PROVAS DE TI
TUDO PARA VOCÊ PASSAR

BANCO DE DADOS
SQL DDL

Prof. Eduardo Neves
edumneves@gmail.com
http://www.itnerante.com.br/profile/EduardoNeves

Dúvidas de Banco de Dados:
http://www.itnerante.com.br/group/bancodedados

Artigos do Blog:
http://goo.gl/Sm6JgV

Qconcursos – estatística

	CESPE	FCC	CESGRANRIO	OUTRAS	TOTAL
SQL	173	131	66	153	523
Data warehouse	96	70	54	50	270
Modelagem	85	62	28	65	240
Conceitos Básicos	112	41	20	62	235
Formas Normais	69	22	27	71	189
Transações	24	18	22	22	86

BD – Distribuição das aulas

- Aula 01
  - Introdução
  - Linguagens – DML, DDL, DCL, TCL
- Aula 02
  - Create Table
- Aula 03
  - Contraints – Primary key, foreign Key, ...
- Aula 04
  - Alter Table
- Aula 05
  - Foreign Key – On UPDATE, On Delete, cascade

BD – Distribuição das aulas

- Aula 06
  - Restrições de Integridade Adiáveis
- Aula 07, 08 e 09
  - Visões
  - Visões atualizáveis
- Aula 10, 11 e 12
  - Exercícios

- 36 Questões ☺

Gostou? Avalie o curso ☺

Excelente **Comentado por DHEISON**

Qualidade ★★★★★
Preço ★★★★★
Conteúdo ★★★★★
Professor ★★★★★

Após ver aulas de 3 professores diferentes tava começando a achar que o problema era comigo, mas finalmente uma boa didática para entender 100% o conteúdo proposto, recomendado! (Postado em 24/02/16)

Excelente Curso de BD **Comentado por RAFAEL NOGUEIRA**

Qualidade ★★★★★
Preço ★★★★★
Conteúdo ★★★★★
Professor ★★★★★

Curso muito didático com explicações na medida! :)
Melhor explicação que ja vi sobre conceitos confusos e abstratos como o ansi-spark e DBA vs AD!

5 Estrelas fácil! (Postado em 03/02/16)

Muito bom! **Comentado por Leonardo**

Qualidade ★★★★★
Preço ★★★★★
Conteúdo ★★★★★
Professor ★★★★★

Excelente Professor!

Teoria na medida certa e muitos exercícios!

Só em Arquitetura ANSI/SPARC foram mais de 30 exercícios!

Recomendado!

Grande Abraço! (Postado em 03/11/15)

Apresentação

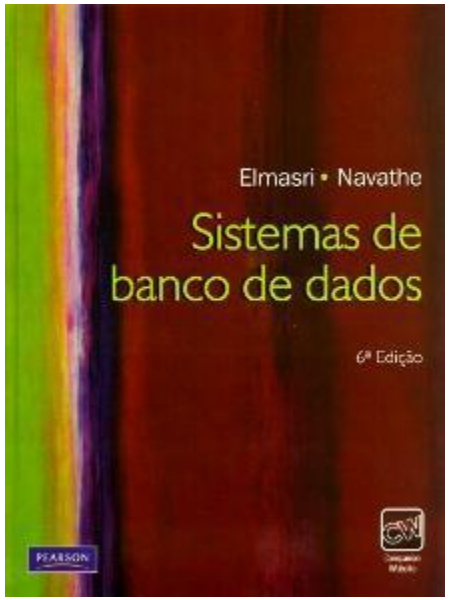
- Formação
  - Bacharel em Ciência da Computação/UFRJ
  - Pós-graduação – Gestão de Projetos
- Analista de Sistemas – BNDES – 2013
- Principais aprovações
  - 4º – BNDES/2012 – CESGRANRIO
  - 19º – BNDES/2011 – CESGRANRIO
  - 30º – Petrobras/ 2011 – CESGRANRIO
  - 12º – FINEP/2011 – CESGRANRIO
  - 3º – Transpetro/2011 – CESGRANRIO
  - 42º – TRT-RJ/2011 – FCC
  - 7º – Petrobras Macaé/2010 – CESGRANRIO
  - 1º – Caixa Econômica Federal/2010 – Nível médio informática – CESPE
  - 18º – BR Distribuidora/2010 – Analista SAP – CESGRANRIO

Dicas de estudo Gerais

- **NÃO DESISTIR!!!!**
- Aprender com os próprios erros
- Estudar as matérias em ciclos
  - http://suficienciacontabil.com.br/wp-content/uploads/2014/12/ciclos-de-estudo-alexandre-meirelles.pdf
- Ter uma forma de revisão
  - Anki
    - http://www.itnerante.com.br/profiles/blogs/deck-anki-do-edu-para-o-bndes-2013-4-lugar
  - Mapa Mental
  - Resumos
- Fazer muitos exercícios
  - Da mesma banca primeiro da mais recente para a mais antiga
  - De outras bancas
  - Usar sites de questões
    - (qconcursos.com, mapadaprova, tecconcursos, ...)
- Timasters
  - Tirar dúvidas, compartilhar conhecimento.

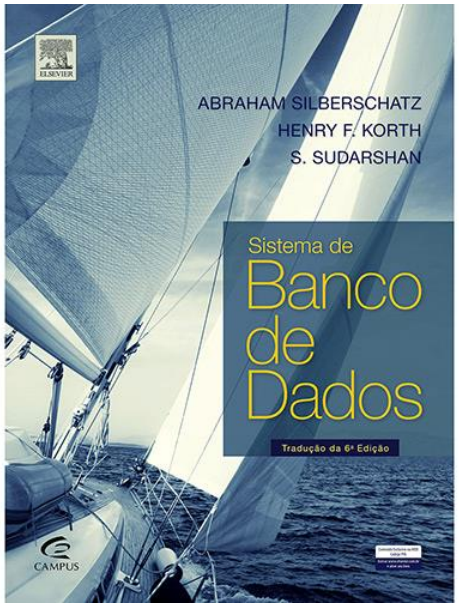
Bibliografia

Sistemas de Banco de Dados – 6ª edição
Ramez E. Elmasri, Shamkant B. **Navathe**
Editora Pearson, 2011



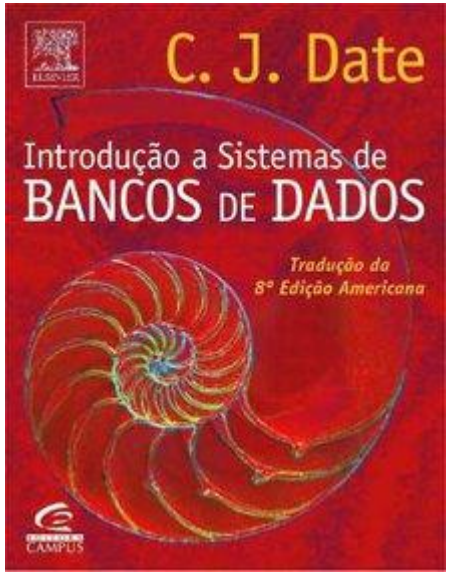
Bibliografia

Sistemas de Banco de Dados – 6ª edição
Abraham **Silberschatz**, Henry F. **Korth**
Editora Campus, 2012



Bibliografia

Introdução a Sistemas de Banco de Dados
C.J. **Date**
Editora Campus, 2004



Histórico

- Linguagem SQUEL desenvolvida pela IBM para um banco de dados experimental R
- Baseada no padrão ANSI e ISO:
  - SQL-86
  - SQL-89
  - SQL-92
  - SQL:1999
  - SQL:2003
- A maioria dos SGBD comerciais suportam o SQL-92, e algumas das caracteristicas das últimas versões

Linguagens de SGBD

- **DDL** – Data Definition Language (Linguagem de Definição de Dados) são usadas para definir a estrutura de banco de dados ou esquema.
  - CREATE- para criar objetos no banco de dados
  - ALTER – altera a estrutura da base de dados
  - DROP – apaga objetos no banco de dados
  - TRUNCATE – remover todos os registros de uma tabela, incluindo todos os espaços alocados para os registros são removidos. Não tem rollback.
  - COMMENT – adicionar comentários ao dicionário de dados
  - RENAME – para renomear um objeto

Linguagens de SGBD

- **DML** – Data Manipulation Language (Linguagem de Manipulação de Dados) são utilizados para o gerenciamento de dados dentro de objetos do banco.
  - SELECT– recuperar dados do banco de dados
  - INSERT – inserir dados em uma tabela
  - UPDATE – atualiza os dados existentes em uma tabela
  - DELETE – exclui registros de uma tabela,
  - CALL – chamar um subprograma PL / SQL
  - EXPLAIN PLAN – explicar o caminho de acesso aos dados
  - LOCK TABLE – controle de concorrência

Linguagens de SGBD

- **TCL** – Transaction Control Language – (Controle de Transações) são usados para gerenciar as mudanças feitas por instruções DML . Ele permite que as declarações a serem agrupadas em transações lógicas .
  - COMMIT – salvar o trabalho feito
  - SAVEPOINT – identificar um ponto em uma transação para que mais tarde você pode efetuar um ROLLBACK
  - ROLLBACK – restaurar banco de dados ao original desde o último COMMIT

Linguagens de SGBD

- **DCL** – Data Control Language (Linguagem de Controle de dados) declarações.
  - GRANT – atribui privilégios de acesso do usuário a objetos do banco de dados
  - REVOKE – remove os privilégios de acesso aos objetos obtidos com o comando GRANT

- **DQL** – *Data Query Language* (Linguagem de Consulta de Dados)
  - SELECT– recuperar dados do banco de dados

  Obs: A maioria das literaturas considera o SELECT como DML, mas existem algumas que o consideram DQL.

Linguagens de SGBD

- **DDL** – Data Definition Language:
  - utilizada para definir os *schemas* conceitual e interno (níveis lógico e físico), em SGBDs em que não há uma separação estrita entre estes dois níveis.
- **SDL** – Storage Definition Language:
  - usada para especificar o esquema interno.
- **VDL** – View Definition Language:
  - usada para especificar as visões dos usuários e o seu mapeamento no esquema conceitual.

CESGRANRIO – EPE – Analista de Gestão Corporativa – 2012

Q1 – 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

CESGRANRIO – EPE – Analista de Gestão Corporativa – 2012

Q1 – 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

Q2 – 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.

Q2 – 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.

Q3 - CESGRANRIO - IBGE - Tecnologia da Informação - 2014

SQL é uma linguagem dedicada à operação de Bancos de Dados relacionais, padronizada internacionalmente, e que pode ser encontrada nos principais SGBD modernos.

