

2

Filtragem e Ordenação de Dados

Objetivos deste Capítulo

- Ao concluir este capítulo, você poderá:
 - Restringir o resultado de uma consulta
 - Ordenar as linhas recuperadas por uma consulta
 - Usar variáveis de substituição (&) no SQL*Plus

2-2

Objetivos deste Capítulo

Numa consulta de banco de dados, normalmente é necessário:

- Restringir o resultado que deve ser exibido por uma consulta
- Especificar a ordem de exibição das linhas

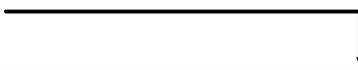
Este capítulo explica as instruções SQL usadas para executar essas ações.

Restrição de Linhas

DEPARTAMENTO

COD_DEPARTAMENTO	NOME_DEPARTAMENTO	COD_GERENTE	COD_LOCALIDADE
10	Administração	200	1700
20	Marketing	201	1800
50	Logística	124	1500
60	Informática	103	1400
80	Vendas	149	2500
90	Executivo	100	1700
110	Contabilidade	205	1700
190	Contratos		1700

"recuperar todos
os departamentos da localidade 1700"



COD_DEPARTAMENTO	NOME_DEPARTAMENTO	COD_GERENTE	COD_LOCALIDADE
10	Administração	200	1700
90	Executivo	100	1700
110	Contabilidade	205	1700
190	Contratos		1700

2-3

Restrição de Linhas

Suponha que você deseja exibir todos os departamentos da localidade 1700. O comando SELECT permite que se restrinja os dados com base em uma condição específica através da cláusula WHERE.

Restrição de Linhas

- A Cláusula WHERE é usada para restringir o resultado de um comando SQL

```
SELECT *|{[DISTINCT] coluna|expressao [apelido],...}  
FROM   tabela  
[WHERE condição(ções)];
```

- A cláusula WHERE é especificada após a cláusula FROM.

Seleção

2-4

Restrição de Linhas (continuação)

É possível restringir as linhas retornadas por uma consulta com a cláusula **WHERE**. Essa cláusula contém uma condição a ser atendida e é inserida imediatamente após a cláusula **FROM**. Se a condição for verdadeira, a linha que atender a essa condição será retornada.

Na sintaxe:

WHERE	restringe a consulta às linhas que atendem a uma condição
<i>condicao</i>	é composta de nomes de colunas, expressões, constantes e um operador de comparação

A cláusula **WHERE** pode comparar valores em colunas, valores literais, expressões aritméticas ou funções. Ela consiste em três elementos:

- Nome de coluna
- Condição de comparação
- Nome de coluna, constante ou lista de valores

A cláusula **WHERE** permite que a linguagem SQL implemente a característica da álgebra relacional chamada Seleção, onde apenas as linhas que satisfazem a condição de seleção são retornadas pelo comando.

Uso da Cláusula WHERE

```
SELECT cod_departamento, nome_departamento,  
       cod_gerente, cod_localidade  
FROM   departamento  
WHERE  cod_localidade = 1700;
```

COD_DEPARTAMENTO	NOME_DEPARTAMENTO	COD_GERENTE	COD_LOCALIDADE
10	Administração	200	1700
90	Executivo	100	1700
110	Contabilidade	205	1700
190	Contratos		1700

2-5

Uso da Cláusula WHERE

No exemplo do slide o comando `SELECT` recupera as informações dos departamentos da localidade 1700.

Literais na Cláusula WHERE

- Literais do tipo caractere e data são delimitados por aspas simples.
- Restrições por valores de caractere fazem distinção entre maiúsculas e minúsculas, e por valores de data fazem distinção de formato.
- O formato padrão de data é DD-MON-RR.

```
SELECT cod_funcionario, nome, data_admissao
FROM   funcionario
WHERE  nome = 'Roberto';
```

2-6

Literais na Cláusula WHERE

Na cláusula `WHERE`, as strings de caracteres e as datas devem ser delimitadas por aspas simples (' '), mas as constantes numéricas não.

Todas as pesquisas de caracteres fazem distinção entre maiúsculas e minúsculas.

No exemplo a seguir, não são retornadas linhas, pois a tabela `FUNCIONARIO` armazena todos os nomes dos funcionario com letras maiúsculas e minúsculas:

```
SELECT cod_funcionario, nome, data_admissao
FROM   funcionario
WHERE  nome = 'ROBERTO';
```

Os bancos de dados Oracle armazenam datas em um formato numérico interno, que representa o século, o ano, o mês, o dia, as horas, os minutos e os segundos (não armazena milissegundo). A exibição padrão de data é DD-MON-RR.

Maiores detalhes sobre o formato RR e como alterar o formato de data padrão serão vistos em um capítulo subsequente.

Operadores de Comparação

Operador	Significado
=	Igual a
>	Maior que
>=	Maior que ou igual a
<	Menor que
<=	Menor que ou igual a
<> ou != ou ^=	Diferente de
BETWEEN ...AND...	Entre dois valores (inclusivo)
IN (conjunto)	Corresponde a qualquer valor do conjunto
LIKE	Corresponde a um padrão de caractere
IS NULL	É um valor nulo
IS NAN	Não é um valor numérico
IS INFINITE	É um valor infinito

2-7

Operadores de Comparação

Com a cláusula WHERE você pode especificar condições para filtrar os dados do resultado.

Expressões contendo operadores de comparação constituem comandos que pode resultar em VERDADEIRO ou FALSO. Quando a condição de comparação resultar em VERDADEIRO então o registro será retornado.

