Objetivos:

- Listar os recursos das instruções SELECT SQL
- Executar uma instrução SELECT básica
- Diferenciar instruções SQL e comandos SQL*Plus

Recursos das Instruções SELECT SQL

Seleção

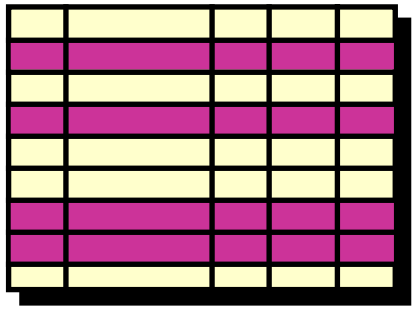


Tabela 1

Projeção

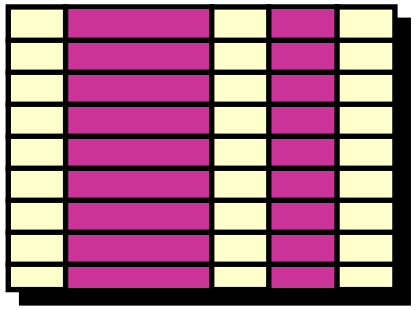


Tabela 1

Junção

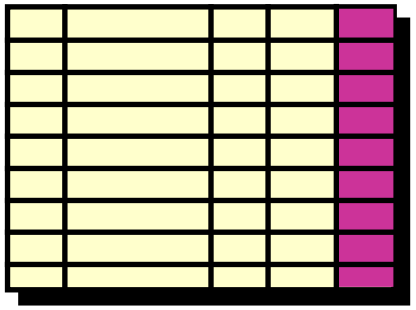


Tabela 1

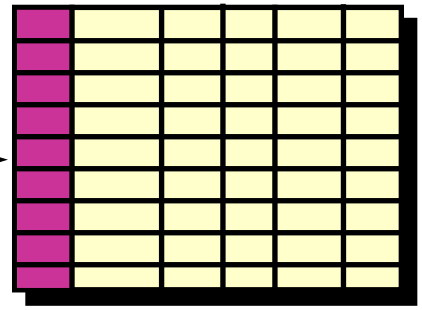


Tabela 2

Instrução SELECT Básica

```
SELECT   [DISTINCT] {*, coluna [apelido],...}
FROM     tabela;
```

- SELECT identifica ***que*** colunas.
- FROM identifica ***qual*** tabela.

Criando Instruções SQL

–Instruções SQL não fazem distinção entre maiúsculas e minúsculas.

–Instruções SQL podem estar em uma ou mais linhas.

–Palavras-chave não podem ser abreviadas ou divididas entre as linhas.

–Normalmente, as cláusulas são colocadas em linhas separadas.

–Guias e endentações são usadas para aperfeiçoar a legibilidade.

Selecionando Todas as Colunas

```
SQL> SELECT *
  2  FROM   dept;
```

```
    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

Selecionando Colunas Específicas

```
SQL> SELECT deptno, loc
  2  FROM   dept;
```

```
   DEPTNO LOC
--------- -------------
       10 NEW YORK
       20 DALLAS
       30 CHICAGO
       40 BOSTON
```

Defaults de Cabeçalho de Coluna

- Justificada default
  - Esquerda: Dados de caractere e data
  - Direita: Dados numéricos
- Exibição default: Letra maiúscula

Expressões Aritméticas

- Criar expressões com dados NUMBER e DATE usando operadores aritméticos

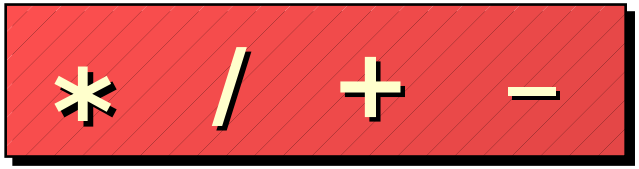
Operador	Descrição
+	Adicionar
-	Subtrair
*	Multiplicar
/	Dividir

Usando Operadores Aritméticos

```
SQL> SELECT ename, sal, sal+300
  2  FROM   emp;
```

```
ENAME             SAL   SAL+300
---------- --------- ---------
KING             5000      5300
BLAKE            2850      3150
CLARK            2450      2750
JONES            2975      3275
MARTIN           1250      1550
ALLEN            1600      1900
...
14 rows selected.
```

Precedência do Operador



–A multiplicação e a divisão têm prioridade sobre a adição e a subtração.

–Os operadores com a mesma prioridade são avaliados da esquerda para a direita.

–Os parênteses são usados para forçar a avaliação e para esclarecer as instruções.

Precedência do Operador

```
SQL> SELECT ename, sal, 12*sal+100
  2  FROM   emp;
```

```
ENAME             SAL 12*SAL+100
---------- --------- -----------
KING             5000       60100
BLAKE            2850       34300
CLARK            2450       29500
JONES            2975       35800
MARTIN           1250       15100
ALLEN            1600       19300
...
14 rows selected.
```

Usando Parênteses

```
SQL> SELECT ename, sal, 12*(sal+100)
  2  FROM   emp;
```

```
ENAME             SAL 12*(SAL+100)
---------- --------- ------------
KING             5000        61200
BLAKE            2850        35400
CLARK            2450        30600
JONES            2975        36900
MARTIN           1250        16200
...
14 rows selected.
```

Definindo um Valor Nulo

–Um valor nulo não está disponível, não é atribuído, é desconhecido ou não é aplicável.

–Um valor nulo não é o mesmo que um zero ou um espaço em branco.

```
SQL> SELECT ename, job, sal, comm
  2  FROM   emp;
```

```
ENAME      JOB              SAL       COMM
---------- --------- ---------- ----------
KING       PRESIDENT       5000
BLAKE      MANAGER         2850
...
TURNER     SALESMAN        1500          0
...
14 rows selected.
```

Valores Nulos nas Expressões Aritméticas

Expressões aritméticas contendo um valor nulo são avaliadas como nulo.

```
SQL> select ename, 12*sal+comm
  2  from   emp
  3  WHERE  ename='KING';
```

```
ENAME      12*SAL+COMM
---------- -----------
KING
```

Definindo um Apelido de Coluna

- Renomeia um cabeçalho de coluna
- É útil para cálculos
- Segue imediatamente o nome da coluna
- Palavra-chave **AS** opcional entre o nome da coluna e o apelido
- Necessita de aspas duplas caso contenha espaços ou caracteres especiais ou faça distinção entre maiúsculas e minúsculas

Usando Apelidos de Coluna

```
SQL> SELECT ename AS name, sal salary
  2  FROM   emp;
```

```
NAME                 SALARY
-------------- ---------

...
```

```
SQL> SELECT ename  "Name",
  2         sal*12 "Annual Salary"
  3  FROM   emp;
```

```
Name           Annual Salary
-------------- -------------

...
```

Operador de Concatenação

- Concatena colunas ou strings de caractere a outras colunas

- É representado por duas barras Verticais - ||

- Cria uma coluna resultante que é uma expressão de caracteres

Usando um Operador de Concatenação

```
SQL> SELECT   ename||job AS "Employees"
  2  FROM     emp;
```

```
Employees
------------------
KINGPRESIDENT
BLAKEMANAGER
CLARKMANAGER
JONESMANAGER
MARTINSALESMAN
ALLENSALESMAN
...
14 rows selected.
```

Strings Literais de Caracteres

–Uma literal é um caractere, um número ou uma data incluída na lista SELECT.

–Os valores literais de caractere e data devem estar entre aspas simples.

–Cada string de caractere é gerada uma vez para cada linha retornada.

Usando Strings Literais de Caracteres

```
SQL> SELECT ename ||' is a '||job
  2                AS "Employee Details"
  3  FROM   emp;
```

```
Employee Details
-------------------------
KING is a PRESIDENT
BLAKE is a MANAGER
CLARK is a MANAGER
JONES is a MANAGER
MARTIN is a SALESMAN
...
14 rows selected.
```

Linhas Duplicadas

•A exibição default das consultas é de todas as linhas, incluindo linhas duplicadas.

```
SQL> SELECT deptno
  2  FROM   emp;
```

```
   DEPTNO
---------
       10
       30
       10
       20
...
14 rows selected.
```

Eliminando Linhas Duplicadas

Elimine linhas duplicadas usando a palavra-chave DISTINCT na cláusula SELECT.

```
SQL> SELECT DISTINCT deptno
  2  FROM   emp;
```

```
    DEPTNO
---------
        10
        20
        30
```

Exibindo a Estrutura de Tabela

Use o comando DESCRIBE do SQL*Plus para exibir a estrutura de uma tabela.

```
DESC[RIBE] nome da tabela
```

```
SQL> DESCRIBE dept
```

```
Name                 Null?    Type
-------------------- -------- ------------
DEPTNO               NOT NULL NUMBER(2)
DNAME                         VARCHAR2(14)
LOC                           VARCHAR2(13)
```

Objetivos:

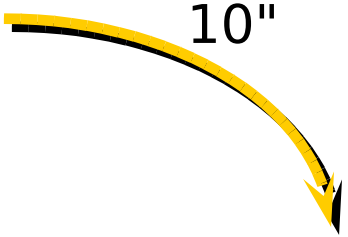
- Limitar linhas recuperadas por uma consulta
- Classificar linhas recuperadas por uma consulta

Limitando Linhas Usando uma Seleção

EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		10
7566	JONES	MANAGER		20
...				

"...recuperar todos os funcionários do departamento 10"



EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7782	CLARK	MANAGER		10
7934	MILLER	CLERK		10

Limitando Linhas Selecionadas

–Restringe as linhas retornadas usando a cláusula WHERE.

```
SELECT      [DISTINCT] {*| coluna [apelido], ...}
FROM        tabela
[WHERE      condição(ões)];
```

–A cláusula WHERE segue a cláusula FROM.

Usando a Cláusula WHERE

```
SQL> SELECT ename, job, deptno
  2  FROM   emp
  3  WHERE  job='CLERK';
```

```
ENAME      JOB          DEPTNO
---------- --------- ---------
JAMES      CLERK            30
SMITH      CLERK            20
ADAMS      CLERK            20
MILLER     CLERK            10
```

Strings de Caractere e Datas

–As strings de caractere e valores de data aparecem entre aspas simples.

–Os valores de caractere fazem distinção entre maiúsculas e minúsculas e os valores de data diferenciam formatos.

–O formato de data default é DD-MON-YY.

```
SQL> SELECT   ename, job, deptno
  2  FROM     emp
  3  WHERE    ename = 'JAMES';
```

Operadores de Comparação

Operador	Significado
=	Igual a
>	Maior do que
>=	Maior do que ou igual a
<	Menor do que
<=	Menor ou igual a
<>	Diferente de

Usando Operadores de Comparação

```
SQL> SELECT ename, sal, comm
  2  FROM   emp
  3  WHERE  sal<=comm;
```

```
ENAME             SAL       COMM
---------- ---------- ----------
MARTIN          1250 <-->    1400
```

Outros Operadores de Comparação

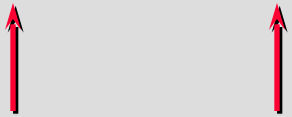
Operador	Significado
BETWEEN ...AND...	Entre dois valores (inclusive)
IN(list)	Vincula qualquer um de uma lista de valores
LIKE	Vincula um padrão de caractere
IS NULL	É um valor nulo
IS NOT NULL	Não é um valor nulo

Usando o Operador BETWEEN

Use o operador BETWEEN para exibir linhas baseadas em uma faixa de valores.

```
SQL> SELECT   ename, sal
  2  FROM     emp
  3  WHERE    sal BETWEEN 1000 AND 1500;
```

```
ENAME           SAL
---------- ---------
MARTIN         1250
TURNER         1500
WARD           1250
ADAMS          1100
MILLER         1300
```



Limite inferior

Limite superior

Usando o Operador IN

Use o operador IN para testar os valores de uma lista.

```
SQL> SELECT   empno, ename, sal, mgr
  2  FROM     emp
  3  WHERE    mgr IN (7902, 7566, 7788);
```

```
     EMPNO ENAME                SAL        MGR
---------- ---------- --------- ----------
      7902 FORD                3000       7566
      7369 SMITH                800       7902
      7788 SCOTT               3000       7566
      7876 ADAMS               1100       7788
```

Usando o Operador LIKE

- Use o operador LIKE para executar pesquisas curinga de valores de string de pesquisa válidos.
- As condições de pesquisa podem conter caracteres literais ou números.
  - % denota zero ou muitos caracteres.
  - _ denota um caractere.

```
SQL> SELECT   ename
  2  FROM     emp
  3  WHERE    ename LIKE 'S%';
```

Usando o Operador LIKE

–Você pode combinar caracteres de vinculação de padrão.

