BANCO DE DADOS
SQL DML – Parte 1 ☺

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

Dúvidas de Banco de Dados:
Fórum do curso ☺

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
  - Insert e Select ALL e SELECT Distinct
- Aula 02
  - Where e Order By
- Aula 03
  - Like Between Is null
- Aula 04
  - IN e EXISTS
- Aula 05
  - EXISTS e NULL
- Aula 06
  - Exercicios

BD – Distribuição das aulas

- Aula 07
  - ANY SOME ALL e DELETE
- Aula 8
  - UPDATE
- Aula 9
  - Funcoes para numeros e para string
- Aula 10
  - UNION INTERSECT MINUS e EXCEPT
- Aula 11
  - LEFT INNER RIGHT FULL CROSS JOIN
- Aula 12, 13, 14 e 15
  - Bateria de questoes

- 29 Questões ☺

Dúvidas de Banco de Dados

Fórum Geral de Dúvidas

Fórum Geral de Dúvidas

Acrescentar um novo tópico de discussão

Tópico	Autor	Comentários	Última mensagem
Dúvida Questão Normalização FNBC	Danniel Albuquerque Araújo	3	Eduardo Neves Dom, 14 Ago 2016, 07:41
Dúvida Questão FGV sobre normalização FNBC	Danniel Albuquerque Araújo	3	Eduardo Neves Dom, 14 Ago 2016, 07:40
Dúvida questão	André Felipe Mendonça Andrade	3	Eduardo Neves Dom, 14 Ago 2016, 07:40
Dúvida questão	Gustavo Mantuan	2	Gustavo Mantuan Dom, 10 Abr 2016, 10:24
Dúvida Aula 5 Questão 32 fgv	Gustavo Mantuan	3	Eduardo Neves Sáb, 9 Abr 2016, 12:36

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

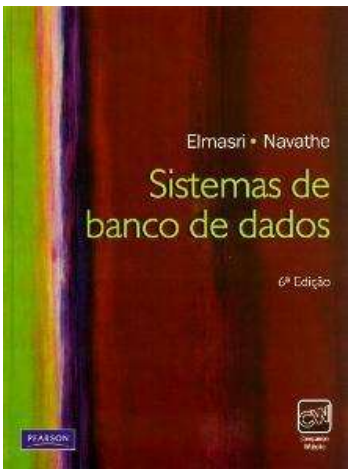
- Formação
  - Bacharel em Ciência da Computação/UFRJ
  - Pós-graduação – Gestão de Projetos
- Analista de Sistemas – BNDES – 2013
- Principais aprovações
  - 4º – BNDES/2012 – CESGRANRIO
  - 19º – BNDES/2011 – CESGRANRIO
  - 30º – Petrobras/ 2011 – CESGRANRIO
  - 12º – FINEP/2011 – CESGRANRIO
  - 3º – 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
    - (https://www.gabaritou.com.br/)
- Timasters
  - Tirar dúvidas, compartilhar conhecimento.

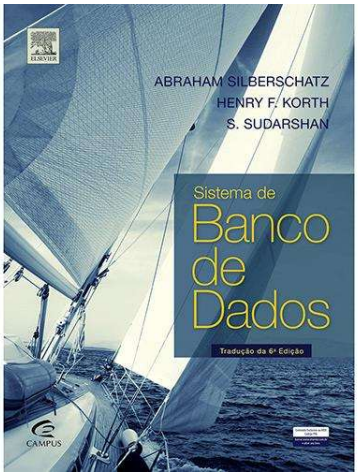
Bibliografia

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

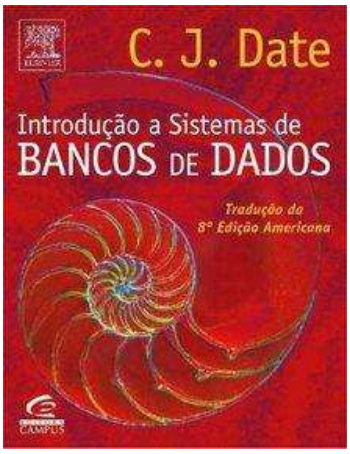


Bibliografia

Introdução a Sistemas de Banco de Dados

C.J. **Date**

Editora Campus, 2004



PROVAS DE TI
TUDO PARA VOCÊ PASSAR

BANCO DE DADOS

SQL DML – Parte 1 ☺

Lição 1

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

Dúvidas de Banco de Dados:
Fórum do curso ☺

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

Comentários

```
/*
Comentário
em
várias
linhas
*/
SELECT *
FROM pessoas
WHERE idade > 18          -- Comentário em uma linha
AND sexo = 'M'
```

Inclusão

- Inclusão de dados:

```
INSERT INTO NOME_TABELA (COL1, COL2,...,COLN)
VALUES (VAL1, VAL2,...,VALN);

-- Insere uma tupla na tabela
Insert Into PESSOA (CPF, NOME, SEXO)
values ('1111111111','João Alberto', 'M');

-- Insere o resultado de um select na tabela
Insert Into PESSOA (CPF, NOME, SEXO)
Select CPF, NOME, SEXO From Aluno
```

- Se não especificar as colunas, é utilizada a “ordem” que as colunas foram criadas na tabela.

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

-- Respeita a ordem das colunas
INSERT INTO motorista
values (‘11111111','Paulo Abreu', ‘AAA2222');

-- Insere o resultado de um select na tabela
Insert Into Motorista
Select CPF, NOME, null From Pessoas
```

CESGRANRIO – LIQUIGAS – Analista de Sistemas – 2013

Considere as tabelas a seguir para responder às questões de nos 51, 52, 53 e 54.
As tabelas a seguir pertencem ao esquema de um banco de dados de atletas de salto em distância.

```
CREATE TABLE ATLETA (
        COD NUMBER(5) NOT NULL,
        NOME VARCHAR2(50) NOT NULL,
        NUM_INSCR NUMBER(7),
        TELEFONE VARCHAR2(8) NOT NULL,
        CONSTRAINT ATLETA_PK PRIMARY KEY (COD))

CREATE TABLE PROVA (
        COD_ATL NUMBER(5) NOT NULL,
        COD_PROVA NUMBER(5) NOT NULL,
        MARCA NUMBER(5,2),
        CONSTRAINT PROVA_PK
                PRIMARY KEY (COD_ATL,COD_PROVA),
        CONSTRAINT PROVA_FK
                FOREIGN KEY (COD_ATL) REFERENCES ATLETA(COD))
```

Observação: A coluna MARCA, localizada na tabela PROVA, contém a distância, em metros, saltada por um atleta em uma prova.

CESGRANRIO – LIQUIGAS – Analista de Sistemas – 2013

Q1 – Qual comando SQL irá inserir corretamente uma nova linha na tabela de atletas?

(A) INSERT INTO ATLETA
VALUES(12345,'JOAO DO PULO','2345-6789')

(B) INSERT INTO ATLETA(NOME,COD,TELEFONE)
VALUES('JOAO DO PULO',12345,'23456789')

(C) INSERT INTO ATLETA
(NOME,COD,TELEFONE,NUM_INSCR)
VALUES('JOAO DO PULO',12345,7777777)

(D) INSERT INTO ATLETA VALUES
(12345,'JOAO DO PULO','2345-6789',7777777)

(E) INSERT INTO ATLETA (NOME,COD,NUM_INSCR)
VALUES('JOAO DO PULO',12345,7777777)

CESGRANRIO – LIQUIGAS – Analista de Sistemas – 2013

Q1 – Qual comando SQL irá inserir corretamente uma nova linha na tabela de atletas?

(A) INSERT INTO ATLETA
VALUES(12345,'JOAO DO PULO','2345-6789')

(B) INSERT INTO ATLETA(NOME,COD,TELEFONE)
VALUES('JOAO DO PULO',12345,'23456789')

(C) INSERT INTO ATLETA
(NOME,COD,TELEFONE,NUM_INSCR)
VALUES('JOAO DO PULO',12345,7777777)

(D) INSERT INTO ATLETA VALUES
(12345,'JOAO DO PULO','2345-6789',7777777)

(E) INSERT INTO ATLETA (NOME,COD,NUM_INSCR)
VALUES('JOAO DO PULO',12345,7777777)

DML – Linguagem de Manipulação de Dados

SELECT [ALL | DISTINCT] *lista-de-seleção*
FROM *tabela* [*nome-correlato*] [, *tabela* [*nome-correlato*]]
[**WHERE** *condição*]
[**GROUP BY** *atributo* [, *atributo*]]
[**HAVING** *condição*]
[**ORDER BY** *atributo* [, *atributo*] [**ASC** | **DESC**]]

Carros

Placa	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	2009
DKL4589	Toyota	Corolla	2009
LME2298	Ford	Fiesta	2010
JPG2356	Toyota	Etios	2014
PPO4554	Honda	Civic	2013

-- Todas as marcas de carros
SELECT marca **FROM** Carros

Marca
Ford
Toyota
Ford
Toyota
Honda

-- Todas as marcas e modelos, ALL é o default
SELECT ALL marca **FROM** Carros

-- Todas as marcas e modelos, sem repetição
SELECT DISTINCT marca **FROM** Carros

Marca
Ford
Toyota
Honda

Carros

Placa	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	2009
DKL4589	Toyota	Corolla	2009
LME2298	Ford	Fiesta	2010
JPG2356	Toyota	Etios	2014
PPO4554	Honda	Civic	2013

/* Mostre a placa e o ano de fabricação de todos os carros que foram fabricados depois de 2009 */

```
SELECT
        c.placa 'Placa veículo',
        c.anoFab as ano_Fabricacao
FROM Carros as c
WHERE c.anoFab > 2009
```

Placa Veículo	Ano_Fabricacao
LME2298	2010
JPG2356	2014
PPO4554	2013

Obs: Podemos usar Alias (apelidos) para as colunas ou tabelas.
O AS pode ser suprimido.

Carros

Placa	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	2009
DKL4589	Toyota	Corolla	2009
LME2298	Ford	Fiesta	2010
JPG2356	Toyota	Etios	2014
PPO4554	Honda	Civic	2013

/* Mostre a placa e o ano de fabricação de todos os carros que foram fabricados depois de 2009 ordenados crescentemente pelo ano de fabricação */