As condições de comparação podem ser usadas também para comparar uma expressão a outra expressão. Elas são usadas na cláusula WHERE no seguinte formato:

Sintaxe:

```
... WHERE expressao operador valor
```

Exemplo:

```
... WHERE data_admissao = '01-JAN-05'  
... WHERE salario >= 6000  
... WHERE nome = 'Roberto'
```

Não é possível usar um apelido na cláusula WHERE.

Operadores de Comparação

```
SQL> SELECT nome_departamento
2 FROM departamento
3 WHERE cod_localidade <> 1700;
```

NOME_DEPARTAMENTO

Marketing
Logística
Informática
Vendas

4 linhas selecionadas.

```
SQL> SELECT nome, salario
2 FROM funcionario
3 WHERE salario >= 17000;
```

NOME SALARIO

Roberto 24000
Nair 17000
Leonardo 17000

3 linhas selecionadas.

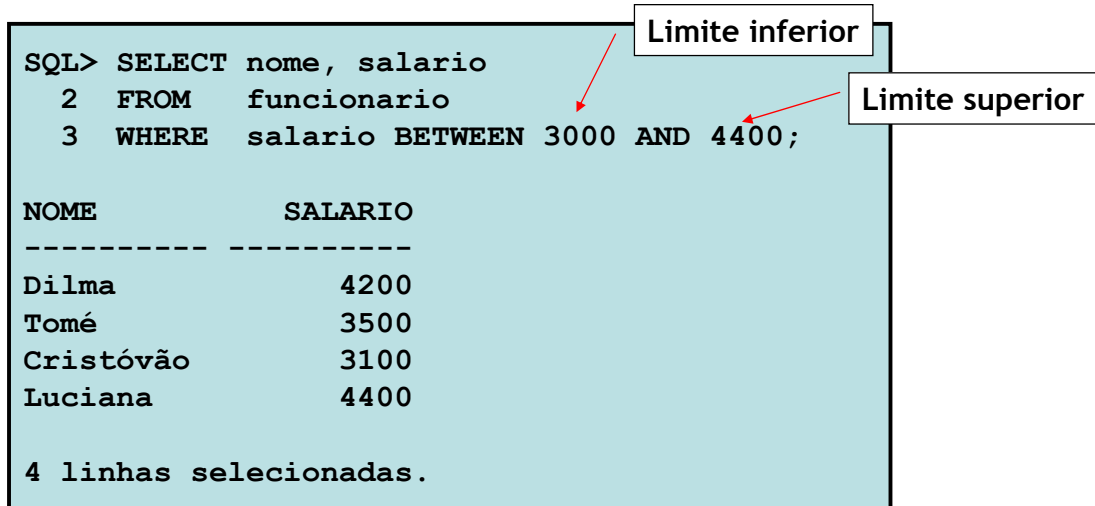
2-8

Operadores de Comparação (continuação)

O slide contém dois exemplos de condições de comparação na cláusula WHERE. No primeiro exemplo é retornado o nome de todo departamento cuja localidade seja diferente de 1700. No segundo exemplo o comando recupera o nome e o salário dos funcionários cujo salário é maior ou igual a R\$ 17.000,00. Observe que são fornecidos valores explícitos nas cláusulas WHERE. Os valores explícitos 1700 e 17000 são comparados aos valores das colunas `cod_localidade` e `salario` respectivamente. A diferença é que no primeiro exemplo serão retornadas todas as linhas cujos valores sejam diferentes da literal, e no segundo exemplo serão retornadas as linhas cujos valores são iguais ao da literal.

Operador BETWEEN

- O operador BETWEEN é usado para verificar se uma expressão está dentro de uma determinada faixa de valores:



```
SQL> SELECT nome, salario
2 FROM funcionario
3 WHERE salario BETWEEN 3000 AND 4400;
```

NOME	SALARIO
Dilma	4200
Tomé	3500
Cristóvão	3100
Luciana	4400

4 linhas selecionadas.

2-9

Operador BETWEEN

É possível exibir linhas com base em uma faixa de valores usando o operador BETWEEN. A faixa especificada contém os limites inferior e superior.

A instrução SELECT do slide retorna as linhas da tabela FUNCIONARIO relativas aos funcionários cujo salário está contido na faixa entre R\$ 3.000,00 e R\$ 4.400,00.

Os valores especificados com o operador BETWEEN são inclusivos, ou seja, os valores iguais ao limite inferior ou limite superior também serão retornados.

Especifique primeiro o limite inferior da faixa, senão a consulta não retornará nenhum registro.

Também é possível usar o operador BETWEEN em valores de caractere:

```
SELECT nome
FROM funcionario
WHERE nome BETWEEN 'Dilma' AND 'Paulo'
```

No caso do uso do BETWEEN com valores caractere, serão retornados os nomes que, ordenados, aparecem dentro da faixa especificada.

Operador IN

- O operador IN é usado para verificar se a expressão encontra-se dentro de um conjunto de valores:

```
SQL> SELECT nome, cod_cargo, cod_gerente, salario
2 FROM funcionario
3 WHERE cod_gerente IN (149,201,205);
```

NOME	COD_CARGO	COD_GERENTE	SALARIO
Pamela	VE_REP	149	11000
Elias	VE_REP	149	8600
Sadako	VE_REP	149	7000
Cosmo	MK_ANA	201	6000
Roberto	CTPUB_GER	205	8300

5 linhas selecionadas.

2-10

Operador IN

Para testar os valores de um conjunto de valores, use o operador IN. Esse operador também é conhecido como condição de associação.

O exemplo do slide exibe nomes de funcionários, código do cargo, código do gerente e salário dos funcionários cujo código do gerente seja 149, 201 ou 205.

