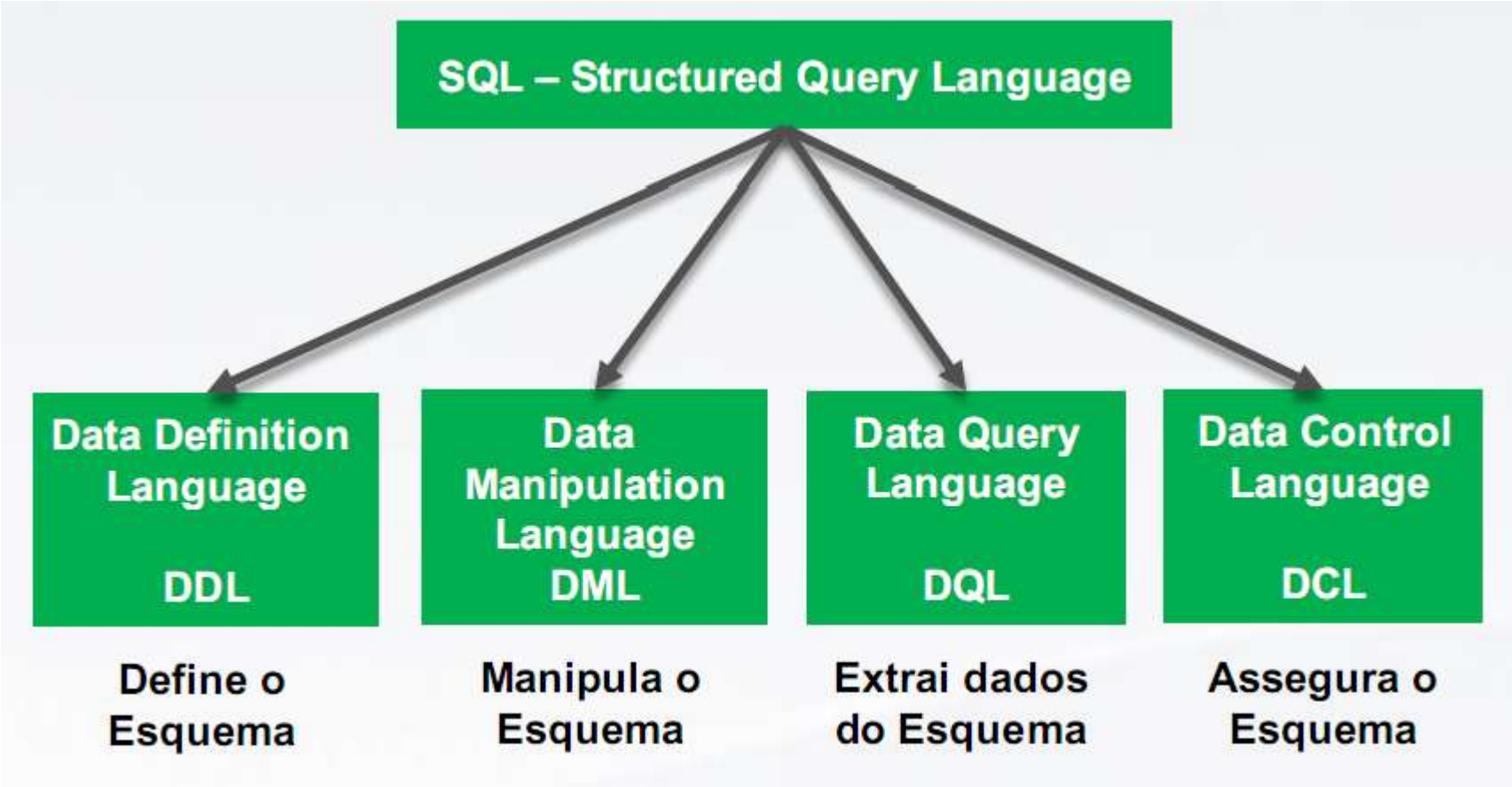
UEA
UNIVERSIDADE
DO ESTADO DO
AMAZONAS

Linguagem SQL

Banco de Dados

André Luiz do Vale Soares

SQL


SQL – Structured Query Language
Data Definition Language
DDL
Define o Esquema
Data Manipulation Language
DML
Manipula o Esquema
Data Query Language
DQL
Extrai dados do Esquema
Data Control Language
DCL
Assegura o Esquema

SQL

- DDL – Data Definition Language
  - Permite a criação dos componentes de BD, como tabelas, índices, etc.
  - Principais comandos:
  - CREATE TABLE
    - Cria uma nova tabela em um BD existente
  - ALTER TABLE
    - Altera uma tabela em um BD existente
  - DROP TABLE
    - Exclui uma tabela em um BD existente

SQL

- DML – Data Manipulation Language
  - Subconjunto de instruções usado para realizar inclusões, alterações e exclusões de dados presentes em registros de uma tabela.
  - Principais comandos:
  - INSERT
    - Insere novos registros em uma tabela
  - UPDATE
    - Atualiza dados já existentes
  - DELETE
    - Exclui registros de tabelas

SQL

- Data Query Language – DQL
  - Permite extrair dados do BD
  - Principal comando:
  - SELECT
    - Usado para realizar consultas a dados em tabelas

SQL

- Data Control Language – DCL
  - Provê segurança interna do BD
  - Principal comando:
  - CREAT USER
  - ALTER USER
  - CREAT SCHEMA

SQL

- Desenvolvida no início dos anos 70;
- Nos laboratórios da IBM;
- Uma linguagem de interface para um projeto de SGBD denominado SYSTEM R ;
- Objetivo: demonstrar a viabilidade da implementação do modelo relacional proposto por E. F. Codd.

