Matéria: Tecnologia da Informação

Assunto: Linguagens de Programação e Banco de Dados

Resumo Teórico do Assunto

Com base nas questões fornecidas, o conhecimento essencial para o tópico "Linguagens de Programação e Banco de Dados" pode ser organizado da seguinte forma:

Resumo Teórico: Linguagens de Programação e Banco de Dados

Este resumo aborda conceitos fundamentais de linguagens de programação modernas e sistemas de gerenciamento de banco de dados, focando em aspectos de interoperabilidade, manipulação de dados e otimização.

 # I. Linguagens de Programação

1. **Kotlin:**
* **Definição:** É uma linguagem de programação moderna, concisa e pragmática, que se tornou a linguagem preferencial para o desenvolvimento de aplicativos **Android**.
* **Interoperabilidade com Java:** Uma de suas características mais importantes é a **compatibilidade total com Java**. Isso significa que:
* Código Kotlin pode chamar e utilizar bibliotecas e funções escritas em Java.
* Código Java pode chamar e utilizar funções e classes escritas em Kotlin.
* É possível construir aplicações com código **parcialmente em Java e parcialmente em Kotlin**, permitindo a coexistência e migração gradual de projetos.

2. **Python e a Biblioteca NumPy:**
* **Python:** Uma linguagem de programação de alto nível, interpretada, de propósito geral, conhecida por sua sintaxe clara e legibilidade. É amplamente utilizada em desenvolvimento web, ciência de dados, inteligência artificial e automação.
* **NumPy (Numerical Python):** É uma biblioteca fundamental para computação científica em Python.
* **Arrays N-dimensionais (`ndarray`):** O principal objeto do NumPy é o `ndarray`, que permite armazenar e manipular grandes coleções de dados numéricos de forma eficiente.
* **Operações com Arrays:**
* `np.array()`: Função para criar um array NumPy a partir de uma lista ou tupla.
* `.transpose()`: Método que retorna a transposta de um array (linhas viram colunas e vice-versa).

* `.dot()`: Função ou método que realiza o **produto escalar** (para vetores) ou **multiplicação de matrizes** (para arrays 2D ou superiores).
* `sum()`: A função `sum()` em Python, quando aplicada a um array NumPy, pode somar todos os elementos ou, se for um array de arrays (como uma matriz), pode somar os elementos de cada sub-array (linha), dependendo do contexto e da dimensão.

 # II. Bancos de Dados

1. **Tipos de Bancos de Dados:**
* **Banco de Dados Relacional (SQL):**
* **Estrutura:** Organiza os dados em **tabelas** (também chamadas de relações), que consistem em linhas (registros) e colunas (atributos).
* **Esquema:** Possui um **esquema de dados rígido e predefinido**. Isso significa que a estrutura das tabelas (nomes das colunas, tipos de dados) deve ser definida antes da inserção dos dados.
* **Linguagem:** Utiliza **SQL (Structured Query Language)** para definir, manipular e consultar os dados.
* **Integridade:** Garante a **integridade dos dados** através de chaves (como **Chave Primária - PK** para identificar unicamente um registro, e **Chave Estrangeira - FK** para estabelecer relacionamentos entre tabelas) e restrições.
* **Uso:** Ideal para dados **bem definidos e estruturados** onde a consistência e a integridade transacional são cruciais (ex: dados financeiros, cadastros de clientes).

* **Banco de Dados NoSQL (Not Only SQL):**
* **Estrutura:** Diferente dos relacionais, não utiliza o modelo de tabelas. Existem vários tipos (documento, chave-valor, grafo, coluna larga), cada um com sua forma de organização.
* **Esquema:** Caracteriza-se por ter um **esquema flexível** ou **sem esquema (schemaless)**. Isso permite armazenar dados com estruturas variáveis, sem a necessidade de definir previamente todas as colunas.
* **Uso:** Adequado para dados **não estruturados ou semi-estruturados** (ex: posts de redes sociais, logs, dados de sensores, documentos JSON), onde a flexibilidade e a escalabilidade horizontal são mais importantes que a rigidez do esquema.
* **Vantagem:** Facilita a adição de novos tipos de dados ou atributos sem a necessidade de alterar o esquema de todo o banco de dados, o que é complexo em bancos relacionais.