É possível usar o operador IN com qualquer tipo de dados. O exemplo a seguir retorna informações da tabela FUNCIONARIO para cada funcionário cujo nome está incluído na lista de nomes da cláusula WHERE:

```
SELECT cod_funcionario, cod_cargo, salario
FROM funcionario
WHERE nome IN ('Pamela', 'Roberto');
```

Lembre-se que literais do tipo caractere ou data devem ser delimitados por aspas simples (' ').

Operador LIKE

- Com o operador LIKE você pode comparar cadeias de caracteres utilizando padrões de comparação.
- Use uma combinação de caracteres literais com os seguintes caracteres especiais:
 - % substitui zero, um ou vários caracteres quaisquer.
 - _ substitui um caractere qualquer.

```
SQL> SELECT nome
      2 FROM   funcionario
      3 WHERE  nome LIKE 'S%';

NOME
-----
Sheila
Sadako

2 linhas selecionadas.
```

2-11

Operador LIKE

Muitas vezes você não sabe exatamente o valor exato a ser pesquisado. Com o uso do operador LIKE é possível selecionar linhas que correspondam a um padrão de caracteres. É possível usar dois símbolos para criar um padrão de pesquisa:

Símbolo	Descrição
%	Substitui qualquer sequência de zero ou mais caracteres
_	Substitui qualquer caractere simples

O comando `SELECT` do slide retorna os nomes dos funcionários que começam com a letra A maiúsculo. Os nomes que começam com “a” (minúsculo) não são retornados.

É possível usar a condição `LIKE` como um atalho para algumas comparações `BETWEEN`. O exemplo a seguir exibe o nome e a data de admissão dos funcionários contratados 2007:

```
SQL> SELECT nome, data_admissao
      2 FROM   funcionario
      3 WHERE  data_admissao LIKE '%07';

NOME          DATA_ADMISSAO
-----
Cristóvão     29/01/2007
Cosmo         17/08/2007
```

Operador LIKE

- Você pode combinar os caracteres especiais de pesquisa com caracteres comuns

```
SQL> SELECT nome
      2 FROM   funcionario
      3 WHERE  nome LIKE '%m_';
```

```
NOME
-----
Dilma
Cosmo
Tomé
```

- Você pode usar o identificador ESCAPE quando desejar encontrar os caracteres “%” e “_”

2-12

Operador LIKE (continuação)

É possível usar os símbolos % e _ em qualquer combinação com caracteres literais. O exemplo do slide exibe os nomes dos funcionários cuja penúltima letra seja a letra “i”.

Opção ESCAPE

Quando você precisar encontrar os caracteres “%” e “_” numa pesquisa, use a opção `ESCAPE`. Essa opção especifica o que é o caractere de escape, ou seja, o caractere que antecederá o “%” ou “_” para indicar que se deve procurar pelo caractere literal.

Por exemplo, para procurar na lista de tabelas do usuário as tabelas cujo nome termina com ‘_CARGO’, você pode usar o seguinte comando:

```
SELECT   table_name
FROM     user_tables
WHERE    table_name LIKE '%\_CARGO' ESCAPE '\';
```

A opção `ESCAPE` especificada no comando identifica a barra invertida (\) como o caractere de escape. Na condição de pesquisa, o caractere de escape precede o sublinhado (_). Isso faz com que o Oracle interprete o sublinhado como um caractere literal.

Operador IS NULL

- Para encontrar registros que não possuem valor para um determinado campo use o operador IS NULL.

```
SQL> SELECT nome, cod_cargo  
2 FROM funcionario  
3 WHERE cod_gerente IS NULL;
```

NOME	COD_CARGO
Roberto	AD_PRES

2-13

Operador IS NULL

O operador `IS NULL` procura valores nulos. Um valor nulo é aquele que não está disponível nem designado e não é conhecido ou não é aplicável. Portanto, não é possível testá-lo com `=`, pois nulo não pode ser igual a um valor, e muito menos diferente de um valor qualquer. O exemplo do slide recupera os nomes e os códigos dos cargos de todos os funcionários que não estão subordinados a um gerente.

Porém, se o comando tivesse sido escrito segundo o exemplo abaixo, o resultado seria nenhum registro.

```
SQL> SELECT nome, cod_cargo  
2 FROM funcionario  
3 WHERE cod_gerente = NULL;
```

não há linhas selecionadas.

Isso ocorre porque nenhum valor é igual a `NULL`. E da mesma forma nenhum valor é diferente de `NULL`, veja:

```
SQL> SELECT nome, cod_cargo  
2 FROM funcionario  
3 WHERE cod_gerente <> NULL;
```

não há linhas selecionadas.

Operadores IS NAN e IS INFINITE

- Operadores usados para pesquisa de valores especiais NAN (Not a Number), INF (infinito positivo) e -INF (infinito negativo) dos tipos de dados BINARY_FLOAT e BINARY_DOUBLE
 - IS NAN: Verifica se a coluna possui o valor especial NAN
 - IS INFINITE: Verifica se a coluna possui o valor especial INF ou -INF
- São usados com tipos de dados numéricos

```
SQL> SELECT dado, valor
2 FROM calculo
3 WHERE valor IS NAN;
```

DADO	VALOR
INDICADOR1	Nan

```
SQL> SELECT dado, valor
2 FROM calculo
3 WHERE valor IS INFINITE;
```

DADO	VALOR
INDICADOR2	Inf
INDICADOR3	-Inf

2-14

Operadores IS NAN e IS INFINITE

Algumas aplicações que trabalham com cálculos numéricos de ponto flutuante, podem precisar representar alguns valores especiais. A partir da versão 10G do banco de dados Oracle, os tipos de dados BINARY_FLOAT e BINARY_DOUBLE suportam os valores especiais:

- NAN: Valor que indica que a expressão não é um numérico (Not a Number).
- INF (ou +INF): Valor que representa o infinito positivo.
- -INF: Valor que representa o infinito negativo.

