FIAP

FIAP GRADUAÇÃO

FIAP

Tecnologia em Análise e Desenvolvimento de Sistemas

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

2019

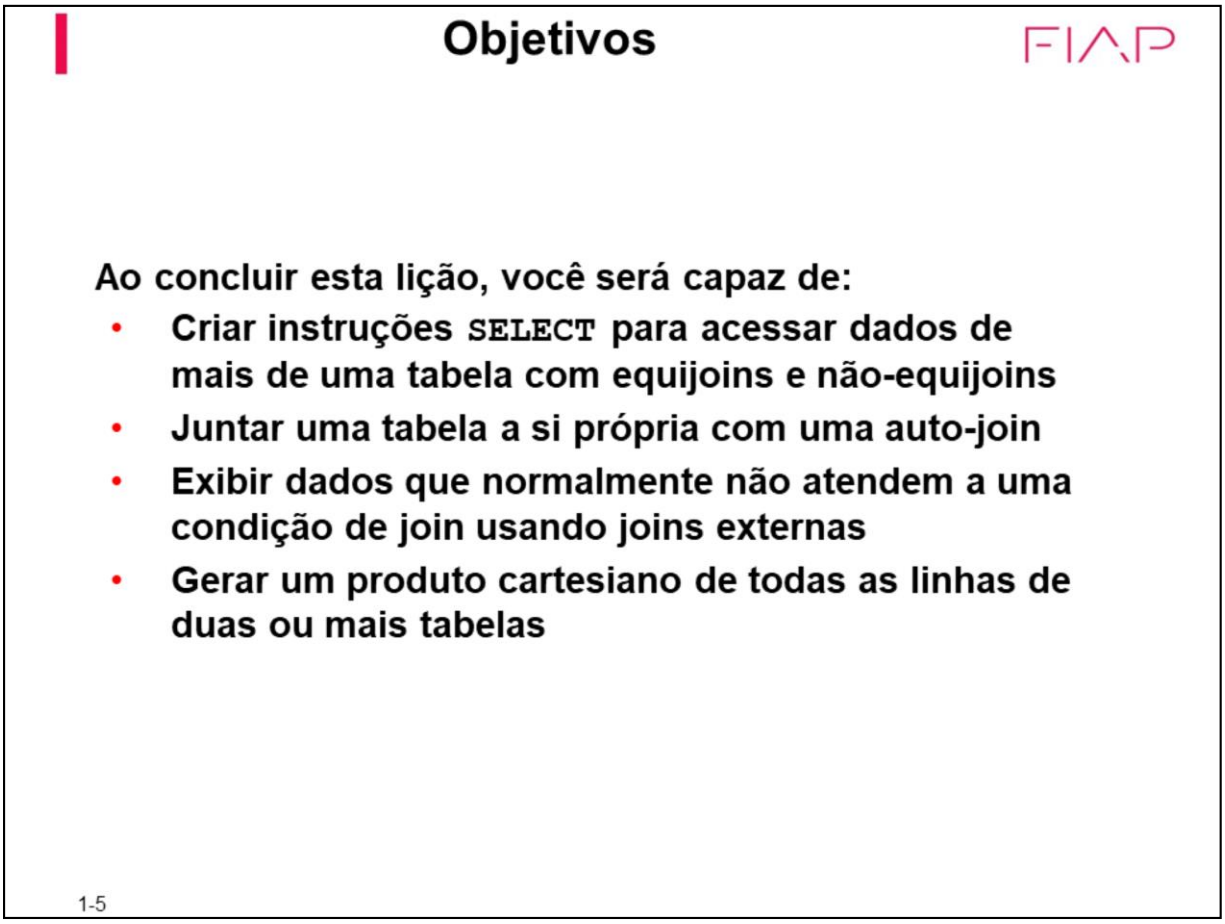
FIAP

Database Application Development

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

2019

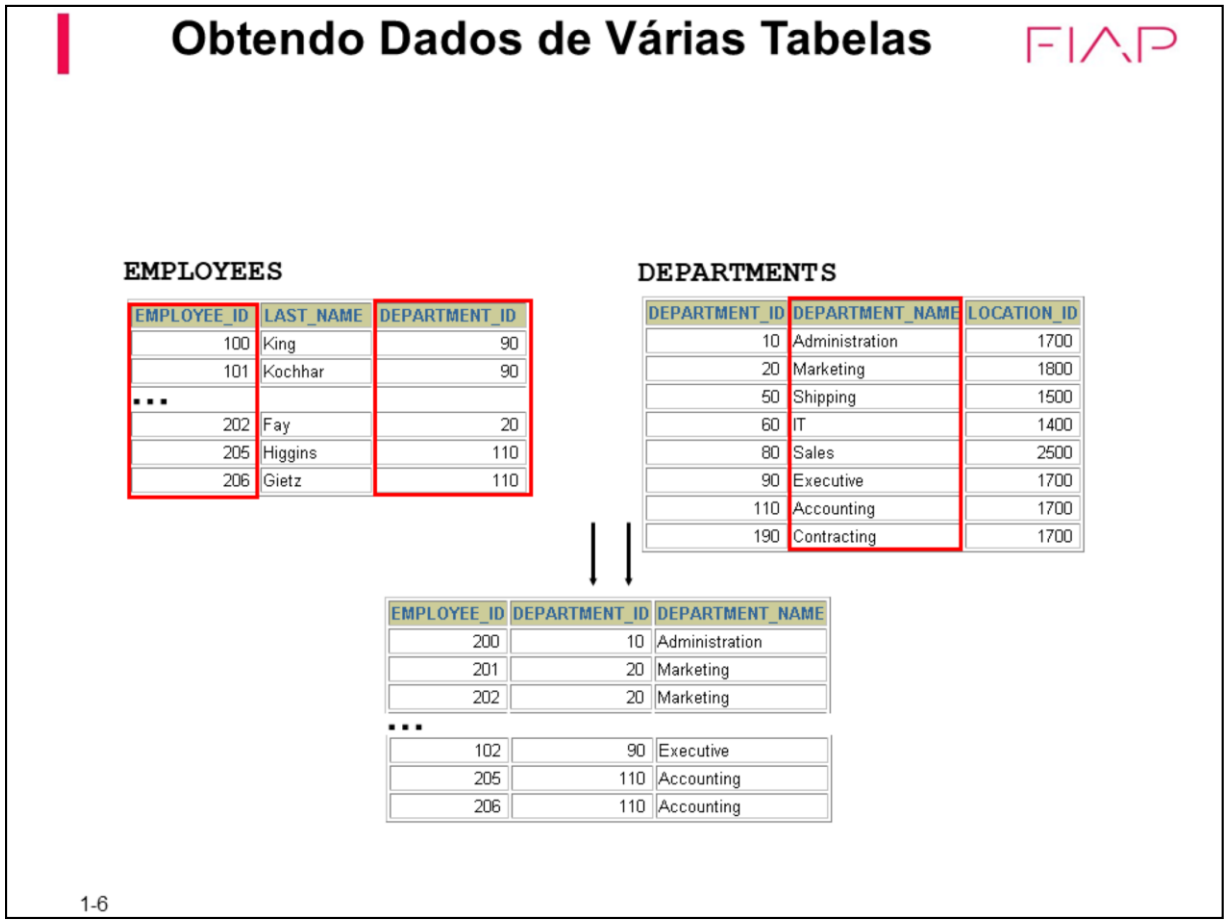
Exibindo Dados de Várias Tabelas



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*.



