

SUPERSPORTS

DIAGRAMA DE ESTRUTURA DE DADOS

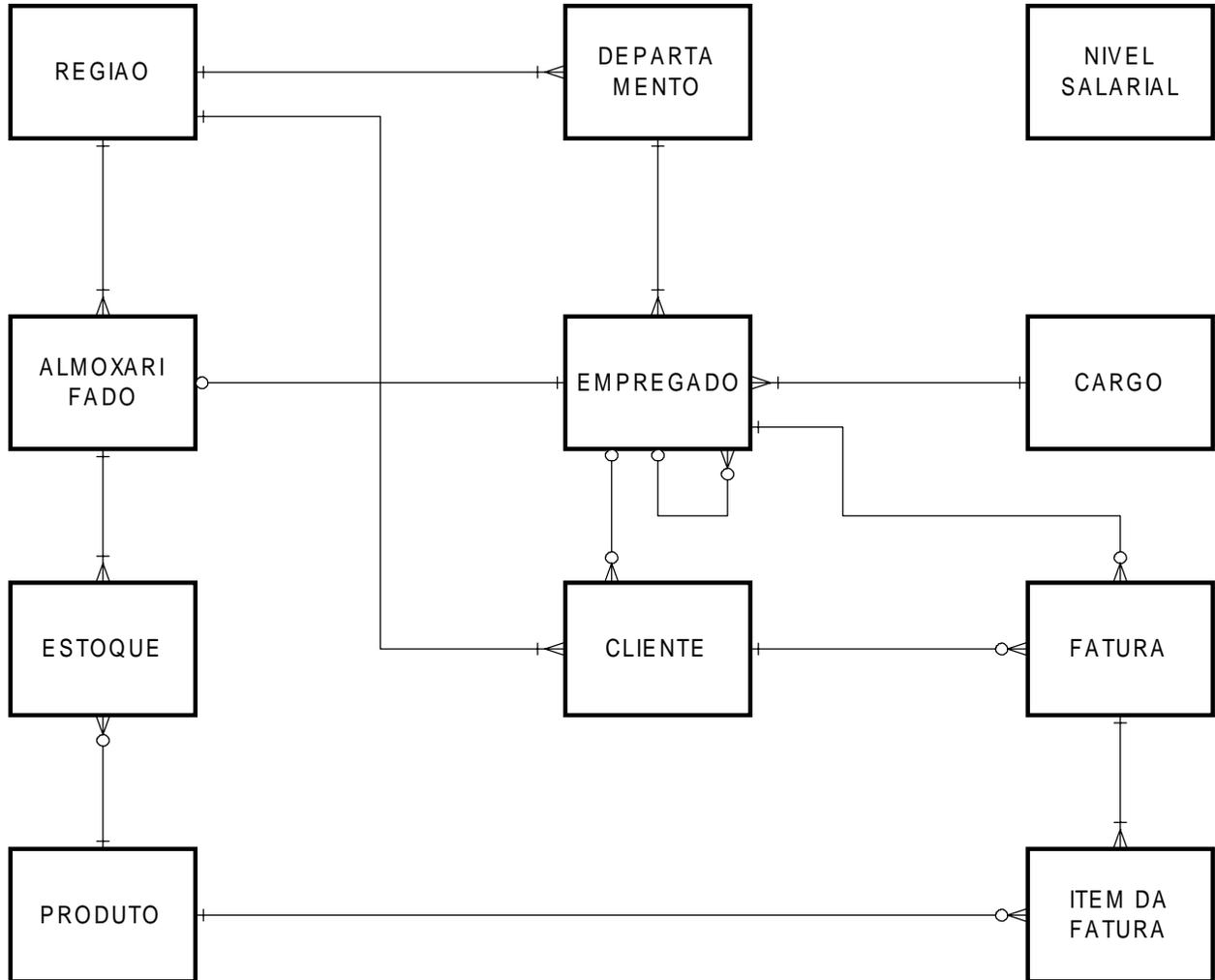


Tabela c_item_fat

Name	Null?	Type
ID_FAT (PK,FK1)	NOT NULL	NUMBER(7)
ID_ITEM (PK)	NOT NULL	NUMBER(7)
ID_PRODUTO (FK2)	NOT NULL	NUMBER(7)
PRECO		NUMBER(11,2)
QTDE		NUMBER(9)
QTDE_EMBARCADA		NUMBER(9)

