Trabalhando com tabelas (restrições)

Prof. Ramon Souza

Restrições

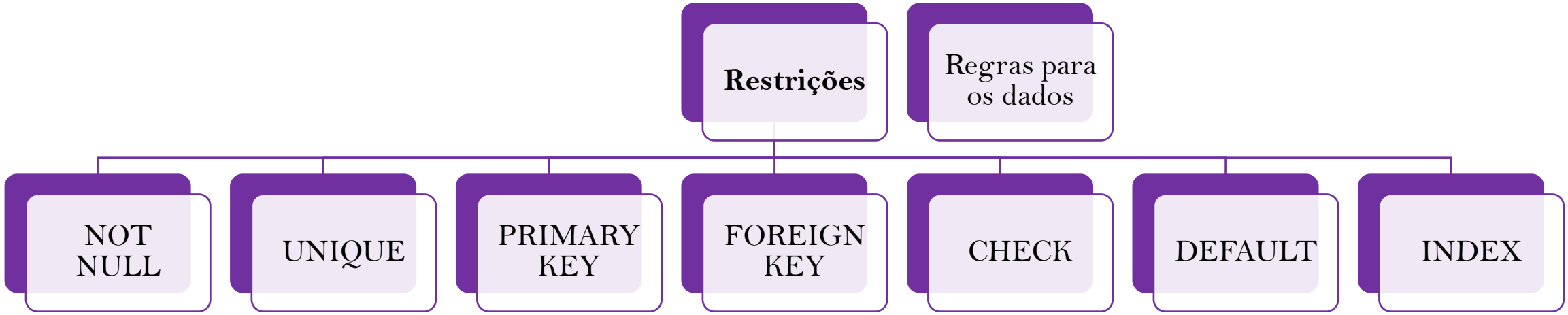
As **restrições SQL (constraints)** são usadas para **especificar regras para os dados em uma tabela**.

As **restrições** são usadas para **limitar o tipo de dados que podem ser colocados em uma tabela**. Isso garante a precisão e a confiabilidade dos dados na tabela. Se houver alguma violação entre a restrição e a ação de dados, a ação será abortada.

Restrições

As **restrições** podem ser no nível da coluna ou no nível da tabela. As restrições de nível de coluna se aplicam a uma coluna e as restrições de nível de tabela se aplicam à tabela inteira.

Restrições



NOT NULL

Por padrão, uma coluna pode conter valores NULL.

A restrição **NOT NULL** impõe a uma coluna a regra para NÃO aceitar valores NULL. Isso **obriga um campo a sempre conter um valor**, o que significa que você não pode inserir um novo registro ou atualizar um registro sem adicionar um valor a esse campo.

NOT NULL


```
CREATE TABLE nome_da_tabela (
    coluna1 tipo_de_dado NOT NULL,
    coluna2 tipo_de_dado,
    coluna3 tipo_de_dado,
   ....
);
OU
ALTER TABLE nome_da_tabela
MODIFY coluna2 tipo_de_dado NOT NULL;
```

NOT NULL (exemplo)


```
CREATE TABLE Pessoas (
    IDPessoa int NOT NULL,
    Sobrenome varchar(255) NOT NULL,
    Nome varchar(255) NOT NULL,
    Endereco varchar(255),
    Cidade varchar(255)
);
```

```
ALTER TABLE Pessoas
MODIFY Cidade varchar(255) NOT NULL;
```

UNIQUE

A restrição **UNIQUE** garante que **todos os valores em uma coluna sejam diferentes**. As restrições UNIQUE e PRIMARY KEY fornecem uma garantia de exclusividade para uma coluna ou conjunto de colunas.

Uma restrição PRIMARY KEY tem automaticamente uma restrição UNIQUE. No entanto, você pode ter muitas restrições UNIQUE por tabela, mas apenas uma restrição PRIMARY KEY por tabela.

UNIQUE

- **No SQL Server / Oracle / MS Access:**

```
CREATE TABLE nome_da_tabela (
    coluna1 tipo_de_dado UNIQUE,
    coluna2 tipo_de_dado,
    coluna3 tipo_de_dado,
);
```

UNIQUE

- **No MySQL**:

```
CREATE TABLE nome_da_tabela (
    coluna1 tipo_de_dado,
    coluna2 tipo_de_dado,
    coluna3 tipo_de_dado,
UNIQUE (coluna1)
);
```

- **No MySQL / SQL Server / Oracle / MS Access para múltiplas colunas**:

```
CREATE TABLE nome_da_tabela (
    coluna1 tipo_de_dado,
    coluna2 tipo_de_dado,
    coluna3 tipo_de_dado,
CONSTRAINT nome_da_restricao UNIQUE (coluna1, coluna2));
);
```

UNIQUE

Também é possível adicionar uma restrição UNIQUE em uma cláusula ALTER:

- **Para uma coluna**:

```
ALTER TABLE nome_da_tabela
ADD UNIQUE (coluna);
```

- **Para múltiplas colunas**:

```
ALTER TABLE nome_da_tabela
ADD CONSTRAINT nome_da_restricao UNIQUE (coluna1, coluna2));
```

Para excluir uma restrição, basta utilizar a cláusula DROP:

- **No MySQL**, usa-se DROP INDEX:

```
ALTER TABLE nome_da_tabela
DROP INDEX nome_da_restricao;
```

- **No SQL Server / Oracle / MS Access**, usa-se DROP CONSTRAINT:

```
ALTER TABLE nome_da_tabela
DROP CONSTRAINT nome_da_restricao;
```

PRIMARY KEY

A restrição **PRIMARY KEY** **identifica exclusivamente cada registro em uma tabela**. As chaves primárias devem conter valores UNIQUE e não podem conter valores NULL.

