
FIAP
FIAP GRADUAÇÃO

FIAP

Tecnologia em Análise e Desenvolvimento de Sistemas

Profº Ms. Alexandre Barcelos
profalexandre.barcelos@fiap.com.br

2019

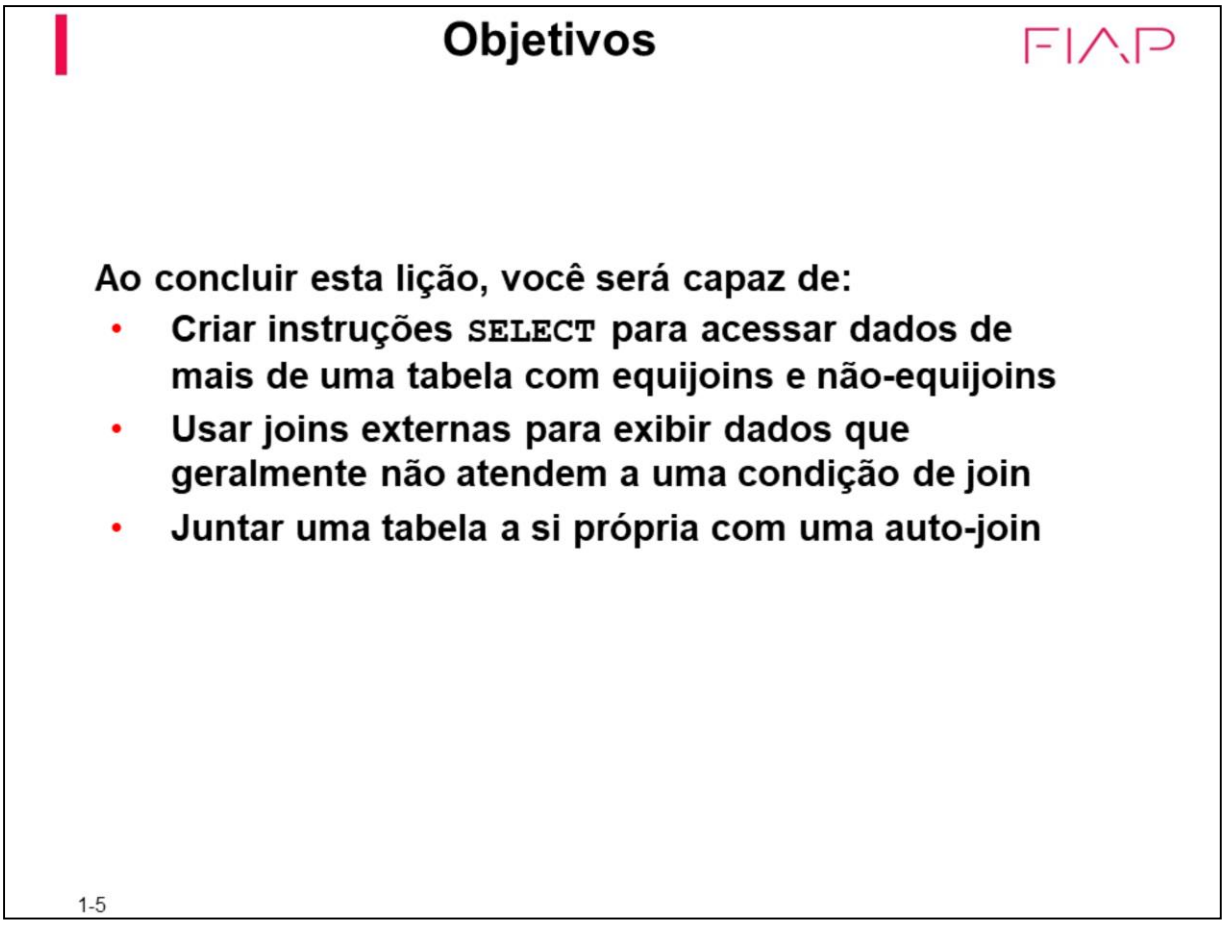
FIAP

Database Application Development

Profº Ms. Alexandre Barcelos
profalexandre.barcelos@fiap.com.br

2019

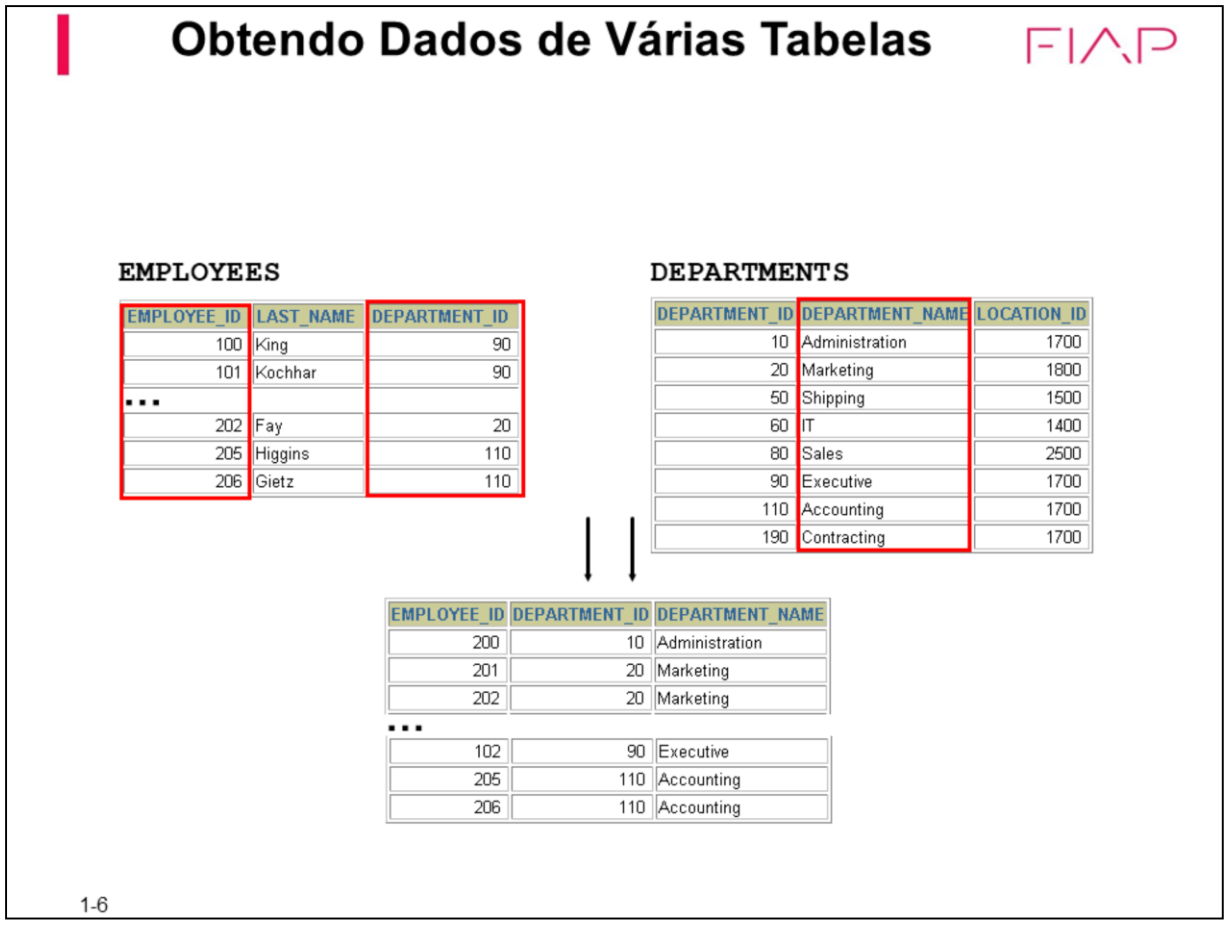
Sintaxe de Join SQL89



Objetivos

Esta lição explica como obter dados de mais de uma tabela. Uma *join* é usada para exibir informações de várias tabelas. Portanto, você pode *juntar* tabelas para exibir informações de mais de uma tabela.

Observação: Para obter informações sobre joins, consulte “SQL Queries and Subqueries: Joins” no manual *Oracle SQL Reference*.

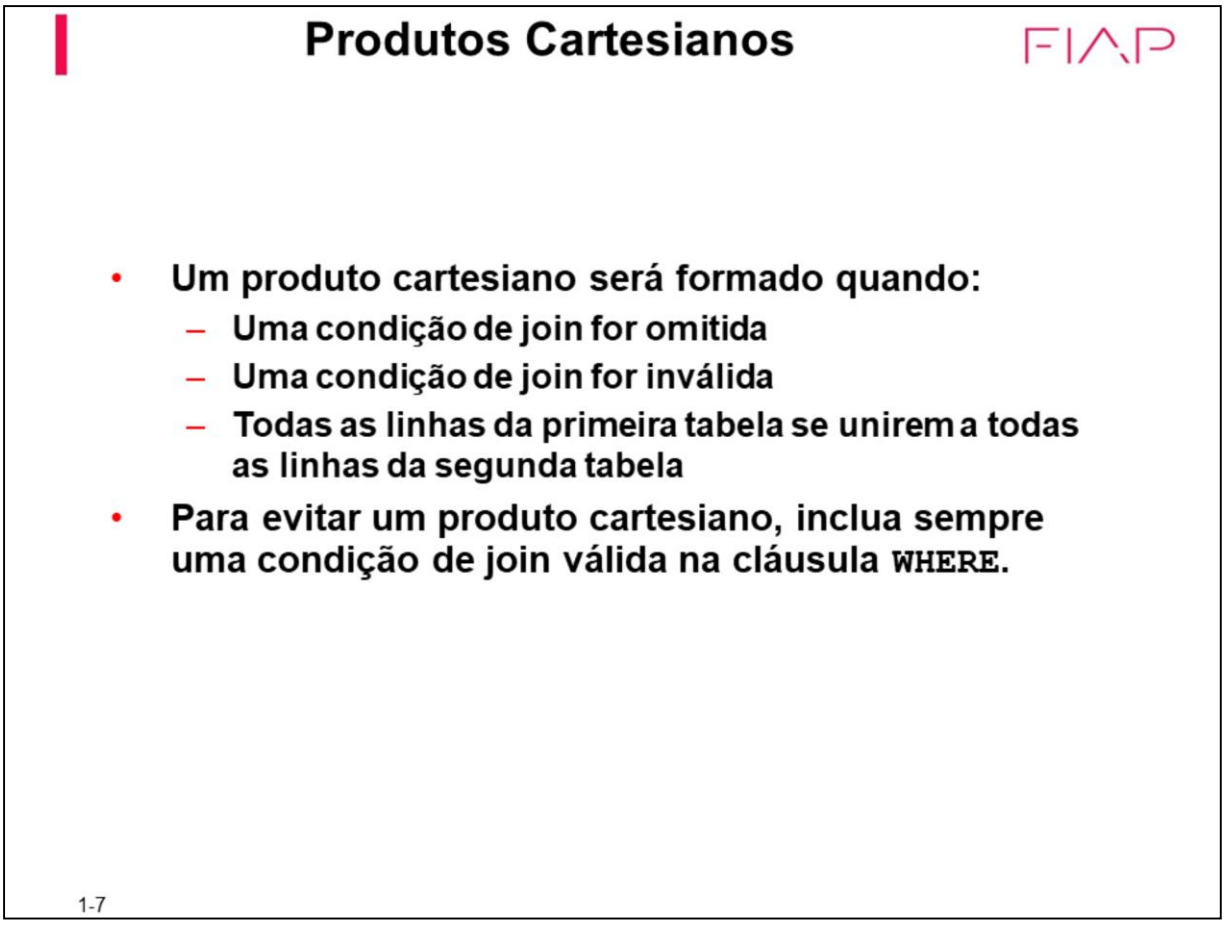


Dados de Várias Tabelas

Às vezes, é necessário usar dados de mais de uma tabela. No exemplo do slide, o relatório exibe dados de duas tabelas distintas:

- Os IDs de funcionário estão na tabela `EMPLOYEES`.
- Os IDs de departamento estão nas tabelas `EMPLOYEES` e `DEPARTMENTS`.
- Os nomes de departamento estão na tabela `DEPARTMENTS`.

Para gerar o relatório, você precisa vincular as tabelas `EMPLOYEES` e `DEPARTMENTS` e acessar os dados dessas duas tabelas.

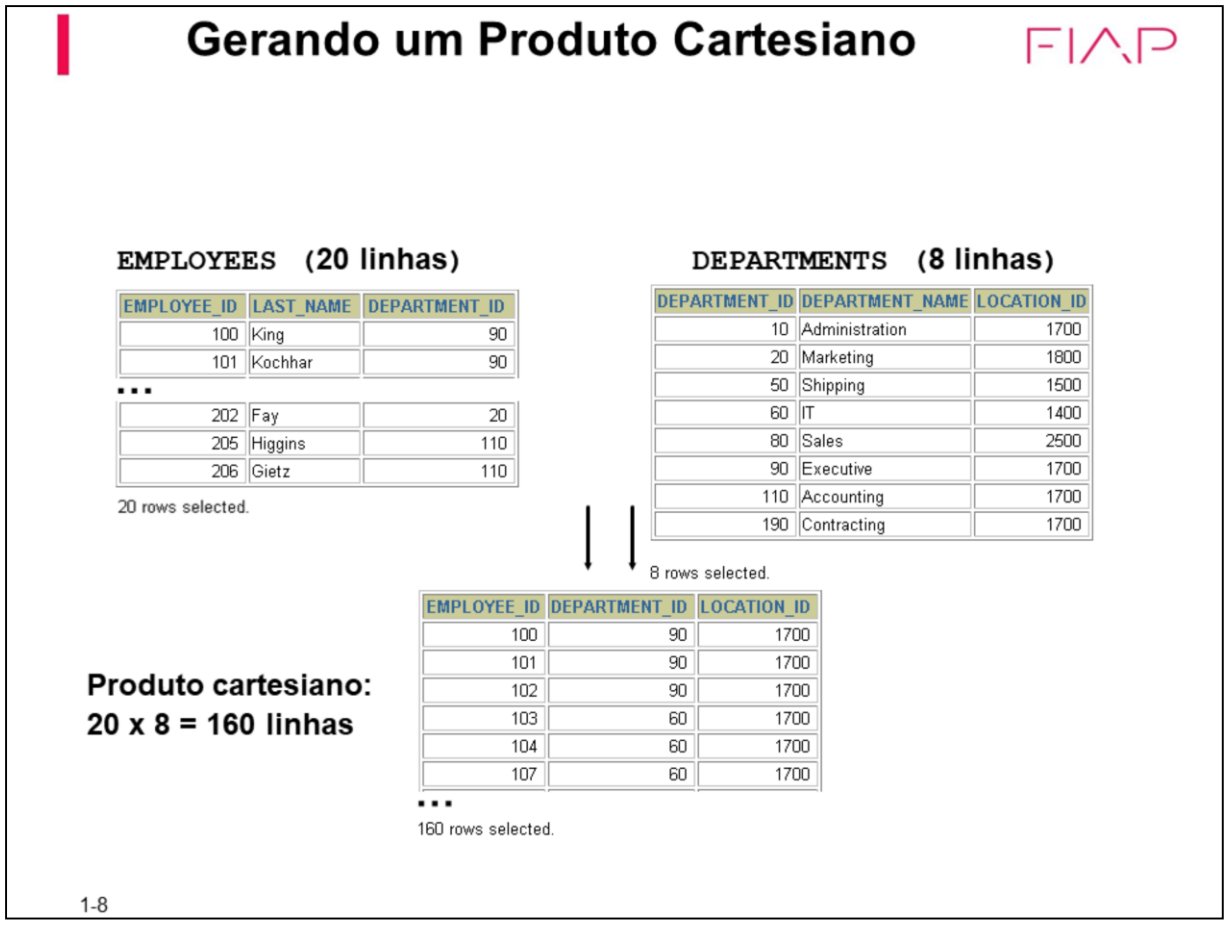


Produtos Cartesianos

Quando uma condição de join é inválida ou completamente omitida, o resultado é um *produto cartesiano*, no qual todas as combinações de linhas são exibidas. Todas as linhas da primeira tabela são unidas a todas as linhas da segunda tabela.

Um produto cartesiano tende a gerar um grande número de linhas e o resultado raramente é útil. Inclua sempre uma condição de join válida, a menos que exista uma necessidade específica de combinar todas as linhas de todas as tabelas.

Os produtos cartesianos são úteis em alguns testes quando é necessário gerar muitas linhas para simular um volume razoável de dados.



Produtos Cartesianos (continuação)

Um produto cartesiano será gerado se uma condição de join for omitida. O exemplo do slide exibe o sobrenome e o nome do departamento dos funcionários com base nas tabelas `EMPLOYEES` e `DEPARTMENTS`. Como não foi especificada uma condição de join, todas as linhas (20) da tabela `EMPLOYEES` são unidas a todas as linhas (8) da tabela `DEPARTMENTS`, gerando 160 linhas na saída.

```
SELECT last_name, department_name dept_name
```

LAST_NAME	DEPT_NAME
King	Administration
Kochhar	Administration
De Haan	Administration

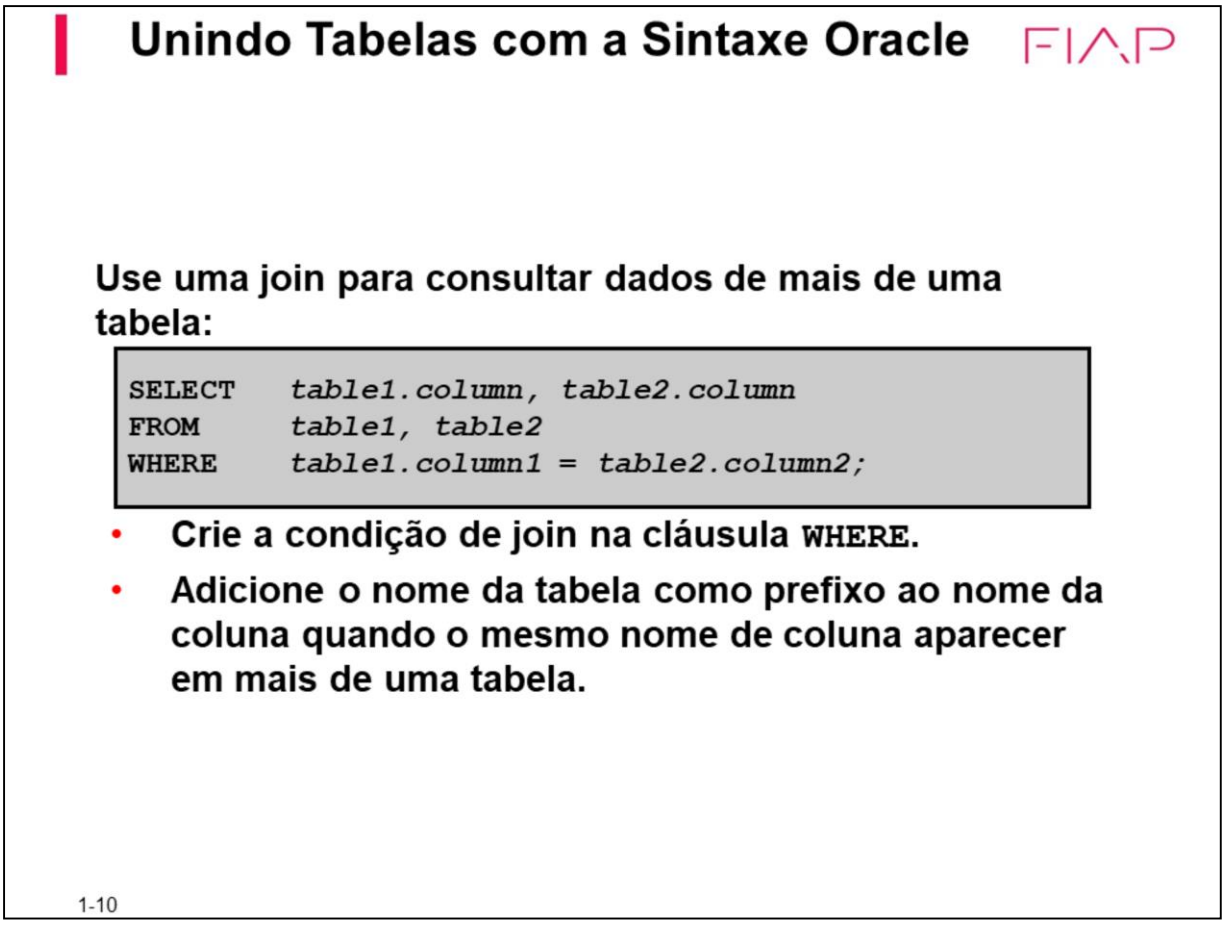
...

