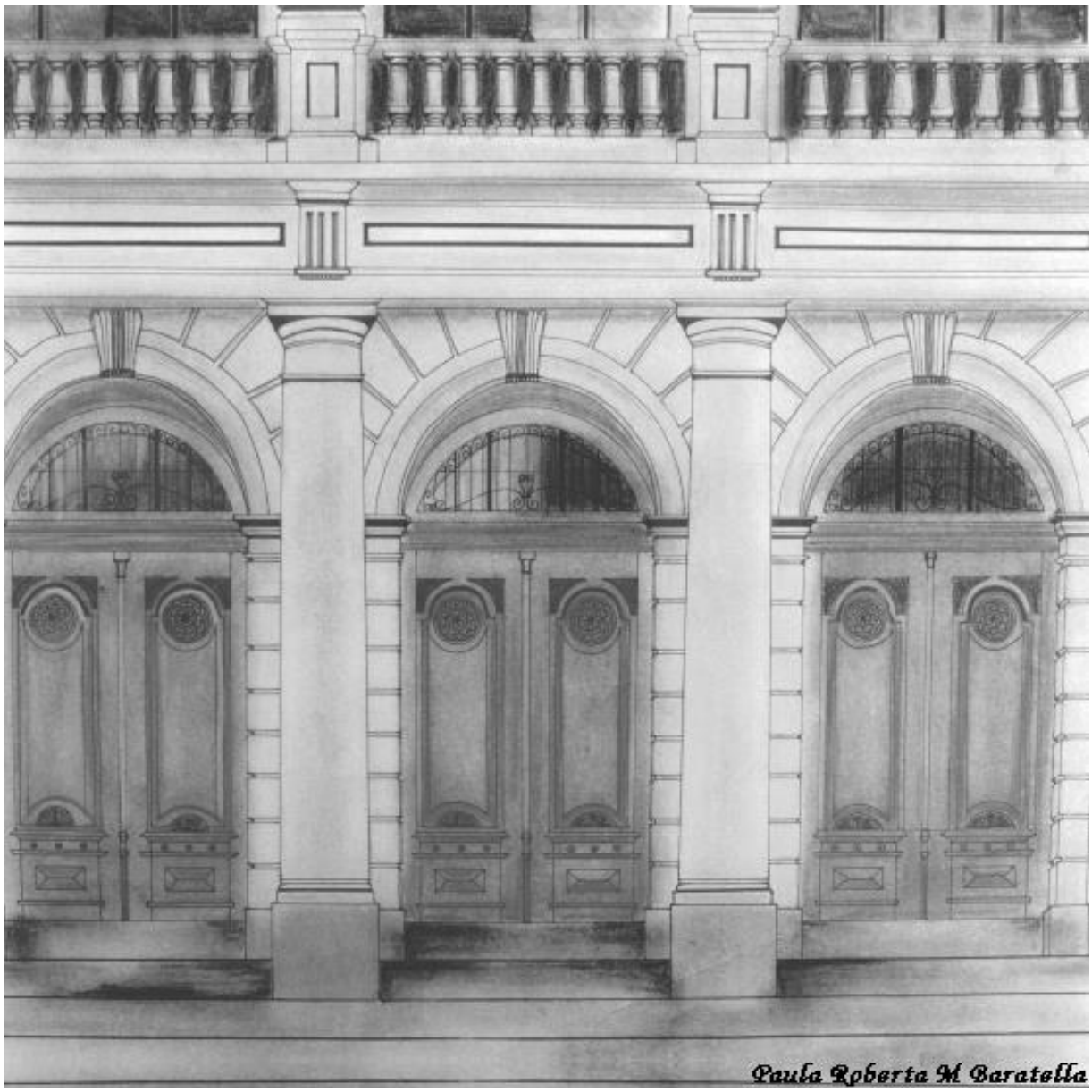

UNICAMP - Colégio Técnico de Campinas
Departamento de Processamento de Dados

Bancos de Dados

Volume 1: DDL+DML+SQL

1º Período do Curso 19
1s2024

Prof. André Luís dos R.G. de Carvalho

Índice

SQL (Structured Query Language)

Introdução

1. SQL é uma linguagem de consulta estruturada a bancos de dados.
2. Toda estrutura SQL deriva da álgebra e do cálculo relacional.
3. O usuário especifica o resultado desejado e não o método de obter o resultado.
4. O formato dos comando SQL é livre, podendo usar uma ou mais linhas.

Objetivos

SQL é uma linguagem unificada de dados para:

1. Definição de dados (criar tabelas, visões);
2. Acesso a dados (consultas);
3. Manipulação de dados (operações);
4. Controle de acesso a dados (definir em que parte do BD o usuário pode manipular).

Modalidades de Uso

SQL existe em duas modalidades:

1. Como linguagem autônoma, interativa, permitindo que através de um terminal ou computador o usuário acesse diretamente os dados que procura;
2. Como sub linguagem de programação, interna à uma outra linguagem como COBOL, Dbase, que se encarregue das estruturas básicas de programação, da formatação de telas, impressão de relatórios, etc.

Vantagens do Modo Programado

Dentre as vantagens de utilizar SQL em modo programado, podemos citar:

1. SQL torna os programas muito mais legíveis e portanto bem mais fáceis de depurar e manter;

2. SQL produz um código significativamente mais compacto que qualquer linguagem procedural; e

3. Toda boa implementação de SQL tem performance superior à das linguagens procedurais na execução das tarefas de processamento às quais SQL se destina (definição, manipulação e controle de dados).

Linguagens Hospedeiras

Ao usar SQL junto com uma outra linguagem, devemos utilizar desta outra linguagem o que ela tem de melhor e que não existe em SQL, ou seja, construções de programação estruturada como comandos condicionais, comandos iterativos, funções de manipulação de strings e datas, comandos de vídeo, comandos de rede, comandos de configuração de ambiente, comandos de formatação de relatório, etc.

Há uma diferença importante entre as duas linguagens, com reflexo direto na construção de programas. SQL se baseia em conjuntos para acessar os dados, e faz esse acesso de modo não procedimental, ou seja, determina os dados necessários e deixa a cargo dos recursos internos da linguagem a localização desses dados.

Linguagens convencionais, por sua vez, obrigam o programador a dizer quais as áreas de trabalho utilizadas, quais os tabelas abertos e quais os registros desejados, por meio de instruções procedimental que a cada passo determina o que deve ser feito.

Além disso, linguagens convencionais não manipulam sobre conjunto de dados, uma vez que trabalha registro a registro (o que é caracterizado pelo uso de um ponteiro de tabela para indicar sempre o registro afetado pela operação em curso).

Na prática, isso significa que existe um ponto de incompatibilidade entre as duas linguagens, e no entanto, não tem maiores conseqüências. Basta que o programador não misture no mesmo programa comandos de criação e acesso aos tabelas de ambas as linguagens, e não inclua em programas que contenham instruções SQL e comandos e funções de linguagens convencionais (voltadas para o posicionamento de ponteiros do tabela), uma vez que o conceito de ponteiros não tem aplicação dentro de SQL.

Quase tudo que é realizado interativamente no SQL pode ser incluído dentro de programas. No entanto, em modo programado, dispõe-se também de alguns comandos novos.

Catálogos

É um grupo de tabelas e visões que contém informações sobre o Banco de Dados. Elas são criadas quando o Banco de Dados é criado.

O catálogo de dados descreve tabelas, colunas, índices, usuários, privilégios de acesso e outros objetos.

Pode-se ler tabelas do catálogo com os comandos SELECT padrões.

Tipos de Dados no SQL Server 7.0

Números

TINYINT	Representa valores inteiros não negativos entre 0 e 255, ocupando 1 byte.
SMALLINT	Representa valores inteiros entre –32.768 e 32.767, ocupando 2 bytes.
INT **ou** **Integer**	Representa valores inteiros entre –2.147.483.648 e 2.147.483.647, ocupando 4 bytes.
BIGINT	Representa valores inteiros entre –9.223.372.036.854.775.808 e 9.223.372.036.854.775.807, ocupando 8 bytes.
DEC (T[,D]) **ou** **Decimal (T[,D])**	Representa números reais de ponto fico com T dígitos, sendo destes, D dígitos decimais (ponto incluso).
NUMERIC (T[,D])	Sinônimo de DEC (T[,D]).
REAL	Representa números reais de ponto flutuante com valores valores negativos entre $–3,40.10^{-38}$ e $–1,18.10^{+38}$ e positivos entre $1,18.10^{-38}$ e $3,40.10^{+38}$ (o valor zero também pode ser armazenado), ocupando

8 bytes.

FLOAT Representa números reais de ponto flutuante com valores negativos entre $-2{,}23.10^{-308}$ e $-1{,}79.10^{+308}$ e valores positivos entre $1{,}79.10^{-308}$ e $2.23.10^{+308}$ (o valor zero também pode ser armazenado), ocupando 8 bytes.

FLOAT (D) Representa números reais de ponto flutuante com mantissa ocupando D bits ($1 \leq D \leq 53$; sinônimo de REAL se D=24 e sinônimo de FLOAT, se D=53).

MONEY Sinônimo de DEC(19,4), ocupando 8 bytes.

SMALLMONEY Sinónimo de DEC(10,4), ocupando 4 bytes.

String

CHAR[(T)]
ou
Character[(T)] Representa uma cadeia de caracteres (possivelmente constituída de letras, dígitos e caracteres especiais) de tamanho fixo T (no máximo 8000) caracteres. O valor padrão de T é 1.

VARCHAR[(T)]
ou
Char Varying[(T)]
ou
Character Varying[(T)] Representa uma cadeia de caracteres (possivelmente constituída de letras, dígitos e caracteres especiais) de tamanho variável limitado a no máximo T (no máximo 8000) caracteres. O valor padrão de T é 1.

NCHAR[(T)] Representa uma cadeia de caracteres UNICODE (possivelmente constituída de letras, dígitos e caracteres especiais) de tamanho fixo T (no máximo 4000) caracteres. O valor padrão de T é 1.

NVARCHAR[(T)] Representa uma cadeia de caracteres UNICODE (possivelmente constituída de letras, dígitos e caracteres especiais) de tamanho variável limitado a no máximo T (no máximo 4000) caracteres. O valor padrão de T é 1.

TEXT Representa uma cadeia de caracteres de tamanho fixo de até 2GB de comprimento.

NTEXT Representa uma cadeia de caracteres UNICODE de tamanho fixo de até 2GB de comprimento.

Binários

BINARY[(T)] Representa uma cadeia de bits de tamanho fixo igual a T (no máximo 8000) bytes. O valor padrão de T é 1.

VARBINARY[(T)] Representa uma cadeia de bits de tamanho variável limitado a no máximo T (no máximo 8000) bytes. O valor padrão de T é 1.