Uma tabela pode ter apenas uma chave primária; e na tabela, essa chave primária pode consistir em colunas únicas ou múltiplas (campos).

PRIMARY KEY

- **No SQL Server / Oracle / MS Access:**

```
CREATE TABLE nome_da_tabela (
    coluna1 tipo_de_dado PRIMARY KEY,
    coluna2 tipo_de_dado;
);
```

PRIMARY KEY

- **No MySQL**:

```
CREATE TABLE nome_da_tabela (
    coluna1 tipo_de_dado,
    coluna2 tipo_de_dado,
PRIMARY KEY (coluna1)
);
```

PRIMARY KEY

- **No MySQL / SQL Server / Oracle / MS Access para múltiplas colunas**:

```
CREATE TABLE nome_da_tabela (
    coluna1 tipo_de_dado,
    coluna2 tipo_de_dado,
    coluna3 tipo_de_dado,
CONSTRAINT nome_da_restricao PRIMARY KEY (coluna1, coluna2));
);
```

PRIMARY KEY

Também é possível adicionar uma restrição **PRIMARY KEY** em uma cláusula ALTER:

- **Para uma coluna**:

```
ALTER TABLE nome_da_tabela
ADD PRIMARY KEY(coluna);
```

- **Para múltiplas colunas**:

```
ALTER TABLE nome_da_tabela
ADD CONSTRAINT nome_da_restricao PRIMARY KEY (coluna1, coluna2));
```

PRIMARY KEY

Para excluir uma restrição, basta utilizar a cláusula DROP:

- **No MySQL**:

```
ALTER TABLE nome_da_tabela
DROP PRIMARY KEY;
```

- **No SQL Server / Oracle / MS Access**:

```
ALTER TABLE nome_da_tabela
DROP CONSTRAINT nome_da_restricao;
```

FOREIGN KEY

Uma **FOREIGN KEY** é uma **chave usada para unir duas tabelas**, sendo um campo (ou conjunto de campos) em uma tabela que se refere à PRIMARY KEY em outra tabela.

A tabela que contém a chave estrangeira é chamada de tabela filha e a tabela que contém a chave candidata é chamada de tabela de referência ou pai.

FOREIGN KEY

- **No SQL Server / Oracle / MS Access:**

```
CREATE TABLE nome_da_tabela (
    coluna1 tipo_de_dado PRIMARY KEY,
    coluna2 tipo_de_dado,
    coluna3 tipo_de_dado FOREIGN KEY REFERENCES
tabela_referenciada(chave),
);
```

FOREIGN KEY

- **No MySQL**:

```
CREATE TABLE nome_da_tabela (
    coluna1 tipo_de_dado,
    coluna2 tipo_de_dado,
    coluna3 tipo_de_dado,
PRIMARY KEY (coluna1),
FOREIGN KEY (coluna2) REFERENCES tabela_referenciada (chave)
);
```

FOREIGN KEY

- **No MySQL / SQL Server / Oracle / MS Access para múltiplas colunas:**

```
CREATE TABLE nome_da_tabela (
    coluna1 tipo_de_dado,
    coluna2 tipo_de_dado,
    coluna3 tipo_de_dado,
CONSTRAINT nome_da_restricao FOREIGN KEY (coluna1, coluna2))
REFERENCES tabela_referenciada (chave1, chave2);
);
```

FOREIGN KEY

Também é possível adicionar uma restrição **FOREIGN KEY** em uma cláusula ALTER:

- **Para uma coluna**:

```
ALTER TABLE nome_da_tabela
ADD FOREIGN KEY(coluna) REFERENCES tabela_referenciada (chave);
```

- **Para múltiplas colunas**:

```
ALTER TABLE nome_da_tabela
ADD CONSTRAINT nome_da_restricao FOREIGN KEY(coluna1, coluna2) REFERENCES tabela_referenciada (chave1, chave2);
```

FOREIGN KEY

Para excluir uma restrição, basta utilizar a cláusula DROP:

- **No MySQL**:

```
ALTER TABLE nome_da_tabela
DROP FOREIGN KEY coluna;
```

- **No SQL Server / Oracle / MS Access**:

```
ALTER TABLE nome_da_tabela
DROP CONSTRAINT nome_da_restricao;
```

FOREIGN KEY

Ao usar a opção **ON DELETE CASCADE**, quando um registro da tabela que possui a chave primária associada a esta chave estrangeira for excluído, então os registros associados também são excluídos.

CONSTRAINT nome_da_restricao **FOREIGN KEY** (coluna1, coluna2)) **REFERENCES** tabela_referenciada (chave1, chave2) **ON DELETE CASCADE**;

Ao usar a opção **ON UPDATE CASCADE**, quando um registro da tabela que possui a chave primária associada a esta chave estrangeira for alterado, então os registros associados também são alterados.