Nesses casos, os operadores IS NAN e IS INFINITE devem ser usados em filtros que se deseja procurar esses valores especiais. O operador IS NAN testa se a coluna aplicada tem o valor 'NAN', e o operador IS INFINITE testa se o valor é 'INF' ou '-INF';

Operadores Lógicos

Operador	Significado
$x \text{ AND } y$	Retornará verdadeiro se ambas as expressões x e y forem verdadeiras
$x \text{ OR } y$	Retornará verdadeiro se uma das expressões x ou y for verdadeira
$\text{NOT } x$	Retornará verdadeiro se a expressão x for falsa e retornará falso se a expressão x for verdadeira.

2-15

Operadores Lógicos

Os operadores lógicos permitem que você limite registros baseado em combinações lógicas.

Uma combinação lógica combina o resultado de duas condições para produzir um único resultado, ou inverte o resultado de uma única condição. Só será retornada uma linha se o resultado geral da condição for verdadeiro.

Três operadores lógicos estão disponíveis em SQL:

- AND
- OR
- NOT

Todos os exemplos até então especificaram apenas uma condição na cláusula `WHERE`. Você pode usar várias condições em uma cláusula `WHERE` com os operadores `AND` e `OR`.

Operador AND

- AND exige que as duas condições sejam verdadeiras:

```
SQL> SELECT nome, cod_cargo, salario
2  FROM    funcionario
3  WHERE   cod_cargo LIKE '%GER'
4  AND     salario > 11000;
```

NOME	COD_CARGO	SALARIO
William	MK_GER	13000
Sheila	CT_GER	12000

2-16

Operador AND

No exemplo do slide as duas condições deverão ser verdadeiras para que sejam selecionados registros. Portanto, somente os funcionários que sejam gerente e que receberem mais de R\$ 11.000,00 de salário, serão retornados.

Todas as pesquisas de caracteres fazem distinção entre maiúsculas e minúsculas. Não serão retornadas linhas se a string '%GER' não estiver com as três últimas letras em maiúscula. Os caracteres literais devem ser delimitados por aspas simples.

A tabela a seguir mostra os resultados da combinação de duas expressões com AND:

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

Operador OR

- OR exige que apenas uma das condições seja verdadeira:

```
SQL> SELECT nome, cod_cargo, salario
2 FROM funcionario
3 WHERE cod_cargo LIKE '%GER'
4 OR salario > 11000;
```

NOME	COD_CARGO	SALARIO
Roberto	AD_PRES	24000
Nair	AD_VP	17000
Leonardo	AD_VP	17000
Manuela	ES_GER	5800
Paula	VE_GER	10500
William	MK_GER	13000
Sheila	CT_GER	12000
Roberto	CTPUB_GER	8300

2-17

Operador OR

No exemplo do slide uma das duas condições poderá ser verdadeira para que sejam selecionados registros. Portanto, os funcionários que forem gerentes ou que receberem mais de R\$ 11.000,00 de salário, serão retornados.

A tabela a seguir mostra os resultados da combinação de duas expressões com OR:

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

Operador NOT

```
SQL> SELECT nome, cod_cargo, salario
2 FROM funcionario
3 WHERE cod_cargo NOT LIKE '%GER';
```

NOME	COD_CARGO	SALARIO
Roberto	AD_PRES	24000
Nair	AD_VP	17000
Leonardo	AD_VP	17000
Alexandre	IT_PROG	9000
Pedro	IT_PROG	6000
Dilma	IT_PROG	4200
Tomé	ES_AUX	3500
...		
Luciana	AD_ASST	4400
Cosmo	MK_ANA	6000

15 linhas selecionadas.

2-18

Operador NOT

No exemplo do slide exibe os dados serão retornados para os funcionários que NÃO forem gerentes.

A tabela a seguir mostra o resultado da aplicação do operador NOT a uma condição:

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL

Observação: Também é possível usar o operador NOT com outros operadores SQL, como BETWEEN, NULL, IN, NAN e INFINITE.

```
... WHERE cod_cargo NOT IN ('AD_PRES', 'IT_PROG')
... WHERE salario NOT BETWEEN 5000 AND 10000
... WHERE cod_gerente IS NOT NULL
```

Cuidado: A condição NOT IN será sempre falsa, ou seja, nunca retornará nenhum registro no caso de existir um valor do tipo NULL dentro do conjunto de valores. Isso é importante porque, como pode ser usada qualquer expressão dentro do conjunto, não apenas valores literais, pode ser difícil determinar que tem um valor NULL no conjunto.

Regras de Precedência

Operador	Significado
1	Operadores aritméticos
2	Operador de concatenação
3	Operadores de comparação
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	Diferente de
7	Operador NOT
8	Operador AND
9	Operador OR

2-19

Regras de Precedência

As regras de precedência determinam a ordem de avaliação e cálculo das expressões. A tabela relaciona a ordem de precedência padrão. Você pode sobrepor a ordem padrão usando parênteses para delimitar as expressões a serem calculadas primeiro.

É uma boa prática de programação utilizar os parênteses para delimitar ordem de precedência de expressões. Isso torna o comando mais claro e fácil de ser entendido.

Regras de Precedência

SQL> SELECT nome, cod_cargo, salario
2 FROM funcionario
3 WHERE cod_cargo like '%GER'
4 OR cod_cargo = 'AD_PRES'
5 AND salario > 10000;

NOME	COD_CARGO	SALARIO
Roberto	AD_PRES	24000
Manuela	ES_GER	5800
Paula	VE_GER	10500
William	MK_GER	13000
Sheila	CT_GER	12000
Roberto	CTPUB_GER	8300

6 linhas selecionadas.

