

**Exibindo Dados de Várias Tabelas**

# Objetivos

Após concluir esta lição, você poderá fazer o seguinte:

- Criar instruções SELECT para obter acesso a dados de mais de uma tabela usando equijunções e não equijunções
- Exibir dados que, em geral, não correspondem a uma condição de junção usando junções externas
- Unir uma tabela a ela mesma por meio de uma autojunção

# Agenda

- **Tipos de JOINS e sua sintaxe**
- Natural join:
  - USING clausula
  - ON clausula
- Self-join
- Nonequijoins
- OUTER join:
  - LEFT OUTER join
  - RIGHT OUTER join
  - FULL OUTER join
- Produto Cartesiano
  - Cross join

# Obtendo Dados de Várias Tabelas

## EMPLOYEES

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	100	King	90
2	101	Kochhar	90
3	102	De Haan	90
...			
18	202	Fay	20
19	205	Higgins	110
20	206	Gietz	110

## DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10	Administration	1700
2	20	Marketing	1800
3	50	Shipping	1500
4	60	IT	1400
5	80	Sales	2500
6	90	Executive	1700
7	110	Accounting	1700
8	190	Contracting	1700

	EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
1	200	10	Administration
2	201	20	Marketing
3	202	20	Marketing
4	124	50	Shipping
5	144	50	Shipping
...			
18	205	110	Accounting
19	206	110	Accounting

# Tipos de Joins

Joins são compatíveis com o padrão SQL:1999 e podem ser:

- Natural joins:
  - NATURAL JOIN **clausula**
  - USING **clausula**
  - ON **clausula**
- OUTER joins:
  - LEFT OUTER JOIN
  - RIGHT OUTER JOIN
  - FULL OUTER JOIN
- Cross joins

# Juntando tabelas usando a sintaxe SQL:1999

Use a instrução **join** para listar os 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.
- Melhore o desempenho usando prefixos de tabela.
- Diferencie colunas que possuem nomes idênticos, mas que residam em tabelas diferentes usando apelidos de coluna.

# Agenda

- Tipos de JOINS e sua sintaxe
- Natural join:
  - USING clausula
  - ON clausula
- Self-join
- Nonequijoins
- OUTER join:
  - LEFT OUTER join
  - RIGHT OUTER join
  - FULL OUTER join
- Produto Cartesiano
  - Cross join



# Criando Junções Naturais

- A cláusula NATURAL JOIN baseia-se em todas as colunas com o mesmo nome nas duas tabelas.
- Ela seleciona 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, será retornado um erro.

# Recuperando Registros com Junções Naturais

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

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
1	60	IT	1400	Southlake
2	50	Shipping	1500	South San Francisco
3	10	Administration	1700	Seattle
4	90	Executive	1700	Seattle
5	110	Accounting	1700	Seattle
6	190	Contracting	1700	Seattle
7	20	Marketing	1800	Toronto
8	80	Sales	2500	Oxford

# Criando Junções com a Cláusula USING

- Se várias colunas tiverem os mesmos nomes mas se os tipos de dados não forem correspondentes, a cláusula NATURAL JOIN poderá ser modificada com a cláusula USING para especificar as colunas que devem ser usadas em uma equijunção.
- Usar a cláusula USING para estabelecer a correspondência com apenas uma coluna quando mais de uma coluna for correspondente.
- Não usar um apelido ou nome de tabela nas colunas às quais foram feitas referências.
- As cláusulas NATURAL JOIN e USING são mutuamente excludentes.

# Recuperando Registros com a Cláusula USING

**EMPLOYEES**

EMPLOYEE_ID	DEPARTMENT_ID
100	90
101	90
102	90
103	60
104	60
107	60
124	50
141	50
142	50
143	50
144	50
149	80
174	80
176	80

...

**DEPARTMENTS**

	DEPARTMENT_ID	DEPARTMENT_NAME
1	10	Administration
2	20	Marketing
3	50	Shipping
4	60	IT
5	80	Sales
6	90	Executive
7	110	Accounting
8	190	Contracting

Primary key

Foreign key

# Recuperando Registros com a Cláusula USING

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

	<small>R2</small> EMPLOYEE_ID	<small>R2</small> LAST_NAME	<small>R2</small> LOCATION_ID	<small>R2</small> DEPARTMENT_ID
1	200	Whalen	1700	10
2	201	Hartstein	1800	20
3	202	Fay	1800	20
4	124	Mourgos	1500	50
5	144	Vargas	1500	50
6	143	Matos	1500	50
7	142	Davies	1500	50
8	141	Rajs	1500	50
9	107	Lorentz	1400	60
10	104	Ernst	1400	60

...

