Structured Query Language

SQL

Data Manipulation Language

DML

Linguagem de Manipulação de Dados

Prof. André Luiz Alves

1. INSERT INTO

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

1.1. Sintaxe

Acréscimo de múltiplos registros:

```
INSERT INTO destino [IN bancodedadosexterno] [(campo1[, campo2[, ...]])]
        SELECT [origem.]campo1[, campo2[, ...]
                FROM expressãodetabela
```

Consulta acréscimo de registro único:

```
INSERT INTO destino [(campo1[, campo2[, ...]])]
        VALUES (valor1[, valor2[, ...])
```

1.2. A instrução INSERT INTO possui as partes a seguir:

Parte	Descrição
destino	O nome da tabela ou consulta à qual acrescentar os registros.
bancodedadosexterno0	O caminho até um banco de dados externo. Para obter uma descrição do caminho, consulte a cláusula IN.
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 (' ').

1.3. 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 especificada.

A tabela origem ou destino pode especificar uma tabela ou uma consulta. Se for especificada uma consulta, o mecanismo de banco de dados Microsoft Jet acrescenta 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, o mecanismo de banco de dados Microsoft Jet 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.

Use a cláusula IN para acrescentar os registros a uma tabela em um outro banco de dados.
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.

2. DELETE

Remove os registros de uma ou mais tabelas relacionadas na cláusula FROM que satisfaçam à cláusula WHERE.

2.1. Sintaxe

DELETE [tabela.*] FROM tabela WHERE critérios

2.2. 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.

2.3. Comentários

DELETE é especialmente útil quando você quer excluir muitos registros.
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. Por exemplo, nos relacionamentos entre as tabelas Clientes e Pedidos, a tabela Clientes está no lado um e a tabela Pedidos está no lado muitos do relacionamento. A exclusão de um registro de Clientes resulta na

exclusão dos registros Pedidos correspondentes se a opção de exclusão em cascata estiver especificada.

O comando DELETE exclui registros inteiros, não somente os dados em campos específicos. Se você quiser excluir valores de um campo específico, crie uma consulta atualização que altere os valores para Null.

2.4. 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.

3. UPDATE

Altera valores de campos em uma tabela especificada, com base em critérios especificados.

3.1. Sintaxe

UPDATE tabela SET novovalor WHERE critérios;

3.2. 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.

3.3. 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. O exemplo a seguir aumenta os valores de Quantia do Pedido em 10% e os valores de Frete em 3% para transportadores no Reino Unido (UK):

```
UPDATE Pedidos
        SET QuantiaDoPedido = QuantiaDoPedido * 1.1,  Frete = Frete * 1.03
                WHERE PaísDeDestino = 'UK';
```

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.

4. SELECT INTO

Cria uma criar tabela.

4.1. Sintaxe

SELECT campo1[, campo2[, ...]] INTO novatabela [IN bancodedadosexterno] FROM origem

4.2. 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.
bancodedadosexterno	O caminho para um banco de dados externo. Para obter uma descrição do caminho, consulte a cláusula IN.
origem	O nome da tabela existente a partir da qual os registros são selecionados. Podem ser tabelas simples ou múltiplas ou um consulta.

4.3. Comentários

Você pode utilizar consultas criar tabela para arquivar registros, fazer cópias de backup das tabelas ou fazer cópias para exportar para um outro banco de dados, ou para usar como base para relatórios que exibem dados sobre um determinado período de tempo. Por exemplo, você poderia produzir um relatório de Vendas Mensais por Região, executando a mesma consulta criar tabela todos os meses.

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.

5. SELECT

Retorna informações do banco de dados como um conjunto de registros.

5.1. Sintaxe

```
SELECT [atributo] { * | tabela.* | [tabela.]campo1 [AS alias1] [, [tabela.]campo2 [AS alias2] [, ...]]}
    FROM expressãodetabela [, ...] [IN bancodedadosexterno]
    [WHERE... ]
    [GROUP BY... ]
    [HAVING... ]
    [ORDER BY... ]
    [WITH OWNERACCESS OPTION]
```

5.2. 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.
bancodedadosexterno	nome do banco de dados que contém as tabelas em expressãodetabela se elas não estiverem no banco de dados atual.

5.3. Comentários

Para executar esta operação, o mecanismo de banco de dados Microsoft Jet procura a 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.

SELECT é geralmente a primeira palavra em uma instrução SQL. A maioria das instruções SQL são instruções SELECT ou SELECT...INTO.