CHECK

A restrição **CHECK** é **usada para limitar o intervalo de valores que pode ser colocado em uma coluna**.

Se você definir uma restrição **CHECK** em uma única coluna, ela permitirá apenas determinados valores para essa coluna.

Se você definir uma restrição **CHECK** em uma tabela, ela poderá limitar os valores em determinadas colunas com base nos valores de outras colunas na linha.

- **No SQL Server / Oracle / MS Access:**

```
CREATE TABLE nome_da_tabela (
    coluna1 tipo_de_dado,
    coluna2 tipo_de_dado CHECK (condicao),
    coluna3 tipo_de_dado,
);
```

CHECK

- **No MySQL**:

```
CREATE TABLE nome_da_tabela (
    coluna1 tipo_de_dado,
    coluna2 tipo_de_dado,
    coluna3 tipo_de_dado,
    CHECK (condicao)
);
```

- **No MySQL / SQL Server / Oracle / MS Access para uma ou múltiplas colunas**:

```
CREATE TABLE nome_da_tabela (
    coluna1 tipo_de_dado,
    coluna2 tipo_de_dado,
    coluna3 tipo_de_dado,
CONSTRAINT nome_da_restricao CHECK (condicao1 AND condicao2));
);
```

CHECK

Também é possível adicionar uma restrição CHECK em uma cláusula ALTER:

- **Para uma coluna**:

```
ALTER TABLE nome_da_tabela
ADD CHECK (condicao);
```

- **Para múltiplas colunas**:

```
ALTER TABLE nome_da_tabela
ADD CONSTRAINT nome_da_restricao CHECK (condicao1 AND condicao2));
```

Para excluir uma restrição, basta utilizar a cláusula DROP:

- **No MySQL**:

```
ALTER TABLE nome_da_tabela
DROP CHECK nome_da_restricao;
```

- **No SQL Server / Oracle / MS Access**:

```
ALTER TABLE nome_da_tabela
DROP CONSTRAINT nome_da_restricao;
```

DEFAULT

A restrição **DEFAULT** é usada para **fornecer um valor padrão para uma coluna**. O valor padrão será adicionado a todos os novos registros SE nenhum outro valor for especificado.

DEFAULT

A definição restrição **DEFAULT** pode ser realizada com a seguinte sintaxe:

```
CREATE TABLE nome_da_tabela (
    coluna1 tipo_de_dado,
    coluna2 tipo_de_dado DEFAULT valor,
    coluna3 tipo_de_dado,
);
```

DEFAULT

Também é possível adicionar uma restrição DEFAULT em uma cláusula ALTER:

- **No MySQL**:

```
ALTER TABLE nome_da_tabela

ALTER coluna SET DEFAULT valor;
```

- **No SQL Server**:

```
ALTER TABLE nome_da_tabela

ADD CONSTRAINT nome_da_restricao DEFAULT valor;
```

- **No Oracle**:

```
ALTER TABLE nome_da_tabela

MODIFY coluna DEFAULT valor;
```

DEFAULT

Para excluir uma restrição, basta utilizar a cláusula DROP:

- **No MySQL**:

```
ALTER TABLE nome_da_tabela
ALTER coluna DROP DEFAULT;
```

- **No SQL Server / Oracle / MS Access**:

```
ALTER TABLE nome_da_tabela
ALTER COLUMN coluna DROP DEFAULT;
```

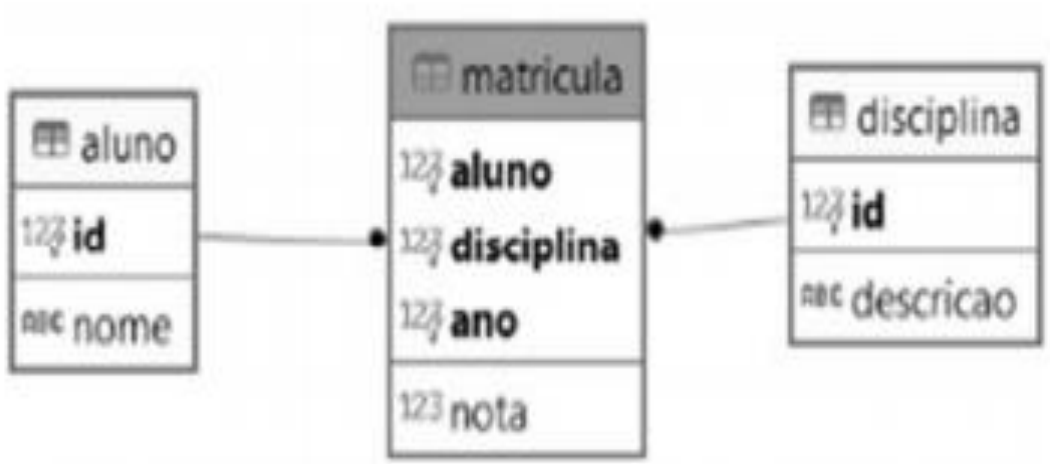
(CESPE / CEBRASPE - 2020 - Ministério da Economia - Tecnologia da Informação - Desenvolvimento de Software)

Tendo como referência o diagrama de entidade relacionamento precedente, julgue o próximo item, a respeito de linguagem de definição de dados e SQL.

