Aula 04: Modelo Relacional e Comandos DDL

Banco de Dados

Roteiro da Aula

- Modelo Relacional
- Criação de Tabelas, Campos e Atributos

Modelo Relacional

- O modelo relacional é um modelo de dados, adequado a ser o modelo de um Sistema Gerenciador de Banco de Dados (SGBD), que se baseia no princípio em que todos os dados estão guardados em **tabelas (*representação bi-dimensional de dados composta de linhas e colunas*).**
- Tornou-se um padrão de fato para aplicações comerciais, devido a sua simplicidade e performance.

Modelo Relacional

- Toda a Informação de um banco de dados relacional é armazenada em **Tabelas**, que na linguagem do MER, também são chamadas de **Entidades**. Por exemplo, posso ter uma Tabela “Alunos", onde seriam armazenadas informações sobre os diversos alunos.
- Sobre cada um dos alunos podem ser armazenadas diversas informações tais como: Nome, RG, Matricula, Rua, Bairro, Telefone, CEP, Sexo, Estado Civil, etc.
- Essas diversas características de cada Aluno são os "**Atributos**" da entidade Aluno , também chamados de campos da tabela Aluno .

Modelo Relacional - Exemplo

- Tabela Aluno:

Num_Matricula	Nome_Aluno	Sexo_Aluno
1	Maria	F
2	João	M
3	Pedro	M
4	Carla	F
5	Sandra	F

- Regras:
  - Nomes de tabelas devem ser únicos no banco de dados;
  - De preferência a nomes curtos.

Modelo Relacional - Atributos ou Colunas

- Considerando a tabela Aluno:
  - Ela tem três colunas Num_Matrícula, Nome_Aluno e Sexo_Aluno;
  - A cada uma destas colunas damos o nome de atributo;
  - Um nome de atributo deve ser **único** em uma tabela e dizer exatamente o tipo de informação que ele representa.

Modelo Relacional - Atributos ou Colunas

- Regras:
  - Uma coluna (atributo) não segue um ordenamento especifico;
  - Nome de uma coluna deve expressar exatamente o que armazena;
  - Sempre que possível utilizar prefixos padronizados, Cod_Dept, Nome_Funcionario, Qtde_Estoque
  - **NUNCA UTILIZAR ACENTOS GRÁFICOS E CARACTERES DA LÍNGUA PORTUGUESA, COMO EXEMPLO "Ç" PARA NOMEAR ATRIBUTOS E TABELAS**

Modelo Relacional - Linhas, Registros ou Tuplas

- A tabela Aluno possui cinco registros;
- Cada registro representa um conjunto de valores;
- A este relacionamento damos o nome de registro, linha ou ainda Tupla;
- Cada linha da tabela é única e possui um atributo identificador (Num_Matrícula);
- Este atributo identificador é chamado de **chave primária**.
- **Regras:**
  - Em uma tabela não devem existir linhas duplicadas;
  - As linhas de uma tabela não seguem uma ordem especifica.

Modelo Relacional - Domínio

- A tabela Aluno possui três atributos;
- Para cada atributo existe um conjunto de valores permitidos chamado domínio daquele atributo:
  - Para o atributo Num_Matrícula o domínio é o conjunto de números naturais;
  - Para o atributo Nome_Aluno o domínio é qualquer nome válido;
  - Enquanto que para Sexo_Aluno o domínio são os mnemônicos **M** ou **F**.

Modelo Relacional – Tabelas e Entidades

- Para a criação de banco de dados, tabelas e atributos em um SGBD, utilizaremos a linguagem SQL que é compostas de comandos de manipulação, **definição** e controle de dados.
- Esses conjuntos de comandos de definição de dados são denominados pela sigla **DDL**(Data Definition Language), que disponibiliza um conjunto de comandos para criação(CREATE), alteração(ALTER) e remoção (DROP) de tabelas e outras estruturas.

