Dados da instalação

1 - Instalar Oracle 10g Express

2 - Instalar Toad 8.5 \ Oracle SQL Developer

3 - Escolher schema HR

Instância = Oracle 10g

usuários = sys e system

pwd = delphi

regras de precedência:

1-operadores aritméticos

2-operadores de concatenação

3-condicionais

4-is [not] null, like, [not] in

5-[not] between

6-not equal to

7-NOT

8-AND

9-OR

Campos nulos são calculados normalmente usando operações aritméticas

```
select job_id, last_name, salary, salary+commission_pct
from hr.employees
```

Alias são usados com AS quando uma palavra e entre aspas com duas ou mais;

```
select job_id AS Job, last_name, salary, salary+commission_pct As "Salary Anual"
from hr.employees
```

Selecionando registros que não são nulos

```
select job_id, last_name ||', '|| substr(first_name,1,1) as Autor, manager_id
from hr.employees
where manager_id is not null
```

Concatenação de campos first_name ||' '|| last_name as Fullname

```
SELECT
        first_name ||' '|| last_name as Fullname,
        hire_date,
        salary,
        salary * 12 as salary_anual,
        department_name
FROM HR.EMPLOYEES emp, HR.DEPARTMENTS depart
WHERE salary like '2%' and
       emp.department_id = depart.department_id

select department_name ||
    q'[, it's asssigned Manager ID:]'
    As "Department and Manager"
from hr.departments
```

o comando describe é usado para descrever estrutura das tabelas

```
desc hr.departments
```

```
SELECT
        sum(salary),
        department_name
FROM HR.EMPLOYEES emp, HR.DEPARTMENTS depart
      WHERE emp.department_id = depart.department_id
GROUP BY department_name
HAVING sum(salary) > 60000



SELECT
        departments.department_name,
    employees.email,
    employees.hire_date
FROM
    HR.DEPARTMENTS departments INNER JOIN HR.employees employees
   ON departments.department_id = employees.department_id
WHERE employees.hire_date = (SELECT MAX(hire_date)
                    FROM hr.employees
                   WHERE departments.department_id = department_id)
GROUP BY
     departments.department_name,
     employees.email,
     employees.hire_date
ORDER BY
   departments.department_name
```

USANDO VARIÁVEIS

```
select last_name, job_id, salary
from hr.employees
where salary > &salary
```

```
select last_name, job_id, &nomeColuna
from hr.employees
where &condicao
order by &colunaOrdenada
```

Reusando a mesma variável

```
select last_name, job_id, &&nomeColuna
from hr.employees
where &nomeColuna > 13000
order by &nomeColuna
```

Utilizando constantes

```
set verify on
define valor = 13000
select last_name, job_id, salary
from hr.employees
where salary > &valor
```

```
SELECT
        first_name ||' '|| last_name as Fullname,
```

```
        hire_date,
        salary,
        salary * 12 as salary_anual,
        department_name
FROM HR.EMPLOYEES emp, HR.DEPARTMENTS depart
WHERE salary like '2%' and
        emp.department_id = depart.department_id and
        hire_date between '15-jan-1999' and '18-mar-2000'
```

FUNÇÕES CARACTER

LOWER('CURSO') = curso

UPPER('curso') = CURSO

INITCAP('CURSO') = Curso

FUNÇÕES DE MANIPULAÇÃO DE CARACTER

CONCAT('Good', 'String') = 'Good String'

SUBSTR('String',1,3) = 'Str'

LENGHT('String') = 6

INSTR('String','r') = 3

LPAD(salary,10,'*') = ******5000

RPAD(salary,10,'*') = 5000******

TRIM('S' from 'SSmith') = mith

REPLACE ('JACK AND JLUE', 'J', 'BL' = ‘BLACK AND BLUE’

FUNÇÕES DE MANIPULAÇÃO NÚMEROS

ROUND()

TRUNC

MOD

FUNÇÕES DE DATA

SYSDATE

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'

CONVERSÃO DE TIPOS

TO_CHAR

```
SELECT hire_date,
        to_char(hire_date, 'fmDD Month YYYY') as fulldate
FROM HR.EMPLOYEES emp, HR.DEPARTMENTS depart
WHERE emp.department_id = depart.department_id

SELECT
          salary,
          to_char(salary,'$999,999,999.99') as Real
FROM HR.EMPLOYEES emp, HR.DEPARTMENTS depart
WHERE emp.department_id = depart.department_id
select
  last_name ||', '|| substr(first_name,1,1) as Empregado,
  to_char(hire_date,'DD/mon/YYYY') as Contratação,
  jobs.job_title as Cargo
from
  hr.employees emp inner join hr.jobs jobs
    on emp.job_id = jobs.job_id
```

```
where
  manager_id is not null
```

TO_DATE

```
select
  last_name ||', '|| substr(first_name,1,1) as Empregado,
  to_char(hire_date,'DD/mon/YYYY') as Contratação,
  jobs.job_title as Cargo
from
  hr.employees emp inner join hr.jobs jobs
    on emp.job_id = jobs.job_id
where
  hire_date > to_date('15-set-1999','dd-mon-yyyy')
```

TO_NUMBER

NVL – retorna um valor padrão caso o conteúdo do campo seja nulo, porém deve ser respeitado o tipo de dados.

```
select
  last_name ||', '|| substr(first_name,1,1) as Empregado,
  to_char(hire_date,'DD/mon/YYYY') as Contratação,
  jobs.job_title as Cargo,
  nvl(emp.manager_id,0) as Gerente
from
  hr.employees emp inner join hr.jobs jobs
    on emp.job_id = jobs.job_id
```