Os principais comandos da sua linguagem de manipulação de dados (DML) são:

(A) ALTER, CREATE e DROP
(B) CREATE, DELETE, READ e UPDATE
(C) CREATE, DESTROY, FIND e INCLUDE
(D) SELECT, DELETE, INSERT e UPDATE
(E) SELECT, JOIN, PROJECT e RENAME

Q3 – CESGRANRIO – IBGE – Tecnologia da Informação – 2014

SQL é uma linguagem dedicada à operação de Bancos de Dados relacionais, padronizada internacionalmente, e que pode ser encontrada nos principais SGBD modernos.

Os principais comandos da sua linguagem de manipulação de dados (DML) são:

(A) ALTER, CREATE e DROP
(B) CREATE, DELETE, READ e UPDATE
(C) CREATE, DESTROY, FIND e INCLUDE
(D) SELECT, DELETE, INSERT e UPDATE
(E) SELECT, JOIN, PROJECT e RENAME

Q4 - BIO-RIO - RJ-RJ - Desenvolvimento - 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.

Q4 – BIO-RIO – RJ-RJ – Desenvolvimento – 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.

Linguagem de Manipulação de Dados (DML)

- A linguagem de manipulação dos dados permite ao usuário manipular os dados da seguinte forma:
- Procedural:
  - o usuário informa qual dado deseja acessar e como obtê-lo
- Não-procedural:
  - o usuário informa qual dado deseja acessar SEM especificar como obtê-lo

Linguagem de Manipulação de Dados (DML)

- Linguagens não-procedurais são usualmente mais fáceis de aprender e usar do que DMLs procedurais
- Se o usuário NÃO especificar COMO obter os dados, as linguagens não-procedurais poderão gerar um código não tão eficiente.

Q5 – ESAF – MF – AFO – 2013

Assinale a opção correta.

a) A DML procedural requer a especificação de que dados são necessários e a forma de obtê-los.
b) A DML associativa requer apenas a especificação de que dados são necessários.
c) A DML funcional requer a especificação de que dados são necessários e a forma de obtê-los.
d) A DML declarativa requer a especificação de que dados são necessários e a forma de obtê-los.
e) A DML procedural requer apenas a especificação de que dados são necessários.

Q5 – ESAF – MF – AFO – 2013

Assinale a opção correta.

a) A DML procedural requer a especificação de que dados são necessários e a forma de obtê-los.
b) A DML associativa requer apenas a especificação de que dados são necessários.
c) A DML funcional requer a especificação de que dados são necessários e a forma de obtê-los.
d) A DML declarativa requer a especificação de que dados são necessários e a forma de obtê-los.
e) A DML procedural requer apenas a especificação de que dados são necessários.

Q6 – FCC – MANAUSPREV – An. Previdênciário – 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.

Q6 – FCC – MANAUSPREV – An. Previdênciário – 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.

Q7 – CESPE – MPU – Suporte – 2013

Julgue os itens subsequentes, acerca dos conceitos relacionados a bancos de dados.

As DML (linguagens de manipulação de dados) procedurais normalmente geram códigos mais eficientes do que as DML não procedurais.

Q7 – CESPE – MPU – Suporte – 2013

Julgue os itens subsequentes, acerca dos conceitos relacionados a bancos de dados.

C As DML (linguagens de manipulação de dados) procedurais normalmente geram códigos mais eficientes do que as DML não procedurais.

Q8 – CESPE – MEC – Desenvolvedor – 2015

Com relação à linguagem de definição de dados (DDL) e à linguagem de manipulação de dados (DML), julgue o próximo item.

A DML utiliza o comando CREATE para inserir um novo registro na tabela de dados.

Q8 – CESPE – MEC – Desenvolvedor – 2015

Com relação à linguagem de definição de dados (DDL) e à linguagem de manipulação de dados (DML), julgue o próximo item.

E A DML utiliza o comando CREATE para inserir um novo registro na tabela de dados.

Q9 – CESPE – SUFRAMA – Tecn. Informação – 2014

Com relação aos sistemas gerenciadores de banco de dados (SGBD), julgue os itens a seguir.

Para controlar os dados de um banco de dados, com o objetivo de manipulá-los adequadamente em operações de atualização e consulta, utilizam-se os comandos grant e revoke da DML (Data Manipulation Language).

Q9 – CESPE – SUFRAMA – Tecn. Informação – 2014

Com relação aos sistemas gerenciadores de banco de dados (SGBD), julgue os itens a seguir.

E Para controlar os dados de um banco de dados, com o objetivo de manipulá-los adequadamente em operações de atualização e consulta, utilizam-se os comandos grant e revoke da DML (Data Manipulation Language).

Q10 – CESPE – ANTT – An. Administrativo – 2013

Julgue os itens subsequentes, relativos a banco de dados.

Existem várias categorias de linguagens de SGBD: a de definição de dados (DDL), usada para especificar esquemas (conceitual e externo); a de manipulação de dados (DML), que provê um conjunto de operações básicas para manipulação de dados; a de definição de visões (VDL), utilizada para especificar a visão do usuário (esquema externo) e seu mapeamento para o esquema conceitual e a de definição de armazenamento (SDL), usada para especificar o esquema interno de armazenamento dos dados no banco.

Q10 – CESPE – ANTT – An. Administrativo – 2013

Julgue os itens subsequentes, relativos a banco de dados.

C Existem várias categorias de linguagens de SGBD: a de definição de dados (DDL), usada para especificar esquemas (conceitual e externo); a de manipulação de dados (DML), que provê um conjunto de operações básicas para manipulação de dados; a de definição de visões (VDL), utilizada para especificar a visão do usuário (esquema externo) e seu mapeamento para o esquema conceitual e a de definição de armazenamento (SDL), usada para especificar o esquema interno de armazenamento dos dados no banco.

Q11 – FGV – TCMSP – Analista de Sistemas – 2015

Analise os três comandos a seguir e as afirmativas a respeito de seus efeitos no âmbito do MS SQL Server.

delete from x

truncate table x

drop table x

I. O comando *delete* e o comando *truncate* removem o mesmo conjunto de registros da tabela X.

II. O comando *drop*, quando usado com a opção "with no removal", produz exatamente o mesmo efeito do comando *truncate*.

III. Devido às suas características operacionais, o comando *delete* é usualmente executado muito mais rapidamente que o comando *truncate*.

Está correto o que se afirma em:

(A) somente I;

(B) somente III;

(C) somente I e II;

(D) somente II e III;

(E) I, II e III.

Q11 – FGV – TCMSP – Analista de Sistemas – 2015

Analise os três comandos a seguir e as afirmativas a respeito de seus efeitos no âmbito do MS SQL Server.

delete from x

truncate table x

drop table x

I. O comando *delete* e o comando *truncate* removem o mesmo conjunto de registros da tabela X.

II. O comando *drop*, quando usado com a opção "with no removal", produz exatamente o mesmo efeito do comando *truncate*.

III. Devido às suas características operacionais, o comando *delete* é usualmente executado muito mais rapidamente que o comando *truncate*.

Está correto o que se afirma em:

(A) somente I;

(B) somente III;

(C) somente I e II;

(D) somente II e III;

(E) I, II e III.

Tipos de Dados

Tipo de Dado	Descrição
CHARACTER	Caractere de tamanho fixo – CHAR.
CHARACTER VARYING	Caractere de tamanho variável – VARCHAR.
CHARACTER LARGE OBJECT	Caractere longo – CLOB.
BINARY LARGE OBJECT	Binário para objetos longos – BLOB.
NUMERIC	Numérico exato.
DECIMAL	Numérico exato.
SMALLINT	Numérico exato.
INTEGER	Numérico exato.
BIGINT	Numérico exato.
FLOAT	Numérico aproximado.
REAL	Numérico aproximado.
DOUBLE PRECISION	Numérico aproximado.
BOOLEAN	Booleano.
DATE	Data com dia, mês e ano.
TIME	Horário com hora, minuto e segundos.
TIMESTAMP	Momento com ano, mês, dia, hora, minuto e segundo.

DDL – Data Definition Language

- Criação de Tabelas:

CREATE TABLE <nome_tabela>(
<campo$_1$> <tipo> [NULL|NOT NULL] [restricao],
[...,
<campo$_n$> <tipo> [NULL|NOT NULL] [restricao],
PRIMARY KEY <chave_primaria>])

DDL – Data Definition Language

- Na criação de tabelas, é possível especificar vários tipos de restrições:
  - Chave Primária: PRIMARY KEY ;
  - Chave Estrangeira: FOREIGN KEY;
  - Chave Alternativa (ou alternada):
    - UNIQUE NOT NULL;
  - Única sem repetição: UNIQUE;
  - Restrição de Domínio: CHECK.

- Pode-se atribuir nomes às restrições de integridade:
  - CONSTRAINT NOME_RESTRIÇÃO TIPO RESTRIÇÃO.

Taxi(placa, chassi, marca, modelo, anoFab)

Cliente(cliid, nome, cpf, sexo)

Corrida(cliid, placa, dataPedido)

cliid referencia Cliente

placa referencia Taxi

```
CREATE TABLE Taxi (
   Placa VARCHAR(7) NOT NULL,
   Chassi VARCHAR(20) NOT NULL UNIQUE,
   Marca VARCHAR(30) NOT NULL,
   Modelo VARCHAR(30) NOT NULL,
   AnoFab INTEGER,
PRIMARY KEY(Placa));
```

```
CREATE TABLE Taxi (
   Placa VARCHAR(7) NOT NULL PRIMARY KEY,
   Chassi VARCHAR(20) NOT NULL UNIQUE,
   Marca VARCHAR(30) NOT NULL,
   Modelo VARCHAR(30) NOT NULL,
   AnoFab INTEGER,
);
```

Taxi(placa, chassi, marca, modelo, anoFab)
Cliente(cliid, nome, cpf, sexo)
Corrida(cliid, placa, dataPedido)
cliid referencia Cliente
placa referencia Taxi

```
CREATE TABLE Cliente (
    CliId VARCHAR(4) NOT NULL,
    Nome VARCHAR(80) NOT NULL,
    CPF VARCHAR(14) NOT NULL,
    Sexo VARCHAR(1) DEFAULT 'M' CHECK (Sexo IN ('M', 'F')),
CONSTRAINT PK_CLIENTE PRIMARY KEY(CliId),
CONSTRAINT AK_CPF UNIQUE (CPF)
);
```

DDL – Data Definition Language

- CREATE TABLE <tabela>

...