```
SELECT placa, anoFab
FROM Carros
WHERE anoFab > 2009
ORDER BY anoFab
```

Placa	AnoFab
LME2298	2010
PPO4554	2013
JPG2356	2014

/* O order by por default é ASC (Ascendente) */
SELECT placa, anoFab
FROM Carros
WHERE anoFab > 2009
ORDER BY anoFab **ASC**

/* Também podemos especificar o número da coluna ao invés do nome do campo */
SELECT placa, anoFab
FROM Carros
WHERE anoFab > 2009
ORDER BY 2

Carros

Placa	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	2009
DKL4589	Toyota	Corolla	2009
LME2298	Ford	Fiesta	2010
JPG2356	Toyota	Etios	2014
PPO4554	Honda	Civic	2013

/* Usar o DESC para ordenação Decrescente */
SELECT placa, anoFab
FROM Carros
WHERE anoFab > 2009
ORDER BY anoFab **DESC**

Placa	AnoFab
JPG2356	2014
PPO4554	2013
LME2298	2010

Carros

Placa	Marca	Preço	AnoFab
DAE6534	Ford	1000	2009
DKL4589	Toyota	500	2009
LME2298	Ford	3000	2010
JPG2356	Toyota	4000	2014
PPO4554	Honda	2000	2013

```
/* Ordenação pode ser feita por mais de uma coluna */
SELECT *
FROM Carros
ORDER BY AnoFab, Preco
```

Placa	Marca	Preço	AnoFab
DKL4589	Toyota	500	2009
DAE6534	Ford	1000	2009
LME2298	Ford	3000	2010
PPO4554	Honda	2000	2013
JPG2356	Toyota	4000	2014

Carros

Placa	Marca	Preço	AnoFab
DAE6534	Ford	1000	2009
DKL4589	Toyota	500	2009
LME2298	Ford	3000	2010
JPG2356	Toyota	4000	2014
PPO4554	Honda	2000	2013

```
/* A direção do ordenamento pode ser diferente para cada coluna */
SELECT *
FROM Carros
ORDER BY AnoFab DESC, Preco
```

Placa	Marca	Preço	AnoFab
JPG2356	Toyota	4000	2014
PPO4554	Honda	2000	2013
LME2298	Ford	3000	2010
DKL4589	Toyota	500	2009
DAE6534	Ford	1000	2009

Carros

Placa	Marca	Preço	AnoFab
DAE6534	Ford	1000	2009
DKL4589	Toyota	500	2009
LME2298	Ford	3000	2010
JPG2356	Toyota	4000	2014
PPO4554	Honda	2000	2013

/* O campo usado na ordenação não precisa ser retornado para funcionar */

```
SELECT placa
FROM Carros
ORDER BY AnoFab DESC, Preco
```

Placa
JPG2356
PPO4554
LME2298
DKL4589
DAE6534

Carros

Placa	Marca	Modelo	AnoFab
DAE6534	Ford	Fiesta	2009
DKL4589	Toyota	Corolla	2009
LME2298	Ford	Fiesta	2010
JPG2356	Toyota	Etios	2014
PPO4554	Honda	Civic	2013

/* Alias só pode ser usado no order by, where, having não pode ser usado */

```
SELECT
        c.placa 'Placa veículo',
        c.anoFab as ano_Fabricacao
FROM Carros as c
WHERE c.anoFab > 2009
Order by ano_Fabricacao
```

Placa Veículo	Ano_Fabricacao
LME2298	2010
PPO4554	2013
JPG2356	2014

Obs: MySQL permite alias no HAVING

Carros

Placa	Preço
DAE6534	1000
DKL4589	*NULL*
LME2298	3000
JPG2356	*NULL*
PPO4554	2000

```
/* O valor NULO vem primeiro na ordenação crescente */
SELECT preco
FROM Carros
ORDER BY Preco
```

Preço
NULL
NULL
1000
2000
3000

```
/* O valor NULO vem por último na ordenação decrescente*/
SELECT preco
FROM Carros
ORDER BY Preco DESC
```

Preço
3000
2000
1000
NULL
NULL

Q2 – CESPE – MPE-PI – 2012

Os atributos utilizados para a classificação do resultado pela cláusula order by não necessitam estar presentes na cláusula select da consulta

Q2 – CESPE – MPE-PI – 2012

C Os atributos utilizados para a classificação do resultado pela cláusula order by não necessitam estar presentes na cláusula select da consulta

Q3 – ESAF – MF – 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.

Q3 – ESAF – MF – 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.

Q4 – CESPE – Banco da Amazonia – 2012

Considere uma tabela de nome carros, com os campos, nome, ano, modelo e cor. Tendo como referência essa tabela, julgue os próximos itens, acerca de SQL.

Para se obter uma listagem dos dados que contenha carros com ano e modelo 2009, deve-se executar o comando select * from carros where ano=2009 and modelo=2009.

Q4 - CESPE - Banco da Amazonia - 2012

Considere uma tabela de nome carros, com os campos, nome, ano, modelo e cor. Tendo como referência essa tabela, julgue os próximos itens, acerca de SQL.

C Para se obter uma listagem dos dados que contenha carros com ano e modelo 2009, deve-se executar o comando select * from carros where ano=2009 and modelo=2009.

Q5 - FGV - DPE-RO - Redes - 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.

Q5 - FGV - DPE-RO - Redes - 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.

Q6 - FCC - TJPE - 2012

Tabela Pessoa

Id	Sobrenome	Nome	Endereço	Cidade
1	Tulio	Nelson	Rua Sete	Santos
2	Madeira	Carala	Av Quadrante	Santos
3	Pereira	Patricia	Pça Julio	Campinas

A expressão SELECT DISTINCT Cidade FROM Pessoa, terá como resultado

a) Santos.
b) Santos e Santos.
c) Santos e Campinas.
d) Campinas.
e) Santos, Santos e Campinas.

Q6 – FCC – TJPE – 2012

Tabela Pessoa

Id	Sobrenome	Nome	Endereço	Cidade
1	Tulio	Nelson	Rua Sete	Santos
2	Madeira	Carala	Av Quadrante	Santos
3	Pereira	Patricia	Pça Julio	Campinas

A expressão SELECT DISTINCT Cidade FROM Pessoa, terá como resultado

a) Santos.

b) Santos e Santos.

c) Santos e Campinas.

d) Campinas.

e) Santos, Santos e Campinas.

Predicados de comparação

- Operadores de comparação válidos:
  - **[not] like**
  - **[not] between**
  - **[not] in**
  - **all | any | some**
  - **[not] exists**
  - **is [not] NULL**

Like

- Usado quando quer-se selecionar strings quaisquer que obedeçam a um certo padrão.
- Caracter-coringa:
  - ‘_’ substitui um caracter
  - ‘%’ substitui uma substring

```
-- Produtos com nome iniciando em c e terminando com a
SELECT nome
FROM produto
WHERE nome LIKE ‘c%a’
```

caixa
cabana
caneta
…

```
/* Funcionarios com nome iniciando por carl e terminando com qualquer caracter */
SELECT nome
FROM funcionario
WHERE nome LIKE ‘carl_’
```

carla
carlo
carls
…

Like

```
-- Produtos que o nome não inicie em c e termine com a
SELECT nome
FROM produto
WHERE nome NOT LIKE ‘c%a’
```

Cabide
Faca
Bandeira
…

```
/* Funcionarios que não tenham o caracter f no nome */
SELECT nome
FROM funcionario
WHERE nome NOT LIKE ‘%f%’
```

Eduardo
Mateus
João
…

```
/* Definimos qual caracter vai funcionar como escape
Todos departamentos cujo nome começa com “_”        */
SELECT * FROM DEPT
WHERE DNAME LIKE '/_%' ESCAPE '/'
```

Like

LIKE 'A%'	Todas as palavras que iniciem com a letra A;
LIKE '%A'	Todas que terminem com a letra A;
LIKE '%A%'	Todas que tenham a letra A em qualquer posição;
LIKE 'A_'	String de dois caracteres que tenham a primeira letra A e o segundo caractere seja qualquer outro;
LIKE '_A'	String de dois caracteres cujo primeiro caractere seja qualquer um e a última letra seja A;
LIKE '_A_'	String de três caracteres cuja segunda letra seja A, independentemente do primeiro ou do último caractere;
LIKE '%A_'	Todos que tenham a letra A na penúltima posição e a última seja qualquer outro caractere;
LIKE '_A%'	Todos que tenham a letra A na segunda posição e o primeiro caractere seja qualquer um;

BETWEEN

Verifica se o valor de um determinado atributo ou expressão pertence a um intervalo, limitado por dois outros atributos ou expressões

Formato:

y **[NOT] BETWEEN** *x* **AND** *z*

$x \leq y$ e $y \leq z$

/* Todos funcionários com salário entre 2000 e 3000, ou seja, 2000 <= salario e salario <= 3000 */

SELECT * **FROM** Funcionário
WHERE salario **BETWEEN** 2000 **AND** 3000

/* Todos funcionários cujo salário NÃO está entre 2000 e 3000, ou seja,
Salario < 2000 e salario > 3000 */

SELECT * **FROM** Funcionário
WHERE salario **NOT BETWEEN** 2000 **AND** 3000

IS NULL

Verifica se o valor de um determinado atributo ou expressão é ou não nulo.

Formato:

expressão **IS [NOT] NULL**

```
/* Todos funcionários que o endereço não está preenchido */
SELECT * FROM Funcionário
WHERE endereco IS NULL

/* Todos funcionários que o endereço está preenchido */
SELECT * FROM Funcionário
WHERE endereco IS NOT NULL
```

IN

O resultado da expressão que contém este predicado será verdadeiro se e somente se o valor do operando aparecer na lista de valores ou na subconsulta subsequente.

Subconsultas são não-correlacionadas.

Formato Geral:

x **[NOT] IN (***lista_de_valores* | *subconsulta***)**

```
-- Funcionários que tenham matrícula igual a 10, 20 ou 30
SELECT * FROM Funcionário
WHERE matricula IN (10, 20, 30)

