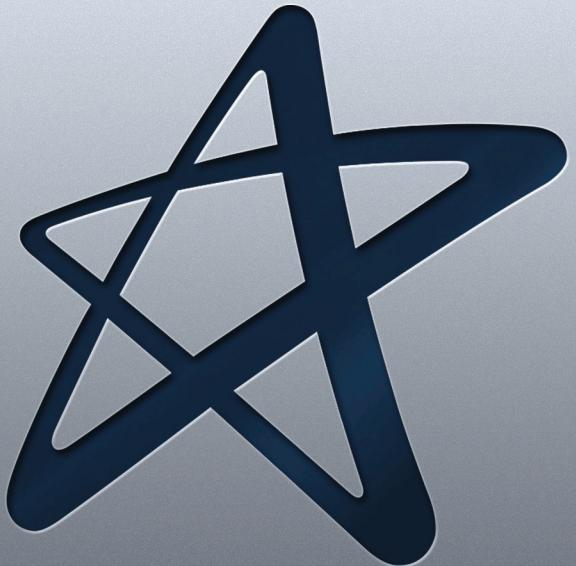


Linguagem de Banco de Dados



Material Teórico



Subquery e Funções de Decisão

Responsável pelo Conteúdo:

Prof. Ms. Luis Carlos Reis

Revisão Textual:

Prof. Ms. Luciano Vieira Francisco

UNIDADE

Subquery e Funções de Decisão



- Introdução
- Subpesquisas Aninhadas
- Operador IN
- Operador ANY
- Operador ALL
- Operador EXISTS
- Funções de Decisão
- CASE



OBJETIVO DE APRENDIZADO

- Aplicar consultas que seriam difíceis ou impossíveis de serem realizadas utilizando somente uma *query*, resolvendo-as com instruções *SELECT*.
- Executar instruções que estarão dentro de outra instrução SQL.
- Aprender funções de decisão dentro de uma instrução *SELECT*.



Orientações de estudo

Para que o conteúdo desta Disciplina seja bem aproveitado e haja uma maior aplicabilidade na sua formação acadêmica e atuação profissional, siga algumas recomendações básicas:



Assim:

- ✓ Organize seus estudos de maneira que passem a fazer parte da sua rotina. Por exemplo, você poderá determinar um dia e horário fixos como o seu “momento do estudo”.
- ✓ Procure se alimentar e se hidratar quando for estudar, lembre-se de que uma alimentação saudável pode proporcionar melhor aproveitamento do estudo.
- ✓ No material de cada Unidade, há leituras indicadas. Entre elas: artigos científicos, livros, vídeos e sites para aprofundar os conhecimentos adquiridos ao longo da Unidade. Além disso, você também encontrará sugestões de conteúdo extra no item **Material Complementar**, que ampliarão sua interpretação e auxiliarão no pleno entendimento dos temas abordados.
- ✓ Após o contato com o conteúdo proposto, participe dos debates mediados em fóruns de discussão, pois irão auxiliar a verificar o quanto você absorveu de conhecimento, além de propiciar o contato com seus colegas e tutores, o que se apresenta como rico espaço de troca de ideias e aprendizagem.

Introdução

Uma subconsulta – ou como é mais conhecida, *subquery* – é uma instrução *SELECT* que está condicionada dentro de outra instrução SQL – podendo ser *SELECT*, *INSERT*, *UPDATE* ou *DELETE*. Como resultado dessa operação, pode-se fazer uso de subconsultas para criarmos consultas que seriam difíceis ou impossíveis de serem realizadas utilizando somente uma *query*. Assim, desde que consigamos codificar as nossas instruções *SELECT*, saberemos como codificar uma subconsulta, já que esta é apenas uma instrução SQL no interior de outra instrução SQL.

Em suma, uma *subquery* é um comando *SELECT* dentro de outro comando *SELECT* que retorna uma ou mais linhas a fim de satisfazer a uma cláusula *WHERE*.

A *subquery* – *inner query* – é executada antes da consulta principal e o seu resultado é, então, avaliado pelo da *query* principal – *outer query*.



Figura 1

Uma subpesquisa – *subquery* – é uma declaração *SELECT* que é aninhada com outra declaração *SELECT*, a qual retorna resultados intermediários.

Vejamos a estrutura de uma *subquery*:



Figura 2

Pré-Requisito para Praticar Nesta Unidade

Como utilizaremos o software *SQLDeveloper* para exemplificar os seguintes comandos, precisaremos da conexão *AulaHR*, a qual foi configurada no tutorial de instalação do software.

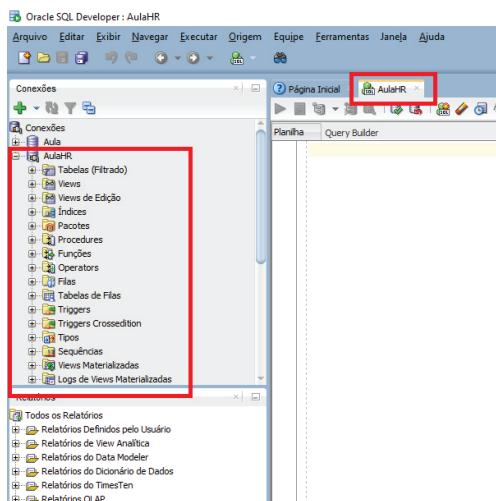


Figura 3

Exemplo 1

Quais funcionários possuem salário maior que o de Abel?