FOREIGN KEY (<coluna_estr>$_1$[,...,<coluna_estr>$_n$])

REFERENCES <tabela_ref>([<coluna_ref>[,...,<coluna_ref]])

Taxi(placa, chassi, marca, modelo, anoFab)
Cliente(cliid, nome, cpf, sexo)
Corrida(cliid, placa, dataPedido)
cliid referencia Cliente
placa referencia Taxi

```
CREATE TABLE Corrida(
    CliId VARCHAR(4) NOT NULL,
    Placa VARCHAR(7) NOT NULL,
    dataPedido DATETIME NOT NULL,
CONSTRAINT PK_CORRIDA PRIMARY KEY(CliId, Placa, dataPedido),
FOREIGN KEY (Cliid) REFERENCES Cliente (Cliid),
CONSTRAINT FK_TAXI
        FOREIGN KEY (Placa) REFERENCES Taxi(Placa)
);
```

Taxi(placa, chassi, marca, modelo, anoFab)
Cliente(cliid, nome, cpf, sexo)
Corrida(cliid, chassi, dataPedido)
cliid referencia Cliente
chassi referencia Taxi

```
CREATE TABLE Corrida(
    CliId VARCHAR(4) NOT NULL,
    Chassi VARCHAR(20) NOT NULL,
    dataPedido DATETIME NOT NULL,
CONSTRAINT PK_CORRIDA PRIMARY KEY(CliId, Chassi, dataPedido),
FOREIGN KEY (Cliid) REFERENCES Cliente (Cliid),
CONSTRAINT FK_TAXI
        FOREIGN KEY (Chassi) REFERENCES Taxi(Chassi)
);
```

Chave estrangeira pode ser para chave candidata.

Taxi(placa, chassi, marca, modelo, anoFab)
Cliente(cliid, nome, cpf, sexo)
Corrida(cliid, chassiTaxi, dataPedido)
cliid referencia Cliente
chassiTaxi referencia Taxi

```
CREATE TABLE Corrida(
    ClIId VARCHAR(4) NOT NULL,
    ChassiTaxi VARCHAR(20) NOT NULL,
    dataPedido DATETIME NOT NULL,
CONSTRAINT PK_CORRIDA PRIMARY KEY(ClIId, ChassiTaxi, dataPedido),
FOREIGN KEY (Cliid) REFERENCES Cliente (Cliid),
CONSTRAINT FK_TAXI
        FOREIGN KEY (ChassiTaxi) REFERENCES Taxi(Chassi)
);
```

Não precisa ser o mesmo nome da coluna.

```
CREATE TABLE Cliente (
   CliId INTEGER NOT NULL,
   Nome VARCHAR(80) NOT NULL,
   CPF VARCHAR(14) NOT NULL,
   Sexo VARCHAR(1) DEFAULT 'M'
            CHECK (Sexo IN ('M', 'F')),
CONSTRAINT PK_CLIENTE PRIMARY KEY(CliId),
CONSTRAINT AK_CPF UNIQUE (CPF)
);
```

Taxi(placa, chassi, marca, modelo, anoFab)
Cliente(cliid, nome, cpf, sexo)
Corrida(cliid, chassi, dataPedido)
cliid referencia Cliente
chassi referencia Taxi

```
CREATE TABLE Corrida(
   CliId INTEGER NOT NULL,
   ChassiTaxi VARCHAR(20) NOT NULL,
   dataPedido DATETIME NOT NULL,
CONSTRAINT PK_CORRIDA PRIMARY KEY(CliId, ChassiTaxi, dataPedido),
FOREIGN KEY (Cliid) REFERENCES Cliente (CPF),
CONSTRAINT FK_TAXI
            FOREIGN KEY (ChassiTaxi) REFERENCES Taxi(Chassi)
);
```

ERRO!!!! A chave estrangeira tem que ter o mesmo número de Colunas e ter os tipos de dados compatíveis.

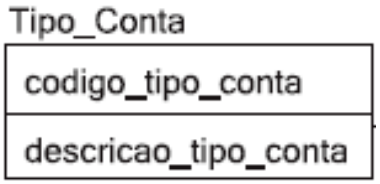
```
CREATE TABLE Cliente (
   CliId VARCHAR(4) NOT NULL,
   Nome VARCHAR(80) NOT NULL,
   CPF VARCHAR(14) NOT NULL,
   Sexo VARCHAR(1) DEFAULT 'M'
            CHECK (Sexo IN ('M', 'F')),
CONSTRAINT PK_CLIENTE PRIMARY KEY(CliId),
CONSTRAINT AK_CPF UNIQUE (CPF)
);
```

Taxi(placa, chassi, marca, modelo, anoFab)
Cliente(cliid, nome, cpf, sexo)
Corrida(cliid, chassi, dataPedido)
cliid referencia Cliente
chassi referencia Taxi

```
CREATE TABLE Corrida(
   CliId VARCHAR(4) NOT NULL,
   ChassiTaxi VARCHAR(20) NOT NULL,
   dataPedido DATETIME NOT NULL,
CONSTRAINT PK_CORRIDA PRIMARY KEY(CliId, ChassiTaxi, dataPedido),
FOREIGN KEY (Cliid) REFERENCES Cliente (CPF),
CONSTRAINT FK_TAXI
            FOREIGN KEY (ChassiTaxi) REFERENCES Taxi(Chassi)
);
```

ERRO!!!! Depende do SGBD, mysql permite,
SQL Server Não permite.

Q12 – CESGRANRIO – Banco da Amazônia – Banco de Dados – 2014



Que comando SQL deve ser dado para criar a Tabela Tipo_Conta?

(A) CREATE TABLE Tipo_Conta (codigo_tipo_conta NUMERIC PRIMARY KEY, descricao_tipo_conta VARCHAR(256))

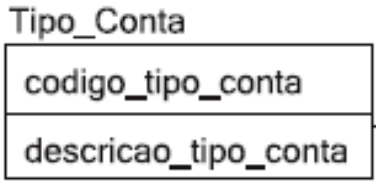
(B) CREATE TABLE Tipo_Conta (codigo_tipo_conta:NUMERIC PRIMARY KEY, descricao_tipo_conta:VARCHAR(256))

(C) CREATE TABLE Tipo_Conta WITH COLUMNS (codigo_tipo_conta NUMERIC PRIMARY KEY, descricao_tipo_conta VARCHAR(256))

(D) CREATE Tipo_Conta AS TABLE (codigo_tipo_conta NUMERIC PRIMARY KEY, descricao_tipo_conta VARCHAR(256))

(E) CREATE Tipo_Conta AS TABLE WITH COLUMNS (codigo_tipo_conta NUMERIC PRIMARY KEY, descricao_tipo_conta VARCHAR(256))

Q12 – CESGRANRIO – Banco da Amazônia – Banco de Dados – 2014



Que comando SQL deve ser dado para criar a Tabela Tipo_Conta?

(A) CREATE TABLE Tipo_Conta (codigo_tipo_conta NUMERIC PRIMARY KEY, descricao_tipo_conta VARCHAR(256))

(B) CREATE TABLE Tipo_Conta (codigo_tipo_conta:NUMERIC PRIMARY KEY, descricao_tipo_conta:VARCHAR(256))

(C) CREATE TABLE Tipo_Conta WITH COLUMNS (codigo_tipo_conta NUMERIC PRIMARY KEY, descricao_tipo_conta VARCHAR(256))

(D) CREATE Tipo_Conta AS TABLE (codigo_tipo_conta NUMERIC PRIMARY KEY, descricao_tipo_conta VARCHAR(256))

(E) CREATE Tipo_Conta AS TABLE WITH COLUMNS (codigo_tipo_conta NUMERIC PRIMARY KEY, descricao_tipo_conta VARCHAR(256))

Q13 – CESGRANRIO – Liquigás – Banco de Dados – 2012

Qual a expressão em SQL que inclui um atributo ENDERECO do tipo VARCHAR(100) na tabela PESSOA?

(A) ADD COLUMN ENDERECO VARCHAR(100) TO TABLE PESSOA

(B) ALTER DATABASE ADD COLUMN ENDERECO VARCHAR(100) TO PESSOA

(C) ALTER TABLE PESSOA MODIFY COLUMN ENDEREÇO VARCHAR(100)

(D) ALTER TABLE PESSOA ADD COLUMN ENDERECO VARCHAR(100)

(E) MODIFY TABLE PESSOA ADD COLUMN ENDERECO VARCHAR(100)

DDL – Data Definition Language

- Alteração de Tabelas:
  - Incluir, modificar e excluir colunas em uma tabela;
  - Adicionar e excluir restrições em uma tabela.

```
<alter table statement> ::=
       ALTER TABLE <table name> <alter table action>

<alter table action> ::=
         <add column definition>
       | <alter column definition>
       | <drop column definition>
       | <add table constraint definition>
       | <drop table constraint definition>
```

```
CREATE TABLE Cliente (
    CliId VARCHAR(4) NOT NULL,
    Nome VARCHAR(80) NOT NULL,
    CPF VARCHAR(14) NOT NULL,
    Sexo VARCHAR(1) DEFAULT 'M' CHECK (Sexo IN ('M', 'F')),
CONSTRAINT PK_CLIENTE PRIMARY KEY(CliId),
CONSTRAINT AK_CPF UNIQUE (CPF)
);
```

SQL ANSI:

```
ALTER TABLE <table name> ADD [COLUMN] <column definition>;
```

SQL Server, Oracle, MySQL:

```
ALTER TABLE Cliente  ADD Tipo_Cli VARCHAR(2) NOT NULL DEFAULT 'PF';
ALTER TABLE Cliente  ADD (DataNascimento DATETIME, Idade INTEGER);
```

PostgreSQL:

```
ALTER TABLE Cliente  ADD COLUMN Tipo_Cli VARCHAR(2) NOT NULL DEFAULT 'PF';

ALTER TABLE Cliente  ADD COLUMN DataNascimento DATETIME, ADD COLUMN Idade
INTEGER;
```

SQL ANSI:

```
<alter column definition> ::=
       ALTER [ COLUMN ] <column name> <alter column action>

<alter column action> ::=
         <set column default clause>
       | <drop column default clause>
       | <add column scope clause>
       | <drop column scope clause>
```

SQL Server

ALTER TABLE Cliente **ALTER COLUMN** Tipo_Cli **VARCHAR**(4);

MySQL e Oracle (antes da 10G)

ALTER TABLE Cliente **MODIFY COLUMN** Tipo_Cli **VARCHAR**(4);

Oracle (depois da 10G)

ALTER TABLE Cliente **MODIFY** Tipo_Cli **VARCHAR**(4);

PostgreSQL:

ALTER TABLE Cliente **ALTER [COLUMN]** Tipo_Cli **VARCHAR**(4)

```
CREATE TABLE Cliente (
    CliId VARCHAR(4) NOT NULL,
    Nome VARCHAR(80) NOT NULL,
    CPF VARCHAR(14) NOT NULL,
    Sexo VARCHAR(1) DEFAULT 'M' CHECK (Sexo IN ('M', 'F')),
CONSTRAINT PK_CLIENTE PRIMARY KEY(CliId),
CONSTRAINT AK_CPF UNIQUE (CPF)
);

ALTER TABLE Cliente  ADD CONSTRAINT C_Tipo_Cli CHECK (Tipo_Cli in ('PF', 'PJ'));

ALTER TABLE Cliente  DROP CONSTRAINT C_Tipo_Cli;

ALTER TABLE Cliente  DROP COLUMN Tipo_Cli;
```

```
CREATE TABLE Motorista (
    CPF VARCHAR(14) NOT NULL,
    Nome VARCHAR(80) NOT NULL,
    PlacaTaxi VARCHAR(7)
);

ALTER TABLE Motorista  ADD CONSTRAINT PK_MOTORISTA PRIMARY KEY (CPF);

ALTER TABLE Motorista
   ADD CONSTRAINT FK_MOTORISTA_TAXI FOREIGN KEY (PlacaTaxi)
       REFERENCES Taxi (Placa);

ALTER TABLE Motorista DISABLE CONSTRAINT FK_MOTORISTA_TAXI;

ALTER TABLE Motorista ENABLE CONSTRAINT FK_MOTORISTA_TAXI;
```

Q13 – CESGRANRIO – Liquigás – Banco de Dados – 2012

Qual a expressão em SQL que inclui um atributo ENDERECO do tipo VARCHAR(100) na tabela PESSOA?

(A) ADD COLUMN ENDERECO VARCHAR(100) TO TABLE PESSOA

(B) ALTER DATABASE ADD COLUMN ENDERECO VARCHAR(100) TO PESSOA

(C) ALTER TABLE PESSOA MODIFY COLUMN ENDEREÇO VARCHAR(100)

(D) ALTER TABLE PESSOA ADD COLUMN ENDERECO VARCHAR(100)

(E) MODIFY TABLE PESSOA ADD COLUMN ENDERECO VARCHAR(100)

Q13 – CESGRANRIO – Liquigás – Banco de Dados – 2012

Qual a expressão em SQL que inclui um atributo ENDERECO do tipo VARCHAR(100) na tabela PESSOA?

(A) ADD COLUMN ENDERECO VARCHAR(100) TO TABLE PESSOA

(B) ALTER DATABASE ADD COLUMN ENDERECO VARCHAR(100) TO PESSOA

(C) ALTER TABLE PESSOA MODIFY COLUMN ENDEREÇO VARCHAR(100)

(D) ALTER TABLE PESSOA ADD COLUMN ENDERECO VARCHAR(100)

(E) MODIFY TABLE PESSOA ADD COLUMN ENDERECO VARCHAR(100)

Q14 - CESGRANRIO - PETROBRAS - Tecnico de Informática - 2012

Ao implantar um banco de dados modelado segundo a abordagem relacional em um SGDB comercial baseado em SQL, o DBA verificou a necessidade de representar uma relação que estava em seu modelo original.

O comando SQL correto para criar a representação dessa relação em um SGDB é

(A) CREATE RELATION

(B) CREATE TABLE

(C) INSERT RELATION

(D) INSERT TABLE

(E) TABLE CREATE

Q14 - CESGRANRIO - PETROBRAS - Tecnico de Informática - 2012

Ao implantar um banco de dados modelado segundo a abordagem relacional em um SGDB comercial baseado em SQL, o DBA verificou a necessidade de representar uma relação que estava em seu modelo original.

O comando SQL correto para criar a representação dessa relação em um SGDB é

(A) CREATE RELATION

(B) CREATE TABLE

(C) INSERT RELATION

(D) INSERT TABLE

(E) TABLE CREATE

Q15 – CESGRANRIO – Petrobras – Técnico em Informática – 2011

Qual a cláusula da instrução SQL create table que inclui a lista de atributos de uma chave candidata?

(A) applicant key

(B) candidate key

(C) foreign key

(D) primary key

(E) unique key

Q15 – CESGRANRIO – Petrobras – Técnico em Informática – 2011

Qual a cláusula da instrução SQL create table que inclui a lista de atributos de uma chave candidata?

(A) applicant key

(B) candidate key

(C) foreign key

(D) primary key

(E) unique key

DDL – Data Definition Language

- CREATE TABLE <tabela>

...

FOREIGN KEY (<coluna_estr>$_1$[,...,<coluna_estr>$_n$])
REFERENCES
<tabela_ref>([<coluna_ref>[,...,<coluna_ref]])
[ON DELETE <acao_ref>]
[ON UPDATE <acao_ref>]

- <acao_ref>
  - NO ACTION impede → Default. A ação na tabela mestre <tabela_ref>, checagem da integridade é feita "depois" de alterar a tabela
  - RESTRICT impede → a ação na tabela mestre <tabela_ref>, checagem da integridade é feita "antes" de alterar a tabela
  - CASCADE → propaga a ação da tabela mestre
  - SET NULL → valores de referencias alterados para nulo
  - SET DEFAULT → valores de referências alterados para default

Taxi

Placa	Modelo	...
T1	Ka	
T2	Siena	
T3	HB20	

Corrida

Placa	Cliid	DataPedido
T1	C1	01/01/2016
T1	C1	12/02/2016
T2	C2	20/01/2015

```
CREATE TABLE Corrida(
    ClilId VARCHAR(4) NOT NULL,
    Placa VARCHAR(7) NOT NULL,
    dataPedido DATETIME NOT NULL,
CONSTRAINT PK_CORRIDA PRIMARY KEY(ClilId, Placa, dataPedido),
FOREIGN KEY (Cliid) REFERENCES Cliente (Cliid),
CONSTRAINT FK_TAXI
            FOREIGN KEY (Placa) REFERENCES Taxi(Placa)
            ON DELETE RESTRICT
            -- Impede a exclusão antes de tentar apagar
);
```

Taxi

Placa	Modelo	...
T1	Ka	
T2	Siena	
T3	HB20	

Corrida

Placa	Cliid	DataPedido
T1	C1	01/01/2016
T1	C1	12/02/2016
T2	C2	20/01/2015

```
CREATE TABLE Corrida(
    ClilId VARCHAR(4) NOT NULL,
    Placa VARCHAR(7) NOT NULL,
    dataPedido DATETIME NOT NULL,
CONSTRAINT PK_CORRIDA PRIMARY KEY(ClilId, Placa, dataPedido),
FOREIGN KEY (Cliid) REFERENCES Cliente (Cliid),
CONSTRAINT FK_TAXI
            FOREIGN KEY (Placa) REFERENCES Taxi(Placa)
            ON UPDATE CASCADE
            -- Propaga a atualização
);
```

Taxi

Placa	Modelo	...
T1	Ka	
T2	Siena	
T3	HB20	

Corrida

Placa	Cliid	DataPedido
T1	C1	01/01/2016
T1	C1	12/02/2016
T2	C2	20/01/2015

```
CREATE TABLE Corrida(
    ClilId VARCHAR(4) NOT NULL,
    Placa VARCHAR(7) NOT NULL,
    dataPedido DATETIME NOT NULL,
CONSTRAINT PK_CORRIDA PRIMARY KEY(ClilId, Placa, dataPedido),
FOREIGN KEY (Cliid) REFERENCES Cliente (Cliid),
CONSTRAINT FK_TAXI
            FOREIGN KEY (Placa) REFERENCES Taxi(Placa)
            ON DELETE SET NULL
            -- Ao deletar atualiza para nulo
);
```

Não dá erro de sintaxe, ou seja, o SGBD não vai criticar.
Mas se apagar um TAXI sem apagar antes a corrida vai dar erro
Pois vai tentar colocar a Placa na corrida como NULL
mas ela é chave primária.

Q16 – FGV – PROCEMPA – 2014

Com referência ao banco BD_CERVEJA, considere que João, analista da empresa, recebeu a tarefa de fazer a engenharia reversa do *script*, e tentou escrever o que, na sua concepção, poderia ser o *script* de criação da tabela Cliente, mostrado a seguir.

```
CREATE TABLE CLIENTE(
 nomeCliente nvarchar(50) NOT NULL,
 nomeFavorita nvarchar(50) NOT NULL,
 Constraint PK_CLIENTE
 PRIMARY KEY (nomeCliente),
 Constraint FK_Cliente_Cerveja
 FOREIGN KEY (nomeFavorita)
   references CERVEJA (nomeCerveja)
   on delete set null
   on update cascade)
```

CLIENTE

nomeCliente	nomeFavorita
Ana	Stella
Mariana	Original
Pedro	Bohemia
Rafael	NULL
Thiago	Stella

CERVEJA

nomeCerveja
Bohemia
Original
Stella

Quando pediu a opinião de uma colega sobre esse *script*, João recebeu os seguintes comentários:

I. Não é possível que haja uma chave estrangeira definida como João imaginou, pois o atributo que constitui a chave estrangeira obrigatoriamente deveria ser denominado **nomeCerveja**, tal qual o atributo da tabela referenciada.

II. Há incompatibilidade entre a semântica do *script* e a instância apresentada para a tabela.

III. Há incompatibilidade entre a semântica do *script* e a declaração dos atributos da tabela.

```
CREATE TABLE CLIENTE(
 nomeCliente nvarchar(50) NOT NULL,
 nomeFavorita nvarchar(50) NOT NULL,
 Constraint PK_CLIENTE
 PRIMARY KEY (nomeCliente),
 Constraint FK_Cliente_Cerveja
 FOREIGN KEY (nomeFavorita)
   references CERVEJA (nomeCerveja)
   on delete set null
   on update cascade)
```

CLIENTE

nomeCliente	nomeFavorita
Ana	Stella
Mariana	Original
Pedro	Bohemia
Rafael	NULL
Thiago	Stella

CERVEJA

nomeCerveja
Bohemia
Original
Stella

I. Não é possível que haja uma chave estrangeira definida como João imaginou, pois o atributo que constitui a chave estrangeira obrigatoriamente deveria ser denominado **nomeCerveja**, tal qual o atributo da tabela referenciada.
II. Há incompatibilidade entre a semântica do *script* e a instância apresentada para a tabela.
III. Há incompatibilidade entre a semântica do *script* e a declaração dos atributos da tabela.

