Documentação Técnica do Projeto de Banco de Dados - Projeto AdoCão

Introdução ao Projeto AdoCão

Descrição do Projeto

Este projeto foi desenvolvido para gerenciar uma ong que resgata cachorros, permitindo que adotantes adotem cachorros, os administradores controlem a exibição dos animais que estão para a adoção, o histórico médico e adicionem novos animais no sistema.

Objetivo do Banco de Dados

O objetivo do banco de dados é organizar e gerenciar informações sobre cachorros e adotantes, garantindo eficiência no processo de adoção e gerenciamento do histórico dos animais.

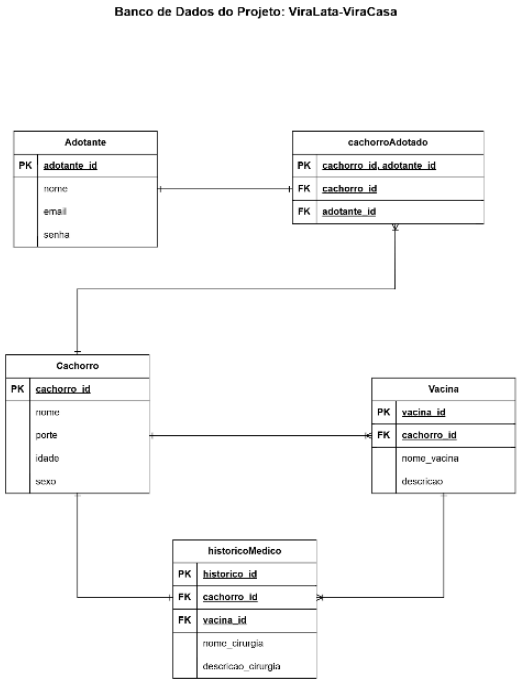
Participantes

- Ashmide JN Baptise - CP3025501
- Melissa Batista Junqueira - CP3029832
- Rafaela Laryssa Mello Neto - CP303061X
- Sophia Ferreira Boonen - CP3031756

Modelagem de Dados

Diagrama Entidade-Relacionamento (DER)

O Diagrama Entidade-Relacionamento (DER) ilustra as entidades e relacionamentos do sistema, como mostrado abaixo:



Estrutura do Banco de Dados

Scripts de Criação do Banco de Dados

Os scripts SQL para criação das tabelas e seus relacionamentos são apresentados abaixo:

```sql
CREATE TABLE adotante(
        adotante_id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(160),
    email VARCHAR(350) NOT NULL,
    senha VARCHAR(12) NOT NULL
);
```

```sql
CREATE TABLE cachorro (
        cachorro_id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    porte VARCHAR(50) NOT NULL,
    idade VARCHAR(10) NOT NULL,
    sexo ENUM('Macho', 'Fêmea')
);
```

```sql
CREATE TABLE vacina (
    vacina_id INT AUTO_INCREMENT PRIMARY KEY,
    nome_vacina VARCHAR(100) NOT NULL,
    descricao TEXT NOT NULL,
    disponivel_adocao BOOLEAN DEFAULT TRUE
);
```

```sql
CREATE TABLE historicoMedico (
    historico_id INT AUTO_INCREMENT PRIMARY KEY,
    nome_cirurgia VARCHAR(100) NOT NULL,
    descricao_cirurgia TEXT NOT NULL,
    cachorro_id INT NOT NULL,
    vacina_id INT NOT NULL,
    FOREIGN KEY (cachorro_id) REFERENCES cachorro(cachorro_id),
    FOREIGN KEY (vacina_id) REFERENCES vacina(vacina_id)
);
```

```sql
CREATE TABLE cachorro_adotado(
    cachorro_id INT NOT NULL,
    adotante_id INT NOT NULL,
    data_adocao DATE NOT NULL,
    data_volta DATE,
    PRIMARY KEY (cachorro_id, adotante_id),
    FOREIGN KEY(cachorro_id) REFERENCES cachorro(cachorro_id),
    FOREIGN KEY(adotante_id) REFERENCES adotante(adotante_id)
);
```

