# Matéria: Tecnologia da Informação

## Assunto: Banco de Dados e SQL

---

### Resumo Teórico do Assunto

Para ter sucesso nas questões sobre Banco de Dados e SQL, é fundamental compreender os conceitos de **Linguagem SQL**, **Definição de Dados (DDL)**, **Tipos de Dados**, **Restrições (Constraints)** e **Objetos de Banco de Dados** como **Views** e **Triggers**, além de estratégias de sincronização de dados.

---

**I. Linguagem SQL (Structured Query Language)**

SQL é a linguagem padrão para gerenciar e manipular bancos de dados relacionais. Ela é dividida em subconjuntos, sendo os mais relevantes para estas questões:

• **DDL (Data Definition Language - Linguagem de Definição de Dados)**: Usada para definir, modificar ou remover a estrutura do banco de dados e seus objetos (tabelas, índices, views, etc.).
• **DML (Data Manipulation Language - Linguagem de Manipulação de Dados)**: Usada para inserir, atualizar, excluir e consultar dados nas tabelas.

**II. Data Definition Language (DDL) e o Comando `CREATE TABLE`**

O comando `CREATE TABLE` é um dos mais importantes da DDL, utilizado para criar novas tabelas em um banco de dados.

**Sintaxe Básica:**

```sql
CREATE TABLE NomeDaTabela (
NomeColuna1 TipoDeDados1 [Restrição1] [Restrição2],
NomeColuna2 TipoDeDados2 [Restrição3],
...
[RestriçãoDeTabela]
);
```

**Componentes Chave:**

1. `**NomeDaTabela**`: O nome que você dará à sua tabela.

2. **`NomeColuna`**: O nome de cada campo (coluna) da tabela.
3. **`TipoDeDados`**: Define o tipo de informação que a coluna pode armazenar.

* **`CHAR(n)`**: Armazena uma string de caracteres de **tamanho fixo** `n`. Se o valor inserido for menor que `n`, ele será preenchido com espaços em branco até atingir o tamanho `n`. É ideal para dados de tamanho constante, como códigos postais ou CNPJs (se o preenchimento com zeros à esquerda for importante e o tamanho for sempre o mesmo).
* **`VARCHAR(n)`**: Armazena uma string de caracteres de **tamanho variável**, com um comprimento máximo de `n`. Ocupa apenas o espaço necessário para o dado, sem preenchimento. É mais eficiente para textos de tamanhos variados, como nomes.
* **`INTEGER`**: Armazena números inteiros. Não é adequado para armazenar sequências de dígitos que precisam manter zeros à esquerda ou que não representam um valor numérico para cálculos (como um CNPJ, que é mais um identificador).

4. **Restrições (Constraints)**: Regras aplicadas às colunas ou à tabela para garantir a integridade e a validade dos dados.

* **`PRIMARY KEY` (Chave Primária)**:
* Identifica unicamente cada registro (linha) em uma tabela.
* Garante que todos os valores na coluna (ou conjunto de colunas) sejam **únicos** e **não nulos** (implicitamente `NOT NULL`).
* Uma tabela pode ter apenas uma chave primária.
* Pode ser definida junto à coluna (`NomeColuna TipoDeDados PRIMARY KEY`) ou no final da definição da tabela (`PRIMARY KEY (NomeColuna)`).

* **`NOT NULL`**:
* Impede que a coluna contenha valores nulos (ausência de valor).
* Se uma coluna é definida como `NOT NULL`, um valor deve ser fornecido para ela ao inserir um novo registro.

* **`NULL`**:
* Permite que a coluna contenha valores nulos.
* É o comportamento padrão se nenhuma restrição de nulidade for especificada.

**III. Objetos de Banco de Dados e Estratégias de Sincronização**

Bancos de dados utilizam diversos objetos para organizar e gerenciar dados e funcionalidades.

1. **`VIEW` (Visão)**:
* Uma **tabela virtual** baseada no resultado de uma consulta SQL.
* Não armazena dados fisicamente por si só; os dados são recuperados das tabelas subjacentes cada vez que a view é acessada.
* **Vantagens**: Simplifica consultas complexas, oferece segurança (expondo apenas um

subconjunto de dados ou colunas), e o mais importante para a questão: os dados em uma view estão **sempre atualizados** em tempo real, pois refletem o estado atual das tabelas base. É ideal para protótipos ou relatórios que precisam de dados em tempo real sem duplicá-los.

2. **`TRIGGER` (Gatilho)**:
* Um procedimento armazenado que é executado **automaticamente** em resposta a um evento específico em uma tabela (ou view).
* Os eventos comuns são `INSERT` (inserção de dados), `UPDATE` (atualização de dados) e `DELETE` (exclusão de dados).
* Podem ser definidos para serem executados `BEFORE` (antes) ou `AFTER` (depois) do evento.
* **Usos**: Auditoria, imposição de regras de negócio complexas, e **sincronização de dados** entre tabelas ou bancos de dados. No entanto, a implementação de triggers para sincronização pode introduzir **overhead de desempenho**, especialmente em sistemas de alta transação, pois cada operação na tabela original dispara uma ação adicional.

