Gabarito

Utilize este gabarito apenas para conferir as respostas dos exercícios propostos. Elas são meramente ilustrativas e não necessariamente as melhores ou as únicas respostas existentes.

CAPÍTULO 1

1. Banco de dados é o conjunto de tabelas, que por sua vez é o conjunto de registros; registro é o conjunto de campos; e campo é a menor unidade de armazenamento.

2. O modelo relacional de banco de dados foi desenvolvido pelo cientista inglês Edgar Frank "Ted" Codd.

3. SQL significa Structured Query Language, ou seja, Linguagem Estruturada de Consulta.

4. Armazena os dados em forma de tabela.

5. SGBD: Sistema de Gerenciamento de Banco de Dados; SDBMS: Relational Data Base Management System.

6. Foi o cientista estadunidense Donald D. Chamberlin.

7. Foi a IBM.

8.

- ADF (Application Development Facility);
- ADRS II (A Departmental Reporting System);
- AS (Application System);
- CSP (Cross System Product Set);
- DMS (Development Management System);
- GIS (Generalized Information System);
- IC/1 (Information Center/1);
- QBE (Query By Example);
- QMF (Query Management Facility);
- TIF (The Information Facility).

CAPÍTULO 3

1. A finalidade é criar um banco de dados.

2. Colocar em uso um banco de dados existente.

3. A finalidade é remover um banco de dados existente.

4. Criar uma nova tabela em um banco de dados existente.

5. A finalidade é efetuar a entrada de registros em uma tabela.

6. A finalidade é fazer uma consulta e apresentar os registros armazenados em determinada tabela.

7. A finalidade é atualizar os registros em determinada tabela.

8. A finalidade é remover (apagar) registros de determinada tabela.

9. A finalidade é alterar a estrutura de determinada tabela.

10. SELECT NOME, FUNCAO FROM cadfun;.

11. SELECT CODFUN, NOME, FUNCAO FROM cadfun;.

12. SELECT CODFUN, NOME, FUNCAO, DEPTO FROM cadfun;.

13. SELECT NOME FROM cadfun WHERE DEPTO = '5';.

14. SELECT NOME, DEPTO FROM cadfun WHERE FUNCAO = 'VENDEDOR';.

15. SELECT SALARIO FROM cadfun WHERE DEPTO = '3';.

16. SELECT NOME, FUNCAO FROM cadfun ORDER BY FUNCAO DESC;.

17. SELECT FUNCAO, NOME FROM cadfun ORDER BY FUNCAO, NOME DESC;.

18. SELECT * FROM cadfun WHERE MONTH(ADMISSAO) = 10;.

CAPÍTULO 4

1. Podem ser utilizados: adição (+), subtração (–), multiplicação (*), divisão (/) e resto de divisão (%).

2. Devem-se usar os parênteses.

3. São: > (maior que), < (menor que), = (igual a), <> (diferente de), >= (maior ou igual a), <= (menor ou igual a) e NULL.

4. São: AND (operador de conjunção), OR (operador de disjunção) e NOT (operador de negação).

5. São: IS NULL (para verificar um campo ou uma coluna vazios), BETWEEN (para verificar um valor em uma faixa de valores), IN (para verificar se um valor existe na tabela) e LIKE (para verificar um valor que faz busca por semelhança).

