Estratégia Concursos

Aula 04 - Profs Diego Carvalho e Emanuele Gouveia

Banco do Brasil (Escriturário - Agente de Tecnologia) Banco de Dados - 2023 (Pós-Edital)

Autor:
Thiago Rodrigues Cavalcanti,
Erick Muzart Fonseca dos Santos,
Diego Carvalho

09 de Janeiro de 2023

Índice

#ATENÇÃO

Avisos Importantes

O curso abrange todos os níveis de conhecimento...

Esse curso foi desenvolvido para ser acessível a **alunos com diversos níveis de conhecimento diferentes.** Temos alunos mais avançados que têm conhecimento prévio ou têm facilidade com o assunto. Por outro lado, temos alunos iniciantes, que nunca tiveram contato com a matéria ou até mesmo que têm trauma dessa disciplina. A ideia aqui é tentar atingir ambos os públicos - iniciantes e avançados - da melhor maneira possível..

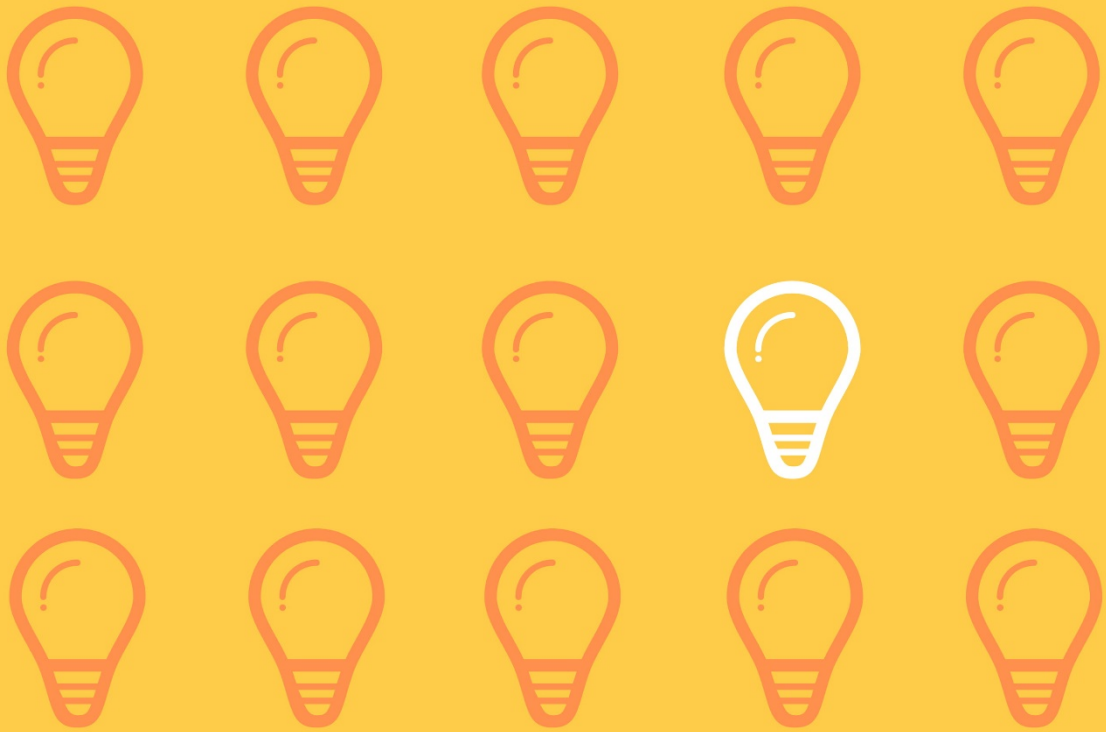
Por que estou enfatizando isso?

O **material completo** é composto de muitas histórias, exemplos, metáforas, piadas, memes, questões, desafios, esquemas, diagramas, imagens, entre outros. Já o **material simplificado** possui exatamente o mesmo núcleo do material completo, mas ele é menor e bem mais objetivo. *Professor, eu devo estudar por qual material?* Se você quiser se aprofundar nos assuntos ou tem dificuldade com a matéria, necessitando de um material mais passo-a-passo, utilize o material completo. Se você não quer se aprofundar nos assuntos ou tem facilidade com a matéria, necessitando de um material mais direto ao ponto, utilize o material simplificado.

Por fim...

O curso contém diversas questões espalhadas em meio à teoria. Essas questões possuem um comentário mais simplificado porque **têm o único objetivo de apresentar ao aluno como bancas de concurso cobram o assunto previamente administrado**. A imensa maioria das questões para que o aluno avalie seus conhecimentos sobre a matéria estão dispostas ao final da aula na lista de exercícios e **possuem comentários bem mais completos, abrangentes e direcionados.**

X



• ATENÇÃO •

Existem muitos exercícios sobre esse tema em sites de questões, no entanto a imensa maioria foi aplicada em provas para cargos específicos de Tecnologia da Informação (TI), os quais podem demandar um conhecimento muito mais aprofundado da matéria.

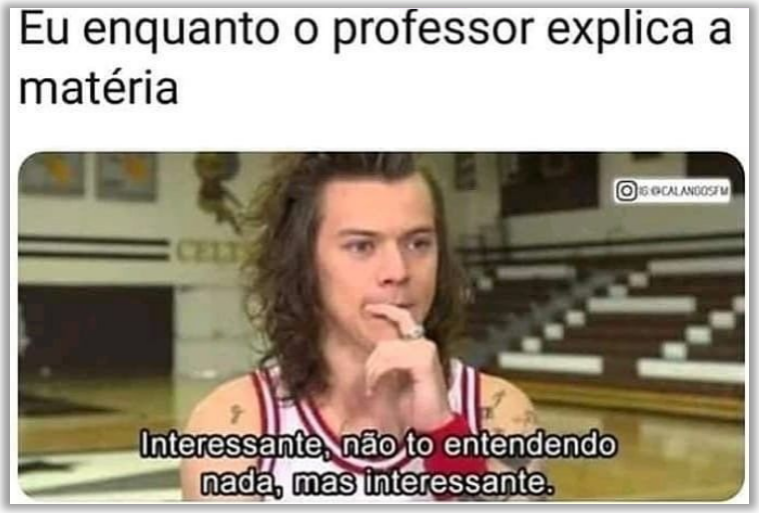
Dessa forma, recomendo que vocês tenham muita atenção na seleção das questões realizadas para que não extrapolem o nível cobrado na sua prova.

Qualquer dúvida, estou à disposição para maiores esclarecimentos!

APRESENTAÇÃO DA AULA

Fala, galera! O assunto da nossa aula de hoje é **SQL (Structured Query Language)**! *É um assunto difícil?* Eu não vou mentir para vocês! A parte básica é simples, mas ele pode entrar em um nível de complexidade de moer o cérebro, sangrar o nariz e fazer a gente chorar em posição fetal no banheiro). Maaaaas eu estou aqui justamente para minimizar o trauma e facilitar a vida de vocês. Vem comigo que vai dar bom ;)

PROFESSOR DIEGO CARVALHO - WWW.INSTAGRAM.COM/PROFESSORDIEGOCARVALHO



Galera, todos os tópicos da aula possuem Faixas de Incidência, que indicam se o assunto cai muito ou pouco em prova. *Diego, se cai pouco para que colocar em aula?* Cair pouco não significa que não cairá justamente na sua prova! A ideia aqui é: se você está com pouco tempo e precisa ver somente aquilo que cai mais, você pode filtrar pelas incidências média, alta e altíssima; se você tem tempo sobrando e quer ver tudo, vejam também as incidências baixas e baixíssimas. *Fechado?*

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

INCIDÊNCIA EM PROVA: BAIXA

INCIDÊNCIA EM PROVA: MÉDIA

INCIDÊNCIA EM PROVA: ALTA

INCIDÊNCIA EM PROVA: ALTÍSSIMA

Além disso, essas faixas não são por banca – é baseado tanto na quantidade de vezes que caiu em prova independentemente da banca e também em minhas avaliações sobre cada assunto...

SQL

Conceitos Básicos

INCIDÊNCIA EM PROVA: BAIXA

Fala, galeeeeeera... o assunto da nossa aula de hoje é SQL! Bem, eu não vou começar logo de cara falando sobre conceitos, definições, tipos, entre outros. Primeiro, nós vamos fazer uma pequena introdução! **Sabemos que um dado é basicamente um fato relacionado a algum objeto em um determinado contexto. Nome, Idade, Altura e Peso são dados sobre mim ou sobre você!** Fotos, imagens, arquivos e videoaulas também podem ser considerados dados.

E um banco de dados? Um banco de dados é um conjunto de dados relacionados. Isso mesmo... um banco de dados não armazena dados aleatórios – ele armazena uma coleção sistemática de dados relacionados. Uma vez que dados podem ser armazenados dentro de um banco de dados, o seu gerenciamento se torna bem mais simples. *E o que seria um Sistema Gerenciador de Bancos de Dados (SGBD)?* **Trata-se de uma coleção de programas que permitem que usuários criem e acessem bancos de dados – assim como permitem que usuários manipulem seus dados**. Ele tem várias outras funções, mas essas são suas funções principais.

Por fim, é importante entender que dados podem ser representados por meio de diversos modelos distintos, sendo o mais popular de todos o Modelo Relacional. *O que é o Modelo Relacional?* **Trata-se de um modelo que se baseia em uma estrutura em que os dados são armazenados em tabelas (também conhecidas como relações)**. Pronto! Agora que nós já apresentamos todas essas definições, nós podemos finalmente ir para o assunto principal da nossa aula...

Prof. Diego, eu gostei desse negócio de banco de dados! O que eu preciso para poder começar a armazenar e manipular dados agora mesmo? **Você precisará de um Banco de Dados (BD), de um Sistema Gerenciador de Banco de Dados Relacional (SGBDR) e também de uma coisinha**

chamada SQL. *Que diabos é isso, professor?* Essa é uma sigla para Structured Query Language (ou Linguagem de Consulta Estruturada). Vamos entender melhor...

SQL é basicamente a linguagem padrão para manipulação de bancos de dados relacionais. **Por meio dela, um usuário pode executar comandos para inserir, pesquisar, atualizar ou deletar registros em um banco de dados, criar ou excluir tabelas, conceder ou revogar permissões para acessar o banco de dados, entre diversos outros recursos**. *Entendido?* Legal, acho que vocês conseguiram entender na teoria, mais à frente veremos como funciona na prática...

> **(TRE/BA – 2010)** Uma linguagem de acesso a bancos de dados relacionais amplamente usada é o SQL.
>
> _____
>
> **Comentários:** SQL é uma linguagem de acesso e manipulação de bancos de dados relacionais (Correto).

Antes de prosseguir, eu acho interessante mencionar algo que cai com bastante frequência em prova. **No mundo dos computadores, as linguagens possuem um papel muito importante!** Da mesma forma que um russo e um colombiano não conseguem se comunicar se não souberem um idioma em comum, um humano e um computador também não conseguem se comunicar sem utilizar uma mesma linguagem[1].

Programadores, em geral, são poliglotas: eles compreendem diversas linguagens (Ex: Java, PHP, Python, Ruby, etc). **Existem muitas linguagens no mundo dos computadores – existem tantas que elas foram categorizadas em diversos paradigmas.** O intuito aqui não é mostrar todos esses paradigmas, somente dois: paradigma declarativo e paradigma procedural. *Diego, qual a diferença entre esses dois paradigmas?*

Em suma: uma linguagem declarativa descreve o que fazer; já uma linguagem procedural descreve como fazer. *Professor, não entendi bulhufas!* É mais ou menos assim: se sua mãe utilizasse uma linguagem procedural, ela diria: "*Filhão, siga reto no corredor; vire à direita; entre na suíte; acesse o banheiro; haverá um armário do lado da pia e na terceira gaveta do lado esquerdo está uma maquiagem cor âmbar-tapioca. Pega pra mamãe, por favor!*".

Agora se sua mãe utilizasse uma linguagem declarativa, ela apenas diria: "*Se vire e me traga a maquiagem âmbar-tapioca, moleque!*". *E qual é a diferença, professor?* A diferença é que a linguagem procedural mostra tim-tim por tim-tim como algo deve ser feito; já a linguagem declarativa não está nem aí para como será feito, ela só quer o resultado! **Essa história toda foi para dizer que a linguagem SQL se trata de uma linguagem declarativa!**

Fiquem tranquilos! Toda essa história fará sentido mais à frente e, ao final da aula, não restará dúvidas sobre o porquê de ser considerada uma linguagem declarativa. Apesaaaaar disso, com o

[1] Lembrando que o SQL é a linguagem comum entre um humano e um SGBD.

passar do tempo, alguns fornecedores introduziram alguns comandos procedurais e algumas extensões à linguagem SQL e lançaram com nomes parecidos, tais como: PL/SQL ou Transact-SQL. No entanto, essas extensões são conhecidas como dialetos SQL e, não, SQL em sua forma original!

Bem... nosso intuito aqui é estudar o SQL tradicional (de raiz!), definido pelo **American National Standards Institute (ANSI)** com uma ou outra observação sobre outros dialetos. *Bacana?*

(UFG – 2018) SQL é uma linguagem utilizada para manipular e consultar os dados das tabelas de um banco de dados. A SQL é considerada uma linguagem:

a) lógica.
b) declarativa.
c) imperativa.
d) funcional.

_____
Comentários: SQL é uma linguagem declarativa (Letra B).

Tipos de Dados

INCIDÊNCIA EM PROVA: BAIXA

Antes de partirmos para as sub-linguagens, é importante estudar os tipos de dados pré-definidos pela linguagem SQL! Galera, em uma tabela armazenada em um banco de dados relacional, temos linhas e colunas. Para que uma tabela seja criada, é necessário especificar qual tipo de dados serão armazenados em suas colunas. O Padrão ANSI/SQL traz um conjunto de tipos de dados básicos padronizados pela linguagem. Não vamos falar sobre todos, mas vamos falar dos principais...

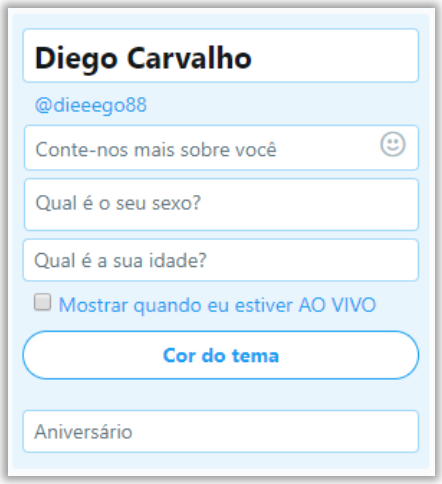
Textual

Também conhecido como literal, esse tipo de dado é basicamente uma cadeia de caracteres. Podem ter um tamanho fixo (Ex: **CHAR(n)**, em que **n** é o número fixo de caracteres) ou um tamanho variável (Ex: **VARCHAR(n)**, em que **n** é o número máximo de caracteres). *Sabe o Twitter? Você pode fazer um tweet com até 280 caracteres, correto?* Logo, no banco de dados do Twitter, isso pode estar modelado com uma coluna de texto do tipo **VARCHAR(280)**.

Ué, Diego! Não deveria ser modelado com CHAR(280)? Não, eu disse que eram **até** 280 caracteres! A modelagem com CHAR(280) obrigaria o usuário a escrever um tweet com exatamente 280 caracteres – nem mais nem menos! O Padrão ANSI/SQL-99 trouxe ainda um tipo de texto chamado **CLOB** (**C**haracter **L**arge **Ob**ject) capaz de armazenar textos gigantescos, sendo que o tamanho máximo depende da especificidade de cada SGBD.

Por fim, é interessante lembrar que – ao inserir um texto diretamente no banco de dados – ele deve ser colocado entre aspas simples (apóstrofos). Ademais, ele é *Case Sensitive* (diferencia maiúsculas de minúsculas). Observação importante: os valores do tipo texto inseridos em uma linha do banco de dados são *Case Sensitive*, mas as palavras-chave que compõem a linguagem, não. Dessa forma, você pode escrever tanto **`CREATE TABLE`** quanto **`create table`**.

Numérico



Esse tipo de dado inclui números inteiros de diversos tamanhos (Ex: **`INTEGER`**, **`INT`** ou **`SMALLINT`**) e números reais (**`FLOAT`**, **`REAL`** e **`DOUBLE PRECISION`**) – também conhecidos como ponto flutuante. *Por que esse nome, professor?* Porque – em números reais – o ponto varia de posição (Ex: 35.2 ou 0.458). Você pode definir a precisão e a escala desses números: a precisão é o número total de dígitos; e a escala é o número de dígitos após o ponto decimal. Respondam-me: *qual é a escala e a precisão do número 527918.436?* Precisão é o número total de dígitos, logo é 9; e a escala é o número de dígitos após o ponto decimal, logo é 3; portanto, esse número poderia estar armazenado em uma coluna do banco de dados como FLOAT(9,3). Voltemos ao Twitter...

Se – durante a inscrição – fosse requisitado informar a idade, ela poderia ser armazenada no banco de dados da aplicação como um número inteiro, como é apresentado na imagem anterior.

Binário

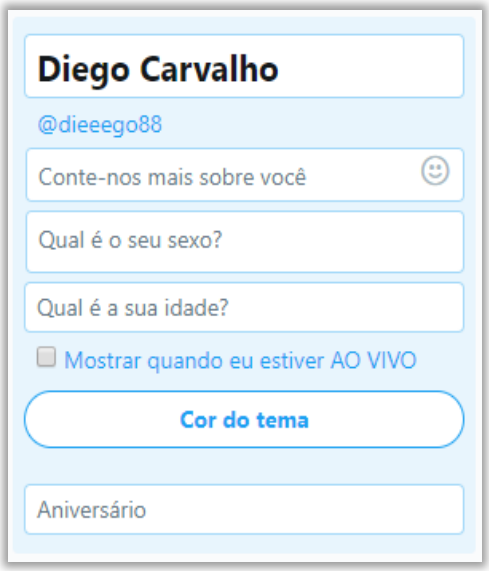
Esse tipo de dado é basicamente uma cadeia de bits. Podem ter um tamanho fixo (Ex: **`BIT(n)`**, em que **`n`** é o número fixo de caracteres) ou um tamanho variável (Ex: **`BIT VARYING(n)`**, em que ***`n`*** é o

número máximo de caracteres). Há ainda o **BLOB** (**B**inary **L**arge **Ob**ject), capaz de armazenar uma quantidade gigantesca de bits. *Para que serve isso, professor?* Voltemos para o exemplo do Twitter: durante a inscrição, é requisitada uma foto de perfil para seu novo Twitter.

Onde será que fica armazenada essa foto? Essa foto fica armazenada em um campo de uma tabela que armazena o endereço de um **BLOB**. Em geral, todo arquivo que você armazena em um banco de dados fica em uma coluna desse tipo de dado porque ele é capaz de suportar uma quantidade gigantesca de bits. E, adivinhem só: um arquivo é basicamente um conjunto de bits (seja ele uma foto, um documento, um software, um vídeo, entre outros).

Booleano

Esse tipo de dado tem como valores tradicionais **TRUE** (Verdadeiro) ou **FALSE** (Falso). Na verdade, há um terceiro valor, que é o **NULL**. É chamado de booleano em homenagem a George Boole, que definiu um sistema de lógica algébrica pela primeira vez na metade do século XIX. O tipo Booleano utiliza apenas 1 byte para seu armazenamento, uma vez que o valor falso é representado por o (oooooooo) e o valor verdadeiro é representado por 1 (oooooooı). *Professor, dá um exemplo?* Claro! Na inscrição do Twitter, é requisitado informar o seu sexo (Masculino ou Feminino) – como mostra a imagem ao lado. Você pode armazenar esse valor em uma coluna do tipo Boolean e considerar o valor 1 como Masculino e o valor o como Feminino. Esse tipo de dado também é muito comum para sinalizar uma linha.



Imaginem uma coluna que armazena um booleano com valor 1 para usuários com mais de um milhão de seguidores e o valor zero para usuários com menos que isso.

Data

Esse tipo de dado possui dez posições, e seus componentes são DAY (Dia), MONTH (Mês) e YEAR (Ano) na forma DD-MM-YYYY (Ex: 30/03/2019). Somente datas válidas devem ser permitidas pela implementação do SQL. Em outras palavras, os meses devem estar entre 1 e 12 e os dias devem estar entre 1 e 31. Além disso, os dias devem ser válidos para o mês correspondente – logo, não é possível haver a data 30/02/2020 porque o mês de fevereiro nunca tem 30 dias. Vejam ao lado que o dia, mês e ano da sua data de aniversário no Twitter podem ser armazenados em um campo cujo tipo de dado seja Data!

Hora

Esse tipo de dado possui pelo menos oito posições compostas por HOUR (Hora), MINUTE (Minuto) e SECOND (Segundo) na forma HH:MM:SS. Somente horas válidas devem ser permitidas pela implementação do SQL. Em outras palavras, as horas devem estar entre 0 e 23, os minutos devem estar entre 0 e 59 e os segundos também devem estar entre 0 e 59. Não encontrei um exemplo no Twitter, mas vocês já conseguem imaginar como seria! *Bacana?*

Outros

Existem outros tipos de dados que foram acrescentados em versões posteriores do Padrão ANSI/SQL. Entre eles, é importante mencionar o **TIMESTAMP**, que é basicamente uma junção da Data com Hora – e até Fuso Horário (Ex: 27-09-2008 09:12:47.648302). Outro tipo interessante é o **INTERVAL**, que permite calcular o intervalo entre Datas ou Horas; e o **DATETIME** combina data e hora em um único tipo, com intervalo de datas.

Como já disse, existem outros tipos de dados (alguns dependentes de SBGD), mas eu não acho que vale a pena entrar em detalhes sobre eles. Por fim, é importante conhecer os valores nulos! Um campo com valor NULL é um campo sem valor. Se um campo em uma tabela for opcional, é possível inserir um novo registro ou atualizar um registro sem adicionar um valor a este campo. Em seguida, o campo será salvo com um valor NULL.

Observação: um valor NULL é diferente de um valor zero ou de um campo que contém espaços – um campo com um valor NULL é aquele que foi deixado em branco durante a criação do registro!

HTTPS://WWW.INSTAGRAM.COM/P/CHQAFU601QP

(TJ/RS – 2014) Os tipos de dados básicos para atributos em SQL incluem:

a) classes, listas e conjuntos ordenados.
b) vetores e matrizes.
c) ponteiros e estruturas.
d) números, cadeia de caractere, cadeia de bits, booleanos, data e hora.
e) listas, pilhas, filas e deques.

_____

Comentários: os tipos de dados incluem números, cadeia de caracteres, cadeia de bits, booleanos, data e hora (Letra D).

(BRDE – 2012) Preencha as lacunas e, em seguida, assinale a alternativa correta. Em SQL-99 temos o tipo de dados de atributos _____, os tipos de dados cadeia de caracteres ou tem tamanho_____ – CHAR(n) ou _____, em que n é o

número de caracteres – ou tem tamanho _____ VARCHAR(n) ou CHAR VARIYNG ou CHARACTER VARIYNG(n), em que n é o número máximo de caracteres.

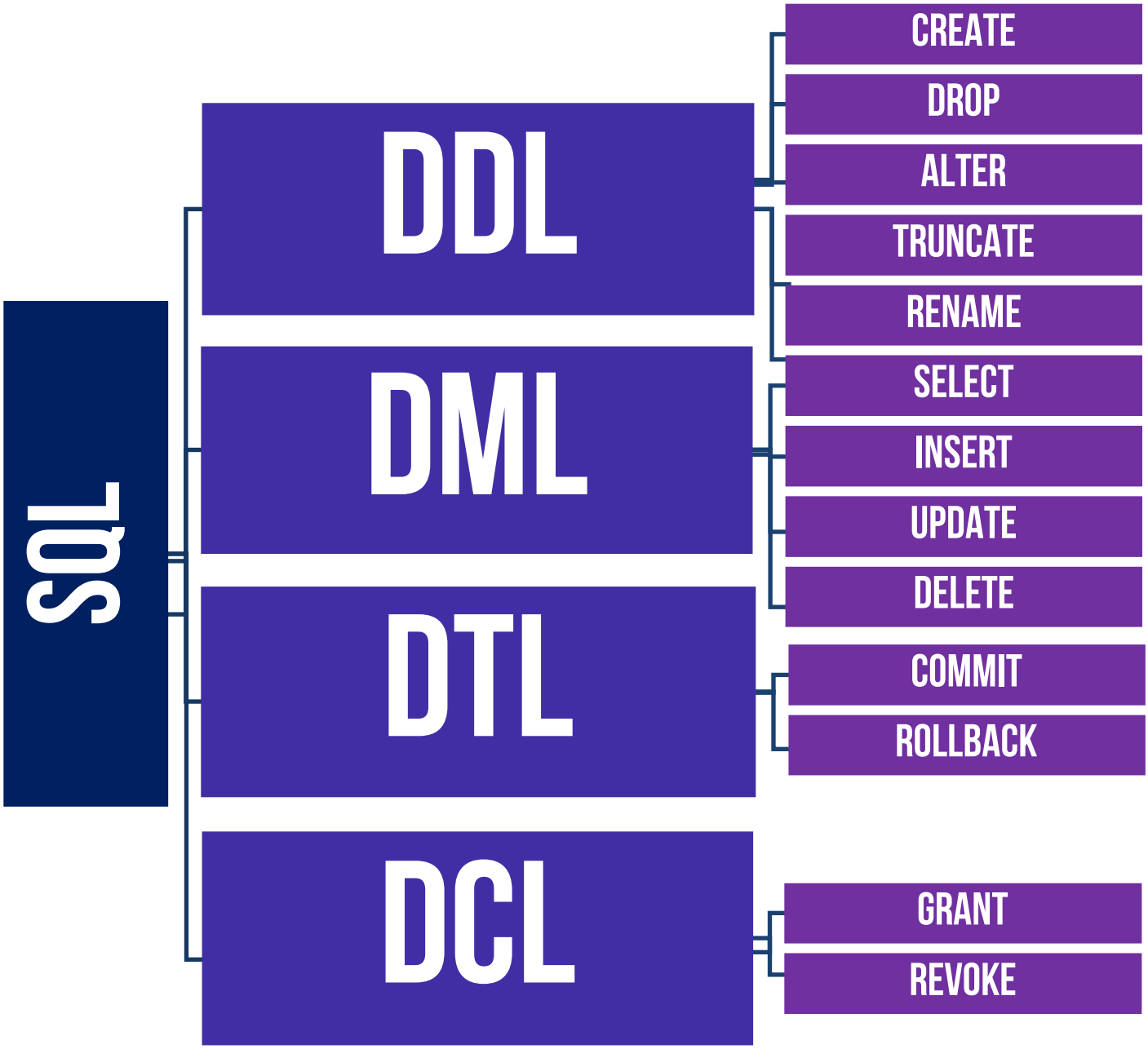
a) cadeia de ponteiros / fixo / CHARACTER(n) / variável
b) cadeia de caracteres / fixo / CHARACTER(n) / variável
c) cadeia de caracteres / fixo / variável / VARCHAR(n)
d) cadeia de caracteres / variável / fixo / VARCHAR(n)
e) cadeia de ponteiros / variável / fixo / VARCHAR(n)

_____

Comentários: no SQL-99, temos o tipo de dados de atributos cadeia de caracteres, os tipos de dados cadeia de caracteres ou tem tamanho fixo – CHAR(n) ou CHARACTER(n), em que n é o número de caracteres – ou tem tamanho variável VARCHAR(n) ou CHAR VARIYNG ou CHARACTER VARIYNG(n) (Letra B).

Sublinguagens SQL

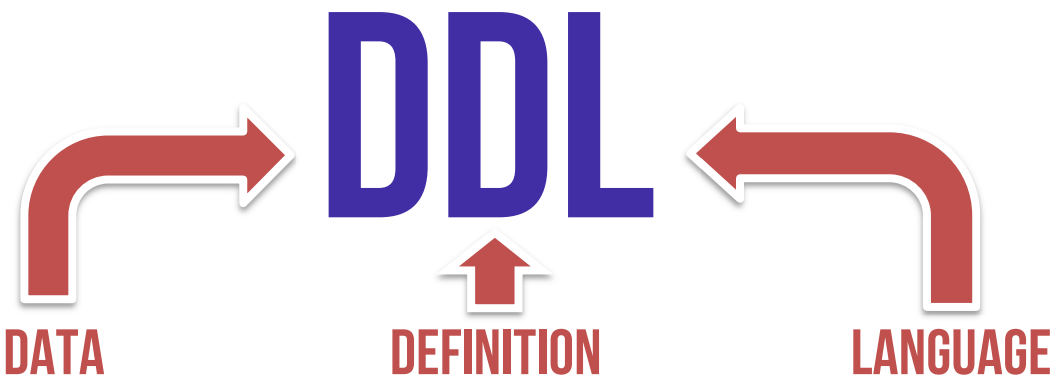
SQL é uma linguagem que permite realizar determinadas operações em um banco de dados relacional por meio de um conjunto de comandos. **Em geral, esses comandos são agrupados basicamente em quatro sublinguagens: DDL, DML[1], DCL e DTL[2]**. Nos próximos tópicos, vamos estudá-las em detalhe! *Professor, eu vou ter que decorar todos os comandos?* Sim, mas calma... existem poucos comandos que caem em prova e eles são bastante intuitivos.



[1] Alguns consideram que o comando SELECT não faz parte da DML, mas de uma subcategoria especial chamada DQL (*Data Query Language*).

[2] Também chamada de TCL (*Transaction Control Language*).

DDL (DATA DEFINITION LANGUAGE)



Em tradução livre, *Data Definition Language* é o mesmo que Linguagem de Definição de Dados! *Logo, essa linguagem serve para quê?* Ela serve para definir dados! *E o que isso significa?* Isso significa que essa linguagem possui um conjunto de comandos que podem ser utilizados para definir um esquema de banco de dados! Eu tenho certeza que vocês se lembram da definição de **esquema**, visto em nossa aula de conceitos básicos...

Um esquema é uma descrição da estrutura de um banco de dados, sendo responsável por descrever tabelas, campos, views, funções, entre outros. Dito isso, nós podemos afirmar que a DDL é um subconjunto dos comandos SQL que podem ser utilizados para definir ou descrever um esquema de um banco de dados, permitindo criar ou modificar a estrutura de objetos de uma base de dados relacional. *Eu sei que isso é bem abstrato, mas isso ficará claro...*

Por enquanto, o que vocês precisam entender é que SQL é uma linguagem com vários comandos e os comandos responsáveis por definir essa estrutura são chamados de DDL. Vejamos...

COMANDOS DDL	DESCRIÇÃO
CREATE	Comando utilizado para criar tabelas (e outros objetos) de um banco de dados.
DROP	Comando utilizado para deletar uma tabela (e outros objetos) de um banco de dados.
TRUNCATE	Comando utilizado para apagar os dados de uma tabela de um banco de dados.
ALTER	Comando utilizado para manipular colunas ou restrições de um banco de dados.
RENAME	Comando utilizado para renomear uma tabela de um banco de dados.

(TRT/SE – 2010) O SQL (Structured Query Language) é uma linguagem de pesquisa declarativa para banco de dados relacional. A DDL permite ao usuário definir tabelas novas e elementos associados. A sigla DDL significa:

a) Data Default List.
b) Definition Data Language.
c) Data Definition Language.
d) Data Default Language.
e) Data Definition List.

_____

Comentários: a sigla DDL significa Data Definition Language (Letra C).

(TRT/AL – 2011) É um comando do tipo DDL (Data Definition Language) no SQL:

a) SELECT
b) DELETE
c) INSERT
d) UPDATE
e) CREATE

_____

Comentários: dentre as opções, o único comando DDL é o CREATE (Letra E).

(TJ/PB – 2012) No contexto da linguagem SQL, assinale a alternativa que indica o comando pertencente a DDL (Data Definition Language).

a) INSERT
b) DELETE
c) UPDATE
d) DROP

_____

Comentários: dentre as opções, o único comando DDL é o DROP (Letra D).

(IF/TO – 2018) O comando TRUNCATE, pertence a qual tipo de linguagem de dados SQL:

a) DCL (Data Control Language)
b) DTL (Data Transaction Language)
c) SDL (Storage Definition Language)
d) DDL (Data Definition Language)
e) DML (Data Manipulation Language)

_____
Comentários: esse comando pertence à DDL (Letra D).

IMPORTANTE

Antes de continuar, é fundamental entender que essa aula é bastante grande e complexa. Logo, existem diversos caminhos didáticos diferentes que podemos percorrer para ter a melhor compreensão sobre os seus conceitos. No entanto, um conceito eventualmente será citado sem maior detalhamento, de forma que seu aprofundamento ocorra somente mais à frente. Por que, professor?

Porque os assuntos são conectados e se torna contraproducente explicar todos os conceitos ao mesmo tempo. Fiquem tranquilos... ao final, tudo fará sentido na cabeça de vocês. Dito isso, todos os tópicos a seguir serão detalhados em relação apenas a tabelas e suas estruturas. Nos tópicos finais, vamos mencionar o funcionamento dos comandos para outros objetos.

Além disso, nós veremos que a sintaxe de alguns comandos se modifica dependendo do tipo específico de SGBD. Sim, existem dezenas de SGBDs no mercado, mas vamos nos ater aqui aos mais importantes (Oracle, SQL Server e MySQL). É muito rara a cobrança de uma ferramenta específica – nós vamos mancionar apenas para que vocês não sejam surpreendidos em eventuais questões de prova.

Comandos DDL

CREATE TABLE

INCIDÊNCIA EM PROVA: ALTA

Esse comando permite criar uma tabela em um banco de dados. Vejamos como seria o seu funcionamento básico...

SINTAXE DO COMANDO

```
CREATE TABLE NOME_DA_TABELA (
      NOME_COLUNA1              TIPO_DE_DADO       RESTRIÇÕES   ,
      NOME_COLUNA2              TIPO_DE_DADO       RESTRIÇÕES   ,
      NOME_COLUNA3              TIPO_DE_DADO       RESTRIÇÕES   ,
      ...
      );
```

EXEMPLO DO COMANDO

```
CREATE TABLE ALUNO (
      NOME                VARCHAR(20)        NOT NULL      ,
      CPF                 INT                PRIMARY KEY   ,
      SEXO                CHAR(1)            NOT NULL      ,
      DATA_NASCIMENTO     DATE               NOT NULL      ,
      CIDADE              VARCHAR(50)        NOT NULL      ,
      VALOR_PAGO          INT                NOT NULL
      );
```

RESULTADO DO COMANDO

ALUNO					
NOME	CPF	SEXO	DATA_NASCIMENTO	CIDADE	VALOR_PAGO

Observem que a sintaxe indica que é necessário especificar o nome da tabela e – para cada coluna – indicar seu nome, tipo de dado e eventuais restrições. No exemplo acima, nós criamos uma tabela (`ALUNO`) que contém seis colunas (`NOME`, `CPF`, `SEXO`, `DATA_NASCIMENTO`, `CIDADE`, `VALOR_PAGO`). Cada coluna possui um tipo (`VARCHAR(20)`, `INT`, `CHAR(1)`, `DATE`, `VARCHAR(50)`, `INT`) e pode armazenar dados apenas desse respectivo tipo.

Eu posso inserir um dado do tipo inteiro na coluna `NOME`*?* Não, apenas texto*! Eu posso inserir um dado do tipo texto dentro da coluna* `DATA_NASCIMENTO`*?* Não, apenas data! *Professor, e o que são aqueles números após alguns tipos de dados?* Eles basicamente indicam quantidade! A coluna que armazena dados do nome do aluno aceitará - no máximo - 20 caracteres; a coluna que armazena dados do sexo do aluno aceitará - no máximo - um caractere.

Por fim, note que as colunas possuem determinadas restrições. *Como é isso, Diego?* Restrições são limitações de uma coluna (Ex: determinada coluna deve ser chave primária; determinada coluna deve ser chave estrangeira; determinada coluna não pode ser nula). O resultado do comando é uma tabela com suas respectivas colunas e restrições, mas sem nenhum dado, uma vez que nós apenas criamos a tabela, mas ainda não inserimos nada dentro...

ALUNO_ESCOLA_ANTIGA					
NOME	CPF	SEXO	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
Alice	111.111.111-11	'F'	01/01/2010	Alexânia	1000,00
Bernardo	222.222.222-22	'M'	02/02/2010	Brasília	2000,00
Caio	333.333.333-33	'M'	03/03/2010	Campinas	3000,00
Denise	444.444.444-44	'F'	04/04/2010	Diadema	4000,00
Elis	555.555.555-55	'F'	05/05/2010	Extrema	5000,00

É possível também criar uma tabela a partir de outra tabela! *Como assim, Diego?* Vamos imaginar que exista uma tabela já populada[3] no banco de dados chamada ALUNO_ESCOLA_ANTIGA. Se nós desejamos criar uma nova tabela a partir dessa tabela, basta seguir a sintaxe apresentada a seguir. Não se preocupem em entender o restante do comando (estudaremos em detalhes mais à frente). Guardem apenas que é possível criar uma tabela a partir de outra pré-existente...

SINTAXE DO COMANDO

```
CREATE TABLE NOME_TABELA_NOVA AS
       SELECT NOME_COLUNA1, NOME_COLUNA2, NOME_COLUNA3, ...
       FROM   NOME_TABELA_ANTIGA
       WHERE ...
```

EXEMPLO DO COMANDO

```
CREATE TABLE ALUNO_ESCOLA_NOVA AS
       SELECT NOME, CPF, SEXO, DATA_NASCIMENTO, CIDADE, VALOR_PAGO
       FROM   ALUNO_ESCOLA_ANTIGA
```

RESULTADO DO COMANDO

ALUNO_ESCOLA_NOVA					
NOME	CPF	SEXO	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
Alice	111.111.111-11	'F'	01/01/2010	Alexânia	1000,00
Bernardo	222.222.222-22	'M'	02/02/2010	Brasília	2000,00
Caio	333.333.333-33	'M'	03/03/2010	Campinas	3000,00
Denise	444.444.444-44	'F'	04/04/2010	Diadema	4000,00
Elis	555.555.555-55	'F'	05/05/2010	Extrema	5000,00

[3] Esse é o termo técnico que indica que foram inseridos dados na tabela (tabela populada = tabela não vazia).

Observem que, ao criar uma tabela a partir de outra, os dados contidos na tabela original também são copiados para a nova tabela.

(SEDF – 2017) O comando CREATE TABLE é responsável pela criação de tabelas, incluindo as colunas e seus tipos de dados. No entanto, com esse comando, não é possível especificar a chave primária da tabela.

_____

Comentários: esse comando permite – sim – especificar a chave primária (Errado).

DROP TABLE

INCIDÊNCIA EM PROVA: BAIXA

Esse comando é utilizado para excluir uma tabela existente em um banco de dados. Ele deve ser utilizado com extremo cuidado porque ele apagará a tabela junto com todos os seus dados.

SINTAXE DO COMANDO

```
DROP TABLE NOME_DA_TABELA;
```

EXEMPLO DO COMANDO

```
DROP TABLE ALUNO_ESCOLA_NOVA;
```

RESULTADO DO COMANDO

ALUNO_ESCOLA_NOVA

NOME	CPF	SE…	DATA…	…TMENTO	CIDADE	VALOR_PAGO
Alice	111.111.111-11			…0	Alexânia	1000,00
Bernardo	222.222.222-22	'M…		…2010	Brasília	2000,00
Caio	333.333.333-33	'M'		'2010	Campinas	3000,00
Denise	444.444.444-44			…0	Diadema	4000,00
Elis	555.555.555-55	'…	05/…	…010	Extrema	5000,00

TABELA EXCLUÍDA

Observem que nós criamos a tabela por meio do comando apresentado no tópico anterior e, por meio do comando apresentado no tópico atual, a tabela foi excluída. Simples assim...

(SUFRAMA – 2014) O comando drop table remove toda a tabela da base de dados. Um exemplo de utilização desse comando é o seguinte: drop table examplo_timestamp;

_____

Comentários: a sintaxe e o exemplo da questão estão perfeitos (Correto).

(DETRAN/RN – 2010) Sobre o comando "drop table pedido;" assinale a alternativa correta:

a) Cria a tabela pedido.
b) Elimina a tabela pedido.
c) Duplica a tabela pedido.
d) Cria uma chave primária na tabela pedido.
e) Extrai dados da tabela pedido.

_____

Comentários: o comando DROP TABLE PEDIDO elimina a tabela pedido (Letra B).

TRUNCATE TABLE

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

Esse comando é utilizado para **apagar todos os dados de uma tabela, porém sem apagar a tabela em si.** Em contraste com o comando anterior, aqui a estrutura da tabela permanece inalterada...

SINTAXE DO COMANDO

```
TRUNCATE TABLE NOME_DA_TABELA;
```

EXEMPLO DO COMANDO

```
TRUNCATE TABLE ALUNO_ESCOLA_NOVA;
```

RESULTADO DO COMANDO

ALUNO_ESCOLA_NOVA					
NOME	CPF	SEXO	DATA_NASCIMENTO	CIDADE	VALOR_PAGO

(TRT/SP – 2015) No SQL há dois comandos que podem eliminar completamente os registros de uma tabela. A diferença entre eles é que o comando I irá eliminar apenas os dados (registros), enquanto o comando II irá eliminar também a tabela. Os comandos I e II são, respectivamente:

a) TRUNCATE TABLE e DROP TABLE.
b) DROP TABLE e DELETE FROM.
c) DELETE RECORD e DELETE TABLE.
d) DROP TABLE e TRUNCATE TABLE.
e) REMOVE RECORD e DROP TABLE.

Comentários: o comando que elimina apenas os dados é o TRUNCATE TABLE e o comando que elimina dados e também a tabela é DROP TABLE (Letra A).

(ANAC – 2009) Quando se executa com sucesso o comando SQL "TRUNCATE TABLE cidades", todos os registros da tabela cidades são removidos.

Comentários: esse comando realmente removerá todos os registros da tabela cidades (Correto).

ALTER TABLE

INCIDÊNCIA EM PROVA: BAIXA

Esse comando é utilizado para adicionar, deletar ou modificar colunas de uma tabela existente – assim como permite modificar restrições. Para adicionar uma coluna, utilizamos a sintaxe:

SINTAXE DO COMANDO

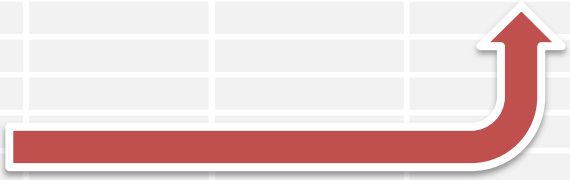
```
ALTER TABLE NOME_DA_TABELA
ADD COLUMN NOME_COLUNA TIPO_DE_DADO;
```

EXEMPLO DO COMANDO

```
ALTER TABLE ALUNO
ADD COLUMN EMAIL VARCHAR(255);
```

RESULTADO DO COMANDO

ALUNO						
NOME	CPF	SEXO	DATA_NASCIMENTO	CIDADE	VALOR_PAGO	EMAIL



Vejam que nós adicionamos uma coluna do tipo texto chamada `EMAIL` capaz de armazenar até 255 caracteres. **Já a sintaxe para excluir uma coluna de uma tabela é:**

SINTAXE DO COMANDO

```
ALTER TABLE NOME_DA_TABELA
DROP COLUMN NOME_COLUNA;
```

EXEMPLO DO COMANDO

```
ALTER TABLE ALUNO
DROP COLUMN SEXO;
```

RESULTADO DO COMANDO

ALUNO					
NOME	CPF	DATA_NASCIMENTO	CIDADE	VALOR_PAGO	EMAIL

Notem que a coluna SEXO foi excluída do resultado. Agora vamos ver como é a sintaxe para modificar o tipo de dado de uma coluna da tabela:

SINTAXE DO COMANDO (SQL SERVER / MS-ACCESS)

```
ALTER TABLE NOME_DA_TABELA
ALTER COLUMN NOME_COLUNA TIPO_DE_DADO;
```

SINTAXE DO COMANDO (MYSQL / ORACLE PRÉ-10G)

```
ALTER TABLE NOME_DA_TABELA
MODIFY COLUMN NOME_COLUNA TIPO_DE_DADO;
```

SINTAXE DO COMANDO (MYSQL / ORACLE 10G EM DIANTE)

```
ALTER TABLE NOME_DA_TABELA
MODIFY NOME_COLUNA TIPO_DE_DADO;
```

Professor, por que há três opções de sintaxe diferentes? Porque existem pequenas diferenças a depender do SGBD utilizado. Vamos ver um exemplo utilizando a sintaxe do SQL Server:

```
ALTER TABLE ALUNO
ALTER COLUMN CPF VARCHAR(14);
```

Notem que alteramos a coluna CPF da tabela ALUNO. *O que foi feito, professor?* Nós apenas convertemos o tipo de dado dessa coluna de INT para VARCHAR. Por fim, é importante mencionar que – por meio desse comando – é possível também inserir ou excluir restrições de uma coluna. Você pode alterar uma coluna para indicar que ela não pode ser nula ou que ela será uma chave estrangeira. Isso ficará mais claro quando estudarmos as restrições...

(MPE/AM – 2013) O comando SQL utilizado para adicionar, modificar ou remover colunas em uma tabela existente é chamado:

a) INSERT INTO.
b) DROP TABLE.
c) CREATE TABLE.
d) ALTER TABLE.
e) TRUNCATE.

_____

Comentários: o comando que permite adicionar, modificar ou remover colunas é o ALTER TABLE (Letra D).

(ANAC – 2009) O comando SQL ALTER TABLE possibilita alterar a estrutura de uma tabela, como por exemplo, adicionando ou removendo uma coluna de uma tabela.

_____

Comentários: adicionar ou remover colunas de uma tabela são de possibilidades do comando ALTER TABLE (Correto).

RENAME TABLE

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

Em algumas situações, administradores de banco de dados desejam alterar o nome de uma tabela a fim de dar um nome mais relevante ou por outra razão. Para tal, é possível fazer:

SINTAXE DO COMANDO

```
RENAME TABLE NOME_DA_TABELA
TO NOVO_NOME_DA_TABELA;
```

EXEMPLO DO COMANDO

```
RENAME TABLE ALUNO
TO ALUNO_ESTRATEGIA;
```

RESULTADO DO COMANDO

ALUNO_ESTRATEGIA					
NOME	CPF	DATA_NASCIMENTO	CIDADE	VALOR_PAGO	EMAIL

Alternativamente, também é possível utilizar o comando `ALTER TABLE` para modificar o nome de uma determinada tabela. Vejamos a sintaxe:

SINTAXE DO COMANDO

```
ALTER TABLE NOME_DA_TABELA
RENAME TO NOVO_NOME_DA_TABELA;
```

EXEMPLO DO COMANDO

```
ALTER TABLE ALUNO
RENAME TO ALUNO_ESTRATEGIA;
```

(BAHIAGÁS – 2016) A Linguagem de Definição de Dados (DDL) é um conjunto de comandos dentro da SQL utilizada para a definição de estrutura dos dados e tabelas. Com relação a este assunto assinale a alternativa correta:

a) Os comandos DDL mais comuns são CREATE, ALTER, DROP, RENAME e TRUNCATE
b) Os comandos DDL mais comuns são ALTER, GRANT, INSERT e TRUNCATE
c) Os comandos DDL mais comuns são ALTER, GRANT, INSERT, RENAME e SELECT
d) Os comandos DDL mais comuns são CREATE, ALTER, GRANT, INSERT e SELECT
e) Os comandos DDL mais comuns são CREATE, ALTER, INSERT, SELECT e TRUNCATE

Comentários: os comandos DDL mais comuns são CREATE, ALTER, DROP, RENAME e TRUNCATE (Letra A).

(Colégio Pedro II – 2013) Qual das alternativas abaixo apresenta APENAS comandos SQL de definição de dados (DDL)?

a) RENAME, DROP, TRUNCATE.
b) INSERT, DELETE, UPDATE.
c) GRANT, REVOKE.
d) ROLLBACK, COMMIT.
e) SAVEPOINT, UNIQUE, ALTER.

_____

Comentários: (a) Correto; (b) Errado, isso é DML; (c) Errado, isso é DCL; (d) Errado, isso é DTL; (e) Errado, SAVEPOINT é DTL, UNIQUE é apenas uma restrição e ALTER é DML(Letra A).

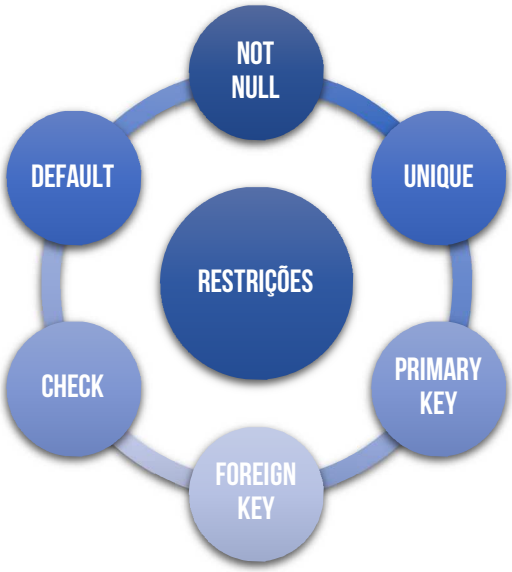
Restrições

Restrições (Constraints) são um conjunto de limitações utilizadas para especificar regras para os dados em uma tabela de um banco de dados relacional. Elas buscam limitar o tipo de dado que pode ser armazenado, o que garante a precisão e confiabilidade aos dados da tabela. Se houver qualquer violação entre a restrição e a ação de dados, a ação será abortada. *Professor, ainda não estou entendendo legal!* Vamos explicar melhor...

Sabe quando você vai criar uma conta em um site e tem que criar uma nova senha? Em geral, existem diversas restrições para escolha da senha (Ex: a senha cadastrada deverá ter ao menos um caractere com letra maiúscula; a senha cadastrada deverá conter ao menos um número; a senha cadastrada deverá conter ao menos um caractere especial; etc). Todas essas são restrições do campo da tabela que armazena as senhas e você pode criar suas próprias restrições personalizadas!

As restrições evitam que determinadas ações violem a integridade da estrutura dos dados especificada no esquema do banco de dados relacional. Elas podem ser no nível de coluna ou no nível de tabela, sendo que as restrições de nível de coluna se aplicam a uma coluna e as restrições de nível de tabela se aplicam a toda tabela. Vamos conhecer nas próximas páginas as principais restrições disponíveis:

CONSTRAINT	DESCRIÇÃO
NOT NULL	Garante que uma coluna não possa ter um valor nulo.
UNIQUE	Garante que todos os valores de uma coluna sejam diferentes entre si.
PRIMARY KEY	Garante que todos os valores de uma coluna sejam diferentes entre si e não nulos.
FOREIGN KEY	Garante que ações não destruam links/relacionamentos entre as tabelas.
CHECK	Garante que os valores em uma coluna satisfaçam uma condição específica.
DEFAULT	Define um valor padrão para uma coluna, se nenhum valor for especificado.



NOT NULL

INCIDÊNCIA EM PROVA: MÉDIA

Por padrão, uma coluna pode conter valores nulos (NULL). *O que isso significa?* Isso significa que – se nada for especificado – não há nenhum problema em existir uma coluna que não contenha nenhum valor. Já a restrição NOT NULL força uma coluna a não aceitar valores nulos. Em outras palavras, essa restrição obriga que determinada coluna contenha valores. Logo, você não pode inserir um novo registro na tabela (ou atualizar um registro existente) sem adicionar valores a esse campo.

Existem basicamente duas maneiras de definir uma coluna como NOT NULL. A primeira é durante a criação da tabela conforme já vimos anteriormente:

SINTAXE DO COMANDO

```
CREATE TABLE NOME_DA_TABELA (
      NOME_COLUNA1        TIPO_DE_DADO        NOT NULL     ,
      ...
      );
```

EXEMPLO DO COMANDO

```
CREATE TABLE ALUNO (
      NOME                VARCHAR(20)         NOT NULL     ,
      CPF                 INT                 PRIMARY KEY  ,
      SEXO                CHAR(1)             NOT NULL     ,
      DATA_NASCIMENTO     DATE                NOT NULL     ,
      CIDADE              VARCHAR(50)                      ,
      VALOR_PAGO          INT
      );
```

Ao criar uma tabela, nós definimos – para cada coluna – nome, tipo e restrição. Nesse caso, basta definir a coluna como NOT NULL. A segunda maneira é por meio do comando ALTER TABLE:

SINTAXE DO COMANDO (MYSQL / ORACLE 10G EM DIANTE)

```
ALTER TABLE ALUNO
MODIFY CIDADE VARCHAR(50) NOT NULL;
```

No final das contas, o que tudo isso significa? Isso significa que a tabela ALUNO não permitirá valores nulos para os campos NOME, CPF, SEXO, DATA_NASCIMENTO e CIDADE. Logo, sempre que um registro for incluído nessa tabela, apenas o campo VALOR_PAGO poderá ficar em branco porque nenhuma restrição foi definida para ele. Todos os outros obrigatoriamente deverão ser preenchidos, caso contrário violarão a restrição especificada e a operação de inserção será abortada.

Professor, uma coluna com valor zero pode ser considerada uma coluna nula? Não! *E uma coluna com espaços em branco?* Também não! Nulo significa ausência de valor, isto é, uma coluna vazia...

UNIQUE

INCIDÊNCIA EM PROVA: BAIXA

Essa restrição garante que todos os valores em uma coluna sejam diferentes! Se uma coluna for definida com essa restrição, nenhum registro poderá ter valores iguais nessa coluna. Exemplo:

EXEMPLO DO COMANDO (SQL SERVER / ORACLE / MS-ACCESS)

```
CREATE TABLE ALUNO (
      NOME              VARCHAR(20)       NOT NULL          ,
      CPF               INT               PRIMARY KEY       ,
      SEXO              CHAR(1)           NOT NULL          ,
      DATA_NASCIMENTO   DATE              NOT NULL          ,
      CIDADE            VARCHAR(50)                         ,
      MATRICULA         INT               UNIQUE            ,
      VALOR_PAGO        INT
      );
```

EXEMPLO DO COMANDO (MYSQL)

```
CREATE TABLE ALUNO (
      NOME              VARCHAR(20)       NOT NULL          ,
      CPF               INT               PRIMARY KEY       ,
      SEXO              CHAR(1)           NOT NULL          ,
      DATA_NASCIMENTO   DATE              NOT NULL          ,
      CIDADE            VARCHAR(50)                         ,
      MATRICULA         INT                                 ,
      VALOR_PAGO        INT
      UNIQUE (MATRICULA)
      );
```

Note que especificamos que a coluna `MATRICULA` é `UNIQUE`! Logo, em todos os registros da tabela, essa coluna não pode ficar vazia nem ter valores repetidos. Um outro ponto interessante é que nós podemos dar um nome a uma restrição de unicidade. *Como assim, Diego?* É isso mesmo que você acabou de ler! É possível nomear uma restrição de unicidade ou defini-la para múltiplas colunas simultaneamente. A sintaxe para ambas as situações é a mesma:

SINTAXE DO COMANDO (MYSQL / SQL SERVER / ORACLE / MS-ACCESS)

```
-- PARA NOMEAR UMA RESTRIÇÃO OU DEFINI-LA PARA MÚLTIPLAS COLUNAS
CREATE TABLE NOME_DA_TABELA (
      NOME_COLUNA1      TIPO_DE_DADO      PRIMARY KEY  ,
      NOME_COLUNA2      TIPO_DE_DADO      NOT NULL     ,
      NOME_COLUNA3      TIPO_DE_DADO                   ,
      NOME_COLUNA4      TIPO_DE_DADO      NOT NULL     ,
      NOME_COLUNA5      TIPO_DE_DADO                   ,
      CONSTRAINT NOME_DA_RESTRICAO UNIQUE (NOME_COLUNA3, NOME_COLUNA5)
      ...
      );
```

É possível também adicionar uma restrição de unicidade a uma coluna de uma tabela pré-existente por meio da sintaxe apresentada a seguir:

SINTAXE DO COMANDO (MYSQL / SQL SERVER / ORACLE / MS-ACCESS)

```
ALTER TABLE NOME_DA_TABELA
ADD UNIQUE (NOME_DA_COLUNA);
```

Professor, o que aconteceria se eu tentasse inserir uma restrição de unicidade a uma coluna que já possua valores repetidos? Antes de inserir a restrição, o Sistema Gerenciador de Banco de Dados (SGBD) analisará os dados da coluna para garantir que todos os valores pré-existentes nela são únicos. Se ela encontrar algum valor duplicado, retornará um erro e não alterará a tabela com a adição da restrição de unidade. *Entendido?*

Por fim, da mesma forma que é possível adicionar uma restrição de unicidade a uma determinada coluna, é também possível retirá-la por meio da seguinte sintaxe:

SINTAXE DO COMANDO (MYSQL)

```
ALTER TABLE NOME_DA_TABELA
DROP INDEX NOME_DA_RESTRICAO;
```

SINTAXE DO COMANDO (SQL SERVER / ORACLE / MS-ACCESS)

```
ALTER TABLE NOME_DA_TABELA
DROP CONSTRAINT NOME_DA_RESTRICAO;
```

PRIMARY KEY

INCIDÊNCIA EM PROVA: MÉDIA

Essa restrição permite identificar unicamente cada registro de uma tabela, o que fornece uma garantia de exclusividade para uma coluna ou conjunto de colunas. A restrição `PRIMARY KEY` combina as duas restrições analisadas nos tópicos anteriores: `PRIMARY KEY = NOT NULL + UNIQUE`! Em outras palavras, uma coluna que seja definida com a restrição `PRIMARY KEY` necessariamente não poderá receber valores nulos nem repetidos.