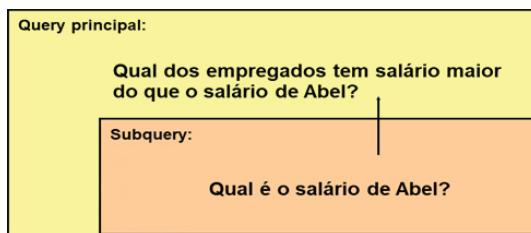


Figura 4

```
SELECT last_name, salary
FROM employees
WHERE salary >
      (SELECT salary
       FROM employees
       WHERE last_name = 'Abel');
```

Figura 5



Figura 6

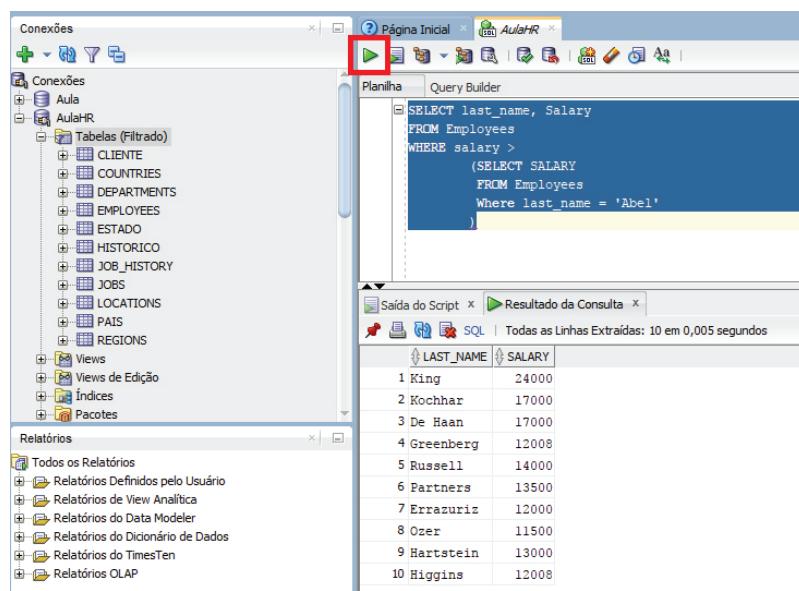


Figura 7



Importante!

Para executar mais de uma linha, selecione-as e depois execute.

Exemplo 2

Mostre o sobrenome – *last_name* – e o cargo – *job_id* – dos funcionários que possuem o mesmo cargo do funcionário de número 141.

Note que primeiramente deveremos descobrir o cargo do empregado com código 141. Neste caso, precisaremos da *subquery*.

```
①
SELECT last_name, job_id
FROM employees
WHERE job_id = (SELECT job_id
                 FROM employees
                 WHERE employee_id = 141);
```

Figura 8

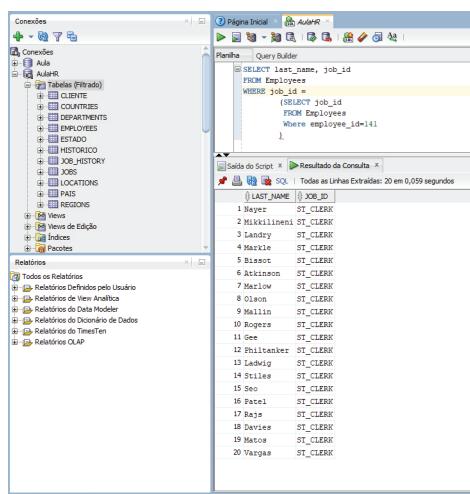


Figura 9

Exemplo 3

Para encontrar todos os empregados que têm o mesmo cargo de David Lee, devemos proceder da seguinte maneira:

```
SELECT FIRST_NAME, LAST_NAME, JOB_ID
FROM EMPLOYEES
```

```
WHERE JOB_ID = (SELECT JOB_ID
```

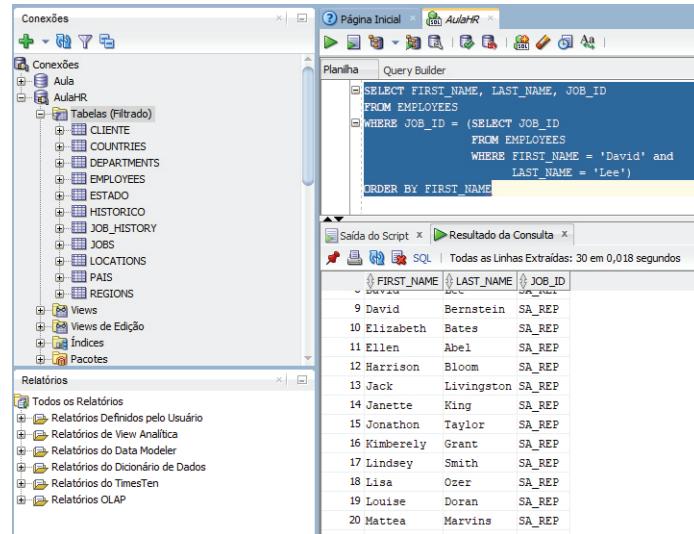
```
FROM EMPLOYEES
```

```
WHERE FIRST_NAME = 'David' and
```

```
LAST_NAME = 'Lee')
```

```
ORDER BY FIRST_NAME
```

A pesquisa interna retorna o cargo de David Lee (SA_REP) o qual é usado na condição *WHERE* da pesquisa principal.



```

SELECT FIRST_NAME, LAST_NAME, JOB_ID
FROM EMPLOYEES
WHERE JOB_ID = (SELECT JOB_ID
                 FROM EMPLOYEES
                 WHERE FIRST_NAME = 'David' and
                       LAST_NAME = 'Lee')
ORDER BY FIRST_NAME
  