IMAGE[(T)] Representa uma cadeia de bits de tamanho fixo T (no máximo 2GB) bytes. O valor padrão de T é 1.

BIT Representa um valor lógico (verdadeiro ou falso), ocupando 1 bit.

Data e Hora

DATETIME Representa uma data e hora, ocupando um espaço equivalente a dois campos int, um para a data e outro para a hora; guarda na data a quantidade de dias decorridos desde 1º de Janeiro de 1753 até 31 de dezembro de 9999; guarda na hora a quantidade de unidades equivalentes a 3 centésimos de segundo decorreram desde a 0h.

SMALLDATETIME Representa uma data e hora, ocupando um espaço equivalente a um inteiro, contendo tanto a data, quanto a hora; guarda na data a quantidade de dias decorridos desde 1º de Janeiro de 1900 até 06 de junho de 2079; guarda na hora a quantidade de minutos decorridos desde a 0h.

Tipos de Dados Derivados

TIMESTAMP Representa um número único em todo o banco de dados; o valor de uma coluna com este domínio é automaticamente atualizado sempre que uma que a linha que o contem for inserida ou atualizada; uma coluna com este domínio que tiver sido definida com a cláusula NOT NULL, equivale a um BINARY(8), ao passo que, uma coluna com esse domínio que não tiver sido definida com a cláusula NOT NULL, equivale a um VARBINARY(8); todo banco de dados tem um TIMESTAMP associado a ele e que é armazenado na variável @@DBTS.

SYSNAME Representa o nome de um objeto de banco de dados no catálogo do sistema, ocupando um espaço equivalente a um NVARCHAR(128).

Novos Tipos de Dados

CURSOR Permite a definição num procedimento armazenado de uma variável que permite o processamento de uma linha por vez.

UNIQUEIDENTIFIER Representa um número de identificação exclusivo que é armazenado na forma de uma cadeia de bits de 16 bytes; a iniciação de uma variável ou coluna deste domínio pode ser feita usando-se a função NEWID, ou então com uma constante string com um formato especial envolvendo hífens e dígitos hexadecimais.

Funções Escalares do SQL Server 7.0

Numéricas

1. **ABS (E):**
   Retorna o valor absoluto da expressão E;

2. **ACOS (N):**
   Retorna o ângulo, em radianos, cujo cosseno é N;

3. **ASIN (N):**
   Retorna o ângulo, em radianos, cujo seno é N;

4. **ATAN (N):**
   Retorna o ângulo, em radianos, cuja tangente é N;

5. **ATN2 (I,F):**
   Retorna o ângulo, em radianos, cuja tangente se encontra entre I e F;

6. **CEILING (N):**
   Retorna o menor valor inteiro maior ou igual a N;

7. **COS (A):**
   Retorna o cosseno do ângulo A expresso em radianos;

8. **COT (A):**
   Retorna a contangente do ângulo A expresso em radianos;

9. **DEGREES (R):**
   Converte R, expresso em radianos, em graus;

10. **EXP (N):**
    Retorna o valor e^n;

11. **FLOOR (N):**
    Retorna o maior inteiro menor ou igual a N;

12. **LOG (N):**
    Retorna o logaritmo natural de N;

13. **LOG10 (E):**
    Retorna o logaritmo na base 10 de N;

14. PI ( ):

Retorna o valor do número pi;

15. POWER (B, E):

Retorna o valor de B^E;

16. RADIAN (G):

Converte G, expresso em graus, em radianos;

17. RAND ( ):

Retorna um número aleatório entre 0 (inclusive) e 1 (exclusive);

18. ROUND (N,P):

Retorna o valor arredondado de N com precisão P;

19. SIN (A):

Retorna o seno do ângulo A expresso em radianos;

20. SIGN (N):

Retorna +1, se N>0, ou –1, se N<0;

21. SQRT (N):

Retorna a raiz quadrada de N;

22. TAN (A):

Retorna a tangente do ângulo A expresso em radianos.

De Data

Todas as funções de data usam uma das seguintes unidades: yy (ano), qq (trimestre), mm (mês), dy (dia do ano), dd (dia do mês), dw (dia da semana), wk (semana), hh (hora), mi (minuto), ss (segundo) ou ms (milissegundo).

1. **DATEPART (U, D):**

Retorna a parte da data D especificada pela unidade U;

2. **DATENAME (U, D):**
   Retorna o nome da parte da data D especificada pela unidade U (se a parte em questão não possuir um nome, retorna o mesmo que a função DATEPART);

3. **DATEDIFF (U, D1, D2):**
   Retorna, na unidade U, a diferença entre as datas D1 e D2;

4. **DATEADD (U, N, D):**
   Retorna, uma data igual a N unidades U somadas à data D;

5. **GETDATE ():**
   Retorna a data e hora atuais do sistema.

De String

1. **ASCII (C):**
   Retorna código ASCII do caractere C;

2. **CAST (E as T[(C)]):**
   Converte, se possível, a expressão E para o tipo T;

3. **CHAR (C):**
   Retorna o caractere que tem código ASCII igual a C;

4. **CONVERT (T[(C)], E):**
   Converte, se possível, a expressão E para o tipo T;

5. **CHARINDEX (S1, S2[, P]):**
   Retorna a posição em S2 onde se pode encontrar S1 como sub string de S2; retorna 0 se S1 não for sub string de S2; se P for fornecido, faz a busca a partir da posição P;

6. **DIFFERENCE (S1, S2):**
   Retorna a diferença de valores SOUNDEX dos strings S1 e S2;

7. **LOWER (S):**
   Retorna uma string equivalente a S com todas suas letras maiúsculas substituídas por letras minúsculas;

8. **LTRIM (S):**
   Retorna uma string equivalente a S com todos os espaços em branco iniciais removidos;

9. **NEWID ( ):**
   Retorna um ID exclusivo que consiste em uma string binária de 16 bytes destinada a ser armazenada em uma variável ou coluna que tenha o domínio UNIQUEIDENTIFIER;

10. **PATINDEX (P, S):**
    Retorna a posição em S onde se pode encontrar o padrão P; P pode conter expressões regulares e caracteres coringa (%); retorna 0 se o padrão P não for encontrado em S;

11. **REPLICATE (S, Q):**
    Retorna uma string contendo Q réplicas da string S concatenadas;

12. **REVERSE (S):**
    Retorna a string S com seus caracteres invertidos;

13. **RIGTH (S, Q):**
    Retorna os Q últimos caracteres de S;

14. **RTRIM (S):**
    Retorna uma string equivalente a S com todos os espaços em branco finais removidos;

15. **SOUNDEX (S):**
    Retorna o código SOUNDEX da string S;

16. **SPACE (Q):**
    Retorna uma string contendo Q espaços em branco;

17. **STR (F [,T [,D]]):**
    Retorna o número real de ponto flutuante F na forma de uma string com no máximo T caracteres, sendo destes, D caracteres para a parte fracionária de F, incluindo o ponto decimal;

18. STUFF (S1, P, T, S2):

Retorna uma string equivalente a S1 com T caracteres a partir da posição P substituídos por S2;

19. SUBSTRING (S, P, Q):

Retorna uma string constituída0 por Q caracteres de S a partir da posição P;

20. UPPER (S):

Retorna uma string equivalente a S com todas suas letras minúsculas substituídas por letras maiúsculas;

De Texto/Imagem

1. **WRITETEXT T.C P [WITH LOG] X:**

   Escreve na coluna C (TEXT, NTEXT ou IMAGE) da tabela T o texto X. P representa uma variável previamente declarada e iniciada. Por exemplo: suponha que decidamos chamar a variável representada por P de @VAR, a tabela representada por T de TAB e a coluna representada por C de COL. Neste caso, a mesma teria que ter sido declarada com o comando DECLARE @VAR BINARY(16) e iniciada com o comando SELECT @VAR=TEXTPTR(COL) FROM TAB WHERE <condição determinística>.

2. **READTEXT T.C P I Q [HOLDLOCK]:**

   Le a coluna C (TEXT, NTEXT ou IMAGE) da tabela T da seguinte forma: (a) posiciona na posição I da coluna C da tabela T (I começa de zero); e (b) le Q posições a partir da posição corrente (zero significa 4 e null significa todos).

   P representa uma variável previamente declarada e iniciada. Por exemplo: suponha que decidamos chamar a variável representada por P de @VAR, a tabela representada por T de TAB e a coluna representada por C de COL. Neste caso, a mesma teria que ter sido declarada com o comando DECLARE @VAR BINARY(16) e iniciada com o comando SELECT @VAR=TEXTPTR(COL) FROM TAB WHERE <condição determinística>.

3. **UPDATETEXT T.C P I Q [WITH LOG] [X]:**

   Atualiza a coluna C (TEXT, NTEXT ou IMAGE) da tabela T da seguinte forma: (a) posiciona na posição I da coluna C da tabela T (I começa de zero e, se for null, é

interpretado como significando o final da coluna); (b) remove Q posições a partir da posição corrente (zero significa nenhum e null significa todos); e (c) insere na posição corrente o texto X, caso o mesmo tenha sido fornecido.

P representa uma variável previamente declarada e iniciada. Por exemplo: suponha que decidamos chamar a variável representada por P de @VAR, a tabela representada por T de TAB e a coluna representada por C de COL. Neste caso, a mesma teria que ter sido declarada com o comando DECLARE @VAR BINARY(16) e iniciada com o comando SELECT @VAR=TEXTPTR(COL) FROM TAB WHERE <condição determinística>.

4. **TEXTPTR (C):**
   Retorna um ponteiro que aponta para o local no qual a coluna C (TEXT, NTEXT ou IMAGE) está armazenada; o resultado desta função pode ser usado com as funções UPDATETEXT, WRITETEXT E READTEXT;

5. **TEXTVALID (' T.C ', P):**
   Retorna 1, se P for um ponteiro válido para a coluna C da tabela T (TEXT, NTEXT ou IMAGE), e 0, caso contrário.

