BANCO DE DADOS II - COMANDOS E CONCEITOS ADICIONAIS

prof. Rodrigo Barbosa
prof2300@iesp.edu.br
(83)991330759

TOP

- O comando TOP é usado para especificar o número de linhas a serem retornadas em uma consulta. Por exemplo:

```
SELECT TOP 5 * FROM clientes;
```

Isso retornaria as cinco primeiras linhas da tabela "clientes".

INTO

- INTO é utilizado para direcionar o resultado de uma consulta para uma nova tabela. Por exemplo:

```
SELECT coluna1, coluna2 INTO nova_tabela FROM tabela_existente WHERE condição;
```

OFFSET-FETCH

- OFFSET-FETCH é usado para paginar resultados, permitindo pular um número de linhas especificado e retornar um número de linhas após esse deslocamento.
Exemplo:

```
SELECT colunas FROM tabela ORDER BY coluna OFFSET 10 ROWS
FETCH NEXT 5 ROWS ONLY;
```

DISTINCT

- DISTINCT é usado para retornar valores únicos em uma coluna. Por exemplo:

```
SELECT DISTINCT coluna FROM tabela;
```

SUM, COUNT, AVG

- COUNT conta o número de linhas que satisfazem uma condição;
- SUM retorna a soma dos valores de uma coluna;
- AVG calcula a média dos valores em uma coluna. Exemplos:

```
SELECT COUNT(*) FROM tabela;
SELECT SUM(coluna) FROM tabela;
SELECT AVG(coluna) FROM tabela;
```

AND, OR e NOT

- Operadores lógicos utilizados para filtrar resultados em consultas. Exemplo:

```
SELECT coluna FROM tabela WHERE condição1 AND condição2 OR NOT condição3;
```

CASE

- CASE é uma expressão condicional utilizada para realizar avaliações e retornar resultados baseados em condições específicas. Exemplo:

```
SELECT coluna,
CASE
    WHEN condição1 THEN resultado1
    WHEN condição2 THEN resultado2
    ELSE resultado_padrao
END AS novo_nome
FROM tabela;
```

BETWEEN, IN e LIKE

- BETWEEN é usado para verificar se um valor está dentro de um intervalo específico;
- IN é utilizado para verificar se um valor está em um conjunto de valores;
- LIKE é usado para comparar padrões de texto. Exemplos:

```
SELECT coluna FROM tabela WHERE coluna BETWEEN valor1 AND valor2;
SELECT coluna FROM tabela WHERE coluna IN (valor1, valor2, valor3);
SELECT coluna FROM tabela WHERE coluna LIKE 'padrão%';
```

IS NULL

- IS NULL é usado para verificar se um valor em uma coluna é nulo. Exemplo:

```
SELECT coluna FROM tabela WHERE coluna IS NULL;
```

GROUP BY e HAVING

- GROUP BY é usado para agrupar linhas que têm o mesmo valor em uma ou mais colunas;

- HAVING é usado em conjunto com GROUP BY para filtrar resultados de grupos.
Exemplos:

```
SELECT coluna, COUNT(*) FROM tabela GROUP BY coluna;

SELECT coluna FROM tabela GROUP BY coluna HAVING COUNT(*) > 1;
```

UNION e EXCEPT

- UNION combina o resultado de duas ou mais consultas em uma única tabela de resultados;

- EXCEPT retorna linhas da primeira consulta que não estão presentes na segunda consulta. Exemplos:

```
SELECT coluna FROM tabela1 UNION SELECT coluna FROM tabela2;

SELECT coluna FROM tabela1 EXCEPT SELECT coluna FROM tabela2;
```

ALIAS

- Alias é usado para atribuir um nome temporário a uma tabela ou coluna em uma consulta. Exemplo:

```
SELECT coluna AS novo_nome FROM tabela;
```

ANY e ALL

- ANY é utilizado para comparar um valor com qualquer valor retornado por uma subconsulta;

- ALL é utilizado para comparar um valor com todos os valores retornados por uma subconsulta. Exemplos:

```
SELECT coluna FROM tabela WHERE coluna > ANY (SELECT coluna FROM outra_tabela);

SELECT coluna FROM tabela WHERE coluna = ALL (SELECT coluna FROM outra_tabela);
```

TRUNCATE

- TRUNCATE é usado para remover todos os registros de uma tabela, porém de uma forma mais rápida do que o comando DELETE. Exemplo:

```
TRUNCATE TABLE tabela;
```

COALESCE

- COALESCE é um comando utilizado para retornar o primeiro valor não nulo de uma lista de expressões.