```
SQL> SELECT   ename
  2  FROM     emp
  3  WHERE    ename LIKE '_A%';
```

```
ENAME
----------
MARTIN
JAMES
WARD
```

É possível usar o identificador ESCAPE para procurar por "%" ou "_".

Usando o Operador IS NULL

Teste para valores nulos com o operador IS NULL.

```
SQL> SELECT  ename, mgr
  2  FROM    emp
  3  WHERE   mgr IS NULL;
```

```
ENAME           MGR
---------- ---------
KING
```

Operadores Lógicos

Operador	Significado
AND	Retorna TRUE se as condições de componentes forem TRUE
OR	Retorna TRUE se uma condição de componente for TRUE
NOT	Retorna TRUE se a condição seguinte for FALSE

Usando o Operador AND

AND exige que ambas as condições sejam TRUE.

```
SQL> SELECT empno, ename, job, sal
  2  FROM   emp
  3  WHERE  sal>=1100
  4  AND    job='CLERK';
```

```
     EMPNO ENAME      JOB              SAL
---------- ---------- --------- ----------
      7876 ADAMS      CLERK            1100
      7934 MILLER     CLERK            1300
```

Usando o Operador OR

OR exige que uma condição seja TRUE.

```
SQL> SELECT empno, ename, job, sal
  2  FROM   emp
  3  WHERE  sal>=1100
  4  OR     job='CLERK';
```

```
     EMPNO ENAME      JOB              SAL
---------- ---------- --------- ----------
      7839 KING       PRESIDENT        5000
      7698 BLAKE      MANAGER          2850
      7782 CLARK      MANAGER          2450
      7566 JONES      MANAGER          2975
      7654 MARTIN     SALESMAN         1250
      ...
      7900 JAMES      CLERK             950
      ...
14 rows selected.
```

Usando o Operador NOT

```
SQL> SELECT ename, job
  2  FROM   emp
  3  WHERE  job NOT IN ('CLERK','MANAGER','ANALYST');
```

```
ENAME      JOB
---------- ---------
KING       PRESIDENT
MARTIN     SALESMAN
ALLEN      SALESMAN
TURNER     SALESMAN
WARD       SALESMAN
```

Regras de Precedência

Ordem de Avaliação	Operador
1	Todos os operadores
	de comparação
2	NOT
3	AND
4	OR

Sobreponha regras de precedência usando parênteses.

Regras de Precedência

```
SQL> SELECT ename, job, sal
  2  FROM   emp
  3  WHERE  job='SALESMAN'
  4  OR     job='PRESIDENT'
  5  AND    sal>1500;
```

ENAME	JOB	SAL
KING	PRESIDENT	5000
MARTIN	SALESMAN	1250
ALLEN	SALESMAN	1600
TURNER	SALESMAN	1500
WARD	SALESMAN	1250

Regras de Precedência

Use parênteses para forçar a prioridade.

```
SQL> SELECT   ename, job, sal
  2  FROM     emp
  3  WHERE    (job='SALESMAN'
  4  OR       job='PRESIDENT')
  5  AND      sal>1500;
```

```
ENAME      JOB              SAL
---------- --------- ----------
KING       PRESIDENT      5000
ALLEN      SALESMAN       1600
```

Cláusula ORDER BY

–Classificar as linhas com a cláusula ORDER BY

- ASC: ordem crescente, default
- DESC: ordem decrescente

–A cláusula ORDER BY vem depois na instrução SELECT.

```
SQL> SELECT    ename, job, deptno, hiredate
  2  FROM      emp
  3  ORDER BY hiredate;
```

```
ENAME      JOB           DEPTNO HIREDATE
---------- --------- --------- ---------
SMITH      CLERK             20 17-DEC-80
ALLEN      SALESMAN          30 20-FEB-81
...
14 rows selected.
```

Classificando em Ordem Decrescente

```
SQL> SELECT    ename, job, deptno, hiredate
  2  FROM      emp
  3  ORDER BY hiredate DESC;
```

```
ENAME      JOB           DEPTNO HIREDATE
---------- --------- --------- ---------
ADAMS      CLERK             20 12-JAN-83
SCOTT      ANALYST           20 09-DEC-82
MILLER     CLERK             10 23-JAN-82
JAMES      CLERK             30 03-DEC-81
FORD       ANALYST           20 03-DEC-81
KING       PRESIDENT         10 17-NOV-81
MARTIN     SALESMAN          30 28-SEP-81
...
14 rows selected.
```

Classificando por Apelido de Coluna

```
SQL> SELECT   empno, ename, sal*12 annsal
  2  FROM     emp
  3  ORDER BY annsal;
```

```
     EMPNO ENAME         ANNSAL
---------- ---------- ---------
      7369 SMITH          9600
      7900 JAMES         11400
      7876 ADAMS         13200
      7654 MARTIN        15000
      7521 WARD          15000
      7934 MILLER        15600
      7844 TURNER        18000
...
14 rows selected.
```

Classificando por Várias Colunas

A ordem da lista ORDER BY é a ordem de classificação.

```
SQL> SELECT    ename, deptno, sal
  2  FROM      emp
  3  ORDER BY  deptno, sal DESC;
```

```
ENAME          DEPTNO        SAL
---------- ---------- ----------
KING               10       5000
CLARK              10       2450
MILLER             10       1300
FORD               20       3000
...
14 rows selected.
```

Você pode classificar por uma coluna que não esteja na lista SELECT.

Sumário

```
SELECT        [DISTINCT] {*| coluna [apelido], ...}
FROM          tabela
[WHERE        condição(ões)]
[ORDER BY     {coluna, expr, apelido} [ASC|DESC]];
```

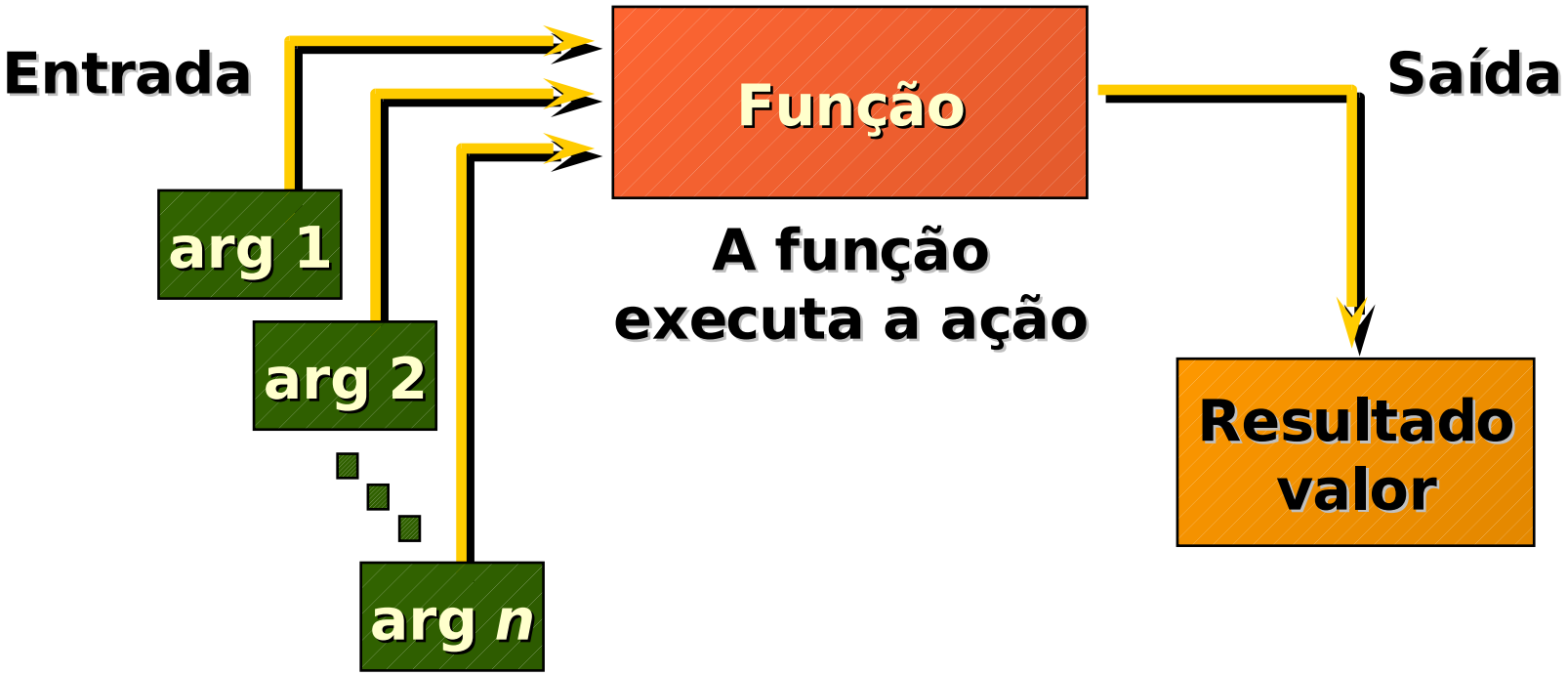
Objetivos:

–Descrever vários tipos de funções disponíveis no SQL

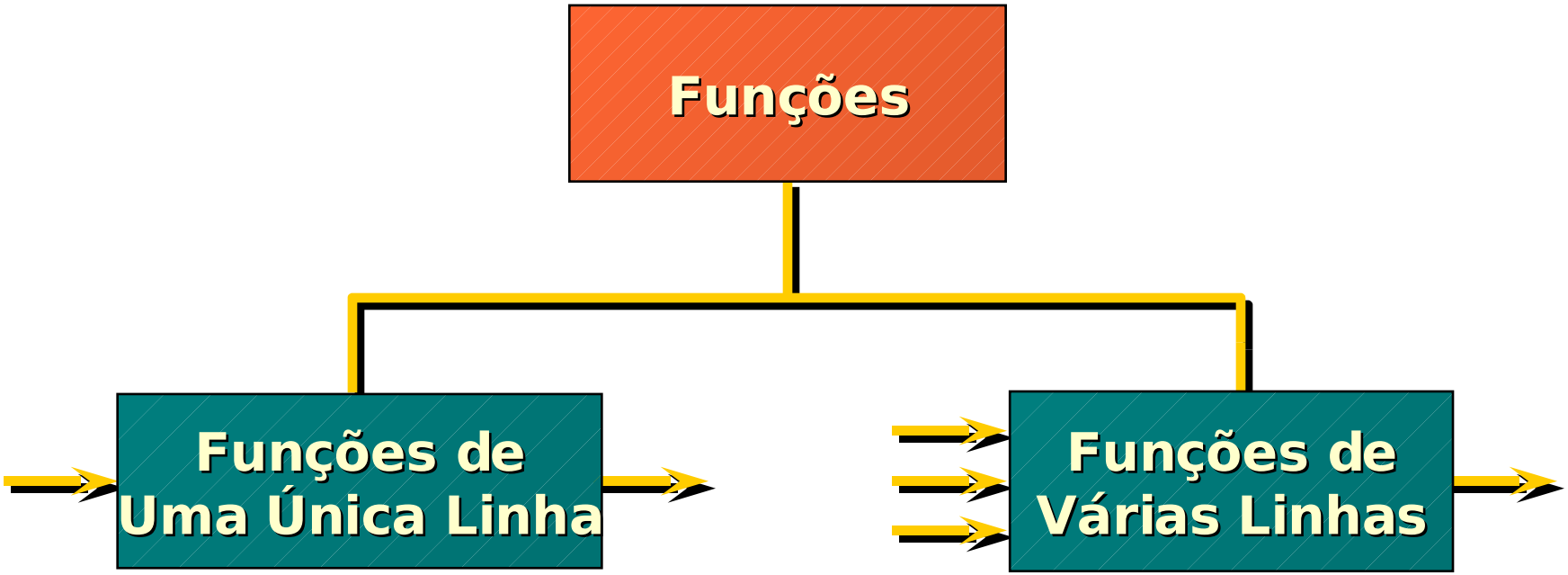
–Usar funções de data, número e caractere nas instruções SELECT

–Descrever o uso das funções de conversão

Funções SQL


Entrada
arg 1
arg 2
arg n
Função
A função executa a ação
Saída
Resultado valor

Dois Tipos de Funções SQL

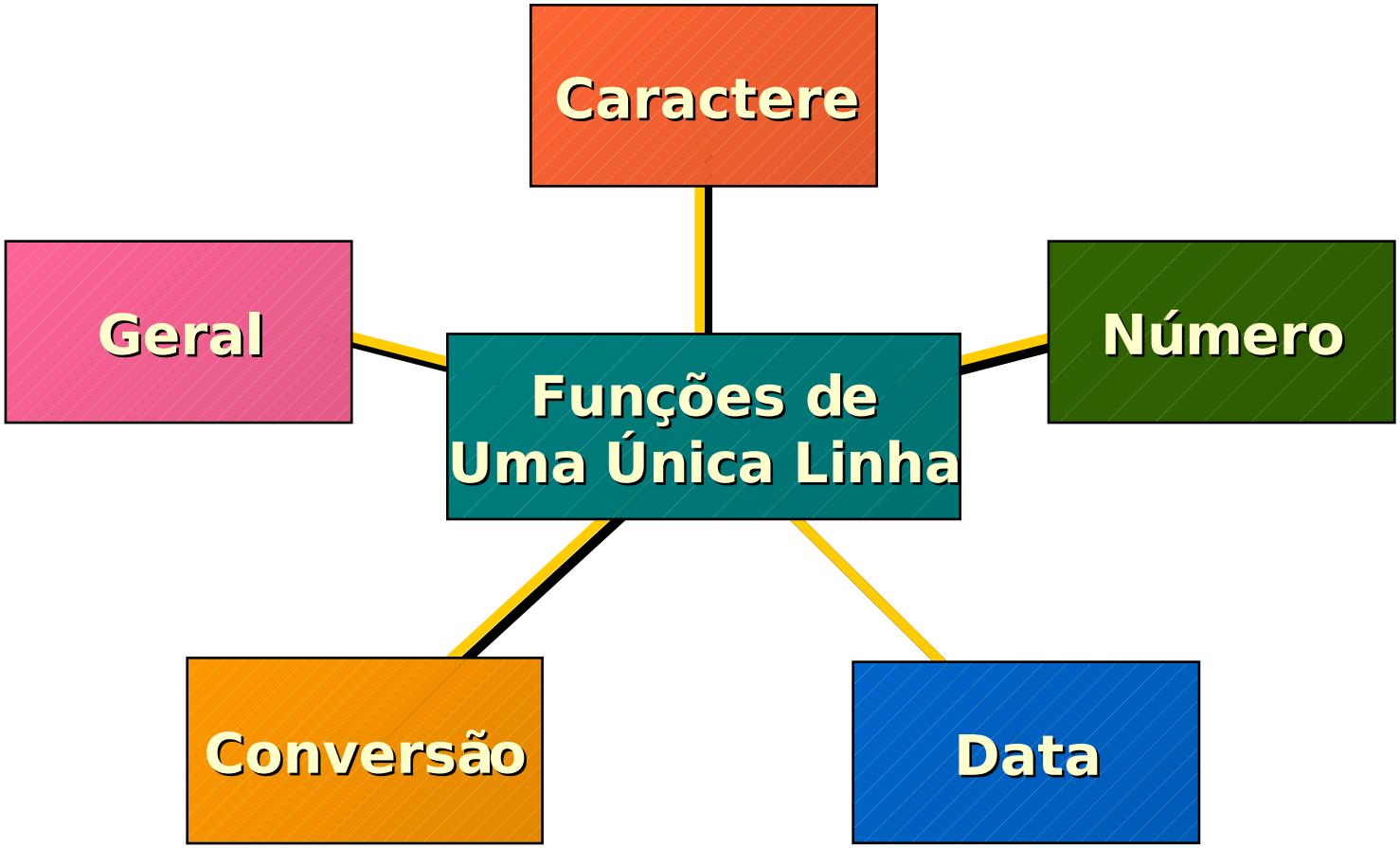


Funções de Uma Única Linha

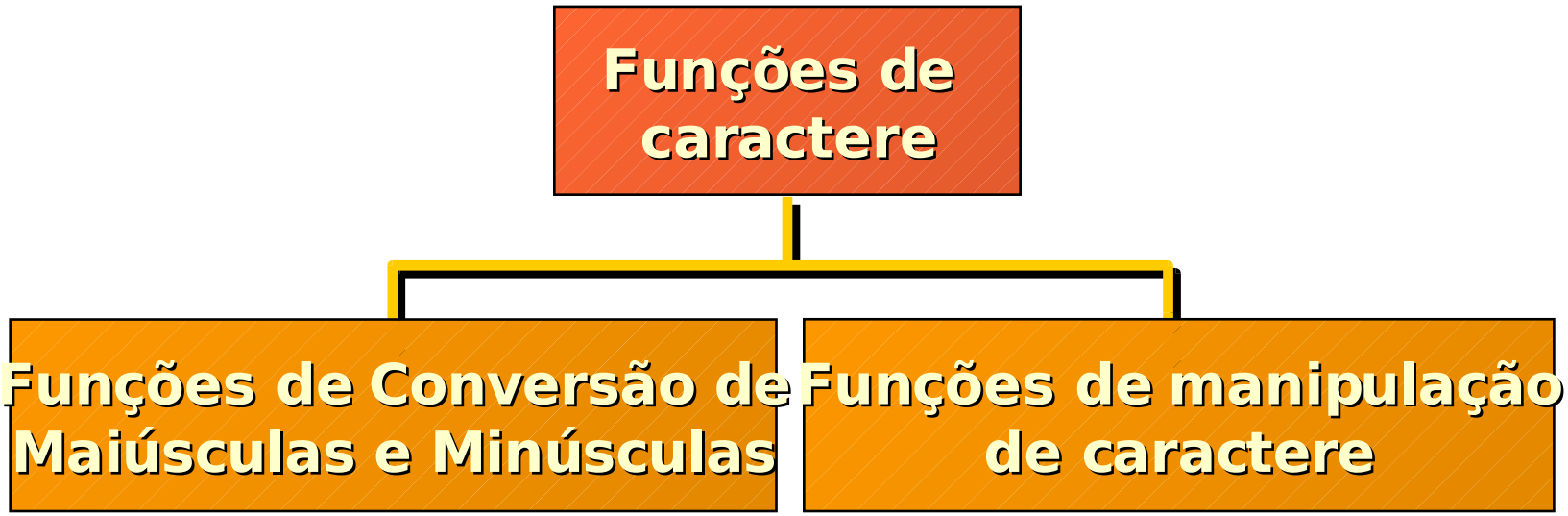
- Manipulam itens de dados
- Aceitam argumentos e retornam um valor
- Agem em cada linha retornada
- Retornam um resultado por linha
- Podem modificar o tipo de dados
- Podem ser aninhadas

