Instituto de Ciências Matemáticas de São Carlos

Instituto de Ciências Matemáticas
Biblioteca
São Carlos

ISSN - 0103-2569

DEDALUS - Acervo - ICMC
30300039618

SQL PARA MICROSOFT ACCESS 97

ENZO SERAPHIM
DINAMÉRICO ALONSO GASPAR

Nº 79

RELATÓRIOS TÉCNICOS DO ICMSC

São Carlos
Ago./1998

SQL para Microsoft Access 97

Autor: Enzo Seraphim (seraphim@icmc.sc.usp.br)
Dinamérico Alonso Gaspar (gaspar@zaz.com.br)

Índice

1) DDL – COMANDOS PARA A DEFINIÇÃO DE RELAÇÕES **2**
1.1) Instrução CREATE TABLE 2
1.1.1) Cláusula CONSTRAIN 2
1.2) Instrução CREATE INDEX 4
1.3) Instrução ALTER TABLE 5
1.4) Instrução DROP 6

2) DML – COMANDOS PARA A MANIPULAÇÃO DE RELAÇÕES **8**
2.1) Instrução SELECT 8
2.1.1) Cláusula FROM 9
2.1.2) Cláusula WHERE 9
2.1.3) Cláusula GROUP BY 10
2.1.4) Cláusula HAVING 11
2.1.5) Cláusula ORDER BY 12
2.1.6) Atributos ALL, DISTINCT, DISTINCTROW e TOP 13
2.1.7) Operação INNER JOIN 14
2.1.8) Operações LEFT JOIN, RIGHT JOIN 15
2.1.9) Operação UNION 16
2.1.10) Operador In 17
2.1.11) Operador Between...And 17
2.1.12) Operador Like 18
2.1.13) Subconsultas SQL 19
2.2) Instrução SELECT...INTO 20
2.3) Instrução INSERT INTO 21
2.4) Instrução UPDATE 23
2.5) Instrução DELETE 24

3) FUNÇÕES AGREGADAS SQL **25**
3.1) Função Avg 25
3.2) Função Count 25
3.3) Funções Min, Max 26
3.4) Função Sum 26

4) COMPARAÇÃO ENTRE O SQL DO MECANISMO DE BANCO DE DADOS MICROSOFT ACCESS E O SQL ANSI **27**
4.1) Recursos avançados do SQL Access não suportados pelo SQL ANSY 27
4.1.1) Instrução TRANSFORM 27
4.1.2) Funções StDev e StDevP 28
4.1.3) Funções Var, VarP 28
4.2) Recursos do SQL ANSI não suportados no SQL do Microsoft Access 29

5) TIPOS DE DADOS SQL **30**
5.1) Tipos de dados SQL ANSI equivalentes 31

1) DDL – Comandos para a definição de Relações

1.1) Instrução CREATE TABLE

Cria uma nova tabela

Sintaxe:
CREATE TABLE *tabela* (*campo1 tipo* [(*tamanho*)] [NOT NULL] [*índice1*]
[, *campo2 tipo* [(*tamanho*)] [NOT NULL] [*índice2*] [, ...]]
[, CONSTRAINT *índicedemulticampos* [, ...]])

A instrução CREATE TABLE tem estas partes:

Parte	Descrição
tabela	O nome da tabela a ser criada.
campo1, campo2	O nome do campo ou campos a serem criados na nova tabela. Você deve criar pelo menos um campo.
tipo	O tipo de dados de campo na nova tabela.
tamanho	O tamanho do campo em caracteres (somente os campos Texto e Binário).
índice1, índice2	Uma cláusula CONSTRAINT que define um índice de campo único. Consulte o tópico da cláusula CONSTRAINT para maiores informações sobre como criar este índice.
Índicedemulticampos	Uma cláusula CONSTRAINT que define um índice de campos múltiplos. Consulte o tópico da cláusula CONSTRAINT para maiores informações sobre como criar este índice.

Comentários:
Use a instrução CREATE TABLE para definir uma nova tabela e seus campos e restrições de campo. Se NOT NULL for especificado para um campo, então é necessário que os novos registros tenham dados válidos naquele campo.
Uma cláusula CONSTRAINT estabelece várias restrições em um campo e pode ser utilizada para estabelecer a chave primária.

1.1.1) Cláusula CONSTRAIN

Uma restrição é semelhante a um índice, embora também possa ser utilizada para estabelecer uma relação com uma outra tabela.
Você utiliza a cláusula CONSTRAINT nas instruções ALTER TABLE e CREATE TABLE para criar ou excluir restrições. Há dois tipos de cláusulas CONSTRAINT: um para criar uma restrição em um campo único e outro para criar uma restrição em mais de um campo.

Sintaxe:

Restrição de campo único:
CONSTRAINT *nome* {PRIMARY KEY | UNIQUE | NOT NULL |
REFERENCES *tabelaexterna* [(*campoexterno1*, *campoexterno2*)]}

Restrição de campos múltiplos:
CONSTRAINT *nome*
{PRIMARY KEY (*primária1*[, *primária2* [, ...]]) |
UNIQUE (*exclusiva1*[, *exclusiva2* [, ...]]) |
NOT NULL (*nãonulo1*[, *nãonulo2* [, ...]]) |
FOREIGN KEY (*ref1*[, *ref2* [, ...]]) REFERENCES *tabelaexterna* [(*campoexterno1* [, *campoexterno2* [, ...]])]}

A cláusula CONSTRAINT tem estas partes:

Parte	Descrição
nome	O nome da restrição a ser criada.
primária1, primária2	O nome do campo ou campos a ser designado(s) à chave primária.
exclusiva1, exclusiva2	O nome do campo ou campos a ser designado(s) como uma chave exclusiva.
nãonulo1, nãonulo2	O nome do campo ou campos que estão restritos a valores não-Null
ref1, ref2	O nome do campo ou campos de uma chave externa que fazem referência a campos em uma outra tabela.
tabelaexterna	O nome da tabela externa contendo o campo ou campos especificados por campoexterno.
campoexterno1, campoexterno2	O nome do campo ou campos na tabelaexterna especificados por ref1, ref2. Você pode omitir esta cláusula se o campo referenciado for a chave primária de tabelaexterna.

Comentários:
É necessário informar o nome da CONSTRAINT.
Você utiliza a sintaxe para uma restrição de campo único na cláusula de definição de campo de uma instrução ALTER TABLE ou CREATE TABLE que se segue imediatamente à especificação do tipo de dados do campo.
Você utiliza a sintaxe para uma restrição de campos múltiplos sempre que utilizar a palavra reservada CONSTRAINT fora de uma cláusula de definição de campo em uma instrução ALTER TABLE ou CREATE TABLE.
Utilizando CONSTRAINT, você pode designar um campo como um dos seguintes tipos de restrições:

- Você pode utilizar a palavra reservada UNIQUE para designar um campo como chave exclusiva. Isto significa que não pode haver dois registros em uma tabela que tenham o mesmo valor neste campo. Você pode restringir qualquer campo ou lista de campos como exclusivo. Se uma restrição de campos múltiplos for designada como uma chave exclusiva, os valores combinados de todos os campos no índice devem ser exclusivos, mesmo que dois ou mais registros tenham o mesmo valor em apenas um dos campos.
- Você pode utilizar as palavras reservadas PRIMARY KEY para designar um campo ou conjunto de campos em uma tabela como uma chave primária. Todos os valores na chave primária devem ser exclusivos e não Nulos e só pode haver uma chave primária para uma tabela.
  Observação: Não defina uma restrição PRIMARY KEY em uma tabela que já tenha uma chave primária; se você o fizer, ocorrerá um erro.
- Você pode utilizar as palavras reservadas FOREIGN KEY para designar um campo como uma chave externa. Se a chave primária da tabela externa consistir em mais de um campo, você deverá utilizar uma definição de restrição de campos múltiplos, listando todos os campos referenciais, o nome da tabela externa e os nomes dos campos referenciados na tabela externa, na mesma ordem em que os campos referenciais são listados. Se o campo ou campos referenciados forem a chave primária da tabela externa, você não precisará especificar os campos referenciados — por padrão, o mecanismo de banco de dados se comporta como se a chave primária da tabela externa fosse os campos referenciados.

Exemplo:

```
/* CRIA TABELA AUTOR */

CREATE TABLE AUTOR(
COD_AUTOR AUTOINCREMENT NOT NULL
  CONSTRAINT CHAVEAUTOR PRIMARY KEY,
NOME VARCHAR(30) NOT NULL,
NASCIMENTO DATE,
CONSTRAINT UNICOAUTOR UNIQUE( NOME, NASCIMENTO) );
```

```
/* CRIA TABELA EDITORA */

CREATE TABLE EDITORA(
COD_EDITORA AUTOINCREMENT NOT NULL
  CONSTRAINT CHAVEAUTOR PRIMARY KEY,
RAZAO VARCHAR(30),
ENDERECO TEXT(30),
CIDADE TEXT(30));

/* CRIA TABELA LIVRO */

CREATE TABLE LIVRO(
TITULO TEXT(30) NOT NULL,
COD_AUTOR INTEGER NOT NULL,
COD_EDITORA INTEGER NOT NULL,
VALOR REAL,
COMENTARIO MEMO,
PUBLICACAO DATE,
VOLUME INTEGER,
CONSTRAINT CHAVELIVRO PRIMARY KEY (TITULO, COD_AUTOR),
CONSTRAINT LIVROINHERITAUTOR FOREIGN KEY (COD_AUTOR)
  REFERENCES AUTOR,
CONSTRAINT LIVROINHERITEDITORA FOREIGN KEY (COD_EDITORA)
  REFERENCES EDITORA);
```

1.2) Instrução CREATE INDEX

Cria um novo índice em uma tabela existente.

