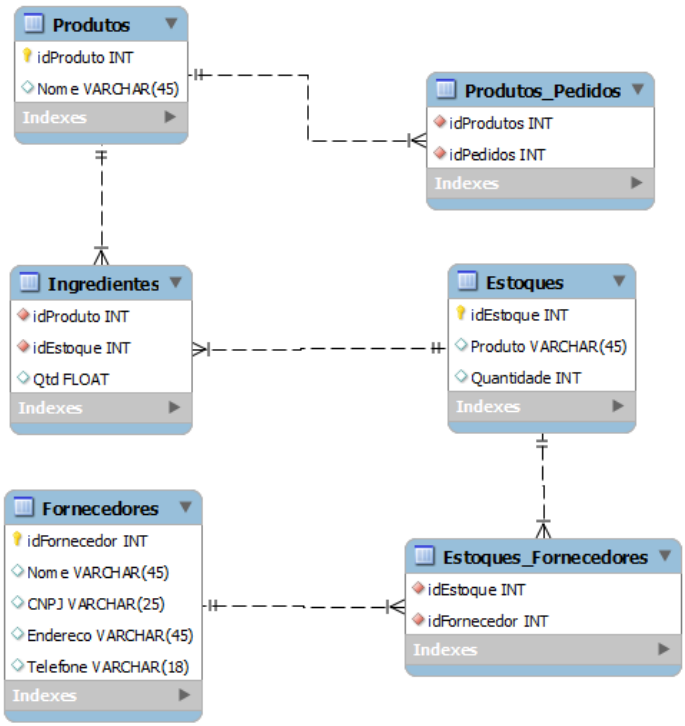


Figura 7 – Modelo Lógico: Produtos, Ingredientes, Produtos_Pedidos, Estoques, Estoques_Fornecedores e Fornecedores.

A Figura 8 mostra a tabela Logs dos funcionários. Essa tabela guarda todas as ações dos funcionários para possível auditorias. É possível observar também que os funcionários têm cargos e cada cargo pode ter muitos funcionários, mas cada funcionários pode ter apenas um cargo na empresa. Como um funcinário pode ser demitido e depois recontradado, existe uma tabela chamada *Funcionarios_Admissoes* onde são salvas as informações a respeito da contratação dos funcionários.

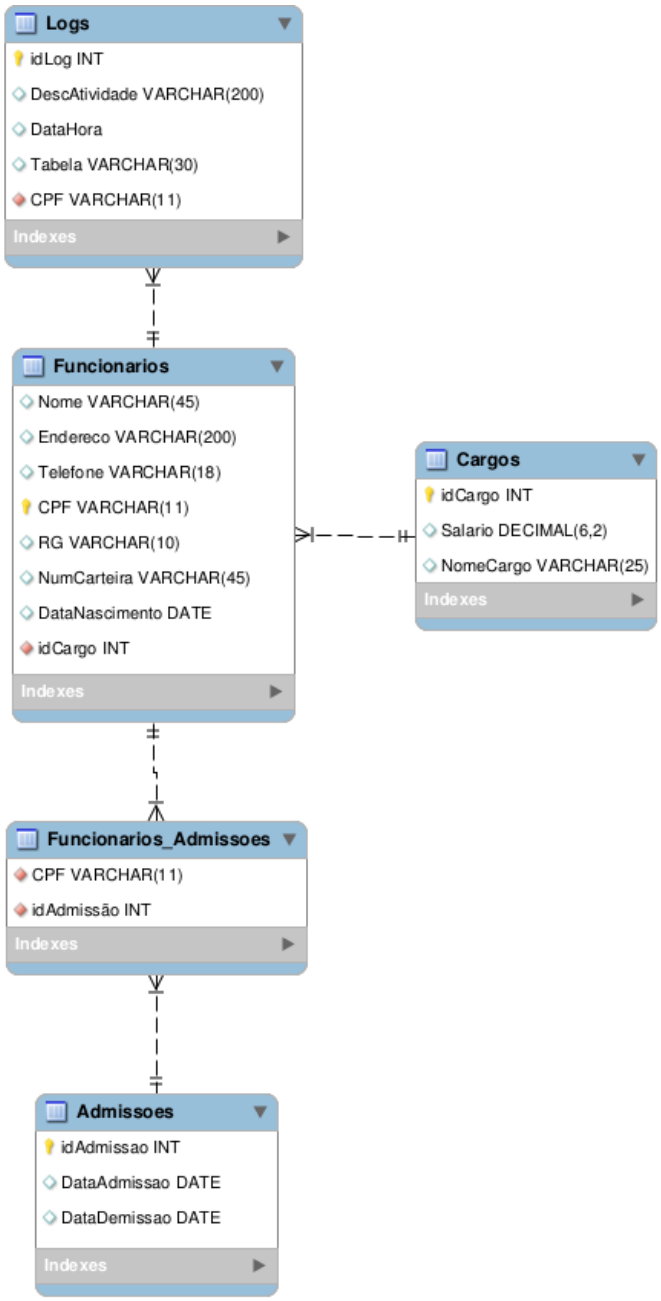


Figura 8 – Modelo Lógico: Logs, Funcionários, Cargos, Funcionarios_Admissoes e Admissoes.

A Figura 9 contém o modelo lógico completo.

