Principais Instruções em SQL

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 [, ...]

[WHERE...]
[GROUP BY...]
[HAVING...]
[ORDER BY...]

A instrução SELECT tem as partes abaixo:

Parte	Descrição
*	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.

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 clientes :

```
SELECT  * FROM PRODUTOS
```

```
SELECT          PRODUTOS.COD_PRODUTO,
                PRODUTOS.DESCRICAO1

FROM PRODUTOS
```

Utilizando Alias

```
SELECT          PRODUTOS.COD_PRODUTO     as codigo,
                PRODUTOS.DESCRICAO1      as descricao

FROM PRODUTOS
```

Utilizando Where

Os operadores podem ser, (=, <>, >,<, >=, <=, in, not in , like, not like, between)

Operador igual (=)

```
SELECT          PRODUTOS.COD_PRODUTO    as codigo,
                PRODUTOS.DESCRICAO1     as descricao

FROM PRODUTOS

WHERE TIPO_PROD = 'MP'
```

Operador (IN)

```
SELECT          PRODUTOS.COD_PRODUTO    as codigo,
                PRODUTOS.DESCRICAO1     as descricao
FROM PRODUTOS
```

```
WHERE TIPO_PROD in ( 'MP', 'MC')
```

Operador (<>)

```
SELECT N_DOCUMENTO          AS TITULO,
                DATA_EMISSAO          AS EMISSAO,
                DATA_VENCIMENTO     AS VENCIMENTO,
                VALOR_INICIAL AS VALOR
FROM LANCAMENTOS

WHERE GERADOR <>'C'
```

Operadores (>, <, >=, <=)

```
SELECT N_DOCUMENTO          AS TITULO,
                DATA_EMISSAO          AS EMISSAO,
                DATA_VENCIMENTO     AS VENCIMENTO,
                VALOR_INICIAL AS VALOR
FROM LANCAMENTOS

WHERE VALOR_INICIAL>0
```

Operador (in, not in)

```
SELECT * FROM CORES WHERE COD_COR IN
('001','002','003','004')
```

Operador (Like, not like)

```
SELECT * FROM CLIENTES WHERE NOME LIKE '%LTDA%'
```

Operador (Between)

```
SELECT          SAIDA,
                ROMANEIO,
                DATA,
                TOTAL
FROM

SAIDAS WHERE DATA BETWEEN '01/01/08' AND '01/31/08'
```

Utilizando Join

Left Join	Mostra Todos os dados da tabela à esquerda mesmo que a da direita não exista registro
Right join	Mostra Todos os dados da tabela à esquerda mesmo que a da esquerda não exista
Inner join	Mostra os dados somente se existis nas duas tabelas relacionadas
Full join	mostra tudo das duas tabelas

(Left Join)

```
SELECT      CLIENTES.NOME,
            SAIDAS.DATA,
            SAIDAS.ROMANEIO,
            SAIDAS.VALOR_FINAL
FROM CLIENTES

LEFT JOIN SAIDAS ON SAIDAS.CLIENTE = CLIENTES.CLIENTE

ORDER BY CLIENTES.NOME
```

(INNER JOIN)

```
SELECT      CLIENTES.NOME,
            SAIDAS.DATA,
            SAIDAS.ROMANEIO,
            SAIDAS.VALOR_FINAL
FROM CLIENTES

INNER JOIN SAIDAS ON SAIDAS.CLIENTE = CLIENTES.CLIENTE

ORDER BY CLIENTES.NOME
```

(RIGHT JOIN)

```
SELECT    CLIENTES.NOME,
          SAIDAS.DATA,
          SAIDAS.ROMANEIO,
          SAIDAS.VALOR_FINAL
FROM CLIENTES

RIGHT JOIN SAIDAS ON SAIDAS.CLIENTE = CLIENTES.CLIENTE

ORDER BY CLIENTES.NOME
```

(FULL JOIN)

```
SELECT      CLIENTES.NOME,
            SAIDAS.DATA,
            SAIDAS.ROMANEIO,
            SAIDAS.VALOR_FINAL
FROM CLIENTES

FULL JOIN SAIDAS ON SAIDAS.CLIENTE = CLIENTES.CLIENTE

ORDER BY CLIENTES.NOME
```

(Utilizando o group by)

```
SELECT      CLIENTES.NOME,
            SUM(saidas.valor_final)

FROM CLIENTES

LEFT JOIN SAIDAS ON SAIDAS.CLIENTE = CLIENTES.CLIENTE

GROUP BY CLIENTES.NOME
```

