





# Restringindo e **Classificando** **Dados**

## Objetivos

Ao concluir esta lição, você será capaz de:

- Limitar as linhas recuperadas por uma consulta
- Classificar as linhas recuperadas por uma consulta

1-3

### Objetivos

Durante a recuperação de dados do banco de dados, talvez seja necessário:

Restringir as linhas de dados a serem exibidas

Especificar a ordem de exibição das linhas

Esta lição explica as instruções SQL usadas para executar essas ações.

## Limitando Linhas por Seleção

### EMPLOYEES

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90
103	Hunold	IT_PROG	60
104	Ernst	IT_PROG	60
107	Lorentz	IT_PROG	60
124	Mourgos	ST_MAN	50

20 rows selected.

"recuperar todos  
os funcionários do  
departamento 90"

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90

1-4

### Limitando Linhas por Seleção

No exemplo do slide, suponha que você queira exibir todos os funcionários do departamento 90. As linhas com o valor 90 na coluna `DEPARTMENT_ID` são as únicas retornadas. Esse método de restrição é a base da cláusula `WHERE` em SQL.

## Limitando as Linhas Seleccionadas

- Restrinja as linhas retornadas com a cláusula **WHERE**:

```
SELECT * | {[DISTINCT] column|expression [alias],...}
FROM table
[WHERE condition(s)];
```

- A cláusula **WHERE** é especificada após a cláusula **FROM**.

1-5

### Limitando as Linhas Seleccionadas

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:

<b>WHERE</b>	restringe a consulta às linhas que atendem a
uma condição	
<i>condition</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 functions. Ela consiste em três elementos:

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

## Usando a Cláusula WHERE

```
SELECT employee_id, last_name, job_id, department_id
FROM employees
WHERE department_id = 90 ;
```

## Strings de Caracteres e Datas

- As strings de caracteres e os valores de data são delimitados por apóstrofo.
- Os valores de caractere fazem distinção entre maiúsculas e minúsculas, e os valores de data fazem distinção de formato.
- O formato default de data no BD FIAP é DD-MM-RR.

```
SELECT last_name, job_id, department_id
FROM employees
WHERE last_name = 'Whalen' ;
```

1-6

### Usando a Cláusula WHERE

No exemplo, a instrução `SELECT` recupera o ID, o nome, o ID do cargo e o número do departamento de todos os funcionários do departamento 90.

### Strings de Caracteres e Datas

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

Todas as pesquisas de caracteres fazem distinção entre maiúsculas e minúsculas. No exemplo a seguir, não são retornadas linhas, pois a tabela `EMPLOYEES` armazena todos os sobrenomes em maiúsculas e minúsculas:

```
SELECT last_name, job_id, department_id
FROM employees
WHERE last_name = 'WHALEN';
```

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. A exibição default de data é DD-MON-RR.

O banco de dados da FIAP está com o seguinte padrão: DD/MM/RR.

## Condições de Comparação

Operador	Significado
=	Igual a
>	Maior que
>=	Maior que ou igual a
<	Menor que
<=	Menor que ou igual a
<>	Diferente de
BETWEEN ...AND...	Entre dois valores (inclusivo)
IN (set)	Corresponde a qualquer
LIKE	Corresponde a um padrão de
IS NULL	É um valor nulo

1-7

### Condições de Comparação

As condições de comparação são usadas em condições que comparam uma expressão a outro valor ou expressão. Elas são usadas na cláusula `WHERE` no seguinte formato:

#### Sintaxe

```
... WHERE expr operator value
```

#### Exemplo

```
... WHERE hire_date='01-JAN-95'
... WHERE salary >=6000
... WHERE last_name='Smith'
```

Não é possível usar um apelido na cláusula `WHERE`.