(A) II, apenas.
(B) II e III, apenas.
(C) I e III, apenas.
(D) I e II, apenas.
(E) I, II e III.

```
CREATE TABLE CLIENTE(
 nomeCliente nvarchar(50) NOT NULL,
 nomeFavorita nvarchar(50) NOT NULL,
 Constraint PK_CLIENTE
 PRIMARY KEY (nomeCliente),
 Constraint FK_Cliente_Cerveja
 FOREIGN KEY (nomeFavorita)
   references CERVEJA (nomeCerveja)
   on delete set null
   on update cascade)
```

CLIENTE

nomeCliente	nomeFavorita
Ana	Stella
Mariana	Original
Pedro	Bohemia
Rafael	NULL
Thiago	Stella

CERVEJA

nomeCerveja
Bohemia
Original
Stella

I. Não é possível que haja uma chave estrangeira definida como João imaginou, pois o atributo que constitui a chave estrangeira obrigatoriamente deveria ser denominado **nomeCerveja**, tal qual o atributo da tabela referenciada.
II. Há incompatibilidade entre a semântica do *script* e a instância apresentada para a tabela.
III. Há incompatibilidade entre a semântica do *script* e a declaração dos atributos da tabela.

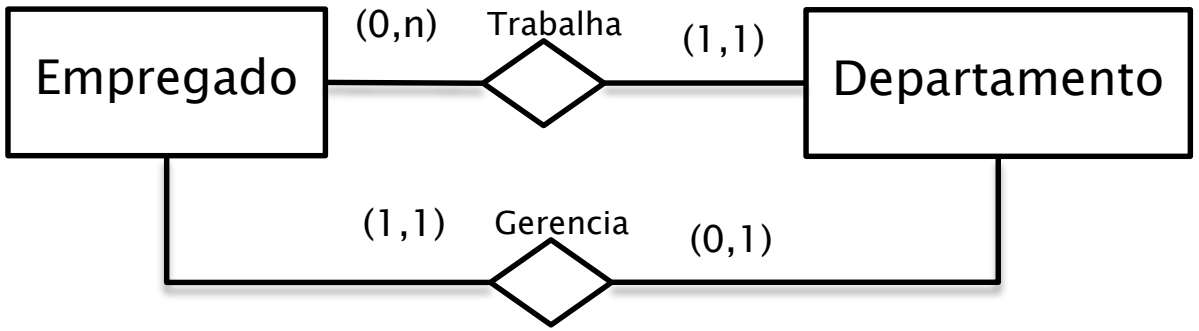
(A) II, apenas.
(B) II e III, apenas.
(C) I e III, apenas.
(D) I e II, apenas.
(E) I, II e III.

Empregado

ID_Emp	Nome	*ID_Dpto*
E1	Caio	D1
E2	Ana	D1
E3	Daniel	D2

Departamento

ID_Dpto	Nome	*ID_Gerente*
D1	RH	E1
D2	TI	E3
D3	Vendas	E2



Empregado trabalha obrigatoriamente em um departamento.
Um departamento tem obrigatoriamente um gerente.

O banco de dados vazio.
Como a gente faz para incluir um Empregado sem Departamento?
E um departamento sem empregado?

```
CREATE TABLE empregado (
Id_emp varchar(2) NOT NULL,
nome varchar(200) NOT NULL,
Id_Depto varchar(2) NOT NULL,
PRIMARY KEY (id_emp)
)
```

```
CREATE TABLE departamento (
Id_Depto varchar(2) NOT NULL,
nome char(200) NOT NULL,
Id_gerente varchar(2) NOT NULL,
PRIMARY KEY (id_Depto)
)
```

```
ALTER TABLE Empregado
    ADD CONSTRAINT FK_Dept  FOREIGN KEY (id_Depto)
            REFERENCES departamento(id_Depto)

ALTER TABLE Departamento
    ADD CONSTRAINT FK_Empregado FOREIGN KEY (id_Gerente)
      REFERENCES empregado(id_Emp)
```

Restrições de Integridade Adiáveis

- NOT DEFERRABLE
  - Padrão.
  - Restrições são aplicadas imediatamente.
- DEFERRABLE
  - Inicialmente setadas para DEFERRED ou IMMEDIATE
    - INITIALLY DEFERRED ou INITIALLY IMMEDIATE
  - DEFERRED
    - Validações são feitas no momento do commit (final da transação)
  - IMMEDIATE
    - Validações são feitas a cada commando (igual ao NOT DEFERRABLE)

```
CREATE TABLE empregado (
Id_emp varchar(2) NOT NULL,
nome varchar(200) NOT NULL,
Id_Depto varchar(2) NOT NULL,
PRIMARY KEY (id_emp)
)
```

```
CREATE TABLE departamento (
Id_Depto varchar(2) NOT NULL,
nome char(200) NOT NULL,
Id_gerente varchar(2) NOT NULL,
PRIMARY KEY (id_Depto)
)
```

```
ALTER TABLE Empregado
    ADD CONSTRAINT FK_Dept  FOREIGN KEY (id_Depto)
            REFERENCES departamento(id_Depto)
            INITIALLY DEFERRED DEFERRABLE;

ALTER TABLE Departamento
    ADD CONSTRAINT FK_Empregado FOREIGN KEY (id_Gerente)
      REFERENCES empregado(id_Emp)
      INITIALLY DEFERRED DEFERRABLE;
```

Q17 – Casa da Moeda – Banco de Dados – 2009

Para que as chaves estrangeiras sejam avaliadas somente ao final de uma transação (no momento do COMMIT), que propriedade pode ser aplicada em uma restrição (*constraint*)?

(A) NO VALIDATE

(B) NULLABLE

(C) DEFERRABLE

(D) IMMEDIATE

(E) NOT NULL

Q17 – Casa da Moeda – Banco de Dados – 2009

Para que as chaves estrangeiras sejam avaliadas somente ao final de uma transação (no momento do COMMIT), que propriedade pode ser aplicada em uma restrição (*constraint*)?

(A) NO VALIDATE

(B) NULLABLE

(C) DEFERRABLE

(D) IMMEDIATE

(E) NOT NULL

Q18 – Casa da Moeda – Desenvolvimento – 2009

Em determinada transação de um sistema de contabilidade, as chaves estrangeiras devem ser avaliadas somente no COMMIT dessa transação. Que propriedade pode ser aplicada em uma restrição (constraint) para atingir esse comportamento?

(A) DEFERRABLE

(B) NO INDEX

(C) NOW

(D) ABORT

(E) CASCADE

Q18 – Casa da Moeda – Desenvolvimento – 2009

Em determinada transação de um sistema de contabilidade, as chaves estrangeiras devem ser avaliadas somente no COMMIT dessa transação. Que propriedade pode ser aplicada em uma restrição (constraint) para atingir esse comportamento?

(A) DEFERRABLE

(B) NO INDEX

(C) NOW

(D) ABORT

(E) CASCADE

Q19 – Casa da Moeda – Analista de Negócios em TI– 2009

Em determinada funcionalidade de um sistema de marketing, as chaves estrangeiras devem ser avaliadas somente no commit de uma transação. Que propriedade pode ser aplicada em uma restrição (*constraint*) para atingir o comportamento descrito?

(A) NOT NOW

(B) CASCADE

(C) RELATED

(D) ASYNC

(E) DEFERRABLE

Q19 – Casa da Moeda – Analista de Negócios em TI– 2009

Em determinada funcionalidade de um sistema de marketing, as chaves estrangeiras devem ser avaliadas somente no commit de uma transação. Que propriedade pode ser aplicada em uma restrição (*constraint*) para atingir o comportamento descrito?

(A) NOT NOW

(B) CASCADE

(C) RELATED

(D) ASYNC

(E) DEFERRABLE

Q20 – BNDES – Desenvolvimento – 2007

Um analista de sistemas elabora um texto explicando um sistema de uma imobiliária. Todo departamento deve possuir um e somente um gerente. Todo empregado deve estar alocado a um e somente um departamento. O Administrador de Dados elabora os comandos SQL para esse sistema.

```
CREATE TABLE empregado (
matrícula number(5) NOT NULL,
nome char(200) NOT NULL,
endereço varchar(300) NULL,
iddepto number(3) NOT NULL,
PRIMARY KEY (matricula),
FOREIGN KEY (iddepto) REFERENCES departamento [.....]
)

CREATE TABLE departamento (
iddepto number(3) NOT NULL,
nome char(200) NOT NULL,
matGerente number(5) NOT NULL,
PRIMARY KEY (iddepto),
FOREIGN KEY (matGerente) REFERENCES empregado [.....]
)
```

Sobre as colunas EMPREGADO.IDDEPTO e DEPARTAMENTO.MATGERENTE e suas restrições de nulidade (NULL ou NOT NULL) e de integridade referencial (chave estrangeira), é correto afirmar que

(A) não é possível ter ambas cadastradas com NOT NULL, pois ao cadastrar o primeiro departamento, um empregado deverá existir, mas não pode existir um empregado sem departamento associado.

(B) não é possível ter ambas cadastradas com NOT NULL, mesmo com a avaliação postergada das restrições, pois no momento do COMMIT o registro referenciado pela chave estrangeira já precisa estar no banco de dados, para ser validado.

(C) ambas podem ser NOT NULL, desde que o nível de isolamento das transações permita leitura suja (*read uncommitted*).

(D) ambas podem ser NOT NULL, desde que o primeiro empregado e o primeiro departamento sejam inseridos na mesma transação e que as chaves estrangeiras sejam avaliadas somente ao final dela (no momento do COMMIT), o que pode ser conseguido declarando-as como sendo de avaliação postergada (DEFERRABLE).

(E) ambas podem ser NOT NULL, mas os sistemas não poderão usar transações para cadastrar os dados nessas tabelas.

Sobre as colunas EMPREGADO.IDDEPTO e DEPARTAMENTO.MATGERENTE e suas restrições de nulidade (NULL ou NOT NULL) e de integridade referencial (chave estrangeira), é correto afirmar que

(A) não é possível ter ambas cadastradas com NOT NULL, pois ao cadastrar o primeiro departamento, um empregado deverá existir, mas não pode existir um empregado sem departamento associado.

(B) não é possível ter ambas cadastradas com NOT NULL, mesmo com a avaliação postergada das restrições, pois no momento do COMMIT o registro referenciado pela chave estrangeira já precisa estar no banco de dados, para ser validado.

(C) ambas podem ser NOT NULL, desde que o nível de isolamento das transações permita leitura suja (*read uncommitted*).

(D) ambas podem ser NOT NULL, desde que o primeiro empregado e o primeiro departamento sejam inseridos na mesma transação e que as chaves estrangeiras sejam avaliadas somente ao final dela (no momento do COMMIT), o que pode ser conseguido declarando-as como sendo de avaliação postergada (DEFERRABLE).

