■ Resumão Banco de Dados II - Completo com Explicações e Sintaxes

Aula 01 - Introdução a Banco de Dados

Banco de Dados é uma coleção organizada de dados. O SGBD (Sistema de Gerenciamento de Banco de Dados) permite criar, gerenciar, atualizar e consultar os dados de forma eficiente e segura. A linguagem padrão usada é a SQL (Structured Query Language).
➡■ Comandos básicos de definição e manipulação:

```
-- Criar e apagar banco de dados
CREATE DATABASE nome_bd;   -- Cria um banco de dados
DROP DATABASE nome_bd;     -- Apaga um banco de dados

-- Criar e apagar tabelas
CREATE TABLE nome_tabela (
    coluna1 TIPO,
    coluna2 TIPO
);                         -- Cria uma tabela com colunas
DROP TABLE nome_tabela;    -- Apaga uma tabela

-- Alterar estrutura de tabela
ALTER TABLE nome_tabela ADD coluna TIPO;         -- Adiciona coluna
ALTER TABLE nome_tabela DROP COLUMN coluna;      -- Remove coluna
ALTER TABLE nome_tabela ALTER COLUMN coluna NOVO_TIPO; -- Modifica tipo

-- Inserir, atualizar e remover dados
INSERT INTO tabela (col1, col2) VALUES (valor1, valor2); -- Insere registro
UPDATE tabela SET coluna = valor WHERE condicao;         -- Atualiza registros
DELETE FROM tabela WHERE condicao;                       -- Remove registros

-- Consultar dados
SELECT col1, col2 FROM tabela WHERE condicao;  -- Seleciona colunas com filtro
```

Aula 02 - Consultas Básicas

Consultas são usadas para recuperar informações do banco. O comando principal é o SELECT.

```
-- Seleção simples
SELECT coluna1, coluna2 FROM tabela;     -- Retorna valores das colunas escolhidas
SELECT DISTINCT coluna FROM tabela;      -- Retorna valores únicos (sem repetição)

-- Filtros com WHERE
SELECT * FROM tabela WHERE coluna = valor;                -- Igualdade
SELECT * FROM tabela WHERE coluna BETWEEN v1 AND v2;      -- Intervalo
SELECT * FROM tabela WHERE coluna IN (v1, v2, v3);        -- Pertence a um conjunto
SELECT * FROM tabela WHERE coluna LIKE 'A%';              -- Começa com 'A'
SELECT * FROM tabela WHERE coluna IS NULL;                -- Valores nulos
SELECT * FROM tabela WHERE coluna IS NOT NULL;            -- Diferente de nulo

-- Ordenação
SELECT * FROM tabela ORDER BY coluna ASC;   -- Ordem crescente
SELECT * FROM tabela ORDER BY coluna DESC;  -- Ordem decrescente

-- Limitar registros
SELECT TOP 5 * FROM tabela;   -- (SQL Server) Retorna 5 primeiros
SELECT * FROM tabela LIMIT 5; -- (MySQL) Retorna 5 primeiros

-- Funções de agregação
SELECT MIN(coluna), MAX(coluna), COUNT(*), AVG(coluna), SUM(coluna)
FROM tabela;   -- Retorna mínimo, máximo, contagem, média e soma
```

