

Ao final deste assunto, você será capaz de:

- Fazer a criação do esquema de um banco de dados em SQL a partir das definições do Projeto Físico de um banco de dados;
- Fazer a definição de um banco de dados utilizando sentenças SQL de um SGBD específico (MySQL, por exemplo);
- Fazer modificações nas definições de uma tabela em SQL

Neste momento, o que se estará tratando são esquemas de dados e, para tanto, pode ser necessário fazer a criação de tabelas, que podem precisar de alterações nas suas definições.

3.1.1. Criação de tabelas

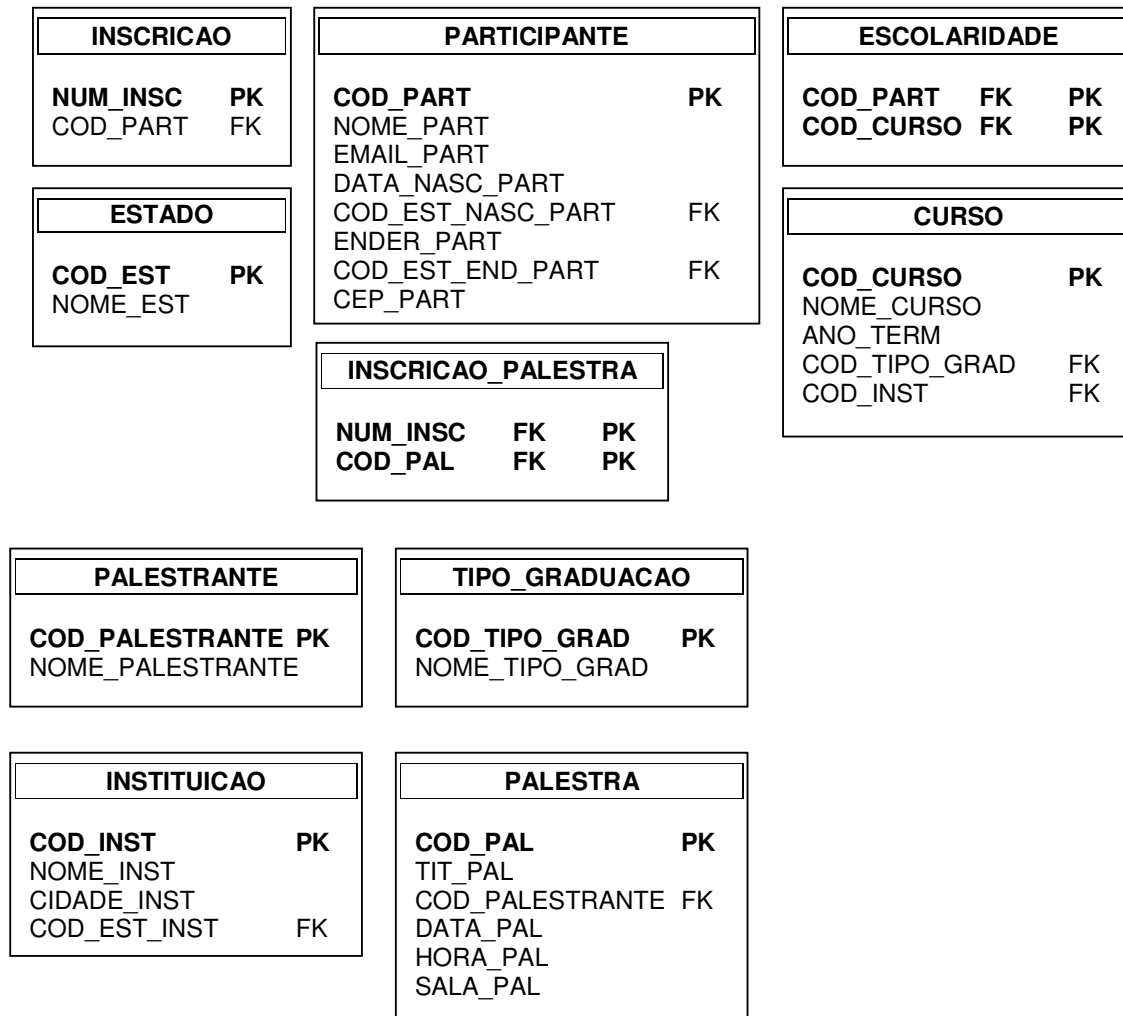
A criação de tabelas em um banco de dados relacional é feita a partir das DDL's produzidas no projeto físico, que devem ser descritas em seqüência pelo DBA segundo um *script*, de modo que a simples submissão desse *script* ao SGBD irá fazer a criação das tabelas descritas no mesmo. Após a criação dessas tabelas no SGBD, o DBA poderá verificar o resultado através de algum software de administração que esse SGBD disponibilize. Mesmo as DDL's submetidas são armazenadas em um Dicionário de Dados ou Arquivo de Metadados, ou qualquer que seja o nome dado a esse fim no SGBD em questão.

