



# ORACLE

## Academy



# Database Foundations

6-9

Unindo Tabelas com JOIN

**ORACLE**  
Academy



# Roteiro

Introdução ao  
Oracle  
Application  
Express (APEX)

Structured  
Query  
Language  
(SQL)

Data  
Definition  
Language  
(DDL)

Data  
Manipulation  
Language  
(DML)

Transaction  
Control  
Language  
(TCL)

Recuperand  
o Dados com  
SELECT

Restringindo  
Dados com  
WHERE

Classificando  
Dados com  
ORDER BY

Unindo Tabelas  
com JOIN

Parte 6

# Objetivos

- Esta lição abrange os seguintes objetivos:
  - Criar instruções SELECT para acessar dados de mais de uma tabela usando equijunções e não-equijunções
  - Usar uma autojunção para unir uma tabela a ela mesma
  - Usar junções externas (OUTER) para exibir dados que geralmente não atendem a uma condição de junção
  - Gerar um produto cartesiano (junção cruzada) de todas as linhas de duas ou mais tabelas



# Obtendo Dados de Várias Tabelas

- Às vezes, é necessário usar dados de mais de uma tabela
- Para produzir o relatório, é necessário vincular as tabelas EMPLOYEES e DEPARTMENTS e acessar os dados das duas:

**EMPLOYEES**

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
149	Zlotkey	80
103	Ernst	60

...

**DEPARTMENTS**

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
60	IT	1400
80	Sales	2500
90	Executive	1700

...



EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
100	90	Executive
149	80	Sales
102	60	IT

# Tipos de Junções

- Junções que estão em conformidade com o padrão SQL:1999:
  - Junção natural com a cláusula NATURAL JOIN
  - Junção com a Cláusula USING
  - Junção com a Cláusula ON
  - Junções OUTER:
    - LEFT OUTER JOIN
    - RIGHT OUTER JOIN
    - FULL OUTER JOIN
  - CROSS JOIN

# Unindo Tabelas com a Sintaxe SQL:1999

- Use uma junção para consultar dados de mais de uma tabela:

```
SELECT    table1.column, table2.column
FROM      table1
[NATURAL JOIN table2] |
[JOIN table2 USING (column_name)] |
[JOIN table2
    ON (table1.column_name = table2.column_name)] |
[LEFT|RIGHT|FULL OUTER JOIN table2
    ON (table1.column_name = table2.column_name)] |
[CROSS JOIN table2];
```



# Qualificando Nomes de Colunas Ambíguos

- Use prefixos de tabela para qualificar nomes de colunas que estão em várias tabelas, evitando ambiguidade
- Use prefixos de tabela para aumentar a velocidade da análise da instrução
- Em vez de prefixos de nomes de tabela completos, use aliases de tabelas
- Um alias de tabela dá à tabela um nome mais curto, mantém o código SQL menor e usa menos memória





# Qualificando Nomes de Colunas Ambíguos

- Use aliases de tabelas para distinguir colunas que tenham nomes idênticos, mas que residam em tabelas diferentes

```
SELECT  e.first_name, d.department_name, d.manager_id
FROM    employees e JOIN departments d
USING   (department_id);
```

- Observação: consulte nas anotações do slide as diretrizes sobre aliases de tabelas

FIRST_NAME	DEPARTMENT_NAME	MANAGER_ID
Jennifer	Administration	200
Michael	Marketing	201

# Criando Junções Naturais.

- A cláusula NATURAL JOIN baseia-se em todas as colunas das duas tabelas que têm o mesmo nome e o mesmo tipo de dados
- Ela seleciona as linhas das duas tabelas que têm valores iguais em todas as colunas correspondentes
- Se as colunas com os mesmos nomes tiverem tipos de dados diferentes, um erro será retornado

# Recuperando Registros com Junções Naturais

- Usa o único campo que é comum às duas tabelas (DEPARTMENT\_ID) para fazer a junção

```
SELECT    department_id, department_name, location_id, city
FROM      departments NATURAL JOIN locations;
```

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
20	Marketing	1800	Toronto
80	Sales	2500	Oxford
60	IT	1400	Southlake
50	Shipping	1500	South San Francisco
10	Administration	1700	Seattle
90	Executive	1700	Seattle

# Criando Junções com a Cláusula USING

- Se várias colunas forem compartilhadas pelas tabelas que estão sendo unidas, todos os campos comuns serão usados na junção
- Use a cláusula USING para especificar uma única coluna para JOIN, em vez de uma NATURAL JOIN
- A cláusula USING também pode ser usada para fazer a correspondência das colunas que têm o mesmo nome, mas tipos de dados diferentes
- As cláusulas NATURAL JOIN e USING são mutuamente exclusivas

# Unindo Nomes de Colunas

- Os valores da coluna DEPARTMENT\_ID devem ser iguais em ambas as tabelas. Isso é denominado equijunção (também chamada junção simples ou interna)

**EMPLOYEES**

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
200	Whalen	10
205	Higgins	110
206	Gietz	110
149	Zlotkey	80
124	Mourgos	50

...