De Sistema

1. **COALESCE (E1, E2,...):**
   Retorna o valor da primeira das expressões E_i que não for NULL;

2. **COL_LENGTH (TV, C):**
   Retorna o comprimento máximo que pode ter a coluna C da tabela ou visão TV;

3. **COL_NAME (TV, C):**
   Retorna o nome da coluna com ID igual a C da tabela ou visão que tem ID TV;

4. **DATALENGTH (E):**
   Retorna o comprimento em bytes da coluna E;

5. **DB_ID ([N]):**
   Retorna o ID do banco de dados de nome N; se N não for especificado, retorna o ID do banco de dados corrente;

6. **DB_NAME ([I]):**
   Retorna o nome do banco de dados que tem ID igual a I; se I não for especificado, retorna o nome do banco de dados corrente;

7. **GETANSINULL ('N'):**
   Retorna 1 se o uso de valores nulos no banco de dados de nome N obedece o padrão ANSI-SQL;

8. **HOST_ID ([M]):**
   Retorna o ID da máquina M;

9. **HOST_NAME ([M]):**
   Retorna o nome da máquina M;

10. **INDEX_COL (T, I, N):**
    Retorna o nome da coluna indexada da tabela T, com ID de índice I e número dentro do índice N;

11. **ISNULL (E, V):**
    Retorna o valor da expressão E, se este valor não for NULL, retorna o valor V, caso contrário;

12. **NULLIF (E1, E2):**
    Retorna NULL se o valor da expressões E1 e E2 forem iguais;

13. **OBJECT_ID (N):**
    Retorna o ID do objeto de nome N;

14. **OBJECT_NAME (I):**
    Retorna o nome do objeto que tem ID igual a I;

15. **SUSER_SID ([L]):**
    Retorna o ID de segurança (SID) do usuário com login L; se L não for especificado, retorna o SID do usuário corrente;

16. SUSER_NAME ([SID]):

Retorna o login do usuário que tem ID de segurança igual a SID; se SID não for especificado, retorna o login do usuário corrente;

17. USER_ID ([L]):

Retorna o ID do usuário com login L; se L não for especificado, retorna o ID do usuário corrente;

18. USER_NAME ([ID]):

Retorna o login do usuário que identificado pelo identificador ID; se ID não for especificado, retorna o login do usuário corrente;

19. CURRENT_TIMESTAMP:

Retorna a data e a hora do sistema;

20. CURRENT_USER:

Retorna o nome do usuário corrente;

21. SYSTEM_USER:

Retorna o ID de login do usuário corrente;

22. USER:

Retorna o nome do usuário corrente.

Operadores do SQL Server 7.0

Operadores Aritméticos

Os operadores aritméticos do SQL Server 7.0 são, em sua maioria, aqueles que usualmente encontramos nas linguagens de programação imperativas. Todos eles operam sobre números e resultam números. São eles:

1. + (mais unário);
2. − (menos unário);
3. + (soma);

4. - (subtração);
5. * (multiplicação);
6. / (divisão); e
7. % (resto da divisão inteira).

Operadores Relacionais

Os operadores relacionais do SQL Server 7.0 são, em sua maioria, aqueles que usualmente encontramos nas linguagens de programação imperativas. Todos eles operam sobre números e resultam valores lógicos. São eles:

1. = (igualdade);
2. <> ou != (desigualdade);
3. < (inferioridade);
4. <= ou !> (inferioridade ou igualdade);
5. > (superioridade); e
6. >= ou !< (superioridade ou igualdade).

Operadores Lógicos

Os operadores lógicos do SQL Server 7.0 são, em sua maioria, aqueles que usualmente encontramos nas linguagens de programação imperativas. Todos eles operam sobre valores lógicos e resultam valores lógicos. São eles:

1. AND (conjunção);
2. OR (disjunção); e
3. NOT (negação).

Operadores de Bit

Os operadores de bit do SQL Server 7.0 são, em sua maioria, aqueles que usualmente encontramos nas linguagens de programação imperativas. Todos eles operam sobre valores integrais e resultam valores integrais. São eles:

1. & (and bit a bit);
2. | (or bit a bit);
3. ^ (xor bit a bit); e
4. ~ (not bit a bit).

Variáveis Globais do SQL Server 7.0

1. **@@CONNECTIONS:**
   Retorna a quantidade de tentativas de login desde a iniciação do SQL Server;

2. **@@CPU_BUSY:**
   Retorna o tempo total de CPU (em unidades de 3 milissegundos) usado desde a iniciação do SQL Server;

3. **@@DBTS:**
   Retorna o TIMESTAMP do banco de dados atual;

4. **@@ERROR:**
   Retorna a informação de se ocorreu ou não um erro na execução do último comando;

5. **@@IDENTITY:**
   Retorna o último valor inserido em uma coluna com propriedade IDENTITY;

6. **@@IDLE:**
   Retorna o tempo total (em unidades de 3 milissegundos) de ociosidade do SQL Server desde sua primeira iniciação;

7. **@@IO_BUSY:**
   Retorna o tempo total (em unidades de 3 milissegundos) que o SQL Server dispendeu em operações de E/S desde sua iniciação;

8. **@@LANGID:**
   Retorna o ID da língua que está sendo utilizada pelo SQL Server;

9. **@@LANGUAGE:**
   Retorna o nome da língua que está sendo utilizada pelo SQL Server;

10. **@@MAX_CONNECTIONS:**
    Retorna a quantidade máxima de conexões simultâneas que o SQL Server aceita;

11. **@@VERSION:**
    Retorna a versão do SQL Server;

12. **@@NESTLEVEL:**
    Retorna o nível de aninhamento do procedimento armazenado que está em execução;

13. **@@PROCID:**
    Retorna o ID do procedimento armazenado que está em execução;

14. **@@ROWCOUNT:**
    Retorna a quantidade de linhas que foram afetadas pelo último comando executado;

15. **@@SERVERNAME:**
    Retorna o nome do SGBD ao qual se está conectado;

16. **@@SPID:**
    Retorna o ID do processo servidor;

17. **@@TEXTSIZE:**
    Retorna a quantidade máxima de bytes que pode ser retornada correntemente por um comando SELECT ao selecionar um campo com domínio TEXT/IMAGE;

18. @@TOTAL_READ:

Retorna a quantidade total de operações de leitura que foram executadas desde que o SQL Server foi iniciado pela primeira vez;

19. @@TOTAL_WRITE:

Retorna a quantidade total de operações de escrita que foram executadas desde que o SQL Server foi iniciado pela primeira vez;

Principais Comandos do SQL Server 7.0

A seguir, abordaremos os principais comandos do SQL Server 7.0.

Todos os exemplos apresentados nesta seção foram elaborados tendo em mente o BD descrito no apêndice BD dos Exemplos de Linguagens de Consulta Teóricas e SQL.

Comandos de Manutenção Estrutural

a) Criação de um Banco de Dados:

- Sintaxe:

```
CREATE DATABASE <nome do bd>
```

- Exemplo:

```
CREATE DATABASE Cartorio
```

b) Eliminação de um Banco de Dados:

- Sintaxe:

```
DROP DATABASE <nome do bd>
```

- Exemplo:

```
DROP DATABASE Cartorio
```

c) Criação de Tipos:

- Sintaxe:

```
sp_addtype <nome do novo tipo>,'<nome do tipo existente>'
        [,'NULL'|'NOT NULL']
```

- Exemplo:

```
sp_addtype Cidade, 'varchar(30)'
```

  - Observações:

    A cláusula NULL, que é asumida em caso de omissão, permite que colunas deste tipo fiquem sem valor, ao passo que a cláusula NOT NULL impede que isto aconteça;

d) Criação de Valores Padrão:

- Sintaxe:

```
CREATE DEFAULT <nome do padrão> AS <expressão constante>
```

- Exemplo:

```
CREATE DEFAULT Cidade_Padrao AS 'Campinas'
```

e) Indicando um Valor Padrão para um Tipo

- Sintaxe:

```
sp_binddefault '<nome do padrão>', '<nome do tipo>'
```

- Exemplo:

```
sp_binddefault 'Cidade_Padrao', 'Cidade'
```

f) Dissociando um Valor Padrão de um Tipo

- Sintaxe:

```
sp_unbinddefault '<nome do tipo>'
```

- Exemplo:

```
sp_unbinddefault 'Cidade'
```

g) Eliminação de Valores Padrão:

- Sintaxe:

```
DROP DEFAULT <nome do padrão>
```

- Exemplo:

```
CREATE DEFAULT Cidade_Padrao
```

h) Eliminação de Tipos:

- Sintaxe:

```
sp_droptype '<nome do tipo>'
```

- Exemplo:

```
sp_droptype 'Cidade'
```

i) Criação de Tabelas:

- Sintaxe:

```
CREATE TABLE [<nome do bd>[.<dono>.].|<dono>.]
             <nome da tabela> (

<definição de coluna calculada>   |
<definição de coluna>   |
<restrição de tabela>

[,...,

<definição de coluna>   |
<definição de coluna calculada>   |
<restrição de tabela>]

Onde:

- <definição de coluna calculada> teria a forma

  <nome da coluna> AS <expressão>

- <definição de coluna> teria a forma

  <nome da coluna> <nome do tipo>
  [DEFAULT <expressão constante> |
   IDENTITY(<valor inicial>,<valor do incremento>)]
  [<restrição de coluna>
  [,...,
   <restrição de coluna>]

- <restrição de coluna> teria a forma

  NULL|NOT NULL   |
```

```
PRIMARY KEY|UNIQUE   |

FOREIGN KEY
REFERENCES <nome da tabela> [(<nome da coluna>)]
[ON DELETE CASCADE|NO ACTION]
[ON UPDATE CASCADE|NO ACTION]]   |

CHECK (condição)
```

- <restrição de tabela> teria a forma

```
CONSTRAINT <nome da restrição>

seguida de

PRIMARY KEY|UNIQUE
(<nome da coluna> [,..., <nome da coluna>])   |


FOREIGN KEY
(<nome da coluna> [,..., <nome da coluna>])
REFERENCES <nome da tabela>
[(<nome da coluna> [,..., <nome da coluna>])]
[ON DELETE CASCADE|NO ACTION]
[ON UPDATE CASCADE|NO ACTION]]   |

CHECK (<condição>)
```

- Exemplo:

```
CREATE TABLE Pessoa
(Numero INT NOT NULL IDENTITY(1,1),
 Nome VARCHAR(50) NOT NULL,
 CONSTRAINT CP_Pessoa PRIMARY KEY (Numero))
```

j) Alterando a Estrutura de uma Tabela

- Sintaxe:

```
ALTER TABLE <nome da tabela>

seguida de

ALTER COLUMN
<nome da coluna> <novo tipo> [NULL|NOT NULL]
[,...,
<nome da coluna> <novo tipo> [NULL|NOT NULL]]   |

ADD
```

```
<definição de coluna calculada>|<definição de coluna>
[,...,
<definição de coluna calculada>|<definição de coluna>   |

DROP
<nome da coluna>[,...,<nome da coluna>]

ADD
<restrição de tabela>[,...,<restrição de tabela>]

DROP CONSTRAINT
<nome da restrição>[,...,<nome da restrição>
```

Onde:

- <definição de coluna calculada> teria a forma

```
<nome da coluna> AS <expressão>
```

- <definição de coluna> teria a forma

```
<nome da coluna> <nome do tipo>
[DEFAULT <expressão constante> |
 IDENTITY(<valor inicial>,<valor do incremento>)]
[<restrição de coluna>
[,...,
 <restrição de coluna>]
```

