Matéria: Tecnologia da Informação

Assunto: Data Warehouse e Programação Java

Resumo Teórico do Assunto

Para resolver as questões apresentadas, é fundamental compreender os conceitos de **Data Warehouse** e **Modelagem Dimensional**, bem como os princípios de **Programação Orientada a Objetos (POO)** em Java, especialmente **Herança**, **Polimorfismo** e a distinção entre métodos **estáticos** e de **instância**.

I. Data Warehouse e Modelagem Dimensional

Um **Data Warehouse (DW)** é um sistema de armazenamento de dados otimizado para consultas e análises, diferente dos bancos de dados transacionais (OLTP). Seu principal objetivo é suportar a **tomada de decisões** e a **inteligência de negócios**, consolidando dados de diversas fontes.

1. **Modelagem Dimensional:**
* A abordagem mais comum para modelar um DW é a **modelagem dimensional**, que organiza os dados em **fatos** e **dimensões**. Isso facilita a compreensão e otimiza o desempenho das consultas analíticas.
* O **modelo estrela (Star Schema)** é o esquema mais simples e amplamente utilizado na modelagem dimensional. Ele consiste em uma **tabela fato** central cercada por várias **tabelas dimensão**.

2. **Tabela Fato (Fact Table):**
* A **tabela fato** armazena as **medidas** (valores numéricos e quantificáveis) que são o foco da análise, como "valor_emprestimo" e "prazo_emprestimo".
* Além das medidas, a tabela fato contém apenas as **chaves estrangeiras (Foreign Keys - FKs)** que se referem às chaves primárias das tabelas dimensão associadas.
* **Regra Essencial:** A tabela fato *não* armazena atributos descritivos das dimensões. Esses atributos pertencem exclusivamente às tabelas dimensão.
* **Granularidade:** A **granularidade** de uma tabela fato define o nível de detalhe dos dados armazenados. "Granularidade mais baixa" significa o nível mais atômico e detalhado possível, ou seja, cada linha da tabela fato representa um evento ou transação individual no seu nível mais fundamental.

3. **Tabela Dimensão (Dimension Table):**
* As **tabelas dimensão** fornecem o **contexto** para as medidas da tabela fato.
* Elas contêm os **atributos descritivos** que qualificam, categorizam e descrevem os dados,

como "dia", "mês", "ano" (para a dimensão tempo) ou "estado", "cidade" (para a dimensão agência).
* Cada tabela dimensão possui uma **chave primária (Primary Key - PK)** que é referenciada como FK na tabela fato.

Em resumo: Para construir uma tabela fato em um modelo estrela, você deve incluir as **medidas** que deseja analisar e as **chaves estrangeiras** que ligam essa tabela às suas respectivas tabelas dimensão. Os atributos descritivos das dimensões (como nome do cliente, nome do produto, etc.) *não* devem estar na tabela fato.

II. Programação Java (Conceitos de POO)

A questão de Java explora conceitos fundamentais de **Programação Orientada a Objetos (POO)**, especificamente **herança**, **polimorfismo** e a distinção crucial entre métodos **estáticos** e de **instância**.

1. **Classes e Objetos:**
* Em Java, **classes** são "plantas" ou modelos para criar **objetos**, que são instâncias dessas classes. Objetos encapsulam dados (atributos) e comportamentos (métodos).

2. **Herança (`extends`):**
* A **herança** permite que uma classe (**subclasse** ou classe filha) herde atributos e métodos de outra classe (**superclasse** ou classe pai). A palavra-chave `extends` é usada para indicar herança (ex: `Va2 extends Va1`).

3. **Polimorfismo:**
* O **polimorfismo** é a capacidade de um objeto assumir múltiplas formas. Em Java, isso é frequentemente visto quando uma referência de superclasse aponta para um objeto de subclasse (ex: `Va1 o = new Va2();`).
* O método que será executado é determinado pelo **tipo real do objeto** em tempo de execução para **métodos de instância**.

4. **Métodos Estáticos (`static`):**
* Um método **estático** pertence à classe, não a uma instância específica dela. Ele é invocado usando o nome da classe (ex: `Va1.getStr()`).
* **Ponto Crucial:** Métodos estáticos *não podem ser sobrescritos* no sentido polimórfico. Se uma subclasse define um método estático com a mesma assinatura, ela está "escondendo" ou "redefinindo" o método da superclasse, mas não o sobrescrevendo. A resolução de chamadas a métodos estáticos ocorre em **tempo de compilação** com base no **tipo da referência** (o tipo declarado da variável).