6. Podem ser usados com SELECT, UPDATE e DELETE.

7. SELECT CODFUN, NOME, SALARIO + 250 FROM cadfun;.

8. SELECT CODFUN, NOME, SALARIO – SALARIO * 0.075 FROM cadfun;.

9. SELECT * FROM cadfun WHERE FUNCAO = 'ANALISTA';.

10. SELECT * FROM cadfun WHERE SALARIO >= 1700;.

11. SELECT * FROM cadfun WHERE SALARIO > 1700;.

12. SELECT * FROM cadfun WHERE SALARIO < 1700;.

13. SELECT * FROM cadfun WHERE SALARIO = 1700;.

14. SELECT * FROM cadfun WHERE (FUNCAO = 'ANALISTA') OR (FUNCAO = 'PROGRAMADOR');.

15. SELECT * FROM cadfun WHERE (FUNCAO = 'ANALISTA') OR (FUNCAO = 'PROGRAMADOR') AND (SALARIO > 1200);.

16. SELECT * FROM cadfun WHERE NOT (FUNCAO = 'ANALISTA') AND NOT (FUNCAO = 'PROGRAMADOR');.

17. SELECT * FROM cadfun WHERE FILHOS BETWEEN 2 AND 4;.

18. SELECT * FROM cadfun WHERE (FILHOS BETWEEN 2 AND 4) AND (SALARIO < 2000);.

19. SELECT * FROM cadfun WHERE FILHOS NOT BETWEEN 2 AND 3;.

20. SELECT * FROM cadfun WHERE (FILHOS NOT BETWEEN 2 AND 3) AND (FILHOS <> 0);.

21. SELECT * FROM cadfun WHERE FILHOS IN (2, 3);.

22. SELECT * FROM cadfun WHERE NOME LIKE '%SILVA';.

23. SELECT * FROM cadfun WHERE NOME LIKE 'SILV%';.

24. SELECT NOME FROM cadfun WHERE NOME LIKE '%SANTOS';.

25. SELECT NOME, DEPTO FROM cadfun WHERE (FUNCAO = 'GERENTE') OR (FUNCAO = 'ANALISTA');.

26. SELECT CODFUN, NOME, DEPTO FROM cadfun WHERE (CODFUN = 2) OR (CODFUN = 5) OR (CODFUN = 9);.

27. SELECT NOME, DEPTO FROM cadfun WHERE DEPTO <> '5';.

28. SELECT * FROM cadfun WHERE NOME LIKE '%SILVA%';.

29. SELECT * FROM cadfun WHERE SALARIO <> 2000;.

CAPÍTULO 5

1. Foram apresentadas funções de agregação, data e hora, numéricas e *strings*.

2. O ideal são as funções numéricas.

3. Devem-se usar as funções de agregação.

4.: Foram apresentadas as funções AVG(), COUNT(), MAX(). MIN() e SUM().

5. Foram apresentadas as funções DATEADD(), DATEDIFF(), DATENAME(), DATEPART(), DAY(), GETDATE(), MONTH() e YEAR().

6. Foram apresentadas as funções ABS(), ACOS(), ASIN(), ATAN(), COS(), DEGREES(), EXP(), LOG(),PI(), POWER(), RADIANS(), ROUND(), SIN(), SQRT() e TAN().

7. Foram apresentadas as funções ASCII(), LOWER(), LEFT(), LEN() e UPPER().

8. Qual é o número médio de filhos por funcionário da empresa?

Resposta: SELECT AVG(FILHOS) FROM cadfun;.

9. SELECT AVG(FILHOS) FROM cadfun WHERE DEPTO = '5';.

10. SELECT SUM(SALARIO) FROM cadfun WHERE DEPTO = '5';.

11. SELECT COUNT(*) FROM cadfun WHERE DEPTO = '5';.

12. SELECT MAX(SALARIO) FROM cadfun WHERE DEPTO = '5';.

13. SELECT SUM(SALARIO) FROM cadfun WHERE FUNCAO = 'ANALISTA';.

14. SELECT SUM(SALARIO) FROM cadfun WHERE (FUNCAO = 'ANALISTA') AND (DEPTO = '2');.

15. SELECT NOME FROM cadfun WHERE DAY(ADMISSAO) = 10;.

16. SELECT NOME, ADMISSAO FROM cadfun WHERE (DAY(ADMISSAO) >= 5) AND (DAY(ADMISSAO) <= 10);.

17.: SELECT NOME, ADMISSAO FROM cadfun WHERE (DAY(ADMISSAO) >= 5) AND (DAY(ADMISSAO) <= 10) ORDER BY ADMISSAO;.