-- Funcionários que NÃO tenham matrícula igual a 10, 20 ou 30
SELECT * FROM Funcionário
WHERE matricula NOT IN (10, 20, 30)
```

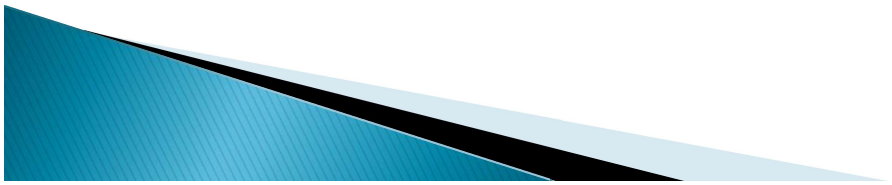
Vendas

id_venda	id_cli	Valor
1	10	7000
2	20	6000
3	50	3000
4	10	4000

Clientes

id_cli	Nome	IDCidade
10	Ana	101
20	Caio	101
30	Lia	103
40	Luiz	103
50	Lais	102

```
/* Mostrar código e nome dos clientes que fizeram compras maiores que 3000 */
SELECT id_cli, nome
FROM clientes
WHERE
        id_cli IN (
                SELECT id_cli
                FROM vendas
                WHERE valor > 3000
        )
```



Vendas

id_venda	id_cli	Valor
1	10	7000
2	20	6000
3	50	3000
4	10	4000

Clientes

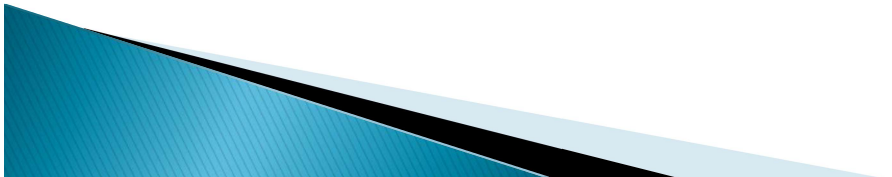
id_cli	Nome	IDCidade
10	Ana	101
20	Caio	101
30	Lia	103
40	Luiz	103
50	Lais	102

```
/* Mostrar código e nome dos clientes que fizeram compras maiores que 3000 */
SELECT id_cli, nome
FROM clientes c
WHERE
        c.id_cli IN (
                SELECT v.id_cli
                FROM vendas v
                WHERE v.valor > 3000
        )
```

id_cli
10
20
10

id_cli	Nome
10	Ana
20	Caio

Consultas normalmente não são correlacionadas



Vendas

id_venda	id_cli	Valor
1	10	7000
2	20	6000
3	50	3000
4	10	4000

Clientes

id_cli	Nome	IDCidade
10	Ana	101
20	Caio	101
30	Lia	103
40	Luiz	103
50	Lais	102

```
SELECT id_cli, nome
FROM clientes c
WHERE
    id_cli IN (
        SELECT id_cli
        FROM vendas v
        WHERE v.id_cli > c.id_cli
)
```

id_cli	Nome

Consulta correlacionada. Subconsulta vai depender da linha analisada.
Fazer a consulta linha a linha.

EXISTS

Formato Geral:

[NOT] EXISTS (*subconsulta*)

- EXISTS
  - VERDADEIRO → Subconsulta retornar uma ou mais linhas.
  - FALSO → Subconsulta retornar zero linhas.

- NOT EXISTS
  - VERDADEIRO → Subconsulta retornar zero linhas.
  - FALSO → Subconsulta retornar uma ou mais linhas.

SUBCONSULTAS normalmente são correlacionadas

Vendas

id_venda	id_cli	Valor
1	10	7000
2	20	6000
3	50	3000
4	10	4000

Clientes

id_cli	Nome	IDCidade
10	Ana	101
20	Caio	101
30	Lia	103
40	Luiz	103
50	Lais	102

```
/* Mostrar código e nome dos clientes que fizeram compras maiores que 3000 */
SELECT id_cli, nome
FROM clientes cli
WHERE
        EXISTS (
           SELECT *
           FROM vendas v
           WHERE v.valor > 3000 AND
                 v.id_cli = cli.id_cli)
```

id_cli	Nome
10	Ana
20	Caio

Consulta correlacionada. Subconsulta vai depender da linha analisada.
Fazer a consulta linha a linha.

Vendas

id_venda	id_cli	Valor
1	10	7000
2	20	6000
3	50	3000
4	10	4000

Clientes

id_cli	Nome	IDCidade
10	Ana	101
20	Caio	101
30	Lia	103
40	Luiz	103
50	Lais	102

```
SELECT id_cli, nome
FROM clientes c
WHERE
     c.id_cli IN (
          SELECT v.id_cli
          FROM vendas v
          WHERE v.valor > 3000)
```

```
SELECT id_cli, nome
FROM clientes cli
WHERE
        EXISTS (
           SELECT *
           FROM vendas v
           WHERE v.valor > 3000 AND
                 v.id_cli = cli.id_cli)
```

Consultas equivalentes. IN e EXISTS.

Vendas

id_venda	id_cli	Valor
1	10	7000
2	20	6000
3	50	3000
4	10	4000

Clientes

id_cli	Nome	IDCidade
10	Ana	101
20	Caio	101
30	Lia	103
40	Luiz	103
50	Lais	102

```
/* Mostrar código e nome dos clientes que NUNCA fizeram compras maiores que 3000 */
SELECT id_cli, nome
FROM clientes cli
WHERE
      NOT EXISTS (
          SELECT *
          FROM vendas v
          WHERE v.valor > 3000 AND
            v.id_cli = cli.id_cli)
```

id_cli	Nome
30	Lia
40	Luiz
50	Lais

Consulta correlacionada. Subconsulta vai depender da linha analisada. Fazer a consulta linha a linha.

Vendas

id_venda	id_cli	Valor
1	10	7000
2	20	6000
3	50	3000
4	10	4000

Clientes

id_cli	Nome	IDCidade
10	Ana	101
20	Caio	101
30	Lia	103
40	Luiz	103
50	Lais	102

```
/* Mostrar código e nome dos clientes se já foi feita alguma compra acima de 3000 */
SELECT id_cli, nome
FROM clientes cli
WHERE
    EXISTS (
        SELECT *
         FROM vendas v
         WHERE v.valor > 3000 )
```

id_cli	Nome
10	Ana
20	Caio
30	Lia
40	Luiz
50	Lais

Consulta NÃO correlacionada. Subconsulta vai ser constante.

Vendas

id_venda	id_cli	Valor
1	10	7000
2	20	6000
3	50	3000
4	10	4000

Clientes

id_cli	Nome	IDCidade
10	Ana	101
20	Caio	101
30	Lia	103
40	Luiz	103
50	Lais	102

```
SELECT id_cli, nome
FROM clientes cli
WHERE
    EXISTS (
        SELECT v.id_venda
        FROM vendas v
        WHERE v.valor > 3000 AND
            v.id_cli = cli.id_cli)
```

id_cli	Nome
10	Ana
20	Caio

Exists verifica se retorna linha, não importando seu conteúdo

Vendas

id_venda	id_cli	Valor
1	10	7000
2	20	6000
3	50	3000
4	10	4000

Clientes

id_cli	Nome	IDCidade
10	Ana	101
20	Caio	101
30	Lia	103
40	Luiz	103
50	Lais	102

```
SELECT id_cli, nome
FROM clientes cli
WHERE
    EXISTS (
        SELECT 1
        FROM vendas v
        WHERE v.valor > 3000 AND
            v.id_cli = cli.id_cli)
```

id_cli	Nome
10	Ana
20	Caio

Exists verifica se retorna linha, não importando seu conteúdo

Vendas

id_venda	id_cli	Valor
1	10	7000
2	20	6000
3	50	3000
4	10	4000

Clientes

id_cli	Nome	IDCidade
10	Ana	101
20	Caio	101
30	Lia	103
40	Luiz	103
50	Lais	102

```
SELECT id_cli, nome
FROM clientes cli
WHERE
    EXISTS (
        SELECT NULL
        FROM vendas v
        WHERE v.valor > 3000 AND
            v.id_cli = cli.id_cli)
```

id_cli	Nome
10	Ana
20	Caio

Exists verifica se retorna linha, não importando seu conteúdo

IS NULL

Verifica se o valor de um determinado atributo ou expressão é ou não nulo.

Formato:

expressão **IS [NOT] NULL**

```
/* Todos funcionários que o endereço não está preenchido */
SELECT * FROM Funcionário
WHERE endereco IS NULL