Eu vou detalhar novamente para não haver confusão: uma coluna que possua a restrição `UNIQUE` jamais poderá se repetir, mas poderá ser nula; uma coluna que possua a restrição `NOT NULL` jamais poderá ser nula, mas poderá se repetir; uma coluna que possua a restrição `PRIMARY KEY` jamais poderá ser nula e jamais poderá se repetir. Uma tabela poderá ter apenas uma chave primária composta de uma coluna (simples) ou mais colunas (composta). Vamos ver a sintaxe:

SINTAXE DO COMANDO (SQL SERVER / ORACLE / MS-ACCESS)

```
CREATE TABLE NOME_DA_TABELA (
      NOME_COLUNA1              TIPO_DE_DADO       PRIMARY KEY  ,
      NOME_COLUNA2              TIPO_DE_DADO       RESTRIÇÕES   ,
      NOME_COLUNA3              TIPO_DE_DADO       RESTRIÇÕES   ,
      ...
      );
```

SINTAXE DO COMANDO (MYSQL)

```
CREATE TABLE NOME_DA_TABELA (
      NOME_COLUNA1              TIPO_DE_DADO       RESTRIÇÕES   ,
      NOME_COLUNA2              TIPO_DE_DADO       RESTRIÇÕES   ,
      NOME_COLUNA3              TIPO_DE_DADO       RESTRIÇÕES   ,
      ...
      PRIMARY KEY (NOME_COLUNA1)
      );
```

SINTAXE DO COMANDO (MYSQL / SQL SERVER / ORACLE / MS-ACCESS)

```
-- PARA NOMEAR UMA RESTRIÇÃO OU DEFINI-LA PARA MÚLTIPLAS COLUNAS
CREATE TABLE NOME_DA_TABELA (
      NOME_COLUNA1              TIPO_DE_DADO       RESTRIÇÕES   ,
      NOME_COLUNA2              TIPO_DE_DADO       RESTRIÇÕES   ,
      NOME_COLUNA3              TIPO_DE_DADO       RESTRIÇÕES   ,
      ...
      CONSTRAINT NOME_DA_RESTRICAO PRIMARY KEY (NOME_COLUNA1, NOME_COLUNA2)
      );
```

Detalhe importante: na sintaxe apresentada acima, nós temos uma – e apenas uma – chave primária. Trata-se de uma chave primária composta, uma vez que seu valor é composto pelos valores de duas outras colunas. *Entendido?* Vamos seguir: da mesma forma que nós vimos

anteriormente, é possível também adicionar uma restrição PRIMARY KEY a uma tabela pré-existente. Para tal, utiliza-se a seguinte sintaxe:

SINTAXE DO COMANDO (MYSQL / SQL SERVER / ORACLE / MS-ACCESS)

```
ALTER TABLE NOME_DA_TABELA
ADD PRIMARY KEY (NOME_COLUNA1);
```

SINTAXE DO COMANDO (MYSQL / SQL SERVER / ORACLE / MS-ACCESS)

```
-- PARA NOMEAR UMA RESTRIÇÃO OU DEFINI-LA PARA MÚLTIPLAS COLUNAS
ALTER TABLE NOME_DA_TABELA
ADD CONSTRAINT NOME_DA_RESTRICAO PRIMARY KEY (NOME_COLUNA1, NOME_COLUNA2);
```

Por fim, da mesma forma que é possível adicionar uma restrição PRIMARY KEY a uma determinada coluna, é também possível retirá-la por meio da seguinte sintaxe:

SINTAXE DO COMANDO (MYSQL)

```
ALTER TABLE NOME_DA_TABELA
DROP PRIMARY KEY;
```

SINTAXE DO COMANDO (SQL SERVER / ORACLE / MS-ACCESS)

```
ALTER TABLE NOME_DA_TABELA
DROP CONSTRAINT NOME_DA_RESTRICAO;
```

FOREIGN KEY

INCIDÊNCIA EM PROVA: MÉDIA

Agora vamos precisar entrar em um pouco mais de detalhes, portanto vou precisar de mais atenção de vocês! Nós sabemos que um banco de dados não é um conjunto de dados aleatórios – eles são definidos como uma coleção logicamente coerente de dados com algum significado inerente. Logo, os dados armazenados em um banco de dados possuem uma relação entre si dentro de um determinado contexto.

Nós também sabemos que – em um banco de dados relacional – dados são armazenados dentro de tabelas! Logo, as tabelas dentro de um banco de dados relacional precisam se interligar de alguma forma. *E que forma seria essa?* Por meio de chaves estrangeiras! As chaves estrangeiras são utilizadas para unir duas tabelas, em que a chave estrangeira de uma tabela referencia uma chave candidata de outra tabela (em geral, a chave primária).

A restrição `FOREIGN KEY` é utilizada justamente para definir uma ou mais colunas como chaves estrangeiras e prevenir que alguma ação possa destruir essa ligação entre tabelas. A tabela com a chave estrangeira é chamada de Tabela Filha, e a tabela com a chave primária é chamada de Tabela Referenciada ou Tabela Pai. *Entendido?* Pois bem... vamos ver a seguir como é a sintaxe para a definição dessa restrição:

SINTAXE DO COMANDO (SQL SERVER / ORACLE / MS-ACCESS)

```
CREATE TABLE NOME_DA_TABELA (
       NOME_COLUNA1  TIPO_DE_DADO  PRIMARY KEY       ,
       NOME_COLUNA2  TIPO_DE_DADO  RESTRIÇÕES       ,
       NOME_COLUNA3  TIPO_DE_DADO  FOREIGN KEY REFERENCES TABELA_REFERENCIADA (CHAVE)
       );
```

SINTAXE DO COMANDO (MYSQL)

```
CREATE TABLE NOME_DA_TABELA (
       NOME_COLUNA1  TIPO_DE_DADO  RESTRIÇÕES       ,
       NOME_COLUNA2  TIPO_DE_DADO  RESTRIÇÕES       ,
       NOME_COLUNA3  TIPO_DE_DADO  RESTRIÇÕES       ,
       PRIMARY KEY (NOME_COLUNA1)                   ,
       FOREIGN KEY (NOME_COLUNA2) REFERENCES TABELA_REFERENCIADA (CHAVE)
       );
```

SINTAXE DO COMANDO (MYSQL / SQL SERVER / ORACLE / MS-ACCESS)

```
-- PARA NOMEAR UMA RESTRIÇÃO OU DEFINI-LA PARA MÚLTIPLAS COLUNAS
CREATE TABLE NOME_DA_TABELA (
       NOME_COLUNA1  TIPO_DE_DADO  RESTRIÇÕES       ,
       NOME_COLUNA2  TIPO_DE_DADO  RESTRIÇÕES       ,
       NOME_COLUNA3  TIPO_DE_DADO  RESTRIÇÕES       ,
       CONSTRAINT NOME_DA_RESTRICAO FOREIGN KEY (NOME_COLUNA1)
       REFERENCES TABELA_REFERENCIADA (CHAVE)
       );
```

Percebam nas sintaxes apresentadas anteriormente que nós atribuímos a restrição `FOREIGN KEY` a uma coluna da Tabela Filha e definimos à qual coluna ela se refere na Tabela Referenciada. É a coluna de uma tabela se referindo à coluna de outra tabela de modo que uma alteração em uma também afeta a outra. *Viram como isso permite a relação entre tabelas?* Pois é! Bem... é possível adicionar essa restrição a uma tabela pré-existente. Vejamos sua sintaxe:

SINTAXE DO COMANDO (MYSQL / SQL SERVER / ORACLE / MS-ACCESS)

```
ALTER TABLE NOME_DA_TABELA
ADD FOREIGN KEY (NOME_COLUNA1) REFERENCES TABELA_REFERENCIADA (CHAVE);
```

SINTAXE DO COMANDO (MYSQL / SQL SERVER / ORACLE / MS-ACCESS)

```
-- PARA NOMEAR UMA RESTRIÇÃO OU DEFINI-LA PARA MÚLTIPLAS COLUNAS
ALTER TABLE NOME_DA_TABELA
ADD CONSTRAINT NOME_DA_RESTRICAO
FOREIGN KEY (NOME_COLUNA1)
REFERENCES TABELA_REFERENCIADA (CHAVE);
```

Por fim, da mesma forma que é possível adicionar uma restrição `FOREIGN KEY` a uma determinada coluna, é também possível retirá-la por meio da seguinte sintaxe:

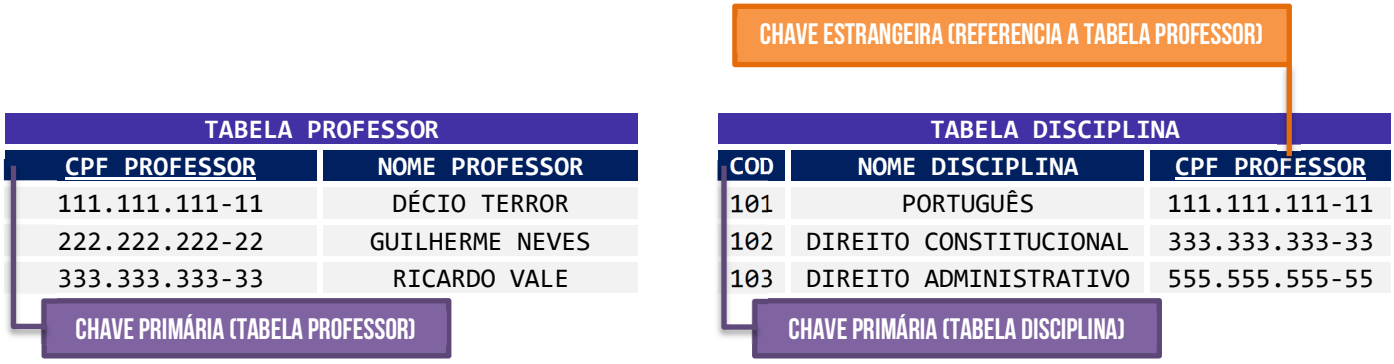
SINTAXE DO COMANDO (MYSQL)

```
ALTER TABLE NOME_DA_TABELA
DROP FOREIGN KEY;
```

SINTAXE DO COMANDO (SQL SERVER / ORACLE / MS-ACCESS)

```
ALTER TABLE NOME_DA_TABELA
DROP CONSTRAINT NOME_DA_RESTRICAO;
```

Agora nós temos um problema bastante interessante: nós sabemos que a chave estrangeira de uma tabela referencia uma chave candidata de outra tabela (em geral, a chave primária). Logo, se algo muda na tabela pai, mudará também na tabela filha. Dito isso, eu te pergunto: *o que aconteceria se eu deletasse um registro da tabela pai?* Ora, todo registro da tabela filha que referenciasse esse registro deletado da tabela pai teria um valor inválido, porque ela perderia a sua referência.



Vejam o exemplo anterior: se excluíssemos o registro da TABELA PROFESSOR cuja chave primária é 111.111.111-11, a disciplina cujo código é 101 da TABELA DISCIPLINA ficaria sem referência. *E como podemos resolver esse problema?* Podemos utilizar a cláusula ON DELETE CASCADE. Essa cláusula basicamente obriga a exclusão dos registros correspondentes das Tabelas Filhas que referenciam o registro excluído da Tabela Pai.

Voltemos ao exemplo: a exclusão do registro da TABELA PROFESSOR cuja chave primária é 111.111.111-11 excluiria também o registro cujo código é 101 da TABELA DISCIPLINA. Sintaxe:

SINTAXE DO COMANDO

```
CREATE TABLE NOME_DA_TABELA (
      NOME_COLUNA1  TIPO_DE_DADO  RESTRIÇÕES       ,
      NOME_COLUNA2  TIPO_DE_DADO  RESTRIÇÕES       ,
      NOME_COLUNA3  TIPO_DE_DADO  RESTRIÇÕES       ,
      ...
      CONSTRAINT NOME_DA_RESTRICAO FOREIGN KEY (NOME_COLUNA1, ...)
      REFERENCES TABELA_REFERENCIADA (CHAVE1, ...)
      ON DELETE CASCADE
      );
```

Existe também a cláusula ON UPDATE, que permite realizar algumas ações quando há uma alteração na tabela pai: CASCADE, RESTRICT, NO ACTION, SET NULL e SET DEFAULT.

CHECK

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

Essa restrição é utilizada para limitar o intervalo de valores que pode ser inserido em uma coluna. É possível defini-la para uma coluna ou para uma tabela. Caso seja definida para uma coluna, ela permitirá apenas alguns valores para esta coluna. Caso seja definida para uma tabela, ela limitará os valores de certas colunas com base nos valores de outras colunas da linha. Vamos ver como tudo isso funciona...

EXEMPLO DO COMANDO (SQL SERVER / ORACLE / MS-ACCESS)

```
CREATE TABLE ALUNO (
      NOME              VARCHAR(20)       NOT NULL          ,
      CPF               INT               PRIMARY KEY       ,
      SEXO              CHAR(1)           NOT NULL          ,
      CIDADE            VARCHAR(50)                         ,
      MATRICULA         INT               UNIQUE            ,
      IDADE             INT               CHECK (IDADE >= 18)
      );
```

EXEMPLO DO COMANDO (MYSQL)

```
CREATE TABLE ALUNO (
      NOME              VARCHAR(20)       NOT NULL          ,
      CPF               INT               PRIMARY KEY       ,
      SEXO              CHAR(1)           NOT NULL          ,
      CIDADE            VARCHAR(50)                         ,
      MATRICULA         INT               UNIQUE            ,
      IDADE             INT                                 ,
      CHECK (IDADE >= 18)
      );
```

EXEMPLO DO COMANDO (MYSQL / SQL SERVER / ORACLE / MS-ACCESS)

```
-- PARA NOMEAR UMA RESTRIÇÃO OU DEFINI-LA PARA MÚLTIPLAS COLUNAS
CREATE TABLE ALUNO (
      NOME              VARCHAR(20)       NOT NULL          ,
      CPF               INT               PRIMARY KEY       ,
      SEXO              CHAR(1)           NOT NULL          ,
      CIDADE            VARCHAR(50)                         ,
      MATRICULA         INT               UNIQUE            ,
      IDADE             INT                                 ,
      CONSTRAINT NOME_DA_RESTRICAO CHECK (IDADE >= 18 AND SEXO = 'F')
      );
```

No último exemplo, temos uma restrição composta, dado que limita a inserção de registros apenas àqueles que tenham `IDADE >= 18` e `SEXO = 'F'`. Em outras palavras, será permitido o armazenamento de registros apenas de mulheres maiores de idade. Conforme vimos nas restrições anteriores, também é possível adicioná-la após a criação da tabela, isto é, em uma tabela pré-existente. Vamos ver como seria a sintaxe:

SINTAXE DO COMANDO (MYSQL / SQL SERVER / ORACLE / MS-ACCESS)

```
ALTER TABLE NOME_DA_TABELA
ADD CHECK (CONDICAO);
```

SINTAXE DO COMANDO (MYSQL / SQL SERVER / ORACLE / MS-ACCESS)

```
-- PARA NOMEAR UMA RESTRIÇÃO OU DEFINI-LA PARA MÚLTIPLAS COLUNAS

ALTER TABLE NOME_DA_TABELA
ADD CONSTRAINT NOME_DA_RESTRICAO CHECK (CONDICAO1 OPERADOR CONDIÇÃO2 ...);
```

Por fim, da mesma forma que é possível adicionar uma restrição de checagem, é também possível retirá-la por meio da seguinte sintaxe:

SINTAXE DO COMANDO (MYSQL)

```
ALTER TABLE NOME_DA_TABELA
DROP CHECK NOME_DA_RESTRICAO;
```

SINTAXE DO COMANDO (SQL SERVER / ORACLE / MS-ACCESS)

```
ALTER TABLE NOME_DA_TABELA
DROP CONSTRAINT NOME_DA_RESTRICAO;
```

DEFAULT

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

Essa restrição é utilizada para configurar um valor padrão para uma coluna. Esse valor padrão é adicionado em todos os novos registros, caso nenhum outro valor tenha sido especificado.

EXEMPLO DO COMANDO (MYSQL / SQL SERVER / ORACLE / MS-ACCESS)

```
CREATE TABLE ALUNO (
      NOME              VARCHAR(20)       NOT NULL          ,
      CPF               INT               PRIMARY KEY       ,
      SEXO              CHAR(1)           NOT NULL          ,
      CIDADE            VARCHAR(50)       DEFAULT 'Brasília' ,
      MATRICULA         INT               UNIQUE            ,
      );
```

No exemplo acima, todo registro que não tenha especificado um valor para a coluna **CIDADE** será automaticamente preenchido com o valor **Brasília** pelo próprio sistema.

SINTAXE DO COMANDO (MYSQL)

```
ALTER TABLE NOME_DA_TABELA
ADD NOME_COLUNA SET DEFAULT VALOR;
```

SINTAXE DO COMANDO (SQL SERVER)

```
ALTER TABLE NOME_DA_TABELA
ADD CONSTRAINT NOME_DA_RESTRICAO DEFAULT VALOR FOR NOME_COLUNA;
```

SINTAXE DO COMANDO (MS-ACCESS)

```
ALTER TABLE NOME_DA_TABELA
ALTER COLUMN NOME_COLUNA SET DEFAULT VALOR;
```

SINTAXE DO COMANDO (ORACLE)

```
ALTER TABLE NOME_DA_TABELA
MODIFY NOME_COLUNA DEFAULT VALOR;
```

Por fim, da mesma forma que é possível adicionar um valor padrão, é também possível retirá-lo por meio da seguinte sintaxe:

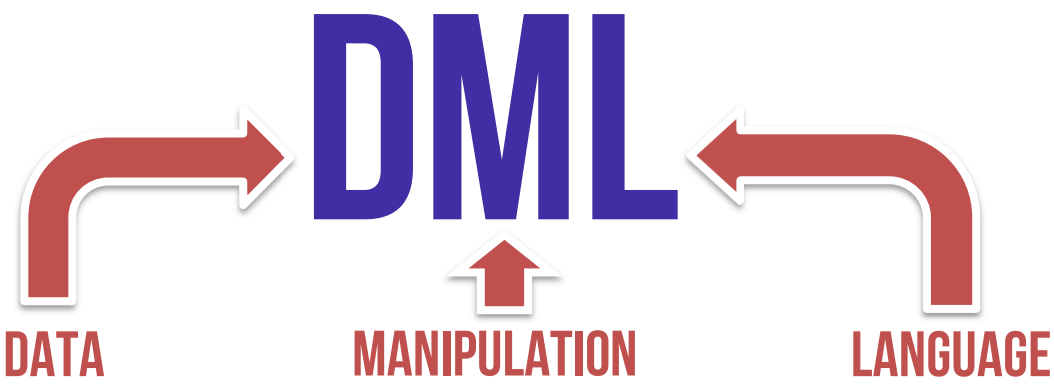
SINTAXE DO COMANDO (MYSQL)

```
ALTER TABLE NOME_DA_TABELA
ALTER NOME_COLUNA DROP DEFAULT;
```

SINTAXE DO COMANDO (SQL SERVER / ORACLE / MS-ACCESS)

```
ALTER TABLE NOME_DA_TABELA
ALTER COLUMN NOME_COLUNA DROP DEFAULT;
```

DML (DATA MANIPULATION LANGUAGE)



Agora chegou o momento de parar de falar um pouco na estrutura de uma tabela e começar a falar um pouco mais sobre como manipular os dados que essa tabela armazenará. *De que adianta uma tabela sem nenhum dado?* De absolutamente nada! Para tal, vamos estudar o Data Manipulation Language (em tradução livre, Linguagem de Manipulação de Dados). *Logo, essa linguagem serve para quê?* Ela serve para manipular dados! *E o que isso significa?*

Isso significa essa linguagem possui um conjunto de comandos que podem ser utilizados para realizar transações em um banco de dados (inserir, excluir, deletar ou consultar):

COMANDOS DML	DESCRIÇÃO
SELECT	Comando utilizado para realizar consultas a dados de uma ou mais tabelas do banco de dados.
INSERT	Comando utilizado para inserir um registro em uma tabela do banco de dados.
UPDATE	Comando utilizado para mudar valores de dados de registros de uma tabela do banco de dados.
DELETE	Comando utilizado para remover registros de uma tabela do banco de dados.

(MPU – 2015) Os comandos SQL INTERT, UPDATE, DELETE e ALTER TABLE fazem parte da DML (Data Manipulation Language).

_____

Comentários: ALTER TABLE também não faz parte da DML, mas da DDL (Errado).

(Banco da Amazônia – 2010) Exemplos de comandos de SQL DML (Data Manipulation Language) incluem SELECT, UPDATE, DELETE, INSERT INTO.

_____

Comentários: perfeito... todos esses são comandos DML (Correto).

Comandos DML

INSERT[4]

INCIDÊNCIA EM PROVA: ALTA

Esse comando é utilizado para inserir novos registros em uma tabela do banco de dados. Há duas formas de inseri-los: completo ou incompleto! *Como é, Diego?* Imagine uma tabela cujos registros contenham seis colunas: se você for inserir um registro por completo, isto é, com valores para todas as colunas, não é necessário indicar o nome das colunas; se for inserir dados para apenas algumas colunas, é necessário indicar quais serão essas colunas e na ordem correta.

SINTAXE DO COMANDO I

```
-- INSERÇÃO DE TODOS OS VALORES PRESCINDE DA ESPECIFICAÇÃO DAS COLUNAS
INSERT INTO NOME_DA_TABELA
VALUES (VALOR_1, VALOR_2, VALOR_3, ...)
```

SINTAXE II DO COMANDO II

```
-- INSERÇÃO DE TODOS OS VALORES PRECISA DA ESPECIFICAÇÃO DAS COLUNAS
INSERT INTO NOME_DA_TABELA (NOME_COLUNA1, NOME_COLUNA2, NOME_COLUNA3, ...)
VALUES (VALOR_1, VALOR_2, VALOR_3, ...)
```

EXEMPLOS DO COMANDO I

```
INSERT INTO ALUNO_ESTRATEGIA
VALUES ('ALICE', '11111111111', 'ALICE@ALICE.COM', '01-01-2001', 'BRASÍLIA', 200.00);

INSERT INTO ALUNO_ESTRATEGIA
VALUES ('BRUNO', '22222222222', 'BRUNO@BRUNO.COM', '02-02-2002', 'SÃO PAULO', 100.00);

INSERT INTO ALUNO_ESTRATEGIA
VALUES ('CAIO' , '33333333333', 'CAIO@CAIO.COM'  , '03-03-2003', 'GOIÂNIA, 150.00);

INSERT INTO ALUNO_ESTRATEGIA
VALUES ('DIEGO', '44444444444', 'DIEGO@DIEGO.COM', '04-04-2004', 'SALVADOR, 250.00);

INSERT INTO ALUNO_ESTRATEGIA
VALUES ('ELIS',  '55555555555', 'ELIS@ELIS.COM'  , '05-05-2005', 'BRASÍLIA', 50.00);
```

EXEMPLOS DO COMANDO II

```
INSERT INTO ALUNO_ESTRATEGIA (NOME, CPF, EMAIL, DATA_NASCIMENTO, CIDADE, VALOR_PAGO)
VALUES ('FABIO', '66666666666', 'FABIO@FABIO.COM', '06-06-2006', 'SALVADOR', 125.00);

INSERT INTO ALUNO_ESTRATEGIA (NOME, CPF, EMAIL, DATA_NASCIMENTO, CIDADE, VALOR_PAGO)
VALUES ('GABI',  '77777777777', 'GABI@GABI.COM'  , '07-07-2007', 'BRASÍLIA', 225.00);

INSERT INTO ALUNO_ESTRATEGIA (NOME, CPF, EMAIL, DATA_NASCIMENTO, CIDADE, VALOR_PAGO)
VALUES ('HUGO',  '88888888888', 'HUGO@HUGO.COM'  , '08-08-2008', 'BRASÍLIA', 50.00);
```

[4] O nome do comando pode ser apenas INSERT ou INSERT INTO.

```
INSERT INTO ALUNO_ESTRATEGIA (NOME, CPF, EMAIL, DATA_NASCIMENTO, CIDADE, VALOR_PAGO)
VALUES ('IGOR',  '99999999999', 'IGOR@IGOR.COM'  , '09-09-2009', 'RECIFE'  , 75.00);

INSERT INTO ALUNO_ESTRATEGIA (NOME, CPF, EMAIL, DATA_NASCIMENTO, CIDADE, VALOR_PAGO)
VALUES ('JOÃO',  '00000000000', 'JOAO@JOAO.COM'  , '10-10-2010', 'NATAL'  , 175.00);
```

RESULTADO DOS COMANDOS

ALUNO_ESTRATEGIA

NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
ALICE	11111111111	ALICE@ALICE.COM	01-01-2001	BRASÍLIA	200.00
BRUNO	22222222222	BRUNO@BRUNO.COM	02-02-2002	SÃO PAULO	100.00
CAIO	33333333333	CAIO@CAIO.COM	03-03-2003	GOIÂNIA	150.00
DIEGO	44444444444	DIEGO@DIEGO.COM	04-04-2004	SALVADOR	250.00
ELIS	55555555555	ELIS@ELIS.COM	05-05-2005	BRASÍLIA	50.00
FABIO	66666666666	FABIO@FABIO.COM	06-06-2006	SALVADOR	125.00
GABI	77777777777	GABI@GABI.COM	07-07-2007	BRASÍLIA	225.00
HUGO	88888888888	HUGO@HUGO.COM	08-08-2008	BRASÍLIA	50.00
IGOR	99999999999	IGOR@IGOR.COM	09-09-2009	RECIFE	75.00
JOÃO	00000000000	JOAO@JOAO.COM	10-10-2010	NATAL	175.00

(TRE/RJ – 2012) INSERT INTO é o comando utilizado para inserir dados em uma tabela.

_____

Comentários: esse comando é realmente utilizado para inserir dados em uma tabela (Correto).

(Prefeitura de Congonhas/MG – 2010) Qual instrução SQL é usada para inserir novos dados em uma tabela do banco de dados?

a) ADD NEW b) ADD RECORD c) INSERT INTO d) INSERT NEW e) INSERT

_____

Comentários: para inserir novos dados em uma tabela do banco de dados, utiliza-se o comando INSERT INTO – sendo que algumas bancas considerariam também apenas o comando INSERT (Letra C).

(STJ – 2008) O comando INSERT INTO é capaz de inserir novos dados em um banco de dados, mas não é classificado como DML nem como DDL.

_____

Comentários: esse comando é classificado como DML (Errado).

(TRE/AL – 2004) O comando INSERT INTO FUNC (coluna1, coluna2) DATA (dado1, dado2) está sintaticamente correto e permite realizar a inserção de um conjunto de dados em uma tabela denominada FUNC.

_____

Comentários: está quase perfeito – exceto pelo DATA, onde deveria ser VALUES (Errado).

UPDATE

INCIDÊNCIA EM PROVA: ALTA

Esse comando é utilizado para atualizar registros existentes em uma tabela do banco de dados. Pode-se atualizar todos os registros de uma tabela ou apenas alguns. Para atualizar registros específicos, devemos utilizar a cláusula **WHERE**. Essa cláusula será detalhada mais à frente, mas – por enquanto – basta saber que ela permite filtrar dados a partir de um conjunto de condições. Vamos conhecer a sintaxe desse comando:

SINTAXE DO COMANDO

```
UPDATE NOME_DA_TABELA
SET NOME_DA_COLUNA_1 = VALOR_1, NOME_COLUNA2 = VALOR_2 ...
WHERE LISTA_DE_CONDICOES
```

EXEMPLO DO COMANDO

```
UPDATE ALUNO_ESTRATEGIA
SET NOME = 'DIOGO', EMAIL = 'DIOGO@GMAIL.COM'
WHERE CPF = 44444444444
```

RESULTADO DOS COMANDOS

ALUNO_ESTRATEGIA					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
ALICE	11111111111	ALICE@ALICE.COM	01-01-2001	BRASÍLIA	200.00
BRUNO	22222222222	BRUNO@BRUNO.COM	02-02-2002	SÃO PAULO	100.00
CAIO	33333333333	CAIO@CAIO.COM	03-03-2003	GOIÂNIA	150.00
DIOGO	44444444444	DIOGO@DIOGO.COM	04-04-2004	SALVADOR	250.00
ELIS	55555555555	ELIS@ELIS.COM	05-05-2005	BRASÍLIA	50.00
FABIO	66666666666	FABIO@FABIO.COM	06-06-2006	SALVADOR	125.00
GABI	77777777777	GABI@GABI.COM	07-07-2007	BRASÍLIA	225.00
HUGO	88888888888	HUGO@HUGO.COM	08-08-2008	BRASÍLIA	50.00
IGOR	99999999999	IGOR@IGOR.COM	09-09-2009	RECIFE	75.00
JOÃO	00000000000	JOAO@JOAO.COM	10-10-2010	NATAL	175.00

Vamos supor que houvesse um erro e o nome de um aluno fosse Diogo em vez de Diego. Vejam no exemplo acima que o código apresentado atualiza (**UPDATE**) a tabela **ALUNO_ESTRATEGIA** de tal forma que se configure (**SET**) o valor da coluna **NOME** para '**Diogo**' e o valor da coluna **EMAIL** para '**diogo@gmail.com**', porém apenas para os registros que tenham **44444444444** como valor de **CPF**. Essa foi uma condição simples, mas você pode inserir a condição que quiser...

(TCE/SP – 2010) Alterações nos valores dos registros de determinada tabela são realizadas em SQL pelo comando:

a) Insert. b) Update. c) Drop. d) Modify. e) Modify_table.

Comentários: o comando utilizado para alterar valores em registros de uma tabela é o UPDATE (Letra B).

DELETE

INCIDÊNCIA EM PROVA: ALTA

Esse comando é utilizado para deletar registros existentes em uma tabela do banco de dados. Pode-se deletar todos os registros de uma tabela ou apenas alguns. Para deletar registros específicos, devemos utilizar a cláusula **WHERE**. Notem que se não for utilizada a cláusula **WHERE**, o **DELETE** funcionará como um **TRUNCATE**. *Vocês se lembram o que esse comando faz?* Ele deleta todos os registros de uma tabela (mas mantém a estrutura).

No exemplo seguinte, nós vamos excluir da tabela **ALUNO_ESTRATEGIA** as linhas/registros cujo valor da coluna **VALOR_PAGO** seja **175.00** ou cujo valor da coluna **CIDADE** seja **RECIFE**.

SINTAXE DO COMANDO

```
DELETE FROM NOME_DA_TABELA WHERE LISTA_DE_CONDICOES
```

EXEMPLO DO COMANDO

```
DELETE FROM ALUNO_ESTRATEGIA WHERE VALOR_PAGO = 175.00 OR CIDADE = 'RECIFE';
```

RESULTADO DO COMANDO

ALUNO_ESTRATEGIA					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
ALICE	11111111111	ALICE@ALICE.COM	01-01-2001	BRASÍLIA	200.00
BRUNO	22222222222	BRUNO@BRUNO.COM	02-02-2002	SÃO PAULO	100.00
CAIO	33333333333	CAIO@CAIO.COM	03-03-2003	GOIÂNIA	150.00
DIOGO	44444444444	DIOGO@DIOGO.COM	04-04-2004	SALVADOR	250.00
ELIS	55555555555	ELIS@ELIS.COM	05-05-2005	BRASÍLIA	50.00
FABIO	66666666666	FABIO@FABIO.COM	06-06-2006	SALVADOR	125.00
GABI	77777777777	GABI@GABI.COM	07-07-2007	BRASÍLIA	225.00
HUGO	88888888888	HUGO@HUGO.COM	08-08-2008	BRASÍLIA	50.00

(EBSERH – 2013) Em SQL a instrução que remove uma ou mais linhas em uma tabela:

a) ERASE b) DELETE c) DROP d) REMOVE

_____

Comentários: a instrução/comando utilizada para remover uma ou mais linhas em uma tabela é o DELETE (Letra B).

(IF/TO – 2018) José, técnico em informática do IFTO, deseja excluir somente um registro de um banco de dados SQL. Qual dos comandos abaixo ele deverá usar:

a) TRUNCATE b) DROP c) DELETE d) INSERT e) COMMIT

_____

Comentários: o comando que deve ser utilizado pelo José para excluir um registro do banco de dados é o DELETE (Letra C).

SELECT

INCIDÊNCIA EM PROVA: ALTÍSSIMA

Esse comando é utilizado para recuperar informações de um banco de dados. *Sabe quando você acessa o site do Estratégia Concursos e faz uma busca?* Você informa algum parâmetro de entrada (Ex: nome de uma disciplina) e o sistema te retorna um resultado de saída (Ex: cursos)! Por trás de tudo isso, existe um banco de dados – e a consulta que você faz é realizada por meio de um SELECT. Na prática, o que você está fazendo é uma seleção de registros da base de dados.

Vocês se lembram que nós estudamos álgebra relacional? Pois é! Nós vimos diversas operações, dentre as quais: projeção, produto cartesiano e seleção. Agora tomem muito cuidado para não confundir a cláusula **SELECT** com a operação de seleção da álgebra relacional. Na verdade, essas três operações podem ser mapeadas para suas cláusulas correspondentes como **SELECT**, **FROM** e **WHERE** respectivamente. *Bacana?* Agora vamos ver como é a sua sintaxe...

SINTAXE DO COMANDO

```
-- AS CLÁUSULAS SÃO OPCIONAIS
SELECT LISTA_DE_COLUNAS FROM LISTA_DE_TABELAS CLAUSULAS;
```

EXEMPLO DO COMANDO

```
SELECT * FROM ALUNO_ESTRATEGIA;
```

RESULTADO DO COMANDO

RESULTADO					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
ALICE	11111111111	ALICE@ALICE.COM	01-01-2001	BRASÍLIA	200.00
BRUNO	22222222222	BRUNO@BRUNO.COM	02-02-2002	SÃO PAULO	100.00
CAIO	33333333333	CAIO@CAIO.COM	03-03-2003	GOIÂNIA	150.00
DIOGO	44444444444	DIOGO@DIOGO.COM	04-04-2004	SALVADOR	250.00
ELIS	55555555555	ELIS@ELIS.COM	05-05-2005	BRASÍLIA	50.00
FABIO	66666666666	FABIO@FABIO.COM	06-06-2006	SALVADOR	125.00
GABI	77777777777	GABI@GABI.COM	07-07-2007	BRASÍLIA	225.00
HUGO	88888888888	HUGO@HUGO.COM	08-08-2008	BRASÍLIA	50.00

Observem no exemplo acima que utilizamos o elemento asterisco (*). *O que isso significa, professor?* Galera, isso significa que queremos recuperar todas as colunas da tabela. Aliás, sempre que vocês virem esse elemento, vocês podem interpretá-lo como "todas as colunas". Vejam no

esquema abaixo o exemplo de comando e a sua respectiva interpretação para que vocês consigam entender isso melhor:

EXEMPLO DE COMANDO			
SELECT	*	FROM	ALUNO_ESTRATEGIA
↓	↓	↓	↓
INTERPRETAÇÃO			
SELECIONE	TODAS AS COLUNAS	DA TABELA	ALUNO_ESTRATEGIA

Professor, mas e se eu quiser selecionar colunas específicas da tabela? Não há problema: nós podemos selecionar, por exemplo, as colunas **NOME** e **DATA_NASCIMENTO**:

EXEMPLO DO COMANDO

```
SELECT NOME, DATA_NASCIMENTO FROM ALUNO_ESTRATEGIA;
```

RESULTADO DO COMANDO

RESULTADO	
NOME	DATA_NASCIMENTO
ALICE	01-01-2001
BRUNO	02-02-2002
CAIO	03-03-2003
DIOGO	04-04-2004
ELIS	05-05-2005
FABIO	06-06-2006
GABI	07-07-2007
HUGO	08-08-2008

COMANDO			
SELECT	NOME, DATA_NASCIMENTO	FROM	ALUNO_ESTRATEGIA
↓	↓	↓	↓
INTERPRETAÇÃO			
SELECIONE	AS COLUNAS NOME E DATA_NASCIMENTO	DA TABELA	ALUNO_ESTRATEGIA

Galera, existem situações em que não é desejável o retorno de valores repetidos de uma coluna do banco de dados. Por exemplo: na tabela original, nós tínhamos uma coluna **CIDADE**. *Vocês notaram que alguns valores se repetem?* Esses valores repetidos são também conhecidos como duplicatas! Se eu desejo saber de quais cidades são os alunos cadastrados na tabela, não é relevante para mim ter valores repetidos – eu só quero saber quais são as cidades. *Como assim, Diego?*

ALUNO_ESTRATEGIA					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
ALICE	11111111111	ALICE@ALICE.COM	01-01-2001	BRASÍLIA	200.00
BRUNO	22222222222	BRUNO@BRUNO.COM	02-02-2002	SÃO PAULO	100.00
CAIO	33333333333	CAIO@CAIO.COM	03-03-2003	GOIÂNIA	150.00
DIOGO	44444444444	DIOGO@DIOGO.COM	04-04-2004	SALVADOR	250.00

ELIS	55555555555	ELIS@ELIS.COM	05-05-2005	BRASÍLIA	50.00
FABIO	66666666666	FABIO@FABIO.COM	06-06-2006	SALVADOR	125.00
GABI	77777777777	GABI@GABI.COM	07-07-2007	BRASÍLIA	225.00
HUGO	88888888888	HUGO@HUGO.COM	08-08-2008	BRASÍLIA	50.00

Notem que essa é a nossa tabela original de referência! Eu poderia selecionar a coluna cidade da tabela por meio do seguinte comando apresentado a seguir:

EXEMPLO DO COMANDO

```
SELECT CIDADE FROM ALUNO_ESTRATEGIA;
```

RESULTADO DO COMANDO

RESULTADO
CIDADE
BRASÍLIA
SÃO PAULO
GOIÂNIA
SALVADOR
BRASÍLIA
SALVADOR
BRASÍLIA
BRASÍLIA

Ora, o resultado retornou cheio de duplicatas! *Já imaginaram se essa tabela contivesse 1000 registros?* Ficaria inviável! Eu quero saber apenas quais são as cidades – sem repetições. Dito isso, existe uma palavra-chave que pode ser utilizada junto com o comando **SELECT** que tem a função justamente de eliminar os registros duplicados. Essa palavra-chave se chama **DINSTINCT**! Vamos ver como é a sua sintaxe:

SINTAXE DO COMANDO

```
-- AS CLÁUSULAS SÃO OPCIONAIS
SELECT DISTINCT LISTA_DE_COLUNAS FROM LISTA_DE_TABELAS CLAUSULAS;
```

EXEMPLO DO COMANDO

```
SELECT DISTINCT CIDADE FROM ALUNO_ESTRATEGIA;
```

RESULTADO DO COMANDO

RESULTADO
CIDADE
BRASÍLIA
SÃO PAULO
GOIÂNIA
SALVADOR

Notem que os registro duplicados foram eliminados porque essa palavra-chave ajuda a selecionar apenas registros distintos (**DISTINCT**). Além dessa palavra-chave, é importante falar sobre os famosos Alias. *O que seria isso, Diego?* Trata-se de um recurso utilizado para dar a uma tabela (ou a uma coluna de uma tabela) um nome temporário – como se fosse um apelido. Em geral, eles são utilizados para tornar os nomes das colunas mais legíveis.

Um alias existe apenas para a duração de uma determinada consulta e é criado por meio da palavra-chave **AS** (que pode ser omitida). Vamos ver como é a sua sintaxe:

SINTAXE DO COMANDO

```
-- ALIAS PARA O NOME DA TABELA
SELECT NOME_COLUNA FROM NOME_DA_TABELA AS APELIDO CLAUSULAS;

-- ALIAS PARA O NOME DA COLUNA
SELECT NOME_COLUNA AS APELIDO FROM NOME_DA_TABELA CLAUSULAS;
```

EXEMPLO DO COMANDO

```
SELECT NOME AS N, DATA_NASCIMENTO AS DN FROM ALUNO_ESTRATEGIA AS AE;
```

Vejam que eu dei apelidos pequenos, o que pode facilitar bastante! Em geral, esse recurso é muito útil quando existe mais de uma tabela envolvida em uma consulta; quando funções são utilizadas dentro de uma consulta; quando nomes de colunas são muito grandes ou pouco intuitivos; e quando duas ou mais colunas são combinadas em uma só. Bem, tudo isso fará mais sentido para vocês no decorrer da aula. *Bacana?* Mudando de assunto...

Uma outra possibilidade é a inserção de constantes em vez de nomes de tabelas. *Como assim, Diego?* Simples, eu posso pedir para a consulta retornar uma constante para cada registro da tabela:

EXEMPLO DO COMANDO

```
SELECT 1 AS RESULTADO FROM ALUNO_ESTRATEGIA;
```

RESULTADO DO COMANDO

RESULTADO
1
1
1
1
1
1
1
1

EXEMPLO DO COMANDO

```
SELECT 'FLAMENGO É O MELHOR' RESULTADO FROM ALUNO_ESTRATEGIA;
```

RESULTADO DO COMANDO

RESULTADO
FLAMENGO É O MELHOR
FLAMENGO É O MELHOR
FLAMENGO É O MELHOR
FLAMENGO É O MELHOR
FLAMENGO É O MELHOR
FLAMENGO É O MELHOR
FLAMENGO É O MELHOR
FLAMENGO É O MELHOR

Por fim, eu decidi deixar um tópico separado para as cláusulas. *Por que, Diego?* Primeiro, porque elas são importantíssimas e despencam em prova; segundo porque algumas delas são transversais e podem ser utilizadas em vários comandos de manipulação de dados (Ex: `FROM` e `WHERE`). Levantem-se, deem uma espreguiçada boa, tomem um café e vamos continuar porque ainda tem muita teoria pela frente. A seguir, uma lista das cláusulas que vamos estudar...

(UFFS – 2014) Qual alternativa informa uma seleção de todas as colunas da tabela USUARIO no banco de dados?

a) SELECT % FROM USUARIO
b) SELECT *.* FROM USUARIO
c) SELECT @ FROM USUARIO
d) SELECT # FROM USUARIO
e) SELECT * FROM USUARIO

_____

Comentários: para selecionar todas as colunas da tabela, deve-se utilizar o asterisco: SELECT * FROM USUÁRIO (Letra E).

(ALGÁS – 2012) Os operadores de Seleção, Projeção e Produto Cartesiano da álgebra relacional são implementados, respectivamente, através das seguintes cláusulas SQL:

a) Select, From e Where, respectivamente.
b) Where, Select e From, respectivamente.
c) Where, From e Select, respectivamente.
d) Select, From e Product, respectivamente.
e) Where, Select e Join, respectivamente.

_____

Comentários: as cláusulas para esses operadores são o WHERE, SELECT e FROM, respectivamente (Letra B).

(UFPA – 2018) O comando SQL utilizado para exibir dados sem repetição em um banco de dados é o:

a) SELECT DISTINCT.
b) SELECT INDIVIDUAL.
c) EXTRACT ONE.
d) EXTRACT ONLYDIFFERENT.
e) RESUME SINGLE.

_____

Comentários: o comando utilizado para exibir dados em repetição é o SELECT DISTINCT (Letra A).

Cláusulas

FROM

INCIDÊNCIA EM PROVA: ALTÍSSIMA

Nós já estudamos essa cláusula e sabemos que ela especifica de onde (quais tabelas) devemos selecionar ou excluir dados – aqui vamos nos focar apenas na seleção de dados. Vejamos:

SINTAXE DO COMANDO

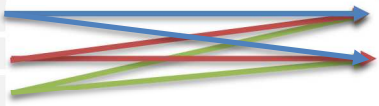
```
SELECT LISTA_DE_COLUNAS FROM TABELA1, TABELA2, ... CLAUSULAS;
```

Observem que é possível especificar mais de uma tabela separada por vírgula! Quando isso ocorre, temos um Produto Cartesiano[5]. Vejamos um exemplo:

EXEMPLO DO COMANDO

```
SELECT * FROM TABELA_PROFESSOR, TABELA_DISCIPLINA;
```

TABELA_PROFESSOR	
NOME_PROFESSOR	CPF
DIEGO CARVALHO	111.111.111-11
RENATO DA COSTA	222.222.222-22
RICARDO VALE	333.333.333-33



TABELA_DISCIPLINA	
NOME_DISCIPLINA	CÓDIGO
INFORMÁTICA	101
DIREITO CONSTITUCIONAL	102

RESULTADO DO COMANDO

RESULTADO			
NOME PROFESSOR	CPF	NOME DISCIPLINA	CÓDIGO
DIEGO CARVALHO	111.111.111-11	INFORMÁTICA	101
RENATO DA COSTA	222.222.222-22	INFORMÁTICA	101
RICARDO VALE	333.333.333-33	INFORMÁTICA	101
DIEGO CARVALHO	111.111.111-11	DIREITO CONSTITUCIONAL	102
RENATO DA COSTA	222.222.222-22	DIREITO CONSTITUCIONAL	102
RICARDO VALE	333.333.333-33	DIREITO CONSTITUCIONAL	102

Notem que as colunas da tabela resultante é basicamente a união das colunas das tabelas especificadas, uma vez que utilizamos o asterisco (*). Já linhas da tabela resultante é basicamente uma combinação de todas as linhas de uma tabela com todas as linhas de outra. Chama-se produto cartesiano justamente porque o resultado é um produto, isto é, o número de linhas de uma tabela (3) vezes o número de linhas de outra tabela (2) retorna uma tabela resultante com 3x2 = 6 linhas.

[5] Há quem considere o Produto Cartesiano como um tipo de Join chamado Cross Join.

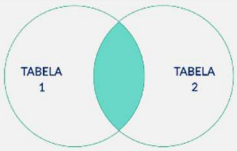
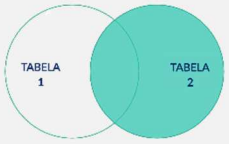
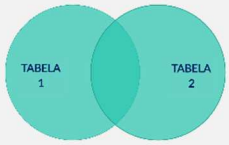
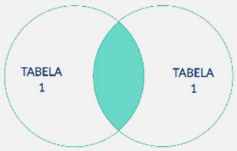
Agora vejam como as coisas se relacionam: quando fazemos combinações entre tabelas, os apelidos (**alias**) – que estudamos algumas páginas atrás – se tornam extremamente úteis. *Por que?* Porque eventualmente podemos fazer o produto de duas ou mais tabelas que possuem atributos com o mesmo nome. *Como assim, Diego?* Ora, imagine o produto cartesiano entre uma tabela chamada `PRODUTO` e outra tabela chamada `EMPRESA`.

Não vou entrar em detalhes, mas imagine que ambas as tabelas possuem uma coluna chamada `CÓDIGO`. Nesse caso, a tabela resultante teria duas colunas com o mesmo nome. Para evitar esse tipo de problema e reduzir a ambiguidade, utilizam-se os alias. Nós podemos chamar a tabela `PRODUTO` de `P` e a tabela `EMPRESA` de `E`. Dessa forma, a tabela resultante teria uma coluna `P.CODIGO` e outra coluna chamada `E.CODIGO`. E assim não temos mais ambiguidade...

JOIN

INCIDÊNCIA EM PROVA: ALTA

Essa cláusula é utilizada para combinar linhas/registros de duas ou mais tabelas, com base em uma coluna em comum entre elas. Elas podem ser de cinco tipos diferentes:

INNER JOIN	LEFT JOIN	RIGHT JOIN	FULL OUTER JOIN	SELF JOIN
				
Retorna registros que possuem valores correspondentes em ambas as tabelas	Retorna todos os registros da tabela da esquerda e seus correspondentes da tabela da direita	Retorna todos os registros da tabela da direita e seus correspondentes da tabela da esquerda	Retorna todos os registros quando há uma correspondência na tabela da esquerda ou da direita	Trata-se de join comum, mas que relaciona registros de uma tabela com ela mesma

O **INNER JOIN** (também chamado apenas de **JOIN**) é uma cláusula que seleciona registros que contenham valores correspondentes em ambas as tabelas. Vejamos sua sintaxe[6]:

SINTAXE DO COMANDO

```
SELECT NOME_DAS_COLUNAS
FROM TABELA1
INNER JOIN TABELA2
ON TABELA1.NOME_COLUNA = TABELA2.NOME_COLUNA;
```

[6] Lembrando que o comando permite utilizar INNER JOIN ou apenas JOIN.

PEDIDOS				
ID_PEDIDO	ID_CLIENTE	ID_FUNCIONARIO	DATA	ID_ENTREGADOR
10308	2	7	18/09/1996	3
10309	37	3	19/09/1996	1
10310	77	8	20/09/1996	2

CLIENTES						
ID_CLIENTE	NOME_CLIENTE	ID_INDICACAO	ENDERECO	CIDADE	CEP	PAIS
1	Alfredo	2	Rua X, 58	Berlin	70.000-00	Alemanha
2	Ana	3	Rua Y, 72	Miami	71.000-00	EUA
3	Antonio		Rua Z, 94	Tijuana	72.000-00	Mexico

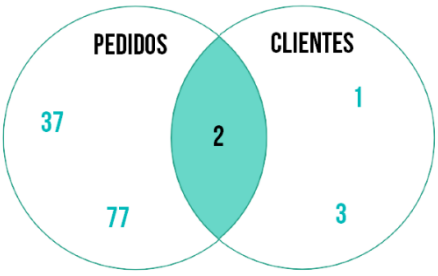
EXEMPLO DO COMANDO

```
SELECT PEDIDOS.ID_PEDIDO, CLIENTES.NOME_CLIENTE
FROM PEDIDOS
INNER JOIN CLIENTES
ON PEDIDOS.ID_CLIENTE = CLIENTES.ID_CLIENTE;
```

O que esse comando está nos dizendo? Ele está nos dizendo para selecionar (**SELECT**) todas as linhas de ambas as tabelas (**FROM PEDIDOS INNER JOIN CLIENTES**) desde que exista uma correspondência entre as colunas (**ON PEDIDOS.ID_CLIENTE = CLIENTES.ID_CLIENTE**) e, após isso, retornar as colunas **PEDIDOS.ID_PEDIDO** e **CLIENTES.NOME_CLIENTE**. O resultado é apresentado na tabela a seguir:

RESULTADO DO COMANDO

RESULTADO	
ID_PEDIDO	NOME_CLIENTE
10308	Ana



Vamos entender melhor? Note que a coluna em comum é **ID_CLIENTE**. A tabela **PEDIDOS** possui três valores para essa coluna: [**2**, 37, 77]; já a tabela **CLIENTES** também possui três valores para essa coluna: [1, **2**, 3]. Como se trata de um **INNER JOIN**, a tabela resultante retornará apenas os registros que possuem correspondência em ambas as tabelas. *Qual é o valor comum entre as tabelas?* É o **2**, que corresponde à cliente Ana.

Lembrando também que é possível fazer um `INNER JOIN` com mais de duas tabelas, conforme mostra o exemplo seguinte (com três tabelas):

EXEMPLO DO COMANDO

```
SELECT PEDIDOS.ID_PEDIDO, CLIENTES.NOME_CLIENTE, ENTREGADORES.NOME_ENTREGADOR
FROM ((PEDIDOS
INNER JOIN CLIENTES ON PEDIDOS.ID_CLIENTE = CLIENTES.ID_CLIENTE)
INNER JOIN ENTREGADORES ON PEDIDOS.ID_ENTREGADOR = ENTREGADORES.ID_ENTREGADOR);
```

Nós vimos que as tabelas que compõem um `INNER JOIN` devem possuir uma coluna em comum. Em geral, essas colunas em comum possuem o mesmo nome conforme vimos nos exemplos. No entanto, isso não é obrigatório – elas podem ter conteúdos correspondentes, mas nomes distintos. De toda forma, caso elas tenham o mesmo nome, é possível utilizar a palavra-chave `USING` para melhorar a leitura do código e sua compreensão. Vejamos como seria sua sintaxe:

SINTAXE DO COMANDO

```
SELECT LISTA_DE_COLUNAS
FROM TABELA1
INNER JOIN TABELA2
USING (NOME_COLUNA_COMUM);
```

EXEMPLO DO COMANDO

```
SELECT PEDIDOS.ID_PEDIDO, CLIENTES.NOME_CLIENTE
FROM PEDIDOS
INNER JOIN CLIENTES
USING (ID_CLIENTE);
```

Já o **`LEFT JOIN`** (também chamado de **`LEFT OUTER JOIN`**) retorna todos os registros da tabela da esquerda, além dos registros correspondentes da tabela da direita. Vejamos a sintaxe:

SINTAXE DO COMANDO

```
SELECT NOME_DAS_COLUNAS
FROM TABELA1
LEFT JOIN TABELA2
ON TABELA1.NOME_COLUNA = TABELA2.NOME_COLUNA;
```

EXEMPLO DO COMANDO

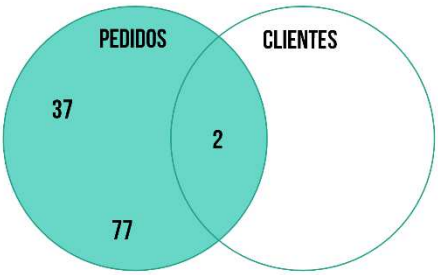
```
SELECT PEDIDOS.ID_PEDIDO, CLIENTES.NOME_CLIENTE
FROM PEDIDOS
LEFT JOIN CLIENTES
ON PEDIDOS.ID_CLIENTE = CLIENTES.ID_CLIENTE;
```

O que esse comando está nos dizendo? Ele está nos dizendo para selecionar (**`SELECT`**) todas as linhas da tabela da esquerda (**`FROM PEDIDOS LEFT JOIN`**), além dos registros da tabela da direita (**`CLIENTES`**) desde que exista uma correspondência entre as colunas (**`ON PEDIDOS.ID_CLIENTE = CLIENTES.ID_CLIENTE`**) e, após isso, retornar as colunas **`PEDIDOS.ID_PEDIDO`** e **`CLIENTES.NOME_CLIENTE`**. O resultado é apresentado na tabela a seguir:

RESULTADO DO COMANDO

RESULTADO	
ID_PEDIDO	NOME_CLIENTE

10308	Ana
10309	NULL
10310	NULL



Vamos entender melhor? Note que a coluna em comum é **ID_CLIENTE**. A tabela **PEDIDOS** possui três valores para essa coluna: [**2**, **37**, **77**]; já a tabela **CLIENTES** também possui três valores para essa coluna: [1, **2**, 3]. Como se trata de um **LEFT JOIN**, a tabela resultante retornará todos os registros da tabela da esquerda e seus valores correspondentes da tabela da direita (se houver). *Entendido?*

Já o **RIGHT JOIN** (também chamado de **RIGHT OUTER JOIN**) retorna todos os registros da tabela da direita, além dos registros correspondentes da tabela da esquerda. Vejamos a sintaxe:

SINTAXE DO COMANDO

```
SELECT NOME_DAS_COLUNAS
FROM TABELA1
RIGHT JOIN TABELA2
ON TABELA1.NOME_COLUNA = TABELA2.NOME_COLUNA;
```

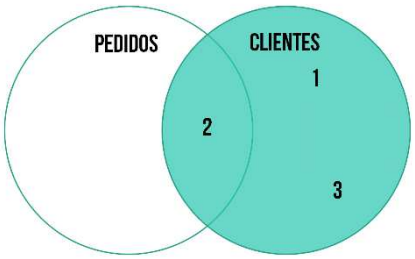
EXEMPLO DO COMANDO

```
SELECT PEDIDOS.ID_PEDIDO, CLIENTES.NOME_CLIENTE
FROM PEDIDOS
RIGHT JOIN CLIENTES
ON PEDIDOS.ID_CLIENTE = CLIENTES.ID_CLIENTE;
```

O que esse comando está nos dizendo? Ele está nos dizendo para selecionar (**SELECT**) todas as linhas da tabela da direita (**CLIENTES**), além dos registros da tabela da esquerda (**FROM PEDIDOS RIGHT JOIN**) desde que exista uma correspondência entre as colunas (**ON PEDIDOS.ID_CLIENTE = CLIENTES.ID_CLIENTE**) e, após isso, retornar as colunas **PEDIDOS.ID_PEDIDO** e **CLIENTES.NOME_CLIENTE**. O resultado é apresentado na tabela a seguir:

RESULTADO DO COMANDO

RESULTADO	
ID_PEDIDO	**NOME_CLIENTE**
NULL	Alfredo
10308	Ana
NULL	Antonio



Vamos entender melhor? Note que a coluna em comum é **ID_CLIENTE**. A tabela **PEDIDOS** possui três valores para essa coluna: [**2**, 37, 77]; já a tabela **CLIENTES** também possui três valores para essa coluna: [**1**, **2**, **3**]. Como se trata de um **RIGHT JOIN**, a tabela resultante retornará todos os registros da tabela da direita e seus valores correspondentes da tabela da esquerda (se houver).

Já o **FULL JOIN** (também chamado de **FULL OUTER JOIN**) retorna todos os registros quando há uma correspondência da tabela esquerda com a direita ou da direita com a esquerda. Vejamos a sintaxe:

SINTAXE DO COMANDO

```
SELECT NOME_DAS_COLUNAS
FROM TABELA1
FULL OUTER JOIN TABELA2
ON TABELA1.NOME_COLUNA = TABELA2.NOME_COLUNA;
```

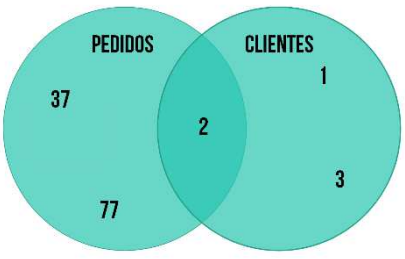
EXEMPLO DO COMANDO

```
SELECT PEDIDOS.ID_PEDIDO, CLIENTES.NOME_CLIENTE
FROM PEDIDOS
FULL OUTER JOIN CLIENTES
ON PEDIDOS.ID_CLIENTE = CLIENTES.ID_CLIENTE;
```

O que esse comando está nos dizendo? Ele está nos dizendo para selecionar (**SELECT**) todas as linhas da tabela da direita e da esquerda (**FROM PEDIDOS FULL OUTER JOIN CLIENTES**) desde que exista uma correspondência entre as colunas (**ON PEDIDOS.ID_CLIENTE = CLIENTES.ID_CLIENTE**) e, após isso, retornar as colunas **PEDIDOS.ID_PEDIDO** e **CLIENTES.NOME_CLIENTE**. O resultado é apresentado na tabela a seguir:

RESULTADO DO COMANDO

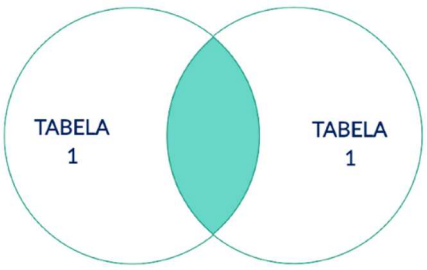
RESULTADO	
ID_PEDIDO	**NOME_CLIENTE**
10309	NULL
10308	Ana
10310	NULL
NULL	Alfredo
NULL	Antonio



Vamos entender melhor? Note que a coluna em comum é **ID_CLIENTE**. A tabela **PEDIDOS** possui três valores para essa coluna: [2, 37, 77]; já a tabela **CLIENTES** também possui três valores para essa coluna: [1, 2, 3]. Como se trata de um **FULL OUTER JOIN**, a tabela resultante retornará todos os registros da tabela da direita e da esquerda e seus valores correspondentes (se houver).

Por fim, o **SELF JOIN** é um join comum, mas que relaciona uma tabela consigo mesma. Imaginem um auto-relacionamento, isto é, uma tabela que se relaciona com ela mesma. Por exemplo:

CLIENTES						
ID_CLIENTE	NOME_CLIENTE	ID_INDICACAO	ENDERECO	CIDADE	CEP	PAIS
1	Alfredo	2	Rua X, 58	Berlin	70.000-00	Alemanha
2	Ana	3	Rua Y, 72	Miami	71.000-00	EUA
3	Antonio		Rua Z, 94	Tijuana	72.000-00	Mexico



Deem uma olhadinha para as três primeiras colunas da tabela e observem que a tabela de clientes armazena dados sobre quem foi o cliente que realizou a indicação. É possível concluir, por exemplo, que Alfredo (1) foi indicado por Ana (2), Ana (2) foi indicada por Antonio (3) e Antonio (3) não foi indicado por ninguém (NULL). Nesse caso, o SELF JOIN retornará quem foi o cliente que indicou outro cliente. Vamos ver um exemplo:

EXEMPLO DO COMANDO

```
SELECT C1.NOME_CLIENTE AS CLIENTE_INDICADOR, C2.NOME_CLIENTE AS CLIENTE_INDICADO
FROM CLIENTES C1
JOIN CLIENTES C2
ON C1.ID_CLIENTE = C2.ID_INDICACAO;
```

O resultado é apresentado na tabela a seguir:

RESULTADO DO COMANDO

RESULTADO	
CLIENTE_INDICADOR	CLIENTE_INDICADO
Ana	Alfredo
Antonio	Ana

(INMETRO – 2015) A operação de consulta de várias tabelas numa mesma operação recebe o nome de join e essa recuperação de informação é feita com base nos relacionamentos estabelecidos. No Oracle isso também é possível de ser feito. Vários tipos de joins podem ser implementados. Implementar a seleção de dados por autorrelacionamento, ou seja, quando se deseja relacionar uma tabela com ela mesma, trata-se do join:

a) Equijoin.
b) Self join.
c) Inner join.
d) Outer join.

e) NonEquijoin.