18. SELECT NOME, ADMISSAO FROM cadfun WHERE (DAY(ADMISSAO) >= 5) AND (DAY(ADMISSAO) <= 10) ORDER BY ADMISSAO DESC;.

19. SELECT * FROM cadfun WHERE ADMISSAO < DATEADD(Month, 1, '2006-09-20');.

20. SELECT NOME, ADMISSAO, DATENAME(Month, ADMISSAO) FROM cadfun WHERE depto = '5';.

21. SELECT LOWER(NOME), DEPTO FROM cadfun WHERE (DEPTO = '3') OR (DEPTO = '5');

22. SELECT NOME, DEPTO, ADMISSAO FROM cadfun WHERE (DAY(ADMISSAO) >= 5) AND (DAY(ADMISSAO) <= 10) AND (DEPTO = '3') OR (DEPTO = '5');.

23. SELECT NOME, DEPTO, FUNCAO, ADMISSAO FROM cadfun WHERE (DAY(ADMISSAO) >= 1) AND (DAY(ADMISSAO) <= 15) AND (FUNCAO = 'ANALISTA');.

24. SELECT NOME, DEPTO, FUNCAO, ADMISSAO FROM cadfun WHERE (MONTH(ADMISSAO) >= 5) AND (MONTH(ADMISSAO) <= 8) AND (FUNCAO = 'ANALISTA');.

25. SELECT NOME, DEPTO, FUNCAO, ADMISSAO FROM cadfun WHERE (MONTH(ADMISSAO) >= 5) AND (MONTH(ADMISSAO) <= 8) AND NOT (FUNCAO = 'ANALISTA');.

26. SELECT COUNT(*) FROM cadfun WHERE MONTH(ADMISSAO) = 7;.

CAPÍTULO 6

1.O agrupamento de dados é, segundo Stephens,Plew & Jones (2011, p. 153), “o processo de combinação de colunas com valores repetidos em uma ordem lógica”.

2. Permitir a definição e o uso do agrupamento de dados.

3. A cláusula GROUP BY deve ser utilizada após a cláusula WHERE e antes da cláusula ORDER BY.

4. É a união de mais de uma consulta em uma única por meio de múltiplos comandos SELECT.

5. SELECT FUNCAO, COUNT(*) FROM cadfun GROUP BY FUNCAO;.

6. SELECT FILHOS, COUNT(*) FROM cadfun WHERE FILHOS > 0 GROUP BY FILHOS;.

7. SELECT * FROM cadfun WHERE DEPTO = '5' UNION SELECT * FROM morto WHERE DEPTO = '5' ORDER BY NOME;.

8. SELECT * FROM cadfun WHERE FUNCAO = 'VENDEDOR' UNION SELECT * FROM morto WHERE FUNCAO = 'VENDEDOR' ORDER BY NOME;.

9. SELECT DEPTO, COUNT(*) FROM cadfun WHERE NOME LIKE '%SILVA' GROUP BY DEPTO;

CAPÍTULO 7

1. Uma junção de tabela possibilita, por meio de um campo comum entre as tabelas existentes, gerar consultas como se as tabelas relacionadas (múltiplas tabelas) na consulta fossem uma só. Uma visualização é uma tabela virtual com base nos dados de uma tabela real. A partir dos dados de uma única tabela real, é possível ter várias visões para facilitar as operações de consultas.

2. Para determinar o relacionamento entre tabelas por meio de junções, é necessário ter no mínimo duas tabelas que possuam algum campo em comum.

3. SELECT cliente.NOME, cliente.ESTADO, venda.DUPLIC, venda.VALOR FROM cliente, venda WHERE (cliente.CODCLI = venda.CODCLI) AND (cliente.ESTADO = 'SP') ORDER BY cliente.NOME;.

4. SELECT cliente.NOME, venda.DUPLIC, venda.VALOR FROM cliente, venda WHERE cliente.CODCLI = venda.CODCLI AND cliente.NOME = 'MICROS INFORMATICA S/A';.