/* Todos funcionários que o endereço está preenchido */
SELECT * FROM Funcionário
WHERE endereco IS NOT NULL
```

NULL

S

A	B
1	NULL
3	8
5	8

T

C	D	E
NULL	0	6
4	9	3
NULL	4	5

SELECT * FROM S **WHERE** B > 0 ⇒

A	B
3	8
5	8

SELECT * FROM S **WHERE** B < 0 ⇒

A	B

NULL

S

A	B
1	NULL
3	8
5	8

T

C	D	E
NULL	0	6
4	9	3
NULL	4	5

SELECT * FROM S, T
WHERE B > C ⇒

A	B	C	D	E	
1	NULL	NULL	0	6	✗
1	NULL	4	9	3	✗
1	NULL	NULL	4	5	✗
3	8	NULL	0	6	✗
3	8	4	9	3	✓
3	8	NULL	4	5	✗
5	8	NULL	0	6	✗
5	8	4	9	3	✓
5	8	NULL	4	5	✗

⇐

A	B	C	D	E
3	8	4	9	3
5	8	4	9	3

NULL

S

A	B
1	NULL
3	8
5	8

T

C	D	E
NULL	0	6
4	9	3
NULL	4	5

SELECT distinct * FROM S, T
WHERE B > C

A	B	C	D	E	
1	NULL	NULL	0	6	✗
1	NULL	4	9	3	✗
1	NULL	NULL	4	5	✗
3	8	NULL	0	6	✗
3	8	4	9	3	✓
3	8	NULL	4	5	✗
5	8	NULL	0	6	✗
5	8	4	9	3	✓
5	8	NULL	4	5	✗

A	B	C	D	E
3	8	4	9	3
5	8	4	9	3

A	B	C	D	E
3	8	4	9	3
5	8	4	9	3

NULL

S

A	B
1	NULL
3	8
5	8

T

C	D	E
NULL	0	6
4	9	3
NULL	4	5

SELECT S.* FROM S, T
WHERE B > C

A	B	C	D	E	
1	NULL	NULL	0	6	✗
1	NULL	4	9	3	✗
1	NULL	NULL	4	5	✗
3	8	NULL	0	6	✗
3	8	4	9	3	✓
3	8	NULL	4	5	✗
5	8	NULL	0	6	✗
5	8	4	9	3	✓
5	8	NULL	4	5	✗

A	B
3	8
5	8

A	B	C	D	E
3	8	4	9	3
5	8	4	9	3

NULL

S

A	B
1	NULL
3	8
5	8

T

C	D	E
NULL	0	6
4	9	3
NULL	4	5

SELECT T.* FROM S, T
WHERE B > C

A	B	C	D	E	
1	NULL	NULL	0	6	✗
1	NULL	4	9	3	✗
1	NULL	NULL	4	5	✗
3	8	NULL	0	6	✗
3	8	4	9	3	✓
3	8	NULL	4	5	✗
5	8	NULL	0	6	✗
5	8	4	9	3	✓
5	8	NULL	4	5	✗

C	D	E
4	9	3
4	9	3

A	B	C	D	E
3	8	4	9	3
5	8	4	9	3

Q7 - FGV - TJ/BA - Tecn. Inform. - 2015

Considere uma tabela relacional S, cuja instância é mostrada a seguir. Sabendo-se que "NULL" denota um valor não preenchido, analise os comandos SQL a seguir.

I. select distinct s1.* from s s1, s s2 where s1.a < s2.c
II. select distinct s1.* from s s1 where exists (select * from s s2 where s1.a != s2.c)
III. select distinct s1.* from s s1, s s2 where s1.a != s2.c
IV. select * from s

Resultados com o mesmo conteúdo do resultado do comando IV são produzidos:

(A) apenas pelo comando I;
(B) apenas pelo comando II;
(C) apenas pelos comandos I e III;
(D) apenas pelos comandos II e III;
(E) pelos comandos I, II e III.

A	C
1	NULL
3	8
5	8

A	C
1	NULL
3	8
5	8

I. select distinct s1.*
from s s1, s s2
where s1.a < s2.c

S1.a	S1.c	S2.a	S2.c	
1	NULL	1	NULL	✗
1	NULL	3	8	✓
1	NULL	5	8	✓
3	8	1	NULL	✗
3	8	3	8	✓
3	8	5	8	✓
5	8	1	NULL	✗
5	8	3	8	✓
5	8	5	8	✓

S1.a	S1.c
1	NULL
3	8
5	8

A	C
1	NULL
3	8
5	8

S1.a	S1.c
1	NULL
3	8
5	8

S2.a	S2.c
1	NULL
3	8
5	8

II. select distinct s1.*
from s s1
where exists
(select * from s s2
  where s1.a != s2.c)

S1.a	S1.c
1	NULL
3	8
5	8

A	C
1	NULL
3	8
5	8

III. select distinct s1.*
from s s1, s s2
where s1.a != s2.c

S1.a	S1.c	S2.a	S2.c	
1	NULL	1	NULL	✗
1	NULL	3	8	✓
1	NULL	5	8	✓
3	8	1	NULL	✗
3	8	3	8	✓
3	8	5	8	✓
5	8	1	NULL	✗
5	8	3	8	✓
5	8	5	8	✓

S1.a	S1.c
1	NULL
3	8
5	8

Q7 – FGV – TJ/BA – Tecn. Inform. – 2015

Considere uma tabela relacional S, cuja instância é mostrada a seguir. Sabendo-se que "NULL" denota um valor não preenchido, analise os comandos SQL a seguir.

I. select distinct s1.* from s s1, s s2 where s1.a < s2.c
II. select distinct s1.* from s s1 where exists (select * from s s2 where s1.a != s2.c)
III. select distinct s1.* from s s1, s s2 where s1.a != s2.c
IV. select * from s

Resultados com o mesmo conteúdo do resultado do comando IV são produzidos:

(A) apenas pelo comando I;
(B) apenas pelo comando II;
(C) apenas pelos comandos I e III;
(D) apenas pelos comandos II e III;
(E) pelos comandos I, II e III.

A	C
1	NULL
3	8
5	8

Gabarito

Q1 – B
Q2 – C
Q3 – B
Q4 – C
Q5 – A
Q6 – C
Q7 – E
Q8 – E
Q9 – E
Q10 – B
Q11 – E
Q12 – D
Q13 – B
Q14 – A
Q15 – D
Q15.1 – C
Q16 – D
Q17 – D
Q18 – A
Q19 – D
Q20 – A
Q21 – A
Q22 – A
Q23 – C
Q24 – E
Q25 – B
Q26 – C
Q27 – C
Q28 – D
Q29 – E

PROVAS DE TI
TUDO PARA VOCÊ PASSAR

BANCO DE DADOS

SQL DML – Parte 1 ☺

Lição 2

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

Dúvidas de Banco de Dados:
Fórum do curso ☺

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

ANY or ALL

Os operadores *any-or-all* podem ser:

= ANY (equivalente ao **IN**)	**= SOME** (equivalente ao **IN**)	**= ALL**
<> ANY	**<> SOME**	**<> ALL**
< ANY	**< SOME**	**< ALL**
<= ANY	**<= SOME**	**<= ALL**
> ANY	**> SOME**	**> ALL**
>= ANY	**>= SOME**	**>= ALL**

/* Mostre os dados dos funcionários que não recebem mais que 5000 */

```
SELECT *
FROM Funcionário
WHERE
matricula <> ALL (
   SELECT matriculaFuncionario
   FROM trabalha
   WHERE salario > 5000)
```

```
SELECT *
FROM Funcionário
WHERE
matricula not in (
   SELECT matriculaFuncionario
   FROM trabalha
   WHERE salario > 5000)
```

CESGRANRIO – Petrobras – Processos – 2010

Q8 – 2 – Considere a expressão em SQL a seguir.

```
SELECT R1.sno
FROM R1
WHERE R1.pno >= ALL (SELECT R2.pno FROM R2)
```

O resultado dessa consulta é

R1

sno	pno
1	1
1	2
1	3
1	4
2	1
2	2
3	2
4	2
4	4

R2

pno
1
2

(A)

sno
1
2
2
2
4
4

(B)

sno
1
1
1
2
2
3
4

(C)

sno
1
2
3
3
4
4
4

(D)

sno
1
2
3
3
4
4

(E)

sno
1
1
1
2
3
4
4

CESGRANRIO – Petrobras – Processos – 2010

Q8 – 2 – Considere a expressão em SQL a seguir.

```
SELECT R1.sno
FROM R1
WHERE R1.pno >= ALL (SELECT R2.pno FROM R2)
```

O resultado dessa consulta é

R1

sno	pno
1	1
1	2
1	3
1	4
2	1
2	2
3	2
4	2
4	4

R2

pno
1
2

(A)

sno
1
2
2
2
4
4

(B)

sno
1
1
1
2
2
3
4

(C)

sno
1
2
3
3
4
4
4

(D)

sno
1
2
3
3
4
4

(E)

sno
1
1
1
2
3
4
4

Deleção

- Deleção:

DELETE FROM NOME_TABELA
WHERE (expressão lógica)

-- Apagando todo funcionário com nome João Alberto
DELETE FROM funcionario
WHERE nome = 'João Alberto';

-- Apagando todo funcionário que não tenha uma venda
DELETE FROM clients
WHERE
id_cli **NOT IN** (**SELECT** id_cli **from vendas**)

Q9 - FGV - TJBA - Tec Inform - 2015

Considere uma tabela relacional TX, cuja instância é mostrada a seguir. Considere também o comando SQL abaixo.

A	B
4	4
2	3
3	4
3	2

```
delete from TX
where exists
      (select * from TX tt where TX.B=tt.A)
```

O número de registros deletados da tabela TX por esse comando é:

(A) 0
(B) 1
(C) 2
(D) 3
(E) 4

Q9 - FGV - TJBA - Tec Inform - 2015

Considere uma tabela relacional TX, cuja instância é mostrada a seguir. Considere também o comando SQL abaixo.

A	B
4	4
2	3
3	4
3	2

```
delete from TX
where exists
      (select * from TX tt where TX.B=tt.A)
```

O número de registros deletados da tabela TX por esse comando é:

(A) 0
(B) 1
(C) 2
(D) 3
(E) 4

Q10 - FGV - TJRO - 2015

47 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.

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


```
delete from S
where not exists
   (select * from R
    where
          R.A=S.C and R.B=S.D)
```

Q10 – FGV – TJRO – 2015

47 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.

FGV – PROCEMPA – 2014

Q11 – Com referência ao banco BD_CERVEJA, analise a seguinte operação.

```
delete from cliente
where exists
(select * from cliente c
where c.nomeCliente=cliente.nomeCliente)
```

Assinale a opção que apresenta a quantidade de registros removidos.

(A) Zero.
(B) Um.
(C) Dois.
(D) Quatro.
(E) Cinco.

CLIENTE

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

Q11 – Com referência ao banco BD_CERVEJA, analise a seguinte operação.

```
delete from cliente
where exists
(select * from cliente c
where c.nomeCliente=cliente.nomeCliente)
```

Assinale a opção que apresenta a quantidade de registros removidos.

(A) Zero.
(B) Um.
(C) Dois.
(D) Quatro.
(E) Cinco.

CLIENTE

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

Atualização

- Atualização:

UPDATE NOME_TABELA
SET COL1=VAL1, COL2=VAL2,...,COLN=VALN
WHERE (expressão lógica)

UPDATE funcionario
SET idade = idade+5
WHERE nome = 'Felipe';

Q12 - CESGRANRIO - IBGE - Desenv 2013

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));
```

Após a criação, a tabela foi preenchida, porém o programador percebeu que todos os Nomes foram colocados no lugar do Sobrenome e vice-versa.
Que instrução SQL pode ser usada para realizar a troca, corrigindo a base?