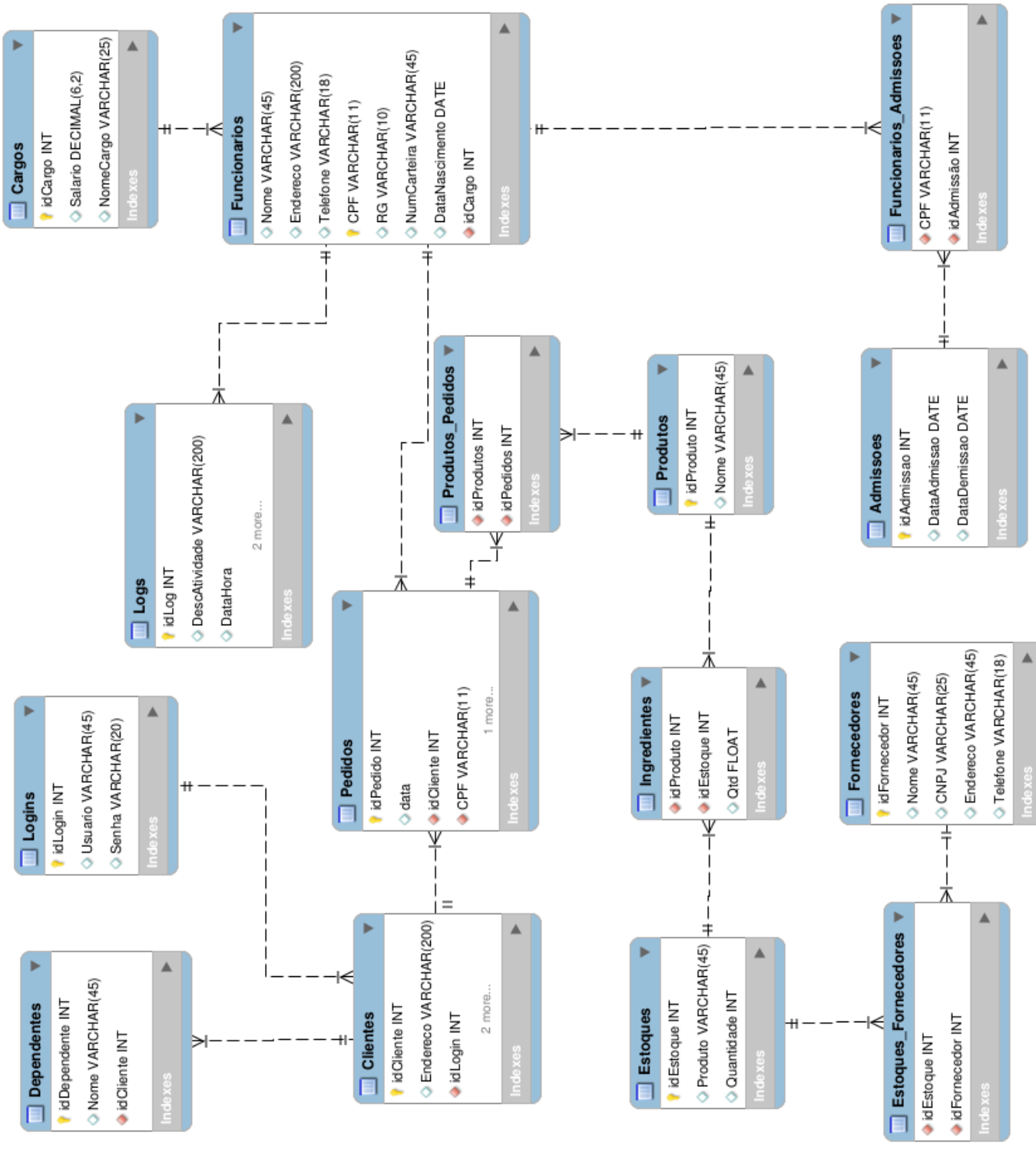


Figura 9 – Modelo Lógico completo.

4.1 DICIONÁRIO DE DADOS

Tabela 1 – Dicionário de Dados - Tabela Admissões

Atributo	Tipo	Nulo	Pk	FK	Ref. Tab.	Descrição
idAdmissao	int	Nao	Sim	Nao	-	PK do registro da tabela admissoes
DataAdmissao	date	Sim	Nao	Nao	-	Data em que o funcionario foi admitido
DataDemissao	date	Sim	Nao	Nao	-	Data em que o funcionario foi demitido

Tabela 2 – Dicionário de Dados - Tabela Cargos

Atributo	Tipo	Nulo	Pk	FK	Ref. Tab.	Descrição
idCargo	int	Nao	Sim	Nao	-	PK do cargo
Salario	decimal	Sim	Nao	Nao	-	Salario correspondente ao cargo
NomeCargo	varchar	Sim	Nao	Nao	-	Nome do cargo

Tabela 3 – Dicionário de Dados - Tabela Clientes

Atributo	Tipo	Nulo	Pk	FK	Ref. Tab.	Descrição
idCliente	int	Nao	Sim	Nao	-	PK dos clientes
Nome	varchar	Nao	Nao	Nao	-	Nome dos clientes
Endereco	varchar	Sim	Nao	Nao	-	Armazena o endereco dos clientes
idLogin	int	Sim	Nao	Sim	Logins	FK login
Telefone	varchar	Sim	Nao	Nao	-	Telefone dos clientes

Tabela 4 – Dicionário de Dados - Tabela Dependentes

Atributo	Tipo	Nulo	Pk	FK	Ref. Tab.	Descrição
idDependentes	int	Nao	Sim	Nao	-	PK dos dependentes
Nome	varchar	Sim	Nao	Nao	-	Nome do dependente
idCliente	int	Nao	Nao	Sim	Clientes	FK do cliente que esta dependendo

Tabela 5 – Dicionário de Dados - Tabela Estoques

Atributo	**Tipo**	**Nulo**	**Pk**	**FK**	**Ref. Tab.**	**Descrição**
idEstoque	int	Nao	Sim	Nao	-	PK do item do estoque
Produto	varchar	Sim	Nao	Nao	-	Nome do produto
Quantidade	int	Sim	Nao	Nao	-	Quantidade do item em estoque
idEstoque	int	Nao	Nao	Sim	Estoques	FK do item do estoque
idFornecedor	int	Nao	Nao	Sim	Fornecedores	FK do fornecedor

Tabela 6 – Dicionário de Dados - Tabela Fornecedores

Atributo	**Tipo**	**Nulo**	**Pk**	**FK**	**Ref. Tab.**	**Descrição**
idFornecedor	int	Nao	Sim	Nao	-	PK do Forcedor
Nome	varchar	Sim	Nao	Nao	-	Nome do Fornecedor
CNPJ	varchar	Sim	Nao	Nao	-	Numero do cadastro de pessoa juridica
Endereco	varchar	Sim	Nao	Nao	-	Endereco do fornecedor
Telefone	varchar	Sim	Nao	Nao	-	Telefone do fornecedor

Tabela 7 – Dicionário de Dados - Tabela Funcionários

Atributo	**Tipo**	**Nulo**	**Pk**	**FK**	**Ref. Tab.**	**Descrição**
Nome	varchar	Sim	Nao	Nao	-	Nome do Funcionario
Endereco	varchar	Sim	Nao	Nao	-	Endereco do Funcionario
Telefone	varchar	Sim	Nao	Nao	-	Telefone do Funcionario
CPF	varchar	Nao	Sim	Nao	-	PK - CPF do funcionario
RG	varchar	Sim	Nao	Nao	-	RG do Funcionario
NumCarteira	varchar	Sim	Nao	Nao	-	Numero da carteira de trabalho
DataNascimento	date	Sim	Nao	Nao	-	Data do nascimento do Funcionario
idCargo	int	Nao	Nao	Sim	Cargos	FK referencia o cargo do funcionario
CPF	varchar	Nao	Nao	Sim	Funcionarios	FK - referencia o funcionario
idAdmissão	int	Nao	Nao	Sim	Admissoes	FK - referencia a admissao

Tabela 8 – Dicionário de Dados - Tabela Ingredientes

Atributo	**Tipo**	**Nulo**	**Pk**	**FK**	**Ref. Tab.**	**Descrição**
idProduto	int	Nao	Nao	Sim	Produtos	FK do Produto
idEstoque	int	Nao	Nao	Sim	Estoques	FK do Estoque
Qtd	float	Nao	Nao	Nao	-	Quantidade do ingrediente utilizado para fazer a pizza

Tabela 9 – Dicionário de Dados - Tabela Logins

Atributo	**Tipo**	**Nulo**	**Pk**	**FK**	**Ref. Tab.**	**Descrição**
idLogin	int	Nao	Sim	Nao	-	PK dos usuarios do site
Usuario	varchar	Sim	Nao	Nao	-	Usuario para efetuar login
Senha	varchar	Sim	Nao	Nao	-	Senha para acessar o site

Tabela 10 – Dicionário de Dados - Tabela Logs

Atributo	**Tipo**	**Nulo**	**Pk**	**FK**	**Ref. Tab.**	**Descrição**
idLog	int	Nao	Sim	Nao	-	PK da tabela Logs
DescAtividade	varchar	Sim	Nao	Nao	-	Armazena a atividade(comandos) do usuario do sistema
DataHora	datetime	Sim	Nao	Nao	-	Armazena a Data da atividade e o horario
Tabela	varchar	Sim	Nao	Nao	-	Armazana o nome da tabela manipulada
CPF	varchar	Nao	Nao	Sim	Funcionarios	FK do funcionario

Tabela 11 – Dicionário de Dados - Tabela Pedidos

Atributo	**Tipo**	**Nulo**	**Pk**	**FK**	**Ref. Tab.**	**Descrição**
idPedido	int	Nao	Sim	Nao	-	PK dos pedidos
data	datetime	Sim	Nao	Nao	-	Data que o pedido foi realizado
idCliente	int	Nao	Nao	Sim	Clientes	FK do cliente que efetuou o pedido
CPF	varchar	Nao	Nao	Sim	Funcionarios	FK do funcionario que atendeu o pedido
Endereco	varchar	Sim	Nao	Nao	-	Endereco do local para entrega do pedido

Tabela 12 – Dicionário de Dados - Tabela Produtos

Atributo	**Tipo**	**Nulo**	**Pk**	**FK**	**Ref. Tab.**	**Descrição**
idProduto	int	Nao	Sim	Nao	-	PK dos produtos vendidos na pizzaria
Nome	varchar	Sim	Nao	Nao	-	Nome do produto
idProduto	int	Nao	Nao	Sim	Produtos	FK dos produtos
idPedido	int	Nao	Nao	Sim	Pedidos	FK dos pedidos