Descrição das Tabelas

- **Tabela adotante**: Armazena informações dos usuários que realizam adoções no sistema.
    - adotante_id: Identificador único do adotante (gerado automaticamente).
    - nome: Nome completo do adotante.
    - email: Endereço de e-mail do adotante (obrigatório).
    - senha: Senha de acesso ao sistema (obrigatória)
- **Tabela cachorro:** Armazena informações dos cães disponíveis para adoção.
    - cachorro_id: Identificador único do cachorro.
    - nome: Nome do cachorro.
    - porte: Tamanho do cachorro (ex: pequeno, médio, grande).
    - idade: Idade do cachorro.
    - sexo: Sexo do cachorro (Macho ou Fêmea).
- **Tabela vacina:** Contém os dados das vacinas cadastradas no sistema.
    - vacina_id: Identificador único da vacina.
    - nome_vacina: Nome da vacina.
    - descricao: Descrição detalhada da vacina.
    - disponivel_adocao: Indica se a vacina está disponível para cães aptos à adoção.
- **Tabela historicoMedico:** Registra o histórico médico dos cães, incluindo cirurgias e vacinas aplicadas.
    - historico_id: Identificador único do registro médico.
    - nome_cirurgia: Nome da cirurgia realizada no cachorro.
    - descricao_cirurgia: Detalhes sobre o procedimento cirúrgico.
    - cachorro_id: Identificador do cachorro (chave estrangeira da tabela cachorro).

    - vacina_id: Identificador da vacina aplicada (chave estrangeira da tabela vacina).
- **Tabela cachorro:** Armazena informações dos cães disponíveis para adoção.
    - cachorro_id: Identificador único do cachorro.
    - nome: Nome do cachorro.
    - porte: Tamanho do cachorro (ex: pequeno, médio, grande).
    - idade: Idade do cachorro.
    - sexo: Sexo do cachorro (Macho ou Fêmea).

Consultas e Funcionalidades

Descrição das Consultas

1. **Consulta mostrando o vínculo entre adotante e cachorro**: Exibi todos os cães adotados e as datas das adoções.

```sql
SELECT a.nome AS 'Adotante', c.nome AS 'Cachorro', ca.data_adocao AS 'Data de Adoção'
FROM adotante AS a
  JOIN cachorro_adotado ca ON a.adotante_id = ca.adotante_id
  JOIN cachorro c ON c.cachorro_id = ca.cachorro_id
WHERE ca.data_volta IS NULL;
```

2. **Consulta para exibir informações de um cachorro**: Exibi todas as informações de um animal incluindo o histórico médico.

```sql
SELECT c.nome AS 'Nome do Cachorro', c.porte AS 'Porte', c.sexo AS 'Sexo', v.nome_vacina AS 'Vacina',
his.nome_cirurgia AS 'Cirurgia', his.descricao_cirurgia AS 'Descrição da Cirurgia'
FROM cachorro c
  JOIN historicoMedico his ON his.cachorro_id = c.cachorro_id
  JOIN vacina v ON his.vacina_id = v.vacina_id
ORDER BY c.nome ASC;
```

3. **Consulta de exibição de adotantes**: Exibe os nomes dos adotantes e quantos cachorros eles adotaram.

```sql
SELECT a.nome AS 'Adotante', COUNT(c.cachorro_id) AS 'Quantidade cachorros'
        FROM adotante a
JOIN cachorro_adotado ca ON ca.adotante_id = a.adotante_id
JOIN cachorro c ON c.cachorro_id = ca.cachorro_id
        GROUP BY a.adotante_id;
```

4. **Consulta de exibição de devoluções**: Exibe os cachorros que foram devolvidos e as datas de devolução.

```sql
SELECT c.nome AS 'Cachorro devolvido', ca.data_volta as 'Data Devolução'
        FROM cachorro c
JOIN cachorro_adotado ca ON ca.cachorro_id = c.cachorro_id
WHERE data_volta IS NOT NULL;
```