(A) `SELECT Nome As Sobrenome, Sobrenome AS Nome FROM Pessoa`
(B) `UPDATE Nome=Sobrenome, Sobrenome=Nome FROM Pessoa`
(C) `UPDATE Pessoa SET Nome,Sobrenome WITH (SELECT Sobrenome,Nome FROM Pessoa)`
(D) `UPDATE Pessoa SET Nome=Sobrenome, Sobrenome=Nome`
(E) `UPDATE Pessoa WITH Nome As Sobrenome, Sobrenome AS Nome`

Q12 - CESGRANRIO - IBGE - Desenv 2013

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));
```

Após a criação, a tabela foi preenchida, porém o programador percebeu que todos os Nomes foram colocados no lugar do Sobrenome e vice-versa.
Que instrução SQL pode ser usada para realizar a troca, corrigindo a base?

(A) `SELECT Nome As Sobrenome, Sobrenome AS Nome FROM Pessoa`
(B) `UPDATE Nome=Sobrenome, Sobrenome=Nome FROM Pessoa`
(C) `UPDATE Pessoa SET Nome,Sobrenome WITH (SELECT Sobrenome,Nome FROM Pessoa)`
(D) `UPDATE Pessoa SET Nome=Sobrenome, Sobrenome=Nome`
(E) `UPDATE Pessoa WITH Nome As Sobrenome, Sobrenome AS Nome`

CESGRANRIO – Banco da Amazônia – Banco de Dados – 2014

Q13 – Que comando SQL deve ser dado para bloquear o crédito da conta 123456, colocando "S" no campo credito_bloqueado_sn?

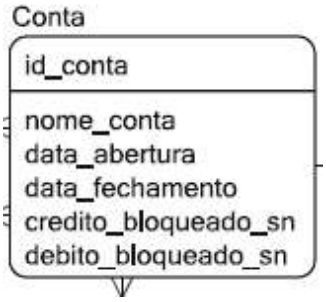
(A) UPDATE Conta SET credito_bloqueado_sn="S" SELECT * FROM Conta WHERE id_conta=123456

(B) UPDATE Conta SET credito_bloqueado_sn="S" WHERE id_conta=123456

(C) UPDATE Conta SET VALUES (id_conta,"S") WHERE id_conta=123456

(D) UPDATE credito_bloqueado_sn="S" From Conta WHERE id_conta=123456

(E) UPDATE INTO Conta VALUES (123456,"S")



CESGRANRIO – Banco da Amazônia – Banco de Dados – 2014

Q13 – Que comando SQL deve ser dado para bloquear o crédito da conta 123456, colocando "S" no campo credito_bloqueado_sn?

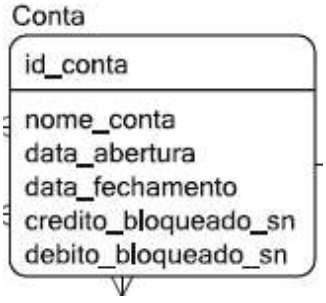
(A) UPDATE Conta SET credito_bloqueado_sn="S" SELECT * FROM Conta WHERE id_conta=123456

(B) UPDATE Conta SET credito_bloqueado_sn="S" WHERE id_conta=123456

(C) UPDATE Conta SET VALUES (id_conta,"S") WHERE id_conta=123456

(D) UPDATE credito_bloqueado_sn="S" From Conta WHERE id_conta=123456

(E) UPDATE INTO Conta VALUES (123456,"S")



Q14 – FGV – DPE-RO – Redes – 2015

24 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) Y seja uma coluna da tabela X e Z seja o nome de um tipo de dados válido.

Q14 – FGV – DPE-RO – Redes – 2015

24 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) Y seja uma coluna da tabela X e Z seja o nome de um tipo de dados válido.

Operações com NULL

not NULL = NULL
NULL or false = NULL
NULL or true = true
NULL or NULL = NULL
NULL and false = false
NULL and true = NULL
NULL and NULL = NULL

Q15 – FGV – DPEMT – 2015

74 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; ?; ?

Q15 – FGV – DPEMT – 2015

74 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; ?; ?

DML – Linguagem de Manipulação de Dados

- Funções para números
  - **ABS** (n)
    - Devolve o valor absoluto de (n).
  - **CEIL** (n)
    - Obtém o valor inteiro imediatamente superior ou igual a "n".
  - **FLOOR** (n)
    - Devolve o valor inteiro imediatamente inferior ou igual a "n".

DML – Linguagem de Manipulação de Dados

- Funções para números
  - **MOD** (m, n)
    - Retorna o resto da divisão de "m" por "n".
  - **POWER** (m, exponente)
    - Calcula a potência de um número.
  - **SQRT** (n)
    - Retorna a raiz quadrada de "n".

DML – Linguagem de Manipulação de Dados

- Funções para cadeias de caracteres:
  - **LPAD** (col, tam, 'char')
    - Adiciona caracteres à esquerda do resultado até que tenha um certo tamanho.
      - Ex: LPAD(nome, 10, '*') = ******Dudu
  - **RPAD** (col, tam, 'char')
    - Adiciona caracteres à direita do resultado até que tenha um certo tamanho.
      - Ex: RPAD(nome, 10, '*') = Dudu******

DML – Linguagem de Manipulação de Dados

- Funções para cadeias de caracteres:
  - **LTRIM** (col, 'chars')
    - Suprime um conjunto de caracteres à esquerda do resultado.
      - Ex: LTRIM(nome, 'E') = Eduardo – duardo
  - **RTRIM** (col, 'chars')
    - Suprime um conjunto de caracteres à direita da cadeia.
      - Ex: RTRIM(nome, 'do') = Eduardo –Eduar

DML – Linguagem de Manipulação de Dados

- Funções:
  - **NVL** (col, valor)
    - Oracle
    - Substitui um valor nulo por outro valor.
  - **ISNULL** (col, valor)
    - SQL Server
    - Substitui um valor nulo por outro valor.
  - **ISNULL** (col)
    - MySQL
    - Se expressão for NULL retorna 1, senão retorna 0
  - **IFNULL** (col, valor)
    - MySQL
    - Substitui um valor nulo por outro valor.

DML – Linguagem de Manipulação de Dados

- **COALESCE** (col, valor)
  - Substitui um valor nulo por outro valor.
- **COALESCE** (col1, col2, col3)
  - retorna o primeiro argumento da lista cujo valor é diferente de NULL.
- **NULLIF** (col, valor)
  - retorna NULL se col = valor;
- **IIF** (boolean_expression, true_value, false_value)

Q15.1 – FCC – TRT23 – An. Jud. – 2016

Considere a instrução abaixo, digitada em um banco de dados Oracle 11g que possui a tabela Produtos, aberta e em condições ideais, contendo os campos NomeProduto – varchar2(50), PrecoUnitario – number(5,2), UnidadesEmEstoque – integer, UnidadesNoPedido – integer:

```
SELECT NomeProduto, PrecoUnitario*(UnidadesEmEstoque+ ..I.. (UnidadesNoPedido,0)) FROM Produtos;
```

Para retornar 0 se o valor de UnidadesNoPedido for nulo, a lacuna I deverá ser corretamente preenchida com

(A) IIF
(B) IFNULL
(C) NVL
(D) COALESCE
(E) ISNULL

Q15.1 – FCC – TRT23 – An. Jud. – 2016

Considere a instrução abaixo, digitada em um banco de dados Oracle 11g que possui a tabela Produtos, aberta e em condições ideais, contendo os campos NomeProduto – varchar2(50), PrecoUnitario – number(5,2), UnidadesEmEstoque – integer, UnidadesNoPedido – integer:

```
SELECT NomeProduto, PrecoUnitario*(UnidadesEmEstoque+ I (UnidadesNoPedido,0)) FROM Produtos;
```

Para retornar 0 se o valor de UnidadesNoPedido for nulo, a lacuna I deverá ser corretamente preenchida com

(A) IIF
(B) IFNULL
(C) NVL
(D) COALESCE
(E) ISNULL

DML – Linguagem de Manipulação de Dados

- Função para conversão de tipos:
- **CAST**
  - CAST(dado_origem as tipo_dado_destino)

SELECT **CAST** (AVG(PRECO) AS DECIMAL (10,2))
FROM PRODUTO;

SELECT **CAST** ('11/05/2016' AS DATE)
FROM DUAL;

UNION

Efetua a união lógica do resultado de duas consultas.

consulta1 **UNION** *consulta2*

O resultado das consultas 1 e 2 devem ser compatíveis, ou seja, os atributos retornados pelas consultas devem ser os mesmos.

```
( SELECT matricula, nome_func, endereço
FROM Funcionário
WHERE salario > 5000)
UNION
( SELECT matricula, nome_func, endereço
FROM Funcionário
WHERE idade > 50)
```

Obs: UNION retira as repetições.

UNION ALL

```
/* UNION ALL não retira as repetições */
( SELECT matricula, nome_func, endereço
FROM Funcionário
WHERE salario > 5000)
UNION ALL
( SELECT matricula, nome_func, endereço
FROM Funcionário
WHERE idade > 50)
```

UNION

Funcionario

Idade
30
31
32
33
34

Medico

Idade
32
33
34
40
42

```
SELECT idade
FROM Funcionário
UNION
SELECT idade
FROM Medico
```

Idade
30
31
32
33
34
32
33
34
40
42

Tira repetições

Idade
30
31
32
33
34
40
42

UNION

Funcionario

Idade
30
30
30
30
30

Medico

Idade
32
33
34

```
SELECT idade
FROM Funcionário
UNION
SELECT idade
FROM Medico
```

Idade
30
30
30
30
30
32
33
34

Tira repetições

Idade
30
32
33
34

UNION

Funcionario

Idade
30
30
30
30
30

Medico

Idade

```
SELECT idade
FROM Funcionário
UNION
SELECT idade
FROM Medico
```

Idade
30
30
30
30
30

Tira repetições

Idade
30

INTERSECT

Interseção:

```
SELECT Placa
FROM Carros
WHERE Marca IN ('Toyota', 'Ford')
INTERSECT
SELECT Placa
FROM Carros
WHERE Marca IN ('Toyota', 'Honda')
```

Carros

Placa	Marca	Preço	AnoFab
DAE6534	Ford	1000	2009
DKL4589	Toyota	500	2009
LME2298	Ford	3000	2010
JPG2356	Toyota	4000	2014
PPO4554	Honda	2000	2013

EXCEPT (MINUS)

Diferença:

SELECT Placa
FROM Carros
WHERE Marca **IN** (‘Toyota’, ‘Ford’)
EXCEPT
SELECT Placa
FROM Carros
WHERE Marca **IN** (‘Toyota’, ‘Honda’)

Carros

Placa	Marca	Preço	AnoFab
DAE6534	Ford	1000	2009
DKL4589	Toyota	500	2009
LME2298	Ford	3000	2010
JPG2356	Toyota	4000	2014
PPO4554	Honda	2000	2013

Observações

- No Oracle o MINUS é equivalente ao EXCEPT
- UNION, EXCEPT e INTERSECT
  - removem duplicatas
- UNION ALL, EXCEPT ALL, INTERSECT ALL
  - Não removem duplicatas

Q16 – FGV – PGE-RO – Desenv – 2015

34 Analise os comandos SQL a seguir.

I.
```
select * from X
UNION ALL
select * from Y
```

II.
```
select * from X
INTERSECT
select * from Y
```

III.
```
select * from X
MINUS
select * from Y
```

Sabendo-se que os comandos I, II e III retornam, respectivamente, 9, 2 e 3 registros, e que as duas tabelas possuem chaves primárias, o número de registros presentes nas tabelas X e Y, respectivamente, é:
(A) 8 e 1;
(B) 7 e 2;
(C) 6 e 3;
(D) 5 e 4;
(E) 4 e 5.

Q16 – FGV – PGE-RO – Desenv – 2015

34 Analise os comandos SQL a seguir.

I.
```
select * from X
UNION ALL
select * from Y
```

II.
```
select * from X
INTERSECT
select * from Y
```

III.
```
select * from X
MINUS
select * from Y
```

Sabendo-se que os comandos I, II e III retornam, respectivamente, 9, 2 e 3 registros, e que as duas tabelas possuem chaves primárias, o número de registros presentes nas tabelas X e Y, respectivamente, é:
(A) 8 e 1;
(B) 7 e 2;
(C) 6 e 3;
(D) 5 e 4;
(E) 4 e 5.

Q17 – FGV – TJRO – 2015

46 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.

Q17 – FGV – TJRO – 2015

46 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.

Funcionario

MatF	NomeF	IdD
10	Ana	D1
20	Caio	D1
30	Lia	D3
40	Luiz	null
50	Lais	null

Departamento

IdD	NomeD
D1	RH
D2	Vendas
D3	TI
D4	Projetos

- Junção Interna:

```
Select *
From   Funcionario F INNER JOIN
       Departamento D  on F.idD = D.idD
```

MatF	NomeF	IdD	IdD	NomeD
10	Ana	D1	D1	RH
20	Caio	D1	D1	RH
30	Lia	D3	D3	TI

Obs: INNER pode ser suprimido, ou seja, pode-se usar apenas JOIN

Cláusula ON é obrigatória

Funcionario

MatF	NomeF	IdD
10	Ana	D1
20	Caio	D1
30	Lia	D3
40	Luiz	null
50	Lais	null

Departamento

IdD	NomeD
D1	RH
D2	Vendas
D3	TI
D4	Projetos

- Junção Interna:

```
Select *
From   Funcionario F, Departamento D
Where  F.idD = D.idD
```

MatF	NomeF	IdD	IdD	NomeD
10	Ana	D1	D1	RH
20	Caio	D1	D1	RH
30	Lia	D3	D3	TI

Funcionario

MatF	NomeF	IdD
10	Ana	D1
20	Caio	D1
30	Lia	D3
40	Luiz	null
50	Lais	null

Departamento

IdD	NomeD
D1	RH
D2	Vendas
D3	TI
D4	Projetos

- Junção Externa à Esquerda:

```
Select *
From   Funcionario F LEFT OUTER JOIN
       Departamento D  on F.idD = D.idD
```

MatF	NomeF	IdD	IdD	NomeD
10	Ana	D1	D1	RH
20	Caio	D1	D1	RH
30	Lia	D3	D3	TI
40	Luiz	null	null	null
50	Lais	null	null	null

Obs: OUTER pode ser suprimido, ou seja, pode-se usar apenas LEFT JOIN

Cláusula ON é obrigatória

Funcionario

MatF	NomeF	IdD
10	Ana	D1
20	Caio	D1
30	Lia	D3
40	Luiz	null
50	Lais	null

Departamento

IdD	NomeD
D1	RH
D2	Vendas
D3	TI
D4	Projetos

- Junção Externa à Direita:

```
Select *
From   Funcionario F RIGHT OUTER JOIN
       Departamento D  on F.idD = D.idD
```

MatF	NomeF	IdD	IdD	NomeD
10	Ana	D1	D1	RH
20	Caio	D1	D1	RH
Null	Null	Null	D2	Vendas
30	Lia	D3	D3	TI
Null	Null	Null	D4	Projetos

Obs: OUTER pode ser suprimido, ou seja, pode-se usar apenas RIGHT JOIN

Cláusula ON é obrigatória

Funcionario

MatF	NomeF	IdD
10	Ana	D1
20	Caio	D1
30	Lia	D3
40	Luiz	null
50	Lais	null

Departamento

IdD	NomeD
D1	RH
D2	Vendas
D3	TI
D4	Projetos

- Junção Externa Completa:

Select *
From Funcionario F **FULL OUTER JOIN**
Departamento D **on** F.idD = D.idD

MatF	NomeF	IdD	IdD	NomeD
10	Ana	D1	D1	RH
20	Caio	D1	D1	RH
Null	Null	Null	D2	Vendas
30	Lia	D3	D3	TI
Null	Null	Null	D4	Projetos
40	Luiz	Null	Null	Null
50	Lais	Null	Null	Null

Obs: OUTER pode ser suprimido, ou seja, pode-se usar apenas FULL JOIN

Cláusula ON é obrigatória

CROSS JOIN

Paciente

Cod_Pac	Nome_Pac
P1	Matias
P2	Gilberto
P3	Lia

Medico

Cod_Med	Nome_Med
M1	Armando
M2	Denise

- Junção Cruzada (produto Cartesiano):

Select Cod_Pac, Nome_Pac, Cod_Med, Nome_Med
From Paciente **CROSS JOIN** Medico

Paciente x Medico

Cod_Pac	Nome_Pac	Cod_Med	Nome_Med
P1	Matias	M1	Armando
P2	Gilberto	M1	Armando
P3	Lia	M1	Armando
P1	Matias	M2	Denise
P2	Gilberto	M2	Denise
P3	Lia	M2	Denise

Obs: ON é opcional no CROSS JOIN

CROSS JOIN

Paciente

Cod_Pac	Nome_Pac
P1	Matias
P2	Gilberto
P3	Lia

Medico

Cod_Med	Nome_Med
M1	Armando
M2	Denise

- Junção Cruzada (produto Cartesiano):

Select Cod_Pac, Nome_Pac, Cod_Med, Nome_Med
From Paciente, Medico

Paciente x Medico

Cod_Pac	Nome_Pac	Cod_Med	Nome_Med
P1	Matias	M1	Armando
P2	Gilberto	M1	Armando
P3	Lia	M1	Armando
P1	Matias	M2	Denise
P2	Gilberto	M2	Denise
P3	Lia	M2	Denise

A	b
1	One
Null	Three
4	Join4

c	D
Null	Two
4	four

```
SELECT *
FROM table1 t1 JOIN table2 t2
  ON t1.a = t2.c
ORDER BY t1.a
```

A	B	c	D
1	One	Null	Two
1	One	4	four
Null	Three	Null	Two
Null	Three	4	four
4	Join4	Null	Two
4	Join4	4	four

A	b	c	D
4	Join4	4	four

Q18 – FGV – TCESE – Segurança da Informação – 2015

49 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

(A)
```
select *
from X FULL JOIN Y on X.a=Y.c
```
(B)
```
select *
from X LEFT JOIN Y on X.a=Y.c
```


Q18 – FGV – TCESE – Segurança da Informação – 2015

49 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

(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
```