- <restrição de coluna> teria a forma

```
NULL|NOT NULL   |

PRIMARY KEY|UNIQUE
[CLUSTERED|NONCLUSTERED]
WITH FILLFACTOR=<valor>   |

FOREIGN KEY
REFERENCES <nome da tabela> [(<nome da coluna>)]
[ON DELETE CASCADE|NO ACTION]
[ON UPDATE CASCADE|NO ACTION]]   |

CHECK (condição)
```

- <restrição de tabela> teria a forma

```
CONSTRAINT <nome da restrição>

seguida de

PRIMARY KEY|UNIQUE
(<nome da coluna> [,..., <nome da coluna>])   |
```

```
FOREIGN KEY
(<nome da coluna> [,..., <nome da coluna>])
REFERENCES <nome da tabela>
[(<nome da coluna> [,..., <nome da coluna>])]
[ON DELETE CASCADE|NO ACTION]
[ON UPDATE CASCADE|NO ACTION]]   |

CHECK (<condição>)
```

- Exemplo 1:

```
ALTER TABLE Pessoas
ALTER COLUMN Nome VARCHAR(30) NOT NULL
```

- Exemplo 2:

```
ALTER TABLE Pessoas
ADD Sexo VARCHAR(1) NOT NULL
```

- Exemplo 3:

```
ALTER TABLE Pessoas
ADD CONSTRAINT DF_Sexo_Pessoa
DEFAULT ('m')
FOR Sexo
```

- Exemplo 4:

```
ALTER TABLE Pessoas
ADD CONSTRAINT PK_Pessoa
PRIMARY KEY (Numero)
```

- Exemplo 5:

```
ALTER TABLE Nascimentos
ADD CONSTRAINT FK_Filho_Nascimento
FOREIGN KEY (Filho)
REFERENCES Pessoa (Numero)
```

- Exemplo 6:

```
ALTER TABLE Pessoas
ADD CONSTRAINT CK_Sexo_Pessoa
CHECK (Sexo IN ('M','F'))
```

- Exemplo 7:

```
ALTER TABLE Pessoas
DROP CONSTRAINT DF_Sexo_Pessoa
```

- Observações :

  - [CLUSTERED|NONCLUSTERED] é um parâmetro opcional que faz com que a ordem física das linhas da tabela indexada seja ou não seja a mesma provocada pela indexação; em caso de omissão, assume-se NONCLUSTERED;

- FILLFACTOR especifica a porcentabem de espaço livre que deve haver em cada página do índice;

k) Eliminação de Tabelas:

- Sintaxe:

```
DROP TABLE <nome da tabela>
```

- Exemplo :

```
DROP TABLE Casamentos
```

l) Criação de Índices:

- Sintaxe:

```
CREATE [UNIQUE] [CLUSTERED|NONCLUSTERED]
INDEX <nome do indice>
ON <nome da tabela>|<nome da visão> (
<nome da coluna> [ASC|DESC]
[,...,
<nome da coluna> [ASC/DESC]])
[WITH <opção de índice>[,...,<opção de índice>]]
```

```
Onde:

- <opção de índice> teria a forma

   PADINDEX   |
   FILLFACTOR=<valor>   |
   IGNORE_DUP_KEY   |
   DROP_EXISTING   |
   STATISTICS_NORECOMPUTE   |
   SORT_IN_TEMPBD
```

- Exemplo :

```
CREATE INDEX Idx_Nome
ON Pessoa
(Nome DESC)
```

- Observações :

- As diversas colunas especificadas na sintaxe servem para desempatar a primeira coluna índice;
- Não é possível misturar chaves ascendentes e descendentes em um mesmo índice;
- Índices são criados automaticamente quando se indica chaves primárias ou estrangeiras;
- Índices só podem estar associado a tabelas e nunca a visões;
- `[UNIQUE]` é um parâmetro opcional que impede valores idênticos para a(s) coluna(s) da chave de indexação em linhas distintas;
- [CLUSTERED|NONCLUSTERED] é um parâmetro opcional que faz com que a ordem física das linhas da tabela indexada seja ou não seja a mesma provocada pela indexação; em caso de omissão, assume-se NONCLUSTERED;
- `[ASC|DESC]` indica que o índice imprimirá ordem ascendente ou descendente nas linhas da tabela indexada; em caso de omissão, assume-se ASC;
- `PAD_INDEX` especifica o espaço livre que deve haver em cada página do índice; é útil apenas quando FILLFACTOR for especificado, já que usa a porcentagem especificada por FILLFACTOR como base de cálculo;
- FILLFACTOR especifica a porcentabem de espaço livre que deve haver em cada página do índice;
- IGNORE_DUP_KEY especifica o que deverá acontecer quando um comando provocar a existência de valores idênticos para a(s) coluna(s) da chave de indexação em linhas distintas. Se IGNORE_DUP_KEY for especificado, o fato será ignorado, ao passo que se IGNORE_DUP_KEY não for especificado, será reportado um erro e o comando que provocou a situação terá seus efeitos desfeitos;
- Se DROP_EXISTING for especificado, um eventual índice homônimo já existente será removido e reconstruído;

- Se STATISTICS_NORECOMPUTE for especificado, as estatísticas relativas ao índice não deverão ser automaticamente recalculadas. Isto pode levar ao uso de formas não otimizada de acesso à tabela indexada;
- Se SORT_IN_TEMPBD for especificado, os dados intermiediários gerados para produzir o índice serão armazenados no banco de dados TempBD.

m) Eliminação de Índices:

- Sintaxe:

```
DROP INDEX <nome da tabela>.<nome do indice>
```

- Exemplo:

```
DROP INDEX Pessoa.Nome
```

n) Criação de Visões:

Visão é uma tabela virtual, ou seja, uma tabela que não existe fisicamente em disco, mas que funciona praticamente como se existisse.

Cada visão pode basear se em diversas outras tabelas e visões, assim como conter colunas resultantes de expressões.

A visão será atualizada toda vez que as tabelas ou visões em que se baseiam forem atualizadas.

- Sintaxe:

```
CREATE VIEW <nome da visão>
AS <comando select>
[WITH CHECK OPTION]
```

- Exemplo:

```
Create View Homens As
(Select *
 From Pessoas
 Where Sexo = 'M')
```

- Observações:

- O comando SELECT especificado na cláusula `AS` seleciona colunas, linhas e tabelas que comporão a visão;
- Qualquer comando SELECT válido pode ser usado na cláusula `AS`, desde que não contenha uma cláusula `ORDER BY`;
- O comando SELECT pode envolver outras visões;
- Pode-se usar uma visão do mesmo modo que uma tabela;
- Pode-se atualizar informações diretamente nas visões, bem como através das tabelas que as definem;
- Para que uma visão seja atualizável o select que a definiu deve obedecer certas restrições: (1) deve-se referir a uma única tabela; (2) não pode conter cláusulas `GROUP BY` e `DISTINCT` e nem funções de grupo;
- `[WITH CHECK OPTION]` é uma cláusula opcional que, quando usada impede a inclusão de linhas que não atendam aos critérios da instrução de definição da visão.

o) Eliminação de Visões:

- Sintaxe:

```
DROP VIEW <nome da visão>
```

- Exemplo:

```
DROP VIEW Homens
```

p) Consulta aos Objetos de Banco de Dados criados:

- Sintaxe Geral:

```
SELECT Name from SysObjects Where xType = <tipo>
```

- Observações:

```
<tipo> pode ser: (1)  'C', para restrições; (2) 'D', para
valores padrão; (3) 'F', para chave estrangeira; (4) 'P',
```

para procedimentos armazenados; (5) 'K', para chave primária ou restrição de valor único; (6) 'R', para regras; (7) 'S', para tabelas de sistema; (8) 'TR', para triggers; (9) 'U', para tabelas de usuário; e (10) 'V', para visões.

- Exemplo:

```
SELECT Name FROM SYSOBJECTS WHERE xType = 'U'
```

q) Obtenção de Informações sobre um Objeto do Banco de Dados:

- Sintaxe Geral:

```
sp_help <nome do objeto>
```

- Exemplo:

```
sp_help Pessoas
```

Comandos de Manutenção da Segurança

a) Criação de Login:

- Sintaxe:

```
sp_addlogin
'<login do usuário>'
[,'<senha inicial do usuário>'
[,'<BD padrão do usuário>'
[,'<língua padrão do usuário>']]]
```

- Exemplo:

```
sp_addlogin 'andre', 'SemSenha'
```

b) Eliminação de um Login:

- Sintaxe:

```
sp_droplogin '<nome>'
```

- Exemplo:

```
sp_droplogin 'andre'
```

c) Alteração de Senha:

- Sintaxe:

```
sp_password
'<senha velha>',
'<senha nova>'
[,'<nome do usuário>']
```

- Exemplo:

```
sp_password 'SemSenha', 'Estrela'
```

- Observações:

Somente o administrador pode fornecer o parâmetro <nome do usuário> e, quando o fizer, deverá fornecer o valor null para o parâmetro <senha velha>.

a) Selecionando um Banco de Dados:

- Sintaxe:

```
USE <nome do BD>
```

- Exemplo:

```
USE Cartorio
```

d) Dando Acesso a um Banco de Dados:

- Sintaxe:

```
sp_grantdbaccess <login do usuário>
```

- Exemplo:

```
USE Cartorio
```

```
sp_grantdbaccess 'andre'
```

e) Revogando Direito de Acesso a um Banco de Dados:

- Sintaxe:

```
sp_revokedbaccess <login>
```

- Exemplo:

```
sp_revokedbaccess 'andre'
```

f) Concessão de Privilégios

Os Sistemas Gerenciadores de Banco de Dados, por definição, armazenam grandes volumes de informações.

Podemos dizer, sem risco de erros, que uma das medidas mais fiéis da qualidade de um Sistema de Gerenciamento de Banco de Dados é a proteção que ele oferece contra o acesso e o uso dessas informações não se perderão ou serão indevidamente modificadas em conseqüência do uso do sistema.

Privilégio é o nome que se dá, em SQL, à autorização fornecida a um usuário para que realize determinadas operações, tais como:

- Criar banco de dados (CREATE DATABASE)
- Criar tabela (CREATE TABLE);
- Criar índice (CREATE INDEX);
- Criar restrições de integridade (CREATE RULE);
- Criar índice (CREATE SP);
- Incluir dados (INSERT);
- Atualizar dados (UPDATE[(<nome da coluna>,...,<nome da coluna>)]);
- Eliminar dados (DELETE);
- Selecionar dados (SELECT[(<nome da coluna>,...,<nome da coluna>)]);
- Executar rotinas armazenadas (EXECUTE).

Todos os privilégios são concedidos com o comando GRANT.