Obtendo 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.

Unindo Tabelas com a Sintaxe SQL:1999

Use uma join para consultar dados de mais de uma tabela:

```
SELECT   table1.column, table2.column
FROM     table1
[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];
```

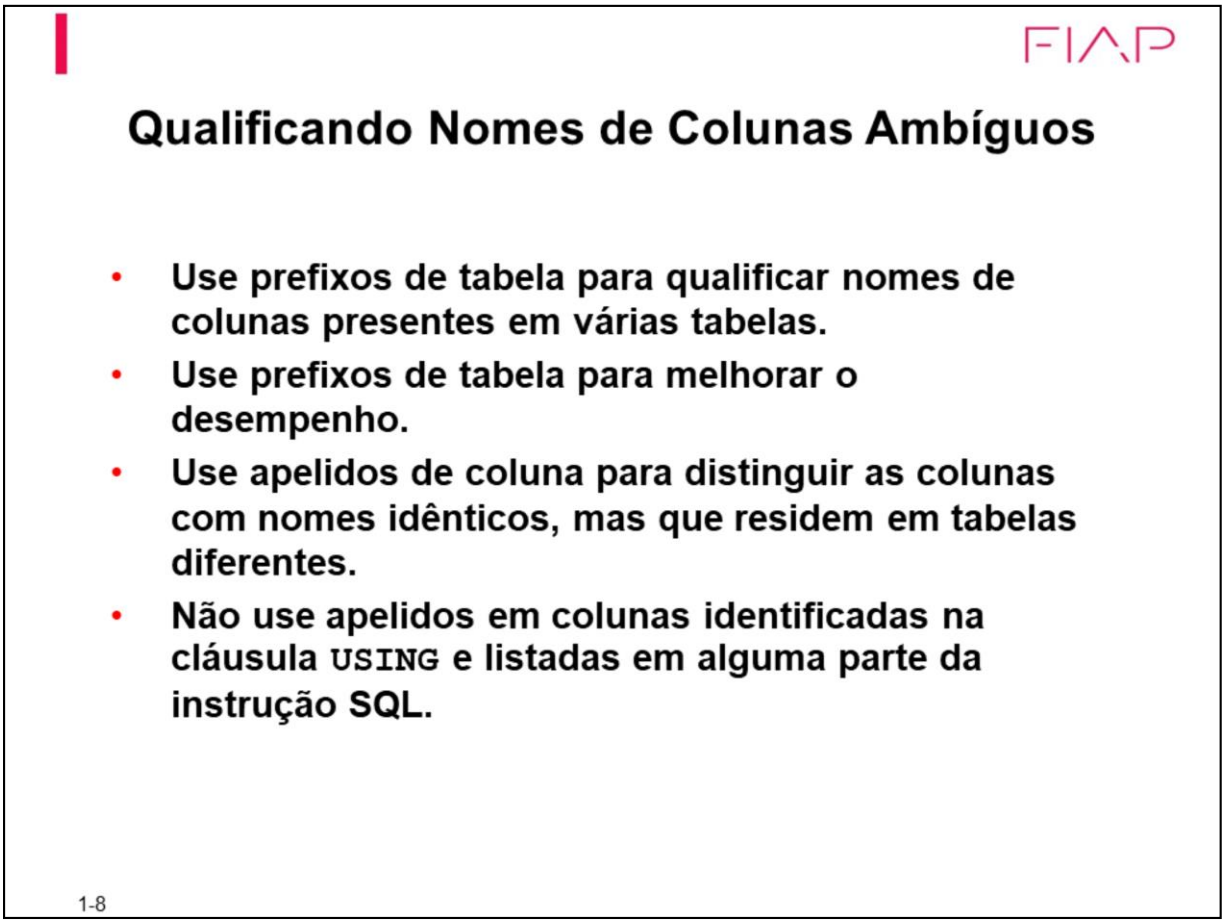
1-7

Definindo Joins

Na sintaxe:

- *table1.column* indica a tabela e a coluna das quais os dados são recuperados
- JOIN *table ON table1.column_name* executa uma operação de equijoin com base na condição da cláusula ON, = *table2.column_name*
- *LEFT/RIGHT/FULL OUTER* executa joins externas
- CROSS JOIN retorna um produto cartesiano das duas tabelas

Para obter mais informações, consulte "SELECT" no manual *Oracle SQL Reference*.