```

FIRST_NAME	LAST_NAME	JOB_ID
9 David	Bernstein	SA_REP
10 Elizabeth	Bates	SA_REP
11 Ellen	Abel	SA_REP
12 Harrison	Bloom	SA_REP
13 Jack	Livingston	SA_REP
14 Janette	King	SA_REP
15 Jonathon	Taylor	SA_REP
16 Kimberly	Grant	SA_REP
17 Lindsey	Smith	SA_REP
18 Lisa	Oser	SA_REP
19 Louise	Doran	SA_REP
20 Mattea	Marvins	SA_REP

Figura 10

Exemplo 4

Para encontrar todos os empregados do mesmo departamento de David Lee, devemos proceder da seguinte maneira:

```
SELECT FIRST_NAME, LAST_NAME, JOB_ID, DEPARTMENT_ID
```

```
FROM EMPLOYEES
```

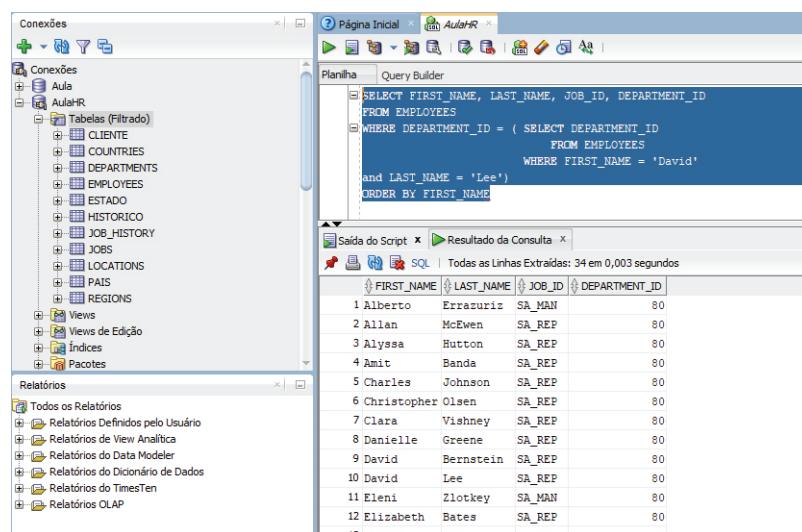
```
WHERE DEPARTMENT_ID = ( SELECT DEPARTMENT_ID
```

```
FROM EMPLOYEES
```

```
WHERE FIRST_NAME = 'David'
```

and LAST_NAME = 'Lee')

```
ORDER BY FIRST_NAME
```



```

SELECT FIRST_NAME, LAST_NAME, JOB_ID, DEPARTMENT_ID
FROM EMPLOYEES
WHERE DEPARTMENT_ID = ( SELECT DEPARTMENT_ID
                         FROM EMPLOYEES
                         WHERE FIRST_NAME = 'David'
                           and LAST_NAME = 'Lee')
ORDER BY FIRST_NAME
  
```

FIRST_NAME	LAST_NAME	JOB_ID	DEPARTMENT_ID
1 Alberto	Erasuriz	SA_MAN	80
2 Allan	McEwen	SA_REP	80
3 Alyssa	Hutton	SA_REP	80
4 Amit	Banda	SA_REP	80
5 Charles	Johnson	SA_REP	80
6 Christopher	Olsen	SA_REP	80
7 Clara	Vishney	SA_REP	80
8 Danielle	Greene	SA_REP	80
9 David	Bernstein	SA_REP	80
10 David	Lee	SA_REP	80
11 Eleni	Zlotkey	SA_MAN	80
12 Elizabeth	Bates	SA_REP	80

Figura 11

Exemplo 5

Para encontrar os empregados que ganham acima da média salarial, devemos proceder da seguinte maneira:

```
SELECT FIRST_NAME, LAST_NAME, JOB_ID,
```

```
DEPARTMENT_ID, SALARY
```

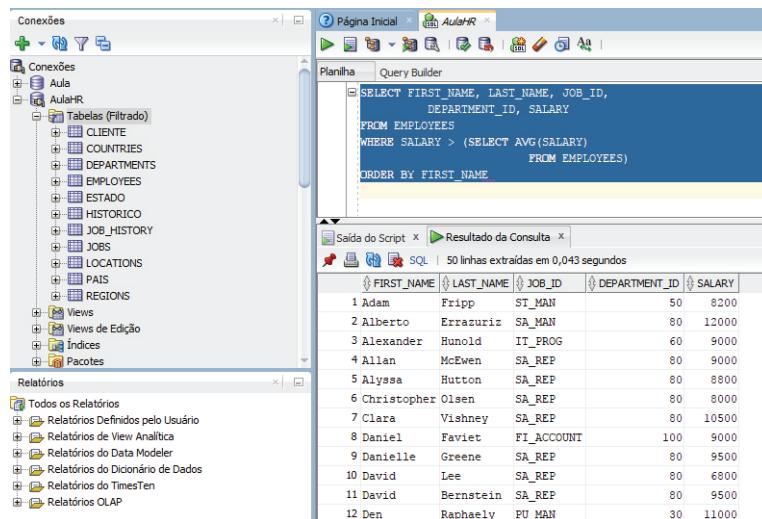
```
FROM EMPLOYEES
```

```
WHERE SALARY > (SELECT AVG(SALARY)
```

```
FROM EMPLOYEES)
```

```
ORDER BY FIRST_NAME
```

 Single Row Subquery
– Retorno de um único valor na subquerie.



