

## Cláusulas para Criação de Tabelas

Quando estamos criando novas tabelas, algumas cláusulas podem ser utilizadas para dar propriedades especiais a um determinado campo. Nesta aula aprenderemos algumas delas.

Para começar, vamos relembrar o conceito de **Chave Primária**. Cada **registro** (linha da tabela) deve possuir uma forma única de nos referirmos a ele para facilitar na hora de realizarmos consultas, alteração ou exclusão.

Imagine uma tabela chamada clientes que possui os seguintes campos e valores:

NOME	ENDEREÇO	IDADE
João da Silva	Rua Projetada S/N	30
Maria da Conceição Souza	Rua Dom Pedro I, 1955	27

Quando temos valores distintos fica fácil localizar o que queremos, mas imagine que um belo dia, aparece outro João da Silva que mora numa outra Rua Projetada S/N e que também possui 30 anos de idade. Aí é onde os problemas começam, pois fica difícil saber quem é quem.

NOME	ENDEREÇO	IDADE
João da Silva	Rua Projetada S/N	30
Maria da Conceição Souza	Rua Dom Pedro I, 1955	27
João da Silva	Rua Projetada S/N	30

Inserir uma **chave primária** na tabela vai fazer a diferença, pois um campo chave possui um **valor que nunca se repete** em todos os registros. Neste caso uma boa escolha seria utilizar o CPF do cliente, deixando a tabela assim:

CPF (chave)	NOME	ENDEREÇO	IDADE
111.111.111-11	João da Silva	Rua Projetada S/N	30
123.123.123-12	Maria da Conceição Souza	Rua Dom Pedro I, 1955	27
444.555.666-77	João da Silva	Rua Projetada S/N	30

Deve-se dar preferência a dados como CPF, MATRÍCULA ou mesmo definir um campo de código para controlar os registros.

Para criar uma chave primária, basta adicionar a cláusula **PRIMARY KEY** após o tipo do campo como no código abaixo:

```
CREATE TABLE cliente (cpf char(14) PRIMARY KEY,
                        nome varchar(40),
                        endereco varchar(60),
                        idade int);
```

Dica: No MySQL use o comando `DESC nome_da_tabela` para ver a estrutura da tabela.

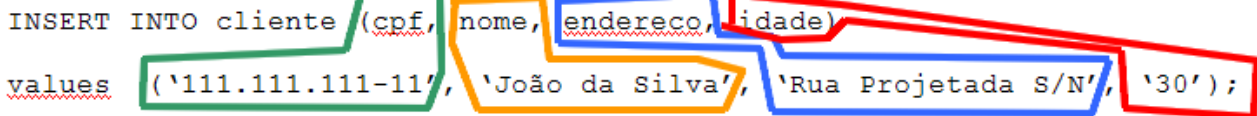
Agora vamos inserir alguns dados para testar nossa tabela. Use o seguinte comando:

```
INSERT INTO nome_da_tabela (campos) VALUES (valores)
```

Para a tabela que acabamos de criar ficaria assim:

```
INSERT INTO cliente (cpf, nome, endereco, idade)
values ('111.111.111-11', 'João da Silva', 'Rua Projetada S/N', '30');
```

**OBS1:** Observe os valores devem ser colocados na MESMA ORDEM em que os nomes dos campos são informados e devem sempre estar ENTRE ASPAS SIMPLES:



```
INSERT INTO cliente (cpf, nome, endereco, idade)
values ('111.111.111-11', 'João da Silva', 'Rua Projetada S/N', '30');
```

Para consultar o que já foi inserido na tabela, use o comando:

```
SELECT * FROM nome_da_tabela;
```

Neste exemplo, `SELECT * FROM cliente;`

Algumas vezes será necessário usar mais de um campo para definir uma chave primária (o que chamamos de chave composta), fazemos isso colocando a cláusula no final da declaração e os campos chave entre parênteses, assim:

```
CREATE TABLE cliente (cpf char(14), nome varchar(40),
                        endereco varchar(60), idade int,
                        PRIMARY KEY (cpf, nome));
```

Se você optar por criar um campo de código para ser a chave primária, há ainda a possibilidade de fazer com que o banco de dados conte e preencha automaticamente usando a cláusula `AUTO_INCREMENT`. Veja o exemplo:

```
CREATE TABLE cliente (codigo int AUTO_INCREMENT primary key,
                        nome varchar(40),
                        endereço varchar(60)
                        idade int);
```

Para testar, faça um `INSERT` omitindo o campo código, como em:

```
INSERT INTO CLIENTE (nome) values ('João');
```

E dê um `SELECT` para conferir como foi inserido.

**OBS2.:** A cláusula `AUTO_INCREMENT` só pode ser usada para **números inteiros**.

**OBS3.:** O Firebird não reconhece a cláusula `AUTO_INCREMENT`.

Outro modificador importante é o **NOT NULL**, que vai exigir o preenchimento do campo para que o registro possa ser salvo. Por exemplo, não podemos cadastrar um cliente que não forneça o nome, assim, podemos fazer com que o banco de dados (não a aplicação em si) “obrigue” o usuário a digitar um nome para poder gravar. Isso é feito adicionando a cláusula da seguinte forma:

```
CREATE TABLE cliente (codigo int auto_increment primary key,
                        nome varchar(40) NOT NULL,
                        endereço varchar(60)
                        idade int);
```

Podemos também definir um **valor padrão** para um determinado campo. Por exemplo, digamos que a empresa armazena o saldo devedor do cliente num dos campos da tabela, entretanto, cada novo cliente cadastrado deve aparecer com saldo devedor igual a zero, para isso, podemos usar a cláusula **DEFAULT** da seguinte maneira:

```
CREATE TABLE cliente (codigo int auto_increment primary key,  
                        nome varchar(40) not null,  
                        endereço varchar(60), idade int,  
                        saldo float DEFAULT '0.00');
```

Se nenhum saldo for informado ao cadastrar o cliente, o valor entre aspas (neste caso, 0.00) será gravado automaticamente no campo.

### Exercícios

1. Crie uma tabela para armazenar os dados dos alunos de uma escola. Defina o campo chave como matrícula.
2. Crie uma tabela semelhante à anterior usando uma chave composta com os campos matrícula e cpf.
3. Modifique o esquema anterior para exigir a digitação de uma data de nascimento para o aluno.
4. Modifique o esquema anterior para preencher automaticamente o campo IRA (Índice de Rendimento Acadêmico) como 0 (zero) para todos os novos cadastros.

### Referências

Manzano, J. A. (2002). *Estudo dirigido de SQL (ANSI/89)*. São Paulo: Érica.

Firebird Project Members. (2008). *Firebird Quick Start Guide*. IBPhoenix Editors.

Disponível em: < <http://www.firebirdsql.org/pdfmanual/Firebird-2-QuickStart.pdf> >