Projeto Laboratório de Banco de Dados

Título do Projeto: Desenvolvimento de um Sistema de Gestão para Clínica Médica

Disciplina: Linguagem de Programação para Banco de Dados

Curso: Ciência da Computação e Sistemas de Informação

Semestre: 3º e 4º Semestre

Tecnologias: SGBD MySQL (via XAMPP) e phpMyAdmin

Grupo: Grupo de APS

Data limite da Entrega: Dia da avaliação de NP2

Postagem: Postagem no Teams em formato DOC ou PDF

1. Contexto e Objetivo

Contexto:

Uma clínica médica de porte médio, que atualmente gerencia seus processos de forma manual (fichas em papel, agendas físicas), busca modernizar suas operações para aumentar a eficiência, reduzir erros e oferecer um melhor serviço aos pacientes. A clínica necessita de uma solução digital robusta para centralizar informações de pacientes, médicos, especialidades, consultas e prontuários.

Objetivo Principal:

O objetivo deste projeto é projetar, implementar e documentar um banco de dados relacional completo para suportar as operações da clínica médica. O aluno deverá demonstrar domínio sobre conceitos avançados de modelagem e administração de bancos de dados, utilizando o SGBD MySQL. O sistema deve ser capaz de gerenciar o cadastro de pacientes e médicos, o agendamento de consultas, o registro de prontuários e fornecer relatórios básicos, garantindo a integridade, a consistência e o desempenho dos dados.

2. Estrutura do Banco de Dados (DDL)

O aluno deverá projetar e implementar a estrutura lógica do banco de dados. A arquitetura deve seguir os princípios de normalização (até a 3ª Forma Normal) e ser descrita utilizando a Linguagem de Definição de Dados (DDL).

Tabelas Propostas (Mínimo Obrigatório):

- `Pacientes` (id_paciente, nome, cpf, data_nascimento, telefone, email)
- `Medicos` (id_medico, nome, crm, id_especialidade, data_nascimento, telefone)
- `Especialidades` (id_especialidade, nome, descricao)

- `Consultas` (id_consulta, id_paciente, id_medico, data_consulta, hora_inicio, hora_fim, status)
- `Prontuarios` (id_prontuario, id_consulta, anamnese, diagnostico, prescricao, data_registro)

Requisitos de Implementação (DDL):

- Criação de Tabelas (`CREATE TABLE`): Definir todas as tabelas com seus respectivos nomes, colunas, tipos de dados adequados (`INT`, `VARCHAR`, `DATE`, `TIME`, `TEXT`, `DECIMAL`, etc.).
- Restrições de Integridade:
    - Integridade de Entidade: Definir chaves primárias (`PRIMARY KEY`) para cada tabela.
    - Integridade Referencial: Definir chaves estrangeiras (`FOREIGN KEY`) para relacionar as tabelas (ex: `id_paciente` em `Consultas` referencia `Pacientes`). Especificar as ações de `ON DELETE` e `ON UPDATE` (ex: `RESTRICT`, `CASCADE`).
    - Integridade de Domínio: Utilizar restrições como `NOT NULL` (campos obrigatórios), `UNIQUE` (valores únicos, como CPF e CRM) e `CHECK` (para validações, ex: `data_consulta >= CURRENT_DATE`).
- Alteração de Tabelas (`ALTER TABLE`): Demonstrar o uso de `ALTER TABLE` para adicionar uma nova coluna (ex: adicionar o campo `logradouro` na tabela `Pacientes`) ou modificar uma coluna existente.
- Eliminação de Tabelas (`DROP TABLE`): Incluir um script de exemplo que demonstre a remoção segura de uma tabela, considerando as dependências.

3. Manipulação de Dados (DML)

Esta seção foca na interação com os dados armazenados. O aluno deve implementar scripts que demonstrem a manipulação das informações.

