|  |  |  |
| --- | --- | --- |
|  | **CENTRO UNIVERSITÁRIO DE PATOS DE MINAS – UNIPAM**  **BACHARELADO EM SISTEMAS DE INFORMAÇÃO**  **SISTEMAS DE BANCO DE DADOS – 4º PERÍODO**  **PROFESSOR RAFAEL MARINHO E SILVA** | |
| **ALUNO (A): Caio César Alves de Freitas** | |
| **VALOR: 4 PONTOS** | **DATA: 07/11/2025** |
| A lista de exercícios deverá ser resolvida manualmente e entregue em formato PDF pelo Portal UNIPAM até o dia 07/11/2025. | | |

**LISTA DE EXERCÍCIOS**

Com base nos conceitos e conteúdos sobre Banco de Dados estudados em aulas expositivas e na resolução de exercícios, utilize os exemplos e materiais das aulas práticas, bem como o livro Sistemas de Banco de Dados, 7ª edição (Capítulos 6 e 7), disponível na biblioteca do UNIPAM, para responder a todas as questões a seguir.

**Instruções gerais**

* Conecte-se ao banco plano\_saude fornecido no roteiro e execute os comandos solicitados.
* Para cada comando SQL solicitado anexe um print contendo (i) o comando executado e (ii) o resultado retornado pelo SGBD.
* **Entrega: arquivo PDF**.

**QUESTÕES TEÓRICAS**

|  |
| --- |
| Para cada pergunta abaixo responda consultando o livro.   1. O que é SQL?   **SQL é a linguagem padrão para Sistemas de Gerenciamento de Banco de Dados Relacionais, fornecendo comandos para estruturar o banco de dados (DDL) e comandos para interagir com os dados armazenados (DML).**   1. Explique o conceito de cada categoria DDL, DML, DQL, DTL e DCL.  * **Linguagem de Definição de Dados (DDL): usada para definir o esquema e estruturar o banco de dados.** * **Linguagem de Manipulação de Dados (DML): usada para manipular os dados do banco.** * **Linguagem de Consulta de Dados (DQL): usada para, especificamente, consultar os dados armazenados, sendo parte da DML.** * **Linguagem de Controle de Dados (DCL): usada para segurança e autorização das estruturas do banco de dados, podendo ser considerada parte da DDL, por alterar a estrutura do banco.** * **Linguagem de Transação de Dados (DTL): usada para gerenciar as transações do banco de dados, fazendo com que as mudanças sejam permanentes ou desfeitas.**  1. Explique o conceito de normalização: motivos, formas normais (1ª, 2ª, 3ª, BCNF) e como aplicá-las.   **A normalização é um processo usado para analisar esquemas de tabelas (relações) com base em suas dependências funcionais e chaves primárias, com o objetivo de decompor tabelas grandes em tabelas menores e bem estruturadas, que quando reunidas, formam os dados originais sem perdas.**  **O principal motivo é evitar a redundância dos dados e a distorção das informações, acontecendo devido a anomalias:**   * **Anomalia de inserção: ocorre quando não é possível adicionar um dado sem que outro dado também esteja presente.** * **Anomalia de Exclusão: ocorre quando a exclusão de um registro acarreta na perda não intencional de outras informações.** * **Anomalia de Modificação: ocorre quando a alteração de um dado exige a atualização de múltiplas linhas.**   **Algumas formas normais são:**   * **Primeira Forma Normal: uma tabela está na primeira forma se todos os seus atributos forem atômicos. Isso significa que nenhum atributo pode conter múltiplos valores ou um conjunto de valores. Por exemplo, se uma tabela Funcionário tem um atributo Dependentes que armazena uma lista de nomes, violando a norma. Para corrigir cria-se uma tabela Dependentes e a relacionamos com Funcionário.** * **Segunda Forma Normal: uma tabela está na segunda forma se ela estiver na primeira forma e todo atributo não chave é totalmente dependente da chave primária, proibindo dependência parcial. Por exemplo, se uma tabela Inscrição tem a chave primária CPF\_Aluno, ID\_Curso e um atributo Nome\_Aluno, violando a norma. Para corrigir cria-se uma tabela Aluno com CPF\_Aluno e Nome\_Aluno e manter a Inscrição com a chave primária.** * **Terceira Forma Normal: uma tabela está na terceira forma se ela estiver na segunda forma e nenhum atributo não chave é transitivamente dependente da chave primária. Por exemplo, se uma tabela Funcionário com CPF, Nome, ID\_Departamento e Nome\_Departamento, temos que para conseguir o ID\_Departamento dependemos do CPF e para conseguir o Nome\_Departamento dependemos de ID\_Departamento, assim Nome\_Departamento depende transitivamente de CPF. Para corrigir cria-se uma tabela Departamento com ID\_Departamento e Nome\_Departamento e Funcionário com CPF, Nome e ID\_Departamento.** * **Forma Normal de Boyce-Codd (BCNF): uma tabela esta na BCNF se ela estiver na terceira forma e se, para cada dependência funcional (Ex: para descobrir o Nome se usa o CPF) que existir, o atributo que faz a dependência for uma superchave. Por exemplo, se uma tabela Funcionário tem Nome, Data\_Nascimento e Idade, para conseguir um dos atributos é difícil, já que podem ter nomes iguais registrados. Para corrigir cria-se o atributo CPF, o qual é único (superchave) para cada Funcionário.**   **Para normalizar um banco de dados é necessário:**   1. **Identificar as dependências funcionais.** 2. **Verificar se a tabela viola a primeira forma. Se sim, decompor.** 3. **Verificar se a tabela viola a segunda forma. Se sim, decompor.** 4. **Verificar se a tabela viola a terceira forma. Se sim, decompor.** 5. **Verificar se a tabela viola a BCNF. Se sim, decompor.** 6. Defina subclasses, superclasses e herança em modelos relacionais/OR-mapeamento.  * **Supercalsse: é um tipo de entidade que inclui um ou mais subgrupos distintos que são significativos para o modelo e precisam ser representados. A superclasse contem todos os atributos e relacionamentos comuns as subclasses.** * **Subclasse: é um subconjunto de entidades de uma superclasse aos outros membros da supercalsse.** * **Herança: é o mecanismo que conecta a subclasse à superclasses, com a subclasse herdando todos os atributos da superclasse.**   **Para simular esses conceitos usando tabelas e chaves existem algumas opções:**   * **Criar uma tabela que representa as superclasse e tabelas separadas para cada subclasse. Na tabela da subclasse, a chave primária da tabela da superclasse é usada como chave estrangeira, conectando-as.** * **Criar tabelas para as subclasses com todos os atributos que seriam herdados nelas, sem uma tabela para a superclasse.** * **Criar uma tabela para a superclasse e nela colocar atributos que vão identificar a qual subclasse elas pertencem.**  1. Explique as alternativas de implementação “tabela por hierarquia”, “tabela por classe concreta” e “tabela por subtipo”. Cite vantagens e desvantagens de cada abordagem.   Tabela por hierarquia:  Toda a hierarquia de classes (superclasse e todas as subclasses) é mapeada para uma **única tabela** no banco de dados. Essa tabela contém colunas para *todos* os atributos (comuns da superclasse e específicos de todas as subclasses). Uma coluna especial, chamada **discriminadora**, é usada para identificar qual classe aquela linha representa (ex: 'ENGENHEIRO' ou 'SECRETARIA').  Vantagens:  **Desempenho em Consultas Polimórficas:** É a abordagem mais rápida para buscar *todos* os tipos de funcionários de uma vez. Uma consulta SELECT \* FROM FUNCIONARIO traz todo mundo, sem a necessidade de JOINs ou UNIONs.  **Simplicidade:** O esquema é simples de entender à primeira vista (é apenas uma tabela).   1. Explique especialização e generalização. Quando usar cada uma no ciclo de projeto de banco de dados? 2. Fundamentos de consultas JOIN: explique os conceitos de cada tipo de consulta, INNER JOIN, LEFT JOIN e RIGHT JOIN. Explique o que cada um retorna e quando usar. |
|  |