1

2

SQL> SELECT nome, cod_cargo, salario
2 FROM funcionario
3 WHERE (cod_cargo like '%GER'
4 OR cod_cargo = 'AD_PRES')
5 AND salario > 10000;

NOME	COD_CARGO	SALARIO
Roberto	AD_PRES	24000
Paula	VE_GER	10500
William	MK_GER	13000
Sheila	CT_GER	12000

2-20

Regras de Precedência (continuação)

1. Exemplo da Precedência do Operador AND

Nesse exemplo as condições são:

- A primeira condição é que o cargo seja Gerente.
- A segunda condição é que cargo seja Presidente e o salário maior que R\$ 10.000,00.

Portanto, a leitura da instrução `SELECT` será esta:

"Selecione o funcionário que for presidente e receber mais de R\$ 10.000,00 de salário, ou o funcionário que for gerente".

2. Exemplo da Utilização de Parênteses

Nesse exemplo as condições são:

- A primeira condição é que o cargo seja Gerente ou Presidente.
- A segunda condição é que o salário seja maior que R\$ 10.000,00.

Portanto, a leitura da instrução `SELECT` será esta:

"Selecione o funcionário que for presidente ou gerente e que ainda receba mais de R\$ 10.000,00 de salário".

Cláusula ORDER BY

- Ordene as linhas retornadas com a cláusula ORDER BY:
 - ASC: ordem crescente, padrão
 - DESC: ordem decrescente
- A cláusula ORDER BY deve ser inserida por último no comando SELECT:

```
SELECT  nome, cod_cargo, salario
FROM    funcionario
ORDER BY salario ;
```

2-21

Cláusula ORDER BY

Se o usuário não especificar nenhuma condição de ordenação, a ordem das linhas retornadas no resultado de uma consulta é indefinida. É possível usar a cláusula ORDER BY para ordenar as linhas. Se você usar essa cláusula, ela deverá ser a última do comando SQL. Você pode especificar uma expressão, um apelido ou uma posição de coluna como a condição de classificação.

Sintaxe:

```
SELECT expressao
FROM   tabela
[WHERE condicao(coes)]
[ORDER BY {coluna, expressao, posicao_numerica}
[ASC|DESC]];
```

onde:

ORDER BY especifica a ordem na qual as linhas recuperadas são exibidas
ASC ordena as linhas em ordem crescente (essa é a ordem padrão)
DESC ordena as linhas em ordem decrescente

Se a cláusula ORDER BY não for usada, a ordem de classificação será indefinida e o servidor Oracle poderá não extrair as linhas na mesma ordem para consultas idênticas. Use a cláusula ORDER BY para exibir as linhas em uma ordem específica.

Ordenação

- Ordenação em ordem decrescente:

```
SELECT  nome, sobrenome, data_admissao
FROM    funcionario
ORDER BY data_admissao DESC;
```

1

- Ordenação por apelido de coluna:

```
SELECT  nome, sobrenome, 13*salario salario_anual
FROM    funcionario
ORDER BY salario_anual;
```

2

- Ordenação por várias colunas:

```
SELECT  nome, sobrenome, cod_cargo, salario
FROM    funcionario
ORDER BY cod_cargo, salario DESC;
```

3

2-22

Ordenação

A ordenação padrão é crescente:

- Os valores numéricos são exibidos com os menores valores primeiro (por exemplo, 1 a 999).
- Os valores de data são exibidos em ordem cronológica (por exemplo, 01-JAN-92 antes de 01-JAN-95).
- Os valores de caractere são exibidos em ordem alfabética (por exemplo, de A a Z).
- Os valores nulos são exibidos por último em sequências crescentes e no início em sequências decrescentes.
- Você pode ordenar por uma coluna não incluída na lista `SELECT`.

Exemplos

1. Para inverter a ordem de exibição das linhas, especifique a palavra-chave `DESC` após o nome da coluna na cláusula `ORDER BY`. O exemplo do slide classifica o resultado de acordo com a data de nascimento do cliente, os clientes mais novos aparecerão primeiro.

2. Você pode usar um apelido de coluna na cláusula `ORDER BY`. O exemplo do slide classifica os funcionários pelo salário anual.

3. Você pode classificar resultados de consultas por mais de uma coluna. O limite de classificação é o número de colunas da tabela. Na cláusula `ORDER BY`, especifique as colunas e separe os nomes correspondentes por vírgulas. Para inverter a ordem de uma coluna, especifique `DESC` após seu nome.

Variáveis de Substituição



2-23

Variáveis de Substituição

As variáveis de substituição são uma ferramenta existente no SQL*Plus e não têm nada a ver com a maneira que o banco de dados processa os comandos.

Quando uma variável de substituição é utilizada em um comando, o SQL*Plus solicita o valor que deverá ser substituído pela variável antes de executar o comando. Depois que o valor da variável é digitada, o SQL*Plus reescreve o comando, substituindo literalmente a variável pelo valor digitado. Só após a reescrita, o comando é enviado ao banco para execução.

O servidor de banco de dados não tem nenhum conhecimento com relação às variáveis de substituição, apenas a ferramenta cliente SQL*Plus guarda os valores das variáveis de substituição.

Variáveis de Substituição

- Armazenar temporariamente valores com substituição de E comercial simples (&) e duplo (&&)
- Variáveis de substituição podem ser usadas para complementar:
 - Condições WHERE
 - Cláusulas ORDER BY
 - Expressões de coluna
 - Nomes de tabelas
 - Instruções SELECT inteiras

2-24

Variáveis de Substituição (continuação)

Utilize variáveis de substituição no SQL*Plus com “E” comercial único (&) para armazenar valores temporariamente.

Você pode predefinir variáveis de substituição no SQL*Plus com o comando `DEFINE`. Esse comando cria e atribui um valor a uma variável de substituição.