```
function_name (coluna|expressão, [arg1, arg2,...])
```

Funções de Uma Única Linha



Funções de Caractere



LOWER
UPPER
INITCAP

CONCAT
SUBSTR
LENGTH
INSTR
LPAD
TRIM

Exibindo Dados de Várias Tabelas

Objetivos:

- Criar instruções SELECT para obter acesso aos dados a partir de mais de uma tabela usando as junções idênticas e não-idênticas
- Visualizar dados que, em geral, não correspondem a uma condição de junção usando junções externas
- Unindo uma tabela a ela mesma

Obtendo Dados de Várias Tabelas

EMP

EMPNO	ENAME	...	DEPTNO
7839	KING	...	10
7698	BLAKE	...	30
...			
7934	MILLER	...	10

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

```
EMPNO  DEPTNO LOC
----- ------- --------
 7839      10 NEW YORK
 7698      30 CHICAGO
 7782      10 NEW YORK
 7566      20 DALLAS
 7654      30 CHICAGO
 7499      30 CHICAGO
...
14 rows selected.
```

O Que É uma Junção?

- Use uma junção para consultar dados a partir de uma ou mais tabelas.

```
SELECT   tabela1.coluna, tabela2.coluna
FROM     tabela1, tabela2
WHERE    tabela1.coluna1 = tabela2.coluna2;
```

  - Criar uma condição de junção na cláusula WHERE.
  - Prefixar o nome da coluna com o nome da tabela quando o mesmo nome da coluna aparecer em mais de uma tabela.

Produto Cartesiano

–Um produto cartesiano é formado quando:

- Uma condição de junção estiver omitida
- Uma condição de junção estiver inválida
- Todas as linhas na primeira tabela estão 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 Produto Cartesiano

EMP (14 linhas)

```
 EMPNO ENAME   ...  DEPTNO
------ -----   ...  ------
  7839 KING    ...      10
  7698 BLAKE   ...      30
   ...
  7934 MILLER  ...      10
```

DEPT (4 linhas)

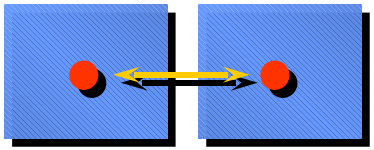
```
DEPTNO DNAME          LOC
------ ---------- --------
    10 ACCOUNTING NEW YORK
    20 RESEARCH   DALLAS
    30 SALES      CHICAGO
    40 OPERATIONS BOSTON
```

"Produto Cartesiano: 14*4=56 linhas"

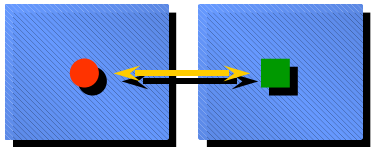
```
ENAME      DNAME
------     ----------
KING       ACCOUNTING
BLAKE      ACCOUNTING
...
KING       RESEARCH
BLAKE      RESEARCH
...
56 rows selected.
```

Tipos de Junções

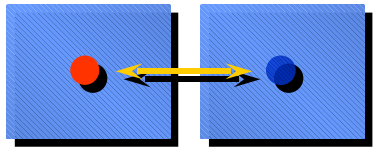
Junção idêntica



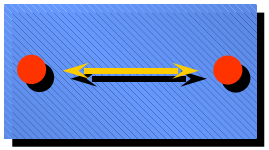
Junção não-idêntica



Junção externa



Autojunção



O Que É uma Junção Idêntica?

EMP

EMPNO	ENAME	DEPTNO
7839	KING	10
7698	BLAKE	30
7782	CLARK	10
7566	JONES	20
7654	MARTIN	30
7499	ALLEN	30
7844	TURNER	30
7900	JAMES	30
7521	WARD	30
7902	FORD	20
7369	SMITH	20

...
14 rows selected.

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
30	SALES	CHICAGO
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
20	RESEARCH	DALLAS
20	RESEARCH	DALLAS

...
14 rows selected.

Chave estrangeira

Chave primária

Recuperando Registros com Junções Idênticas

```
SQL> SELECT   emp.empno,   emp.ename, emp.deptno,
  2           dept.deptno, dept.loc
  3  FROM     emp, dept
  4  WHERE    emp.deptno=dept.deptno;
```