- Sintaxe básica:

```
COALESCE(valor1, valor2, valor3, ...);
```

- O comando COALESCE avalia as expressões na ordem em que são passadas e retorna o primeiro valor não nulo encontrado.

COALESCE

- Imagine uma situação onde você tem colunas com valores nulos e precisa de um valor padrão caso encontre um nulo. Por exemplo:

```
SELECT COALESCE(nome, 'Nome não disponível') AS nome_com_valor_padrao FROM clientes;
```

- Neste exemplo, se a coluna "nome" estiver nula, o COALESCE retornará 'Nome não disponível' no lugar do valor nulo.

COALESCE

- É possível utilizar COALESCE com múltiplas colunas para encontrar o primeiro valor não nulo entre elas. Por exemplo:

```
SELECT COALESCE(coluna1, coluna2, coluna3) AS primeiro_nao_nulo
FROM tabela;
```

- O COALESCE aqui retornará o primeiro valor não nulo entre as colunas "coluna1", "coluna2" e "coluna3".

COALESCE

- Flexibilidade: Permite definir valores padrão ou alternativos quando os valores originais são nulos.

- Clareza: Torna o código mais legível ao especificar de forma explícita como lidar com valores nulos.

FUNCTIONS, TRIGGERS, PROCEDURES

- Uma FUNCTION é um objeto no SQL que realiza uma operação específica e retorna um valor ou um conjunto de valores.

- Funcionalidade:

    - Pode aceitar parâmetros e retorna um valor específico.

    - Pode ser utilizada em consultas SQL, outras funções ou procedimentos armazenados.

    - Existem funções embutidas (built-in) e funções definidas pelo usuário.

```
CREATE FUNCTION nome_funcao
(parâmetros)
RETURNS tipo_retorno
AS
BEGIN
-- lógica da função
RETURN valor;
END;
```

FUNCTIONS, TRIGGERS, PROCEDURES

- Um TRIGGER é um tipo de procedimento armazenado que é automaticamente executado ou disparado em resposta a determinados eventos no banco de dados. Esses eventos podem ser INSERT, UPDATE ou DELETE em uma tabela.

- Funcionalidade:

    - Acionado automaticamente quando ocorre uma operação específica (INSERT, UPDATE, DELETE).

    - Pode ser usado para manter a integridade dos dados, realizar auditorias, aplicar ações baseadas em eventos, etc.

```
CREATE TRIGGER nome_trigger
AFTER INSERT ON tabela
FOR EACH ROW
BEGIN
-- lógica a ser executada após a inserção
END;
```

FUNCTIONS, TRIGGERS, PROCEDURES

- Uma STORED PROCEDURE é um bloco nomeado de instruções SQL que pode aceitar parâmetros, ser invocado e executado a qualquer momento.

- Funcionalidade:

    - Armazena um conjunto de instruções SQL para execução repetida.

    - Pode aceitar parâmetros para personalização das operações.

    - Oferece reutilização de código e maior segurança.

```
CREATE PROCEDURE nome_procedimento
    (parâmetros)
    AS
    BEGIN
    -- conjunto de instruções SQL
    END;
```

COMPARAÇÃO ENTRE OS CONCEITOS

- TRIGGER: Disparado automaticamente por eventos em tabelas.

- STORED PROCEDURE: Conjunto de instruções SQL nomeadas para execução sob demanda.

- FUNCTION: Bloco de código que aceita parâmetros e retorna um valor específico.

VANTAGENS

- TRIGGER: Garante integridade dos dados, controla auditorias, aplica ações automatizadas.

- STORED PROCEDURE: Reutilização de código, segurança dos dados, execução de tarefas complexas.

- FUNCTION: Modularidade, reutilização de lógica, simplificação de consultas complexas.

REFERÊNCIAS

Usando JOIN em consultas SQL para combinar tabelas | SQL para Análise de Dados EP.6 - YouTube

Como usar GROUP BY, HAVING e CASE em consultas SQL na prática | SQL para Análise de Dados EP.4 - YouTube

COMO FAZER SELECT CASE NO SQL - YouTube

SQL SERVER - 37 - TRIGGER - Criar gatilhos no seu banco de dados - YouTube

SQL SERVER - 38 - PROCEDURES - Como criar, executar e apagar - YouTube

Vídeo Aula - SQLServer - Criar Funções (Functions) - YouTube