Tabela c_fatura

Name	Null?	Type
ID (PK)	NOT NULL	NUMBER(7)
ID_CLIENTE (FK1)	NOT NULL	NUMBER(7)
DT_FAT	NOT NULL	DATE
DT_EMB	NOT NULL	DATE
ID_REPR_VENDAS (FK2)		NUMBER(7)
TOTAL		NUMBER(11,2)
TIPO_PAGAMENTO		VARCHAR2(8)
IND_ATEND_FAT		VARCHAR2(1)

Tabela c_produto

Name	Null?	Type
ID (PK)	NOT NULL	NUMBER(7)
NOME	NOT NULL	VARCHAR2(25)
DESC_RESUMIDA		VARCHAR2(50)
PRECO_ATACADO_SUGERIDO		NUMBER(11,2)
UNID_ESTOQUE		VARCHAR2(25)

Tabela c_regiao

Name	Null?	Type
ID (PK)	NOT NULL	NUMBER(7)
NOME	NOT NULL	VARCHAR2(20)

Tabela c_nivel_salarial

Name	Null?	Type
NIVEL (PK)	NOT NULL	NUMBER(7)
SALARIO_DE	NOT NULL	NUMBER(7)
SALARIO_ATE	NOT NULL	NUMBER(7)

Tabela c_depto

Name	Null?	Type
ID (PK)	NOT NULL	NUMBER(7)
NOME	NOT NULL	VARCHAR2(15)
ID_REGIAO (FK)	NOT NULL	NUMBER(7)

Tabela c_empr

Name	Null?	Type
ID (PK)	NOT NULL	NUMBER(7)
ULT_NOME	NOT NULL	VARCHAR2(20)
PRIM_NOME		VARCHAR2(15)
SENHA		VARCHAR2(8)
DT_ADMISSAO		DATE
COMENTARIOS		VARCHAR2(50)
ID_GERENTE (FK1)		NUMBER(7)
CARGO (FK2)		VARCHAR2(25)
ID_DEPTO (FK3)	NOT NULL	NUMBER(7)
SALARIO		NUMBER(11,2)
PERC_COMISSAO		NUMBER(4,2)

Tabela c_estoque

Name	Null?	Type
ID_PRODUTO (PK,FK1)	NOT NULL	NUMBER(7)
ID_ALMOX (PK,FK2)	NOT NULL	NUMBER(7)
QTD_ESTOQUE		NUMBER(9)
PONTO_RESSUP		NUMBER(9)
QTD_MAX_ESTOQUE		NUMBER(9)
MOTIVO_FALTA_ESTOQUE		VARCHAR2(255)
DT_RESSUP		DATE

ESTRUTURAS DAS TABELAS

Tabela c_almoxarifado

Name	Null?	Type
ID (PK)	NOT NULL	NUMBER(7)
ID_REGIAO (FK1)	NOT NULL	NUMBER(7)
ENDER		LONG
CIDADE		VARCHAR2(30)
ESTADO		VARCHAR2(20)
PAIS		VARCHAR2(30)
CEP		VARCHAR2(75)
TELEFONE		VARCHAR2(25)
ID_GERENTE (FK2)		NUMBER(7)

Tabela c_cargo

Name	Null?	Type
CARGO (PK)	NOT NULL	VARCHAR2(25)

Tabela c_cliente

Name	Null?	Type
ID (PK)	NOT NULL	NUMBER(7)
NOME	NOT NULL	VARCHAR2(30)
TELEFONE		VARCHAR2(20)
ENDER		VARCHAR2(50)
CIDADE		VARCHAR2(20)
ESTADO		VARCHAR2(20)
PAIS		VARCHAR2(20)
CEP		VARCHAR2(15)
SIT_CRED		VARCHAR2(9)
ID_REPR_VENDAS (FK1)		NUMBER(7)
ID_REGIAO (FK2)	NOT NULL	NUMBER(7)
COMENTARIOS		VARCHAR2(255)

ANEXOS

HAVING COUNT (*) > 2

- Ex07_30 - mostrar departamento, quantidade de empregados no departamento (com cabeçalho “Qtd.empregados”) e total anual de salários no departamento (com cabeçalho “Total anual salários”), para os departamentos que tenham mais que um empregado.
 - SELECT id_depto,COUNT (*) "Qtd. empregados",
SUM (salario)*12 "Total anual salários"
FROM c_empr
GROUP BY id_depto
HAVING COUNT (*) > 1
- Ex07_31 - mostrar cargo e total mensal de salários por cargo (com cabeçalho “Total mensal salários”), quando estes totais que excederem à \$5000. Não devem ser incluídos os vice-presidentes. Ordenar por total mensal de salários.
 - SELECT cargo,SUM (salario) "Total mensal salários"
FROM c_empr
WHERE cargo NOT LIKE 'VP%'
GROUP BY cargo
HAVING SUM (salario) > 5000
ORDER BY SUM (salario)

WHERE condição

GROUP BY expressão

onde

expressão especifica as colunas cujos valores determinam a base para o grupo de linhas;
group by produz uma linha sumarizada para cada grupo de linhas selecionado.