Sintaxe
CREATE [UNIQUE] INDEX *índice*
ON *tabela* (*campo* [ASC|DESC][, *campo* [ASC|DESC], ...])
[WITH { PRIMARY | DISALLOW NULL | IGNORE NULL }]

A instrução CREATE INDEX tem estas partes:

Parte	Descrição
índice	O nome do índice a ser criado.
tabela	O nome da tabela existente que conterá o índice.
campo	O nome do campo ou campos a serem indexados. Para criar um índice de campo único, liste o nome do campo entre parênteses após o nome da tabela. Para criar um índice de campos múltiplos, liste o nome de cada campo a ser incluído no índice. Para criar índices descendentes, use a palavra reservada DESC; caso contrário, assume-se que os índices são ascendentes.

Comentários:
O nome do índice é obrigatório.
Para proibir valores duplicados no campo ou campos indexados de diferentes registros, use a palavra reservada UNIQUE.
Na cláusula WITH opcional, você pode impor regras de validação de dados. Você pode:

- Proibir entradas Null no campo ou campos indexados dos novos registros, utilizando a opção DISALLOW NULL.
- Impedir que registros com valores Null no campo ou campos indexados sejam incluídos no índice utilizando a opção IGNORE NULL.

- Designar o campo ou campos indexados como a chave primária utilizando a palavra reservada PRIMARY. Isto significa que a chave é exclusiva e, portanto, você pode omitir a palavra reservada UNIQUE.

Você também pode utilizar a instrução ALTER TABLE para adicionar um índice de campo único ou de campos múltiplos a uma tabela e pode utilizar a instrução ALTER TABLE ou a instrução DROP para remover um índice criado com ALTER TABLE ou CREATE INDEX.

Observação:
Não use a palavra reservada PRIMARY ao criar um novo índice numa tabela que já tenha uma chave primária; se você o fizer, ocorrerá um erro.

Exemplo:

```
/* CRIA INDICE DO CAMPO NASCIMENTO,
ORDEM DESCENDENTE PARA TABELA AUTOR */

CREATE INDEX INDNASCIMENTO
ON AUTOR (NASCIMENTO DESC)
WITH IGNORE NULL;

/* CRIA INDICE DO CAMPO RAZAO,
ORDEM ASCENDENTE PARA TABELA EDITORA */

CREATE INDEX INDRAZAO
ON EDITORA (RAZAO ASC);
```

1.3) Instrução ALTER TABLE

Modifica a estrutura de uma tabela depois de ter sido criada com a instrução CREATE TABLE.

Sintaxe:
ALTER TABLE *tabela* {ADD {COLUMN *campo tipo*[(*tamanho*)] [NOT NULL] [CONSTRAINT *índice*] |
CONSTRAINT *índicedemulticampos*} |
DROP {COLUMN *campo* I CONSTRAINT *nomedoíndice*} }

A instrução ALTER TABLE tem estas partes:

Parte	Descrição
tabela	O nome da tabela a ser alterada.
campo	O nome do campo a ser adicionado ou excluído da tabela.
tipo	O tipo de dados de campo.
tamanho	O tamanho do campo em caracteres (somente os campos Texto e Binário).
índice	O índice para campo. Consulte o tópico da cláusula CONSTRAINT para maiores informações sobre como construir este índice.
Índicedemulticampos	A definição de um índice de campos múltiplos a ser adicionado à tabela. Consulte o tópico da cláusula CONSTRAINT para maiores informações sobre como construir esta cláusula.
nomedoíndice	O nome do índice de campo múltiplo a ser removido.

Comentários:
Através da instrução ALTER TABLE, você pode alterar uma tabela existente de diversas maneiras. Você pode:

- Utilizar ADD COLUMN para adicionar um novo campo à tabela. Você especifica o nome do campo, tipo de dados e (para campos Texto e Binário) um tamanho opcional.
- Você também pode definir um índice naquele campo.

- Se você especificar NOT NULL para um campo, então os novos registros deverão ter dados válidos naquele campo.
- Utilizar ADD CONSTRAINT para adicionar um índice de campos múltiplos.
- Utilizar DROP COLUMN para excluir um campo. Você especifica somente o nome do campo.
- Utilizar DROP CONSTRAINT para excluir um índice de campos múltiplos. Você especifica somente o nome do índice após a palavra reservada CONSTRAINT.

Observações:
- Você não pode adicionar ou excluir mais de um campo ou índice de cada vez.
- Você pode utilizar NOT NULL em um campo único ou dentro de uma cláusula CONSTRAINT nomeada que se aplica tanto a uma CONSTRAINT de campo único ou campos múltiplos. Contudo, você pode aplicar a restrição NOT NULL somente uma vez a um campo pois, senão, ocorrerá um erro em tempo de execução.

Exemplo:

```
/* EXCLUI A CHAVE ESTRANGEIRA
LIVROINHERITEDITORA DA TABELA LIVRO */

ALTER TABLE LIVRO
DROP CONSTRAINT LIVROINHERITEDITORA ;

/* INCLUI A CHAVE ESTRANGEIRA
LIVROINHERITEDITORA DA TABELA LIVRO */

ALTER TABLE LIVRO
ADD CONSTRAINT LIVROINHERITEDITORA
  FOREIGN KEY (COD_EDITORA)
  REFERENCES EDITORA;

/* EXCLUI O CAMPO VOLUME NA TABELA LIVRO */

ALTER TABLE LIVRO
DROP COLUMN VOLUME;

/* ACRESCENTA O CAMPO EDICAO NA TABELA LIVRO */

ALTER TABLE LIVRO
ADD COLUMN EDICAO INTEGER;
```

1.4) Instrução DROP

Exclui uma tabela existente de um banco de dados ou exclui um índice existente de uma tabela.

Sintaxe:
DROP {TABLE tabela | INDEX índice ON tabela}

A instrução DROP tem estas partes:

Parte	Descrição
tabela	O nome da tabela a ser excluída ou a tabela a partir da qual um índice deve ser excluído.
índice	O nome do índice a ser excluído da tabela.

Comentários:
Você também pode utilizar ALTER TABLE para excluir um índice de uma tabela.

Você pode utilizar CREATE TABLE para criar uma tabela e CREATE INDEX ou ALTER TABLE para criar um índice. Para modificar uma tabela, use ALTER TABLE.

Exemplo:

```
/* APAGA TABELA LIVRO */
DROP TABLE LIVRO;

/* APAGA TABELA AUTOR */
DROP TABLE AUTOR;

/* APAGA TABELA EDITORA */
DROP TABLE EDITORA;
```

2) DML – Comandos para a manipulação de Relações

2.1) Instrução SELECT

Efetua consultas em uma Base de Dados.

Sintaxe:
SELECT [*atributo*] { * | *tabela.* * | [*tabela.*]*campo1* [AS *alias1*] [, [*tabela.*]*campo2* [AS *alias2*] [, ...]]}
FROM *expressãodetabela* [, ...]
[WHERE...]
[GROUP BY...]
[HAVING...]
[ORDER BY...]

A instrução SELECT possui as partes a seguir:

Parte	Descrição
atributo	Um dos atributos a seguir: ALL, DISTINCT, DISTINCTROW ou TOP. Você utiliza o atributo para restringir o número de registros retornados. Se nenhum for especificado, o padrão será ALL.
*	Especifica que todos os campos da tabela ou tabelas especificadas estão selecionados.
tabela	O nome da tabela contendo os campos a partir dos quais os registros são selecionados.
campo1, *campo2*	Os nomes dos campos contendo os dados que você deseja recuperar. Se você incluir mais de um campo, eles serão recuperados na ordem listada.
alias1, *alias2*	Os nomes a serem utilizados como cabeçalhos de coluna em lugar dos nomes originais de coluna em tabela.
Expressãodetabela	O nome da tabela ou tabelas contendo os dados que você deseja recuperar.

Comentários
Esta operação procura na tabela ou tabelas especificadas, extrai as colunas escolhidas, seleciona as linhas que atendem aos critérios e classifica ou agrupa as linhas resultantes na ordem especificada.
As instruções SELECT não alteram os dados no banco de dados.
A sintaxe mínima para uma instrução SELECT é: SELECT campos FROM tabela

Exemplo:

Recuperar nome, nascimento da tabela autor:
SELECT NOME, NASCIMENTO
FROM AUTOR;

Você pode utilizar um asterisco (*) para selecionar todos os campos em uma tabela. O exemplo a seguir seleciona todos os campos na tabela Autor:
SELECT * FROM AUTOR;

Se desejar um nome de campo diferente ou um nome não for sugerido pela expressão utilizada para gerar o campo, utilize a palavra reservada AS. O exemplo a seguir renomeia nome para autor:
SELECT NOME AS AUTOR, NASCIMENTO
FROM AUTOR;

Ao digitar um nome de campo que contém um espaço ou pontuação, coloque o nome entre colchetes ([ ]).
SELECT NOME AS [NOME DO AUTOR], NASCIMENTO
FROM AUTOR;

2.1.1) Cláusula FROM

Especifica as tabelas ou consultas que contêm os campos listados na instrução SELECT.