5 IMPLEMENTAÇÃO

O banco de dados foi implementado utilizando o *software SQL Server 2008*. Segue o código de execução para a criação das tabelas:

```
1   USE master
2   GO
3
4   IF EXISTS (select name from sys.databases where name = 'Pizzaria
        ')
5           DROP DATABASE Pizzaria
6   go
7
8   CREATE DATABASE Pizzaria
9   go
10
11  USE Pizzaria
12  go
13
14  SET DATEFORMAT dmy
15  go
16
17  -- ------------------------------------------------------
18  -- Table Pizzaria.Logins
19  -- ------------------------------------------------------
20  CREATE TABLE Logins (
21    idLogin INT NOT NULL,
22    Usuario VARCHAR(45) NULL,
23    Senha VARCHAR(20) NULL,
24    PRIMARY KEY (idLogin)
25  )
26  GO
27
28  -- ------------------------------------------------------
29  -- Table Pizzaria.Clientes
30  -- ------------------------------------------------------
31  CREATE TABLE Clientes (
32    idCliente INT NOT NULL PRIMARY KEY,
33    Nome VARCHAR (200) NOT NULL,
34    Endereco VARCHAR(200) NULL,
35    idLogin INT DEFAULT NULL,
36    Telefone VARCHAR(18) NULL,
37    CONSTRAINT fk_Clientes_Logins
38      FOREIGN KEY (idLogin)
39      REFERENCES Logins (idLogin)
```

```
40            ON DELETE NO ACTION
41            ON UPDATE NO ACTION
42        )
43        GO
44
45        -- -----------------------------------------------------
46        -- Table Pizzaria.Cargos
47        -- -----------------------------------------------------
48        CREATE TABLE Cargos (
49          idCargo INT NOT NULL,
50          Salario DECIMAL(6,2) NULL,
51          NomeCargo VARCHAR(25) NULL,
52          PRIMARY KEY (idCargo)
53        )
54        GO
55
56        -- -----------------------------------------------------
57        -- Table Pizzaria.Funcionarios
58        -- -----------------------------------------------------
59        CREATE TABLE Funcionarios (
60          Nome VARCHAR(45) NULL,
61          Endereco VARCHAR(200) NULL,
62          Telefone VARCHAR(18) NULL,
63          CPF VARCHAR(11) NOT NULL,
64          RG VARCHAR(10) NULL,
65          NumCarteira VARCHAR(45) NULL,
66          DataNascimento DATE NULL,
67          idCargo INT NOT NULL,
68          PRIMARY KEY (CPF),
69          CONSTRAINT fk_Funcionarios_Cargos
70            FOREIGN KEY (idCargo)
71            REFERENCES Cargos (idCargo)
72            ON DELETE NO ACTION
73            ON UPDATE NO ACTION
74        )
75        GO
76
77        -- -----------------------------------------------------
78        -- Table Pizzaria.Pedidos
79        -- -----------------------------------------------------
80        CREATE TABLE Pedidos (
81          idPedido INT NOT NULL,
82          data DATETIME NULL,
83          idCliente INT NOT NULL,
84          CPF VARCHAR(11) NOT NULL,
85          Endereco VARCHAR(200) NULL,
86          PRIMARY KEY (idPedido),
```

```
87          CONSTRAINT fk_Pedidos_Clientes
88            FOREIGN KEY (idCliente)
89            REFERENCES Clientes (idCliente)
90            ON DELETE NO ACTION
91            ON UPDATE NO ACTION,
92          CONSTRAINT fk_Pedidos_Funcionarios
93            FOREIGN KEY (CPF)
94            REFERENCES Funcionarios (CPF)
95            ON DELETE NO ACTION
96            ON UPDATE NO ACTION
97        )
98        GO
99
100       -- -----------------------------------------------------
101       -- Table Pizzaria.Dependentes
102       -- -----------------------------------------------------
103       CREATE TABLE Dependentes (
104         idDependentes INT NOT NULL,
105         Nome VARCHAR(45) NULL,
106         idCliente INT NOT NULL,
107         PRIMARY KEY (idDependentes),
108         CONSTRAINT fk_Dependentes_Clientes
109           FOREIGN KEY (idCliente)
110           REFERENCES Clientes (idCliente)
111           ON DELETE NO ACTION
112           ON UPDATE NO ACTION
113       )
114       GO
115
116       -- -----------------------------------------------------
117       -- Table Pizzaria.Produtos
118       -- -----------------------------------------------------
119       CREATE TABLE Produtos (
120         idProduto INT NOT NULL,
121         Nome VARCHAR(45) NULL,
122         PRIMARY KEY (idProduto)
123       )
124       GO
125
126       -- -----------------------------------------------------
127       -- Table Pizzaria.Estoques
128       -- -----------------------------------------------------
129       CREATE TABLE Estoques (
130         idEstoque INT NOT NULL,
131         Produto VARCHAR(45) NULL,
132         Quantidade INT NULL,
133         PRIMARY KEY (idEstoque)
```

```
134         )
135         GO
136
137
138         -----------------------------------------------------
139         -- Table Pizzaria.Ingredientes
140         -----------------------------------------------------
141         CREATE TABLE Ingredientes (
142           idProduto INT NOT NULL,
143           idEstoque INT NOT NULL,
144           Qtd FLOAT NOT NULL,
145           FOREIGN KEY (idProduto)
146               REFERENCES Produtos (idProduto)
147               ON DELETE NO ACTION
148               ON UPDATE NO ACTION,
149           FOREIGN KEY (idEstoque)
150               REFERENCES Estoques (idEstoque)
151               ON DELETE NO ACTION
152               ON UPDATE NO ACTION
153         )
154         GO
155
156         -- -----------------------------------------------------
157         -- Table Pizzaria.Fornecedores
158         -- -----------------------------------------------------
159         CREATE TABLE Fornecedores (
160           idFornecedor INT NOT NULL,
161           Nome VARCHAR(45) NULL,
162           CNPJ VARCHAR(25) NULL,
163           Endereco VARCHAR(95) NULL,
164           Telefone VARCHAR(18) NULL,
165           PRIMARY KEY (idFornecedor)
166         )
167         GO
168
169         -- -----------------------------------------------------
170         -- Table Pizzaria.Estoques_Fornecedores
171         -- -----------------------------------------------------
172         CREATE TABLE Estoques_Fornecedores (
173           idEstoque INT NOT NULL,
174           idFornecedor INT NOT NULL,
175           CONSTRAINT fk_Estoque_has_Fornecedor_Estoque
176             FOREIGN KEY (idEstoque)
177             REFERENCES Estoques (idEstoque)
178             ON DELETE NO ACTION
179             ON UPDATE NO ACTION,
180           CONSTRAINT fk_Estoque_has_Fornecedor_Fornecedor
```

```
181             FOREIGN KEY (idFornecedor)
182             REFERENCES Fornecedores (idFornecedor)
183             ON DELETE NO ACTION
184             ON UPDATE NO ACTION
185         )
186         GO
187
188         -- -----------------------------------------------------
189         -- Table Pizzaria.Produtos_Pedidos
190         -- -----------------------------------------------------
191         CREATE TABLE Produtos_Pedidos (
192           idProduto INT NOT NULL,
193           idPedido INT NOT NULL,
194           CONSTRAINT fk_Produtos_has_Pedidos_Produtos
195             FOREIGN KEY (idProduto)
196             REFERENCES Produtos (idProduto)
197             ON DELETE NO ACTION
198             ON UPDATE NO ACTION,
199           CONSTRAINT fk_Produtos_has_Pedidos_Pedidos
200             FOREIGN KEY (idPedido)
201             REFERENCES Pedidos (idPedido)
202             ON DELETE NO ACTION
203             ON UPDATE NO ACTION
204         )
205         GO
206
207         -- -----------------------------------------------------
208         -- Table Pizzaria.Admissoes
209         -- -----------------------------------------------------
210         CREATE TABLE Admissoes (
211           idAdmissao INT NOT NULL,
212           DataAdmissao DATE NULL,
213           DataDemissao DATE NULL,
214           PRIMARY KEY (idAdmissao)
215         )
216         GO
217
218         -- -----------------------------------------------------
219         -- Table Pizzaria.Funcionarios_Admissoes
220         -- -----------------------------------------------------
221         CREATE TABLE Funcionarios_Admissoes (
222           CPF VARCHAR(11) NOT NULL,
223           idAdmissão INT NOT NULL,
224           CONSTRAINT fk_Funcionarios_has_Admissão_Funcionarios
225             FOREIGN KEY (CPF)
226             REFERENCES Funcionarios (CPF)
227             ON DELETE NO ACTION
```

```
228             ON UPDATE NO ACTION,
229           CONSTRAINT fk_Funcionarios_has_Admissão_Admissão
230             FOREIGN KEY (idAdmissão)
231             REFERENCES Admissoes (idAdmissao)
232             ON DELETE NO ACTION
233             ON UPDATE NO ACTION
234         )
235         GO
236
237         -- -----------------------------------------------------
238         -- Table Pizzaria.Logs
239         -- -----------------------------------------------------
240         CREATE TABLE Logs (
241           idLog INT NOT NULL,
242           DescAtividade VARCHAR(200) NULL,
243           DataHora DATETIME NULL,
244           CPF VARCHAR(11) NOT NULL,
245           PRIMARY KEY (idLog),
246           CONSTRAINT fk_Log_Funcionarios
247             FOREIGN KEY (CPF)
248             REFERENCES Funcionarios (CPF)
249             ON DELETE NO ACTION
250             ON UPDATE NO ACTION
251         )
252         GO
```

