Matéria: Tecnologia da Informação

Assunto: Banco de Dados e Programação

Resumo Teórico do Assunto

Olá! Para dominar as questões de Banco de Dados e Programação, é fundamental compreender os conceitos por trás das operações e estruturas de dados. Vamos explorar a teoria necessária para abordar as questões apresentadas.

---

1. Banco de Dados Relacionais (SQL)

As questões abordam operações de consulta e garantia de integridade de dados em bancos de dados relacionais, utilizando a linguagem **SQL (Structured Query Language)**.

 # 1.1. Operações de JOIN

As cláusulas `JOIN` são usadas para combinar linhas de duas ou mais tabelas com base em uma coluna relacionada entre elas.

• **`INNER JOIN` (ou simplesmente `JOIN`)**:
* **Definição**: Retorna apenas as linhas que possuem **correspondência** em *ambas* as tabelas, com base na condição de junção especificada. Linhas sem correspondência em qualquer uma das tabelas são **excluídas** do resultado.
* **Comportamento**: Se a Tabela A tem 100 linhas e a Tabela B tem 80, e apenas 70 linhas têm valores correspondentes na coluna de junção em ambas, um `INNER JOIN` resultará em 70 linhas.
* **Sintaxe Comum**: `SELECT * FROM T1 JOIN T2 ON T1.coluna = T2.coluna;`
* **`USING (coluna)`**: É um atalho para `ON T1.coluna = T2.coluna` quando a coluna de junção tem o **mesmo nome** em ambas as tabelas. Ex: `SELECT * FROM T1 JOIN T2 USING (CHAVE);`

• **`LEFT JOIN` (ou `LEFT OUTER JOIN`)**:
* **Definição**: Retorna **todas as linhas** da tabela da **esquerda** (a primeira tabela mencionada no `FROM` ou antes do `LEFT JOIN`), e as linhas correspondentes da tabela da direita. Se não houver correspondência na tabela da direita para uma linha da tabela da esquerda, os campos da tabela da direita para essa linha serão preenchidos com **NULL**.
* **Comportamento**: Se a Tabela A (esquerda) tem 100 linhas e a Tabela B (direita) tem 80, e apenas 70 linhas têm correspondência, um `LEFT JOIN` resultará em 100 linhas. As 30 linhas da Tabela A que não tiveram correspondência na Tabela B aparecerão com valores `NULL` para as colunas de B.

* **Diferença em relação ao `INNER JOIN`**: O `LEFT JOIN` sempre retornará um número de linhas **igual ou maior** que um `INNER JOIN` nas mesmas tabelas, a menos que a tabela da esquerda seja vazia ou todas as linhas da tabela da esquerda tenham correspondência na tabela da direita. A diferença no número de linhas entre um `LEFT JOIN` e um `INNER JOIN` indica a presença de linhas na tabela da esquerda que **não possuem correspondência** na tabela da direita.

 # 1.2. Restrições de Integridade (Constraints)

As **restrições (constraints)** são regras aplicadas às colunas de uma tabela para limitar o tipo de dados que podem ser inseridos nela, garantindo a **integridade e consistência** dos dados.

• **`CHECK` Constraint**:
* **Definição**: Permite definir uma **condição lógica** que deve ser verdadeira para cada linha da tabela. Se a condição for violada durante uma operação de `INSERT` ou `UPDATE`, a operação é rejeitada.
* **Uso**: Ideal para impor regras de negócio complexas ou validações entre colunas.
* **Exemplo**: `ALTER TABLE MinhaTabela ADD CONSTRAINT CK_Datas CHECK (data_de_emissao_RG > data_de_nascimento);` Esta restrição garante que o valor da coluna `data_de_emissao_RG` seja sempre posterior ao valor da coluna `data_de_nascimento`.