- Sintaxe:

```
GRANT [ALL [PRIVILEGES]]

<especificação do privilégio>
[,...,
<especificação do privilégio>

ON [TABLE]
<nome da tabelas>|<nome da visão>
[,...,
<nome da tabelas>|<nome da visão>

TO PUBLIC   |
TO <login do usuário>[,...,<login do usuário>]

[WITH GRANT OPTION]
```

- Exemplo 1:

```
GRANT DELETE, INSERT, SELECT
ON TABLE Pessoas
TO PUBLIC
```

Dá a todos os usuários os privilégios de remover, inserir e listar linhas da tabela Pessoas.

- Exemplo 2:

```
GRANT ALL PRIVILEGES
ON TABLE Casamentos
TO Andre
WITH GRANT OPTION
```

O usuário Andre recebe todos os privilégios sobre a tabela Casamento, podendo conceder esse privilégios a outros usuários.

- Observações:
  - ALL implica a concessão de todos os privilégios disponíveis;
  - [PRIVILEGES] é um indicador opcional desnecessário, servindo apenas para tornar o comando mais claro;
  - Os <privilégio> representam a relação dos privilégios que estão sendo concedidos;
  - [TABLE] é um termo opcional de uso equivalente ao [PRIVILEGES];

- Os <tabela>|<visao> indicam as tabelas ou visões às quais os privilégios se referem;
- TO PUBLIC representa a concessão dos privilégios indicados a todos os usuários;
- Os <login do usuário> indicam os usuários a quem os privilégios estão sendo concedidos;
- [WITH GRANT OPTION] dá aos usuários a capacidade de conceder a outros usuários os mesmos privilégios que a eles foram concedidos por este comando.

g) Revogação de Privilégios

Da mesma forma que são concedidos,os privilégios podem ser revogados.

- Sintaxe:

```
REVOKE [ALL [PRIVILEGES]]

<especificação do privilégio>
[,...,
<especificação do privilégio>

ON [TABLE]
<nome da tabelas>|<nome da visão>
[,...,
<nome da tabelas>|<nome da visão>

FROM PUBLIC   |
FROM <login do usuário>[,...,<login do usuário>]
```

As cláusulas `REVOKE` têm significado análogo às cláusulas `GRANT`, sendo que a primeira revoga privilégios, enquanto a última concede.

- Exemplo:

```
REVOKE UPDATE
ON Pessoas
FROM Andre
```

Retira o privilégio de alterar a tabela notas do usuário Andre.

É interessante ressaltar que, se o privilégio concedido a um usuário com a cláusula WITH GRANT OPTION for revogado, todos os usuários a quem ele tiver repassado o privilégio perderão esse direito.

Comandos de Atualização e Consulta

a) Inclusão de uma Linha:

- Sintaxe:

```
INSERT INTO <nome da tabela>
[(<nome da coluna>
[,...,
<nome da coluna>]
VALUES
(<valor>
[,...,
<valor>]);
```

- Exemplo:

```
INSERT INTO Pessoa
(Nome, Sexo)
VALUE
('Victor Hugo Rodrigues Gomes de Carvalho', 'M')
```

b) Exclusão de Dados:

- Sintaxe:

```
DELETE FROM <nome da tabela>
[WHERE <condição>]
```

- Exemplo:

```
DELETE FROM Pessoa
WHERE Sexo = 'F'
```

- Observações:

`[WHERE <condicao>]` é uma cláusula que determina as linhas que serão eliminadas.

c) Atualização de Dados:

- Sintaxe:

```
UPDATE <nome da tabela>
SET <nome da coluna> = <valor>
[,...,
<nome da coluna> = <valor>]
[WHERE <condição>]
```

- Exemplo:

```
UPDATE Pessoa
SET Nome = "Lilia Rodrigues Gomes de Carvalho"
WHERE Nome = "Lilia de Carvalho e Silva Rodrigues"
```

d) Consultas:

- Seleção Simples:

  - Exemplo:

```
SELECT *
FROM Pessoas
WHERE Nome > 'E' and Nome < 'F'
```

  - Resultado:

```
Numero      Nome                                               Sexo
----------- -------------------------------------------------- ----
44          Edi Aparecida Cervan                               f
45          Edson de Almeida                                   m
46          Edson Kowask Bezerra                               m
47          Eliza Balboni Skuja                                f
48          Érica Cristina Olher Rubi da Cunha                 f
49          Érica Rodrigues Pereira                            f
50          Escolástica Pires Monteiro                         f

(7 row(s) affected)
```

- Projeção Simples:

  - Exemplo:

```
SELECT Data, Local
FROM Mortes
```

- Resultato:

```
Data                        Local
--------------------------- ------------------------------
1981-11-06 00:00:00.000     Taquaritinga
1988-05-20 00:00:00.000     Campinas
1921-02-14 00:00:00.000     Campinas
1964-06-06 00:00:00.000     Campinas
NULL                        Campinas
1962-09-20 00:00:00.000     São José do Rio Preto
1990-09-26 00:00:00.000     São José do Rio Preto
1996-12-22 00:00:00.000     Santos
1942-03-18 00:00:00.000     Campinas
1978-01-07 00:00:00.000     Guarulhos
1998-01-04 00:00:00.000     Brasília
1991-01-02 00:00:00.000     São José do Rio Preto
1988-06-09 00:00:00.000     Campinas
1945-01-17 00:00:00.000     Campinas
1994-09-18 00:00:00.000     São José do Rio Preto

(15 row(s) affected)
```

- Linhas iniciais:

  - Exemplo 1:

```
SELECT TOP 5
FROM Pessoas
```

  - Resultado:

```
Numero      Nome                                               Sexo
----------- -------------------------------------------------- ----
1           Adolfo Tufaile                                     m
2           Adolfo Tufaile Júnior                              m
3           Adriana Ferreira dos Santos                        f
4           Adriana Rodrigues Tufaile                          f
5           Alessandra de Carvalho e Silva Skuja               f

(5 row(s) affected)
```

  - Exemplo 2:

```
SELECT TOP 5 PERCENT
FROM Pessoas
```

  - Resultado:

```
Numero      Nome                                               Sexo
----------- -------------------------------------------------- ----
1           Adolfo Tufaile                                     m
2           Adolfo Tufaile Júnior                              m
3           Adriana Ferreira dos Santos                        f
4           Adriana Rodrigues Tufaile                          f
5           Alessandra de Carvalho e Silva Skuja               f
6           Alexandre Rodrigues Tufaile                        m
7           Alexandrina de Souza Portugal                      f
```

```
8           Alice dos Santos Madi                          f

(8 row(s) affected)
```

- Observações:

  Ao comando SELECT com a cláusula TOP <n> [PERCENT], pode-se acrescentar a expressão [WITH TIES], desde que o comando SELECT empregue também a cláusula ORDER BY.

  Caso isto ocorra, o comando poderá retornar uma quantidade de linhas maior do que a especificada por <n>, já que também retornará as linhas posteriores à n-ézima que possuírem o mesmo valor que esta para a(s) coluna(s) mencionadas na cláusula ORDER BY.

- Seleção com Resultados Ordenados:

  - Exemplo:

```
SELECT *
FROM Pessoa
ORDER BY sexo DESC
WHERE Nome > 'E' and Nome < 'F'
```

  - Resultado:

```
Numero      Nome                                           Sexo
----------- ---------------------------------------------- ----
46          Edson Kowask Bezerra                           m
45          Edson de Almeida                               m
50          Escolástica Pires Monteiro                     f
49          Érica Rodrigues Pereira                        f
48          Érica Cristina Olher Rubi da Cunha             f
47          Eliza Balboni Skuja                            f
44          Edi Aparecida Cervan                           f

(7 row(s) affected)
```

- BETWEEN...AND...:

  Testa o conteúdo de uma coluna e extrai apenas os dados que se encontram dentro da faixa especificada.

  - Exemplo:

```
SELECT *
```

```
FROM Pessoas
WHERE Nome Between 'E' and 'F'
```

- Resultado:

```
Numero      Nome                                               Sexo
----------- -------------------------------------------------- ----
44          Edi Aparecida Cervan                               f
45          Edson de Almeida                                   m
46          Edson Kowask Bezerra                               m
47          Eliza Balboni Skuja                                f
48          Érica Cristina Olher Rubi da Cunha                 f
49          Érica Rodrigues Pereira                            f
50          Escolástica Pires Monteiro                         f

(7 row(s) affected)
```

- IN (<lista de valores>):

Opera sobre uma lista de valores e testa se o valor da coluna especificada pertence a esta lista.

- Exemplo:

```
SELECT *
FROM Pessoa
WHERE Numero IN (11,22,33,44,55,66,77)
```

- Resultado:

```
Numero      Nome                                               Sexo
----------- -------------------------------------------------- ----
11          Ana Regina Ferreira dos Santos                     f
22          Antônio Russo                                      m
33          Cybelle Monteiro de Carvalho e Silva               f
44          Edi Aparecida Cervan                               f
55          Francisco Monteiro de Carvalho e Silva             m
66          Henrique Capana Filho                              m
77          João Russo                                         m

(7 row(s) affected)
```

- LIKE:

Quando não se sabe o valo exato que está sendo procurado, usa-se o operador LIKE que reconhece expressões regulares, bem como caracteres coringas, a saber: **%** , que corresponde a uma seqüência qualquer de 0 ou mais caracteres, e **_** , que corresponde a qualquer caracter.

- Exemplo:

```
SELECT Nome
```

```
FROM Pessoa
WHERE Nome LIKE '%Russo%'
```

- Resultado:

```
Nome
-------------------------------------------------
Andressa Russo Zotto
Antônio Russo
Haílton Russo
Jefferson Russo
Kátia Russo
Sílvia Maria de Carvalho e Silva Russo

(6 row(s) affected)
```

- IS [NOT] NULL:

  Verifica se a coluna especificada possui ou não um valor nulo.

- GROUP BY:

  Agrupamento de linhas permite que as linhas da tabela de resultado sejam agrupadas, de acordo com critérios especificados pelo usuário.

  - Funções de Grupo:

    M**AX(<espressão>)**

    Calcula o maior dos valores da <expessão> para os valores das linhas selecionadas; Ignora nulos.

    MIN(<expressao>]

    Calcula o menor dos valores da <expessão> para os valores das linhas selecionadas; Ignora nulos.

    SUM ([DISTINCT] *|<expressão>)

    Calcula a soma dos valores da <expessão> para os valores das linhas selecionadas; Ignora nulos;

    AVG([DISTINCT] *|<expressão>)

    Calcula a soma dos valores da <expessão> para os valores das linhas selecionadas; Ignora nulos;

COUNT([DISTINCT] *|<expressão>)

Conta o número de vezes que a expressão retorna qualquer coisa diferente de nulo.

STDDEV ([DISTINCT] *|<expressão>)

Calcula o desvio padrão dos valores da <expessão> para os valores das linhas selecionadas; Ignora nulos.

