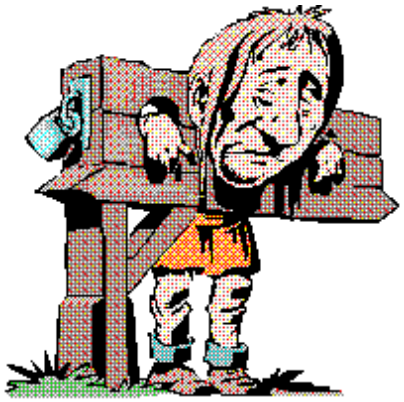


Principais Instruções em SQL

Contidas nesta apostila as principais instruções em SQL para a manutenção em Bancos de Dados.

Atenção:

Esta apostila foi desenvolvida com o **auxílio on-Line** do banco **MS-ACCESS**, O **SQL** para este banco não é totalmente compatível com o **SQL Padrão ANSI**, que é o oficial na maioria dos bancos de dados, então algumas cláusulas podem não funcionar em outros bancos.

Instrução SELECT

Instrui o programa principal do banco de dados para retornar a informação como um conjunto de registros.

Sintaxe

SELECT [predicado { * | tabela.* | [tabela.]campo1 [AS alias1] [, [tabela.]campo2 [AS alias2] [, ...]]}
FROM expressãotabela [, ...] [IN bancodedadosexterno]
[WHERE...]
[GROUP BY...]
[HAVING...]

[ORDER BY...]
[WITH OWNERACCESS OPTION]

A instrução SELECT tem as partes abaixo:

Parte	**Descrição**
predicado	Um dos seguintes predicados: ALL, DISTINCT, DISTINCTROW ou TOP. Você usa o predicado para restringir o número de registros que retornam. Se nenhum for especificado, o padrão será ALL.
*	Especifica que todos os campos da tabela ou tabelas especificadas são selecionados.
tabela	O nome da tabela que contém os campos dos quais os registros são selecionados.
campo1, campo2	Os nomes dos campos dos quais os dados serão recuperados. Se você incluir mais de um campo, eles serão recuperados na ordem listada.
alias1, alias2	Os nomes que serão usados como títulos de colunas em vez dos nomes originais das colunas na tabela.
expressãotabela	O nome da tabela ou tabelas contendo os dados que você quer recuperar.
bancodedadosexterno	O Nome do banco de dados que contém as tabelas em expressãotabela se não estiver no banco de dados atual.

Comentários

Para executar esta operação, o programa principal de banco de dados procura a tabela ou tabelas especificadas, extrai as colunas escolhidas, seleciona as linhas que satisfazem o critério e classifica ou agrupa as linhas resultantes na ordem especificada.

A instrução SELECT **não muda** os dados no banco de dados.

SELECT é normalmente a primeira palavra em uma instrução SQL. A maior parte das instruções SQL são instruções SELECT.

A sintaxe mínima da instrução SELECT é:
SELECT campos FROM tabela

Você pode usar um asterisco (*) para selecionar todos os campos na tabela. O exemplo abaixo seleciona todos os campos na tabela Funcionários:
SELECT * FROM Funcionários;

Se o nome de um campo estiver incluído em mais de uma tabela na cláusula FROM, preceda-o com o nome da tabela e o operador . (ponto). No exemplo abaixo, o campo Departamento está nas tabelas Funcionários e Supervisores. A instrução SQL seleciona Departamento da tabela Funcionários e NomeSupv da tabela Supervisores:

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

Ao criar um objeto Recordset, o programa principal de banco de dados do Jet usa o nome do campo da tabela como o nome do objeto Field no objeto Recordset. Se você quiser um nome de campo diferente ou um nome que não esteja implícito na expressão usada para gerar o campo, use a palavra reservada AS. O exemplo abaixo usa o título Nasc para nomear o objeto Field retornado no objeto Recordset resultante:

```
SELECT DataNasc AS Nasc FROM Funcionários;
```

Sempre que você usar funções aggregate ou consultas que retornem nomes de objetos Field ambíguos ou duplicados, você precisará usar a cláusula AS para fornecer um nome alternativo para o objeto Field. O exemplo abaixo usa o título Contagem para nomear o objeto Field retornado no objeto Recordset resultante:

```
SELECT COUNT(FuncionárioID) AS Contagem FROM Funcionários;
```

Você pode usar outras cláusulas na instrução SELECT para restringir e organizar posteriormente os seus dados retornados.