Sintaxe:
SELECT *listadecampos*
FROM *expressãodetabela*

Uma instrução SELECT contendo uma cláusula FROM possui as partes a seguir:

Parte	Descrição
listadecampos	O nome do campo ou campos a serem recuperados juntamente com quaisquer aliases de nome de campo, funções agregadas SQL, atributos de seleção (ALL, DISTINCT, DISTINCTROW ou TOP) ou outras opções da instrução SELECT.
expressãodetabela	Uma expressão que identifica uma ou mais tabelas a partir das quais os dados são recuperados. A expressão pode ser um simples nome de tabela, um nome de consulta salvo ou uma composição resultante de um INNER JOIN, LEFT JOIN, ou RIGHT JOIN.

Comentários:
FROM é exigido e segue-se a qualquer instrução SELECT.
A ordem dos nomes de tabela em expressãodetabela não é importante.

2.1.2) Cláusula WHERE

Especifica quais registros das tabelas listadas na cláusula FROM são afetados por uma instrução SELECT, UPDATE ou DELETE.

Sintaxe:
SELECT *listadecampos*
FROM *expressãodetabela*
WHERE *critérios*

Uma instrução SELECT contendo uma cláusula WHERE possui as partes a seguir:

Parte	Descrição
listadecampos	O nome do campo ou campos a serem recuperados juntamente com quaisquer aliases de nome de campo, atributos de seleção (ALL, DISTINCT, DISTINCTROW ou TOP) ou outras opções da instrução SELECT.
expressãodetabela	O nome da tabela ou tabelas a partir das quais os dados são recuperados.
critérios	Uma expressão que os registros devem atender para serem incluídos nos resultados da consulta.

Comentários:
São selecionados os registros que satisfazem às condições listadas na cláusula WHERE. Se você não especificar uma cláusula WHERE, a consulta retornará todas as linhas da tabela. Se você especificar mais de uma tabela na consulta e não tiver incluído uma cláusula WHERE ou uma cláusula JOIN, a consulta irá gerar um produto cartesiano das tabelas.
WHERE é opcional, mas quando incluído, se segue ao FROM.
WHERE é semelhante a HAVING. WHERE determina quais registros são selecionados. Da mesma forma, depois de os registros serem agrupados com GROUP BY, HAVING determina quais registros serão exibidos.

Utilize a cláusula WHERE para eliminar os registros que você não deseja que sejam agrupados por uma cláusula GROUP BY.
Utilize várias expressões para determinar quais registros a instrução SQL retorna.
Uma cláusula WHERE pode conter até 40 expressões vinculadas por operadores lógicos, como And e Or.

Exemplo:

A instrução SQL a seguir seleciona todos os livros cujo valor sejam maiores que R$ 60,00:
SELECT TITULO, VALOR
FROM LIVRO
WHERE VALOR > 60;

Quando você especifica o argumento critérios, os literais de data deverão estar no formato dos EUA. Por exemplo, 10 de maio de 1994 é escrito 10/5/94 no Brasil e 5/10/94 nos Estados Unidos. Certifique-se de colocar os literais de data entre sinais de número (#), como mostrado nos exemplos a seguir.
Para encontrar os livros publicados após 10 de maio de 1994 em um banco de dados dos Estados Unidos, você deve utilizar a instrução SQL a seguir:
SELECT *
FROM LIVRO
WHERE PUBLICACAO > #5/10/93#;

Você também pode utilizar a função DateValue, que é sensível às configurações internacionais estabelecidas pelo Microsoft Windows. Por exemplo se no Painel de Controle, Configurações Regionais, Data, Estilo da data abreviada = DD/MM/AA:
SELECT *
FROM LIVRO
WHERE PUBLICACAO > DATEVALUE('10/5/94');

Se um nome de campo for incluído em mais de uma tabela na cláusula FROM, coloque antes dele o nome da tabela e o operador . (ponto). No exemplo a seguir, o campo Cod_Autor está tanto na tabela Autor quanto na tabela Livro. A instrução SQL seleciona os nomes a partir da tabela Autor e os titulos a partir da tabela Livro:
SELECT AUTOR.NOME, LIVRO.TITULO
FROM AUTOR, LIVRO
WHERE AUTOR.COD_AUTOR = LIVRO.COD_AUTOR;

Pode-se renomear a tabela para facilitar a digitação, usando o mesmo exemplo:
SELECT AU.NOME, LI.TITULO
FROM AUTOR AU, LIVRO LI
WHERE AU.COD_AUTOR = LI.COD_AUTOR;

2.1.3) Cláusula GROUP BY

Combina registros com valores idênticos na lista de campos especificada em um único registro. Um valor de resumo é criado para cada registro se você incluir uma função agregada SQL, como Sum ou Count, na instrução SELECT.

Sintaxe:

SELECT *listadecampos*
FROM *tabela*
WHERE *critérios*
[GROUP BY *listadecamposdegrupo*]

Uma instrução SELECT contendo uma cláusula GROUP BY possui as partes a seguir:

Parte	Descrição
listadecampos	O nome do campo ou dos campos a serem recuperados juntamente com qualquer alias de nome de campo, funções agregadas SQL, atributos de seleção (ALL, DISTINCT, DISTINCTROW ou TOP) ou outras opções da instrução SELECT.
tabela	O nome da tabela a partir da qual os registros são recuperados. Para obter maiores informações, consulte a cláusula FROM.
critérios	Critérios de seleção. Se a instrução incluir uma cláusula WHERE, agrupará valores depois de aplicar as condições WHERE aos registros.
listadecamposdegrupo	Os nomes de até 10 campos utilizados para agrupar os registros. A ordem dos nomes de campo em listadecamposdegrupo determina os níveis de agrupamento do nível mais alto ao mais baixo do agrupamento.

Comentários:
GROUP BY é opcional.
Os valores de resumo são omitidos se não houver uma função agregada SQL na instrução SELECT.
Valores Null nos campos GROUP BY são agrupados e não são omitidos. Contudo, valores Null não são avaliados em nenhuma função SQL agregada.
Utilize a cláusula WHERE para excluir linhas que você não deseja que permaneçam agrupadas e utilize a cláusula HAVING para filtrar os registros depois de eles terem sido agrupados.
A menos que contenha dados de Memorando ou Objeto OLE, um campo na lista de campos GROUP BY pode se referir a qualquer campo em qualquer tabela listada na cláusula FROM, mesmo que o campo não esteja incluído na instrução SELECT, desde que a instrução SELECT inclua pelo menos uma função SQL agregada.
Todos os campos na lista de campos SELECT devem estar incluídos na cláusula GROUP BY ou serem incluídos como argumentos em uma função SQL agregada.

Exemplo:

```
/* TODAS EDITORAS QUE PUBLICARAM LIVRO */

SELECT Livro.Cod_Editora, Editora.Razao
FROM Livro, Editora
WHERE Livro.Cod_Editora
GROUP BY Cod_Editora;
```

2.1.4) Cláusula HAVING

Especifica quais registros agrupados são exibidos na instrução SELECT com uma cláusula GROUP BY. Depois de GROUP BY combinar os registros, HAVING exibirá qualquer registro agrupado pela cláusula GROUP BY que satisfaça às condições da cláusula HAVING.

Sintaxe:
SELECT *listadecampos*
FROM *tabela*
WHERE *critériosdeseleção*
GROUP BY *listadecamposdegrupo*
[HAVING *critériosdegrupo*]

Uma instrução SELECT contendo uma cláusula HAVING possui as partes a seguir:

Parte	Descrição
listadecampos	O nome do campo ou campos a serem recuperados juntamente com qualquer alias de nome de campo, funções SQL agregadas, atributos de seleção (ALL, DISTINCT, DISTINCTROW ou TOP) ou outras opções da instrução SELECT.
tabela	O nome da tabela a partir da qual os registros são recuperados. Para obter maiores informações, consulte a cláusula FROM.
critériosdeseleção	Critérios de seleção. Se a instrução inclui uma cláusula WHERE, agrupa valores depois de aplicar as condições WHERE aos registros.
listadecamposdegrupo	Os nomes de até 10 campos utilizados para agrupar registros. A ordem dos nomes de campo em listadecamposdegrupo determina os níveis de agrupamento do nível mais alto ao mais baixo de agrupamento.
critériosdegrupo	Uma expressão que determina quais registros agrupados exibir.

Comentários:
HAVING é opcional.
HAVING é semelhante a WHERE, que determina quais registros são selecionados. Depois de os registros serem agrupados com o GROUP BY, HAVING determina os registros a serem exibidos:
Uma cláusula HAVING pode conter até 40 expressões vinculadas por operadores lógicos, como And e Or.

Exemplo:

```
/* TOTAL DE PUBLICAÇÕES DAS EDITORAS MAIOR QUE 1*/

SELECT COD_EDITORA AS [EDITORA],
  COUNT(COD_EDITORA) AS [TOTAL PUBLICACAO]
FROM LIVRO
GROUP BY COD_EDITORA
HAVING COUNT(COD_EDITORA) > 1;
```

2.1.5) Cláusula ORDER BY

Classifica os registros resultantes de uma consulta em um campo ou campos especificados, em ordem crescente ou decrescente.