As variáveis de substituição podem receber atribuição de valores das seguintes maneiras:

- Entrada direta de um usuário;
- Arquivo texto de entrada;
- Outra instrução SQL.

Variáveis de Substituição com &

- Use uma variável que tenha como prefixo o “&” e o SQL*Plus solicitará um valor ao usuário em cada execução:

```
SELECT cod_funcionario, nome, cod_cargo, salario
FROM funcionario
WHERE cod_funcionario = &var_codigo_funcionario;
```

```
Informe o valor para var_codigo_funcionario: 100
antigo 3: WHERE cod_funcionario = &var_codigo_funcionario
novo 3: WHERE cod_funcionario = 100
```

COD_FUNCIONARIO	NOME	COD_CARGO	SALARIO
100	Roberto	AD_PRES	24000

2-25

Variável de Substituição com &

Para que o SQL*Plus utilize valores diferentes em cada execução de um comando SQL, pode-se utilizar as variáveis de substituição com o &. Caso as variáveis não sejam previamente definidas, no momento da execução do comando SQL, o SQL*Plus solicitará um valor para cada variável.

Notação	Descrição
<i>&variavel</i>	Indica uma variável de substituição. Se a variável não for definida previamente, o SQL*Plus solicitará um valor ao usuário. Se a variável não for predefinida, a cada execução do comando SQL o SQL*Plus solicitará novamente o valor da variável.

O exemplo do slide cria uma variável de substituição do SQL*Plus para o código do funcionário. Quando o comando é executado, o SQL*Plus solicita ao usuário que seja digitado o código do funcionário e, em seguida, exibe o resultado da consulta reescrita com o valor digitado.

Utilizando-se apenas um único &, o usuário é solicitado a especificar essa informação a cada execução do comando caso a variável não seja previamente definida.

Caractere e Data com Variáveis de Substituição

- Use aspas simples para delimitar literais do tipo data e caractere:

```
SELECT cod_funcionario, nome, cod_cargo, salario
FROM funcionario
WHERE nome = '&var_nome_funcionario';
```

```
Informe o valor para var_nome_funcionario: Nair
antigo 3: WHERE nome = '&var_nome_funcionario'
novo 3: WHERE nome = 'Nair'
```

COD_FUNCIONARIO	NOME	COD_CARGO	SALARIO
101	Nair	AD_VP	17000

2-26

Valores de caractere e Data com Variáveis de Substituição

Em uma cláusula `WHERE`, é necessário delimitar as literais do tipo data e caractere por aspas simples. A mesma regra se aplica às variáveis de substituição.

Delimite a variável por aspas simples no próprio comando SQL.

O slide mostra uma consulta para recuperar dados dos funcionários com base no valor do nome do funcionário que é especificado através de uma variável de substituição do SQL*Plus.

Nomes de Colunas, Expressões e Texto

```
SELECT    cod_funcionario, nome, sobrenome, &nome_coluna
FROM      funcionario
WHERE     &condicao
ORDER BY  &ordem;
```

```
Informe o valor para nome_coluna: salario
antigo  1: SELECT    cod_funcionario, nome, sobrenome, &nome_coluna
novo    1: SELECT    cod_funcionario, nome, sobrenome, salario
Informe o valor para condicao: salario > 11000
antigo  3: WHERE     &condicao
novo    3: WHERE     salario > 11000
Informe o valor para ordem: nome
antigo  4: ORDER BY &ordem
novo    4: ORDER BY nome
```

COD_FUNCIONARIO	NOME	SOBRENOME	SALARIO
102	Leonardo	da Silva	17000
101	Nair	Martins	17000
100	Roberto	Carlos	24000
205	Sheila	Almeida	12000
201	William	Thacker	13000

2-27

Especificando Nomes de Colunas, Expressões e Texto

Você pode usar as variáveis de substituição não apenas na cláusula `WHERE` de um comando SQL, mas também nos casos em que deseje substituir nomes de colunas, expressões ou qualquer texto.

Exemplo

O exemplo do slide exibe o código, o nome e o sobrenome dos funcionários, bem como qualquer outra coluna especificada pelo usuário durante a execução do comando, com base na tabela `FUNCIONARIO`. Para cada variável de substituição na instrução `SELECT`, um valor será solicitado.

Se você não informar um valor para a variável de substituição, receberá um erro quando executar o comando.

Uma variável de substituição pode ser usada em qualquer local no comando `SELECT`, exceto como a primeira palavra especificada no comando.

Variáveis de Substituição com &&

- Use o símbolo de E comercial duplo (&&) para reutilizar o valor da variável sem solicitar novamente um valor ao usuário.

```
SQL> SELECT    cod_funcionario, nome, sobrenome, &&nome_coluna
  2  FROM      funcionario
  3  ORDER BY  &nome_coluna;
Informe o valor para nome_coluna: salario
antigo  1: SELECT    cod_funcionario, nome, sobrenome, &&nome_coluna
novo    1: SELECT    cod_funcionario, nome, sobrenome, salario
antigo  3: ORDER BY &nome_coluna
novo    3: ORDER BY salario
```

COD_FUNCIONARIO	NOME	SOBRENOME	SALARIO
144	Pedro	Chaves	2500
143	Rafael	Miranda	2600
142	Cristóvão	Cabral	3100
141	Tomé	Lopes	3500

(...)

2-28

Variáveis de Substituição com &&

É possível usar a variável de substituição com E comercial duplo (&&) para reutilizar o valor da variável sem solicitar novamente um valor ao usuário. O valor é solicitado ao usuário apenas uma vez. No exemplo do slide, o valor da variável `nome_coluna` é solicitado apenas uma vez ao usuário. O valor fornecido por ele (`salario`) é usado para exibir e ordenar os dados.