5. **Consulta de exibição de animais adotados**: Exibe os cachorros que foram adotados em 2024.

```sql
SELECT c.nome AS 'Cachorro adotado', ca.data_adocao AS 'Data Adoção'
        FROM cachorro c
JOIN cachorro_adotado ca ON ca.cachorro_id = c.cachorro_id
WHERE YEAR(ca.data_adocao) = 2024;
```

6. **Consulta de exibição da idade dos animais**: Exibe os cachorros com idade maior ou igual a 3 anos que realizaram cirurgias.

```sql
SELECT c.nome AS 'Nome do Cachorro', c.idade AS 'Idade' , his.nome_cirurgia AS 'Cirurgia',
       his.descricao_cirurgia AS 'Descrição da Cirurgia'
FROM cachorro c
  JOIN historicoMedico his ON his.cachorro_id = c.cachorro_id
WHERE c.idade LIKE '%3 anos%';
```

7. **Consulta de exibição de cachorros vacinados**: Exibe os animais que receberam a vacina V10.

```sql
SELECT c.nome AS 'Nome cachorro'
        FROM cachorro c
JOIN historicoMedico his ON his.cachorro_id = c.cachorro_id
JOIN vacina v ON v.vacina_id = his.vacina_id
        WHERE nome_vacina = 'V10';
```

8. **Consulta de exibição de animais por cirurgia**: Exibe a quantidade de cachorros por tipo de cirurgia.

```sql
SELECT nome_cirurgia, COUNT(*) AS quantidade
        FROM historicoMedico
GROUP BY nome_cirurgia
ORDER BY quantidade DESC;
```

9. **Consulta de exibição de histórico médico do cachorro**: Exibe o histórico médico completo do animal mesmo se ele não for vacinado.

```sql
SELECT c.nome AS 'Cachorro', his.nome_cirurgia AS 'Cirurgia', v.nome_vacina AS 'Vacina'
        FROM cachorro c
  LEFT JOIN historicoMedico his ON c.cachorro_id = his.cachorro_id
  LEFT JOIN vacina v ON his.vacina_id = v.vacina_id;
```

10. **Consulta de exibição de cachorros para adoção**: Exibe os animais que ainda não foram adotados.

```sql
SELECT c.nome AS 'Cachorro disponível', c.cachorro_id
        FROM cachorro c
  LEFT JOIN cachorro_adotado ca ON c.cachorro_id = ca.cachorro_id AND ca.data_volta IS NULL
WHERE ca.cachorro_id IS NULL;
```

11. **Consulta de exibição do nome das vacinas**: Exibe o nome de todas as vacinas associadas a cada cachorro.

```sql
SELECT c.nome AS 'Cachorro', GROUP_CONCAT(DISTINCT v.nome_vacina SEPARATOR ', ') AS 'Vacinas Aplicadas'
        FROM cachorro c
  LEFT JOIN historicoMedico his ON c.cachorro_id = his.cachorro_id
  LEFT JOIN vacina v ON v.vacina_id = his.vacina_id
GROUP BY c.cachorro_id, c.nome
ORDER BY c.nome;
```

Descrição das Funcionalidades Views

1. **Exibição do status do cachorro**: Uma *view* que exibe se o animal está disponível para adoção e se não estiver mostra em que ele foi adotado.

```sql
CREATE VIEW view_status_cachorros AS
       SELECT c.cachorro_id AS 'ID', c.nome AS 'Nome Cachorro', c.porte AS 'Porte', c.idade AS 'Idade', c.sexo AS 'Sexo',
    IF(
        EXISTS (
            SELECT 1 FROM cachorro_adotado ca
            WHERE ca.cachorro_id = c.cachorro_id AND ca.data_volta IS NULL
        ),
        'Indisponível', 'Disponível'
    ) AS status,
    (   SELECT DATEDIFF(CURDATE(), ca.data_adocao)
                    FROM cachorro_adotado ca
        WHERE ca.cachorro_id = c.cachorro_id AND ca.data_volta IS NULL
        LIMIT 1
    ) AS dias_com_adotante
FROM cachorro c;

# Mostrar view
SELECT * FROM view_status_cachorros;
```