SQL – Structured Query Language

– Instruções de DML, DDL, VDL, SDL

– Implementa a maior parte dos operadores da Álgebra Relacional

– Origem: SEQUEL (Structured English Query Language), uma linguagem de interface para um projeto de SGBD denominado SYSTEM R (IBM)

– SQL1 ou SQL86

– SQL2 ou SQL92 - usa expressões regulares de emparelhamento, *queries* recursivas e gatilhos (*triggers*)

– SQL3 : Conceitos XML, de Bancos de Dados Orientados a Objetos e Relacional Estendido.

SQL

- Embora padronizado pela ANSI e ISO, possui muitas variações e extensões.

- Tipicamente a linguagem pode ser migrada de plataforma para plataforma sem mudanças estruturais principais.

Recursos das Instruções SELECT SQL

Seleção

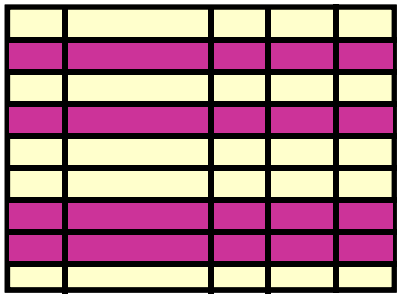


Tabela 1

Projeção

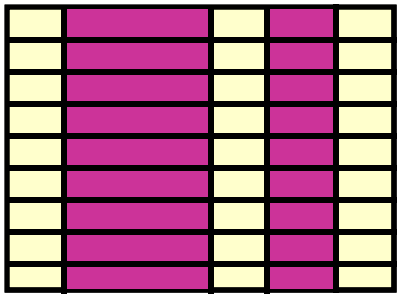


Tabela 1

Junção

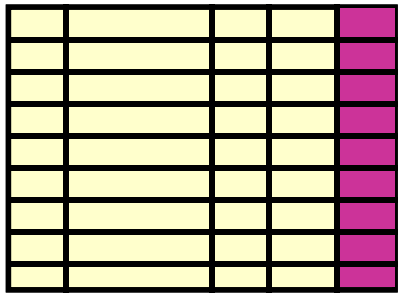


Tabela 1

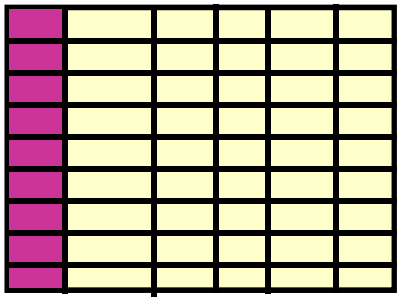


Tabela 2

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.

Projetando Todas as Colunas

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

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

Projetando Colunas Específicas (Projeção)

```
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

```
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 (Rebatizamento)

–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
```

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 (Seleção)

–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-YYYY.

```
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(lista)	Vincula qualquer um de uma lista de valores
LIKE	Vincula um padrão de caractere
IS NULL IS NOT NULL	É um valor nulo 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.

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]];
```

UNION

• Combina os resultados de dois SELECT em uma única tabela para todas as linhas correspondentes.

•As duas consultas devem ser compatíveis para união.

•Registros duplicados são automaticamente removidos a menos que use UNION ALL.

UNION - EXEMPLO

vendas2005

pessoa	quantia
João	1000
Alex	2000
Roberto	5000

vendas2006

person	amount
João	2000
Alex	2000
Isaque	35000

```
SELECT * FROM vendas2005
UNION
SELECT * FROM vendas2006;
```

pessoa	quantia
João	1000
Alex	2000
Roberto	5000
João	2000
Isaque	35000