Para exemplificar a criação de tabelas em um SGBD, vejamos o exemplo a seguir:

Uma empresa de eventos deseja fazer o controle das inscrições dos participantes de um congresso em palestras e, para tanto, indicou uma ficha de inscrição como sendo a sua fonte de dados.

3.1.2. Mapeamento Relacional de Dados

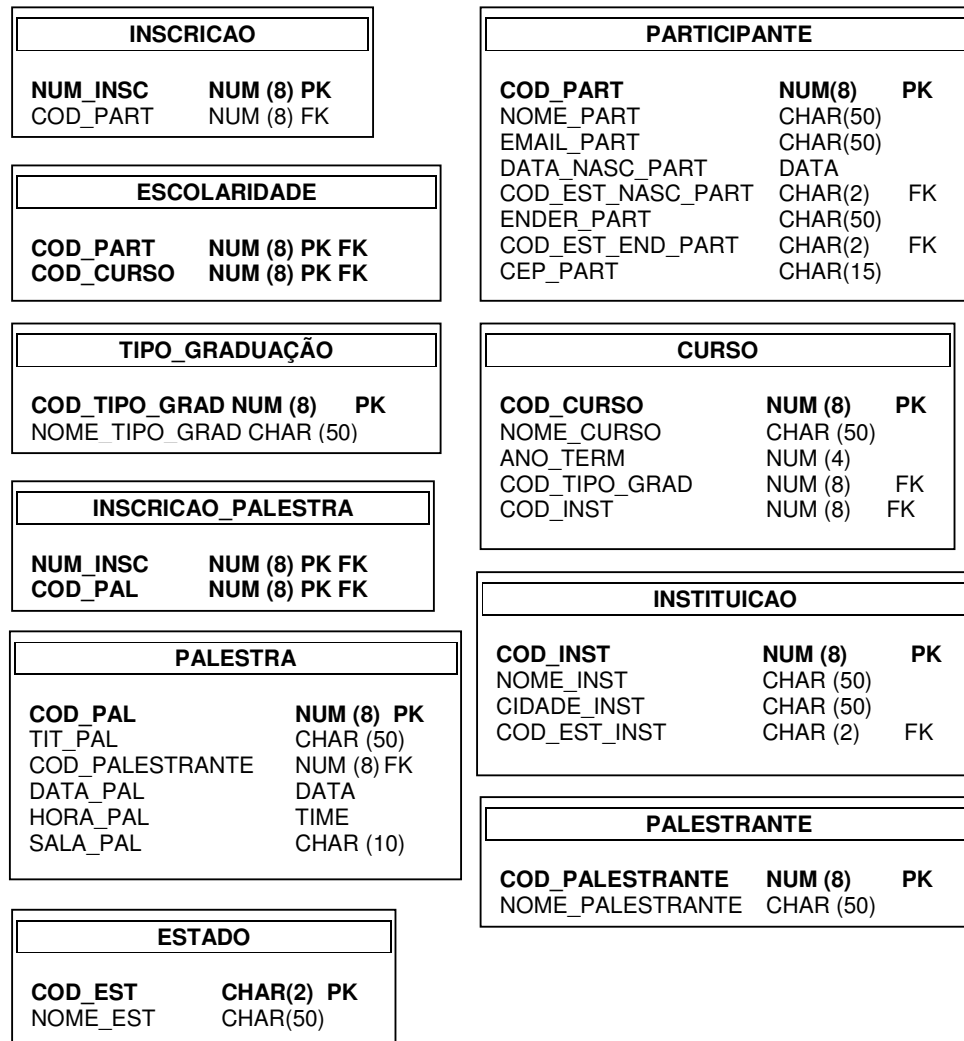
O projeto lógico do banco de dados que atende esse exemplo produziu o seguinte mapeamento relacional de dados:



3.1.3. Definição de Domínios

O projeto físico do banco de dados correspondente a esse exemplo produziu, a partir desse mapeamento de estruturas, a definição de domínios a seguir, onde convencionamos que:

- Os domínios numéricos são representados por NUM e o valor entre parênteses indica a quantidade máxima de caracteres numéricos permitida;
- Os domínios caracteres quaisquer são representados por CHAR e o valor entre parênteses indica a quantidade máxima de caracteres quaisquer permitidos.



3.1.4. Implementação do *script* DDL em SQL ANSI

A partir dessa definição de domínios, as DDL's produzidas em SQL ANSI são as seguintes:

Script de criação do Banco de Dados EVENTO

```
CREATE DATABASE EVENTO;
```

```
CREATE TABLE ESTADO (
    COD_EST CHAR (2) NOT NULL,
    NOME_EST CHAR (50) NOT NULL,
    PRIMARY KEY (COD_EST));
```

```
CREATE TABLE PARTICIPANTE (  
    COD_PART          NUMERIC (8, 0) NOT NULL,  
    NOME_PART         CHAR (50) NOT NULL,  
    EMAIL_PART        CHAR (50),  
    DATA_NASC_PART   DATE,  
    COD_EST_NASC_PART CHAR (2) NOT NULL,  
    ENDER_PART        CHAR (50),  
    COD_EST_END_PART  CHAR (2) NOT NULL,  
    CEP_PART          CHAR (15),  
    PRIMARY KEY (COD_PART),  
    FOREIGN KEY (COD_EST_END_PART) REFERENCES ESTADO  
    (COD_EST),  
    FOREIGN KEY (COD_EST_NASC_PART) REFERENCES ESTADO  
    (COD_EST));
```

```
CREATE TABLE INSCRICAO (  
    NUM_INSC NUMERIC (8, 0) NOT NULL,  
    COD_PART NUMERIC (8, 0) NOT NULL,  
    PRIMARY KEY (NUM_INSC),  
    FOREIGN KEY (COD_PART) REFERENCES PARTICIPANTE  
    (COD_PART));
```

```
CREATE TABLE TIPO_GRADUACAO (  
    COD_TIPO_GRAD     NUMERIC (8, 0) NOT NULL,  
    NOME_TIPO_GRAD    CHAR (50) NOT NULL,  
    PRIMARY KEY (COD_TIPO_GRAD));
```

```
CREATE TABLE INSTITUICAO (  
    COD_INST          NUMERIC (8, 0) NOT NULL,  
    NOME_INST         CHAR (50) NOT NULL,  
    CIDADE_INST       CHAR (50) NOT NULL,
```

```
COD_EST_INST CHAR (2) NOT NULL,  
PRIMARY KEY (COD_INST),  
FOREIGN KEY (COD_EST_INST) REFERENCES ESTADO (COD_EST));
```

```
CREATE TABLE CURSO (  
    COD_CURSO NUMERIC (8, 0) NOT NULL,  
    NOME_CURSO CHAR (50) NOT NULL,  
    ANO_TERM NUMERIC (4, 0),  
    COD_TIPO_GRAD NUMERIC (8, 0) NOT NULL,  
    COD_INST NUMERIC (8, 0) NOT NULL,  
    PRIMARY KEY (COD_CURSO),  
    FOREIGN KEY (COD_INST) REFERENCES INSTITUICAO (COD_INST),  
    FOREIGN KEY (COD_TIPO_GRAD) REFERENCES TIPO_GRADUACAO  
    (COD_TIPO_GRAD));
```

```
CREATE TABLE ESCOLARIDADE (  
    COD_PART NUMERIC (8, 0) NOT NULL,  
    COD_CURSO NUMERIC (8, 0) NOT NULL,  
    PRIMARY KEY (COD_PART, COD_CURSO),  
    FOREIGN KEY (COD_CURSO)  
    REFERENCES CURSO (COD_CURSO),  
    FOREIGN KEY (COD_PART) REFERENCES PARTICIPANTE  
    (COD_PART));
```

```
CREATE TABLE PALESTRANTE (  
    COD_PALESTRANTE NUMERIC (8, 0) NOT NULL,  
    NOME_PALESTRANTE CHAR (50) NOT NULL,  
    PRIMARY KEY (COD_PALESTRANTE));
```

```
CREATE TABLE PALESTRA (  
    COD_PAL NUMERIC (8, 0) NOT NULL,  
    TIT_PAL CHAR (50) NOT NULL,
```

```
COD_PALESTRANTE    NUMERIC (8, 0) NOT NULL,  
DATA_PAL           DATE NOT NULL,  
SALA_PAL           CHAR (10) NOT NULL,  
PRIMARY KEY (COD_PAL),  
FOREIGN KEY (COD_PALESTRANTE) REFERENCES PALESTRANTE  
(COD_PALESTRANTE));
```

```
CREATE TABLE INSCRICAO_PALESTRA (  
    NUM_INSC NUMERIC (8, 0) NOT NULL,  
    COD_PAL  NUMERIC (8, 0) NOT NULL,  
    PRIMARY KEY (NUM_INSC, COD_PAL),  
    FOREIGN KEY (COD_PAL) REFERENCES PALESTRA (COD_PAL),  
    FOREIGN KEY (NUM_INSC) REFERENCES INSCRICAO (NUM_INSC));
```