**DEPARTMENTS**

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
50	Shipping	1500
60	IT	1400
80	Sales	2500
90	Executive	1700
110	Accounting	1700

...

**Chave estrangeira**

**Chave primária**

# Recuperando Registros com a Cláusula USING

- A cláusula USING especifica que a junção seja realizada com a coluna DEPARTMENT\_ID, não com a coluna MANAGER\_ID, que também é uma coluna comum

```
SELECT  employee_id, last_name, location_id,  
        department_id  
FROM    employees JOIN departments  
USING   (department_id);
```

EMPLOYEE_ID	LAST_NAME	LOCATION_ID	DEPARTMENT_ID
200	Whalen	1700	10
201	Hartstein	1800	20
202	Fay	1800	20
124	Mourgos	1500	50

...

# Usando Aliases de Tabelas com a Cláusula USING

- Não use um nome nem um alias de tabela na cláusula USING
- Se a mesma coluna for usada em outro lugar na instrução SQL, não atribua um alias a ela

```
SELECT  l.city, d.department_name  
FROM    locations l JOIN departments d  
USING   (location_id)  
WHERE   d.location_id = 1400;
```



ORA-25154: column part of USING clause cannot have qualifier



# Criando Junções com a Cláusula ON

- Uma NATURAL JOIN cria uma equijunção de todas as colunas com o mesmo nome e tipo de dados
- Use a cláusula ON para especificar condições arbitrárias ou especificar colunas a serem unidas
- A condição da junção é separada de outras condições de pesquisa

# Criando Junções com a Cláusula ON

- A cláusula ON facilita o entendimento do código
- Uma cláusula USING cria uma equijunção entre duas tabelas usando uma coluna com o mesmo nome, independentemente do tipo de dados
- Uma cláusula ON cria uma equijunção entre duas tabelas usando uma coluna de cada tabela, independentemente do nome ou do tipo de dados

# Recuperando Registros com a Cláusula ON

- Você também pode usar a cláusula ON para unir colunas que tenham nomes ou tipos de dados diferentes

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON     (e.department_id = d.department_id);
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
200	Whalen	10	10	1700
201	Hartstein	20	20	1800
202	Fay	20	20	1800
124	Mourgos	50	50	1500
141	Rajs	50	50	1500
142	Davies	50	50	1500



# Criando Junções Tridirecionais com a Cláusula ON

- Ao unir três tabelas, deverão existir duas instruções de junção, conforme mostrado:

```
SELECT  employee_id, city, department_name
FROM    employees e JOIN departments d
ON      d.department_id = e.department_id
JOIN    locations l
ON      d.location_id = l.location_id;
```

EMPLOYEE_ID	CITY	DEPARTMENT_NAME
201	Toronto	Marketing
202	Toronto	Marketing
149	Oxford	Sales
174	Oxford	Sales
176	Oxford	Sales

# Cenário de Caso: Cláusula ON



Recuperando  
dados de  
três tabelas

```
SELECT  b.title as "BOOK TITLE",  
        a.name as "AUTHOR",  
        t.id as "BOOK TRANSACTION"  
FROM    authors a JOIN books b  
ON      a.id = b.author_id  
JOIN    book_transactions t  
ON      b.id = t.book_id;
```

Recuperação bem-sucedida dos  
dados usando a cláusula ON

BOOK_TITLE	AUTHOR	BOOK_TRANSACTION
The Clicking of Cuthbert	P.G. Wodehouse	0D0002
War and Peace	Leo Tolstoy	0D0001
An Unsocial Socialist	George Bernard Shaw	0D0003

# Aplicando Condições Adicionais a uma Junção

- Use a cláusula AND ou WHERE para aplicar condições adicionais:

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON     (e.department_id = d.department_id)  
AND    e.manager_id = 149 ;
```

Ou

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON     (e.department_id = d.department_id)  
WHERE  e.manager_id = 149 ;
```

# Exercício 1 do Projeto

- DFo\_6\_9\_1\_Project
  - Banco de Dados da Oracle Baseball League
  - Criar Instruções SELECT com Dados de Várias Tabelas Usando Equijunções e Não-equijunções
  - Junções Naturais, Cláusula USING e ON, Junções Tridirecionais





# Unindo uma Tabela a Ela Mesma

**EMPLOYEES (WORKER)**

EMPLOYEE_ID	LAST_NAME	MANAGER_ID
100	King	-
101	Kochhar	100
102	De Haan	100
200	Whalen	101
205	Higgins	101
206	Gietz	205
149	Zlotkey	100
174	Abel	149
176	Taylor	149
201	Hartstein	100
202	Fay	201

...