2. **Linguagem SQL (Structured Query Language):**
* **Propósito:** Linguagem padrão para gerenciar e manipular dados em bancos de dados relacionais.
* **Comandos Essenciais para Consulta (DML - Data Manipulation Language):**
* **`SELECT`:** Usado para recuperar dados de uma ou mais tabelas.
* `SELECT *`: Retorna todas as colunas.
* `SELECT coluna1, coluna2`: Retorna colunas específicas.

* **`FROM`:** Especifica a(s) tabela(s) de onde os dados serão recuperados.
* **`WHERE`:** Filtra as linhas retornadas com base em uma ou mais condições.
* **`JOIN`:** Combina linhas de duas ou mais tabelas com base em uma coluna relacionada entre elas (geralmente uma chave estrangeira). O tipo mais comum é o `INNER JOIN` (ou apenas `JOIN`), que retorna apenas as linhas onde há correspondência em ambas as tabelas.
* **Subconsultas (Subqueries):** Uma consulta `SELECT` aninhada dentro de outra consulta SQL.
* **`NOT IN`:** Operador usado na cláusula `WHERE` para filtrar registros cujos valores de uma coluna *não* estão presentes no conjunto de resultados de uma subconsulta. É fundamental para encontrar registros que *não* possuem uma determinada característica ou relacionamento.

3. **Otimização e Recursos de Banco de Dados:**
* **Desempenho:** A velocidade (baixo tempo de resposta) das consultas é crucial para a experiência do usuário e a eficiência do sistema.
* **Recursos para Melhorar o Desempenho:**
* **Índices:** São estruturas de dados especiais que melhoram significativamente a velocidade de recuperação de dados em tabelas grandes. Funcionam como um índice de livro, permitindo que o banco de dados localize rapidamente as linhas desejadas sem ter que escanear a tabela inteira. A criação de índices é uma das principais estratégias para otimizar consultas `SELECT`.
* **Regras de Integridade:** Garantem a validade e consistência dos dados (ex: unicidade de chaves primárias, referencial de chaves estrangeiras). Embora não sejam diretamente para desempenho de consulta, são vitais para a qualidade dos dados.
* **Visões (Views):** São tabelas virtuais baseadas no resultado de uma consulta SQL. Simplificam consultas complexas e podem restringir o acesso a dados específicos. Visões "não materializadas" não armazenam dados fisicamente, sendo apenas a definição da consulta.
* **Sequências:** Objetos de banco de dados que geram números únicos sequenciais, frequentemente usados para preencher chaves primárias automaticamente.
* **Gatilhos (Triggers):** Procedimentos armazenados que são executados automaticamente em resposta a eventos específicos (como `INSERT`, `UPDATE` ou `DELETE`) em uma tabela. Usados para impor regras de negócio complexas ou para auditoria.

Questões de Provas Anteriores

Fonte: escriturario_agente_de_tecnologia.pdf, Página: 15

pcimarkpci MjgwNDowMTRkOjE0YTU6OTI1ODozOGQ2OjNhMGM6NTM0MzplZmI1:U3V uLCAyNyBKdWwgMjAyNSAyMzo0NzozMSAtMDMwMA==
www.pciconcursos.com.br
AGENTE DE TECNOLOGIA - Microrregião 158 -TI
15
GABARITO 1
BANCO DO BRASIL
50
Kotlin é uma linguagem de programação usada no desenvolvimento Android.
Entre suas características, está um grau de compatibilidade com Java, que permite

(A) chamar funções feitas em Java, apenas, mas não permite que suas funções Kotlin sejam chamadas por Java.
(B) ler dados que foram salvos por apps Java, apenas.
(C) ler e escrever dados que podem ser lidos e escritos por apps Java, apenas.
(D) ter suas funções chamadas por Java, apenas, mas não consegue chamar funções feitas em Java.
(E) construir apps com código parcialmente em Java e parcialmente em Kotlin, sem restrições.