Requisitos de Implementação (DML):

- Inclusão (`INSERT`): Criar scripts para popular o banco de dados com dados de exemplo para todas as tabelas.
- Modificação (`UPDATE`): Desenvolver comandos `UPDATE` para modificar dados em diferentes cenários (ex: atualizar o telefone de um paciente, alterar o status de uma consulta).
- Exclusão (`DELETE`): Implementar comandos `DELETE` para remover registros, demonstrando o cuidado necessário para não violar a integridade referencial.
- Impacto da Integridade Referencial: Elaborar um relatório ou comentários no código que expliquem o impacto das regras de integridade referencial definidas no DDL. Por exemplo:
    - O que acontece ao tentar excluir um `Medico` que possui consultas agendadas? (Se `ON DELETE RESTRICT`, a operação falhará).
    - Qual o efeito de atualizar o `id_paciente` na tabela `Pacientes` se a chave estrangeira em `Consultas` tiver `ON UPDATE CASCADE`? (O ID será atualizado automaticamente nas consultas relacionadas).

4. Procedimentos e Funções

O aluno deve utilizar as extensões procedurais do SQL para encapsular lógica de negócio diretamente no banco de dados.

Requisitos de Implementação:

- Procedimento Armazenado (`PROCEDURE`): Criar um procedimento para agendar uma nova consulta. O procedimento deve receber o ID do paciente, ID do médico, data e hora como parâmetros. Internamente, ele deve verificar se o médico já possui uma consulta no mesmo horário antes de inserir o novo registro. Se houver conflito, deve retornar uma mensagem de erro.
- Função (`FUNCTION`): Desenvolver uma função escalar que calcule a idade de um paciente ou médico a partir da sua data de nascimento. A função deve receber a data de nascimento e retornar a idade em anos.
- Gatilho (`TRIGGER`): Implementar um gatilho para auditoria. Criar uma tabela `auditoria_prontuarios` e um `TRIGGER` que seja acionado sempre que um registro na tabela `Prontuarios` for atualizado (`AFTER UPDATE`). O gatilho deve inserir na tabela de auditoria o ID do prontuário alterado, o antigo diagnóstico, o novo diagnóstico e a data/hora da alteração.

5. Controle de Transações

O projeto deve garantir a atomicidade e a consistência das operações que envolvem múltiplos passos.

Requisitos de Implementação:

- Plano de Controle de Transações: Identificar uma operação crítica que deva ser tratada como uma unidade lógica de trabalho. O agendamento de consulta (procedimento da seção anterior) é um exemplo perfeito.
- Implementação: O procedimento de agendamento deve ser reescrito para utilizar comandos de controle de transação: `START TRANSACTION`, `COMMIT` e `ROLLBACK`.
    - A transação deve iniciar, verificar a disponibilidade do horário e, em caso de sucesso, inserir a consulta e efetivar a transação com `COMMIT`.
    - Se qualquer etapa falhar (ex: horário indisponível), a transação deve ser desfeita com `ROLLBACK` para garantir que nenhum dado parcial seja salvo.
- Controle de Concorrência: Descrever brevemente como o SGBD MySQL (com o motor InnoDB) lida com o acesso concorrente aos dados. Mencionar o uso de bloqueios (locks) e os níveis de isolamento de transação (`READ COMMITTED`, `REPEATABLE READ`) como mecanismos para evitar problemas como leituras sujas ou leituras não repetíveis.

6. Otimização de Consultas

O aluno deve demonstrar a capacidade de analisar e melhorar o desempenho das consultas ao banco de dados.