As expressões DDL a seguir permitem a criação das tabelas presentes no diagrama apresentado.

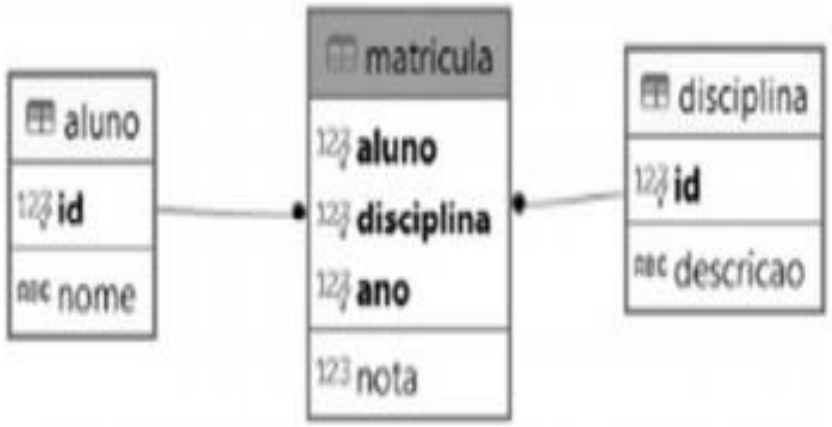
Questão de Concurso


```
create table aluno (
        id integer primary key,
        nome varchar(40) );
        create table disciplina (
        id integer primary key,
        descricao varchar(60)
);
```



Questão de Concurso


```
create table matricula (
    aluno integer,
    disciplina integer,
    ano integer,
    nota numeric,
    constraint pk_matricula primary key (aluno,
    disciplina, ano),
    constraint fk_matricula_aluno foreign key
    (aluno)
    references aluno,
    constraint fk_matricula_disciplina foreign
    key (disciplina)
    references disciplina );
```

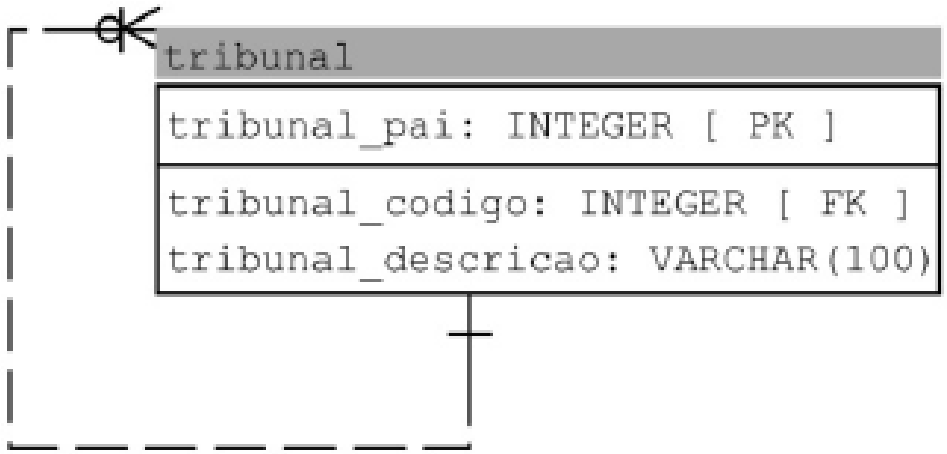


(CESPE - 2016 - TCE-PA - Auditor de Controle Externo - Área Informática - Analista de Sistema)

Julgue o próximo item, relativo à linguagem de definição de dados (DDL).

A expressão DDL abaixo cria a tabela referente ao diagrama de entidade e relacionamento apresentado a seguir.

```
create table tribunal(
tribunal_codigo integer ,
tribunal_descricao varchar(100),
tribunal_pai integer primary key,
constraint fk_tribunal
foreign key (tribunal_codigo )
references tribunal
```



(CESPE - 2016 - POLÍCIA CIENTÍFICA - PE - Perito Criminal - Ciência da Computação) Na linguagem SQL, o comando create table é usado para criar uma tabela no banco de dados; enquanto o relacionamento entre duas tabelas pode ser criado pela declaração

a) null.

b) primary key.

c) constraint.

d) auto_increment.

e) not null.

(CESPE - 2015 - MEC – Desenvolvedor)

```
CREATE TABLE PESSOA (
ID INTEGER NOT NULL,
NOME CHAR(50) NOT NULL UNIQUE,
CPF DECIMAL (11,0) NULL,
NACIONALIDADE INTEGER NOT NULL,
PRIMARY KEY (ID),
FOREIGN KEY (NACIONALIDADE)
REFERENCES TABELA_NACIONALIDADE(CODIGO_NACIONALIDADE)
);
```

Com base no comando SQL apresentado, julgue o item subsequente.

A cláusula NULL na coluna CPF indica que o conteúdo dessa coluna pode ser zero, já que ela é do tipo DECIMAL (11,0).