A sintaxe mínima para uma instrução SELECT é:

SELECT campos FROM tabela

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

SELECT * FROM Funcionários;

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 Departamento está tanto na tabela Funcionários quanto na tabela Supervisores. A instrução SQL seleciona os departamentos a partir da tabela Funcionários e os nomes de supervisores a partir da tabela Supervisores:

```
SELECT Funcionários.Departamento, Supervisores.SupvNome
FROM Funcionários INNER JOIN Supervisores
WHERE Funcionários.Departamento = Supervisores.Departamento;
```

Sempre que utilizar funções agregadas ou consultas que retornam nomes de objetos Field ambíguos ou duplicados, você deve utilizar a cláusula AS para fornecer um nome alternativo para o objeto Field. O exemplo a seguir utiliza o título ContagemDePessoas para nomear o objeto Field retornado no objeto Recordset resultante:

```
SELECT COUNT(CódigoDoFuncionário)
AS ContagemDePessoas FROM Funcionários;
```

Você pode utilizar as outras cláusulas em uma instrução SELECT para ampliar a restrição e organizar os dados retornados. Para maiores informações, consulte o tópico da Ajuda para a cláusula que você está utilizando.

6. Atributos ALL, DISTINCT, DISTINCTROW e TOP

Especifica registros selecionados com consultas SQL.

6.1. Sintaxe

```
SELECT [ALL | DISTINCT | DISTINCTROW | [TOP n [PERCENT]]]
    FROM tabela
```

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

Parte	Descrição
ALL	Adotada quando você não inclui um dos atributos. O mecanismo de banco de dados Microsoft Jet seleciona todos os registros que atendam às condições na instrução SQL. Os dois exemplos a seguir são equivalentes e retornam todos os registros da tabela Funcionários:

```
SELECT ALL *
    FROM Funcionários
    ORDER BY CódigoDoFuncionário;
```

```
SELECT *
    FROM Funcionários
    ORDER BY CódigoDoFuncionário;
```

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. Por exemplo, vários funcionários listados em uma tabela Funcionários podem ter o mesmo sobrenome. Se dois registros contiverem Smith no campo Sobrenome, a instrução SQL a seguir retornará somente um deles:

```
SELECT DISTINCT Sobrenome
    FROM Funcionários;
```

Se você omitir DISTINCT, esta consulta retornará os dois registros Smith.Se a cláusula SELECT contiver mais de um campo, a combinação de valores de todos os campos deverão ser exclusivos para que um dado registro seja incluído nos resultados.A saída de uma consulta que utiliza DISTINCT não é atualizável e não reflete alterações subseqüentes feitas por outros usuários.

DISTINCTROW	Omite dados baseado em registros duplicados completos, e não somente campos duplicados. Por exemplo, você poderia criar uma consulta que associasse as tabelas Cliente e Pedidos no campo CódigoDoCliente. A tabela Clientes não contém campos CódigoDoCliente duplicados, mas a tabela Pedidos contém, pois cada cliente pode fazer vários pedidos. A instrução SQL a seguir mostra como você pode utilizar DISTINCTROW para produzir uma lista de empresas que têm pelo menos um pedido, mas sem exibir detalhes sobre esses pedidos:

```
SELECT DISTINCTROW NomeDaEmpresa
    FROM Clientes INNER JOIN Pedidos
    ON Clientes.CódigoDoCliente = Pedidos.CódigoDoCliente
    ORDER BY NomeDaEmpresa;
```

Se você omitir DISTINCTROW, esta consulta produzirá várias linhas para cada empresa que tenha mais de um pedido.DISTINCTROW produz um efeito somente quando você seleciona campos de algumas, mas não todas, as tabelas utilizadas na consulta. DISTINCTROW será

ignorado se a consulta incluir somente uma tabela ou se você obtiver saída de campos de todas as tabelas.