VARIANCE ([DISTINCT] *|<expressão>)

Calcula a variância dos valores da <expessão> para os valores das linhas selecionadas; Ignora nulos.

- Exemplo:

```
SELECT Pai, COUNT(*) AS QtosFilhos
FROM Nascimentos
GROUP BY Pai
HAVING COUNT(*)>5
ORDER BY QtosFilhos
```

- Resultado:

```
Pai         QtosFilhos
----------- -----------
62          8
28          10

(2 row(s) affected)
```

- Observações:

A diferença entre as cláusulas `HAVING` e `WHERE` é que, a cláusula `WHERE` define as linhas de resultado como um todo, já a cláusula `HAVING` opera sobre um grupo de linhas.

- Sub Queries:

Uma subconsulta é um comando SELECT que está aninhado dentro de um comando SELECT, INSERT, UPDATE, ou DELETE.

Subqueries constituem operandos dos operandos de comparação.

Como um comando SELECT pode retornar uma lista de valores, é necessário que o operador seja adequado ao resultado produzido pelo comando. A alternativa é que, no caso de listas, o outro operando também seja uma lista.

- Exemplo:

```
SELECT Nome
FROM Pessoas
WHERE Numero IN
   (SELECT Pai FROM Nascimentos
    WHERE Data >= '01/01/2000')
```

- Resultado:

```
Nome
--------------------------------------------------
Alexandre Rodrigues Tufaile
Jefferson Russo

(2 row(s) affected)
```

- Observações:

  A cláusulas `DISTINCT, ORDER BY e UNION` só podem ser utilizadas uma única vez na instrução `SELECT` inteira.

  Subqueries não podem ter as cláusulas `ORDER BY`, `GROUP BY` e `HAVING`.

- `ALL|DISTINCT`:

  Como default de `SELECT` é `ALL`, sempre que a instrução `SELECT` é executada ela selecionará todas as linhas que atenderem a condição especificada, mesmo que haja duplicidade na tabela de resultado.

  Com a cláusula `DISTINCT` as linhas repetidas desaparecem da tabela de resultados (embora continuem existindo dentro da tabela de dados).

- Os Predicados ANY e ALL:

  Os predicados ANY e ALL estão sempre associados a uma subconsulta.

  ANY determina se a condição estabelecida pela cláusula WHERE é verdadeira para pelo menos uma das linhas da tabela de resultado da subconsulta.

- Exemplo 1:

```
SELECT Nome
FROM Pessoas
WHERE Numero = ANY
   (SELECT Pai FROM Nascimentos
    WHERE Data >= '01/01/2001')
```

- Resultado:

```
Nome
--------------------------------------------------
Alexandre Rodrigues Tufaile
Jefferson Russo

(2 row(s) affected)
```

ALL determina se a condição estabelecida pela cláusula WHERE é verdadeira para todas as linhas da tabela resultado da subconsulta.

- Exemplo:

```
Select Nome
From Pessoas
Where Numero != All (select Pai From Nascimentos) and
      Numero != All (select Mae From Nascimentos)
```

- Resultado:

```
Nome
--------------------------------------------------
Ana Regina Ferreira dos Santos
Cláudio Aparecido Ribeiro
Dirceu Roque da Luz
José Pereira dos Santos
Luiz Pereira
Maria Castro da Luz
Tânia Maria Monteiro Ribeiro

(7 row(s) affected)
```

- O Predicado EXISTS:

O predicado EXISTS está sempre associado a uma subconsulta e determina se a subconsulta seleciona ou não alguma linha.

Em outras palavras, o predicado EXISTS permite que um SELECT selecione as linhas que fazem com que a subconsulta selecione ou não alguma linha.

O predicado EXISTS é o único que permite o uso do asterisco (*) na subconsulta. Outra observação é que a subconsulta envolver informações da query principal.

- Exemplo:

```
Select Nome
From Pessoas as P
Where Nome Like 'A%' and Exists
     (Select 1 From Nascimentos as N
      Where P.Numero = N.Mae and Not Exists
           (Select 1 From Casamentos as C
            Where N.Mae = C.Esposa))
```

- Resultado:

```
Nome
--------------------------------------------------
Alexandrina de Souza Portugal
Alice dos Santos Madi
Ana Paiva Pereira
Anna Francisca de Oliveira Salgado
Antônia Pereira de Souza Cruz

(5 row(s) affected)
```

- Consultas Correlacionadas:

A subconsulta é executada para cada linha selecionada no SELECT principal, diferente dos subqueries simples que são executados somente uma vez. Para isso ocorrer, o subconsulta deve fazer referência a uma coluna referente a uma tabela selecionada do query principal.

- Exemplo:

```
Select Nome
From Pessoas As P
Where Numero in
    (Select Pai
     From Nascimentos As N
     Where P.Numero = N.Pai and N.Filho =
         (Select Numero
          From Pessoas As F
          Where Nome = 'Victor Hugo Rodrigues Gomes de Carvalho'));
```

- Resultado:

```
Nome
--------------------------------------------------
André Luís dos Reis Gomes de Carvalho

(1 row(s) affected)
```

- Junção de Tabelas:

  As junções de tabelas possibilitam a criação de tabelas de resultado contendo informações extraídas de duas tabelas de dados. Os atributos da tabela resultado de uma junção de duas tabelas serão os atributos da primeira tabela, seguidos pelos atributos da segunda tabela. As linhas da tabela resultado de uma junção serão obtidas pela concatenação de linhas da primeira tabela com linhas da segunda tabela.

  Existem vários tipos de junção, a saber: produto cartesiano, junção interna, junção externa esquerda, junção externa direita e junção externa completa.

  O PRODUTO CARTESIANO consiste em combinar CADA UMA das linhas da primeira tabela com TODAS as linhas da segunda tabela. Este tipo de junção não exige a existência de um ou mais atributos comuns às duas tabelas.

  A JUNÇÃO INTERNA pressupõe a existência de um ou mais atributos comuns às duas tabelas. SOMENTE as linhas cujos valores desses atributos comuns forem coincidentes serão combinadas na junção interna; as linhas da primeira tabela cujos valores desses atributos comuns não coincidirem com os valores destes mesmos atributos de nenhuma linha da segunda tabela NÃO serão combinadas na junção interna; as linhas da segunda tabela cujos valores desses atributos comuns não coincidirem com os valores destes mesmos atributos de nenhuma linha da primeira tabela NÃO serão combinadas na junção interna.

  A JUNÇÃO EXTERNA ESQUERDA também pressupõe a existência de um ou mais atributos comuns às duas tabelas. As linhas cujos valores desses atributos comuns forem coincidentes SERÃO combinadas neste tipo de junção. As linhas da primeira tabela cujos valores desses atributos comuns não coincidirem com os valores destes mesmos atributos de nenhuma linha da segunda tabela TAMBÉM serão combinadas neste tipo de junção (os valores dos atributos provindos da segunda tabela ficarão nulos). As linhas da segunda tabela cujos valores desses atributos comuns não

coincidirem com os valores destes mesmos atributos de nenhuma linha da primeira tabela NÃO serão combinadas neste tipo de junção.

A JUNÇÃO EXTERNA DIREITA também pressupõe a existência de um ou mais atributos comuns às duas tabelas. As linhas cujos valores desses atributos comuns forem coincidentes SERÃO combinadas neste tipo de junção. As linhas da primeira tabela cujos valores desses atributos comuns não coincidirem com os valores destes mesmos atributos de nenhuma linha da segunda tabela NÃO serão combinadas neste tipo de junção. As linhas da segunda tabela cujos valores desses atributos comuns não coincidirem com os valores destes mesmos atributos de nenhuma linha da primeira tabela TAMBÉM serão combinadas neste tipo de junção (os valores dos atributos provindos da primeira tabela ficarão nulos).

A JUNÇÃO EXTERNA COMPLETA também pressupõe a existência de um ou mais atributos comuns às duas tabelas. As linhas cujos valores desses atributos comuns forem coincidentes SERÃO combinadas neste tipo de junção. As linhas da primeira tabela cujos valores desses atributos comuns não coincidirem com os valores destes mesmos atributos de nenhuma linha da segunda tabela TAMBÉM serão combinadas neste tipo de junção (os valores dos atributos provindos da segunda tabela ficarão nulos). As linhas da segunda tabela cujos valores desses atributos comuns não coincidirem com os valores destes mesmos atributos de nenhuma linha da primeira tabela TAMBÉM serão combinadas neste tipo de junção (os valores dos atributos provindos da primeira tabela ficarão nulos).

Os atributos comuns que serão levados em conta para combinar linhas da primeira tabela com linhas da segunda tabela podem ser indicados explicitamente ou não.

No caso de uma junção NATURAL eles não são indicados explicitamente; TODOS os atributos comuns serão levados em conta e eles aparecerão UMA ÚNICA VEZ na tabela resultante da junção.

No caso de junções não naturais, o critério de junção precisará ser explicitado com ON ou com USING. No primeiro caso (ON), os atributos comuns às duas tabelas aparecerão DUPLICADOS na tabela resultante da junção. No segundo caso (USING),

os atributos comuns às duas tabelas aparecerão UMA ÚNICA VEZ na tabela resultante da junção.

Junções são sempre usadas na cláusula FROM de um comando SELECT.

- Sintaxe:

```
<nome da tabela>
[NATURAL]
CROSS | INNER | LEFT OUTER | RIGHT OUTER | FULL OUTER
JOIN
<nome da tabela>

[ON <condição>]   |
[USING (<nome da coluna>[,...,, <nome da coluna>]]
```

- Exemplo:

```
Select F.Nome, N.Local From
    Pessoas as F
        inner join
    Nascimentos as N
        On (F.Numero = N.Filho
            And
            N.Data >= '01/01/2000')
```

- Resultato:

```
Nome                                               Local
-------------------------------------------------- --------------------
Giovana Russo                                      Cutitiba
Rafael Ferreira Tufaile                            Guarulhos

(2 row(s) affected)
```

- Observações:

  AUTO JUNÇÕES (Self Joins) são o tipo de junção em que uma tabela é juntada consigo própria.

- União de Tabelas:

  A união de tabelas permite agrupar, em uma única tabela de resultado, as linhas de duas outras tabelas. É imprescindível que as tabelas envolvidas tenham a mesma estrutura, i.e., a mesma quantidade de colunas, com os mesmos nomes e tipos. Este

tipo de operação só poderá ser realizada se as tabelas envolvidas residirem no mesmo BD.