**QUESTÕES PRÁTICAS**

|  |
| --- |
| 1. Liste todos os pacientes (nome e data\_cadastro)   Anexar print do comando + resultado   1. Liste todos os médicos (nome, crm, especialidade)   Anexar print do comando + resultado   1. Criar nova tabela endereco e migrar os dados:   Passos a executar (cada passo com comando e print):   * Criar tabela endereco com colunas atômicas (id, logradouro, numero, complemento, bairro, cidade, uf, cep). * Popular endereco com os dados existentes (faça uma transformação simplificada se o formato atual for Rua X, 100). * Alterar pessoa removendo colunas de endereço e adicionando endereco\_id FK. * Atualizar pessoa.endereco\_id para referenciar [endereco.id](http://endereco.id). * Mostrar SELECT p.nome, e.logradouro, e.numero FROM pessoa p JOIN endereco e ON p.endereco\_id = e.id; com print.  1. LEFT JOIN vs RIGHT JOIN: escreva e execute duas queries que relacionem paciente e consulta com LEFT JOIN e RIGHT JOIN (na sequência); capture os resultados e explique a diferença entre eles, indicando quais registros aparecem em cada resultado e por quê. 2. INNER JOIN vs RIGHT JOIN (igualdade de resultados): escreva e execute queries INNER JOIN e RIGHT JOIN para medico + consulta e paciente + consulta, explique em quais circunstâncias os resultados serão idênticos e por quê.   Anexar print dos comandos + resultados |

**Boa Atividade!!!**