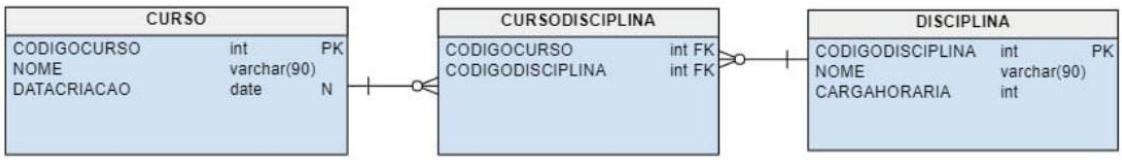
Criação de Banco de Dados: Empregar comandos para manipular linhas nas tabelas

MANIPULAÇÃO DE LINHAS NAS TABELAS

Quando usamos o termo manipulação, fazemos referência às operações de inserção, atualização ou mesmo eliminação de dados. Em uma linguagem mais comercial, existe o termo **CRUD**, que representa quatro operações básicas: criação, consulta, atualização e remoção de dados, respectivamente.

Create: INSERT → **R**oad: SELECT → **U**pdate:UPDATE → **D**elete:DELETE

Modelo para exemplos:




```
--Comando para criar tabela CURSO;
CREATE TABLE CURSO (
        CODIGOCURSO int NOT NULL,
        NOME varchar(90) NOT NULL,
        DATACRIACAO date NULL,
        CONSTRAINT CURSO_pk PRIMARY KEY (CODIGOCURSO)
);
--Comando para criar tabela DISCIPLINA;
CREATE TABLE DISCIPLINA (
        CODIGODISCIPLINA int NOT NULL,
        NOME varchar(90) NOT NULL,
        CARGAHORARIA int NOT NULL,
        CONSTRAINT DISCIPLINA_pk PRIMARY KEY (CODIGODISCIPLINA)
);
--Comando para criar CURSODISCIPLINA;
CREATE TABLE CURSODISCIPLINA (
        CODIGOCURSO int NOT NULL,
        CODIGODISCIPLINA int NOT NULL,
```

```
        CONSTRAINT CURSODISCIPLINA_pk PRIMARY KEY
(CODIGOCURSO,CODIGODISCIPLINA),

        CONSTRAINT CURSODISCIPLINA_CURSO FOREIGN KEY
(CODIGOCURSO) REFERENCES CURSO(CODIGOCURSO) ON DELETE
CASCADE,

        CONSTRAINT CURSODISPLINA_DISCIPLINA FOREIGN KEY
(CODIGODISCIPLINA) REFERENCES DISCIPLINA (CODIGODISCIPLINA)

);


COMMIT
```

O modelo é útil para gerenciar os dados de cursos, disciplinas e do relacionamento entre esses objetos. Em especial, cada linha da tabela CURSODISCIPLINA representa uma associação entre curso e disciplina.

INSERÇÃO DE LINHAS EM TABELA(CREATE)

A inserção de linhas em tabela é realizada com o auxílio do comando INSERT da SQL. Sua sintaxe **básica** está expressa a seguir:

INSERT INTO <NOMETABELA> (COLUNA1, COLUNA2,...,COLUNAn) VALUES (VALOR1, VALOR2,...,VALORn);

Caso vc vá inserir informações em todas as linhas da tabela, não necessita indicar o nome das colunas, assim o a estrutura do código passa a ser:

INSERT INTO <NOMETABELA> VALUES (VALOR1, VALOR2,...,VALORn);

Iremos cadastrar quatro cursos. O comando SQL a seguir pode ser utilizado:

```
INSERT INTO CURSO (CODIGOCURSO,NOME,DATACRIACAO)
VALUES( 1,'Sistemas de Informação','19/06/1999');
INSERT INTO CURSO (CODIGOCURSO,NOME,DATACRIACAO)
VALUES( 2,'Medicina','10/05/1990');
INSERT INTO CURSO (CODIGOCURSO,NOME,DATACRIACAO)
VALUES( 3,'Nutrição','19/02/2012');
INSERT INTO CURSO (CODIGOCURSO,NOME,DATACRIACAO)
VALUES( 4,'Pedagogia','19/06/1999');
```

Agora, faremos um procedimento semelhante, cadastrando quatro disciplinas. O comando SQL a seguir pode ser utilizado:

```
INSERT INTO DISCIPLINA (CODIGODISCIPLINA,NOME,CARGAHORARIA)
VALUES( 1,'Leitura e Produção de Textos',60);
INSERT INTO DISCIPLINA (CODIGODISCIPLINA,NOME,CARGAHORARIA)
VALUES( 2,'Redes de Computadores',60);
INSERT INTO DISCIPLINA (CODIGODISCIPLINA,NOME,CARGAHORARIA)
```

```
VALUES( 3,'Computação Gráfica',40);
INSERT INTO DISCIPLINA (CODIGODISCIPLINA,NOME,CARGAHORARIA)
VALUES( 4,'Anatomia',60);
```