- Sintaxe:

```
<comando select>
UNION
<comando select>
```

- Seleção com Resultado em uma Tabela:
  - Sintaxe:

```
<comando select>
INTO <nome da tabela>
```

  - Exemplo:

```
SELECT * FROM Pessoas INTO Todos
WHERE Sexo = 'M'
```

- Inclusão de Múltiplas Linhas:
  - Sintaxe:

```
INSERT INTO <nome da tabela>
[(<nome da coluna>[,...,<nome da coluna>]]
<comando select>
```

  - Exemplo:

```
INSERT INTO Todos
SELECT * FROM Pessoas
WHERE Sexo = 'F'
```

e) Processamento de Transações

Transação é uma série de operações consideradas em conjunto. Assim, um transação só será concluída se todas as operações que a compõem forem completadas com sucesso.

Com o uso de transações, torna-se possível garantir a integridade das informações do Banco de Dados, uma vez que nenhum processo será executado parcialmente.

Caso algum problema ocasione a interrupção da transação, todas as tabelas serão revertidas à situação que se encontravam no início da transação.

- Sintaxe:

```
BEGIN TRANSACTION
    <comando SQL>
    ...
    <comando SQL>
[COMMIT TRANSACTION|ROLLBACK TRANSACTION]
```

As transações podem ser usadas tanto interativamente quanto dentro de programas. Como é fácil perceber, as transações retardam a execução dos programas e por isso devem ser usadas com critério.

As inclusões, alterações e exclusões só podem ser revertidas dentro de transações. Fora de transações elas são registradas fisicamente. As instruções de criação e eliminação de elementos do Banco de Dados não podem ser utilizadas dentro de transações.

Apêndice A:
Palavras Reservadas de SQL

CHAR	EXEC	MODULE	SELECT
CHARACTER	EXISTS	NOT	SET
CHECK	FETCH	NULL	SMALLINT
CLOSE	FLOAT	NUMERIC	SOME
COBOL	FOR	OF	SQL
COMMIT	FORTRAN	ON	SQLCODE
CONTINUE	GRANT	OPEN	SQLERROR
COUNT	GROUP	OPTION	SUM
CRATE	HAVING	OR	TABLE
CURRENT	IN	ORDER	TO
CURSOR	INDICATOR	PASCAL	UNION
DEC	INSERT	PLI	UNIQUE
DECIMALDECL	INT	PRECISION	UPDATE
ARE	INTEGER	PRIVILEGES	USER
DELETE	INTO	PROCEDURE	VALUES
DESC	IS	PUBLIC	VIEW
DISTINCT	LANGUAGE	REAL	WHENEVER
DOUBLE	LIKE	ROLLBACK	WHERE
END	MAX	SCHEMA	WITH
ESCAPE	MIN	SECTION	WORK

Apêndice B:
BD

PESSOAS

Numero	Nome	Sexo
1	Adolfo Tufaile	m
2	Adolfo Tufaile Júnior	m
3	Adriana Ferreira dos Santos	f
4	Adriana Rodrigues Tufaile	f
5	Alessandra de Carvalho e Silva Skuja	f
6	Alexandre Rodrigues Tufaile	m
7	Alexandrina de Souza Portugal	f
8	Alice dos Santos Madi	f
9	Álisson de Carvalho e Silva Skuja	m
10	Ana Paiva Pereira	f
11	Ana Regina Ferreira dos Santos	f
12	André Hairan de Carvalho e Silva Almeida	m
13	André Luís dos Reis Gomes de Carvalho	m
14	Andressa Russo Zotto	f
15	Anna Francisca de Oliveira Salgado	f
16	Anna Maria Monteiro de Carvalho e Silva	f
17	Antônia Pereira de Souza Cruz	f
18	Antônio Carlos Zotto	m
19	Antônio de Abreu Sampaio	m
20	Antônio Monteiro de Carvalho e Silva	m
21	Antônio Oscar da Silva	m
22	Antônio Russo	m
23	Ariane Tufaile Paiva	f
24	Arlindo Demétrio Madi	m
25	Armando de Souza Portugal	m
26	Bruna Letícia Olher Maluf	f
27	Célia Monteiro de Carvalho e Silva	f
28	Celso Monteiro de Carvalho e Silva	m
29	Cláudio Aparecido Ribeiro	m
30	Cleso Monteiro de Carvalho e Silva	m
31	Cleusa Monteiro de Carvalho e Silva	f
32	Cleusa de Carvalho e Silva Rodrigues F	f
33	Cybelle Monteiro de Carvalho e Silva	f
34	Daniel Skuja	m
35	Daniela de Carvalho e Silva Skuja	f
36	David Celso de Souza Cruz Rodrigues	m
37	David Venâncio de Souza Cruz	m
38	Deodato Eleutério Rodrigues	m
39	Deodato Eleutério Rodrigues Neto	m
40	Diego Iniesta	m

41	Dirce Monteiro de Carvalho e Silva	f
42	Dirce de Carvalho e Silva Rodrigues F	f
43	Dirceu Roque da Luz	m
44	Edi Aparecida Cervan	f
45	Edson de Almeida	m
46	Edson Kowask Bezerra	m
47	Eliza Balboni Skuja	f
48	Érica Cristina Olher Rubi da Cunha	f
49	Érica Rodrigues Pereira	f
50	Escolástica Pires Monteiro	f
51	Fabiano Ricardo Rodrigues Maluf	m
52	Felippe Abrahão Maluf	m
53	Fernanda Rodrigues Madi	f
54	Flora Lisboa Rodrigues	f
55	Francisco Monteiro de Carvalho e Silva	m
56	Gabriela Rodrigues da Silva	f
57	Genoveva Maluf	f
58	Giovana Russo	f
59	Giulliano Humberto Capana	m
60	Haílton Russo	m
61	Haroldo Cesar de Souza Cruz Rodrigues	m
62	Haroldo Lisboa Rodrigues	m
63	Haroldo Lisboa Rodrigues Júnior	m
64	Heloisa Monteiro de Carvalho e Silva	f
65	Heloisa de Carvalho e Silva Rodrigues	f
66	Henrique Capana Filho	m
67	Henrique Rodrigues da Silva	m
68	Iara Maria Pereira	f
69	Ingrid Tufaile Kowask Bezerra	f
70	Izabel Gimenez	f
71	Jair da Costa Pereira	m
72	Jefferson Russo	m
73	Jéssica Cristina Cervan Rodrigues	f
74	João Antônio Salgado Júnior	m
75	João Gabriel Cervan Rodrigues	m
76	João Moreira da Cunha	m
77	João Russo	m
78	José Antônio Iniesta	m
79	José Antônio Iniesta Júnior	m
80	José Gomes de Carvalho	m
81	José Luiz Santana	m
82	José Monteiro de Carvalho e Silva	m
83	José Pereira da Silveira	m
84	José Pereira dos Santos	m
85	José Tufaile Huaixan	m
86	Júlia Rodrigues Madi	f
87	Juliana Rodrigues Tufaile	f
88	Juventina Monteiro de Carvalho e Silva	f
89	Kátia Russo	f
90	Leonina Alboredo Russo	f
91	Lília de Carvalho e Silva Rodrigues	f
92	Lilian Iniesta	f
93	Luciana Rodrigues Tufaile	f
94	Luciano Zotto	m
95	Luiz Celso Lisboa Rodrigues	m
96	Luiz Henrique de Souza Cruz Rodrigues	m

97	Luiz Paiva Pereira	m
98	Luiz Pereira	m
99	Luiz Ricardo Rodrigues Maluf	m
100	Luzia Maria da Costa Iniesta	f
101	Magda da Luz	f
102	Manoel Cervan	m
103	Márcia Regina Santana	f
104	Maria Castro da Luz	f
105	Maria das Dores de Carvalho Sampaio	f
106	Maria de Lourdes Oliveira	f
107	Maria Fernanda de Carvalho e Silva	f
108	Maria Helena Rubi da Cunha	f
109	Maria Izabel Tufaile	f
110	Maria Lúcia Zotto	f
111	Maria Luiza Pereira Rodrigues	f
112	Maria de Oliveira Salgado	f
113	Maria Terezinha Kowask Bezerra	f
114	Michela de Souza Cruz Rodrigues	f
115	Neusa Maria Palota Capana	f
116	Neusa Pereira dos Reis Carvalho	f
117	Olga Canezin	f
118	Ozélia Mota Santana	f
119	Páscoa Cervan	f
120	Paulo Cesar Madi	m
121	Paulo Madi Filho	m
122	Petras Skuja	m
123	Rafael Ferreira Tufaile	m
124	Regina Célia de Carvalho e Silva Rodrigues	f
125	Renan Tufaile Paiva	m
126	Renata Rodrigues Madi	f
127	Renata Santana Rodrigues	f
128	Ricardo Felippe Maluf	m
129	Ricardo Felippe Maluf Filho	m
130	Rogério Tufaile Kowask Bezerra	m
131	Sandra Regina de Carvalho e Silva	f
132	Sílvia Maria de Carvalho e Silva	f
133	Tânia Maria Monteiro Ribeiro	f
134	Teíla de Souza Cruz	f
135	Theodoro Silva Bezerra	m
136	Theolinda Monteiro de Carvalho e Silva	f
137	Theolinda de Abreu Sampaio	f
138	Thereza Souza Portugal	f
139	Vanessa de Carvalho e Silva Skuja	f
140	Vanessa Santana Rodrigues	f
141	Vera Regina Tufaile	f
142	Victor Hugo Rodrigues Gomes de Carvalho	m
143	Viviane Santana Rodrigues	f
144	Zique Tufaile	m