O SQL*Plus armazena o valor fornecido com o comando `DEFINE` e reutiliza esse valor sempre que é feita referência ao nome da variável. Após a definição de uma variável de usuário, é necessário usar o comando `UNDEFINE` para remover a definição, segundo o exemplo abaixo:

```
UNDEFINE nome_coluna
```

Comando DEFINE do SQL*Plus

- Use o comando DEFINE do SQL*Plus para criar e atribuir um valor a uma variável.
- Use o comando UNDEFINE do SQL*Plus para remover uma variável.

```
SQL> DEFINE codigo_funcionario = 176
SQL> SELECT      cod_funcionario, nome, sobrenome, salario
   2  FROM        funcionario
   3  WHERE       cod_funcionario = &codigo_funcionario;
antigo  3: WHERE       cod_funcionario = &codigo_funcionario
novo    3: WHERE       cod_funcionario = 176

COD_FUNCIONARIO NOME          SOBRENOME          SALARIO
-----
              176 Elias      Voorhees              8600

SQL> UNDEFINE codigo_funcionario
```

2-29

Comando DEFINE e UNDEFINE do SQL*Plus

O exemplo mostrado cria uma variável de substituição do SQL*Plus e atribui o valor do código de um funcionário com o comando `DEFINE`. Durante a execução do comando, são exibidos o código, o nome, o sobrenome e o salário desse funcionário.

Como a variável é criada com o comando `DEFINE` do SQL*Plus, não é solicitado ao usuário um valor relativo ao código do funcionário. Nesse caso, o valor definido da variável é substituído automaticamente no comando `SELECT`.

A variável de substituição `COD_FUNCIONARIO` existirá durante toda sessão, até que o usuário remova sua definição com o comando `UNDEFINE` ou saia da sessão do SQL*Plus.

Comando VERIFY

- Use o comando VERIFY para ligar ou desligar a opção do SQL*Plus de exibir o conteúdo da linha antes e depois da substituição das variáveis de substituição.

```
SQL> SET VERIFY ON
SQL> SELECT cod_funcionario, nome, cod_cargo, salario
2 FROM funcionario
3 WHERE cod_funcionario = &var_codigo_funcionario;
```

```
antigo 3: WHERE cod_funcionario = &var_codigo_funcionario
novo 3: WHERE cod_funcionario = 149
```

COD_FUNCIONARIO	NOME	COD_CARGO	SALARIO
149	Paula	VE_GER	10500

2-30

Comando VERIFY

Para confirmar as alterações na instrução SQL, use o comando VERIFY do SQL*Plus. A definição de SET VERIFY ON força o SQL*Plus a exibir o texto da linha do comando antes e depois de substituir as variáveis por valores.

O exemplo do slide exibe os valores antigo e novo da terceira linha do comando, onde é substituída a variável pelo código do funcionário.

Variáveis de Sistema do SQL*Plus

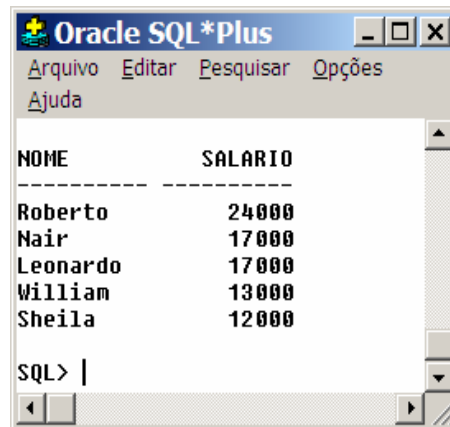
O SQL*Plus usa diversas variáveis de sistema que controlam o ambiente de trabalho. Uma delas é VERIFY. Para obter uma lista completa de todas as variáveis de sistema, execute o comando SHOW ALL.

Exercício 2

2-31


Exercício 2

1. Crie um comando que retorne os nomes e salário de todos os funcionários que ganham mais de R\$ 10.000,00 e salve o comando no arquivo texto `cap_02_01.sql`. Execute a consulta.



NOME	SALARIO
Roberto	24000
Nair	17000
Leonardo	17000
William	13000
Sheila	12000

2. Crie um relatório que exiba o nome e o código do cargo do funcionário 107.



NOME	COD_CARGO
Dilma	IT_PROG

3. Modifique `cap_02_01.sql` para exibir o nome e o salário de todos os funcionários cuja faixa salarial não esteja entre R\$ 6.000,00 e R\$ 12.000,00. Salve o comando no arquivo texto `cap_02_03.sql` e o execute.



NOME	SALARIO
Alexandre	9000
Pedro	6000
Paula	10500
Pamela	11000
Elias	8600
Sadako	7000
Cosmo	6000
Sheila	12000
Roberto	8300

9 linhas selecionadas.

Exercício 2 (continuação)

4. Crie um relatório para exibir o nome, sobrenome, código do cargo e salário dos funcionários cujos sobrenomes sejam Cabral e Martins. Ordene o relatório em ordem crescente de salários.

Oracle SQL*Plus			
Arquivo Editar Pesquisar Opções Ajuda			
NOME	SOBRENOME	COD_CARGO	SALARIO
Cristóvão	Cabral	ES_AUX	3100
Nair	Martins	AD_UP	17000

5. Exiba o nome e o código da localidade dos departamento cujo código da localidade são 1700 e 1800 em ordem alfabética crescente por nome.