```
EMPNO ENAME   DEPTNO DEPTNO LOC
----- ------ ------ ------ ----------
 7839 KING       10     10 NEW YORK
 7698 BLAKE      30     30 CHICAGO
 7782 CLARK      10     10 NEW YORK
 7566 JONES      20     20 DALLAS
...
14 rows selected.
```

Qualificando Nomes de Coluna Ambíguos

–Use os prefixos de tabela para qualificar nomes de coluna que estão em várias tabelas.

–Melhore o desempenho usando os prefixos de tabela.

–Diferencie colunas que possuem nomes idênticos, mas que residam em tabelas diferentes usando apelidos de coluna.

Condições de Pesquisa Adicional Usando o Operador AND

EMP

EMPNO	ENAME	DEPTNO
7839	KING	10
7698	BLAKE	30
7782	CLARK	10
7566	JONES	20
7654	MARTIN	30
7499	ALLEN	30
7844	TURNER	30
7900	JAMES	30
7521	WARD	30
7902	FORD	20
7369	SMITH	20

...
14 rows selected.

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
30	SALES	CHICAGO
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
20	RESEARCH	DALLAS
20	RESEARCH	DALLAS

...
14 rows selected.

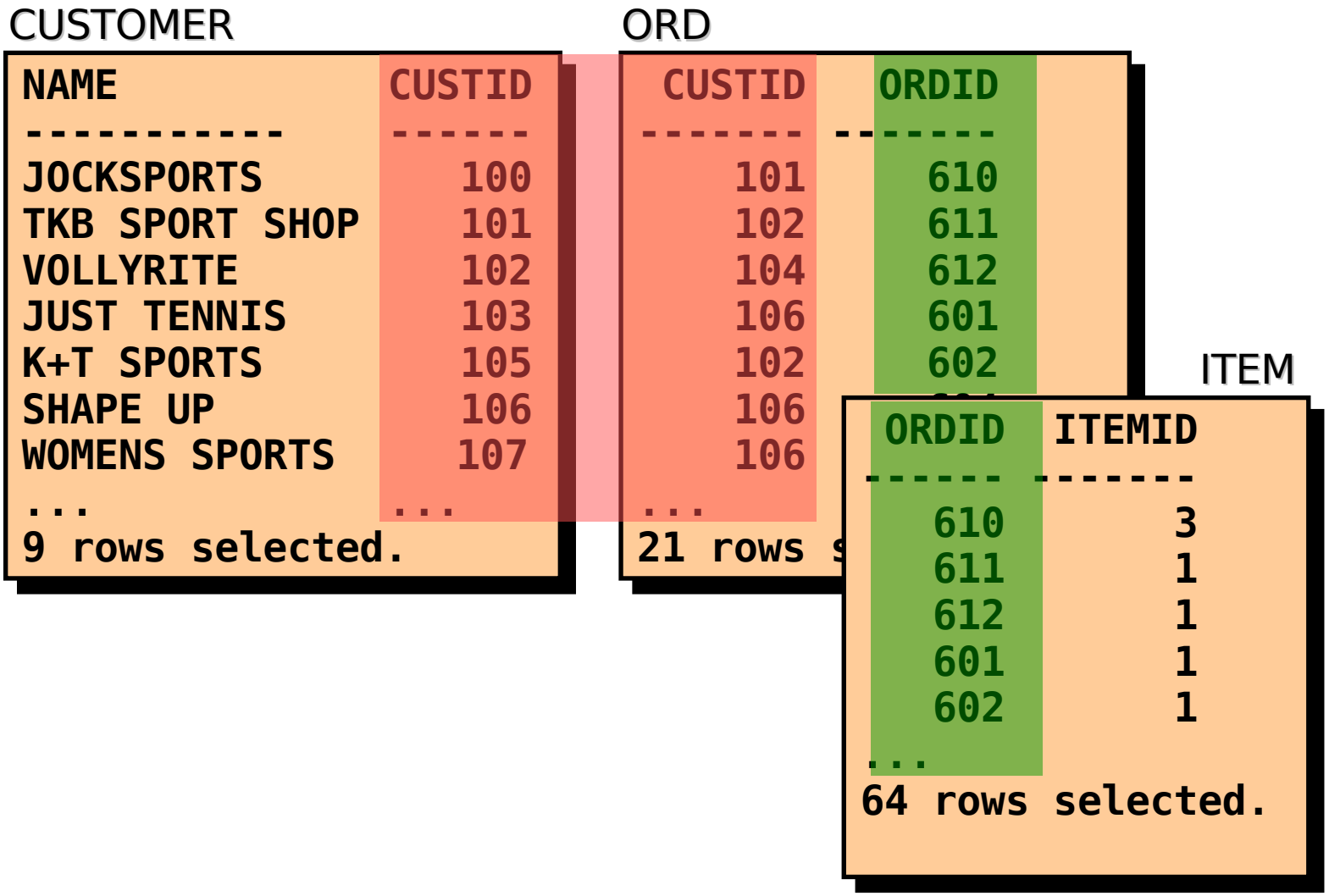
Usando Apelidos de Tabela

Simplifique consultas usando apelidos de tabela.

```
SQL> SELECT emp.empno,   emp.ename, emp.deptno,
  2         dept.deptno, dept.loc
  3  FROM   emp, dept
  4  WHERE  emp.deptno=dept.deptno;
```

```
SQL> SELECT e.empno,  e.ename, e.deptno,
  2         d.deptno, d.loc
  3  FROM   emp e,    dept d
  4  WHERE  e.deptno= d.deptno;
```

Unindo Mais de Duas Tabelas



Junções Não-idênticas

EMP

EMPNO	ENAME	SAL
7839	KING	5000
7698	BLAKE	2850
7782	CLARK	2450
7566	JONES	2975
7654	MARTIN	1250
7499	ALLEN	1600
7844	TURNER	1500
7900	JAMES	950

...
14 rows selected.

SALGRADE

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

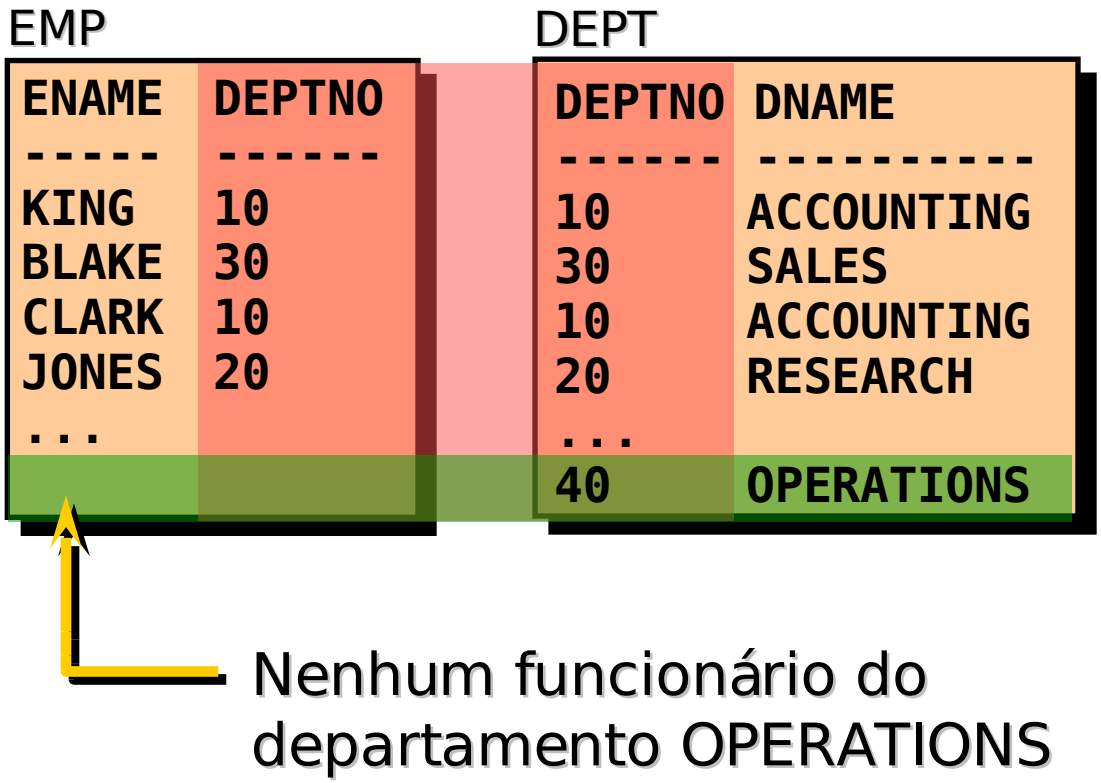
"o salário na tabela EMP está entre salário inferior e salário superior na tabela SALGRADE"

Recuperando Registros com Junções Não-idênticas

```
SQL>   SELECT   e.ename, e.sal, s.grade
   2   FROM     emp e,   salgrade s
   3   WHERE    e.sal
   4   BETWEEN  s.losal AND s.hisal;
```

```
ENAME             SAL      GRADE
---------- --------- ---------
JAMES             950          1
SMITH             800          1
ADAMS            1100          1
...
14 rows selected.
```

Junções Externas



Junções Externas

- Use uma junção externa para consultar também todas as linhas que em geral não atendem à condição de junção.
- O operador de junção externo é um sinal de adição (+).

```
SELECT tabela1.coluna, tabela2.coluna
FROM   tabela1, tabela2
WHERE  tabela1.coluna(+) = tabela2.coluna;
```

```
SELECT tabela1.coluna , tabela2.coluna
FROM   tabela1, tabela2
WHERE  tabela1.coluna = tabela2.coluna(+);
```

Usando Junções Externas

```
SQL> SELECT   e.ename, d.deptno, d.dname
  2  FROM     emp e,   dept d
  3  WHERE    e.deptno(+) = d.deptno
  4  ORDER BY e.deptno;
```

```
ENAME          DEPTNO DNAME
---------- --------- --------------
KING               10 ACCOUNTING
CLARK              10 ACCOUNTING
...
                   40 OPERATIONS
15 rows selected.
```

Autojunções

EMP (WORKER)

EMPNO	ENAME	MGR
7839	KING	
7698	BLAKE	7839
7782	CLARK	7839
7566	JONES	7839
7654	MARTIN	7698
7499	ALLEN	7698

EMP (MANAGER)

EMPNO	ENAME
7839	KING
7839	KING
7839	KING
7698	BLAKE
7698	BLAKE

"MGR na tabela WORKER é igual a EMPNO na tabela MANAGER"

Unindo uma Tabela a Ela Mesma

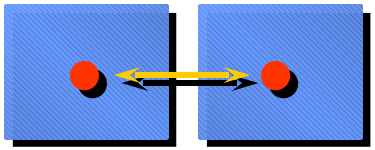
```
SQL> SELECT worker.ename||' works for '||manager.ename
  2  FROM   emp worker, emp manager
  3  WHERE  worker.mgr = manager.empno;
```

```
WORKER.ENAME||'WORKSFOR'||MANAG
------------------------------
BLAKE works for KING
CLARK works for KING
JONES works for KING
MARTIN works for BLAKE
...
13 rows selected.
```

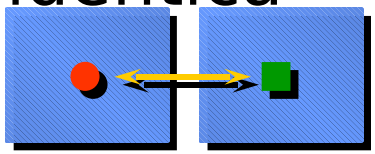
Sumário

```
SELECT   tabela1.coluna, tabela2.coluna
FROM     tabela1, tabela2
WHERE    tabela1.coluna1 = tabela2.coluna2;
```

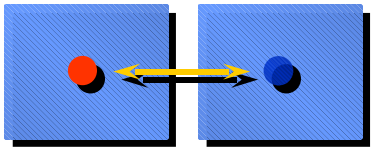
Junção idêntica



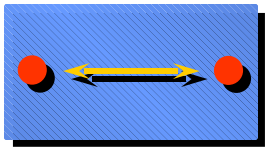
Junção não-idêntica



Junção externa



Autojunção



SQL: Lista de Exercício

1. Escreva uma query para mostrar o nome do empregado, número e nome do departamento para todos os empregados

2. Crie uma única lista de todos os cargos que estão no departamento 30.

3. Escreva uma query para mostrar o nome do empregado, nome e localização do departamento de todos os empregados que ganham comissão

4. Mostre o nome do empregado e nome do departamento para todo os empregado que tenha um A em seu nome. Salve em p4q4.sql.

5. Escreva uma query para mostrar o nome, cargo, número e nome do departamento de todos os empregados que trabalham em DALLAS

6. Mostre o nome e número do empregado com o seu respectivo gerente, nome e número. Nomeie as colunas como Employee, emp#, Manager, and Mgr#, respectivamente. Salve em p4q6.sql

7. Modifique p4q6.sql para mostrar todos os empregados, incluindo King, que não tem gerente. Salve em p4q7.sql. Execute.

SQL: Lista de Exercício

8. Crie uma query que mostre o nome do empregado, número do departamento e todos os empregados que trabalham no mesmo departamento. Nomeie cada coluna apropriadamente.

9. Mostre a estrutura da tabela SALGRADE. Crie uma query que mostre o nome, cargo, nome do departamento, salário e a faixa salarial de todos os empregados.

10. Crie uma query para mostrar o nome e data de contratação de todos empregados contratado após o Blake.

11. Mostre todos os nomes dos empregados com suas datas de contratações, nome dos gerentes e datas de contratações dos empregados que foram contratados antes dos seus gerentes. Nomeie as colunas como Employee, Emp Hiredate, Manager, and Mgr Hiredate, respectivamente.

12. Crie uma query que mostre o nome do empregado e salário como um montante de asteriscos. Cada asterisco significa centenas de dólares. Ordene os dados em ordem descendente de salário. Nomeie a coluna como EMPLOYEE_AND_THEIR_SALARIES.

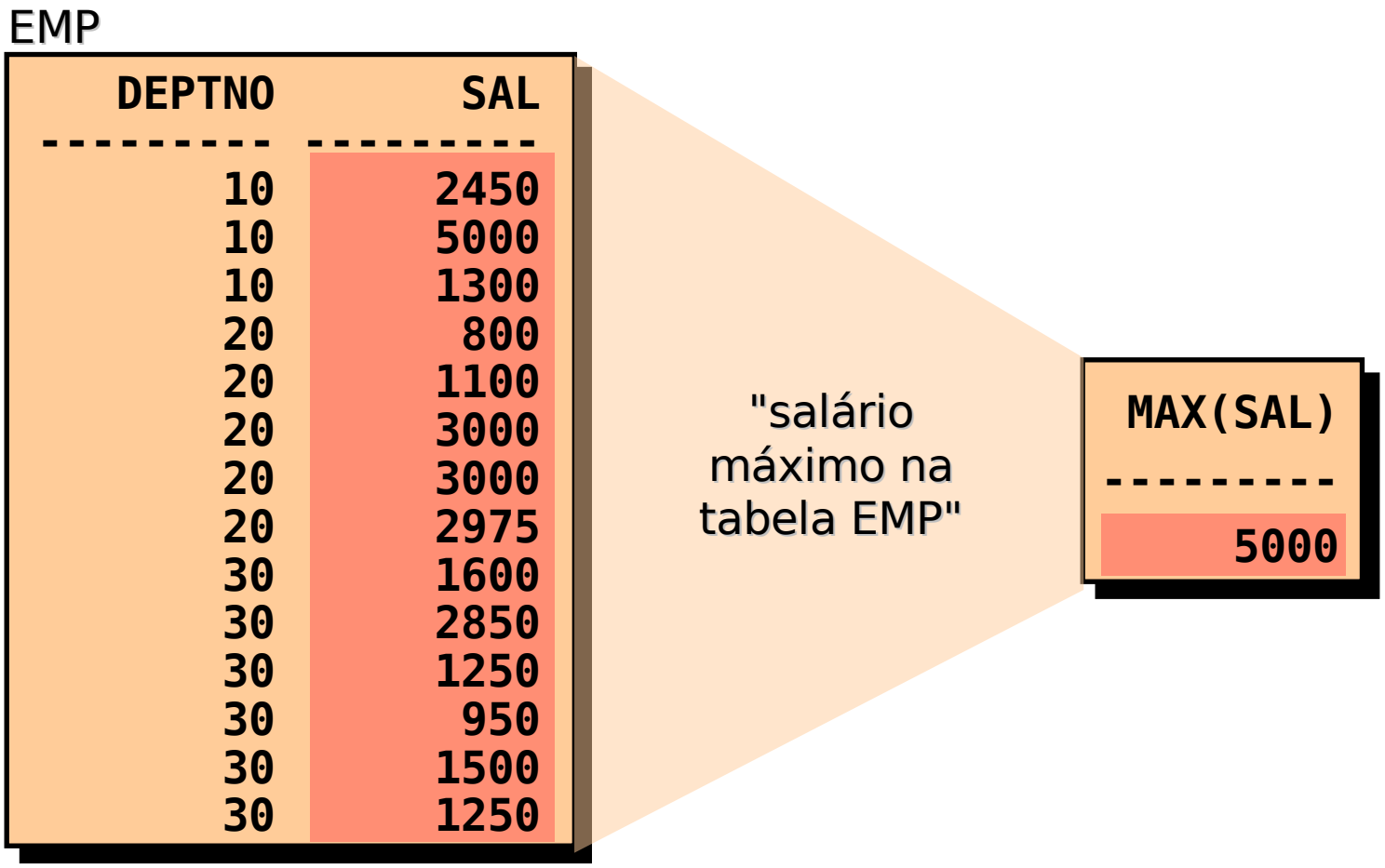
Agregando Dados Usando Funções de Grupo

Objetivos:

- Identificar as funções de grupo disponíveis
- Descrever o uso de funções de grupo
- Agrupar dados usando a cláusula GROUP BY
- Incluir ou excluir linhas agrupadas usando a cláusula HAVING

O Que São Funções de Grupo?

As funções de grupo operam em conjuntos de linhas para fornecer um resultado por grupo.



Tipos de Funções de Grupo

- AVG
- COUNT
- MAX
- MIN
- STDDEV
- SUM
- VARIANCE

Usando Funções de Grupo