3. **Outras Estratégias de Sincronização/Gerenciamento (para contraste):**

* **Backup e Restauração (`DUMP`)**: Envolve a criação de uma cópia completa (ou parcial) do banco de dados em um ponto no tempo (`DUMP`) e sua posterior restauração. Não é uma solução em tempo real para sincronização e pode ter alto impacto de desempenho se feita frequentemente.
* **Particionamento de Tabelas**: Uma técnica para dividir uma tabela grande em partes menores e mais gerenciáveis, chamadas partições. Isso melhora o desempenho de consultas e a manutenção, mas ocorre **dentro de um único banco de dados** e não é uma estratégia para sincronizar dados entre bancos de dados separados.

Ao entender esses conceitos, você estará apto a analisar as opções das questões e identificar a solução mais adequada para cada cenário proposto.

---

## Questões de Provas Anteriores

**Fonte: escriturario_agente_de_tecnologia (1).pdf, Página: 15**

**pcimarkpci MjgwNDowMTRkOjE0YTU6OTI1ODozOGQ2OjNhMGM6NTM0MzplZmI1:U3VuLCAyNyBKdWwgMjAyNSAyMzo0Nzo0MCAtMDMwMA==**
**www.pciconcursos.com.br**
**15**
**BANCO DO BRASIL**
**AGENTE DE TECNOLOGIA - Microrregião 16 DF-TIGABARITO 1**
**45**
**Após um treinamento em SQL padrão 2008, compatível com ambiente MS SQL Server**

**2008, um escriturá-**
**rio do Banco Z precisou utilizar os conhecimentos adquiridos para criar uma tabela no sistema de banco de dados**
**desse Banco. A tabela a ser criada é de fornecedores, e tem os seguintes campos: CNPJ, nome do fornecedor e país de**
**origem. As características gerais da tabela são:**
**. o campo CNPJ é chave primária e contém 14 caracteres, sendo que os caracteres devem se ater aos numéricos**
**["0" a "9"], e o caractere zero "0" não pode ser ignorado, seja qual for a posição dele (início, meio ou fim da chave);**
**. o campo NOME contém 20 caracteres e aceita valor nulo;**
**. o campo PAIS contém 15 caracteres e não aceita valor nulo.**
**Nesse contexto, o comando SQL2008 que cria uma tabela com as características descritas acima é**

(A) CREATE TABLE Fornecedores
(CNPJ INTEGER PRIMARY KEY,
NOME VARCHAR(20) ACCEPT NULL,
PAIS VARCHAR(15) NOT NULL)
(B) CREATE TABLE Fornecedores
(CNPJ CHAR(14) PRIMARY KEY,
NOME VARCHAR(20),
PAIS VARCHAR(15) NOT NULL)
(C) CREATE TABLE Fornecedores
(CNPJ CHAR(14) NOT NULL,
NOME VARCHAR(20) NOT NULL,
PAIS VARCHAR(15))
(D) CREATE TABLE Fornecedores
(CNPJ CHAR(14),
NOME VARCHAR(20) NOT NULL,
PAIS VARCHAR(15) NOT NULL),
PRIMARY KEY (CNPJ)
(E) CREATE TABLE Fornecedores
(CNPJ INTEGER(14) NOT NULL,
NOME VARCHAR(20),
PAIS VARCHAR(15) NOT NULL),
PRIMARY KEY (CNPJ)
46
Para que fosse mais fácil entender um sistema em desenvolvimento, um desenvolvedor usou um modelo de dados rela-
cional (protótipo) mais simples do que o do banco de dados do sistema corporativo original, sendo que ambos utilizavam
o mesmo SGDB PostgreSQL. Cabe ressaltar que esse protótipo utilizava apenas um subconjunto dos dados do sistema

corporativo original e realizava apenas consultas.
Uma forma de garantir que os dados desse protótipo estejam sempre completamente atualizados em relação aos dados
reais, com baixo impacto tanto na operação quanto no desempenho do sistema corporativo original, é
(A) criar apenas VIEWS no protótipo, definidas com consultas sobre as tabelas do sistema corporativo original.
(B) implantar TRIGGERS a cada INSERT, em todas as tabelas do sistema corporativo original, atualizando as tabelas do
protótipo.
(C) implantar TRIGGERS de atualização a cada SELECT, em todas as tabelas do protótipo.
(D) particionar as tabelas da base do sistema corporativo original escolhendo um RANGE adequado ao trabalho do pro-
tótipo.
(E) utilizar DUMP da base do sistema corporativo original e PSQL para a base do protótipo, a cada seção de trabalho, para
atualizar a base do protótipo.