5. **Métodos de Instância:**
* Um método de instância pertence a um objeto específico. Ele é invocado usando uma

referência ao objeto (ex: `o.fin()`).
* Métodos de instância *podem ser sobrescritos* por subclasses. A resolução de chamadas a métodos de instância ocorre em **tempo de execução** com base no **tipo real do objeto** (polimorfismo).

6. **Sobrescrita de Método (Method Overriding):**
* Ocorre quando uma subclasse fornece uma implementação específica para um método que já é definido em sua superclasse, mantendo a mesma assinatura (nome, tipo de retorno e parâmetros). Isso é fundamental para o polimorfismo em métodos de instância.

7. **Manipulação de Strings:**
* A classe `String` em Java oferece métodos úteis como:
* `length()`: Retorna o comprimento da string.
* `substring(startIndex, endIndex)`: Retorna uma nova string que é uma parte da string original, começando em `startIndex` (inclusive) e terminando em `endIndex` (exclusivo). Lembre-se que os índices em Java são baseados em zero.

Em resumo: Ao analisar o código Java, preste atenção ao tipo da referência (`Va1 o`) e ao tipo real do objeto (`new Va2()`). Para métodos de instância, o polimorfismo garante que a versão da subclasse (se sobrescrita) seja chamada. Para métodos estáticos, a versão chamada é determinada pelo tipo da *referência*, não pelo tipo do objeto.

Questões de Provas Anteriores

Fonte: escriturario_agente_de_tecnologia (1).pdf, Página: 17

pcimarkpci MjgwNDowMTRkOjE0YTU6OTI1ODozOGQ2OjNhMGM6NTM0MzplZmI1:U3VuLCAyNyBKdWwgMjAyNSAyMzo0Nzo0MCAtMDMwMA==

BANCO DO BRASIL
AGENTE DE TECNOLOGIA - Microrregião 16 DF-TIGABARITO 1
49
Ao desenvolver um Data Warehouse para o Banco W, um programador decidiu criar um modelo conceitual com base no
modelo estrela para cada fato analisado. Ao criar a primeira tabela fato, relativa ao valor e ao prazo de empréstimos, foram
identificadas as seguintes dimensões, com os seus atributos descritos em parênteses: tempo (dia, mês e ano), agência
(estado, cidade, bairro e número da agência), produto (nome do produto e juros do produto) e cliente (conta e nome do
cliente).
Segundo as regras e as práticas da modelagem dimensional, e usando a granularidade

mais baixa, que atributos devem constar da tabela fato?

(A) fato_id, dia, mes, ano, estado, cidade, bairro, numero_agencia, nome_produto, juros_mensais_produto, conta_cliente, nome_cliente, valor_emprestimo, prazo_emprestimo
(B) fato_id, emprestimo_id, valor_emprestimo, prazo_emprestimo
(C) fato_id, tempo_id, agencia_id, produto_id, cliente_id, emprestimo_id
(D) fato_id, tempo_id, agencia_id, produto_id, cliente_id, dia, mes, ano, estado, cidade, bairro, numero_agencia, nome_produto, juros_mensais_produto, conta_cliente, nome_cliente, valor_emprestimo, prazo_emprestimo
(E) fato_id, tempo_id, agencia_id, produto_id, cliente_id, valor_emprestimo, prazo_emprestimo
50
Sejam as seguintes classes Java:

```
public class Va1 {
public static String getStr() {
return “abcdefghijklmnop“;
}

public String ini(String s, int cpr) {
return s.substring(0, cpr);
}

public String fin(String s, int cpr) {
return ini(s, cpr)+s.substring(s.length()-cpr, s.length());
}
}
public class Va2 extends Va1 {

public static String getStr() {
return “0123456789ABCDEF“;
}

public String ini(String s, int cpr) {
return s.substring(s.length()-cpr, s.length());
}
public static void main(String[] args) {
Va1 o=new Va2();

System.out.println(o.fin(o.getStr(), 5));
}
}
```

O que será exibido no console quando o método main for executado?
(A) 0123BCDE

(B) BCDEFBCDEF
(C) 01234BCDEF
(D) abcdelmnop
(E) lmnoplmnop