```
SELECT * FROM vendas2005
UNION ALL
SELECT * FROM vendas2006;
```

pessoa	quantia
João	1000
João	2000
Alex	2000
Alex	2000
Roberto	5000
Isaque	35000

Exemplo: Wiki

INTERSECT

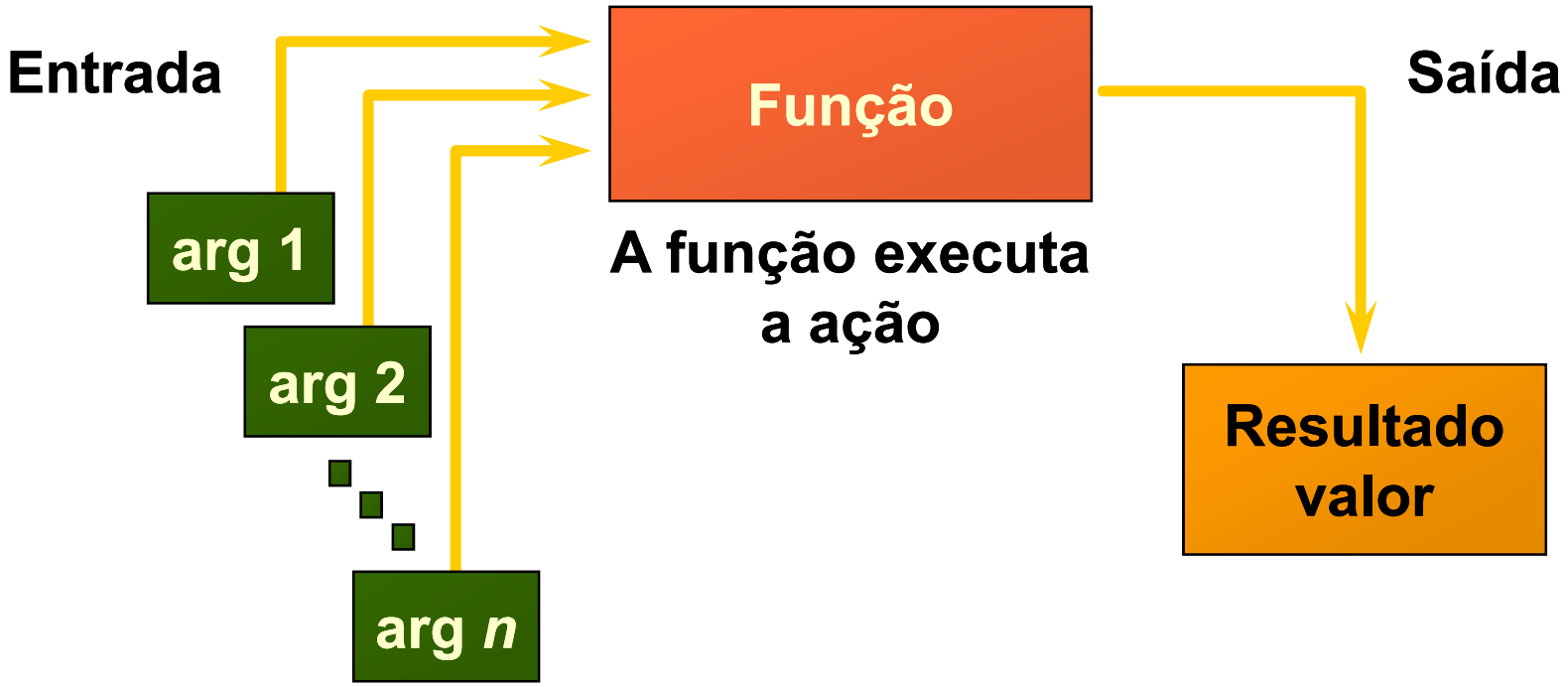
- Retorna a interseção entre o resultado de duas consultas.
- Remove linhas duplicadas do conjunto resultante final.