• **Outras Restrições Comuns (para contexto)**:
* **`PRIMARY KEY`**: Identifica unicamente cada registro em uma tabela (não nula e única).
* **`FOREIGN KEY`**: Garante a integridade referencial entre tabelas (referencia uma `PRIMARY KEY` em outra tabela).
* **`UNIQUE`**: Garante que todos os valores em uma coluna sejam diferentes.
* **`NOT NULL`**: Garante que uma coluna não possa ter valores `NULL`.

---

2. Programação com Python (Pandas)

A questão aborda a manipulação de dados usando a biblioteca **Pandas** em Python, amplamente utilizada para análise de dados.

• **`DataFrame` Pandas**:
* **Definição**: É a estrutura de dados principal do Pandas, representando uma tabela bidimensional com linhas e colunas rotuladas. É similar a uma planilha ou uma tabela de banco de dados.
* **Uso**: Armazena e permite a manipulação eficiente de grandes conjuntos de dados.

• **Seleção de Colunas em um DataFrame**:
* Para selecionar **uma única coluna**, você usa colchetes simples com o nome da coluna

como string: `dp['nome_da_coluna']`. O resultado é uma **Series** (estrutura de dados unidimensional do Pandas).
* Para selecionar **múltiplas colunas**, você deve passar uma **lista de nomes de colunas** dentro de colchetes. Isso significa que você usará **colchetes duplos**: `dp[['coluna1', 'coluna2', 'coluna3']]`. O resultado é um novo **DataFrame** contendo apenas as colunas selecionadas.
* **Exemplo**: `dp[['pais', 'ano', 'renda per capita', 'expectativa de vida']]` seleciona as quatro colunas especificadas e retorna um novo DataFrame.

---

3. Programação com TypeScript

A questão explora conceitos avançados de **TypeScript**, um superset de JavaScript que adiciona tipagem estática.

• **TypeScript (TS)**:
* **Definição**: É uma linguagem de programação que compila para JavaScript. Seu principal benefício é adicionar **tipagem estática** ao JavaScript, o que ajuda a detectar erros em tempo de desenvolvimento, melhorar a legibilidade do código e facilitar a manutenção.

• **Generics (Tipos Genéricos)**:
* **Definição**: Permitem escrever componentes (funções, classes, interfaces) que podem trabalhar com uma variedade de tipos, em vez de um único tipo específico. Isso promove a **reutilização de código** e a **segurança de tipo**.
* **Sintaxe**: Geralmente representados por uma letra maiúscula entre `< >`, como `` (de "Type"), `` (de "Key"), `` (de "Value"), etc.

• **Type Constraints (Restrições de Tipo)**:
* **Definição**: Permitem restringir os tipos que podem ser usados com um tipo genérico. Você usa a palavra-chave `extends` para indicar que o tipo genérico deve ser compatível com (ou "estender") um tipo específico.
* **Sintaxe**: ``. Isso significa que o tipo `T` deve ter, no mínimo, todas as propriedades e métodos de `SomeType`.
* **Exemplo**: `const a = (obj: T) => { ... };`
* Aqui, `T` é um tipo genérico.
* `extends {b: string}` é a **restrição de tipo**. Isso significa que qualquer tipo `T` que for passado para a função `a` **deve ser um objeto que possua, obrigatoriamente, uma propriedade `b` do tipo `string`**. O objeto `T` pode ter outras propriedades, mas `b: string` é um requisito mínimo.
* `obj: T` indica que o parâmetro `obj` da função será do tipo `T`.
* `=> { ... }` é a sintaxe de uma **função de seta (arrow function)** em JavaScript/TypeScript.

• **Funções em TypeScript**:
* Podem receber parâmetros com tipos definidos e, opcionalmente, ter um tipo de retorno

definido.
* No exemplo, a função `a` recebe um objeto `obj` cujo tipo `T` é restrito a ter a propriedade `b` como string. O que a função retorna dependeria do ``.

Compreender esses conceitos fundamentais de SQL (JOINs e Constraints), manipulação de DataFrames com Pandas em Python, e tipagem avançada com Generics e Type Constraints em TypeScript, permitirá que você analise e resolva as questões de forma eficaz.