_____

Comentários: o tipo de join que implementa um autorrelacionamento é o Self Join (Letra B).

WHERE

INCIDÊNCIA EM PROVA: ALTÍSSIMA

Galera, vejam como nós estamos avançando rapidamente: nós já sabemos como criar uma tabela, alterá-la, excluí-la e renomeá-la. Também sabemos como adicionar diversos tipos de restrições que especificam regras para os dados inseridos. Por falar nisso, também sabemos como inserir, atualizar e deletar dados. Por fim, nós já sabemos selecionar registros distintos, com apelidos ou não, e por meio de diversas formas de relacionamento entre tabelas.

Agora chegou o momento de estudar uma das cláusulas mais importantes: **WHERE**! Ela é responsável por permitir a filtragem dos registros de uma tabela por meio de uma ou mais condições. Vejamos:

ALUNO_ESTRATEGIA

NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
ALICE	11111111111	ALICE@ALICE.COM	01-01-2001	BRASÍLIA	200.00
BRUNO	22222222222	BRUNO@BRUNO.COM	02-02-2002	SÃO PAULO	100.00
CAIO	33333333333	CAIO@CAIO.COM	03-03-2003	GOIÂNIA	150.00
DIOGO	44444444444	DIOGO@DIOGO.COM	04-04-2004	SALVADOR	250.00
ELIS	55555555555	ELIS@ELIS.COM	05-05-2005	BRASÍLIA	50.00
FABIO	66666666666	FABIO@FABIO.COM	06-06-2006	SALVADOR	125.00
GABI	77777777777	GABI@GABI.COM	07-07-2007	BRASÍLIA	225.00
HUGO	88888888888	HUGO@HUGO.COM	08-08-2008	BRASÍLIA	50.00

SINTAXE DO COMANDO

```
SELECT NOME_COLUNA1, NOME_COLUNA2, ...
FROM NOME_DA_TABELA1
WHERE CONDICAO;
```

EXEMPLO DO COMANDO

```
SELECT *
FROM ALUNO_ESTRATEGIA
WHERE CIDADE = 'BRASÍLIA';
```

RESULTADO DO COMANDO

RESULTADO

NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
ALICE	11111111111	ALICE@ALICE.COM	01-01-2001	BRASÍLIA	200.00
ELIS	55555555555	ELIS@ELIS.COM	05-05-2005	BRASÍLIA	50.00
GABI	77777777777	GABI@GABI.COM	07-07-2007	BRASÍLIA	225.00
HUGO	88888888888	HUGO@HUGO.COM	08-08-2008	BRASÍLIA	50.00

ELECIONE	TODAS AS COLUNAS	DA TABELA	ALUNO_ESTRATEGIA	ONDE	A CIDADE SEJA BRASÍLIA

Essa cláusula não é utilizada apenas com o comando **SELECT**. Ela também pode ser utilizada junto com os comandos **UPDATE** e **DELETE**.

E a condição, Diego? A condição é basicamente uma expressão booleana, isto é, uma expressão que retornará um valor **TRUE** ou **FALSE**.

Existem diversas maneiras de definir uma condição e, para tal, utilizam-se operadores relacionais e lógicos para comparar valores.

Pronto, agora vocês podem comprar esse casaco da foto ao lado! Dito isso, vamos começar a estudar os operadores relacionais...

OPERADORES RELACIONAIS

OPERADOR	DESCRIÇÃO	EXEMPLO
=	IGUAL	`... WHERE NOME = 'DIEGO';`
>	MAIOR	`... WHERE VALOR_PAGO > 1000.00;`
>=	MAIOR OU IGUAL	`... WHERE IDADE >= 18;`
<	MENOR	`... WHERE DATA_NASCIMENTO < '01/01/2000';`
<=	MENOR OU IGUAL	`... WHERE VELOCIDADE <= 100;`
<>	DIFERENTE	`... WHERE CIDADE <> 'São Paulo';`

OPERADORES AND, OR, NOT, BETWEEN

Os operadores lógicos são um pouco mais complexos, logo vamos vê-los com um pouco mais de detalhes. Vamos começar pelos operadores AND, OR e NOT! Os dois primeiros são utilizados para filtrar registros baseado em mais de uma condição, de forma que o AND exibe um registro se todas as condições separadas por ele forem verdadeiras; e o OR exibe um registro se qualquer uma das condições separadas por ele for verdadeira. Bem simples...

Já o NOT é basicamente uma negação que inverte o significado de um operador lógico. Vamos ver a sintaxe de cada um e seus respectivos exemplos:

SINTAXE DO COMANDO

```
SELECT NOME_COLUNA1, NOME_COLUNA2, ...
FROM NOME_DA_TABELA1
WHERE CONDICAO1 AND CONDICAO2 AND CONDICAO3 ... ;
```

EXEMPLO DO COMANDO

```
SELECT *
FROM ALUNO_ESTRATEGIA
WHERE CIDADE = 'SALVADOR' AND VALOR_PAGO >= 200.00 ;
```

RESULTADO DO COMANDO

RESULTADO					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
DIOGO	44444444444	DIOGO@DIOGO.COM	04-04-2004	SALVADOR	250.00

Vejam que o resultado filtrou a tabela original, retornando apenas os registros cuja cidade era **Salvador E** cujo valor pago foi maior ou igual a **250.00**.

SINTAXE DO COMANDO

```
SELECT NOME_COLUNA1, NOME_COLUNA2, ...
FROM NOME_DA_TABELA1
WHERE CONDICAO1 OR CONDICAO2 OR CONDICAO3 ... ;
```

EXEMPLO DO COMANDO

```
SELECT *
FROM ALUNO_ESTRATEGIA
WHERE CIDADE = 'SALVADOR' OR VALOR_PAGO >= 200.00 ;
```

RESULTADO DO COMANDO

RESULTADO					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
ALICE	11111111111	ALICE@ALICE.COM	01-01-2001	BRASÍLIA	200.00
DIOGO	44444444444	DIOGO@DIOGO.COM	04-04-2004	SALVADOR	250.00
FABIO	66666666666	FABIO@FABIO.COM	06-06-2006	SALVADOR	125.00
GABI	77777777777	GABI@GABI.COM	07-07-2007	BRASÍLIA	225.00

Vejam que o resultado filtrou a tabela original, retornando apenas os registros cuja cidade era **Salvador OU** cujo valor pago foi maior ou igual a **250.00**.

SINTAXE DO COMANDO

```
SELECT NOME_COLUNA1, NOME_COLUNA2, ...
FROM NOME_DA_TABELA1
WHERE NOT CONDICAO1;
```

EXEMPLO DO COMANDO

```
SELECT *
FROM ALUNO_ESTRATEGIA
WHERE NOT CIDADE = 'BRASÍLIA';
```

RESULTADO DO COMANDO

RESULTADO					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
BRUNO	22222222222	BRUNO@BRUNO.COM	02-02-2002	SÃO PAULO	100.00
CAIO	33333333333	CAIO@CAIO.COM	03-03-2003	GOIÂNIA	150.00
DIOGO	44444444444	DIOGO@DIOGO.COM	04-04-2004	SALVADOR	250.00
FABIO	66666666666	FABIO@FABIO.COM	06-06-2006	SALVADOR	125.00

Vejam que o resultado filtrou a tabela original, retornando apenas os registros cuja cidade **NÃO** era **Brasília** – tanto que ele retorna registros com São Paulo, Goiânia e Salvador. *Professor, é possível combinar esses operadores?* Claro que é – sem problema algum! Pessoal, agora vamos falar sobre mais quatro operadores: `BETWEEN`, `IN`, `LIKE` e `IS NULL`. O `BETWEEN` permite selecionar valores (números, textos ou datas) dentro de um determinado intervalo.

Os valores dentro do intervalo incluem as extremidades? Sim, esse comando é inclusivo, isto é, os valores inicial e final são incluídos. Vamos ver a sintaxe:

SINTAXE DO COMANDO

```
SELECT NOME_COLUNA1, NOME_COLUNA2, ...
FROM NOME_DA_TABELA1
WHERE NOME_COLUNA1 BETWEEN VALOR1 AND VALOR2;
```

EXEMPLO DO COMANDO

```
SELECT *
FROM ALUNO_ESTRATEGIA
WHERE VALOR_PAGO BETWEEN 150.00 AND 300.00;
```

RESULTADO DO COMANDO

RESULTADO					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
ALICE	11111111111	ALICE@ALICE.COM	01-01-2001	BRASÍLIA	200.00
CAIO	33333333333	CAIO@CAIO.COM	03-03-2003	GOIÂNIA	150.00
DIOGO	44444444444	DIOGO@DIOGO.COM	04-04-2004	SALVADOR	250.00
GABI	77777777777	GABI@GABI.COM	07-07-2007	BRASÍLIA	225.00

Observem que o resultado contém valores entre 150.00 e 300.00 incluindo o 150.00 e 300.00. Por essa razão, retornou o registro cujo nome é Caio! Lembrando que é possível combinar esse operador com outros operadores, além de poder utilizá-lo também com textos ou datas. Já o operador `LIKE` é utilizado em uma cláusula `WHERE` para pesquisar um padrão especificado em uma coluna por meio da utilização de caracteres curingas (*wildcards*).

OPERADOR LIKE

Professor, o que é um caractere curinga? Trata-se de um caractere utilizado para substituir um ou mais caracteres em uma string (cadeia de caracteres). Existem diversos caracteres curingas, mas

não vamos entrar em detalhes porque eles se modificam a depender do SGBD utilizado. De toda forma, existem dois caracteres curingas principais que são frequentemente utilizados com o operador LIKE: sinal de porcentagem (%) e sinal de sublinhado (_).

O primeiro substitui zero, um ou mais caracteres e o segundo substitui um único caractere – ambos podem ser utilizados de forma combinada e com outros operadores. Vejamos a sintaxe:

SINTAXE DO COMANDO

```
SELECT NOME_COLUNA1, NOME_COLUNA2, ...
FROM NOME_DA_TABELA1
WHERE NOME_COLUNA1 LIKE PADRAO
```

OPERADOR	DESCRIÇÃO
...WHERE NOME LIKE 'A%'	Retorna valores que comecem com "A".
...WHERE NOME LIKE '%A'	Retorna valores que terminem com "A".
...WHERE NOME LIKE '%IO%'	Retorna valores que possuam "IO" em qualquer posição.
...WHERE NOME LIKE '_R%'	Retorna valores que possuam um caractere e depois a letra "R".
...WHERE NOME LIKE '%A_'	Retorna valores que terminem com "A" mais apenas um caractere.
...WHERE NOME LIKE 'A__%'	Retorna valores que comecem com "A" e possuem ao menos 3 caracteres.
...WHERE NOME LIKE '%A%O'	Retorna valores que possuam "A" depois "O" (imediatamente ou não).

Vejamos – para cada operador curinga – quais palavras poderiam ser aceitas (em verde) e quais não poderiam (em vermelho):

LIKE '%A'	A	BOLA	PERSPECTIVA	GLÓRIA	PUXAR	ARCO
LIKE 'A%'	A	ACRÉSCIMO	ALVENARIA	ARCO	BOLA	PUXAR
LIKE '%IO%'	IO	CAIO	DIOGO	FABIO	ELIAS	LOURA
LIKE '_R%'	ARCO	IRMÃ	ORATÓRIA	URANIO	PUXAR	BARCO
LIKE '%A_'	AR	ARMAS	AURICULAR	BAÚ	A	ALÔ
LIKE 'A__%'	AVÔ	AVERIGUAR	ALÔ	AMOR	AR	A
LIKE '%A%O'	LAGO	AVÔ	AO	AVERIGUADO	LOBA	PIVOTAR

Dada a nossa clássica tabela apresentada abaixo, vamos ver como seriam diversos exemplos de comandos utilizando o operador LIKE e seus caracteres curingas:

ALUNO_ESTRATEGIA					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
ALICE	11111111111	ALICE@ALICE.COM	01-01-2001	BRASÍLIA	200.00
BRUNO	22222222222	BRUNO@BRUNO.COM	02-02-2002	SÃO PAULO	100.00
CAIO	33333333333	CAIO@CAIO.COM	03-03-2003	GOIÂNIA	150.00
DIOGO	44444444444	DIOGO@DIOGO.COM	04-04-2004	SALVADOR	250.00
ELIS	55555555555	ELIS@ELIS.COM	05-05-2005	BRASÍLIA	50.00
FABIO	66666666666	FABIO@FABIO.COM	06-06-2006	SALVADOR	125.00
GABI	77777777777	GABI@GABI.COM	07-07-2007	BRASÍLIA	225.00
HUGO	88888888888	HUGO@HUGO.COM	08-08-2008	BRASÍLIA	50.00

EXEMPLO DO COMANDO

```
SELECT *
FROM ALUNO_ESTRATEGIA
WHERE NOME LIKE 'A%';
```

RESULTADO DO COMANDO

RESULTADO					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
ALICE	11111111111	ALICE@ALICE.COM	01-01-2001	BRASÍLIA	200.00

EXEMPLO DO COMANDO

```
SELECT *
FROM ALUNO_ESTRATEGIA
WHERE NOME LIKE '%A';
```

RESULTADO DO COMANDO

RESULTADO					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO

EXEMPLO DO COMANDO

```
SELECT *
FROM ALUNO_ESTRATEGIA
WHERE NOME LIKE '%IO';
```

RESULTADO DO COMANDO

RESULTADO					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
CAIO	33333333333	CAIO@CAIO.COM	03-03-2003	GOIÂNIA	150.00
FABIO	66666666666	FABIO@FABIO.COM	06-06-2006	SALVADOR	125.00

EXEMPLO DO COMANDO

```
SELECT *
FROM ALUNO_ESTRATEGIA
WHERE NOME LIKE '_R%';
```

RESULTADO DO COMANDO

RESULTADO					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
BRUNO	22222222222	BRUNO@BRUNO.COM	02-02-2002	SÃO PAULO	100.00

EXEMPLO DO COMANDO

```
SELECT *
FROM ALUNO_ESTRATEGIA
```

```
WHERE NOME LIKE '%A_';
```

RESULTADO DO COMANDO

RESULTADO					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO

EXEMPLO DO COMANDO

```
SELECT *
FROM ALUNO_ESTRATEGIA
WHERE NOME LIKE 'A__%';
```

RESULTADO DO COMANDO

RESULTADO					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
ALICE	11111111111	ALICE@ALICE.COM	01-01-2001	BRASÍLIA	200.00

EXEMPLO DO COMANDO

```
SELECT *
FROM ALUNO_ESTRATEGIA
WHERE NOME LIKE '%A%O';
```

RESULTADO DO COMANDO

RESULTADO					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
CAIO	33333333333	CAIO@CAIO.COM	03-03-2003	GOIÂNIA	150.00
FABIO	66666666666	FABIO@FABIO.COM	06-06-2006	SALVADOR	125.00

OPERADOR IS NULL

Os operadores `IS NULL` e `IS NOT NULL` são utilizados para avaliar se uma coluna é nula ou não, visto que operadores relacionais não podem ser utilizados para comparar valores nulos[7]. Vejamos...

SINTAXE DO COMANDO

```
SELECT NOME_COLUNA1, NOME_COLUNA2, ...
FROM NOME_DA_TABELA1
WHERE NOME_COLUNA1 IS NULL;
```

SINTAXE DO COMANDO

```
SELECT NOME_COLUNA1, NOME_COLUNA2, ...
FROM NOME_DA_TABELA1
WHERE NOME_COLUNA1 IS NOT NULL;
```

[7] Atenção: não dará erro, mas retornará vazio. Caso queira verificar se um valor é efetivamente nulo, utilize os operadores IS NULL ou IS NOT NULL.

OPERADOR IN

ALUNO_ESTRATEGIA					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
ALICE	11111111111	ALICE@ALICE.COM	01-01-2001	BRASÍLIA	200.00
BRUNO	22222222222	BRUNO@BRUNO.COM	02-02-2002	SÃO PAULO	100.00
CAIO	33333333333	CAIO@CAIO.COM	03-03-2003	GOIÂNIA	150.00
DIOGO	44444444444	DIOGO@DIOGO.COM	04-04-2004	SALVADOR	250.00
ELIS	55555555555	ELIS@ELIS.COM	05-05-2005	BRASÍLIA	50.00
FABIO	66666666666	FABIO@FABIO.COM	06-06-2006	SALVADOR	125.00
GABI	77777777777	GABI@GABI.COM	07-07-2007	BRASÍLIA	225.00
HUGO	88888888888	HUGO@HUGO.COM	08-08-2008	BRASÍLIA	50.00

Esse operador permite especificar múltiplos valores dentro de uma cláusula WHERE[8]. Ele também pode ser utilizado como **NOT IN**. Vejamos como é a sua sintaxe:

SINTAXE DO COMANDO

```
SELECT NOME_COLUNA1, NOME_COLUNA2, ...
FROM NOME_DA_TABELA1
WHERE NOME_COLUNA1 IN (VALOR1, VALOR2,...);
```

EXEMPLO DO COMANDO

```
SELECT *
FROM ALUNO_ESTRATEGIA
WHERE CIDADE IN ('SALVADOR', 'GOIANIA');
```

RESULTADO DO COMANDO

RESULTADO					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
CAIO	33333333333	CAIO@CAIO.COM	03-03-2003	GOIÂNIA	150.00
DIOGO	44444444444	DIOGO@DIOGO.COM	04-04-2004	SALVADOR	250.00
FABIO	66666666666	FABIO@FABIO.COM	06-06-2006	SALVADOR	125.00

Note que esse comando retorna todas as colunas da tabela em que a cidade seja **Salvador OU Goiânia**. Aliás, esse operador é como a abreviação para várias condições **OR**. Vamos comparar:

COMPARATIVO DE COMANDOS

```
SELECT *
FROM ALUNO_ESTRATEGIA
WHERE CIDADE = 'SALVADOR' OR CIDADE = 'GOIANIA';
```

[8] Aliás, é permitido também comparar várias colunas (Ex: ... WHERE (NomeColuna1, NomeColuna2) IN (SELECT NomeColuna1, NomeColuna2 FROM NomeTabela)).

=

```
SELECT *
FROM ALUNO_ESTRATEGIA
WHERE CIDADE IN ('SALVADOR', 'GOIANIA');
```

Vocês viram como é basicamente a mesma coisa? Pois é! Agora uma observação bastante importante: há outra maneira de utilizar o operador IN. *Como, professor?* Por meio de subconsultas! *O que seria uma subconsulta, Diego?* Também chamado de subqueries ou de consultas aninhadas, trata-se basicamente de uma consulta dentro de outra consulta (pode ser um SELECT, INSERT, UPDATE ou DELETE). *Professor, eu viajei agora! Tem como mostrar isso daí?* Claro...

SINTAXE DO COMANDO

```
SELECT NOME_COLUNA1, NOME_COLUNA2, ...
FROM NOME_DA_TABELA1
WHERE NOME_COLUNA1 IN (SELECT ... FROM ...);
```

Vocês viram que existe uma consulta dentro de outra? Pois é, ela é chamada de subconsulta! Elas podem ser de dois tipos: correlacionadas ou não correlacionadas. As subconsultas correlacionadas são consultas que dependem e fazem referências às colunas de consultas externas a qual estão contidas; já as consultas não correlacionadas são consultas independentes das consultas externas nas quais estão contidas. Para ver como funciona, vamos adotar as tabelas seguintes:

CAPITAIS				
CODIGO	CAPITAL	ESTADO	SIGLA	REGIAO
001	MACEIÓ	ALAGOAS	AL	NORDESTE
002	SALVADOR	BAHIA	BA	NORDESTE
003	BELÉM	PARÁ	PA	NORTE
004	MANAUS	AMAZONAS	AM	NORTE
005	GOIÂNIA	GOIÁS	GO	CENTRO-OESTE
006	SÃO LUIS	MARANHÃO	MA	NORDESTE
007	CURITIBA	PARANÁ	PR	SUL
008	PORTO ALEGRE	RIO GRANDE DO SUL	RS	SUL

ALUNO_ESTRATEGIA					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
ALICE	11111111111	ALICE@ALICE.COM	01-01-2001	BRASÍLIA	200.00
BRUNO	22222222222	BRUNO@BRUNO.COM	02-02-2002	SÃO PAULO	100.00
CAIO	33333333333	CAIO@CAIO.COM	03-03-2003	GOIÂNIA	150.00
DIOGO	44444444444	DIOGO@DIOGO.COM	04-04-2004	SALVADOR	250.00
ELIS	55555555555	ELIS@ELIS.COM	05-05-2005	BRASÍLIA	50.00
FABIO	66666666666	FABIO@FABIO.COM	06-06-2006	SALVADOR	125.00
GABI	77777777777	GABI@GABI.COM	07-07-2007	BRASÍLIA	225.00
HUGO	88888888888	HUGO@HUGO.COM	08-08-2008	BRASÍLIA	50.00

EXEMPLO DO COMANDO

```
SELECT *
FROM ALUNO_ESTRATEGIA
WHERE CIDADE IN (SELECT CAPITAL FROM CAPITAIS)
```

Esse caso trata de uma subconsulta não correlacionada. Note que primeiro executamos a consulta interna e depois executamos a consulta externa. Vamos ver o resultado da consulta interna:

RESULTADO INTERMEDIÁRIO

RESULTADO
CAPITAL
MACEIÓ
SALVADOR
BELÉM
MANAUS
GOIÂNIA
SÃO LUIS
CURITIBA
PORTO ALEGRE

Agora vamos executar a consulta externa – observe que ela retornará todas as colunas da tabela ALUNO_ESTRATEGIA em que a cidade esteja dentre as cidades da tabela acima:

RESULTADO DO COMANDO

RESULTADO					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
CAIO	33333333333	CAIO@CAIO.COM	03-03-2003	GOIÂNIA	150.00
DIOGO	44444444444	DIOGO@DIOGO.COM	04-04-2004	SALVADOR	250.00
FABIO	66666666666	FABIO@FABIO.COM	06-06-2006	SALVADOR	125.00

Já a subconsulta correlacionada será explicada dentro do contexto do operador EXISTS, logo vamos entendê-lo primeiro em detalhes...

OPERADORES EXISTS

Esse operador permite testar a existência de qualquer registro em uma subconsulta. Ele retorna TRUE se a subconsulta retornar um ou mais registros; caso contrário, retorna FALSE. Vejamos:

SINTAXE DO COMANDO

```
SELECT NOME_COLUNA1, NOME_COLUNA2, ...
FROM NOME_DA_TABELA1
WHERE EXISTS (SELECT ... FROM ... WHERE ...);
```

EXEMPLO DO COMANDO

```
SELECT *
```

```
FROM CAPITAIS C
WHERE EXISTS (SELECT CIDADE FROM ALUNO_ESTRATEGIA AE WHERE AE.CIDADE = C.CAPITAL);
```

RESULTADO DO COMANDO

RESULTADO				
CODIGO	CAPITAL	ESTADO	SIGLA	REGIAO
002	SALVADOR	BAHIA	BA	NORDESTE
005	GOIÂNIA	GOIÁS	GO	CENTRO-OESTE

Note que esse comando retorna todos os registros da tabela de CAPITAIS desde que a capital exista como uma das cidades da tabela ALUNO_ESTRATEGIA. Esse é um clássico exemplo de subconsulta correlacionada. Ao contrário da subconsulta não correlacionada, essa não pode ser executada independentemente da consulta externa, dado que contém uma ou mais referências a colunas da consulta externa. Inclusive, ela retornará erro caso se tente executá-la de forma independente.

Dito isso, uma subconsulta correlacionada (interna) será executada uma vez para cada linha candidata da consulta externa. Os valores de cada linha da coluna candidata serão utilziados para fornecer valores para as colunas da consulta externa no interior de cada execução da subconsulta correlacionada. Os resultados finais serão baseados nos resultados de cada execução da subconsulta correlacionada. Eu sei! É realmente muito abstrato, portanto vamos ver exemplos...

ARVORE_GENEALOGICA	
ASCENDENTE	DESCENDENTE
ALICE	LAURA
BRUNO	ELIS
BRUNO	HUGO
ELIS	CAIO
GABI	ALICE
HUGO	GABI
JUDITH	LAURA

Considerem a tabela acima com a base para os exemplos que vamos analisar. Vamos começar por meio da seguinte consulta:

EXEMPLO DO COMANDO

```
SELECT *
FROM ARVORE_GENEALOGICA
WHERE EXISTS (SELECT * FROM ARVORE_GENEALOGICA AG
              WHERE AG.ASCENDENTE = 'BRUNO'
              AND AG.DESCENDENTE = ARVORE_GENEALOGICA.ASCENDENTE);
```

Em primeiro lugar, podemos notar que se trata de uma consulta aninhada, dado que temos uma consulta dentro de outra. Em segundo lugar, podemos notar que a consulta interna é uma subconsulta correlacionada, dado que temos uma referência a uma coluna da consulta externa. Dito

isso, agora vamos analisar – para cada tupla da consulta externa – se a consulta interna retorna algum valor. Se sim, retornaremos também para a consulta externa; caso contrário, não.

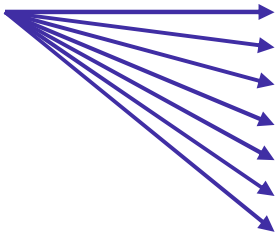
Além disso, note que temos a consulta interna e a consulta externa tratando da mesma tabela, porém uma é chamada de ARVORE_GENEALOGICA e a outra tem um *alias* chamado AG:

ARVORE_GENEALOGICA	
ASCENDENTE	DESCENDENTE
ALICE	LAURA
BRUNO	ELIS
BRUNO	HUGO
ELIS	CAIO
GABI	ALICE
HUGO	GABI
JUDITH	LAURA

AG	
ASCENDENTE	DESCENDENTE
ALICE	LAURA
BRUNO	ELIS
BRUNO	HUGO
ELIS	CAIO
GABI	ALICE
HUGO	GABI
JUDITH	LAURA

Agora vamos pegar cada tupla da consulta externa e vamos validar a consulta interna, de modo que a tupla da consulta externa será retornada se, e somente se, a consulta interna retornar alguma tupla. Podemos ver que a consulta interna retornará todas as tuplas da tabela AG desde que a coluna ASCENDENTE da tabela AG seja "Bruno" e a coluna DESCENDENTE da tabela AG seja igual à coluna ASCENDENTE da tabela ARVORE_GENEALOGICA. Agora vamos para a prática...

ARVORE_GENEALOGICA	
ASCENDENTE	DESCENDENTE
ALICE	LAURA
BRUNO	ELIS
BRUNO	HUGO
ELIS	CAIO
GABI	ALICE
HUGO	GABI
JUDITH	LAURA



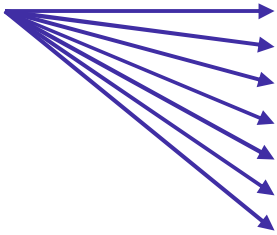
AG	
ASCENDENTE	DESCENDENTE
ALICE	LAURA
BRUNO	ELIS
BRUNO	HUGO
ELIS	CAIO
GABI	ALICE
HUGO	GABI
JUDITH	LAURA

A primeira tupla da tabela da consulta externa é (**ALICE**, **LAURA**). Essa tupla será retornada apenas se a consulta interna retornar alguma tupla. *Vamos ver se ela retorna?* A primeira tupla da consulta interna também é (**ALICE**, **LAURA**). Porém, a consulta interna só retornará essa tupla se AG.ASCENDENTE = "Bruno". Ora, acabamos de ver que – para essa tupla – AG.ASCENDENTE é "Alice", logo nem precisamos ver o restante da consulta porque essa tupla já não será retornada.

Vamos agora para a segunda tupla da tabela da consulta interna: (**BRUNO**, **ELIS**). A consulta interna só retornará essa tupla se AG.ASCENDENTE = "Bruno". É realmente "Bruno", mas ainda não acabou: AG.DESCENDENTE deve ser igual a ARVORE_GENEALOGICA.ASCENDENTE. Ora, AG.DESCENDENTE é ELIS e ARVORE_GENEALOGICA.ASCENDENTE é ALICE. Logo, essa tupla também não será retornada porque não cumpriu as duas condições do operador AND. *Viram que nós temos que fazer um por um?*

Em tese, teríamos que fazer 7 x 7 = 49 avaliações. No entanto, é possível identificar alguns atalhos: o nome de AG.ASCENDENTE deve ser "Bruno". Isso só ocorre em duas oportunidades:

ARVORE_GENEALOGICA	
ASCENDENTE	DESCENDENTE
ALICE	LAURA
BRUNO	ELIS
BRUNO	HUGO
ELIS	CAIO
GABI	ALICE
HUGO	GABI
JUDITH	LAURA



AG	
ASCENDENTE	DESCENDENTE
ALICE	LAURA
BRUNO	ELIS
BRUNO	HUGO
ELIS	CAIO
GABI	ALICE
HUGO	GABI
JUDITH	LAURA

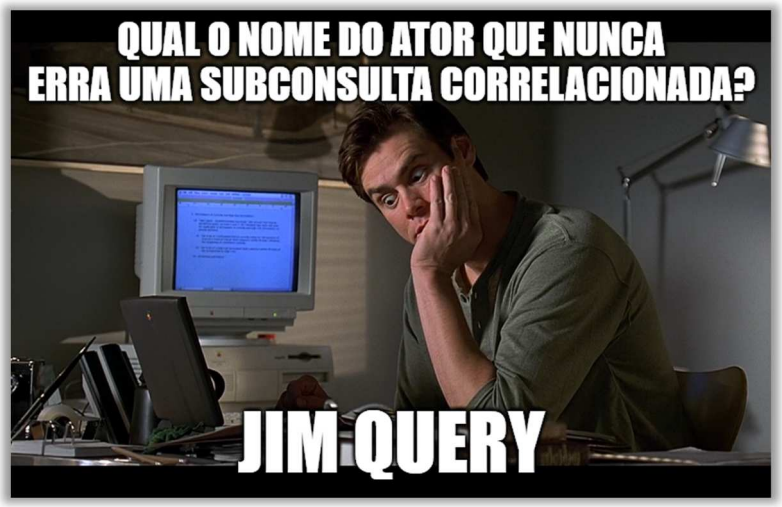
Vejam que ocorre somente nas tuplas (**BRUNO**, **ELIS**) e (**BRUNO**, **HUGO**). Nesses dois casos, temos AG.DESCENDENTE = (**ELIS**, **HUGO**). Por fim, vejam que temos ELIS ou HUGO como ASCENDENTE na tabela ARVORE_GENEALOGICA apenas em duas oportunidades: (**ELIS**, **CAIO**) e (**HUGO**, **GABI**). Logo, essas são as tuplas que serão retornadas pela consulta externa. *Interessante, não?* Vejam como isso seria em termos de tabela:

ARVORE_GENEALOGICA	
ASCENDENTE	DESCENDENTE
ALICE	LAURA
BRUNO	ELIS
BRUNO	HUGO
ELIS	CAIO
GABI	ALICE
HUGO	GABI
JUDITH	LAURA

AG	
ASCENDENTE	DESCENDENTE
ALICE	LAURA
BRUNO	ELIS
BRUNO	HUGO
ELIS	CAIO
GABI	ALICE
HUGO	GABI
JUDITH	LAURA

RESULTADO DO COMANDO

RESULTADO	
ASCENDENTE	DESCENDENTE
ELIS	CAIO
HUGO	GABI



Galera, eu considero esse o assunto mais difícil de SQL! De looonde, essa é disparada a parte mais complicada de entender na aula. Eu mesmo demorei um bom tempo para fixar...

Logo, não fiquem frustrados se vocês não compreenderam porque é um assunto realmente complexo. O que eu sugiro é tentar assistir os vídeos sobre o assunto. *Fechou?* Então vamos seguir ;)

GROUP BY

INCIDÊNCIA EM PROVA: MÉDIA

A cláusula **GROUP BY** foi criada e adicionada à linguagem SQL porque a cláusula **WHERE** não podia ser utilizada com funções de agregação. *Como assim, Diego?* Galera, imaginem que vocês queiram um relatório com alguma quantidade, soma, valores máximos, valores mínimos, média, entre outros. Para isso, você precisará utilizar uma função de agregação junto com a cláusula **GROUP BY**. Vamos ver a seguir sintaxes e exemplos...

ALUNO_ESTRATEGIA					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
ALICE	11111111111	ALICE@ALICE.COM	01-01-2001	BRASÍLIA	200.00
BRUNO	22222222222	BRUNO@BRUNO.COM	02-02-2002	SÃO PAULO	100.00
CAIO	33333333333	CAIO@CAIO.COM	03-03-2003	GOIÂNIA	150.00
DIOGO	44444444444	DIOGO@DIOGO.COM	04-04-2004	SALVADOR	250.00
ELIS	55555555555	ELIS@ELIS.COM	05-05-2005	BRASÍLIA	50.00
FABIO	66666666666	FABIO@FABIO.COM	06-06-2006	SALVADOR	125.00
GABI	77777777777	GABI@GABI.COM	07-07-2007	BRASÍLIA	225.00
HUGO	88888888888	HUGO@HUGO.COM	08-08-2008	BRASÍLIA	50.00

SINTAXE DO COMANDO

```
SELECT LISTA_DE_COLUNAS, FUNCAO_DE_AGREGACAO(COLUNA)
FROM NOME_DA_TABELA
WHERE CONDIÇOES --OPCIONAL
GROUP BY LISTA_DE_COLUNAS;
```

EXEMPLO DO COMANDO

```
SELECT CIDADE, COUNT(CPF)
FROM ALUNO_ESTRATEGIA
GROUP BY CIDADE;
```

COMANDO					
SELECT	CIDADE, COUNT(CPF)	FROM	ALUNO_ESTRATEGIA	GROUP BY	CIDADE
↓	↓	↓	↓	↓	↓
INTERPRETAÇÃO					
SELECIONE	AS COLUNAS CIDADE E COUNT(CPF)	DA TABELA	ALUNO_ESTRATEGIA	E AGRUPE POR	CIDADE

Essa cláusula buscará registros de uma tabela que possuem um valor em comum para um ou mais atributos e os agrupará baseado em algum critério de agrupamento (soma, média, quantidade, etc). No caso do comando acima, ele buscará registros que tenham o mesmo valor para o atributo CIDADE e os agrupará pela quantidade (Brasília tem quatro aparições; São Paulo tem uma aparição; Goiânia tem uma aparição; e Salvador tem duas aparições). Vejam a tabela resultante a seguir:

RESULTADO DO COMANDO

RESULTADO

CIDADE	COUNT(CPF)
BRASÍLIA	4
SÃO PAULO	1
GOIÂNIA	1
SALVADOR	2

Na tabela a seguir, podemos ver outras funções de agregação – lembrando que todas elas podem ser utilizadas com o operador DISTINCT! Em seguida, veremos outros exemplos com essas funções:

FUNÇÕES	AGREGAÇÃO	DESCRIÇÃO
COUNT()	Quantidade	Essa função conta a quantidade total de dados de um dado campo.
SUM()	Soma	Essa função soma valores numéricos de um dado campo[9].
AVG()	Média	Essa função calcula a média aritmética simples de um conjunto de valores numéricos.
MAX()	Máximo	Essa função retorna o maior valor encontrado de um dado campo.
MIN()	Mínimo	Essa função retorna o menor valor encontrado de um dado campo.

EXEMPLO DO COMANDO

```
SELECT CIDADE, SUM(VALOR_PAGO)
FROM ALUNO_ESTRATEGIA
GROUP BY CIDADE;
```

RESULTADO DO COMANDO

RESULTADO	
CIDADE	SUM(VALOR_PAGO)
BRASÍLIA	525.00
SÃO PAULO	100.00
GOIÂNIA	150.00
SALVADOR	375.00

EXEMPLO DO COMANDO

```
SELECT CIDADE, AVG(VALOR_PAGO)
FROM ALUNO_ESTRATEGIA
GROUP BY CIDADE;
```

RESULTADO DO COMANDO

RESULTADO	
CIDADE	AVG(VALOR_PAGO)
BRASÍLIA	131.25
SÃO PAULO	100.00
GOIÂNIA	150.00

[9] Um detalhe que cai sobre a função SUM(): é possível inserir um valor de tal modo que, para cada registro encontrado, esse valor seja somado. Por exemplo: SUM(2) somará duas unidades para cada registro encontrado. Logo, se encontrou 3 registros, retornará 2x3 = 6. Além disso, é importante mencionar que SUM(1) = COUNT(*) = COUNT(1).

SALVADOR	187.50

EXEMPLO DO COMANDO

```
SELECT CIDADE, MAX(VALOR_PAGO)
FROM ALUNO_ESTRATEGIA
GROUP BY CIDADE;
```

RESULTADO DO COMANDO

RESULTADO	
CIDADE	MAX(VALOR_PAGO)
BRASÍLIA	225.00
SÃO PAULO	100.00
GOIÂNIA	150.00
SALVADOR	250.00

EXEMPLO DO COMANDO

```
SELECT CIDADE, MIN(VALOR_PAGO)
FROM ALUNO_ESTRATEGIA
GROUP BY CIDADE;
```

RESULTADO DO COMANDO

RESULTADO	
CIDADE	MIN(VALOR_PAGO)
BRASÍLIA	50.00
SÃO PAULO	100.00
GOIÂNIA	150.00
SALVADOR	125.00

Lembrando que as funções de agregação podem ser utilizadas também em outras cláusulas sem necessariamente a presença do **GROUP BY**, como é mostrado a seguir:

EXEMPLO DO COMANDO

```
SELECT MAX(VALOR_PAGO)
FROM ALUNO_ESTRATEGIA
```

RESULTADO DO COMANDO

RESULTADO					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
DIOGO	44444444444	DIOGO@DIOGO.COM	04-04-2004	SALVADOR	250.00

HAVING

INCIDÊNCIA EM PROVA: MÉDIA

A cláusula **WHERE** filtra linhas de acordo com alguma condição, já a cláusula **HAVING** filtra agrupamentos também de acordo com alguma condição. Dessa forma, podemos concluir que a cláusula **HAVING** só pode existir se houver anteriormente uma cláusula **GROUP BY**. No exemplo a seguir, queremos filtrar o agrupamento de modo que só mostre as linhas cuja função agregada **COUNT(CPF)** seja maior que 1.

SINTAXE DO COMANDO

```
SELECT LISTA_DE_COLUNAS, FUNCAO_DE_AGREGACAO(COLUNA)
FROM NOME_DA_TABELA
WHERE CONDIÇOES --OPCIONAL
GROUP BY LISTA_DE_COLUNAS
HAVING CONDIÇOES;
```

EXEMPLO DO COMANDO

```
SELECT CIDADE, COUNT(CPF)
FROM ALUNO_ESTRATEGIA
GROUP BY CIDADE
HAVING COUNT(CPF) > 1;
```

RESULTADO DO COMANDO

RESULTADO	
CIDADE	COUNT(CPF)
BRASÍLIA	4
SALVADOR	2

COMANDO							
SELECT	CIDADE, COUNT(CPF)	FROM	ALUNO_ESTRATEGIA	GROUP BY	CIDADE	HAVING	COUNT(CPF) >1
↓	↓	↓	↓	↓	↓	↓	↓
INTERPRETAÇÃO							
SELECIONE	AS COLUNAS CIDADE E COUNT(CPF)	DA TABELA	ALUNO_ESTRATEGIA	E AGRUPE POR	CIDADE	QUE TENHA	CONTAGEM >1

A coluna utilizada na cláusula **HAVING** deve necessariamente estar na lista de colunas selecionadas no **SELECT** ou estar contida dentro de uma função de agregação. Dessa forma, se fizermos um **SELECT** de **CIDADE** junto de uma função de agregação selecionada, a cláusula **HAVING** poderá filtrar por **CIDADE**, pela função de agregação selecionada ou por uma coluna (Ex: **VALOR_PAGO**) desde que ela esteja contida dentro de uma função de agregação. *Entendido?* Vejamos um exemplo:

EXEMPLO DO COMANDO

```
SELECT CIDADE, COUNT(CPF)
FROM ALUNO_ESTRATEGIA
GROUP BY CIDADE
HAVING MAX(VALOR_PAGO) > 100;
```

Note que **VALOR_PAGO** não pode ser utilizado no **HAVING** porque não consta do **SELECT**, mas como está dentro de uma função de agregação (**MAX**), sua utilização é permitida.

RESULTADO DO COMANDO

RESULTADO	
CIDADE	COUNT(CPF)
BRASÍLIA	4
GOIÂNIA	1
SALVADOR	2

ORDER BY

INCIDÊNCIA EM PROVA: ALTA

A consulta apresentada no tópico anterior retorna apenas os agrupamentos que satisfizeram a condição estabelecida na sintaxe, ou seja, cidades que possuem mais de um aluno. No entanto, eu quero que os dados sejam apresentados em ordem crescente e – para tal – é necessário utilizar a nossa penúltima cláusula: **ORDER BY**. Por meio dela, é possível ordenar registros/linhas de uma tabela em ordem crescente (**ASC**) ou decrescente (**DESC**).

SINTAXE DO COMANDO

```
SELECT LISTA_DE_COLUNAS, FUNCAO_DE_AGREGACAO(COLUNA)
FROM NOME_DA_TABELA
WHERE CONDIÇOES --OPCIONAL
GROUP BY LISTA_DE_COLUNAS --OPCIONAL
HAVING CONDIÇOES --OPCIONAL
ORDER BY COLUNA1 ASC | DESC, COLUNA2 ASC | DESC, ...;
```

EXEMPLO DO COMANDO C/ ASC

```
SELECT CIDADE, COUNT(CPF)
FROM ALUNO_ESTRATEGIA
GROUP BY CIDADE
HAVING COUNT(CPF) > 1
ORDER BY COUNT(CPF) ASC;
```

RESULTADO DO COMANDO C/ ASC

RESULTADO	
CIDADE	COUNT(CPF)
SALVADOR	2
BRASÍLIA	4

EXEMPLO DO COMANDO C/ DESC

```
SELECT CIDADE, COUNT(CPF)
FROM ALUNO_ESTRATEGIA
GROUP BY CIDADE
HAVING COUNT(CPF) > 1
ORDER BY COUNT(CPF) DESC;
```

RESULTADO DO COMANDO C/ DESC

RESULTADO	
CIDADE	COUNT(CPF)
BRASÍLIA	4
SALVADOR	2

INTERPRETAÇÃO									
SELECT	CIDADE, COUNT(CPF)	FROM	ALUNO_ESTRATEGIA	GROUP BY	CIDADE	HAVING	COUNT(CPF) >1	ORDER BY	ASC

INTERPRETAÇÃO									
SELECIONE	AS COLUNAS CIDADE E COUNT(CPF)	DA TABELA	ALUNO_ESTRATEGIA	E AGRUPE POR	CIDADE	QUE TENHA	CONTAGEM >1	EM ORDEM	CRESC

A coluna utilizada para ordenação na cláusula **ORDER BY** deve necessariamente estar na lista de colunas do **SELECT**, em uma função de agregação qualquer ou ainda em uma coluna definida em uma tabela do **FROM**. Dessa forma, se fizermos um **SELECT** de **NOME** e **CIDADE**, a cláusula **ORDER BY** poderá ordenar por **NOME**, por **CIDADE** ou por outra coluna (Ex: **VALOR_PAGO**) desde que ela esteja contida dentro de uma função de agregação. *Entendido?* Vejamos a sintaxe do comando...

Algumas observações: caso não seja indicado se é **ASC** ou **DESC**, o valor *default* é **ASC**. Veja também que várias dessas cláusulas são opcionais, logo é possível utilizar o **ORDER BY** apenas com **SELECT** e **FROM**. É possível também representar a coluna responsável pela ordenação por meio de um número que indique a ordem da coluna (Ex: 1 é primeira coluna, 2 é segunda coluna, 3 é terceira coluna, etc). Por fim, é possível indicar mais de uma coluna para ordenação (no caso de empates).

EXEMPLO DO COMANDO

```
SELECT *
FROM ALUNO_ESTRATEGIA
ORDER BY 6,1;
```

RESULTADO DO COMANDO

RESULTADO					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
ELIS	55555555555	ELIS@ELIS.COM	05-05-2005	BRASÍLIA	50.00
HUGO	88888888888	HUGO@HUGO.COM	08-08-2008	BRASÍLIA	50.00
BRUNO	22222222222	BRUNO@BRUNO.COM	02-02-2002	SÃO PAULO	100.00
FABIO	66666666666	FABIO@FABIO.COM	06-06-2006	SALVADOR	125.00
CAIO	33333333333	CAIO@CAIO.COM	03-03-2003	GOIÂNIA	150.00
ALICE	11111111111	ALICE@ALICE.COM	01-01-2001	BRASÍLIA	200.00
GABI	77777777777	GABI@GABI.COM	07-07-2007	BRASÍLIA	225.00
DIOGO	44444444444	DIOGO@DIOGO.COM	04-04-2004	SALVADOR	250.00

Note que ele ordenou pela coluna #6 (**VALOR_PAGO**), mas – como houve um empate de **VALOR_PAGO** de **50.00** – ele ordenou pela coluna #1 (**NOME**).

LIMIT

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

Essa cláusula é utilizada para restringir o conjunto de resultados a um número fixo de linhas. É claro que, se o conjunto de resultados completo tiver menos linhas do que o número limite que definimos, o banco de dados retornará menos registros do que o número limite. Essa cláusula não faz parte do padrão SQL puro, logo outros dialetos possuem variações: MS-SQL Server chama de **TOP** e o Oracle chama de **ROWNUM**.

SINTAXE DO COMANDO

```
SELECT LISTA_DE_COLUNAS
FROM NOME_DA_TABELA
LIMIT QTD_LINHAS;
```

EXEMPLO DO COMANDO

```
SELECT *
FROM ALUNO_ESTRATEGIA
LIMIT 2;
```

RESULTADO DO COMANDO

RESULTADO					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
ALICE	11111111111	ALICE@ALICE.COM	01-01-2001	BRASÍLIA	200.00
BRUNO	22222222222	BRUNO@BRUNO.COM	02-02-2002	SÃO PAULO	100.00

(TJDFT – 2008) Na linguagem de consulta SQL (structured query language), é possível obter o resultado de uma consulta SELECT ordenado pelo valor de um ou mais atributos.

_____

Comentários: é perfeitamente possível obter resultados de uma consulta por meio da cláusula ORDER BY (Correto).

Por fim, vamos falar de um operador chamado **UNION**! Não se trata de uma cláusula, trata-se de um operador utilizado para combinar os resultados de duas ou mais instruções **SELECT**. Para que funcione, cada uma dessas instruções deve conter o mesmo número de colunas, sendo que as colunas devem ter tipos de dados semelhantes – além de ter a mesma ordem. Vamos ver como é a sua sintaxe:

SINTAXE DO COMANDO

```
SELECT LISTA_DE_COLUNAS
FROM NOME_DA_TABELA1

UNION

SELECT LISTA_DE_COLUNAS
FROM NOME_DA_TABELA2;
```

Esse comando elimina eventuais linhas duplicadas. Caso se queira permitir linhas duplicadas, utiliza-se a instrução **UNION ALL**:

SINTAXE DO COMANDO

```
SELECT LISTA_DE_COLUNAS
FROM NOME_DA_TABELA1

UNION ALL

SELECT LISTA_DE_COLUNAS
FROM NOME_DA_TABELA2;
```

O resultado é muito simples: esse comando basicamente une/junta as linhas das tabelas que compõem a união. Se tinham 10 linhas em cada tabela, teremos 20 (exceto se houver duplicatas).

IMPORTANTE

CLÁSULAS	DESCRIÇÃO
FROM	Comando utilizado para indicar de onde os dados devem ser selecionados.
JOIN	Comando utilizado para combinar linhas tabelas, com base em uma coluna em comum entre elas.
WHERE	Comando utilizado para filtrar os dados.
GROUP BY	Comando utilizado para agregar um conjunto de dados.
HAVING	Comando utilizado para filtrar dados agregados.
ORDER BY	Comando utilizado para ordenar os dados recuperados.
LIMIT	Comando utilizado para limitar a quantidade de resultados.

A única cláusula obrigatória do **SELECT** é o **FROM** – todas as outras são opcionais! A única ressalva é que elas devem vir na ordem acima e, se houver um **HAVING**, antes deve existir um **GROUP BY**.

MÁ NOTÍCIA ☹

Galera, eu tenho uma boa e uma má notícia! Vou começar pela má: tudo que vimos nas últimas 30 páginas pode ser combinado dentro de um SELECT. *Como é, Diego?* Sim, nós vimos a maioria dos comandos e cláusulas de forma separada, mas é possível fazer consultas gigantescas e absurdamente complexas juntando tudo– isso porque nem vimos tudo que existe dentro do SQL. Vamos ver um exemplo de questão bastante complexa...

(AL/RO – 2018) Considere o comando a seguir utilizando a tabela arvore seguinte.

Pessoa	Descendente
Ana	Vitoria
João	Maria
João	Rafael
Maria	Tiago
Natalia	Ana
Rafael	Natalia
Ana	Vitoria

```
insert into arvore
select distinct a1.pessoa, a2.descendente
from arvore a1, arvore a2
where a1.descendente = a2.pessoa and
      not exists (select * from arvore a3
                  where a3.pessoa = a1.pessoa
                  and a3.descendente = a2.descendente)
```

Sabe-se que esse comando foi executado duas vezes, sobre a instância acima, e que não houve nenhuma outra alteração na tabela entre as duas execuções. Assinale o número de registros inseridos na segunda execução do comando:

a) Zero.
b) Um.
c) Dois.
d) Três.
e) Quatro.

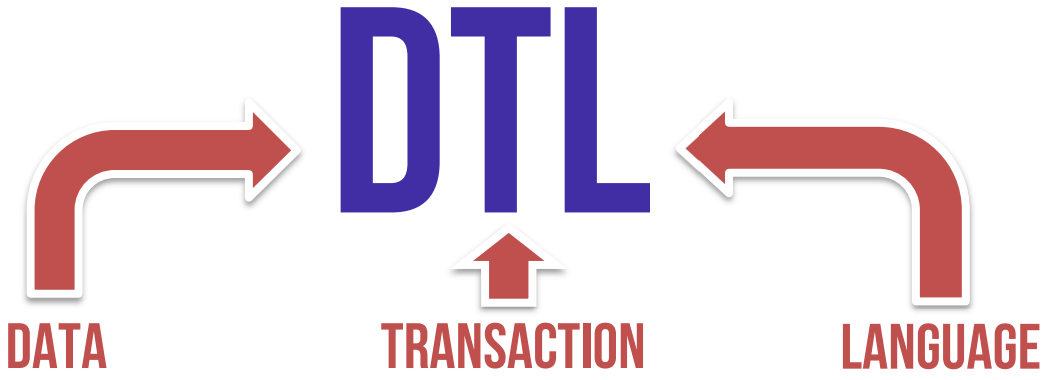
Comentários: não vou nem tentar explicar porque essa questão é de um nível tão pesado que até concurseiros da área de banco de dados dentro do concurseiros de tecnologia da informação teriam dificuldade.

BOA NOTÍCIA

Professor, cadê a boa notícia pelo amor de Santo Cristo? Galera, a boa notícia é que essas consultas absurdamente complexas geralmente não caem para concursos fora da área de Tecnologia da Informação. O nível das questões para área geral tem aumentado, mas não chegou nesse ponto da questão acima. Nós veremos questões em um nível que consideramos adequado, mas muito cuidado ao fazer questões em sites por aí. *Fechado?*

DTL (DATA TRANSACTION LANGUAGE)

Comandos DTL



Agora nós já sabemos como manipular dados de um banco de dados. Essa manipulação se dá por meio de transações: modificar, inserir, remover e consultar! *Vocês se lembram que uma transação deve ser atômica, consistente, isolada e durável?* Pois é! **Para alcançar esses objetivos, faz-se uso de comandos capazes de controlar transações** por meio do Data Transaction Language (DTL). *Professor, não entendi nada!* Calma, vai ficar mais claro a seguir...

COMANDOS DTL	DESCRIÇÃO
COMMIT	Comando utilizado para finalizar/confirmar uma transação dentro de um SGBD.
ROLLBACK	Comando utilizado para descartar mudanças nos dados desde o último COMMIT ou ROLLBACK.

Há ainda os comandos **SAVEPOINT** e **SET TRANSACTION**, mas nós não vamos entrar em detalhes sobre eles porque eles raramente caem em prova.

(Prefeitura de Teresina/PI – 2016) Em um banco de dados relacional os comandos são classificados em:

– DDL – Data Definition Language.
– DML – Data Manipulation Language.
– DCL – Data Control Language.
– TCL – Transaction Control Language.

Os seguintes comandos: COMMIT, CREATE, ROLLBACK, DELETE, REVOKE e UPDATE, são respectivos a:

a) DDL, DDL, TCL, DML, DML e DCL.
b) TCL, DDL, TCL, DML, DCL e DML.
c) DML, TCL, DDL, DDL, DCL e DML.
d) TCL, DDL, TCL, DDL, DCL e DML.

e) DCL, DDL, TCL, DDL, DCL e DML.

_____

Comentários: COMMIT é TCL; CREATE é DDL; ROLLBACK é TCL; DELETE é DML; REVOKE é DCL; e UPDATE é DML (Letra B).

COMMIT

INCIDÊNCIA EM PROVA: BAIXA

Comando utilizado para salvar permanentemente uma transação em um banco de dados. Quando nós utilizamos transações DML (Ex: **INSERT**, **UPDATE** ou **DELETE**), as mudanças feitas por esses comandos não são permanentes. Até que a sessão se encerre, as mudanças realizadas por esses comandos ainda não foram confirmadas. Utiliza-se o comando **COMMIT** para efetivar as mudanças realizadas pelas transações.

Vamos supor que você precise inserir, modificar ou deletar diversos valores em um banco de dados. O banco de dados realiza essas transações e as deixa em um estado intermediário. No entanto, ele só altera os dados de fato quando se executa o comando **COMMIT**. Esse comando salva todas as transações de uma base de dados desde o último **COMMIT** ou **ROLLBACK**. **O SGBD saberá, assim, que as transações executadas estão de fato ser confirmadas.**

Vamos imaginar um cenário: você precisa fazer uma transferência bancária para outra pessoa. Na tela inicial, você insere a agência e conta dessa outra pessoa, e também o valor a ser transferido. Na tela seguinte, o aplicativo bancário exibe todos esses dados mais um estado intermediário que mostra quanto será o seu saldo em conta após a transferência – além de pedir uma senha para confirmar a transferência. **Adivinhem só: o COMMIT seria equivalente a essa confirmação**!

SINTAXE DO COMANDO

```
COMMIT;
```

ESTADO INICIAL

ALUNO_ESTRATEGIA					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
ALICE	11111111111	ALICE@ALICE.COM	01-01-2001	BRASÍLIA	200.00
BRUNO	22222222222	BRUNO@BRUNO.COM	02-02-2002	SÃO PAULO	100.00
CAIO	33333333333	CAIO@CAIO.COM	03-03-2003	GOIÂNIA	150.00
DIOGO	44444444444	DIOGO@DIOGO.COM	04-04-2004	SALVADOR	250.00
ELIS	55555555555	ELIS@ELIS.COM	05-05-2005	BRASÍLIA	50.00
FABIO	66666666666	FABIO@FABIO.COM	06-06-2006	SALVADOR	125.00
GABI	77777777777	GABI@GABI.COM	07-07-2007	BRASÍLIA	225.00
HUGO	88888888888	HUGO@HUGO.COM	08-08-2008	BRASÍLIA	50.00

TRANSAÇÃO DML

```
INSERT INTO ALUNO_ESTRATEGIA
VALUES ('ZICO', 12345678901, 'ZICO@ZICO.COM', '03-03-1953', 'RIO DE JANEIRO', 500.00);
```

ESTADO INTERMEDIÁRIO

ALUNO_ESTRATEGIA					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
ALICE	11111111111	ALICE@ALICE.COM	01-01-2001	BRASÍLIA	200.00

BRUNO	22222222222	BRUNO@BRUNO.COM	02-02-2002	SÃO PAULO	100.00
CAIO	33333333333	CAIO@CAIO.COM	03-03-2003	GOIÂNIA	150.00
DIOGO	44444444444	DIOGO@DIOGO.COM	04-04-2004	SALVADOR	250.00
ELIS	55555555555	ELIS@ELIS.COM	05-05-2005	BRASÍLIA	50.00
FABIO	66666666666	FABIO@FABIO.COM	06-06-2006	SALVADOR	125.00
GABI	77777777777	GABI@GABI.COM	07-07-2007	BRASÍLIA	225.00
HUGO	88888888888	HUGO@HUGO.COM	08-08-2008	BRASÍLIA	50.00
ZICO	12345678901	ZICO@ZICO.COM	03-03-1953	RIO DE JANEIRO	500.00

CONFIRMAÇÃO DA TRANSAÇÃO

```
COMMIT;
```

RESULTADO DO COMANDO

RESULTADO					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
ALICE	11111111111	ALICE@ALICE.COM	01-01-2001	BRASÍLIA	200.00
BRUNO	22222222222	BRUNO@BRUNO.COM	02-02-2002	SÃO PAULO	100.00
CAIO	33333333333	CAIO@CAIO.COM	03-03-2003	GOIÂNIA	150.00
DIOGO	44444444444	DIOGO@DIOGO.COM	04-04-2004	SALVADOR	250.00
ELIS	55555555555	ELIS@ELIS.COM	05-05-2005	BRASÍLIA	50.00
FABIO	66666666666	FABIO@FABIO.COM	06-06-2006	SALVADOR	125.00
GABI	77777777777	GABI@GABI.COM	07-07-2007	BRASÍLIA	225.00
HUGO	88888888888	HUGO@HUGO.COM	08-08-2008	BRASÍLIA	50.00
ZICO	12345678901	ZICO@ZICO.COM	03-03-1953	RIO DE JANEIRO	500.00

(Itaipu Binacional – 2017) Assinale a alternativa que identifica corretamente o comando SQL usado para tornar permanentes as alterações realizadas desde o início de uma transação.

a) COMMIT
b) SAVE
c) SYNC
d) FLUSH
e) APPEND

_____

Comentários: o comando usado para tornar permanentes as alterações realizadas é o COMMIT (Letra A).

(Banco da Amazônia – 2012) Uma transação é uma coleção de instruções SQL DML tratada como uma unidade lógica, de forma que não seja necessário o uso de commit, mesmo que implícito, para tornar as alterações permanentes.

_____

Comentários: na verdade, ele é utilizado com o intuito de confirmar alterações permanentes realizadas justamente por instruções DML (Errado).

ROLLBACK

INCIDÊNCIA EM PROVA: BAIXA

Comando utilizado para cancelar transações e retornar para o último estado em que foi realizado COMMIT. Se forem realizadas inserções, modificações ou remoções em uma base de dados e – por alguma razão – percebermos que essas transações não são mais necessárias, pode-se utilizar o comando **ROLLBACK** para desfazer as transações – serão desfeitas transações desde o último **COMMIT** ou **ROLLBACK**.