Cláusula GROUP BY

GROUP BY é opcional. Valores de resumo são omitidos se não houver qualquer função aggregate SQL na instrução SELECT. Os valores Null nos campos GROUP BY são agrupados e não omitidos. No entanto, os valores Null não são avaliados em qualquer função aggregate SQL. Use a cláusula WHERE para excluir linhas que você não quer agrupadas e use a cláusula HAVING para filtrar os registros após eles terem sido agrupados.

A não ser que contenha dados Memo ou OLE Object, um campo na lista de campos GROUP BY pode fazer referência a qualquer campo em qualquer tabela listada na cláusula FROM. Mesmo que o campo não esteja incluído na instrução SELECT, fornecida a instrução SELECT, inclua pelo menos uma função SQL. O programa principal de banco de dados do Jet não pode agrupar campos Memo ou OLE Objects.

Todos os campos na lista de campos SELECT devem ser incluídos na cláusula GROUP BY ou incluídos como argumentos em uma função aggregate SQL.

Cláusula HAVING

HAVING é opcional. HAVING é semelhante a WHERE, que determina quais registros são selecionados. Depois que os registros são agrupados com GROUP BY, HAVING determina quais registros são exibidos:

```
SELECT CategoriaID, Sum(UnidadesNoEstoque) FROM Produtos
GROUP BY CategoriaID
HAVING Sum(UnidadesNoEstoque) > 100 AND LIKE "BOS*";
```

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

Cláusula ORDER BY

ORDER BY é opcional. Entretanto, se você quiser exibir seus dados na ordem classificada, você deve utilizar ORDER BY. O padrão ordem de classificação é ascendente (A a Z, 0 a 9). Os dois exemplos abaixo classificam os nomes dos funcionários pelo sobrenome.

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

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

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

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

Se você especificar um campo que contém dados Memo ou OLE Objects na cláusula ORDER BY, um erro ocorrerá. O programa principal de banco de dados do Jet não classifica campos deste tipo. ORDER BY é normalmente o último item em uma instrução SQL.

Você pode incluir campos adicionais na cláusula ORDER BY. Os registros são classificados primeiro pelo primeiro campo listado depois de ORDER BY. Os registros que tiverem valores iguais naquele campo são classificados pelo valor no segundo campo listado e assim por diante.

Cláusula WITH OWNERACCESS OPTION

A declaração WITH OWNERACCESS OPTION é opcional. O exemplo abaixo habilita o usuário a ver as informações de salário (mesmo que não tenha outra permissão para ver a tabela Folha de Pagamentos) desde que o proprietário da consulta tenha tal permissão:

```
SELECT Sobrenome, Nome, Salário FROM Funcionários ORDER BY Sobrenome
WITH OWNERACCESS OPTION;
```

Se, por outro lado, um usuário for impedido de criar ou anexar a uma tabela, você poderá usar WITH OWNERACCESS OPTION para habilitá-lo a executar uma consulta construção de tabela ou consulta anexação. Se você quiser reforçar as configurações de segurança do grupo de trabalho e as permissões dos usuários, não inclua a declaração WITH OWNERACCESS OPTION. Esta opção exige que você tenha acesso ao arquivo System.mda associado ao banco de dados. É realmente útil em implementações de multiusuários seguras.

Exemplo da instrução SELECT, cláusula FROM

Esse exemplo seleciona os campos "Sobrenome" e "Nome" de todos os registros da tabela "Funcionários".