51

A seguir, é apresentado um fragmento de código em Python.

```
import numpy as np
b = np.array([[1,2,3,5]])
c = b.transpose()
print(b.dot(c),sum(b),sum(c))
```

O fragmento de código acima provoca a seguinte saída:
(A) 39 [11] [11]
(B) 39 [1 2 3 5] 11
(C) [39] [1 2 3 5] 11
(D) [[39]] [11] [11]
(E) [[39]] [1 2 3 5] [11]

52

Considere uma empresa que possui dados de clientes, todos bem definidos e estruturados (ex: CPF, nome, e-mail, endereço), armazenados em um banco de dados relacional. Uma oportunidade surge para a empresa enriquecer esse banco de dados com dados de outra natureza, porém não muito bem definidos e pouco estruturados.

Uma solução pode ser adotar um banco de dados NoSQL, de tal forma que:
(A) a ausência de um esquema de dados bem definido para os dados necessários de um cliente possa ser corretamente modelada e implementada em um ge - renciador de banco de dados adequado.
(B) a linguagem SQL utilizada para acesso aos dados dos clientes possa ser substituída por outra linguagem de acesso a dados organizados em tabelas segundo o modelo relacional, porém com maior eficiência.
(C) esse novo banco de dados relacional possa ser me - lhorado, com os dados não muito bem definidos, sem um esquema rígido.
(D) o gerenciador de banco de dados relacional utilizado possa ser atualizado para uma versão mais recente, que não utilize a linguagem SQL.
(E) os atributos que hoje representam chaves primárias e estrangeiras sejam mais bem controlados.
53
Uma empresa de investimentos financeiros busca iden - tificar novas oportunidades de negócio para pessoas jurídicas, em especial dentre aquelas que têm como ca - racterística a adoção de governança ambiental, social e corporativa (conhecida como ESG, uma sigla em inglês).
Considere que existe um banco de dados nessa empresa com as seguintes tabelas (todas as chaves primárias são numéricas):
Empresa (CNPJ, razaoSocial, endereco)
Caracteristica (cod, sigla, nome)
Tem (CNPJ, cod)
Que comando SELECT do SQL retorna apenas o CNPJ e a razão social das empresas que não têm "ESG" como característica?
(A) SELECT *
FROM Empresa
WHERE Caracteristica.Sigla <> 'ESG'
(B) SELECT E.CNPJ, E.razaoSocial
FROM Empresa E
JOIN Tem T ON (E.CNPJ = T.CNPJ)
WHERE Tem.cod = 'ESG'
(C) SELECT E.CNPJ, E.razaoSocial
FROM Empresa E
JOIN Tem T ON (E.CNPJ = T.CNPJ)
JOIN Caracteristica C ON (C.cod = T.cod)

WHERE C.nome <> 'ESG'
(D) SELECT E.CNPJ, E.razaoSocial
FROM Empresa E
WHERE E.CNPJ NOT IN (
SELECT T.CNPJ
FROM Tem T
JOIN Caracteristica C ON (T.cod = C.cod)
WHERE C.sigla = 'ESG'
)
(E) SELECT Empresa.*
FROM Empresa, Tem
WHERE Empresa.CNPJ = Tem.cod
AND Tem.cod <> 'ESG'
54
Um banco de dados (BD) persiste dados de forma organizada e controlada. Em adição, um BD deve prover recursos para permitir que consultas que necessitem de velocidade (baixo tempo de resposta) no acesso aos dados possam ter um bom desempenho.
Um dos recursos que um profissional de tecnologia da informação tem à disposição para configurar um BD, de modo a melhorar o desempenho de consultas selecionadas, é a criação de
(A) regras de integridade
(B) visões não materializadas
(C) índices
(D) sequências
(E) gatilhos