(CESPE - 2015 - MEC – Desenvolvedor)

```
CREATE TABLE PESSOA (
ID INTEGER NOT NULL,
NOME CHAR(50) NOT NULL UNIQUE,
CPF DECIMAL (11,0) NULL,
NACIONALIDADE INTEGER NOT NULL,
PRIMARY KEY (ID),
FOREIGN KEY (NACIONALIDADE)
REFERENCES TABELA_NACIONALIDADE(CODIGO_NACIONALIDADE)
);
```

Com base no comando SQL apresentado, julgue o item subsequente.

Mais de uma PESSOA pode ter o mesmo NOME e a mesma NACIONALIDADE.

(CESPE - 2014 - ANATEL - Analista Administrativo - Tecnologia da Informação e Comunicação) Nos comandos em linguagem de consulta estruturada (SQL) apresentados a seguir, as chaves primárias estão sublinhadas e apenas horas_gastas é do tipo numérico, os demais campos são do tipo caractere. Em uma tarefa, com a utilização de um mesmo veículo, pode haver a participação de mais de um motorista na função de auxiliar.

MOTORISTA (cod_mot, nome, cnh)

TAREFA (cod_tarefa, cod_mot, cod_mot_auxiliar, placa, descricao, horas_gastas)

VEÍCULO (placa, ano, modelo)

Tendo como base as informações acima, julgue o item a seguir.

O comando a seguir exclui a chave primária, cod_tarefa, da tabela TAREFA.

Alter table tarefa drop primary key cod_tarefa;

(CESPE - 2013 - BACEN - Analista - Análise e Desenvolvimento de Sistemas) Julgue os itens seguintes, a respeito das linguagens de banco de dados.

```
CREATE TABLE Pessoa
(
Id int NULL,
Matricula int NOT NULL,
Nome varchar(255) NOT NULL,
DataNascimento date NULL)

CREATE TABLE EnderecoPessoa
(Id int NOT NULL,
TipoEndereco char (1) NOT NULL,
Endereco varchar(255),
Cidade char(55),
UF varchar (2)
)
```

Considerando os scripts acima para criação das Tabelas Pessoa e EnderecoPessoa, julgue o item seguinte.

Considerando que o campo Id na tabela Pessoa esteja corretamente configurado como chave primária simples, para se criar uma chave estrangeira entre as tabelas Pessoa e EnderecoPessoa, deve-se executar o comando a seguir.

```
ALTER TABLE EnderecoPessoa
ADD CONSTRAINT fk_Endereco_Pessoa FOREIGN KEY (P_id) REFERENCES Pessoa (Id)
```

(CESPE - 2013 - BACEN - Analista - Análise e Desenvolvimento de Sistemas) Julgue os itens seguintes, a respeito das linguagens de banco de dados.

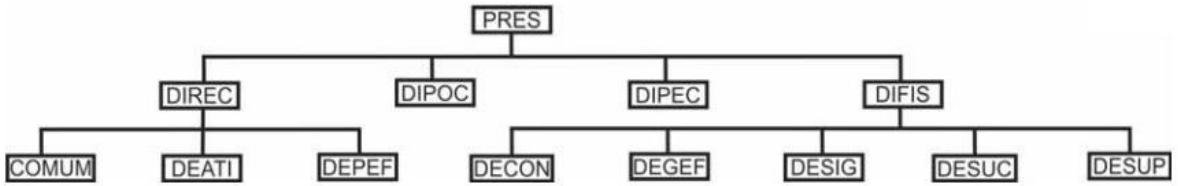
```
CREATE TABLE Pessoa
(
Id int NULL,
Matricula int NOT NULL,
Nome varchar(255) NOT NULL,
DataNascimento date NULL)

CREATE TABLE EnderecoPessoa
(Id int NOT NULL,
TipoEndereco char (1) NOT NULL,
Endereco varchar(255),
Cidade char(55),
UF varchar (2)
)
```

Considerando os scripts acima para criação das Tabelas Pessoa e EnderecoPessoa, julgue o item seguinte.

Para criar uma chave primária composta na Tabela Pessoa, deve-se executar o seguinte comando.

ALTER TABLE Pessoa ADD CONSTRAINT pk_PessoaID PRIMARY KEY (Id, Matricula)

(CESPE - 2013 - BACEN - Analista - Suporte à Infraestrutura de Tecnologia da Informação)



A expressão SQL abaixo cria uma tabela com estrutura que permite armazenar informações acerca dos órgãos e sua hierarquia.

```
create table orgao (
codigo integer,
sigla char(10),
codigo_pai integer,
constraint pk_orgao primary key (codigo),
constraint fk_orgao foreign key (codigo_pai) references órgão
)
```

(CESPE - 2013 - TRT - 10ª REGIÃO (DF e TO) - Técnico Judiciário - Tecnologia da Informação) Em relação aos comandos da linguagem SQL, julgue os itens seguintes.