Comando CREATE DATABASE

- A maioria dos SGBDs disponibiliza ferramentas que permitem a criação de Bancos de Dados, mas é possível criar o próprio Banco de Dados a partir de um comando SQL.
- A sintaxe do comando é:

```
CREATE DATABASE nome_do_banco_de_dados
```

```
CREATE DATABASE IFBA
```

Comando DROP DATABASE

- O comando DROP DATABASE permite remover um determinado Banco de Dados, apagando todas as tabelas e estruturas associadas e, conseqüentemente, todos os dados existentes nelas
- A sintaxe do comando é:

```
DROP DATABASE nome_do_banco_de_dados
```

```
DROP DATABASE IFBA
```

Comando CREATE TABLE

- O comando CREATE TABLE é o principal comando DDL da linguagem SQL. A criação de tabelas é realizada em SQL utilizando este comando. Sua sintaxe básica é a seguinte:

```
CREATE TABLE nome_da_tabela(Coluna1 Tipo,Coluna2 Tipo,
ColunaN Tipo)
```

```
CREATE TABLE Empregado(Id INTEGER, Nome CHAR(50),
Data_Nasc DATE, Salario FLOAT)
```

Comando CREATE TABLE

```
CREATE TABLE Empregado(Id INTEGER, Nome CHAR(50),
Data_Nasc DATE, Salario FLOAT)
```

Id	Nome	Data_Nasc	Salario

Tipos de Dados

- Em SQL os tipos de dados são agrupados em 3 categorias:
  - Caracteres (Strings)
  - Numéricos
  - Tempo e Data
- No slide a seguir é apresentado uma tabela com os tipos de dados da linguagem SQL

Tipos de Dados

- Tabela com tipos de Dados SQL

TIPO DE DADOS	DESCRIÇÃO
CHAR (n)	Tipo de dados STRING com comprimento fixo (n > 0)
CHAR	Um único caractere
VARCHAR (n)	Tipo de dados STRING com comprimento variável (n > 0)
BIT	Valores True/False ou 0/1
NUMERIC ou NUMERIC (n) ou NUMEROC (n,d)	Valor numérico constituído por n dígitos e sinal e com d casas decimais
DATETIME	Um valor de data ou hora entre os anos 100 e 9999.
INTEGER ou INT	Inteiro
SMALLINT	Pequeno Inteiro
FLOAT	Nº de Dupla Precisão
DATE	Datar
TIME	Hora

Tipos de Dados

- Apesar dos tipos de dados citados anteriormente serem padrões em banco de dados que utilizam SQL, a maioria dos fabricantes disponibiliza outros tipos de dados, como exemplo o MONEY(valores monetários), COUNTER (números inteiros incrementados automaticamente pelo banco de dados.

Exemplo 01

- Escrever um comando de SQL que permita criar uma tabela com o nome Caixa_Postal, capaz de armazenar um inteiro de até quatro dígitos e uma string com 45 caracteres

```
CREATE TABLE Caixa_Postal(Codigo NUMERIC, Localidade CHAR(45))
```

Codigo	Localidade

Exemplo 02

- Escrever um comando de SQL que permita criar uma tabela com o nome Pessoa, com o seguintes atributos: ID, Nome, Idade, Salario, Telefone e Código Postal)

```
CREATE TABLE Pessoa(Codigo INTEGER, nome CHAR(45), idade
INTEGER, salario NUMERIC(10,2), telefone CHAR(12),
Codigo_Postal CHAR(9))
```

Codigo	Nome	Idade	Salario	Telefone	Codigo_Postal

Exercício Prático

- Escrever um comando para criar uma base de dados com o nome de ESCOLA.
- Criar duas tabelas:
  - A primeira tabela deverá ter o nome DISCIPLINA com os atributos: id, nome, carga horária, período e ementa,
  - A segunda tabela deverá ter o nome LIVROS, com os atributos: id, isbn, nome, autor, editora, edicao, disciplina, data de aquisição
- Ao final exclua as duas tabelas e o banco de dados criado.

Características das Colunas

- Para execução do comando CREATE TABLE é necessário indicar qual o nome da tabela e, para cada uma das colunas, o nome da coluna e o tipo de dados.
- No entanto, podem ser indicadas as características próprias de cada uma das colunas, tais como:Que valores pode admitir ?Qual o valor padrão?O campo representa um atributo identificador(chave primária)?

Atributos NOT NULL

- No exemplo apresentado anteriormente:

```
CREATE TABLE Caixa_Postal(Codigo NUMERIC, Localidade CHAR(45))
```

- Estamos admitindo que a tabela é composta por duas colunas(código e localidade) e que qualquer uma delas pode admitir valores nulos(ou seja, o usuário poderá informar dados vazios para os campos).
- Se quisermos que uma coluna não admita valores nulos, usamos a cláusula **NOT NULL**.