6 EXECUÇÃO E TESTES

As execuções e os testes foram feitos utilizando o *software SQL Server Management Studio 2010*.

6.1 CONSULTAS

A consulta a seguir foi realizada utilizando as tabelas Estoques e Fornecedores e o resultado pode ser visualizado na figra 10

```
1  -- ------------------------------------------------------
2  -- Lista alimentos e seus fornecedores
3  -- ------------------------------------------------------
4  SELECT  Estoques.Produto as [Alimento],
5          Fornecedores.Nome as [Fornecedor]
6      FROM Estoques_Fornecedores
7        INNER JOIN Estoques ON
8            Estoques.idEstoque = Estoques_Fornecedores.idEstoque
9        INNER JOIN Fornecedores ON
10           Fornecedores.idFornecedor = Estoques_Fornecedores.idFornecedor
11   ORDER BY Fornecedores.Nome, Estoques.Produto
12 GO
```

	Alimento	Fornecedor
1	Abobrinha	Alimentos Já
2	Bacon	Alimentos Já
3	Beringela	Alimentos Já
4	Calabresa	Alimentos Já
5	Carne Seca	Alimentos Já
6	Champignon	Alimentos Já
7	Farinha de Trigo	Alimentos Já
8	Lombo	Alimentos Já
9	Ovo	Alimentos Já
10	Requeijão Cre...	Alimentos Já
11	Bróculis	Boa Massa
12	Cebola	Boa Massa
13	Extrato de To...	Boa Massa
14	Frango desfiado	Boa Massa
15	Manjericão	Boa Massa
16	Oregano	Boa Massa
17	Palmito	Boa Massa
18	Queijo Mussar...	Boa Massa
19	Queijo parmesão	Boa Massa
20	Tomate	Boa Massa

Figura 10 – Resultado do select Lista alimentos e seus fornecedores

A consulta a seguir foi realizada utilizando as tabelas Produtos e Estoques e o resultado pode ser visualizado na figura 11

```
1  -- ------------------------------------------------------
2  -- Lista os nomes dos produtos, seus ingredientes e a
3  -- quantidade em estoque
4  -- ------------------------------------------------------
5  SELECT  Produtos.Nome,
6          Estoques.Produto,
7          Estoques.Quantidade
8      FROM Ingredientes
9        INNER JOIN Produtos ON
10           Produtos.idProduto = Ingredientes.idProduto
11       INNER JOIN Estoques ON
12           Estoques.idEstoque = Ingredientes.idEstoque
13   ORDER BY Produtos.Nome, Estoques.Produto
14 GO
```

	Nome	Produto	Quantidade
1	Calabresa	Calabresa	7
2	Calabresa	Cebola	13
3	Calabresa	Extrato de Tomate	12
4	Calabresa	Queijo Mussarela	10
5	Frango C/ Catupiry	Extrato de Tomate	12
6	Frango C/ Catupiry	Frango desfiado	14
7	Frango C/ Catupiry	Requeijão Cremoso	10
8	Frango Especial	Bacon	18
9	Frango Especial	Extrato de Tomate	12
10	Frango Especial	Frango desfiado	14
11	Frango Especial	Oregano	4
12	Frango Especial	Requeijão Cremoso	10
13	Lombo	Extrato de Tomate	12
14	Lombo	Extrato de Tomate	12
15	Lombo	Queijo Mussarela	10
16	Margarita	Extrato de Tomate	12
17	Margarita	Manjericão	7
18	Margarita	Queijo Mussarela	10
19	Margarita	Queijo parmesão	13

Figura 11 – Resultado do select

A consulta a seguir foi realizada utilizando as tabelas Logins e Clientes e o resultado pode ser visualizado na figra 12.

```
1  -- ------------------------------------------------------
2  -- Lista os clientes e os logins de quem o tiver.
3  -- ------------------------------------------------------
4  CREATE VIEW ClientesComLogin
5  AS
6    SELECT  Logins.Usuario,
7        Clientes.idCliente FROM Logins
8          RIGHT JOIN Clientes ON
9        Logins.idLogin = Clientes.idLogin
10 GO
```

	Usuario	idCliente
1	Guilherme	1
2	Matheus	2
3	Victor	3
4	Marcelo	4
5	Pedro	5
6	Joao	6
7	NULL	7
8	NULL	8
9	NULL	9
10	NULL	10

Figura 12 – Resultado do select lista os clientes e os logins de quem o tiver.

A consulta a seguir foi realizada utilizando as tabelas Produtos e Pedidos e o resultado pode ser visualizado na figra 13.

```
1  -- ------------------------------------------------------
2  -- Lista produtos pedidos
3  -- ------------------------------------------------------
4  CREATE VIEW PedidosRealizados
5  AS
6    SELECT  Produtos.Nome AS [Produto],
7            Pedidos.idCliente
8          FROM Produtos_Pedidos
9          INNER JOIN Produtos ON
10             Produtos.idProduto = Produtos_Pedidos.idProduto
11         INNER JOIN Pedidos ON
12             Pedidos.idPedido = Produtos_Pedidos.idPedido
13 GO
```

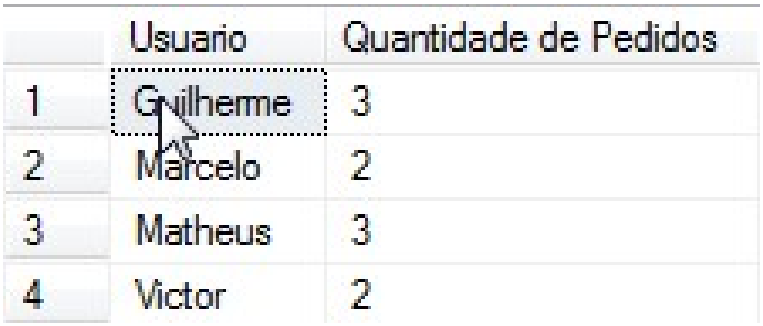
	Produto	idCliente
1	Calabresa	1
2	Frango C/ Catupiry	1
3	Lombo	1
4	Margarita	2
5	Portuguesa	2
6	Napolitana	4
7	Frango Especial	4
8	Toscana	3
9	Nordestina	2
10	Vegetariana	3

Figura 13 – Resultado do select lista produtos pedidos.