160 rows selected.

Tipos de Join

Antes da release do Banco de Dados Oracle9*i*, a sintaxe de join era proprietária.

Observação: A sintaxe de join compatível com o SQL:1999 não oferece benefícios de desempenho em relação à sintaxe de join de propriedade Oracle existente nas releases anteriores. Para obter informações detalhadas sobre a sintaxe de join compatível com o SQL:1999, consulte a Lição 5.



Definindo Joins

Quando são necessários dados de mais de uma tabela do banco de dados, é usada uma condição de *join*. É possível unir linhas de uma tabela a linhas de outra, de acordo com valores comuns existentes em colunas correspondentes (isto é, geralmente colunas de chave primária e estrangeira).

Para exibir os dados de duas ou mais tabelas relacionadas, crie uma condição de join simples na cláusula `WHERE`.

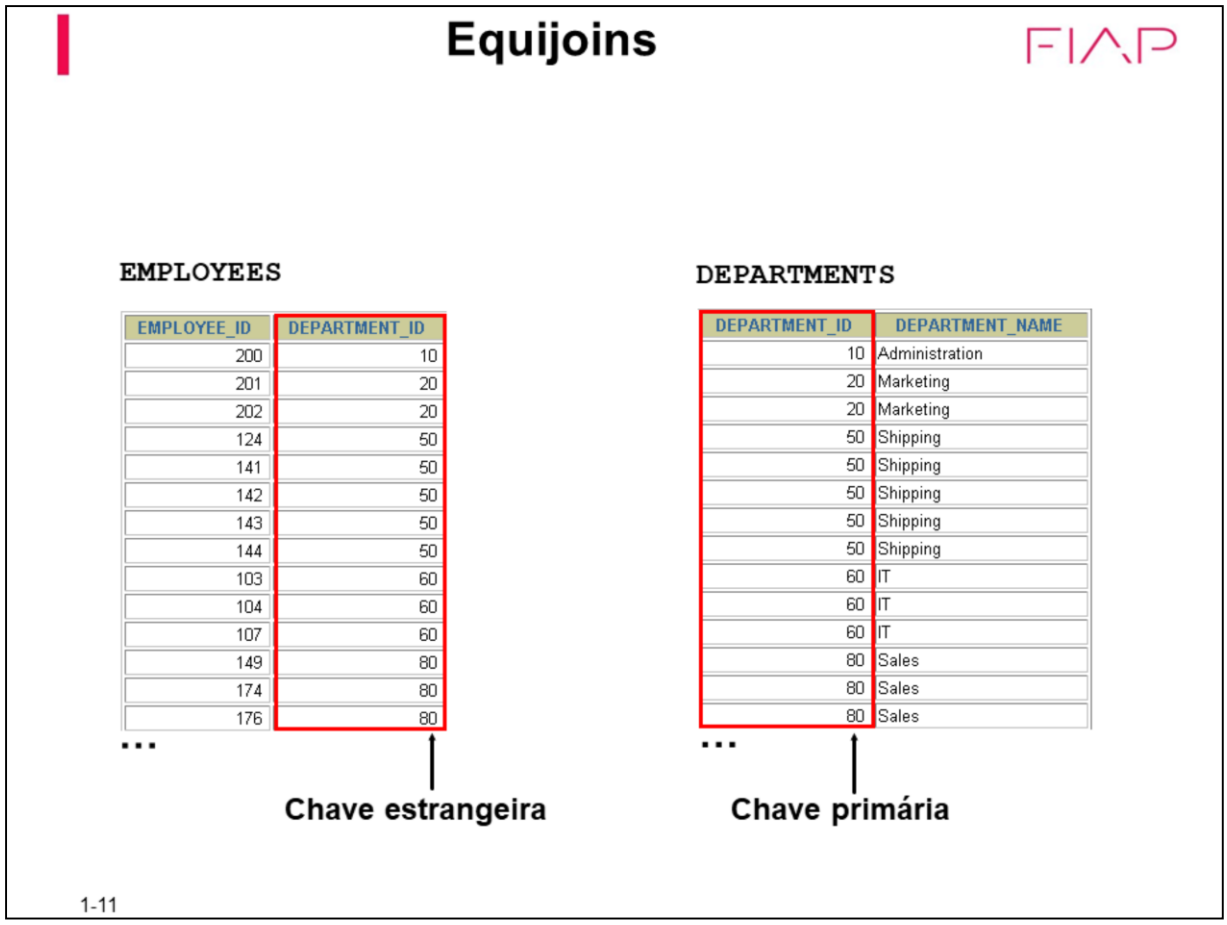
Na sintaxe:

`table1.column`	indica a tabela e a coluna das quais os dados são recuperados
`table1.column1 = table2.column2`	é a condição que une (ou relaciona) as tabelas

Diretrizes

- Quando criar uma instrução `SELECT` para unir tabelas, preceda o nome da coluna pelo nome da tabela por uma questão de clareza e para melhorar o acesso ao banco de dados.
- Se o mesmo nome de coluna aparecer em mais de uma tabela, adicione o nome da tabela como um prefixo ao nome da coluna.
- Para unir *`n`* tabelas, você precisará de, pelo menos, `n-1` condições de join. Por

exemplo, para unir quatro tabelas, é necessário um mínimo de três joins. Essa regra não será aplicada se a tabela tiver uma chave primária concatenada; nesse caso, será necessário mais de uma coluna para identificar cada linha com exclusividade.



Equijoins

Para determinar o nome do departamento de um funcionário, compare o valor da coluna `DEPARTMENT_ID` na tabela `EMPLOYEES` com os valores de `DEPARTMENT_ID` na tabela `DEPARTMENTS`. O relacionamento entre as tabelas `EMPLOYEES` e `DEPARTMENTS` é uma *equijoin* – isto é, os valores da coluna `DEPARTMENT_ID` nas duas tabelas devem ser iguais. Com freqüência, esse tipo de join envolve complementos de chave primária e estrangeira.

Observação: As equijoins são chamadas de *joins simples* ou *joins internas*.