```
SELECT        [coluna,] group_function(coluna)
FROM          tabela
[WHERE        condição]
[GROUP BY     coluna]
[ORDER BY     coluna];
```

Usando Funções AVG e SUM

Você pode usar AVG e SUM para dados numéricos.

```
SQL> SELECT   AVG(sal), MAX(sal),
  2           MIN(sal), SUM(sal)
  3  FROM     emp
  4  WHERE    job LIKE 'SALES%';
```

```
AVG(SAL)  MAX(SAL)  MIN(SAL)  SUM(SAL)
--------  --------  --------  --------
    1400      1600      1250      5600
```

Usando Funções MIN e MAX

Você pode usar MIN e MAX para qualquer tipo de dados.

```
SQL> SELECT   MIN(hiredate), MAX(hiredate)
  2  FROM     emp;
```

```
MIN(HIRED MAX(HIRED
--------- ---------
17-DEC-80 12-JAN-83
```

Usando a Função COUNT

COUNT(*) retorna o número de linhas em uma tabela.

```
SQL> SELECT   COUNT(*)
  2  FROM     emp
  3  WHERE    deptno = 30;
```

```
 COUNT(*)
---------
         6
```

Usando a Função COUNT

COUNT(*expr*) retorna o número de linhas não nulas.

```
SQL> SELECT   COUNT(comm)
  2  FROM     emp
  3  WHERE    deptno = 30;
```

```
COUNT(COMM)
-----------
          4
```

Funções de Grupo e Valores Nulos

As funções de grupo ignoram valores nulos na coluna.

```
SQL> SELECT AVG(comm)
  2  FROM   emp;
```

```
 AVG(COMM)
---------
       550
```

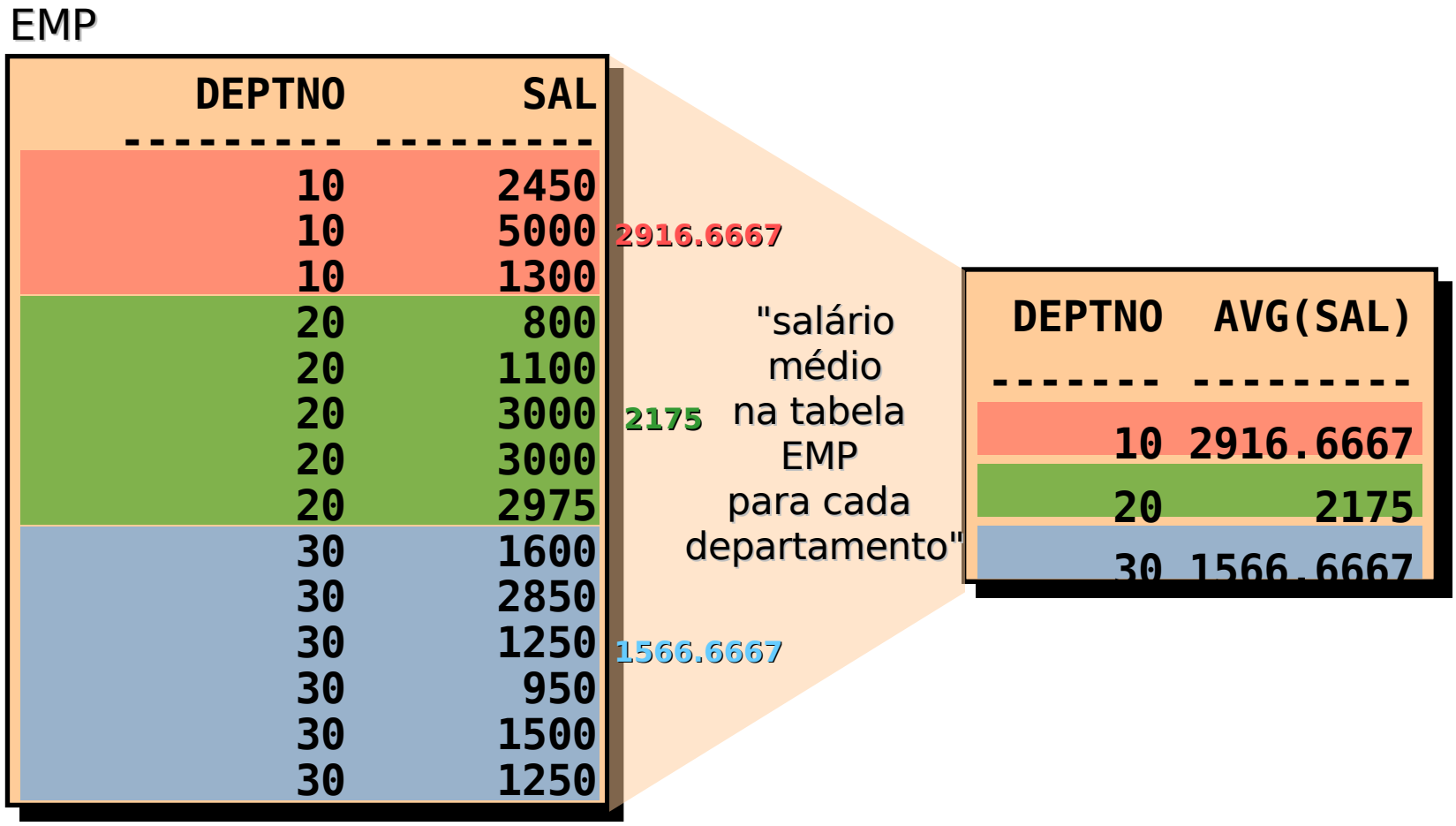
Usando a Função NVL com Funções de Grupo

A função NVL força as funções de grupo a incluírem valores nulos.

```
SQL> SELECT AVG(NVL(comm,0))
  2  FROM   emp;
```

```
AVG(NVL(COMM,0))
----------------
      157.14286
```

Criando Grupos de Dados



Criando Grupos de Dados: Cláusula GROUP BY

```
SELECT      coluna, group_function(coluna)
FROM        tabela
[WHERE      condição]
[GROUP BY   group_by_expression]
[ORDER BY   coluna];
```

Divida linhas de uma tabela em grupos menores usando a cláusula GROUP BY.

Usando a Cláusula GROUP BY

- Todas as colunas na lista SELECT que não estejam em funções de grupo devem estar na cláusula GROUP BY

```
SQL> SELECT   deptno, AVG(sal)
  2  FROM     emp
  3  GROUP BY deptno;
```

```
    DEPTNO  AVG(SAL)
---------- ---------
        10 2916.6667
        20       2175
        30 1566.6667
```

Usando a Cláusula GROUP BY

- A coluna GROUP BY não precisa estar na lista SELECT

```
SQL> SELECT   AVG(sal)
  2  FROM     emp
  3  GROUP BY deptno;
```

```
 AVG(SAL)
---------
2916.6667
      2175
1566.6667
```

Agrupando por Mais de Uma Coluna

EMP

DEPTNO	JOB	SAL
10	MANAGER	2450
10	PRESIDENT	5000
10	CLERK	1300
20	CLERK	800
20	CLERK	1100
20	ANALYST	3000
20	ANALYST	3000
20	MANAGER	2975
30	SALESMAN	1600
30	MANAGER	2850
30	SALESMAN	1250
30	CLERK	950
30	SALESMAN	1500
30	SALESMAN	1250

"soma de salários na tabela EMP para cada cargo, agrupados por departamento"

DEPTNO	JOB	SUM(SAL)
10	CLERK	1300
10	MANAGER	2450
10	PRESIDENT	5000
20	ANALYST	6000
20	CLERK	1900
20	MANAGER	2975
30	CLERK	950
30	MANAGER	2850
30	SALESMAN	5600

Usando a Cláusula GROUP BY em Várias Colunas

```
SQL> SELECT   deptno, job, sum(sal)
  2  FROM     emp
  3  GROUP BY deptno, job;
```

```
    DEPTNO JOB         SUM(SAL)
---------- --------- ----------
        10 CLERK           1300
        10 MANAGER         2450
        10 PRESIDENT       5000
        20 ANALYST         6000
        20 CLERK           1900
...
9 rows selected.
```

Consultas Ilegais Usando Funções de Grupo

- Qualquer coluna ou expressão na lista SELECT que não seja uma função agregada deve estar na cláusula GROUP BY.

```
SQL> SELECT   deptno, COUNT(ename)
  2  FROM     emp;
```

Coluna ausente na cláusula GROUP BY

```
SELECT deptno, COUNT(ename)
       *
ERROR at line 1:
ORA-00937: Nenhuma função de grupo de grupo único
(Not a single-group group function)
```

Consultas Ilegais Usando Funções de Grupo

–Não é possível usar a cláusula WHERE para restringir grupos.

–Use a cláusula HAVING para restringir grupos.

```
SQL> SELECT    deptno, AVG(sal)
  2  FROM      emp
  3  WHERE     AVG(sal) > 2000
  4  GROUP BY  deptno;
```

```
WHERE AVG(sal) > 2000
      *
ERROR at line 3:
ORA-00934: A função de grupo não é permitida aqui
(Group function is not allowed here)
```

Não é possível usar a cláusula WHERE para restringir grupos

Excluindo Resultados do Grupo

EMP

DEPTNO	SAL
10	2450
10	5000
10	1300
20	800
20	1100
20	3000
20	3000
20	2975
30	1600
30	2850
30	1250
30	950
30	1500
30	1250

5000

3000

2850

"salário máximo por departamento maior do que US$ 2.900"

DEPTNO	MAX(SAL)
10	5000
20	3000

Excluindo Resultados do Grupo: Cláusula HAVING

- Use a cláusula HAVING para restringir grupos
  - –As linhas são agrupadas.
  - –A função de grupo é aplicada.
  - –Os grupos que correspondem à cláusula HAVING são exibidos.