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. Suponha que você deseje obter os nomes dos 25 melhores estudantes da classe de 1994:

```
SELECT TOP 25 Nome, Sobrenome
    FROM Estudantes
    WHERE AnoDeGraduação = 1994
    ORDER BY MédiaDeNotas DESC;
```

Se você não incluir a cláusula ORDER BY, a consulta retornará um conjunto arbitrário de 25 registros da tabela Estudantes que satisfaçam à cláusula WHERE.O atributo TOP não escolhe entre valores iguais. No exemplo anterior, se a vigésima quinta e a vigésima sexta melhores médias de notas forem iguais, a consulta retornará 26 registros.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. Suponha que, em vez dos 25 melhores estudantes, você queira os 10% inferiores da classe:

```
SELECT TOP 10 PERCENT Nome, Sobrenome
    FROM Estudantes
    WHERE AnoDeGraduação = 1994
    ORDER BY MédiaDeNotas ASC;
```

O atributo ASC especifica um retorno de valores inferiores. O valor que se segue a TOP deve ser um Integer sem sinal.TOP não afeta o fato de a consulta ser ou não atualizável.

tabela — O nome da tabela a partir da qual os registros são recuperados.

7. Cláusula FROM

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

7.1 Sintaxe

```
SELECT listadecampos
    FROM expressãodetabela [IN bancodedadosexterno]
```

7.2 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.

bancodedadosexterno	O caminho completo de um banco de dados externo contendo todas as tabelas em expressãodetabela.

7.3 Comentários

FROM é exigido e segue-se a qualquer instrução SELECT.

A ordem dos nomes de tabela em expressãodetabela não é importante.

Para um melhor desempenho e facilidade de utilização, convém utilizar uma tabela vinculada em vez de uma cláusula IN para recuperar dados de um banco de dados externo.
O exemplo a seguir mostra como você pode recuperar os dados da tabela Funcionários:

```
SELECT Sobrenome, Nome
    FROM Funcionários;
```

8 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.

8.1 Sintaxe

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

8.2 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, o mecanismo de banco de dados Microsoft Jet 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.

8.3 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.

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.

9. 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.

9.1 Sintaxe

```
SELECT listadecampos
    FROM tabela
    WHERE critériosdeseleção
    GROUP BY listadecamposdegrupo
    [HAVING critériosdegrupo]
```

9.2 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, o mecanismo de banco de dados Microsoft Jet 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.

9.3 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:

```
SELECT CódigoDaCategoria, Sum(UnidadesNoEstoque)
    FROM Produtos
    GROUP BY CódigoDaCategoria
    HAVING Sum(UnidadesNoEstoque) > 100 And Like "BOS*";
```

Uma cláusula HAVING pode conter até 40 expressões vinculadas por operadores lógicos, como And e Or.

10. Cláusula IN

Identifica tabelas em qualquer banco de dados externo ao qual o mecanismo de banco de dados Microsoft Jet pode se conectar, como um banco de dados dBASE ou Paradox ou um banco de dados Microsoft Jet externo.

10.1. Sintaxe

Para identificar uma tabela de destino:

```
[SELECT | INSERT] INTO destino IN
        {caminho | ["caminho" "tipo"] | ["" [tipo; DATABASE = caminho]]}
```

Para identificar uma tabela de origem:

```
FROM expressãodetabela IN
        {caminho | ["caminho" "tipo"] | ["" [tipo; DATABASE = caminho]]}
```

10.2. Uma instrução SELECT contendo uma cláusula IN possui as partes a seguir:

Parte	Descrição
destino	O nome da tabela externa à qual os dados são inseridos.
expressãodetabela	O nome da tabela ou tabelas a partir das quais os dados são recuperados. Este argumento pode ser um único nome de tabela, uma consulta salva ou uma composição resultante de um INNER JOIN, LEFT JOIN ou RIGHT JOIN.
caminho	O caminho completo para o diretório ou arquivo contendo tabela.
tipo	O nome do tipo de banco de dados utilizado para criar tabela se um banco de dados não for um banco de dados Microsoft Jet (por exemplo, dBASE III, dBASE IV, Paradox 3.x ou Paradox 4.x).

10.3. Comentários

Você pode utilizar IN para se conectar a um único banco de dados externo de cada vez.
Em alguns casos, o argumento caminho refere-se ao diretório que contém os arquivos do banco de dados. Por exemplo, ao trabalhar com tabelas de banco de dados dBASE, FoxPro ou Paradox, o argumento caminho especifica o diretório contendo os arquivos .dbf ou .db. O nome do arquivo de tabela deriva do argumento destino ou expressãodetabela.

Para especificar um banco de dados não-Microsoft Jet, anexe um ponto-e-vírgula (;) ao nome e coloque-o entre aspas simples (' ') ou duplas (" "). Por exemplo, tanto 'dBASE IV;' quanto "dBASE IV;" são aceitos.