(E) ambas podem ser NOT NULL, mas os sistemas não poderão usar transações para cadastrar os dados nessas tabelas.

DDL – Data Definition Language

- Criação de tabela a partir de outra tabela

```
CREATE TABLE Cliente_Simples
AS SELECT cliid, nome, cpf, sexo
FROM Cliente;
```

Já cria a tabela e adiciona as linhas do select

Q21 – CESGRANRIO – IBGE – Suporte Operacional – 2013

56 Em um banco de dados, a tabela Pessoa foi criada com a seguinte instrução:

```
CREATE TABLE Pessoa (   PessoaID int,
                        Nome varchar(255),
                        Sobrenome varchar(255),
                        Endereco varchar(255),
                        Cidade varchar(255)
                        );
```

Que instrução SQL acrescenta um campo CEP do tipo varchar(9) a essa tabela?

(A) ADD COLUMN CEP varchar(9) INTO TABLE
(B) ALTER TABLE Pessoa ADD CEP varchar(9)
(C) ALTER TABLE Pessoa INSERT COLUMN CEP varchar(9)
(D) ALTER TABLE Pessoa ALTER COLUMN CEP varchar(9)
(E) ALTER TABLE Pessoa MODIFY COLUMN ADD CEP varchar(9)

SQL ANSI:

ALTER TABLE <table name> ADD [COLUMN] <column definition>;

SQL Server, Oracle, MySQL:

ALTER TABLE Cliente **ADD** Tipo_Cli **VARCHAR**(2) **NOT NULL DEFAULT** 'PF';

ALTER TABLE Cliente **ADD** (DataNascimento DATETIME, Idade INTEGER);

PostgreSQL:

ALTER TABLE Cliente **ADD COLUMN** Tipo_Cli **VARCHAR**(2) **NOT NULL DEFAULT** 'PF';

ALTER TABLE Cliente **ADD COLUMN** DataNascimento **DATETIME**, **ADD COLUMN** Idade **INTEGER**;

(A) ADD COLUMN CEP varchar(9) INTO TABLE
(B) ALTER TABLE Pessoa ADD CEP varchar(9)
(C) ALTER TABLE Pessoa INSERT COLUMN CEP varchar(9)
(D) ALTER TABLE Pessoa ALTER COLUMN CEP varchar(9)
(E) ALTER TABLE Pessoa MODIFY COLUMN ADD CEP varchar(9)

Q21 – CESGRANRIO – IBGE – Suporte Operacional – 2013

56 Em um banco de dados, a tabela Pessoa foi criada com a seguinte instrução:

```
CREATE TABLE Pessoa (   PessoaID int,
                        Nome varchar(255),
                        Sobrenome varchar(255),
                        Endereco varchar(255),
                        Cidade varchar(255)
                        );
```

Que instrução SQL acrescenta um campo CEP do tipo varchar(9) a essa tabela?

(A) ADD COLUMN CEP varchar(9) INTO TABLE

(B) ALTER TABLE Pessoa ADD CEP varchar(9)

(C) ALTER TABLE Pessoa INSERT COLUMN CEP varchar(9)

(D) ALTER TABLE Pessoa ALTER COLUMN CEP varchar(9)

(E) ALTER TABLE Pessoa MODIFY COLUMN ADD CEP varchar(9)

DDL – Data Definition Language

- **Visão**: é um meio de prover ao usuário um "modelo personalizado" do banco de dados
- Objetivos:
  - Simplificar consultas
  - Autorização de acesso (segurança)
- O SGBD armazena a definição da visão, mas ela é instanciada quando uma consulta sobre ela for executada.
- Nem toda visão pode ser atualizada.

CREATE VIEW <nome-de-visão> [(lista-de-colunas)]

AS <comando-de-seleção>

[WITH CHECK OPTION]

CREATE VIEW view_AltoEscalao

AS SELECT

nom as nome, end as endereco, sexo,

dat_nasc as Data_Nascimento

FROM EMPREGADO

WHERE salario > 10000

CREATE VIEW <nome-de-visão> [(lista-de-colunas)]

AS <comando-de-seleção>

[WITH CHECK OPTION]

CREATE VIEW view_AltoEscalao (Nome, Endereco, Sexo, Data_Nascimento)

AS **SELECT** (nom, end, sexo, dat_nasc)

FROM EMPREGADO

WHERE salario > 10000

CONTAS

Numero	Nome	Valor
1	Paulo	200
2	André	300
3	Ana	900
4	Daniele	1000
5	Rafaela	550

VW_CONTAS

Numero	Valor
3	900
4	1000
5	550

```
CREATE VIEW VW_CONTAS AS
SELECT NUMERO, VALOR
FROM CONTAS
WHERE VALOR > 500;
```

INSERT INTO VW_CONTAS **VALUES** (6, 400); → 1 registro inserido

SELECT COUNT(*) FROM CONTAS; → 6 registros retornados

SELECT COUNT(*) FROM VW_CONTAS; → 3 registros retornados

INSERT INTO VW_CONTAS **VALUES** (7, 700); → 1 registro inserido

SELECT COUNT(*) FROM CONTAS; → 7 registros retornados

SELECT COUNT(*) FROM VW_CONTAS; → 4 registros retornados

CONTAS

Numero	Nome	Valor
1	Paulo	200
2	André	300
3	Ana	900
4	Daniele	1000
5	Rafaela	550

VW_CONTAS

Numero	Valor
3	900
4	1000
5	550

CREATE VIEW VW_CONTAS **AS**

SELECT NUMERO, VALOR

FROM CONTAS

WHERE VALOR > 500;

1 registro alterado

UPDATE VW_CONTAS **SET** VALOR = 400 **WHERE** NUMERO = 4;

CONTAS

Numero	Nome	Valor
1	Paulo	200
2	André	300
3	Ana	900
4	Daniele	400
5	Rafaela	550

VW_CONTAS

Numero	Valor
3	900
4	400
5	550

```
CREATE VIEW VW_CONTAS AS
SELECT NUMERO, VALOR
FROM CONTAS
WHERE VALOR > 500;
```

SELECT COUNT(*) FROM VW_CONTAS; → 2 registros retornados

SELECT COUNT(*) FROM CONTAS; → 5 registros retornados

CONTAS

Numero	Nome	Valor
1	Paulo	200
2	André	300
3	Ana	900
4	Daniele	1000
5	Rafaela	550

VW_CONTAS

Numero	Valor
3	900
4	1000
5	550

```
CREATE VIEW VW_CONTAS AS
SELECT NUMERO, VALOR
FROM CONTAS
WHERE VALOR > 500;
```

0 registros alterados

```
UPDATE VW_CONTAS SET VALOR = 2000 WHERE NUMERO = 2;
```

CONTAS

Numero	Nome	Valor
1	Paulo	200
2	André	300
3	Ana	900
4	Daniele	1000
5	Rafaela	550

VW_CONTAS

Numero	Valor
3	900
4	1000
5	550

CREATE VIEW VW_CONTAS **AS**

SELECT NUMERO, VALOR

FROM CONTAS

WHERE VALOR > 500;

1 registro alterado

UPDATE CONTAS **SET** VALOR = 2000 **WHERE** NUMERO = 2;

CONTAS

Numero	Nome	Valor
1	Paulo	200
2	André	2000
3	Ana	900
4	Daniele	1000
5	Rafaela	550

VW_CONTAS

Numero	Valor
2	2000
3	900
4	1000
5	550

```
CREATE VIEW VW_CONTAS AS
SELECT NUMERO, VALOR
FROM CONTAS
WHERE VALOR > 500;
```

SELECT COUNT(*) FROM VW_CONTAS; → 4 registros retornados

SELECT COUNT(*) FROM CONTAS; → 5 registros retornados

CONTAS

Numero	Nome	Valor
1	Paulo	200
2	André	300
3	Ana	900
~~4~~	~~Daniele~~	~~1000~~
5	Rafaela	550

VW_CONTAS

Numero	Valor
3	900
~~4~~	~~1000~~
5	550

```
CREATE VIEW VW_CONTAS
    AS  SELECT NUMERO, VALOR
        FROM CONTAS
        WHERE VALOR > 500;
```

```
DELETE FROM VW_CONTAS WHERE NUMERO = 2;
```
→ 0 registros apagados

```
DELETE FROM VW_CONTAS WHERE NUMERO = 4;
```
→ 1 registro apagado

- WITH CHECK OPTION
  - Operações INSERT e UPDATE sobre a visão serão rejeitadas se violarem qualquer restrição de integridade implícita na expressão de definição de visão.

CONTAS

Numero	Nome	Valor
1	Paulo	200
2	André	300
3	Ana	900
4	Daniele	1000
5	Rafaela	550

VW_CONTAS

Numero	Valor
3	900
4	1000
5	550

```
CREATE VIEW VW_CONTAS
    AS  SELECT NUMERO, VALOR
        FROM CONTAS
        WHERE VALOR > 500
        WITH CHECK OPTION;
```

```
INSERT INTO VW_CONTAS VALUES (6, 400);
```
→ ERRO!

```
INSERT INTO VW_CONTAS VALUES (7, 700);
```
→ OK! 1 registro inserido

CONTAS

Numero	Nome	Valor
1	Paulo	200
2	André	300
3	Ana	900
4	Daniele	1000
5	Rafaela	550
7		700

VW_CONTAS

Numero	Valor
3	900
4	1000
5	550
7	700

```
CREATE VIEW VW_CONTAS AS
SELECT NUMERO, VALOR
FROM CONTAS
WHERE VALOR > 500
WITH CHECK OPTION;
```

UPDATE VW_CONTAS **SET** VALOR = 100 **WHERE** NUMERO = 3; → ERRO!

UPDATE VW_CONTAS **SET** VALOR = 2000 **WHERE** NUMERO = 2; → OK! 0 registros alterados

UPDATE VW_CONTAS **SET** VALOR = 2000 **WHERE** NUMERO = 4; → OK! 1 registro alterado

DDL – Data Definition Language

- Alteração de view

```
ALTER VIEW VW_CONTAS  AS
        SELECT NUMERO, NOME, VALOR
        FROM CONTAS;
```

- Deleção de view (Estrutura)

```
DROP VIEW VW_CONTAS;
```

DDL – Data Definition Language

- Critérios para atualização de visão:
  - A query não pode conter join, ou seja, deve ser baseada apenas em uma tabela.
  - A query deve conter todas as colunas not null da tabela referenciada.
  - A query não pode conter operadores de conjunto: UNION, EXCEPT e INTERSECT.
  - A query não pode conter o operador DISTINCT.
  - A query não pode conter funções de agregação (SUM, MAX, AVG, MIN, COUNT, ...).
  - A query não pode conter GROUP BY.