```
SELECT *
FROM   Ordens
WHERE  Quantidade BETWEEN 1 AND 100

INTERSECT

SELECT *
FROM   Ordens
WHERE  Quantidade BETWEEN 50 AND 200;
```

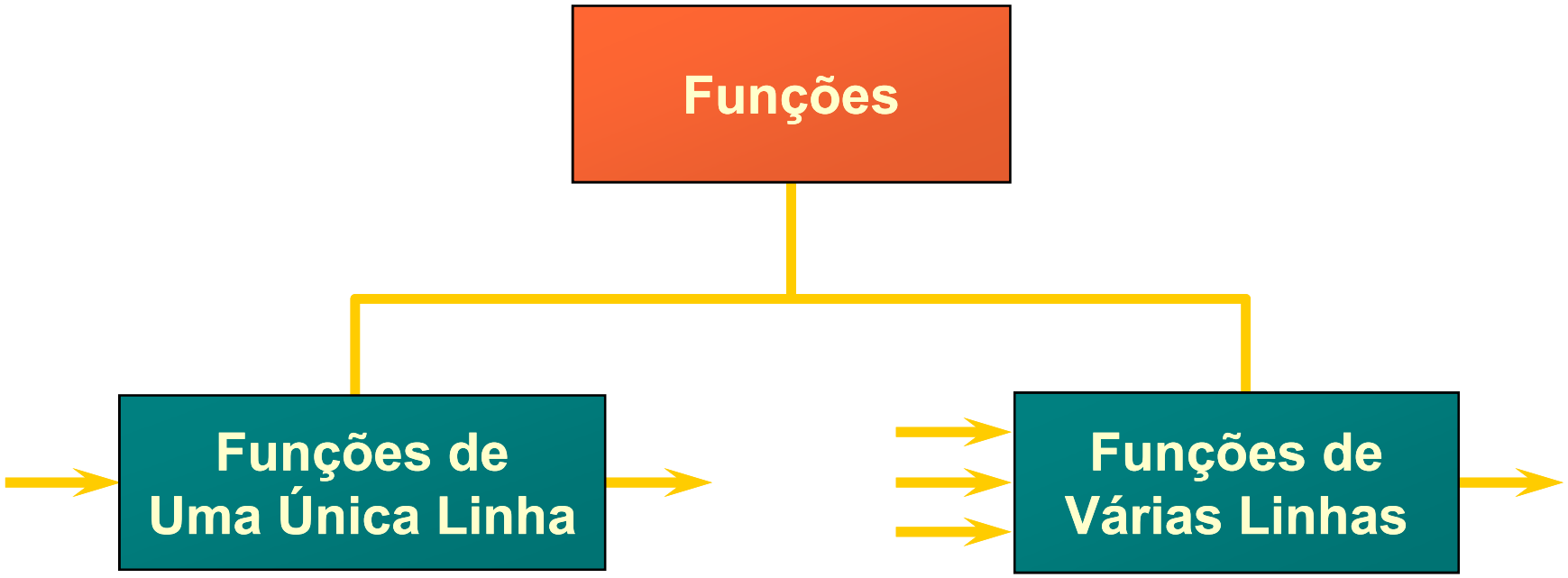
Exercícios

- Elabore consultas SQL que recuperem:
  1. Os nomes dos departamentos da Empresa em ordem crescente.
  2. Os nomes dos funcionários do sexo masculino da Empresa.
  3. Os nomes das esposas dos funcionários da empresa que estiverem no cadastro de dependentes.
  4. O nome da localização do projeto de número 5.
  5. O RG do gerente do departamento 'Contabilidade'.
  6. Os RGs dos empregados que trabalhem mais de 20 horas em qualquer projeto.
  7. Os números de projetos em que o empregado de RG 10 trabalha.
  8. Os nomes dos empregados dos sexo feminino, supervisionadas pelo empregado de RG 120.
  9. Os nomes dos departamentos gerenciados pelo empregado de RG 534 ou gerenciados a partir de '01-Janeiro-2004'.

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

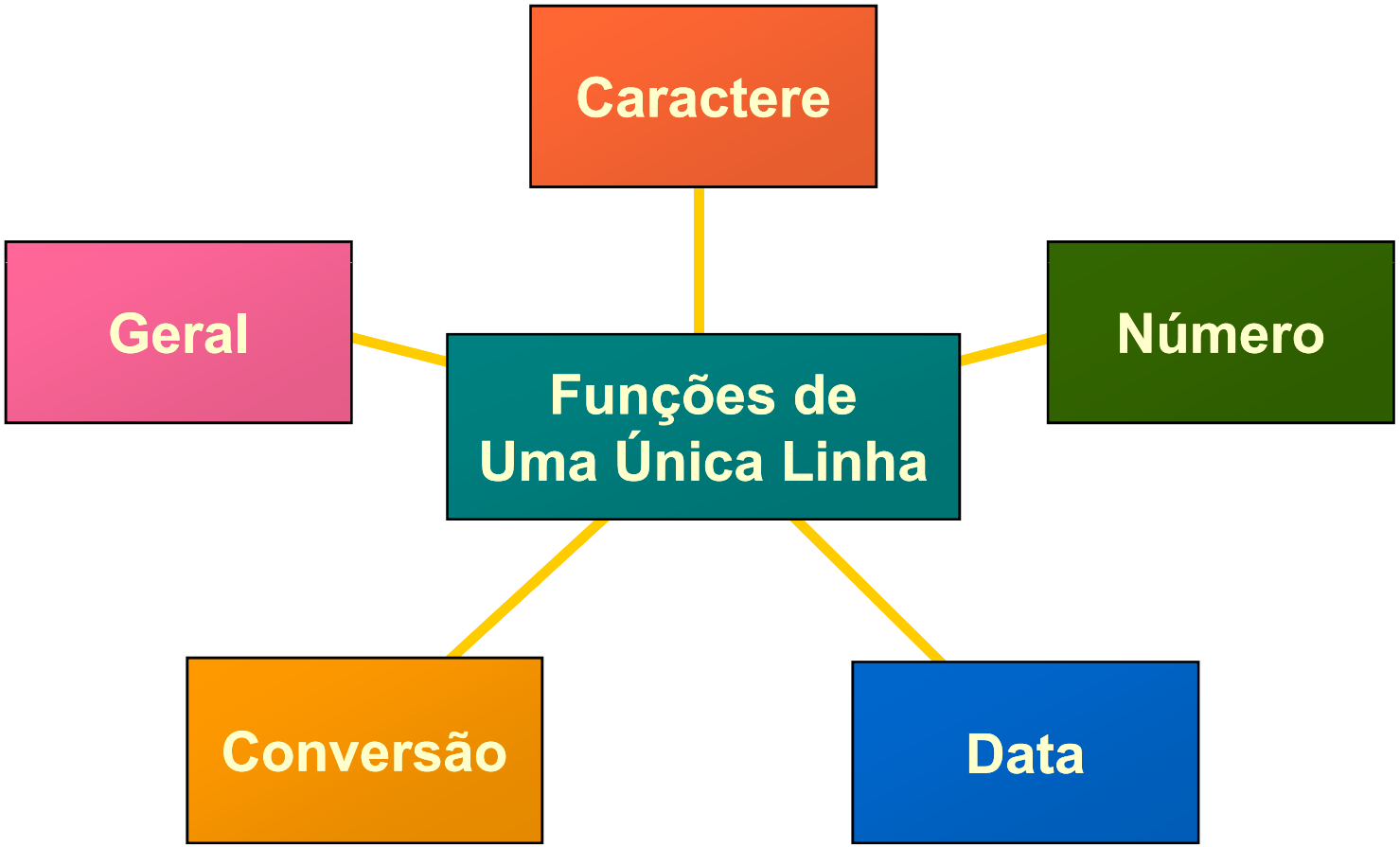
–Retornam um resultado por linha

–Podem modificar o tipo de dados

–Podem ser aninhadas

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

Funções de Uma Única Linha



Funções de Caractere

Funções de caractere

Funções de Conversão de Maiúsculas e Minúsculas

- LOWER
- UPPER
- INITCAP

Funções de manipulação de caractere

- CONCAT
- SUBSTR
- LENGTH
- INSTR
- LPAD
- TRIM
- REPLACE

Funções de Conversão de Maiúsculas e Minúsculas

Converter maiúsculas em minúsculas para strings de caractere

Função	Resultado
LOWER('SQL Course')	sql course
UPPER('SQL Course')	SQL COURSE
INITCAP('SQL Course')	Sql Course

Usando Funções de Conversão de Maiúsculas e Minúsculas

Exibir o número de funcionário, nome e número de departamento do funcionário Blake.