Agora, vamos registrar na tabela CURSODISCIPLINA algumas associações entre cursos e disciplinas. O comando SQL a seguir pode ser utilizado:

```
INSERT INTO CURSODISCIPLINA(CODIGOCURSO,CODIGODISCIPLINA)
VALUES (1,1);
INSERT INTO CURSODISCIPLINA(CODIGOCURSO,CODIGODISCIPLINA)
VALUES (1,2);
INSERT INTO CURSODISCIPLINA(CODIGOCURSO,CODIGODISCIPLINA)
VALUES (1,3);
INSERT INTO CURSODISCIPLINA(CODIGOCURSO,CODIGODISCIPLINA)
VALUES (2,1);
INSERT INTO CURSODISCIPLINA(CODIGOCURSO,CODIGODISCIPLINA)
VALUES (2,3);
INSERT INTO CURSODISCIPLINA(CODIGOCURSO,CODIGODISCIPLINA)
VALUES (3,1);
INSERT INTO CURSODISCIPLINA(CODIGOCURSO,CODIGODISCIPLINA)
VALUES (3,3);
```

E se submetermos ao SGBD o comando a seguir?

```
INSERT INTO CURSODISCIPLINA(CODIGOCURSO,CODIGODISCIPLINA)
VALUES (3,30);
```

O SGBD não realizará a inserção e retornará uma mensagem de erro informando que 30 não é um valor previamente existente na chave primária da tabela DISCIPLINA. Isso acontece porque, quando definimos (linha 16 do *script* da seção anterior) a chave estrangeira da tabela CURSODISCIPLINA, nós delegamos ao SGBD a tarefa de realizar esse tipo de validação com objetivo de sempre manter a integridade dos dados do banco de dados. Note que não existe disciplina identificada de código 30 na tabela DISCIPLINA.

MECANISMO DE CHAVE PRIMÁRIA EM AÇÃO

Já vimos que o SGBD é responsável por manter a integridade dos dados ao longo de todo o ciclo de vida do banco de dados. A consequência disso pode ser percebida ao tentarmos executar (novamente) o comando a seguir:

```
INSERT INTO DISCIPLINA (CODIGODISCIPLINA, NOME,
CARGAHORARIA) VALUES (4,'Anatomia',60);
```

Como já existe um registro com valor de CODIGODISCIPLINA igual a 4, o SGBD exibirá mensagem de erro informando que o referido valor já existe no banco de dados. Semelhantemente, devemos lembrar que todo valor de chave primária é obrigatório.

Vamos agora tentar inserir uma disciplina sem valor para CODIGODISCIPLINA, conforme comando SQL a seguir:

```
INSERT INTO DISCIPLINA (CODIGODISCIPLINA, NOME,
CARGAHORARIA) VALUES (NULL,'Biologia Celular e Molecular',60);
```

O SGBD exibirá mensagem de erro informando que o valor da coluna CODIGODISCIPLINA não pode ser nulo.

ATUALIZAÇÃO DE LINHAS EM TABELA(UPDATE):

A atualização de linhas em tabela é realizada com o auxílio do comando UPDATE da SQL. Sua sintaxe **básica** está expressa a seguir:

```
UPDATE <NOMETABELA>
SET COLUNA1=VALOR1, COLUNA2=VALOR2,…,COLUNAn=VALORn
WHERE <CONDIÇÃO>;
```

Breve exemplo:

```
UPDATE EMPREGADO
SET SALARIO = SALARIO + 1000;
```

Sem a condição WHERE o camando a seguir irá executar para todos os registros da tabela empregado, o o camando a seguir com a o WHERE determinar que para todos o sempregados com id 10 ou 20 o salario é de 3000;

```
UPDATE EMPREGADO
SET SALARIO = 3000
WHERE ID = 10 OR = 20;
```

Alteraremos para 70 a carga horária da disciplina Redes de Computadores. Para isso, basta executar o comando a seguir:

```
UPDATE DISCIPLINA SET CARGAHORARIA=70 WHERE CODIGODISCIPLINA=2;
```

Suponha agora que houve a necessidade de alterar em 20% a carga horária de todas as disciplinas cadastradas no banco de dados. Podemos executar o seguinte comando:

```
UPDATE DISCIPLINA SET CARGAHORARIA=CARGAHORARIA*1.2;
```

ATUALIZAÇÃO DE COLUNA CHAVE PRIMÁRIA

Devemos ter especial cuidado ao planejarmos alterar o valor de coluna com o papel de chave primária em uma tabela.