Recuperando Registros com Equijoins

FIAP

```
SELECT employees.employee_id, employees.last_name,
       employees.department_id, departments.department_id,
       departments.location_id
FROM   employees, departments
WHERE  employees.department_id = departments.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
143	Matos	50	50	1500
144	Vargas	50	50	1500

...

19 rows selected.

1-12

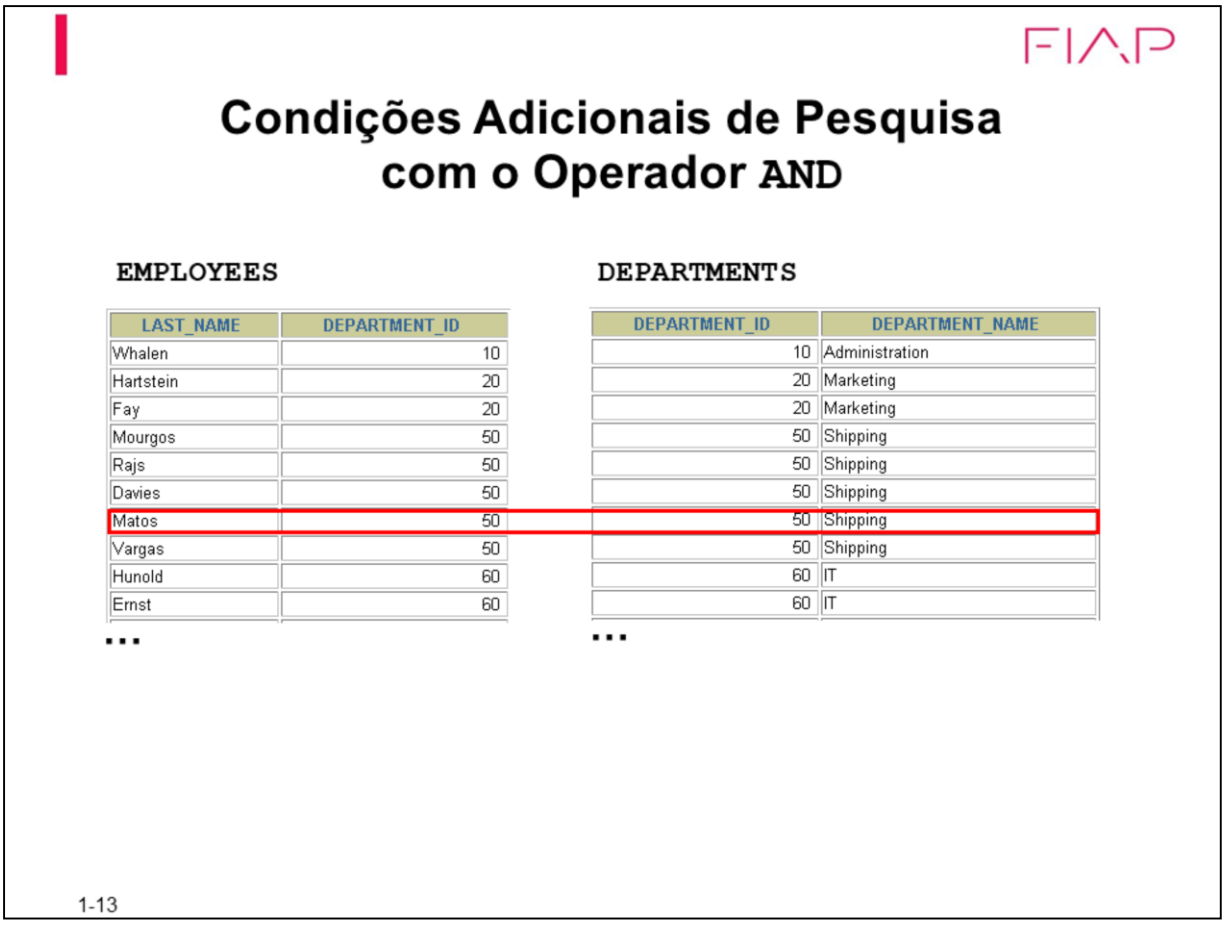
Recuperando Registros com Equijoins

No exemplo do slide:

- A cláusula `SELECT` especifica os nomes das colunas a serem recuperadas:
  - Sobrenome do funcionário, número do funcionário e número do departamento, que são colunas da tabela `EMPLOYEES`
  - Número do departamento, nome do departamento e ID do local, que são colunas da tabela `DEPARTMENTS`
- A cláusula `FROM` especifica as duas tabelas que o banco de dados deve acessar:
  - Tabela `EMPLOYEES`
  - Tabela `DEPARTMENTS`
- A cláusula `WHERE` especifica como as tabelas serão unidas:

  `EMPLOYEES.DEPARTMENT_ID = DEPARTMENTS.DEPARTMENT_ID`

Como a coluna `DEPARTMENT_ID` é comum às duas tabelas, é necessário adicionar o nome da tabela como prefixo ao nome dessa coluna para evitar ambigüidade.

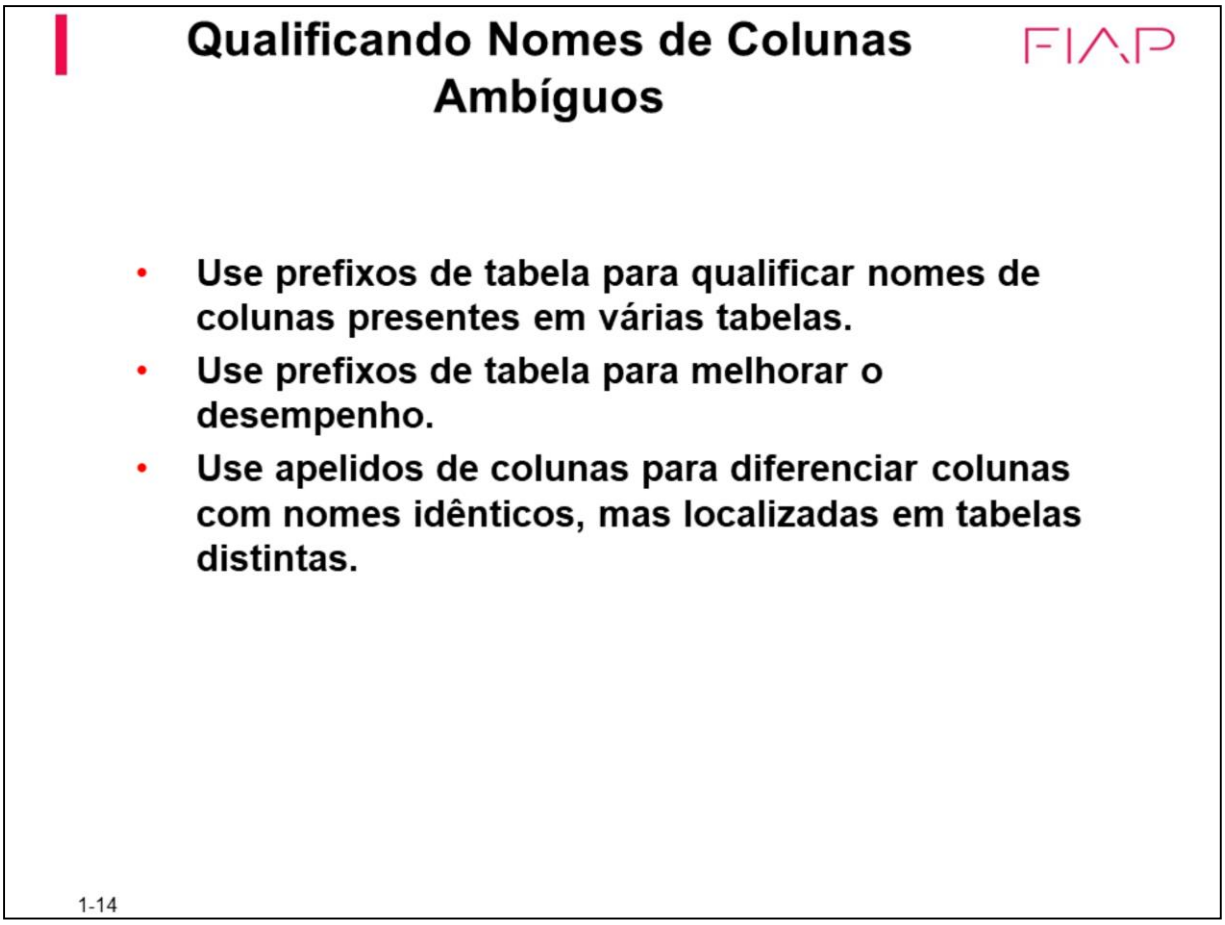


Condições Adicionais de Pesquisa

Além da join, você pode especificar critérios na cláusula `WHERE` para restringir as linhas a serem consideradas em uma ou mais tabelas da join. Por exemplo, para exibir o número e o nome do departamento do funcionário Matos, você precisa de uma condição adicional na cláusula `WHERE`.

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

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Matos	50	Shipping

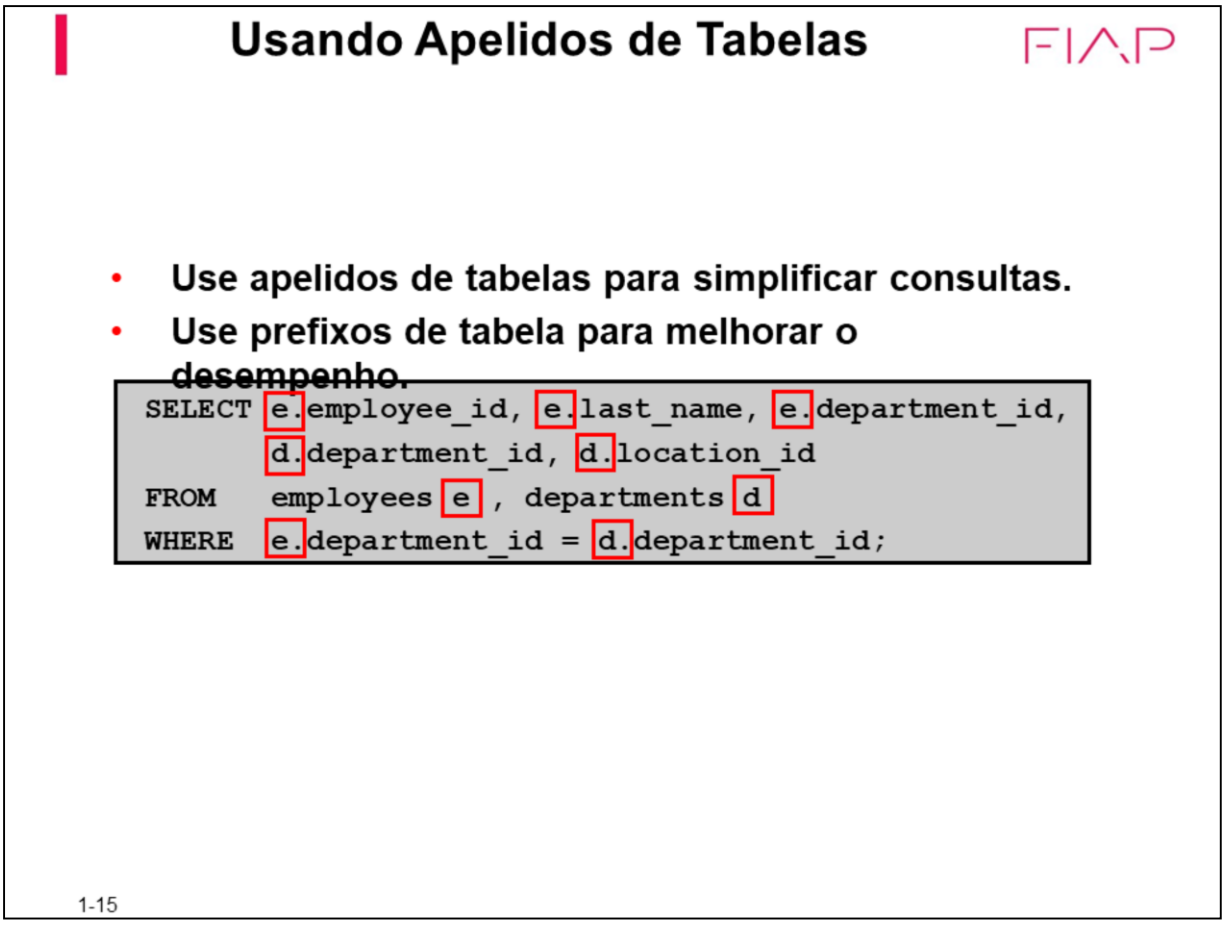


Qualificando Nomes de Colunas Ambíguos

Você precisa qualificar os nomes das colunas na cláusula `WHERE` com o nome da tabela para evitar ambigüidades. Sem os prefixos das tabelas, a coluna `DEPARTMENT_ID` poderá ser originária da tabela `DEPARTMENTS` ou `EMPLOYEES`. É necessário adicionar o prefixo da tabela para executar a consulta.

Se não houver nomes de colunas comuns entre as duas tabelas, não será preciso qualificar as colunas. No entanto, o uso do prefixo da tabela melhora o desempenho, pois você informa ao servidor Oracle exatamente onde localizar as colunas.

O requisito para qualificar nomes de colunas ambíguos também é aplicado a colunas que possam gerar ambigüidade em outras cláusulas, como a cláusula `SELECT` ou `ORDER BY`.



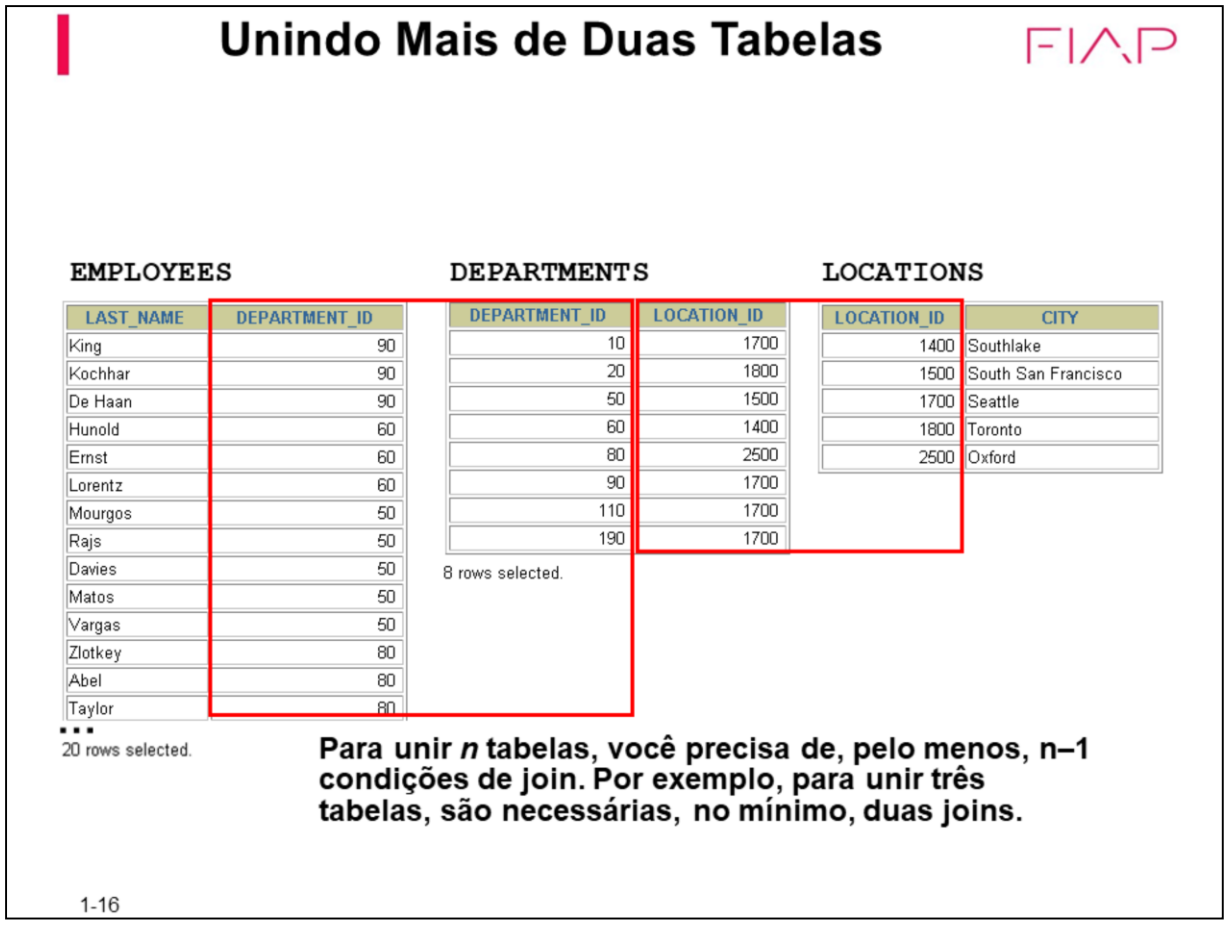
Usando Apelidos de Tabelas

A qualificação dos nomes de colunas com nomes de tabelas pode consumir muito tempo, especialmente se os *nomes das tabelas* forem longos. Você pode usar os apelidos das tabelas em vez dos nomes. Assim como um apelido de coluna fornece outro nome a uma coluna, um apelido de tabela fornece um outro nome a uma tabela. Os apelidos de tabelas ajudam a reduzir o tamanho do código SQL, utilizando menos memória.

Observe como os apelidos de tabelas são identificados na cláusula `FROM` do exemplo. O nome da tabela é especificado por inteiro, seguido por um espaço e, depois, o apelido da tabela. A tabela `EMPLOYEES` recebeu o apelido e, e a tabela `DEPARTMENTS`, o apelido `d`.

Diretrizes

- Um apelido de tabela pode conter até 30 caracteres, mas é recomendável especificar o menor nome possível.
- Se um apelido de tabela for usado para um nome de tabela específico na cláusula `FROM`, ele deverá ser substituído pelo nome da tabela em toda a instrução `SELECT`.
- Os apelidos de tabelas devem ser significativos.
- Um apelido de tabela é válido somente para a instrução `SELECT` atual.



Condições Adicionais de Pesquisa

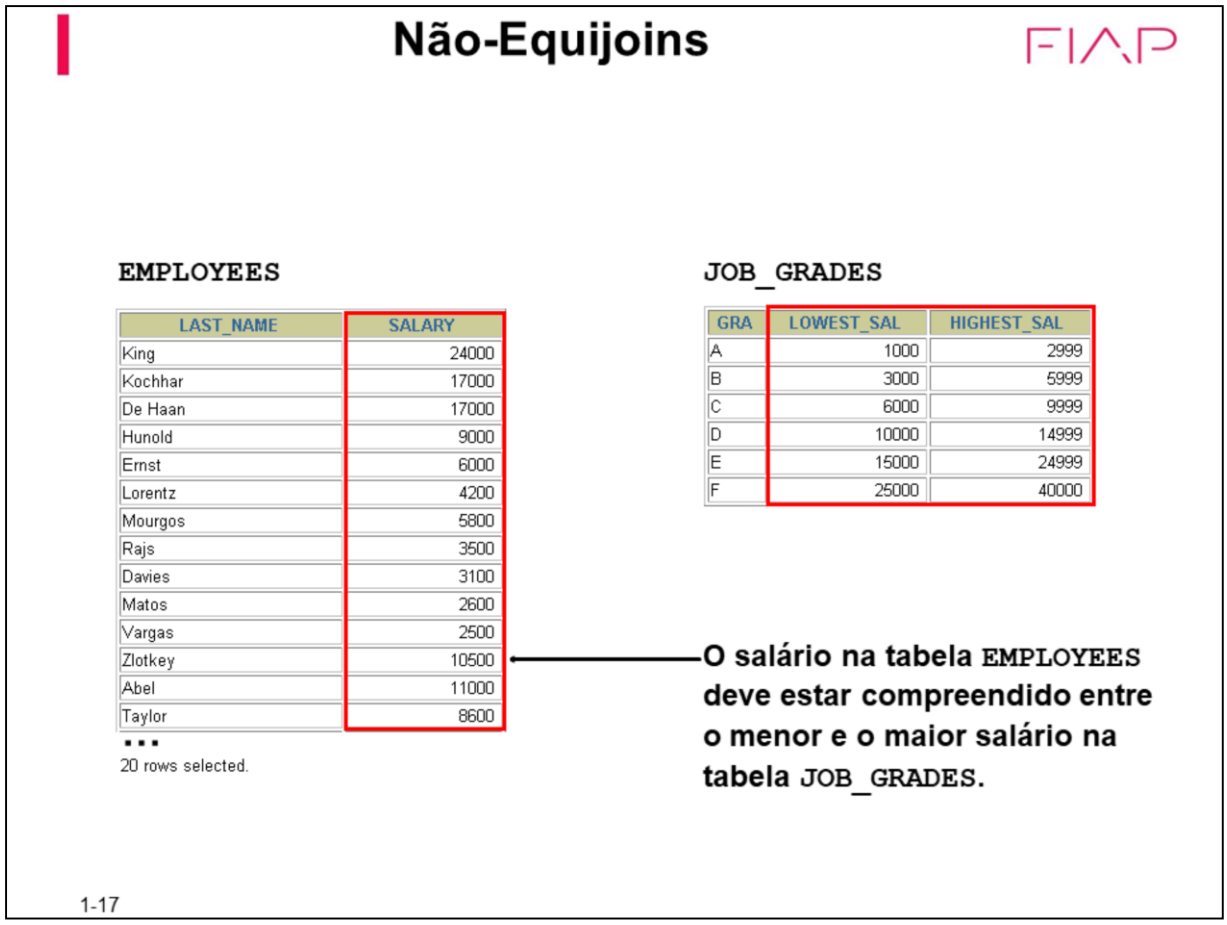
Às vezes, talvez seja necessário unir mais de duas tabelas. Por exemplo, para exibir o sobrenome, o nome do departamento e a cidade de cada funcionário, você precisa unir as tabelas `EMPLOYEES`, `DEPARTMENTS` e `LOCATIONS`.