Sintaxe:
SELECT *listadecampos*
FROM *tabela*
WHERE *critériosdeseleção*
[ORDER BY *campo1* [ASC | DESC][, *campo2* [ASC | DESC]][, ...]]]

Uma instrução SELECT contendo uma cláusula ORDER BY possui as partes a seguir:

Parte	Descrição
listadecampos	O nome do campo ou campos a serem recuperados juntamente com qualquer alias de nome de campo, funções SQL agregadas, atributos de seleção (ALL, DISTINCT, DISTINCTROW ou TOP) ou outras opções da instrução SELECT.
tabela	O nome da tabela a partir da qual os registros são recuperados. Para obter maiores informações, consulte a cláusula FROM.
critériosdeseleção	Critérios de seleção. Se a instrução incluir uma cláusula WHERE, ordenará os valores depois de aplicar as condições WHERE aos registros.
campo1, campo2	Os nomes dos campos nos quais os registros devem ser classificados.

Comentários:
ORDER BY é opcional. Entretanto, se desejar que os dados sejam exibidos em ordem classificada, então você deverá utilizar ORDER BY.
A ordem de classificação padrão é ascendente (de A a Z, de 0 a 9).
A ordem de classificação descendente (de Z a A, de 9 a 0).
Se você especificar um campo contendo dados de Memorando ou Objeto OLE na cláusula ORDER BY, um erro será gerado.
ORDER BY é geralmente o último item em uma instrução SQL.
Você pode incluir campos adicionais na cláusula ORDER BY. Os registros são classificados pelo primeiro campo listado após ORDER BY. Os registros que têm valores iguais naquele campo serão então classificados pelo valor no segundo campo listado e assim por diante.

Exemplo:

```
/* SELECIONA NOME E NASCIMENTO DA TABELA AUTOR
EM ORDER DE NOME DO AUTOR */

SELECT NOME, NASCIMENTO
FROM AUTOR
ORDER BY NOME;

/* SELECIONA NOME E NASCIMENTO DA TABELA AUTOR
EM ORDER DOS AUTORES MAIS NOVOS */

SELECT NOME, NASCIMENTO
FROM AUTOR
ORDER BY NASCIMENTO DESC;
```

2.1.6) Atributos ALL, DISTINCT, DISTINCTROW e TOP

Especifica registros selecionados com consultas SQL.

Sintaxe:
SELECT [*ALL* | *DISTINCT* | *DISTINCTROW* | [*TOP n [PERCENT]*]]
FROM *tabela*

Uma instrução SELECT contendo estes atributos possui as partes a seguir:

Parte	Descrição
ALL	Adotada quando você não inclui um dos atributos. Seleciona todos os registros que atendam às condições na instrução SQL.
DISTINCT	Omite registros que contêm dados duplicados nos campos selecionados. Para serem incluídos nos resultados da consulta, os valores de cada campo listado na instrução SELECT devem ser exclusivos.
DISTINCTROW	Omite dados baseado em registros duplicados completos, e não somente campos duplicados.
TOP n [PERCENT]	Retorna um certo número de registros que caem no topo ou na base de um intervalo especificado por uma cláusula ORDER BY.
tabela	O nome da tabela a partir da qual os registros são recuperados.

Comentário:
Você também pode utilizar a palavra reservada PERCENT para retornar uma certa porcentagem de registros que se situem no topo ou na base de um intervalo especificado pela cláusula ORDER BY.

Exemplo:

```
/* SELECIONA AS EDITORAS QUE PUBLICARAM LIVROS NO ANO DE 1994,
SEM REPETIÇAO */

SELECT DISTINCT ED.RAZAO
FROM EDITORA ED, LIVRO LI
WHERE (ED.COD_EDITORA = LI.COD_EDITORA AND
LI.PUBLICACAO >= DATEVALUE('01/01/94') AND
LI.PUBLICACAO <= DATEVALUE('31/12/94') );

/* SELECIONA OS TRES AUTORES MAIS VELHOS */

SELECT TOP 3 *
FROM AUTOR
ORDER BY NASCIMENTO;

/* SELECIONA 50 % DOS AUTORES MAIS VELHOS */

SELECT TOP 50 PERCENT *
FROM AUTOR
ORDER BY NASCIMENTO;
```

2.1.7) Operação INNER JOIN

Combina registros de duas tabelas sempre que houver valores correspondentes em um campo comum.

Sintaxe:
FROM *tabela1* INNER JOIN *tabela2*
ON *tabela1.campo1 opercomp tabela2.campo2*

Você pode também vincular várias cláusulas ON em uma instrução JOIN, usando a sintaxe a seguir:

```
SELECT campos
FROM tabela1 INNER JOIN tabela2
ON tabela1.campo1 opercomp tabela2.campo1 AND
ON tabela1.campo2 opercomp tabela2.campo2) OR
ON tabela1.campo3 opercomp tabela2.campo3)];
```

Você pode também aninhar instruções JOIN usando a seguinte sintaxe:

```
SELECT campos
FROM tabela1 INNER JOIN
(tabela2 INNER JOIN [( ]tabela3
[INNER JOIN [( ]tabelax [INNER JOIN ...)]
ON tabela3.campo3 opercomp tabelax.campox)]
ON tabela2.campo2 opercomp tabela3.campo3)
ON tabela1.campo1 opercomp tabela2.campo2;
```

A operação INNER JOIN possui as partes a seguir:

Parte	Descrição
tabela1, tabela2	Os nomes das tabelas das quais os registros são combinados.
campo1, campo2	Os nomes dos campos que são associados. Se não forem numéricos, os campos deverão ser do mesmo tipo de dados e conter o mesmo tipo de dados, mas não precisarão ter o mesmo nome.
opercomp	Qualquer operador de comparação relacional: "=," "<," ">," "<=," ">=," ou "<>."

Comentários
Você pode usar uma operação INNER JOIN em qualquer cláusula FROM. Este é o tipo mais comum de associação. As associações internas combinam registros de duas tabelas sempre que houver valores correspondentes em um campo comum a ambas.
Você pode associar quaisquer dois campos numéricos de tipos semelhantes. Por exemplo, você pode associar em campos AutoNumeração e Longo pois são tipos semelhantes. Entretanto, você não pode associar os tipos de campo Único e Duplo.
Uma instrução LEFT JOIN ou RIGHT JOIN pode ser aninhada dentro de uma INNER JOIN, mas uma INNER JOIN não pode ser aninhada dentro de uma LEFT JOIN ou de uma RIGHT JOIN.

Exemplo:

```
/* SELECIONA NOME DA TABELA AUTOR E TITULO DA TABELA LIVRO */

SELECT AUTOR.NOME, LIVRO.TITULO
FROM AUTOR INNER JOIN LIVRO
ON AUTOR.COD_AUTOR = LIVRO.COD_AUTOR;

/* SELECIONA OS NOME DE AUTOR, TITULO DE LIVRO E RAZAO DA EDITORA */

SELECT AUTOR.NOME, LIVRO.TITULO, EDITORA.RAZAO
FROM AUTOR INNER JOIN (LIVRO INNER JOIN EDITORA
ON LIVRO.COD_EDITORA = EDITORA.COD_EDITORA)
ON AUTOR.COD_AUTOR = LIVRO.COD_AUTOR;
```

2.1.8) Operações LEFT JOIN, RIGHT JOIN

Combina registros da tabela de origem quando usados em qualquer cláusula FROM.

Sintaxe:
FROM *tabela1* [LEFT | RIGHT] JOIN *tabela2*
ON *tabela1.campo1 opercomp tabela2.campo2*

As operações LEFT JOIN e RIGHT JOIN possuem as partes a seguir:

Parte	Descrição
tabela1, tabela2	Os nomes das tabelas das quais os registros são combinados.
campo1, campo2	Os nomes dos campos que são associados. Os campos devem ser do mesmo tipo de dados e conter o mesmo tipo de dados, mas não precisam ter o mesmo nome.
opercomp	Qualquer operador de comparação relacional: "=," "<," ">," "<=," ">=," ou "<>."

Comentários:
Utilize uma operação LEFT JOIN para criar uma associação externa esquerda. As associações externas esquerdas incluem todos os registros da primeira (esquerda) de duas tabelas, mesmo que não haja valores correspondentes para os registros na segunda tabela (direita).
Utilize uma operação RIGHT JOIN para criar uma associação externa direita. As associações externas direitas incluem todos os registros da segunda (direita) de duas tabelas, mesmo que não haja valores correspondentes para registros na primeira (esquerda) tabela.

Observações:
- Para criar uma consulta que inclua somente registros nos quais os dados nos campos associados sejam os mesmos, utilize uma operação INNER JOIN.

- Uma operação LEFT JOIN ou RIGHT JOIN pode ser aninhada dentro de uma operação INNER JOIN, mas uma INNER JOIN não pode ser aninhada dentro de uma LEFT JOIN ou de uma RIGHT JOIN. Consulte o tópico INNER JOIN para ver como aninhar associações dentro de outras associações.
- Você pode vincular várias cláusulas ON. Consulte o tópico INNER JOIN para ver como isso é feito.
- Se você tentar associar campos contendo dados de Memorando ou Objeto OLE, ocorrerá um erro.