A consulta a seguir foi realizada utilizando as views ClientesComLogin e PedidosRealizados e o resultado pode ser visualizado na figra 14.

```
-- ------------------------------------------------------
-- Lista dos clientes que fizeram pedidos.
-- ------------------------------------------------------
CREATE VIEW ClientesQueFizeramPedidos
AS
SELECT  ClientesComLogin.Usuario,
    PedidosRealizados.Produto
     FROM PedidosRealizados
  INNER JOIN ClientesComLogin ON
      ClientesComLogin.idCliente = PedidosRealizados.idCliente
GO


SELECT  ClientesQueFizeramPedidos.Usuario,
        COUNT(*) AS [Quantidade de Pedidos]
    FROM ClientesQueFizeramPedidos
      GROUP BY ClientesQueFizeramPedidos.Usuario
```

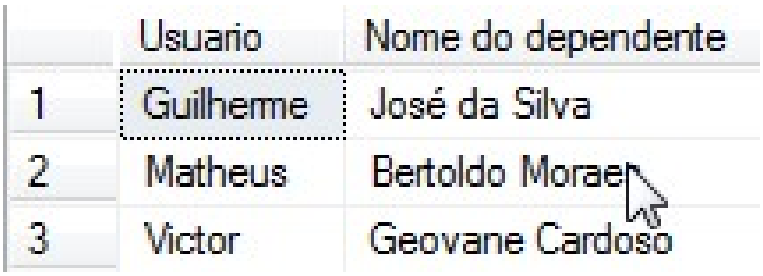


	Usuario	Quantidade de Pedidos
1	Guilherme	3
2	Marcelo	2
3	Matheus	3
4	Victor	2

Figura 14 – Resultado do select lista dos clientes que fizeram pedidos.

A consulta a seguir foi realizada utilizando a view ClientesComLogin e a tabela Dependentes e o resultado pode ser visualizado na figra 15.

```
1 -- ------------------------------------------------------
2 -- Clientes e seus dependentes
3 -- ------------------------------------------------------
4 SELECT  ClientesComLogin.Usuario,
5         Dependentes.Nome [Nome do dependente]
6     FROM Dependentes
7       INNER JOIN ClientesComLogin ON
8           ClientesComLogin.idCliente = Dependentes.idCliente
```



	Usuario	Nome do dependente
1	Guilherme	José da Silva
2	Matheus	Bertoldo Morae
3	Victor	Geovane Cardoso

Figura 15 – Resultado do select clientes e seus dependentes

A consulta a seguir foi realizada utilizando as tabelas Funcionários e Cargos e o resultado pode ser visualizado na figra 16.

```
-- ------------------------------------------------------
-- Funcionários e Cargos
-- ------------------------------------------------------
SELECT  Funcionarios.Nome,
        Funcionarios.CPF,
        Cargos.NomeCargo,
        Cargos.Salario
    FROM Funcionarios
      INNER JOIN Cargos ON
          Cargos.idCargo = Funcionarios.idCargo
  ORDER BY Cargos.NomeCargo, Funcionarios.Nome
GO
```

	Nome	CPF	NomeCargo	Salario
1	Amanda Silveira	12332112365	Balconista	1000.00
2	Carlos Eduardo	12332112366	Balconista	1000.00
3	Catarina Santos	12332112361	Balconista	1000.00
4	Miguel de Souza	12332112362	Entregador	1500.00
5	Sérgio Malandro	12332112363	Entregador	1500.00
6	Carlos Belozo	12332112368	Garçon	1500.00
7	Sandra de Sá	12332112369	Garçon	1500.00
8	Roberto Jefferson	12332112364	Gerente	2500.00
9	José Benedito	12332112360	Pizzaiolo	2000.00
10	Miguel de Arrais	12332112367	Pizzaiolo	2000.00

Figura 16 – Resultado do select Funcionários e Cargos

A consulta a seguir foi realizada utilizando as tabelas Funcionários e Cargos e o resultado pode ser visualizado na figra 17.

```
1
2  -- ------------------------------------------------------
3  -- Funcionários, cargos e suas admissões
4  -- ------------------------------------------------------
5  SELECT  Funcionarios.Nome,
6          Admissoes.DataAdmissao,
7          Cargos.NomeCargo,
8          Cargos.Salario
9      FROM Funcionarios_Admissoes
10       INNER JOIN Funcionarios ON
11           Funcionarios.CPF = Funcionarios_Admissoes.CPF
12       INNER JOIN Admissoes
13           ON Admissoes.idAdmissao = Funcionarios_Admissoes.idAdmissão
14       INNER JOIN Cargos
15           ON Cargos.idCargo = Funcionarios.idCargo
16 GO
```

	Nome	DataAdmissao	NomeCargo	Salario
1	José Benedito	2005-08-30	Pizzaiolo	2000.00
2	Catarina Santos	2007-04-28	Balconista	1000.00
3	Miguel de Souza	2009-06-30	Entregador	1500.00
4	Sérgio Malandro	2009-10-14	Entregador	1500.00
5	Roberto Jefferson	2010-08-15	Gerente	2500.00
6	Amanda Silveira	2010-08-25	Balconista	1000.00
7	Carlos Eduardo	2011-09-30	Balconista	1000.00
8	Miguel de Arrais	2011-10-01	Pizzaiolo	2000.00
9	Carlos Belozo	2011-11-30	Garçon	1500.00
10	Sandra de Sá	2012-04-01	Garçon	1500.00

Figura 17 – Resultado do select funcionários, cargos e suas admissões

6.2 PROCEDIMENTOS ARMAZENADOS

Procedimento armazenado para cálculo do aniversários de cada funcionário na empresa.

```
1  CREATE PROCEDURE usp_idadeFuncionarios
2  AS
3    SELECT   Nome,
4             DATEDIFF(YEAR, DataNascimento, GETDATE()) - CASE
5             WHEN GETDATE() < DATEADD(YEAR,
6                 DATEDIFF(YEAR,DataNascimento,
7                                       GETDATE()),
8                           DataNascimento)
9               THEN 1
10              ELSE 0
11            END AS 'Idade',
12                   CONVERT(VARCHAR(10),
13                   DataNascimento, 103) As 'Data de Nascimento'
14       FROM Funcionarios
15 GO
16
17 EXEC usp_idadeFuncionarios
18 GO
```

Depois de executado, o reultado obtido pode ser visualizado na Figura 18.

	Nome	Idade	Data de Nascimento
1	José Benedito	31	14/09/1982
2	Catarina Santos	32	18/09/1981
3	Miguel de Souza	26	08/08/1987
4	Sérgio Malandro	28	18/04/1985
5	Roberto Jefferson	44	01/12/1969
6	Amanda Silveira	39	23/07/1974
7	Carlos Eduardo	40	02/03/1973
8	Miguel de Arrais	22	09/12/1990
9	Carlos Belozo	23	15/08/1990
10	Sandra de Sá	22	11/09/1991

Figura 18 – Procedimento Armazenado para calcular idade.

```
CREATE PROCEDURE usp_pedidosRealizados
  @nome VARCHAR(45)
AS
  SELECT  F.Nome,
          C.NomeCargo as 'Cargo',
          Prod.Nome,
          CONVERT(VARCHAR(10),P.data, 103) As 'Data do Pedido'
        FROM Funcionarios F
            INNER JOIN Cargos C ON
                C.idCargo = F.idCargo
            INNER JOIN Pedidos P ON
                P.CPF = F.CPF
            INNER JOIN Produtos_Pedidos PP ON
                PP.idPedido = P.idPedido
            INNER JOIN Produtos Prod ON
                Prod.idProduto = PP.idProduto
        WHERE F.Nome = @nome
GO
```

Depois de executado, o reultado obtido pode ser visualizado na Figura 19.