Vamos supor que seja necessário alterar para 6 o valor de CODIGOCURSO referente ao curso de Pedagogia. Podemos, então, executar o comando a seguir:

```
UPDATE CURSO SET CODIGOCURSO=6 WHERE CODIGOCURSO=4;
```

Perceba que o SGBD processará a alteração, visto que não há vínculo na tabela CURSODISCIPLINA envolvendo este curso. No entanto, o que aconteceria se tentássemos alterar para 10 o valor de CODIGOCURSO referente ao curso de Sistemas de Informação?

Seguindo a mesma linha de raciocínio:

```
UPDATE CURSO SET CODIGOCURSO=10 WHERE CODIGOCURSO=1;
```

O SGBD não realizará a alteração e retornará uma mensagem de erro indicando que o valor 1 está registrado na tabela CURSODISCIPLINA, coluna CODIGOCURSO. Isso significa que, se o SGBD aceitasse a alteração, a tabela CURSODISCIPLINA ficaria com dados inconsistentes, o que não deve ser permitido.

Assim, de modo semelhante ao que aprendemos na seção Mecanismo de chave primária em ação, deixaremos o SGBD realizar as alterações necessárias para manter a integridade dos dados. Vamos, então, submeter o comando a seguir:

```
ALTER TABLE CURSODISCIPLINA
DROP CONSTRAINTCURSODISCIPLINA_CURSO,
ADD CONSTRAINT CURSODISCIPLINA_CURSO
FOREIGN KEY (CODIGOCURSO) REFERENCES CURSO (CODIGOCURSO)
ON UPDATE CASCADE ;
```

O que fizemos? Usamos o comando ALTER TABLE para alterar a estrutura da tabela CURSODISCIPLINA:, removemos a chave estrangeira (comando DROP CONSTRAINT) e, por último, recriamos a chave (ADD CONSTRAINT), especificando a operação de atualização (UPDATE) em cascata.

Assim, após o processamento da alteração anterior, podemos então submeter o comando, conforme a seguir:

```
UPDATE CURSO SET CODIGOCURSO=10 WHERE CODIGOCURSO=1;
```

REMOÇÃO DE LINHAS EM TABELA(DELETE)

A remoção de linhas em tabela é realizada com o auxílio do comando DELETE da SQL. Sua sintaxe **básica** está expressa a seguir:

```
DELETE FROM <NOMETABELA>
WHERE <CONDIÇÃO>;
```

Atenção!

O DELETE sem o WHERE executa para todas as linhas da tabela, ou seja, ele apaga todos os registros da tabela;

Caso ocorra isso, sem querer não faça commit, e realize um rollback para cancelar o comando;

Exemplo:

```
DELETE FROM <NOME DA TABELA>;
```

Queremos apagar do banco de dados a disciplina de Anatomia. Podemos, então, usar o código a seguir:

```
DELETE FROM DISCIPLINA WHERE CODIGODISCIPLINA=4;
```

O SGBD localiza na tabela DISCIPLINA a linha cujo conteúdo da coluna CODIGODISCIPLINA seja igual a 1. Em seguida, remove do banco de dados a linha em questão.

Suponhamos que queremos remover a disciplina de leitura e produção de textos:

```
DELETE FROM DISCIPLINA WHERE CODIGODISCIPLINA=1;
```

Isso daria erro, pq existem dados associado a essa disciplina, pois se caso isso acontecesse o banco de dados ficaria com dados inconsistentes.

Assim, de modo semelhante ao que aprendemos na seção Mecanismo de chave primária em ação, deixaremos o SGBD realizar as alterações necessárias para manter a integridade dos dados. Vamos, então, submeter o comando a seguir:

```
ALTER TABLE CURSODISCIPLINA
DROP CONSTRAINT CURSODISCIPLINA_DISCIPLINA,
ADD CONSTRAINT CURSODISCIPLINA_DISCIPLINA
FOREIGN KEY (CODIGODISCIPLINA) REFERENCES DISCIPLINA
(CODIGODISCIPLINA)
ON DELETE CASCADE ;
```

O que fizemos? Usamos o comando ALTER TABLE para alterar a estrutura da tabela CURSODISCIPLINA:, removemos a chave estrangeira (comando DROP CONSTRAINT) e, por último, recriamos a chave (ADD CONSTRAINT), especificando operação de remoção (DELETE) em cascata.

Assim, após o processamento da alteração anterior, podemos submeter o comando conforme a seguir:

```
DELETE FROM DISCIPLINA WHERE CODIGODISCIPLINA=1;
```

E se quiséssemos eliminar todos os registros de todas as tabelas do banco de dados?

Para realizarmos esta operação, precisaremos identificar quais tabelas são mais independentes e quais são as que possuem vínculos de chave estrangeira.

No nosso caso, CUSODISCIPLINA possui 2 chaves estrangeiras o que a torna mais dependente, já as demais não possuem chaves estrangeiras, assim, sendo menos dependentes:

```
DELETE FROM CURSODISCIPLINA;
DELETE FROM CURSO;
DELETE FROM DISCIPLINA;
```