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

Esse exemplo seleciona todos os campos da tabela "Funcionários".

```
SELECT Funcionários.* FROM Funcionários;
```

Esse exemplo conta o número de registros que têm uma entrada no campo "CódigoPostal" e nomeia o campo retornado como "Tcp".

```
SELECT Count(CódigoPostal) AS Tcp FROM Clientes;
```

Esse exemplo mostra qual seria o salário se cada funcionário recebesse um aumento de 10 porcento. Não altera o valor original dos salários.

```
SELECT Sobrenome, Salário AS Atual, Salário * 1.1 AS Proposto FROM
Funcionários;
```

Esse exemplo coloca o título Nome no topo da coluna "Sobrenome". O título Salário é exibido no topo da coluna "Salário".

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

Esse exemplo mostra o número de funcionários e os salários médio e máximo.

```
SELECT Count(*) AS [Total de Funcionários], Avg(Salário) AS [Salário Médio],
Max(Salário) AS [Salário Máximo] FROM Funcionários;
```

Para cada registro, mostra Sobrenome e Salário no primeiro e último campos. A seqüência de caracteres "tem um salário de" é retornada como o campo do meio de cada registro.

```
SELECT Sobrenome, 'tem um salário de', Salário FROM Funcionários;
```

Exemplo de cláusula GROUP BY

Esse exemplo cria uma lista de nomes de departamentos únicos e o número de funcionários em cada um destes departamentos.

```
SELECT Departamento, Count([Departamento]) AS Tbc FROM Funcionários
GROUP BY Departamento;
```

Para cada título de função único, calcula o número de funcionários do departamento de Vendas que têm este título.

```
SELECT Título, Count(Título) AS Tbc FROM Funcionários
WHERE Departamento = 'Vendas' GROUP BY Título;
```

Esse exemplo calcula o número de itens em estoque para cada combinação de número e cor do item.

```
SELECT Item, Sum(Unidades) AS Tbc FROM ItensEmEstoque
GROUP BY Item, Cor;
```

Exemplo de cláusula HAVING

Esse exemplo seleciona os títulos de cargos do departamento de Produção atribuídos a mais de 50 funcionários.

```
SELECT Título, Count(Título) FROM Funcionários WHERE Departamento =
'Produção'
GROUP BY Título HAVING Count(Título) > 50;
```

Esse exemplo seleciona os departamentos que tenham mais de 100 funcionários.

```
SELECT Departamento, Count([Departamento]) FROM Funcionários
GROUP BY Departamento HAVING Count(Departamento) > 100;
```

Exemplo de cláusula ORDER BY

As instruções SQL mostradas abaixo usam a cláusula ORDER BY para classificar os registros em ordem alfabética e depois por categoria.

Esse exemplo ordena os registros pelo sobrenome, em ordem descendente (Z-A).

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

Esse exemplo ordena, primeiro, por categoria ID e depois por nome do produto.

```
SELECT CategoriaID, ProdutoNome, PreçoUnit FROM Produtos
ORDER BY CategoriaID, NomeProduto;
```

Instrução INSERT INTO

Adiciona um ou vários registros a uma tabela. Isto é referido como consulta anexação.

Sintaxe

Consulta anexação de vários registros:

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

Consulta anexação de um único registro:

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

A instrução INSERT INTO tem as partes abaixo:

Parte	**Descrição**

destino O nome da tabela ou consulta em que os registros devem ser anexados.

bancodedadosexterno O caminho para um banco de dados externo. Para uma descrição do caminho, consulte a cláusula IN.

origem O nome da tabela ou consulta de onde os dados devem ser copiados.

campo1, campo2 Os nomes dos campos aos quais os dados devem ser anexados, se estiverem após um argumento destino ou os nomes dos campos dos quais se deve obter os dados, se estiverem após um argumento origem.

expressãodetabela O nome da tabela ou tabelas das quais registros são inseridos. Este argumento pode ser um único nome de tabela ou uma combinação resultante de uma operação INNER JOIN, LEFT JOIN ou RIGHT JOIN ou de uma consulta gravada.

valor1, valor2 Os valores para inserir em 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 textos entre aspas (" ").