- Ex07_25 - mostrar cada situação de crédito possível e a quantidade de clientes em cada uma destas situações (com cabeçalho “Qtd. clientes”).
 - SELECT sit_cred Crédito, COUNT (*) "Qtd. clientes"
FROM c_cliente
GROUP BY sit_cred
- Ex07_26 - mostrar identificador de departamento e quantos empregados estão lotados em cada um (com cabeçalho “Qtd. empregados”).
 - SELECT id_depto, COUNT (*) "Qtd. empregados"
FROM c_empr
GROUP BY id_depto
- Ex07_27 - mostrar identificador de departamento, cargo e a quantidade de empregados lotados em cada departamento e em cada cargo (com cabeçalho “Qtd. empregados”).
 - SELECT id_depto,cargo, COUNT (*) "Qtd. empregados"
FROM c_empr
GROUP BY id_depto,cargo
- Ex07_28 - mostrar cargo, identificador de departamento e a quantidade de empregados em cada cargo e em cada departamento (com cabeçalho “Qtd. empregados”).
 - SELECT cargo, id_depto,COUNT (*) "Qtd. empregados"
FROM c_empr
GROUP BY cargo,id_depto
- Mostrar linhas específicas ou grupos específicos.
 - SELECT nome da coluna [, nome da coluna]
FROM nome da tabela
WHERE condição 1
GROUP BY expressão
HAVING condição 2

onde

condição 2 restringe os grupos de linhas retornados àqueles grupos para os quais a condição especificada é verdadeira.
- Ex07_29 - mostrar cargo, média anual de salários no cargo (com cabeçalho “Sal.médio anual”) e quantidade de empregados no cargo (com cabeçalho “Qtd.empregados”), para os cargos que tenham mais que dois empregados.
 - SELECT cargo,AVG (salario)*12 "Sal.médio anual",
COUNT (*) "Qtd. empregados"
FROM c_empr
GROUP BY cargo

- Executar cálculos sumarizados. Funções
 - **AVG (x)**
 - Retorna o valor médio da coluna **x**.
 - Exemplo: AVG (salario)
 - Ignora os valores nulos.
 - **MAX (x)**
 - Retorna o valor máximo da coluna **x**.
 - Exemplo: MAX (salario)
 - Ignora os valores nulos.
 - **MIN (x)**
 - Retorna o valor mínimo da coluna **x**.
 - Exemplo: MIN (salario)
 - Ignora os valores nulos.
 - **SUM (x)**
 - Retorna a soma da coluna **x**.
 - Exemplo: SUM (salario)
 - Ignora os valores nulos.
 - **COUNT (x)**
 - Retorna o número de valores não nulos da coluna **x**.
 - Exemplo: COUNT (perc_comissao)
 - **COUNT (*)**
 - Retorna o número de linhas de uma tabela.
 - Exemplo: COUNT (*)
 - Considera os valores nulos.
 - Ex07_21 - mostrar salário médio, maior salário, menor salário e total dos salários dos representantes de venda.
 - ```
SELECT AVG (salario) "Sal. médio", MAX (salario) "Sal. maior",
MIN (salario) "Sal. menor", SUM (salario) "Sal. total"
FROM c_empr
WHERE cargo = 'Representante de Vendas'
```
  - Ex07\_22 - mostrar último nome que é o primeiro em ordem alfabética.
    - ```
SELECT MIN (ult_nome) "Primeiro ordem alfabética"  
FROM c_empr
```
 - Ex07_23 - mostrar último nome que é o último em ordem alfabética.
 - ```
SELECT MAX (ult_nome) "Último ordem alfabética"
FROM c_empr
```
  - Ex07\_24 - mostrar o total de linhas da tabela de empregados.
    - ```
SELECT COUNT (*) "Total de linhas tabela C_EMPR"  
FROM c_empr
```
- Agrupar informações.
 - ```
SELECT nome da coluna [, nome da coluna]
FROM nome da tabela
```

```
WHERE LOWER(SUBSTR(nome,1,3)) = 'ace'
```