SINTAXE DO COMANDO

```
ROLLBACK;
```

ESTADO INICIAL

ALUNO_ESTRATEGIA

NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
ALICE	11111111111	ALICE@ALICE.COM	01-01-2001	BRASÍLIA	200.00
BRUNO	22222222222	BRUNO@BRUNO.COM	02-02-2002	SÃO PAULO	100.00
CAIO	33333333333	CAIO@CAIO.COM	03-03-2003	GOIÂNIA	150.00
DIOGO	44444444444	DIOGO@DIOGO.COM	04-04-2004	SALVADOR	250.00
ELIS	55555555555	ELIS@ELIS.COM	05-05-2005	BRASÍLIA	50.00
FABIO	66666666666	FABIO@FABIO.COM	06-06-2006	SALVADOR	125.00
GABI	77777777777	GABI@GABI.COM	07-07-2007	BRASÍLIA	225.00
HUGO	88888888888	HUGO@HUGO.COM	08-08-2008	BRASÍLIA	50.00
ZICO	12345678901	ZICO@ZICO.COM	03-03-1953	RIO DE JANEIRO	500.00

TRANSAÇÃO DML

```
UPDATE ALUNO_ESTRATEGIA
SET NOME = 'ARTHUR'
WHERE CPF = 12345678901;
```

ESTADO INTERMEDIÁRIO

ALUNO_ESTRATEGIA

NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
ALICE	11111111111	ALICE@ALICE.COM	01-01-2001	BRASÍLIA	200.00
BRUNO	22222222222	BRUNO@BRUNO.COM	02-02-2002	SÃO PAULO	100.00
CAIO	33333333333	CAIO@CAIO.COM	03-03-2003	GOIÂNIA	150.00
DIOGO	44444444444	DIOGO@DIOGO.COM	04-04-2004	SALVADOR	250.00
ELIS	55555555555	ELIS@ELIS.COM	05-05-2005	BRASÍLIA	50.00
FABIO	66666666666	FABIO@FABIO.COM	06-06-2006	SALVADOR	125.00
GABI	77777777777	GABI@GABI.COM	07-07-2007	BRASÍLIA	225.00
HUGO	88888888888	HUGO@HUGO.COM	08-08-2008	BRASÍLIA	50.00
ARTHUR	12345678901	ZICO@ZICO.COM	03-03-1953	RIO DE JANEIRO	500.00

CANCELAMENTO DA TRANSAÇÃO

```
ROLLBACK;
```

RESULTADO DO COMANDO

RESULTADO					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
ALICE	11111111111	ALICE@ALICE.COM	01-01-2001	BRASÍLIA	200.00
BRUNO	22222222222	BRUNO@BRUNO.COM	02-02-2002	SÃO PAULO	100.00
CAIO	33333333333	CAIO@CAIO.COM	03-03-2003	GOIÂNIA	150.00
DIOGO	44444444444	DIOGO@DIOGO.COM	04-04-2004	SALVADOR	250.00
ELIS	55555555555	ELIS@ELIS.COM	05-05-2005	BRASÍLIA	50.00
FABIO	66666666666	FABIO@FABIO.COM	06-06-2006	SALVADOR	125.00
GABI	77777777777	GABI@GABI.COM	07-07-2007	BRASÍLIA	225.00
HUGO	88888888888	HUGO@HUGO.COM	08-08-2008	BRASÍLIA	50.00
ZICO	12345678901	ZICO@ZICO.COM	03-03-1953	RIO DE JANEIRO	500.00

(TRE/MG – 2005) Para controlar a execução das transações, o SQL utiliza o comando:

a) Commit para desfazer todas as operações confirmadas pelo último comando Rollback.

b) Rollback para desfazer todas as operações confirmadas pelo último comando Commit.

c) Rollback para desfazer as operações confimadas por todos os comandos Commit.

d) Commit para desfazer as operações realizadas até o último comando Rollback.

e) Rollback para desfazer as operações realizadas até o último comando Commit.

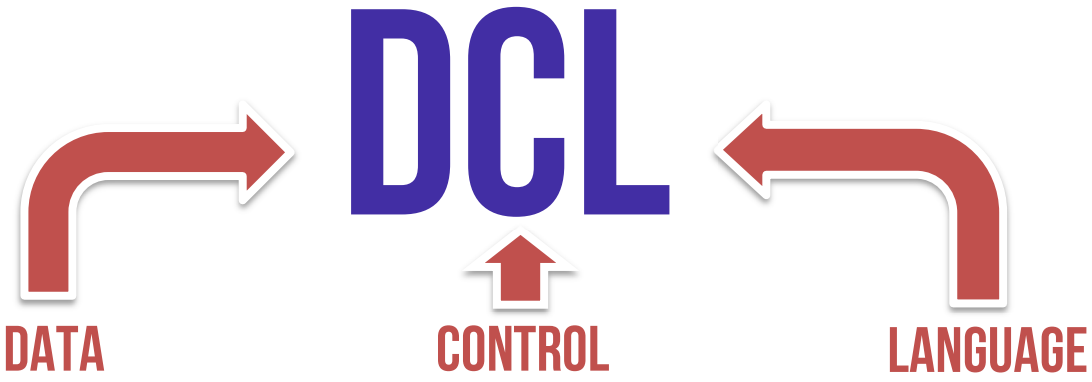
Comentários: (a) Errado. COMMIT não desfaz operações, ele confirma; (b) ROLLBACK desfaz operações realizadas e, não, confirmadas; (c) ROLLBACK desfaz operações realizadas e, não, confirmadas; (d) COMMIT não desfaz operações, ele confirma; (e) Correto. ROLLBACK desfaz operações realizadas (e ainda não confirmadas) até o último comando COMMIT (Letra E).

(MEC – 2011) Quando uma transação é abortada, todas as mudanças que ocorreram no banco de dados devem ser desfeitas. Essa operação pode ser executada por meio do comando ROLLBACK da linguagem SQL.

Comentários: perfeito... o rollback desfaz todas as mudanças não efetivadas (Correto).

DCL (DATA CONTROL LANGUAGE)

Comandos DCL



Em tradução livre, *Data Control Language* é o mesmo que Linguagem de Controle de Dados! *Logo, essa linguagem serve para quê?* Ela serve para controlar dados! *E o que isso significa?* Isso significa essa linguagem possui um conjunto de comandos que podem ser utilizados para lidar com **autorizações de dados e licenças de usuários para controlar quem tem acesso para ver ou manipular dados dentro do banco de dados**.

Como assim, professor? Galera, muitas pessoas e aplicações têm acesso a um banco de dados. *Você vai permitir que todos tenham acesso a tudo? Já imaginaram uma área atualizando dados de outra área de uma empresa?* Não vai dar certo! **Por isso, o administrador do banco de dados pode utilizar alguns comandos que permitem que determinada área só consulte dados**; outra área consiga consultar e inserir; outra só pode atualizar; e assim por diante.

COMANDOS DCL	DESCRIÇÃO
GRANT	Comando utilizado para conceder permissão a um usuário em relação a algum objeto.
REVOKE	Comando utilizado para remover/restringir a capacidade de um usuário de executar operações.

(EBSERH – 2015) Acerca dos conceitos de segurança dos sistemas de banco de dados, entre os "comandos" que estruturam o SQL, existem aqueles, que compõem um grupo, e são utilizados para atribuir as permissões que os usuários irão ter dentro de um banco de dados (GRANT, DENY, REVOKE). Eles são classificados como:

a) DML.
b) DLL.
c) DDL.
d) DSQL.
e) DCL.

_____

Comentários: todos esses comandos são exemplos de DCL – Data Control Language (Letra E).

GRANT

INCIDÊNCIA EM PROVA: BAIXA

Comando utilizado para conceder permissões a usuários em relação a objetos. Há nove funções permitidas: SELECT, INSERT, UPDATE, DELETE, REFERENCES, USAGE, UNDER, TRIGGER e EXECUTE.

SINTAXE DO COMANDO

```
GRANT LISTA_DE_PRIVILEGIOS ON OBJETO TO LISTA_DE_USUARIOS;
```

EXEMPLO DO COMANDO

```
GRANT SELECT     ON ALUNO_ESTRATEGIA TO PROFESSOR;
GRANT INSERT     ON ALUNO_ESTRATEGIA TO GERENTE;
GRANT UPDATE     ON ALUNO_ESTRATEGIA TO HEBER;
GRANT DELETE     ON ALUNO_ESTRATEGIA TO RICARDO;
GRANT REFERENCES ON ALUNO_ESTRATEGIA TO DUDU;
```

RESULTADO DO COMANDO

```
--TODOS OS COMANDOS GARANTEM ALGUM TIPO DE PERMISSÃO À TABELA ALUNO_ESTRATEGIA

SELECT      - PERMITE QUE PROFESSOR CONSULTE DADOS;
INSERT      - PERMITE QUE GERENTE INSIRA DADOS;
UPDATE      - PERMITE QUE HEBER MODIFIQUE DADOS;
DELETE      - PERMITE QUE RICARDO DELETE DADOS;
REFERENCES   - PERMITE QUE DUDU REFERENCIE OUTRA TABELA;
```

(Companhia de Águas de Joinville – 2010) Em um banco de dados, qual o objetivo do comando GRANT?

a) O comando GRANT concede permissões específicas no objeto (tabela, visão) para um ou mais usuários ou grupos de usuário.

b) O comando GRANT retorna o maior valor da coluna de uma tabela/visão armazenada no banco de dados.

c) O comando GRANT é utilizado para revogar os privilégios de um usuário.

d) O comando GRANT atribui que o campo informado será o maior que o valor máximo já existente na coluna de uma tabela ou visão de um banco de dados.

e) O comando GRANT retorna o tamanho de um objeto (variável, tabela, visão).

_____

Comentários: (a) Correto, questão perfeita; (b) Errado, esse item não faz o menor sentido; (c) Errado, essa é uma função do comando REVOKE; (d) Errado, esse item não faz o menor sentido; (e) Errado, esse item não faz o menor sentido (Letra A).

REVOKE

INCIDÊNCIA EM PROVA: BAIXA

Comando usado para revogar permissões a usuários em relação a objetos. Há nove funções: SELECT, INSERT, UPDATE, DELETE, REFERENCES, USAGE, UNDER, TRIGGER e EXECUTE.

SINTAXE DO COMANDO

```
REVOKE LISTA_DE_PRIVILEGIOS ON OBJETO FROM LISTA_DE_USUARIOS;
```

EXEMPLO DO COMANDO

```
REVOKE SELECT     ON ALUNO_ESTRATEGIA FROM PROFESSOR;
REVOKE INSERT     ON ALUNO_ESTRATEGIA FROM GERENTE;
REVOKE UPDATE     ON ALUNO_ESTRATEGIA FROM HEBER;
REVOKE DELETE     ON ALUNO_ESTRATEGIA FROM RICARDO;
REVOKE REFERENCES ON ALUNO_ESTRATEGIA FROM DUDU;
```

RESULTADO DO COMANDO

```
//TODOS OS COMANDOS REVOGAM ALGUM TIPO DE PERMISSÃO À TABELA ALUNO_ESTRATEGIA

SELECT      - REVOGA A PERMISSÃO DE QUE PROFESSOR CONSULTE DADOS;
INSERT      - REVOGA A PERMISSÃO DE QUE GERENTE INSIRA DADOS;
UPDATE      - REVOGA A PERMISSÃO DE QUE HEBER MODIFIQUE DADOS;
DELETE      - REVOGA A PERMISSÃO DE QUE RICARDO DELETE DADOS;
REFERENCES  - REVOGA A PERMISSÃO DE QUE DUDU REFERENCIE OUTRA TABELA;
```

(EBSERH – 2013) O comando REVOKE da linguagem SQL é utilizado para controle de:

a) bloqueios de transação.
b) acesso dos usuários do sistema.
c) monitoração e otimização de desempenho.
d) backup e restauração de dados.

_____

Comentários: esse comando é utilizado para controle de acesso dos usuários do sistema (Letra B).

(COBRA – 2013) Em SQL, um comando típico classificado como Linguagem de controle de dados (LCD) é:

a) SELECT. b) ALTER. c) REVOKE. d) INSERT.

_____

Comentários: o comando típico classificado como DCL (Data Control Language) é o REVOKE (Letra C).

Conceitos Avançados

Databases

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

CREATE DATABASE

Esse comando permite criar um banco de dados. Vejamos sintaxe e exemplo:

SINTAXE DO COMANDO

```
CREATE DATABASE NOME_BANCO;
```

EXEMPLO DO COMANDO

```
CREATE DATABASE BANCO_ESTRATEGIA;
```

DROP DATABASE

Esse comando permite excluir um banco de dados. Vejamos sintaxe e exemplo:

SINTAXE DO COMANDO

```
DROP DATABASE NOME_BANCO;
```

EXEMPLO DO COMANDO

```
DROP DATABASE BANCO_ESTRATEGIA;
```

Duas informações importantes: (1) ao excluir um banco de dados, os dados contidos no banco evidentemente também serão deletados, logo essa exclusão só pode ser realizada por usuários que tenham privilégios de administrador; (2) apesar de estarmos estudando em tópicos separados e ao final da aula, ambos os comandos de bancos de dados fazem parte da DDL e precedem todos os outros, visto que é necessário criar um banco de dados para poder manipulá-lo.

Views

INCIDÊNCIA EM PROVA: BAIXA

Uma visão (*view*) é uma tabela virtual baseada no conjunto de resultados de uma instrução SQL. Ela contém linhas e colunas, assim como uma tabela real. Os campos em uma visão são campos de uma ou mais tabelas reais no banco de dados. Você pode adicionar instruções e funções a uma visualização e apresentar os dados como se fossem provenientes de uma única tabela. Dito isso, vamos ver seus principais comandos a partir da tabela abaixo:

ALUNO_ESTRATEGIA					
NOME	CPF	EMAIL	DATA_NASCIMENTO	CIDADE	VALOR_PAGO
ALICE	11111111111	ALICE@ALICE.COM	01-01-2001	BRASÍLIA	200.00
BRUNO	22222222222	BRUNO@BRUNO.COM	02-02-2002	SÃO PAULO	100.00
CAIO	33333333333	CAIO@CAIO.COM	03-03-2003	GOIÂNIA	150.00
DIOGO	44444444444	DIOGO@DIOGO.COM	04-04-2004	SALVADOR	250.00
ELIS	55555555555	ELIS@ELIS.COM	05-05-2005	BRASÍLIA	50.00
FABIO	66666666666	FABIO@FABIO.COM	06-06-2006	SALVADOR	125.00
GABI	77777777777	GABI@GABI.COM	07-07-2007	BRASÍLIA	225.00
HUGO	88888888888	HUGO@HUGO.COM	08-08-2008	BRASÍLIA	50.00

CREATE VIEW

SINTAXE DO COMANDO

```
CREATE VIEW [NOME_VIEW] AS
SELECT NOME_COLUNA1, NOME_COLUNA2, ...
FROM NOME_DA_TABELA
WHERE CONDICAO;
```

EXEMPLO DO COMANDO

```
CREATE VIEW [BRASILIENSES] AS
SELECT NOME, CIDADE
FROM ALUNO_ESTRATEGIA
WHERE CIDADE = “BRASÍLIA”;
```

Vejam no exemplo anterior que nós criamos uma view para retornar apenas alunos brasilienses, logo nós fizemos uma seleção de `NOME` e `CIDADE` da tabela `ALUNO_ESTRATEGIA` em que a cidade seja `BRASÍLIA`. Opa... uma vez criada a view, eu posso consultá-la diretamente da seguinte forma: **`SELECT * FROM [BRASILIENSES]`**. Lembrando que uma visão sempre mostra dados atualizados – o mecanismo de banco de dados recria a visualização sempre que um usuário a consulta.

RESULTADO DOS COMANDOS

BRASILIENSES	
NOME	CIDADE
ALICE	BRASÍLIA
ELIS	BRASÍLIA

GABI	BRASÍLIA
HUGO	BRASÍLIA

CREATE OR REPLACE VIEW

SINTAXE DO COMANDO

```
CREATE OR REPLACE VIEW [NOME_VIEW] AS
SELECT NOME_COLUNA1, NOME_COLUNA2, ...
FROM NOME_DA_TABELA
WHERE CONDICAO;
```

EXEMPLO DO COMANDO

```
CREATE OR REPLACE VIEW [BRASILIENSES] AS
SELECT NOME, E-MAIL, CIDADE,
FROM ALUNO_ESTRATEGIA
WHERE CIDADE = “BRASÍLIA”;
```

Esse comando permite alterar uma view existente ou, caso ela ainda não exista, permite criá-la. No exemplo acima, nós alteramos a view para que ela retorne também o e-mail:

RESULTADO DOS COMANDOS

BRASILIENSES		
NOME	EMAIL	CIDADE
ALICE	ALICE@ALICE.COM	BRASÍLIA
ELIS	ELIS@ELIS.COM	BRASÍLIA
GABI	GABI@GABI.COM	BRASÍLIA
HUGO	HUGO@HUGO.COM	BRASÍLIA

DROP VIEW

SINTAXE DO COMANDO

```
DROP VIEW [NOME_VIEW];
```

EXEMPLO DO COMANDO

```
DROP VIEW [BRASILIENSES];
```

A ideia aqui é simplesmente excluir uma view. Lembrando que a manipulação de views faz parte da DDL, sendo considerada uma sublinguagem chamada VDL (*View Definition Language*).

Stored Procedures

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

Uma Stored Procedure é um código pré-preparado que você pode salvar, para que ele possa ser reutilizado em outras ocasiões repetidamente. Logo, se você possui uma consulta SQL que utiliza frequentemente, é possível salvá-la como uma espécie de procedimento armazenado e, em seguida, apenas chamá-lo para que a consulta seja executada. Lembrando que você também pode passar parâmetros de entrada para um procedimento armazenado. Vejamos...

SINTAXE DO COMANDO

```
CREATE PROCEDURE NOME_PROCEDIMENTO
      @NOME_PARAMETRO1    TIPO, @NOME_PARAMETRO2    TIPO, ...
AS
      DECLARACOES_SQL
GO;
```

EXEMPLO DO COMANDO

```
CREATE PROCEDURE RETORNA_CIDADE_ALUNO
      @CIDADE VARCHAR(20)
AS
      SELECT NOME, E-MAIL, CIDADE
      FROM ALUNO_ESTRATEGIA
      WHERE CIDADE = @CIDADE
GO;
```

CHAMADA DO COMANDO

```
EXEC RETORNA_CIDADE_ALUNO CIDADE = "BRASÍLIA";
```

RESULTADO DOS COMANDOS

BRASILIENSES	
NOME	CIDADE
ALICE	BRASÍLIA
ELIS	BRASÍLIA
GABI	BRASÍLIA
HUGO	BRASÍLIA

Note que o resultado foi idêntico ao da view. Então, você pode me perguntar: *professor, uma view é a mesma coisa que uma stored procedure?* Não, uma visão representa uma Tabela Virtual! Você pode juntar várias tabelas em uma visão e utilizá-la para apresentar os dados como se todos eles viessem de uma única tabela. Uma Stored Procedure utiliza parâmetros para executar uma atividade – seja atualizando e inserindo dados ou retornando valores únicos ou conjuntos de dados.

Ordens de Precedência

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

Vamos analisar a seguinte query:

SINTAXE DO COMANDO

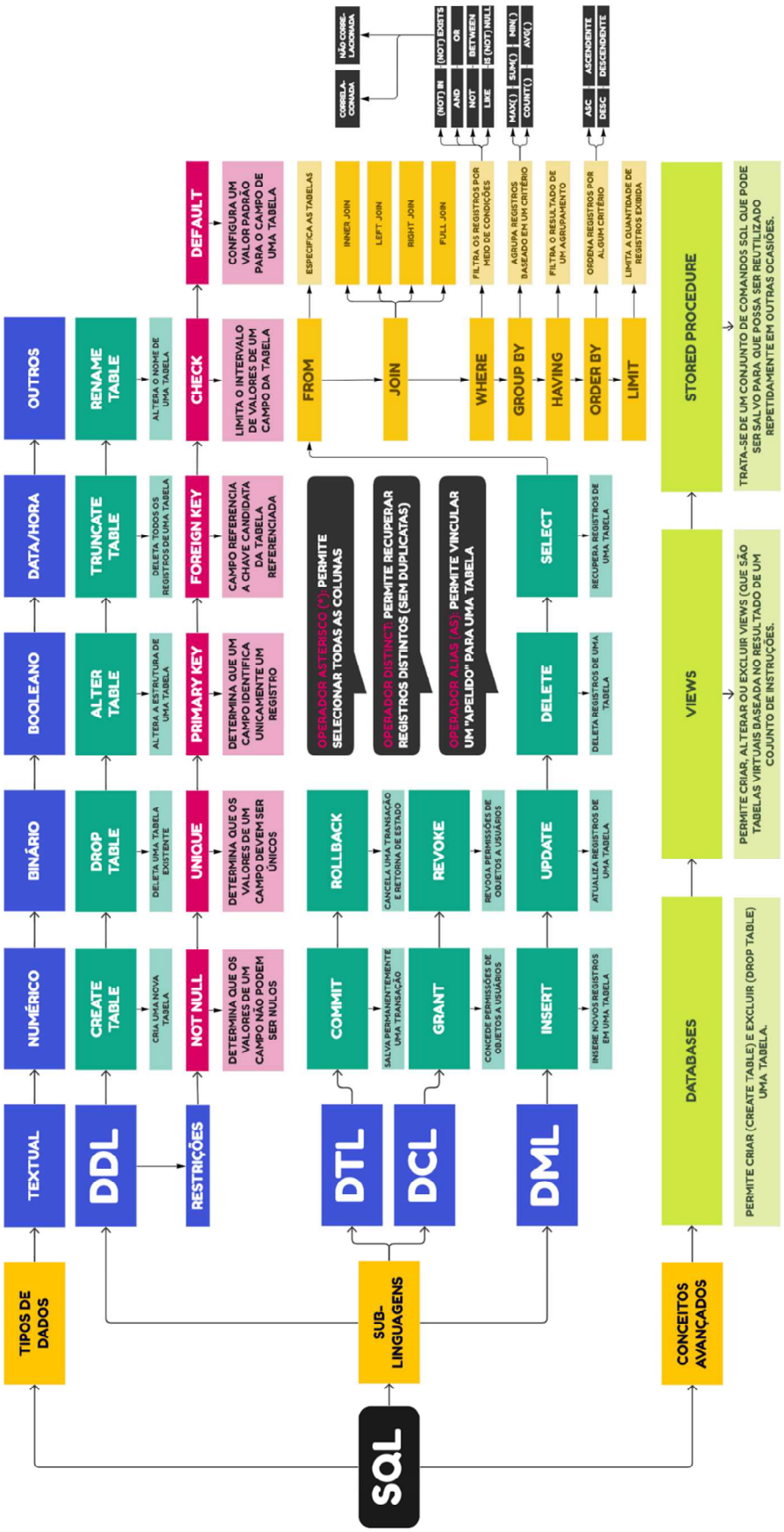
```
SELECT ...
FROM ...
WHERE COLUNA IN (1,2,3) OR COLUNA IN (4,5) AND ...
```

Note que nós temos vários operadores! *E aí, qual tem precedência sobre qual?* É rara a cobrança desse assunto em prova, mas é importante saber que os operadores seguem uma precedência[1]...

1	Operadores Aritméticos (Ex: +, -, *, /)
2	Operadores de Concatenação (Ex: II)
3	Operadores de Comparação (Ex: =, >, <, >=, <=)
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	NOT EQUAL TO
7	NOT
8	AND
9	OR

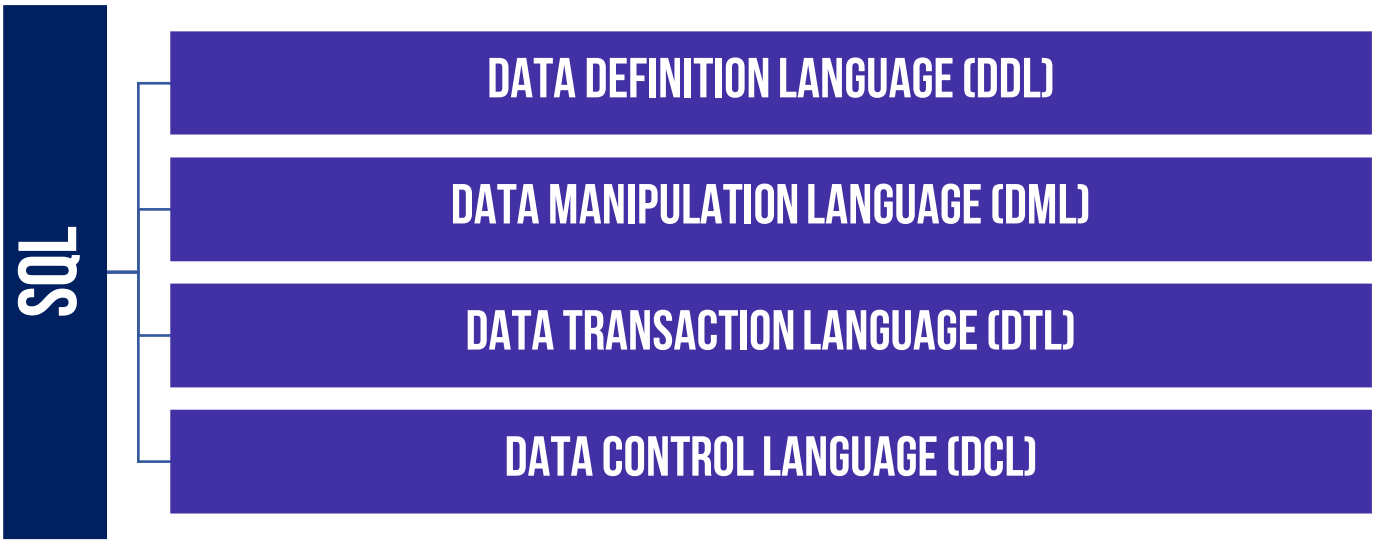
[1] Lembrando que a utilização de parênteses sobrescreve essas regras de precedência.

RESUMO


SQL
SUB-LINGUAGENS
TIPOS DE DADOS
CONCEITOS AVANÇADOS
TEXTUAL
DDL
DTL
DCL
DML
RESTRIÇÕES
DATABASES
PERMITE CRIAR (CREATE TABLE) E EXCLUIR (DROP TABLE) UMA TABELA.
VIEWS
PERMITE CRIAR, ALTERAR OU EXCLUIR VIEWS (QUE SÃO TABELAS VIRTUAIS BASEADA NO RESULTADO DE UM CONJUNTO DE INSTRUÇÕES.
STORED PROCEDURE
TRATA-SE DE UM CONJUNTO DE COMANDOS SQL QUE PODE SER SALVO PARA QUE POSSA SER REUTILIZADO REPETIDAMENTE EM OUTRAS OCASIÕES.
NUMÉRICO
BINÁRIO
BOOLEANO
DATA/HORA
OUTROS
CREATE TABLE
DROP TABLE
ALTER TABLE
TRUNCATE TABLE
RENAME TABLE
CRIA UMA NOVA TABELA
DELETA UMA TABELA EXISTENTE
ALTERA A ESTRUTURA DE UMA TABELA
DELETA TODOS OS REGISTROS DE UMA TABELA
ALTERA O NOME DE UMA TABELA
NOT NULL
UNIQUE
PRIMARY KEY
FOREIGN KEY
CHECK
DEFAULT
DETERMINA QUE OS VALORES DE UM CAMPO NÃO PODEM SER NULOS
DETERMINA QUE OS VALORES DE UM CAMPO DEVEM SER ÚNICOS
DETERMINA QUE UM CAMPO IDENTIFICA UNICAMENTE UM REGISTRO
CAMPO REFERENCIA A CHAVE CANDIDATA DA TABELA REFERENCIADA
LIMITA O INTERVALO DE VALORES DE UM CAMPO DA TABELA
CONFIGURA UM VALOR PADRÃO PARA O CAMPO DE UMA TABELA
COMMIT
ROLLBACK
GRANT
REVOKE
SALVA PERMANENTEMENTE UMA TRANSAÇÃO
CANCELA UMA TRANSAÇÃO E RETORNA DE ESTADO
CONCEDE PERMISSÕES DE OBJETOS A USUÁRIOS
REVOGA PERMISSÕES DE OBJETOS A USUÁRIOS
INSERT
UPDATE
DELETE
SELECT
INSERE NOVOS REGISTROS EM UMA TABELA
ATUALIZA REGISTROS DE UMA TABELA
DELETA REGISTROS DE UMA TABELA
RECUPERA REGISTROS DE UMA TABELA
OPERADOR ASTERISCO (*): PERMITE SELECIONAR TODAS AS COLUNAS
OPERADOR DISTINCT: PERMITE RECUPERAR REGISTROS DISTINTOS (SEM DUPLICATAS)
OPERADOR ALIAS (AS): PERMITE VINCULAR UM "APELIDO" PARA UMA TABELA
FROM
JOIN
WHERE
GROUP BY
HAVING
ORDER BY
LIMIT
ESPECIFICA AS TABELAS
INNER JOIN
LEFT JOIN
RIGHT JOIN
FULL JOIN
FILTRA OS REGISTROS POR MEIO DE CONDIÇÕES
AGRUPA REGISTROS BASEADO EM UM CRITÉRIO
FILTRA O RESULTADO DE UM AGRUPAMENTO
ORDENA REGISTROS POR ALGUM CRITÉRIO
LIMITA A QUANTIDADE DE REGISTROS EXIBIDA
CORRELACIONADA
NÃO CORRELACIONADA
(NOT) IN
AND
NOT
LIKE
(NOT) EXISTS
OR
BETWEEN
IS (NOT) NULL
MAX()
SUM()
COUNT()
MIN()
AVG()
ASC
DESC
ASCENDENTE
DESCENDENTE

DEFINIÇÃO DE SQL
SQL é basicamente a linguagem padrão declarativa para manipulação de bancos de dados relacionais. Por meio dela, um usuário pode executar comandos para inserir, pesquisar, atualizar ou deletar registros em um banco de dados, criar ou excluir tabelas, conceder ou revogar permissões para acessar o banco de dados, entre diversos outros recursos.

TIPOS DE DADOS	DESCRIÇÃO
TEXTUAL	Esse tipo de dado – também conhecido como literal – é basicamente uma cadeia de caracteres – também chamada de string.
NUMÉRICO	Esse tipo de dado trata de números inteiros de diversos tamanhos (Ex: `INTEGER`, `INT` ou `SMALLINT`) e números reais (`FLOAT`, `REAL` e `DOUBLE PRECISION`).
BINÁRIO	Esse tipo de dado é basicamente uma cadeia de bits de tamanho fixo (Ex: `BIT(n)`, em que n é o número fixo de caracteres) ou tamanho variável (Ex: `BIT VARYING(n)`, em que n é o número máximo de caracteres.
BOOLEANO	Esse tipo de dado tem como valores tradicionais `TRUE` (Verdadeiro) ou `FALSE` (Falso).
DATA	Esse tipo de dado possui dez posições, e seus componentes são `DAY` (Dia), `MONTH` (Mês) e `YEAR` (Ano) na forma `DD-MM-YYYY` (Ex: 30/03/2019).
HORA	Esse tipo de dado possui pelo menos oito posições compostas por `HOUR` (Hora), `MINUTE` (Minuto) e `SECOND` (Segundo) na forma `HH:MM:SS`.
OUTROS	Existem outros tipos de dados que foram acrescentados em versões posteriores do Padrão ANSI/SQL (Ex: `TIMESTAMP` (junção da Data com Hora), `INTERVAL` (calcula o intervalo entre Datas/Horas); e `DATETIME` (combina data e hora em um único tipo, com intervalo de datas)).



COMANDOS DDL	DESCRIÇÃO
CREATE	Comando utilizado para criar tabelas (e outros objetos) de um banco de dados.

DROP	Comando utilizado para deletar uma tabela (e outros objetos) de um banco de dados.
TRUNCATE	Comando utilizado para apagar dados de uma tabela (e, não, a tabela em si) de um banco de dados.
ALTER	Comando utilizado para manipular colunas ou restrições de um banco de dados.
RENAME	Comando utilizado para renomear uma tabela de um banco de dados.

CREATE TABLE

SINTAXE DO COMANDO

```
CREATE TABLE NOME_DA_TABELA (
      NOME_COLUNA1              TIPO_DE_DADO       RESTRIÇÕES   ,
      NOME_COLUNA2              TIPO_DE_DADO       RESTRIÇÕES   ,
      NOME_COLUNA3              TIPO_DE_DADO       RESTRIÇÕES   ,
      ...
      );
```

SINTAXE DO COMANDO

```
CREATE TABLE NOME_TABELA_NOVA AS
      SELECT NOME_COLUNA1, NOME_COLUNA2, NOME_COLUNA3, ...
      FROM   NOME_TABELA_ANTIGA
      WHERE ...
```

DROP TABLE

SINTAXE DO COMANDO

```
DROP TABLE NOME_DA_TABELA;
```

TRUNCATE TABLE

SINTAXE DO COMANDO

```
TRUNCATE TABLE NOME_DA_TABELA;
```

ALTER TABLE

SINTAXE DO COMANDO PARA ADICIONAR COLUNA

```
ALTER TABLE NOME_DA_TABELA
ADD COLUMN NOME_COLUNA TIPO_DE_DADO;
```

SINTAXE DO COMANDO PARA EXCLUIR COLUNA

```
ALTER TABLE NOME_DA_TABELA
DROP COLUMN NOME_COLUNA;
```

SINTAXE DO COMANDO (SQL SERVER / MS-ACCESS)

```
ALTER TABLE NOME_DA_TABELA
ALTER COLUMN NOME_COLUNA TIPO_DE_DADO;
```

SINTAXE DO COMANDO (MYSQL / ORACLE PRÉ-10G)

```
ALTER TABLE NOME_DA_TABELA
MODIFY COLUMN NOME_COLUNA TIPO_DE_DADO;
```

SINTAXE DO COMANDO (MYSQL / ORACLE 10G EM DIANTE)

```
ALTER TABLE NOME_DA_TABELA
MODIFY NOME_COLUNA TIPO_DE_DADO;
```

RENAME TABLE

SINTAXE DO COMANDO

```
RENAME TABLE NOME_DA_TABELA
TO NOVO_NOME_DA_TABELA;
```

EXEMPLO DO COMANDO

```
RENAME TABLE ALUNO
TO ALUNO_ESTRATEGIA;
```

SINTAXE DO COMANDO

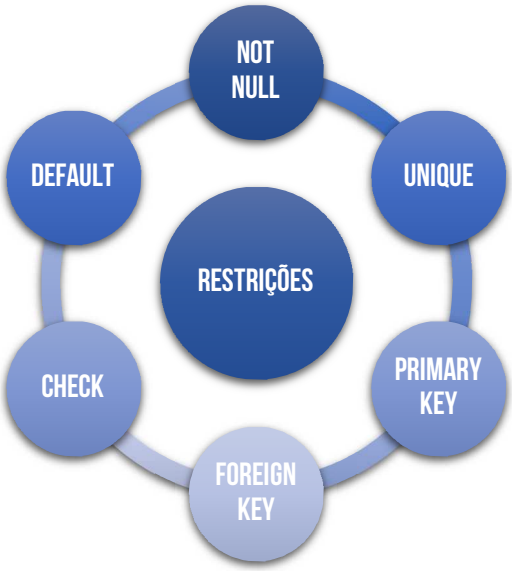
```
ALTER TABLE NOME_DA_TABELA
RENAME TO NOVO_NOME_DA_TABELA;
```

RESTRIÇÕES

RESTRIÇÕES (CONSTRAINTS)
Trata-se de um conjunto de limitações utilizadas para especificar regras para os dados em uma tabela de um banco de dados relacional. Elas buscam limitar o tipo de dado que pode ser armazenado, o que garante a precisão e confiabilidade aos dados da tabela. Se houver qualquer violação entre a restrição e a ação de dados, a ação será abortada.

CONSTRAINT	DESCRIÇÃO
NOT NULL	Garante que uma coluna não possa ter um valor nulo.
UNIQUE	Garante que todos os valores de uma coluna sejam diferentes entre si.
PRIMARY KEY	Garante que todos os valores de uma coluna sejam diferentes entre si e não nulos.

FOREIGN KEY	Garante que ações não destruam links/relacionamentos entre as tabelas.
CHECK	Garante que os valores em uma coluna satisfaçam uma condição específica.
DEFAULT	Define um valor padrão para uma coluna, se nenhum valor for especificado.



NOT NULL

SINTAXE DO COMANDO

```
CREATE TABLE NOME_DA_TABELA (
       NOME_COLUNA1        TIPO_DE_DADO        NOT NULL     ,
       ...
       );
```

UNIQUE

EXEMPLO DO COMANDO (SQL SERVER / ORACLE / MS-ACCESS)

```
CREATE TABLE ALUNO (
       NOME                VARCHAR(20)         NOT NULL           ,
       CPF                 INT                 PRIMARY KEY        ,
       SEXO                CHAR(1)             NOT NULL           ,
       DATA_NASCIMENTO     DATE                NOT NULL           ,
       CIDADE              VARCHAR(50)                            ,
       MATRICULA           INT                 UNIQUE             ,
       VALOR_PAGO          INT
       );
```

PRIMARY KEY

SINTAXE DO COMANDO (SQL SERVER / ORACLE / MS-ACCESS)

```
CREATE TABLE NOME_DA_TABELA (
       NOME_COLUNA1              TIPO_DE_DADO        PRIMARY KEY  ,
       NOME_COLUNA2              TIPO_DE_DADO        RESTRIÇÕES   ,
```

```
      NOME_COLUNA3              TIPO_DE_DADO       RESTRIÇÕES   ,
      ...
      );
```

FOREIGN KEY

SINTAXE DO COMANDO (SQL SERVER / ORACLE / MS-ACCESS)

```
CREATE TABLE NOME_DA_TABELA (
      NOME_COLUNA1  TIPO_DE_DADO  PRIMARY KEY      ,
      NOME_COLUNA2  TIPO_DE_DADO  RESTRIÇÕES       ,
      NOME_COLUNA3  TIPO_DE_DADO  FOREIGN KEY REFERENCES TABELA_REFERENCIADA (CHAVE)
      );
```

CHECK

EXEMPLO DO COMANDO (SQL SERVER / ORACLE / MS-ACCESS)

```
CREATE TABLE ALUNO (
      NOME              VARCHAR(20)       NOT NULL          ,
      CPF               INT               PRIMARY KEY       ,
      SEXO              CHAR(1)           NOT NULL          ,
      CIDADE            VARCHAR(50)                         ,
      MATRICULA         INT               UNIQUE            ,
      IDADE             INT               CHECK (IDADE >= 18)
      );
```

DEFAULT

EXEMPLO DO COMANDO (MYSQL / SQL SERVER / ORACLE / MS-ACCESS)

```
CREATE TABLE ALUNO (
      NOME              VARCHAR(20)       NOT NULL          ,
      CPF               INT               PRIMARY KEY       ,
      SEXO              CHAR(1)           NOT NULL          ,
      CIDADE            VARCHAR(50)       DEFAULT 'Brasília' ,
      MATRICULA         INT               UNIQUE            ,
      );
```

COMANDOS DML	DESCRIÇÃO
SELECT	Comando utilizado para realizar consultas a dados de uma ou mais tabelas do banco de dados.

INSERT	Comando utilizado para inserir um registro em uma tabela do banco de dados.
UPDATE	Comando utilizado para mudar valores de dados de registros de uma tabela do banco de dados.
DELETE	Comando utilizado para remover registros de uma tabela do banco de dados.

INSERT INTO

SINTAXE DO COMANDO I

```
-- INSERÇÃO DE TODOS OS VALORES PRESCINDE DA ESPECIFICAÇÃO DAS COLUNAS
INSERT INTO NOME_DA_TABELA
VALUES (VALOR_1, VALOR_2, VALOR_3, ...)
```

SINTAXE II DO COMANDO II

```
-- INSERÇÃO DE TODOS OS VALORES PRECISA DA ESPECIFICAÇÃO DAS COLUNAS
INSERT INTO NOME_DA_TABELA (NOME_COLUNA1, NOME_COLUNA2, NOME_COLUNA3, ...)
VALUES (VALOR_1, VALOR_2, VALOR_3, ...)
```

UPDATE

SINTAXE DO COMANDO

```
UPDATE NOME_DA_TABELA
SET NOME_DA_COLUNA_1 = VALOR_1, NOME_COLUNA2 = VALOR_2 ...
WHERE LISTA_DE_CONDICOES
```

DELETE

SINTAXE DO COMANDO

```
DELETE FROM NOME_DA_TABELA WHERE LISTA_DE_CONDICOES
```

SELECT

SINTAXE DO COMANDO

```
-- AS CLÁUSULAS SÃO OPCIONAIS
SELECT LISTA_DE_COLUNAS FROM LISTA_DE_TABELAS CLAUSULAS;
```

SELECT DISTINCT

SINTAXE DO COMANDO

```
-- AS CLÁUSULAS SÃO OPCIONAIS
SELECT DISTINCT LISTA_DE_COLUNAS FROM LISTA_DE_TABELAS CLAUSULAS;
```

ALIASES

SINTAXE DO COMANDO

```
-- ALIAS PARA O NOME DA TABELA
SELECT NOME_COLUNA FROM NOME_DA_TABELA AS APELIDO CLAUSULAS;

-- ALIAS PARA O NOME DA COLUNA
SELECT NOME_COLUNA AS APELIDO FROM NOME_DA_TABELA CLAUSULAS;
```

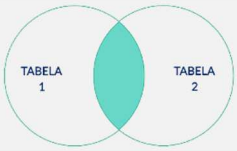
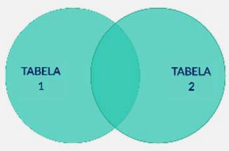
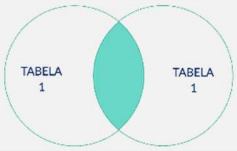
CLÁSULAS	DESCRIÇÃO
FROM	Comando utilizado para indicar de onde os dados devem ser selecionados.
JOIN	Comando utilizado para combinar linhas tabelas, com base em uma coluna em comum entre elas.
WHERE	Comando utilizado para filtrar os dados.
GROUP BY	Comando utilizado para agregar um conjunto de dados.
HAVING	Comando utilizado para filtrar dados agregados.
ORDER BY	Comando utilizado para ordenar os dados recuperados.
LIMIT	Comando utilizado para limitar a quantidade de resultados.

FROM

SINTAXE DO COMANDO

```
SELECT LISTA_DE_COLUNAS FROM TABELA1, TABELA2, ... CLAUSULAS;
```

JOIN

INNER JOIN	LEFT JOIN	RIGHT JOIN	FULL OUTER JOIN	SELF JOIN
				
Retorna registros que possuem valores correspondentes em ambas as tabelas	Retorna todos os registros da tabela da esquerda e seus correspondentes da tabela da direita	Retorna todos os registros da tabela da direita e seus correspondentes da tabela da esquerda	Retorna todos os registros quando há uma correspondência na tabela da esquerda ou da direita	Trata-se de join comum, mas que relaciona registros de uma tabela com ela mesma

WHERE

SINTAXE DO COMANDO

```
SELECT NOME_COLUNA1, NOME_COLUNA2, ...
FROM NOME_DA_TABELA1
WHERE CONDICAO;
```

OPERADORES RELACIONAIS

OPERADOR	DESCRIÇÃO	EXEMPLO
=	IGUAL	... WHERE NOME = 'DIEGO';
>	MAIOR	... WHERE VALOR_PAGO > 1000.00;
>=	MAIOR OU IGUAL	... WHERE IDADE >= 18;
<	MENOR	... WHERE DATA_NASCIMENTO < '01/01/2000';
<=	MENOR OU IGUAL	... WHERE VELOCIDADE <= 100;
<>	DIFERENTE	... WHERE CIDADE <> 'São Paulo';

OPERADORES AND, OR, NOT, BETWEEN

SINTAXE DO COMANDO

```
SELECT NOME_COLUNA1, NOME_COLUNA2, ...
FROM NOME_DA_TABELA1
WHERE CONDICAO1 AND CONDICAO2 AND CONDICAO3 ... ;
```

SINTAXE DO COMANDO

```
SELECT NOME_COLUNA1, NOME_COLUNA2, ...
FROM NOME_DA_TABELA1
WHERE CONDICAO1 OR CONDICAO2 OR CONDICAO3 ... ;
```

SINTAXE DO COMANDO

```
SELECT NOME_COLUNA1, NOME_COLUNA2, ...
FROM NOME_DA_TABELA1
WHERE NOT CONDICAO1;
```

SINTAXE DO COMANDO

```
SELECT NOME_COLUNA1, NOME_COLUNA2, ...
FROM NOME_DA_TABELA1
WHERE NOME_COLUNA1 BETWEEN VALOR1 AND VALOR2;
```

OPERADOR LIKE

SINTAXE DO COMANDO

```
SELECT NOME_COLUNA1, NOME_COLUNA2, ...
FROM NOME_DA_TABELA1
WHERE NOME_COLUNA1 LIKE PADRAO
```

OPERADOR	DESCRIÇÃO
...WHERE NOME LIKE 'A%'	Retorna valores que comecem com "A".

`...WHERE NOME LIKE '%A'`	Retorna valores que terminem com "A".
`...WHERE NOME LIKE '%IO%'`	Retorna valores que possuam "IO" em qualquer posição.
`...WHERE NOME LIKE '_R%'`	Retorna valores que possuam um caractere e depois a letra "R".
`...WHERE NOME LIKE '%A_'`	Retorna valores que terminem com "A" mais apenas um caractere.
`...WHERE NOME LIKE 'A__%'`	Retorna valores que comecem com "A" e possuem ao menos 3 caracteres.
`...WHERE NOME LIKE '%A%O'`	Retorna valores que possuam "A" depois "O" (imediatamente ou não).

OPERADOR IS NULL

SINTAXE DO COMANDO

```
SELECT NOME_COLUNA1, NOME_COLUNA2, ...
FROM NOME_DA_TABELA1
WHERE NOME_COLUNA1 IS NULL;
```

OPERADOR IN

SINTAXE DO COMANDO

```
SELECT NOME_COLUNA1, NOME_COLUNA2, ...
FROM NOME_DA_TABELA1
WHERE NOME_COLUNA1 IN (VALOR1, VALOR2,...);
```

OPERADOR EXISTS

-

SINTAXE DO COMANDO

```
SELECT NOME_COLUNA1, NOME_COLUNA2, ...
FROM NOME_DA_TABELA1
WHERE EXISTS (SELECT ... FROM ... WHERE ...);
```

GROUP BY

SINTAXE DO COMANDO

```
SELECT LISTA_DE_COLUNAS, FUNCAO_DE_AGREGACAO(COLUNA)
FROM NOME_DA_TABELA
WHERE CONDIÇOES --OPCIONAL
GROUP BY LISTA_DE_COLUNAS;
```

FUNÇÕES	AGREGAÇÃO	DESCRIÇÃO
`COUNT()`	Quantidade	Essa função conta a quantidade total de dados de um dado campo.
`SUM()`	Soma	Essa função soma valores numéricos de um dado campo.
`AVG()`	Média	Essa função calcula a média aritmética simples de um conjunto de valores numéricos.
`MAX()`	Máximo	Essa função retorna o maior valor encontrado de um dado campo.
`MIN()`	Mínimo	Essa função retorna o menor valor encontrado de um dado campo.

HAVING

SINTAXE DO COMANDO

```
SELECT LISTA_DE_COLUNAS, FUNCAO_DE_AGREGACAO(COLUNA)
FROM NOME_DA_TABELA
WHERE CONDIÇOES --OPCIONAL
GROUP BY LISTA_DE_COLUNAS
HAVING CONDIÇOES;
```

ORDER BY

SINTAXE DO COMANDO

```
SELECT LISTA_DE_COLUNAS, FUNCAO_DE_AGREGACAO(COLUNA)
FROM NOME_DA_TABELA
WHERE CONDIÇOES --OPCIONAL
GROUP BY LISTA_DE_COLUNAS --OPCIONAL
HAVING CONDIÇOES --OPCIONAL
ORDER BY COLUNA1 ASC | DESC, COLUNA2 ASC | DESC, ...;
```

LIIMT

SINTAXE DO COMANDO

```
SELECT LISTA_DE_COLUNAS
FROM NOME_DA_TABELA
LIMIT QTD_LINHAS;
```

COMANDOS DTL	DESCRIÇÃO
COMMIT	Comando utilizado para finalizar/confirmar uma transação dentro de um SGBD.

ROLLBACK	Comando utilizado para descartar mudanças nos dados desde o último COMMIT ou ROLLBACK.

SINTAXE DO COMANDO

```
COMMIT;
```

SINTAXE DO COMANDO

```
ROLLBACK;
```

COMANDOS DCL	DESCRIÇÃO
GRANT	Comando utilizado para conceder permissão a um usuário em relação a algum objeto.
REVOKE	Comando utilizado para remover/restringir a capacidade de um usuário de executar operações.

SINTAXE DO COMANDO

```
GRANT LISTA_DE_PRIVILEGIOS ON OBJETO TO LISTA_DE_USUARIOS;
```

SINTAXE DO COMANDO

```
REVOKE LISTA_DE_PRIVILEGIOS ON OBJETO FROM LISTA_DE_USUARIOS;
```

DATABASES

SINTAXE DO COMANDO

```
CREATE DATABASE NOME_BANCO;
```

SINTAXE DO COMANDO

```
DROP DATABASE NOME_BANCO;
```

VIEWS

SINTAXE DO COMANDO

```
CREATE VIEW [NOME_VIEW] AS
SELECT NOME_COLUNA1, NOME_COLUNA2, ...
FROM NOME_DA_TABELA
WHERE CONDICAO;
```

SINTAXE DO COMANDO

```
CREATE OR REPLACE VIEW [NOME_VIEW] AS
SELECT NOME_COLUNA1, NOME_COLUNA2, ...
FROM NOME_DA_TABELA
WHERE CONDICAO;
```

SINTAXE DO COMANDO

```
DROP VIEW [NOME_VIEW];
```

STORED PROCEDURES

SINTAXE DO COMANDO

```
CREATE PROCEDURE NOME_PROCEDIMENTO
      @NOME_PARAMETRO1    TIPO, @NOME_PARAMETRO2    TIPO, ...
AS
      DECLARACOES_SQL
GO;
```

CHAMADA DO COMANDO

```
EXEC RETORNA_CIDADE_ALUNO CIDADE = “BRASÍLIA”;
```

QUESTÕES COMENTADAS – CESPE

1. **(CESPE / SEFAZ-SE – 2022)** A respeito do código SQL (Structured Query Language) anteriormente apresentado, assinale a opção correta.

```
select C.CPF as CPF, C.NOME as NOME
from CONTRIBUINTE as C, PARCELAMENTO as P
where C.CPF=P.CPF
and P.TIPO='IPVA'
and P.DATAADESAO between '01/01/2021' and
'31/12/2021'
and P.STATUS='ADIMPLENTE'
```

a) Há um erro de sintaxe em `and P.DATAADESAO between '01/01/2021' and '31/12/2021'`, pois não é permitida a utilização do operador `and` mais de uma vez na mesma linha.

b) Há uma junção (JOIN) nesse código, a qual é especificada no trecho `from CONTRIBUINTE as C, PARCELAMENTO as P.`

c) O objetivo do código é mostrar o CPF e o nome de todos os contribuintes que não aderiram ao programa de parcelamento do IPVA no ano de 2021.

d) A palavra reservada `between` foi inserida no código equivocadamente, pois somente deveria ser usada nos comandos de `update` e `delete`.

e) A finalidade do código é mostrar o CPF e o nome de todos os contribuintes que aderiram ao parcelamento do IPVA no ano de 2021 e que estão com o seu parcelamento em dia.

Comentários:

(a) Errado, pode ser utilizado mais de uma vez na mesma linha; (b) Errado, isso é um produto cartesiano e, não, uma junção; (c) Errado, o objetivo é mostrar o CPF e NOME de todos os contribuintes que aderiram ao programa de parcelamento do IPVA no ano de 2021; (d) Errado, não existe essa restrição; (e) Correto.

Gabarito: Letra E

---

2. **(CESPE / PETROBRAS – 2022)** Por meio do comando SQL a seguir, é possível recuperar o nome dos pesquisadores responsáveis por projetos, seguido pelo nome de seu orientador, mas apenas os projetos orientados por **Pedro**.

```
select responsavel.nome nomeresponsavel,
orientador.nome    nomeorientador,
tituloProjeto
```

```
from Pesquisador responsavel, Pesquisador orientador, Projeto
where orientador.nome = 'Pedro'
andcodPesquisadorResponsavel = codPesquisador
and codPesquisadorOrientador = codPesquisador;
```

Comentários:

No diagrama, temos dois tipos de relacionamentos entre pesquisador e projeto: responsável e orientador. Podemos inferir que existe o pesquisador responsável por um projeto, assim como um pesquisador orientador de um projeto. Logo, uma mesma entidade pode exercer papeis diferentes e, por conta disso, a questão dá apelidos (*alias*) diferentes para a entidade Pesquisador. Dessa forma, podemos concluir que:

```
1. Pesquisador = responsavel;
2. Pesquisador = orientador;
3. responsavel.nome = nomeresponsavel;
4. orientador.nome = nomeorientador.
```

Pronto, isso facilita bastante a leitura do código! Nas três primeiras linhas, podemos identificar que o código deseja retornar o nome do responsável, nome do orientador e o título de um projeto. *E onde essa consulta deverá ser feita?* Do produto cartesiano entre **responsavel**, **orientador** e **Projeto**. Do resultado desse produto cartesiano, devemos filtrar as linhas em que o orientador seja chamado "Pedro" e o código único de responsável seja igual ao código único de pesquisador, assim como o código único de orientador seja igual ao código único de pesquisador.

Agora que vem o problema: *que código único de pesquisador?* A questão menciona apenas **codPesquisador**, mas esse atributo pode ser de **responsavel** ou de **orientador**. *E agora?* Há uma ambiguidade e esse é o erro da questão! Para estar correta, ela deveria colocar os alias corretos conforme é apresentado a seguir:

```
where orientador.nome = 'Pedro'
and codPesquisadorResponsavel = responsavel.codPesquisador
and codPesquisadorOrientador = orientador.codPesquisador;
```

Gabarito: Errado

---

3. **(CESPE / Petrobrás - 2022)** Duas expressões SQL são equivalentes se e somente se elas tiverem os mesmos comandos em suas respectivas sequências.

Comentários:

Se duas expressões SQL são tem os mesmos comandos, elas serão equivalentes. Entretanto, se elas tiverem comandos diferentes, ainda é possível que elas sejam equivalentes, basta que o resultado das expressões seja o mesmo.

Gabarito: Errado

---

4. **(CESPE / Petrobrás - 2022)** O comando truncate PESSOA; permite excluir todos os registros da tabela de nome PESSOA.

Comentários:

Perfeito! O comando TRUNCATE deleta todos os registros de uma tabela (mas mantém a estrutura).

Gabarito: Correto

---

5. **(CESPE / Petrobrás - 2022)** A expressão SQL a seguir está sintaticamente correta e permite inserir dois alunos de nomes Pedro e Maria na tabela alunos.

   INSERT VALUES ('Pedro', 'Maria') INTO alunos;

Comentários:

A sintaxe correta para o comando INSERT é a seguinte:
INSERT INTO NOME_DA_TABELA
VALUES (VALOR_1, VALOR_2, VALOR_3, ...)

Desse modo, o correto seria:
INSERT INTO alunos VALUES ('Pedro', 'Maria');

Gabarito: Errado

---

6. **(CESPE / TJ-RJ - 2021)** Processo (codprocesso, autor, reu, dataultimamovimentacao, assunto, codjuiz) Juiz (codjuiz, nome).
   Considerando as tabelas anteriores, de um banco de dados relacional, assinale a opção cuja consulta em SQL mostra os nomes dos juízes para os quais não há processos distribuídos (relacionados).

   a) SELECT J.nome
   FROM Juiz AS J
   WHERE J.nome NOT IN (SELECT P.codjuiz
   FROM Processo AS P);

   b) SELECT J.nome
   FROM Juiz AS J, Processo AS P
   WHERE J.codjuiz inner join P.codjuiz;

c) SELECT J.nome
FROM Juiz AS J
WHERE J.codjuiz NOT IN (SELECT P.codjuiz
FROM Processo AS P);

d) SELECT J.nome
FROM Juiz AS J
WHERE J.codjuiz LIKE (SELECT P.codjuiz
FROM Processo AS P);

e) SELECT J.nome
FROM Juiz AS J, Processo AS P
WHERE J.nome NOT EXISTS (P.codjuiz);

Comentários:

(a) Errado, o WHERE procura pelo nome do juiz (J.nome) e a consulta interna procura pelo código do juiz (P.codjuiz) não retornando nenhum resultado; (b) Errado, essa opção faz um inner join; (c) Correto, essa opção irá retornar os nomes dos juízes que não tem processos distribuídos; (d) Errado, essa opção utiliza o operador LIKE; (e) Errado, essa opção utiliza o operador NOT EXISTS com o código do juiz.

Gabarito: Letra C

---

7. (CESPE / DPE-RO – 2021)

```
create table aluno (
    id integer not null primary key,
    nome varchar,
    datanascimento date
);
create table cidade (
    ibge bigint not null primary key,
    município varchar
);

create table alunocidade (
    cidade bigint,
    aluno integer,
   tipo varchar,
    constraint fkcidade foreign key (cidade)
references cidade,
    constraint fkaluno foreign key (aluno)
references aluno,
    constraint pkcidade primary key
```

```
(cidade,aluno,tipo)
);
```

Para a expressão SQL anterior, a cardinalidade entre as entidades aluno e cidade é:

a) zero-para-muitos.
b) muitos-para-muitos.
c) um-para-um.
d) muitos-para-um.
e) um-para-muitos.

Comentários:

De acordo com o código SQL, temos três tabelas: ALUNO, CIDADE e ALUNOCIDADE. A tabela ALUNOCIDADE é chamada de tabela associativa – ela normalmente surge em um relacionamento com cardinalidade N:N. Ademais, constraints (restrições) são um conjunto de limitações utilizadas para especificar regras para os dados em uma tabela de um banco de dados relacional. Não havendo nenhuma restrição para as cardinalidades, trata-se de um relacionamento N:N (muitos-para-muitos).

Gabarito: Letra B

---

8. **(CESPE / DPE-RO – 2021)** Assinale a opção que apresenta o comando SQL usado para excluir todos os registros de uma tabela de nome aluno, mantendo-se a estrutura da tabela:

a) delete aluno
b) erase aluno
c) erase from aluno
d) delete from aluno
e) drop from aluno

Comentários:

O comando DELETE é utilizado para remover registros de uma tabela do banco de dados. *Professor, não poderia ser o DROP?* Não, o DROP é o comando utilizado para deletar uma tabela de um banco de dados junto com seus dados. Na questão, queremos excluir apenas os registros!

Gabarito: Letra D

---

9. **(CESPE / APEX-BRASIL – 2021)** *create database pessoa*;

O comando SQL apresentado anteriormente criará:

a) um banco de dados denominado pessoa;

b) uma tabela denominada pessoa;
c) um tipo de dados denominado pessoa;
d) um esquema denominado pessoa;

Comentários:

Esse comando criará um banco de dados denominado pessoa.

Gabarito: Letra A

---

10. (CESPE / Polícia Federal – 2021) Na linguagem SQL (*structured query language*), DTL (*data transaction language*) são comandos responsáveis por gerenciar diferentes transações ocorridas dentro de um banco de dados.