Q22 – FCC – TRE RR – 2015

Considere a instrução Oracle PL/SQL a seguir.

```
CREATE VIEW valores (nome, minsal, maxsal, medsal)
AS SELECT d.depnome, MIN(e.sal), MAX(e.sal), AVG(e.sal)
FROM empregado e, departamento d
WHERE e.depnro=d.depnro
GROUP BY d.depnome;
```

Considere a existência das tabelas departamento e empregado, relacionadas de forma que cada departamento possa ter um ou muitos empregados ligados a ele. Na tabela departamento existem os campos depnro (chave primária) e depnome e na tabela empregado existem os campos empnro (chave primária), empnome, cargo, sal e depnro (chave estrangeira). Considere que em ambas as tabelas existem registros cadastrados relacionando adequadamente departamentos a empregados.

A instrução acima

```
CREATE VIEW valores (nome, minsal, maxsal, medsal)
AS SELECT d.depnome, MIN(e.sal), MAX(e.sal), AVG(e.sal)
FROM empregado e, departamento d
WHERE e.depnro=d.depnro
GROUP BY d.depnome;
```

A instrução acima

a) está incorreta, pois não é possível criar view para exibir valores a partir de duas ou mais tabelas.
b) está incorreta, pois a subconsulta que define a view não pode conter a cláusula GROUP BY
c) está correta, porém, os apelidos definidos para as colunas não serão aplicados, pois eles deveriam estar na subconsulta e não após a cláusula CREATE VIEW.
d) está incorreta, pois a função para obter a média dos valores contidos no campo sal é MED e não AVG.
e) está correta, e a view será criada com os nomes de departamento e os valores mínimo, máximo e médio dos salários por departamento.

```
CREATE VIEW valores (nome, minsal, maxsal, medsal)
AS SELECT d.depnome, MIN(e.sal), MAX(e.sal), AVG(e.sal)
FROM empregado e, departamento d
WHERE e.depnro=d.depnro
GROUP BY d.depnome;
```

A instrução acima

a) está incorreta, pois não é possível criar view para exibir valores a partir de duas ou mais tabelas.
b) está incorreta, pois a subconsulta que define a view não pode conter a cláusula GROUP BY
c) está correta, porém, os apelidos definidos para as colunas não serão aplicados, pois eles deveriam estar na subconsulta e não após a cláusula CREATE VIEW.
d) está incorreta, pois a função para obter a média dos valores contidos no campo sal é MED e não AVG.
e) está correta, e a view será criada com os nomes de departamento e os valores mínimo, máximo e médio dos salários por departamento.

Q23 – CESGRANRIO – BNDES – Desenvolvimento – 2011

Na base de dados de um sistema de controle de clientes, foi criada a tabela CLIENTES, que conta com as colunas: ID, NOME, ENDERECO, CIDADE e UF. Os valores da coluna ID não se repetem. Sobre essa tabela CLIENTES foi criada a visão VCLIENTES_RJ, que busca apresentar os clientes do estado do Rio de Janeiro. O comando de criação da visão VCLIENTES_RJ é:

```
CREATE VIEW VCLIENTES_RJ
AS SELECT ID, NOME, ENDERECO, CIDADE, UF
FROM CLIENTES WHERE UF = 'RJ'
```

Um usuário submeteu o seguinte comando para execução pelo gerenciador do banco de dados:

```
UPDATE VCLIENTES_RJ SET NOME ='JOAO'
WHERE ID IN (1,2,3) AND UF = 'SP'
```

O comando UPDATE acima, quando submetido para execução, resulta na atualização de:

```
CREATE VIEW VCLIENTES_RJ
AS SELECT ID, NOME, ENDERECO, CIDADE, UF
FROM CLIENTES WHERE UF = 'RJ'
```

```
UPDATE VCLIENTES_RJ SET NOME ='JOAO'
WHERE ID IN (1,2,3) AND UF = 'SP'
```

(A) nenhuma linha, pois, como a visão VCLIENTES_RJ somente apresenta clientes do Rio de Janeiro, não é possível atualizar o nome de um cliente de São Paulo.

(B) nenhuma linha, pois não é possível realizar atualização sobre visões.

(C) até três das linhas da visão, cujo novo valor para a coluna Nome pode ser verificado através de consulta à própria visão VCLIENTES_RJ.

(D) até três linhas da visão VCLIENTES_RJ, não sendo atualizadas linhas da tabela CLIENTES.

(E) até três linhas da tabela CLIENTES.

```
CREATE VIEW VCLIENTES_RJ
AS SELECT ID, NOME, ENDERECO, CIDADE, UF
FROM CLIENTES WHERE UF = 'RJ'
```

```
UPDATE VCLIENTES_RJ SET NOME ='JOAO'
WHERE ID IN (1,2,3) AND UF = 'SP'
```

(A) nenhuma linha, pois, como a visão VCLIENTES_RJ somente apresenta clientes do Rio de Janeiro, não é possível atualizar o nome de um cliente de São Paulo.

(B) nenhuma linha, pois não é possível realizar atualização sobre visões.

(C) até três das linhas da visão, cujo novo valor para a coluna Nome pode ser verificado através de consulta à própria visão VCLIENTES_RJ.

(D) até três linhas da visão VCLIENTES_RJ, não sendo atualizadas linhas da tabela CLIENTES.

(E) até três linhas da tabela CLIENTES.

Gabarito preliminar, E

```
CREATE VIEW VCLIENTES_RJ
AS SELECT ID, NOME, ENDERECO, CIDADE, UF
FROM CLIENTES WHERE UF = 'RJ'
```

```
UPDATE VCLIENTES_RJ SET NOME ='JOAO'
WHERE ID IN (1,2,3) AND UF = 'SP'
```

(A) nenhuma linha, pois, como a visão VCLIENTES_RJ somente apresenta clientes do Rio de Janeiro, não é possível atualizar o nome de um cliente de São Paulo.

(B) nenhuma linha, pois não é possível realizar atualização sobre visões.

(C) até três das linhas da visão, cujo novo valor para a coluna Nome pode ser verificado através de consulta à própria visão VCLIENTES_RJ.

(D) até três linhas da visão VCLIENTES_RJ, não sendo atualizadas linhas da tabela CLIENTES.

(E) até três linhas da tabela CLIENTES.

Gabarito preliminar, E

Q24 – CESGRANRIO – Petrobras – Eng Software – 2011

Sobre visões em bancos de dados relacionais, considere as afirmativas a seguir.

I – O uso de visões permite restringir o acesso a dados das tabelas por razões de segurança.

II – Fazer insert em uma visão gerada a partir de uma única tabela, e que não contenha a chave primária da tabela nessa visão, gera erro.

III – É impossível fazer update em visões geradas por junções em mais de uma tabela.

Está correto APENAS o que se afirma em

(A) I

(B) II

(C) III

(D) I e II

(E) II e III

Q24 – CESGRANRIO – Petrobras – Eng Software – 2011

Sobre visões em bancos de dados relacionais, considere as afirmativas a seguir.

I – O uso de visões permite restringir o acesso a dados das tabelas por razões de segurança.

II – Fazer insert em uma visão gerada a partir de uma única tabela, e que não contenha a chave primária da tabela nessa visão, gera erro.

III – É impossível fazer update em visões geradas por junções em mais de uma tabela.

Está correto APENAS o que se afirma em

(A) I

(B) II

(C) III

(D) I e II

(E) II e III

Q24 – CESGRANRIO – Petrobras – Eng Software – 2011

Sobre visões em bancos de dados relacionais, considere as afirmativas a seguir.

I – O uso de visões permite restringir o acesso a dados das tabelas por razões de segurança.

II – Fazer insert em uma visão gerada a partir de uma única tabela, e que não contenha a chave primária da tabela nessa visão, gera erro.

III – É impossível fazer update em visões geradas por junções em mais de uma tabela.

Está correto APENAS o que se afirma em

(A) I

(B) II

(C) III

(D) I e II

(E) II e III

Mudou com os recursos para A

Q25 – CESGRANRIO – BNDES – Desenvolvimento – 2007

Um funcionário, encarregado de verificar o correto funcionamento de uma base de dados relacional, faz o seguinte teste:

```
select nome from emp where matr = 123;
```

O resultado é vazio. Então ele executa:

```
insert into emp(matr, nome, salario, ativo)
values (123, 'José da Silva', 2000, 'N');
commit;
```

O banco de dados não retorna erro e informa que inseriu uma linha. Por fim, para verificar, ele consulta novamente:

```
select nome from emp where matr = 123;
```

O resultado continua vazio.

Supondo que o sistema gerenciador de banco de dados esteja funcionando corretamente, que opção explica o ocorrido?

(A) Como o funcionário executou o primeiro SELECT momentos antes de executar o INSERT, o resultado ficou na memória cache do computador e não foi executado pelo banco de dados na segunda vez. Somente após o protocolo LRU ter retirado do *cache* o resultado do SELECT é que ele será novamente executado.

(B) Como “emp” é uma visão e uma visão é nada menos que uma consulta gravada no banco de dados, nunca é possível usá-la em operações de manipulação de dados. O COMMIT ignora a inserção anterior.

(C) “emp” é uma visão que retorna todos os empregados ativos (ativo=’S’), mas foi criada sem a expressão WITH CHECK OPTION, que evitaria o problema acima.

(D) “emp” não é uma tabela, mas uma visão que retorna todos os empregados ativos (ativo=’S’) e foi criada com a expressão WITH CHECK OPTION. Dessa forma, como o empregado José da Silva não está ativo, o banco de dados não gravou o registro no momento do COMMIT.

(E) O funcionário executou o SELECT pouco tempo após a inserção do registro. Mesmo finalizando a transação com o COMMIT, o registro está em memória e ainda não foi gravado no disco. Somente após o CHECKPOINT é que o registro estará disponível para consulta.

Supondo que o sistema gerenciador de banco de dados esteja funcionando corretamente, que opção explica o ocorrido?

(A) Como o funcionário executou o primeiro SELECT momentos antes de executar o INSERT, o resultado ficou na memória cache do computador e não foi executado pelo banco de dados na segunda vez. Somente após o protocolo LRU ter retirado do *cache* o resultado do SELECT é que ele será novamente executado.

(B) Como “emp” é uma visão e uma visão é nada menos que uma consulta gravada no banco de dados, nunca é possível usá-la em operações de manipulação de dados. O COMMIT ignora a inserção anterior.