- Ex07\_18 - mostrar nome completo e cargo dos vice-presidentes. Ao mostrar o cargo, omitir os caracteres iniciais “VP,”. As informações devem ter o cabeçalho “Vice-presidentes”.
  - ```
SELECT prim_nome || ' ' || ult_nome || SUBSTR(cargo,3)
"Vice-presidentes"
FROM c_empr
WHERE cargo LIKE 'VP%'
```
- Ex07_19 - para os empregados do departamento 42, mostrar ult_nome e o número de caracteres existentes no último nome.
 - ```
SELECT ult_nome,LENGTH (ult_nome)
FROM c_empr
WHERE id_depto = 42
```
- Ex07\_20 - para os empregados que tenham o último nome com 6 caracteres, mostrar ult\_nome.
  - ```
SELECT ult_nome
FROM c_empr
WHERE LENGTH(ult_nome) = 6
```

- Exemplos. Supondo-se que o nome do cliente esteja gravado como “delhi sports”.
 - SUBSTRING (nome,1,4)
 - Retorna “delh”.
 - SUBSTRING (nome,2,6)
 - Retorna “elhi s”.
- **LENGTH (x)**
 - Retorna o tamanho do string **x**.
 - Exemplo. Supondo-se que o nome do cliente esteja gravado como “delhi sports”.
 - LENGTH (nome)
 - Retorna “12”.
- Ex07_12 - mostrar senha (com o cabeçalho “Senha”), último nome e cargo dos empregados que são vice-presidentes. As senhas devem aparecer com a primeira letra maiúscula e as demais minúsculas.
 - SELECT INITCAP(senha) Senha,ult_nome,cargo
FROM c_empr
WHERE cargo LIKE 'VP%'
- Ex07_13 - mostrar primeiro nome e último nome de todos os empregados com último nome “PIRES”.
 - SELECT prim_nome,ult_nome
FROM c_empr
WHERE ult_nome = 'PIRES'
 - Nenhuma selecionada porque o nome está gravado só com a primeira letra maiúscula, o que não ser do conhecimento do usuário.
- Ex07_14 - mostrar primeiro nome e último nome de todos os empregados com último nome “PIRES”.
 - SELECT prim_nome,ult_nome
FROM c_empr
WHERE UPPER(ult_nome) = 'PIRES'
 - O nome é mostrado como está gravado no banco de dados.
- Ex07_15 - mostrar primeiro nome, último nome e cargo de todos os empregados do departamento 42. Mostrar todas as informações em letras minúsculas.
 - SELECT LOWER(prim_nome),LOWER(ult_nome),LOWER(cargo)
FROM c_empr
WHERE id_depto = 50
- Ex07_16 - mostrar identificador e nome dos produtos que têm “ski” no nome.
 - SELECT id,nome
FROM c_produto
WHERE LOWER(nome) LIKE '%ski%'
- Ex07_17 - mostrar identificador e nome dos produtos que têm “ACE” no início do nome.
 - SELECT id,nome
FROM c_produto

- ```
SELECT ult_nome, salario, perc_comissao,
salario * (perc_comissao / 100) COMISSÃO
FROM c_empr
WHERE salario >= 1500
```