Comentários:

Perfeito! Uma das sublinguagens da Linguagem SQL é a DTL, que possui comandos responsáveis por gerenciar diferentes transações ocorridas dentro de um banco de dados (Ex: Commit e Rollback).

Gabarito: Correto

---

11. (CESPE / TJ-AM – 2019) Em SQL, o comando RIGHT OUTER JOIN exibe a união entre duas tabelas, apresentando as linhas da segunda tabela que também existem na primeira tabela, descartando-se as demais situações.

Comentários:

Em SQL, o comando `RIGHT OUTER JOIN` exibe a união entre duas tabelas, apresentando todas as linhas da segunda tabela, além daquelas em que também existe uma correspondência na primeira tabela, descartando-se as demais situações. A descrição da questão se aproxima mais de um `INNER JOIN`.

Gabarito: Errado

---

12. (CESPE / TRE/MT – 2015) Assinale a opção que apresenta a sintaxe correta para se obter, empregando-se a cláusula select em um banco de dados, uma lista com nomes não repetidos dos cargos de uma empresa.

a) select * from cargos
b) select nome from cargos
c) select distinct nome from cargos

d) select codigo, cargo fro cargos where codigo m>1
e) select nome from cargos order by nome asc

Comentários:

Vamos analisar o enunciado: Obter uma lista com nomes não repetidos (**SELECT DINSTINCT NOME**) dos cargos de uma empresa. (**FROM CARGOS**). Logo, ficaria assim:

```
SELECT DINSTINCT NOME FROM CARGOS
```

Gabarito: Letra C

---

13. (CESPE / MEC – 2015) A DML utiliza o comando CREATE para inserir um novo registro na tabela de dados.

Comentários:

A DML utiliza o comando ~~CREATE~~ INSERT para inserir um novo registro na tabela de dados. CREATE é um comando DDL para criar um novo objeto em um banco de dados.

Gabarito: Errado

---

14. (CESPE / STM – 2011) Os comandos do grupo DDL (Data Definition Language) do SQL permitem gerar os dados das tabelas que formam um banco de dados.

Comentários:

Essa questão teve seu gabarito modificado de Errado para Certo sob a seguinte justificativa: *"Os comandos do grupo DDL - Data Definition Language - do SQL permitem gerar tabelas que formam um banco de dados, porém, as estruturas e os conteúdos das tabelas devem ser definidos anteriormente. Dessa forma, opta-se pela alteração do gabarito".*

A banca forçou a barra e interpretou que há comandos DDL (Ex: CREATE TABLE) que criam estruturas que permitem gerar dados posteriormente. Outra interpretação seria pensar que gerar os dados da tabela seria gerar os dados que descrevem a tabela, suas restrições de integridade e tipos de dados, isto é, metadados. De toda forma, questão polêmica!

Gabarito: Correto

---

15. (CESPE / TRE-BA - 2010) A instrução GRANT que altera as permissões em objetos-esquema, é uma instrução de DDL (data definition language).

Comentários:

Nope! A instrução GRANT realmente altera as permissões em objetos-esquema, mas se trata de uma instrução de DCL (Data Control Language).

Gabarito: Errado

---

16. (CESPE / STJ – 2008) O comando CREATE INDEX, usado para criar um parâmetro relacionado com uma tabela para buscar dados mais rapidamente, é considerado como DDL.

Comentários:

Perfeito! O comando CREATE INDEX realmente cria um parâmetro relacionado com uma tabela para buscar dados mais rapidamente e se trata de um comando DDL.

Gabarito: Correto

---

17. (CESPE / STJ – 2008) O comando DELETE, capaz de excluir dados de um banco de dados, é considerado como DDL.

Comentários:

DELETE é um comando DML.

Gabarito: Errado

QUESTÕES COMENTADAS – FCC

18.(FCC / SEFAZ-PE – 2022)

```
CREATE TABLE nfe (
Numero_NFe VARCHAR(9) NOT NULL,
Modelo_NFe VARCHAR(2) NULL,
Serie_NFe VARCHAR(3) NULL,
codigo_UF VARCHAR(2) NULL,
ano_Emissao VARCHAR(2) NULL,
mes_Emissao VARCHAR(2) NULL,
CNPJ_Emitente VARCHAR(14) NULL,
Codigo_Chave VARCHAR(8) NULL,
Digito_Chave VARCHAR(1) NULL,
PRIMARY KEY (Numero_NFe));
```

Para inserir um registro com valores de teste na tabela nfe, utiliza-se a instrução SQL:

a) INSERT INTO nfe VALUES('123456789','55','1','RJ','22','10','15145076000106','87654321','6');
b) APPEND TO nfe VALUES('123456789','55','1','RJ','22','10','15145076000106','87654321','6');
c) INSERT INTO nfe('123456789','55','1','RJ','22','10','15145076000106','87654321','6');
d) INSERT TO nfe VALUES('123456789','55','1','RJ','22','10','15145076000106','87654321','6');
e) ADD INTO nfe('123456789','55','1','RJ','22','10','15145076000106','87654321','6');

Comentários:

A instrução utilizada para inserir registros em uma tabela do banco de dados segue um dos seguintes formatos:

```
INSERT INTO Nome_Tabela (Coluna1, Coluna2, ...)
VALUES (Valor1, Valor2, ...)
```

Ou (quando se está adicionando valores para todas as colunas da tabela):

```
INSERT INTO Nome_Tabela
VALUES (Valor1, Valor2, ...)
```

A única alternativa que segue o modelo apresentado é a letra (a):

```
INSERT INTO nfe
VALUES ('123456789','55','1','RJ','22','10','15145076000106','87654321','6');
```

Gabarito: Letra A

19.(FCC / SEFAZ-PE – 2022)

```
CREATE TABLE nfe (
Numero_NFe VARCHAR(9) NOT NULL,
Modelo_NFe VARCHAR(2) NULL,
Serie_NFe VARCHAR(3) NULL,
codigo_UF VARCHAR(2) NULL,
ano_Emissao VARCHAR(2) NULL,
mes_Emissao VARCHAR(2) NULL,
CNPJ_Emitente VARCHAR(14) NULL,
Codigo_Chave VARCHAR(8) NULL,
Digito_Chave VARCHAR(1) NULL,
PRIMARY KEY (Numero_NFe));
```

Para selecionar na tabela nfe todas as notas fiscais eletrônicas cujo conteúdo do campo codigo_UF inicie pela letra S e tenha qualquer letra na sequência, utiliza-se a instrução SQL:

SELECT * FROM nfe WHERE

a) codigo_UF CONTAINS 'S*';
b) codigo_UF LIKE 'S%';
c) codigo_UF LIKE 'S*';
d) codigo_UF = 'S%';
e) codigo_UF LIKE '%S';

Comentários:

Para selecionar registros em uma tabela por meio de uma pesquisa no conteúdo de um campo do tipo texto, utiliza-se o operador LIKE de acordo com o seguinte formato:

```
SELECT Nome_Coluna
FROM Nome_Tabela
WHERE Nome_Campo LIKE 'Valor'
```

Logo, para selecionar registros da tabela **nfe** cujo conteúdo do campo **codigo_UF** se inicie pela letra S e tenha qualquer letra em sequência, utilizamos:

```
SELECT *
FROM nfe
WHERE código_UF LIKE 'S%'
```

Lembrem-se que o símbolo de porcentagem indica 'qualquer sequência de caracteres', logo essa consulta poderia retornar palavras como, por exemplo, "Salgado", "S", "Sã", "S138".

Gabarito: Letra B

---

20. (FCC / SEFAZ-AP – 2022) Em um banco de dados SQL aberto em condições ideais, considere a existência de uma tabela chamada clientes com os campos clienteID e nomeCliente e de uma tabela chamada pedidos com os campos pedidoID, clienteID e dataPedido.

A coluna clienteID na tabela pedidos se refere ao clienteID na tabela clientes, ou seja, a relação entre elas ocorre por meio da coluna clienteID. Para selecionar pedidoID, nomeCliente e dataPedido apenas de registros que possuam valores correspondentes nas duas tabelas utiliza-se a instrução SQL:

a) SELECT pedidos.pedidoID, clientes.nomeCliente, pedidos.dataPedido FROM pedidos INNER JOIN clientes ON pedidos.clienteID=clientes.clienteID;

b) SELECT pedidos.pedidoID, clientes.nomeCliente, pedidos.dataPedido FROM pedidos AND SELECT clientes WHERE pedidos.clienteID=clientes.clienteID;

c) SELECT pedidos.pedidoID, clientes.nomeCliente, pedidos.dataPedido FROM pedidos UNION clientes ON pedidos.clienteID=clientes.clienteID;

d) SELECT pedidos.pedidoID, clientes.nomeCliente, pedidos.dataPedido FROM pedidos FULL INTER JOIN clientes ON pedidos.clienteID=clientes.clienteID;

e) SELECT pedidos.pedidoID, clientes.nomeCliente, pedidos.dataPedido FROM pedidos INNER JOIN FROM clientes WHERE pedidos.clienteID=clientes.clienteID;

Comentários:

Vamos desenhar as relações:

Deseja-se selecionar pedidoID, nomeCliente e dataPedido apenas de registros que possuam valores correspondentes nas duas tabelas. Vamos analisar item a item:

(a) Correto. Observem que essa consulta retorna as três colunas relacionando as duas tabelas por meio de um INNER JOIN, isto é, retornará apenas os registros que possuem valores correspondentes nas duas tabelas para o atributo clienteID;

(b) Errado. Note que essa consulta não faz nenhuma relação entre as tabelas, logo não há como referenciar clientes.nomeCliente da tabela Pedidos, uma vez que essa coluna pertence à tabela Cliente;

(c) Errado. Sintaxe completamente errada – nada faz sentido nessa consulta com SELECT ... FROM ... UNION ... ON; (d) Errado. Não existe FULL **INTER** JOIN e, sim, FULL OUTER JOIN; (e) Errado. A sintaxe está incorreta – há dois FROM na consulta.

Gabarito: Letra A

---

21. (FCC / SEFAZ-AP – 2022) Usando a linguagem SQL, um fiscal escreveu corretamente, em uma consulta, a expressão WHERE Nome-Contribuinte LIKE 'p%o'. Com isso ele obteve nomes de contribuintes como, por exemplo,

a) Paula ou Patricio.
b) Paulo ou Pedro.
c) Augusto e Jordão.
d) Paulo ou Paula.
e) Álvaro ou Augusto.

Comentários:

Vamos analisar a consulta: **(...) WHERE Nome-Contribuinte LIKE 'p%o'.**

Essa consulta retornará todos os registros da coluna Nome-Contribuinte cujo valor inicie pela letra 'p', em seguida tenha qualquer quantidade de caracteres, e termine com a letra 'o'. Dessa forma, não é possível retornar Paula, Augusto, Jordão ou Álvaro. Logo, restam: Paulo e Pedro.

Gabarito: Letra B

---

22. (FCC / TJ-MA – 2019) Considere a existência de um banco de dados aberto e em condições ideais, no qual a tabela Processo possui diversos campos, sendo um deles, o campo numero_processo, do tipo cadeia de caracteres (varchar). Para exibir todos os processos cujo número inicie por qualquer caractere seguido de "009.51.01.87348-6", utiliza-se a instrução SQL:

a) SELECT *.* FROM Processo WHERE numero_processo LIKE '_009.51.01.87348-6';
b) SELECT * FROM Processo WHERE numero_processo='#009.51.01.87348-6';
c) SELECT * FROM Processo WHERE numero_processo EQUALS '%009.51.01.87348-6';
d) SELECT * FROM Processo WHERE numero_processo LIKE '_009.51.01.87348-6';

e) SELECT *.* FROM Processo WHERE numero_processo LIKE '%009.51.01.87348-6';

Comentários:

Para exibir todos os processos, utiliza-se o asterisco (*):
- `SELECT *`
Para exibir da tabela Processo, temos que:
- `SELECT * FROM Processo`
Para filtrar por numero_processo, temos que:
- `SELECT * FROM Processo WHERE numero_processo`
Para filtrar por aqueles que iniciem por qualquer caractere seguido de "009.51.01.87348-6":
- `SELECT * FROM Processo WHERE numero_processo LIKE '_009.51.01.87348-6'`

Professor, por que não se utiliza o '%'? Porque a questão deseja filtrar por processos cujo número inicie por qualquer caractere (_) e, não, quaisquer caracteres (%).

Gabarito: Letra D

---

23. (FCC / TRT4 – 2019) Uma Analista digitou o comando TRUNCATE TABLE processos; em um banco de dados SQL aberto em condições ideais para:

a) excluir os dados da tabela, mas não a tabela em si.
b) excluir a estrutura da tabela e os dados nela contidos.
c) juntar a tabela aberta na memória com a tabela processos.
d) bloquear a tabela processos para uso exclusivo de seu usuário.
e) editar a estrutura da tabela em modo gráfico.

Comentários:

Esse comando é utilizado para excluir todos s dados de uma tabela, mas não a tabela sem si.

Gabarito: Letra A

---

24. (FCC / TRT4 – 2019) Em uma tabela chamada itemfatura há diversos registros em que constam o mesmo valor no campo idfatura. Para mostrar a quantidade de valores de idfatura diferentes que estão cadastrados na tabela, utiliza-se o comando:

a) SELECT DISTINCT (idfatura) FROM itemfatura;
b) SELECT * FROM itemfatura WHERE idfatura IS DIFFERENT;
c) SELECT SUM(DISTINCT idfatura) FROM itemfatura;
d) SELECT COUNT(DISTINCT idfatura) FROM itemfatura;
e) SELECT COUNT(DIFFERENT idfatura) FROM itemfatura;

Comentários:

Note que temos vários registros diferentes com o mesmo valor de idfatura. Para mostrar a quantidade (**count**()) de valores de idfatura (count(**id_fatura**)) diferentes (count(**distinct** idfatura)) que estão cadastrados na tabela, pode-se utilizar o comando:

```
SELECT COUNT(DISTINCT idfatura) FROM itemfatura;
```

Gabarito: Letra D

---

25. (FCC / TRT4 – 2019) Um Técnico Judiciário necessitou usar a linguagem padrão SQL para recuperar, de uma tabela do banco de dados relacional denominada tabela1,

I. o menor valor em uma determinada coluna denominada coluna1.
II. um padrão de valores denominado padrão_desejado em uma outra coluna denominada coluna2.

Para tanto, em duas operações distintas, ele utilizou, respectivamente, as expressões

1. SELECT I .....
   FROM tabela1
   WHERE condição;
2. SELECT coluna2
   FROM tabela1
   WHERE II .....;

I e II são, correta e respectivamente,

a) MINVALUE(coluna1) e padrão_desejado %LIKE coluna2
b) THIN (coluna1) e coluna2 = padrão_desejado
c) SMALL(coluna1) e padrão_desejado = coluna2
d) MIN(coluna1) e coluna2 LIKE padrão_desejado
e) GETSMLL(coluna1) e padrão_desejado % coluna2

Comentários:

- O menor valor de uma determinada coluna denominada coluna1 é MIN(coluna1);
- Um padrão de valores denominado padrão_desejado é LIKE padrão_desejado.

Gabarito: Letra D

---

26. (FCC / SEFAZ-BA – 2019) Em uma tabela chamada Contribuinte de um banco de dados padrão SQL aberto e em condições ideais 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;

Comentários:

Note que se deseja alterar dados e, não, a definição de tabelas, logo utiliza-se o comando UPDATE e, não, ALTER TABLE (eliminamos as letras (a) e (d)). A sintaxe do comando UPDATE é UPDATE NomeTabela e o nome da tabela é Contribuinte (eliminamos a letra (c) e (e)). Logo, a resposta correta é:

```
UPDATE Contribuinte SET nomeContribuinte='Marcos Silva' WHERE idContribuinte =
1
```

Gabarito: Letra B

---

27. (FCC / SEFAZ-BA – 2019) Para buscar na tabela Contribuintes todos os nomes de contribuintes (campo nomeContribuinte) que terminam com a letra s, um Auditor utilizou corretamente a instrução SQL

a) SEARCH * FROM Contribuintes WHERE nomeContribuinte LIKE '%s';
b) SELECT nomeContribuinte FROM Contribuintes WHERE nomeContribuinte LIKE '*s';
c) SELECT * FROM Contribuintes WHERE nomeContribuinte FINISHED BY '%s';
d) SEARCH nomeContribuinte FROM Contribuintes WHERE nomeContribuinte FINISHED BY 's';
e) SELECT * FROM Contribuintes WHERE nomeContribuinte LIKE '%s';

Comentários:

O comando que recupera valores é o SELECT, logo já podemos eliminar as letras (a) e (d); como se deseja buscar todos os nomes de contribuintes, utiliza-se asterisco (*), logo já podemos eliminar a letra (b); por fim, o operador utilizado para comparar substituir um ou mais caracteres em uma string é o LINK e, não, FINISHED. Logo, a resposta correta é:

```
SELECT * FROM Contribuintes WHERE nomeContribuinte LIKE '%s'
```

Gabarito: Letra E

28.(FCC / AFAP – 2019) Fernando está usando a linguagem SQL (ANSI) e pretende fazer uma atualização nos dados Nome_Cli e End_Cli do cliente cujo Cod_Cli é Clio1, na tabela Cliente. Nome_Cli passará a ser Ariana e End_Cli passará a ser Rua ABC. O código SQL correto que Fernando escreveu foi:

..I.. Cliente
..II.. Nome_Cli = 'Ariana', End _Cli = 'Rua ABC'
..III.. Cod_Cli = 'Clio1';

Para que o código esteja correto, as lacunas I, II e III devem ser preenchidas, respectivamente, por

a) SET - WHERE - UPDATE
b) UPDATE - SET - WHERE
c) UPDATE - WHERE - SET
d) WHERE - SET - UPDATE
e) SET - UPDATE - WHERE

Comentários:

Nesse caso, teremos que fazer:

```
UPDATE Cliente
SET Nome_Cli = ‘Ariana’, End_Cli = ‘Rua ABC’
WHERE Cod_Cli = ‘Clio1’;
```

Gabarito: Letra B

29.(FCC / DPE-AM – 2018) Para apagar todos os registros da tabela copia_eleitores utiliza-se a instrução SQL:

a) DELETE FROM copia_eleitores; ou TRUNCATE * FROM copia_eleitores;
b) DELETE RECORDS copia_eleitores; ou DROP RECORDS FROM copia_eleitores;
c) DELETE * FROM copia_eleitores; ou DELETE RECORDS copia_eleitores;
d) DELETE FROM copia_eleitores; ou DELETE * FROM copia_eleitores;
e) DELETE RECORDS copia_eleitores; ou TRUNCATE TABLE copia_eleitores;

Comentários:

(a) Errado, deveria ser TRUNCATE TABLE copia_eleitores; (b) Errado, não existe a cláusula RECORDS; (c) Errado, Errado, não existe a cláusula RECORDS; (d) Correto, pero no mucho. Essa questão estaria correta, mas ela coloca um asterisco após o DELETE – que não é suportado por esse

comando – no entanto a banca não anulou a questão e considerou esse item como correto; (e) Errado, não existe a cláusula RECORDS.

Gabarito: Letra D

---

30. (FCC / TST – 2017) Um Programador:

I. criou uma tabela e uma view em um banco de dados relacional.
II. alterou a estrutura da tabela.
III. incluiu registros na tabela.

a) DDL – DML – DDL.
b) DML – DML – DDL.
c) DML – DDL – DDL.
d) DDL – DML – DML.
e) DDL – DDL – DML.

Comentários:

(I) Criar estruturas (Ex: CREATE TABLE ou CREATE VIEW) é um recurso da DDL; (II) Alterar estruturas de tabelas (Ex: ALTER TABLE) é um recurso da DDL; (III) Incluir registros (Ex: INSERT INTO) em uma tabela é um recurso da DML.

Gabarito: Letra E

---

31. (FCC / MANAUSPREV – 2015) A linguagem SQL é dividida em subconjuntos de acordo com as operações que se deseja efetuar sobre um banco de dados. Considere os grupos de comandos:

I. CREATE, ALTER, DROP.
II. GRANT, REVOKE.
III. DELETE, UPDATE, INSERT.

Os comandos listados em:

a) I correspondem à Data Control Language - DCL e II à Data Definition Language - DDL.
b) I correspondem à Data Manipulation Language - DML e III à Data Control Language - DCL.
c) II correspondem à Data Manipulation Language - DML e III à Data Control Language - DCL.
d) I correspondem à Data Definition Language - DDL e III à Data Manipulation Language - DML.
e) II correspondem à Data Control Language - DCL e III à Data Definition Language - DDL.

Comentários:

(I) CREATE, ALTER e DROP são comandos DDL; (II) GRANT e REVOKE são comandos DCL; (III) DELETE, UPDATE e INSERT são comandos DML.

Gabarito: Letra D

---

32. (FCC / TRT-19 - 2011) Considere as linguagens inseridas no contexto SQL: DML, DDL, DTL, DCL e DQL. Desta forma, Grant, Commit, Update, Delete e Alter, correspondem, respectivamente, a:

a) DCL, DCL, DDL, DDL e DDL.
b) DTL, DCL, DML, DDL e DQL.
c) DCL, DTL, DML, DML e DDL.
d) DML, DTL, DCL, DQL e DML.
e) DQL, DDL, DML, DML e DDL.

Comentários:

GRANT é DCL; COMMIT é DTL; UPDATE é DML; DELETE é DML; e ALTER é DDL.

Gabarito: Letra C

---

33. (FCC / TCM-PA – 2010) Na linguagem SQL são, respectivamente, expressões DDL e DML:

a) UPDATE e DROP INDEX.
b) UPDATE e INSERT INTO.
c) ALTER DATABASE e UPDATE.
d) ALTER TABLE e CREATE INDEX.
e) INSERT INTO e DELETE.

Comentários:

(a) UPDATE é **DML** e DROP INDEX é **DDL**; (b) UPDATE é **DML** e INSERT INTO é **DML**; (c) ALTER DATABASE é **DDL** e UPDATE é **DML**; (d) ALTER TABLE é **DDL** e CREATE INDEX é **DDL**; (e) INSERT INTO é **DML** e DELETE é **DML**;

Gabarito: Letra C

---

34. (FCC / METRÔ-SP – 2010) O SQL (Structured Query Language) é uma linguagem de pesquisa declarativa para banco de dados relacional. A DDL permite ao usuário definir tabelas novas e elementos associados. A sigla DDL significa:

a) Data Definition List.
b) Data Default Language.
c) Data Definition Language.
d) Data Default List.

e) Definition Data Language.

Comentários:

A sigla **DDL** significa **Data Definition Language**.

Gabarito: Letra C

---

35. (FCC / TRE-RS – 2010) São declarações SQL, associadas, respectivamente, a três declarações DML (Data Manipulation Language) e a três declarações DDL (Data Definition Language):

a) UPDATE, INSERT, SELECT, DROP INDEX, ALTER TABLE, CREATE VIEW.
b) DROP VIEW, DROP INDEX, UPDATE, DELETE, ALTER TABLE, INSERT.
c) CREATE, UPDATE, DELETE, DROP VIEW, INSERT, SELECT.
d) CREATE INDEX, CREATE VIEW, DROP VIEW, UPDATE, DELETE, INSERT.
e) INSERT, DELETE, DROP INDEX, ALTER INDEX, UPDATE, SELECT.

Comentários:

(a) Correto. UPDATE **(DML)**, INSERT **(DML)**, SELECT **(DML)**, DROP INDEX **(DDL)**, ALTER TABLE **(DDL)**, CREATE VIEW **(DDL)**.

(b) Errado. DROP VIEW **(DDL)**, DROP INDEX **(DDL)**, UPDATE **(DML)**, DELETE **(DML)**, ALTER TABLE **(DDL)**, INSERT **(DML)**.

(c) Errado. CREATE, UPDATE **(DML)**, DELETE **(DML)**, DROP VIEW **(DDL)**, INSERT **(DML)**, SELECT **(DML)**.

(d) Errado. CREATE INDEX **(DDL)**, CREATE VIEW **(DDL)**, DROP VIEW **(DDL)**, UPDATE **(DML)**, DELETE **(DML)**, INSERT **(DML)**.

(e) Errado. INSERT **(DML)**, DELETE **(DML)**, DROP INDEX **(DDL)**, ALTER INDEX **(DDL)**, UPDATE **(DML)**, SELECT **(DML)**.

Gabarito: Letra A

---

36. (FCC / TRF-4ª Região – 2010) DROP é um comando utilizado para apagar um objeto do banco de dados e é parte integrante do subconjunto da linguagem SQL denominado:

a) DML - Linguagem de Manipulação de Dados.
b) DTL - Linguagem de Transação de Dados.
c) DCL - Linguagem de Controle de Dados.
d) DDL - Linguagem de Definição de Dados.

e) DQL - Linguagem de Consulta de Dados.

Comentários:

DROP é um comando DDL – Linguagem de Definição de Dados.

Gabarito: Letra D

---

37. **(FCC / MPE-SE – 2010)** Em relação às linguagens de definição e de manipulação de dados no SQL, é correto afirmar:

a) no grupo DDL, apagar tabelas e índices da base de dados é função do comando DROP.

b) no grupo DML, conceder acesso à base de dados e aos seus objetos é função do comando ALTER.

c) no grupo DDL, o comando SELECT é utilizado para extrair e alterar dados da base de dados.

d) o grupo DDL contém os comandos para criar e alterar novas tuplas no banco de dados.

e) o comando ALTER, do grupo DML, tem como função alterar linhas já existentes no banco de dados.

Comentários:

(a) Correto. São os comandos DROP TABLE e DROP INDEX; (b) Errado. Esses são comandos do grupo DCL; (c) Errado. Esse é um comando do grupo DML; (d) Errado. Esses são comandos do grupo DML; (e) Errado. Esse é um comando do grupo DDL e tem a função de alterar a estrutura do banco.

Gabarito: Letra A

---

38. **(FCC / DPE-SP – 2010)** O SQL (Structured Query Language) é uma linguagem de pesquisa declarativa para banco de dados relacional. A DML é um subconjunto da linguagem usada para inserir, atualizar e apagar dados. A sigla DML significa:

a) Data Main Language.
b) Data Manager Language.
c) Data Manipulation List.
d) Data Manager List.
e) Data Manipulation Language.

Comentários:

A sigla significa Data Manipulation Language.

Gabarito: Letra E

---

39. (FCC / TRE-AM – 2010) Em SQL, a deleção de linhas em uma tabela é feita por meio da expressão geral:

a) WHERE nome_coluna FROM nome_tabela = valor_qualquer DELETE nome_coluna.
b) WHERE nome_coluna = valor_qualquer DELETE nome_coluna FROM nome_tabela.
c) DELETE nome_coluna FROM nome_tabela WHERE nome_coluna = valor_qualquer.
d) DELETE WHERE nome_tabela.nome_coluna = valor_qualquer
e) DELETE FROM nome_tabela WHERE nome_coluna = valor_qualquer.

Comentários:

A sintaxe correta é: DELETE FROM nome_tabela WHERE nome_coluna = valor_qualquer.

Gabarito: Letra E

---

40. (FCC / TRE-RN – 2010) Na SQL, é o comando principal da Linguagem de Consulta de Dados:

a) REVOKE.
b) DROP.
c) WHERE.
d) SELECT.
e) HAVING.

Comentários:

O comando principal da SQL (Linguagem de Consulta de Dados) é o SELECT.

Gabarito: Letra D

---

41. (FCC / TCE-GO – 2009) Considere:

Select (X) from (Y) order by (Z)

Na SQL, X, Y e Z são, respectivamente,

a) nome de tabela, nome de coluna e nome de coluna.
b) nome de coluna, nome de tabela e nome de coluna.

c) condição, nome de tabela e nome de coluna.
d) nome de tabela, condição e nome de coluna.
e) nome de coluna, nome de tabela e condição.

Comentários:

X é o nome de coluna, Y é o nome de tabela e Z é o nome de coluna.

Gabarito: Letra B

---

42. (FCC / TRE-PI – 2009) Na linguagem SQL, considere os comandos relativos à Linguagem de Definição de Dados ? DDL e à Linguagem de Manipulação de Dados ? DML:

a. ALTER
b. CREATE
c. DELETE
d. DROP
e. INSERT
f. SELECT
g. SET

A associação entre os comandos e suas respectivas linguagens de definição e manipulação de dados está correta em:

a) DDL - abcd // DML - efg.
b) DDL - bfg // DML - acde.
c) DDL - abf // DML - cdeg.
d) DDL - abd // DML - cefg.
e) DDL - acg // DML - bdef.

Comentários:

DDL é ALTER, CREATE, DROP (abdg); e DML é DELETE, INSERT, SELECT e SET (cefg)

Gabarito: Letra D

---

43. (FCC / TRE-SE – 2007) Em SQL-ANSI, Count:

a) é um comando de intersecção no contexto da DML.
b) é uma função de agregação no contexto da DML.
c) é um operador de conjunto no contexto da DDL.
d) é uma função de restrição no contexto da DML.
e) é uma expressão de seleção no contexto da DDL.

Comentários:

COUNT é uma função de agregação no contexto da DML capaz de retornar a quantidade de linhas de uma determinada query.

Gabarito: Letra B

QUESTÕES COMENTADAS – FGV

```
create table X(A int not null primary key,
               B int)
create table Y(A int not null UNIQUE,
               constraint fk
               foreign key (A) references X(A)
               on delete cascade)
```

Para todos os efeitos, suponha que o número de linhas em cada tabela é diferente de zero.

44. (FGV / TRT-MA – 2022) Assinale a afirmativa correta a respeito do esquema relacional apresentado.

a) a tabela X admite linhas duplicadas.
b) a tabela X não pode ter mais linhas que a tabela Y
c) a tabela Y admite linhas duplicadas.
d) a tabela Y não pode ter mais linhas que a tabela X.
e) as tabelas X e Y não podem ter o mesmo número de linhas.

Comentários:

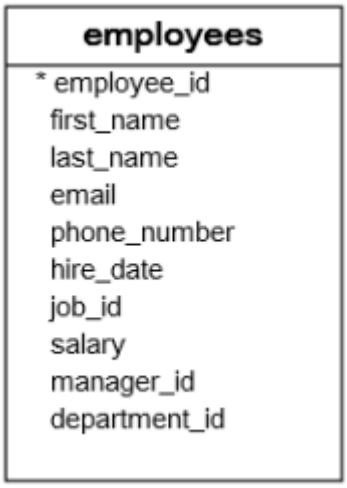
Observem que a Tabela X tem dois atributos: A, que é chave primária; e B, que é um inteiro. Já a Tabela Y tem somente um atributo: A, que é também chave primária (dado que é único e não nulo) e esse atributo é uma chave estrangeira que referencia o atributo A da Tabela X.

Se o atributo A da Tabela Y referencia o atributo A da Tabela X, então a Tabela Y jamais poderá ter mais linhas que a Tabela X. Já a Tabela X pode ter mais (ou iguais) linhas que a Tabela Y sem nenhum problema. E nenhuma das tabelas admite linhas duplicadas porque possuem atributos únicos e não nulos.

Gabarito: Letra D

---

45. (FGV / SEFAZ-BA – 2022) Considere a seguinte tabela em um banco de dados relacional.



Assinale a opção que indica o comando SQL utilizado para localizar todos os nomes completos dos *employees*, cujos primeiros nomes começam com as letras Ma.

a) SELECT
first_name;
last_name;
FROM
employees;

b) SELECT *
FROM
employees
WHERE
first_name = 'Ma';

c) SELECT *
FROM
employees
WHERE
first_name = 'Ma*';

d) SELECT
employee_id,
first_name,
last_name
FROM
employees
WHERE
first_name LIKE 'Ma%';

e) SELECT

```
        employee_id,
        first_name,
        last_name
FROM
        employees
WHERE
        first_name IN 'Ma_';
```

Comentários:

(a) Errado. Esse comando está retornando o nome completo (primeiro e último nome) de todos os empregados;

(b) Errado. Esse comando está retornando todas as informações (e não apenas o nome completo) de todos os funcionários cujo primeiro nome é "Ma"

c) Errado. Esse comando está retornando todas as informações (e não apenas o nome completo) de todos os funcionários cujo primeiro nome é "Ma";

(d) Correto. Traz o primeiro e último nome (além do ID) de todos os funcionários cujo primeiro nome começa com "Ma". O operador correto é o LIKE e o % indica uma cadeia de caracteres.

(e) Errado. IN é um operador que permite especificar múltiplos valores dentro de uma cláusula. O operador correto seria o LIKE e mesmo assim o underline (_) indica um único caractere e a questão pede qualquer funcionário cujo primeiro nome comece com "Ma" mais qualquer cadeia de caracteres, logo deveria ser LIKE 'Ma%'

Gabarito: Letra D

Tabela T

sequencia	caracteristica
1	23987
2	9845
3	NULL
4	40983
6	48750
7	NULL
8	NULL
10	48750
12	48750

Tabela TX

sequencia	caracteristica
2	9845
3	998034
4	50932
5	24390
6	48750
6	50296
7	NULL
8	998746
9	32746
9	NULL
9	22798

Tabela DUAL

x
NULL

46.(FGV/ TCU – 2022) Considere que é preciso atualizar os dados da tabela T a partir dos dados da tabela TX, ambas definidas anteriormente. A consolidação é feita por meio da alteração na tabela T a partir de registros de TX. O comando SQL utilizado nessa atualização é exibido a seguir.

```
update T
set caracteristica =
   select max(caracteristica) x from TX tx
   where tx.sequencia = t.sequencia
         and not (tx.caracteristica is null))
where
( exists
 (select * from TX tx
 where tx.sequencia = t.sequencia
   and not (tx.caracteristica is null))
 and
 ( t.caracteristica is null
   or
   t.caracteristica <
         (select max(caracteristica) x from TX tx
          where tx.sequencia = t.sequencia
               and not (tx.caracteristica is null))
 )
)
```

O número de registros da tabela T afetados pela execução do comando SQL acima é:

a) zero;

b) três;
c) quatro;
d) seis;
e) nove.

Comentários:

Uma dica que eu sempre dou ao ver queries tão grandes é abstrair a sua estrutura. *Como assim, Diego?* Vamos retirar alguns detalhes e imaginá-la em partes macro:

```
update T
set caracteristica =
        [SELECT])
where
( exists
 (CONDIÇÃO 1)
 and
 ( CONDIÇÃO 2
   or
   CONDIÇÃO 3)
 )
)
```

Vocês viram que eu retirei vários detalhes? Pois é, agora nós conseguimos visualizar melhor algumas coisas interessantes. O primeiro de tudo é que ocorre um `EXISTS`. Esse operador permite testar a existência de qualquer registro em uma subconsulta. Ele retorna `TRUE` se a subconsulta retornar um ou mais registros; caso contrário, retorna `FALSE`. No caso, se ele retornar `TRUE`, uma determinada linha da tabela T será atualizada com um determinado valor; caso contrário, não.

Além disso, podemos notar que se trata de uma consulta aninhada, dado que temos uma consulta dentro de outra. A consulta interna é uma subconsulta correlacionada, dado que temos uma referência a uma coluna da consulta externa. Dito isso, vamos ter que analisar – para cada tupla da consulta externa – se a consulta interna retorna algum valor. Se sim, retornaremos também para a consulta externa; caso contrário, não. Por fim, note que temos três condições:

(CONDIÇÃO 1 AND (CONDIÇÃO 2 OR CONDIÇÃO 3))

Lembrem-se que o operador `AND` retorna verdadeiro somente se ambas as condições forem verdadeiras; já o operador `OR` retorna verdadeira caso qualquer das condições forem verdadeiras. Pronto! Agora estamos prontos para analisar a questão. Como temos uma subconsulta correlacionada com um `EXISTS`, nós precisamos passar por cada uma das tuplas da consulta externa e fazendo verificações na consulta interna. A primeira tupla da consulta externa é:

TX

sequencia	caracteristica
2	9845
3	998034
4	50932
5	24390
6	48750
6	50296
7	NULL
8	998746
9	32746
9	NULL
9	22798

T

sequencia	caracteristica
1	23987
2	9845
3	NULL
4	40983
6	48750
7	NULL
8	NULL
10	48750
12	48750

Agora vamos rodar toda a query baseado nessa primeira tupla. Note que a questão só deseja saber a quantidade de registros que serão atualizados e, não, qual valor específico de atualização. Logo, podemos ignorar solenemente toda essa parte do `select` apresentado a seguir:

```
select max(caracteristica) x from TX tx
where tx.sequencia = t.sequencia
and not (tx.caracteristica is null)
```

Logo, vamos entrar agora efetivamente no `where` apresentado no código abaixo. Lembrem-se que, para cada tupla da Tabela T, vamos ver se a consulta abaixo retorna ao menos um valor `TRUE`.

```
where
( exists
 (select * from TX tx
 where tx.sequencia = t.sequencia
      and not (tx.caracteristica is null))
 and
 ( t.caracteristica is null
      or
      t.caracteristica <
            (select max(caracteristica) x from TX tx
             where tx.sequencia = t.sequencia
               and not (tx.caracteristica is null))
 )
)
```

Vejam o início da subconsulta! Ela nos diz para retornar todas ocorrências da Tabela TX em que `tx.sequencia` seja igual a 1. *Elas existem?* Não, logo nem precisamos prosseguir porque quando temos um valor falso no operador `AND`, ele sempre retornará `FALSE`.

TX

sequencia	caracteristica
2	9845
3	998034
4	50932
5	24390
6	48750
6	50296
7	NULL
8	998746
9	32746
9	NULL
9	22798

T

sequencia	caracteristica
1	23987
2	9845
3	NULL
4	40983
6	48750
7	NULL
8	NULL
10	48750
12	48750

Agora é a vez da segunda tupla da Tabela T. Vamos procurar na Tabela TX todas as ocorrências em que o valor de `tx.sequencia` seja igual a 2. *Elas existem?* Sim, logo vamos prosseguir! Nas ocorrências encontradas, o valor de `tx.caracteristica` não é nulo? Verdadeiro! Opa, então nossa primeira condição já foi satisfeita. Nas ocorrências encontradas, o valor de `t.caracteristica` é nulo? Falso! Ainda não acabou: o valor de `t.caracteristica` é menor que o maior valor de `tx.característica`? Ora, só há um valor de `tx.caracteristica` encontrado e ele tem exatamente o mesmo valor que `t.caracteristica`, logo não é menor, então retorna falso! Dessa forma, nossas condições ficaram:

[VERDADEIRO AND (FALSO OU FALSO)] = [VERDADEIRO AND FALSO] = FALSO

Podemos concluir – portanto – que essa tupla não será atualizada na Tabela T dado que o operador `exists` retornou `FALSE`!

TX

sequencia	caracteristica
2	9845
3	998034
4	50932
5	24390
6	48750
6	50296
7	NULL
8	998746
9	32746
9	NULL
9	22798

T

sequencia	caracteristica
1	23987
2	9845
3	NULL
4	40983
6	48750
7	NULL
8	NULL
10	48750
12	48750

Agora é a vez da terceira tupla da Tabela T. Vamos procurar na Tabela TX todas as ocorrências em que o valor de `tx.sequencia` seja igual a 3. *Elas existem?* Sim, logo vamos prosseguir! Nas ocorrências encontradas, o valor de `tx.caracteristica` não é nulo? Verdadeiro! Opa, então nossa

primeira condição já foi satisfeita. Nas ocorrências encontradas, o valor de `t.caracteristica` é nulo? Verdadeiro! Logo, nem precisamos continuar porque sabemos que o operador OU sempre retornará verdadeiro quando alguma das condições for verdadeira. Dessa forma, nossas condições ficaram:

[VERDADEIRO AND (VERDADEIRO OU ?)] = [VERDADEIRO AND VERDADEIRO] = VERDADEIRO

Podemos concluir – portanto – que essa tupla será atualizada na Tabela T dado que o operador `exists` retornou `TRUE`!

TX

sequencia	caracteristica
2	9845
3	998034
4	50932
5	24390
6	48750
6	50296
7	NULL
8	998746
9	32746
9	NULL
9	22798

T

sequencia	caracteristica
1	23987
2	9845
3	NULL
4	40983
6	48750
7	NULL
8	NULL
10	48750
12	48750

Agora é a vez da quarta tupla da Tabela T. Vamos procurar na Tabela TX todas as ocorrências em que o valor de `tx.sequencia` seja igual a 4. *Elas existem?* Sim, logo vamos prosseguir! Nas ocorrências encontradas, o valor de `tx.caracteristica` não é nulo? Verdadeiro! Opa, então nossa primeira condição já foi satisfeita. Nas ocorrências encontradas, o valor de `t.caracteristica` é nulo? Falso! Ainda não acabou: o valor de `t.caracteristica` (40983) é menor que o maior valor de `tx.característica` (50932)? Verdadeiro! Dessa forma, nossas condições ficaram:

[VERDADEIRO AND (FALSO OU VERDADEIRO)] = [VERDADEIRO AND VERDADEIRO] = VERDADEIRO

Podemos concluir – portanto – que essa tupla será atualizada na Tabela T dado que o operador `exists` retornou `TRUE`!

TX

sequencia	caracteristica
2	9845
3	998034
4	50932
5	24390
6	48750
6	50296
7	NULL
8	998746
9	32746
9	NULL
9	22798

T

sequencia	caracteristica
1	23987
2	9845
3	NULL
4	40983
6	48750
7	NULL
8	NULL
10	48750
12	48750

Agora é a vez da quinta tupla da Tabela T. Vamos procurar na Tabela TX todas as ocorrências em que o valor de `tx.sequencia` seja igual a 6. *Elas existem?* Sim, logo vamos prosseguir! Nas ocorrências encontradas, o valor de `tx.caracteristica` não é nulo? Verdadeiro! Opa, então nossa primeira condição já foi satisfeita. Nas ocorrências encontradas, o valor de `t.caracteristica` é nulo? Falso! Ainda não acabou: o valor de `t.caracteristica` (48750) é menor que o maior valor de `tx.característica` (50296)? Verdadeiro! Dessa forma, nossas condições ficaram:

[VERDADEIRO AND (FALSO OU VERDADEIRO)] = [VERDADEIRO AND VERDADEIRO] = VERDADEIRO

Podemos concluir – portanto – que essa tupla será atualizada na Tabela T dado que o operador `exists` retornou `TRUE`!

TX

sequencia	caracteristica
2	9845
3	998034
4	50932
5	24390
6	48750
6	50296
7	NULL
8	998746
9	32746
9	NULL
9	22798

T

sequencia	caracteristica
1	23987
2	9845
3	NULL
4	40983
6	48750
7	NULL
8	NULL
10	48750
12	48750

Agora é a vez da sexta tupla da Tabela T. Vamos procurar na Tabela TX todas as ocorrências em que o valor de `tx.sequencia` seja igual a 7. *Elas existem?* Sim, logo vamos prosseguir! Nas ocorrências encontradas, o valor de `tx.caracteristica` não é nulo? Falso! Logo, já podemos encerrar por aqui! As nossas condições ficaram:

[FALSO AND (? OU ?)] = [FALSO AND ?] = FALSO

Podemos concluir – portanto – que essa tupla será atualizada na Tabela T dado que o operador `exists` retornou `TRUE`!

TX

sequencia	caracteristica
2	9845
3	998034
4	50932
5	24390
6	48750
6	50296
7	NULL
8	998746
9	32746
9	NULL
9	22798

T

sequencia	caracteristica
1	23987
2	9845
3	NULL
4	40983
6	48750
7	NULL
8	NULL
10	48750
12	48750

Agora é a vez da sétima tupla da Tabela T. Vamos procurar na Tabela TX todas as ocorrências em que o valor de `tx.sequencia` seja igual a 8. *Elas existem?* Sim, logo vamos prosseguir! Nas ocorrências encontradas, o valor de `tx.caracteristica` não é nulo? Verdadeiro! Opa, então nossa primeira condição já foi satisfeita. Nas ocorrências encontradas, o valor de `t.caracteristica` é nulo? Verdadeiro! Logo, nem precisamos continuar porque sabemos que o operador OU sempre retornará verdadeiro quando alguma das condições for verdadeira. Dessa forma, nossas condições ficaram:

[VERDADEIRO AND (VERDADEIRO OU ?)] = [VERDADEIRO AND VERDADEIRO] = VERDADEIRO

Podemos concluir – portanto – que essa tupla será atualizada na Tabela T dado que o operador `exists` retornou `TRUE`!

Nem precisamos continuar porque os valores de sequência 10 e 12 não existem na Tabela TX. Logo, tivemos 4 atualizações na Tabela T.

Gabarito: Letra C

---

47. (FGV/ TCU – 2022) Analise os cinco comandos SQL exibidos abaixo, utilizando a tabela DUAL apresentada anteriormente.

(1) select * from dual where x = null
(2) select * from dual where x <> null
(3) select * from dual where x > 10

(4) select * from dual where not x > 10
(5) select * from dual where x > 10
union
select * from dual where x <= 10

Se os resultados desses comandos fossem separados em grupos homogêneos, de modo que em cada grupo todos sejam idênticos e distintos dos elementos dos demais grupos, haveria:

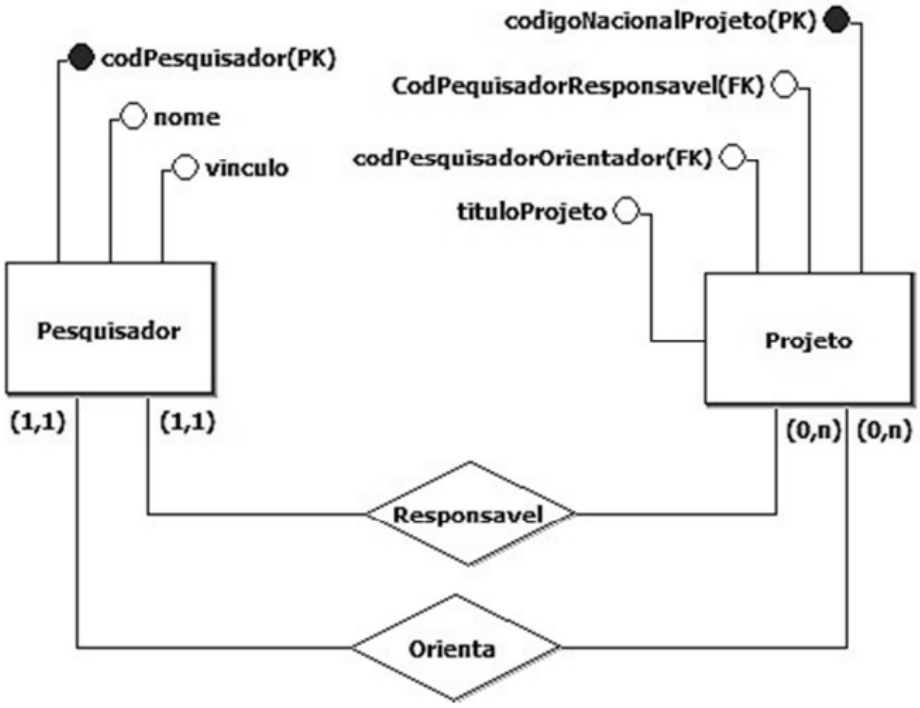
a) apenas um grupo;
b) apenas dois grupos;
c) apenas três grupos;
d) apenas quatro grupos;
e) cinco grupos.

Comentários:

Nós sabemos que – se quisermos verificar se um valor é nulo ou não – devemos utilizar os operadores IS NULL ou IS NOT NULL. Os dois primeiros comandos utilizaram um operador relacional – isso não ocasiona nenhum erro, mas retornará vazio. O terceiro comando não retornará nada (retornará vazio), dado que x não é maior que 10. O quarto comando é a grande pegadinha da questão: se o terceiro comando retornou vazio, a negação de vazio também é vazio. Parece contraintuitivo: seria de se pensar que retornaria todo o restante, mas isso não é verdade no contexto de SQL. Por fim, o último comando retornará a união de vazio com vazio = vazio.

O enunciado pergunta quantos grupos existiriam se os resultados desses comandos fossem separados em grupos homogêneos, de modo que em cada grupo todos sejam idênticos e distintos dos elementos dos demais grupos. Todos os comandos retornaram vazio, logo apenas um grupo.

Gabarito: Letra A



Com base no modelo entidade-relacionamento (MER) precedente, que apresenta a representação das regras de uma instituição de pesquisa, existe um Pesquisador cadastrado com o nome Pedro. Todos os atributos do MER são do tipo caractere e um dos comandos SQL usados para a construção do modelo é mostrado a seguir.

```
create table Projeto
codNacionalProjeto char(2),
codPesquisadorResponsavel char(2),
codPesquisadorOrientador char(2),
tituloProjeto char(50),
primary key(codNacionalProjeto));
```

A partir das informações constantes no modelo e dos dados sobre o conteúdo dos atributos, julgue os itens subsecutivos.

48.(FGV / FUNSÚDE-CE – 2021) Atenção: na próxima questão, considere a definição e as instâncias das tabelas de bancos de dados CLUBE e JOGO exibidas a seguir.

CLUBE
nome
Barcelona
Boca Juniors
The Strongest

JOGO

mandante	**visitante**	**golsM**	**golsV**
Barcelona	Boca Juniors	1	0

Barcelona	The Strongest	NULL	NULL
Boca Juniors	Barcelona	0	0
Boca Juniors	The Strongest	3	0
The Strongest	Barcelona	2	0
The Strongest	Boca Juniors	2	0

Cada clube deve jogar quatro vezes, duas como mandante e duas como visitante. As colunas golsM e golsV registram o número de gols dos times mandantes e visitantes, respectivamente, em cada jogo. Ambas são nulas enquanto o jogo não for realizado.

Analise o comando SQL a seguir, à luz das definições e instâncias das tabelas CLUBE e JOGO, apresentadas anteriormente.

```
select distinct mandante, visitante from JOGO, CLUBE
```

Assinale o número de linhas, sem incluir os títulos, produzidas pela execução desse comando:

a) 4.
b) 6.
c) 10.
d) 24.
e) 48.

Comentários:

Vamos analisar a consulta apresentada na questão: (1) ela possui um **distinct**, logo registros repetidos (duplicatas) serão eliminados; (2) ela possui um produto cartesiano, dado que há uma vírgula separando as tabelas **JOGO** e **CLUBE**. Dito isso, sabemos que a primeira tabela possui três linhas e a segunda possui seis linhas, logo teríamos 3x6 = 18 linhas. No entanto, há uma pegadinha na questão: ela deseja retornar apenas as colunas mandante e visitante, e essas colunas pertencem somente à tabela **JOGO**. Como há um distinct na consulta, é irrelevante o produto cartesiano, visto que as repetições serão ignoradas. Logo, ela retornará apenas seis linhas:

mandante	**visitante**
Barcelona	Boca Juniors
Barcelona	The Strongest
Boca Juniors	Barcelona
Boca Juniors	The Strongest
The Strongest	Barcelona
The Strongest	Boca Juniors

Gabarito: Letra B

49.(FGV / FUNSÚDE-CE – 2021) Analise o comando SQL a seguir, à luz das definições e instâncias das tabelas CLUBE e JOGO, definidas anteriormente.

```
select c.nome from CLUBE c where  (
   select count(*) from JOGO j  where c.nome = j.mandante) <> 2  or  (
        select count(*) from JOGO j  where c.nome = j.visitante) <> 2
```

O resultado produzido pela execução desse comando é a lista de todos os clubes que:

a) aparecem em quatro jogos.
b) não aparecem em quatro jogos.
c) não aparecem em dois jogos como mandante ou que não aparecem em dois jogos como visitante.
d) aparecem em dois jogos como mandante ou que aparecem em dois jogos como visitante.
e) aparecem em dois jogos como mandante e aparecem em dois jogos como visitante.

Comentários:

Esse comando basicamente busca, para cada clube, se a quantidade de vezes em que ele foi mandante é diferente de dois e se a quantidade de vezes em que ele foi visitante é diferente de dois. Em outras palavras, o resultado produzido pela execução desse comando é a lista de todos os clubes que não aparecem em dois jogos como mandante ou que não aparecem em dois jogos como visitante.

Gabarito: Letra C

A	B	C	D
12	2	3	1
14	3	8	2
18	2	9	3
21	5	4	4

50. **(FGV / TCE-AM – 2021)** Considerando-se a instância da tabela T (descrita anteriormente), analise o comando SQL abaixo.

```
delete from T where b + d = c
```

O número de registros da tabela T afetados pela execução desse comando é:

a) zero;
b) um;
c) dois;
d) três;

e) quatro.

Comentários:

Linha 1: b = 2, d = 1, c = 3; 2 + 1 = 3? Sim, portanto deleta a primeira linha;
Linha 2: b = 3, d = 2, c = 8; 3 + 2 = 8? Não, portanto não deleta a segunda linha;
Linha 3: b = 2, d = 3, c = 9; 2 + 3 = 9? Não, portanto não deleta a terceira linha;
Linha 4: b = 5, d = 4, c = 4; 5 + 4 = 4? Não, portanto não deleta a quarta linha.

Gabarito: Letra B

A	B	C	D
12	2	3	1
14	3	8	2
18	2	9	3
21	5	4	4

51. **(FGV / TCE-AM – 2021)** Considerando-se a instância da tabela T (descrita anteriormente), analise o comando SQL abaixo.

```
update T
set a = a + 32
where
exists (select * from T t2 where T.c > t2.D)
```

O número de registros da tabela T afetados pela execução desse comando é:

a) zero;
b) um;
c) dois;
d) três;
e) quatro.

Comentários:

(1) Vamos examinar o código:

```
update T
set a = a + 32
where
exists (select * from T t2 where T.c > t2.D)
```

Linha 1: O comando **UPDATE** será utilizado para mudar os valores dos registros da tabela **T**.
Linha 2: O comando **SET** irá realizar a operação **a = a + 32**
Linha 3: O comando **WHERE** irá filtrar dados a partir de um conjunto de condições
Linha 4: O comando **EXISTS** permite testar a existência de qualquer registro em uma subconsulta. Ele retorna **TRUE** se a subconsulta retornar um ou mais registros; caso contrário, retorna **FALSE**.

Linha 4 (subconsulta dentro do **EXISTS**): Basicamente, o comando "**select * from T t2 where T.c > t2.D**" irá selecionar as linhas da tabela **T** em que os registros da coluna **C** são maiores que os registros da coluna **D**.

Importante: **t2** é um alias da tabela **T**. O que isso significa? Um alias é um nome alternativo e temporário para colunas, tabelas, views etc em uma consulta. E por que ele é usado? Para ajudar o usuário em consultas complexas. Em suma, **t2** é igual a **T**.

2) Analisando os valores

Para **C = 3** e **D = 1, 3 > 1**? Sim, logo podemos parar e **A = 12 + 32 = 44**;

Nesse caso, a subconsulta encontrou na primeira linha da tabela um registro da coluna **C** que é maior que o da coluna **D**. Logo, o **EXISTS** irá resultar em **TRUE** e o **SET** atualizará o valor correspondente da linha na coluna A. Lembre-se que o **EXISTS** irá retornar **TRUE** se a consulta retornar um ou mais registros, então basta um registro ser verdadeiro para que se atualize todas as linhas da coluna **A**. A nova tabela ficaria da seguinte forma:

A	B	C	D
44	2	3	1
46	3	8	2
50	2	9	3
53	5	4	4

Por fim, observa-se que foram alterados quatro registros.

Gabarito: Letra E

A	B	C	D
12	2	3	1
14	3	8	2
18	2	9	3
21	5	4	4

52. **(FGV / TCE-AM – 2021)** Considerando-se a instância da tabela T (descrita anteriormente), analise o comando SQL abaixo.

```
select distinct * from T t1, T t2, T t3
```

A execução desse comando produz um resultado que, além da linha de títulos, contém:

a) 8 linhas;
b) 24 linhas;
c) 32 linhas;
d) 64 linhas;
e) 128 linhas.

Comentários:

A tabela T possui quatro linhas, logo o produto cartesiano de t1, t2, t3 será 4 x 4 x 4 = 64 linhas.

Gabarito: Letra D

---

53. (FGV / IMBEL - 2021) Considere a instância da tabela R1 e o comando SQL exibidos a seguir.

A	B
1	2
2	2
3	3
4	3
4	2
4	1
5	0

```
select distinct A
from R1
where A not in
        (select B from R1)
```

Assinale a lista de números que é exibida quando esse comando SQL é executado.

a) 5.
b) 1,2.
c) 2,3.
d) 4,5.
e) 1, 2, 3, 4.

Comentários:

Esse comando busca valores da coluna A {1, 2, 3, 4, 5} que não constam na coluna B {0, 1, 2, 3}. Logo, retornará 4,5.

Gabarito: Letra D

---

54. (FGV / IMBEL – 2021) Considere o comando SQL a seguir, executado num banco de dados relacional com duas tabelas, R1 e R2, contendo 2.000 e 5.000 registros, respectivamente. R1 e R2 possuem chaves primárias definidas.

```
SELECT DISTINCT * FROM A, B
```

Assinale o número de linhas produzidas na execução:

a) 1.
b) 2.000.
c) 5.000.
d) 7.000.
e) 10.000.000

Comentários:

Trata-se de um produto cartesiano, logo teremos 2.000 x 5.000 = 10.000.000 linhas.

Gabarito: Letra E

---

55. (FGV / IMBEL – 2021) Considere a instância da tabela R1 e o comando SQL exibidos a seguir.

A	B
1	2
2	2
3	3
4	3
4	2
4	1
5	0

```
select distinct A
from R1
where exists
     (select * from R1 x Where x.B > R1.A)
```

Assinale a lista de números que é exibida quando esse comando SQL é executado:

a) 5.
b) 1,2.
c) 2,3.
d) 3,4,5.

e) 1, 2, 3, 4.

Comentários:

Para cada valor de A, vamos verificar se os valores da coluna B são maiores. Se ao menos um valor for maior, esse valor de A será retornado. Vejamos:

- Para A = **1** e B = 2; 2 > 1? Sim, portanto podemos parar e retornar A = 1;

- Para A = **2** e B = 2; 2 > 2? Não, portanto vamos verificar o próximo;
- Para A = **2** e B = 2; 2 > 2? Não, portanto vamos verificar o próximo;
- Para A = **2** e B = 3; 3 > 2? Sim, portanto podemos parar e retornar A = 2;

- Para A = **3** e B = 2; 2 > 3? Não, portanto vamos verificar o próximo;
- Para A = **3** e B = 2; 2 > 3? Não, portanto vamos verificar o próximo;
- Para A = **3** e B = 3; 3 > 3? Não, portanto vamos verificar o próximo;
- Para A = **3** e B = 3; 3 > 3? Não, portanto vamos verificar o próximo;
- Para A = **3** e B = 2; 2 > 3? Não, portanto vamos verificar o próximo;
- Para A = **3** e B = 1; 1 > 3? Não, portanto vamos verificar o próximo;
- Para A = **3** e B = 0; 0 > 3? Não, portanto esse valor de A não será retornado.

Seguindo essa lógica, podemos ver que mais nenhum outro valor poderá retornar A. Logo, o resultado desse comando é 1, 2.