3.1.5. Implementação do *script* DDL em SQL de um SGBD específico (MySQL)

Quando da implementação de um banco de dados, é necessário que tenha sido definido previamente o SGBD. Todos os SGBD's relacionais atualmente atendem ao padrão SQL ANSI. Entretanto, cada um acrescenta às sentenças SQL características próprias, como pontuação diferente, obrigatoriedade de descrição de sentenças em minúsculas, nomes de dados entre aspas ou apóstrofes, ordem dos fatores diferentes do ANSI. Portanto, à vezes é necessário fazer adaptações ao *script* SQL ANSI produzido no projeto físico. A seguir, apresentamos a descrição do *script* SQL correspondente ao exemplo anterior, quando aplicado para o SGBD MySQL:

Script de criação do Banco de Dados EVENTO

```
create database evento;  
  
create table estado (  
    cod_est char(2) not null,  
    nome_est char(50) not null,
```

```
primary key(cod_est));
```

```
create table participante (  
cod_part decimal(8,0) not null,  
nome_part char(50) not null,  
email_part char(50),  
data_nasc_part date,  
cod_est_nasc_part char(2) not null,  
ender_part char(50),  
cod_est_end_part char(2) not null,  
cep_part char(15),  
primary key (cod_part),  
constraint foreign key estado_ender_fkey(cod_est_end_part) references  
estado(cod_est),  
constraint foreign key estado_nasc_fkey(cod_est_nasc_part) references  
estado(cod_est));
```

```
create table inscricao (  
num_insc decimal(8,0) not null,  
cod_part decimal(8,0) not null,  
primary key (num_insc),  
constraint foreign key participante_fkey(cod_part) references  
participante(cod_part));
```

```
create table tipo_graduacao (  
cod_tipo_grad decimal(8,0) not null,  
nome_tipo_grad char(50) not null,  
primary key (cod_tipo_grad));
```

```
create table instituicao (  
cod_inst decimal(8,0) not null,  
nome_inst char(50) not null,
```

```
cidade_inst char(50) not null,  
cod_est_inst char(2) not null,  
primary key(cod_inst),  
constraint foreign key estado_fkey(cod_est_inst) references estado(cod_est));
```

```
create table curso (  
cod_curso decimal(8,0) not null,  
nome_curso char(50) not null,  
ano_term decimal(4,0),  
cod_tipo_grad decimal(8,0) not null,  
cod_inst decimal(8,0) not null,  
primary key(cod_curso),  
constraint foreign key instituicao_fkey(cod_inst) references  
instituicao(cod_inst),  
constraint foreign key tipo_graduacao_fkey(cod_tipo_grad) references  
tipo_graduacao(cod_tipo_grad));
```

```