```
SQL> SELECT   empno, ename, deptno
  2  FROM     emp
  3  WHERE    ename = 'blake';
no rows selected
```

```
SQL> SELECT   empno, ename, deptno
  2  FROM     emp
  3  WHERE    ename = UPPER('blake');
```

```
     EMPNO ENAME          DEPTNO
---------- ---------- ----------
      7698 BLAKE              30
```

Funções de Manipulação de Caractere

Manipular strings de caractere

Função	Resultado
CONCAT('Good', 'String')	GoodString
SUBSTR('String',1,3)	Str
LENGTH('String')	6
INSTR('String', 'r')	3
LPAD(sal,10,'*')	******5000
TRIM('S' FROM 'SSMITH')	MITH
REPLACE('CAMA','A','$')	CM

Usando as Funções de Manipulação de Caractere

```
SQL> SELECT ename, CONCAT (ename, job), LENGTH(ename),
  2         INSTR(ename, 'A')
  3 FROM    emp
  4 WHERE   SUBSTR(job,1,5) = 'SALES';
```

ENAME	CONCAT(ENAME,JOB)	LENGTH(ENAME)	INSTR(ENAME,'A')
MARTIN	MARTINSALESMAN	6	2
ALLEN	ALLENSALESMAN	5	1
TURNER	TURNERSALESMAN	6	0
WARD	WARDSALESMAN	4	2

Funções Numéricas

–ROUND: Arredonda valor para determinado decimal

ROUND(45.926, 2) → 45.93

–TRUNC: Trunca valor para determinado decimal

TRUNC(45.926, 2) → 45.92

–MOD: Retorna o restante da divisão

MOD(1600, 300) → 100

Usando a Função ROUND

```
SQL> SELECT ROUND(45.923,2), ROUND(45.923,0),
  2         ROUND(45.923,-1)
  3  FROM   DUAL;
```

```
ROUND(45.923,2) ROUND(45.923,0) ROUND(45.923,-1)
--------------- --------------- ----------------
          45.92              46               50
```

Usando a Função TRUNC

```
SQL> SELECT TRUNC(45.923,2), TRUNC(45.923),
  2         TRUNC(45.923,-1)
  3  FROM   DUAL;
```

TRUNC(45.923,2)	TRUNC(45.923)	TRUNC(45.923,-1)
45.92	45	40

Usando a Função MOD

Calcular o restante da proporção do salário para comissão de todos os funcionários cujo cargo é **salesman**.

```
SQL> SELECT   ename, sal, comm, MOD(sal, comm)
  2  FROM     emp
  3  WHERE    job = 'SALESMAN';
```

ENAME	SAL	COMM	MOD(SAL,COMM)
MARTIN	1250	1400	1250
ALLEN	1600	300	100
TURNER	1500	0	1500
WARD	1250	500	250

O Que São Funções de Grupo?

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

"salário máximo na tabela EMP"

MAX(SAL)
5000

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]
[HAVING]
[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

EMP

DEPTNO	SAL	
10	2450	
10	5000	2916.6667
10	1300	
20	800	
20	1100	
20	3000	2175
20	3000	
20	2975	
30	1600	
30	2850	
30	1250	1566.6667
30	950	
30	1500	
30	1250	

"salário médio na tabela EMP para cada departamento"

DEPTNO	AVG(SAL)
10	2916.6667
20	2175
30	1566.6667

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

"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];
```

UEA
UNIVERSIDADE
DO ESTADO DO
AMAZONAS

GOVERNO DO ESTADO DO
AMAZONAS

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
```

Trabalhando com Datas

–O Oracle armazena datas em um formato numérico interno: século, ano, mês, dia, horas, minutos, segundos.

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

–SYSDATE é uma função de retorno de data e hora.

–DUAL é uma tabela fictícia usada para visualizar SYSDATE.

Aritmética com Datas

–Adicionar ou subtrair um número de, ou para, uma data para um valor de ***data*** resultante.

–Subtrair duas datas a fim de localizar o ***número*** de dias entre estas datas.

–Adicionar ***horas*** para uma data dividindo o número de dias por 24.

Usando Operadores Aritméticos com Datas

```
SQL> SELECT ename, (SYSDATE-hiredate)/7 WEEKS
  2  FROM   emp
  3  WHERE  deptno = 10;
```

```
ENAME          WEEKS
---------- ---------
KING       830.93709
CLARK      853.93709
MILLER     821.36566
```

Funções de Data

Função	Descrição
MONTHS_BETWEEN	Número de meses entre duas datas
ADD_MONTHS	Adiciona meses de calendário para a data
NEXT_DAY	Dia seguinte da data especificada
LAST_DAY	Último dia do mês
ROUND	Data de arredondamento
TRUNC	Data para truncada

Usando Funções de Data

- MONTHS_BETWEEN ('01-SEP-95','11-JAN-94') → 19.6774194
- ADD_MONTHS ('11-JAN-94',6) → '11-JUL-94'
- NEXT_DAY ('01-SEP-95','FRIDAY') → '08-SEP-95'
- LAST_DAY('01-SEP-95') → '30-SEP-95'

Usando Funções de Data

- ROUND('25-JUL-95','MONTH') → 01-AUG-95
- ROUND('25-JUL-95','YEAR') → 01-JAN-96
- TRUNC('25-JUL-95','MONTH') → 01-JUL-95
- TRUNC('25-JUL-95','YEAR') → 01-JAN-95

Função TO_CHAR com Datas