```
SELECT e.last_name, d.department_name, l.city
FROM   employees e, departments d, locations l
WHERE  e.department_id = d.department_id
```

LAST_NAME	DEPARTMENT_NAME	CITY
Hunold	IT	Southlake
Ernst	IT	Southlake
Lorentz	IT	Southlake
Mourgos	Shipping	South San Francisco
Rajs	Shipping	South San Francisco
Davies	Shipping	South San Francisco

▪ ▪ ▪

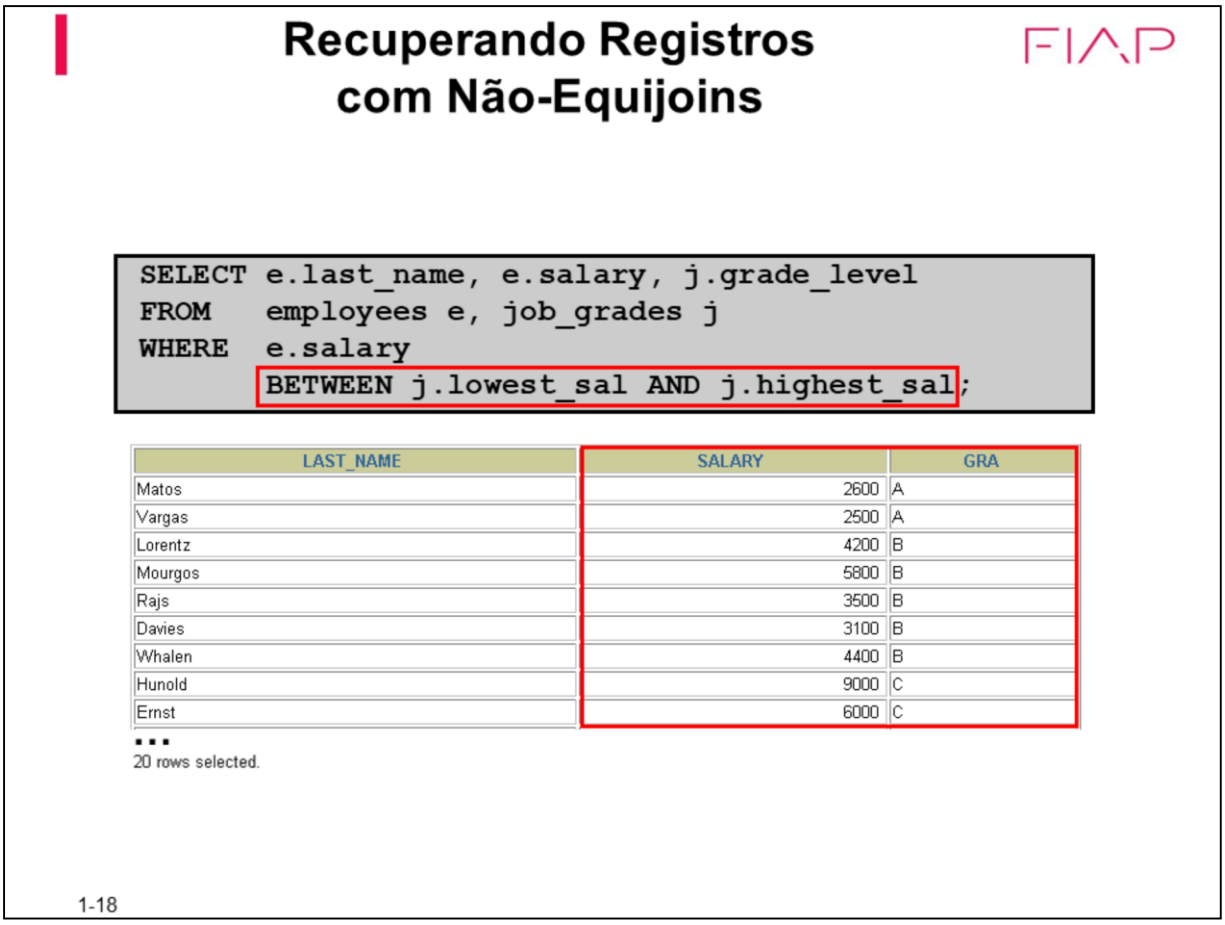
19 rows selected.



Não-Equijoins

Uma não-equijoin é uma condição de join que contém algo diferente de um operador de igualdade.

O relacionamento entre as tabelas `EMPLOYEES` e `JOB_GRADES` é um exemplo de uma não-equijoin. Em um relacionamento entre as duas tabelas, os valores da coluna `SALARY` da tabela `EMPLOYEES` devem estar compreendidos entre os valores das colunas `LOWEST_SALARY` e `HIGHEST_SALARY` da tabela `JOB_GRADES`. O relacionamento é obtido com um operador diferente de igualdade (=).



Não-Equijoins (continuação)

O exemplo do slide cria uma não-equijoin para avaliar o nível salarial de um funcionário. O salário deve estar compreendido *entre* qualquer par de faixas de salário mais baixo e mais alto.

É importante observar que todos os funcionários aparecem uma única vez quando essa consulta é executada. Os funcionários não são repetidos na lista. Existem dois motivos para isso:

- Nenhuma das linhas da tabela de níveis de cargos contém níveis sobrepostos. Isto é, o valor do salário de um funcionário somente pode estar entre os valores de salário mais alto e mais baixo de uma das linhas da tabela de níveis salariais.
- Os salários de todos os funcionários estão compreendidos entre os limites fornecidos pela tabela de níveis de cargos. Isto é, nenhum funcionário recebe menos que o valor mais baixo contido na coluna `LOWEST_SAL` ou mais que o valor mais alto contido na coluna `HIGHEST_SAL`.

Observação: É possível usar outras condições (como <= e >=), mas `BETWEEN` é a mais simples. Quando usar `BETWEEN`, lembre-se de informar primeiro o valor mais baixo e depois o valor mais alto.

Foram especificados apelidos de tabelas no exemplo do slide por questões de desempenho, e não para evitar uma possível ambigüidade.

Retornando Registros sem Correspondência Direta com Joins Externas

Se não atender a uma condição de join, a linha não aparecerá no resultado da consulta. Por exemplo, na condição de equijoin das tabelas `EMPLOYEES` e `DEPARTMENTS`, o nome da funcionária Grant não é exibido, pois não existe um ID de departamento para ela na tabela `EMPLOYEES`. Em vez de conter 20 funcionários, o conjunto de resultados conterá 19 registros.

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

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Hartstein	20	Marketing
Fay	20	Marketing
Mourgos	50	Shipping

...

19 rows selected.

Sintaxe de Joins Externas

FIAP

- Use uma join externa para ver as linhas que não atendem à condição de join.
- O operador de join externa é o sinal de adição (+).