- Converter um valor nulo em um valor real usando a função NVL.
  - **NVL ( expressão 1 , expressão 2 )**
    - **expressão 1** é o valor fonte que pode conter nulos.
    - **expressão 2** é o valor que vai substituir o valor nulo.
  - Ex07\_10 - para os empregados com salários maiores ou iguais a \$1500, mostrar ult\_nome, salário, percentual de comissão e valor da comissão (com cabeçalho “Comissão”). Nem todos os empregados têm um percentual de comissão; neste caso, considerar o valor de comissão igual a 0.
    - ```
SELECT ult_nome, salario, NVL(perc_comissao,0),
salario * (NVL(perc_comissao,0) / 100) COMISSÃO
FROM c_empr
WHERE salario >= 1500
```
- Manipular strings de caracteres.
 - Concatenar strings de caracteres com o operador de concatenação ||.
 - Ex07_11 - mostrar os nomes completos dos empregados do departamento 41, separados por um espaço e com o cabeçalho “Nome dos Empregados”.
 - ```
SELECT prim_nome || ' ' || ult_nome "Nomes dos Empregados"
FROM c_empr
WHERE id_depto = 41
```
- Funções.
  - **INITCAP (x)**
    - Retorna o string **x** com a primeira letra de cada palavra em maiúscula e as restantes em minúsculas.
    - Exemplo. Supondo-se que o nome do cliente esteja gravado como “delhi sports”.
      - **INICAP (nome)**
        - Retorna “Delhi Sports”.
  - **UPPER (x)**
    - Retorna o string **x** com todas as letras em maiúsculas.
    - Exemplo. Supondo-se que o nome do cliente esteja gravado como “delhi sports”.
      - **UPPER (nome)**
        - Retorna “DELHI SPORTS”.
  - **LOWER (x)**
    - Retorna o string **x** com todas as letras em minúsculas.
    - Exemplo. Supondo-se que o nome do cliente esteja gravado como “delhi sports”.
      - **LOWER (nome)**
        - Retorna “delhi sports”.
  - **SUBSTRING (x , y , z )**
    - Retorna o string **x** , iniciando com o caracter localizado na posição **y** , com o tamanho de **z** ; se **z** for omitido retorna até o fim do string **x** .

- Ex07\_05 - mostrar nome do produto, preço, imposto e imposto arredondado de todos os itens da fatura com preço unitário entre \$20 e \$30 (inclusive). O imposto é de 30% do preço do item. O imposto é arredondado para o centavo mais próximo. O imposto deve ter uma coluna com o cabeçalho “Imposto”; o imposto arredondado deve ter uma coluna com o cabeçalho “Imposto Arredondado”. As linhas não devem ser repetidas.
  - ```
SELECT DISTINCT p.nome PRODUTO, i.preco,
i.preco * .03 IMPOSTO,
ROUND ((i.preco * .03),2) "IMPOSTO ARREDONDADO"
FROM c_item_fat i, c_produto p
WHERE i.preco BETWEEN 20 AND 30
AND i.id_produto = p.id;
```
- Ex07_06 - mostrar ult_nome, salário diário e salário diário truncado de todos os empregados do departamento 42. O salário mensal considera o mês com 22 dias. O salário diário truncado é truncado para o inteiro. O salário diário deve ter uma coluna com o cabeçalho “Salário diário”; o salário diário truncado deve ter uma coluna com o cabeçalho “Sal. diário truncado”.
 - ```
SELECT ult_nome,salario,(salario / 22) "SALÁRIO DIÁRIO",
TRUNC (salario / 22,0) "SAL. DIÁRIO TRUNCADO"
FROM c_empr
WHERE id_depto = 42
```
- Ex07\_07 - para todos os itens com preço no atacado sugerido entre \$12 e \$34 (inclusive) mostrar preço no atacado sugerido, preço no atacado sugerido com desconto e preço no atacado sugerido com desconto truncado. O desconto é de 37% do preço no atacado sugerido. O preço no atacado sugerido com desconto truncado é truncado para o centavo mais próximo. O preço no atacado sugerido deve ter uma coluna com o cabeçalho “Preço”; o preço no atacado sugerido com desconto deve ter uma coluna com o cabeçalho “Preço com desconto”; o preço no atacado sugerido com desconto truncado deve ter uma coluna com o cabeçalho “Preço truncado”.
  - ```
SELECT id,preco_atacado_sugerido PREÇO,
preco_atacado_sugerido * .63 "PREÇO COM DESCONTO",
TRUNC ((preco_atacado_sugerido * .63),2) "PREÇO TRUNCADO"
FROM c_produto
WHERE preco_atacado_sugerido BETWEEN 12 AND 34
```
- Ex07_08 - para os empregados admitidos a partir de 01/01/92, mostrar ult_nome, data de admissão, total de dias (com o cabeçalho “Total dias”) e total de semanas (com o cabeçalho “Total semanas”) e o complemento em dias (com o cabeçalho “Compl Dias”) na empresa até a data de hoje.O total de dias e o total de semanas devem ser truncados e o complemento em dias deve ser arredondado, para o inteiro mais próximo.
 - ```
SELECT ult_nome,dt_admissao,
TRUNC((SYSDATE - dt_admissao),0) "TOTAL DIAS",
TRUNC(((SYSDATE - dt_admissao)/7),0) "TOTAL SEMANAS",
ROUND(MOD((SYSDATE - dt_admissao),7),0) "COMPL DIAS"
FROM c_empr
WHERE dt_admissao > '01-jan-92'
```
- Ex07\_09 - para os empregados com salários maiores ou iguais a \$1500, mostrar ult\_nome, salário, percentual de comissão e valor da comissão (com cabeçalho “Comissão”). Nem todos os empregados têm um percentual de comissão.

- Funções numéricas.
  - **ROUND ( n , m )**
    - Retorna **n** arredondado de **m** casas decimais à direita da vírgula decimal. O valor default de **m** é 0; se **m** for negativo, **n** será arredondado de **m** casas inteiras à esquerda da vírgula decimal.
    - Exemplos
      - round (salario, 1)
        - o salário de \$2550,25 passa para \$2550,30
        - o salário de \$1450,52 passa para \$1450,50
      - round (salario, 0)
        - o salário de \$2550,25 passa para \$2550
        - o salário de \$1450,52 passa para \$1451
      - round (salario, -2)
        - o salário de \$2550 passa para \$2600
        - o salário de \$1515 passa para \$1500
    - **TRUNC ( n , m )**
      - Retorna **n** truncado **m** casas decimais à direita da vírgula decimal. O valor default de **m** é 0; se **m** for negativo, **n** será truncado à esquerda da vírgula decimal.
      - Exemplos
        - trunc (salario, 1)
          - o salário de \$2550,25 passa para \$2550,20
          - o salário de \$1450,52 passa para \$1450,50
        - trunc (salario, 0)
          - o salário de \$2550,25 passa para \$2550
          - o salário de \$1450,52 passa para \$1450
        - trunc (salario, -2)
          - o salário de \$2550 passa para \$2500
          - o salário de \$1515 passa para \$1500
      - **MOD ( n , m )**
        - Retorna o resto da divisão de **n** por **m**.
        - Exemplo
          - mod ( 7 , 5 )
            - retorna 2, que é o resto da divisão de 7 por 5.
    - Ex07\_04 - mostrar ult\_nome, salário diário e salário diário de todos os empregados do departamento 42. O salário mensal considera o mês com 22 dias. O salário diário arredondado é arredondado para o inteiro mais próximo. O salário diário deve ter uma coluna com o cabeçalho “Salário diário”; o salário diário arredondado deve ter uma coluna com o cabeçalho “Sal. diário arred”.
      - ```
SELECT ult_nome,salario, (salario / 22) "SALÁRIO DIÁRIO",  
ROUND(salario / 22,0) "SAL. DIÁRIO ARRED"  
FROM c_empr  
WHERE id_depto = 42
```
 - Observar que o salário diário pode ter sido arredondado para cima como para baixo.