The screenshot shows the Oracle SQL Developer interface. On the left, the 'Conexões' (Connections) sidebar shows a connection named 'AulaHR'. Below it, the 'Tabelas (Filtrado)' (Tables Filtered) section lists various tables from the 'AulaHR' schema, including CLIENTE, COUNTRIES, DEPARTMENTS, EMPLOYEES, ESTADO, HISTORICO, JOB_HISTORY, JOBS, LOCATIONS, PAIS, and REGIONS. The 'Relatórios' (Reports) section shows a list of reports defined by the user. In the center, the 'Planilha' (Worksheet) tab displays the following SQL query:

```
SELECT FIRST_NAME, LAST_NAME, JOB_ID,
       DEPARTMENT_ID, SALARY
  FROM EMPLOYEES
 WHERE SALARY > (SELECT AVG(SALARY)
                  FROM EMPLOYEES)
 ORDER BY FIRST_NAME
```

Below the worksheet, the 'Saída do Script' (Script Output) tab shows the results of the query, which is a table with columns FIRST_NAME, LAST_NAME, JOB_ID, DEPARTMENT_ID, and SALARY. The results are as follows:

FIRST_NAME	LAST_NAME	JOB_ID	DEPARTMENT_ID	SALARY
1 Adam	Fripp	ST_MAN	50	8200
2 Alberto	Errazuriz	SA_MAN	80	12000
3 Alexander	Hunold	IT_PROG	60	9000
4 Allan	McEwen	SA_REP	80	9000
5 Alyssa	Hutton	SA_REP	80	8800
6 Christopher	Olsen	SA_REP	80	8000
7 Clara	Vishney	SA_REP	80	10500
8 Daniel	Faviet	FI_ACCOUNT	100	9000
9 Danielle	Greene	SA_REP	80	9500
10 David	Lee	SA_REP	80	6800
11 David	Bernstein	SA_REP	80	9500
12 Den	Raphaely	PU_MAN	30	11000

Figura 12

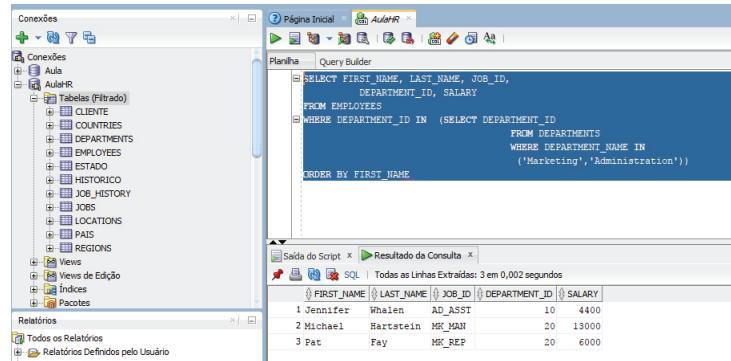
Exemplo 6

Para encontrarmos os empregados que trabalham nos departamentos de Marketing e Administração, devemos proceder da seguinte maneira:

```
SELECT FIRST_NAME, LAST_NAME, JOB_ID,
       DEPARTMENT_ID, SALARY
  FROM EMPLOYEES
 WHERE DEPARTMENT_ID IN (SELECT DEPARTMENT_ID
                           FROM DEPARTMENTS
                          WHERE DEPARTMENT_NAME IN
                                ('Marketing', 'Administration'))
```

```
ORDER BY FIRST_NAME
```

 Multiple Row Subquery
– Retorno de vários valores na subquerie



```

Conexões
  + Conexões
    + Aula
      + AulaHR
        + Tabelas (Filtrado)
          + CLIENTE
          + COUNTRIES
          + DEPARTMENTS
          + EMPLOYEES
          + ESTADO
          + HISTORICO
          + JOB_HISTORY
          + JOBS
          + LOCATIONS
          + PAIS
          + REGIONS
        + Views
        + Views de Edição
        + Índices
        + Pacotes
      + Relatórios
        + Todos os Relatórios
        + Relatórios Definidos pelo Usuário
      + Relatórios de Visão Analítica

Página Inicial | AulaHR | Planilha | Query Builder | Saída do Script | Resultado da Consulta | Ajuda | Ferramentas | Sobre | Sair

SELECT FIRST_NAME, LAST_NAME, JOB_ID,
       DEPARTMENT_ID, SALARY
  FROM EMPLOYEES
 WHERE DEPARTMENT_ID IN (SELECT DEPARTMENT_ID
                           FROM DEPARTMENTS
                          WHERE DEPARTMENT_NAME IN
                                ('Marketing', 'Administration'))
 ORDER BY FIRST_NAME

Saída do Script | Resultado da Consulta | SQL | Todas as Linhas Extraídas: 3 em 0,002 segundos

FIRST_NAME | LAST_NAME | JOB_ID | DEPARTMENT_ID | SALARY |
1 Jennifer Whalen AD_ASST 10 4400
2 Michael Hartstein MK_MAN 20 13000
3 Pat Fury MR_REP 20 6000
  
```

Figura 13

Subpesquisas Aninhadas

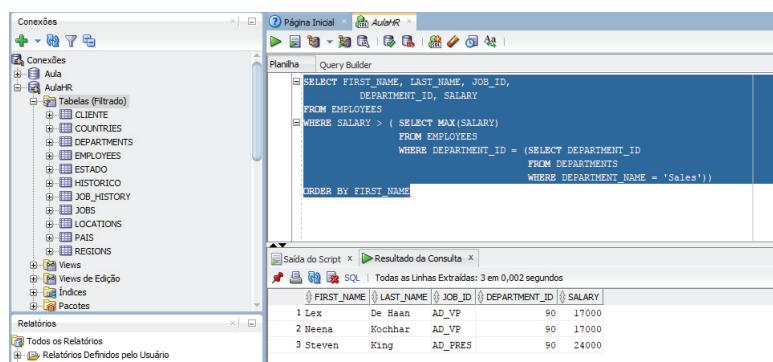
Muitas vezes é necessário envolver vários níveis de *subquery's*, como veremos adiante.