2. **Exibição de animais vacinados**: Uma *view* que exibe todas as vacinas que um cachorro tomou, as agrupando por nome.

```
CREATE VIEW view_vacinas_por_cachorro AS
        SELECT c.cachorro_id AS 'ID', c.nome AS 'Nome Cachorro',
          GROUP_CONCAT(DISTINCT v.nome_vacina SEPARATOR ', ') AS 'Vacinas'
                FROM cachorro c
          LEFT JOIN historicoMedico h ON c.cachorro_id = h.cachorro_id
          LEFT JOIN vacina v ON v.vacina_id = h.vacina_id
        GROUP BY c.cachorro_id, c.nome;

# Mostrar a view
SELECT * FROM view_vacinas_por_cachorro;
```

3. **Exibição de cachorros adotados:** Mostra os animais adotados que ainda estão com os adotantes.

```
CREATE VIEW view_adocoes_ativas AS
       SELECT a.nome AS 'Nome Adotante', a.email AS 'E-mail', c.nome AS 'Nome Cachorro', ca.data_adocao as 'Data de Adoção'
                FROM cachorro_adotado ca
         JOIN adotante a ON a.adotante_id = ca.adotante_id
         JOIN cachorro c ON c.cachorro_id = ca.cachorro_id
    WHERE ca.data_volta IS NULL;

#Mostrar a view
SELECT * FROM view_adocoes_ativas;
```

4. **Exibição histórico completo do cachorro:** Mostra todo o histórico do animal, desde vacinas e histórico médico até se foi adotado, data de adoção e nome de adotante.

```
CREATE VIEW view_historico_completo AS
        SELECT c.cachorro_id AS 'ID', c.nome AS 'Nome Cachorro', hist.nome_cirurgia AS 'Cirurgia',
      hist.descricao_cirurgia AS 'Descrição Cirurgia', v.nome_vacina AS 'Vacina',
      a.nome AS 'Nome Adotante', ca.data_adocao AS 'Data de Adoção'
                FROM cachorro c
        LEFT JOIN historicoMedico hist ON c.cachorro_id = hist.cachorro_id
        LEFT JOIN vacina v ON hist.vacina_id = v.vacina_id
        LEFT JOIN cachorro_adotado ca ON c.cachorro_id = ca.cachorro_id AND ca.data_volta IS NULL
        LEFT JOIN adotante a ON ca.adotante_id = a.adotante_id;

# Mostrar view
SELECT * FROM view_historico_completo;
```

Descrição das Funcionalidades Triggers

1. **Criação de Log**: Log criado para registrar todos os dados de cachorros adotados como forma de manter os dados seguros.

```
CREATE TABLE log_adocao (
    log_id INT AUTO_INCREMENT PRIMARY KEY,
    cachorro_id INT,
    adotante_id INT,
    data_adocao DATETIME,
    data_log TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

2. **Adição de dados:** Trigger para adicionar os dados de cachorro adotado para o log como forma de manter os dados salvos em um "backup" de uma forma mais segura e sem perder os dados registrados de adoção.

```
DELIMITER //
CREATE TRIGGER after_adocao_insert
        AFTER INSERT ON cachorro_adotado
                FOR EACH ROW
BEGIN
    INSERT INTO log_adocao (cachorro_id, adotante_id, data_adocao)
                VALUES (NEW.cachorro_id, NEW.adotante_id, NEW.data_adocao);
END//
DELIMITER ;
```

3. **Evitar Duplicatas:** Trigger para evitar que duas pessoas adotem um mesmo cachorro ao mesmo tempo.