```
TO_CHAR(data, 'fmt')
```

- O modelo de formato
  - Deve estar entre aspas simples e fazer distinção entre maiúsculas e minúsculas
  - Pode incluir qualquer elemento de formato de data válido
  - Tem um elemento *fm* para remover espaços preenchidos ou suprimir zeros à esquerda
  - É separado do valor de data por uma vírgula

Elementos de Modelo de Formato de Data

YYYY	Valor de 4 dígitos para o Ano
YEAR	Ano por extenso
MM	Valor de dois dígitos para mês
MONTH	Mês por extenso
DY	Abreviação de três letras do dia da semana
DAY	Dia por extenso
DD	Valor de dois dígitos para o dia
D	Valor do dia da semana: 1:domingo,...,7:sábado

Elementos de Modelo de Formato de Data

- Elementos de hora formatam a parte de hora da data.

HH24:MI:SS AM	**15:45:32 PM**

- Adicionar strings de caractere incluindo-as entres aspas.

DD "of" MONTH	**12 of OCTOBER**

- Sufixos de número escrevem os números por extenso.

ddspth	**fourteenth**

Usando a Função TO_CHAR com Datas

```
SQL> SELECT ename,
  2         TO_CHAR(hiredate, 'fmDD Month YYYY') HIREDATE
  3  FROM   emp;
```

```
ENAME      HIREDATE
---------- ------------------
KING       17 November 1981
BLAKE      1 May 1981
CLARK      9 June 1981
JONES      2 April 1981
MARTIN     28 September 1981
ALLEN      20 February 1981
...
14 rows selected.
```

Função TO_CHAR com Números

```
TO_CHAR(número, 'fmt')
```

Use estes formatos com a função TO_CHAR para exibir um valor de número como um caractere:

9	Representa um número
0	Obriga que um 0 seja mostrado
$	Coloca um sinal de dólar flutuante
L	Usa o símbolo da moeda local flutuante
.	Imprime um ponto decimal
,	Imprime um indicador de milhar

Usando a Função TO_CHAR com Números

```
SQL> SELECT   TO_CHAR(sal,'$99,999') SALARY
  2  FROM     emp
  3  WHERE    ename = 'SCOTT';
```

```
SALARY
--------
  $3,000
```

Funções TO_NUMBER e TO_DATE

Converter uma string de caractere para um formato de número usando a função TO_NUMBER

```
TO_NUMBER(carac[, 'fmt'])
```

Converter uma string de caractere para um formato de data usando a função TO_DATE

```
TO_DATE(carac[, 'fmt'])
```

Função NVL

- Converte nulo para um valor real
  - –Os tipos de dados que podem ser usados são data, caractere e número.
  - –Os tipos de dados devem corresponder com
    - NVL(comm,0)
    - NVL(hiredate,'01-JAN-97')
    - NVL(job,'No Job Yet')

Usando a Função NVL

```
SQL> SELECT ename, sal, comm, (sal*12)+NVL(comm,0)
  2  FROM   emp;
```

```
ENAME             SAL       COMM (SAL*12)+NVL(COMM,0)
---------- --------- --------- --------------------
KING             5000                         60000
BLAKE            2850                         34200
CLARK            2450                         29400
JONES            2975                         35700
MARTIN           1250      1400                16400
ALLEN            1600       300                19500
...
14 rows selected.
```

Aninhando Funções

```
SQL> SELECT   ename,
  2           NVL(TO_CHAR(mgr),'No Manager')
  3   FROM    emp
  4   WHERE   mgr IS NULL;
```

```
ENAME      NVL(TO_CHAR(MGR),'NOMANAGER')
---------- ------------------------------
KING       No Manager
```

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 duas 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.

Junção com ANSI SQL

- INNER JOIN.

```
SELECT   tabela1.coluna, tabela2.coluna
FROM     tabela1 inner join tabela2
ON       tabela1.coluna1 = tabela2.coluna2;
```

–Criar uma condição de junção na cláusula ON.

–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.
```

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.

Chave estrangeira

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 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.
```

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;
```

Junção Interna

- **Junção Interna:** todas linhas de uma tabela se relacionam com todas as linhas de outras tabelas se elas tiverem ao menos 1 campo em comum

```
SELECT e.nome, s.local
FROM eleitor e, secao s
WHERE e.secao = s.numero;
```

Unindo Mais de Duas Tabelas

CUSTOMER

NAME	CUSTID
JOCKSPORTS	100
TKB SPORT SHOP	101
VOLLYRITE	102
JUST TENNIS	103
K+T SPORTS	105
SHAPE UP	106
WOMENS SPORTS	107
...	...

9 rows selected.

ORD

CUSTID	ORDID
101	610
102	611
104	612
106	601
102	602
106	
106	
...	

21 rows s

ITEM

ORDID	ITEMID
610	3
611	1
612	1
601	1
602	1
...	

64 rows selected.

Junção com mais de duas tabelas

```
SELECT   c.name, c.custid, o.ordid, i.itemid
FROM     customer c, order o, item i
WHERE    c.cust_id = o.cust_id
AND      o.ordid = i.ordid
```

Junção com mais de duas tabelas

```
SELECT e.nome, c.nome, v.ano

FROM eleitor e, votacao v, candidato c

WHERE v.eleitor = e.titulo AND
          v.candidato = c.numero
```

- Formas junção:
  - a explícita: JOIN
  - a implícita: utiliza ',' para separar as tabelas a combinar na cláusula FROM do SELECT. Então sempre é gerado o produto cruzado do qual são selecionadas as combinações que cumpram a cláusula WHERE.

```
SELECT e.nome, s.local

FROM eleitor e INNER JOIN  secao s

ON e.secao = s.numero;
```

Junção com mais de duas tabelas – ANSI SQL

```
SELECT    c.name, c.custid, o.ordid, i.itemid
FROM      (customer c inner join order o
ON        c.cust_id = o.cust_id)
          inner join item i
ON        o.ordid = i.ordid
```

UEA
UNIVERSIDADE
DO ESTADO DO
AMAZONAS

```
SELECT e.nome, c.nome, v.ano
FROM (eleitor e INNER JOIN votacao V
ON      v.eleitor = e.titulo  ) INNER JOIN candidato c
ON      v.candidato = c.numero
```

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

EMP

ENAME	DEPTNO
KING	10
BLAKE	30
CLARK	10
JONES	20
...	