5. SELECT cliente.NOME FROM cliente, venda WHERE (cliente.CODCLI = venda.CODCLI) AND (MONTH(venda.VENCTO) = 4);.

6. SELECT cliente.NOME, COUNT(*) FROM cliente, venda WHERE cliente.CODCLI = venda.CODCLI GROUP BY cliente.NOME;.

7. SELECT YEAR(venda.VENCTO), COUNT(*) FROM cliente, venda WHERE cliente.CODCLI = venda. CODCLI GROUP BY YEAR(venda.VENCTO);.

8. SELECT MONTH(venda.VENCTO), YEAR(venda.VENCTO), COUNT(*) FROM cliente, venda WHERE cliente.CODCLI = venda. CODCLI GROUP BY YEAR(venda.VENCTO), MONTH(venda.VENCTO);.

9. SELECT cliente.NOME, venda.VALOR, venda.VENCTO FROM cliente, venda WHERE cliente.CODCLI = venda.CODCLI AND VENCTO BETWEEN '2000-01-01' AND '2003-12-31' ORDER BY cliente.NOME;.

CAPÍTULO 8

1. O índice é uma estrutura de arquivo complementar que armazena o campo-chave de determinada tabela e o local onde aquele campo é encontrado na tabela. Segundo Stephens,Plew & Jones (2011, p. 255) "um índice em um banco de dados é muito semelhante ao índice na parte de trás de um livro", ou seja, semelhante ao índice remissivo. No índice remissivo, é possível localizar uma palavra-chave e a página em que se encontra o assunto relacionado àquela palavra-chave selecionada.

2. Os índices de uma tabela podem ser classificados em simples, exclusivo e composto.

3. O índice simples baseia-se na definição do valor de apenas uma coluna (campo).

4. O índice exclusivo baseia-se na definição de um valor que não pode ser repetido na tabela.

5. O índice composto baseia-se em valores de mais de uma coluna.

6. CREATE [UNIQUE] INDEX <índice> ON <tabela (campos)>;

7. DROP INDEX <índice> ON <tabela>;

8. Um índice deve ser evitado em tabelas pequenas ou em colunas (campos) que tenham um grande número de valores NULL. Deve-se também evitar o uso de índices em colunas que sofrem alto impacto de atualização, pois degradam a velocidade de acesso aos dados.

9. O campo de chave primária é usado para evitar a duplicidade de registros.

10. É o campo que armazena determinado valor que poderia ser considerado campo de chave primária. Normalmente se utiliza um campo de chave candidata para armazenar valores como CPF e CNPJ, que são exclusivos para cada pessoa física ou jurídica e não podem ser repetidos para mais de uma entidade de registro.

11. A chave estrangeira dá a possibilidade de estabelecer o relacionamento lógico de determinada tabela (em que se tem definido um campo-chave) com o campo de chave primária de outra tabela.

12. Há três formas de relacionamentos (cardinalidade) que podem ser utilizadas, sendo:

- relacionamento de 1 para 1;
- relacionamento de 1 para N ou de N para 1 quando for o caso;
- relacionamento de N para N.

13. O relacionamento de 1 para 1 (um para um) representado pela cardinalidade (1:1) é a relação de um campo-chave de um registro de determinada tabela vinculado a um campo-chave de um outro registro de outra tabela.

14. O relacionamento de 1 para N (um para muitos) representado pela cardinalidade (1:N) é a relação de um campo-chave de um registro de determinada tabela vinculado ao campo-chave de muitos registros de outra tabela.

15. O relacionamento de N para N (muitos para muitos) representado pela cardinalidade (N:N) é a relação de um campo-chave de muitos registros de determinada tabela vinculado ao campo-chave de muitos registros de outra tabela.

16. Usa-se a cláusula FOREIGN KEY.

17. Os dois conceitos são muito diferentes. A relação chave primária x chave estrangeira é realizada por vínculos explicitamente definidos entre os campos das tabelas envolvidas na relação. Já uma junção é uma relação estabelecida de forma lógica no momento de determinada consulta.