```
SELECT        coluna, group_function
FROM          tabela
[WHERE        condição]
[GROUP BY     group_by_expression]
[HAVING       group_condition]
[ORDER BY     coluna];
```

Usando a Cláusula HAVING

```
SQL> SELECT   deptno, max(sal)
  2  FROM     emp
  3  GROUP BY deptno
  4  HAVING   max(sal)>2900;
```

```
    DEPTNO  MAX(SAL)
---------- ---------
        10      5000
        20      3000
```

Usando a Cláusula HAVING

```
SQL> SELECT    job, SUM(sal) PAYROLL
  2  FROM      emp
  3  WHERE      job NOT LIKE 'SALES%'
  4  GROUP BY  job
  5  HAVING    SUM(sal)>5000
  6  ORDER BY  SUM(sal);
```

```
JOB          PAYROLL
--------- ---------
ANALYST         6000
MANAGER         8275
```

Aninhando Funções de Grupo

Exiba o salário médio máximo

```
SQL> SELECT   max(avg(sal))
  2  FROM     emp
  3  GROUP BY deptno;
```

```
MAX(AVG(SAL))
-------------
    2916.6667
```

Sumário

```
SELECT        coluna, group_function(coluna)
FROM          tabela
[WHERE        condição]
[GROUP BY     group_by_expression]
[HAVING       group_condition]
[ORDER BY     coluna];
```

- Ordem de avaliação das cláusulas:
  - cláusula WHERE
  - cláusula GROUP BY
  - cláusula HAVING

SQL: Lista de Exercício

Determine se verdadeiro(V) ou falso(F) as seguintes declarações:

1. Funções de grupo trabalham em muitas linhas para produzir um resultado.

2. Funções de grupo usam nulls nos seus cálculos.

3. A cláusula WHERE restringe linhas antes de incluí-las em cálculos de funções de grupos.

4. Mostre o maior, o menor, a soma e a média dos salários de todos os empregados. Nomeie as colunas como Maximum, Minimum, Sum, and Average, respectivamente. Arredonde os resultados para inteiro. Salve em p5q4.sql.

5. Modifique p5q4.sql para mostrar o menor, o maior, a soma e a média dos salários para cada tipo de cargo. Salve em p5q5.sql.

6. Escreva uma query para mostrar o número de empregados com o mesmo cargo.

7. Determine o número de gerentes sem listá-los. Nomeie a coluna como Number of Managers.

SQL: Lista de Exercício

Determine se verdadeiro(V) ou falso(F) as seguintes declarações:

8. Escreva uma query que mostre a diferença entre o maior e menor salário. Nomeie a coluna como DIFFERENCE.

9. Mostre o número do gerente e o salário mais baixo pago aos funcionários daquele gerente. Exclua o empregado que não possua gerente. Exclua qualquer grupo where o menor salário seja menor que $1000. Ordene por salário (descendente).

10. Escreva uma query para mostrar o nome do departamento, nome da localização, número de empregados, e média de salário para todos os empregados daquele departamento. Nomeie as colunas como dname, loc, Number of People, and Salary, respectivamente.

11. Crie uma query que mostre o número total de empregados e daquele total, o número que foram contratados em 1980, 1981, 1982, e 1983. Nomeie as colunas de forma apropriada.

Objetivos:

- Descrever os tipos de problemas que as subconsultas podem resolver
- Definir as subconsultas
- Listar os tipos de subconsultas
- Criar subconsultas de uma única linha e de várias linhas

Usando uma Subconsulta para Resolver um Problema

"Quem tem um salário maior que o de Jones?"

Consulta principal

"Que funcionários têm um salário maior que o salário de Jones?"

Subconsulta

"Qual é o salário de Jones?"

Subconsultas

```
SELECT   select_list
FROM     tabela
WHERE    operador expr
                    (SELECT      select_list
                    FROM         tabela);
```

–A subconsulta (consulta interna) é executada uma vez antes da consulta principal.

–O resultado da subconsulta é usado pela consulta principal (consulta externa).

Usando uma Subconsulta

```
SQL> SELECT ename
  2  FROM   emp
  3  WHERE  sal >
  4                (SELECT sal
  5                 FROM   emp
  6                 WHERE  empno=7566);
```

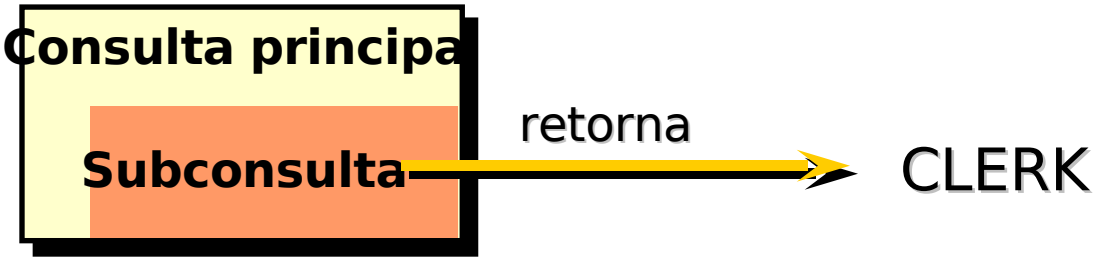
2975

```
ENAME
----------
KING
FORD
SCOTT
```

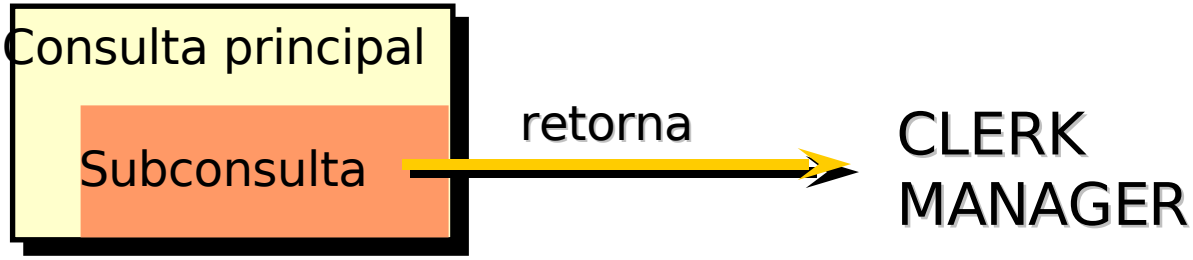
Diretrizes para o Uso de Subconsultas

- Coloque as subconsultas entre parênteses.
- Coloque as subconsultas no lado direito do operador de comparação.
- Não adicione uma cláusula ORDER BY a uma subconsulta.
- Use operadores de uma única linha com subconsultas de uma única linha.
- Use operadores de várias linhas com subconsultas de várias linhas.

Subconsulta de uma única linha



Subconsulta de várias linhas



Subconsulta de várias colunas

Subconsultas de uma Única Linha

–Retorne somente uma linha

–Use operadores de comparação de uma única linha

Operado	Significado
r	Igual a
=	Maior do que
>	Maior do que ou igual a
>=	Menor do que
<	Menor ou igual a
<=	Diferente de
<>	

Executando Subconsultas de uma Única Linha

```
SQL> SELECT    ename, job
  2  FROM      emp
  3  WHERE     job =
  4                    (SELECT      job
  5                    FROM         emp
  6                    WHERE        empno = 7369)
  7  AND       sal >
  8                    (SELECT      sal
  9                    FROM         emp
  10                   WHERE        empno = 7876);
```

CLERK

1100

```
ENAME      JOB
---------- ---------
MILLER     CLERK
```

Usando Funções de Grupo em uma Subconsulta

```
SQL> SELECT   ename, job, sal
  2  FROM     emp
  3  WHERE    sal =
  4                  (SELECT       MIN(sal)
  5                  FROM          emp);
```

800

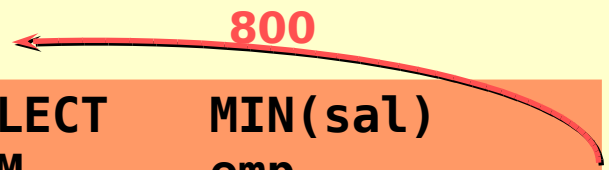
```
ENAME      JOB              SAL
---------- --------- ---------
SMITH      CLERK            800
```

Cláusula HAVING com Subconsultas

–O Oracle Server primeiro executa as subconsultas.

–O Oracle Server retorna os resultados para a cláusula HAVING da consulta principal.

```
SQL> SELECT       deptno, MIN(sal)
  2  FROM         emp
  3  GROUP BY     deptno
  4  HAVING       MIN(sal) >
  5                      (SELECT    MIN(sal)
  6                      FROM       emp
  7                      WHERE      deptno = 20);
```



O que Há de Errado com esta Instrução?

```
SQL> SELECT empno, ename
  2  FROM   emp
  3  WHERE  sal =
  4                (SELECT   MIN(sal)
  5                FROM      emp
  6                GROUP BY  deptno);
```

Operador de uma única linha com subconsulta de várias linhas

```
ERROR:
ORA-01427: A subconsulta de uma única linha retorna
mais de uma linha (Single-row subquery returns
more than one row)

no rows selected
```

Esta Instrução Irá Funcionar?

```
SQL> SELECT ename, job
  2  FROM   emp
  3  WHERE  job =
  4                (SELECT job
  5                FROM    emp
  6                WHERE   ename='SMYTHE');
```

```
no rows selected
```

A subconsulta não retorna nenhum valor

Subconsultas de Várias Linhas

- Retorne mais de uma linha
- Use operadores de comparação de várias linhas

Operador	Significado
IN	Igual a qualquer membro na lista
ANY	Compare o valor a cada valor retornado pela subconsulta
ALL	Compare o valor a todo valor retornado pela subconsulta

Usando o Operador ANY em Subconsultas de Várias Linhas

```
SQL> SELECT   empno, ename, job
  2  FROM     emp
  3  WHERE    sal < ANY
  4                  (SELECT   sal
  5                  FROM      emp
  6                  WHERE     job = 'CLERK')
  7  AND      job <> 'CLERK';
```

1300
1100
800
950

```
     EMPNO ENAME      JOB
---------- ---------- ---------
      7654 MARTIN     SALESMAN
      7521 WARD       SALESMAN
```

Usando o Operador ALL em Subconsultas de Várias Linhas

```
SQL> SELECT  empno, ename, job
  2  FROM    emp
  3  WHERE   sal > ALL
  4                 (SELECTavg(sal)
  5                 FROM          emp
  6                 GROUP BY      deptno);
```

1566.6667
2175
2916.6667

```
     EMPNO ENAME      JOB
---------- ---------- ---------
      7839 KING       PRESIDENT
      7566 JONES      MANAGER
      7902 FORD       ANALYST
      7788 SCOTT      ANALYST
```

As subconsultas são úteis quando uma consulta baseia-se em valores desconhecidos.