DEPT

DEPTNO	DNAME
10	ACCOUNTING
30	SALES
10	ACCOUNTING
20	RESEARCH
...	
40	OPERATIONS

Nenhum funcionário do departamento OPERATIONS

Junção Externa

- É uma seleção que não requer que os registros de uma tabela possuam registros equivalentes em outra.
- O registro é mantido na pseudo-tabela se não existe outro registro que lhe corresponda.

Junção Externa

- Subdivisão:
  - dependendo da tabela do qual admitiremos os registros que não possuem correspondência:
    - a tabela esquerda (**Left Outer Join)**,
    - a direita (**Right Outer Join**)
    - ou ambas (**Full Outer Join**).

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(+);
```

Left Outer Join

- Resultado: sempre contém todos os registros da tabela esquerda (a primeira tabela mencionada), mesmo quando não exista registros correspondentes na tabela direita.

- Retorna todos os valores da tabela esquerda com os valores da tabela direita correspondente, ou NULL (quando não há correspondência).

Left Outer Join

- Ex.:
  - Seleção trará todos os dados da tabela Cliente, e os correspondentes na tabela Profissao, e quando não houver estes correspondentes, trará o valor NULL.

```
SELECT distinct *
FROM Cliente LEFT OUTER JOIN Profissao
ON Cliente.Profissao=Profissao.Codigo;
```

Rigth Outer Join

- Retorna: todos os registros da tabela à direita (a segunda tabela mencionada na consulta), mesmo se não existir registro correspondente na tabela à esquerda.
- O valor NULL é retornado quando não há correspondência.

Rigth Outer Join

- Ex.: há diversas Profissões com códigos que não possuem correspondentes na tabela Clientes. Esta consulta traz todas estas Profissões mesmo que não haja esta correspondência:O valor NULL é retornado.

```
SELECT *
FROM Cliente RIGHT OUTER JOIN Profissao
ON Cliente.Profissao = Profissao.Codigo;
```

Full Outer Join

- apresenta todos os dados das tabelas à esquerda e à direita, mesmo que não possuam correspondência em outra tabela.

```
SELECT *
FROM Cliente FULL OUTER JOIN Profissao
ON Cliente.Profissao=Profissao.Codigo;
```

Junções Externas – ANSI SQL

- OUTER JOIN
- LEFT, RIGHT ou FULL.

```
SELECT  tabela1.coluna, tabela2.coluna
FROM    tabela1 right outer join tabela2
ON      tabela1.coluna = tabela2.coluna;
```

```
SELECT  tabela1.coluna , tabela2.coluna
FROM    tabela1 left outer join tabela2
ON      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.
```

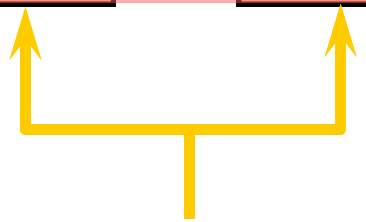
Autojunções

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"

UEA
UNIVERSIDADE
DO ESTADO DO
AMAZONAS

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.
```

Exercícios

- Elabore consultas SQL para listar:
  1. Os nomes dos empregados que trabalham no departamento 'Administração'
  2. Os nomes e parentescos dos dependentes do empregado 'Patrick Silva de Menezes'
  3. O nome do gerente do departamento 'Informática'
  4. As Localidades do departamento 'Contabilidade'
  5. Os nomes dos projetos em que o empregado 'Patrick Silva de Menezes' trabalha
  6. Os nomes e parentescos dos dependentes dos empregados do departamento 'Informática'

Exercícios

- Elabore consultas SQL para listar:
  7. Os nomes dos empregados supervisionados por 'Mariana Baracho'
  8. Os nomes dos projetos em que os empregados do departamento 'Informática' trabalham
  9. Os nomes dos supervisores dos empregados que trabalham no projeto 'Downsizing'
  10. Os nomes dos empregados que trabalham em departamentos com localizações em 'Manaus'
  11. Os nomes dos gerentes dos departamentos que controlam projetos em que o empregado 'Paulo Salim' trabalha
  12. Os nomes dos empregados do departamento 'Informática' e de suas esposas, caso existam.

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
```

UEA
UNIVERSIDADE
DO ESTADO DO
AMAZONAS

Subconsultas

- Quais candidatos são do mesmo partido do candidato Serafim?

```
SELECT `nome`, `partido`
FROM `candidato`
WHERE `partido` =
        (SELECT `partido` FROM `candidato`
          WHERE `nome`='Serafim');
```

UEA
UNIVERSIDADE
DO ESTADO DO
AMAZONAS

Subconsultas

- Todos os eleitores que nasceram no mesmo ano da Antonia Souza?