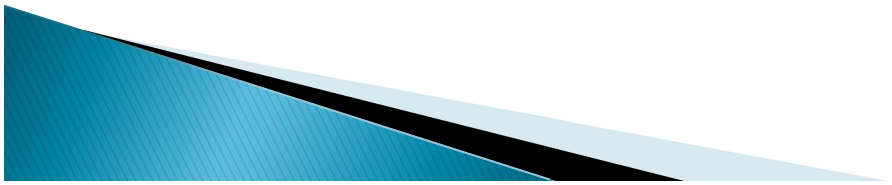
X	
a	b
1	2
3	3
4	5
5	7

Y	
c	d
1	2
3	4
5	6
7	8
9	1

(E)

```
select *
from X INNER JOIN Y on X.a=Y.c
```

a	b	c	d
1	2	1	2
3	3	3	4
5	7	5	6



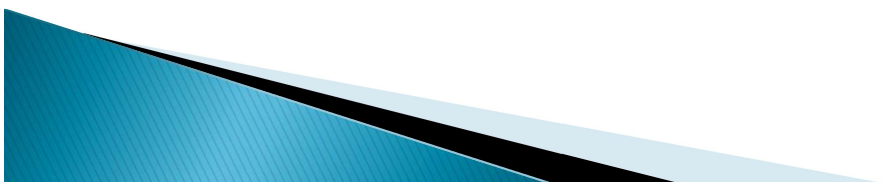
X	
a	b
1	2
3	3
4	5
5	7

Y	
c	d
1	2
3	4
5	6
7	8
9	1

(D)

```
select *
from X CROSS JOIN Y on X.a=Y.c
```

a	b	c	d
1	2	1	2
3	3	3	4
5	7	5	6



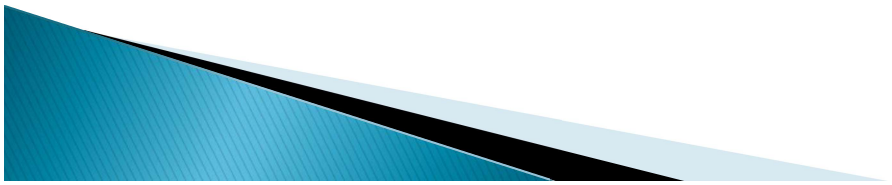
X	
a	b
1	2
3	3
4	5
5	7

Y	
c	d
1	2
3	4
5	6
7	8
9	1

(C)

```
select *
from Y RIGHT JOIN X on X.a=Y.c
```

a	b	c	d
1	2	1	2
3	3	3	4
5	7	5	6
4	5	*NULL*	*NULL*



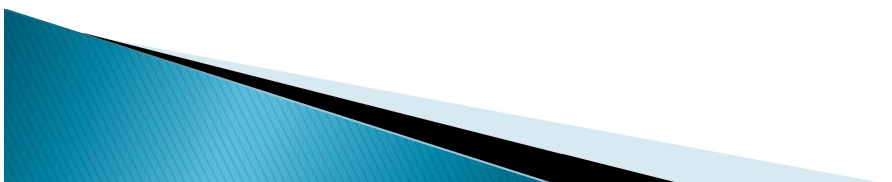
X	
a	b
1	2
3	3
4	5
5	7

Y	
c	d
1	2
3	4
5	6
7	8
9	1

(B)

```
select *
from X LEFT JOIN Y on X.a=Y.c
```

a	b	c	d
1	2	1	2
3	3	3	4
5	7	5	6
4	5	*NULL*	*NULL*



X	
a	b
1	2
3	3
4	5
5	7

Y	
c	d
1	2
3	4
5	6
7	8
9	1

(A)

```
select *
from X FULL JOIN Y on X.a=Y.c
```

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

Q19 – FGV – PGE-RO – Micro – 2015

27 Sabe-se que as tabelas T1 e T2 têm, cada uma:

- 1.000 registros;
- 5 colunas;
- uma coluna intitulada “A” totalmente preenchida com valores nulos.

Pode-se concluir que o comando SQL

```
select *
from T1 x left join T2 y on x.A=y.A
```

produz um resultado que contém:

(A) 5 colunas e 1 linha;
(B) 10 colunas e 1 linha;
(C) 5 colunas e 1.000 linhas;
(D) 10 colunas e 1.000 linhas;
(E) 9 colunas e 1.000.000 de linhas.

Q19 – FGV – PGE-RO – Micro – 2015

27 Sabe-se que as tabelas T1 e T2 têm, cada uma:

- 1.000 registros;
- 5 colunas;
- uma coluna intitulada "A" totalmente preenchida com valores nulos.

Pode-se concluir que o comando SQL

```
select *
from T1 x left join T2 y on x.A=y.A
```

produz um resultado que contém:

(A) 5 colunas e 1 linha;
(B) 10 colunas e 1 linha;
(C) 5 colunas e 1.000 linhas;
(D) 10 colunas e 1.000 linhas;
(E) 9 colunas e 1.000.000 de linhas.

Q20 – FGV – TJSC – Analista de Sistemas – 2015

36 Considerando duas tabelas relacionais R e S, tal que haja uma chave primária definida para cada uma delas e que a instância de R contenha um número maior de registros que a instância de S, analise os comandos SQL a seguir.

I. I. select * from r full outer join s on r.a = s.a
II. select * from r left outer join s on r.a = s.a union
select * from r right outer join s on r.a = s.a

Sabendo-se que as instâncias de R e S não são vazias, é correto concluir que:

(A) os resultados de I e II contêm as mesmas linhas;
(B) o resultado de I contém mais linhas que o de II;
(C) o resultado de II contém mais linhas que o de I;
(D) os resultados de I e II não possuem interseção e têm o mesmo número de linhas;
(E) os resultados de I e II possuem interseção e não têm o mesmo número de linhas.

Q20 – FGV – TJSC – Analista de Sistemas – 2015

I. select * from r full outer join s on r.a = s.a

r

a	b
1	1
2	2
3	3

s

a	c
1	1
1	2
4	5

a	b	a	c
1	1	1	1
1	1	1	2
2	2	null	Null
3	3	null	Null
null	null	4	5

Q20 – FGV – TJSC – Analista de Sistemas – 2015

II. select * from r left outer join s on r.a = s.a union
select * from r right outer join s on r.a = s.a

r

a	b
1	1
2	2
3	3

s

a	c
1	1
1	2
4	5

a	b	a	c
1	1	1	1
1	1	1	2
2	2	null	Null
3	3	null	null

a	b	a	c
1	1	1	1
1	1	1	2
null	Null	4	5

a	b	a	c
1	1	1	1
1	1	1	2
2	2	null	Null
3	3	null	null
null	Null	4	5

Q20 – FGV – TJSC – Analista de Sistemas – 2015

36 Considerando duas tabelas relacionais R e S, tal que haja uma chave primária definida para cada uma delas e que a instância de R contenha um número maior de registros que a instância de S, analise os comandos SQL a seguir.

I. I. select * from r full outer join s on r.a = s.a

II. select * from r left outer join s on r.a = s.a union
select * from r right outer join s on r.a = s.a

Sabendo-se que as instâncias de R e S não são vazias, é correto concluir que:

(A) os resultados de I e II contêm as mesmas linhas;

(B) o resultado de I contém mais linhas que o de II;

(C) o resultado de II contém mais linhas que o de I;

(D) os resultados de I e II não possuem interseção e têm o mesmo número de linhas;

(E) os resultados de I e II possuem interseção e não têm o mesmo número de linhas.

Q21 – FGV – COMPESA – 2014

49 Observe a utilização de junções externas nos dois comandos SQL

mostrados abaixo.

```
select *
from x left join y on x.a=y.a
UNION
select *
from x right join y on x.a=y.a

select *
from x full join y on x.a=y.a
```

Assinale a opção que indica a condição necessária e suficiente para que os dois comandos sempre produzam o mesmo resultado.

```
select *
from x left join y on x.a=y.a
UNION
select *
from x right join y on x.a=y.a

select *
from x full join y on x.a=y.a
```

Assinale a opção que indica a condição necessária e suficiente para que os dois comandos sempre produzam o mesmo resultado.
(A) Não haja duplicação de registros nas instâncias de “x” e “y”.
(B) As duas tabelas possuam o mesmo número de registros.
(C) Nenhum dos valores da coluna “a” da tabela “x” seja nulo (NULL).
(D) Nenhuma das tabelas possua uma instância vazia.
(E) Uma das tabelas possua mais registros que a outra.

II. select * from r full outer join s on r.a = s.a

r

a	b
1	1
2	2
3	3
3	3
3	3

s

a	c
1	1
1	2
4	5

a	b	a	c
1	1	1	1
1	1	1	2
2	2	null	Null
3	3	null	null
3	3	null	null
3	3	null	null
null	Null	4	5

II. select * from r left outer join s on r.a = s.a union
select * from r right outer join s on r.a = s.a

r

a	b
1	1
2	2
3	3
3	3
3	3

s

a	c
1	1
1	2
4	5

a	b	a	c
1	1	1	1
1	1	1	2
2	2	null	Null
3	3	null	null
3	3	null	null
3	3	null	null

a	b	a	c
1	1	1	1
1	1	1	2
null	Null	4	5

a	b	a	c
1	1	1	1
1	1	1	2
2	2	null	Null
3	**3**	**null**	**null**
null	Null	4	5

```
select *
from x left join y on x.a=y.a
UNION
select *
from x right join y on x.a=y.a

select *
from x full join y on x.a=y.a
```

Assinale a opção que indica a condição necessária e suficiente para que os dois comandos sempre produzam o mesmo resultado.

(A) Não haja duplicação de registros nas instâncias de “x” e “y”.
(B) As duas tabelas possuam o mesmo número de registros.
(C) Nenhum dos valores da coluna “a” da tabela “x” seja nulo (NULL).
(D) Nenhuma das tabelas possua uma instância vazia.
(E) Uma das tabelas possua mais registros que a outra.

Q22 – FGV – TJBA – Tec Inform Reaplicada – 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.

Q22 – FGV – TJBA – Tec Inform Reaplicada – 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.

Q23 – FGV – TCESE – Desenvolvimento – 2015

X	
a	b
1	2
3	3
4	5
5	7

Y	
c	d
1	2
3	4
5	6
7	8
9	1

```
select x.a from x
where not exists
  (select * from y where y.c = x.a+1)
```

Considerando-se as tabelas e o comando SQL mostrados acima, é correto concluir que esse comando produz um resultado com uma única coluna contendo somente o(s) valor(es):

(A) 4
(B) 3, 4
(C) 1, 3, 5
(D) 3, 4, 5
(E) 1, 3, 4, 5

Q23 – FGV – TCESE – Desenvolvimento – 2015

X	
a	b
1	2
3	3
4	5
5	7

Y	
c	d
1	2
3	4
5	6
7	8
9	1

```
select x.a from x
where not exists
  (select * from y where y.c = x.a+1)
```

Considerando-se as tabelas e o comando SQL mostrados acima, é correto concluir que esse comando produz um resultado com uma única coluna contendo somente o(s) valor(es):

(A) 4
(B) 3, 4
(C) 1, 3, 5
(D) 3, 4, 5
(E) 1, 3, 4, 5

Q24 – FGV – TCESE – Desenvolvimento – 2015

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

Q24 – FGV – TCESE – Desenvolvimento – 2015

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

CESGRANRIO – BNDES – Suporte – 2013

Q25 – 67 – O modelo relacional a seguir representa um banco de dados simplificado de uma empresa de comércio. As chaves estão sublinhadas.

CLIENTE(NomeC, EnderecoC)

PRODUTO(NomeP)

FORNECEDOR(NomeF)

PRODUZ(NomeF,NomeP,Preco)

PEDIDO(NomeC,NomeF,NomeP,Quantidade)

Se o dono da empresa deseja saber quais clientes nunca pediram um produto do fornecedor cujo nome é "Barateira", que consulta SQL deve fazer?