O comando abaixo permite adicionar a tabela disciplinas a uma chave estrangeira com o nome fk_curso, do campo id_curso que pertence à tabela cursos.

```
alter table disciplinas
alter column fk_curso references cursos (id_curso);
```

(FCC - 2019 - SEFAZ-BA - Auditor Fiscal - Tecnologia da Informação - Prova II) Em um banco de dados aberto e em condições ideais há uma tabela chamada Contribuinte cuja chave primária é idContribuinte. Há também uma tabela chamada Imposto cuja chave primária é idimposto. Para criar uma tabela de associação chamada Contribuinte_imposto cuja chave primária é composta pelos campos idContribuinte e idImposto, que são chaves estrangeiras resultantes da relação dessa tabela com as tabelas Contribuinte e Imposto, utiliza-se a instrução SQL

a) CREATE TABLE Contribuinte_Imposto(idContribuinte INT, idImposto INT, PRIMARY KEY (idContribuinte), FOREIGN KEY (idContribuinte) REFERENCES Contribuinte (idContribuinte), PRIMARY KEY (idImposto), FOREIGN KEY (idContribuinte) REFERENCES Contribuinte (idContribuinte));

b) CREATE TABLE Contribuinte_Imposto(idContribuinte INT NOT NULL, idImposto INT NOT NULL, PRIMARY KEY (idContribuinte, idImposto), CONSTRAINT fk1 FOREIGN KEY (idContribuinte) REFERENCES Contribuinte (idContribuinte), CONSTRAINT fk2 FOREIGN KEY (idImposto) REFERENCES Imposto (idImposto));

c) CREATE TABLE Contribuinte_Imposto(idContribuinte INT NOT NULL, idImposto INT NOT NULL, PRIMARY KEY (idContribuinte, idImposto), FOREIGN KEY (idContribuinte) SOURCE Contribuinte (idContribuinte), FOREIGN KEY (idImposto) SOURCE Imposto (idImposto));

d) CREATE TABLE Contribuinte_Imposto(idContribuinte INT NOT NULL, idImposto INT NOT NULL, PRIMARY KEY (idContribuinte, idImposto), FOREIGN KEY (idContribuinte, idImposto) REFERENCES (Contribuinte!idContribuinte, Imposto!idImposto));

e) CREATE TABLE Contribuinte_Imposto(idContribuinte INT NOT NULL, idImposto INT NOT NULL, PRIMARY KEY (idContribuinte, idImposto), FOREIGN KEY (idContribuinte, idImposto) REFERENCES all parents);

(FCC - 2019 - SANASA Campinas - Analista de Tecnologia da Informação - Suporte de DBA-Banco de Dados)

Considere o código SQL abaixo, que gerou a tabela ItemFatura.

```
CREATE TABLE ItemFatura(
idItemFatura INT NOT NULL,
idFatura INT NOT NULL,
descItemFatura VARCHAR(45),
valorItemFatura DOUBLE,
..I..
);
```

Considerando que a tabela ItemFatura possui chave primária composta pelos campos idItemFatura e idFatura, e que se uma fatura for excluída, automaticamente serão excluídos todos os seus itens, a lacuna I deve ser preenchida corretamente por

a) PRIMARY KEY (idItemFatura, FOREIGN KEY(idFatura)) REFERENCES Fatura(idFatura) ON DELETE CASCADE

b) PRIMARY KEY (idItemFatura, idFatura), FOREIGN KEY(idFatura) FROM Fatura(idFatura) WITH DELETE CASCADE

c) PRIMARY KEY (idItemFatura, idFatura), FOREIGN KEY (idFatura) REFERENCES Fatura(idFatura) ON DELETE CASCADE

d) PRIMARY KEY (idItemFatura, idFatura), FOREIGN KEY(idFatura) REFERENCES Fatura(idFatura)

e) PRIMARY KEY (idItemFatura, idFatura), FOREIGN KEY(idFatura) FROM Fatura(idFatura) ON DELETE CASCADE

(FCC - 2018 - SABESP - Estagiário - Nível Médio Técnico)

Considere que foi criada uma tabela em um banco de dados relacional capaz de armazenar os dados de clientes da SABESP, usando o comando SQL: CREATE TABLE Clientes_Sabesp (Cliente VARCHAR (50), MedidorAnt INTEGER NOT NULL, MedidorAtual INTEGER NOT NULL, Periodo DATE NOT NULL, Conta DECIMAL (10,2), PRIMARY KEY (Cliente));
É correto afirmar que

a) a instrução PRIMARY KEY define que o campo Cliente será a chave primária simples da tabela e este campo não pode ser nulo nem repetido.

b) com exceção do campo Cliente todos os outros campos são do tipo numérico inteiro.

c) para criar uma chave estrangeira, basta acrescentar FOREIGN KEY (Cliente) após a definição da PRIMARY KEY.

d) a chave estrangeira de um banco de dados relacional é usada para criar relacionamentos com as demais tabelas do banco de dados, por isso deve ter o mesmo nome da chave primária seguido de _FK. Nesta tabela seria Cliente_FK.

e) há um erro neste comando: todos os campos da tabela devem ser NOT NULL e os campos Cliente e Conta não têm esta restrição.