Exemplo 7

Mostrar o primeiro e último nome, cargo, código do departamento e salário para os empregados cujo salário é maior que o maior salário no Departamento *Sales*:

```

SELECT FIRST_NAME, LAST_NAME, JOB_ID,
       DEPARTMENT_ID, SALARY
  FROM EMPLOYEES
 WHERE SALARY > ( SELECT MAX(SALARY)
   FROM EMPLOYEES
  WHERE DEPARTMENT_ID = (SELECT DEPARTMENT_ID
   FROM DEPARTMENTS
  WHERE DEPARTMENT_NAME = 'Sales'))
 ORDER BY FIRST_NAME
  
```



```

Conexões
  + Conexões
    + Aula
      + AulaHR
        + Tabelas (Filtrado)
          + CLIENTE
          + COUNTRIES
          + DEPARTMENTS
          + EMPLOYEES
          + ESTADO
          + HISTORICO
          + JOB_HISTORY
          + JOBS
          + LOCATIONS
          + PAIS
          + REGIONS
        + Views
        + Views de Edição
        + Índices
        + Pacotes
      + Relatórios
        + Todos os Relatórios
        + Relatórios Definidos pelo Usuário
      + Relatórios de Visão Analítica

Página Inicial | AulaHR | Planilha | Query Builder | Saída do Script | Resultado da Consulta | SQL | Todas as Linhas Extraídas: 3 em 0,002 segundos

SELECT FIRST_NAME, LAST_NAME, JOB_ID,
       DEPARTMENT_ID, SALARY
  FROM EMPLOYEES
 WHERE SALARY > ( SELECT MAX(SALARY)
   FROM EMPLOYEES
  WHERE DEPARTMENT_ID = (SELECT DEPARTMENT_ID
   FROM DEPARTMENTS
  WHERE DEPARTMENT_NAME = 'Sales'))
 ORDER BY FIRST_NAME

Saída do Script | Resultado da Consulta | SQL | Todas as Linhas Extraídas: 3 em 0,002 segundos

FIRST_NAME | LAST_NAME | JOB_ID | DEPARTMENT_ID | SALARY |
1 Lex De Haan AD_VP 90 17000
2 Neena Kochhar AD_VP 90 17000
3 Steven King AD_PRES 90 24000
  
```

Figura 14

Quadro 1 – Operadores de comparação para múltiplas linhas

Operador	Significado
IN	Igual a qualquer membro da lista
ANY	Compara o valor com cada valor retornado pela <i>subquery</i>
ALL	Compara o valor com todos os valores retornados pela <i>subquery</i>
EXISTS	Verifica se um valor existe

Fonte: elaborado pelo professor confeudista

Operador IN

A seguinte pesquisa se propõe a encontrar os empregados que trabalham nos departamentos de Marketing e Administração.

Perceba que a *subquery* retorna uma lista de valores, portanto, neste caso devemos utilizar o operador *IN*:

```
SELECT FIRST_NAME, LAST_NAME, JOB_ID,
       DEPARTMENT_ID, SALARY
  FROM EMPLOYEES
 WHERE DEPARTMENT_ID IN (SELECT DEPARTMENT_ID)
                           FROM DEPARTMENTS
                           WHERE DEPARTMENT_NAME IN
                               ('Marketing', 'Administration'))
```



Multiple Row *Subquery*
– Retorno de vários valores na subquerie

```
ORDER BY FIRST_NAME
```

FIRST_NAME	LAST_NAME	JOB_ID	DEPARTMENT_ID	SALARY
Jennifer	Whalen	AD_ASST	10	4400
Michael	Hartstein	MK_MAN	20	13000
Pat	Pay	MK_REP	20	6000

Figura 15

Operador ANY

Compara o valor com cada valor retornado pela *subquery*, onde:

- $< \text{ANY}$ corresponde ao menor que o maior valor;
- $> \text{ANY}$ corresponde ao maior que o menor valor.

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary < ANY
      9000,6000,4800,4200
      (SELECT salary
       FROM employees
       WHERE job_id = 'IT_PROG')
AND job_id <> 'IT_PROG';
```

Figura 16

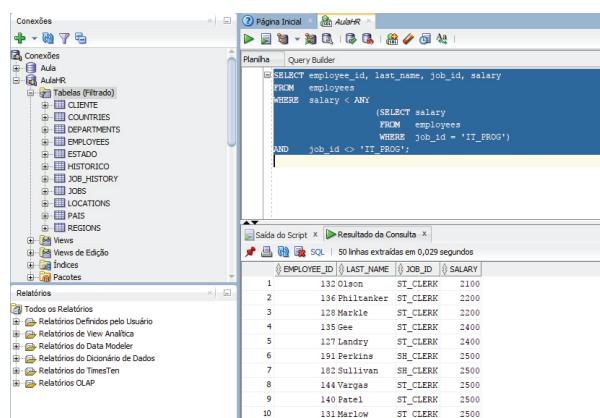


Figura 17

Operador ALL

Compara o valor com todos os valores retornados pela *subquery*.

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary < ALL
      9000,6000,4800,4200
      (SELECT salary
       FROM employees
       WHERE job_id = 'IT_PROG')
AND job_id <> 'IT_PROG';
```

