**BD2 – Prova - Resumo**

**Criando o banco de dados**

**Criar o Banco de Dados**:

CREATE DATABASE clinicavet; // cria o BD

USE clinicavet; // usa o BD

**Criar usuário e conceder privilégios a somente ao banco clinicavet**:

create user uservet@localhost identified by 'cmpvet'; // criar usuário uservet

grant all privileges on clinicavet.\* to uservet@localhost; // conceder privilégios de acesso

**Script de Criação de Tabelas**

CREATE TABLE autores ( -- Criação da tabela de Autores

id\_autor INT PRIMARY KEY,

nome\_autor VARCHAR(100) NOT NULL

);

CREATE TABLE livros ( -- Criação da tabela de Livros

id\_livro INT PRIMARY KEY,

titulo VARCHAR(200) NOT NULL,

id\_autor INT,

FOREIGN KEY (id\_autor) REFERENCES autores(id\_autor)

);

**Script de Povoamento (Inserção de Dados)**

INSERT INTO autores (id\_autor, nome\_autor) VALUES (1, 'Machado de Assis'); -- Inserção de autores

INSERT INTO autores (id\_autor, nome\_autor) VALUES (2, 'Clarice Lispector');

INSERT INTO livros (id\_livro, titulo, id\_autor) -- Inserção de livros

VALUES (1, 'Dom Casmurro', 1);

INSERT INTO livros (id\_livro, titulo, id\_autor)

VALUES (2, 'A Hora da Estrela', 2);

**Consultas Iniciais**

SELECT \* FROM X; // listar todas as informações de uma tabela

SELECT titulo, ano\_publicacao // buscar apenas livros publicados após o ano 1950

FROM livros

WHERE ano\_publicacao > 1950;

SELECT COUNT(\*) AS total\_livros FROM livros; // contar quantos livros há no sistema

**SQL básico/intermediário**

**WHERE (Condição): filtra o que você quer ver**

SELECT \* FROM pessoas WHERE nome = 'Ana';

**Operadores Aritméticos (+, -, \*, /): usado para fazer contas matemáticas simples**

SELECT 10 + 5 AS resultado; -- Resultado: 15

SELECT 10 - 3 AS resultado; -- Resultado: 7

SELECT 4 \* 2 AS resultado; -- Resultado: 8

SELECT 20 / 4 AS resultado; -- Resultado: 5

**Operadores de Comparação (=, >, <, >=, <=, <>, BETWEEN)**

= (igual)

SELECT \* FROM pessoas WHERE nome = 'Ana';

> (maior que)

SELECT \* FROM pessoas WHERE id > 5;

< (menor que)

SELECT \* FROM pessoas WHERE id < 10;

>= (maior ou igual)

SELECT \* FROM pessoas WHERE id >= 5;

<= (menor ou igual)

SELECT \* FROM pessoas WHERE id <= 10;

<> (diferente de)

SELECT \* FROM pessoas WHERE nome <> 'Ana';

BETWEEN (entre dois valores)

SELECT \* FROM pessoas WHERE id BETWEEN 1 AND 5; -- (Pega quem tem ID entre 1 e 5, incluindo o 1 e o 5.)

**Cláusula DISTINCT: traz apenas valores diferentes (sem repetir)**

SELECT DISTINCT nome FROM pessoas; -- mostra uma lista de nomes sem repetições

**Cláusula LIKE: usado para buscar parecidos em uma coluna de texto, usando curingas % e \_**

% = qualquer quantidade de letras

\_ = apenas 1 letra

SELECT \* FROM pessoas WHERE nome LIKE 'A%'; -- começando com "A"

SELECT \* FROM pessoas WHERE nome LIKE '%o'; -- terminando com "o"

SELECT \* FROM pessoas WHERE nome LIKE 'Ma%'; -- começando com "Ma" e mais qualquer coisa depois

**Comparação envolvendo NULL: NULL significa "vazio", "sem valor"**

SELECT \* FROM pessoas WHERE nome IS NULL; -- encontrar registros sem nome (nome vazio)

SELECT \* FROM pessoas WHERE nome IS NOT NULL; -- encontrar registros com nome preenchido

**Cláusula ALL: compara um valor com TODOS os valores -> normalmente usada com operadores de comparação**

SELECT \* FROM pessoas

WHERE id > ALL ( -- pega quem tem ID maior que todos os IDs menores que 5

SELECT id

FROM pessoas

WHERE id < 5

);

**Cláusula UNION: junta resultados de dois SELECTs em uma só lista - tira duplicados automaticamente**

SELECT nome FROM pessoas

UNION -- mostra nomes de pessoas e de cidades juntos

SELECT nome\_cidade FROM cidades;

**Cláusula ORDER BY: organiza os resultados em ordem, de forma ascendente ou decrescente.**

SELECT \* FROM pessoas ORDER BY nome ASC; -- ordem crescente (A-Z / menor para maior)

SELECT \* FROM pessoas ORDER BY nome DESC; -- ordem decrescente (Z-A / maior para menor)

SELECT \* FROM pessoas ORDER BY nome ASC, id DESC; -- ordenar por nome (cresc) e depois por ID (decresc)