Aula 03 - Consultas com Joins

Joins permitem combinar dados de várias tabelas usando chaves de ligação.

```
-- INNER JOIN (traz apenas correspondências)
SELECT t1.col, t2.col
FROM tabela1 t1
INNER JOIN tabela2 t2 ON t1.coluna = t2.coluna;
```

```sql
-- LEFT JOIN (traz todos da esquerda, mesmo sem correspondência)
SELECT t1.col, t2.col
FROM tabela1 t1
LEFT JOIN tabela2 t2 ON t1.coluna = t2.coluna;

-- RIGHT JOIN (traz todos da direita, mesmo sem correspondência)
SELECT t1.col, t2.col
FROM tabela1 t1
RIGHT JOIN tabela2 t2 ON t1.coluna = t2.coluna;

-- CROSS JOIN (produto cartesiano)
SELECT * FROM tabela1 CROSS JOIN tabela2;

-- SELF JOIN (tabela unida com ela mesma)
SELECT a.col, b.col
FROM tabela a, tabela b
WHERE a.coluna = b.coluna;

-- UNION (combina resultados e elimina duplicados)
SELECT coluna FROM tabela1
UNION
SELECT coluna FROM tabela2;

-- UNION ALL (combina resultados e mantém duplicados)
SELECT coluna FROM tabela1
UNION ALL
SELECT coluna FROM tabela2;

-- INTERSECT (apenas valores em comum)
SELECT coluna FROM tabela1
INTERSECT
SELECT coluna FROM tabela2;

-- EXCEPT (valores de uma tabela que não estão em outra)
SELECT coluna FROM tabela1
EXCEPT
SELECT coluna FROM tabela2;

-- GROUP BY (agrupamento) e HAVING (filtro de grupo)
SELECT coluna, COUNT(*)
FROM tabela
GROUP BY coluna
HAVING COUNT(*) > 2;

-- EXISTS (verifica existência em subconsulta)
SELECT * FROM tabela1 t1
WHERE EXISTS (SELECT 1 FROM tabela2 t2 WHERE t1.coluna = t2.coluna);

-- ANY / ALL (comparações com subconsultas)
SELECT * FROM tabela WHERE coluna > ANY (SELECT coluna FROM tabela2);
SELECT * FROM tabela WHERE coluna > ALL (SELECT coluna FROM tabela2);
```

Aula 04 - Variáveis, Conversões, IF/ELSE, While

SQL Server permite o uso de variáveis, conversões e estruturas de controle (IF/ELSE e WHILE).

```sql
-- Declarar variáveis
DECLARE @valor INT, @texto VARCHAR(50), @data DATE;

-- Atribuir valores
SET @valor = 10;                                    -- Definindo valor direto
SELECT @valor = coluna FROM tabela WHERE id = 1;   -- Pegando valor de uma tabela

-- Exibir valores
SELECT @valor AS Resultado;   -- Mostra valor armazenado

-- Conversões
SELECT CAST(coluna AS VARCHAR) FROM tabela;             -- Conversão de tipo
SELECT CONVERT(VARCHAR, coluna, 103) FROM tabela;       -- Conversão de data (dd/mm/yyyy)

-- Estruturas condicionais
IF @valor > 10
    PRINT 'Maior que 10';
ELSE
    PRINT 'Menor ou igual a 10';

-- Estrutura de repetição (laço)
```

```
DECLARE @i INT = 0;
WHILE @i < 5
BEGIN
    PRINT @i;       -- Exibe o valor de i
    SET @i = @i + 1;
END;
```

Aula 05 - Funções e Procedures

Funções retornam valores (escalar ou tabela). Procedures executam blocos de comandos SQL.

```
-- Função Escalar (retorna um único valor)
CREATE FUNCTION fn_Dobro(@num INT)
RETURNS INT
AS
BEGIN
    RETURN @num * 2;    -- Retorna o dobro do número
END;

-- Usando função
SELECT dbo.fn_Dobro(5);   -- Retorna 10

-- Função Inline (retorna uma tabela)
CREATE FUNCTION fn_FuncDepto(@id INT)
RETURNS TABLE
AS
RETURN (
    SELECT * FROM FUNCIONARIO WHERE depto_id = @id
);

-- Função Multi-Statement (tabela criada dentro da função)
CREATE FUNCTION fn_SalarioTotal(@id INT)
RETURNS @Tabela TABLE (Nome VARCHAR(50), Salario DECIMAL(10,2))
AS
BEGIN
    INSERT INTO @Tabela
    SELECT Nome, Salario*14   -- Inclui 12 meses + férias + 13º
    FROM FUNCIONARIO WHERE id=@id;
    RETURN;
END;

-- Procedure simples (sem parâmetros)
CREATE PROCEDURE sp_Teste
AS
BEGIN
    SELECT 'Teste Procedure';
END;

-- Executando uma procedure
EXEC sp_Teste;

-- Procedure com parâmetros de entrada
CREATE PROCEDURE sp_AtualizaSalario(@cpf CHAR(11), @novoSal DECIMAL(10,2))
AS
BEGIN
    UPDATE FUNCIONARIO SET Salario=@novoSal WHERE CPF=@cpf;
END;

-- Procedure com parâmetro de saída
CREATE PROCEDURE sp_Soma(@a INT, @b INT, @res INT OUTPUT)
AS
BEGIN
    SET @res = @a + @b;
END;

-- Executando e recebendo saída
DECLARE @resultado INT;
EXEC sp_Soma 5, 7, @resultado OUTPUT;
PRINT @resultado;  -- Exibe 12
```