Exemplo:

```
/* SELECIONAR TODAS AS EDITORAS QUE NÃO
TENHAM LIVROS ASSOCIADOS A ELAS */

SELECT EDITORA.RAZAO
FROM EDITORA LEFT JOIN LIVRO
ON EDITORA.COD_EDITORA = LIVRO.COD_EDITORA
WHERE LIVRO.COD_EDITORA = NULL;
```

A operação LEFT JOIN faz uma junção de todas as editora com todos os livros, obedecendo o critério após ON e adiciona também as editora que não possuem correspondência em livros. A cláusula WHERE filtra a seleção dos que em Livro.cod_editora é null.

2.1.9) Operação UNION

Cria uma consulta união, que combina os resultados de duas ou mais consultas ou tabelas independentes.

Sintaxe:
[TABLE] *consulta1* UNION [ALL] [TABLE] *consulta2* [UNION [ALL] [TABLE] *consultan* [...]]

A operação UNION possui as partes a seguir:

Parte	Descrição
consulta1-n	Uma instrução SELECT, o nome de uma consulta armazenada ou o nome de uma tabela armazenada precedida da palavra-chave TABLE.

Comentários:
Você pode mesclar os resultados de duas ou mais consultas, tabelas e instruções SELECT, em qualquer combinação, em uma única operação UNION.
Como padrão, nenhum registro duplicado é retornado quando você usa uma operação UNION; entretanto, você pode incluir o atributo ALL para assegurar que todos os registros sejam retornados, mesmo duplicados (isso faz com que a execução da consulta seja mais rápida).
Todas as consultas em uma operação UNION devem solicitar o mesmo número de campos; contudo, os campos deverão ter o mesmo tamanho ou tipo de dados.
Use aliases somente na primeira instrução SELECT pois eles são ignorados em qualquer outra. Na cláusula ORDER BY, refira-se aos campos pelo que são chamados na primeira instrução SELECT.

Observações:
- Você pode usar uma cláusula GROUP BY ou HAVING em cada argumento consulta para agrupar os dados retornados.
- Você pode usar uma cláusula ORDER BY no fim do último argumento consulta para exibir os dados retornados em uma ordem especificada.

Exemplo:

```
/* UNIR NOME E NASCIMENTO DA TABELA FUNCIONÁRIO COM
NOME E NASCIMENTO DA TABELA CLIENTE, SEM QUE HAJA REPETIÇÃO,
POIS HÁ CLIENTES QUE SÃO FUNCIONÁRIOS DA EMPRESA */
```

```
SELECT NOME, NASCIMENTO
FROM FUNCIONARIO
UNION
  SELECT NOME, NASCIMENTO
  FROM CLIENTE;
```

2.1.10) Operador In

Determina se o valor de uma expressão é igual a algum dos vários valores em uma lista especificada.

Sintaxe:
expr [Not] In(*valor1, valor2, . . .*)

A sintaxe do operador In possui as partes a seguir:

Parte	Descrição
expr	Expressão que identifica o campo que contém os dados a serem avaliados.
valor1, valor2	Expressão ou lista de expressões em relação a qual você deseja avaliar expr.

Comentário:
Se expr for encontrado na lista de valores, o operador In retornará True; caso contrário, retornará False.
Você pode incluir o operador lógico Not para avaliar a condição oposta (isto é, se expr não está na lista de valores).

Exemplo:

```
/* SELECIONAR EDITORA LOCALIZADAS NAS CIDADES DE
SÃO PAULO, SÃO CARLOS E RIO DE JANEIRO */

SELECT *
FROM EDITORA
WHERE CIDADE IN ('SÃO PAULO', 'SÃO CARLOS', 'RIO DE JANEIRO');
```

2.1.11) Operador Between...And

Determina se o valor de uma expressão se situa dentro de um intervalo especificado de valores. Você pode utilizar este operador em instruções SQL.

Sintaxe:
expr [Not] Between *valor1* And *valor2*

A sintaxe do operador Between...And possui as partes a seguir:

Parte	Descrição
expr	Expressão que identifica o campo que contém os dados a serem avaliados.
valor1, valor2	Expressões em relação as quais você deseja avaliar expr.

Comentários:
Se o valor de expr estiver entre valor1 e valor2 (inclusive), o operador Between...And retornará True; caso contrário, retornará False.
Você pode incluir o operador lógico Not para avaliar a condição oposta (isto é, se expr estiver situado fora do intervalo definido por valor1 e valor2).
Você poderia utilizar Between...And para determinar se o valor de um campo está situado em um intervalo numérico especificado.

Exemplo:

/* SELECIONAR LIVROS COM VALOR ENTRE R$ 50,00 e R$ 60,00 */

```
SELECT TITULO, VALOR
FROM LIVRO
WHERE VALOR BETWEEN 50 AND 60;
```

2.1.12) Operador Like

Compara uma expressão de seqüência com um padrão em uma expressão SQL.

Sintaxe:
expressão Like "*padrão*"

A sintaxe do operador Like possui as partes a seguir:

Parte	Descrição
expressão	Expressão SQL utilizada em uma cláusula WHERE.
padrão	Seqüência ou literal de seqüência de caracteres em relação a qual expressão é comparada.

Comentários:
Você pode utilizar o operador Like para localizar valores em um campo que correspondam ao padrão especificado. Para padrão, você pode especificar o valor completo (por exemplo, Like 'MAKROMBOOKS') ou pode utilizar caracteres curinga para encontrar um intervalo de valores (por exemplo, Like 'MAK*').
A correspondência interna de padrões oferece uma ferramenta versátil para fazer comparações de seqüências.
A tabela a seguir mostra os caracteres curinga que você pode utilizar com o operador Like e o número de dígitos ou seqüências aos quais eles podem corresponder.

Caractere(s) em padrão	Coincide com expressão
?	Qualquer caractere isolado
*	Zero ou mais caracteres
#	Qualquer dígito isolado (0 — 9)
[listadecaract]	Qualquer caractere isolado em listadecaract
[!listadecaract]	Qualquer caractere isolado não-presente em listadecaract

Você pode utilizar um grupo de um ou mais caracteres (listadecaract) entre colchetes ([ ]) para coincidir com qualquer caractere isolado em expressão, e listadecaract pode incluir praticamente qualquer caractere do conjunto de caracteres ANSI, inclusive dígitos. De fato, você pode utilizar os caracteres especiais colchete de abertura ([ ), ponto de interrogação (?), sinal de número (#) e asterisco (*) para que correspondam diretamente a eles mesmos somente se estiverem entre colchetes. Você pode utilizar o colchete de fechamento ( ]) dentro de um grupo para que corresponda a ele mesmo, mas pode utilizá-lo fora de um grupo como um caractere individual.
Além de uma simples lista de caracteres entre colchetes, listadecaract pode especificar um intervalo de caracteres através da utilização de um hífen (-) para separar os limites superior e inferior do intervalo. Por exemplo, a utilização de [A-Z] em padrão resultará em uma correspondência se a posição do caractere correspondente em expressão contiver qualquer uma das letras maiúsculas no intervalo de A a Z. Você pode incluir vários intervalos entre os colchetes sem delimitar os intervalos. Por exemplo, [a-zA-Z0-9] corresponde a qualquer caractere alfanumérico.
Outras regras importantes para correspondência de padrão incluem:

- Um ponto de exclamação (!) no início de listadecaract significa que uma correspondência será feita se qualquer caractere, exceto aqueles na listadecaract, for encontrado em expressão. Quando utilizado sem colchetes, o ponto de exclamação corresponderá a si mesmo.

- Você pode utilizar o hífen (-) seja no início (depois de um ponto de exclamação, se este for utilizado) ou no fim de listadecaract para corresponder a si mesmo. Se estiver em qualquer outro local, o hífen identificará um intervalo de caracteres ANSI.
- Quando você especifica um intervalo de caracteres, os caracteres devem aparecer em ordem de classificação crescente (A-Z ou 0-100). [A-Z] é um padrão válido, mas [Z-A] não é.
- A seqüência de caracteres [ ] é ignorada; ela é considerada uma seqüência de comprimento zero ("").

A tabela a seguir mostra como você pode utilizar Like para testar expressões para diferentes padrões.

Tipo de correspondência	Padrão	Coincidente (retorna True)	Não coincidente (retorna False)
Vários caracteres	a*a	aa, aBa, aBBBa	aBC
	ab	abc, AABB, Xab	aZb, bac
Caractere especial	a[*]a	a*a	aaa
Vários caracteres	ab*	abcdefg, abc	cab, aab
Um único caractere	a?a	aaa, a3a, aBa	aBBBa
Um único dígito	a#a	a0a, a1a, a2a	aaa, a10a
Intervalo de caracteres	[a-z]	f, p, j	2, &
Fora de um intervalo	[!a-z]	9, &, %	b, a
Nenhum dígito	[!0-9]	A, a, &, ~	0, 1, 9
Combinado	a[!b-m]#	An9, az0, a99	abc, aj0

Exemplo:

```
/* SELECIONAR NOME DE AUTOR QUE COMECE COM
LETRA 'A' ATÉ LETRA 'C' */

SELECT NOME
FROM AUTOR
WHERE NOME LIKE '[A-C]*';
```

2.1.13) Subconsultas SQL

Uma subconsulta é uma instrução SELECT aninhada dentro de uma instrução SELECT, SELECT...INTO, INSERT...INTO, DELETE ou UPDATE ou de uma outra subconsulta.