(FCC - 2017 - DPE-RS - Técnico - Informática) Um Técnico está criando uma tabela filha chamada funcionario, que será relacionada a uma tabela pai chamada departamento, por meio da chave estrangeira. Como parte do comando CREATE TABLE, usado para criar a tabela filha, ele deseja estabelecer uma restrição de chave estrangeira chamada emp_dept_fk para o campo department_id, que fará referência ao campo department_id que é chave primária na tabela departamento. Esta restrição será criada corretamente se for utilizada, na criação da tabela funcionario, a instrução SQL

a) RESTRICTION emp_dept_fk FOREIGN KEY (department_id) PRIMARY KEY departmento(department_id)

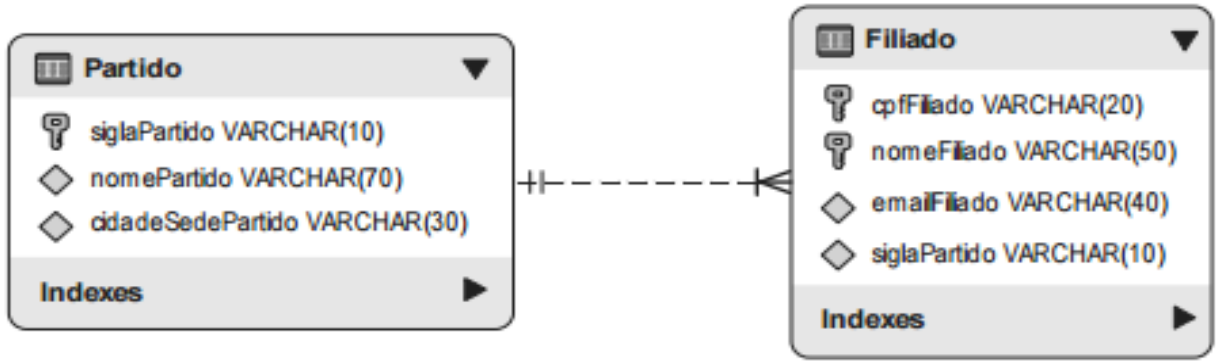
b) FOREIGN KEY emp_dept_fk FIELD(department_id) REFERENCES departmento(department_id)

c) CONSTRAINT emp_dept_fk FOREIGN KEY (department_id) REFERENCES departmento(department_id)

d) DEFINE CONSTRAINT emp_dept_fk FOREIGN KEY (department_id) WITH REFERENCES departmento department_id)

e) CONSTRAINT emp_dept_fk FOREIGN KEY (department_id) PRIMARY KEY departmento(department_id)

(FCC - 2016 - AL-MS - Técnico de Informática) Para responder à questão, considere o modelo mostrado na imagem abaixo, oriundo de uma situação hipotética:



Após criadas as tabelas Partido e Filiado, foram incluídos, respectivamente, os seguintes registros:

siglaPartido	nomePartido	cidadeSedePartido
PDT	Partido Democrático Trabalhista	Brasília
PMDB	Partido do Movimento Democrático Brasileiro	Brasília
PSDB	Partido da Social Democracia Brasileira	São Paulo

cpfFiliado	nomeFiliado	emailFiliado	siglaPartido
124.179.156-10	André Braga	braga@hotmail.com	PMDB
147.189.237-18	Marcos Pereira	mpereira@hotmail.com	PDT
154.496.172-14	Pedro Silva	pedro@gmail.com	PDT
192.345.176-01	Maria Souza	maria@ig.com.br	PSDB

Questão de Concurso

Após a tabela Partido ser criada, para criar a tabela Filiado foi utilizada a instrução abaixo:

```
CREATE TABLE IF NOT EXISTS Filiado (
 cpfFiliado VARCHAR(20) NOT NULL,
 nomeFiliado VARCHAR(50),
 emailFiliado VARCHAR(40),
 siglaPartido VARCHAR(10) NOT NULL,
 PRIMARY KEY (cpfFiliado),
 FOREIGN KEY (siglaPartido)
I Partido (siglaPartido)
);
```

A lacuna I deverá ser corretamente preenchida por

a) CASCADE CONSTRAINT

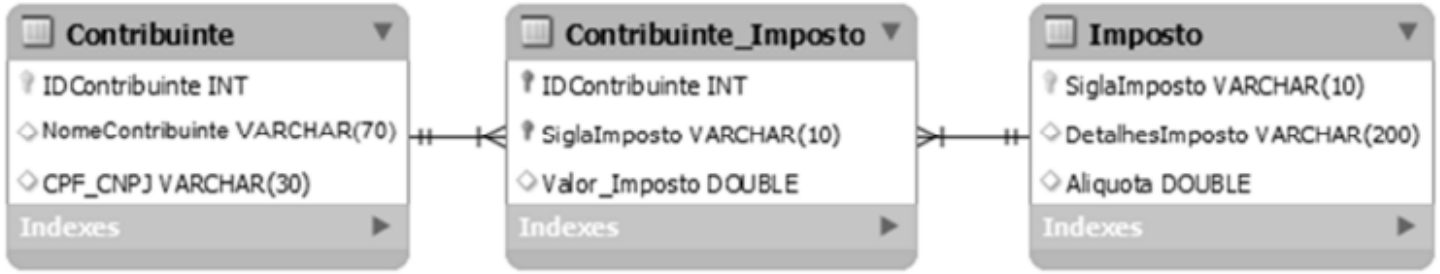
b) REFERENCES

c) REFERENCE CONSTRAINT

d) EXTENDS

e) IMPLEMENTS

(FCC - 2016 - SEGEP-MA - Técnico da Receita Estadual - Tecnologia da Informação - Conhecimentos Específicos) Atenção: Para responder às questões, considere a figura abaixo.