CLIENTE(NomeC, EnderecoC)

PRODUTO(NomeP)

FORNECEDOR(NomeF)

PRODUZ(NomeF,NomeP,Preco)

PEDIDO(NomeC,NomeF,NomeP,Quantidade)

Se o dono da empresa deseja saber quais clientes nunca pediram um produto do fornecedor cujo nome é "Barateira", que consulta SQL deve fazer?

(A)
```
SELECT * FROM CLIENTE
 WHERE CLIENTE. NOMEC IN
    (SELECT NOMEC FROM PEDIDO WHERE NOMEF="Barateira")
```

(B)
```
SELECT * FROM CLIENTE
 WHERE CLIENTE.NOMEC NOT IN
    (SELECT NOMEC FROM PEDIDO WHERE NOMEF="Barateira")
```

(C)
```
SELECT * FROM CLIENTE
 WHERE CLIENTE.NOMEC=PEDIDO.NOMEC AND
   CLIENTE.NOMEC NOT IN
      (SELECT NOMEC FROM PEDIDO WHERE NOMEF="Barateira")
```

CLIENTE(NomeC, EnderecoC)
PRODUTO(NomeP)
FORNECEDOR(NomeF)
PRODUZ(NomeF,NomeP,Preco)
PEDIDO(NomeC,NomeF,NomeP,Quantidade)

Se o dono da empresa deseja saber quais clientes nunca pediram um produto do fornecedor cujo nome é "Barateira", que consulta SQL deve fazer?

(D) **SELECT * FROM CLIENTE,PEDIDO**
WHERE CLIENTE.NOMEC=PEDIDO.NOMEC AND
CLIENTE.NOMEC IN
(SELECT NOMEC FROM PEDIDO WHERE NOMEF="Barateira")

(E) **SELECT * FROM CLIENTE,PEDIDO**
WHERE CLIENTE.NOMEC=PEDIDO.NOMEC AND
NOMEF<>"Barateira"

CLIENTE(NomeC, EnderecoC)
PRODUTO(NomeP)
FORNECEDOR(NomeF)
PRODUZ(NomeF,NomeP,Preco)
PEDIDO(NomeC,NomeF,NomeP,Quantidade)
Se o dono da empresa deseja saber quais clientes nunca pediram um produto do fornecedor cujo nome é "Barateira", que consulta SQL deve fazer?

(A) **SELECT * FROM CLIENTE**
WHERE CLIENTE. NOMEC IN
(SELECT NOMEC FROM PEDIDO WHERE NOMEF="Barateira")

(B) **SELECT * FROM CLIENTE**
WHERE CLIENTE.NOMEC NOT IN
(SELECT NOMEC FROM PEDIDO WHERE NOMEF="Barateira")

(C) **SELECT * FROM CLIENTE**
WHERE CLIENTE.NOMEC=PEDIDO.NOMEC AND
CLIENTE.NOMEC NOT IN
(SELECT NOMEC FROM PEDIDO WHERE NOMEF="Barateira")

Q26 – CESPE – Banco da Amazonia – 2012

Em SQL, para se listarem os nomes dos departamentos da tabela departamento que não possuem colaboradores, é correto utilizar o código abaixo

```
SELECT nomedepartamento
FROM departamento d
EXCEPT
(SELECT nomedepartamento
FROM departamento d, colaborador c
WHERE d.codigodepartamento =
c.codigodepartamento);
```

Q26 – CESPE – Banco da Amazonia – 2012

C Em SQL, para se listarem os nomes dos departamentos da tabela departamento que não possuem colaboradores, é correto utilizar o código abaixo


```
SELECT nomedepartamento
FROM departamento d
EXCEPT
(SELECT nomedepartamento
FROM departamento d, colaborador c
WHERE d.codigodepartamento =
c.codigodepartamento);
```

Q27 – FCC – TRT19 – 2011

Uma tabela A contém apenas o código (cod_pac) de todos os pacientes internados em um hospital. Uma tabela B, com a mesma estrutura, contém o código (cod_pac) de todos os pacientes deste hospital que estão internados há mais de 15 dias. Para listar o código de todos os pacientes deste hospital internados por um período menor ou igual a 15 dias, deve-se utilizar a seguinte instrução SQL:

a) SELECT cod_pac FROM A DIFF SELECT cod_pac FROM B

b) SELECT cod_pac FROM (A UNION B)

c) SELECT cod_pac FROM A where
cod_pac not in (SELECT cod_pac FROM B)

d) SELECT cod_pac FROM A where cod_pac in (A MINUS B)

e) SELECT cod_pac FROM (A DIFF B)

Q27 – FCC – TRT19 – 2011

Uma tabela A contém apenas o código (cod_pac) de todos os pacientes internados em um hospital. Uma tabela B, com a mesma estrutura, contém o código (cod_pac) de todos os pacientes deste hospital que estão internados há mais de 15 dias. Para listar o código de todos os pacientes deste hospital internados por um período menor ou igual a 15 dias, deve-se utilizar a seguinte instrução SQL:

a) SELECT cod_pac FROM A DIFF SELECT cod_pac FROM B

b) SELECT cod_pac FROM (A UNION B)

c) SELECT cod_pac FROM A where
cod_pac not in (SELECT cod_pac FROM B)

d) SELECT cod_pac FROM A where cod_pac in (A MINUS B)

e) SELECT cod_pac FROM (A DIFF B)

Q28 - FCC - MPE-AP - 2012

```
CREATE TABLE Livro (ISBN INT, Nome VARCHAR(40),Autor INT, Editora INT);
CREATE TABLE Autor (Codigo INT, NOME VARCHAR(40));
CREATE TABLE Editora (Codigo INT, Nome VARCHAR(40));
INSERT INTO Livro VALUES (12345, "Programas em C",1,1);
INSERT INTO Livro VALUES (67890, "Métodos Ágeis",1,2);
INSERT INTO Autor VALUES (1, "Manoel da Silva");
INSERT INTO Editora VALUES (1, "Editora Livros");
```

Note que os exemplos abaixo consideram que as linhas apresentadas acima já foram executadas. Para receber como resultado apenas a string Programas em C, é necessário executar o comando

a) SELECT Nome FROM Editora WHERE Editora.Codigo = Livro.Editora AND Autor.Codigo = Livro.Autor;
b) SELECT b.Nome FROM Autor a, Livro c, Editora c WHERE a.Autor = b.Codigo AND a.Editora = c.Codigo;
c) SELECT * FROM Livro a WHERE (SELECT Codigo FROM Autor WHERE Codigo = a.Autor) AND (SELECT Codigo FROM Editora WHERE Codigo = a.Editora);
d) SELECT Nome FROM Livro WHERE Autor IN (SELECT Codigo FROM Autor) AND Editora IN (SELECT Codigo FROM Editora);
e) SELECT * FROM Livro WHERE Livro.Autor = (SELECT Codigo FROM Autor) AND Livro.Editora = (SELECT Codigo FROM Editora);

Q28 - FCC - MPE-AP - 2012

CREATE TABLE Livro (ISBN INT, Nome VARCHAR(40),Autor INT, Editora INT);
CREATE TABLE Autor (Codigo INT, NOME VARCHAR(40));
CREATE TABLE Editora (Codigo INT, Nome VARCHAR(40));
INSERT INTO Livro VALUES (12345, "Programas em C",1,1);
INSERT INTO Livro VALUES (67890, "Métodos Ágeis",1,2);
INSERT INTO Autor VALUES (1, "Manoel da Silva");
INSERT INTO Editora VALUES (1, "Editora Livros");

Note que os exemplos abaixo consideram que as linhas apresentadas acima já foram executadas. Para receber como resultado apenas a string Programas em C, é necessário executar o comando

a) SELECT Nome FROM Editora WHERE Editora.Codigo = Livro.Editora AND Autor.Codigo = Livro.Autor;
b) SELECT b.Nome FROM Autor a, Livro c, Editora c WHERE a.Autor = b.Codigo AND a.Editora = c.Codigo;
c) SELECT * FROM Livro a WHERE (SELECT Codigo FROM Autor WHERE Codigo = a.Autor) AND (SELECT Codigo FROM Editora WHERE Codigo = a.Editora);
d) SELECT Nome FROM Livro WHERE Autor IN (SELECT Codigo FROM Autor) AND Editora IN (SELECT Codigo FROM Editora);
e) SELECT * FROM Livro WHERE Livro.Autor = (SELECT Codigo FROM Autor) AND Livro.Editora = (SELECT Codigo FROM Editora);

Q29 – CESPE – ANTAQ – 2014

Acerca de sistemas de gerenciamento de banco de dados relacional (SGBDR), julgue o item a seguir.

Considerando-se a inexistência de concorrência de transação, a existência de permissão de leitura; e considerando-se também que as tabelas A e B, além de possuírem o atributo ID, estejam disponíveis, é correto afirmar que a execução do comando SQL mostrado abaixo terá como resultado uma quantidade de registros igual à soma dos registros das duas tabelas.

```
SELECT ID FROM A
UNION
SELECT ID FROM B
```

Q29 – CESPE – ANTAQ – 2014

Acerca de sistemas de gerenciamento de banco de dados relacional (SGBDR), julgue o item a seguir.

E Considerando-se a inexistência de concorrência de transação, a existência de permissão de leitura; e considerando-se também que as tabelas A e B, além de possuírem o atributo ID, estejam disponíveis, é correto afirmar que a execução do comando SQL mostrado abaixo terá como resultado uma quantidade de registros igual à soma dos registros das duas tabelas.

```
SELECT ID FROM A
UNION
SELECT ID FROM B
```

Gabarito

Q1 – B
Q2 – C
Q3 – B
Q4 – C
Q5 – A
Q6 – C
Q7 – E
Q8 – E
Q9 – E
Q10 – B
Q11 – E
Q12 – D
Q13 – B
Q14 – A
Q15 – D
Q15.1 – C
Q16 – D
Q17 – D
Q18 – A
Q19 – D
Q20 – A
Q21 – A
Q22 – A
Q23 – C
Q24 – E
Q25 – B
Q26 – C
Q27 – C
Q28 – D
Q29 – E