Sintaxe
Você pode usar três formas de sintaxe para criar uma subconsulta:

comparação [ANY | ALL | SOME] (*instruçãosql*)
expressão [NOT] IN (*instruçãosql*)
[NOT] EXISTS (*instruçãosql*)

Uma subconsulta possui as partes a seguir:

Parte	Descrição
comparação	Uma expressão e um operador de comparação que compara a expressão com os resultados da subconsulta.
expressão	Uma expressão para a qual o conjunto de resultados da subconsulta é procurado.
instruçãosql	Uma instrução SELECT, que segue o mesmo formato e regras de qualquer outra instrução SELECT. Deve estar entre parênteses.

Comentários:
Você pode usar uma consulta em lugar de uma expressão na lista de campos de uma instrução SELECT ou ema cláusula WHERE ou HAVING. Em uma subconsulta, você usa uma instrução SELECT para proporcionar um conjunto de um ou mais valores específicos para avaliar na expressão de cláusula WHERE ou HAVING.

Use o atributo ANY ou SOME, que são sinônimos, para recuperar registros na consulta principal que satisfaçam a comparação com qualquer registro recuperado na subconsulta.
Use o atributo ALL para recuperar somente os registros da consulta principal que satisfaçam a comparação com todos os registros recuperados na subconsulta.
Use o atributo IN para recuperar somente os registros da consulta principal para os quais algum registro na subconsulta contenha um valor igual. Da mesma forma, você pode usar NOT IN para recuperar somente os registros na consulta principal para os quais nenhum registro na subconsulta contenha um valor igual.
Use o atributo EXISTS (com a palavra reservada NOT opcional) em comparações verdadeiro/falso para determinar se a subconsulta retorna algum registro.
Você pode também usar aliases de nome de tabela em uma consulta para se referir a tabelas relacionadas em uma cláusula FROM fora da subconsulta.
Algumas subconsultas são permitidas em consultas de tabela de referência cruzada — especificamente, como atributos (os da cláusula WHERE). Subconsultas como saída (as da lista SELECT) não são permitidas em consultas de tabela de referência cruzada.

Exemplo:

```
/* SELECIONA OS LIVROS QUE AUTOR TEM DATA
NASCIMENTO < 01/01/60 */

SELECT COD_AUTOR, TITULO
FROM LIVRO
WHERE COD_AUTOR IN (
SELECT COD_AUTOR
FROM AUTOR
WHERE NASCIMENTO < DATEVALUE('01/01/60') );

/* SELECIONA OS LIVROS QUE DATA PUBLICACAO
SEJA MAIOR QUE TODA DATA DE NASCIMENTO DOS AUTORES */

SELECT TITULO
FROM LIVRO
WHERE PUBLICACAO > ANY(
SELECT NASCIMENTO
FROM AUTOR);
```

2.2) Instrução SELECT...INTO

Cria uma consulta criar tabela.

Sintaxe:
SELECT *campo1*[, *campo2*[, ...]] INTO *novatabela*
FROM *origem*

A instrução SELECT...INTO possui as partes a seguir:

Parte	Descrição
campo1, campo2	Os nomes dos campos a ser copiados na nova tabela.
novatabela	O nome da tabela a ser criada. Deve ser compatível com às convenções de nomenclatura padrão. Se novatabela for igual ao nome de uma tabela existente, ocorrerá um erro interceptável.
origem	O nome da tabela existente a partir da qual os registros são selecionados. Podem ser tabelas simples ou múltiplas ou um consulta.

Comentários
Você pode utilizar consultas criar tabela para arquivar registros, fazer cópias de backup das tabelas ou para usar como base para relatórios que exibem dados sobre um determinado período de tempo.

Observações:
- Convém definir uma chave primária para a nova tabela. Quando você cria a tabela, os campos na nova tabela herdam o tipo de dados e tamanho de campo de cada campo das tabelas base da consulta, mas nenhuma outra propriedade do campo ou da tabela é transferida.
- Para adicionar dados a uma tabela existente, use a instrução INSERT INTO em vez de criar uma consulta acréscimo.
- Para descobrir quais registros serão selecionados antes de executar a consulta criar tabela, examine antes os resultados de uma instrução SELECT que use os mesmos critérios de seleção.

Exemplo:

```
/* CRIA UMA TABELA AUTORVELHO COM OS
AUTORES QUE TEM NASCIMENTO < 01/01/60 */

SELECT *
INTO AUTORVELHO
FROM AUTOR
WHERE NASCIMENTO < DATEVALUE('01/01/60');
```

2.3) Instrução INSERT INTO

Adiciona um registro ou múltiplos registros a uma tabela. Isto é chamado de consulta acréscimo.

Sintaxe:
Consulta acréscimo de múltiplos registros:
INSERT INTO *destino* [(*campo1*[, *campo2*[, ...]])]
SELECT [*origem.*]*campo1*[, *campo2*[, ...]
FROM *expressãodetabela*

Consulta acréscimo de registro único:
INSERT INTO *destino* [(*campo1*[, *campo2*[, ...]])]
VALUES (*valor1*[, *valor2*[, ...])

A instrução INSERT INTO possui as partes a seguir:

Parte	Descrição
destino	O nome da tabela ou consulta à qual acrescentar os registros.
origem	O nome da tabela ou consulta a partir da qual os registros vão ser copiados.
campo1, campo2	Nomes dos campos aos quais os dados serão acrescentados, se se seguirem a um argumento destino, ou os nomes dos campos a partir dos quais os dados serão obtidos, se se seguirem a um argumento origem.
Expressãodetabela	O nome da tabela ou das tabelas das quais os registros são inseridos. Este argumento pode ser um nome de tabela simples ou um composto resultante de uma operação INNER JOIN, LEFT JOIN ou RIGHT JOIN ou uma consulta salva.
valor1, valor2	Os valores a serem inseridos nos campos específicos do novo registro. Cada valor é inserido no campo que corresponde à posição do valor na lista: valor1 é inserido no campo1 do novo registro, valor2 no campo2, e assim por diante. Você deve separar os valores com uma vírgula e colocar os campos de texto entre aspas (' ').

Comentários:
Você pode usar a instrução INSERT INTO para adicionar um único registro a uma tabela usando a sintaxe da consulta acréscimo de registro único como mostrado acima. Nesse caso, o código especifica o nome e o valor de cada campo do registro. Você deve especificar cada um dos campos do registro ao qual será atribuído um valor e um valor para aquele campo. Quando você não especifica os campos, o valor padrão ou Null é inserido para colunas ausentes. Os registros são adicionados ao fim da tabela.
Você pode também usar INSERT INTO para acrescentar um conjunto de registros de uma outra tabela ou consulta utilizando a cláusula SELECT ... FROM, como mostrado acima na sintaxe da consulta acréscimo de múltiplos registros. Nesse caso, a cláusula SELECT especifica os campos a acrescentar à tabela destino já especificada.
A tabela origem ou destino pode especificar uma tabela ou uma consulta. Se for especificada uma consulta, é acrescentado registros a todas as tabelas especificadas pela consulta.
INSERT INTO é opcional, mas, quando incluída, precede a instrução SELECT.
Se a tabela de destino contiver uma chave primária, certifique-se de acrescentar valores exclusivos não-Null ao campo ou campos da chave primária; se não o fizer, não acrescentará os registros.
Se você acrescentar os registros a uma tabela com um campo AutoNumeração e quiser renumerar os registros acrescentados, não inclua esse campo. Inclua o campo AutoNumeração na consulta se quiser conservar os valores originais do campo.
Para criar uma nova tabela, use a instrução SELECT... INTO em lugar de criar uma consulta criar tabela.
Para descobrir quais registros serão acrescentados antes de executar a consulta acréscimo, primeiro execute e visualize os resultados de uma consulta seleção que use os mesmos critérios de seleção.
Uma consulta acréscimo copia registros de uma ou mais tabelas para outra. As tabelas que contêm os registros que você acrescenta não são afetadas pela consulta acréscimo.
Em vez de acrescentar registros existentes de uma outra tabela, você pode especificar o valor de cada campo em um único registro novo, usando a cláusula VALUES. Se você omitir a lista de campos, a cláusula VALUES deverá incluir valores para todos os campos da tabela; caso contrário, a operação INSERT falhará. Use uma instrução INSERT INTO adicional com uma cláusula VALUES para cada registro adicional que quiser criar.