7 - EXECUTAR OPERAÇÕES COM DADOS

- Objetivos
 - Executar cálculos com números usando operadores aritméticos e funções.
 - Executar cálculos com datas usando operadores aritméticos e funções.
 - Reformatar números e datas.
 - Manipular strings de caracteres e usá-los com funções.
 - Executar cálculos sumarizados com grupos de linhas usando funções de grupo.
- Executar cálculos com números usando operadores aritméticos e funções.
 - Operadores aritméticos
 - + somar
 - - subtrair
 - * multiplicar
 - / dividir
 - Os operadores aritméticos podem ser usados em qualquer cláusula de um comando SQL, exceto na cláusula FROM.
 - Ex07_01 - mostrar identificador, ult_nome, salário, percentual de comissão e valor da comissão de todos os representantes de vendas, em ordem crescente de valor da comissão. O valor da comissão deve ter uma coluna com o cabeçalho “Comissão”.
 - ```
SELECT id,ult_nome,salario,perc_comissao,
 salario * (perc_comissao/100) COMISSÃO
FROM c_empr
WHERE cargo = 'Representante de Vendas'
ORDER BY salario * (perc_comissao/100)
```
  - Ex07\_02 - mostrar identificador, ult\_nome, salário, percentual de comissão e valor da comissão de todos os representantes de vendas cujo valor da comissão é maior que \$150, em ordem crescente de valor da comissão. O valor da comissão deve ter uma coluna com o cabeçalho “Comissão”.
    - ```
SELECT id, ult_nome, salario, perc_comissao,
      salario * (perc_comissao/100) COMISSÃO
FROM c_empr
WHERE cargo = 'Representante de Vendas'
      AND salario * perc_comissao/100 > 150
ORDER BY salario * perc_comissao/100
```
 - Ex07_03 - mostrar identificador, ult_nome, salário e gratificação anual de todos os empregados do departamento 41, em ordem decrescente do valor da gratificação anual. A gratificação anual corresponde a 12 salários e uma bonificação fixa de \$150. A gratificação anual deve ter uma coluna com o cabeçalho “Gratificação anual”.
 - ```
SELECT id,ult_nome,salario,
 (salario * 12) + 150 "GRATIFICAÇÃO ANUAL"
FROM c_empr
WHERE id_depto = 41
ORDER BY (salario * 12) + 150
```

- Ex06\_09 - mostrar identificador, nome e situação de crédito de todos os clientes localizados na região North America ou que tenham o empregado com último nome Nogueira como representante de vendas.