FUNÇÃO IF/CASE (DECODE)

```
SELECT
        job_id as Profissional,
        to_char(salary,'$999,999,999.99') as salaryreal,
        to_char(decode(job_id, 'IT_PROG', salary*1.1,
                                'CLERK', salary*1.15,
                                'MANAGER', salary*1.20,
                                salary),'$999,999,999.99')  AS Salary

FROM HR.EMPLOYEES emp, HR.DEPARTMENTS depart
WHERE emp.department_id = depart.department_id
```

DECODE (MELHOR QUE CASE)

```
select
  last_name ||', '|| substr(first_name,1,1) as Empregado,
  emp.job_id,
  emp.salary,
  decode(emp.job_id,
     'IT_PROG', salary *1.15,
     'ST_CLERK', salary *1.25,
     'SA_REP', salary *1.35,
     salary) Corrigido
from
  hr.employees emp inner join hr.jobs jobs
```

```
   on emp.job_id = jobs.job_id
```

FUNÇÕES DE AGREGAÇÃO

```
select

  emp.job_id,

  MAX(emp.salary), MIN(emp.salary), AVG(emp.salary), trunc(STDDEV(emp.salary),2),
SUM(emp.salary), COUNT(emp.salary)

from

  hr.employees emp inner join hr.jobs jobs

    on emp.job_id = jobs.job_id

GROUP BY emp.job_id
```

HAVING

```
select

  emp.job_id,

  MAX(emp.salary), MIN(emp.salary), AVG(emp.salary), trunc(STDDEV(emp.salary),2),
SUM(emp.salary), COUNT(emp.salary)

from

  hr.employees emp inner join hr.jobs jobs

    on emp.job_id = jobs.job_id

GROUP BY emp.job_id

HAVING MAX(emp.salary) > 8000 AND MIN(emp.salary) > 2000
```

TIPOS DE JOINS:

EQUIJOINS

NON EQUIJOINS

INNER JOINS

LEFT OTHER JOINS

RIGHT OTHER JOINS

CROSS JOINS

NATURAL JOINS

SELFJOINS

NATURAL JOIN (MAIS RÁPIDO QUE INNER JOIN and ON)

```
select
  job_id,
  MAX(salary), MIN(salary), AVG(salary), trunc(STDDEV(salary),2), SUM(salary), COUNT(salary)
from
  hr.employees NATURAL join hr.jobs
GROUP BY job_id
```

USING (MAIS RÁPIDO QUE NATURAL JOIN)

```
select
  job_id,
  MAX(salary), MIN(salary), AVG(salary), trunc(STDDEV(salary),2), SUM(salary), COUNT(salary)
from
  hr.employees join hr.jobs
using(job_id)
GROUP BY job_id
```

RANK

```
SELECT
        rownum as Rank,
        last_name||','||first_name as Name,
        job_id as Profissional,
        to_char(salary,'$999,999,999.99') as Salary,
        to_char(decode(job_id, 'IT_PROG', salary*1.1,
                                'CLERK', salary*1.15,
                                'MANAGER', salary*1.20,
                                salary),'$999,999,999.99') as Growth

FROM HR.EMPLOYEES emp, HR.DEPARTMENTS depart
WHERE emp.department_id = depart.department_id and
      rownum <=10
ORDER BY salary DESC



CREATE PROCEDURE adjust_salary(emp_id NUMBER, sal IN OUT NUMBER)
            IS emp_job VARCHAR2(10);
                avg_sal NUMBER(8,2);
BEGIN

SELECT job_id INTO emp_job FROM HR.employees WHERE employee_id = emp_id;


SELECT AVG(salary) INTO avg_sal FROM HR.employees WHERE job_id = emp_job;
```

```
DBMS_OUTPUT.PUT_LINE ('The average salary for ' || emp_job || ' employees: ' ||
TO_CHAR(avg_sal));


sal := (sal + avg_sal)/2; -- adjust sal value which is returned

END;


 --TESTING PROCEDURE


DECLARE


new_sal hr.employees.SALARY%type;

emp_id hr.employees.EMPLOYEE_ID%type;


BEGIN


SELECT AVG(salary) INTO new_sal FROM hr.employees;


emp_id := 120;


DBMS_OUTPUT.PUT_LINE ('The average salary for all employees: ' || TO_CHAR(new_sal));


adjust_salary(emp_id, new_sal); -- assigns a new value to new_sal


DBMS_OUTPUT.PUT_LINE ('The adjusted salary for employee ' || TO_CHAR(emp_id)|| ' is ' ||
TO_CHAR(new_sal)); -- sal has new value


END;
```

```
--LOOP



BEGIN

FOR someone IN (SELECT * FROM hr.employees WHERE employee_id < 120 )

        LOOP

                DBMS_OUTPUT.PUT_LINE('First name = ' || someone.first_name || ', Last
name = ' || someone.last_name);

        END LOOP;

END;



--CURSOR

DECLARE

CURSOR c1 IS
        SELECT last_name, salary, hire_date, job_id FROM employees
        WHERE employee_id = 120;

-- declare record variable that represents a row fetched from the employees table
```

```
employee_rec c1%ROWTYPE;

BEGIN

-- open the explicit cursor and use it to fetch data into employee_rec

  OPEN c1;

                FETCH c1 INTO employee_rec;

                        DBMS_OUTPUT.PUT_LINE('Employee name: ' ||
employee_rec.last_name);

END;

DESCRIBE DEPARTMENTS

select object_name from user_objects where object_type = 'TABLE'

user_tables

user_catalog
```