



# ORACLE

## Academy



# Database Programming with SQL

6-2

Cláusulas de Junção

**ORACLE**  
Academy



# Objetivos

- Esta lição abrange os seguintes objetivos:
  - Construir e executar uma junção com a cláusula USING do ANSI-99
  - Construir e executar uma junção com a cláusula ON do ANSI-99
  - Construir e executar uma consulta do ANSI-99 que junte três tabelas

# Finalidade

- À medida que acrescenta mais comandos ao seu vocabulário de banco de dados, você será mais capaz de criar consultas que retornem o resultado desejado
- O propósito de uma junção é unir os dados, entre tabelas, sem repetir todos os dados de cada uma delas
- Por que solicitar mais dados do que você realmente precisa?

# Cláusula USING

- Em uma junção natural, se as tabelas tiverem colunas com o mesmo nome, mas tipos de dados diferentes, a junção causará um erro
- Para evitar essa situação, a cláusula de junção pode ser modificada com a cláusula USING
- A cláusula USING especifica as colunas que devem ser usadas para a junção

# Cláusula USING

- A consulta mostrada é um exemplo da cláusula USING
- As colunas referenciadas na cláusula USING não devem ter um qualificador (nome de tabela ou alias) em qualquer lugar da instrução SQL

```
SELECT first_name, last_name, department_id, department_name  
FROM employees JOIN departments USING (department_id);
```

| FIRST_NAME | LAST_NAME | DEPARTMENT_ID | DEPARTMENT_NAME |
|------------|-----------|---------------|-----------------|
| Jennifer   | Whalen    | 10            | Administration  |
| Michael    | Hartstein | 20            | Marketing       |
| Pat        | Fay       | 20            | Marketing       |
| ...        | ...       | ...           | ...             |

# Cláusula USING

- A cláusula USING permite usar WHERE para restringir as linhas de uma ou ambas as tabelas:

```
SELECT first_name, last_name, department_id, department_name
FROM employees JOIN departments USING (department_id)
WHERE last_name = 'Higgins';
```

| FIRST_NAME | LAST_NAME | DEPARTMENT_ID | DEPARTMENT_NAME |
|------------|-----------|---------------|-----------------|
| Shelley    | Higgins   | 110           | Accounting      |

# Cláusula ON

- E se as colunas a serem unidas tiverem nomes diferentes ou se a junção usar operadores de comparação de não igualdade, como  $<$ ,  $>$  ou BETWEEN?
- Não podemos usar USING. Em vez disso, usamos uma cláusula ON
- Isso permite especificar uma variedade maior de condições de junção
- A cláusula ON também permite usar WHERE para restringir as linhas de uma ou ambas as tabelas



# Exemplo da Cláusula ON

- Neste exemplo, a cláusula ON é usada para juntar a tabela de funcionários com a tabela de cargos

```
SELECT last_name, job_title  
FROM employees e JOIN jobs j  
ON (e.job_id = j.job_id);
```

- Uma cláusula ON de junção é necessária quando as colunas em comum têm nomes diferentes nas duas tabelas

| LAST_NAME | JOB_TITLE                        |
|-----------|----------------------------------|
| King      | President                        |
| Kochhar   | Administration<br>Vice President |
| De Haan   | Administration<br>Vice President |
| Whalen    | Administration<br>Assistant      |
| Higgins   | Accounting<br>Manager            |
| Gietz     | Public<br>Accountant             |
| Zlotkey   | Sales Manager                    |
| Abel      | Sales<br>Representative          |
| Taylor    | Sales<br>Representative          |
| ...       |                                  |

# Exemplo da Cláusula ON

- Ao usar uma cláusula ON em colunas com o mesmo nome em ambas as tabelas, você precisa adicionar um qualificador (o nome da tabela ou o alias), caso contrário, um erro será retornado. O exemplo acima usa aliases de tabela como qualificadores (e.job\_id = j.job\_id), mas os nomes das tabelas também poderiam ter sido gravados (employees.job\_id = jobs.job\_id)

```
SELECT last_name, job_title  
FROM employees e JOIN jobs j  
ON (e.job_id = j.job_id);
```

| LAST_NAME | JOB_TITLE                        |
|-----------|----------------------------------|
| King      | President                        |
| Kochhar   | Administration<br>Vice President |
| De Haan   | Administration<br>Vice President |
| Whalen    | Administration<br>Assistant      |
| Higgins   | Accounting<br>Manager            |
| Gietz     | Public<br>Accountant             |
| Zlotkey   | Sales Manager                    |
| Abel      | Sales<br>Representative          |
| Taylor    | Sales<br>Representative          |
| ...       |                                  |