Figura 18

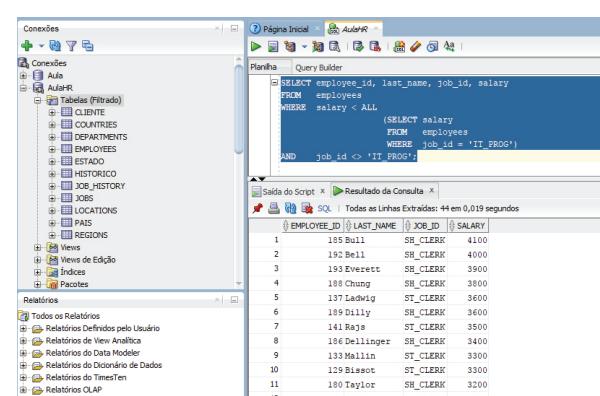


Figura 19

Operador *EXISTS*

O operador *EXISTS* é frequentemente utilizado com subpesquisas correlatas. Testa quando um valor existe; já *NOT EXISTS* garante que não existe. Se o valor existir, será retornado verdadeiro; se não existir, será retornado **falso**. Por exemplo, para encontrar os empregados que figuram em um mínimo para uma pessoa subordinada aos quais, faça o seguinte:

```

Conexões
  + - Conexões
    + - Aulas
      + - AulaR
        + - Tabelas (Filtrado)
          + - CLIENTE
          + - COUNTRIES
          + - DEPARTMENTS
          + - EMPLOYEES
          + - ESTADO
          + - HISTORICO
          + - JOBS
          + - LOCATIONS
          + - PAIS
          + - REGIONS
          + - Vagas
          + - Vagas de Edição
          + - Índices
          + - Pacotes
        + - Relatórios
          + - Todos os Relatórios
          + - Relatórios Definidos pelo Usuário
          + - Relatórios de Visão Analítica
          + - Relatórios do Data Modeler
          + - Relatórios do Dicionário de Dados
          + - Relatórios do TimesTen
          + - Relatórios OLAP
      + - Planilha Query Builder
        SELECT EMPLOYEE_ID, MANAGER_ID, FIRST_NAME, LAST_NAME,
               JOB_ID, DEPARTMENT_ID, SALARY
        FROM EMPLOYEES E
       WHERE EXISTS (SELECT EMPLOYEE_ID
                      FROM EMPLOYEES
                     WHERE E.MANAGER_ID =
                           EMPLOYEES.EMPLOYEE_ID)
        ORDER BY EMPLOYEE_ID;
    + - Saída do Script Resultado da Consulta
      + - SQL | 50 linhas extraídas em 0,083 segundos
        + - EMPLOYEE_ID | MANAGER_ID | FIRST_NAME | LAST_NAME | JOB_ID | DEPARTMENT_ID | SALARY
        1   101   100 Neena Kochhar AD_VP         90 17000
        2   102   100 Lex De Haan AD_VP         90 17000
        3   103   102 Alexander Hunold IT_PROG     60 9000
        4   104   103 Bruce Ernst IT_PROG     60 6000
        5   105   103 David Austin IT_PROG     60 4800
        6   106   103 Valli Pataballa IT_PROG     60 4800
        7   107   103 Diana Lorentz IT_PROG     60 4200
        8   108   101 Nancy Greenberg FI_MGR     100 12000
        9   109   108 Daniel Faviet FI_ACCOUNT 100 9000
       10   110   108 John Chen FI_ACCOUNT 100 8200
        ...
  
```

Figura 20

Agora, para encontrar os empregados que não têm departamento relacionado na tabela *DEPARTMENTS*, faça o seguinte:

```

Conexões
  + - Conexões
    + - Aulas
      + - AulaR
        + - Tabelas (Filtrado)
          + - CLIENTE
          + - COUNTRIES
          + - DEPARTMENTS
          + - EMPLOYEES
          + - ESTADO
          + - HISTORICO
          + - JOBS
          + - LOCATIONS
          + - PAIS
          + - REGIONS
          + - Vagas
          + - Vagas de Edição
          + - Índices
          + - Pacotes
      + - Planilha Query Builder
        SELECT EMPLOYEE_ID, MANAGER_ID, FIRST_NAME, LAST_NAME,
               JOB_ID, DEPARTMENT_ID, SALARY
        FROM EMPLOYEES
       WHERE NOT EXISTS (SELECT DEPARTMENT_ID
                           FROM DEPARTMENTS
                          WHERE EMPLOYEES.DEPARTMENT_ID = DEPARTMENTS.DEPARTMENT_ID);
    + - Saída do Script Resultado da Consulta
      + - SQL | Todas as Linhas Extraídas: 1 em 0,019 segundos
        + - EMPLOYEE_ID | MANAGER_ID | FIRST_NAME | LAST_NAME | JOB_ID | DEPARTMENT_ID | SALARY
        1   178   149 Kimberly Grant SA_REP      (null) 7000
  
```

Figura 21

Funções de Decisão

DECODE

Temos duas formas de trabalhar com a condição dentro de consultas em *Oracle SQL*: *CASE* e *DECODE*. Ambas têm a mesma função, que é permitir, de forma dinâmica e prática, como obter um retorno de uma coluna com base em uma condição, ou seja, ter a possibilidade de usar condições semelhantes ao *IF-THEN-ELSE* em consultas. Vejamos a sintaxe:

DECODE(col/expressão, procurado1, resultado1,...,padrão)

Col/expressão é comparado com cada um dos valores procurados e separados por vírgula, retornando o resultado se *col/expressão* for igual ao valor procurado.