```
SELECT table1.column, table2.column
FROM   table1, table2
WHERE  table1.column(+) = table2.column;
```

```
SELECT table1.column, table2.column
FROM   table1, table2
WHERE  table1.column = table2.column(+);
```

1-20

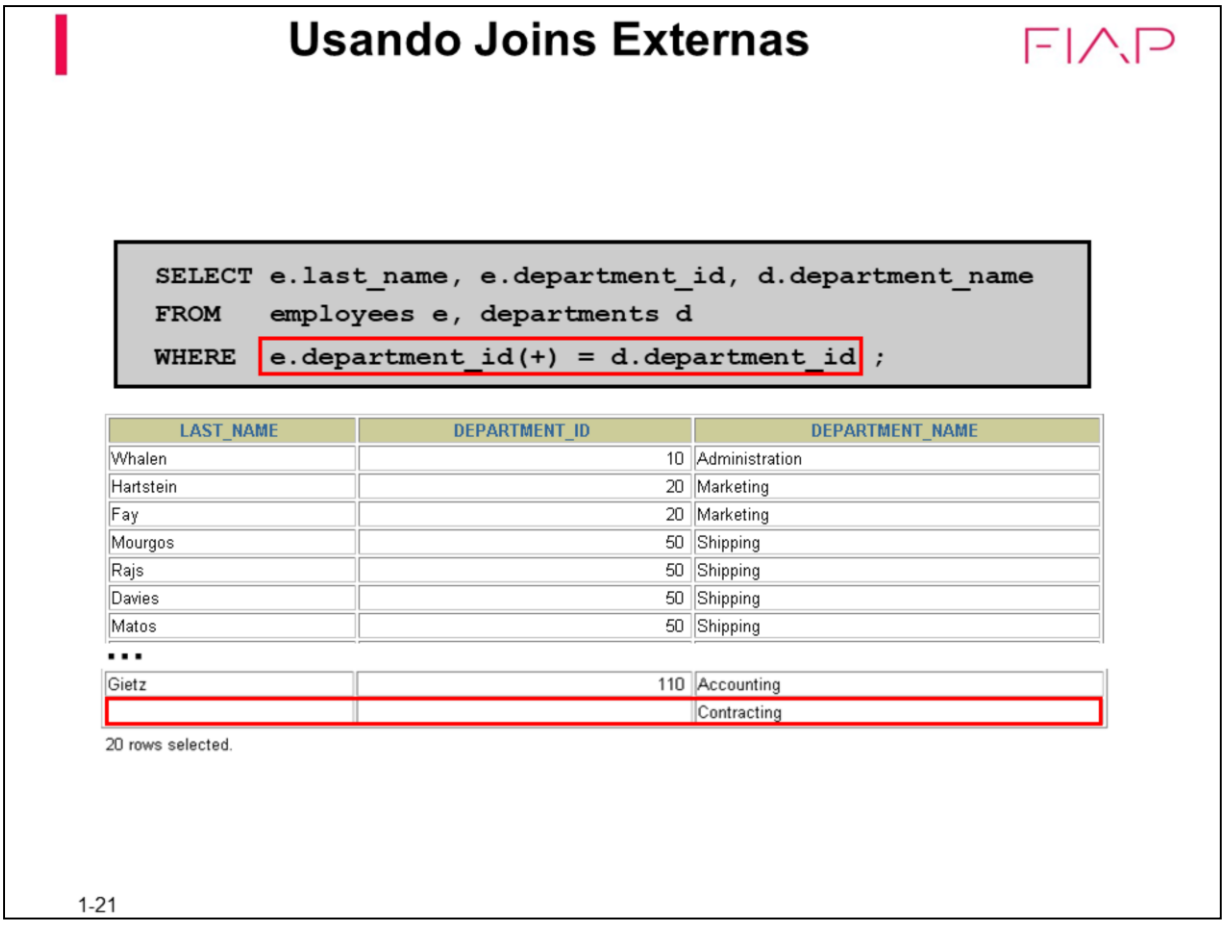
Usando Joins Externas para Retornar Registros sem Correspondência Direta

Para retornar as linhas sem correspondência, utilize um operador de *join externa* na condição de join. O operador é um sinal de adição entre parênteses (+), colocado ao "lado" da join em que há falta de informações. Esse operador cria uma ou mais linhas nulas, às quais é possível unir uma ou mais linhas da tabela com registros correspondentes.

Na sintaxe:

`table1.column` = é a condição que une (ou relaciona) as tabelas

`table2.column` (+) é o símbolo de join externa, que pode ser colocado em qualquer lado da condição da cláusula `WHERE`, mas não em ambos. (Coloque o símbolo de join externa após o nome da coluna na tabela sem as linhas correspondentes.)



Usando Joins Externas para Retornar Registros sem Correspondência Direta (continuação)

O exemplo do slide exibe sobrenomes, IDs de departamento e nomes de departamento. O departamento Contracting não tem funcionários. Nenhum valor é mostrado na saída.

Restrições de Joins Externas

- O operador de join externa só pode aparecer em *um* lado da expressão – o lado no qual faltam informações. Ele retorna as linhas de uma tabela sem uma correspondência direta na outra tabela.
- Uma condição que envolva uma join externa não pode usar o operador `IN` ou ser vinculada a outra condição pelo operador `OR`.

Auto-Joins

FIAP

EMPLOYEES (WORKER)

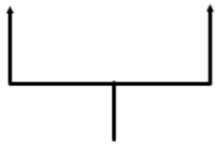
EMPLOYEE_ID	LAST_NAME	MANAGER_ID
100	King	
101	Kochhar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103
107	Lorentz	103
124	Mourgos	100

...

EMPLOYEES (MANAGER)

EMPLOYEE_ID	LAST_NAME
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst
107	Lorentz
124	Mourgos

...



MANAGER_ID na tabela WORKER é igual a EMPLOYEE_ID na tabela MANAGER.

1-22

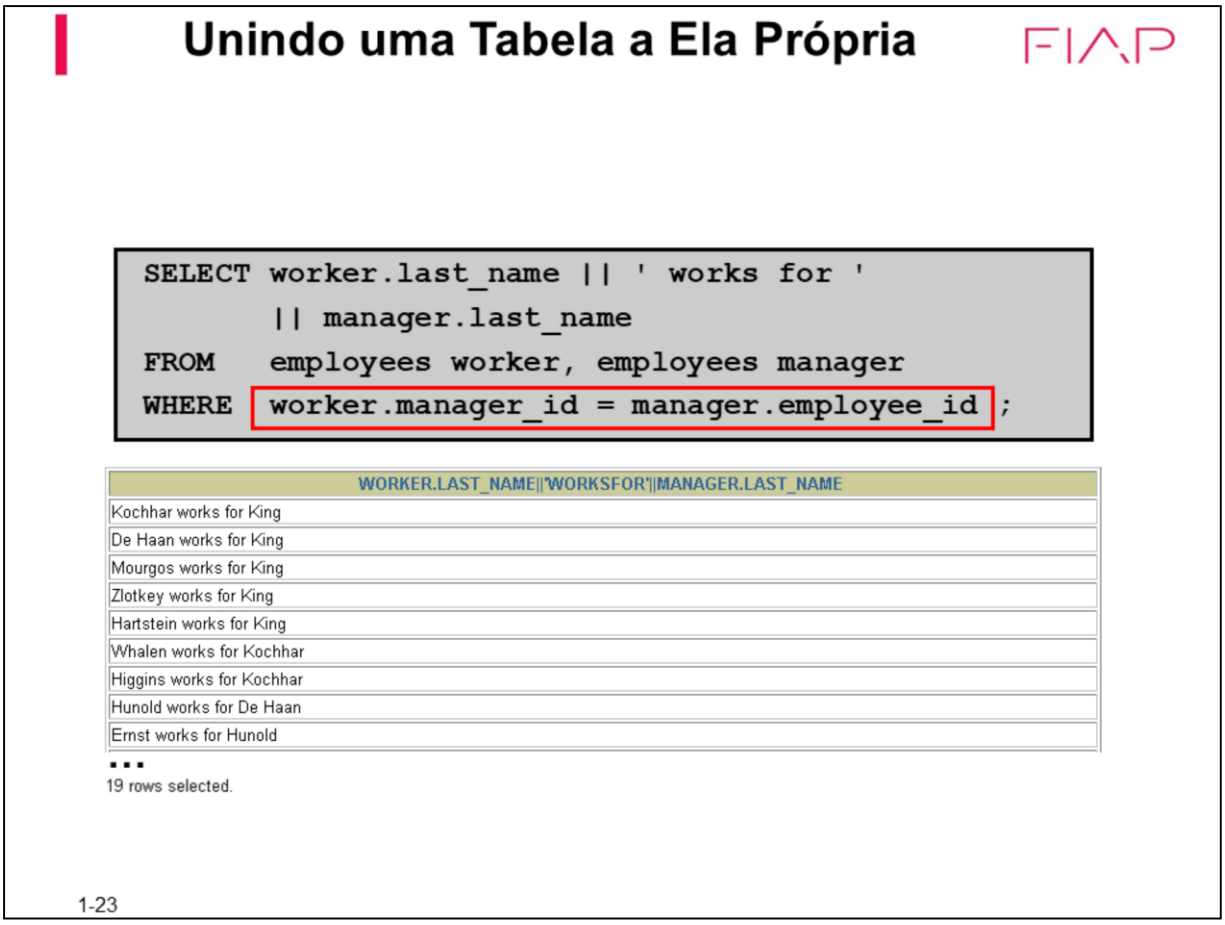
Unindo uma Tabela a Ela Própria

Às vezes, é necessário unir uma tabela a ela própria. Para obter o nome do gerente de cada funcionário, você precisa unir a tabela `EMPLOYEES` a ela própria; esse tipo de join é denominado *auto-join*.

Por exemplo, para obter o nome do gerente de Lorentz, você precisa:

- Localizar Lorentz na tabela `EMPLOYEES` examinando a coluna `LAST_NAME`.
- Localizar o número do gerente de Lorentz examinando a coluna `MANAGER_ID`. O número do gerente de Lorentz é 103.
- Localizar o nome do gerente com o valor de `EMPLOYEE_ID` 103 examinando a coluna `LAST_NAME`. O número de funcionário de Hunold é 103, portanto, Hunold é o gerente de Lorentz.

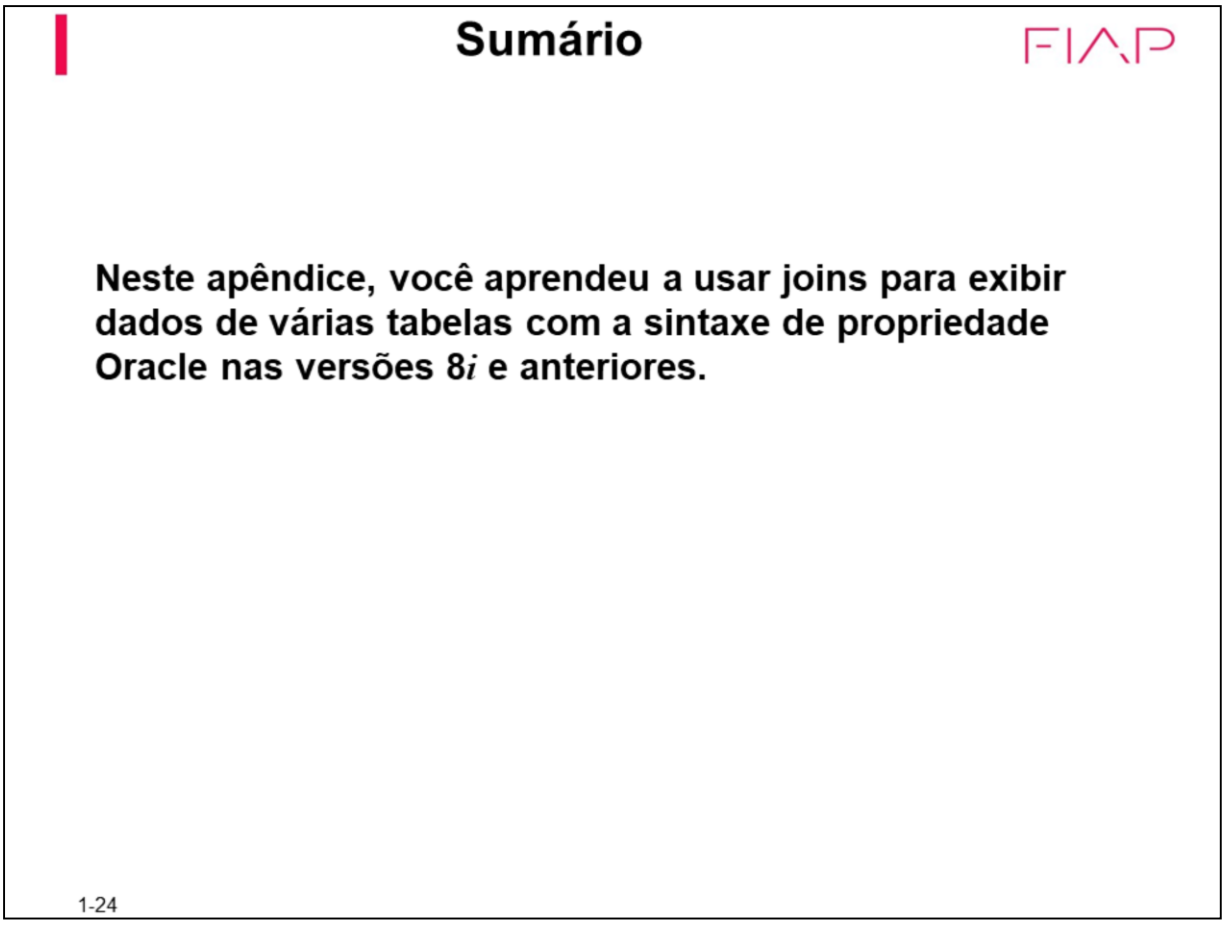
Nesse processo, você examinará a tabela duas vezes. Na primeira vez, você examinará a tabela para localizar Lorentz na coluna `LAST_NAME` e o valor de `MANAGER_ID` 103. Na segunda vez, você examinará a coluna `EMPLOYEE_ID` para localizar 103 e a coluna `LAST_NAME` para localizar Hunold.



Unindo uma Tabela a Ela Própria (continuação)

O exemplo do slide une a tabela `EMPLOYEES` a ela própria. Para simular duas tabelas na cláusula `FROM`, existem dois apelidos, `w` e `m`, para a mesma tabela, `EMPLOYEES`.

Neste exemplo, a cláusula `WHERE` contém a join que significa “onde o número do gerente de um funcionário corresponde ao número de funcionário do gerente”.



Sumário

Há várias maneiras de unir tabelas.

Tipos de Join

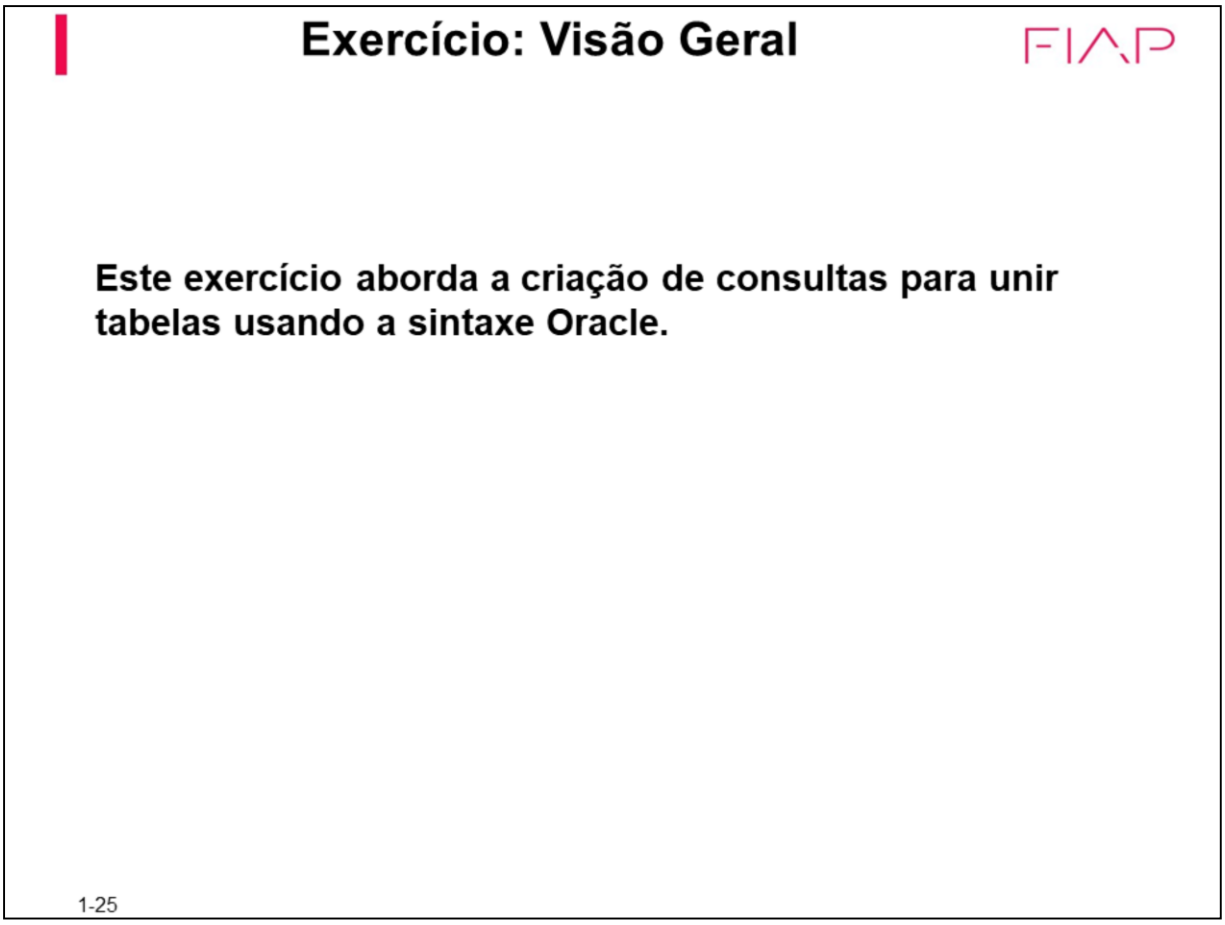
- Produtos cartesianos
- Equijoins
- Não-equijoins
- Joins externas
- Auto-joins

Produtos Cartesianos

Um produto cartesiano resulta na exibição de todas as combinações de linhas. Para obter esse resultado, omita a cláusula `WHERE`.

Apelidos de Tabelas

- Os apelidos de tabelas aceleram o acesso ao banco de dados.
- Eles podem ajudar a reduzir o código SQL, preservando a memória.



Exercício C: Visão Geral

O objetivo deste exercício é proporcionar várias atividades de junção de tabelas com a sintaxe Oracle abordada neste apêndice.

Exercício C

1. Crie uma consulta para o departamento de recursos humanos a fim de gerar os endereços de todos os departamentos. Use as tabelas `LOCATIONS` e `COUNTRIES`. Mostre o ID do local, o endereço, a cidade, o estado e o país na saída. Use `NATURAL JOIN` para gerar os resultados.

LOCATION_ID	STREET_ADDRESS	CITY	STATE_PROVINCE	COUNTRY_NAME
1400	2014 Jabberwocky Rd	Southlake	Texas	United States of America