NASCIMENTOS

Numero	Data	Local	Filho	Mae	Pai
1	1919-06-22 00:00:00.000	Paraguai, Rosário	1	70	85
2	1960-11-28 00:00:00.000	São José do Rio Preto	2	16	1
3	1970-02-25 00:00:00.000	São Paulo	4	65	144
4	1982-03-05 00:00:00.000	Sumaré	5	131	34
5	1973-08-19 00:00:00.000	São Paulo	6	65	144
6	1988-05-04 00:00:00.000	Campinas	9	131	34
7	1996-05-15 00:00:00.000	Campinas	12	107	45
8	1966-01-19 00:00:00.000	Araçatuba	13	116	80
9	1993-12-15 00:00:00.000	Curitiba	14	89	94
10	1932-02-15 00:00:00.000	Campinas	16	112	28
11	1865-05-26 00:00:00.000	Campinas	20	50	55
12	1947-05-07 00:00:00.000	Brotas	22	90	77
13	1998-04-16 00:00:00.000	Cafelândia	26	48	51
14	1922-04-14 00:00:00.000	Campinas	27	112	28
15	1897-07-08 00:00:00.000	Campinas	28	137	20
16	1927-06-19 00:00:00.000	Campinas	30	112	28
17	1930-01-24 00:00:00.000	Campinas	31	112	28
18	1965-01-20 00:00:00.000	São José do Rio Preto	32	31	62
19	1928-12-29 00:00:00.000	Campinas	33	112	28
20	1952-06-10 00:00:00.000	Carapicuiba	34	47	122
21	1978-10-24 00:00:00.000	Piracicaba	35	131	34
22	1973-01-18 00:00:00.000	Brasília	36	134	95
23	1963-10-25 00:00:00.000	São José do Rio Preto	39	31	62
24	1924-10-10 00:00:00.000	Campinas	41	112	28
25	1924-10-10 00:00:00.000	Campinas	42	41	62
26	NULL		44	119	102
27	1958-03-02 00:00:00.000	Paranaguá	46	113	135
28	1976-02-29 00:00:00.000		48	108	76
29	1977-08-22 00:00:00.000	Brasília	49	68	71
30	1977-02-10 00:00:00.000	Santos	51	124	128
31	1986-09-18 00:00:00.000	São José do Rio Preto	53	42	120
32	1925-04-17 00:00:00.000	São José do Rio Preto	56	32	21
33	2001-01-24 00:00:00.000	Cutitiba	58	101	72
34	1979-08-03 00:00:00.000	São Paulo	59	115	66
35	1978-02-25 00:00:00.000	Campinas	60	132	22
36	1979-11-11 00:00:00.000	Brasília	61	134	95
37	1923-04-22 00:00:00.000	São José do Rio Preto	62	54	38
38	1957-08-12 00:00:00.000	São josé de Rio Preto	63	41	62
39	1937-04-20 00:00:00.000	Campinas	64	112	28
40	1946-07-16 00:00:00.000	São José de Rio Preto	65	41	62
41	1990-06-06 00:00:00.000	São José do Rio Preto	67	32	21
42	1983-06-01 00:00:00.000	Campinas	69	141	46
43	1974-04-26 00:00:00.000	Campinas	72	132	22
44	1993-07-28 00:00:00.000	Araraquara	73	44	63
45	1995-09-05 00:00:00.000	Araraquara	75	44	63
46	1953-06-22 00:00:00.000	São Caetano	78	100	40
47	1953-06-22 00:00:00.000	São Caetano do Sul	79	109	78
48	1926-04-23 00:00:00.000	Campinas	82	112	28
49	1993-05-14 00:00:00.000	São José de Rio Preto	86	42	120
50	1978-07-27 00:00:00.000	São Paulo	87	65	144
51	1934-01-10 00:00:00.000	Campinas	88	112	28
52	1973-04-15 00:00:00.000	Campinas	89	132	22
53	1962-01-22 00:00:00.000	São José de Rio Preto	91	41	62
54	1978-10-08 00:00:00.000	São Caetano do Sul	92	109	78
55	1977-11-08 00:00:00.000	São Paulo	93	65	144
56	NULL		94	110	18
57	1947-12-28 00:00:00.000	São José de Rio |Preto	95	41	62
58	1975-12-21 00:00:00.000	Campinas	96	134	95
59	1966-07-31 00:00:00.000	Tarumirim	97	10	83
60	1979-04-03 00:00:00.000	Cafelândia	99	124	128
61	NULL	São José do Rio Preto	103	118	81
62	1971-07-09 00:00:00.000	Campinas	107	106	30
63	1957-01-11 00:00:00.000	Paulo de Faria	109	16	1
64	1997-11-02 00:00:00.000	Brasília	111	49	36
65	1897-04-11 00:00:00.000	Campinas	112	15	74
66	1982-04-01 00:00:00.000	Brasília	114	134	95
67	1953-01-05 00:00:00.000	São José do Rio Preto	120	8	24
68	1976-08-01 00:00:00.000	São José de Rio Preto	121	42	120
69	2001-03-02 00:00:00.000	Guarulhos	123	3	6
70	1950-08-23 00:00:00.000	Campinas	124	41	62
71	1994-04-11 00:00:00.000	Guarulhos	125	4	97
72	1976-10-21 00:00:00.000	São José de Rio Preto	126	42	120
73	1948-03-03 00:00:00.000	São José do Rio Preto	128	57	52
74	1973-03-07 00:00:00.000	São José de Rio Preto	129	124	128
75	1986-11-03 00:00:00.000	Campinas	130	141	46
76	1956-12-24 00:00:00.000	Campinas	131	138	82
77	1954-10-24 00:00:00.000	Campinas	132	138	82
78	1951-07-15 00:00:00.000	Roraima	134	17	37
79	1923-08-18 00:00:00.000	Campinas	136	112	28
80	1875-12-15 00:00:00.000	Campinas	137	105	19
81	1927-01-31 00:00:00.000	Sertãozinho	138	7	25
82	1982-05-03 00:00:00.000	Sumaré	139	131	34
83	1991-04-22 00:00:00.000	São José do Rio Preto	140	103	39
84	1959-09-08 00:00:00.000	Paulo de Faria	141	16	1
85	1926-07-29 00:00:00.000	Campinas	142	91	13
86	1945-10-03 00:00:00.000	Paulo de Faria	144	117	1

87	1999-06-10 00:00:00.000	Guarulhos	23	4	97
88	1994-09-18 00:00:00.000	São José do Rio Preto	143	103	39
89	1999-09-04 00:00:00.000	São José do Rio Preto	127	103	39

MORTES

Numero	Data	Local	Morto
1	1981-11-06 00:00:00.000	Taquaritinga	1
2	1988-05-20 00:00:00.000	Campinas	16
3	1921-02-14 00:00:00.000	Campinas	20
4	1964-06-06 00:00:00.000	Campinas	28
5	NULL	Campinas	30
6	1962-09-20 00:00:00.000	São José do Rio Preto	41
7	1990-09-26 00:00:00.000	São José do Rio Preto	54
8	1996-12-22 00:00:00.000	Santos	63
9	1942-03-18 00:00:00.000	Campinas	64
10	1978-01-07 00:00:00.000	Guarulhos	93
11	1998-01-04 00:00:00.000	Brasília	96
12	1991-01-02 00:00:00.000	São José do Rio Preto	98
13	1988-06-09 00:00:00.000	Campinas	112
14	1945-01-17 00:00:00.000	Campinas	137
15	1994-09-18 00:00:00.000	São José do Rio Preto	143

CASAMENTOS

Numero	Data	Local	Esposo	Esposa	NNEsposa
1	1956-10-06 00:00:00.000	São José do Rio Preto	1	16	Anna Maria de Carvalho e Silva Tufaile
2	1970-04-26 00:00:00.000	São Paulo	6	3	Adriana Ferreira Tufaile
3	1989-03-18 00:00:00.000	Campinas	13	91	Lília Rodrigues Gomes de Carvalho
4	1865-05-26 00:00:00.000	Campinas	20	137	Theolinda Sampaio de Carvalho e Silva
5	1989-11-04 00:00:00.000	São José do Rio Preto	21	32	NULL
6	1972-07-01 00:00:00.000	Campinas	22	132	Silvia Maria de Carvalho e Silva Russo
7	1921-06-23 00:00:00.000	Jaguary	28	112	Maria Salgado de Carvalho e Silva
8	NULL	Campinas	30	106	Maria de Lourdes de Carvalho e Silva
9	1977-12-16 00:00:00.000	Campinas	34	131	Sandra Regina de Carvalho e Silva Skuja
10	1977-08-22 00:00:00.000	Brasilia	36	49	NULL
11	NULL	NULL	38	54	NULL
12	1989-12-23 00:00:00.000	São José do Rio Preto	39	103	Márcia Regina Santana Rodrigues
13	1983-09-24 00:00:00.000	Campinas	46	141	Vera Regina Tufaile Kowask Bezerra
14	NULL	NULL	51	48	Érica Cristina Olher da Cunha Maluf
15	1997-11-22 00:00:00.000	São Paulo	59	87	NULL
16	1963-01-26 00:00:00.000	São José do Rio Preto	62	31	Cleusa de Carvalho e Silva Rodrigues
17	1945-10-14 00:00:00.000	Campinas	62	41	Dirce de Carvalho e Silva Rodrigues
18	1991-12-20 00:00:00.000	Araraquara	63	44	Edi Aparecida Cervan Rodrigues
19	1977-01-05 00:00:00.000	Sao Caetano do Sul	78	109	Maria Izabel Tufaile Iniesta
20	1953-12-24 00:00:00.000	Campinas	82	138	Thereza Portugal de Carvalho e Silva
21	1969-12-15 00:00:00.000	Roraima	95	134	Teila de Souza Cruz Rodrigues
22	1992-03-21 00:00:00.000	São Paulo	97	4	Adriana Tufaile Paiva
23	NULL	NULL	98	54	NULL
24	1974-09-27 00:00:00.000	São José do Rio Preto	120	42	Dirce de Carvalho e Silva Rodrigues Madi
25	1972-07-15 00:00:00.000	São José do Rio Preto	128	124	Regina Célia Rodrigues Maluf
26	1969-01-18 00:00:00.000	São José do Rio Preto	144	65	Heloisa Rodrigues Tufaile

SEPARACOES

Numero	Data	Local	Casamento

Apêndice C:
Exercícios de Linguagens de SQL

1. Considere o seguinte BD:

   Motoristas {Cart, DtVencCart, Nom, End, Cid,Est}

   Veículos {Plac, Propr, EndProp, CidProp, EstPropVei, Comb, Chas, Ano}

   IPVA {PlacVei, Ano, Qta, ValQta, DtVencQta, DtPagQta}

   Infrações {Nro, Descr}

   Multas {PlacVei, Dt, Hor, NroInfr, Local, Cid, Est}

   MultasPagas {PlacVei, Dt, Hor, NroInfr, Local, Cid, Est}

   Responda usando SQL às seguintes consultas:

   - Quais os proprietários multados por excesso de velocidade?
   - Quais os proprietários que nunca foram multados?
   - Quais os proprietários que foram multados pelo menos uma vez?

2. Considere o seguinte BD:

   Médicos {CRM, Nom}

   Cirurgiões {CRM, NroBIP}

   Clínicos {CRM, Tel}

   Especialidades {Cod, Nom}

   EspecDosMed {CRM, Cod}

   Enfermarias {Cod, Nom}

   Trabalha {CRM, Cod}

   Coordena {CRM, Cod}

   Responda usando SQL às seguintes consultas:

- Quais as especialidades do coordenador da enfermaria deemergência?
- Qual o nome de todos os cirurgiões da enfermaria de pediatriaespecialistas em correção?

3. Considere o seguinte BD:

   HospitaisGeriátricos {Cod, Nom, NroLeit}
   HospitaisPsiquiátricos {Cod, Nom, NroLeit}
   Laboratórios {Nro, Nom, Tel}
   HospLab {Cod, Nro}

   Responda usando SQL às seguintes consultas:

   - Qual o nome dos laboratórios que não trabalham com nenhum hospital?
   - Qual o nome e o telefone dos laboratórios que trabalham apenas com hospitais que são só geriátricos?
   - Qual o número dos laboratórios que trabalham apenas com hospitais que são geriátricos e psiquiátricos?