19	205	Higgins	1700	110
----	-----	---------	------	-----

# Criando Junções com a Cláusula ON

- A condição da junção natural é basicamente uma equijunção de todas as colunas com o mesmo nome.
- Para especificar condições arbitrárias ou colunas a serem unidas, é usada a cláusula ON.
- A condição de junção é separada de outras condições de pesquisa.
- A cláusula ON facilita a compreensão do código.

# Recuperando Registros com a Cláusula ON



```
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_1	LOCATION_ID
1	200	Whalen	10	10	1700
2	201	Hartstein	20	20	1800
3	202	Fay	20	20	1800
4	124	Mourgos	50	50	1500
5	144	Vargas	50	50	1500
6	143	Matos	50	50	1500
7	142	Davies	50	50	1500
8	141	Rajs	50	50	1500
9	107	Lorentz	60	60	1400
10	104	Ernst	60	60	1400

...

# Criando Junções Triplas com a Cláusula ON

```
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
1	100	Seattle	Executive
2	101	Seattle	Executive
3	102	Seattle	Executive
4	103	Southlake	IT
5	104	Southlake	IT
6	107	Southlake	IT
7	124	South San Francisco	Shipping
8	141	South San Francisco	Shipping

...



# Condições adicionais em um JOIN

Use a cláusula `AND` ou a cláusula `WHERE` para aplicar condições adicionais ao JOIN:

```
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 ;
```

# Agenda

- Tipos de JOINS e sua sintaxe
- Natural join:
  - USING clausula
  - ON clausula
- Self-join
- Nonequijoins
- OUTER join:
  - LEFT OUTER join
  - RIGHT OUTER join
  - FULL OUTER join
- Produto Cartesiano
  - Cross join

# Auto-Junção

**EMPLOYEES (WORKER)**

	EMPLOYEE_ID	LAST_NAME	MANAGER_ID
1	100	King	(null)
2	101	Kochhar	100
3	102	De Haan	100
4	103	Hunold	102
5	104	Ernst	103
6	107	Lorentz	103
7	124	Mourgos	100
8	141	Rajs	124
9	142	Davies	124
10	143	Matos	124

...

**EMPLOYEES (MANAGER)**

EMPLOYEE_ID	LAST_NAME
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst
107	Lorentz
124	Mourgos
141	Rajs
142	Davies
143	Matos

...

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

# Auto-Junção

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

	EMP	MGR
1	Hunold	De Haan
2	Fay	Hartstein
3	Gietz	Higgins
4	Lorentz	Hunold
5	Ernst	Hunold
6	Zlotkey	King
7	Mourgos	King
8	Kochhar	King
9	Hartstein	King
10	De Haan	King

...

# Agenda

- Tipos de JOINS e sua sintaxe
- Natural join:
  - USING clausula
  - ON clausula
- Self-join
- **Nonequijoins**
- OUTER join:
  - LEFT OUTER join
  - RIGHT OUTER join
  - FULL OUTER join
- Produto Cartesiano
  - Cross join

# Nonequijoins

**EMPLOYEES**

R#	LAST_NAME	R#	SALARY
1	King		24000
2	Kochhar		17000
3	De Haan		17000
4	Hunold		9000
5	Ernst		6000
6	Lorentz		4200
7	Mourgos		5800
8	Rajs		3500
9	Davies		3100
10	Matos		2600
...			
19	Higgins		12000
20	Gietz		8300

**JOB\_GRADES**

R#	GRADE_LEVEL	R#	LOWEST_SAL	R#	HIGHEST_SAL
1	A		1000		2999
2	B		3000		5999
3	C		6000		9999
4	D		10000		14999
5	E		15000		24999
6	F		25000		40000

A tabela **JOB\_GRADES** define o intervalo de valores **LOWEST\_SAL** e **HIGHEST\_SAL** para cada **GRADE\_LEVEL**. Portanto, a coluna **GRADE\_LEVEL** pode ser usada para atribuir a faixa salarial de cada funcionário.

# Obtendo dados com Nonequijoins

```
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;
```

	RZ	LAST_NAME	RZ	SALARY	RZ	GRADE_LEVEL
1		Vargas		2500		A
2		Matos		2600		A
3		Davies		3100		B
4		Rajs		3500		B
5		Lorentz		4200		B
6		Whalen		4400		B
7		Mourgos		5800		B
8		Ernst		6000		C
9		Fay		6000		C
10		Grant		7000		C

...