Comentários

Você pode usar a instrução INSERT INTO para adicionar um único registro a uma tabela usando a sintaxe de consulta anexação de um único registro como mostrado acima. Neste caso, seu código especifica o nome e o valor de cada campo do registro. Você precisa especificar cada um dos campos do registro para os quais um valor deve ser designado e um valor para este campo. Quando você não especifica cada campo, o valor padrão ou Null é inserido nas colunas omitidas. Os registros são adicionados no final da tabela.

Você também pode usar INSERT INTO para anexar um conjunto de registros de outra tabela ou consulta usando a cláusula SELECT ... FROM como é mostrado acima na sintaxe consulta anexação de vários registros. Neste caso, a cláusula SELECT especifica os campos para acrescentar à tabela destino especificada.

A tabela de origem ou de destino pode especificar uma tabela ou uma consulta. Se uma consulta for especificada, o programa principal de banco de dados do Microsoft anexa a qualquer e a todas as tabelas especificadas pela consulta.

INSERT INTO é opcional, mas quando incluída, precede a instrução SELECT.

Se sua tabela de destino contém uma chave primária, você deve acrescentar valores únicos, não Null ao campo ou campos da chave primária. Caso contrário, o programa principal de banco de dados do Jet não anexará os registros.

Se você anexar registros a uma tabela com um campo Counter e quiser numerar novamente os registros anexados, não inclua o campo Counter em sua consulta. Inclua o campo Counter na consulta se quiser manter os valores originais do campo.

Use a cláusula IN para anexar registros a uma tabela de outro banco de dados. Para achar quais registros serão anexados, antes de você executar a consulta anexação, primeiro execute e veja os resultados de uma consulta seleção que use o mesmo critério de seleção.

Uma operação de consulta anexação copia os registros de uma ou mais tabelas em outra. As tabelas que contêm os registros que você anexa não são afetadas pela operação de consulta anexação.

Em lugar de acrescentar registros existentes de 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 campo, a cláusula VALUES deve incluir um valor para cada campo na tabela; caso contrário, um erro ocorrerá em INSERT. Use uma instrução adicional INSERT INTO com uma cláusula VALUES para cada registro adicional que você quiser criar.

Exemplo de instrução INSERT INTO

Esse exemplo seleciona todos os registros de uma tabela hipotética "Novos Clientes" e os adiciona à tabela "Clientes" (quando não são designadas colunas individuais, os nomes das colunas das tabelas SELECT devem corresponder exatamente aos da tabela INSERT INTO).

```
INSERT INTO Clientes SELECT [Novos Clientes].*
FROM [Novos Clientes];
```

Esse exemplo cria um novo registro na tabela "Funcionários"

```
INSERT INTO Funcionários (Nome,Sobrenome, Título)
VALUES ("André", "Pereira", "Estagiário");
```

Esse exemplo seleciona todos os estagiários de uma tabela hipotética "Estagiários" que foram contratados há mais de 30 dias e adiciona seus registros à tabela "Funcionários".

```
INSERT INTO Funcionários SELECT Estagiários.*
FROM Estagiários WHERE DataContrato < Now() - 30;
```

Declaração UPDATE

Cria uma consulta atualização que altera os valores dos campos em uma tabela especificada com base em critérios específicos.

Sintaxe

```
UPDATE tabela
SET valornovo
WHERE critério;
```

A instrução UPDATE tem as partes abaixo:

Parte	Descrição
tabela	O nome da tabela cujos os dados você quer modificar.
valornovo	Uma expressão que determina o valor a ser inserido em um campo específico nos registros atualizados.
critério	Uma expressão que determina quais registros devem ser atualizados. Só os registros que satisfazem a expressão são atualizados.