Requisitos de Implementação:

- Identificação de Consultas Críticas: Selecionar pelo menos duas consultas `SELECT` complexas que seriam frequentes no sistema (ex: "Listar todas as consultas de um paciente nos últimos 6 meses" ou "Encontrar o próximo horário disponível de um determinado médico").
- Análise com `EXPLAIN`: Utilizar o comando `EXPLAIN` nas consultas selecionadas para analisar o plano de execução gerado pelo MySQL antes da otimização.
- Criação de Índices: Criar índices (`CREATE INDEX`) em colunas apropriadas para otimizar as consultas identificadas (ex: índice em `id_paciente` e `data_consulta` na tabela `Consultas`).
- Verificação da Otimização: Executar o `EXPLAIN` novamente nas mesmas consultas após a criação dos índices para comprovar a melhoria no plano de execução (ex: mudança de "full table scan" para uso de índice).
- Estruturas de Índices: Explicar conceitualmente como os índices funcionam, mencionando que o MySQL InnoDB utiliza estruturas de Árvores B+ para seus índices, o que os torna eficientes para buscas por igualdade e por intervalo.

7. Implementação e Testes

O aluno deve apresentar um plano claro para o desenvolvimento e validação do projeto.

Roadmap de Implementação:

- Fase 1 - Modelagem: Criação do Modelo Entidade-Relacionamento (MER) e do projeto lógico.
- Fase 2 - DDL e Carga de Dados: Escrita e execução dos scripts DDL. Carga de dados de exemplo usando DML.
- Fase 3 - Lógica de Negócio: Implementação de procedimentos, funções e triggers.
- Fase 4 - Transações e Otimização: Implementação do controle de transações e análise/criação de índices.
- Fase 5 - Testes e Documentação: Execução da bateria de testes e compilação da documentação final.

Metodologia de Testes:

- Testes de Integridade: Tentativas de inserir dados inválidos (ex: CPF duplicado, consulta com médico inexistente) para verificar se as restrições são acionadas.
- Testes de Lógica: Testar o procedimento de agendamento com cenários de sucesso e de falha (conflito de horário).
- Testes de Gatilhos: Atualizar um prontuário e verificar se a tabela de auditoria foi populada corretamente.
- Testes de Desempenho: Comparar o tempo de execução de consultas complexas antes e depois da criação de índices.

8. Considerações Finais

Melhores Práticas do Setor:

- Normalização: Justificar as escolhas de normalização do banco.
- Nomenclatura: Utilizar uma convenção de nomes clara e consistente (ex: `snake_case` para nomes de tabelas e colunas).

- Segurança: Mencionar que, em um ambiente real, senhas nunca seriam armazenadas em texto plano, e sim utilizando hashes (ex: bcrypt).
- Versionamento: Sugerir o uso de um sistema de controle de versão (como o Git) para gerenciar os scripts SQL.

Possíveis Desafios e Soluções:

- Desafio: Gerenciar o agendamento de consultas em um ambiente com múltipos usuários (recepção) ao mesmo tempo.
    - Solução: Utilizar transações com um nível de isolamento adequado e bloqueios explícitos (`SELECT ... FOR UPDATE`) para evitar a dupla alocação do mesmo horário.
- Desafio: Migração de dados de um sistema legado.
    - Solução: Desenvolver scripts ETL (Extract, Transform, Load) para limpar e importar os dados antigos para a nova estrutura.

Recursos Necessários:

- Tempo: Estimativa de 6 a 8 semanas.
- Habilidades: Conhecimento em modelagem de dados, SQL (DDL, DML, DCL, TCL), SQL procedural e conceitos de otimização.
- Ferramentas: Pacote XAMPP, MySQL Workbench ou phpMyAdmin, editor de texto.

Critérios de Sucesso e Prazos:

- Entrega Mínima: Um arquivo `.sql` com todos os scripts de criação (DDL, DML, procedimentos, etc.) e um relatório em PDF (até 10 páginas) documentando o projeto (MER, justificativas de design, resultados dos testes e da otimização).
- Critérios de Avaliação:
    - Correção e completude do modelo de dados e DDL (30%).
    - Funcionalidade e complexidade da lógica procedural (DML, Procedures, Functions, Triggers) (30%).
    - Aplicação correta de transações e otimização de consultas (20%).
    - Qualidade da documentação e clareza do relatório (20%).
- Prazo Final: Data de entrega definida no calendário acadêmico da disciplina.