# Agenda

- Tipos de JOINS e sua sintaxe
- Natural join:
  - USING clausula
  - ON clausula
- Self-join
- Nonequijoins
- OUTER join:
  - LEFT OUTER join
  - RIGHT OUTER join
  - FULL OUTER join
- Produto Cartesiano
  - Cross join



# OUTER Joins

## DEPARTMENTS

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

Não há empregados no departamento 190.

O empregado "Grant" ainda não está alocado em um departamento.

## Equijoin with EMPLOYEES

	DEPARTMENT_ID	LAST_NAME
1	90	King
2	90	Kochhar
3	90	De Haan
4	60	Hunold
5	60	Ernst
6	60	Lorentz
7	50	Mourgos
8	50	Rajs
9	50	Davies
10	50	Matos

...

18	110	Higgins
19	110	Gietz

# INNER Versus OUTER Joins

- Na sintaxe SQL: 1999, a junção de duas tabelas que retorna apenas linhas correspondentes é uma junção interna.
- Uma junção entre duas tabelas que retorna os resultados da junção interna assim como linhas não correspondentes em tabelas esquerdas (ou direitas) é uma junção externa esquerda (ou direita).
- Uma junção entre duas tabelas que retorna os resultados de uma junção interna assim como os resultados de uma junção esquerda ou direita é uma junção externa completa.

# LEFT OUTER JOIN

```
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
1	Whalen	10	Administration
2	Fay	20	Marketing
3	Hartstein	20	Marketing
4	Vargas	50	Shipping
5	Matos	50	Shipping

...

17	King	90	Executive
18	Gietz	110	Accounting
19	Higgins	110	Accounting
20	Grant	(null)	(null)

# RIGHT OUTER JOIN

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

	A Z	LAST_NAME	A Z	DEPARTMENT_ID	A Z	DEPARTMENT_NAME
1		Whalen		10		Administration
2		Hartstein		20		Marketing
3		Fay		20		Marketing
4		Mourgos		50		Shipping

...

18		Gietz		110		Accounting
19		Higgins		110		Accounting
20		(null)		190		Contracting

# FULL OUTER JOIN

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

	A2	LAST_NAME	A2	DEPARTMENT_ID	A2	DEPARTMENT_NAME
1		King		90		Executive
2		Kochhar		90		Executive
3		De Haan		90		Executive
4		Hunold		60		IT

...

15		Grant		(null)		(null)
16		Whalen		10		Administration
17		Hartstein		20		Marketing
18		Fay		20		Marketing
19		Higgins		110		Accounting
20		Gietz		110		Accounting
21		(null)		190		Contracting

# Agenda

- Tipos de JOINS e sua sintaxe
- Natural join:
  - USING clausula
  - ON clausula
- Self-join
- Nonequijoin
- OUTER join:
  - LEFT OUTER join
  - RIGHT OUTER join
  - FULL OUTER join
- Produto Cartesiano
  - Cross join

# Produto Cartesianos

- Um produto cartesiano será formado quando:
  - – Uma condição de junção for omitida.
  - – Uma condição de junção for inválida.
  - – Todas as linhas da primeira tabela forem unidas a todas as linhas da segunda tabela.
- Para evitar um produto Cartesiano, sempre inclua uma condição de junção válida em uma cláusula WHERE.

# Gerando um Produto Cartesiano

**EMPLOYEES (20 linhas)**

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	100	King	90
2	101	Kochhar	90
3	102	De Haan	90
4	103	Hunold	60

...

19	205	Higgins	110
20	206	Gietz	110

**DEPARTMENTS (8 linhas)**

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10	Administration	1700
2	20	Marketing	1800
3	50	Shipping	1500
4	60	IT	1400
5	80	Sales	2500
6	90	Executive	1700
7	110	Accounting	1700
8	190	Contracting	1700

**Produto**

**Cartesiano:**

**20 x 8 = 160 linhas**

	EMPLOYEE_ID	DEPARTMENT_ID	LOCATION_ID
1	100	90	1700
2	101	90	1700
3	102	90	1700
4	103	60	1700



...

159	205	110	1700
160	206	110	1700



# Gerando um Produto Cartesiano

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

	 LAST_NAME	 DEPARTMENT_NAME
1	Abel	Administration
2	Davies	Administration
3	De Haan	Administration
4	Ernst	Administration
5	Fay	Administration

...

159	Whalen	Contracting
160	Zlotkey	Contracting

# Summary

Nesta lição, você aprendeu a usar junções para exibir os dados de várias tabelas com:

- Equijoins
- Nonequijoins
- `OUTER` joins
- Self-joins
- Cross joins
- Natural joins
- Full `OUTER` joins