Comentários

UPDATE é especialmente útil quando você quer alterar muitos registros ou quando os registros que você quer alterar estão em várias tabelas. Você pode alterar vários campos ao mesmo tempo. O exemplo abaixo aumenta o Valor do Pedido em 10 por cento e o valor do Frete em 3 por cento para embarques do Reino Unido:

```
UPDATE Pedidos SET ValorPedido = ValorPedido * 1.1, Frete = Frete * 1.03
WHERE PaísEmbarque = 'RU';
```

UPDATE não gera um conjunto de resultados. Se você quiser saber quais resultados serão alterados, examine primeiro os resultados da consulta seleção que use os mesmos critérios e então execute a consulta atualização.

Exemplo de instrução UPDATE

Esse exemplo muda os valores no campo "RelatórioPara" para 5 para todos os registros de funcionários que atualmente têm valores de RelatórioPara de 2.

```
UPDATE Funcionários SET RelatórioPara = 5 WHERE RelatórioPara = 2;
```

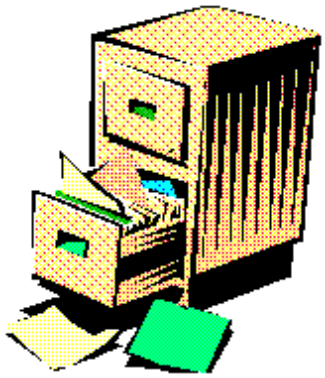
Esse exemplo aumenta o "PreçoUnit" de todos os produtos não suspensos do fornecedor 8 em 10 porcento.

```
UPDATE Produtos SET PreçoUnit = PreçoUnit * 1.1
WHERE FornecedorID = 8 AND Suspenso = No;
```

Esse exemplo reduz o PreçoUnit de todos os produtos não suspensos fornecidos pela Tokyo Traders em 5 porcento. As tabelas "Produtos" e "Fornecedores" têm uma relação um para vários.

```
UPDATE Fornecedores INNER JOIN Produtos

ON Fornecedores.FornecedorID = Produtos.FornecedorID SET PreçoUnit =
PreçoUnit * .95
WHERE NomeEmpresa = 'Tokyo Traders' AND Suspenso = No;
```



Instrução DELETE

Cria uma consulta exclusão que remove registros de uma ou mais tabelas listadas na cláusula FROM que satisfaz a cláusula WHERE.

Sintaxe

```
DELETE [tabela.*]
FROM tabela
WHERE critério
```

A instrução DELETE tem as partes abaixo:

Parte	Descrição
tabela.*	O nome opcional da tabela da qual os registros são excluídos.
tabela	O nome da tabela da qual os registros são excluídos.
critério	Uma expressão que determina qual registro deve ser excluído.

Comentários

DELETE é especialmente útil quando você quer excluir muitos registros. Para eliminar uma tabela inteira do banco de dados, você pode usar o método Execute com uma instrução DROP.

Entretanto, se você eliminar a tabela, a estrutura é perdida. Por outro lado, quando você usa DELETE, apenas os dados são excluídos. A estrutura da tabela e todas as propriedades da tabela, como atributos de campo e índices, permanecem intactos.

Você pode usar DELETE para remover registros de tabelas que estão em uma relação um por vários com outras tabelas. Operações de exclusão em cascata fazem com que os registros das tabelas que estão no lado "vários" da relação sejam excluídos quando os registros correspondentes do lado "um" da relação são excluídos na consulta. Por exemplo, nas relações entre as tabelas Clientes e Pedidos, a tabela Clientes está do lado "um" e a tabela Pedidos está no lado "vários" da relação. Excluir um registro em Clientes faz com que os registros correspondentes em Pedidos sejam excluídos se a opção de exclusão em cascata for especificada.