```
SELECT   select_list
FROM     tabela
WHERE    operador expr
                    (SELECT select_list
                    FROM    tabela);
```

SQL: Lista de Exercício

1. Escreva uma query para mostrar o nome do empregado e data de contratação para todos os empregados do departamento do Blake. Exclua o Blake.

2. Crie uma query para mostrar o número e nome dos empregados que ganham acima da máeia salarial. Ordene os resultados por salário (decrescente).

3. Escre uma query que mostre o número e nome dos empregados que trabalham em um departamento que tem um empregado cujo nome contem um *T*. salve em *p6q3.sql.*

4. Mostre o nome do empregado, número do departamento e cargo para todos eo empregados lotados em Dallas.

5. Mostre o nome do empregado e salário de todos os empregados do King.

6. Mostre o número do departamento, nome do empregado e cargo de todos os empregados do departamento Sales.

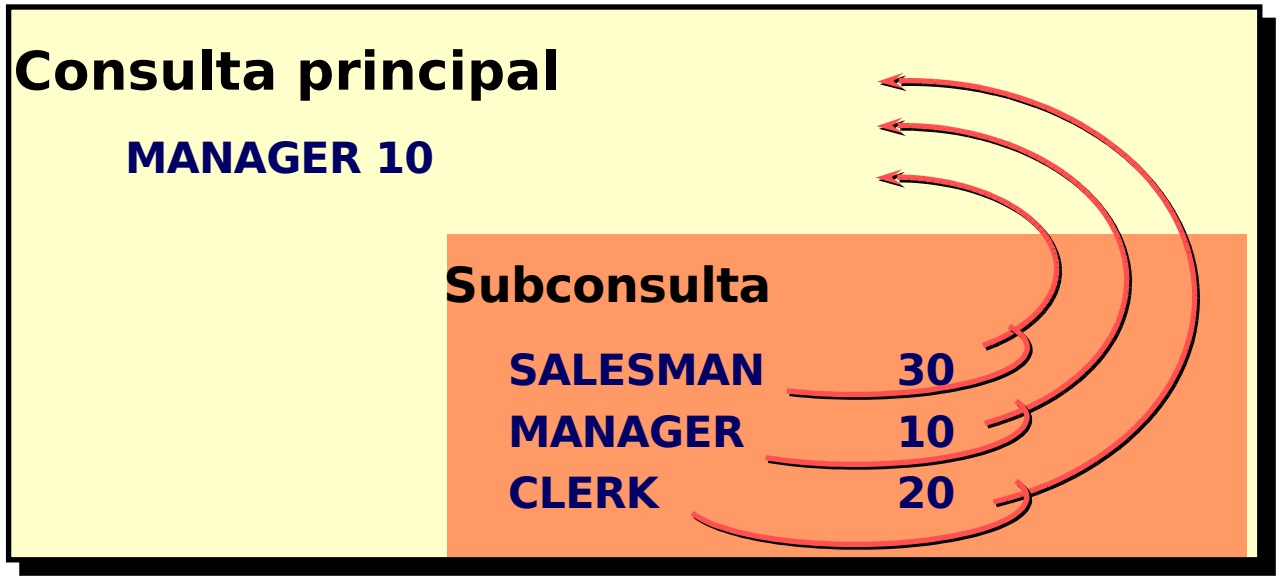
7. Modifiique *p6q3.sql* para mostrar o número e nome do empregado, salário de todos os empregados que ganham mais que média salarial e que trabalham em um departamento que possui um empregado com um *T* em seu nome. Salve como *p6q7.sql*. Execute.

Subconsultas de Várias Colunas

Objetivos:

- Criar uma subconsulta de várias colunas
- Descrever e explicar o comportamento de subconsultas quando valores nulos forem recuperados
- Criar uma subconsulta em uma cláusula FROM

Subconsultas de Várias Colunas



A consulta principal compara a Valores de uma subconsulta de várias linhas e de várias colunas

MANAGER 10

SALESMAN 30
MANAGER 10
CLERK 20

Usando Subconsultas de Várias Colunas

Exiba a ID da ordem, a ID do produto e a quantidade de itens na tabela de itens que corresponde à ID do produto e à quantidade de um item na ordem 605.

```
SQL> SELECT   ordid, prodid, qty
  2  FROM     item
  3  WHERE    (prodid, qty) IN
  4                          (SELECT prodid, qty
  5                           FROM   item
  6                           WHERE  ordid = 605)
  7  AND      ordid <> 605;
```

Usando Subconsultas de Várias Colunas

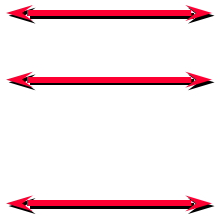
Exiba o número da ordem, o número do produto e a quantidade de qualquer item em que o número do produto e a quantidade correspondam ao número do produto e à quantidade de um item na ordem 605.

```
SQL> SELECT   ordid, prodid, qty
  2  FROM     item
  3  WHERE    (prodid, qty) IN
  4                          (SELECT prodid, qty
  5                           FROM   item
  6                           WHERE  ordid = 605)
  7  AND      ordid <> 605;
```

Comparações de Coluna

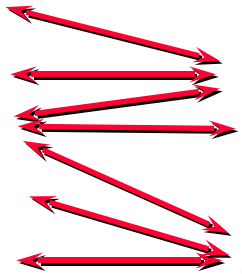
Aos pares

PRODID	QTY
101863	100
100861	100
102130	10
100890	5
100870	500
101860	50



Sem ser aos pares

PRODID	QTY
101863	100
100861	100
102130	10
100890	5
100870	500
101860	50



Subconsulta de Comparação que Não Seja aos Pares

Exiba o número da ordem, o número do produto e a quantidade de qualquer item em que o número do produto e a quantidade correspondam a qualquer número do produto e quantidade de um item na ordem 605.

```
SQL> SELECT   ordid, prodid, qty
  2  FROM     item
  3  WHERE    prodid IN  (SELECT          prodid
  4                       FROM           item
  5                       WHERE          ordid = 605)
  6  AND      qty    IN  (SELECT          qty
  7                       FROM           item
  8                       WHERE          ordid = 605)
  9  AND      ordid <> 605;
```

Subconsulta que Não Seja aos Pares

```
    ORDID     PRODID        QTY
--------- ---------- ----------
      609     100870          5
      616     100861         10
      616     102130         10
      621     100861         10
      618     100870         10
      618     100861         50
      616     100870         50
      617     100861        100
      619     102130        100
      615     100870        100
      617     101860        100
      621     100870        100
      617     102130        100
        . . .
16 rows selected.
```

Valores Nulos em uma Subconsulta

```
SQL> SELECT   employee.ename
  2  FROM     emp employee
  3  WHERE    employee.empno NOT IN
  4                          (SELECT manager.mgr
  5                           FROM   emp manager);
no rows selected.
```

Usando uma Subconsulta na Cláusula FROM

```
SQL> SELECT  a.ename, a.sal, a.deptno, b.salavg
  2  FROM    emp a, (SELECT   deptno, avg(sal) salavg
  3                  FROM     emp
  4                  GROUP BY deptno) b
  5  WHERE   a.deptno = b.deptno
  6  AND     a.sal > b.salavg;
```

```
ENAME                SAL     DEPTNO     SALAVG
---------- --------- --------- ----------
KING              5000         10  2916.6667
JONES             2975         20       2175
SCOTT             3000         20       2175
...
6 rows selected.
```

SQL: Lista de Exercício

1. Escreva uma query para mostrar nome do empregado, número do departamento e salário de qualquer empregado cujo número do departamento e salário casam ambos com o número do departamento e salário de qualquer empregado que ganha comissão.

2. Mostre o nome do empregado, nome do departamento e salário de qualquer empregado cujo salário e comissão casam ambos com com o salário e comissão de qualquer empregado lotado em Dallas.

3. Crie uma query para mostrar o nome, data de contratação e salário para todos os empregados que tenham o mesmo salário e comissão do Scott.

4. Crie uma query para mostrar os empregados que ganham salário maior que qualquer CLERKS. Ordene o resultado por salário (decrescente).

Objetivos:

- Descrever cada instrução DML
- Inserir linhas em uma tabela
- Atualizar linhas em uma tabela
- Deletar linhas de uma tabela
- Controlar transações

DML (Data Manipulation Language)

–Uma instrução DML é executada quando você:

- Adiciona novas linhas a uma tabela
- Modifica linhas existentes em uma tabela
- Remove linhas existentes de uma tabela

–Uma *transação* consiste em um conjunto de instruções DML que formam uma unidade lógica de trabalho.

Adicionando uma Nova Linha em uma Tabela

50	DEVELOPMENT	DETROIT

Nova linha

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

"... inserir uma nova linha na tabela DEPT..."

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	DEVELOPMENT	DETROIT

A Instrução INSERT

–Adicione novas linhas em uma tabela usando a instrução INSERT.

```
INSERT INTO   tabela [(coluna [, coluna...])]
VALUES        (valor [, valor...]);
```

–Somente uma linha é inserida por vez com esta sintaxe.

Inserindo Novas Linhas

–Insira uma nova linha contendo valores para cada coluna.

–Liste valores na ordem default das colunas na tabela.

–Liste opcionalmente as colunas na cláusula INSERT.

```
SQL> INSERT INTO    dept (deptno, dname, loc)
  2  VALUES          (50, 'DEVELOPMENT', 'DETROIT');
1 row created.
```

– Coloque os valores de data e caractere entre aspas simples.

Inserindo Linhas com Valores Nulos

Método implícito: Omita a coluna da lista de colunas.

```
SQL> INSERT INTO    dept (deptno, dname )
  2  VALUES          (60, 'MIS');
1 row created.
```

Método explícito: Especifique a palavra-chave NULL.

```
SQL> INSERT INTO    dept
  2  VALUES          (70, 'FINANCE', NULL);
1 row created.
```

Inserindo Valores Especiais

A função SYSDATE registra a data e hora atuais.

```
SQL> INSERT INTO     emp (empno, ename, job,
  2                  mgr, hiredate, sal, comm,
  3                  deptno)
  4  VALUES          (7196, 'GREEN', 'SALESMAN',
  5                  7782, SYSDATE, 2000, NULL,
  6                  10);
1 row created.
```

Inserindo Valores Específicos de Data

Adicionar um novo funcionário.

```
SQL> INSERT INTO emp
  2  VALUES      (2296,'AROMANO','SALESMAN',7782,
  3               TO_DATE('FEB 3, 1997', 'MON DD, YYYY'),
  4               1300, NULL, 10);
1 row created.
```

Verifique sua adição.

```
EMPNO ENAME   JOB       MGR   HIREDATE  SAL COMM DEPTNO
----- ------- --------- ---- ---------- ---- ---- ------
 2296 AROMANO SALESMAN 7782 03-FEB-97 1300           10
```

Inserindo Valores Usando Variáveis de Substituição

Crie um script interativo usando parâmetros de substituição do SQL*Plus

```
SQL> INSERT INTO     dept (deptno, dname, loc)
  2  VALUES          (&department_id,
  3                  '&department_name', '&location');
```

```
Enter value for department_id: 80
Enter value for department_name: EDUCATION
Enter value for location: ATLANTA

1 row created.
```

Criando um Script com Prompts Personalizados

–ACCEPT armazena o valor em uma variável.

–PROMPT exibe o texto personalizado.

```
ACCEPT        department_id PROMPT 'Please enter the -
              department number:'
ACCEPT        department_name PROMPT 'Please enter -
              the department name:'
ACCEPT        location PROMPT 'Please enter the -
              location:'
INSERT INTO   dept (deptno, dname, loc)
VALUES        (&department_id, '&department_name',
              '&location');
```

Copiando Linhas a partir de Outra Tabela

–Crie a instrução INSERT com uma subconsulta.

```
SQL> INSERT INTO managers(id, name, salary, hiredate)
  2                 SELECT empno, ename, sal, hiredate
  3                 FROM   emp
  4                 WHERE  job = 'MANAGER';
3 rows created.
```

–Não use a cláusula VALUES.

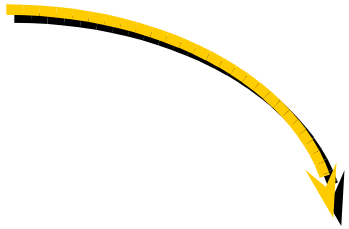
–Faça a correspondência do número de colunas na cláusula INSERT com o número de colunas na subconsulta.

Alterando os Dados em uma Tabela

EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		10
7566	JONES	MANAGER		20
...				

"...atualize uma linha em uma tabela EMP..."



EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		20
7566	JONES	MANAGER		20
...				

A instrução UPDATE

–Modifique linhas existentes com a instrução UPDATE.

```
UPDATE      tabela
SET         coluna = valor [, coluna = valor, ...]
[WHERE      condição];
```

–Atualize mais de uma linha por vez, se necessário.

Atualizando Linhas em uma Tabela

–Uma linha ou linhas específicas são modificadas quando você especifica a cláusula WHERE.

```
SQL> UPDATE   emp
  2  SET      deptno = 20
  3  WHERE    empno = 7782;
1 row updated.
```

Todas as linhas na tabela são modificadas quando você omite a cláusula WHERE.

```
SQL> UPDATE   employee
  2  SET      deptno = 20;
14 rows updated.
```

Atualizando com Subconsulta de Várias Colunas

- Atualize o cargo e o departamento do funcionário 7698 para coincidir com o do funcionário 7499.

```
SQL> UPDATE  emp
  2  SET     (job, deptno) =
  3                          (SELECT job, deptno
  4                          FROM    emp
  5                          WHERE   empno = 7499)
  6  WHERE   empno = 7698;
1 row updated.
```

Atualizando Linhas Baseadas em Outra Tabela

- Use subconsultas em instruções UPDATE para atualizar linhas em uma tabela baseada em valores de outra tabela.