**EMPLOYEES (MANAGER)**

EMPLOYEE_ID	LAST_NAME
100	King
101	Kochhar
102	De Haan
200	Whalen
205	Higgins
206	Gietz
149	Zlotkey
174	Abel
176	Taylor
201	Hartstein
202	Fay

...



**MANAGER\_ID na tabela WORKER é igual a  
EMPLOYEE\_ID na tabela MANAGER.**

# Autojunções com a Cláusula ON

- A cláusula ON também pode ser usada para unir colunas que tenham nomes diferentes, dentro da mesma tabela ou em outra

```
SELECT worker.last_name emp, manager.last_name mgr
FROM   employees worker JOIN employees manager
ON     (worker.manager_id = manager.employee_id);
```

EMP	MGR
Kochhar	King
De Haan	King
Zlotkey	King
Mourgos	King
Hartstein	King

...



# Não-equijunções

- A tabela JOB\_GRADES define o intervalo de valores LOWEST\_SAL e HIGHEST\_SAL para cada GRADE\_LEVEL

```
SELECT *  
FROM 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

# Não-equijunções

- Dessa forma, a coluna `GRADE_LEVEL` pode ser usada para atribuir níveis salariais a cada funcionário com base no respectivo salário

**EMPLOYEES**

EMPLOYEE_ID	SALARY
100	24000
101	17000
102	17000
200	4400
205	12000
206	8300
149	10500
174	11000
176	8600
178	7000

**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





# Recuperando Registros com Não-equijunções

- Este exemplo cria uma não-equijunção para avaliar o nível salarial de um funcionário. O salário deve estar entre qualquer par de intervalos entre os salários mínimo e máximo

```
SELECT e.last_name, e.salary, j.grade_level  
FROM employees e JOIN job_grades j  
ON e.salary BETWEEN j.lowest_sal AND j.highest_sal;
```

LAST_NAME	SALARY	GRADE_LEVEL
Vargas	2500	A
Matos	2600	A
Davies	3100	B
Rajs	3500	B
Lorentz	4200	B
Whalen	4400	B

# Retornando Registros sem Correspondência Direta Usando Junções OUTER

## DEPARTMENTS

DEPARTMENT_NAME	DEPARTMENT_ID
Administration	10
Marketing	20
Shipping	50
IT	60
Sales	80
Executive	90
Accounting	110
Contracting	190

Não há funcionários no departamento 190

## Equijunção com EMPLOYEES

DEPARTMENT_ID	LAST_NAME
90	King
90	Kochhar
90	De Haan
10	Whalen
80	Taylor
-	Grant
50	Mourgos
20	Fay

...

Não foi atribuído ao funcionário "Grant" um ID de departamento

# Junções INNER Versus OUTER

- No SQL:1999, a junção de duas tabelas que retorna apenas as linhas correspondentes é denominada junção INNER (interna). (Cláusulas NATURAL JOIN, USING, ON)
- Uma junção entre duas tabelas que retorna os resultados da junção INNER, bem como as linhas sem correspondência da tabela esquerda (ou direita) é denominada junção OUTER (externa) esquerda ou direita
- Uma junção entre duas tabelas que retorna os resultados de uma junção INNER, bem como os resultados de junções OUTER esquerdas e direitas é uma junção OUTER completa



# LEFT OUTER JOIN

- Aqui desejamos ver todos os registros de funcionários (tabela esquerda), mesmo que eles não estejam atribuídos a um departamento

```
SELECT e.last_name, e.department_id, d.department_name
FROM employees e LEFT OUTER JOIN departments d
ON (e.department_id = d.department_id);
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Fay	20	Marketing
Hartstein	20	Marketing
Vargas	50	Shipping
Matos	50	Shipping
Higgins	110	Accounting
Grant	-	-

# RIGHT OUTER JOIN

- Aqui desejamos ver todos os registros de departamentos (tabela direita), mesmo que eles não tenham funcionários

```
SELECT e.last_name, e.department_id, d.department_name
FROM employees e RIGHT OUTER JOIN departments d
ON (e.department_id = d.department_id);
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Hartstein	20	Marketing
Fay	20	Marketing
Mourgos	50	Shipping
Rajs	50	Shipping
-	-	<b>Contracting</b>

...

# FULL OUTER JOIN

- Aqui desejamos ver todos os registros de funcionários e todos os registros de departamentos

```
SELECT e.last_name, e.department_id, d.department_name
FROM employees e FULL OUTER JOIN departments d
ON (e.department_id = d.department_id);
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
King	90	Executive
Kochhar	90	Executive
Taylor	80	Sales
<b>Grant</b>	-	-
Mourgos	50	Shipping
Fay	20	Marketing
-	-	<b>Contracting</b>

...