Uma consulta de exclusão exclui registros inteiros e não apenas dados em campos específicos. Se você quiser excluir valores de um campo específico, crie uma consulta atualização que mude os valores para Null.

Importante

Após remover os registros usando uma consulta exclusão, você não poderá desfazer a operação. Se quiser saber quais arquivos foram excluídos, primeiro examine os resultados de uma consulta seleção que use o mesmo critério e então, execute a consulta exclusão. Mantenha os backups de seus dados. Se você excluir os registros errados, poderá recuperá-los a partir dos seus backups.

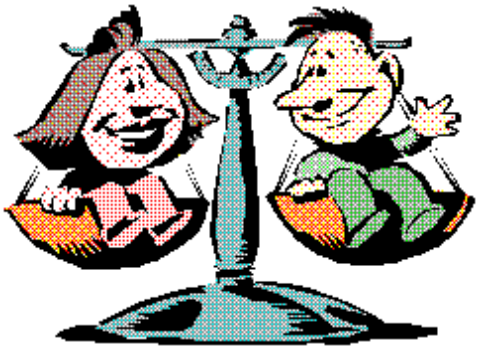
Exemplo de instrução DELETE

Esse exemplo exclui todos os registros de funcionários cujo título seja Estagiário. Quando a cláusula FROM inclui apenas uma tabela, não é necessário indicar o nome da tabela na instrução DELETE.

```
DELETE *FROM Funcionários WHERE Título = 'Estagiário';
```

Esse exemplo exclui todos os registros de funcionários cujo título seja Estagiário e que também tenham um registro na tabela "FolhadePagamento". As tabelas "Funcionários" e "FolhadePagamento" têm uma relação um por um.

```
DELETE Funcionários.* FROM Funcionários INNER JOIN FolhaDePagamento
ON Funcionários.FuncionárioID = FolhadePagamento.FuncionárioID
WHERE Funcionários.Título = 'Estagiário';
```



Subconsultas SQL

Uma subconsulta é uma instrução **SELECT** aninhada dentro de uma instrução **SELECT**, **INSERT**, **DELETE** ou **UPDATE** ou dentro 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 tem as partes abaixo:

Parte	Descrição
comparação	Uma expressão e um operador de comparação que compara a expressão com o resultado da subconsulta.
expressão	Uma expressão para a qual o resultado definido da subconsulta é procurado.

instruçãosqlt Uma instrução SELECT de acordo com as mesmas regras e formato de qualquer outra
instrução SELECT. Ela deve estar entre parênteses.

Comentários

Você pode usar uma subconsulta em vez de uma expressão na lista de campo de uma instrução SELECT ou em uma cláusula WHERE ou HAVING. Em uma subconsulta, você usa uma instrução SELECT para fornecer um conjunto de um ou mais valores específicos para avaliar as expressões das cláusulas WHERE ou HAVING.

Use o predicado ANY ou SOME, que são sinônimos, para recuperar registros na consulta principal que satisfaçam a comparação com quaisquer registros recuperados na subconsulta. O exemplo abaixo retorna todos os produtos cujo preço unitário é maior que o preço de qualquer produto vendido com um desconto de 25 por cento ou mais:

```
SELECT * FROM Produtos WHERE PreçoUnit > ANY
(SELECT PreçoUnit FROM PedidoDetalhes WHERE Desconto >= .25);
```

Use o predicado ALL para recuperar apenas os registros na consulta principal que satisfaçam a comparação com todos os registros recuperados na subconsulta. Se você mudou ANY para ALL no exemplo acima, a consulta retornaria apenas os produtos cujo preço unitário fosse maior que o de todos os produtos vendidos com um desconto de 25 por cento ou mais. Isto é muito mais restritivo.