Exemplo:

```
/* INSERE UM REGISTRO NA TABELA AUTOR */

INSERT INTO AUTOR( NOME, NASCIMENTO )
VALUES ('RAMAKRISHMAN, R.', '13/05/60');

/* INSERE UM REGISTRO NA TABELA EDITORA */

INSERT INTO EDITORA( RAZAO, ENDERECO, CIDADE )
VALUES ('MCGRAW - HILL', 'R. SANTA HELENA, 321', 'SÃO JOSÉ DO RIO PRETO');

/* INSERE UM REGISTRO NA TABELA LIVRO */

INSERT INTO LIVRO
VALUES ('DATABASE MANAGEMENT SYSTEMS', 1, 1, 78.00,
'CONCEITOS FUNDAMENTAIS E MODELOS CONVENCIONAIS DE BANCO DE DADOS',
'15/03/1998', 1);

/* CRIA TABELA AUTORVELHO EM SEGUIDA
INSERE NA TABELA AUTORVELHO, OS AUTORES
COM NASCIMENTO < 01/01/60 */

CREATE TABLE AUTORVELHO(
COD_AUTOR AUTOINCREMENT NOT NULL
  CONSTRAINT CHAVEAUTOR PRIMARY KEY,
NOME VARCHAR(30) NOT NULL,
```

```
NASCIMENTO DATE,
CONSTRAINT UNICOAUTOR UNIQUE( NOME, NASCIMENTO) );

INSERT INTO AUTORVELHO
SELECT *
FROM AUTOR
WHERE NASCIMENTO < DATEVALUE('01/01/60');
```

2.4) Instrução UPDATE

Cria uma consulta atualização que altera valores de campos em uma tabela especificada, com base em critérios especificados.

Sintaxe:
UPDATE *tabela*
SET *novovalor*
WHERE *critérios*;

A instrução UPDATE possui as partes a seguir:

Parte	Descrição
tabela	O nome da tabela contendo os dados que você quer modificar.
novovalor	Uma expressão que determina o valor a ser inserido em um campo específico dos registros atualizados.
critérios	Uma expressão que determina quais registros serão atualizados. Apenas os registros que satisfazem à expressão são atualizados.

Comentários:
UPDATE é especialmente útil quando você quer alterar vários registros ou quando os registros que você quer alterar estão em várias tabelas.
Você pode alterar vários campos ao mesmo tempo.

Importante:
- UPDATE não gera um conjunto de resultados. Além disso, depois de atualizar os registros usando uma consulta de atualização, você não poderá desfazer a operação. Se quiser saber quais os registros que foram atualizados, examine antes os resultados de uma consulta seleção que usem os mesmos critérios e, depois, execute a consulta de atualização.
- Mantenha sempre cópias de backup dos dados. Se você atualizar os registros errados, poderá recuperá-los a partir das cópias.

Exemplo:

```
/* ALTERAR O ENDEREÇO E CIDADE DA TABELA
EDITORA COM COD_EDITORA = 1 */

UPDATE EDITORA
SET ENDERECO = 'AV. N.S. DE FATIMA, 123',
CIDADE = 'SANTA FÉ'
WHERE COD_EDITORA = 1;

/* REAJUSTA O VALOR DE TODOS DOS LIVROS EM 10% */

UPDATE LIVRO
SET VALOR = VALOR * 1.1
```

2.5) Instrução DELETE

Cria uma consulta exclusão que remove os registros de uma ou mais tabelas relacionadas na cláusula FROM que satisfaçam à cláusula WHERE.

Sintaxe:
DELETE [*tabela.**]
FROM *tabela*
WHERE *critérios*

A instrução DELETE possui as partes a seguir:

Parte	Descrição
*tabela.**	O nome opcional da tabela da qual são excluídos registros.
tabela	O nome da tabela da qual são excluídos registros.
critérios	Uma expressão que determina os registros a ser excluídos.

Comentários:
Para excluir uma tabela inteira do banco de dados, você pode usar o método Execute com uma instrução DROP. No entanto, se você excluir a tabela a estrutura será perdida. Em contrapartida, quando você usa DELETE, somente os dados são excluídos; a estrutura da tabela e todas as suas propriedades, como atributos de campo e índices, permanecem intactos.
Você pode usar DELETE para remover registros de tabelas que estão em um relacionamento um-para-muitos com outras tabelas. As operações de exclusão em cascata fazem com que os registros em tabelas que estão no lado muitos do relacionamento sejam excluídos quando o registro correspondente no lado um do relacionamento é excluído na consulta.

Importante:
- Depois de remover os registros utilizando uma consulta exclusão, você não poderá desfazer a operação. Se quiser saber quais registros foram excluídos, examine antes os resultados de uma consulta seleção que use os mesmos critérios e, depois, execute a consulta exclusão.
- Mantenha sempre cópias de backup dos dados. Assim, se você excluir os registros errados, poderá recuperá-los a partir das cópias.

Exemplo:

```
/* DELETA COM COD_AUTOR = 1 DA TABELA AUTOR */

DELETE
FROM AUTOR
WHERE COD_AUTOR = 1;

/* DELETA TODOS REGISTRO DA TABELA LIVRO */

DELETE
FROM LIVRO;
```

Instituto de Ciências Matemáticas - São Carlos - Biblioteca

3) Funções agregadas SQL

3.1) Função Avg

Calcula a média aritmética de um conjunto de valores contido em um campo especificado em uma consulta.

Sintaxe:
Avg(expr)

O espaço reservado expr representa uma expressão de seqüência que identifica o campo que contém os dados numéricos dos quais você quer tirar uma média ou uma expressão que realiza um cálculo utilizando os dados naquele campo. Os operandos em expr podem incluir o nome de um campo de tabela, uma constante ou uma função.

Comentários:
A média calculada por Avg é a média aritmética (a soma dos valores dividida pelo número de valores). Você poderia utilizar o Avg.
A função Avg não inclui nenhum campo Null no cálculo.

Exemplo:

```
/* MEDIA DE DATA VALOR DOS LIVROS */

SELECT AVG(VALOR) AS [MEDIA VALOR]
FROM LIVRO;
```

3.2) Função Count

Calcula o número de registros retornado por uma consulta.

Sintaxe:
Count(*expr*)

O espaço reservado expr representa uma expressão de seqüência de caracteres que identifica o campo que contém os dados que você quer contar ou uma expressão que realiza um cálculo utilizando os dados no campo. Os operandos em expr podem incluir o nome de um campo de tabela ou função. Você pode contar qualquer tipo de dados, inclusive texto.

Comentários:
Você pode utilizar Count para contar o número de registros em uma consulta base.
Embora expr possa realizar um cálculo em um campo, Count simplesmente computa o número de registros, independentemente dos valores armazenados nos registros.
A função Count não conta registros que tenham campos Null, exceto quando expr for o caractere curinga asterisco (*). Se você utilizar um asterisco, Count calculará o número total de registro, inclusive aqueles que contêm campos Null. Count(*) é consideravelmente mais rápido que Count([Nome da Coluna]). Não coloque o asterisco entre aspas (' ').
Se expr identificar vários campos, a função Count contará um registro somente se um dos campos não for Null. Se todos os campos especificados forem Null, o registro não será contado. Separe os nomes de campo com um e-comercial (&).

Exemplo:

```
/* CONTAR OS LIVROS PUBLICADO POR CADA EDITORA */

SELECT ED.RAZAO, COUNT(*) AS [TOTAL PUBLICADO]
FROM EDITORA ED, LIVRO LI
```

```
WHERE ED.COD_EDITORA = LI.COD_EDITORA
GROUP BY ED.RAZAO;

/* CONTAR QUANTOS LIVROS NÃO TEM PUBLICACAO E VALOR COM NULL */

SELECT COUNT ('PUBLICACAO & VALOR')
FROM LIVRO;
```

3.3) Funções Min, Max

Retornam o mínimo ou o máximo de um conjunto de valores contido em um campo especificado em uma consulta.

Sintaxe:
Min(*expr*)
Max(*expr*)

O espaço reservado expr representa uma expressão de seqüência que identifica o campo que contém os dados que você quer avaliar ou uma expressão que realiza um cálculo utilizando os dados naquele campo. Os operandos em expr podem incluir o nome de um campo de tabela, uma constante ou uma função.

Comentários:
Você pode utilizar Min e Max para determinar o menor e o maior valor em um campo, com base na agregação ou agrupamento especificado.

Exemplo:

```
/* SELECIONAR O VALOR DO LIVRO MAIS CARO */
SELECT MAX(VALOR) AS [VALOR MAIS CARO]
FROM LIVRO;

/* SELECIONAR O VALOR DO LIVRO MAIS BARATO */
SELECT MIN(VALOR) AS [VALOR MAIS BARATO]
FROM LIVRO;
```

3.4) Função Sum

Retorna a soma de um conjunto de valores contido em um campo especificado em uma consulta.

Sintaxe:
Sum(*expr*)

O espaço reservado expr representa uma expressão de seqüência que identifica o campo que contém os dados numéricos que você quer adicionar ou uma expressão que realiza um cálculo utilizando os dados naquele campo. Os operandos em expr podem incluir o nome de um campo de tabela, uma constante ou uma função.

Comentários:
A função Sum totaliza os valores em um campo.
A função Sum ignora os registros que contenham campos Null.

Exemplo:

```
/* SOMA DOS VALORES DOS LIVROS PULICADOS
PELA EDITORA COM COD_EDITORA = 2 */
SELECT SUM(VALOR) AS [SOMA EDITORA]
FROM LIVRO
WHERE COD_EDITORA = 2;
```

4) Comparação entre o SQL do mecanismo de banco de dados Microsoft Access e o SQL ANSI

O SQL do mecanismo de banco de dados Microsoft Access é geralmente compatível com ANSI-89 Nível 1. Entretanto, certos recursos SQL ANSI não são implementados no SQL do Microsoft Access. Por outro lado, o SQL do Microsoft Access inclui palavras reservadas de recursos não-suportados no SQL ANSI.

Principais diferenças:

- SQL do Microsoft Access e o SQL ANSI têm, cada um, diferentes palavras reservadas e tipos de dados.