**Observação:** Os símbolos `!=` e `^=` também podem representar a condição *diferente de*.

## Usando Condições de Comparação

```
SELECT last_name, salary
FROM employees
WHERE salary <= 3000 ;
```

Use a condição **BETWEEN** para exibir linhas com base em uma faixa de valores:

```
SELECT last_name, salary
FROM employees
WHERE salary BETWEEN 2500 AND 3500 ;
```

↑                      ↑  
Limite inferior    Limite superior

1-8

### Usando Condições de Comparação

No exemplo, a instrução **SELECT** recupera, com base na tabela **EMPLOYEES**, o sobrenome e o salário de todos os funcionários cujo salário é menor que US\$ 3.000 ou igual a esse valor. Observe que foi fornecido um valor explícito na cláusula **WHERE**. O valor explícito **3000** é comparado ao valor do salário na coluna **SALARY** da tabela **EMPLOYEES**.

### Usando a Condição **BETWEEN**

É possível exibir linhas com base em uma faixa de valores usando a condição de faixa **BETWEEN**. A faixa especificada contém os limites inferior e superior. A instrução **SELECT** do slide retorna as linhas da tabela **EMPLOYEES** relativas aos funcionários cujo salário está contido na faixa entre US\$ 2.500 e US\$ 3.500.

Os valores especificados com a condição **BETWEEN** são inclusivos.

Especifique o limite inferior primeiro.

Também é possível usar a condição **BETWEEN** em valores de caractere:

```
SELECT last_name
FROM employees
WHERE last_name BETWEEN 'King' AND 'Smith';
```



## Usando a Condição IN

Use a condição de associação **IN** para testar os valores de uma lista:

```
SELECT employee_id, last_name, salary, manager_id
FROM employees
WHERE manager_id IN (100, 101, 201) ;
```

## Usando a Condição LIKE

- Use a condição **LIKE** para executar pesquisas com curinga de valores válidos de strings de pesquisa.
- As condições de pesquisa podem conter números ou caracteres literais:
  - % indica zero ou vários caracteres.
  - \_ indica um caractere.

```
SELECT first_name
FROM employees
WHERE first_name LIKE 'S%';
```

1-9

### Usando a Condição IN

Para testar os valores de um conjunto de valores especificado, use a condição **IN**. Essa condição também é conhecida como *condição de associação*.

O exemplo do slide exibe números de funcionários, sobrenomes, salários e números de funcionário de gerentes relativos a todos os funcionários cujo número de funcionário do gerente seja 100, 101 ou 201.

É possível usar a condição **IN** com qualquer tipo de dados. O exemplo a seguir retorna uma linha da tabela **EMPLOYEES** para cada funcionário cujo sobrenome está incluído na lista de nomes da cláusula **WHERE**:

```
SELECT employee_id, manager_id, department_id
FROM employees
WHERE last_name IN ('Hartstein', 'Vargas');
```

Se forem usados caracteres ou datas na lista, eles deverão ser delimitados por apóstrofo ( ' ' ).

# Usando a Condição LIKE

FIAP

• • • • •  
• • • • •  
• + •  
+ •  
  
|  
+

•  
• +  
■ □ •  
• •  
•  
• + • •  
• • •

1-10

## Usando a Condição LIKE

Nem sempre você sabe o valor exato a ser pesquisado. É possível selecionar linhas que correspondam a um padrão de caracteres usando a condição `LIKE`. A operação de correspondência a um padrão de caracteres é conhecida como pesquisa com *curinga*. É possível usar dois símbolos para criar a string de pesquisa.

Símbolo	Descrição
%	A instrução <code>SELECT</code> do slide retorna, com base na tabela <code>EMPLOYEES</code> , os nomes de todos os funcionários que começam com a letra <code>S</code> . Observe o <code>S</code> maiúsculo. Os nomes que começam com <code>s</code> não são retornados.
—	Representa qualquer caractere simples.