1500	2011 Interiors Blvd	South San Francisco	California	United States of America
1700	2004 Charade Rd	Seattle	Washington	United States of America
1800	460 Bloor St. W.	Toronto	Ontario	Canada
2500	Magdalen Centre, The Oxford Science Park	Oxford	Oxford	United Kingdom

2. ento
é o nome do departamento de todos os funcionários.

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Hartstein	20	Marketing
Fay	20	Marketing
Mourgos	50	Shipping
Rajs	50	Shipping
Davies	50	Shipping
Vargas	50	Shipping
...		
De Haan	90	Executive
Higgins	110	Accounting
Gietz	110	Accounting

19 rows selected.

Exercício C (continuação)

3. O departamento de recursos humanos precisa de um relatório dos funcionários baseados em Toronto. Exiba o sobrenome, o cargo, o número do departamento e o nome do departamento de todos os funcionários que trabalham em Toronto.

LAST_NAME	JOB_ID	DEPARTMENT_ID	DEPARTMENT_NAME
Hartstein	MK_MAN	20	Marketing
Fay	MK_REP	20	Marketing

4. sobrenome e o número dos respectivos gerentes. Atribua às colunas os labels `Employee`, `Emp#`, `Manager` e `Mgr#`, respectivamente. Inclua a instrução SQL no arquivo de texto `lab_c_04.sql`.

Employee	EMP#	Manager	Mgr#
Kochhar	101	King	100
De Haan	102	King	100
Mourgos	124	King	100
Zlotkey	149	King	100
Hartstein	201	King	100
Whalen	200	Kochhar	101
Higgins	205	Kochhar	101
Hunold	103	De Haan	102
Ernst	104	Hunold	103
Lorentz	107	Hunold	103
Rajs	141	Mourgos	124
Davies	142	Mourgos	124
Matos	143	Mourgos	124
Vargas	144	Mourgos	124
Employee	**EMP#**	**Manager**	**Mgr#**
Abel	174	Zlotkey	149
Taylor	176	Zlotkey	149
Grant	178	Zlotkey	149
Fay	202	Hartstein	201
Gietz	206	Higgins	205

19 rows selected.

Exercício C (continuação)

5. Modifique `lab_c_04.sql` para exibir todos os funcionários, inclusive King, que não possui gerente. Ordene os resultados pelo número do funcionário. Inclua a instrução SQL no arquivo de texto `lab_c_05.sql`. Execute a consulta em `lab_c_05.sql`.

Employee	EMP#	Manager	Mgr#
King	100		
Kochhar	101	King	100
De Haan	102	King	100
Hunold	103	De Haan	102
Ernst	104	Hunold	103
Lorentz	107	Hunold	103
Mourgos	124	King	100

...

20 rows selected.

6. Crie um relatório para o departamento de recursos humanos que exiba os sobrenomes e os números de departamento dos funcionários, bem como todos os funcionários que trabalham no mesmo departamento como um funcionário específico. Atribua um label apropriado a cada coluna. Salve o script no arquivo `lab_c_06.sql`.

DEPARTMENT	EMPLOYEE	COLLEAGUE
20	Fay	Hartstein
20	Hartstein	Fay
50	Davies	Matos
50	Davies	Mourgos
50	Davies	Rajs
50	Davies	Vargas
50	Matos	Davies
50	Matos	Mourgos
50	Matos	Rajs
50	Matos	Vargas
50	Mourgos	Davies
50	Mourgos	Matos
50	Mourgos	Rajs
50	Mourgos	Vargas

...

42 rows selected.

Exercício C (continuação)

7. O departamento de recursos humanos precisa de um relatório sobre níveis de cargos e salários. Para se familiarizar com a tabela `JOB_GRADES`, primeiro mostre a estrutura dessa tabela. Em seguida, crie uma consulta que exiba o sobrenome, o cargo, o nome do departamento, o salário e o nível de todos os funcionários.

Name	Null?	Type
GRADE_LEVEL		VARCHAR2(3)
LOWEST_SAL		NUMBER
HIGHEST_SAL		NUMBER

LAST_NAME	JOB_ID	DEPARTMENT_NAME	SALARY	GRA
Matos	ST_CLERK	Shipping	2600	A
Vargas	ST_CLERK	Shipping	2500	A
Lorentz	IT_PROG	IT	4200	B
Mourgos	ST_MAN	Shipping	5800	B
Rajs	ST_CLERK	Shipping	3500	B
Davies	ST_CLERK	Shipping	3100	B
Whalen	AD_ASST	Administration	4400	B

Se quis
19 rows selected.

8. O departamento de recursos humanos deseja determinar os nomes de todos os de admissão de todos os funcionários admitidos após Davies.

LAST_NAME	HIRE_DATE
Lorentz	07-FEB-99
Mourgos	16-NOV-99
Matos	15-MAR-98
Vargas	09-JUL-98
Zlotkey	29-JAN-00
Taylor	24-MAR-98
Grant	24-MAY-99
Fay	17-AUG-97

8 rows selected.

Exercício C (continuação)

9. O departamento de recursos humanos precisa obter o nome e a data de admissão de todos os funcionários admitidos antes dos respectivos gerentes, além do nome e da data de admissão desses gerentes. Atribua às colunas os labels `Employee`, `Emp Hired`, `Manager` e `Mgr Hired`, respectivamente. Salve o script no arquivo `lab_c_09.sql`.

LAST_NAME	HIRE_DATE	LAST_NAME	HIRE_DATE
Whalen	17-SEP-87	Kochhar	21-SEP-89
Hunold	03-JAN-90	De Haan	13-JAN-93
Rajs	17-OCT-95	Mourgos	16-NOV-99
Davies	29-JAN-97	Mourgos	16-NOV-99
Matos	15-MAR-98	Mourgos	16-NOV-99
Vargas	09-JUL-98	Mourgos	16-NOV-99
Abel	11-MAY-96	Zlotkey	29-JAN-00
Taylor	24-MAR-98	Zlotkey	29-JAN-00
Grant	24-MAY-99	Zlotkey	29-JAN-00

9 rows selected.