Gabarito: Letra B

---

pessoa1	pessoa2	relação
João	Rafael	pai
Maria	Rafael	mãe
Rafael	Gabriela	pai
Gabriela	Rita	mãe
Rita	Bruna	mãe
Bruna	Ana	mãe
Rafael	Rita	avo

56. (FGV / DPE-RJ – 2019) Considere a tabela FAMILIA descrita anteriormente e o comando SQL a seguir.

```
select relação, sum(1)
from familia
group by relação
having count(*) > 1
order by 2 desc, 1
```

Os valores exibidos pela execução desse comando, na ordem, são:

a) `mãe 4`
`pai 2`
`avo 1`

b) `mãe 2`
`pai 4`

c) `pai 2`
`mãe 4`

d) `mãe 4`
`pai 2`

e) `mãe 4`
`pai 2`
`avo ø`

Comentários:

Note que essa consulta retornará os registros agrupados por relação em que, para cada relação, teremos uma coluna com seu somatório de ocorrências. No entanto, somente serão exibidas relações que tenham mais de uma ocorrência e o resultado final será ordenado pela segunda coluna (sum(1)) de forma decrescente e – nos casos de empate – serão ordenados pela primeira coluna de forma ascendente. Observações importantes: sum(1) é o mesmo que count(*) ou count(1); além disso, quando não se informa o tipo de ordenação, o padrão é ascendente.

Dito isso, é possível agrupar relação por pai, mãe e avô – sendo que há 2 ocorrências de pai, 4 ocorrências de mãe e 1 ocorrência de avô. Como a função de agregação foi filtrada de modo que só se retorne aquelas com mais de uma ocorrência, retornará apenas mãe 4 e pai 2.

Gabarito: Letra D

---

57. (FGV / Prefeitura de Niterói-RJ – 2018) A otimização de consultas em gerenciadores de bancos de dados é fundamental para o desempenho do sistema. Consultas escritas em SQL são particularmente propícias à otimização, porque essa linguagem:

a) não é procedural, permitindo a criação de diferentes planos de execução.
b) tem uma sintaxe simples e é largamente utilizada.
c) suporta todas as operações da Álgebra Relacional.
d) permite o uso de subconsultas, facilitando os processos de busca.
e) permite a criação de camadas de software de persistência.

Comentários:

(a) Correto, ela não é procedural – ela é declarativa. Os diferentes planos de execução permitem chegar ao mesmo resultado com pior ou melhor desempenho; (b) Errado, isso não é relevante para otimização; (c) Errado, isso é um requisito de um SGBD e não se relaciona com a otimização; (d) Errado, isso também não é particularmente relevante para otimização de consultas; (e) Errado, isso é uma característica do SGBD e não tem relação com otimização de consultas.

Gabarito: Letra A

Pessoa	Descendente
Ana	Vitoria
João	Maria
João	Rafael
Maria	Tiago
Natalia	Ana
Rafael	Natalia
Ana	Vitoria

58. **(FGV / AL-RO – 2018)** Analise o comando a seguir utilizando a tabela arvore, definida anteriormente.

```
delete from arvore
where exists
          (select * from arvore a
          where a.pessoa = 'João'
          and a.descendente = arvore.pessoa)
```

Assinale o número de registros que é removido na execução desse comando.

a) Zero.
b) Um.
c) Dois.
d) Três.
e) Quatro.

Comentários:

Note que temos uma subconsulta correlacionada. Logo, para cada tupla de **arvore**, temos que verificar em **a** se **a** = "**João**" e **a.descendente** = **arvore.pessoa**. Em **a**, **pessoa** só é "**João**" nas linhas 3 e 4. Nesses casos, os descendentes são **Maria** e **Rafael**, logo duas linhas serão apagadas (**Maria**, **Tiago**) e (**Rafael**, **Natalia**).

Gabarito: Letra C

---

59. (FGV / AL-RO – 2018) Considere o seguinte comando SQL numa instalação MS SQL Server.

```
select A, count(*) X
from T1
where B > 2
group by A
```

Assinale a cláusula order by que seria inválida nesse comando.

a) order by A
b) order by B
c) order by X
d) order by 1
e) order by avg(B)

Comentários:

Imagine uma hipotética tabela T1 que tenha as colunas A e B.

A	B
1	4
2	6
1	7
3	3
2	5

O comando apresentado no enunciado busca selecionar os registros da tabela T1, desde que a coluna B seja maior que 2, de modo que eles estejam agrupados pela coluna A e a coluna X indique quantas vezes os valores de A se repetiram. Vejamos:

A	X
1	2
2	2
3	1

Note que agrupamos A de acordo com a quantidade de vezes que esses valores se repetiram. Nesse cenário, é possível ordenar o resultado por A, por X ou por 1. *Onde vem está dúvida?* Há uma regra no ORDER BY que diz que as colunas dessa cláusula devem estar contidas no SELECT ou dentro de uma função de agregação. A letra (e) está correta, visto que temos uma coluna não referenciada no SELECT, mas está dentro de uma função de agregação. Já a letra (b) é inválida porque a coluna B não está contida nem dentro do SELECT nem em uma função de agregação.

Gabarito: Letra B

60.(FGV / AL-RO – 2018) Considere uma tabela relacional T, com atributos A e B, onde A, isoladamente, constitui a chave primária de T. Considere ainda o comando SQL a seguir.

```
select * from T t1, T t2
where t1.A = t2.A
and exists (select * from T t3
            where t3.A <> t2.A)
```

Dado que essa tabela possui 10 registros, assinale o número de linhas que, além dos títulos, aparece no resultado produzido pela execução desse comando.

a) 0
b) 1
c) 10
d) 100
e) 1000

Comentários:

Antes de começar, temos que fazer quatro observações: (1) note que A é chave primária, logo nenhum valor se repetirá para essa coluna; (2) note que o atributo B é irrelevante para a consulta; (3) note que temos três instâncias da mesma tabela T: t1, t2, t3; (4) note que temos um produto cartesiano (t1, t2) que foi filtrado na cláusula `WHERE` de modo que t1.A = t2.A, logo o resultado é a mesma tabela original T. O examinador fez isso apenas para confundir, mas não tem função!

Dito isso, agora vamos imaginar uma tabela hipotética T com 10 valores: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}. Vamos analisar cada valor e a consulta externa vai retorná-lo caso t3.A <> t2.A. Vejamos:

- O primeiro valor de t3.A é 1 e o primeiro valor de T2.A também é 1. Como é igual, retorna falso;
- O primeiro valor de t3.A é 1 e o segundo valor de T2.A é 2. Como é diferente, retorna verdadeiro;
- Pronto! Não precisamos continuar porque significa que a consulta externa retornará 1;

- O segundo valor de t3.A é 2 e o primeiro valor de T2.A é 1. Como é diferente, retorna verdadeiro;
- Pronto! Não precisamos continuar porque significa que a consulta externa retornará 2;

- O terceiro valor de t3.A é 3 e o primeiro valor de T2.A é 1. Como é diferente, retorna verdadeiro;
- Pronto! Não precisamos continuar porque significa que a consulta externa retornará 3;

...

- O décimo valor de t3.A é 10 e o primeiro valor de T2.A é 1. Como é diferente, retorna verdadeiro;
- Pronto! Não precisamos continuar porque significa que a consulta externa retornará 10.

Entenderam a lógica? Pois é, serão retornadas 10 linhas!

Gabarito: Letra C

---

61. (FGV / MPE-AL – 2018) Considere o comando SQL a seguir.

```
select * from teste where nota <> null
```

Sabendo-se que a tabela teste tem 100 registros, dos quais 15 estão com a coluna nota ainda não preenchida (null), o comando acima, executado no MS SQL Server ou no Oracle, retorna

a) um erro, porque o literal null não pode ser usado diretamente numa comparação.
b) zero linhas.
c) quinze linhas.
d) oitenta e cinco linhas.
e) cem linhas.

Comentários:

Operadores relacionais não podem ser utilizados para comparar valores nulos, portanto essa consulta não retornará nenhuma linha.

Gabarito: Letra B

---

62. (FGV / BANESTES – 2018) Considere um banco de dados com duas tabelas, R e S, contendo 4 e 3 registros, respectivamente. Em R, os valores da coluna A são 1, 2, 5 e 7. Em S, os valores da coluna B são 2, 4 e 7. Excetuando-se a linha de títulos, o número de linhas no resultado do comando SQL

```
select * from R full outer join S on A=B
```

É:

a) 3
b) 4
c) 5
d) 7
e) 12

Comentários:

O `FULL OUTER JOIN` retorna todos os registros mesmo quando não há uma correspondência da tabela esquerda com a direita ou da direita com a esquerda. Portanto, se houver linhas em "Clientes" que não tenham correspondências em "Pedidos", ou se houver linhas em "Pedidos" que

não tenham correspondências em "Clientes", essas linhas também serão listadas. Logo, ele retornaria algo como:

(1, null)
(2, 2)
(null, 4)
(5, null)
(7, 7)

Isso ocorre pois, quando nenhuma linha da tabela da esquerda corresponde a uma linha da tabela da direita, retorna-se NULL; quando nenhuma linha da tabela da direita corresponde a uma linha da tabela da esquerda, também se retorna NULL; e quando há uma correspondência, retorna os dois valores.

Gabarito: Letra C

---

63. (FGV / Prefeitura de Niterói-RJ – 2018) A questão deve ser respondida a partir das tabelas de banco de dados t1 e t2, a seguir.

T1

A	B	C
1	2	4
2	3	5
4	2	4
6	2	NULL

T2

D	E	F
1	2	NULL
2	5	5
4	2	1
7	12	1

Analise o comando SQL exibido abaixo:

```
select  *  from  T1  where  C > 5
UNION
select  * from  T1  where  C <= 5
```

A execução desse comando no MS SQL Server produz um resultado que contém, além da linha de títulos, n linhas. Assinale o valor de n.

a) 3
b) 4
c) 5
d) 6
e) 8

Comentários:

A primeira consulta retorna registros de T1 em que C > 5, logo não retornará nenhum registro porque nenhum deles possui valor maior que 5; a segunda consulta retorna registros de T1 em que C <= 5, logo retornará três registros. *Não seriam quatro registros, professor?* Não, porque operadores relacionais não funcionam com valores nulos, portanto consideram-se apenas os três primeiros registros.

Gabarito: Letra A

64.(FGV / BANESTES – 2018) Considere a tabela de banco de dados R, com a seguinte instância.

a	b
1	2
2	3
4	5

Após a execução do comando SQL

```
update r set a = a + b where b > (select max(a) from r)
```

a instância da referida tabela é, na respectiva ordem de colunas e linhas:

a) 1 2
2 3
4 5

b) 10 2
10 3
10 5

c) 3 2
3 3
4 5

d) 3 2
5 3
9 5

e) 1 2
2 3
9 5

Comentários:

Esse comando nos diz para atualizar a tabela r, de forma que a se torna a + b sempre que um registro tiver b maior que o maior valor de a. O max(a) é 4, logo o único registro que possui b > 4 é o terceiro, já que b = 5 > 4. Nesse caso, a = a + b = 4 + 5 = 9. Dessa forma, o resultado final é:

1 2
2 3
9 5

Gabarito: Letra E

---

65. (FGV / BANESTES – 2018) Considere um banco de dados com duas tabelas. A primeira tabela, números, possui dez registros e apenas uma coluna, cujos valores são 1, 2, 3, 4, 5, 5, 9, 9, 9, 10. A segunda tabela, denominada teste, com cinco registros, também possui apenas uma coluna, cujos valores são 1, 3, 3, 4, 5. Considere ainda o seguinte comando SQL

```
insert into teste select numero from numeros n where not exists (select * from teste t where t.numero = n.numero)
```

Quando da execução desse comando, o número de registros inseridos na tabela teste é:

a) 2;
b) 3;
c) 5;
d) 8;
e) 10.

Comentários:

Essa consulta basicamente nos diz para inserir na tabela teste os números da primeira tabela {1, 2, 3, 4, 5, 5, 9, 9, 9, 10} que não existem na segunda tabela {1, 3, 3, 4, 5}, logo seriam inseridos {2, 9, 9, 9, 10}.

Gabarito: Letra C

---

66. (FGV / SEFIN-RO – 2018) Considere as tabelas de bancos de dados T1, T2 e T3, que contêm, respectivamente, 10, 500 e 2.000 registros, e o comando SQL a seguir.

```
select count(*) FROM T1, T2, T3
```

Assinale a opção que apresenta o número exibido no resultado da execução desse comando.

a) 10000000
b) 1000000

c) 2000
d) 500
e) 10

Comentários:

Temos uma vírgula separando o nome das tabelas, portanto temos um produto cartesiano. Dessa forma, a tabela resultante terá #LinhasT1 x #LinhasT2 x #LinhasT3: 10 x 500 x 2.000 = 10.000.000.

Gabarito: Letra A

---

67. (FGV / IBGE – 2017) A linguagem mais comum para elaboração de consultas em bancos de dados é a SQL. Ao elaborar uma consulta nessa linguagem, emprega-se a cláusula WHERE quando se deseja:

a) especificar a tabela onde será realizada a consulta;
b) especificar o diretório onde os dados estão armazenados;
c) especificar o endereço IP onde os dados estão armazenados;
d) especificar condições a que as instâncias selecionadas devem atender;
e) extrair a geometria do objeto selecionado na consulta.

Comentários:

A cláusula WHERE é responsável por permitir a filtragem dos registros de uma tabela por meio de uma ou mais condições a que as instâncias selecionadas devem atender.

Gabarito: Letra D

R	
a	b
1	2
2	3
4	5

S	
c	d
3	2
4	2
6	1

68. (FGV / MPE-BA – 2017) Considerando as tabelas R e S apresentadas anteriormente, o comando SQL

```
SELECT a FROM R UNION ALL SELECT d FROM S
```

produz um resultado que contém, além dos títulos:

a) 1 linha;
b) 3 linhas;
c) 4 linhas;
d) 5 linhas;
e) 6 linhas.

Comentários:

O UNION ALL unirá as colunas a e d respectivamente das tabelas (R e S), incluindo valores duplicados. O resultado será: {1, 2, 4, 2, 2, 1}, logo retornará seis linhas.

Gabarito: Letra E

R	
a	b
1	2
2	3
4	5

S	
c	d
3	2
4	2
6	1

69. **(FGV / MPE-BA – 2017)** Considere as tabelas R e S apresentadas anteriormente e o comando SQL a seguir.

```
update R set a = NULL
where b >= (select max(d) from S)
```

Após execução desse comando, os valores na coluna a da tabela R seriam, de cima para baixo:

a) NULL, NULL, NULL
b) 1, 2, 4
c) 1, 2, NULL
d) 1, NULL, NULL
e) NULL, 2, 4

Comentários:

Vamos começar pela consulta: seleciona o maior valor da coluna d da tabela S, logo retornará 2. O comando externo nos diz para atualizar a tabela R, inserindo a = NULL sempre que b >= 2. Ao olhar a tabela R, observamos que todos os valores de B são maiores ou iguais a 2, logo atualizará a coluna a para NULL, NULL, NULL.

Gabarito: Letra A

R	
a	b
1	2
2	3
4	5

S	
c	d
3	2
4	2
6	1

70. (FGV / MPE-BA – 2017) Considerando as tabelas R e S apresentadas anteriormente, o resultado

a	b
1	2
2	3

seria obtido pela execução do comando SQL:

a) select * from R
where not exists (select * FROM S where a=c)

b) select * from R
where not exists (select * FROM S where a=d)

c) select * from S
where not exists (select * FROM R where a=c)

d) select * from S
where not exists (select * FROM R where a=d)

e) select * from R
where exists (select * FROM S where b=d)

Comentários:

(a) Correto. Essa consulta basicamente seleciona todas as linhas de R nas quais a é diferente de c (dado que temos um **NOT EXISTS**). 1 e 2 da tabela R não existem em S, logo retorna {(1,2), (2,3)}; (b) Errado, isso retornaria apenas {(4,5)}; (c) Errado, isso retornaria {(3,2), (6,1)}; (d) Errado, isso retornaria nulo; (e) Errado, isso retornaria {((1,2)}.

Gabarito: Letra A

71. (FGV / COMPESA – 2016) Analise o comando SQL a seguir, no contexto das tabelas R1, R2 e R3.

R1

A1	B1
2	3
2	4
1	1

R2

A2	B2
2	3
1	1
1	1

R3

A3	B3	C3
2	3	4
1	2	3
1	1	1

```
select * from R1
where not exists
          (select * from R2
           where R1.A1 = R2.A2 and R1.B1 = R2.B2)
```

Assinale a soma dos valores numéricos exibidos no resultado produzido pelo referido comando.

a) 2
b) 4
c) 6
d) 8
e) 11

Comentários:

A consulta interna basicamente deseja retornar os registros de R2 {(2,3), (1,1), (1,1)} que sejam iguais aos registros de R1 {(2,3), (2,4), (1,1)}. Isso retornará todos os registros de R2, dado que eles estão contidos em R1. No entanto, a consulta externa deseja retornar todos os registros de R1 {(2,3), (2,4), (1,1)} que não existam no resultado da consulta interna {(2,3), (1,1), (1,1)}, logo retornará (2,4). A soma dos valores numéricos será 2 + 4 = 6.

Gabarito: Letra C

---

72. (FGV / IBGE – 2016) O comando SQL

```
select a, sum(b) x, COUNT(*) y
from T
group by a
```

produz como resultado as linhas abaixo.

a	x	y
1	6	1
3	6	2
4	4	1
5	1	1

Na tabela T, composta por duas colunas, a e b, nessa ordem, há um registro duplicado que contém os valores:

a) 1 e 3
b) 3 e 3
c) 3 e 6
d) 4 e 2
e) 5 e 1

Comentários:

A questão afirma que existe uma tabela T com duas colunas: **a** e **b**.

TABELA T	
a	b

Quando executamos a consulta apresentada, retorna a tabela mostrada no enunciado. Agora note que os valores dessa tabela estão agrupados por a. *O que isso significa?* Significa que valores repetidos serão agrupados por essa coluna. Agora perceba que o campo **`count(*)`** conta a quantidade de registros, logo basta procurar o valor de **`count(*) > 1`**. *Como é, Diego?*

Eu estou à procura de um registro duplicado e quem pode me dizer isso é o campo **`count(*)`**. Se ele for maior que 1, significa que o valor agrupado por **a** está repetido. Veja que isso ocorre na segunda linha: (3,6,2). Agora ficou fácil: se a = 3, count(*) = 2 e sum(b) = 6, então b = 3. *Por que, Diego?* Porque a tabela original possuía duas vezes o registro (3,3).

```
select a, sum(b) x, COUNT(*) y
from T
group by a
```

Note que, para a = 3, temos que sum(b) = 6 (3 + 3) e count(*) = 2 (1+1).

Gabarito: Letra B

73. (FGV / IBGE – 2016) Os comandos SQL

```
create table R (a int, b int)
create table S (c int, d int)
insert into R values(1,2)
insert into R values(2,3)
insert into R values(2,3)
insert into R values(3,5)
insert into R values(4,1)
insert into S values(1,2)
insert into S values(2,1)
insert into S values(2,3)
insert into S values(3,5)
select r.a,r.b from R
where not exists
   (select * from S where s.c=r.a and s.d=r.b)
```

Produzem um resultado que, além da linha de títulos, contém:

a) uma linha;
b) duas linhas;
c) três linhas;
d) quatro linhas;
e) quatro linhas;

Comentários:

As linhas 1-2 criam duas tabelas R e S com duas colunas de valores inteiros; as linhas 3-11 inserem valores nas duas tabelas:

TABELA R	
a	b
1	2
2	3
2	3
3	5
4	1

TABELA S	
c	d
1	2
2	1
2	3
3	5

A linha 14 seleciona todas as colunas de S em que s.c = r.a e s.d = r.b. Em outras palavras, selecionará todos os registros em que as colunas de S são iguais às colunas de R. Logo, retornará os registros (1,2), (2,3) e (3,5). Por fim, a linha 12 retornará as colunas a e b de R desde que não sejam os registros encontrados na consulta interna: {(1,2), (2,3), (2,3), (3,5), (4,1)} – {(1,2), (2,1), (2,3), (3,5)} = (4,1). Logo, retornará apenas uma linha.

Gabarito: Letra A

---

74. (FGV / TJ-PI – 2015) Considerando a tabela T, analise o comando SQL a seguir.

T

a	b	c
1	2	NULL
2	3	NULL
5	NULL	NULL
4	2	NULL

```
delete from T
where exists
          (select * from T t1
           where T.a > t1.c or T.a <= t1.c)
```

O número de registros da tabela T removidos pela execução desse comando é:

a) zero;
b) um;
c) dois;
d) três;
e) quatro.

Comentários:

Esse comando deleta da **Tabela T** todos os registros em que o valor da coluna **a** seja maior que o valor da coluna **c** ou que o valor da coluna **a** seja menor ou igual ao valor da coluna **c**. No entanto, note que a coluna **c** é toda **NULL** e não há como utilizar operadores relacionais com valores nulos, portanto retornará falso para todas as linhas e nenhum registro será retornado. Para realizar comparações com valores nulos, utilizam-se os operadores **IS NULL** ou **IS NOT NULL**.

Gabarito: Letra A

---

75. **(FGV / PGE-RO – 2015)** No SQL, a outorga de privilégios sobre objetos de um banco de dados é efetuada por meio do comando:

a) CREATE;
b) GRANT;
c) LICENSE;
d) PERMIT;
e) REVOKE.

Comentários:

O comando DCL responsável por outorgar privilégios sobre objetos de um banco de dados é o comando **GRANT**.

Gabarito: Letra B

---

76. (FGV / DPE-RO – 2015) Observe o comando SQL a seguir.

```
update X set Y = 'Z'
```

Para que esse comando esteja corretamente formulado, quando analisado isoladamente, pressupõe-se que:

a) Y seja uma coluna da tabela X;
b) X seja uma coluna da tabela Y;
c) X e Y sejam tabelas;
d) X seja um banco de dados e Y seja uma tabela;
e) X seja um banco de dados e Y seja uma tabela;

Comentários:

Para esse comando funcionar corretamente, assume-se que X é uma tabela e que Y é uma de suas colunas. Vamos lembrar da sintaxe:

SINTAXE DO COMANDO

```
UPDATE NOME_DA_TABELA
SET NOME_DA_COLUNA_1 = VALOR_1, NOME_COLUNA2 = VALOR_2 ...
WHERE LISTA_DE_CONDICOES
```

Gabarito: Letra A

---

77. (FGV / DPE-RO – 2015) Analise o comando SQL a seguir.

```
select distinct 1 from X
```

Sabendo-se que a instância da tabela X não é vazia, conclui-se que a execução desse comando produz um resultado com:

a) apenas uma linha;
b) um conjunto de linhas contendo o valor NULL;
c) um conjunto de linhas contendo todos os atributos de X;
d) um número de linhas igual ao número de diferentes valores do primeiro atributo de X;
e) um número de linhas igual ao número de registros de X.

Comentários:

Esse comando retornará o valor 1 para cada linha da tabela, mas como há um operador distinct, ele elimina duplicatas e retorna apenas uma linha com o valor 1.

Gabarito: Letra A

---

78. (FGV / TJ-BA – 2015) Considere que as instâncias das tabelas T1, T2 e T3 têm, respectivamente, 1.000, 10.000 e 100.000 registros. O comando SQL:

```
select 1 from t1
union
select 2 from t2
union
select 3 from t3
```

produz um resultado com:

a) 3 linhas;
b) 1.000 linhas;
c) 10.000 linhas;
d) 100.000 linhas;
e) 111.000 linhas.

Comentários:

Pegadinha clássica: **UNION** basicamente une tabelas que tenham estruturas idênticas. Logo, a tabela resultante teria 1.000 + 10.000 + 100.000 = 111.000 registros. No entanto, observe que há um número após o **select**, em vez do nome de uma coluna. Quando há uma constante em vez do nome de uma coluna, o comando retorna esse número para cada um dos registros. Logo, ele retorna 1.000 vezes o valor 1; depois retorna 10.000 vezes o valor 2; e depois retorna 100.000 vezes o valor 3.

select 1 from t1 - retorna as 1.000 linhas contendo o número 1.
select 2 from t2 - retorna as 10.000 linhas contendo o número 2.
select 3 from t3 - retorna as 100.000 linhas contendo o número 3.

Professor, isso não daria 111.000 registros do mesmo jeito? Não, porque o **UNION** elimina duplicadas, então o comando apresentado no enunciado retornaria apenas três linhas: 1, 2, 3. Para retornar as 111.000 linhas, deveria ser utilizado o comando **UNION ALL**.

Gabarito: Letra A

---

79. (FGV / PGE-RO – 2015) No SQL, a outorga de privilégios sobre objetos de um banco de dados é efetuada por meio do comando:

a) CREATE;

b) GRANT;
c) LICENSE;
d) PERMIT;
e) REVOKE.

Comentários:

A outorga (sinônimo de concessão) de privilégios sobre objetos de um banco de dados é efetuada por meio do comando **GRANT**.

Gabarito: Letra B

80.(FGV / TCE-SE – 2015) Considere duas tabelas X e Y, com as seguintes instâncias:

X	
a	b
1	2
3	3
4	5
5	7

Y	
c	d
1	2
3	4
5	6
7	8
9	1

O comando SQL que retorna

a	b	c	d
1	2	1	2
3	3	3	4
4	5	NULL	NULL
5	7	5	6
NULL	NULL	7	8
NULL	NULL	9	1

é:

a) select * from X FULL JOIN Y on X.a=Y.c
b) select * from X LEFT JOIN Y on X.a=Y.c
c) select * from Y RIGHT JOIN X on X.a=Y.c
d) select * from X CROSS JOIN Y on X.a=Y.c
e) select * from X INNER JOIN Y on X.a=Y.c

Comentários:

Note que o parâmetro de comparação é X.a = Y.c. Sabemos que X.a = {1, 3, 4, 5} e Y.c = {1, 3, 5, 7, 9}. Nem é necessário olhas as outras colunas, basta verificar as colunas (a,c) conforme vemos a seguir:

Se fosse um `INNER JOIN`, retornaria {(1,1),(3,3),(5,5)};

a	b	c	d
1	2	1	2
3	3	3	4
5	7	5	6

Se fosse um `LEFT JOIN`, retornaria {(1,1),(3,3),(4,NULL),(5,5)}

a	b	c	d
1	2	1	2
3	3	3	4
4	5	NULL	NULL
5	7	5	6

Se fosse um `RIGHT JOIN`, retornaria {(1,1),(3,3),(5,5),(NULL,7),(NULL,9)}

a	b	c	d
1	2	1	2
3	3	3	4
5	7	5	6
NULL	NULL	7	8
NULL	NULL	9	1

Se fosse um `FULL JOIN`, retornaria {(1,1),(3,3),(4,NULL),(5,5),(NULL,7),(NULL,9)}

a	b	c	d
1	2	1	2
3	3	3	4
4	5	NULL	NULL
5	7	5	6
NULL	NULL	7	8
NULL	NULL	9	1

O `CROSS JOIN` é um produto cartesiano, logo teríamos uma tabela com 16 linhas.

Gabarito: Letra A

---

81. (FGV / TJ-RO – 2015) Considere as seguintes tabelas relacionais e respectivas instâncias.

R

A	B
1	3
2	3
3	4
5	5
5	7
8	8

S

C	D
1	9
2	3
3	4
3	4
3	4

Analise o comando SQL a seguir.

```
DELETE FROM S
WHERE NOT EXISTS
(SELECT * FROM R
WHERE R.A = S.C AND R.B = S.D)
```

O número de registros deletados por esse comando é:

a) 0;
b) 1;
c) 2;
d) 4;
e) 5.

Comentários:

Note que temos uma subconsulta correlacionada, logo vamos deletar da tabela S tuplas que sejam iguais a alguma tupla da tabela R.

Temos que R(A,B) e S(C,D). Logo, vamos analisar as tuplas:

Para S = (1,9):

- R.A (1) = S.C (1)? Sim. R.B (3) = S.D (9)? Não, logo vamos para o próximo;
- R.A (2) = S.C (1)? Não, logo vamos para o próximo;
- R.A (3) = S.C (1)? Não, logo vamos para o próximo;
...
- R.A (8) = S.C (1)? Não! Como nenhum registro é igual, retorna TRUE e o registro é deletado;

Para S = (2,3):

R.A (2) = S.C (1)? Não, logo vamos para o próximo;
R.A (2) = S.C (2)? Sim. R.B (3) = S.D (3)? Sim, logo retorna FALSO e o registro não é deletado;

Vocês entenderam o padrão? Nós verificamos – para cada tupla de S – se ela não existe em R. Se não existir, ela é apagada. Note que isso só ocorre uma única vez, porque apenas S(1,9) não existe em R(A,B). Todas as outras tuplas de S existem em R.

Gabarito: Letra B

---

82. (FGV / TJ-RO – 2015) Considere as seguintes tabelas relacionais e respectivas instâncias.

R

A	B
1	3
2	3
3	4
5	5
5	7
8	8

S

C	D
1	9
2	3
3	4
3	4
3	4

Analise o comando SQL a seguir.

```
SELECT * FROM R UNION SELECT * FROM S
```

O número de linhas produzidas por esse comando, excetuada a linha de títulos de colunas, é:

a) 2;
b) 5;
c) 6;
d) 7;
e) 11.

Comentários:

O UNION elimina tuplas duplicadas, logo ela retornará 7 tuplas: (1,3), (2,3), (3,4), (5,5), (5,7), (8,8), (1,9).

Gabarito: Letra D

---

83. (FGV / DPE-RO – 2015) Analise o comando SQL a seguir.

```
SELECT DISTINCT 1 FROM X
```

Sabendo-se que a instância da tabela X não é vazia, conclui-se que a execução desse comando produz um resultado com:

a) apenas uma linha;
b) um conjunto de linhas contendo o valor NULL;
c) um conjunto de linhas contendo todos os atributos de X;

d) um número de linhas igual ao número de diferentes valores do primeiro atributo de X;
e) um número de linhas igual ao número de registros de X.

Comentários:

Se o comando fosse `SELECT 1 FROM X`, retornaria o valor constante 1 para todas as linhas de X. *Professor, mas eu não sei nem quantas linhas existem em X.* Não precisa, porque – quando inserimos o `DISTINCT` – retornará apenas valores distintos. Como só existe valor 1, não existem valores distintos, logo retornará apenas uma linha.

Gabarito: Letra A

84.(FGV / TCE-SE – 2015) Considere duas tabelas X e Y, com as seguintes instâncias:

X	
a	b
1	2
3	3
4	5
5	7

Y	
c	d
1	2
3	4
5	6
7	8
9	1

```
delete from y
where y.c in
(select a from x union select c from y)
```

Considerando-se as tabelas e o comando SQL mostrados acima, é correto concluir que o número de registros removidos da tabela Y pela execução desse comando é:

a) 1
b) 2
c) 3
d) 4
e) 5

Comentários:

A subconsulta interna une a coluna **a** de **x** com **c** de **y**, eliminando as duplicatas: {1, 3, 4, 5, 7, 9}. Dessa forma, vamos deletar as tuplas de y em que y.c {1, 3, 5, 7, 9} esteja dentro de {1, 3, 4, 5, 7, 9}. Logo, temos cinco valores: {1, 3, 5, 7, 9}.

Gabarito: Letra E

85.(FGV / DPE-MT – 2015) Analise o comando SQL a seguir.

```
select
   x,
   sum (nota) as soma,
   count (nota) as numero
from inscrição
group by x
having ???
```

Assinale a opção que indica a expressão que, ao ser utilizada para substituir o trecho "???", INVALIDA o comando SQL acima.

a) count (nota) > 1
b) x=2
c) max (nota) = 10
d) nota > 7
e) (select max(nota) from inscricao) > 5

Comentários:

As colunas contidas no **HAVING** devem estar entre aquelas do **SELECT** (**x**, **soma**, **numero**) ou estar contida dentro de uma função de agregação. Logo, temos que: (a) Correto, está dentro de uma função de agregação; (b) Correto, está entre as colunas do **SELECT**; (c) Correto, está dentro de uma função de agregação; (d) Errado, não obedece a nenhuma das regras; (e) Correto, está dentro de uma função de agregação.

Gabarito: Letra D

---

86. (FGV / DPE-MT – 2015) Na maioria das implementações SQL, pode-se considerar que as expressões lógicas possam assumir três valores, verdadeiro (T), falso (F) e desconhecido (?). Isso decorre principalmente da manipulação de valores nulos (NULL). Assim sendo, analise as quatro expressões lógicas a seguir.

```
not ?
F or ?
T and ?
? or T
```

Assinale a opção que apresenta os valores finais das expressões lógicas acima, na ordem de cima para baixo.

a) F; ?; T; T
b) F; F; T; T
c) ?; ?; ?; ?
d) ?; ?; ?; T

e) ?; F; ?; ?

Comentários:

Essa é uma questão mais de lógica do que de SQL. Nós sabemos que: (1) quando temos um **OR**, basta que um dos operandos seja verdadeiro (T) para retornar verdadeiro, caso contrário nada se pode afirmar; (2) quando temos um **AND**, basta que um dos operandos seja falso (F) para retornar falso, caso contrário nada se pode afirmar; (3) quando temos um **NOT**, nada se pode afirmar porque dependerá do valor original. Logo, temos que:

(1) not ? = **?**
(2) F or ? = **?**
(3) T and ? = **?**
(4) ? or T = **T**

Gabarito: Letra D

---

87. (FGV / TJ-AM – 2013) A SQL é constituída pela Data Control Language (DCL), a Data Definition Language (DDL) e a Data Manipulation Language (DML).

Assinale a alternativa que apresenta os três comandos que são parte integrante da DDL:

a) REVOKE, GRANT e DELETE
b) GRANT, DELETE e UPDATE
c) DELETE, UPDATE e CREATE
d) CREATE, ALTER e DROP
e) ALTER, DROP e REVOKE

Comentários:

(a) REVOKE é DCL; GRANT é DCL; e DELETE é DML; (b) GRANT é DCL, DELETE é DML; e UPDATE é DML; (c) DELETE é DML; UPDATE é DML; e CREATE é DDL; (d) CREATE é DDL; ALTER é DDL; e DROP é DDL; (e) ALTER é DDL; DROP é DDL; e REVOKE é DCL;

Gabarito: Letra D

---

88. (FGV / MPE-MS – 2013) Observe o comando SQL a seguir:

```
SELECT nome, sobrenome, PIS, anos_de_servico FROM Empregados
```

A cláusula que deve ser adicionada ao comando acima para ordenar os registros por anos de serviço, com os empregados que estão há mais tempo na empresa aparecendo primeiro na listagem, é:

a) ORDER 'anos_de_servico' BY ASC
b) ORDER BY 'anos_de_servico'
c) ORDER BY anos_de_servico DESC
d) SORTED BY anos_de_servico DESC
e) ORDER BY anos_de_servico ASC

Comentários:

Para ordenar os registros por anos de serviço (`ORDER BY`), com os empregados que estão há mais tempo na empresa aparecendo primeiro na listagem (`DESC`). Logo, temos:

```
ORDER BY anos_de_servico DESC.
```

Gabarito: Letra C

QUESTÕES COMENTADAS – DIVERSAS BANCAS

89. (FEPESE / ISS-Criciúma – 2022) No contexto de uma consulta SQL com o objetivo de retornar os resultados de duas relações, sendo que o resultado desejado envolve todos os registros da primeira relação em conjunto com os resultados dos registros correspondentes à condição da junção da segunda relação, qual instrução deve ser utilizada?

a) JOIN
b) INNER JOIN
c) OUTER JOIN
d) RIGHT JOIN
e) LEFT JOIN

Comentários:

Objetivo de retornar os resultados de duas relações? O resultado envolve todos os registros da primeira relação em conjunto com os resultados dos registros correspondentes à direção da junção da segunda relação? Ora, trata-se do LEFT JOIN porque queremos retornar todos os registros da relação da esquerda com seus respectivos correspondentes da direita.

Gabarito: Letra E

---

90. (FADESP / SEFA-PA – 2022) A linguagem de banco de dados relacional SQL (Structured Query Language) é um exemplo de linguagem de banco de dados abrangente que representa uma combinação de:

a) TTL, VDL e DML.
b) TDL, GDL e DML.
c) DDL, VDL e DML.
d) DDL, VDL e BML.
e) DDL, GDL e BML.

Comentários:

Sendo bastante abrangente, podemos afirmar que: SQL = DML + DDL + DCL + DTL + DQL + VDL. A questão se limitou a exemplificar apenas: DML (Manipulação), DDL (Definição) e VDL (Visão). Lembrando que não existe TTL, GDL ou BML.

Gabarito: Letra C

As questões 06, 07 e 08 baseiam-se na Figura 6, que mostra, esquematicamente, um Diagrama Entidade-Relacionamento (DER) elaborado no MySQL Workbench 8.0, no qual se inseriu,

intencionalmente, nos locais apontados pelas setas nº 1 e 2, retângulos para ocultar os relacionamentos existentes nesses locais. Nesse DER, constam as entidades "Produto", "Aquisicao" e "Cliente", implementadas de acordo com as seguintes regras de negócio:

(1) um cliente poderá adquirir um ou mais produtos, inclusive os mesmos produtos mais de uma vez, em data/hora diferentes;

(2) um produto poderá ser adquirido por um ou mais clientes, inclusive o mesmo cliente, mais de uma vez;

(3) deve ser possível cadastrar qualquer produto ou cliente, no banco de dados, sem associá-los a qualquer outra tabela;

(4) ao se associar um cliente a um produto, armazena-se, no banco de dados, a quantidade adquirida, a correspondente data/hora de aquisição e o preço efetivamente pago (que poderá ser diferente do preço de tabela do produto, devido ao cliente ter recebido um desconto no preço do produto).

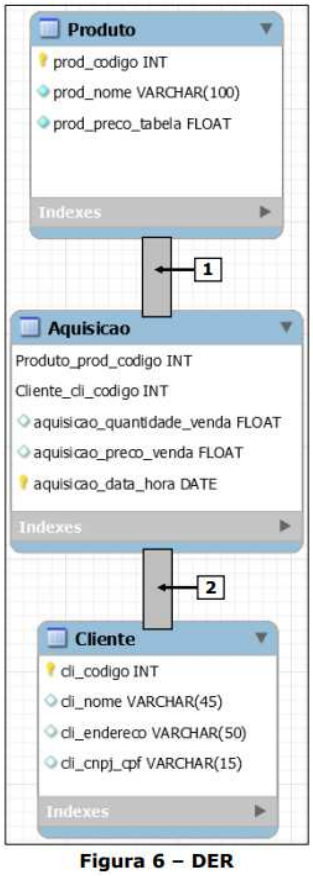


Figura 6 – DER

91.(FUNDATEC / ISS-Porto Alegre – 2022) Sabe-se que, a partir do DER mostrado na Figura 6, foram criadas e populadas as tabelas correspondentes em um Sistema Gerenciador de Banco de Dados Relacional (SGBDR), tendo se respeitado, rigorosamente, os conceitos do modelo

relacional. Nesse caso, para criar a tabela "Aquisicao", bastou executar a seguinte declaração, em SQL padrão ANSI:

a)
```
CREATE TABLE Aquisicao (
  Produto_prod_codigo INT NOT NULL,
  Cliente_cli_codigo INT NOT NULL,
  aquisicao_quantidade_venda FLOAT NOT NULL,
  aquisicao_preco_venda FLOAT NOT NULL,
  aquisicao_data_hora DATE NOT NULL,
  PRIMARY KEY (aquisicao_data_hora, Produto_prod_codigo, Cliente_cli_codigo),
  FOREIGN KEY (Produto_prod_codigo) REFERENCES Produto (prod_codigo),
  FOREIGN KEY (Cliente_cli_codigo) REFERENCES Cliente (cli_codigo)
);
```

b)
```
CREATE TABLE Aquisicao (
  Produto_prod_codigo INT NOT NULL,
  Cliente_cli_codigo INT NOT NULL,
  aquisicao_quantidade_venda FLOAT NULL,
  aquisicao_preco_venda FLOAT NULL,
  aquisicao_data_hora DATE NOT NULL,
  PRIMARY KEY (Produto_prod_codigo, Cliente_cli_codigo),
  FOREIGN KEY (Produto_prod_codigo) REFERENCES Produto (prod_codigo),
  FOREIGN KEY (Cliente_cli_codigo) REFERENCES Cliente (cli_codigo)
);
```

c)
```
CREATE TABLE Aquisicao (
  Produto_prod_codigo INT PRIMARY KEY,
  Cliente_cli_codigo INT PRIMARY KEY,
  aquisicao_quantidade_venda FLOAT NOT NULL,
  aquisicao_preco_venda FLOAT NOT NULL,
  aquisicao_data_hora DATE PRIMARY KEY,
  FOREIGN KEY (Produto_prod_codigo) REFERENCES Produto (prod_codigo),
  FOREIGN KEY (Cliente_cli_codigo) REFERENCES Cliente (cli_codigo)
);
```

d)
```
CREATE TABLE Aquisicao (
  Produto_prod_codigo INT PRIMARY KEY REFERENCES Produto (prod_codigo),
  Cliente_cli_codigo INT PRIMARY KEY ) REFERENCES Cliente (cli_codigo),
  aquisicao_quantidade_venda FLOAT NULL,
  aquisicao_preco_venda FLOAT NULL,
  aquisicao_data_hora DATE PRIMARY KEY
);
```

e)
```
CREATE TABLE Aquisicao (
  Produto_prod_codigo INT PRIMARY KEY REFERENCES Produto (prod_codigo) NOT NULL,
  Cliente_cli_codigo INT PRIMARY KEY ) REFERENCES Cliente (cli_codigo) NOT NULL,
  aquisicao_quantidade_venda FLOAT NOT NULL,
  aquisicao_preco_venda FLOAT NOT NULL,
  aquisicao_data_hora DATE NOT NULL
);
```

Comentários:

(a) Correto. Observe que temos: **Produto_prod_codigo** e **Cliente_cli_codigo**, que são atributos do tipo inteiro e não-nulos (dado que fazem parte da chave primária composta); **aquisição_quantidade_venda** e **aquisicao_preco_venda**, que são atributo do tipo ponto flutuante e não-nulos (dado a questão afirma que, ao associar um cliente a um produto, deve-se armazenar a

quantidade adquirida e o preço pago); e **aquisicao_data_hora**, que é um atributo do tipo data e não-nulo (dado que a questão afirma que, ao associar um cliente a um produto, deve-se armazenar a data/hora de aquisição). Por fim, a chave primária é composta pelos atributos **aquisicao_data_hora**, **Produto_prod_codigo**, **Cliente_cli_codigo**; e as chaves estrangeiras são **Produto_prod_codigo**, que referencia **prod_codigo** e **Cliente_cli_codigo**, que referencia **cli_codigo**. Logo, nenhum erro no item!

(b) Errado, dado que o atributo **aquisicao_data_hora** deve fazer parte da chave primária composta; (c) Errado, quando se tem uma chave primária composta, a indicação deve ser feita em uma cláusula separada e, não, ao lado de cada atributo; (d) Errado, quando se tem uma chave primária composta, a indicação deve ser feita em uma cláusula separada e, não, ao lado de cada atributo; (e) Errado, quando se tem uma chave primária composta, a indicação deve ser feita em uma cláusula separada e, não, ao lado de cada atributo.

Gabarito: Letra A

---

92. (INAZ DO PARÁ / CORE-SP – 2019) SQL é uma linguagem de banco de dados abrangente, que possui instruções para definições de dados, consultas e atualizações. Apresenta alguns comandos principais e, entre eles, o "CREATE TABLE", que serve para:

a) Definir uma nova tabela de banco de dados relacional, criando as relações necessárias aos registros dessa tabela.
b) Especificar uma nova relação, dando-lhe um nome e especificando seus atributos e restrições iniciais.
c) Criar uma nova tabela a partir dos registros encontrados em tabelas diversificadas de banco de dados distintos.
d) Juntamente com o comando "CREATE DOMAIN", estabelecer todos os endereços dos registros de uma tabela.
e) Definir uma tabela centralizada a partir dos registros encontrados em banco de dados distribuídos.

Comentários:

(a) Errado, ela serve para definir uma nova tabela de banco de dados relacional, criando as ~~relações~~ relacionamentos necessárias aos registros dessa tabela – questão polêmica; (b) Correto; (c) Errado, isso é até possível por meio de uma subquery, mas não se trata de seu objetivo principal; (d) Errado, esse comando não existe; (e) Errado, isso é até possível por meio de uma subquery, não se trata de seu objetivo principal.

Galera, péssima questão que dá margem para várias interpretações...

Gabarito: Letra B

---

93. (INAZ DO PARÁ / CORE-SP – 2019) "Embora o SQL tenha sido originalmente criado pela IBM, rapidamente surgiram vários "dialectos" desenvolvidos por outros produtores. Essa expansão levou à necessidade de ser criado e adaptado um padrão para a linguagem. Esta tarefa foi realizada pela American National Standards Institute (ANSI) em 1986 e ISO em 1987."

Disponível em: https://pt.wikipedia.org/wiki/SQL. Acesso em: 13.12.2018

Qual o código SQL contém comandos do tipo DDL?

a) drop table questoes.
b) insert into questoes select * from questoesmodelo.
c) delete from questoes.
d) select * from questoes.
e) select * from questoes.

Comentários:

(a) Correto; (b) Errado, trata-se de um comando DML; (c) Errado, trata-se de um comando DML; (d) Errado, trata-se de um comando DML; (e) Errado, trata-se de um comando DML.

Gabarito: Letra A

---

94. (CCV / UFC – 2019) Em alguns cenários, é necessário definir que uma coluna em um banco de dados não deve permitir a inserção de valores repetidos. Qual das cláusulas abaixo deverá ser usada no comando SQL (Structured Query Language) para aplicar essa restrição no momento da criação da coluna?

a) CHECK
b) DEFAULT
c) UNIQUE
d) DISTINCT
e) CONSTRAINT

Comentários:

A restrição que impede valores repetidos é chamada de UNIQUE.

Gabarito: Letra C

---

95. (CCV / UFC – 2019) Uma tabela chamada Area possui dois campos: arecod e aredes. Como podemos inserir um novo registro na tabela "Area"?

a) INSERT INTO Area (arecod, aredes) VALUES (100, "Técnico"), (200, "TI").
b) INSERT (100, "Técnico"), (200, "TI") INTO Area VALUES(arecod, aredes).
c) INSERT (arecod, aredes) INTO Area VALUES (100, "Técnico"), (200, "TI").
d) INSERT INTO (arecod, aredes) Area VALUES (100, "Técnico"), (200, "TI").
e) INSERT (100, "Técnico"), (200, "TI") INTO Area (arecod, aredes).

Comentários:

Para inserir um novo registro na tabela "Área" que possui os campos arecod e aredes, usa-se:

```
INSERT INTO Area (arecod, aredes) VALUES (?, ?), (?, ?), (?,?)
```

As letras (b), (c) e (e) já podem ser eliminados por o comando é INSERT INTO. Já a letra (d) está errada porque é `INSERT INTO Tabela (Colunas)` e, não, `INSERT INTO (Colunas) Tabela`.

Gabarito: Letra A

---

96. (CCV / UFC – 2019) Utilizando SQL, como selecionamos todos os registros de uma tabela chamada "Pessoas" onde o valor da coluna "PrimeiroNome " começa com "a"?

a) SELECT * FROM Pessoas WHERE PrimeiroNome='a'
b) SELECT * FROM Pessoas WHERE PrimeiroNome LIKE 'a%'
c) SELECT * FROM Pessoas WHERE PrimeiroNome='%a%'
d) SELECT * FROM Pessoas WHERE PrimeiroNome LIKE '%a'
e) SELECT * FROM Pessoas WHERE PrimeiroNome HAVING='%a%'

Comentários:

Se começa com "a", basta utilizar o operador LIKE 'a%'.

Gabarito: Letra B

---

97. (NC-UFPR / Itaipu Binacional – 2019) A recursividade presente em consultas realizadas com SQL na forma SELECT a.id,... FROM a WHERE ... IN (SELECT atributo FROM b WHERE b.x=a.id) pode ser evitada por meio:

a) da substituição do operador IN por EXISTS.
b) da junção externa do tipo RIGHT JOIN com a verificação de atributos de b com o valor nulo.
c) da junção interna – INNER JOIN.
d) da junção externa do tipo LEFT JOIN com a verificação de atributos de b com o valor nulo.
e) da utilização de expressões de tabelas comuns (CTE).

Comentários:

Vamos pensar um pouquinho: quando temos consultas aninhadas com o operador IN, queremos saber se valores do SELECT externo coincidem com valores do SELECT interno. *Quem faz exatamente a mesma coisa?* INNER JOIN! Ele retorna elementos em comum entre dois conjuntos!

Gabarito: Letra C

```
SELECT DISTINCT cra
FROM TRABALHO
WHERE (forma_atuacao, uf) IN (
      SELECT forma_atuacao, uf FROM TRABALHO WHERE cra=2019);
```

98. (QUADRIX / CRA-PR – 2019) Uma consulta aninhada pode retornar tanto um único atributo quanto vários atributos e(ou) várias tuplas.

Comentários:

O resultado de uma subconsulta é utilizado como argumento para uma consulta externa e pode conter uma única linha, múltiplas linhas ou múltiplas linhas e colunas.

Gabarito: Correto

```
SELECT DISTINCT cra
FROM TRABALHO
WHERE (forma_atuacao, uf) IN (
      SELECT forma_atuacao, uf FROM TRABALHO WHERE cra=2019);
```

99. (QUADRIX / CRA-PR – 2019) O operador DISTINCT não pode ser utilizado em consultas aninhadas.

Comentários:

Claro que pode! Não existe essa limitação...

Gabarito: Errado

```
SELECT DISTINCT cra
FROM TRABALHO
WHERE (forma_atuacao, uf) IN (
      SELECT forma_atuacao, uf FROM TRABALHO WHERE cra=2019);
```

100. (QUADRIX / CRA-PR – 2019) A instrução demonstra que é permitido o uso de tuplas de valores em comparações, colocando-os entre parênteses, em consultas do tipo aninhada

Comentários:

Perfeito! É permitido, sim, utilizar uma tupla de valores de colunas para realizar comparações em consultas aninhadas.

Gabarito: Correto

```
SELECT DISTINCT cra
FROM TRABALHO
WHERE (forma_atuacao, uf) IN (
     SELECT forma_atuacao, uf FROM TRABALHO WHERE cra=2019);
```

101. (QUADRIX / CRA-PR – 2019) A instrução contém erro clássico de construção, pois, em uma consulta aninhada ou subconsulta, não é permitido o uso de nomes de tabelas repetidos, como, nesse caso, ocorre com a tabela TRABALHO.

Comentários:

É permitido, sim, o uso de nomes de tabelas repetidos!

Gabarito: Errado

---

102. (NC-UFPR / Itaipu Binacional – 2019) Considerando a linguagem SQL (Structured Query Language) para sistemas de banco de dados, assinale a alternativa que remove linhas de uma tabela chamada CLIENTE.

a) REMOVE FROM CLIENTE ...
b) CUT FROM CLIENTE ...
c) DELETE FROM CLIENTE WHERE ...
d) ERASE FROM CLIENTE ...
e) CLEAR FROM CLIENTE ...

Comentários:

O comando que permite remover linhas de uma tabela é:

```
DELETE FROM CLIENTE
WHERE Condição
```

Gabarito: Letra C

---

103. (IADES / CRF - TO - 2019) A Linguagem de Consulta Estruturada (SQL – Structured Query Language) foi padronizada para utilização em bancos de dados em 1986 e é amplamente utilizada por diferentes Sistemas Gerenciadores de Bancos de Dados (SGBDs). Essa linguagem é dividida em quatro conjuntos, sendo eles linguagens:

a) de estruturação, de dados, para argumentação de controles e orientada a objetos.
b) orientada à conexão, estruturada, de manipulação de dados e de paralelismo.
c) para argumentação de controles, de definição de dados, orientada à conexão e de paralelismo.
d) para controle de acesso a dados, para transações, orientada a objetos e de estruturação.
e) de manipulação de dados, de definição de dados, para controle de transações e para controle de acesso a dados.

Comentários:

Essa linguagem é dividida em quatro conjuntos, sendo eles linguagens de manipulação de dados (DML), definição de dados (DDL), controle de transações (DTL) e controle de acesso a dados (DCL).

Gabarito: Letra E

---

104. (Gestão Concurso / EMATER MG – 2018) Para se implementar um banco de dados relacional, bem como para armazenar os dados e posteriormente recuperá-los, faz-se uso da linguagem:

a) DDL.
b) DML.
c) SQL.
d) SGBD.

Comentários:

Implementar um banco de dados relacional assim como armazenar dados e recuperá-los são todos recursos providos pela linguagem SQL (Structured Query Language).

Gabarito: Letra C

---

105. (SUGEP - UFRPE / UFRPE – 2018) A respeito da linguagem SQL, analise as proposições abaixo.

1) Um dos comandos do DML (Data Manipulation Language) é o INSERT.
2) CREATE e DELETE são comandos da DDL (Data Definition Language).
3) Dois comandos do DCL (Data Control Language) são UPDATE E DROP.

Está(ão) correta(s), apenas:

a) 1.
b) 1 e 2.
c) 2 e 3.

d) 3.
e) 1 e 3.

Comentários:

(1) Correto, INSERT é DML; (2) Errado, DELETE é DML; (3) Errado, UPDATE é DML e DROP é DDL.

Gabarito: Letra A

---

106. (FAURGS / UFRGS - 2018) Analise o trecho de código abaixo, escrito em SQL.

```
SELECT nome_pessoa
FROM PESSOA_FISICA
WHERE nome_pessoa NOT IN (‘Benedita Silva’, ‘José
Silva’, ‘Lúcia Silva’, ‘João Silva’)
```

Com base nesse código, é correto afirmar que o resultado da consulta é:

a) Benedita Silva.
b) José Silva.
c) João Silva.
d) Lúcia Silva.
e) Rosa Silva.

Comentários:

Essa consulta retornará nomes de todas as pessoas física que não se chamem: “Benedita Silva”, José Silva”, “Lúcia Silva” ou “João Silva”. Logo, a única possibilidade entre aquelas apresentadas na questão é “Rosa Silva”.

Gabarito: Letra E

---

107. (COPESE / UFPI – 2018) As definições de uma tabela básica ou de outros elementos do esquema que possuírem denominação poderão, em linguagem SQL, ser alteradas pelo comando:

a) ALTER
b) DROP.
c) RESET.
d) RESET.
e) TABLE.

Comentários:

Alterações de definições de uma tabela podem ser feitas por meio do comando ALTER.

Gabarito: Letra A

```
SELECT Cadastro.NomeCliente, Ordens.NumOrdem
FROM Cadastro
LEFT JOIN Ordens
ON Cadastro.NumCliente=Ordens.NumCliente
ORDER BY NomeCliente;
```

108. (QUADRIX / CRM-PR – 2018) O resultado será uma lista mostrada por ordem alfabética de NomeCliente e poderá ocorrer mais de uma linha contendo o mesmo registro NomeCliente.

Comentários:

Perfeito! Quando não se especifica a ordenação, considera-se que será ascendente. Além disso, não há nenhum operador distinct, logo pode haver repetição do registro NomeCliente.

Gabarito: Correto

```
SELECT Cadastro.NomeCliente, Ordens.NumOrdem
FROM Cadastro
LEFT JOIN Ordens
ON Cadastro.NumCliente=Ordens.NumCliente
ORDER BY NomeCliente;
```

109. (QUADRIX / CRM-PR – 2018) Há um erro de programação SQL na quarta linha, pois o registro NumCliente foi declarado em ambas as tabelas, Cadastro e Ordens.

Comentários:

SQL não é uma linguagem de programação, portanto não é possível existir um erro de programação. Além disso, não há nenhum problema em NumCliente estar declarado em ambas as tabelas porque ele existe em ambas as tabelas e fazem a ligação entre elas.

Gabarito: Errado

```
SELECT Cadastro.NomeCliente, Ordens.NumOrdem
FROM Cadastro
LEFT JOIN Ordens
ON Cadastro.NumCliente=Ordens.NumCliente
ORDER BY NomeCliente;
```

110. (QUADRIX / CRM-PR – 2018) Com relação aos comandos SQL apresentados acima, julgue o item a seguir.

O resultado será uma lista com todas as ocorrências de NomeCliente da tabela Cadastro e os respectivos NumOrdem da tabela Ordens onde houver coincidência do NumCliente.

Comentários:

Perfeito! Eu diria ainda que os valores de NomeCliente serão retornados ainda que não haja coincidência de NumCliente e o resultado será ordenado de forma ascendente por NomeCliente.

Gabarito: Correto

---

111. (IBADE/ Câmara de Porto Velho-RO – 2018) O comando SQL para extrair uma informação de um banco de dados é:

a) Extract.
b) Get.
c) Select.
d) Import.
e) Open.

Comentários:

O comando utilizado para extrair informações ou consultar dados é o SELECT.

Gabarito: Letra C

---

112. (FUNDEP / UFVJM-MG – 2017) Qual comando SQL é necessário para listar os itens na tabela de PRODUTOS ordenados por NOME, começando pelo final do alfabeto?

a) SELECT * FROM PRODUTOS ORDER BY NOME
b) SELECT ALL FROM PRODUTOS ORDER BY NOME
c) SELECT * FROM PRODUTOS ORDER BY NOME INVERT
d) SELECT * FROM PRODUTOS ORDER BY NOME DESC.

Comentários:

Vamos analisar o enunciado: Listar os itens (**SELECT *****) na tabela de PRODUTOS (**FROM** PRODUTOS) ordenador por NOME (**ORDER BY NOME**), começando pelo final do alfabeto (**DESC**).

Gabarito: Letra D

---

113. (AOCP / EBSERH – 2017) Usando como referência o padrão SQL-92, assinale a alternativa que exemplifica corretamente um comando de inserção de dados em uma tabela usando a linguagem SQL.

a) INSERT SALDO SET CONTA = 25 AND CLIENTE = 12
b) INSERT INTO CONTA (ID_CONTA, ID_CLIENTE, SALDO) VALUES (1, 12, 993.56)
c) INSERT CONTA WHERE SALDO < 12.01
d) CREATE REGISTER CONTA FROM VALOR = 598.01 AND ID = 58
e) UPDATE CONTA SET CLIENTE = 31 AND SALDO = 554,32

Comentários:

(a) Errado, a cláusula INSERT deve vir com o INTO – INSERT INTO; (a) Correto, a sintaxe está perfeita; (a) Errado, a cláusula INSERT deve vir com o INTO – INSERT INTO; (a) Errado, a cláusula CREATE cria estruturas de banco de dados; (a) Errado, a cláusula UPDATE atualiza valores do banco de dados;

Gabarito: LETRA B

---

114. (AOCP / EBSERH – 2017) A linguagem SQL é subdividida em subconjuntos, conforme a especificação das operações realizadas em um Banco de Dados (BD). Dentre essas subdivisões, temos a DML (Data Manipulation Linguage), que é responsável por realizar exclusões, inclusões, atualizações e consultas aos dados presentes do BD. Assinale a alternativa que exemplifica um comando de exclusão de registro.

a) DELETE SALDO SET CONTA = 25
b) ALTER CONTA SET SALDO < 12.01
c) DELETE CONTA FROM VALOR = 598.01 AND ID = 58
d) DELETE FROM CONTA WHERE CLIENTE = 31 AND SALDO = 554,32
e) DROP TABLE CONTA FROM ID = 33 AND CLIENTE = 12

Comentários:

(a) Errado. A sintaxe correta seria `DELETE FROM NOME_TABELA WHERE LISTA_CONDICOES;` (b) Errado. ALTER altera estruturas do banco de dados – não excluir dados; (c) Errado. A sintaxe correta seria: `DELETE FROM CONTA WHERE VALOR = 598.01 AND ID = 58;` (d) Correto. A sintaxe está perfeita: `DELETE FROM CONTA WHERE CLIENTE = 31 AND SALDO = 554,325;` (e) Errado. DROP deleta uma tabela e a sintaxe está incorreta.