```
SQL> UPDATE   employee
  2  SET      deptno =  (SELECT   deptno
  3                     FROM     emp
  4                     WHERE    empno = 7788)
  5  WHERE    job    =  (SELECT   job
  6                     FROM     emp
  7                     WHERE    empno = 7788);
2 rows updated.
```

Atualizando Linhas: Erro de Restrição de Integridade

```
SQL> UPDATE   emp
  2  SET      deptno = 55
  3  WHERE    deptno = 10;
```

```
UPDATE emp
       *
ERROR at line 1:
ORA-02291: integrity constraint (USR.EMP_DEPTNO_FK)
violated - parent key not found
```

- Não existe o número de departamento 55

Removendo uma Linha de uma Tabela

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	DEVELOPMENT	DETROIT
60	MIS	
...		

"... remova uma linha da tabela DEPT..."

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
60	MIS	
...		

A Instrução DELETE

Você pode remover linhas existentes de uma tabela usando a instrução DELETE.

```
DELETE [FROM]   tabela
[WHERE          condição];
```

Deletando Linhas de uma Tabela

–Linhas específicas são deletadas quando você especifica a cláusula WHERE.

```
SQL> DELETE FROM    department
  2  WHERE           dname = 'DEVELOPMENT';
1 row deleted.
```

-Todas as linhas na tabela serão deletadas se você omitir a cláusula WHERE.

```
SQL> DELETE FROM    department;
4 rows deleted.
```

Deletando Linhas Baseadas em Outra Tabela

- Use subconsultas em instruções DELETE para remover linhas de uma tabela baseadas em valores de outra tabela.

```
SQL> DELETE FROM    employee
  2  WHERE           deptno =
  3                         (SELECT   deptno
  4                          FROM     dept
  5                          WHERE    dname ='SALES');
6 rows deleted.
```

Deletando Linhas: Erro de Restrição de Integridade

```
SQL> DELETE FROM    dept
  2  WHERE          deptno = 10;
```

```
DELETE FROM dept
            *
ERROR at line 1:
ORA-02292: integrity constraint (USR.EMP_DEPTNO_FK)
violated - child record found
```

Você não pode deletar uma linha que contenha uma chave primária usada como chave estrangeira em outra tabela.

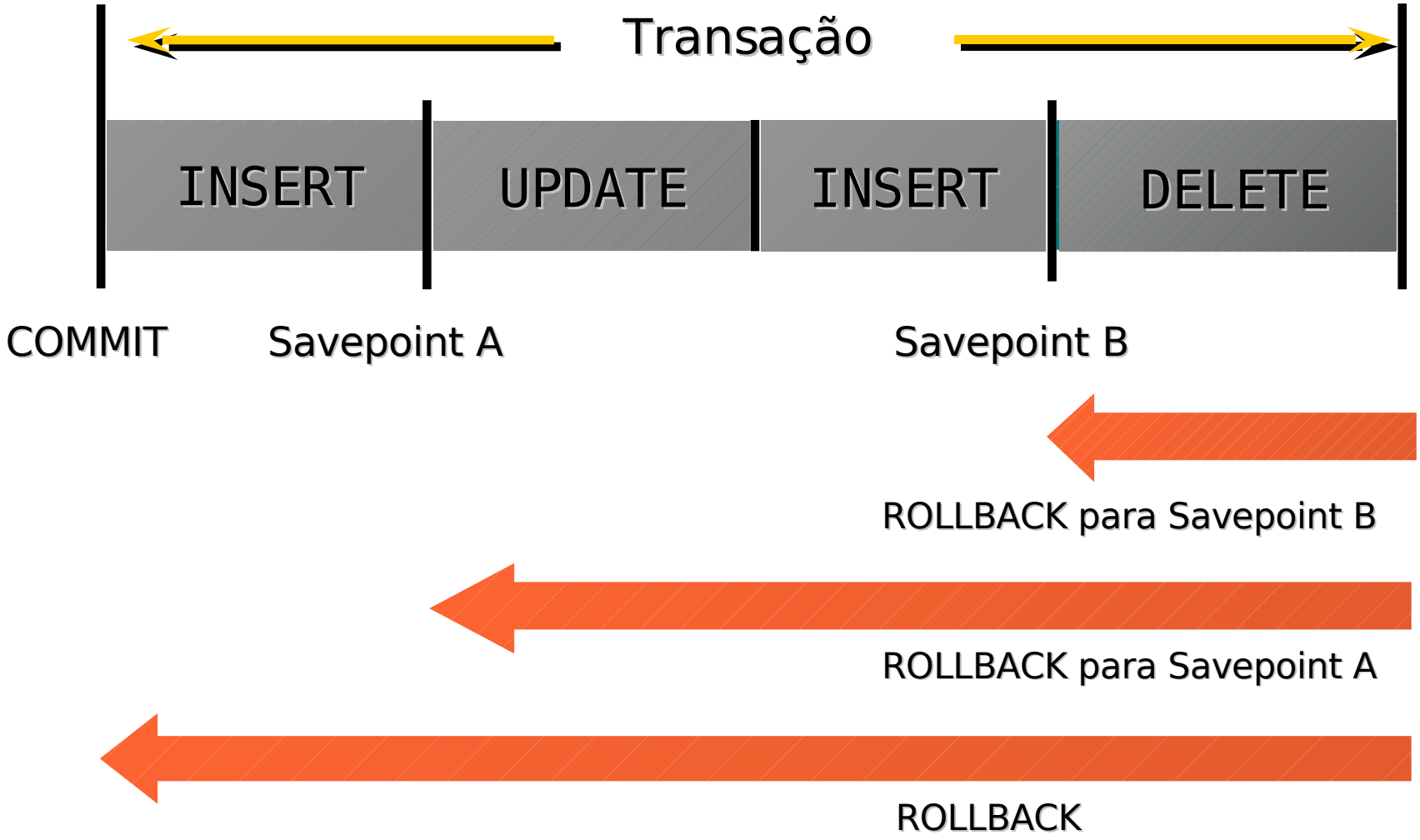
- Consistem de uma das seguintes instruções:
  - Instruções DML que fazem uma alteração consistente nos dados
  - Uma instrução DDL
  - Uma instrução DCL

- Começa quando for executada a primeira instrução SQL executável
- Termina com um dos seguintes eventos:
  - COMMIT ou ROLLBACK é emitida
  - Instrução DDL ou DCL é executada (commit automático)
  - O usuário sai
  - O sistema cai

Vantagens das Instruções COMMIT e ROLLBACK

- Garantir consistência de dados
- Visualizar alterações nos dados antes de fazer as alterações permanentemente
- Agrupar operações relacionadas logicamente

Controlando Transações



Estado dos Dados Antes de COMMIT ou ROLLBACK

- O estado anterior dos dados pode ser recuperado.
- O usuário atual pode revisar os resultados das operações DML usando a instrução SELECT.
- Outros usuários *não poderão* ver os resultados das instruções DML do usuário atual.
- As linhas afetadas são *bloqueadas*, outros usuários não poderão alterar os dados dentro das linhas afetadas.

Estado dos Dados Após COMMIT

- As alterações nos dados são feitas permanentemente no banco de dados.
- O estado anterior dos dados é perdido permanentemente.
- Todos os usuários podem ver os resultados.
- As linhas afetadas são desbloqueadas, essas linhas estão disponíveis para serem manipuladas por outros usuários.
- Todos os savepoints são apagados.

Submetendo Dados a Commit

Fazer as alterações.

```
SQL> UPDATE   emp
  2  SET      deptno = 10
  3  WHERE    empno = 7782;
1 row updated.
```

Submeter alterações a commit.

```
SQL> COMMIT;
Commit complete.
```

Estado dos Dados Após ROLLBACK

- Descarte todas as alterações pendentes usando a instrução ROLLBACK.
  - As alterações nos dados são desfeitas.
  - O estado anterior dos dados é restaurado.
  - As linhas afetadas são desbloqueadas.

```
SQL> DELETE FROM    employee;
14 rows deleted.
SQL> ROLLBACK;
Rollback complete.
```

Fazendo Roll Back de Alterações para um Marcador

–Crie um marcador em uma transação atual usando a instrução SAVEPOINT.

–Faça roll back do marcador usando a instrução ROLLBACK TO SAVEPOINT.

```
SQL> UPDATE...
SQL> SAVEPOINT update_done;
Savepoint created.
SQL> INSERT...
SQL> ROLLBACK TO update_done;
Rollback complete.
```

Rollback no Nível da Instrução

- Se uma única instrução DML falhar durante a execução, será feito roll back somente dessa instrução.
- O Oracle Server implementa um savepoint implícito.
- Todas as outras alterações são mantidas.
- O usuário deve finalizar as transações explicitamente usando uma instrução COMMIT ou ROLLBACK.

Consistência na Leitura

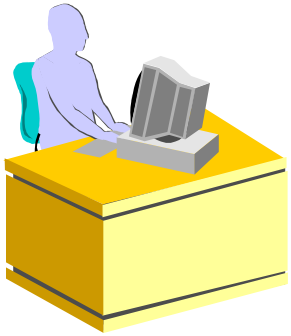
–A consistência na leitura garante sempre uma exibição consistente dos dados.

–As alterações feitas por um usuário não entram em conflito com as alterações feitas por outro usuário.

–A consistência na leitura garante que nos mesmos dados:

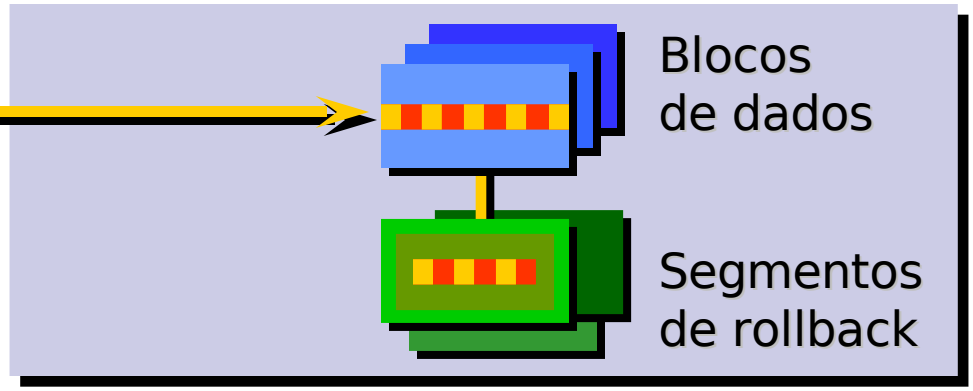
- Os leitores não esperem pelos autores
- Os autores não esperem pelos leitores

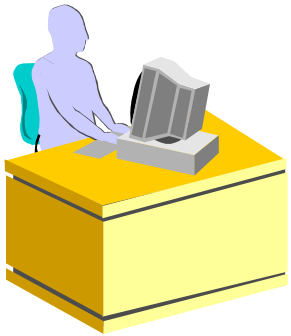
Implementação da Consistência na Leitura



Usuário A

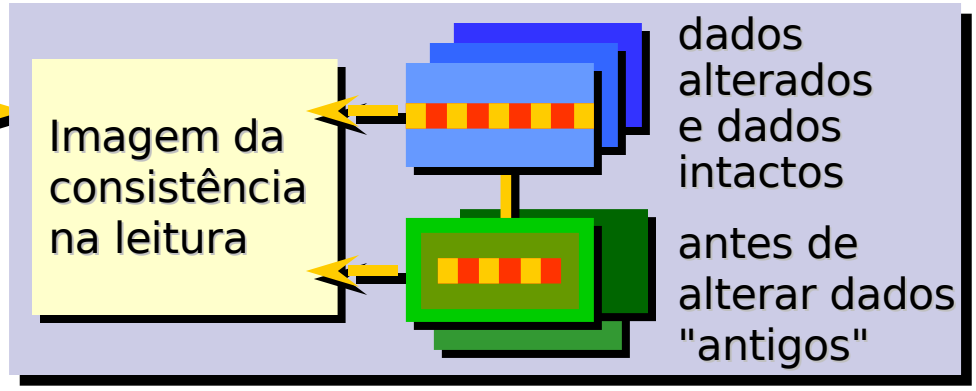
```
UPDATE emp
SET    sal = 2000
WHERE  ename =
       'SCOTT';
```





Usuário B

```
SELECT  *
FROM   emp;
```



Bloqueando

- Bloqueios::
  - Impedem a interação destrutiva entre transações simultâneas
  - Não requerem ação do usuário
  - Usam automaticamente o nível mais baixo de restrição
  - São mantidos durante a duração da transação
  - Há dois modos básicos:
    - Exclusivo
    - Compartilhado

Sumário

Instrução	Descrição
INSERT	Adiciona uma nova linha à tabela
UPDATE	Modifica linhas existentes na tabela
DELETE	Remove linhas existentes da tabela
COMMIT	Torna permanente todas as alterações pendentes
SAVEPOINT	Permite um rollback no marcador do savepoint
ROLLBACK	Descarta todas as alterações nos dados pendentes