Banco de Dados Avançado

Prof. Me. Nisston Moraes Tavares de Melo

prof2279@iesp.edu.br

Revisão

- Apresentação da interface do SQL Server;
- Criando o Banco de Dados;
- Criando as Tabelas;
- Povoando as Tabelas;
- Operações com INSERT, UPDATE e DELETE.
- Operações básicas do SELECT.
  - BETWEEN 2600 and 5600
  - IN
  - LIKE
  - IS NULL
  - AS (apelido das colunas)
  - ORDER BY (ASC e DESC)
  - DISTINCT
- Trabalhando com o JOIN
- Funções de agregação
  - Avg()
  - Count()
  - Max()
  - Min()
  - SUM()
- Trabalhando com SubQuery

Construção de Constraints

- Uma constraint, em um contexto de banco de dados, é uma regra ou restrição aplicada a uma ou mais colunas em uma tabela, com o objetivo de garantir a integridade, consistência e validade dos dados armazenados.
- Essas regras podem ser impostas para garantir que os dados atendam a certos critérios específicos, como restrições de chave primária, chave estrangeira, unicidade, não nulos e condições de verificação.
- As constraints são essenciais para manter a qualidade e a confiabilidade dos dados em um banco de dados, impedindo a inserção, atualização ou exclusão de registros que violam as regras definidas.

Importância das Constraints

- Destaque para a importância das constraints na manutenção da integridade dos dados.
- Benefícios incluem a prevenção de inserção de dados inválidos, restrição de operações de atualização/deleção que violam regras de integridade e garantia de consistência dos dados.

Características das Constraints

- Definição de regras de integridade para assegurar a validade dos dados.
- Aplicação automática e imediata em todas as operações de modificação de dados.
- Capacidade de serem definidas a nível de coluna ou de tabela.
- Diversos tipos de constraints, como UNIQUE, PRIMARY KEY, FOREIGN KEY e CHECK.

Tipos de constraint

- **PRIMARY KEY**: Define uma ou mais colunas como chaves primárias, garantindo que cada linha na tabela tenha um valor único para essa coluna (ou conjunto de colunas). Uma chave primária também impede a inserção de valores nulos.
- **FOREIGN KEY**: Define uma relação de integridade referencial entre duas tabelas, garantindo que os valores em uma coluna correspondam aos valores em outra coluna em uma tabela relacionada.
- **UNIQUE**: Garante que os valores em uma coluna (ou conjunto de colunas) sejam exclusivos em cada linha da tabela, permitindo apenas um valor nulo, caso a coluna permita valores nulos.
- **CHECK**: Define uma condição que os valores em uma coluna devem satisfazer para serem inseridos ou atualizados na tabela. Isso pode ser usado para aplicar regras de validação personalizadas aos dados.
- **NOT NULL**: Garante que os valores em uma coluna não sejam nulos, ou seja, é obrigatório fornecer um valor para essa coluna ao inserir ou atualizar registros na tabela.

Exemplo 03

- Neste exemplo, a constraint UNIQUE é aplicada à coluna "Email" da tabela "Clientes", garantindo que nenhum endereço de e-mail seja duplicado na tabela.

```
CREATE TABLE Clientes (
    ID INT PRIMARY KEY,
    Nome VARCHAR(50),
    Email VARCHAR(100) UNIQUE
);
```

Exemplo 06

- Neste exemplo, a constraint NOT NULL é aplicada à coluna "Nome" da tabela "Funcionarios", garantindo que nenhum valor nulo seja permitido nesta coluna.

```
CREATE TABLE Funcionarios (
    ID INT PRIMARY KEY,
    Nome VARCHAR(50) NOT NULL,
    Cargo VARCHAR(50)
);
```

Exemplo 02

- Exemplo de criação de uma constraint CHECK em SQL Server para restringir o intervalo de valores permitidos em uma coluna:

```
ALTER TABLE Produtos
  ADD CONSTRAINT CHK_PrecoValido CHECK (Preco >= 0);
```

- Explicação do código: Esta constraint CHECK garante que o valor da coluna "Preco" na tabela "Produtos" seja maior ou igual a zero.

Exemplo 01

- Exemplo de criação de uma constraint CHECK para restringir o intervalo de valores permitidos na coluna UF:
  - CREATE TABLE funcao(
    codigo int,nome varchar(40) NOT NULL,
    uf varchar(2) CONSTRAINT CK_CARGO_NIVEL CHECK (uf IN
    (‘PB’,’PE’,’CE’,’PI’,’BA’)),
    idade int NOT NULL,
    salario real,
    id_funcionario int,
    FOREIGN KEY(id_funcionario) REFERENCES Disciplina(IDDisciplina))

Exemplo 04

- Neste exemplo, a constraint PRIMARY KEY é aplicada à coluna "ID" da tabela "Alunos", garantindo que cada aluno tenha um identificador único na tabela..

```
CREATE TABLE Pedidos (
    ID INT PRIMARY KEY,
    ClienteID INT,
    DataPedido DATE,
    FOREIGN KEY (ClienteID) REFERENCES Clientes(ID)
);
```

Exemplo 05

- Neste exemplo, a constraint CHECK é aplicada às colunas "Quantidade" e "Preco" da tabela "Produtos", garantindo que os valores inseridos sejam maiores ou iguais a zero.

```
CREATE TABLE Produtos (
    ID INT PRIMARY KEY,
    Nome VARCHAR(50),
    Quantidade INT,
    Preco DECIMAL(10,2), CHECK (Quantidade >= 0 AND Preco >= 0)
);
```

Dúvidas ?

Exercício


uniesp
Centro Universitário