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

```
SELECT
        c.nome,
        s.romaneio,
        s.data,
```

```
        p.cod_produto,
        pe.quantidade,
        sum(pe.quantidade*pe.preco)as valor  FROM saidas S

INNER JOIN EVENTOS E  ON E.EVENTO = S.evento and e.tipo_saida = 'V'
inner join PRODUTOS_EVENTOS pe on pe.tipo_operacao = 'S' and pe.cod_operacao = s.saida
inner join clientes c on c.cliente = s.cliente
inner join produtos p on p.produto = pe.produto

where s.data>='04/01/08'

group by
  c.nome,
    s.romaneio,
    s.data,
    p.cod_produto,
    pe.quantidade
```

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

```
SELECT
        c.nome,
        s.romaneio,
        s.data,
        p.cod_produto,
        pe.quantidade,
        sum(pe.quantidade*pe.preco)as valor  FROM saidas S

INNER JOIN EVENTOS E  ON E.EVENTO = S.evento and e.tipo_saida = 'V'
inner join PRODUTOS_EVENTOS pe on pe.tipo_operacao = 'S' and pe.cod_operacao = s.saida
inner join clientes c on c.cliente = s.cliente
inner join produtos p on p.produto = pe.produto

where s.data>='04/01/08'

group by
  c.nome,
```

```
   s.romaneio,
   s.data,
   p.cod_produto,
   pe.quantidade

having sum(pe.quantidade*pe.preco)>20000
```

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
   c.nome,
   s.romaneio,
   s.data,
   p.cod_produto,
   pe.quantidade,
   count(s.saida),
   sum(pe.quantidade*pe.preco)as valor  FROM saidas S

INNER JOIN EVENTOS E  ON E.EVENTO = S.evento and e.tipo_saida = 'V'
inner join PRODUTOS_EVENTOS pe on pe.tipo_operacao = 'S' and pe.cod_operacao = s.saida
inner join clientes c on c.cliente = s.cliente
inner join produtos p on p.produto = pe.produto

where s.data>='04/01/08'

group by
  c.nome,
   s.romaneio,
   s.data,
   p.cod_produto,
   pe.quantidade

having sum(pe.quantidade*pe.preco)>2000
order by s.data
```

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
   c.nome,
   s.romaneio,
   s.data,
   p.cod_produto,
   pe.quantidade,
   count(s.saida),
   sum(pe.quantidade*pe.preco)as valor  FROM saidas S

INNER JOIN EVENTOS E  ON E.EVENTO = S.evento and e.tipo_saida = 'V'
inner join PRODUTOS_EVENTOS pe on pe.tipo_operacao = 'S' and pe.cod_operacao = s.saida
inner join clientes c on c.cliente = s.cliente
inner join produtos p on p.produto = pe.produto

where s.data>='04/01/08'

group by
  c.nome,
   s.romaneio,
   s.data,
   p.cod_produto,
   pe.quantidade

having sum(pe.quantidade*pe.preco)>2000
order by s.data  desc
```

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 Pedido sem transportadora, sendo alterado por um valor predefinido.

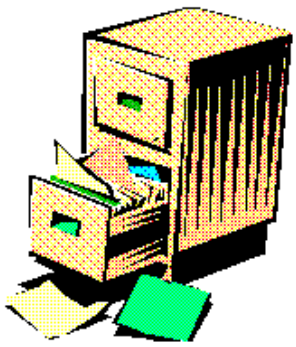
```
update pedido_venda set transportadora = 1 where cliente = 10 and transportadora is null
```

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 aumenta o "Preco" de todos os produtos de um pedido em 10%

```
update produto_pedidov set preco = preco*1.1 where pedidov = 10
```



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.

Millennium
NETWORK
SISTEMAS DE GESTÃO EMPRESARIAL

Utilitários SQL (Millennium)

Comandos prontos e relacionamentos
\\192.192.192.91\cd_Tecnicos\BANCO_DADOS\UTEIS\COMANDOS_SQL_RELACIONAMENTO

Organizador de Script.
\\192.192.192.91\cd_Tecnicos\BANCO_DADOS\INSTALADORES\SQL_INFORM

Driver ODBC
\\192.192.192.91\cd_Tecnicos\BANCO_DADOS\INSTALADORES\ODBC_DRIVER