É possível usar a condição `LIKE` como um atalho para algumas comparações `BETWEEN`. O exemplo a seguir exibe os sobrenomes e as datas de admissão de todos os funcionários admitidos entre janeiro de 1995 e dezembro de 1995:

```
SELECT last_name, hire_date
FROM employees
WHERE hire_date LIKE '%95';
```

## Usando a Condição LIKE

- Pode ser utilizada combinação de caracteres com padrões correspondentes:

```
SELECT last_name  
FROM employees  
WHERE last_name LIKE 'o%';
```

1-11

### Combinando Caracteres Curinga

É possível usar os símbolos % e \_ em qualquer combinação com caracteres literais. O exemplo do slide exibe os nomes de todos os funcionários cujo sobrenome inclui a letra o como o segundo caractere.

## Usando a Condição IS NULL

- Teste valores nulos com o operador IS NULL.

```
SELECT last_name, manager_id
FROM employees
WHERE manager_id IS NULL;
```

1-12

### Usando as Condições NULL

As condições NULL incluem IS NULL e IS NOT NULL.

A condição IS NULL testa valores nulos. Um valor nulo é aquele que não está disponível nem designado e não é conhecido ou aplicável. Portanto, não é possível testá-lo com =, pois não pode ser igual a um valor ou diferente dele. O exemplo do slide recupera os sobrenomes e os gerentes de todos os funcionários que não estão subordinados a um gerente.

Veja outro exemplo: Para exibir o sobrenome, o ID do cargo e a comissão de todos os funcionários *sem* direito a comissão, use a seguinte instrução SQL:

```
SELECT last_name, job_id, commission_pct
FROM employees
WHERE commission_pct IS NULL;
```

EMPLOYEE_ID	LAST_NAME	JOB_ID
149	Zlotkey	SA_MAN
174	Abel	SA_REP
176	Taylor	SA_REP
178	Grant	SA_REP

## Condições Lógicas

Operador	Significado
AND	Retornará TRUE se as duas condições componentes forem verdadeiras
OR	Retornará TRUE se uma das condições componentes for verdadeira
NOT	Retornará TRUE se a condição seguinte for falsa

1-13

### Condições Lógicas

Uma condição lógica combina o resultado de duas condições componentes para produzir um único resultado com base nessas condições, 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`.

## Usando o Operador AND

AND exige que as duas condições sejam verdadeiras:

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary >= 10000
AND job_id LIKE '%MAN%';
```

## Usando o Operador OR

OR exige que uma das condições seja verdadeira:

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary >= 10000
OR job_id LIKE '%MAN%';
```

1-14

### Usando o Operador AND

No exemplo, as duas condições deverão ser verdadeiras para que sejam selecionados registros. Portanto, somente os funcionários cujos cargos contiverem a string 'MAN' e que receberem US\$ 10.000 ou mais serão selecionados.

Todas as pesquisas de caracteres fazem distinção entre maiúsculas e minúsculas. Não serão retornadas linhas se a string 'MAN' não estiver em maiúsculas. As strings de caracteres devem ser delimitadas por aspas.

#### Tabela de Valores Verdadeiros com AND

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

</

## Usando o Operador NOT

```
SELECT last_name, job_id
FROM employees
WHERE job_id
      NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP') ;
```

1-16

### Usando o Operador NOT

O exemplo do slide exibe o sobrenome e o ID do cargo de todos os funcionários cujo ID de cargo é *diferente* de IT\_PROG, ST\_CLERK ou SA\_REP.

#### Tabela de Valores Verdadeiros com NOT

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

NOT	TRUE	FALSE	NULL
SQL, como BETWEEN, LIKE e NULL	FALSE	TRUE	NULL

```
... WHERE job_id NOT IN ('AC_ACCOUNT', 'AD_VP')
... WHERE salary NOT BETWEEN 10000 AND 15000
... WHERE last_name NOT LIKE '%A%'
... WHERE commission_pct IS NOT NULL
```



## Regras de Precedência

Operado	Significado
1	Operadores aritméticos
2	Operador de concatenação
3	Condições de comparação
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	Diferente de <> ou !=
7	Condição lógica NOT
8	Condição lógica AND
9	Condição lógica OR

Você pode usar parênteses para sobrepor as regras de precedência.