	Nome	Cargo	Nome Produto	Data do Pedido
1	Sérgio Malandro	Entregador	Calabresa	01/12/2013
2	Sérgio Malandro	Entregador	Frango C/ Catupiry	01/12/2013
3	Sérgio Malandro	Entregador	Lombo	01/12/2013
4	Miguel de Souza	Entregador	Margarita	30/11/2013
5	Miguel de Souza	Entregador	Portuguesa	30/11/2013
6	Miguel de Souza	Entregador	Napolitana	30/11/2013
7	Miguel de Souza	Entregador	Frango Especial	30/11/2013
8	Sérgio Malandro	Entregador	Toscana	30/11/2013
9	Miguel de Souza	Entregador	Nordestina	30/11/2013
10	Sérgio Malandro	Entregador	Vegetariana	30/11/2013

Figura 19 – Resultado do procedimento armazenado que retorna os pedidos realizados.

```
CREATE PROCEDURE usp_pedidosRealizadosCliente
  @nome VARCHAR(45)
AS
  SELECT  Cli.Nome,
          Prod.Nome,
          CONVERT(VARCHAR(10),
          P.data, 103) As 'Data do Pedido'
    FROM Clientes Cli
        INNER JOIN Pedidos P ON
            P.idCliente = Cli.idCliente
        INNER JOIN Produtos_Pedidos PP
            ON PP.idPedido = P.idPedido
        INNER JOIN Produtos Prod
            ON Prod.idProduto = PP.idProduto
        WHERE Cli.Nome = @nome
GO
```

Depois de executado, o reultado obtido pode ser visualizado na Figura 20 e também na Figura **??**.

	Nome	Nome	Data do Pedido
1	Robervaldo	Calabresa	01/12/2013
2	Robervaldo	Frango C/ Catupiry	01/12/2013
3	Robervaldo	Lombo	01/12/2013

Figura 20 – Procedimento Armazenado que retorna os pedidos de um determinado cliente via parâmetro do nome.

	Nome	Nome	Data do Pedido
1	Valdomiro	Margarita	30/11/2013
2	Valdomiro	Portuguesa	30/11/2013
3	Valdomiro	Nordestina	30/11/2013

Figura 21 – Procedimento Armazenado que retorna os pedidos de um determinado cliente via parâmetro do nome.

6.3 TRIGGERS

Foram criadas duas triggers. A primeira verifica quando houver um INSERT em Produtos_Pedidos e decrementa a quantidade do produto no estoque. A segunda verifica quando houver um INSERT, UPDATE ou DELETE na tabela Pedidos e armazena a operação executada na tabela de logs. Segue o código das triggers:

```
1  /* ******************************************************
2  *  Verifica quando houver um INSERT em Produtos_Pedidos
3  *  e decrementa a quantida de do produto no estoque.
4  * ******************************************************/
5  CREATE TRIGGER tg_Produtos_Pedidos
6  ON Produtos_Pedidos
7  AFTER INSERT
8  AS
9  BEGIN
10   -- Declara as variaveis a ser utilizada dentro da TRIGGER.
11   DECLARE @idPedido INT
12   DECLARE @idProduto INT
13   DECLARE @idEstoque INT
14
15   -- Atribui o código do pedido a variavel.
16   SET @idPedido = (SELECT inserted.idPedido FROM inserted)
17
18   -- Atribui a variavel @DataLocacao o valor da data de locacao da
        tabela locacao.
19   SET @idProduto = (SELECT inserted.idProduto FROM inserted)
20
21   DECLARE @cursorEstoque CURSOR;
22
23   SET @cursorEstoque = CURSOR FOR
24       SELECT e.idEstoque, e.Quantidade, i.Qtd
25         FROM Estoques e
26         INNER JOIN Ingredientes i
27         ON e.idEstoque = i.idEstoque
28         WHERE i.idProduto = @idProduto;
29
30   DECLARE @ESTOQUE INT;
31   DECLARE @ESTOQUE_QUANTIDADE INT;
32   DECLARE @INGREDIENTES_QUANTIDADE INT;
33
34   OPEN @cursorEstoque;
35
36   IF(CURSOR_STATUS('variable','@cursorEstoque') = 1)
37   BEGIN
38     FETCH NEXT FROM @cursorEstoque INTO
39     @ESTOQUE, @ESTOQUE_QUANTIDADE, @INGREDIENTES_QUANTIDADE;
40
41     WHILE (@@FETCH_STATUS = 0)
42     BEGIN
43
44     UPDATE Estoques SET Quantidade = @ESTOQUE_QUANTIDADE -
          @INGREDIENTES_QUANTIDADE
45     WHERE idEstoque = @ESTOQUE;
```

```
46
47      FETCH NEXT FROM @cursorEstoque INTO
48      @ESTOQUE, @ESTOQUE_QUANTIDADE, @INGREDIENTES_QUANTIDADE;
49      END
50    END
51
52    CLOSE @cursorEstoque;
53    DEALLOCATE @cursorEstoque;
54  END
55  GO
56
57  /* Select para verificar se a trigger foi executada com sucesso. */
58
59  SELECT * FROM Produtos_Pedidos;
60
61  INSERT INTO Produtos_Pedidos VALUES (1, 1);
62
63
64  SELECT * FROM Estoques e
65  INNER JOIN Ingredientes i
66  ON e.idEstoque = i.idEstoque
67  WHERE i.idProduto = 3
68
69  SELECT * FROM Estoques
70  SELECT * FROM Ingredientes
71
72
73  /* ******************************************************
74  *  Verifica quando houver um INSERT, UPDATE e DELETE
75  *  na Pedidos e armazena a operação executada na tabela
76  *  de logs.                                             *
77  * ******************************************************/
78  CREATE TRIGGER tg_PEDIDOS_LOG
79  ON Pedidos FOR INSERT, UPDATE, DELETE
80  AS
81    -- @idPedido: Variável que armazenará os registros afetados
82    -- por uma instrução INSERT, UPDATE, DELETE.
83    DECLARE @idPedido INT;
84    DECLARE @CPF VARCHAR(11);
85
86    -- @Acao: Indicará a instrução feita (INSERT, UPDATE, DELETE).
87    DECLARE @Acao NVARCHAR(10);
88
89    -- @@CursorPEDIDOS: CURSOR que fará com que seja manipulado
90    -- registro por registro de uma tabela, podendo ser
91    -- a tabelas temporárias inserted ou deleted
92    DECLARE @CursorPEDIDOS CURSOR;
```

```
93
94    -- Condição que verifica se há regitros na tabela inserted e não há
95    -- registros na tabela deleted, o que caracteriza uma instrução INSERT
          .
96    -- Se houver registros na tabela inserted e houver registros na tabela
97    -- deleted, caracteriza uma instrução UPDATE. Caso contrário, não há
98    -- registros na tabela inserted e há registros na tabela deleted,
99    -- caracterizando uma instrução DELETE.
100   -- Dependendo da condição, atribui uma determinada instrução SELECT
101   -- para o cursor @CursorCLIENTE e atribui uma ação à variável @Acao
102   IF EXISTS (SELECT * FROM inserted) AND NOT EXISTS (SELECT * FROM
          deleted)
103     BEGIN
104       SET @CursorPEDIDOS = CURSOR FOR SELECT idPedido, CPF FROM inserted
              ;
105       SET @Acao = 'INSERT';
106       OPEN @CursorPEDIDOS;
107     END
108   ELSE IF EXISTS (SELECT * FROM inserted) AND EXISTS (SELECT * FROM
          deleted)
109     BEGIN
110       SET @CursorPEDIDOS = CURSOR FOR SELECT idPedido, CPF FROM deleted;
111       SET @Acao = 'UPDATE';
112       OPEN @CursorPEDIDOS;
113     END
114   ELSE IF NOT EXISTS (SELECT * FROM inserted) AND EXISTS (SELECT * FROM
          deleted)
115     BEGIN
116       SET @CursorPEDIDOS = CURSOR FOR SELECT idPedido, CPF FROM deleted;
117       SET @Acao = 'DELETE';
118       OPEN @CursorPEDIDOS;
119     END
120
121   -- CURSOR_STATUS: Função que verifica o status da variával cursor
          @CursorCLIENTE
122   -- retornando 1 caso ele esteja aberto(open), 0 caso não esteja
          atribuido nenhuma
123   -- instrução SELECT, -1 caso esteja fechado(close), entre outros.
124   IF(CURSOR_STATUS('variable','@CursorPEDIDOS') = 1)
125   BEGIN
126     -- Obtem o próximo registro do cursor @CursorCLIENTE e
127     -- armazena na variável @cod_CLIENTE
128     FETCH NEXT FROM @CursorPEDIDOS INTO @idPedido, @CPF
129
130     -- Laço que irá executar enquanto a variável @@FETCH_STATUS for
131     -- igual a 0, o que indica que ainda há registros no cursor
            @CursorCLIENTE,
```

```
132     -- quando não houver registros a variável @@FETCH_STATUS será igual
            a -1.
133     WHILE (@@FETCH_STATUS = 0)
134     BEGIN
135       INSERT INTO Logs (DescAtividade, DataHora, Tabela, CPF)
136       VALUES (@Acao + ':' + CONVERT(VARCHAR, @idPedido), GETDATE(), '
              Pedidos', @CPF);
137
138       FETCH NEXT FROM @CursorPEDIDOS INTO @idPedido, @CPF;
139     END
140
141     -- Desaloca o cursor da memória.
142     CLOSE @CursorPEDIDOS;
143
144     -- Remove as todas referências do cursor.
145     DEALLOCATE @CursorPEDIDOS;
146   END
147 GO
148
149 /* Select para verificar se a trigger foi executada com sucesso. */
150 SELECT * FROM Pedidos
151
152 INSERT INTO Pedidos VALUES
153     (15, '05-12-2013', 1, '12332112363', 'AV Andrômeda, 720, Centro.');
154
155     DELETE FROM Pedidos WHERE idPedido = 15;
156
157 SELECT * FROM Logs;
```