- ```
SELECT id,nome, sit_cred
FROM c_cliente
WHERE id_regiao =
      (SELECT id
       FROM c_regiao
       WHERE nome = 'North America')
OR id_repr_vendas =
      (SELECT id
       FROM c_empr
       WHERE ult_nome = 'Nogueira')
```

ou

- ```
SELECT distinct c_cliente.id,c_cliente.nome,c_cliente.sit_cred
FROM c_regiao,c_cliente,c_empr
WHERE (c_regiao.nome = 'North America'
AND c_regiao.id = c_cliente.id_regiao)
OR (c_empr.ult_nome = 'Nogueira'
AND c_empr.id = c_cliente.id_repr_vendas);
```

- Ex06\_10 - mostrar identificador, último nome, salário e identificador do departamento de todos os empregados lotados no departamento de Vendas ou que estejam lotados em departamentos localizados na região 2.

- ```
SELECT id,ult_nome,salario,id_depto
FROM c_empr
WHERE id_depto IN
      (SELECT id
       FROM c_depto
       WHERE nome = 'Vendas')
OR id_depto IN
      (SELECT id
       FROM c_depto
       WHERE id_regiao = 2)
```

ou

- ```
SELECT c_empr.id,c_empr.ult_nome,c_empr.salario,c_empr.id_depto
FROM c_depto,c_empr
WHERE (c_depto.nome = 'Vendas'
AND c_depto.id = c_empr.id_depto)
OR (c_depto.id_regiao = 2
AND c_depto.id = c_empr.id_depto);
```

- Ex06\_06 - mostrar identificador, último nome, salário e identificador do departamento de todos os empregados que estão lotados em departamentos que tenham algum empregado com Pires como último nome.
  - ```
SELECT id,ult_nome,salario,id_depto
FROM c_empr
WHERE id_depto IN
      (SELECT id_depto
       FROM c_empr
       WHERE ult_nome = 'Pires')
```
- Ex06_07 - mostrar identificador, último nome, salário e identificador do departamento de todos os empregados que estão lotados em departamentos que tenham algum empregado com Machado como último nome.
 - ```
SELECT id,ult_nome,prim_nome,salario,id_depto
FROM c_empr
WHERE id_depto IN
 (SELECT id_depto
 FROM c_empr
 WHERE ult_nome = 'Machado');
```
- Ex06\_08 - mostrar identificador, último nome, primeiro nome, salário e data de admissão de todos os empregados cujos salários estão abaixo da média e que estão lotados em departamentos que tenham algum empregado com Pires como último nome.
  - ```
SELECT id,ult_nome,prim_nome,salario,dt_admissao
FROM c_empr
WHERE salario <
      (SELECT AVG (salario)
       FROM c_empr)
AND id_depto IN
      (SELECT id_depto
       FROM c_empr
       WHERE ult_nome = 'Pires')
```

```
(SELECT id_depto
FROM c_empr
WHERE ult_nome = 'Hubert')
```

- EX06_03 - mostrar número das faturas do cliente Muench Sports.

```
• SELECT id
FROM c_fatura
WHERE id_cliente =
  (SELECT id
   FROM c_cliente
   WHERE nome = 'Muench Sports')
```

OU

```
• SELECT c_fatura.id
FROM c_cliente,c_fatura
WHERE c_cliente.nome = 'Muench Sports'
AND c_cliente.id = c_fatura.id_cliente;
```

- Mostrar dados através de uma query principal que tem uma subquery com uma função de grupo.

- Ex06_04 - mostrar último nome, cargo e salário de todos os empregados que ganham menos que o salário médio.

```
• SELECT ult_nome,cargo,salario
FROM c_empr
WHERE salario <
  (SELECT AVG (salario)
   FROM c_empr)
```

- Subqueries que retornam várias linhas. As linhas mostradas pela query principal dependem de várias linhas obtidas pela subquery.

- Ex06_05 - mostrar identificador, último nome, salário e identificador do departamento de todos os empregados que estão lotados em departamentos que pertençam à região 2.

```
• SELECT id,ult_nome,salario,id_depto
FROM c_empr
WHERE id_depto IN
  (SELECT id
   FROM c_depto
   WHERE id_regiao = 2)
```