- Diferentes regras se aplicam à construção Between...And, que possui a sintaxe a seguir:
  Expr1 [NOT] Between valor1 And valor2
  No SQL do Microsoft Access, valor1 pode ser maior que valor2; no SQL ANSI, valor1 deve ser igual ou menor que valor2.

- Diferentes caracteres curinga são utilizados com o operador Like.

Caractere correspondente	SQL do Microsoft Access	SQL ANSI
Qualquer caractere isolado	?	_ (sublinhado)
Zero ou mais caracteres	*	%

- Em geral, o SQL do Microsoft Access é menos restritivo. Por exemplo, ele permite agrupamento e classificação nas expressões.

- SQL do Microsoft Access suporta menos expressões avançadas.

4.1) Recursos avançados do SQL Access não suportados pelo SQL ANSY

4.1.1) Instrução TRANSFORM

Cria uma consulta tabela de referência cruzada. Uma consulta tabela de referência cruzada é uma consulta que calcula a soma, média, contagem ou outro tipo de total dos registros e em seguida, agrupa os resultados em dois tipos de informações – um ao longo do lado esquerdo de uma grade (títulos de linhas) e outro através do topo (títulos de colunas).

Sintaxe:
TRANSFORM *funçãoagrg*
instruçãoselect
PIVOT *campocentral* [IN (*valor1*[, *valor2*[, ...]])]

A instrução TRANSFORM possui as partes a seguir:

Parte	Descrição
funçãoagrg	Uma função agregada SQL que opera sobre os dados selecionados.
instruçãoselect	Uma instrução SELECT.
campodinâmicol	O campo ou expressão que você quer usar para criar títulos de coluna no conjunto de resultados da consulta.
valor1, valor2	Valores fixos usados para criar títulos de colunas.

Comentários:
Quando resume dados utilizando uma consulta tabela de referência cruzada, você seleciona valores de campos ou expressões especificadas como títulos de colunas para poder visualizar os dados em um formato mais compacto do que com uma consulta seleção.

A instrução TRANSFORM é opcional, mas quando for incluída será a primeira instrução em uma seqüência SQL. Ela antecede uma instrução SELECT que especifica os campos usados como títulos de linhas e uma cláusula GROUP BY que especifica o agrupamento de linhas. Opcionalmente, você pode incluir outras cláusulas, como WHERE, que especifiquem critérios adicionais de seleção ou de classificação. Você pode também usar subconsultas como atributos — especificamente, aqueles em uma cláusula WHERE — em uma consulta tabela de referência cruzada.
Os valores retornados em campodinâmico são usados como títulos de colunas no conjunto de resultados da consulta.

4.1.2) Funções StDev e StDevP

Retornam estimativas do desvio padrão para uma população ou amostra de população representada como um conjunto de valores contido em um campo especificado em uma consulta.

Sintaxe:
StDev(*expr*)
StDevP(*expr*)

O espaço reservado expr representa uma expressão de seqüência que identifica o campo que contém os dados numéricos que você quer avaliar ou uma expressão que realiza um cálculo utilizando os dados naquele campo. Os operandos em expr podem incluir o nome de um campo de tabela, uma constante ou uma função.

Comentários:
A função StDevP avalia uma população e a função StDev avalia uma amostra da população.
Se a consulta base contiver menos que dois registros (ou nenhum registro, para a função StDevP), estas funções retornarão um valor Null (que indica que não é possível calcular um desvio padrão).

Exemplo:

```
/* CALCULA O DESVIO PADRAO DO VALOR DOS LIVRO */

SELECT STDEV(VALOR) AS [DESVIO PADÃO]
FROM LIVRO;
```

4.1.3) Funções Var, VarP

Retornam estimativas da variação de uma população ou uma amostra da população representadas como um conjunto de valores contido em um campo especificado em uma consulta.

Sintaxe:
Var(*expr*)
VarP(*expr*)

O espaço reservado expr representa uma expressão de seqüência que identifica o campo que contém os dados numéricos que você quer avaliar ou uma expressão que realiza um cálculo utilizando os dados naquele campo. Os operandos em expr podem incluir o nome de um campo de tabela, uma constante ou uma função.

Comentários:
A função VarP avalia uma população e a função Var avalia uma amostra da população.
Se a consulta base contiver menos de dois registros, as funções Var e VarP retornarão um valor Null, que indica que não é possível calcular uma variância.

Exemplo:

```
/* CALCULA A VARIANCIA DO VALOR DOS LIVRO */
SELECT VAR(VALOR) AS [VARIÂNCIA]
FROM LIVRO;
```

4.2) Recursos do SQL ANSI não suportados no SQL do Microsoft Access

O SQL do Microsoft Access não suporta os recursos do SQL ANSI a seguir:

- Instruções de segurança, como COMMIT, GRANT e LOCK.
- Referências à função agregada DISTINCT. Por exemplo, o SQL do Microsoft Access não permite SUM(DISTINCT nomedecoluna).
- A cláusula LIMIT TO nn ROWS utilizada para limitar o número de linhas retornados por uma consulta. Você só pode utilizar a cláusula WHERE para limitar o escopo de uma consulta.

5) Tipos de dados SQL

Os tipos de dados SQL consistem em 13 tipos de dados primários e vários sinônimos válidos reconhecidos para esses tipos de dados.
A tabela a seguir lista os tipos de dados primários. Os sinônimos estão identificados em Palavras reservadas SQL :

Tipo de dados	Tamanho	Descrição
BINARY	1 byte por caractere	Qualquer tipo de dados que pode ser armazenado em um campo deste tipo. Não é feita nenhuma conversão dos dados (por exemplo, para texto). O modo como os dados são inseridos em um campo binário dita o modo como eles aparecerão como saída.
BIT	1 byte	Valores e campos Yes e No que contêm somente um entre dois valores.
BYTE	1 byte	Um valor inteiro entre 0 e 255.
COUNTER	4 bytes	Um número incrementado automaticamente sempre que um novo registro é adicionado à tabela. No mecanismo de banco de dados Microsoft Access, o tipo de dados para este valor é Long.
CURRENCY	8 bytes	Um número inteiro escalado entre – 922.337.203.685.477,5808 e 922.337.203.685.477,5807.
DATETIME (Consulte DOUBLE)	8 bytes	Um valor de data ou hora entre os anos 100 e 9999.
GUID	128 bits	Um número de identificação exclusivo utilizado com chamadas de procedimento remoto.
SINGLE	4 bytes	Um valor de ponto flutuante de precisão simples com um intervalo entre – 3,402823E38 a – 1,401298E-45 para valores negativos, 1,401298E-45 a 3,402823E38 para valores positivos e 0.
DOUBLE	8 bytes	Um valor de ponto flutuante de dupla precisão com um intervalo de – 1,79769313486232E308 a – 4,94065645841247E-324 para valores negativos, 4,94065645841247E-324 a 1.79769313486232E308 para valores positivos e 0.
SHORT	2 bytes	Um inteiro curto entre – 32.768 e 32.767.
LONG	4 bytes	Um inteiro longo entre – 2.147.483.648 e 2.147.483.647.
LONGTEXT	1 byte por caractere	Zero a um máximo de 1,2 gigabytes.
LONGBINARY	Conforme exigido	Zero a um máximo de 1,2 gigabytes. Utilizado para objetos OLE.
TEXT	1 byte por caractere	Zero a 255 caracteres.

Observação:
Você também pode utilizar a palavra reservada VALUE em instruções SQL.

5.1) Tipos de dados SQL ANSI equivalentes

A tabela a seguir lista os tipos de dados SQL ANSI e os tipos de dados equivalentes do Access e seus sinônimos válidos.

Tipo de dados SQL ANSI	Tipo de dados SQL do Access	Sinônimo
BIT, BIT VARYING	BINARY (Consulte Observações)	VARBINARY
Não-suportado	BIT (Consulte Observações)	BOOLEAN, LOGICAL, LOGICAL1, YESNO
Não-suportado	BYTE	INTEGER1
Não-suportado	COUNTER	AUTOINCREMENT
Não-suportado	CURRENCY	MONEY
DATE, TIME, TIMESTAMP	DATETIME	DATE, TIME, TIMESTAMP
Não-suportado	GUID	
DECIMAL	Não-suportado	
REAL	SINGLE	FLOAT4, IEEESINGLE, REAL
DOUBLE PRECISION, FLOAT	DOUBLE	FLOAT, FLOAT8, IEEEDOUBLE, NUMBER, NUMERIC
SMALLINT	SHORT	INTEGER2, SMALLINT
INTEGER	LONG	INT, INTEGER, INTEGER4
INTERVAL	Não-suportado	
Não-suportado	LONGBINARY	GENERAL, OLEOBJECT
Não-suportado	LONGTEXT	LONGCHAR, MEMO, NOTE
CHARACTER, CHARACTER VARYING	TEXT	ALPHANUMERIC, CHAR, CHARACTER, STRING, VARCHAR
Não-suportado	VALUE (Consulte Observações)	

Observações:

- Tipo de dados BIT do SQL ANSI não corresponde ao tipo de dados BIT do SQL Access, mas corresponde ao tipo de dados BINARY. Não há equivalente SQL ANSI para o tipo de dados.
- A palavra reservada VALUE não representa um tipo de dados definido.