6.4 ESQUEMA DE BACKUP

No script abaixo, pode-se notar a implementação de um backup completo da base de dados para o local “C:\\Pizzaria\\PizzariaCompleto.BAK”

```
1  USE master
2  GO
3
4  ALTER DATABASE Pizzaria SET RECOVERY FULL
5  GO
6
7  :setvar diretorio "C:\Pizzaria"
8
9  EXEC XP_CREATE_SUBDIR '$(diretorio)'
10 GO
11
12 BACKUP DATABASE Pizzaria
```

```
13    TO DISK = 'C:\Pizzaria\PizzariaCompleto.BAK'
14  GO
```

CONSIDERAÇÕES FINAIS

Apesar de parecer simples, criar um banco de dados para uma pizzaria mostrou-se uma tarefa cheia de detalhes a se pensar. Ao ser implementado, tornou-se funcional, sendo possível utilizá-lo em um ambiente real.

REFERÊNCIAS

Anexos

ANEXO A – DADOS INSERIDOS PARA TESTE

```
USE master
GO

-- ----------------------------------------------------
-- Table Pizzaria.Logins
-- ----------------------------------------------------
INSERT INTO Logins VALUES
        (1, 'Guilherme', 'egmdc321'),
        (2, 'Matheus', 'egmdc321'),
        (3, 'Victor', 'egmdc321'),
        (4, 'Marcelo', 'egmdc321'),
        (5, 'Pedro', 'egmdc321'),
        (6, 'Joao', 'egmdc321');
GO

-- ----------------------------------------------------
-- Table Pizzaria.Clientes
-- ----------------------------------------------------
INSERT INTO Clientes VALUES
        (1, 'Robervaldo', 'Av Ministro Nelson Hungria, 280, Centro,
                Santo Antônio do Pinhal-SP - CEP 12450-000', 1, '(12)
                   3674-3689'),
        (2, 'Valdomiro', 'Av Coronel Sebastião Marcondes da Silva,
                149, Centro, Santo Antônio do Pinhal-SP - CEP 12450-000'
                   , 2, '(12)3654-5709'),
        (3, 'Cleidiane', 'Rua Sao João, 455, Centro, São José de Campos-
           SP
                 - CEP 12440-123', 3, '(12)3644-5610'),
        (4, 'Wanilda', 'Rua Quinze De Novembro, 394, Centro, Taubaté-SP
           -
                CEP 12440-123', 4, '(12)3644-5160'),
        (5, 'Soleneusa', 'Rua Sao Sebastiao, 289, Centro, Tremembé-SP -
                CEP 12440-123', 5, '(12)3644-6510'),
        (6, 'Godofredo', 'Rua Santos Dumont, 876, Centro, Ubatuba-SP -
                CEP 12440-123', 6, '(12)3644-1560'),
        (7, 'Jaime', 'Rua Belo Horizonte, 255, Centro, Londrina-PR -
                CEP 12440-123', NULL, '(32)3644-5560'),
        (8, 'Jean', 'Rua José Bonifácio, 580, Centro, Maringá-PR -
                CEP 12440-123', NULL, '(32)3644-5660'),
        (9, 'Claudisney', 'Rua Vinte e Três, 290, Centro, Barbosa-SP -
```

```
37                  CEP 12440-123', NULL, '(12)3644-5980'),
38          (10, 'Fúlvio', 'Rua Santa Rita, 276, Centro, Manaus-AM -
39                  CEP 12440-123', NULL, '(98)3644-5130');
40  GO
41
42  -- ------------------------------------------------------
43  -- Table Pizzaria.Cargos
44  -- ------------------------------------------------------
45  INSERT INTO Cargos VALUES
46          (1, 1500, 'Entregador'),
47          (2, 1000, 'Balconista'),
48          (3, 2500, 'Gerente'),
49          (4, 2000, 'Pizzaiolo'),
50          (5, 1500, 'Garçon');
51  GO
52
53  -- ------------------------------------------------------
54  -- Table Pizzaria.Funcionarios
55  -- ------------------------------------------------------
56  INSERT INTO Funcionarios VALUES
57          ('Roberto Jefferson', 'Rua Conde de Bobadela, 225,
58                  Centro, Rio Branco', '9-9909-4413', '12332112364',
59                  '490808800','9999999999','01/12/1969',3),
60          ('Amanda Silveira', 'Rua Senador Rocha Lagoa, 235, Centro,
61                  Cuiabá', '9-9909-4413', '12332112365','490808800',
62                  '9999999999','23/07/1974',2),
63          ('Carlos Eduardo', 'Praça Reinaldo Alves de Brito, 325,
64                  Centro, Curitiba', '9-9909-4413', '12332112366','
                       490808800',
65                  '9999999999','2/03/1973',2),
66          ('Miguel de Arrais', 'Rua Conde de Bobadela, 223, Centro,
67                  João Pessoa', '9-9909-4413', '12332112367','490808800',
68                  '9999999999','09/12/1990',4),
69          ('Carlos Belozo', 'Praça Silviano Brandão, 245, Centro, Belém',
70                  '9-9909-4413', '12332112368','490808800','9999999999',
71                  '15/08/1990',5),
72          ('Sandra de Sá', 'Rua Conde de Bobadela, 224, Centro, Teresina',
73                  '9-9909-4413', '12332112369','490808800','9999999999','
                       11/09/1991',5),
74          ('Sérgio Malandro', 'Rua Alvarenga, 425, Centro, Natal', '
               9-9909-4413',
75                  '12332112363','490808800','9999999999','18/04/1985',1),
76          ('Miguel de Souza', 'Rua Randolfo Bretas, 525, Centro, Porto
               Alegre',
77                  '9-9909-4413', '12332112362','490808800','9999999999','
                       08/08/1987',1),
```

```
78          ('Catarina Santos', 'Rua Antônio de Albuquerque, 255, Centro,
               Florianópolis',
79                   '9-9909-4413', '12332112361','490808800','9999999999','
                        18/09/1981',2),
80          ('José Benedito', 'Praça Barão do Rio Branco, 909, Centro,
               Aracajú',
81                   '9-9909-4413', '12332112360','490808800','9999999999','
                        14/09/1982',4);
82  GO
83
84  -- ----------------------------------------------------
85  -- Table Pizzaria.Pedidos
86  -- ----------------------------------------------------
87  INSERT INTO Pedidos VALUES
88      (1, '01-12-2013', 1, '12332112363',
89              'AV Andrômeda, 720, Centro.'),
90      (2, '30-11-2013', 2, '12332112362',
91              'Av Anahanguera, 820, Bairro das Flores'),
92      (3, '30-11-2013', 3, '12332112363',
93              'Rua São João, 520, Centro'),
94      (4, '30-11-2013', 4, '12332112362',
95              'Rua Nelson de Fátima, 400, Bairro Sertãozinho'),
96      (5, '29-11-2013', 5 , '12332112362',