# Cláusula ON com a Cláusula WHERE

- Eis a mesma consulta com uma cláusula WHERE para restringir as linhas selecionadas

```
SELECT last_name, job_title  
FROM employees e JOIN jobs j  
    ON (e.job_id = j.job_id)  
WHERE last_name LIKE 'H%';
```

| LAST_NAME | JOB_TITLE          |
|-----------|--------------------|
| Higgins   | Accounting Manager |
| Hunold    | Programmer         |
| Hartstein | Marketing Manager  |

# Cláusula ON com operador de não igualdade

- Às vezes, você pode precisar recuperar dados de uma tabela que não tenha uma coluna correspondente em outra tabela
- Vamos supor que queremos saber a classificação do salário de cada funcionário
- A tabela job\_grades não tem uma coluna em comum com a tabela de funcionários
- O uso de uma cláusula ON nos permite unir as duas tabelas

tabela job\_grades

| GRADE_LEVEL | LOWEST_SAL | HIGHEST_SAL |
|-------------|------------|-------------|
| A           | 1000       | 2999        |
| B           | 3000       | 5999        |
| C           | 6000       | 9999        |
| D           | 10000      | 14999       |
| E           | 15000      | 24999       |
| F           | 25000      | 40000       |

# Cláusula ON com operador de não igualdade

```
SELECT last_name, salary, grade_level, lowest_sal,  
highest_sal  
FROM employees JOIN job_grades  
    ON(salary BETWEEN lowest_sal AND highest_sal);
```

| LAST_NAME | SALARY | GRADE_LEVEL | LOWEST_SAL | HIGHEST_SAL |
|-----------|--------|-------------|------------|-------------|
| Vargas    | 2500   | A           | 1000       | 2999        |
| Matos     | 2600   | A           | 1000       | 2999        |
| Davies    | 3100   | B           | 3000       | 5999        |
| Rajs      | 3500   | B           | 3000       | 5999        |
| Lorentz   | 4200   | B           | 3000       | 5999        |
| Whalen    | 4400   | B           | 3000       | 5999        |
| Mourgos   | 5800   | B           | 3000       | 5999        |
| Fay       | 6000   | C           | 6000       | 9999        |
| ...       |        |             |            |             |

# Juntando Três Tabelas

- Tanto USING quanto ON podem ser usados para juntar três ou mais tabelas
- Vamos supor que precisamos de um relatório sobre os nossos funcionários, o departamento deles e a cidade onde o departamento está localizado
- Precisamos juntar três tabelas: as de funcionários, departamentos e locais



# Exemplo de Junção de Três Tabelas

```
SELECT last_name, department_name AS "Department", city
FROM employees JOIN departments USING (department_id)
      JOIN locations USING (location_id);
```



| LAST_NAME | Departemen | CITY                |
|-----------|------------|---------------------|
| Hartstein | Marketing  | Toronto             |
| Fay       | Marketing  | Toronto             |
| Zlotkey   | Sales      | Oxford              |
| Abel      | Sales      | Oxford              |
| Taylor    | Sales      | Oxford              |
| Hunold    | IT         | Southlake           |
| Ernst     | IT         | Southlake           |
| Lorentz   | IT         | Southlake           |
| Mourgos   | Shipping   | South San Francisco |
| ...       |            |                     |

# Terminologia

- Estes são os principais termos usados nesta lição:
  - Cláusula ON
  - Cláusula USING



# Resumo

- Nesta lição, você deverá ter aprendido a:
  - Construir e executar uma junção com a cláusula USING do ANSI-99
  - Construir e executar uma junção com a cláusula ON do ANSI-99
  - Construir e executar uma consulta do ANSI-99 que junte três tabelas



# ORACLE

## Academy