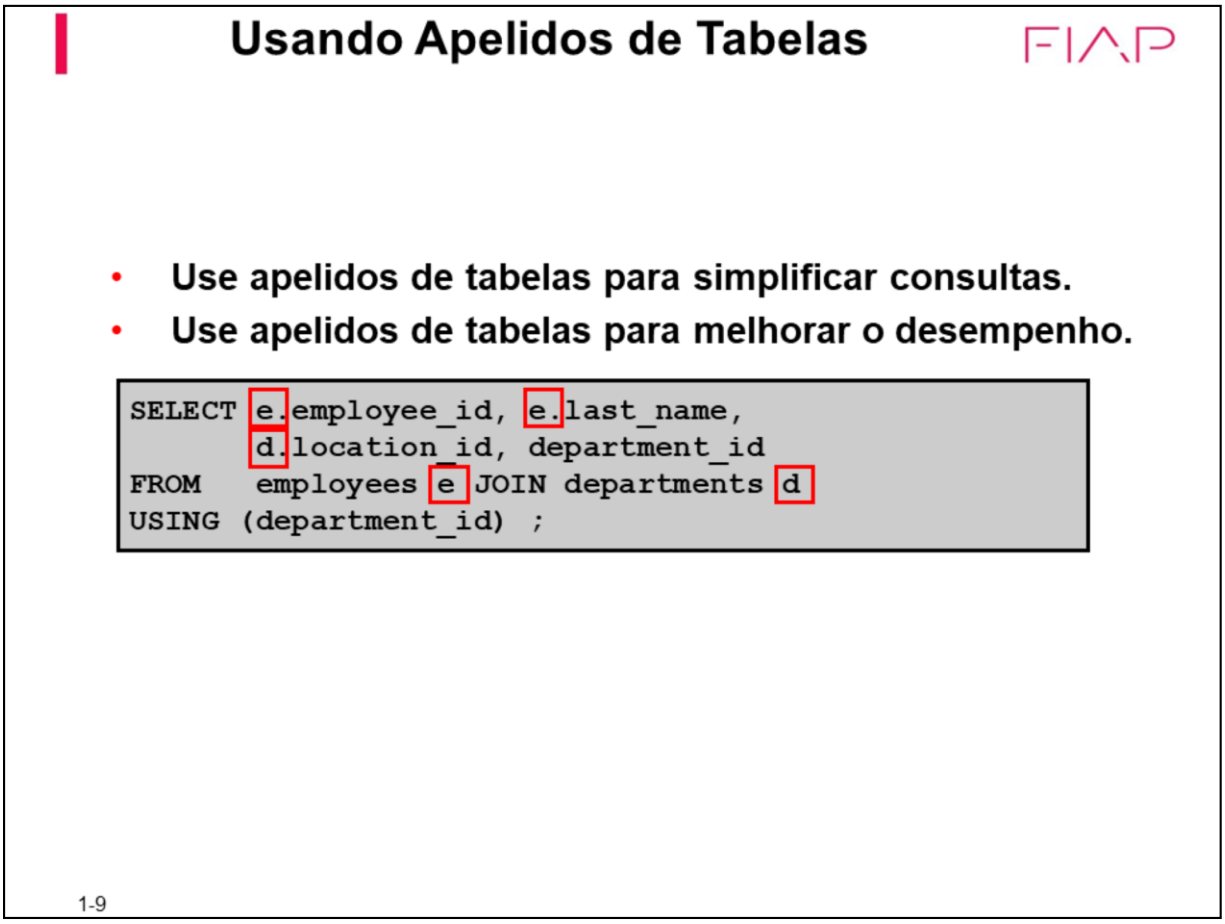
Qualificando Nomes de Colunas Ambíguos

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

```
SELECT employees.employee_id, employees.last_name,
       departments.department_id, departments.location_id
FROM   employees JOIN departments
ON     employees.department_id = departments.department_id;
```

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.

Observação: Ao efetuar uma operação de join com a cláusula `USING`, você não poderá qualificar uma coluna usada nessa própria cláusula. Além disso, se essa coluna for usada em alguma parte da instrução SQL, ela não poderá ser utilizada como apelido.



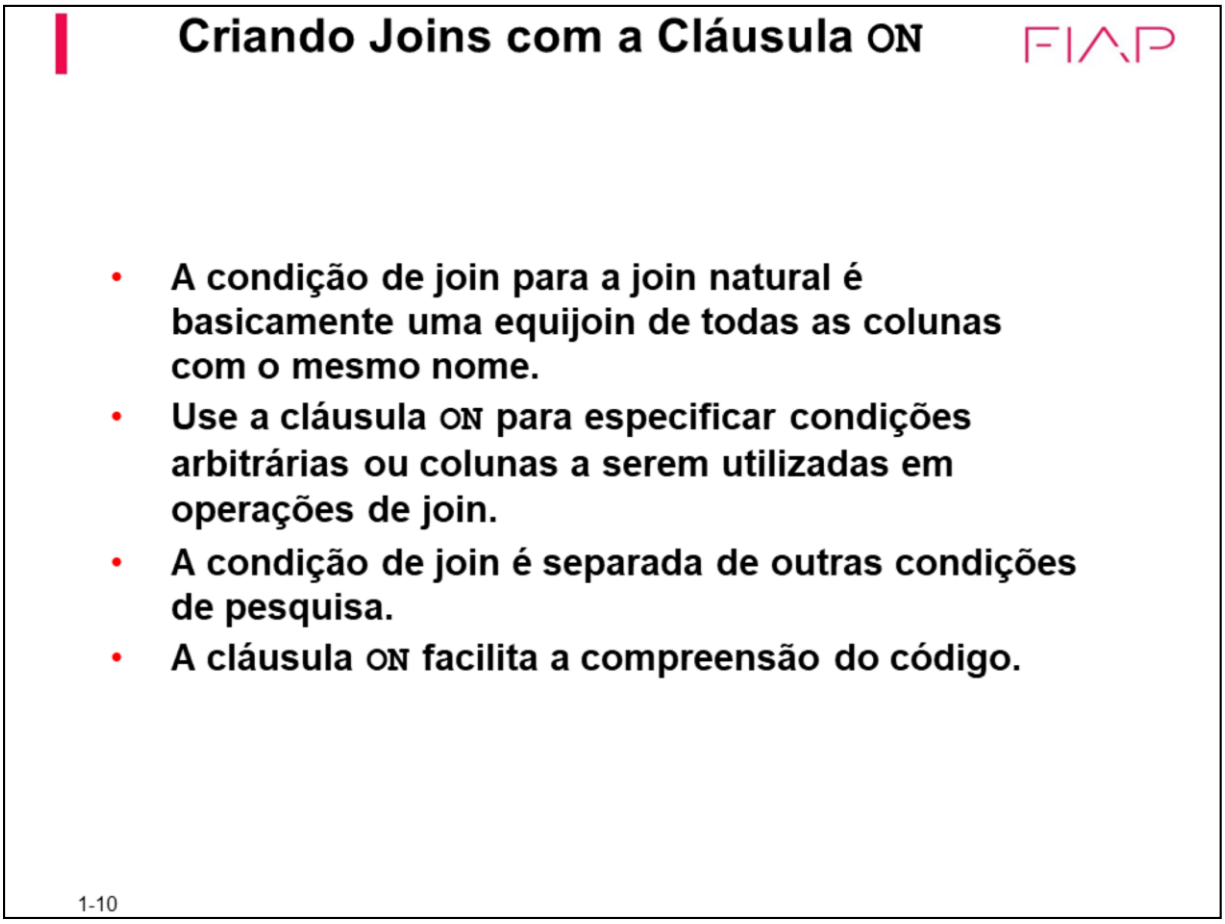
Usando Apelidos de Tabelas

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 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.
- O apelido de tabela é válido somente para a instrução `SELECT` atual.



Cláusula ON

Use a cláusula `ON` para especificar uma condição de join. Assim, você pode especificar condições de join separadas de condições de filtro e pesquisa na cláusula `WHERE`.

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	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

...

19 rows selected.

1-11

Criando Joins com a Cláusula ON

Neste exemplo, as colunas DEPARTMENT_ID das tabelas EMPLOYEES e DEPARTMENTS são unidas com a cláusula ON. Sempre que um ID de departamento na tabela EMPLOYEES for igual ao ID de departamento na tabela DEPARTMENTS, a linha será retornada.

Também é possível usar a cláusula ON para unir colunas com nomes distintos.

Auto-Joins Usando a Cláusula ON

FIAP

EMPLOYEES (WORKER)

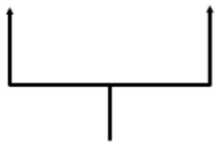
EMPLOYEE_ID	LAST_NAME	MANAGER_ID
100	King	
101	Kochhar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103
107	Lorentz	103
124	Mourgos	100

...

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.