(C) “emp” é uma visão que retorna todos os empregados ativos (ativo=’S’), mas foi criada sem a expressão WITH CHECK OPTION, que evitaria o problema acima.

(D) “emp” não é uma tabela, mas uma visão que retorna todos os empregados ativos (ativo=’S’) e foi criada com a expressão WITH CHECK OPTION. Dessa forma, como o empregado José da Silva não está ativo, o banco de dados não gravou o registro no momento do COMMIT.

(E) O funcionário executou o SELECT pouco tempo após a inserção do registro. Mesmo finalizando a transação com o COMMIT, o registro está em memória e ainda não foi gravado no disco. Somente após o CHECKPOINT é que o registro estará disponível para consulta.

CESPE – BASA – Banco de Dados – 2012

Q26 – Com relação a visões (views) e SQL ANSI, julgue os itens consecutivos.

99. Em SQL, caso uma view tenha de atualizar dados no banco de dados, a cláusula WITH CHECK deverá ser acrescentada ao final da definição da view.

100. Em SQL, tanto CREATE VIEW quanto CREATE LIST TABLE são comandos que permitem que seja especificada uma view.

101. As visões definidas sobre várias tabelas por meio de junções, em geral, não são atualizáveis.

CESPE – BASA – Banco de Dados – 2012

Q26 – Com relação a visões (views) e SQL ANSI, julgue os itens consecutivos.

E

99. Em SQL, caso uma view tenha de atualizar dados no banco de dados, a cláusula WITH CHECK deverá ser acrescentada ao final da definição da view.

100. Em SQL, tanto CREATE VIEW quanto CREATE LIST TABLE são comandos que permitem que seja especificada uma view.

101. As visões definidas sobre várias tabelas por meio de junções, em geral, não são atualizáveis.

CESPE – BASA – Banco de Dados – 2012

Q26 – Com relação a visões (views) e SQL ANSI, julgue os itens consecutivos.

E 100. Em SQL, caso uma view tenha de atualizar dados no banco de dados, a cláusula WITH CHECK deverá ser acrescentada ao final da definição da view.

E Em SQL, tanto CREATE VIEW quanto CREATE LIST TABLE são comandos que permitem que seja especificada uma view.

101. As visões definidas sobre várias tabelas por meio de junções, em geral, não são atualizáveis.

CESPE – BASA – Banco de Dados – 2012

Q26 – Com relação a visões (views) e SQL ANSI, julgue os itens consecutivos.

E Em SQL, caso uma view tenha de atualizar dados no banco de dados, a cláusula WITH CHECK deverá ser acrescentada ao final da definição da view.

E Em SQL, tanto CREATE VIEW quanto CREATE LIST TABLE são comandos que permitem que seja especificada uma view.

C As visões definidas sobre várias tabelas por meio de junções, em geral, não são atualizáveis.

Q27 – CESPE – MPU – Tecnologica da Informação e Comunicação – 2013

110. Os comandos SQL INSERT, UPDATE, DELETE e ALTER TABLE fazem parte da DML (data manipulation language).

Q27 – CESPE – MPU – Tecnologica da Informação e Comunicação – 2013

E .Os comandos SQL INSERT, UPDATE, DELETE e ALTER TABLE fazem parte da DML (data manipulation language).

Q28 – FUNCAB – MDA – 2014

O compilador que processa as definições de esquema que são especificadas e armazena as descrições dos esquemas (metadados) no catálogo do SGBD tema denominação:

a) DML

b) DCL

c) DDL

d) DLL

e) DMC

Q28 – FUNCAB – MDA – 2014

O compilador que processa as definições de esquema que são especificadas e armazena as descrições dos esquemas (metadados) no catálogo do SGBD tema denominação:

a) DML

b) DCL

c) DDL

d) DLL

e) DMC

Q29 – IDECAN – INMETRO – 2015

A linguagem SQL (Structured Query Language), uma linguagem padrão para utilização em banco de dados, é declarativa, ao contrário das linguagens tradicionais, que são do tipo procedimental. A linguagem SQL é constituída por três sublinguagens. Relacione adequadamente as colunas acerca das sublinguagens.

1. DML Data Manipulation Language.
2. DDL Data Definition Language.
3. DCL Data Control Language.

( ) Grant.
( ) Select.
( ) Insert.
( ) Create.
( ) Revoke.
( ) Update.
( ) Alter.
( ) Drop.
( ) Delete.

a) 3, 1, 1, 2, 3, 1, 2, 2, 1.
b) 2, 2, 1, 3, 3, 1, 2, 1, 3.
c) 1, 3, 1, 2, 2, 3, 3, 1, 2.
d) 3, 2, 1, 1, 2, 3, 2, 2, 1.
e) 2, 1, 1, 3, 1, 2, 1, 3, 2.

Q29 – IDECAN – INMETRO – 2015

A linguagem SQL (Structured Query Language), uma linguagem padrão para utilização em banco de dados, é declarativa, ao contrário das linguagens tradicionais, que são do tipo procedimental. A linguagem SQL é constituída por três sublinguagens. Relacione adequadamente as colunas acerca das sublinguagens.

1. DML Data Manipulation Language.
2. DDL Data Definition Language.
3. DCL Data Control Language.

( ) Grant.
( ) Select.
( ) Insert.
( ) Create.
( ) Revoke.
( ) Update.
( ) Alter.
( ) Drop.
( ) Delete.

a) 3, 1, 1, 2, 3, 1, 2, 2, 1.
b) 2, 2, 1, 3, 3, 1, 2, 1, 3.
c) 1, 3, 1, 2, 2, 3, 3, 1, 2.
d) 3, 2, 1, 1, 2, 3, 2, 2, 1.
e) 2, 1, 1, 3, 1, 2, 1, 3, 2.

Q30 – 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.

Q30 – 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.

Q31 – CESPE – ANATEL – 2014

Julgue os itens seguintes, a respeito das linguagens de banco de dados.

A linguagem de manipulação de dados DML (data manipulation language) permite a recuperação, inclusão, exclusão ou modificação de informações do banco de dados, que são operações modificadoras da instância do banco de dados.

Q31 – CESPE – ANATEL – 2014

Julgue os itens seguintes, a respeito das linguagens de banco de dados.

E A linguagem de manipulação de dados DML (data manipulation language) permite a recuperação, inclusão, exclusão ou modificação de informações do banco de dados, que são operações modificadoras da instância do banco de dados.

Q32 – CESPE – MPU – 2013

A respeito de banco de dados, julgue os itens de 112 a 118.

As manipulações típicas de banco de dados são recuperação, inserção, remoção e modificação dos dados. Para essa finalidade, o SGBD fornece uma série de operações ou uma linguagem de manipulação de dados (data manipulation language – DML).

Q32 – CESPE – MPU – 2013

A respeito de banco de dados, julgue os itens de 112 a 118.

C As manipulações típicas de banco de dados são recuperação, inserção, remoção e modificação dos dados. Para essa finalidade, o SGBD fornece uma série de operações ou uma linguagem de manipulação de dados (data manipulation language – DML).

Q33 – FCC – TRF3 – 2014

ID	Nome	Idade
1	José Emírio	42
2	Antônio Silva	36

O comando em SQL utilizado para criar a tabela, é

a) ALTER TABLE Cadastro (ID TEXT, Nome VARCHAR, Idade INT);
b) CREATE INTO TABLE Cadastro VALUES(ID INTEGER, Nome TEXT, Idade TEXT);
c) CREATE TABLE Cadastro (ID INTEGER, Nome VARCHAR(30), Idade INTEGER);
d) CREATE Cadastro (ID INT PRIMARY KEY, Nome TEXT, Idade INT);
e) CREATE Cadastro NAMES (ID, Nome, Idade) TYPES(INT, TEXT,INT);

Q33 – FCC – TRF3 – 2014

ID	Nome	Idade
1	José Emírio	42
2	Antônio Silva	36

O comando em SQL utilizado para criar a tabela, é

a) ALTER TABLE Cadastro (ID TEXT, Nome VARCHAR, Idade INT);
b) CREATE INTO TABLE Cadastro VALUES(ID INTEGER, Nome TEXT, Idade TEXT);
c) CREATE TABLE Cadastro (ID INTEGER, Nome VARCHAR(30), Idade INTEGER);
d) CREATE Cadastro (ID INT PRIMARY KEY, Nome TEXT, Idade INT);
e) CREATE Cadastro NAMES (ID, Nome, Idade) TYPES(INT, TEXT,INT);

Q34 – MSCONCURSOS – IF-AC – 2014

São comandos de definição de dados (DDL) em banco de dados.

a) alter, drop, create

b) Select, update, insert

c) Grant, Revoke

d) Views, triggers

Q34 – MSCONCURSOS – IF-AC – 2014

São comandos de definição de dados (DDL) em banco de dados.

a) alter, drop, create

b) Select, update, insert

c) Grant, Revoke

d) Views, triggers

Q35 – BIORIO – EMGEPRON – 2014

Os SGBDs possuem compiladores que são necessários para
implementar as seguintes especificações:

I – definir os esquemas

II– acesso aos dados compatível com o modelo de dados

III– concessões de autorizações

As denominações desses compiladores, na ordem em que são especificados, são:

(A) DDL, DCL e DML

(B) DDL, DML e DCL

(C) DML, DDL e DCL

(D) DCL, DML e DDL

Q35 – BIORIO – EMGEPRON – 2014

Os SGBDs possuem compiladores que são necessários para
implementar as seguintes especificações:

I – definir os esquemas

II– acesso aos dados compatível com o modelo de dados

III– concessões de autorizações

As denominações desses compiladores, na ordem em que são especificados, são:

(A) DDL, DCL e DML

(B) DDL, DML e DCL

(C) DML, DDL e DCL

(D) DCL, DML e DDL

Q36 – IBFC – EBSERH – 2013

40) Relacione os commandos 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

Q36 – IBFC – EBSERH – 2013

40) Relacione os commandos típicos utilizados em cada linguagem conforme tabela abaixo:

(A) ŬPDĂTE

(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

Gabarito

Q1 – C
Q2 – E
Q3 – D
Q4 – B
Q5 – A
Q6 – D
Q7 – C
Q8 – E
Q9 – E
Q10 – C
Q11 – A
Q12 – A
Q13 – D
Q14 –B
Q15 – E

Q16 – B
Q17 – C
Q18 – A
Q19 – E
Q20 – D
Q21 – B
Q22 – E
Q23 – A
Q24 – A
Q25 – C
Q26 – E, E, C
Q27 – E
Q28 – C
Q29 – A
Q30 – C

Q31 – E
Q32 – C
Q33 – C
Q34 – A
Q35 – B
Q36 – D