1-17

### Regras de Precedência

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

## Regras de Precedência

```
SELECT last_name, job_id, salary
FROM employees
WHERE job_id = 'SA_REP'
OR job_id = 'AD_PRES'
AND salary > 15000;
```

1

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000
Abel	SA_REP	11000
Taylor	SA_REP	8600
Grant	SA_REP	7000

```
SELECT last_name, job_id, salary
FROM employees
WHERE (job_id = 'SA_REP'
OR job_id = 'AD_PRES')
AND salary > 15000;
```

2

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000

1-18

### 1. Exemplo da Precedência do Operador AND

Neste exemplo, existem duas condições:

A primeira condição é que o ID do cargo seja AD\_PRES e o salário seja maior que US\$ 15.000.

A segunda condição é que o ID do cargo seja SA\_REP.

Portanto, a leitura da instrução SELECT será esta:

"Selecione a linha se o funcionário for presidente e receber mais de US\$15.000, ou se ele for representante de vendas".

### 2. Exemplo da Utilização de Parênteses

Neste exemplo, existem duas condições:

A primeira condição é que o ID do cargo seja AD\_PRES ou SA\_REP.

A segunda condição é que o salário seja maior que US\$ 15.000.

Portanto, a leitura da instrução SELECT será esta:

"Selecione a linha se o funcionário for presidente ou representante de vendas e receber mais de US\$15.000".

## Usando a Cláusula ORDER BY

- **Classifique as linhas recuperadas com a cláusula ORDER BY:**
  - ASC: ordem crescente, default
  - DESC: ordem decrescente
- **A cláusula ORDER BY é inserida por último na instrução SELECT:**

```
SELECT last_name, job_id, department_id, hire_date
FROM employees
ORDER BY hire_date
```

1-19

### Usando a Cláusula ORDER BY

A ordem das linhas retornadas no resultado de uma consulta é indefinida. É possível usar a cláusula ORDER BY para classificar as linhas. Se você usar essa cláusula, ela deverá ser a última da instrução SQL. Você pode especificar uma expressão, um apelido ou uma posição de coluna como a condição de classificação.

#### Sintaxe

```
SELECT          expr
FROM            table
[WHERE          condition(s)]
[ORDER BY {column, expr, numeric_position}
[ASC|DESC]];
```

Na sintaxe:

ORDER BY	especifica a ordem na qual as linhas recuperadas são exibidas
ASC	ordena as linhas em ordem crescente (essa é a ordem default)
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 (fetch) as linhas na mesma ordem para consultas idênticas. Use a cláusula ORDER BY para exibir as linhas em uma ordem específica.

## Classificação

- **Classificação em ordem decrescente:**

```
SELECT last_name, job_id, department_id, hire_date
FROM employees
ORDER BY hire_date DESC;
```

1

- **Classificação por apelido de coluna:**

```
SELECT employee_id, last_name, salary*12 annsal
FROM employees
ORDER BY annsal ;
```

2

- **Classificação por várias colunas:**

```
SELECT last_name, department_id, salary
FROM employees
ORDER BY department_id, salary DESC;
```

3

- **Classificação posicional**

```
SELECT last_name, department_id, salary
FROM employees
ORDER BY 2, 3 DESC;
```

4

1-20

### Ordenação de Dados Default

A ordem de classificação default 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 seqüências crescentes e no início em seqüências decrescentes.

Você pode classificar 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 o funcionário admitido mais recentemente.
2. Você pode usar um apelido de coluna na cláusula `ORDER BY`. O exemplo do slide classifica os dados por 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.

4. Na cláusula `ORDER BY`, especifique posição da coluna que deve ser ordenada, de acordo com a sua posição na lista de seleção da instrução `SELECT`

OBRIGADO



profalexandre.barcelos@fiap.com.br



<https://www.linkedin.com/in/alexandrebarcelos>

FIAP

Copyright © 2023 | Professor Me. Alexandre Barcelos  
Todos os direitos reservados. Reprodução ou divulgação total ou parcial deste documento, é expressamente  
proibido sem consentimento formal, por escrito, do professor/autor.