Gabarito: Letra D

---

115. (IFB / IFB – 2017) Considerando-se a descrição sobre SQL (Structured Query Language), assinale a única alternativa onde a sequência de comandos SQL (da esquerda para a direita) está

associada corretamente à sequência de categorias {comando DML, comando DDL, comando TCL}.

a) {CREATE, UPDATE, ROOLBACK}
b) {DELETE, COMMIT, INSERT}
c) {UPDATE, ALTER, COMMIT}
d) {CREATE, REVOKE, COMMIT}
e) {INSERT, DROP, GRANT}

Comentários:

(a) {CREATE (DDL), UPDATE (DML), ROOLBACK (NÃO EXISTE)}; (b) {DELETE (DML), COMMIT (DTL), INSERT (DML)}; (c) {UPDATE (DML), ALTER (DDL), COMMIT (DTL)}; (d) {CREATE (DDL), REVOKE (DCL), COMMIT (DTL)}; (e) {INSERT (DML), DROP (DDL), GRANT (DCL)}

Gabarito: Letra C

---

116. (COSEAC / UFF – 2017) Na linguagem de manipulação de dados DML, os comandos que permitem remover linhas existentes em uma tabela e mudar os valores de dados em uma ou mais linhas da tabela existente são, respectivamente:

a) DELETE e UPDATE.
b) DROP e ALTER TABLE.
c) REVOKE e UPDATE.
d) DELETE e CREATE.
e) DROP e ALTER INDEX.

Comentários:

(a) DELETE (DML) – remove linhas; e UPDATE (DML) – muda valores; (b) DROP (DDL) e ALTER TABLE (DDL); (c) REVOKE (DCL) e UPDATE (DML); (d) DELETE (DML) e CREATE (DDL); (e) DROP (DDL) e ALTER INDEX (DDL).

Gabarito: Letra A

---

117. (FCM / IF Farroupilha-RS – 2016) O comando SELECT da linguagem Structured Query Language (SQL) permite acessar dados de um banco de dados.

id_aluno	nome	email	datacadastro
1	Wilson	wilson@ifrs.edu.br	2016-01-21
2	Jennifer	jennifer@ifrs.edu.br	2016-01-21
3	Lauro	lauro@ifrs.edu.br	2016-01-21
4	Renato	renato@ifrs.edu.br	2016-01-21

Uma consulta em linguagem SQL que exiba somente as colunas id_aluno e nome da tabela ALUNO apresenta-se abaixo em:

a) SELECT * FROM ALUNO
b) SELECT id_aluno, email FROM ALUNO
c) SELECT id_aluno, nome FROM ALUNO
d) SELECT email, datacasdastro FROM ALUNO
e) SELECT nome as nome_aluno FROM ALUNO

Comentários:

O enunciado afirma que deve-se exibir somente as colunas id_aluno e nome da tabela ALUNO. *Qual é a cláusula que seleciona colunas de uma tabela?* O resultado seria:

```
SELECT id_aluno, nome FROM ALUNO
```

Gabarito: Letra C

---

118. (IBFC / EBSERH - 2016) Com base no comando SQL abaixo assinale a alternativa que tenha a interpretação técnica correta:

DELETE FROM func WHERE tipo IN ('X','Y');

a) apaga registros de uma tabela tipo que contenha no campo func caracteres iguais a 'X' e 'Y'
b) apaga o campo tipo nos registros de uma tabela func que contenham os caracteres iguais a 'X' e 'Y'
c) apaga registros de uma tabela func que contenha no campo tipo caracteres iguais a 'X' e 'Y'
d) apaga o campo func nos registros de uma tabela tipo que contenham os caracteres iguais a 'X' e 'Y'
e) apaga qualquer registro de uma tabela func que contenha o campo tipo com caracteres iguais a 'XY'

Comentários:

O Operador IN permite especificar múltiplos valores em uma cláusula WHERE e pode ser lido simplesmente como um OU. *Dito isso, vamos traduzir essa query?*

Apague registros (**DELETE**) da tabela func (**FROM** FUNC) em que (**WHERE**) a coluna tipo contenha os caracteres 'X' ou 'Y' (`tipo IN ('X','Y')`).

Gabarito: Letra C

---

119. (IF-SE / IF-SE – 2016) Em relação aos conceitos de linguagem SQL, avalie as afirmações abaixo:

I. São exemplos de DCL (Linguagem de Controle de Dados): grant e revoke.

II. São exemplos de DML (Linguagem de Manipulação de Dados): select, insert, update e drop.

III. São exemplos de DDL (Linguagem de Definição de Dados): create user, alter table e delete table.

a) Apenas a I é verdadeira
b) Apenas a II é verdadeira.
c) Apenas a III é verdadeira.
d) As três são verdadeiras.

Comentários:

(I) Correto. GRANT e REVOKE são ambos exemplos de DCL; (II) Errado. SELECT, INSERT, UPDATE são exemplos de DML, mas DROP é DDL; (III) Errado. CREATE USER e ALTER TABLE são exemplos de DDL, mas DELETE é DML.

Gabarito: Letra A

---

120. (FUNRIO / IF-BA – 2016) Um dos mecanismos de segurança em um sistema de banco de dados é o subsistema de autorização, que permite a usuários que têm privilégios específicos concederem de forma seletiva e dinâmica esses privilégios a outros usuários e, subsequentemente, revogarem esses privilégios, se desejarem. Os comandos SQL que permitem a um usuário conceder privilégios a outros usuários e revogar privilégios concedidos a outros usuários são, respectivamente:

a) INSERT PRIVILEGES e DELETE PRIVILEGES.
b) CREATE ROLE e DROP ROLE.
c) CONCEDE e EXCLUDE.
d) GRANT e REVOKE.
e) ALLOW e DISALLOW.

Comentários:

O comando que permite a um usuário conceder privilégios é o GRANT e o comando que permite revogar privilégios conceitos a outros usuários é o REVOKE.

Gabarito: Letra D

---

121. (IBFC / EBSERH – 2016) Relacione as duas colunas quanto aos comandos SQL:

(1) Linguagem de Definição
(2) Linguagem de Manipulação

(A) INSERT
(B) CREATE
(C) DROP
(D) UPDATE

a) 1AD-2BC
b) 1BC-2AD
c) 1AB-2CD
d) 1CD-2AB
e) 1AC-2BD

Comentários:

(A) INSERT é um comando da Linguagem de Manipulação (2); (B) CREATE é um comando da Linguagem de Definição (1); (C) DROP é um comando da Linguagem de Definição (1); (D) UPDATE é um comando da Linguagem de Manipulação (2);

Gabarito: Letra B

---

122. (IBFC / EBSERH – 2016) Relacione os subconjuntos do SQL da coluna da esquerda com os seus respectivos comandos da coluna da direita:

(1) DDL
(2) DML

(A) UPDATE
(B) CREATE
(C) INSERT
(D) DROP.

a) 1A - 1C - 2B - 2D
b) 1A - 1B - 2C - 2D
c) 1C - 1D - 2A - 2B
d) 1B - 1C - 2A - 2D
e) 1B - 1D - 2A - 2C

Comentários:

UPDATE (DML); CREATE (DDL); INSERT (DML); DROP (DDL). Logo, 1B - 1D - 2A - 2C.

Gabarito: Letra E

---

123. (QUADRIX / CRO-PR – 2016) Nos bancos de dados SQL, para exibir todos os registros cadastrados em uma tabela chamada clientes utiliza-se a instrução:

a) SHOW ALL clientes;
b) SELECT * TO clientes;
c) SELECT ALL clientes;
d) SHOW * FROM clientes;
e) SELECT * FROM clientes;

Comentários:

Para exibir todos os registros dessa tabela, utiliza-se o SELECT * FROM clientes.

Gabarito: Letra E

---

124. (BIO-RIO / IF-RJ – 2015) "A linguagem SQL é do tipo declarativa e constituída das três sublinguagens a seguir:

(1) ____ - inclui os comandos SELECT, INSERT, UPDATE e DELETE;
(2) ____ - inclui os comandos CREATE, ALTER e DROP;
(3) ____ - inclui os comandos GRANT e REVOKE."

As siglas que completam corretamente as lacunas do fragmento acima são respectivamente:

a) DML, DCL e DDL.
b) DML, DDL e DCL.
c) DCL, DDL e DML.
d) DDL, DML e DCL.
e) DDL, DCL e DML.

Comentários:

(1) A sublinguagem que inclui os comandos SELECT, INSERT, UPDATE e DELETE é a DML; (2) A sublinguagem que inclui os comandos CREATE, ALTER e DROP é a DDL; (3) A sublinguagem que inclui os comandos GRANT e REVOKE é a DCL;

Gabarito: Letra B

---

125. (EXATUS / BANPARÁ – 2015) É um comando do tipo DDL (Data Definition Language) no SQL:

a) SELECT.
b) CREATE.
c) DELETE.
d) INSERT.
e) UPDATE.

Comentários:

(a) Errado, trata-se de um comando DML; (b) Correto, trata-se de um comando DDL; (c) Errado. Trata-se de um comando DML; (d) Errado. Trata-se de um comando DML; (e) Errado. Trata-se de um comando DML;

Gabarito: Letra B

---

126. (CETRO / AMAZUL – 2015) A linguagem SQL é dividida em subconjuntos de acordo com as operações que se deseja efetuar sobre um banco de dados. Um desses subconjuntos é a DDL. Assinale a alternativa que apresenta apenas comandos de DDL.

a) Select e Insert.
b) Create e Drop.
c) Update e Select.
d) Revoke e Alter.
e) Delete e Truncate.

Comentários:

(a) SELECT (**DML**) e INSERT (**DML**); (b) CREATE (**DDL**) e DROP (**DDL**); (c) UPDATE (**DML**) e SELECT (**DML**); (d) REVOKE (**DCL**) e ALTER (**DDL**); (e) DELETE (**DML**) e TRUNCATE (**DDL**);

Gabarito: Letra B

---

127. (AOCP / EBSERH – 2015) De acordo com o conceito da DDL – Linguagem de definição de dados – quais comandos são utilizados na linguagem SQL?

a) SELECT, GRANT, DENY, REVOKE.
b) INSERT, UPDATE, GRANT, DROP.
c) CREATE, ALTER, DROP.
d) ALTER, DENY, GRANT.
e) GRANT, DENY, REVOKE.

Comentários:

(a) SELECT (DML), GRANT (DCL), DENY (DCL), REVOKE (DCL); (b) INSERT (DML), UPDATE (DML), GRANT (DCL), DROP (DDL); (c) CREATE (DDL), ALTER (DDL), DROP (DDL); (d) ALTER (DDL), DENY (DCL), GRANT (DCL); (e) GRANT (DCL), DENY (DCL), REVOKE (DCL). DENY é um comando para negar uma permissão.

Gabarito: Letra C

---

128. (NUCEPE / SEDUC-PI – 2015) O SQL é uma linguagem de manipulação de banco de dados. Assinale a alternativa CORRETA sobre os subconjuntos do SQL:

a) DML – Linguagem de Transação de Dados.
b) DDL – Linguagem de Controle de Dados.
c) DCL – Linguagem de Definição de Dados.
d) DTL – Linguagem de Manipulação de Dados.
e) DQL – Linguagem de Consulta de Dados.

Comentários:

(a) DML – Linguagem de ~~Transação~~ Manipulação de Dados; (b) DDL – Linguagem de ~~Controle~~ Definição de Dados; (c) DCL – Linguagem de ~~Definição~~ Controle de Dados; (d) DTL – Linguagem de ~~Manipulação~~ Transação de Dados; (e) DQL – Linguagem de Consulta de Dados. Lembrando que algumas questões consideram SELECT como DQL e, não, DML.

Gabarito: Letra E

---

129. (CETAP - CPCM – 2015) Na linguagem SQL, a sentença que permite atualizar dois campos A e B em uma tabela X e:

a) UPDATE X WHERE A=2 AND B-3
b) UPDATE X SET A=2, B=3 WHERE C=4.
c) UPDATE X SET A=2 WHERE B=3
d) UPDATE X SET B=2 WHERE A=3

e) UPDATE A=2 AND B=3 WHERE X

Comentários:

(a) Errado, falta a cláusula SET; (b) Correto, sintaxe perfeita; (c) Errado, não atualiza o campo B; (d) Errado, não atualiza o campo A; (e) Errado, falta a cláusula SET.

Gabarito: Letra B

---

130. (COMPERVE / UFRN – 2015) O comando na linguagem SQL que permite a remoção de todas as linhas de uma tabela X é:

a) REMOVE * FROM X
b) REMOVE ALL FROM X
c) DELETE FROM X
d) DELETE * FROM X

Comentários:

(a) Errado, REMOVE não é um comando válido; (b) Errado, REMOVE não é um comando válido; (c) Correto, eliminará todas as linhas da Tabela X; (d) Errado, não é necessário utilizar o asterisco.

Gabarito: Letra C

---

131. (FAFIPA / UFFS – 2014) Os comandos SQL podem ser divididos basicamente em três categorias: DML, DDL e DCL. Assinale a alternativa que apresenta um comando DCL:

a) SELECT
b) INSERT
c) ALTER
d) GRANT
e) DROP

Comentários:

(a) SELECT (**DML**); (b) INSERT (**DML**); (c) ALTER (**DDL**); (d) GRANT (**DCL**); (E) DROP (**DDL**);

Gabarito: Letra D

---

132. (FUNCAB / PRODAM-AM – 2014) Se trata de um comando considerado parte da linguagem de definição de dados (DDL) e um comando considerado parte da linguagem de manipulação de dados (DML) do SQL, respectivamente:

a) CREATE e DROP
b) DROP e UPDATE
c) DELETE e CREATE
d) INSERTe UPDATE
e) SELECT e TRUNCATE

Comentários:

(a) CREATE (DDL) e DROP (DDL); (b) DROP (DDL) e UPDATE (DML); (c) DELETE (DML) e CREATE (DDL); (d) INSERT (DML) e UPDATE (DML); (e) SELECT (DML) e TRUNCATE (DDL)

Gabarito: Letra B

---

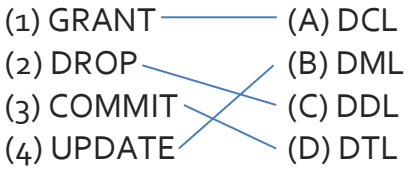
133. (IBFC / TRE-AM – 2014) Relacione as duas colunas, quanto aos comandos SQL e os respectivos subconjuntos da linguagem SQL:

(1) GRANT	(A) DCL
(2) DROP	(B) DML
(3) COMMIT	(C) DDL
(4) UPDATE	(D) DTL

Assinale a alternativa correta:

a) 1A-2C-3D-4B
b) 1A-2D-3C-4B
c) 1B-2C- 3D- 4A
d) 1C-2A-3D-4B

Comentários:



Gabarito: Letra A

---

134. (UFBA / UNILAB – 2014) Os comandos commit e rollback são utilizados, respectivamente, para confirmar e desfazer instruções do tipo DML e DDL em um banco de dados.

Comentários:

Os comandos COMMIT e ROLLBACK são utilizados, respectivamente, para confirmar e desfazer instruções apenas do tipo DML.

Gabarito: Errado

---

135. (ESPP / MPE-PR – 2013) Ao se utilizar a linguagem SQL, aceitamos o conjunto de relações como dado que deve ser especificado no sistema gerenciador de banco de dados por meio de uma linguagem. A linguagem_____permite não só a especificação de um conjunto de relações, como também informações acerca de cada uma das relações, incluindo: o esquema de cada relação, o domínio dos valores associados a cada atributo, as regras de integridade, o conjunto de índices para manutenção de cada relação, informações sobre segurança e autoridade sobre cada relação e a estrutura de armazenamento físico de cada relação. Assinale a alternativa que completa corretamente a lacuna.

a) DDL
b) DSL
c) DTL
d) DQL
e) DCL

Comentários:

A questão trata de uma linguagem que permite a especificação de um conjunto de relações (tabelas), assim como informações acerca de cada uma (esquema, domínio, regras de integridade, índices, segurança, etc). Logo, só pode estar tratando sobre a Linguagem DDL.

Gabarito: Letra A

---

136. (CCV-UFC / UFC – 2013) Assinale a alternativa em que são apresentados dois comandos da linguagem de definição de dados (DDL).

a) CREATE e ALTER
b) CREATE e GRANT
c) INSERT e DELETE
d) GRANT e REVOKE
e) GRANT e ROLLBACK

Comentários:

(a) CREATE (**DDL**) e ALTER (**DDL**); (b) CREATE (**DDL**) e GRANT (**DCL**); (c) INSERT (**DML**) e DELETE (**DML**); (d) GRANT (**DCL**) e REVOKE (**DCL**); (e) GRANT (**DCL**) e ROLLBACK (**DTL**);

Gabarito: Letra A

---

137. (CEPERJ / CEPERJ – 2013) A linguagem SQL para bancos de dados, é constituída das sublinguagens "Data Manipulation Language – DML", "Data Definition Language – DDL" e "Data Control Language – DCL". Fazem parte da DCL os seguintes comandos:

a) SELECT e DROP
b) DROP e GRANT
c) GRANT e REVOKE
d) REVOKE e DELETE
e) DELETE e SELECT

Comentários:

(a) SELECT (DML) e DROP (DDL); (b) DROP (DDL) e GRANT (DCL); (c) GRANT (DCL) e REVOKE (DCL); (d) REVOKE (DCL) e DELETE (DML); (e) DELETE (DML) e SELECT (DML)

Gabarito: Letra C

---

138. (IBFC / EBSERH – 2013) Relacione os comandos típicos utilizados em cada Linguagem conforme tabela abaixo:

(A) UPDATE
(B) GRANT
(C) CREATE

(D) Linguagem de definição de dados (DDL)
(E) Linguagem de manipulação de dados (DML)
(F) Linguagem de controle de dados (DCL)

a) AD - BE - CF
b) AD - BF - CE
c) AE - BD - CF
d) AE - BF - CD

Comentários:

(A) UPDATE – (E) Linguagem de manipulação de dados (DML); (B) GRANT – (F) Linguagem de controle de dados (DCL); (C) CREATE – (D) Linguagem de definição de dados (DDL).

Gabarito: Letra D

---

139. (IBFC / PC-RJ – 2013) A "linguagem" SQL (Structured Query Language) revolucionou a forma em que os programadores extraem informações de um Banco de Dados. Um perito com sólidos

conhecimentos de SQL pode obter em frações de segundos uma grande massa de dados e posterior processamento visando a extração de algum dado ou sequência de dados, como por exemplo possíveis suspeitos em uma amostra populacional de uma cidade. São quatro os principais comandos do DML (Data Manipulation Language) do SQL. Identifique a alternativa que NÃO é um desses comandos:

a) INSERT
b) SELECT
c) UPDATE
d) DELETE
e) CREATE.

Comentários:

Todos são comandos DML – exceto CREATE.

Gabarito: Letra E

---

140. (QUADRIX / CRQ-SP – 2013) Assinale a alternativa com o uso correto do comando Insert do SQL.

a)
INSERT (CODIGO, NOME, SALARIO, SECAO)
VALUES(1, "LUCKY LUCIANO", 120, 1)
INTO EMPREGADOS

b)
INSERT INTO EMPREGADOS
VALUES(1, "LUCKY LUCIANO", 120, 1)
(CODIGO, NOME, SALARIO, SECAO)

c)
INSERT INTO EMPREGADOS(CODIGO, NOME, SALARIO, SECAO)

d)
INSERT INTO EMPREGADOS(CODIGO, NOME, SALARIO, SECAO)
VALUES(1, "LUCKY LUCIANO", 120, 1)

e)
INSERT VALUES(1, "LUCKY LUCIANO", 120, 1)
INTO EMPREGADOS

Comentários:
(a) Errado, falta a cláusula INTO; (b) Errado, o segundo parêntese deveria vir após EMPREGADOS; (c) Errado, faltam os valores a serem inseridos; (d) Correto, sintaxe perfeita; (e) Errado, a sintaxe é INSERT INTO.

Gabarito: Letra D

---

141. (ESPP / COBRA – 2013) Em SQL se um empregado for promovido a gerente devemos alterar o cargo (CARGO) do empregado número (NUM_EMP) '48729' na tabela EMPREGADOS para o valor GERENTE, usando a seguinte query:

a) UPDATE GERENTE WHERE NUM_EMP = '48729';
b) UPDATE CARGO = GERENTE WHERE NUM_EMP = '48729';
c) UPDATE EMPREGADOS SET CARGO = GERENTE WHERE NUM_EMP = '48729';
d) UPDATE EMPREGADOS SET NUM_EMP = '48729';

Comentários:

(a) Errado, deve-se atualizar a tabela EMPREGADOS e, não, GERENTE; (b) Errado, deve-se atualizar a tabela EMPREGADOS e, não, CARGO; (c) Correto, sintaxe perfeita – a questão atualiza (UPDATE) a tabela EMPREGADOS, setando a coluna CARGO para o valor GERENTE onde o NUM_EMP é '48729'; (d) Errado, essa query atualiza o valor de NUM_EMP para '48729' – o que não foi especificado pela questão.

Gabarito: Letra C

---

142. (ESAF / Ministério da Fazenda – 2013) As três cláusulas de uma consulta SQL são:

a) start, from, to.
b) select, from, where.
c) select, up, what.
d) start, from, who.
e) select, initial, final.

Comentários:

As três cláusulas de uma consulta são SELECT, FROM e WHERE.

Gabarito: Letra B

---

143. (CESGRANRIO / EPE – 2012) A DDL (Data Definition Language) ou linguagem de definição de dados é um conjunto de comandos SQL responsável pela definição das estruturas de dados em um SGBD. Qual comando faz parte da DDL?

a) Select
b) Update
c) Create
d) Delete
e) Insert

Comentários:

(a) SELECT é DML; (b) UPDATE é DML; (c) CREATE é DDL; (d) DELETE é DML; (e) INSERT é DML;

Gabarito: Letra C

---

144. (CONSULPLAN / TSE – 2012) Ao contrário das linguagens tradicionais, que são procedimentais, SQL é uma linguagem declarativa, que integra três sublinguagens: Data Manipulation Language (DML), Data Definition Language (DDL) e Data Control Language (DCL). Um comando DML e outro DDL são, respectivamente,

a) Drop e Grant.
b) Grant e Delete.
c) Delete e Update.
d) Update e Drop.

Comentários:

(a) DROP é DDL; GRANT é DCL; (b) GRANT é DCL; DELETE é DML; (c) DELETE é DML; UPDATE é DML; (d) UPDATE é DML; DROP é DDL.

Gabarito: Letra D

---

145. (FUMARC / TJ-MG – 2012) A linguagem SQL possui comandos de definição de dados (DDL - Data Defnition Language), dos quais faz parte o seguinte comando:

a) SELECT
b) DELETE
c) ALTER
d) UPDATE

Comentários:

(a) SELECT (**DML**); (b) DELETE (**DML**); (c) ALTER (**DDL**); (d) UPDATE (**DML**)

Gabarito: Letra C

---

146. (CEPERJ / PROCON-RJ – 2012) SQL representa uma linguagem declarativa, não procedural, que permite interação com bancos de dados, sendo constituída de três sublinguagens, a Data Manipulation Language (DML), a Data Definition Language (DDL) e a Data Control Language (DCL). Como comandos DCL, um permite conceder determinado privilégio a um usuário e outro permite retirar o privilégio concedido. Esses comandos são, respectivamente:

a) GET e PUT
b) CREATE e DROP
c) SELECT e DELETE
d) GRANT e REVOKE
e) INSERT e REMOVE

Comentários:

(a) GET (NÃO EXISTE) e PUT (NÃO EXISTE); (b) CREATE (DDL) e DROP (DDL); (c) SELECT (DML) e DELETE (DML); (d) GRANT (DCL) e REVOKE (DCL); (e) INSERT (DML) e REMOVE (NÃO EXISTE).

Gabarito: Letra D

---

147. (IESES / CRF-SC – 2012) Relacione a primeira coluna com a segunda e em seguida identifique a alternativa que apresenta a ordem correta dos números de cima para baixo:

(1) – DDL ( ) - É um subconjunto de comandos SQL que serve para a definição das estruturas de dados de um banco de dados, como por exemplo, criar tabelas, índices, views, etc.

(2) – DML ( ) - É um subconjunto de comandos SQL que permite a DBAs controlar o acesso aos dados de um banco de dados.

(3) – DCL ( ) - É um subconjunto de comandos SQL que serve para acesso, inclusão, alteração e exclusão dos dados de um banco de dados.

a) 1 – 3 – 2
b) 3 – 1 – 2
c) 1 – 2 – 3
d) 2 – 3 – 1

Comentários:

É um subconjunto de comandos SQL que serve para a definição das estruturas de dados de um banco de dados, como por exemplo, criar tabelas, índices, views, etc – (1) DDL;

É um subconjunto de comandos SQL que permite a DBAs controlar o acesso aos dados de um banco de dados – (3) DCL;

É um subconjunto de comandos SQL que serve para acesso, inclusão, alteração e exclusão dos dados de um banco de dados – (2) DML;

Gabarito: Letra A

---

148. (FUNCAB / MPE-RO – 2012) A cláusula FROM de comando SELECT é utilizada para especificar:

a) a ordem de seleção das linhas da tabela.
b) a condição de seleção das linhas da tabela.
c) os campos cujas informações deverão ser selecionadas.
d) as tabelas das quais as informações serão selecionadas.
e) o agrupamento das informações selecionadas.

Comentários:

Essa cláusula é utilizada para especificar as tabelas das quais as informações serão selecionadas.

Gabarito: Letra D

---

149. (FMP / TCE-RS – 2011) Qual comando SQL é utilizado para obter um conjunto de dados em uma tabela em um banco de dados?

a) INSERT
b) UPDATE
c) JOIN
d) SELECT
e) GET

Comentários:

O comando utilizado para obter um conjunto de dados em uma tabela é o SELECT.

Gabarito: Letra D

---

150. (Instituto Ânima / Companhia Águas de Joinville – 2010) Para manipular dados em um banco de dados, usamos uma linguagem de consulta estruturada. A SQL (Structured Query Language) é a linguagem usada pela maioria dos bancos de dados. Esta é composta de três outras linguagens, quais são elas?

a) DML, DDL e DLL.
b) DML, DDL e DCL.
c) MLL, DLL e CLL.
d) SQL, LQL e CQL.
e) DDL, DCL e DGL

Comentários:

SQL é composta por DML, DDL e DCL (a questão poderia ter mencionado também a DTL).

Gabarito: Letra B

---

151. (IF-RJ / IF-RJ– 2010) A linguagem SQL possui sublinguagens. Dentre elas, destacamos a DDL, de definição de dados; a DML, de manipulação de dados e a DCL, de controle de dados.

Marque a alternativa que possui um comando de cada sublinguagem.

a) Create, Drop, Select
b) Create, Insert, Alter
c) Select, Insert, Update
d) Insert, Revoke, Update
e) Create, Delete, Grant

Comentários:

(a) Create (DDL), Drop (DDL), Select (DML); (b) Create (DDL), Insert, Alter (DDL); (c) Select (DML), Insert (DML), Update (DML); (d) Insert (DML), Revoke (DCL), Update (DML); (e) Create (DDL), Delete (DML), Grant (DCL).

Gabarito: Letra E

---

152. (NCE-UFRJ / UFRJ – 2009) Um sistema que suporta o processamento de transações, garante que se a transação executar algumas atualizações e então ocorrer uma falha antes que esta alcance seu término normal, aquelas atualizações não serão feitas. Consequentemente a atualização é executada na sua totalidade ou cancelada. Os comandos em SQL, usados para desfazer uma transação mal sucedida e confirmar uma bem sucedida, são, respectivamente:

a) COMMIT e ROLLBACK;
b) GRANT e REVOKE;
c) ROLLBACK e COMMIT;
d) REVOKE e GRANT;
e) ROLLBACK e GRANT.

Comentários:

O comando para desfazer uma transação mal sucedida é o ROLLBACK e o comando para confirmar uma transação bem sucedida é o COMMIT.

Gabarito: Letra C

---

153. (CESGRANRIO / Casa da Moeda – 2009) Um administrador de dados de uma empresa deve, excepcionalmente, atualizar o endereço de um funcionário registrado em uma tabela do banco de dados, que não guarda histórico e registra somente o endereço atual em uma única linha. Para a atualização dos dados, que comando SQL deverá ser utilizado?

a) SELECT.
b) CHANGE.
c) INSERT.
d) UPDATE.
e) DELETE.

Comentários:

O comando utilizado para atualizar dados é o UPDATE.

Gabarito: Letra D

---

154. (ESAF / UFSJ – 2005) Com relação ao uso da SQL na manipulação de dados, caso se queira eliminar linhas repetidas do conjunto resultado, deve-se utilizar a palavra-chave DISTINCT, da seguinte forma:

a) SELECT DISTINCT {colunas} FROM {tabelas}.
b) DISTINCT SELECT {colunas} FROM {tabelas}.
c) SELECT {colunas} FROM {tabelas} DISTINCT.
d) SELECT FROM {tabelas} DISTINCT {colunas}.

Comentários:

O comando para eliminar linhas repetidas é o SELECT DISTINCT {colunas} FROM {tabelas}.

Gabarito: Letra A

LISTA DE QUESTÕES – CESPE

1. **(CESPE / SEFAZ-SE – 2022)** A respeito do código SQL (Structured Query Language) anteriormente apresentado, assinale a opção correta.

```
select C.CPF as CPF, C.NOME as NOME
from CONTRIBUINTE as C, PARCELAMENTO as P
where C.CPF=P.CPF
and P.TIPO='IPVA'
and P.DATAADESAO between '01/01/2021' and
'31/12/2021'
and P.STATUS='ADIMPLENTE'
```

a) Há um erro de sintaxe em `and P.DATAADESAO between '01/01/2021' and '31/12/2021'`, pois não é permitida a utilização do operador `and` mais de uma vez na mesma linha.

b) Há uma junção (JOIN) nesse código, a qual é especificada no trecho `from CONTRIBUINTE as C, PARCELAMENTO as P.`

c) O objetivo do código é mostrar o CPF e o nome de todos os contribuintes que não aderiram ao programa de parcelamento do IPVA no ano de `2021`.

d) A palavra reservada `between` foi inserida no código equivocadamente, pois somente deveria ser usada nos comandos de `update` e `delete`.

e) A finalidade do código é mostrar o CPF e o nome de todos os contribuintes que aderiram ao parcelamento do IPVA no ano de `2021` e que estão com o seu parcelamento em dia.

2. **(CESPE / PETROBRAS – 2022)** Por meio do comando SQL a seguir, é possível recuperar o nome dos pesquisadores responsáveis por projetos, seguido pelo nome de seu orientador, mas apenas os projetos orientados por **`Pedro`**.

```
select responsavel.nome nomeresponsavel,
orientador.nome    nomeorientador,
tituloProjeto
from Pesquisador responsavel, Pesquisador orientador, Projeto
where orientador.nome = 'Pedro'
andcodPesquisadorResponsavel = codPesquisador
and codPesquisadorOrientador = codPesquisador;
```

3. **(CESPE / Petrobrás - 2022)** Duas expressões SQL são equivalentes se e somente se elas tiverem os mesmos comandos em suas respectivas sequências.

4. **(CESPE / Petrobrás - 2022)** O comando truncate PESSOA; permite excluir todos os registros da tabela de nome PESSOA.

5. **(CESPE / Petrobrás - 2022)** A expressão SQL a seguir está sintaticamente correta e permite inserir dois alunos de nomes Pedro e Maria na tabela alunos.

   INSERT VALUES ('Pedro', 'Maria') INTO alunos;

6. **(CESPE / TJ-RJ - 2021)** Processo (codprocesso, autor, reu, dataultimamovimentacao, assunto, codjuiz) Juiz (codjuiz, nome).
   Considerando as tabelas anteriores, de um banco de dados relacional, assinale a opção cuja consulta em SQL mostra os nomes dos juízes para os quais não há processos distribuídos (relacionados).

   a) SELECT J.nome
   FROM Juiz AS J
   WHERE J.nome NOT IN (SELECT P.codjuiz
   FROM Processo AS P);

   b) SELECT J.nome
   FROM Juiz AS J, Processo AS P
   WHERE J.codjuiz inner join P.codjuiz;

   c) SELECT J.nome
   FROM Juiz AS J
   WHERE J.codjuiz NOT IN (SELECT P.codjuiz
   FROM Processo AS P);

   d) SELECT J.nome
   FROM Juiz AS J
   WHERE J.codjuiz LIKE (SELECT P.codjuiz
   FROM Processo AS P);

   e) SELECT J.nome
   FROM Juiz AS J, Processo AS P
   WHERE J.nome NOT EXISTS (P.codjuiz);

7. **(CESPE / DPE-RO – 2021)**

```
create table aluno (
    id integer not null primary key,
    nome varchar,
    datanascimento date
);
create table cidade (
    ibge bigint not null primary key,
    município varchar
```

```
);

create table alunocidade (
    cidade bigint,
    aluno integer,
   tipo varchar,
    constraint fkcidade foreign key (cidade)
references cidade,
    constraint fkaluno foreign key (aluno)
references aluno,
    constraint pkcidade primary key
(cidade,aluno,tipo)
);
```

Para a expressão SQL anterior, a cardinalidade entre as entidades aluno e cidade é:

a) zero-para-muitos.
b) muitos-para-muitos.
c) um-para-um.
d) muitos-para-um.
e) um-para-muitos.

8. **(CESPE / DPE-RO – 2021)** Assinale a opção que apresenta o comando SQL usado para excluir todos os registros de uma tabela de nome aluno, mantendo-se a estrutura da tabela:

a) delete aluno
b) erase aluno
c) erase from aluno
d) delete from aluno
e) drop from aluno

9. **(CESPE / APEX-BRASIL – 2021)** *create database pessoa*;

O comando SQL apresentado anteriormente criará:

a) um banco de dados denominado pessoa;
b) uma tabela denominada pessoa;
c) um tipo de dados denominado pessoa;
d) um esquema denominado pessoa;

10. **(CESPE / Polícia Federal – 2021)** Na linguagem SQL (*structured query language*), DTL (*data transaction language*) são comandos responsáveis por gerenciar diferentes transações ocorridas dentro de um banco de dados.

11. **(CESPE / TJ-AM – 2019)** Em SQL, o comando RIGHT OUTER JOIN exibe a união entre duas tabelas, apresentando as linhas da segunda tabela que também existem na primeira tabela, descartando-se as demais situações.

12. **(CESPE / TRE/MT – 2015)** Assinale a opção que apresenta a sintaxe correta para se obter, empregando-se a cláusula select em um banco de dados, uma lista com nomes não repetidos dos cargos de uma empresa.

a) select * from cargos
b) select nome from cargos
c) select distinct nome from cargos
d) select codigo, cargo fro cargos where codigo m>1
e) select nome from cargos order by nome asc

13. **(CESPE / MEC – 2015)** A DML utiliza o comando CREATE para inserir um novo registro na tabela de dados.

14. **(CESPE / STM – 2011)** Os comandos do grupo DDL (Data Definition Language) do SQL permitem gerar os dados das tabelas que formam um banco de dados.

15. **(CESPE / TRE-BA - 2010)** A instrução GRANT que altera as permissões em objetos-esquema, é uma instrução de DDL (data definition language).

16. **(CESPE / STJ – 2008)** O comando CREATE INDEX, usado para criar um parâmetro relacionado com uma tabela para buscar dados mais rapidamente, é considerado como DDL.

17. **(CESPE / STJ – 2008)** O comando DELETE, capaz de excluir dados de um banco de dados, é considerado como DDL.

LISTA DE QUESTÕES – FCC

18.(FCC / SEFAZ-PE – 2022)

```
CREATE TABLE nfe (
Numero_NFe VARCHAR(9) NOT NULL,
Modelo_NFe VARCHAR(2) NULL,
Serie_NFe VARCHAR(3) NULL,
codigo_UF VARCHAR(2) NULL,
ano_Emissao VARCHAR(2) NULL,
mes_Emissao VARCHAR(2) NULL,
CNPJ_Emitente VARCHAR(14) NULL,
Codigo_Chave VARCHAR(8) NULL,
Digito_Chave VARCHAR(1) NULL,
PRIMARY KEY (Numero_NFe));
```

Para inserir um registro com valores de teste na tabela nfe, utiliza-se a instrução SQL:

a) INSERT INTO nfe VALUES('123456789','55','1','RJ','22','10','15145076000106','87654321','6');
b) APPEND TO nfe VALUES('123456789','55','1','RJ','22','10','15145076000106','87654321','6');
c) INSERT INTO nfe('123456789','55','1','RJ','22','10','15145076000106','87654321','6');
d) INSERT TO nfe VALUES('123456789','55','1','RJ','22','10','15145076000106','87654321','6');
e) ADD INTO nfe('123456789','55','1','RJ','22','10','15145076000106','87654321','6');

19.(FCC / SEFAZ-PE – 2022)

```
CREATE TABLE nfe (
Numero_NFe VARCHAR(9) NOT NULL,
Modelo_NFe VARCHAR(2) NULL,
Serie_NFe VARCHAR(3) NULL,
codigo_UF VARCHAR(2) NULL,
ano_Emissao VARCHAR(2) NULL,
mes_Emissao VARCHAR(2) NULL,
CNPJ_Emitente VARCHAR(14) NULL,
Codigo_Chave VARCHAR(8) NULL,
Digito_Chave VARCHAR(1) NULL,
PRIMARY KEY (Numero_NFe));
```

Para selecionar na tabela nfe todas as notas fiscais eletrônicas cujo conteúdo do campo codigo_UF inicie pela letra S e tenha qualquer letra na sequência, utiliza-se a instrução SQL:

SELECT * FROM nfe WHERE

a) codigo_UF CONTAINS 'S*';
b) codigo_UF LIKE 'S%';
c) codigo_UF LIKE 'S*';
d) codigo_UF = 'S%';
e) codigo_UF LIKE '%S';

20. (FCC / SEFAZ-AP – 2022) Em um banco de dados SQL aberto em condições ideais, considere a existência de uma tabela chamada clientes com os campos clienteID e nomeCliente e de uma tabela chamada pedidos com os campos pedidoID, clienteID e dataPedido.

A coluna clienteID na tabela pedidos se refere ao clienteID na tabela clientes, ou seja, a relação entre elas ocorre por meio da coluna clienteID. Para selecionar pedidoID, nomeCliente e dataPedido apenas de registros que possuam valores correspondentes nas duas tabelas utiliza-se a instrução SQL:

a) SELECT pedidos.pedidoID, clientes.nomeCliente, pedidos.dataPedido FROM pedidos INNER JOIN clientes ON pedidos.clienteID=clientes.clienteID;

b) SELECT pedidos.pedidoID, clientes.nomeCliente, pedidos.dataPedido FROM pedidos AND SELECT clientes WHERE pedidos.clienteID=clientes.clienteID;

c) SELECT pedidos.pedidoID, clientes.nomeCliente, pedidos.dataPedido FROM pedidos UNION clientes ON pedidos.clienteID=clientes.clienteID;

d) SELECT pedidos.pedidoID, clientes.nomeCliente, pedidos.dataPedido FROM pedidos FULL INTER JOIN clientes ON pedidos.clienteID=clientes.clienteID;

e) SELECT pedidos.pedidoID, clientes.nomeCliente, pedidos.dataPedido FROM pedidos INNER JOIN FROM clientes WHERE pedidos.clienteID=clientes.clienteID;

21. (FCC / SEFAZ-AP – 2022) Usando a linguagem SQL, um fiscal escreveu corretamente, em uma consulta, a expressão WHERE Nome-Contribuinte LIKE 'p%o'. Com isso ele obteve nomes de contribuintes como, por exemplo,

a) Paula ou Patricio.
b) Paulo ou Pedro.
c) Augusto e Jordão.
d) Paulo ou Paula.
e) Álvaro ou Augusto.

22. (FCC / TJ-MA – 2019) Considere a existência de um banco de dados aberto e em condições ideais, no qual a tabela Processo possui diversos campos, sendo um deles, o campo

numero_processo, do tipo cadeia de caracteres (varchar). Para exibir todos os processos cujo número inicie por qualquer caractere seguido de "009.51.01.87348-6", utiliza-se a instrução SQL:

a) SELECT *.* FROM Processo WHERE numero_processo LIKE '_009.51.01.87348-6';
b) SELECT * FROM Processo WHERE numero_processo='#009.51.01.87348-6';
c) SELECT * FROM Processo WHERE numero_processo EQUALS '%009.51.01.87348-6';
d) SELECT * FROM Processo WHERE numero_processo LIKE '_009.51.01.87348-6';
e) SELECT *.* FROM Processo WHERE numero_processo LIKE '%009.51.01.87348-6';

23. **(FCC / TRT4 – 2019)** Uma Analista digitou o comando TRUNCATE TABLE processos; em um banco de dados SQL aberto em condições ideais para:

a) excluir os dados da tabela, mas não a tabela em si.
b) excluir a estrutura da tabela e os dados nela contidos.
c) juntar a tabela aberta na memória com a tabela processos.
d) bloquear a tabela processos para uso exclusivo de seu usuário.
e) editar a estrutura da tabela em modo gráfico.

24. **(FCC / TRT4 – 2019)** Em uma tabela chamada itemfatura há diversos registros em que constam o mesmo valor no campo idfatura. Para mostrar a quantidade de valores de idfatura diferentes que estão cadastrados na tabela, utiliza-se o comando:

a) SELECT DISTINCT (idfatura) FROM itemfatura;
b) SELECT * FROM itemfatura WHERE idfatura IS DIFFERENT;
c) SELECT SUM(DISTINCT idfatura) FROM itemfatura;
d) SELECT COUNT(DISTINCT idfatura) FROM itemfatura;
e) SELECT COUNT(DIFFERENT idfatura) FROM itemfatura;

25. **(FCC / TRT4 – 2019)** Um Técnico Judiciário necessitou usar a linguagem padrão SQL para recuperar, de uma tabela do banco de dados relacional denominada tabela1,

I. o menor valor em uma determinada coluna denominada coluna1.
II. um padrão de valores denominado padrão_desejado em uma outra coluna denominada coluna2.

Para tanto, em duas operações distintas, ele utilizou, respectivamente, as expressões

1. SELECT I .....
   FROM tabela1
   WHERE condição;
2. SELECT coluna2
   FROM tabela1
   WHERE II .....;

I e II são, correta e respectivamente,

a) MINVALUE(coluna1) e padrão_desejado %LIKE coluna2
b) THIN (coluna1) e coluna2 = padrão_desejado
c) SMALL(coluna1) e padrão_desejado = coluna2
d) MIN(coluna1) e coluna2 LIKE padrão_desejado
e) GETSMLL(coluna1) e padrão_desejado % coluna2

26. (FCC / SEFAZ-BA – 2019) Em uma tabela chamada Contribuinte de um banco de dados padrão SQL aberto e em condições ideais 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;

27. (FCC / SEFAZ-BA – 2019) Para buscar na tabela Contribuintes todos os nomes de contribuintes (campo nomeContribuinte) que terminam com a letra s, um Auditor utilizou corretamente a instrução SQL

a) SEARCH * FROM Contribuintes WHERE nomeContribuinte LIKE '%s';
b) SELECT nomeContribuinte FROM Contribuintes WHERE nomeContribuinte LIKE '*s';
c) SELECT * FROM Contribuintes WHERE nomeContribuinte FINISHED BY '%s';
d) SEARCH nomeContribuinte FROM Contribuintes WHERE nomeContribuinte FINISHED BY 's';
e) SELECT * FROM Contribuintes WHERE nomeContribuinte LIKE '%s';

28. (FCC / AFAP – 2019) Fernando está usando a linguagem SQL (ANSI) e pretende fazer uma atualização nos dados Nome_Cli e End_Cli do cliente cujo Cod_Cli é Clio1, na tabela Cliente. Nome_Cli passará a ser Ariana e End_Cli passará a ser Rua ABC. O código SQL correto que Fernando escreveu foi:

..I.. Cliente
..II.. Nome_Cli = 'Ariana', End _Cli = 'Rua ABC'
..III.. Cod_Cli = 'Clio1';

Para que o código esteja correto, as lacunas I, II e III devem ser preenchidas, respectivamente, por

a) SET - WHERE - UPDATE
b) UPDATE - SET - WHERE
c) UPDATE - WHERE - SET
d) WHERE - SET - UPDATE
e) SET - UPDATE - WHERE

29. **(FCC / DPE-AM – 2018)** Para apagar todos os registros da tabela copia_eleitores utiliza-se a instrução SQL:

a) DELETE FROM copia_eleitores; ou TRUNCATE * FROM copia_eleitores;
b) DELETE RECORDS copia_eleitores; ou DROP RECORDS FROM copia_eleitores;
c) DELETE * FROM copia_eleitores; ou DELETE RECORDS copia_eleitores;
d) DELETE FROM copia_eleitores; ou DELETE * FROM copia_eleitores;
e) DELETE RECORDS copia_eleitores; ou TRUNCATE TABLE copia_eleitores;

30. **(FCC / TST – 2017)** Um Programador:

I. criou uma tabela e uma view em um banco de dados relacional.
II. alterou a estrutura da tabela.
III. incluiu registros na tabela.

a) DDL – DML – DDL.
b) DML – DML – DDL.
c) DML – DDL – DDL.
d) DDL – DML – DML.
e) DDL – DDL – DML.

31. **(FCC / MANAUSPREV – 2015)** A linguagem SQL é dividida em subconjuntos de acordo com as operações que se deseja efetuar sobre um banco de dados. Considere os grupos de comandos:

I. CREATE, ALTER, DROP.
II. GRANT, REVOKE.
III. DELETE, UPDATE, INSERT.

Os comandos listados em:

a) I correspondem à Data Control Language - DCL e II à Data Definition Language - DDL.
b) I correspondem à Data Manipulation Language - DML e III à Data Control Language - DCL.
c) II correspondem à Data Manipulation Language - DML e III à Data Control Language - DCL.
d) I correspondem à Data Definition Language - DDL e III à Data Manipulation Language - DML.
e) II correspondem à Data Control Language - DCL e III à Data Definition Language - DDL.

32. **(FCC / TRT-19 - 2011)** Considere as linguagens inseridas no contexto SQL: DML, DDL, DTL, DCL e DQL. Desta forma, Grant, Commit, Update, Delete e Alter, correspondem, respectivamente, a:

a) DCL, DCL, DDL, DDL e DDL.
b) DTL, DCL, DML, DDL e DQL.
c) DCL, DTL, DML, DML e DDL.
d) DML, DTL, DCL, DQL e DML.
e) DQL, DDL, DML, DML e DDL.

33. **(FCC / TCM-PA – 2010)** Na linguagem SQL são, respectivamente, expressões DDL e DML:

a) UPDATE e DROP INDEX.
b) UPDATE e INSERT INTO.
c) ALTER DATABASE e UPDATE.
d) ALTER TABLE e CREATE INDEX.
e) INSERT INTO e DELETE.

34. **(FCC / METRÔ-SP – 2010)** O SQL (Structured Query Language) é uma linguagem de pesquisa declarativa para banco de dados relacional. A DDL permite ao usuário definir tabelas novas e elementos associados. A sigla DDL significa:

- a) Data Definition List.
b) Data Default Language.
c) Data Definition Language.
d) Data Default List.
e) Definition Data Language.

35. **(FCC / TRE-RS – 2010)** São declarações SQL, associadas, respectivamente, a três declarações DML (Data Manipulation Language) e a três declarações DDL (Data Definition Language):

a) UPDATE, INSERT, SELECT, DROP INDEX, ALTER TABLE, CREATE VIEW.
b) DROP VIEW, DROP INDEX, UPDATE, DELETE, ALTER TABLE, INSERT.
c) CREATE, UPDATE, DELETE, DROP VIEW, INSERT, SELECT.
d) CREATE INDEX, CREATE VIEW, DROP VIEW, UPDATE, DELETE, INSERT.
e) INSERT, DELETE, DROP INDEX, ALTER INDEX, UPDATE, SELECT.

36. **(FCC / TRF-4ª Região – 2010)** DROP é um comando utilizado para apagar um objeto do banco de dados e é parte integrante do subconjunto da linguagem SQL denominado:

a) DML - Linguagem de Manipulação de Dados.
b) DTL - Linguagem de Transação de Dados.
c) DCL - Linguagem de Controle de Dados.
d) DDL - Linguagem de Definição de Dados.

e) DQL - Linguagem de Consulta de Dados.

37. **(FCC / MPE-SE – 2010)** Em relação às linguagens de definição e de manipulação de dados no SQL, é correto afirmar:

a) no grupo DDL, apagar tabelas e índices da base de dados é função do comando DROP.

b) no grupo DML, conceder acesso à base de dados e aos seus objetos é função do comando ALTER.

c) no grupo DDL, o comando SELECT é utilizado para extrair e alterar dados da base de dados.

d) o grupo DDL contém os comandos para criar e alterar novas tuplas no banco de dados.

e) o comando ALTER, do grupo DML, tem como função alterar linhas já existentes no banco de dados.

38. **(FCC / DPE-SP – 2010)** O SQL (Structured Query Language) é uma linguagem de pesquisa declarativa para banco de dados relacional. A DML é um subconjunto da linguagem usada para inserir, atualizar e apagar dados. A sigla DML significa:

a) Data Main Language.
b) Data Manager Language.
c) Data Manipulation List.
d) Data Manager List.
e) Data Manipulation Language.

39. **(FCC / TRE-AM – 2010)** Em SQL, a deleção de linhas em uma tabela é feita por meio da expressão geral:

a) WHERE nome_coluna FROM nome_tabela = valor_qualquer DELETE nome_coluna.
b) WHERE nome_coluna = valor_qualquer DELETE nome_coluna FROM nome_tabela.
c) DELETE nome_coluna FROM nome_tabela WHERE nome_coluna = valor_qualquer.
d) DELETE WHERE nome_tabela.nome_coluna = valor_qualquer
e) DELETE FROM nome_tabela WHERE nome_coluna = valor_qualquer.

40. **(FCC / TRE-RN – 2010)** Na SQL, é o comando principal da Linguagem de Consulta de Dados:

a) REVOKE.
b) DROP.
c) WHERE.
d) SELECT.

e) HAVING.

41. **(FCC / TCE-GO – 2009)** Considere:

Select (X) from (Y) order by (Z)

Na SQL, X, Y e Z são, respectivamente,

a) nome de tabela, nome de coluna e nome de coluna.
b) nome de coluna, nome de tabela e nome de coluna.
c) condição, nome de tabela e nome de coluna.
d) nome de tabela, condição e nome de coluna.
e) nome de coluna, nome de tabela e condição.

42. **(FCC / TRE-PI – 2009)** Na linguagem SQL, considere os comandos relativos à Linguagem de Definição de Dados ? DDL e à Linguagem de Manipulação de Dados ? DML:

a. ALTER
b. CREATE
c. DELETE
d. DROP
e. INSERT
f. SELECT
g. SET

A associação entre os comandos e suas respectivas linguagens de definição e manipulação de dados está correta em:

a) DDL - abcd // DML - efg.
b) DDL - bfg // DML - acde.
c) DDL - abf // DML - cdeg.
d) DDL - abd // DML - cefg.
e) DDL - acg // DML - bdef.

43. **(FCC / TRE-SE – 2007)** Em SQL-ANSI, Count:

a) é um comando de intersecção no contexto da DML.
b) é uma função de agregação no contexto da DML.
c) é um operador de conjunto no contexto da DDL.
d) é uma função de restrição no contexto da DML.
e) é uma expressão de seleção no contexto da DDL.

LISTA DE QUESTÕES – FGV

```
create table X(A int not null primary key,
               B int)
create table Y(A int not null UNIQUE,
               constraint fk
               foreign key (A) references X(A)
               on delete cascade)
```

Para todos os efeitos, suponha que o número de linhas em cada tabela é diferente de zero.

44. (FGV / TRT-MA – 2022) Assinale a afirmativa correta a respeito do esquema relacional apresentado.

a) a tabela X admite linhas duplicadas.
b) a tabela X não pode ter mais linhas que a tabela Y
c) a tabela Y admite linhas duplicadas.
d) a tabela Y não pode ter mais linhas que a tabela X.
e) as tabelas X e Y não podem ter o mesmo número de linhas.

Comentários:

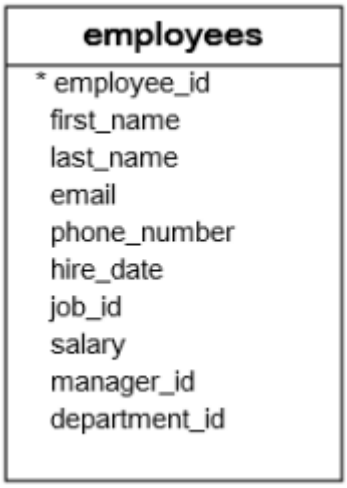
Observem que a Tabela X tem dois atributos: A, que é chave primária; e B, que é um inteiro. Já a Tabela Y tem somente um atributo: A, que é também chave primária (dado que é único e não nulo) e esse atributo é uma chave estrangeira que referencia o atributo A da Tabela X.

Se o atributo A da Tabela Y referencia o atributo A da Tabela X, então a Tabela Y jamais poderá ter mais linhas que a Tabela X. Já a Tabela X pode ter mais (ou iguais) linhas que a Tabela Y sem nenhum problema. E nenhuma das tabelas admite linhas duplicadas porque possuem atributos únicos e não nulos.

Gabarito: Letra D

---

45. (FGV / SEFAZ-BA – 2022) Considere a seguinte tabela em um banco de dados relacional.



Assinale a opção que indica o comando SQL utilizado para localizar todos os nomes completos dos *employees*, cujos primeiros nomes começam com as letras Ma.

a) SELECT
first_name;
last_name;
FROM
employees;

b) SELECT *
FROM
employees
WHERE
first_name = 'Ma';

c) SELECT *
FROM
employees
WHERE
first_name = 'Ma*';

d) SELECT
employee_id,
first_name,
last_name
FROM
employees
WHERE
first_name LIKE 'Ma%';

e) SELECT

```
        employee_id,
        first_name,
        last_name
FROM
        employees
WHERE
        first_name IN 'Ma_';
```

Tabela T

sequencia	caracteristica
1	23987
2	9845
3	NULL
4	40983
6	48750
7	NULL
8	NULL
10	48750
12	48750

Tabela TX

sequencia	caracteristica
2	9845
3	998034
4	50932
5	24390
6	48750
6	50296
7	NULL
8	998746
9	32746
9	NULL
9	22798

Tabela DUAL

x
NULL

46.(FGV/ TCU – 2022) Considere que é preciso atualizar os dados da tabela T a partir dos dados da tabela TX, ambas definidas anteriormente. A consolidação é feita por meio da alteração na tabela T a partir de registros de TX. O comando SQL utilizado nessa atualização é exibido a seguir.

```
update T
set caracteristica =
   select max(caracteristica) x from TX tx
   where tx.sequencia = t.sequencia
         and not (tx.caracteristica is null))
where
( exists
 (select * from TX tx
 where tx.sequencia = t.sequencia
   and not (tx.caracteristica is null))
 and
 ( t.caracteristica is null
   or
   t.caracteristica <
         (select max(caracteristica) x from TX tx
          where tx.sequencia = t.sequencia
               and not (tx.caracteristica is null))
 )
)
```

O número de registros da tabela T afetados pela execução do comando SQL acima é:

a) zero;
b) três;
c) quatro;
d) seis;
e) nove.

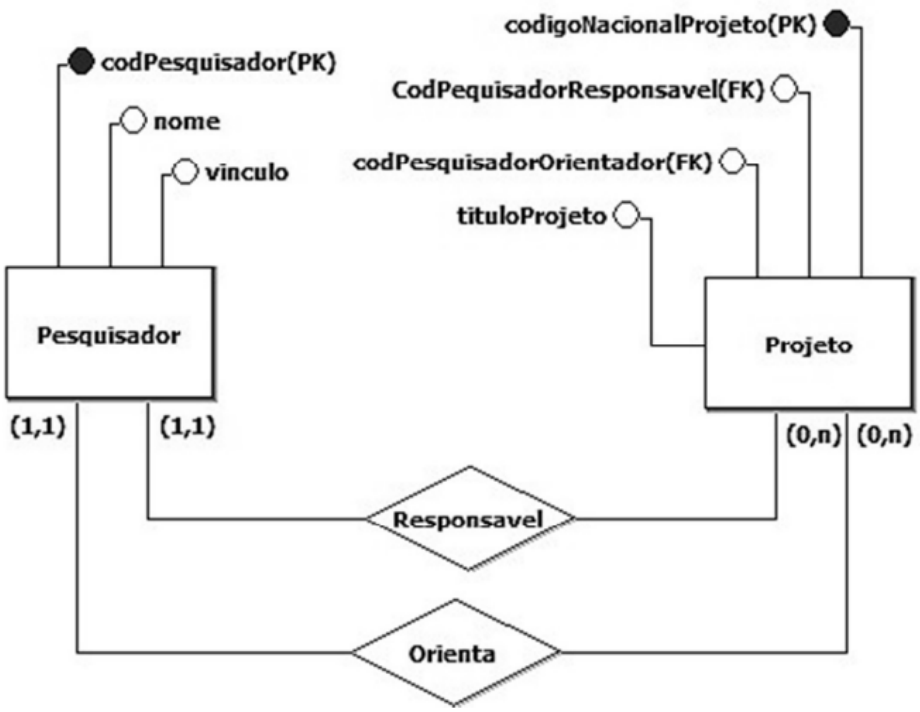
47.(FGV/ TCU – 2022) Analise os cinco comandos SQL exibidos abaixo, utilizando a tabela DUAL apresentada anteriormente.

(1) select * from dual where x = null
(2) select * from dual where x <> null
(3) select * from dual where x > 10
(4) select * from dual where not x > 10
(5) select * from dual where x > 10
    union
    select * from dual where x <= 10

Se os resultados desses comandos fossem separados em grupos homogêneos, de modo que em cada grupo todos sejam idênticos e distintos dos elementos dos demais grupos, haveria:

a) apenas um grupo;

b) apenas dois grupos;
c) apenas três grupos;
d) apenas quatro grupos;
e) cinco grupos.



Com base no modelo entidade-relacionamento (MER) precedente, que apresenta a representação das regras de uma instituição de pesquisa, existe um Pesquisador cadastrado com o nome Pedro. Todos os atributos do MER são do tipo caractere e um dos comandos SQL usados para a construção do modelo é mostrado a seguir.

```
create table Projeto
codNacionalProjeto char(2),
codPesquisadorResponsavel char(2),
codPesquisadorOrientador char(2),
tituloProjeto char(50),
primary key(codNacionalProjeto));
```

A partir das informações constantes no modelo e dos dados sobre o conteúdo dos atributos, julgue os itens subsecutivos.

48.(FGV / FUNSÚDE-CE – 2021) Atenção: na próxima questão, considere a definição e as instâncias das tabelas de bancos de dados CLUBE e JOGO exibidas a seguir.

```
CLUBE
nome
Barcelona
Boca Juniors
```

The Strongest

JOGO

mandante	visitante	golsM	golsV
Barcelona	Boca Juniors	1	0
Barcelona	The Strongest	NULL	NULL
Boca Juniors	Barcelona	0	0
Boca Juniors	The Strongest	3	0
The Strongest	Barcelona	2	0
The Strongest	Boca Juniors	2	0

Cada clube deve jogar quatro vezes, duas como mandante e duas como visitante. As colunas golsM e golsV registram o número de gols dos times mandantes e visitantes, respectivamente, em cada jogo. Ambas são nulas enquanto o jogo não for realizado.