1-12

Unindo uma Tabela a Ela Própria

Às vezes, é necessário unir uma tabela a ela própria. Para descobrir o nome do gerente de cada funcionário, você precisa unir a tabela `EMPLOYEES` a ela própria ou executar uma auto-join. Por exemplo, para descobrir 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 103 relativo a `MANAGER_ID`. Na segunda vez, você examinará a coluna `EMPLOYEE_ID` para localizar 103 e a coluna `LAST_NAME` para localizar Hunold.

Auto-Joins Usando a Cláusula ON

FIAP

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

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

...

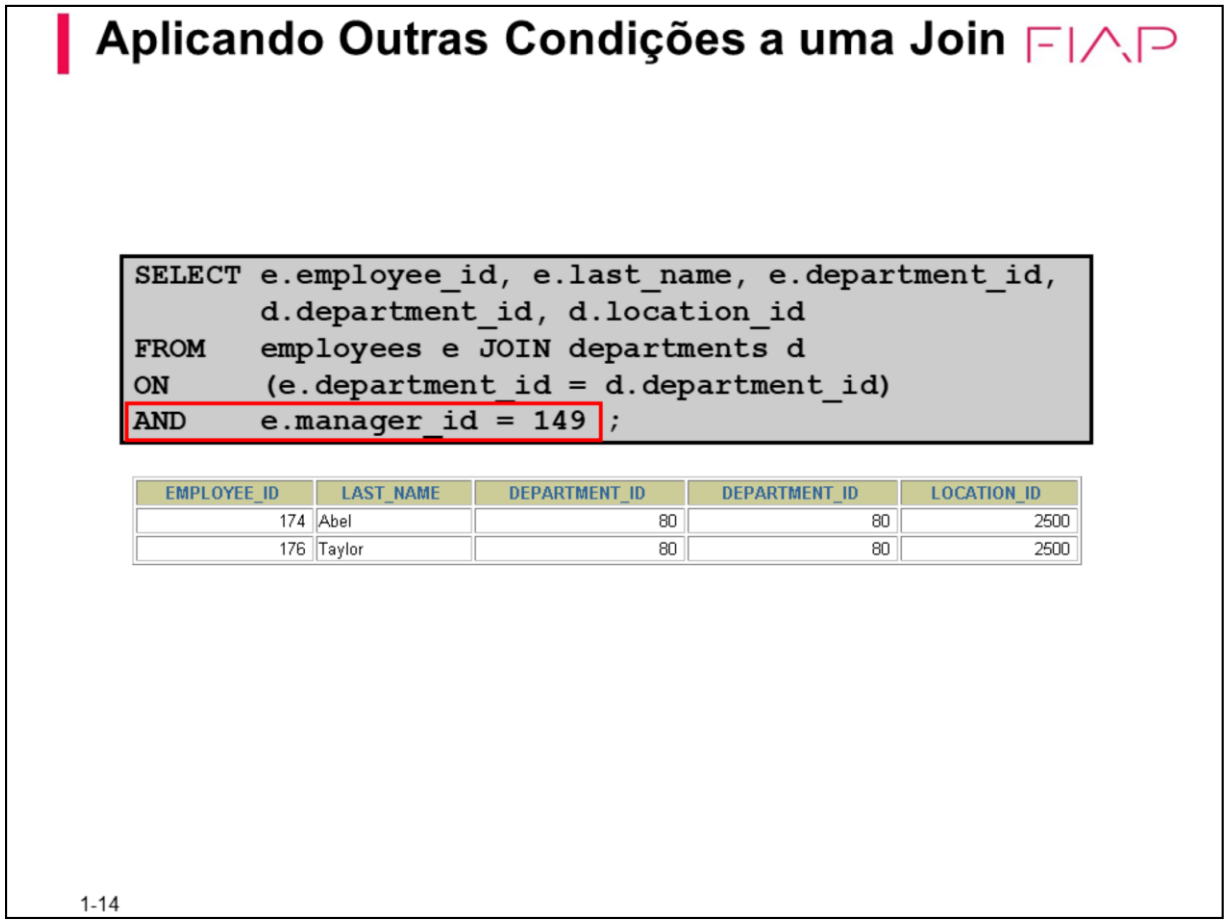
19 rows selected.

1-13

Unindo uma Tabela a Ela Própria (continuação)

Também é possível usar a cláusula `ON` para unir colunas com nomes distintos na mesma tabela ou em uma tabela diferente.

O exemplo mostrado é uma auto-join da tabela `EMPLOYEES`, com base nas colunas `EMPLOYEE_ID` e `MANAGER_ID`.



Aplicando Outras Condições a uma Join

Você pode aplicar outras condições à join.

O exemplo mostrado executa uma operação de join nas tabelas `EMPLOYEES` e `DEPARTMENTS`, além de exibir apenas os funcionários com o ID de gerente 149. Para adicionar outras condições à cláusula `ON`, especifique cláusulas `AND`. Como opção, você pode usar uma cláusula `WHERE` para aplicar outras condições:

```
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;
```