```
DELIMITER //
CREATE TRIGGER before_adocao_insert
        BEFORE INSERT ON cachorro_adotado
                FOR EACH ROW
BEGIN
    DECLARE ja_adotado INT;

    SELECT COUNT(*) INTO ja_adotado
                FROM cachorro_adotado
    WHERE cachorro_id = NEW.cachorro_id AND data_volta IS NULL;

    IF ja_adotado > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Este cachorro já está adotado e ainda não foi devolvido.';
    END IF;
END//
DELIMITER ;
```

4. **Adição de dados:** Trigger para adicionar a data de adoção, caso não seja informada.

```
DELIMITER //
CREATE TRIGGER before_adocao_default_data
        BEFORE INSERT ON cachorro_adotado
                FOR EACH ROW
BEGIN
    IF NEW.data_adocao IS NULL THEN
        SET NEW.data_adocao = CURDATE();
    END IF;
END//
DELIMITER ;
```

5. **Criação de Log:** Log criado para registrar alterações no histórico médico dos cachorros.

```
CREATE TABLE log_historico_medico (
    log_id INT AUTO_INCREMENT PRIMARY KEY,
    cachorro_id INT,
    vacina_id INT,
    nome_cirurgia VARCHAR(100),
    data_log TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    acao VARCHAR(20)
);
```

6. **Adição de dados:** Trigger para todas as adições novas no histórico médico.

```
DELIMITER //
CREATE TRIGGER after_historico_insert
        AFTER INSERT ON historicoMedico
                FOR EACH ROW
BEGIN
    INSERT INTO log_historico_medico (
        cachorro_id, vacina_id, nome_cirurgia, acao
    )
    VALUES (
        NEW.cachorro_id, NEW.vacina_id, NEW.nome_cirurgia, 'INSERÇÃO'
    );
END//
DELIMITER ;
```

7. **Alteração de dados:** Trigger para alterar o estado de adoção disponível, caso seja adotado.

```
DELIMITER //
CREATE TRIGGER after_adocao_cachorro_indisponivel
        AFTER INSERT ON cachorro_adotado
                FOR EACH ROW
BEGIN
    UPDATE cachorro
    SET disponivel_adocao = FALSE
    WHERE cachorro_id = NEW.cachorro_id;
END//
DELIMITER ;
```

8. **Adição de dados:** Trigger para caso o cachorro retorne.

```
DELIMITER //
CREATE TRIGGER after_devolucao_cachorro_disponivel
        AFTER UPDATE ON cachorro_adotado
                FOR EACH ROW
BEGIN
    IF
                NEW.data_volta IS NOT NULL
                        AND OLD.data_volta IS NULL
        THEN
        UPDATE cachorro
        SET disponivel = TRUE
        WHERE cachorro_id = NEW.cachorro_id;
    END IF;
END//
DELIMITER ;
```

Descrição das Funcionalidades Procedures

1. **Adição de dados:** Procedure para já inserir um cachorro adotado com a data de adoção para a atual.

```
DELIMITER //
CREATE PROCEDURE registrar_adocao (IN p_cachorro_id INT, IN p_adotante_id INT)
BEGIN
    INSERT INTO cachorro_adotado (cachorro_id, adotante_id, data_adocao)
    VALUES (p_cachorro_id, p_adotante_id, CURDATE());
END//
DELIMITER ;

CALL registrar_adocao(2, 3); ## Terá um aviso, por conta do trigger limitando a adoção de um cachorro já adotado
CALL registrar_adocao(4, 5); ## Deu certo
```

2. **Adição de dados:** Procedure para inserir um cachorro na tabela com o status "disponível" já definido.