Analise o comando SQL a seguir, à luz das definições e instâncias das tabelas CLUBE e JOGO, apresentadas anteriormente.

```
select distinct mandante, visitante from JOGO, CLUBE
```

Assinale o número de linhas, sem incluir os títulos, produzidas pela execução desse comando:

a) 4.
b) 6.
c) 10.
d) 24.
e) 48.

49. **(FGV / FUNSÚDE-CE – 2021)** Analise o comando SQL a seguir, à luz das definições e instâncias das tabelas CLUBE e JOGO, definidas anteriormente.

```
select c.nome from CLUBE c where  (
   select count(*) from JOGO j  where c.nome = j.mandante) <> 2  or  (
        select count(*) from JOGO j  where c.nome = j.visitante) <> 2
```

O resultado produzido pela execução desse comando é a lista de todos os clubes que:

a) aparecem em quatro jogos.
b) não aparecem em quatro jogos.
c) não aparecem em dois jogos como mandante ou que não aparecem em dois jogos como visitante.
d) aparecem em dois jogos como mandante ou que aparecem em dois jogos como visitante.
e) aparecem em dois jogos como mandante e aparecem em dois jogos como visitante.

A	B	C	D
12	2	3	1
14	3	8	2
18	2	9	3
21	5	4	4

50. **(FGV / TCE-AM – 2021)** Considerando-se a instância da tabela T (descrita anteriormente), analise o comando SQL abaixo.

```
delete from T where b + d = c
```

O número de registros da tabela T afetados pela execução desse comando é:

a) zero;
b) um;
c) dois;
d) três;
e) quatro.

A	B	C	D
12	2	3	1
14	3	8	2
18	2	9	3
21	5	4	4

51. **(FGV / TCE-AM – 2021)** Considerando-se a instância da tabela T (descrita anteriormente), analise o comando SQL abaixo.

```
update T
set a = a + 32
where
exists (select * from T t2 where T.c > t2.D)
```

O número de registros da tabela T afetados pela execução desse comando é:

a) zero;
b) um;
c) dois;
d) três;

e) quatro.

A	B	C	D
12	2	3	1
14	3	8	2
18	2	9	3
21	5	4	4

52. (FGV / TCE-AM – 2021) Considerando-se a instância da tabela T (descrita anteriormente), analise o comando SQL abaixo.

```
select distinct * from T t1, T t2, T t3
```

A execução desse comando produz um resultado que, além da linha de títulos, contém:

a) 8 linhas;
b) 24 linhas;
c) 32 linhas;
d) 64 linhas;
e) 128 linhas.

53. (FGV / IMBEL - 2021) Considere a instância da tabela R1 e o comando SQL exibidos a seguir.

A	B
1	2
2	2
3	3
4	3
4	2
4	1
5	0

```
select distinct A
from R1
where A not in
        (select B from R1)
```

Assinale a lista de números que é exibida quando esse comando SQL é executado.

a) 5.
b) 1,2.
c) 2,3.
d) 4,5.

e) 1, 2, 3, 4.

54. (FGV / IMBEL – 2021) Considere o comando SQL a seguir, executado num banco de dados relacional com duas tabelas, R1 e R2, contendo 2.000 e 5.000 registros, respectivamente. R1 e R2 possuem chaves primárias definidas.

```
SELECT DISTINCT * FROM A, B
```

Assinale o número de linhas produzidas na execução:

a) 1.
b) 2.000.
c) 5.000.
d) 7.000.
e) 10.000.000

55. (FGV / IMBEL – 2021) Considere a instância da tabela R1 e o comando SQL exibidos a seguir.

A	B
1	2
2	2
3	3
4	3
4	2
4	1
5	0

```
select distinct A
from R1
where exists
     (select * from R1 x Where x.B > R1.A)
```

Assinale a lista de números que é exibida quando esse comando SQL é executado:

a) 5.
b) 1,2.
c) 2,3.
d) 3,4,5.
e) 1, 2, 3, 4.

pessoa1	pessoa2	relação
João	Rafael	pai
Maria	Rafael	mãe
Rafael	Gabriela	pai
Gabriela	Rita	mãe
Rita	Bruna	mãe
Bruna	Ana	mãe
Rafael	Rita	avo

56. **(FGV / DPE-RJ – 2019)** Considere a tabela FAMILIA descrita anteriormente e o comando SQL a seguir.

```
select relação, sum(1)
from familia
group by relação
having count(*) > 1
order by 2 desc, 1
```

Os valores exibidos pela execução desse comando, na ordem, são:

a) `mãe 4`
`pai 2`
`avo 1`

b) `mãe 2`
`pai 4`

c) `pai 2`
`mãe 4`

d) `mãe 4`
`pai 2`

e) `mãe 4`
`pai 2`
`avo ∅`

57. **(FGV / Prefeitura de Niterói-RJ – 2018)** A otimização de consultas em gerenciadores de bancos de dados é fundamental para o desempenho do sistema. Consultas escritas em SQL são particularmente propícias à otimização, porque essa linguagem:

a) não é procedural, permitindo a criação de diferentes planos de execução.
b) tem uma sintaxe simples e é largamente utilizada.
c) suporta todas as operações da Álgebra Relacional.
d) permite o uso de subconsultas, facilitando os processos de busca.

e) permite a criação de camadas de software de persistência.

Pessoa	Descendente
Ana	Vitoria
João	Maria
João	Rafael
Maria	Tiago
Natalia	Ana
Rafael	Natalia
Ana	Vitoria

58. **(FGV / AL-RO – 2018)** Analise o comando a seguir utilizando a tabela arvore, definida anteriormente.

```
delete from arvore
where exists
          (select * from arvore a
          where a.pessoa = 'João'
          and a.descendente = arvore.pessoa)
```

Assinale o número de registros que é removido na execução desse comando.

a) Zero.
b) Um.
c) Dois.
d) Três.
e) Quatro.

59. **(FGV / AL-RO – 2018)** Considere o seguinte comando SQL numa instalação MS SQL Server.

```
select A, count(*) X
from T1
where B > 2
group by A
```

Assinale a cláusula order by que seria inválida nesse comando.

a) order by A
b) order by B
c) order by X
d) order by 1
e) order by avg(B)

60. **(FGV / AL-RO – 2018)** Considere uma tabela relacional T, com atributos A e B, onde A, isoladamente, constitui a chave primária de T. Considere ainda o comando SQL a seguir.

```
select * from T t1, T t2
where t1.A = t2.A
and exists (select * from T t3
            where t3.A <> t2.A)
```

Dado que essa tabela possui 10 registros, assinale o número de linhas que, além dos títulos, aparece no resultado produzido pela execução desse comando.

a) 0
b) 1
c) 10
d) 100
e) 1000

61. (FGV / MPE-AL – 2018) Considere o comando SQL a seguir.

```
select * from teste where nota <> null
```

Sabendo-se que a tabela teste tem 100 registros, dos quais 15 estão com a coluna nota ainda não preenchida (null), o comando acima, executado no MS SQL Server ou no Oracle, retorna

a) um erro, porque o literal null não pode ser usado diretamente numa comparação.
b) zero linhas.
c) quinze linhas.
d) oitenta e cinco linhas.
e) cem linhas.

62. (FGV / BANESTES – 2018) Considere um banco de dados com duas tabelas, R e S, contendo 4 e 3 registros, respectivamente. Em R, os valores da coluna A são 1, 2, 5 e 7. Em S, os valores da coluna B são 2, 4 e 7. Excetuando-se a linha de títulos, o número de linhas no resultado do comando SQL

```
select * from R full outer join S on A=B
```

É:

a) 3
b) 4
c) 5
d) 7
e) 12

63. (FGV / Prefeitura de Niterói-RJ – 2018) A questão deve ser respondida a partir das tabelas de banco de dados t1 e t2, a seguir.

T1

A	B	C
1	2	4
2	3	5
4	2	4
6	2	NULL

T2

D	E	F
1	2	NULL
2	5	5
4	2	1
7	12	1

Analise o comando SQL exibido abaixo:

```
select  *  from  T1  where  C > 5
UNION
select  * from  T1  where  C <= 5
```

A execução desse comando no MS SQL Server produz um resultado que contém, além da linha de títulos, n linhas. Assinale o valor de n.

a) 3
b) 4
c) 5
d) 6
e) 8

64.(FGV / BANESTES – 2018) Considere a tabela de banco de dados R, com a seguinte instância.

a	b
1	2
2	3
4	5

Após a execução do comando SQL

```
update r set a = a + b where b > (select max(a) from r)
```

a instância da referida tabela é, na respectiva ordem de colunas e linhas:

a) 1 2
2 3
4 5

b) 10 2
10 3

10 5

c) 3 2
3 3
4 5

d) 3 2
5 3
9 5

e) 1 2
2 3
9 5

65. **(FGV / BANESTES – 2018)** Considere um banco de dados com duas tabelas. A primeira tabela, números, possui dez registros e apenas uma coluna, cujos valores são 1, 2, 3, 4, 5, 5, 9, 9, 9, 10. A segunda tabela, denominada teste, com cinco registros, também possui apenas uma coluna, cujos valores são 1, 3, 3, 4, 5. Considere ainda o seguinte comando SQL

```
insert into teste select numero from numeros n where not exists (select * from teste t where t.numero = n.numero)
```

Quando da execução desse comando, o número de registros inseridos na tabela teste é:

a) 2;
b) 3;
c) 5;
d) 8;
e) 10.

66. **(FGV / SEFIN-RO – 2018)** Considere as tabelas de bancos de dados T1, T2 e T3, que contêm, respectivamente, 10, 500 e 2.000 registros, e o comando SQL a seguir.

```
select count(*) FROM T1, T2, T3
```

Assinale a opção que apresenta o número exibido no resultado da execução desse comando.

a) 10000000
b) 1000000
c) 2000
d) 500
e) 10

67. **(FGV / IBGE – 2017)** A linguagem mais comum para elaboração de consultas em bancos de dados é a SQL. Ao elaborar uma consulta nessa linguagem, emprega-se a cláusula WHERE quando se deseja:

a) especificar a tabela onde será realizada a consulta;
b) especificar o diretório onde os dados estão armazenados;
c) especificar o endereço IP onde os dados estão armazenados;
d) especificar condições a que as instâncias selecionadas devem atender;
e) extrair a geometria do objeto selecionado na consulta.

R	
a	b
1	2
2	3
4	5

S	
c	d
3	2
4	2
6	1

68. **(FGV / MPE-BA – 2017)** Considerando as tabelas R e S apresentadas anteriormente, o comando SQL

```
SELECT a FROM R UNION ALL SELECT d FROM S
```

produz um resultado que contém, além dos títulos:

a) 1 linha;
b) 3 linhas;
c) 4 linhas;
d) 5 linhas;
e) 6 linhas.

R	
a	b
1	2
2	3
4	5

S	
c	d
3	2
4	2
6	1

69. **(FGV / MPE-BA – 2017)** Considere as tabelas R e S apresentadas anteriormente e o comando SQL a seguir.

```
update R set a = NULL
where b >= (select max(d) from S)
```

Após execução desse comando, os valores na coluna a da tabela R seriam, de cima para baixo:

a) NULL, NULL, NULL
b) 1, 2, 4
c) 1, 2, NULL
d) 1, NULL, NULL
e) NULL, 2, 4

R	
a	b
1	2
2	3
4	5

S	
c	d
3	2
4	2
6	1

70. **(FGV / MPE-BA – 2017)** Considerando as tabelas R e S apresentadas anteriormente, o resultado

a	b
1	2
2	3

seria obtido pela execução do comando SQL:

a) select * from R
where not exists (select * FROM S where a=c)

b) select * from R
where not exists (select * FROM S where a=d)

c) select * from S
where not exists (select * FROM R where a=c)

d) select * from S

where not exists (select * FROM R where a=d)

e) select * from R
where exists (select * FROM S where b=d)

71. (FGV / COMPESA – 2016) Analise o comando SQL a seguir, no contexto das tabelas R1, R2 e R3.

R1

A1	B1
2	3
2	4
1	1

R2

A2	B2
2	3
1	1
1	1

R3

A3	B3	C3
2	3	4
1	2	3
1	1	1

```
select * from R1
where not exists
          (select * from R2
           where R1.A1 = R2.A2 and R1.B1 = R2.B2)
```

Assinale a soma dos valores numéricos exibidos no resultado produzido pelo referido comando.

a) 2
b) 4
c) 6
d) 8
e) 11

72. (FGV / IBGE – 2016) O comando SQL

```
select a, sum(b) x, COUNT(*) y
from T
group by a
```

produz como resultado as linhas abaixo.

a	x	y
1	6	1
3	6	2
4	4	1
5	1	1

Na tabela T, composta por duas colunas, a e b, nessa ordem, há um registro duplicado que contém os valores:

a) 1 e 3

b) 3 e 3
c) 3 e 6
d) 4 e 2
e) 5 e 1

73. (FGV / IBGE – 2016) Os comandos SQL

```
create table R (a int, b int)
create table S (c int, d int)
insert into R values(1,2)
insert into R values(2,3)
insert into R values(2,3)
insert into R values(3,5)
insert into R values(4,1)
insert into S values(1,2)
insert into S values(2,1)
insert into S values(2,3)
insert into S values(3,5)
select r.a,r.b from R
where not exists
   (select * from S where s.c=r.a and s.d=r.b)
```

Produzem um resultado que, além da linha de títulos, contém:

a) uma linha;
b) duas linhas;
c) três linhas;
d) quatro linhas;
e) quatro linhas;

74. (FGV / TJ-PI – 2015) Considerando a tabela T, analise o comando SQL a seguir.

T

a	b	c
1	2	NULL
2	3	NULL
5	NULL	NULL
4	2	NULL

```
delete from T
where exists
          (select * from T t1
           where T.a > t1.c or T.a <= t1.c)
```

O número de registros da tabela T removidos pela execução desse comando é:

a) zero;
b) um;
c) dois;

d) três;
e) quatro.

75. **(FGV / PGE-RO – 2015)** No SQL, a outorga de privilégios sobre objetos de um banco de dados é efetuada por meio do comando:

a) CREATE;
b) GRANT;
c) LICENSE;
d) PERMIT;
e) REVOKE.

76. **(FGV / DPE-RO – 2015)** Observe o comando SQL a seguir.

```
update X set Y = 'Z'
```

Para que esse comando esteja corretamente formulado, quando analisado isoladamente, pressupõe-se que:

a) Y seja uma coluna da tabela X;
b) X seja uma coluna da tabela Y;
c) X e Y sejam tabelas;
d) X seja um banco de dados e Y seja uma tabela;
e) X seja um banco de dados e Y seja uma tabela;

77. **(FGV / DPE-RO – 2015)** Analise o comando SQL a seguir.

```
select distinct 1 from X
```

Sabendo-se que a instância da tabela X não é vazia, conclui-se que a execução desse comando produz um resultado com:

a) apenas uma linha;
b) um conjunto de linhas contendo o valor NULL;
c) um conjunto de linhas contendo todos os atributos de X;
d) um número de linhas igual ao número de diferentes valores do primeiro atributo de X;
e) um número de linhas igual ao número de registros de X.

78. **(FGV / TJ-BA – 2015)** Considere que as instâncias das tabelas T1, T2 e T3 têm, respectivamente, 1.000, 10.000 e 100.000 registros. O comando SQL:

```
select 1 from t1
union
select 2 from t2
```

```
union
select 3 from t3
```

produz um resultado com:

a) 3 linhas;
b) 1.000 linhas;
c) 10.000 linhas;
d) 100.000 linhas;
e) 111.000 linhas.

79. **(FGV / PGE-RO – 2015)** No SQL, a outorga de privilégios sobre objetos de um banco de dados é efetuada por meio do comando:

a) CREATE;
b) GRANT;
c) LICENSE;
d) PERMIT;
e) REVOKE.

80. **(FGV / TCE-SE – 2015)** Considere duas tabelas X e Y, com as seguintes instâncias:

X	
a	b
1	2
3	3
4	5
5	7

Y	
c	d
1	2
3	4
5	6
7	8
9	1

O comando SQL que retorna

a	b	c	d
1	2	1	2
3	3	3	4
4	5	NULL	NULL
5	7	5	6
NULL	NULL	7	8
NULL	NULL	9	1

é:

a) select * from X FULL JOIN Y on X.a=Y.c
b) select * from X LEFT JOIN Y on X.a=Y.c
c) select * from Y RIGHT JOIN X on X.a=Y.c

d) select * from X CROSS JOIN Y on X.a=Y.c
e) select * from X INNER JOIN Y on X.a=Y.c

81. (FGV / TJ-RO – 2015) Considere as seguintes tabelas relacionais e respectivas instâncias.

R

A	B
1	3
2	3
3	4
5	5
5	7
8	8

S

C	D
1	9
2	3
3	4
3	4
3	4

Analise o comando SQL a seguir.

```
DELETE FROM S
WHERE NOT EXISTS
(SELECT * FROM R
WHERE R.A = S.C AND R.B = S.D)
```

O número de registros deletados por esse comando é:

a) 0;
b) 1;
c) 2;
d) 4;
e) 5.

82. (FGV / TJ-RO – 2015) Considere as seguintes tabelas relacionais e respectivas instâncias.

R

A	B
1	3
2	3
3	4
5	5
5	7
8	8

S

C	D
1	9
2	3
3	4
3	4
3	4

Analise o comando SQL a seguir.

```
SELECT * FROM R UNION SELECT * FROM S
```

O número de linhas produzidas por esse comando, excetuada a linha de títulos de colunas, é:

a) 2;

b) 5;
c) 6;
d) 7;
e) 11.

83. (FGV / DPE-RO – 2015) Analise o comando SQL a seguir.

```
SELECT DISTINCT 1 FROM X
```

Sabendo-se que a instância da tabela X não é vazia, conclui-se que a execução desse comando produz um resultado com:

a) apenas uma linha;
b) um conjunto de linhas contendo o valor NULL;
c) um conjunto de linhas contendo todos os atributos de X;
d) um número de linhas igual ao número de diferentes valores do primeiro atributo de X;
e) um número de linhas igual ao número de registros de X.

84. (FGV / TCE-SE – 2015) Considere duas tabelas X e Y, com as seguintes instâncias:

X	
a	b
1	2
3	3
4	5
5	7

Y	
c	d
1	2
3	4
5	6
7	8
9	1

```
delete from y
where y.c in
(select a from x union select c from y)
```

Considerando-se as tabelas e o comando SQL mostrados acima, é correto concluir que o número de registros removidos da tabela Y pela execução desse comando é:

a) 1
b) 2
c) 3
d) 4
e) 5

85. (FGV / DPE-MT – 2015) Analise o comando SQL a seguir.

```
select
   x,
```

```
   sum (nota) as soma,
   count (nota) as numero
from inscrição
group by x
having ???
```

Assinale a opção que indica a expressão que, ao ser utilizada para substituir o trecho "???", INVALIDA o comando SQL acima.

a) count (nota) > 1
b) x=2
c) max (nota) = 10
d) nota > 7
e) (select max(nota) from inscricao) > 5

86. (FGV / DPE-MT – 2015) Na maioria das implementações SQL, pode-se considerar que as expressões lógicas possam assumir três valores, verdadeiro (T), falso (F) e desconhecido (?). Isso decorre principalmente da manipulação de valores nulos (NULL). Assim sendo, analise as quatro expressões lógicas a seguir.

```
not ?
F or ?
T and ?
? or T
```

Assinale a opção que apresenta os valores finais das expressões lógicas acima, na ordem de cima para baixo.

a) F; ?; T; T
b) F; F; T; T
c) ?; ?; ?; ?
d) ?; ?; ?; T
e) ?; F; ?; ?

87. (FGV / TJ-AM – 2013) A SQL é constituída pela Data Control Language (DCL), a Data Definition Language (DDL) e a Data Manipulation Language (DML).

Assinale a alternativa que apresenta os três comandos que são parte integrante da DDL:

a) REVOKE, GRANT e DELETE
b) GRANT, DELETE e UPDATE
c) DELETE, UPDATE e CREATE
d) CREATE, ALTER e DROP
e) ALTER, DROP e REVOKE

88. (FGV / MPE-MS – 2013) Observe o comando SQL a seguir:

```
SELECT nome, sobrenome, PIS, anos_de_servico FROM Empregados
```

A cláusula que deve ser adicionada ao comando acima para ordenar os registros por anos de serviço, com os empregados que estão há mais tempo na empresa aparecendo primeiro na listagem, é:

a) ORDER 'anos_de_servico' BY ASC
b) ORDER BY 'anos_de_servico'
c) ORDER BY anos_de_servico DESC
d) SORTED BY anos_de_servico DESC
e) ORDER BY anos_de_servico ASC

LISTA DE QUESTÕES – DIVERSAS BANCAS

89. **(FEPESE / ISS-Criciúma – 2022)** No contexto de uma consulta SQL com o objetivo de retornar os resultados de duas relações, sendo que o resultado desejado envolve todos os registros da primeira relação em conjunto com os resultados dos registros correspondentes à condição da junção da segunda relação, qual instrução deve ser utilizada?

a) JOIN
b) INNER JOIN
c) OUTER JOIN
d) RIGHT JOIN
e) LEFT JOIN

90. **(FADESP / SEFA-PA – 2022)** A linguagem de banco de dados relacional SQL (Structured Query Language) é um exemplo de linguagem de banco de dados abrangente que representa uma combinação de:

a) TTL, VDL e DML.
b) TDL, GDL e DML.
c) DDL, VDL e DML.
d) DDL, VDL e BML.
e) DDL, GDL e BML.

As questões 06, 07 e 08 baseiam-se na Figura 6, que mostra, esquematicamente, um Diagrama Entidade-Relacionamento (DER) elaborado no MySQL Workbench 8.0, no qual se inseriu, intencionalmente, nos locais apontados pelas setas nº 1 e 2, retângulos para ocultar os relacionamentos existentes nesses locais. Nesse DER, constam as entidades "Produto", "Aquisicao" e "Cliente", implementadas de acordo com as seguintes regras de negócio:

(1) um cliente poderá adquirir um ou mais produtos, inclusive os mesmos produtos mais de uma vez, em data/hora diferentes;

(2) um produto poderá ser adquirido por um ou mais clientes, inclusive o mesmo cliente, mais de uma vez;

(3) deve ser possível cadastrar qualquer produto ou cliente, no banco de dados, sem associá-los a qualquer outra tabela;

(4) ao se associar um cliente a um produto, armazena-se, no banco de dados, a quantidade adquirida, a correspondente data/hora de aquisição e o preço efetivamente pago (que poderá ser diferente do preço de tabela do produto, devido ao cliente ter recebido um desconto no preço do produto).

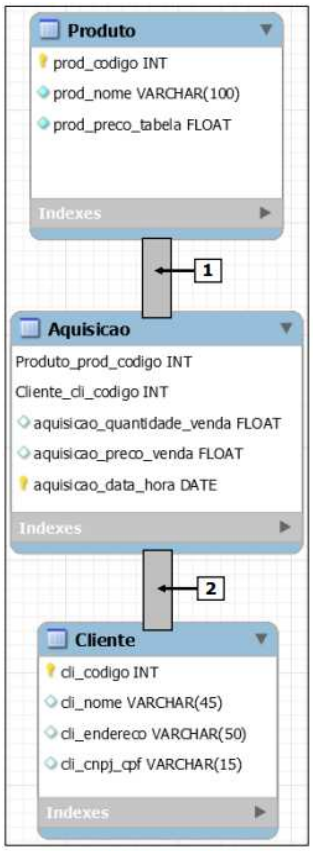


Figura 6 – DER

91. **(FUNDATEC / ISS-Porto Alegre – 2022)** Sabe-se que, a partir do DER mostrado na Figura 6, foram criadas e populadas as tabelas correspondentes em um Sistema Gerenciador de Banco de Dados Relacional (SGBDR), tendo se respeitado, rigorosamente, os conceitos do modelo relacional. Nesse caso, para criar a tabela "Aquisicao", bastou executar a seguinte declaração, em SQL padrão ANSI:

a)
```
CREATE TABLE Aquisicao (
  Produto_prod_codigo INT NOT NULL,
  Cliente_cli_codigo INT NOT NULL,
  aquisicao_quantidade_venda FLOAT NOT NULL,
  aquisicao_preco_venda FLOAT NOT NULL,
  aquisicao_data_hora DATE NOT NULL,
  PRIMARY KEY (aquisicao_data_hora, Produto_prod_codigo, Cliente_cli_codigo),
  FOREIGN KEY (Produto_prod_codigo) REFERENCES Produto (prod_codigo),
  FOREIGN KEY (Cliente_cli_codigo) REFERENCES Cliente (cli_codigo)
);
```

b)
```
CREATE TABLE Aquisicao (
  Produto_prod_codigo INT NOT NULL,
  Cliente_cli_codigo INT NOT NULL,
  aquisicao_quantidade_venda FLOAT NULL,
  aquisicao_preco_venda FLOAT NULL,
  aquisicao_data_hora DATE NOT NULL,
  PRIMARY KEY (Produto_prod_codigo, Cliente_cli_codigo),
  FOREIGN KEY (Produto_prod_codigo) REFERENCES Produto (prod_codigo),
  FOREIGN KEY (Cliente_cli_codigo) REFERENCES Cliente (cli_codigo)
);
```

c)
```
CREATE TABLE Aquisicao (
  Produto_prod_codigo INT PRIMARY KEY,
  Cliente_cli_codigo INT PRIMARY KEY,
  aquisicao_quantidade_venda FLOAT NOT NULL,
  aquisicao_preco_venda FLOAT NOT NULL,
  aquisicao_data_hora DATE PRIMARY KEY,
  FOREIGN KEY (Produto_prod_codigo) REFERENCES Produto (prod_codigo),
  FOREIGN KEY (Cliente_cli_codigo) REFERENCES Cliente (cli_codigo)
);
```

d)
```
CREATE TABLE Aquisicao (
  Produto_prod_codigo INT PRIMARY KEY REFERENCES Produto (prod_codigo),
  Cliente_cli_codigo INT PRIMARY KEY ) REFERENCES Cliente (cli_codigo),
  aquisicao_quantidade_venda FLOAT NULL,
  aquisicao_preco_venda FLOAT NULL,
  aquisicao_data_hora DATE PRIMARY KEY
);
```

e)
```
CREATE TABLE Aquisicao (
  Produto_prod_codigo INT PRIMARY KEY REFERENCES Produto (prod_codigo) NOT NULL,
  Cliente_cli_codigo INT PRIMARY KEY ) REFERENCES Cliente (cli_codigo) NOT NULL,
  aquisicao_quantidade_venda FLOAT NOT NULL,
  aquisicao_preco_venda FLOAT NOT NULL,
  aquisicao_data_hora DATE NOT NULL
);
```

92. (INAZ DO PARÁ / CORE-SP – 2019) SQL é uma linguagem de banco de dados abrangente, que possui instruções para definições de dados, consultas e atualizações. Apresenta alguns comandos principais e, entre eles, o "CREATE TABLE", que serve para:

a) Definir uma nova tabela de banco de dados relacional, criando as relações necessárias aos registros dessa tabela.
b) Especificar uma nova relação, dando-lhe um nome e especificando seus atributos e restrições iniciais.
c) Criar uma nova tabela a partir dos registros encontrados em tabelas diversificadas de banco de dados distintos.
d) Juntamente com o comando "CREATE DOMAIN", estabelecer todos os endereços dos registros de uma tabela.
e) Definir uma tabela centralizada a partir dos registros encontrados em banco de dados distribuídos.

93. (INAZ DO PARÁ / CORE-SP – 2019) "Embora o SQL tenha sido originalmente criado pela IBM, rapidamente surgiram vários "dialectos" desenvolvidos por outros produtores. Essa expansão levou à necessidade de ser criado e adaptado um padrão para a linguagem. Esta tarefa foi realizada pela American National Standards Institute (ANSI) em 1986 e ISO em 1987."
Disponível em: https://pt.wikipedia.org/wiki/SQL. Acesso em: 13.12.2018

Qual o código SQL contém comandos do tipo DDL?

a) drop table questoes.

b) insert into questoes select * from questoesmodelo.
c) delete from questoes.
d) select * from questoes.
e) select * from questoes.

94.(CCV / UFC – 2019) Em alguns cenários, é necessário definir que uma coluna em um banco de dados não deve permitir a inserção de valores repetidos. Qual das cláusulas abaixo deverá ser usada no comando SQL (Structured Query Language) para aplicar essa restrição no momento da criação da coluna?

a) CHECK
b) DEFAULT
c) UNIQUE
d) DISTINCT
e) CONSTRAINT

95.(CCV / UFC – 2019) Uma tabela chamada Area possui dois campos: arecod e aredes. Como podemos inserir um novo registro na tabela "Area"?

a) INSERT INTO Area (arecod, aredes) VALUES (100, "Técnico"), (200, "TI").
b) INSERT (100, "Técnico"), (200, "TI") INTO Area VALUES(arecod, aredes).
c) INSERT (arecod, aredes) INTO Area VALUES (100, "Técnico"), (200, "TI").
d) INSERT INTO (arecod, aredes) Area VALUES (100, "Técnico"), (200, "TI").
e) INSERT (100, "Técnico"), (200, "TI") INTO Area (arecod, aredes).

96. (CCV / UFC – 2019) Utilizando SQL, como selecionamos todos os registros de uma tabela chamada "Pessoas" onde o valor da coluna "PrimeiroNome " começa com "a"?

a) SELECT * FROM Pessoas WHERE PrimeiroNome='a'
b) SELECT * FROM Pessoas WHERE PrimeiroNome LIKE 'a%'
c) SELECT * FROM Pessoas WHERE PrimeiroNome='%a%'
d) SELECT * FROM Pessoas WHERE PrimeiroNome LIKE '%a'
e) SELECT * FROM Pessoas WHERE PrimeiroNome HAVING='%a%'

97.(NC-UFPR / Itaipu Binacional – 2019) A recursividade presente em consultas realizadas com SQL na forma SELECT a.id,... FROM a WHERE ... IN (SELECT atributo FROM b WHERE b.x=a.id) pode ser evitada por meio:

a) da substituição do operador IN por EXISTS.
b) da junção externa do tipo RIGHT JOIN com a verificação de atributos de b com o valor nulo.
c) da junção interna – INNER JOIN.
d) da junção externa do tipo LEFT JOIN com a verificação de atributos de b com o valor nulo.
e) da utilização de expressões de tabelas comuns (CTE).

```
SELECT DISTINCT cra
FROM TRABALHO
WHERE (forma_atuacao, uf) IN (
      SELECT forma_atuacao, uf FROM TRABALHO WHERE cra=2019);
```

98. **(QUADRIX / CRA-PR – 2019)** Uma consulta aninhada pode retornar tanto um único atributo quanto vários atributos e(ou) várias tuplas.

```
SELECT DISTINCT cra
FROM TRABALHO
WHERE (forma_atuacao, uf) IN (
      SELECT forma_atuacao, uf FROM TRABALHO WHERE cra=2019);
```


99. **(QUADRIX / CRA-PR – 2019)** O operador DISTINCT não pode ser utilizado em consultas aninhadas.

```
SELECT DISTINCT cra
FROM TRABALHO
WHERE (forma_atuacao, uf) IN (
      SELECT forma_atuacao, uf FROM TRABALHO WHERE cra=2019);
```


100. **(QUADRIX / CRA-PR – 2019)** A instrução demonstra que é permitido o uso de tuplas de valores em comparações, colocando-os entre parênteses, em consultas do tipo aninhada

---


```
SELECT DISTINCT cra
FROM TRABALHO
WHERE (forma_atuacao, uf) IN (
      SELECT forma_atuacao, uf FROM TRABALHO WHERE cra=2019);
```


101. **(QUADRIX / CRA-PR – 2019)** A instrução contém erro clássico de construção, pois, em uma consulta aninhada ou subconsulta, não é permitido o uso de nomes de tabelas repetidos, como, nesse caso, ocorre com a tabela TRABALHO.

---

102. **(NC-UFPR / Itaipu Binacional – 2019)** Considerando a linguagem SQL (Structured Query Language) para sistemas de banco de dados, assinale a alternativa que remove linhas de uma tabela chamada CLIENTE.

a) REMOVE FROM CLIENTE ...
b) CUT FROM CLIENTE ...
c) DELETE FROM CLIENTE WHERE ...
d) ERASE FROM CLIENTE ...
e) CLEAR FROM CLIENTE ...

103. (IADES / CRF - TO - 2019) A Linguagem de Consulta Estruturada (SQL – Structured Query Language) foi padronizada para utilização em bancos de dados em 1986 e é amplamente utilizada por diferentes Sistemas Gerenciadores de Bancos de Dados (SGBDs). Essa linguagem é dividida em quatro conjuntos, sendo eles linguagens:

a) de estruturação, de dados, para argumentação de controles e orientada a objetos.
b) orientada à conexão, estruturada, de manipulação de dados e de paralelismo.
c) para argumentação de controles, de definição de dados, orientada à conexão e de paralelismo.
d) para controle de acesso a dados, para transações, orientada a objetos e de estruturação.
e) de manipulação de dados, de definição de dados, para controle de transações e para controle de acesso a dados.

104. (Gestão Concurso / EMATER MG – 2018) Para se implementar um banco de dados relacional, bem como para armazenar os dados e posteriormente recuperá-los, faz-se uso da linguagem:

a) DDL.
b) DML.
c) SQL.
d) SGBD.

105. (SUGEP - UFRPE / UFRPE – 2018) A respeito da linguagem SQL, analise as proposições abaixo.

1) Um dos comandos do DML (Data Manipulation Language) é o INSERT.
2) CREATE e DELETE são comandos da DDL (Data Definition Language).
3) Dois comandos do DCL (Data Control Language) são UPDATE E DROP.

Está(ão) correta(s), apenas:

a) 1.
b) 1 e 2.
c) 2 e 3.
d) 3.
e) 1 e 3.

106. (FAURGS / UFRGS - 2018) Analise o trecho de código abaixo, escrito em SQL.

```
SELECT nome_pessoa
FROM PESSOA_FISICA
WHERE nome_pessoa NOT IN ('Benedita Silva', 'José
Silva', 'Lúcia Silva', 'João Silva')
```

Com base nesse código, é correto afirmar que o resultado da consulta é:

a) Benedita Silva.
b) José Silva.
c) João Silva.
d) Lúcia Silva.
e) Rosa Silva.

107. (COPESE / UFPI – 2018) As definições de uma tabela básica ou de outros elementos do esquema que possuírem denominação poderão, em linguagem SQL, ser alteradas pelo comando:

a) ALTER
b) DROP.
c) RESET.
d) RESET.
e) TABLE.

```
SELECT Cadastro.NomeCliente, Ordens.NumOrdem
FROM Cadastro
LEFT JOIN Ordens
ON Cadastro.NumCliente=Ordens.NumCliente
ORDER BY NomeCliente;
```

108. (QUADRIX / CRM-PR – 2018) O resultado será uma lista mostrada por ordem alfabética de NomeCliente e poderá ocorrer mais de uma linha contendo o mesmo registro NomeCliente.

```
SELECT Cadastro.NomeCliente, Ordens.NumOrdem
FROM Cadastro
LEFT JOIN Ordens
ON Cadastro.NumCliente=Ordens.NumCliente
ORDER BY NomeCliente;
```

109. (QUADRIX / CRM-PR – 2018) Há um erro de programação SQL na quarta linha, pois o registro NumCliente foi declarado em ambas as tabelas, Cadastro e Ordens.

```
SELECT Cadastro.NomeCliente, Ordens.NumOrdem
FROM Cadastro
LEFT JOIN Ordens
ON Cadastro.NumCliente=Ordens.NumCliente
ORDER BY NomeCliente;
```

110. (QUADRIX / CRM-PR – 2018) Com relação aos comandos SQL apresentados acima, julgue o item a seguir.

O resultado será uma lista com todas as ocorrências de NomeCliente da tabela Cadastro e os respectivos NumOrdem da tabela Ordens onde houver coincidência do NumCliente.

111. (IBADE/ Câmara de Porto Velho-RO – 2018) O comando SQL para extrair uma informação de um banco de dados é:

a) Extract.
b) Get.
c) Select.
d) Import.
e) Open.

112. (FUNDEP / UFVJM-MG – 2017) Qual comando SQL é necessário para listar os itens na tabela de PRODUTOS ordenados por NOME, começando pelo final do alfabeto?

a) SELECT * FROM PRODUTOS ORDER BY NOME
b) SELECT ALL FROM PRODUTOS ORDER BY NOME
c) SELECT * FROM PRODUTOS ORDER BY NOME INVERT
d) SELECT * FROM PRODUTOS ORDER BY NOME DESC.

113. (AOCP / EBSERH – 2017) Usando como referência o padrão SQL-92, assinale a alternativa que exemplifica corretamente um comando de inserção de dados em uma tabela usando a linguagem SQL.

a) INSERT SALDO SET CONTA = 25 AND CLIENTE = 12
b) INSERT INTO CONTA (ID_CONTA, ID_CLIENTE, SALDO) VALUES (1, 12, 993.56)
c) INSERT CONTA WHERE SALDO < 12.01
d) CREATE REGISTER CONTA FROM VALOR = 598.01 AND ID = 58
e) UPDATE CONTA SET CLIENTE = 31 AND SALDO = 554,32

114. (AOCP / EBSERH – 2017) A linguagem SQL é subdividida em subconjuntos, conforme a especificação das operações realizadas em um Banco de Dados (BD). Dentre essas subdivisões, temos a DML (Data Manipulation Language), que é responsável por realizar exclusões, inclusões,

atualizações e consultas aos dados presentes do BD. Assinale a alternativa que exemplifica um comando de exclusão de registro.

a) DELETE SALDO SET CONTA = 25
b) ALTER CONTA SET SALDO < 12.01
c) DELETE CONTA FROM VALOR = 598.01 AND ID = 58
d) DELETE FROM CONTA WHERE CLIENTE = 31 AND SALDO = 554,32
e) DROP TABLE CONTA FROM ID = 33 AND CLIENTE = 12

115. (IFB / IFB – 2017) Considerando-se a descrição sobre SQL (Structured Query Language), assinale a única alternativa onde a sequência de comandos SQL (da esquerda para a direita) está associada corretamente à sequência de categorias {comando DML, comando DDL, comando TCL}.

a) {CREATE, UPDATE, ROOLBACK}
b) {DELETE, COMMIT, INSERT}
c) {UPDATE, ALTER, COMMIT}
d) {CREATE, REVOKE, COMMIT}
e) {INSERT, DROP, GRANT}

116. (COSEAC / UFF – 2017) Na linguagem de manipulação de dados DML, os comandos que permitem remover linhas existentes em uma tabela e mudar os valores de dados em uma ou mais linhas da tabela existente são, respectivamente:

a) DELETE e UPDATE.
b) DROP e ALTER TABLE.
c) REVOKE e UPDATE.
d) DELETE e CREATE.
e) DROP e ALTER INDEX.

117. (FCM / IF Farroupilha-RS – 2016) O comando SELECT da linguagem Structured Query Language (SQL) permite acessar dados de um banco de dados.

id_aluno	nome	email	datacadastro
1	Wilson	wilson@ifrs.edu.br	2016-01-21
2	Jennifer	jennifer@ifrs.edu.br	2016-01-21
3	Lauro	lauro@ifrs.edu.br	2016-01-21
4	Renato	renato@ifrs.edu.br	2016-01-21

Uma consulta em linguagem SQL que exiba somente as colunas id_aluno e nome da tabela ALUNO apresenta-se abaixo em:

a) SELECT * FROM ALUNO
b) SELECT id_aluno, email FROM ALUNO
c) SELECT id_aluno, nome FROM ALUNO
d) SELECT email, datacasdastro FROM ALUNO
e) SELECT nome as nome_aluno FROM ALUNO

118. (IBFC / EBSERH - 2016) Com base no comando SQL abaixo assinale a alternativa que tenha a interpretação técnica correta:

DELETE FROM func WHERE tipo IN ('X','Y');

a) apaga registros de uma tabela tipo que contenha no campo func caracteres iguais a 'X' e 'Y'
b) apaga o campo tipo nos registros de uma tabela func que contenham os caracteres iguais a 'X' e 'Y'
c) apaga registros de uma tabela func que contenha no campo tipo caracteres iguais a 'X' e 'Y'
d) apaga o campo func nos registros de uma tabela tipo que contenham os caracteres iguais a 'X' e 'Y'
e) apaga qualquer registro de uma tabela func que contenha o campo tipo com caracteres iguais a 'XY'

119. (IF-SE / IF-SE – 2016) Em relação aos conceitos de linguagem SQL, avalie as afirmações abaixo:

I. São exemplos de DCL (Linguagem de Controle de Dados): grant e revoke.

II. São exemplos de DML (Linguagem de Manipulação de Dados): select, insert, update e drop.

III. São exemplos de DDL (Linguagem de Definição de Dados): create user, alter table e delete table.

a) Apenas a I é verdadeira
b) Apenas a II é verdadeira.
c) Apenas a III é verdadeira.
d) As três são verdadeiras.

120. (FUNRIO / IF-BA – 2016) Um dos mecanismos de segurança em um sistema de banco de dados é o subsistema de autorização, que permite a usuários que têm privilégios específicos concederem de forma seletiva e dinâmica esses privilégios a outros usuários e, subsequentemente, revogarem esses privilégios, se desejarem. Os comandos SQL que permitem a um usuário conceder privilégios a outros usuários e revogar privilégios concedidos a outros usuários são, respectivamente:

a) INSERT PRIVILEGES e DELETE PRIVILEGES.

b) CREATE ROLE e DROP ROLE.
c) CONCEDE e EXCLUDE.
d) GRANT e REVOKE.
e) ALLOW e DISALLOW.

121. (IBFC / EBSERH – 2016) Relacione as duas colunas quanto aos comandos SQL:

(1) Linguagem de Definição
(2) Linguagem de Manipulação

(A) INSERT
(B) CREATE
(C) DROP
(D) UPDATE

a) 1AD-2BC
b) 1BC-2AD
c) 1AB-2CD
d) 1CD-2AB
e) 1AC-2BD

122. (IBFC / EBSERH – 2016) Relacione os subconjuntos do SQL da coluna da esquerda com os seus respectivos comandos da coluna da direita:

(1) DDL
(2) DML

(A) UPDATE
(B) CREATE
(C) INSERT
(D) DROP.

a) 1A - 1C - 2B - 2D
b) 1A - 1B - 2C - 2D
c) 1C - 1D - 2A - 2B
d) 1B - 1C - 2A - 2D
e) 1B - 1D - 2A - 2C

123. (QUADRIX / CRO-PR – 2016) Nos bancos de dados SQL, para exibir todos os registros cadastrados em uma tabela chamada clientes utiliza-se a instrução:

a) SHOW ALL clientes;
b) SELECT * TO clientes;

c) SELECT ALL clientes;
d) SHOW * FROM clientes;
e) SELECT * FROM clientes;

124. (BIO-RIO / IF-RJ – 2015) "A linguagem SQL é do tipo declarativa e constituída das três sublinguagens a seguir:

(1) ____ - inclui os comandos SELECT, INSERT, UPDATE e DELETE;
(2) ____ - inclui os comandos CREATE, ALTER e DROP;
(3) ____ - inclui os comandos GRANT e REVOKE."

As siglas que completam corretamente as lacunas do fragmento acima são respectivamente:

a) DML, DCL e DDL.
b) DML, DDL e DCL.
c) DCL, DDL e DML.
d) DDL, DML e DCL.
e) DDL, DCL e DML.

125. (EXATUS / BANPARÁ – 2015) É um comando do tipo DDL (Data Definition Language) no SQL:

a) SELECT.
b) CREATE.
c) DELETE.
d) INSERT.
e) UPDATE.

126. (CETRO / AMAZUL – 2015) A linguagem SQL é dividida em subconjuntos de acordo com as operações que se deseja efetuar sobre um banco de dados. Um desses subconjuntos é a DDL. Assinale a alternativa que apresenta apenas comandos de DDL.

a) Select e Insert.
b) Create e Drop.
c) Update e Select.
d) Revoke e Alter.
e) Delete e Truncate.

127. (AOCP / EBSERH – 2015) De acordo com o conceito da DDL – Linguagem de definição de dados – quais comandos são utilizados na linguagem SQL?

a) SELECT, GRANT, DENY, REVOKE.
b) INSERT, UPDATE, GRANT, DROP.

c) CREATE, ALTER, DROP.
d) ALTER, DENY, GRANT.
e) GRANT, DENY, REVOKE.

128. (NUCEPE / SEDUC-PI – 2015) O SQL é uma linguagem de manipulação de banco de dados. Assinale a alternativa CORRETA sobre os subconjuntos do SQL:

a) DML – Linguagem de Transação de Dados.
b) DDL – Linguagem de Controle de Dados.
c) DCL – Linguagem de Definição de Dados.
d) DTL – Linguagem de Manipulação de Dados.
e) DQL – Linguagem de Consulta de Dados.

129. (CETAP - CPCM – 2015) Na linguagem SQL, a sentença que permite atualizar dois campos A e B em uma tabela X e:

a) UPDATE X WHERE A=2 AND B-3
b) UPDATE X SET A=2, B=3 WHERE C=4.
c) UPDATE X SET A=2 WHERE B=3
d) UPDATE X SET B=2 WHERE A=3
e) UPDATE A=2 AND B=3 WHERE X

130. (COMPERVE / UFRN – 2015) O comando na linguagem SQL que permite a remoção de todas as linhas de uma tabela X é:

a) REMOVE * FROM X
b) REMOVE ALL FROM X
c) DELETE FROM X
d) DELETE * FROM X

131. (FAFIPA / UFFS – 2014) Os comandos SQL podem ser divididos basicamente em três categorias: DML, DDL e DCL. Assinale a alternativa que apresenta um comando DCL:

a) SELECT
b) INSERT
c) ALTER
d) GRANT
e) DROP

132. (FUNCAB / PRODAM-AM – 2014) Se trata de um comando considerado parte da linguagem de definição de dados (DDL) e um comando considerado parte da linguagem de manipulação de dados (DML) do SQL, respectivamente:

a) CREATE e DROP
b) DROP e UPDATE
c) DELETE e CREATE
d) INSERTe UPDATE
e) SELECT e TRUNCATE

133. (IBFC / TRE-AM – 2014) Relacione as duas colunas, quanto aos comandos SQL e os respectivos subconjuntos da linguagem SQL:

(1) GRANT	(A) DCL
(2) DROP	(B) DML
(3) COMMIT	(C) DDL
(4) UPDATE	(D) DTL

Assinale a alternativa correta:

a) 1A-2C-3D-4B
b) 1A-2D-3C-4B
c) 1B-2C- 3D- 4A
d) 1C-2A-3D-4B

134. (UFBA / UNILAB – 2014) Os comandos commit e rollback são utilizados, respectivamente, para confirmar e desfazer instruções do tipo DML e DDL em um banco de dados.

135. (ESPP / MPE-PR – 2013) Ao se utilizar a linguagem SQL, aceitamos o conjunto de relações como dado que deve ser especificado no sistema gerenciador de banco de dados por meio de uma linguagem. A linguagem_____permite não só a especificação de um conjunto de relações, como também informações acerca de cada uma das relações, incluindo: o esquema de cada relação, o domínio dos valores associados a cada atributo, as regras de integridade, o conjunto de índices para manutenção de cada relação, informações sobre segurança e autoridade sobre cada relação e a estrutura de armazenamento físico de cada relação. Assinale a alternativa que completa corretamente a lacuna.

a) DDL
b) DSL
c) DTL
d) DQL
e) DCL

136. (CCV-UFC / UFC – 2013) Assinale a alternativa em que são apresentados dois comandos da linguagem de definição de dados (DDL).

a) CREATE e ALTER

b) CREATE e GRANT
c) INSERT e DELETE
d) GRANT e REVOKE
e) GRANT e ROLLBACK

137. (CEPERJ / CEPERJ – 2013) A linguagem SQL para bancos de dados, é constituída das sublinguagens "Data Manipulation Language – DML", "Data Definition Language – DDL" e "Data Control Language – DCL". Fazem parte da DCL os seguintes comandos:

a) SELECT e DROP
b) DROP e GRANT
c) GRANT e REVOKE
d) REVOKE e DELETE
e) DELETE e SELECT

138. (IBFC / EBSERH – 2013) Relacione os comandos típicos utilizados em cada Linguagem conforme tabela abaixo:

(A) UPDATE
(B) GRANT
(C) CREATE

(D) Linguagem de definição de dados (DDL)
(E) Linguagem de manipulação de dados (DML)
(F) Linguagem de controle de dados (DCL)

a) AD - BE - CF
b) AD - BF - CE
c) AE - BD - CF
d) AE - BF - CD

139. (IBFC / PC-RJ – 2013) A "linguagem" SQL (Structured Query Language) revolucionou a forma em que os programadores extraem informações de um Banco de Dados. Um perito com sólidos conhecimentos de SQL pode obter em frações de segundos uma grande massa de dados e posterior processamento visando a extração de algum dado ou sequência de dados, como por exemplo possíveis suspeitos em uma amostra populacional de uma cidade. São quatro os principais comandos do DML (Data Manipulation Language) do SQL. Identifique a alternativa que NÃO é um desses comandos:

a) INSERT
b) SELECT
c) UPDATE
d) DELETE

e) CREATE.

140. (QUADRIX / CRQ-SP – 2013) Assinale a alternativa com o uso correto do comando Insert do SQL.

a)
INSERT (CODIGO, NOME, SALARIO, SECAO)
VALUES(1, "LUCKY LUCIANO", 120, 1)
INTO EMPREGADOS

b)
INSERT INTO EMPREGADOS
VALUES(1, "LUCKY LUCIANO", 120, 1)
(CODIGO, NOME, SALARIO, SECAO)

c)
INSERT INTO EMPREGADOS(CODIGO, NOME, SALARIO, SECAO)

d)
INSERT INTO EMPREGADOS(CODIGO, NOME, SALARIO, SECAO)
VALUES(1, "LUCKY LUCIANO", 120, 1)

e)
INSERT VALUES(1, "LUCKY LUCIANO", 120, 1)
INTO EMPREGADOS

141. (ESPP / COBRA – 2013) Em SQL se um empregado for promovido a gerente devemos alterar o cargo (CARGO) do empregado número (NUM_EMP) '48729' na tabela EMPREGADOS para o valor GERENTE, usando a seguinte query:

a) UPDATE GERENTE WHERE NUM_EMP = '48729';
b) UPDATE CARGO = GERENTE WHERE NUM_EMP = '48729';
c) UPDATE EMPREGADOS SET CARGO = GERENTE WHERE NUM_EMP = '48729';
d) UPDATE EMPREGADOS SET NUM_EMP = '48729';

142. (ESAF / Ministério da Fazenda – 2013) As três cláusulas de uma consulta SQL são:

a) start, from, to.
b) select, from, where.
c) select, up, what.

d) start, from, who.
e) select, initial, final.

143. (CESGRANRIO / EPE – 2012) A DDL (Data Definition Language) ou linguagem de definição de dados é um conjunto de comandos SQL responsável pela definição das estruturas de dados em um SGBD. Qual comando faz parte da DDL?

a) Select
b) Update
c) Create
d) Delete
e) Insert

144. (CONSULPLAN / TSE – 2012) Ao contrário das linguagens tradicionais, que são procedimentais, SQL é uma linguagem declarativa, que integra três sublinguagens: Data Manipulation Language (DML), Data Definition Language (DDL) e Data Control Language (DCL). Um comando DML e outro DDL são, respectivamente,

a) Drop e Grant.
b) Grant e Delete.
c) Delete e Update.
d) Update e Drop.

145. (FUMARC / TJ-MG – 2012) A linguagem SQL possui comandos de definição de dados (DDL - Data Defnition Language), dos quais faz parte o seguinte comando:

a) SELECT
b) DELETE
c) ALTER
d) UPDATE

146. (CEPERJ / PROCON-RJ – 2012) SQL representa uma linguagem declarativa, não procedural, que permite interação com bancos de dados, sendo constituída de três sublinguagens, a Data Manipulation Language (DML), a Data Definition Language (DDL) e a Data Control Language (DCL). Como comandos DCL, um permite conceder determinado privilégio a um usuário e outro permite retirar o privilégio concedido. Esses comandos são, respectivamente:

a) GET e PUT
b) CREATE e DROP
c) SELECT e DELETE
d) GRANT e REVOKE
e) INSERT e REMOVE

147. (IESES / CRF-SC – 2012) Relacione a primeira coluna com a segunda e em seguida identifique a alternativa que apresenta a ordem correta dos números de cima para baixo:

(1) – DDL ( ) - É um subconjunto de comandos SQL que serve para a definição das estruturas de dados de um banco de dados, como por exemplo, criar tabelas, índices, views, etc.

(2) – DML ( ) - É um subconjunto de comandos SQL que permite a DBAs controlar o acesso aos dados de um banco de dados.

(3) – DCL ( ) - É um subconjunto de comandos SQL que serve para acesso, inclusão, alteração e exclusão dos dados de um banco de dados.

a) 1 – 3 – 2
b) 3 – 1 – 2
c) 1 – 2 – 3
d) 2 – 3 – 1

148. (FUNCAB / MPE-RO – 2012) A cláusula FROM de comando SELECT é utilizada para especificar:

a) a ordem de seleção das linhas da tabela.
b) a condição de seleção das linhas da tabela.
c) os campos cujas informações deverão ser selecionadas.
d) as tabelas das quais as informações serão selecionadas.
e) o agrupamento das informações selecionadas.

149. (FMP / TCE-RS – 2011) Qual comando SQL é utilizado para obter um conjunto de dados em uma tabela em um banco de dados?

a) INSERT
b) UPDATE
c) JOIN
d) SELECT
e) GET

150. (Instituto Ânima / Companhia Águas de Joinville – 2010) Para manipular dados em um banco de dados, usamos uma linguagem de consulta estruturada. A SQL (Structured Query Language) é a linguagem usada pela maioria dos bancos de dados. Esta é composta de três outras linguagens, quais são elas?

a) DML, DDL e DLL.

b) DML, DDL e DCL.
c) MLL, DLL e CLL.
d) SQL, LQL e CQL.
e) DDL, DCL e DGL

151. (IF-RJ / IF-RJ– 2010) A linguagem SQL possui sublinguagens. Dentre elas, destacamos a DDL, de definição de dados; a DML, de manipulação de dados e a DCL, de controle de dados.

Marque a alternativa que possui um comando de cada sublinguagem.

a) Create, Drop, Select
b) Create, Insert, Alter
c) Select, Insert, Update
d) Insert, Revoke, Update
e) Create, Delete, Grant

152. (NCE-UFRJ / UFRJ – 2009) Um sistema que suporta o processamento de transações, garante que se a transação executar algumas atualizações e então ocorrer uma falha antes que esta alcance seu término normal, aquelas atualizações não serão feitas. Consequentemente a atualização é executada na sua totalidade ou cancelada. Os comandos em SQL, usados para desfazer uma transação mal sucedida e confirmar uma bem sucedida, são, respectivamente:

a) COMMIT e ROLLBACK;
b) GRANT e REVOKE;
c) ROLLBACK e COMMIT;
d) REVOKE e GRANT;
e) ROLLBACK e GRANT.

153. (CESGRANRIO / Casa da Moeda – 2009) Um administrador de dados de uma empresa deve, excepcionalmente, atualizar o endereço de um funcionário registrado em uma tabela do banco de dados, que não guarda histórico e registra somente o endereço atual em uma única linha. Para a atualização dos dados, que comando SQL deverá ser utilizado?

a) SELECT.
b) CHANGE.
c) INSERT.
d) UPDATE.
e) DELETE.

154. (ESAF / UFSJ – 2005) Com relação ao uso da SQL na manipulação de dados, caso se queira eliminar linhas repetidas do conjunto resultado, deve-se utilizar a palavra-chave DISTINCT, da seguinte forma:

a) SELECT DISTINCT {colunas} FROM {tabelas}.
b) DISTINCT SELECT {colunas} FROM {tabelas}.
c) SELECT {colunas} FROM {tabelas} DISTINCT.
d) SELECT FROM {tabelas} DISTINCT {colunas}.

GABARITO

1. LETRA E
2. ERRADO
3. ERRADO
4. CORRETO
5. ERRADO
6. LETRA C
7. LETRA B
8. LETRA D
9. LETRA A
10. CORRETO
11. ERRADO
12. LETRA C
13. ERRADO
14. CORRETO
15. ERRADO
16. CORRETO
17. ERRADO
18. LETRA A
19. LETRA B
20. LETRA A
21. LETRA B
22. LETRA D
23. LETRA A
24. LETRA D
25. LETRA D
26. LETRA B
27. LETRA E
28. LETRA B
29. LETRA D
30. LETRA E
31. LETRA D
32. LETRA C
33. LETRA C
34. LETRA C
35. LETRA A
36. LETRA D
37. LETRA A
38. LETRA E
39. LETRA E
40. LETRA D
41. LETRA B
42. LETRA D
43. LETRA B
44. LETRA D
45. LETRA D
46. LETRA C
47. LETRA A
48. LETRA B
49. LETRA C
50. LETRA B
51. LETRA E
52. LETRA D
53. LETRA D
54. LETRA E
55. LETRA B
56. LETRA D
57. LETRA A
58. LETRA C
59. LETRA B
60. LETRA C
61. LETRA B
62. LETRA C
63. LETRA A
64. LETRA E
65. LETRA C
66. LETRA A
67. LETRA D
68. LETRA E
69. LETRA A
70. LETRA A
71. LETRA C
72. LETRA B
73. LETRA A
74. LETRA A
75. LETRA B
76. LETRA A
77. LETRA A
78. LETRA A
79. LETRA B
80. LETRA A
81. LETRA B
82. LETRA D
83. LETRA A
84. LETRA E
85. LETRA D
86. LETRA D
87. LETRA D
88. LETRA C
89. LETRA E
90. LETRA C
91. LETRA A
92. LETRA B
93. LETRA A
94. LETRA C
95. LETRA A
96. LETRA B
97. LETRA C
98. CORRETO
99. ERRADO
100. CORRETO
101. ERRADO
102. LETRA C
103. LETRA E
104. LETRA C
105. LETRA A
106. LETRA E
107. LETRA A
108. CORRETO
109. ERRADO
110. CORRETO
111. LETRA C
112. LETRA D
113. LETRA B
114. LETRA D
115. LETRA C
116. LETRA A
117. LETRA C
118. LETRA C
119. LETRA A
120. LETRA D

121. LETRA B
122. LETRA E
123. LETRA E
124. LETRA B
125. LETRA B
126. LETRA B
127. LETRA C
128. LETRA E
129. LETRA B
130. LETRA C
131. LETRA D
132. LETRA B
133. LETRA A
134. ERRADO
135. LETRA A
136. LETRA A
137. LETRA C
138. ERRADO
139. LETRA E
140. LETRA D
141. LETRA C
142. LETRA B
143. LETRA C
144. LETRA D
145. LETRA C
146. LETRA D
147. LETRA A
148. LETRA D
149. LETRA D
150. LETRA B
151. LETRA E
152. LETRA C
153. LETRA D
154. LETRA A

ESSA LEI TODO MUNDO CONHECE:
PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.

1. 

Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.

2. 

Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).

3. 

Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).

4. 

Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).

5. 

Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).

6. 

Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).

7. 

Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.

8. 

O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.

Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.