Criando Joins Tridimensionais com a Cláusula ON

FIAP

```
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
103	Southlake	IT
104	Southlake	IT
107	Southlake	IT
124	South San Francisco	Shipping
141	South San Francisco	Shipping
142	South San Francisco	Shipping
143	South San Francisco	Shipping
144	South San Francisco	Shipping

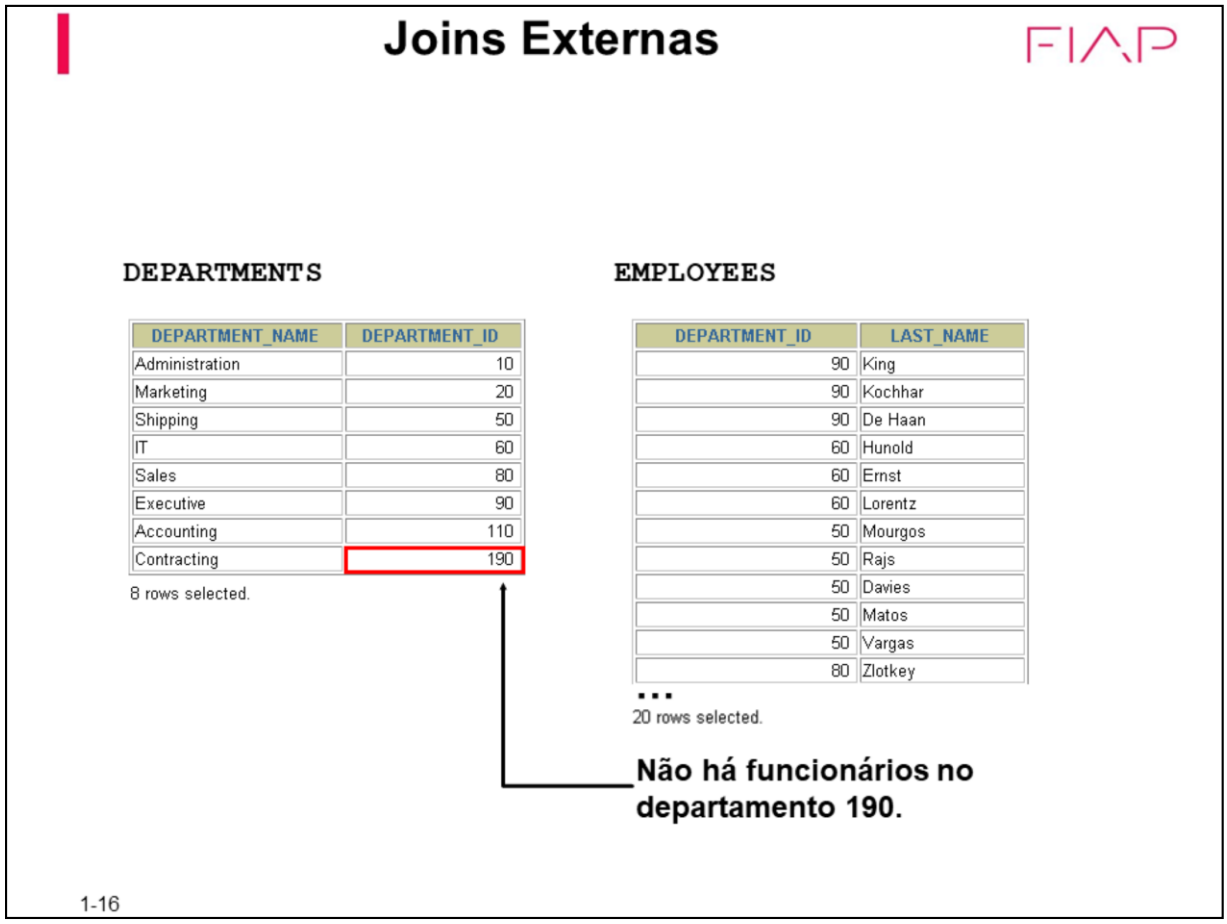
...

19 rows selected.

1-15

Joins Tridimensionais

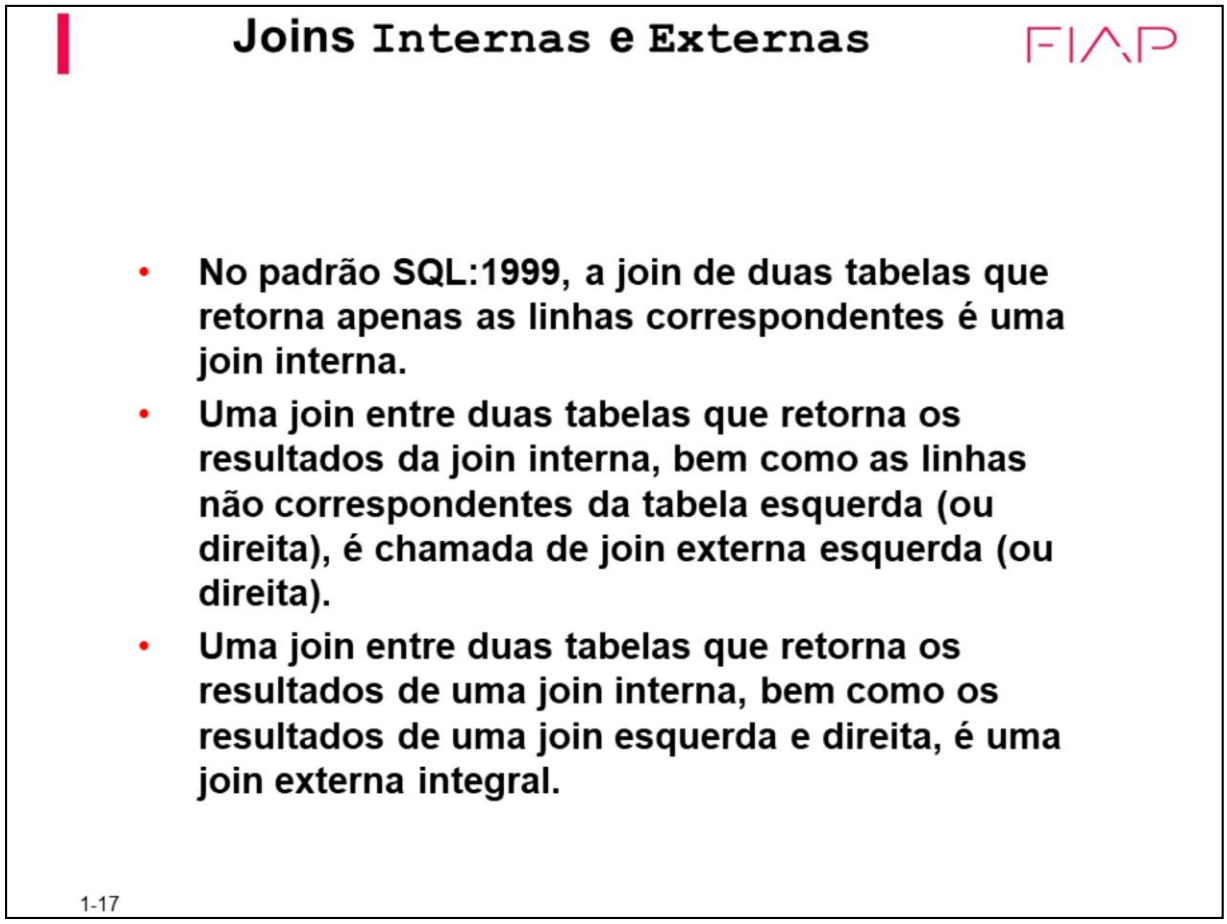
Uma join tridimensional é uma join de três tabelas. Na sintaxe compatível com o padrão SQL:1999, as joins são executadas da esquerda para a direita. Portanto, a primeira join a ser executada é `EMPLOYEES JOIN DEPARTMENTS`. A primeira condição de join pode fazer referência a colunas de `EMPLOYEES` e `DEPARTMENTS`, mas não a colunas de `LOCATIONS`. A segunda condição de join pode fazer referência a colunas de todas as três tabelas.



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 ID de departamento 190 não é exibido, pois não existem funcionários com esse ID registrado na tabela `EMPLOYEES`. Em vez de conter 20 funcionários, o conjunto de resultados conterá 19 registros.

Para retornar o registro de um departamento sem funcionários, use uma join externa.