```
SELECT `nome`,DATE_FORMAT(`data_nascimento`,'%d/%m/%Y')
FROM `eleitor` WHERE EXTRACT(YEAR FROM `data_nascimento`) =
        (SELECT EXTRACT(YEAR FROM `data_nascimento`)
         FROM eleitor WHERE `nome` LIKE '%antonia souza%');
```

Diretrizes para o Uso de Subconsultas

–Coloque as subconsultas entre parênteses.

–Coloque as subconsultas no lado direito do operador de comparação.

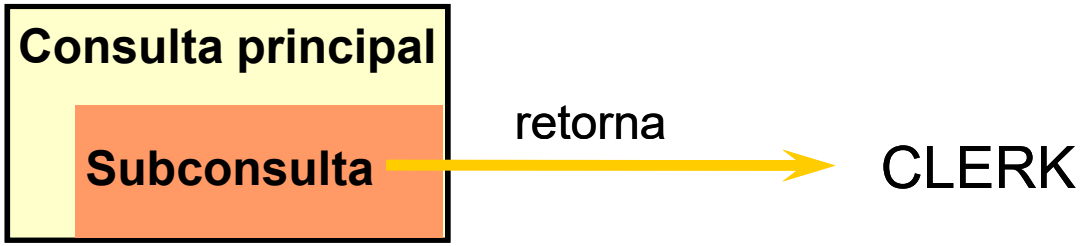
–Não adicione uma cláusula ORDER BY a uma subconsulta.

–Use operadores de uma única linha com subconsultas de uma única linha.

–Use operadores de várias linhas com subconsultas de várias linhas.

Tipos de Subconsultas

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

Operador	Significado
=	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

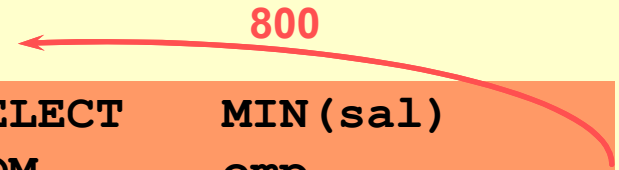
```
ENAME      JOB              SAL
---------- --------- ----------
SMITH      CLERK            800
```

Cláusula HAVING com Subconsultas

–O SGBD primeiro executa as subconsultas.

–O SGBD 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
```

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 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;
```

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.
```

Exercícios

- Elabore consultas utilizando linguagem SQL para listar:
  1. O nome do empregado mais jovem da empresa
  2. O nome do departamento que controla o maior número de projetos
  3. O total de horas trabalhadas nos projetos, por empregado
  4. O nome do empregado que possui maior número de dependentes
  5. As quantidades de empregados dos sexos masculino e feminino que trabalham na empresa
  6. O nome do departamento que mais gasta com pagamento de salários da empresa
  7. O nome de cada empregado da empresa e, se for o caso, o departamento que o mesmo gerencia

Manipulação de Dados

Objetivos:

–Descrever cada instrução DML

–Inserir linhas em uma tabela

–Atualizar linhas em uma tabela

–Deletar linhas de uma tabela

–Criação de tabelas

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

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.

Criando Tabelas

Crie a tabela.

```
SQL> CREATE TABLE dept
  2          (deptno  NUMBER(2),
  3           dname   VARCHAR2(14),
  4           loc     VARCHAR2(13));
Table created.
```

Confirme a criação da tabela

```
SQL> DESCRIBE dept
```

Name	Null?	Type
DEPTNO		NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)

Tipos de Dados

Tipo de Dados	Descrição
VARCHAR(*tamanho*)	**Dados de caractere de comprimento variável**
CHAR(*tamanho*)	**Dados de caractere de comprimento fixo**
NUMBER(*p,s*)	**Dados numéricos de comprimento variável**
DATE , DATETIME, TIMESTAMP, TIME, YEAR	**Valores de data e hora**
LONG	**Dados de caractere de comprimento variável até 2 gigabytes**
BLOB, MEDIUMBLOB, LONGBLOB	**Dados binários de até 4 gigabytes**:imagens, pdf, word, vídeos

https://dev.mysql.com/doc/refman/5.7/en/data-types.html

Criando uma Tabela Usando uma Subconsulta

```
SQL> CREATE TABLE   dept30
  2  AS
  3     SELECT    empno, ename, sal*12 ANNSAL, hiredate
  4     FROM      emp
  5     WHERE     deptno = 30;
Table created.
```

```
SQL> DESCRIBE dept30
```

```
Name                                      Null?    Type
----------------------------------------- -------- -----
EMPNO                                     NOT NULL NUMBER(4)
ENAME                                              VARCHAR2(10)
ANNSAL                                             NUMBER
HIREDATE                                           DATE
```

Adicionando uma Coluna

Use a cláusula ADD para adicionar colunas

```
SQL> ALTER TABLE dept30
  2  ADD        (job VARCHAR2(9));
Table altered.
```

A coluna nova torna-se a última coluna

```
    EMPNO ENAME          ANNSAL HIREDATE  JOB
--------- ---------- ---------- --------- ----
     7698 BLAKE           34200 01-MAY-81
     7654 MARTIN          15000 28-SEP-81
     7499 ALLEN           19200 20-FEB-81
     7844 TURNER          18000 08-SEP-81
...
6 rows selected.
```

Modificando uma Coluna

–Você pode alterar um tipo de dados, tamanho e valor default de uma coluna.

```
SQL> ALTER TABLE    dept30
  2  MODIFY          (ename VARCHAR2(15));
Table altered.
```

–Uma alteração no valor default afeta somente as inserções subseqüentes à tabela.

Eliminando uma Coluna

Use a cláusula DROP COLUMN para eliminar colunas que você não precisa mais na tabela.

```
SQL> ALTER TABLE    dept30
  2  DROP  COLUMN   job ;
Table altered.
```