Considere que na tabela Contribuinte estão cadastrados os seguintes dados:

IDContribuinte	NomeContribuinte	CPF_CNPJ
1	Paulo da Silva	154.246.037-12
2	Maria Pereira	143.172.129-50

Após as tabelas Imposto e Contribuinte terem sido criadas, para criar a tabela Contribuinte_Imposto deve ser utilizada a seguinte instrução SQL:

```
CREATE TABLE Contribuinte_Imposto (IDContribuinte INT NOT NULL, SiglaImposto VARCHAR(10) NOT
NULL, Valor_Imposto DOUBLE, PRIMARY KEY (IDContribuinte, SiglaImposto), ..... I .....);
```

A lacuna I é corretamente preenchida por:

a) FOREIGN KEY (IDContribuinte) EXTENDS Contribuinte (IDContribuinte), FOREIGN KEY (SiglaImposto) EXTENDS Imposto (SiglaImposto)

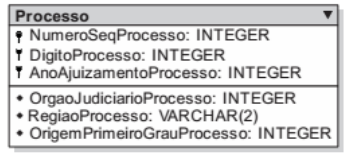
b) FOREIGN_KEY (IDContribuinte) CONSTRAINT Contribuinte (IDContribuinte), FOREIGN_KEY (SiglaImposto) CONSTRAINT Imposto (SiglaImposto)

c) FOREIGN KEY (IDContribuinte) REFERENCES Contribuinte (IDContribuinte), FOREIGN KEY(SiglaImposto) REFERENCES Imposto (SiglaImposto)

d) FOREIGN KEY (IDContribuinte, Contribuinte), FOREIGN KEY (SiglaImposto, Imposto)

e) REFERENCES Contribuinte (IDContribuinte) FOREIGN KEY, REFERENCES Imposto (SiglaImposto) FOREIGN KEY

(FCC - 2016 - TRF - 3ª REGIÃO - Técnico Judiciário - Informática) Para responder a questão, considere as informações abaixo.



Registros cadastrados:

NumeroSeqProcesso	DigitoProcesso	AnoAjuizamentoProcesso	OrgaoJudiciarioProcesso	RegiaoProcesso	OrigemPrimeiroGrauProcesso
15472	49	2002	4	3	3300
16592	44	2014	4	3	4500
17543	45	1999	4	3	3300
24535	23	2002	4	3	3300
29670	41	2012	4	2	2200
36535	45	2000	4	1	3400
44672	40	2012	4	2	3400
45891	43	2007	4	1	4400
67234	39	1997	4	1	3500

Considere que a tabela Processo foi criada sem chave primária. Nesse caso, para definir a chave primária, antes de serem inseridos registros, deve-se utilizar a instrução SQL

a) ADD TO Processo PRIMARY KEY(NumeroSeqProcesso, DigitoProcesso, AnoAjuizamentoProcesso);

b) INSERT INTO Processo PRIMARY KEY(NumeroSeqProcesso, DigitoProcesso, AnoAjuizamentoProcesso);

c) ALTER TABLE Processo ADD PRIMARY KEY(NumeroSeqProcesso, DigitoProcesso, AnoAjuizamentoProcesso);

d) ADD CONTRAINT PRIMARY KEY(NumeroSeqProcesso, DigitoProcesso, AnoAjuizamentoProcesso) from Processo;

e) UPDATE TABLE Processo ADD PRIMARY KEY(NumeroSeqProcesso, DigitoProcesso, AnoAjuizamentoProcesso);

(FGV - 2018 - MPE-AL - Analista do Ministério Público - Desenvolvimento de Sistemas) Analise o comando de criação da tabela teste, com cinco constraints, exibido a seguir.

```
create table teste (
a int null,
b varchar (40) not null,
c int,
constraint pk_a primary key (a),
constraint fk_a_c foreign key (c) references teste,
constraint uq_b unique (b),
constraint up_c unique (c),
constraint ch_1 check ( a >= 0.5 * c ) )
```

Assinale a constraint desse script que seria rejeitada no MS SQL Server.

a) A primeira. b) A segunda. c) A terceira.

d) A quarta. e) A quinta.

(FGV - 2015 - Câmara Municipal de Caruaru - PE - Analista Legislativo - Informática) Em geral, a definição de chaves estrangeiras em bancos de dados relacionais pode vir acompanhada da especificação de procedimentos adicionais a serem adotados quando da exclusão/alteração de valores nos registros da tabela estrangeira.

create table R2 (

a int not null primary key,

b int null,

x int not null,

constraint FK foreign key (x) references

R1(x))

Considerando o script acima, as opções complementares compatíveis para a definição da chave estrangeira FK são:

a) on delete cascade

on update set null

b) on delete cascade

on update cascade

c) on delete no action

on update set null

d) on delete set null

on update restrict

e) on delete restrict

on update set null

Trabalhando com tabelas (restrições)

Prof. Ramon Souza