```
--1
SELECT location_id, street_address, city, state_province,
country_name
FROM   locations, countries
WHERE  locations.country_id = countries.country_id;


--2
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e, departments d
WHERE  e.department_id = d.department_id;


--3
SELECT e.last_name, e.job_id, e.department_id,
```

```
d.department_name
FROM   employees e, departments d , locations l
WHERE  e.department_id = d.department_id
AND    d.location_id = l.location_id
AND    LOWER(l.city) = 'toronto';

--4
SELECT w.last_name "Employee", w.employee_id "EMP#",
       m.last_name "Manager", m.employee_id  "Mgr#"
FROM   employees w, employees m
WHERE  w.manager_id = m.employee_id

--5
SELECT w.last_name "Employee", w.employee_id "EMP#",
       m.last_name "Manager", m.employee_id  "Mgr#"
FROM   employees w, employees m
WHERE  w.manager_id  = m.employee_id (+);

--6
SELECT e.department_id department, e.last_name employee,
       c.last_name colleague
FROM   employees e, employees c
WHERE  e.department_id = c.department_id
AND    e.employee_id <> c.employee_id
ORDER BY e.department_id, e.last_name, c.last_name;

--7
DESC JOB_GRADES
```

```
SELECT e.last_name, e.job_id, d.department_name,
       e.salary, j.grade_level
FROM   employees e, departments d, job_grades j
WHERE  e.department_id = d.department_id
AND    e.salary BETWEEN j.lowest_sal AND j.highest_sal;

--8
SELECT e.last_name, e.hire_date
FROM   employees e , employees davies
WHERE  davies.last_name = 'Davies'
AND    davies.hire_date < e.hire_date;

--9
SELECT w.last_name, w.hire_date, m.last_name, m.hire_date
FROM   employees w , employees m
WHERE  w.manager_id = m.employee_id
AND    w.hire_date <  m.hire_date;
```

Bibliografia Utilizada

FIAP

Oracle SQL References: http://docs.oracle.com
Manuais Oracle – Introdução ao Oracle e SQL I e II