Se não for encontrado nenhum dos valores procurados, a função *DECODE* retorna o valor padrão; se o valor padrão for omitido, retornará um valor nulo.

Ademais, *DECODE* deve ter, no mínimo, três parâmetros ou argumentos, vejamos alguns:

- *COL/EXPRESSÃO* – nome da coluna ou expressão a ser avaliada;
- *PROCURADO1* – o primeiro valor para ser testado;
- *RESULTADO1* – o valor para ser retornado se *PROCURADO1* for encontrado;
- *PROCURADO1* e *RESULTADO1* – podem ser repetidos quantas vezes se fizer necessário, por exemplo:
- (*PROCURADO2*, *RESULTADO2*, *PROCURADO3*, *RESULTADO3*,...)
- *PADRÃO* – o valor a ser retornado se nenhum procurado for encontrado.

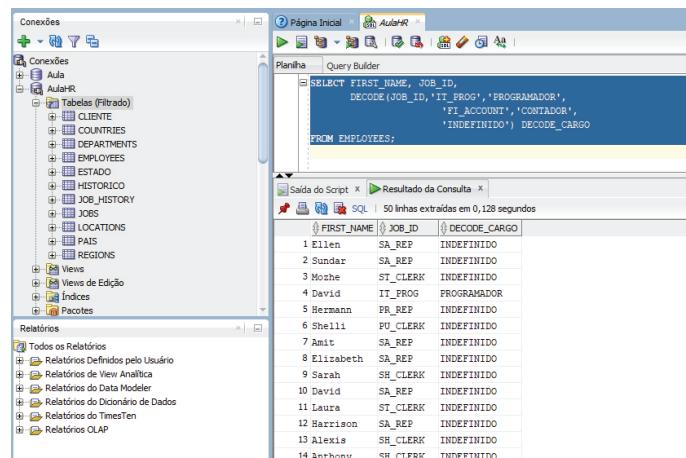


Col/expressão pode corresponder a vários tipos de dados, de modo que *PROCURADO* deve ser um dado do tipo coluna ou expressão.

O seguinte exemplo decodifica unicamente os cargos dos tipos *IT_PROG* e *CLERK*; enquanto que outros cargos serão padrões alterados para *INDEFINIDO*:

```
SELECT FIRST_NAME, JOB_ID,
       DECODE(JOB_ID, 'IT_PROG', 'PROGRAMADOR',
              'FI_ACCOUNT', 'CONTADOR',
              'INDEFINIDO') DECODE_CARGO
FROM EMPLOYEES;
```

FROM EMPLOYEES;



FIRST_NAME	JOB_ID	DECODE_CARGO
Ellen	SA_REP	INDEFINIDO
Sundar	SA_REP	INDEFINIDO
Moche	ST_CLERK	INDEFINIDO
David	IT_PROG	PROGRAMADOR
Hermann	PR_REP	INDEFINIDO
Shelli	PU_CLERK	INDEFINIDO
Amit	SA_REP	INDEFINIDO
Elizabeth	SA_REP	INDEFINIDO
Sarah	SH_CLERK	INDEFINIDO
David	SA_REP	INDEFINIDO
Laura	ST_CLERK	INDEFINIDO
Harrison	SA_REP	INDEFINIDO
Alexis	SH_CLERK	INDEFINIDO
Larken	SH_CLERK	INDEFINIDO

Figura 22

Já no próximo exemplo, deve-se retornar o salário incrementado de acordo com o tipo de cargo:

```
SELECT FIRST_NAME, JOB_ID, SALARY,
       DECODE(  JOB_ID,'IT_PROG',SALARY * 1.1,
               'FI_ACCOUNT',SALARY * 1.2,
               'AD_VP', SALARY * 0.95,
               SALARY) DECODE_CARGO
  FROM EMPLOYEES;
```

The screenshot shows the Oracle SQL Developer interface. On the left, the 'Conexões' (Connections) sidebar lists 'Aula' and 'AulaHR'. Under 'AulaHR', the 'Tabelas (Filtrado)' (Filtered Tables) section contains 'CLIENTE', 'COUNTRIES', 'DEPARTMENTS', 'EMPLOYEES', 'ESTADO', 'HISTORICO', 'JOB_HISTORY', 'JOBS', 'LOCATIONS', 'PAIS', and 'REGIONS'. Below this is the 'Views' section. On the right, the 'Planilha' (Worksheet) tab displays the following SQL query:

```
SELECT FIRST_NAME, JOB_ID, SALARY,
       DECODE(  JOB_ID,'IT_PROG',SALARY * 1.1,
               'FI_ACCOUNT',SALARY * 1.2,
               'AD_VP', SALARY * 0.95,
               SALARY) DECODE_CARGO
  FROM EMPLOYEES;
```

Below the worksheet is the 'Saída do Script' (Script Output) tab, which shows the execution time: '50 linhas extraídas em 0,016 segundos'. The 'Resultado da Consulta' (Query Result) tab shows the output of the query:

FIRST_NAME	JOB_ID	SALARY	DECODE_CARGO
1 Steven	AD_PRES	24000	24000
2 Neena	AD_VP	17000	16150
3 Lex	AD_VP	17000	16150
4 Alexander	IT_PROG	9000	9900
5 Bruce	IT_PROG	6000	6600
6 David	IT_PROG	4800	5280
7 Valli	IT_PROG	4800	5280
8 Diana	IT_PROG	4200	4620
9 Nancy	FI_MGR	12008	12008
10 Daniel	FI_ACCOUNT	9000	10800
11 John	FI_ACCOUNT	8200	9840
12 Ismael	FI_ACCOUNT	7700	9240
13 Jose Manuel	FI_ACCOUNT	7800	9360
14 Luis	FI_ACCOUNT	6900	8280
15 Den	PU_MAN	11000	11000