```
DELIMITER //
CREATE PROCEDURE cadastrar_cachorro (IN p_nome VARCHAR(100), IN p_porte VARCHAR(50), IN p_idade VARCHAR(10),
        IN p_sexo ENUM('Macho', 'Fêmea'), IN disponivel_adocao BOOLEAN)
BEGIN
    INSERT INTO cachorro (nome, porte, idade, sexo, disponivel_adocao) VALUES (p_nome, p_porte, p_idade, p_sexo, TRUE);
END//
DELIMITER ;

# Chamando a procedure cadastrar_cachorro() para o cachorro Chico (id
CALL cadastrar_cachorro('Chico', 'Médio', '2 anos', 'Macho');
```

3. **Adição de dados:** Procedure para inserir histórico médico em um cachorro com restrição de verificar se o cachorro existe e se a vacina existe, antes de inserir.

```
DELIMITER //
CREATE PROCEDURE inserir_historico (IN p_cachorro_id INT, IN p_vacina_id INT, IN p_nome_cirurgia VARCHAR(100),
        IN p_descricao TEXT)
BEGIN
    IF EXISTS (SELECT 1 FROM cachorro WHERE cachorro_id = p_cachorro_id) AND
       EXISTS (SELECT 1 FROM vacina WHERE vacina_id = p_vacina_id) THEN
                    INSERT INTO historicoMedico (cachorro_id, vacina_id, nome_cirurgia, descricao_cirurgia)
                            VALUES (p_cachorro_id, p_vacina_id, p_nome_cirurgia, p_descricao);
    END IF;
END//
DELIMITER ;

# Chamando a procedure inserir_historico() para o cachorro de id 2 (Luna), com a vacina 3 (Vacina V8)
CALL inserir_historico(2, 3, 'Extração dentária', 'Remoção de dente infectado após exame odontológico.');
```

Descrição das Funcionalidades Functions

1. **Verificação de dados:** Function para verificar se o cachorro está disponível ou não para adoção.

```
DELIMITER //
CREATE FUNCTION status_disponibilidade(p_cachorro_id INT)
        RETURNS VARCHAR(20) DETERMINISTIC
BEGIN
    DECLARE adotado INT;

    SELECT COUNT(*) INTO adotado
                FROM cachorro_adotado
    WHERE cachorro_id = p_cachorro_id AND data_volta IS NULL;

    RETURN IF(adotado = 0, 'Disponível', 'Indisponível');
END//
DELIMITER ;
```

2. **Verificação de dados:** Function para ver quantas vezes um cachorro foi devolvido.

```
DELIMITER //
CREATE FUNCTION total_devolucoes(p_cachorro_id INT)
        RETURNS INT DETERMINISTIC
BEGIN
    DECLARE total INT;

    SELECT COUNT(*) INTO total
    FROM cachorro_adotado
    WHERE cachorro_id = p_cachorro_id AND data_volta IS NOT NULL;

    RETURN total;
END//
DELIMITER ;

# Chamando a function
SELECT nome, total_devolucoes(cachorro_id) AS 'Devoluções' FROM cachorro;
```

3. **Cálculo de dias:** Function para calcular quantos dias o cachorro está com o adotante, caso ele ainda não tenha sido devolvido.

```
DELIMITER //
CREATE FUNCTION dias_com_adotante(p_cachorro_id INT)
        RETURNS INT DETERMINISTIC
BEGIN
    DECLARE dias INT;

    SELECT DATEDIFF(NOW(), data_adocao) INTO dias
    FROM cachorro_adotado
    WHERE cachorro_id = p_cachorro_id AND data_volta IS NULL
    LIMIT 1;

    RETURN IFNULL(dias, 0);
END//
DELIMITER ;

# Chamando a function
SELECT nome, dias_com_adotante(cachorro_id) AS 'Dias com adotante' FROM cachorro;
```

Descrição das Funcionalidades Transaction

1. **Registro de dados:** Transaction registra a adoção do cachorro e atualiza o status de disponibilidade dele.

```
START TRANSACTION;

INSERT INTO cachorro_adotado (cachorro_id, adotante_id, data_adocao)
VALUES (10, 2, NOW());

UPDATE cachorro SET disponivel = FALSE WHERE cachorro_id = 10;

COMMIT;
```