Valores por padrão(*default*)

- Caso um valor não seja inserido em uma coluna o valor padrão (*default*) armazenado nela é **NULL**. No entanto, é possível associar um outro valor default através da cláusula **DEFAULT**.
- Se quisermos por exemplo que a localidade padrão (*default*) se chame **Ilhéus**, então teremos que fazer o seguinte:

Valores por padrão(*default*)

- Veja o exemplo:

```
CREATE TABLE Caixa_Postal(Codigo NUMERIC NOT NULL,
Localidade CHAR(45))
```

- O código acima está **enfatizando** que o atributo **Codigo** da tabela **Caixa Postal, NÃO ACEITA** valores nulos, ou seja, o campo Código deverá possuir **SEMPRE**, qualquer valor diferente de vazio.

```
CREATE TABLE Caixa_Postal(Codigo NUMERIC NOT NULL,
Localidade CHAR(45) DEFAULT 'Ilhéus')
```

Restrições (*constraints*)

- Restrições são regras a que os valores de uma ou mais colunas devem obedecer. Por exemplo, o conteúdo da coluna Sexo só poderá conter os valores "F" ou "M", a coluna Idade não poderá conter valores negativos, o salário não poderá ser inferior ao salário mínimo(R$ 510,00).
- A utilização de restrições é a única garantia que temos de que os dados existentes nas colunas estão de acordo com as regras especificadas no projeto do Banco de Dados.

Restrições (*constraints*)

- Existem alguns tipos distintos de restrições que se podem aplicar a colunas:
  - *Constraint* **NOT NULL**
  - *Constraint* **CHECK**
  - *Constraint* **UNIQUE**
  - *Constraint* **PRIMARY KEY**
  - *Constraint* **REFERENCES**

Constraint CHECK

- A *constraint* **CHECK** permite realizar a validação dos dados inseridos na coluna, através da especificação de uma condição. São admitidos apenas os dados cujo resultado da avaliação da condição seja verdadeiro.
- No slide a seguir veremos alguns exemplos utilizando a *constraint* CHECK

Constraint CHECK

- Exemplos com ***constraints***

```
CREATE TABLE Dados_Pessoais
(
Codigo NUMERIC NOT NULL,
Nome CHAR(60) CHECK(Nome NOT LIKE '%Regilan%'),
Idade INTEGER NOT NULL CHECK(Idade >= 0 AND Idade <=
150),
Sexo CHAR CHECK (SEXO IN('M', 'F')),
Tempo_Servico INTEGER CHECK(Tempo_Servico >= 0)
)
```

Constraint UNIQUE

- A *constraint* **UNIQUE** indica que os valores dessa coluna não podem se repetir.
- Em uma tabela podem existir tantas colunas UNIQUE quantas forem necessárias.
- Veja o exemplo:

```
CREATE TABLE Dados_Pessoais
(
Codigo NUMERIC NOT NULL,
Nome CHAR(60) UNIQUE,
CPF CHAR(15) UNIQUE,
Tempo_Servico INTEGER CHECK(Tempo_Servico >= 0)
)
```

Comando DROP TABLE

- O comando DROP TABLE permite remover uma determinada tabela de um Banco de Dados, e conseqüentemente, todos os dados existentes nela.
- A sintaxe do comando é:

```
DROP TABLE nome_da_tabela
```

```
DROP TABLE Dados_Pessoais
```

Exemplo 03

- Criar um banco de dados **Clínica** com as seguintes características:
- **Tabela Medicos**
  - Atributo CRM: caractere, único e não vazio
  - Atributo Nome: caractere e não vazio
  - Atributo Idade: inteiro e não poderá ser maior que 23 e menor que 70
  - Atributo Especialidade: caractere e não poderá possuir especialização em Ortopedia
- **Tabela Paciente**
  - Atributo CPF: caractere e único,
  - Atributo Nome: caractere e não vazio
  - Atributo Doença: caractere e não poderá ter valores como fratura e torção

Exemplo 03 - Resolução

```
CREATE DATABASE CLINICA
```

```
CREATE TABLE Medicos
(
Crm CHAR(15) NOT NULL UNIQUE,
Nome CHAR(100) NOT NULL,
Idade INTEGER CHECK(Idade > 23 AND Idade < 70),
Especialidade CHAR(50) CHECK(Especialidade NOT LIKE
'%ORTOPEDIA%')
)
```

```
CREATE TABLE Paciente(
Cpf CHAR(15) UNIQUE,
Nome CHAR(100) NOT NULL,
Doenca CHAR(50) CHECK(Doenca NOT LIKE '%FRATURA%' AND
'%TORÇÃO')
)
```

Próxima Aula

- Relacionamento entre tabelas
- Primary Key e Foreign Key
- Comando ALTER TABLE