Figura 23

CASE

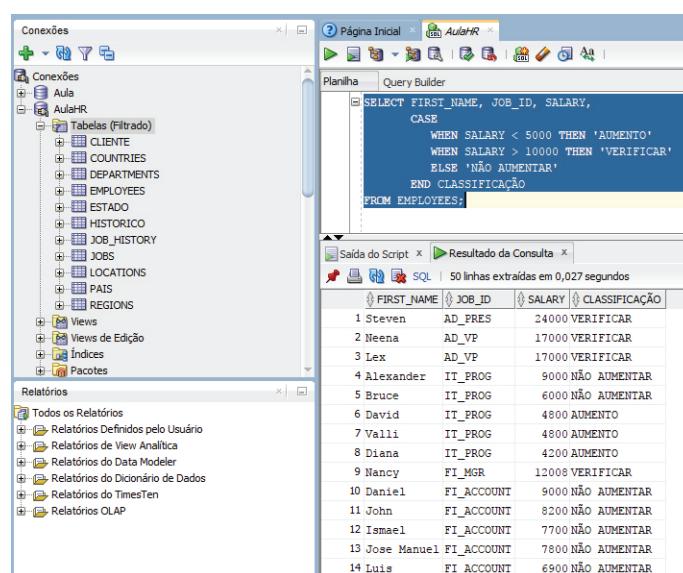
A função *CASE* permite o processamento condicional que exija o tratamento de várias hipóteses, portanto, possibilita maior flexibilidade na elaboração de condições; permite também a utilização das condições *AND* e *OR*.

Assim, o seguinte exemplo avalia duas expressões lógicas e ainda oferece uma terceira possibilidade – quando as duas anteriores resultarem falsas:

```

SELECT FIRST_NAME, JOB_ID, SALARY,
CASE
    WHEN SALARY < 5000 THEN 'AUMENTO'
    WHEN SALARY > 10000 THEN 'VERIFICAR'
    ELSE 'NÃO AUMENTAR'
END CLASSIFICAÇÃO

FROM EMPLOYEES;
  
```



The screenshot shows the Oracle SQL Developer interface. On the left, the 'Conexões' (Connections) sidebar shows a connection named 'AulaHR'. Below it, the 'Tabelas (Filtrado)' (Tables Filtered) section lists tables such as CLIENTE, COUNTRIES, DEPARTMENTS, EMPLOYEES, ESTADO, HISTORICO, JOB_HISTORY, JOBS, LOCATIONS, PAIS, and REGIONS. The 'Relatórios' (Reports) sidebar shows various report categories. The main window displays a SQL query in the 'Planilha' (Worksheet) tab:

```

SELECT FIRST_NAME, JOB_ID, SALARY,
CASE
    WHEN SALARY < 5000 THEN 'AUMENTO'
    WHEN SALARY > 10000 THEN 'VERIFICAR'
    ELSE 'NÃO AUMENTAR'
END CLASSIFICAÇÃO
FROM EMPLOYEES;
  
```

The 'Resultado da Consulta' (Query Result) tab shows the output of the query:

FIRST_NAME	JOB_ID	SALARY	CLASSIFICAÇÃO
1 Steven	AD_PRES	24000	VERIFICAR
2 Neena	AD_VP	17000	VERIFICAR
3 Lex	AD_VP	17000	VERIFICAR
4 Alexander	IT_PROG	9000	NÃO AUMENTAR
5 Bruce	IT_PROG	6000	NÃO AUMENTAR
6 David	IT_PROG	4800	AUMENTO
7 Valli	IT_PROG	4800	AUMENTO
8 Diana	IT_PROG	4200	AUMENTO
9 Nancy	FI_MGR	12008	VERIFICAR
10 Daniel	FI_ACCOUNT	9000	NÃO AUMENTAR
11 John	FI_ACCOUNT	8200	NÃO AUMENTAR
12 Ismael	FI_ACCOUNT	7700	NÃO AUMENTAR
13 Jose Manuel	FI_ACCOUNT	7800	NÃO AUMENTAR
14 Luis	FI_ACCOUNT	6900	NÃO AUMENTAR

Figura 24

Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta Unidade:



Livros

SQL : guia prático

Ler capítulo V do livro: Costa, Rogério Luis de C. SQL : guia prático. 2. ed. Rio de Janeiro : Brasport, 2006.

Oracle Database 11g Sql

Como complemento a esta Unidade, leia o sexto capítulo do livro de J. Price, intitulado Oracle Database 11g Sql, publicado em 2009, pela editora Bookman, de Porto Alegre, RS.



Vídeos

Oracle - Sub-Queries

<https://youtu.be/ocysliQ0DZA>

Sub Select em tabelas de histórico - SQL Aula 08

https://youtu.be/EYoS_AGNMvo

Referências

FANDERUFF, D. **Dominando o Oracle 9i:** modelagem e desenvolvimento. São Paulo: Pearson Education do Brasil, 2003.

MORELLI, E. M. T. **Oracle 9i fundamental:** Sql, Pl/SQL e administração. São Paulo: Érica, 2002.

PRICE, J. **Oracle Database 11g Sql.** Porto Alegre, RS: Bookman, 2009.



Cruzeiro do Sul Virtual
Educação a Distância

www.cruzeirodosulvirtual.com.br
Campus Liberdade
Rua Galvão Bueno, 868
CEP 01506-000
São Paulo - SP - Brasil
Tel: (55 11) 3385-3000



Cruzeiro do Sul
Educacional