Você também pode utilizar a palavra reservada DATABASE para especificar o banco de dados externo. Por exemplo, as linhas a seguir especificam a mesma tabela:

```
... FROM Table IN "" [dBASE IV; DATABASE=C:\DBASE\DATA\SALES;];
... FROM Table IN "C:\DBASE\DATA\SALES" "dBASE IV;"
```

Para um melhor desempenho e facilidade de uso, utilize uma tabela vinculada em lugar de IN.

Você também pode utilizar a palavra reservada IN como um operador de comparação em uma expressão. Para obter maiores informações, consulte o operador In.

11. Cláusula ORDER BY

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

11.1. Sintaxe

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

11.2. 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, o mecanismo de banco de dados Microsoft Jet ordenará os valores depois de aplicar as condições WHERE aos registros.
campo1, campo2	Os nomes dos campos nos quais os registros devem ser classificados.

11.3. 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). Os dois exemplos a seguir classificam os nomes dos funcionários pela ordem do sobrenome:

```
SELECT Sobrenome, Nome
    FROM Funcionários
    ORDER BY Sobrenome;
```

```
SELECT Sobrenome, Nome
    FROM Funcionários
    ORDER BY Sobrenome ASC;
```

Para classificar em ordem decrescente (de Z a A, de 9 a 0), adicione a palavra reservada DESC ao final de cada campo que você queira classificar em ordem decrescente. O exemplo a seguir seleciona os salários e os classifica em ordem decrescente:

```
SELECT Sobrenome, Salário
    FROM Funcionários
    ORDER BY Salário DESC, Sobrenome;
```

Se você especificar um campo contendo dados de Memorando ou Objeto OLE na cláusula ORDER BY, um erro será gerado. O mecanismo de banco de dados Microsoft Jet não classifica campos desses tipos.

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.

12. Cláusula WHERE

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

12.1. Sintaxe

```
SELECT listadecampos
    FROM expressãodetabela
    WHERE critérios
```

12.2. 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.

12.3. Comentários

O mecanismo de banco de dados Microsoft Jet seleciona 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. Por exemplo, você pode selecionar todos os funcionários no departamento de vendas (WHERE Depto = 'Vendas') ou todos os clientes com idades entre 18 e 30 anos (WHERE Idade Between 18 And 30).

Se você não utilizar uma cláusula JOIN para executar operações de associação SQL em várias tabelas, o objeto Recordset resultante não será atualizável.

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. Por exemplo, a instrução SQL a seguir seleciona todos os funcionários cujos salários sejam maiores que R$21.000:

```
SELECT Sobrenome, Salário
        FROM Funcionários
        WHERE Salário > 21000;
```

Uma cláusula WHERE pode conter até 40 expressões vinculadas por operadores lógicos, como And e Or.

Ao digitar um nome de campo que contém um espaço ou pontuação, coloque o nome entre colchetes ([ ]). Por exemplo, uma tabela de informações sobre o cliente poderia incluir informações sobre clientes específicos:

SELECT [Restaurante Favorito do Cliente]

Quando você especifica o argumento critérios, os literais de data deverão estar no formato dos EUA, mesmo que você não esteja utilizando a versão norte-americana do mecanismo de banco de dados Microsoft Jet. Por exemplo, 10 de maio de 1996 é escrito 10/5/96 no Brasil e 5/10/96 nos Estados Unidos. Certifique-se de colocar os literais de data entre sinais de número (#), como mostrado nos exemplos a seguir.

Para encontrar registros datados de 10 de maio de 1996 em um banco de dados dos Estados Unidos, você deve utilizar a instrução SQL a seguir:

```
SELECT *
    FROM Pedidos
    WHERE DataDeEnvio = #5/10/96#;
```

Você também pode utilizar a função DateValue, que é sensível às configurações internacionais estabelecidas pelo Microsoft Windows. Por exemplo, utilize este código para os Estados Unidos:

```
SELECT *
    FROM Pedidos
    WHERE DataDeEnvio = DateValue('5/10/96');
```

E utilize este código para o Brasil:

```
SELECT *
    FROM Pedidos
    WHERE DataDeEnvio = DateValue('10/5/96');
```

Se a coluna referenciada na sequência de critérios for do tipo GUID, a expressão de critérios utilizará uma sintaxe ligeiramente diferente:

```
WHERE ReplicaID = {GUID {12345678-90AB-CDEF-1234-567890ABCDEF}}
```

Não deixe de incluir as chaves e hifens embutidos, como mostrado.