Questões de Provas Anteriores

Fonte: escriturario_agente_de_tecnologia (1).pdf, Página: 13

pcimarkpci MjgwNDowMTRkOjE0YTU6OTI1ODozOGQ2OjNhMGM6NTM0MzplZmI1:U3VuLCAyNyBKdWwgMjAyNSAyMzo0Nzo0MCAtMDMwMA==

13
BANCO DO BRASIL
AGENTE DE TECNOLOGIA - Microrregião 16 DF-TIGABARITO 1
38
Para gerar um gráfico de dispersão, um programador precisava consultar duas tabelas, T1 e T2. Ele decidiu, então, usar um LEFT JOIN, como em
SELECT * FROM T1 LEFT JOIN T2 USING (CHAVE);
Essa consulta resultou em 214 linhas.
Por motivos de segurança, ele fez outra consulta semelhante, apenas trocando o LEFT JOIN por um JOIN, e essa segunda consulta resultou em 190 linhas.
O que pode explicar corretamente a quantidade diferente de linhas nas consultas realizadas?

(A) CHAVE é a chave primária de T1, mas apenas um campo da chave primária de T2.
(B) CHAVE é a chave primária de T2, mas apenas um campo da chave primária de T1.
(C) T1 possui linhas cujo valor de CHAVE não está presente na T2.
(D) T2 possui linhas cujo valor de CHAVE não está presente na T1.
(E) T2 possui linhas com todas as chaves presentes em T1, mas com campos nulos.
39
Ao analisar um conjunto de dados com Python, um programador resolveu usar um dataframe Pandas de nome dp para guardá-los. Em um certo momento, ele resolveu que precisaria usar, apenas, quatro colunas de dados do dataframe:

“pais“, “ano“, “renda per capita“ e “expectativa de vida“.
Que fragmento de código Python 3 deve ser usado para selecionar, apenas, essas quatro colunas do dataframe dp?
(A) dp[“pais“,“ano“,“expectativa de vida“,“renda per capita“]
(B) dp[[“pais“,“ano“,“expectativa de vida“,“renda per capita“]]
(C) dp(“pais“,“ano“,“expectativa de vida“,“renda per capita“)
(D) dp([“pais“,“ano“,“expectativa de vida“,“renda per capita“])
(E) dp[dp[“pais“,“
ano“,“expectativa de vida“,“renda per capita“]]
40
Ao coletar dados em um sistema compatível com SQL 2008 para fazer uma análise de dados, um programador percebeu
que havia dois campos, data_de_nascimento e data_de_emissão_RG, em que o valor de data_de_emissão_RG sempre
deve ser mais recente que data_de_nascimento. Percebeu, porém, que em 10% das linhas acontecia o inverso, isto é,
data_de_nascimento era mais recente que data_de_emissão_RG. Ele corrigiu os dados nessas linhas, verificando que es-
tavam consistentemente trocados, mas, preocupado que tal problema voltasse a acontecer, resolveu solicitar ao DBA uma
alteração da tabela, de forma que data_de_emissão_RG sempre tivesse que ser mais recente que data_de_nascimento.
O DBA atendeu adequadamente a esse pedido do programador por meio de uma restrição em SQL 2008 do tipo
(A) CHECK
(B) INSPECT
(C) TEST
(D) VALIDATE
(E) VERIFY
41
Considere o fragmento de código TypeScript a seguir.
const a = <T extends {b: string}>(obj: T) => {
<código removido>
};
Com relação ao código apresentado acima, a(o)
(A) função a() retorna um objeto do tipo string.
(B) variável a é uma lista de objetos do tipo string.
(C) variável a é um dicionário cujas chaves são objetos do tipo string.
(D) objeto que for passado para a função a() deve ter um campo b do tipo string.
(E) valor retornado pela função a() é um objeto que estende um objeto do tipo string.