create table escolaridade (  
cod_part decimal(8,0) not null,  
cod_curso decimal(8,0) not null,  
primary key(cod_part, cod_curso),  
constraint foreign key curso_fkey(cod_curso) references curso(cod_curso),  
constraint foreign key participante_fkey(cod_part) references  
participante(cod_part));
```

```
create table palestrante (  
cod_palestrante decimal(8,0) not null,  
nome_palestrante char(50) not null,  
primary key (cod_palestrante));
```

```
create table palestra (  
cod_pal decimal(8,0) not null,
```



```
tit_pal char(50) not null,  
cod_palestrante decimal(8,0) not null,  
data_pal date not null,  
hora_pal time not null,  
sala_pal char(10) not null,  
primary key(cod_pal),  
constraint foreign key palestrante_fkey(cod_palestrante) references  
palestrante(cod_palestrante));
```

```
create table inscricao_palestra (  
num_insc decimal(8,0) not null,  
cod_pal decimal(8,0) not null,  
primary key(num_insc, cod_pal),  
constraint foreign key palestra_fkey(cod_pal) references palestra(cod_pal),  
constraint foreign key inscricao_fkey(num_insc) references  
inscricao(num_insc));
```

3.1.6. Manutenção de tabelas

Sempre é possível que erros possam ser cometidos quando da transcrição das DDL's do projeto físico para a construção do *script* a ser submetido ao SGBD. Também é possível que se deseje fazer modificações no esquema do banco de dados. Portanto, para sanar esses problemas, o SQL ANSI dispõe de sentenças específicas para tratar alterações de esquemas ou mesmo eliminação de partes destes.

a) Alteração de tabelas

Quando da alteração de uma tabela, normalmente, o que se quer fazer é acrescentar uma ou mais colunas a essa tabela, modificar características do domínio da mesma, ou até eliminar uma coluna. A seguir são apresentadas, através de exemplos, as sentenças SQL ANSI que permitem essa alteração de tabelas.

Acrescentando colunas:

```
ALTER TABLE PARTICIPANTE ADD (  
    BAIRRO_PART CHAR (20) NOT NULL,  
    APELIDO_PART CHAR (20) NOT NULL);
```

- Foram acrescentadas as colunas BAIRRO_PART e APELIDO_PART às demais colunas da tabela PARTICIPANTE.

Alterando especificações de colunas:

```
ALTER TABLE PARTICIPANTE MODIFY (  
    BAIRRO_PART CHAR (30) NOT NULL);
```

- Observe que o tamanho da coluna BAIRRO_PART foi alterado de 20 para 30.

Eliminando colunas:

```
ALTER TABLE PARTICIPANTE DROP (  
    BAIRRO_PART);
```

- A coluna BAIRRO_PART foi eliminada da tabela PARTICIPANTE.