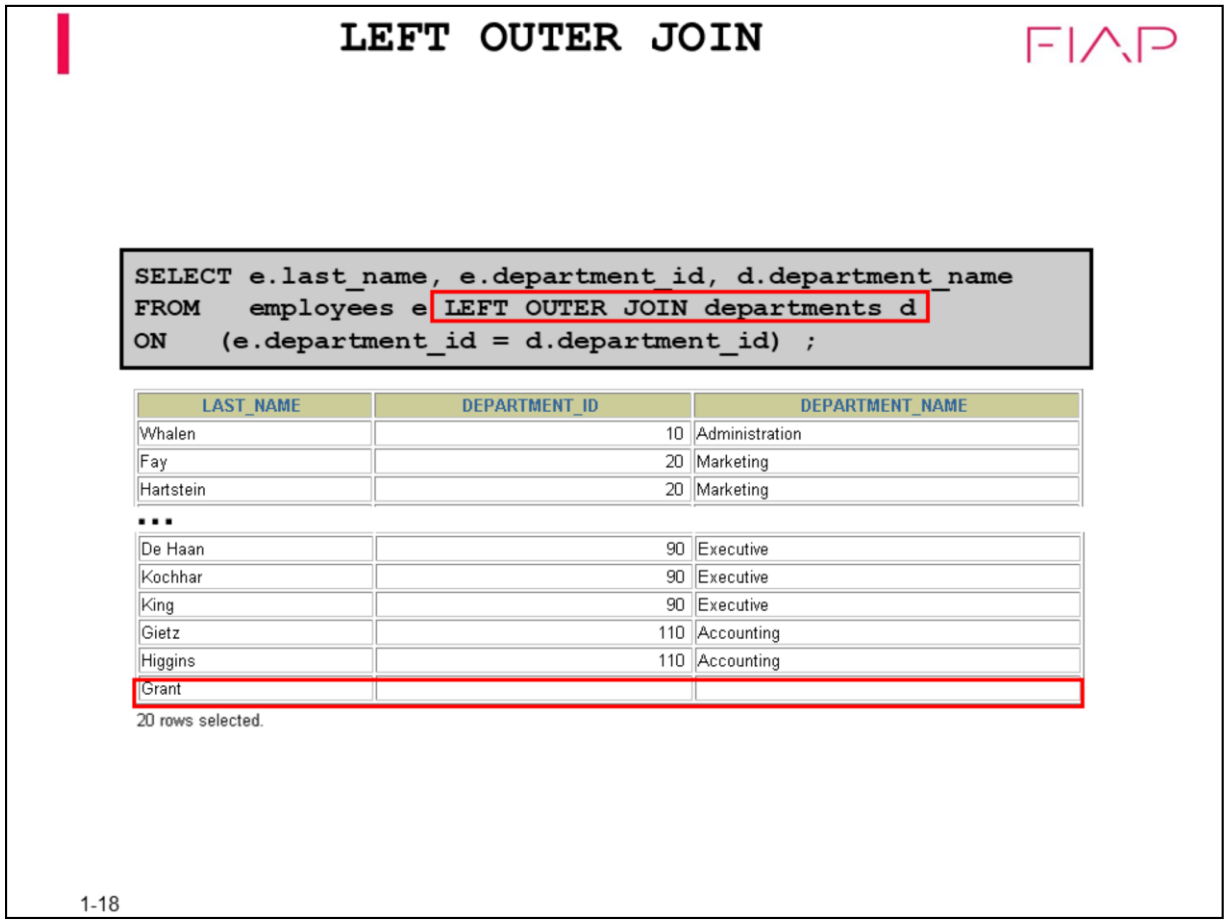


Joins Internas e Externas

A união de tabelas com as cláusulas `NATURAL JOIN`, `USING` ou `ON` resulta em uma join interna. As linhas não correspondentes não são exibidas na saída. Para retornar as linhas não correspondentes, use uma join externa. Uma join externa retorna todas as linhas que atendem à condição de join, bem como algumas ou todas as linhas de uma tabela para as quais nenhuma linha da outra tabela atende à condição de join.

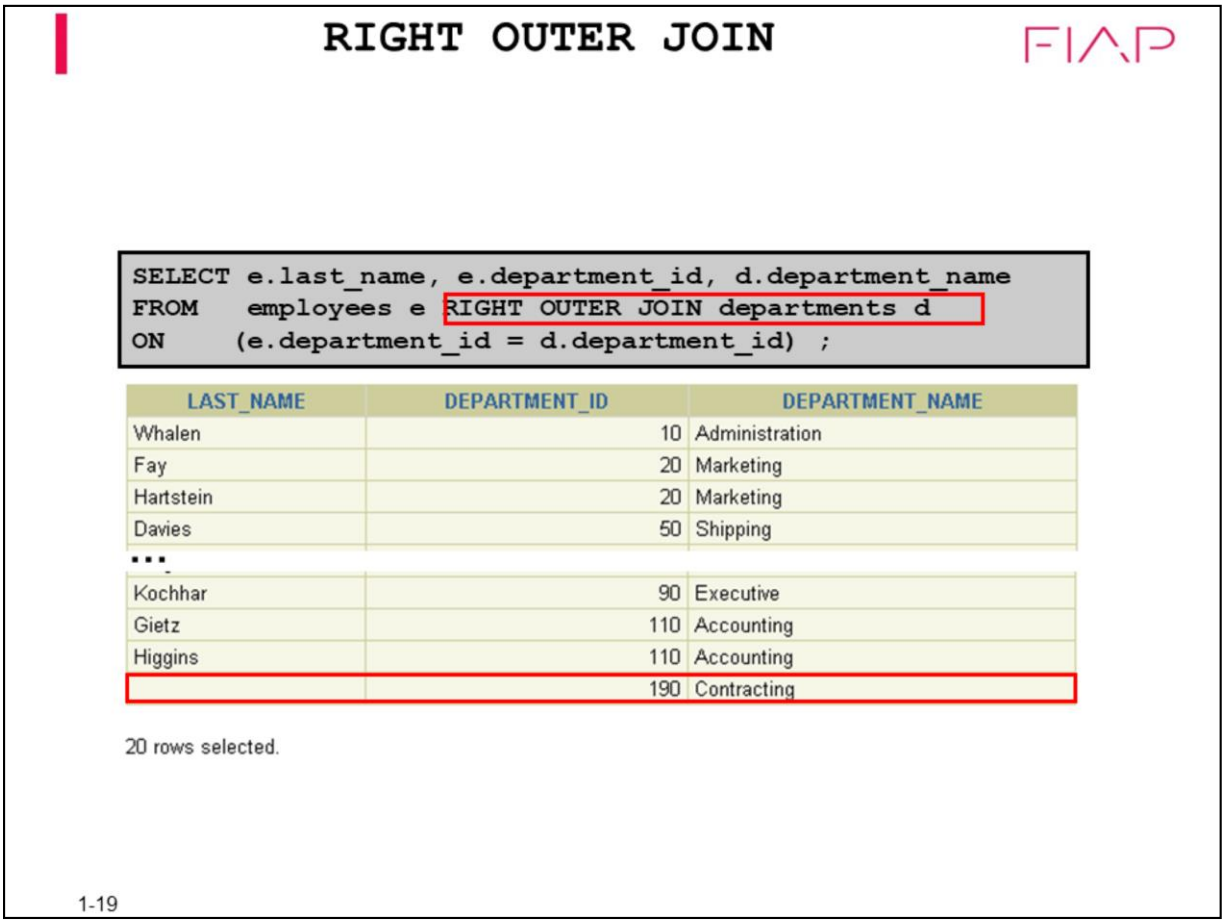
Há três tipos de joins externas:

- Externa Esquerda `(LEFT OUTER JOIN)`
- Externa Direita `(RIGHT OUTER JOIN)`
- Externa Integral `(FULL OUTER JOIN)`



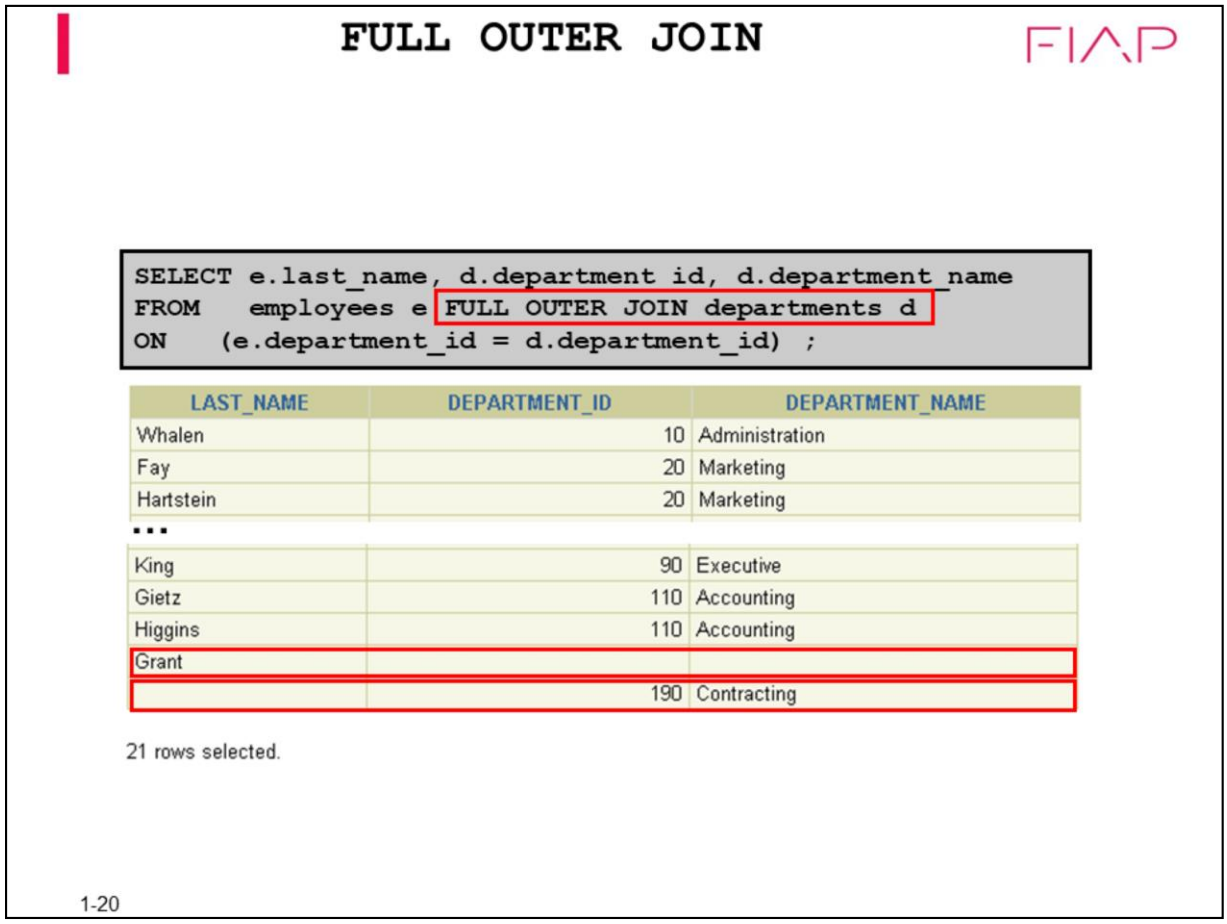
Exemplo de Join Externa Esquerda (`LEFT OUTER JOIN`)

Esta consulta recupera todas as linhas da tabela `EMPLOYEES`, que é a tabela esquerda, mesmo quando não há correspondência na tabela `DEPARTMENTS`.



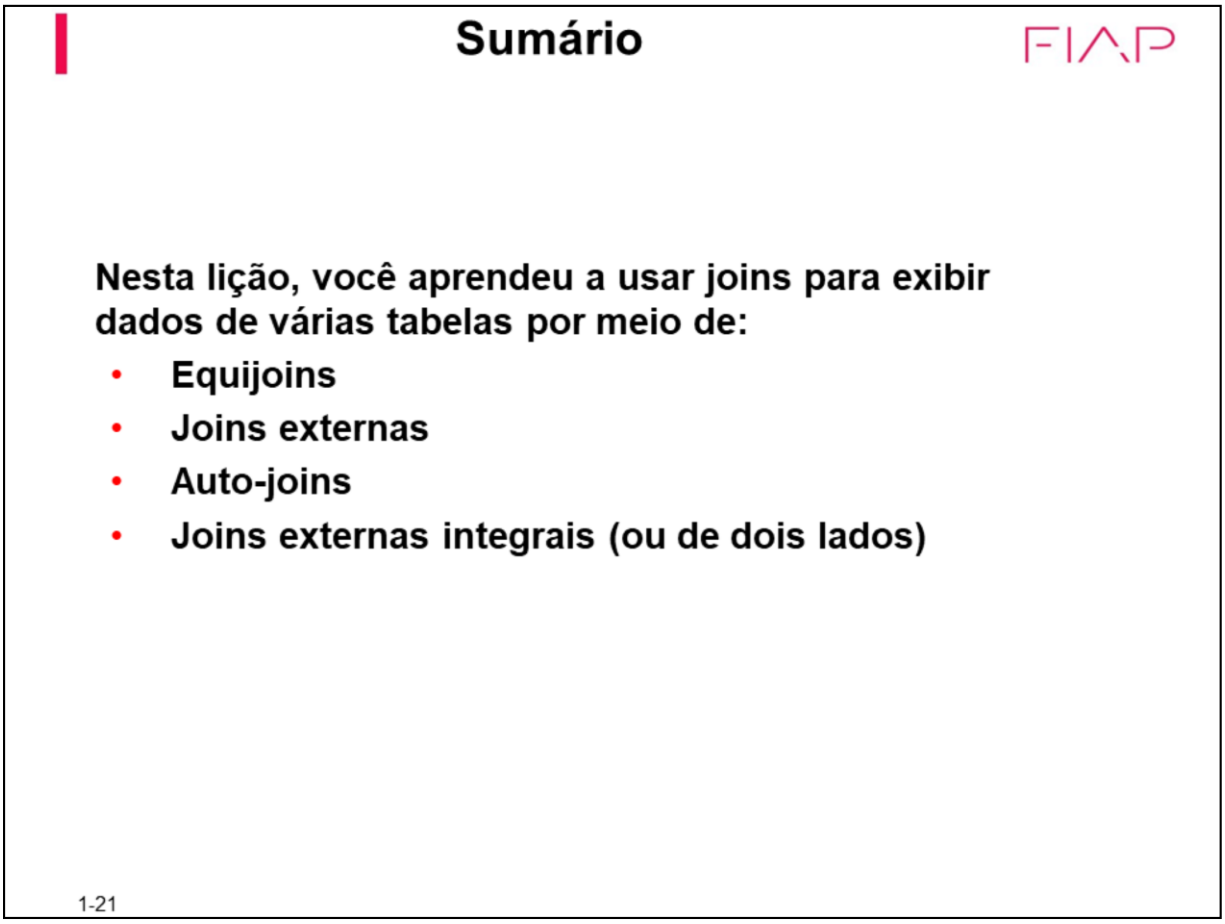
Exemplo de Join Externa Direita (`RIGHT OUTER JOIN`)

Esta consulta recupera todas as linhas da tabela `DEPARTMENTS`, que é a tabela direita, mesmo quando não há correspondência na tabela `EMPLOYEES`.



Exemplo de Join Externa Integral (`FULL OUTER JOIN`)