# Produtos Cartesianos

- Um produto Cartesiano ocorre quando todas as combinações de linhas são exibidas. Todas as linhas da primeira tabela são unidas a todas as linhas da segunda

```
SELECT last_name, department_name  
FROM   employees, departments;
```

- Um produto Cartesiano será gerado quando uma condição de junção for omitida ou inválida
- Sempre inclua uma condição de junção válida se quiser evitar um produto Cartesiano

```
SELECT last_name, department_name  
FROM   employees e, departments d  
WHERE  e.department_id = d.department_id;
```

# Gerando um Produto Cartesiano

EMPLOYEES (40 linhas)

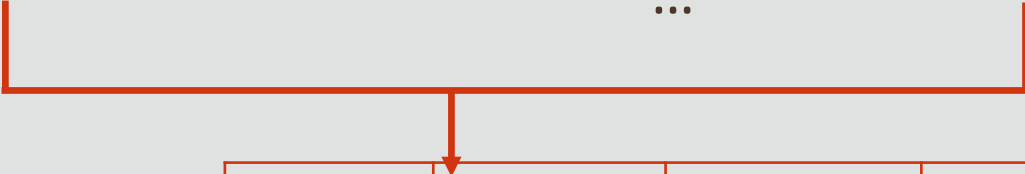
EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
149	Zlotkey	80
103	Ernst	60

...

DEPARTMENTS (9 linhas)

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
60	IT	1400
80	Sales	2500
90	Executive	1700

...



EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
100	King	90	Administration	1700
101	Kochhar	90	Administration	1700
102	De Haan	90	Administration	1700
200	Whalen	10	Administration	1700
205	Higgins	110	Administration	1700
206	Gietz	110	Administration	1700
149	Zlotkey	80	Administration	1700

**Produto Cartesiano:**  
**40 x 9 = 360 linhas**

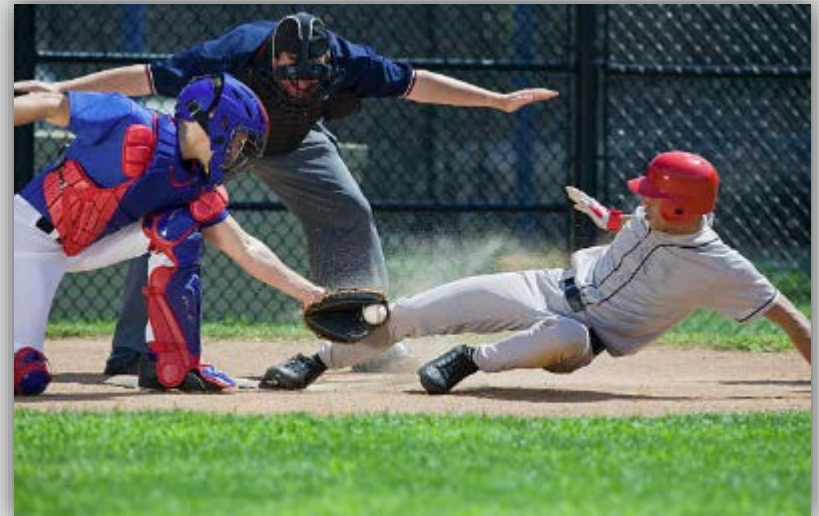
# Criando Junções Cruzadas

- A cláusula CROSS JOIN cria o produto cruzado de duas tabelas
- Isso também é denominado produto Cartesiano entre as duas tabelas

```
SELECT last_name, department_name  
FROM employees  
CROSS JOIN departments ;
```

# Exercício 2 do Projeto

- DFo\_6\_9\_2\_Project
  - Banco de Dados da Oracle Baseball League
  - Criar Instruções SELECT com Dados de Várias Tabelas Usando Equijunções e Não-equijunções
  - Autojunções, JUNÇÕES OUTER, Produtos Cartesianos





# Resumo

- Nesta lição, você deverá ter aprendido a:
  - Criar instruções SELECT para acessar dados de mais de uma tabela usando equijunções e não-equijunções
  - Usar uma autojunção para unir uma tabela a ela mesma
  - Usar junções externas (OUTER) para exibir dados que geralmente não atendem a uma condição de junção
  - Gerar um produto Cartesiano (junção cruzada) de todas as linhas de duas ou mais tabelas





# ORACLE

## Academy