97              'Rua Sebastião da Rosa, s/n, Bairro Matadouro'),
98      (6, '01-12-2013', 6, '12332112363',
99              'AV Andrômeda, 720, Centro.'),
100     (7, '30-11-2013', 7, '12332112362',
101             'Av Anahanguera, 820, Bairro das Flores'),
102     (8, '30-11-2013', 8, '12332112363',
103             'Rua São João, 520, Centro'),
104     (9, '30-11-2013', 9, '12332112362',
105             'Rua Nelson de Fátima, 400, Bairro Sertãozinho'),
106     (10, '29-11-2013', 10 , '12332112362',
107             'Rua Sebastião da Rosa, s/n, Bairro Matadouro')
108 GO
109
110 -- ----------------------------------------------------
111 -- Table Pizzaria.Dependentes
112 -- ----------------------------------------------------
113 INSERT INTO Dependentes VALUES
114   (1, 'José da Silva', 1),
115   (2, 'Bertoldo Moraes', 2),
116   (3, 'Geovane Cardoso', 3)
117 GO
118
119 -- ----------------------------------------------------
120 -- Table Pizzaria.Produtos
```

```
121 -- -----------------------------------------------------
122 INSERT INTO Produtos VALUES
123         (1,'Calabresa'),
124         (2,'Frango C/ Catupiry'),
125         (3,'Lombo'),
126         (4,'Margarita'),
127         (5,'Portuguesa'),
128         (6,'Napolitana'),
129         (7,'Frango Especial'),
130         (8,'Toscana'),
131         (9,'Nordestina'),
132         (10,'Vegetariana')
133 GO
134
135 -- -----------------------------------------------------
136 -- Table Pizzaria.Estoques
137 -- -----------------------------------------------------
138 INSERT INTO Estoques VALUES
139         (1,'Extrato de Tomate',12),
140         (2,'Requeijão Cremoso',10),
141         (3,'Farinha de Trigo',20),
142         (4,'Queijo Mussarela',10),
143         (5,'Frango desfiado',14),
144         (6,'Oregano',4),
145         (7,'Calabresa',7),
146         (8,'Bacon',18),
147         (9,'Ovo',29),
148         (10,'Cebola',13),
149         (11,'Queijo parmesão',13),
150         (12,'Manjericão',7),
151         (13,'Abobrinha',3),
152         (14,'Beringela',9),
153         (15,'Bróculis',10),
154         (16,'Palmito',21),
155         (17,'Champignon',12),
156         (18,'Lombo',11),
157         (19,'Tomate',2),
158         (20,'Carne Seca',2);
159 GO
160
161 -- -----------------------------------------------------
162 -- Table Pizzaria.Ingredientes
163 -- -----------------------------------------------------
164 INSERT INTO Ingredientes VALUES
165         (1,1,1),
166         (1,4,1),
167         (1,7,1),
```

```
168         (1,10,1),
169
170         (2,1,1),
171         (2,5,1),
172         (2,2,1),
173
174         (3,1,1),
175         (3,4,1),
176         (3,1,1),
177
178         (4,1,1),
179         (4,4,1),
180         (4,19,1),
181         (4,11,1),
182         (4,12,1),
183
184         (5,1,1),
185         (5,9,1),
186         (5,4,1),
187         (5,10,1),
188
189         (6,1,1),
190         (6,4,1),
191         (6,11,1),
192         (6,19,1),
193
194         (7,1,1),
195         (7,5,1),
196         (7,2,1),
197         (7,8,1),
198         (7,6,1),
199
200         (8,1,1),
201         (8,4,1),
202         (8,7,1),
203         (8,6,1),
204
205         (9,1,1),
206         (9,2,1),
207         (9,20,1),
208         (9,10,1),
209
210         (10,1,1),
211         (10,13,1),
212         (10,14,1),
213         (10,15,1),
214         (10,16,1),
```

```
215         (10,17,1);
216 GO
217
218 -- -----------------------------------------------------
219 -- Table Pizzaria.Fornecedores
220 -- -----------------------------------------------------
221 INSERT INTO Fornecedores VALUES
222         (1,'Alimentos Já', '01010101-01010','Rua Carlos Bom Tempo, 2215,
223                 Centro, Rio Branco', '9-9909-4413'),
224         (2,'Boa Massa', '02020202-02020','Rua Conde de Monte Cristo,
225                 21, São Paulo', '9-9909-4413'),
226         (3, 'Frutas ATC', '091942-00130', 'Av. Ministro Celso de Melo,
227                 242, São Paulo', '12495-045'),
228         (4, 'RMC Verduras', '192814-9049', 'Av do Povo, 545,
229                 Taubaté', '5432-090')
230 GO
231
232 -- -----------------------------------------------------
233 -- Table Pizzaria.Estoques_Fornecedores
234 -- -----------------------------------------------------
235 INSERT INTO Estoques_Fornecedores  VALUES
236         (1,2),
237         (2,1),
238         (3,1),
239         (4,2),
240         (5,2),
241         (6,2),
242         (7,1),
243         (8,1),
244         (9,1),
245         (10,2),
246         (11,2),
247         (12,2),
248         (13,1),
249         (14,1),
250         (15,2),
251         (16,2),
252         (17,1),
253         (18,1),
254         (19,2),
255         (20,1)
256  GO
257
258 -- -----------------------------------------------------
259 -- Table Pizzaria.Produtos_Pedidos
260 -- -----------------------------------------------------
261  INSERT INTO Produtos_Pedidos VALUES
```

```
  (1, 1),
  (2, 1),
  (3, 1),
  (4, 2),
  (5, 2),
  (6, 4),
  (7, 4),
  (8, 3),
  (9, 2),
  (10, 3)
GO

-- ----------------------------------------------------
-- Table Pizzaria.Admissoes
-- ----------------------------------------------------
INSERT INTO Admissoes VALUES
  (1, '30-08-2005', ''),
  (2, '28-04-2007', '11-07-2007'),
  (3, '30-06-2009', ''),
  (4, '14-10-2009', ''),
  (5, '15-08-2010', ''),
  (6, '25-08-2010', ''),
  (7, '30-09-2011', ''),
  (8, '01-10-2011', ''),
  (9, '30-11-2011', ''),
  (10, '01-04-2012', '')
GO

-- ----------------------------------------------------
-- Table Pizzaria.Funcionarios_Admissoes
-- ----------------------------------------------------
INSERT INTO Funcionarios_Admissoes VALUES
  ('12332112360', 1),
  ('12332112361', 2),
  ('12332112362', 3),
  ('12332112363', 4),
  ('12332112364', 5),
  ('12332112365', 6),
  ('12332112366', 7),
  ('12332112367', 8),
  ('12332112368', 9),
  ('12332112369', 10)
GO
```