Esta consulta recupera todas as linhas da tabela `EMPLOYEES`, mesmo quando não há correspondência na tabela `DEPARTMENTS`. Ela também recupera todas as linhas da tabela `DEPARTMENTS`, mesmo quando não há correspondência na tabela `EMPLOYEES`.



Sumário

Há várias maneiras de unir tabelas.

Tipos de Join

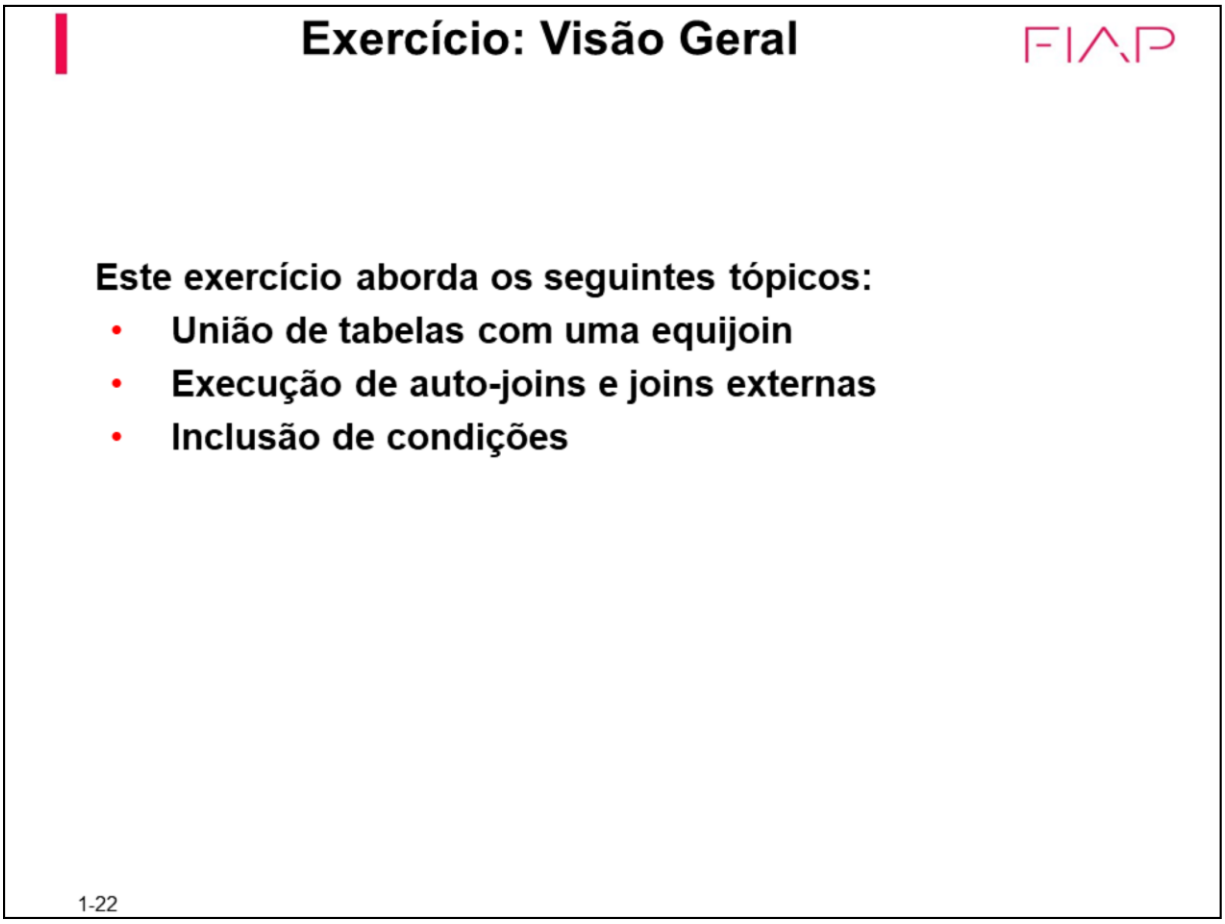
- Equijoins
- Não-equijoins
- Joins externas
- Auto-joins
- Joins cruzadas
- Joins naturais
- Joins externas integrais (ou de dois lados)

Produtos Cartesianos

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

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: Visão Geral

Este exercício tem como objetivo proporcionar a você um treinamento prático de como extrair dados de mais de uma tabela com joins compatíveis com o padrão SQL:1999.

Exercício

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.

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. O departamento de recursos humanos precisa de um relatório de todos os funcionários. Crie uma consulta para exibir o sobrenome, o número do departamento e 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 (continuação)

3. O departamento de recursos humanos precisa de um relatório dos funcionários 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. Crie um relatório para exibir o sobrenome e o número dos funcionários, bem como o 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_05_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 (continuação)

5. Modifique `lab_05_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_05_05.sql`. Execute a consulta em `lab_05_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_05_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 (continuação)

7. O departamento de recursos humanos deseja determinar os nomes de todos os funcionários admitidos após Davies. Crie uma consulta para exibir o nome e a data 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.

8. O departamento de recursos humanos precisa obter os nomes e as datas de admissão de todos os funcionários admitidos antes dos respectivos gerentes, além dos nomes e das datas de admissão desses gerentes. Salve o script no arquivo `lab5_08.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.

Bibliografia Utilizada

FIAP

Database SQL Language Reference: http://docs.oracle.com/database/121/SQLRF/toc.htm

Manuais Oracle – Oracle Database 12c: SQL Workshop I/II