NOME_DEPARTAMENTO	COD_LOCALIDADE
Administração	1700
Contabilidade	1700
Contratos	1700
Executivo	1700
Marketing	1800

6. Modifique `cap_02_03.sql` para exibir o nome, o código do cargo e o salário dos funcionários que ganham entre R\$ 3.000,00 e R\$ 7.000,00 e que possuem o código do cargo 'IT_PROG' ou 'VE_REP'. Atribua às colunas os apelidos FUNCIONARIO, FUNCAO e SALARIO MENSAL, respectivamente. Salve o comando como `cap_02_06.sql` e o execute.

FUNCIONARIO	FUNCAO	SALARIO MENSAL
Pedro	IT_PROG	6000
Dilma	IT_PROG	4200
Sadako	VE_REP	7000

Exercício 2 (continuação)

7. Crie uma consulta que exiba o nome, sobrenome e a data de admissão de todos os funcionários contratados em 2007.

NOME	SOBRENOME	DATA_ADMISSAO
Cristóvão	Cabral	29/01/2007
Cosmo	Kramer	17/08/2007

8. Crie um relatório que exiba o nome e o código do cargo de todos os funcionários não subordinados a um gerente.

NOME	COD_CARGO
Roberto	AD_PRES

9. Crie um relatório para exibir o nome, o salário e a comissão de todos os funcionários que ganham comissão. Ordene os dados em ordem decrescente de salário e comissão.

NOME	SALARIO	COMISSAO
Paula	10500	,2
Pamela	11000	,3
Elias	8600	,2
Sadako	7000	,15

10. Crie um relatório que exiba o nome e o salário dos funcionários que ganham mais do que uma quantia especificada pelo usuário durante a execução da consulta. Salve essa consulta no arquivo `cap_02_10.sql`. Se você informar 12000 quando a quantia for solicitada, o relatório exibirá estes resultados:

NOME	SALARIO
Roberto	24000
Nair	17000
Leonardo	17000
William	13000

Exercício 2 (continuação)

11. Crie uma consulta que solicite código de gerente ao usuário e retorne o código do funcionário, o nome e o salário dos funcionários desse gerente. Além disso também deve ser solicitado ao usuário o nome de uma coluna que será utilizada para ordenar os dados.

Você pode testar os dados com os seguintes valores:

Código do gerente = 103, ordenando pelo nome do funcionário:

COD_FUNCIONARIO	NOME	SALARIO
107	Dilma	4200
104	Pedro	6000

Código do gerente = 100, ordenando pelo salário do funcionário:

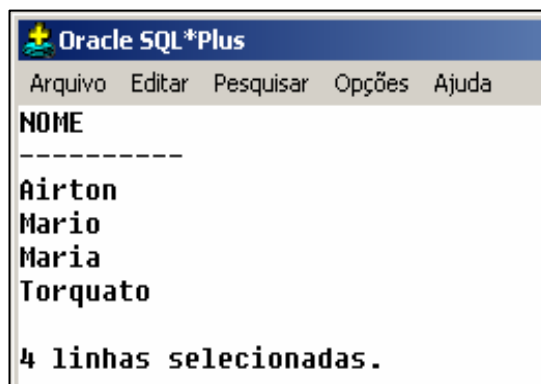
COD_FUNCIONARIO	NOME	SALARIO
124	Manuela	5800
149	Paula	10500
201	William	13000
102	Leonardo	17000
101	Nair	17000

Código do gerente = 149, ordenando pelo código do funcionário:

COD_FUNCIONARIO	NOME	SALARIO
174	Pamela	11000
176	Elias	8600
178	Sadako	7000

Exercício 2 (continuação)

12. Exiba todos os nomes dos funcionários cuja terceira letra do nome seja *r*.



Oracle SQL*Plus

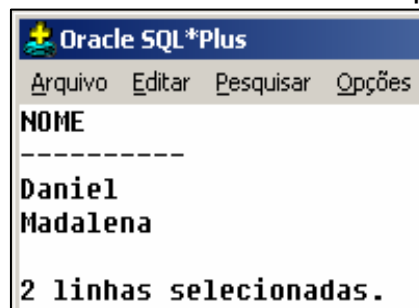
Arquivo Editar Pesquisar Opções Ajuda

NOME

Airton
Mario
Maria
Torquato

4 linhas selecionadas.

13. Exiba o nome de todos os funcionários que contenham *a* e *e*.



Oracle SQL*Plus

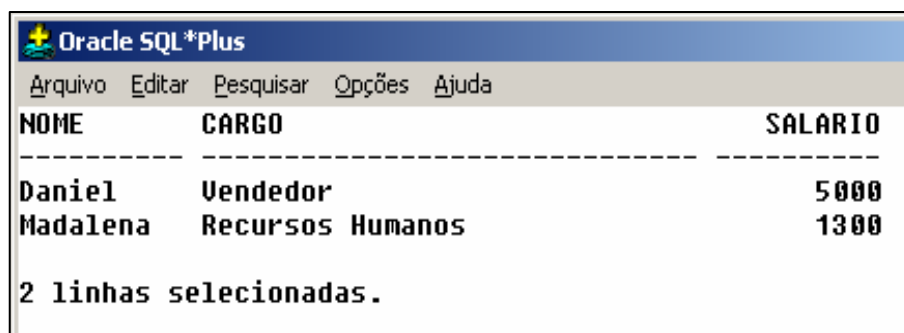
Arquivo Editar Pesquisar Opções

NOME

Daniel
Madalena

2 linhas selecionadas.

14. Exiba o nome, o cargo e o salário de todos os funcionários cujo cargo seja Vendedor ou Recursos Humanos e cujo salário seja diferente de R\$ 1.200,00, R\$ 1.500,00 ou R\$ 2.000,00.



Oracle SQL*Plus

Arquivo Editar Pesquisar Opções Ajuda

NOME	CARGO	SALARIO
-----	-----	-----
Daniel	Vendedor	5000
Madalena	Recursos Humanos	1300

2 linhas selecionadas.