13. Declaração WITH OWNERACCESS OPTION

Em um ambiente multiusuários com um grupo de trabalho seguro, use esta declaração com uma consulta para conceder ao usuário que executa a consulta as mesmas permissões que as do proprietário da consulta.

13.1. Sintaxe

instruçãosql WITH OWNERACCESS OPTION

13.2. Comentários

A declaração WITH OWNERACCESS OPTION é opcional.

O exemplo a seguir permite que o usuário visualize informações sobre salários (mesmo que, de outra forma, o usuário não tenha permissão para visualizar a tabela FolhaDePagamento), desde que o proprietário da consulta tenha essa permissão:

SELECT Sobrenome, Nome, Salário

```
FROM Funcionários
ORDER BY Sobrenome
WITH OWNERACCESS OPTION;
```

Se de alguma outra forma o usuário for impedido de criar ou adicionar a uma tabela, você pode usar WITH OWNERACCESS OPTION para permitir que ele execute uma consulta criar tabela ou consulta acréscimo.

Se você deseja impor configurações de segurança de grupo de trabalho e permissões de usuários, não inclua a declaração WITH OWNERACCESS OPTION.

Essa opção requer que você tenha acesso ao arquivo System.mdw associado ao banco de dados. É realmente útil somente nas implementações de multiusuários com segurança.

14. Funções agregadas SQL

Utilizando os SQL funções agregadas, você pode determinar várias estatísticas em conjuntos de valores. Você pode utilizar estas funções em uma consulta e em expressões agregadas na propriedade SQL de um objeto QueryDef ou ao criar um objeto Recordset baseado em uma consulta SQL.

Funções Agregadadas

Função Avg
Função Count
Funções Min, Max
Funções StDev, StDevP
Função Sum
Funções Var, VarP

15. Função Avg

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

15.1. Sintaxe

Avg(expr)

O espaço reservado expr representa uma expressão de sequência 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 (que pode ser intrínseca ou definida pelo usuário, mas nenhuma das outras funções agregadas SQL).

15.2. Comentários

A média calculada por Avg é a média aritmética (a soma dos valores dividida pelo número de valores). Você poderia utilizar o Avg, por exemplo, para calcular o custo médio do frete.

A função Avg não inclui nenhum campo Null no cálculo.

Você pode utilizar Avg em uma expressão de consulta e na propriedade SQL de um objeto QueryDef ou ao criar um objeto Recordset baseado em uma consulta SQL.

16. Função Count

Calcula o número de registros retornado por uma consulta.

16.1. Sintaxe

Count(expr)

O espaço reservado expr representa uma expressão de sequência 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 (que pode ser intrínseca ou definida pelo usuário, mas nenhuma outra função agregada SQL). Você pode contar qualquer tipo de dados, inclusive texto.

16.2. Comentários

Você pode utilizar Count para contar o número de registros em uma consulta base. Por exemplo, você poderia utilizar Count para contar o número de pedidos enviados a um determinado país.

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 (' '). O exemplo a seguir calcula o número de registros na tabela Pedidos:

```
SELECT Count(*) AS TotalDePedidos
    FROM Pedidos;
```

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 (&). O exemplo a seguir mostra como você poderia limitar a contagem aos registros nos quais DataDeEnvio ou Frete não fosse Null:

```
SELECT Count('DataDeEnvio & Frete') AS [Not Null]
    FROM Pedidos;
```

Você pode utilizar Count em uma expressão de consulta. Você também pode utilizar esta expressão na propriedade SQL de um objeto QueryDef ou durante a criação de um objeto Recordset baseado em uma consulta SQL.

17. Função Sum

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

17.1. Sintaxe

Sum(expr)

O espaço reservado expr representa uma expressão de sequência 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 (que pode ser intrínseca ou definida pelo usuário, mas nenhuma das outras funções agregadas SQL).

17.2. Comentários

A função Sum totaliza os valores em um campo. Por exemplo, você poderia utilizar a função Sum para determinar o custo total dos encargos de frete.

A função Sum ignora os registros que contenham campos Null. O exemplo a seguir mostra como você pode calcular a soma dos produtos dos campos PreçoUnitário e Quantidade:

```
SELECT Sum(PreçoUnitário * Quantidade) [AS Receita Total]
    FROM [Detalhes do Pedido];
```

Você pode utilizar a função Sum em uma expressão de consulta. Você também pode utilizar esta expressão na propriedade SQL de um objeto QueryDef ou durante a criação de um objeto Recordset, com base em uma consulta SQL.