**ADMINISTRADOR DE BANCO DE DADOS**

**Resolução**

**LISTA DE EXERCÍCIOS SQL**

**Exercício 1:**

Explique o que é a linguagem SQL e quais são suas subdivisões. Dica: são 5 ao todo.

**RESPOSTA:**

A linguagem SQL é uma linguagem padrão de consulta comum aos SGBDs, com definições de estrutura em tabelas, a linguagem orquestra uma interação completa com o banco de dados, isto é, sua administração completa. Ela é uma linguagem não-procedural, e é composta por inúmeros comandos, os quais são agrupados em 5 subdivisões:

**DDL** - *Data Definition Language* - Linguagem de Definição de Dados.

São os comandos que interagem com os objetos do banco.

São comandos DDL : CREATE, ALTER e DROP

**DML** - *Data Manipulation Language* - Linguagem de Manipulação de Dados.

São os comandos que interagem com os dados dentro das tabelas.

São comandos DML : INSERT, DELETE e UPDATE

**DQL** - *Data Query Language* - Linguagem de Consulta de Dados.

São os comandos de consulta.

São comandos DQL : SELECT (é o comando de consulta)

Aqui cabe um parênteses. Em alguns livros o SELECT fica na DML e em outros tem esse grupo próprio.

**DTL** - *Data Transaction Language* - Linguagem de Transação de Dados.

São os comandos para controle de transação.

São comandos DTL : BEGIN TRANSACTION, COMMIT E ROLLBACK

**DCL** - *Data Control Language* - Linguagem de Controle de Dados.

São os comandos para controlar a parte de segurança do banco de dados.

São comandos DCL : GRANT, REVOKE e DENY.

**Exercício 2:**

O que são CONSTRAINTS? Cite pelo menos 3 exemplos e explique o que cada uma delas faz.

**RESPOSTA:**

As CONSTRAINTS são regras que são aplicadas a uma tabela em um banco de dados para garantir a integridade de dados. Elas podem ser usadas para limitar os tipos de dados que podem ser inseridos em uma tabela, para exigir que os campos tenham valores únicos ou para exigir que um campo tenha um valor obrigatório. As CONSTRAINTS são especificadas quando a tabela é criada, mas também podem ser adicionadas ou removidas posteriormente.

**Principais:**

**NOT NULL:** exige que um campo tenha um valor.

UNIQUE: exige que todos os valores em um campo sejam únicos.

**PRIMARY KEY:** exige que todos os valores em um campo sejam únicos e não nulos. Além disso, uma tabela só pode ter uma chave primária.

**FOREIGN KEY:** exige que os valores em um campo correspondam aos valores em uma chave primária em outra tabela.

**CHECK:** exige que os valores em um campo atendam a uma condição especificada.

**DEFAULT:** especifica um valor padrão para um campo, que será usado se nenhum valor for fornecido quando um novo registro for inserido.

**INDEX:** cria um índice para uma tabela, o que pode melhorar a performance das consultas que usam essa tabela.

**FULLTEXT:** cria um índice full-text para uma tabela, o que permite a pesquisa de palavras-chave em um campo de texto.

**SPATIAL**: cria um índice para armazenar informações espaciais, como coordenadas geográficas.

**Exercício 3:**

Na linguagem SQL, o comando CREATE TABLE é usado para criar uma tabela no banco de dados; já o relacionamento entre duas tabelas pode ser expresso através do comando:

**A)** null

**B)** primary key

**C)** constraint foreign key

**D)** auto\_increment

**E)** not null

**RESPOSTA:**

**FOREIGN KEY:** exige que os valores em um campo correspondam aos valores em uma chave primária em outra tabela. **(TABELA RELAÇÃO)**

**LETRA C**

**Exercício 4:**

O comando SELECT é utilizado para efetuar consultas no banco de dados. O resultado de uma consulta é chamado de Result Set. Sabendo disso, julgue as afirmações a seguir como Verdadeiro ou Falso:

I. O Result Set deve possuir pelo menos um registro.

II. Uma consulta só pode ser executada em uma única tabela; não é possível juntar várias tabelas para produzir uma consulta.

III. O comando SELECT \* FROM nome\_tabela retorna todos os campos da tabela selecionada.

IV. O comando utilizado para ordenar os registros de um Result Set a partir de um ou mais campos é o GROUP BY.

**Exercício 5:**

A cláusula WHERE é utilizada para filtrar resultados de uma consulta. Sabendo disso, cite o que fazem os seguintes operadores utilizados em conjunto com a cláusula WHERE:

**<>**; **<=**; **>=**; **<**; **>**; **=**; **IN**; **LIKE**; **BETWEEN**

**RESPOSTA:**

**Operadores aritméticos:**

**Adição:** SELECT 5 **+** 4;

**Subtração:** SELECT 5 **-** 4;

**Multiplicação:** SELECT 5 **\*** 4;

**Divisão:** SELECT 5 **/** 4;

**Operadores de comparação:**

**Igualdade:** SELECT \* FROM customers WHERE customer\_id **=** 1;

**Diferente:** SELECT \* FROM customers WHERE customer\_id **<>** 1;

**Maior que:** SELECT \* FROM customers WHERE customer\_age **>** 25;

**Menor que:** SELECT \* FROM customers WHERE customer\_age **<** 25;

**Maior ou igual a**: SELECT \* FROM customers WHERE customer\_age **>=** 25;

**Menor ou igual a**: SELECT \* FROM customers WHERE customer\_age **<=** 25;

**Operadores de conjunção:**

**AND:** SELECT \* FROM customers WHERE customer\_age < 25 AND customer\_gender = 'F';

**OR:** SELECT \* FROM customers WHERE customer\_age < 25 OR customer\_gender = 'F';

Operadores de expressão:

**IN:** permite especificar uma lista de valores que devem ser buscados em um campo específico. Por exemplo:

*SELECT \* FROM customers WHERE customer\_age IN (25, 30, 35);*

**LIKE:** permite especificar um padrão para buscar em um campo de texto. O comodim % é usado como curinga para corresponder a qualquer sequência de zero ou mais caracteres. Por exemplo:

*SELECT \* FROM customers WHERE customer\_name LIKE 'J%';*

**BETWEEN:** permite especificar um intervalo de valores para buscar em um campo. Por exemplo:

*SELECT \* FROM customers WHERE customer\_age BETWEEN 25 AND 35;*

**Exercício 6:**

Em uma tabela chamada Contribuinte de um banco de dados relacional há o campo idContribuinte do tipo inteiro e chave primária. Há também o campo nomeContribuinte que é do tipo varchar. Nessa tabela, um Auditor Fiscal deseja alterar o nome do contribuinte de id 1 para 'Marcos Silva'. Para isso, terá que utilizar o comando:

**A.** ALTER TABLE Contribuinte SET nomeContribuinte='Marcos Silva' WHERE idContribuinte=1;

**B**. UPDATE Contribuinte SET nomeContribuinte='Marcos Silva' WHERE idContribuinte=1;

**C.** UPDATE nomeContribuinte TO 'Marcos Silva' FROM Contribuinte WHERE idContribuinte=1;

**D.** ALTER TABLE Contribuinte FIELD nomeContribuinte='Marcos Silva' WHERE idContribuinte=1;

**E**. UPDATE TABLE Contribuinte FIELD nomeContribuinte='Marcos Silva' WHERE idContribuinte=1;

**RESPOSTA:** **LETRA B**

*A instrução UPDATE é usada para modificar os registros existentes em uma tabela (Atualiza os valores das linhas de uma tabela).*

*UPDATE table\_name (contribuinte)*

*SET column1 = value1, column2 = value2, … (valor desejado para ser alterado na tabela)*

*WHERE condition; (condição para inserir o valor)*

*A instrução ALTER TABLE é usada para Modificar um objeto dentro do banco de dados (tabela, view, etc). Tipos (varchar, inter etc)*

**Exercício 7:**

O que aconteceria no banco de dados caso o Auditor Fiscal do exercício anterior não utilizasse a cláusula WHERE no comando SQL?

**RESPOSTA:**

Se a instrução ao ser executada sem a condição WHERE alteraria todos os valores dos registos(linhas) da coluna (nomeContribuinte) para “Marcos Silva".

**Exercício 8:**

Considerando a tabela Aluno (id, curso, disciplina, notafinal) de um banco de dados relacional, escreva o comando SQL que permite obter o nome da disciplina e a média das notas finais por disciplina apenas para os alunos para as disciplinas com média abaixo de 5.

**REPOSTA:**

**SELECT** disciplina, **AVG**(notafinal)

**FROM** Aluno

**GROUP BY** disciplina

**HAVING** **AVG**(notafinal) **<** 5;

\*A cláusula HAVING é usada para especificar condições de filtragem em grupos de registros ou agregações. Uso na cláusula GROUP BY para filtrar as colunas agrupadas.

**Exercício 9:**

Filtros simples são aqueles que utilizam apenas uma condição. Filtros compostos são aqueles que utilizam mais de uma condição. Para criar um filtro composto são utilizados junto à cláusula WHERE os operadores lógicos. Liste quais são e explique as diferenças entre eles.

**RESPOSTA:**

**AND:** retorna verdadeiro se ambas as condições forem verdadeiras.

*Exemplo: SELECT \* FROM customers WHERE customer\_age < 25 AND customer\_gender = 'F';*

**OR:** retorna verdadeiro se pelo menos uma das condições for verdadeira.

Exemplo: *SELECT \* FROM customers WHERE customer\_age < 25 OR customer\_gender = 'F';*

**NOT:** inverte o resultado de uma condição.

Exemplo: *SELECT \* FROM customers WHERE NOT customer\_age < 25;*

**Exercício 10:**

Assinale, das alternativas abaixo, a única que identifica corretamente as funções de agregação da linguagem SQL.

**A)** AVG – COUNT – SUM – MAX – MIN

**B)** AVG – LIKE – UNION – MAX – MIN

**C)** AVG – COUNT – AND – MAX – LIKE

**D)** FROM – COUNT – SUM – MAX – MIN

R: **LETRA A -**

**Funções de agregação:**

**COUNT:** retorna o número de linhas em um conjunto de resultados.

**SUM:** retorna a soma de todos os valores em um campo numérico.

**AVG:** retorna a média de todos os valores em um campo numérico.

**MIN:** retorna o menor valor em um campo.

**MAX:** retorna o maior valor em um campo.