Use o predicado IN para recuperar apenas os registros na consulta principal para os quais alguns registros na subconsulta contêm um valor igual. O exemplo abaixo retorna todos os produtos com um desconto de 25 por cento ou mais:

```
SELECT * FROM Produtos WHERE ProdutoID IN
(SELECT ProdutoID FROM PedidoDetalhes WHERE Desconto >= .25);
```

De maneira contrária, você pode usar NOT IN para recuperar apenas os registros na consulta principal para os quais não existam registros com valores iguais na subconsulta. Utilize o predicado EXISTS (com a palavra reservada NOT opcionalmente) em comparações true/false para determinar se a subconsulta retorna algum registro.

Você também pode usar aliases de nomes de tabelas em uma subconsulta para fazer referência a tabelas listadas em uma cláusula FROM fora da subconsulta. O exemplo abaixo retorna os nomes dos funcionários cujos salários sejam iguais ou superiores à média de salários de todos os funcionários na mesma função. Para a tabela Funcionários é dada o alias "T1":

```
SELECT Sobrenome, Nome, Título, Salário FROM Funcionários AS T1
WHERE Salário >= (SELECT Avg(Salário)
FROM Funcionários WHERE T1. T1.Título = Funcionários.Título) Order by Title;
```

No exemplo acima, a palavra reservada AS é opcional. Algumas subconsultas são aceitas em consultas de tabela cruzada especialmente como predicados (as da cláusula WHERE). Subconsultas como saída (as da lista SELECT) não são aceitas em tabelas de referência cruzada.

Exemplos de subconsultas SQL

Esse exemplo lista o nome, título e salário de todos os representantes de vendas cujos salários sejam superiores aos de todos os gerentes e diretores.

```
SELECT Sobrenome, Nome, Título, Salário FROM Funcionários
WHERE Título LIKE "*Repr Vendas*" AND Salário > ALL
(SELECT Salário FROM Funcionários WHERE (Título LIKE "*Gerente*")
OR (Título LIKE "*Diretor*"));
```

Esse exemplo lista o nome e preço unitário de todos os produtos cujo preço unitário seja igual ao do Licor de Cacau.

```
SELECT NomeProduto, PreçoUnit FROM Produtos
WHERE PreçoUnit = (SELECT PreçoUnit FROM [Produtos]
WHERE NomeProduto = "Licor de Cacau");
```

Esse exemplo lista a empresa e o contato de cada empresa de todos os clientes que fizeram pedidos no segundo trimestre de 1995.

```
SELECT NomeContato, NomeEmpresa, ContatoTítulo, Fone FROM Clientes
WHERE ClienteID IN (SELECT ClienteID FROM Pedidos
WHERE DataPedido BETWEEN #1/04/95# AND #1/07/95#);
```

Esse exemplo lista os funcionários cujo salário seja maior que a média dos salários de todos os funcionários.

```
SELECT Sobrenome, Nome, Título, Salário FROM Funcionários T1
WHERE Salário >= (SELECT AVG(Salário) FROM Funcionários
WHERE Funcionários.Título = T1.Título) ORDER BY Título;
```

Esse exemplo seleciona o nome de todos os funcionários que tenham registrado pelo menos um pedido. Isto também poderia ser feito com INNER JOIN.

```
SELECT Nome, Sobrenome FROM Funcionários AS E
WHERE EXISTS (SELECT * FROM Pedidos AS O
WHERE O.FuncionárioID = E.FuncionárioID);
```

Altera o campo Efetuado do arquivo de serviços para 2 caso o mesmo tenha parecer técnico da entidade encaminhanhamento diferente de nulo.

```
UPDATE servico SET efetuado = 2
WHERE numero_servico = ANY (SELECT servico.numero_servico
FROM servico INNER JOIN encaminhamento
ON (servico.numero_servico = encaminhamento. numero_servico)
AND (servico. ano_servico = encaminhamento.ano_servico)
WHERE (((servico.efetuado) Is Null) AND ((encaminhamento.parecer_tecnico) Is
Not Null))
GROUP BY servico.numero_servico ORDER BY servico.numero_servico);
```