1.1. ATUALIZAÇÃO DE DADOS

Ao final deste assunto, você será capaz de:

- Fazer a atualização dos dados de uma tabela previamente criada num banco de dados através da utilização de sentenças SQL

Neste momento, o que se estará tratando são as instâncias de dados e, para tanto, pode ser necessário fazer a atualização dos dados de tabelas através da inclusão, alteração ou exclusão de dados dessas tabelas.

Inclusão de Dados

Uma vez criadas as tabelas, a primeira coisa a ser feita é popular essas tabelas, ou seja, incluir dados, de modo que as tabelas possam ter dados para estarem disponíveis quando necessário.

A sentença SQL que possibilita essa inclusão de dados nas linhas de uma tabela é apresentada através do exemplo a seguir:

```
INSERT INTO ESTADO VALUES ('PA', 'PARÁ');  
INSERT INTO ESTADO VALUES ('AM', 'AMAZONAS');
```

Desta forma, estarão sendo incluídos os dados dos estados do PARÁ e AMAZONAS.

Outra forma de escrever as mesmas sentenças é a seguinte:

```
INSERT INTO ESTADO (NOME_EST, COD_EST)  
VALUES ('PARÁ', 'PA');  
  
INSERT INTO ESTADO (NOME_EST, COD_EST)  
VALUES ('AMAZONAS', 'AM');
```

A visualização do resultado dessa inclusão de dados seria o seguinte:

COD_EST	NOME_EST
AM	AMAZONAS
PA	PARÁ

Alteração de Dados

Como parte do processamento normal dos sistemas, os dados inseridos em tabelas precisam ser atualizados e, neste caso, isso pode implicar em modificação de valores de algumas colunas de tabelas.

A sentença SQL que possibilita essa alteração de dados nas linhas de uma tabela é apresentada através do exemplo a seguir

Supondo que foi criada uma palestra na tabela PALESTRA, vista a seguir:

COD_PAL	TIT_PAL	COD_PALESTRANTE	DATA_PAL	HORA_PAL	SALA_PAL
110	DESIGN PATTERNS	114	14/10/2003	18:30	D-200
213	QoS EM VOZ SOBRE IP	23	23/11/2003	19:30	LA-211
235	DATA MINING UTILIZANDO REDES NEURAIAS	142	05/09/2004	20:00	D-200

Pode ser necessário fazer alteração em valores das linhas dessa tabela, como a mudança de sala de uma determinada palestra, como na sentença seguinte:

```
UPDATE PALESTRA SET SALA_PAL = 'B-100' WHERE COD_PAL = 213;
```

Após essa modificação a tabela ficaria da seguinte forma:

COD_PAL	TIT_PAL	COD_PALESTRANTE	DATA_PAL	HORA_PAL	SALA_PAL
110	DESIGN PATTERNS	114	14/10/2003	18:30	D-200
213	QoS EM VOZ SOBRE IP	23	23/11/2003	19:30	B-100
235	DATA MINING UTILIZANDO REDES NEURAIAS	142	05/09/2004	20:00	D-200

Exclusão de Dados

Ainda, como parte do processamento normal dos sistemas, os dados, é possível que se precise eliminar linhas de uma tabela, ou por erro ou simplesmente como parte da lógica de algum tipo de rotina.

A sentença SQL que possibilita essa exclusão de dados nas linhas de uma tabela é apresentada através do exemplo a seguir.

Considerando que a tabela PALESTRA tem as seguintes linhas:

COD_PAL	TIT_PAL	COD_PALESTRANTE	DATA_PAL	HORA_PAL	SALA_PAL
110	DESIGN PATTERNS	114	14/10/2003	18:30	D-200
213	QoS EM VOZ SOBRE IP	23	23/11/2003	19:30	B-100
235	DATA MINING UTILIZANDO REDES NEURAIAS	142	05/09/2004	20:00	D-200

Ao submeter ao SGBD a seguinte sentença:

DELETE FROM PALESTRA WHERE SALA_PAL = 'D-200';

Teremos o seguinte resultado:

COD_PAL	TIT_PAL	COD_PALESTRANTE	DATA_PAL	HORA_PAL	SALA_PAL
213	QoS EM VOZ SOBRE IP	23	23/11/2003	19:30	B-100

Considerações finais

Quando for necessária a execução de sentenças de atualização em bloco, ou seja, diversas sentenças INSERT, UPDATE e DELETE tiverem que ser submetidas ao SGBD de modo que todas sejam aceitas ou rejeitadas em conjunto, é necessário que se utilize outras sentenças SQL para esse fim:

- Para aceitar o bloco de atualizações é preciso submeter a sentença **COMMIT** após esse conjunto de sentenças;
- Para rejeitar o bloco de atualizações é preciso submeter a sentença **ROLLBACK** após esse conjunto de sentenças.

Observações:

- Alguns SGBDs tem a submissão de COMMIT automaticamente após a submissão de INSERTs, UPDATEs e DELETEs. Neste caso, o ROLLBACK não irá funcionar.

- Em alguns SGBDs a configuração de COMMIT automático é configurável.

ATIVIDADE ⁽¹⁾

1. Crie o banco de dados relacional que foi visto na 1ª avaliação utilizando o MS-Access:

Reside (nome_pessoa, rua, cidade)

Trabalha (nome_pessoa, nome_companhia, salário)

Localizada_em (nome_companhia, cidade)

Gerencia (nome_pessoa, nome_gerente)

Para criar cada tabela utilize os seguintes comandos:

CREATE TABLE Reside

(nome_pessoa varchar(10),
rua varchar(10),
cidade varchar(10),

PRIMARY KEY (nome_pessoa));

CREATE TABLE Trabalha

(nome_pessoa varchar(10),
nome_companhia varchar(15),
salario money,

PRIMARY KEY (nome_pessoa, nome_companhia),

FOREIGN KEY (nome_pessoa) REFERENCES Reside,

FOREIGN KEY (nome_companhia) REFERENCES Localizada_em);

CREATE TABLE Localizada_em

(nome_companhia varchar(15),
cidade varchar(10),

PRIMARY KEY (nome_companhia));

CREATE TABLE Gerencia

(nome_pessoa varchar(10),

¹ ROCHA NETO, Aluizio Ferreira da. Apostila de BANCO DE DADOS I: 3º. Ano de Bacharelado em Sistema de Informação da Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte (FARN), 1º. Semestre de 2001.

```
nome_gerente varchar(10),  
PRIMARY KEY (nome_pessoa, nome_gerente),  
FOREIGN KEY (nome_pessoa) REFERENCES Reside,  
FOREIGN KEY (nome_gerente) REFERENCES Reside(nome_pessoa));
```

De acordo com os comandos acima:

- a) Informe a chave primária de cada tabela.
- b) Informe as restrições de domínio de cada atributo das tabelas.
- c) Informe a integridade referencial entre as tabelas.
- d) Clique no botão “Relacionamentos” do MS-Access e observe o resultado. O que você está vendo como resultado?
- e) Informe, a partir do seu entendimento do Sistema de Informação em questão, as dependências funcionais entre os atributos de cada tabela.

2. Crie o banco de dados relacional que foi visto prova de reposição da 1a avaliação utilizando o MS-Access:

Clientes (cpf, nome, bairro, cidade)

Funcionários (cpf, nome, departamento, salário)

Pedidos (numped, cpf_cliente, cpf_func, data)

PedidosItens (numped, numitem)

Itens (numitem, descrição, valor)

Para criar cada tabela, observe (e teste) a cláusula SQL para:

- a) Identificar a(s) chave(s) primária(s) de cada tabela.
- b) Quais são as restrições de domínio de cada atributo.
- c) Identificar a integridade referencial entre as tabelas.
- d) Clique novamente no botão “Relacionamentos” do Access e observe o resultado.
- e) Quais são as dependências funcionais entre os atributos de cada tabela.