```
SELECT c_empr.id,c_empr.ult_nome,c_empr.salario,c_empr.id_depto
FROM c_empr, c_depto
WHERE c_depto.id_regiao = 2
AND c_depto.id = c_empr.id_depto;
```

6 - UTILIZAR SUBQUERIES

- Objetivos
 - Passar uma linha de dados recuperada por uma subquery para a query principal.
 - Passar várias linhas de dados recuperadas por uma subquery para a query principal.
 - Encadear múltiplas subqueries dentro da query principal.
- Uma subquery é um comando *select* que está embutido em uma cláusula de um outro comando SQL.
- Mostrar dados usando uma query que depende dos resultados de outra query.
 - **SELECT { nome da coluna [, nome da coluna ...] }**
FROM nome da tabela
WHERE nome da coluna operador
 (**SELECT { nome da coluna [, nome da coluna ...] }**
 FROM nome da tabela
 WHERE condição)
 - onde
 - operador
 - pode ser >, =, <, IN
 - condição
 - pode ser =, <>, >, >=, <, <=
 - Processamento de uma subquery
 - O comando select interno é executado primeiro.
 - O resultado é passado para a query principal
 - Pode-se colocar uma subquery
 - Em uma cláusula *where*
 - Em uma cláusula *having*
- Subqueries que retornam uma só linha. As linhas mostradas pela query principal dependem de uma só linha obtida pela subquery.
 - EX06_01 - mostrar identificador, último nome e cargo dos empregados que têm o mesmo cargo que o empregado que tem como último nome Machado.
 - **SELECT id,ult_nome,cargo**
FROM c_empr
WHERE cargo =
 (**SELECT cargo**
 FROM c_empr
 WHERE ult_nome = 'Machado')
 - EX06_02 - mostrar identificador, último nome e código do departamento dos empregados que pertencem ao mesmo departamento que o empregado com último nome Hubert.
 - **SELECT id,ult_nome,cargo,id_depto**
FROM c_empr
WHERE id_depto =

- Mostrar dados de diferentes linhas na mesma tabela (dados de um auto-relacionamento). Self-join
 - Ex05_13 - mostrar a hierarquia da SuperSports apresentando o último nome de cada gerente e de seu subordinado direto, colocando os cabeçalhos Gerente e Subordinado, respectivamente.
 - ```
SELECT gerente.ult_nome GERENTE,subordinado.ult_nome SUBORDINADO
FROM c_empr gerente,
 c_empr subordinado
WHERE subordinado.id_gerente = gerente.id;
```
  - Ex05\_14 - mostrar o último nome de cada empregado e de seu gerente, ordenados por último nome do empregado, colocando os cabeçalhos Empregado e Empregado-Gerente.
    - ```
SELECT subordinado.ult_nome EMPREGADO,
       gerente.ult_nome "EMPREGADO-GERENTE"
FROM   c_empr gerente, c_empr subordinado
WHERE  subordinado.id_gerente = gerente.id
ORDER BY subordinado.ult_nome
```
 - Ex05_15 - mostrar o último nome de todos empregados e de seus gerentes, ordenados por último nome do empregado, colocando os cabeçalhos Empregado e Empregado-Gerente. Um empregado pode não ter gerente.
 - ```
SELECT subordinado.ult_nome EMPREGADO,
 gerente.ult_nome "EMPREGADO-GERENTE"
FROM c_empr gerente, c_empr subordinado
WHERE subordinado.id_gerente = gerente.id (+)
ORDER BY subordinado.ult_nome
```
- Criar uma visão de com múltiplas tabelas.
  - Ex05\_16 - criar a visão *v\_empr\_depto* contendo último nome e nome do departamento de todos os empregados.
    - ```
CREATE VIEW v_empr_depto
AS
SELECT c_empr.ult_nome,c_depto.nome Departamento
FROM   c_empr,c_depto
WHERE  c_empr.id_depto = c_depto.id
```
 - Ex05_17 - mostrar último nome, código e nome do departamento de todos os empregados, usando a visão *v_empr_depto*.
 - ```
SELECT * FROM v_empr_depto
```

- Ex05\_08- mostrar nome do cliente, código do produto e quantidade faturada dos clientes que cujas faturas totalizaram mais que 100000.

```

• SELECT cl.nome,i.id_produto,i.qtde
FROM c_cliente cl,c_fatura f,c_item_fat i
WHERE f.total > 100000
AND f.id_cliente = cl.id
AND f.id = i.id_fat

```

- Non-equijoin

- Ex05\_09 - mostrar último nome, função, salário e nível salarial de cada empregado.

```

• SELECT e.ult_nome, e.cargo, e.salario, s.nivel
FROM c_empr e, c_nivel_salarial s
WHERE e.salario BETWEEN s.salario_de AND s.salario_ate

```

- Outer-join.

- Mostrar dados de tabelas relacionadas onde algumas linhas de uma das tabelas não têm correspondência direta com linhas da outra tabela.

```

• SELECT nome da tabela1.nome da coluna, nome da tabela2 .nome da coluna
FROM nome da tabela1, nome da tabela2
WHERE nome da tabela1.nome da coluna (+) = nome da tabela2.nome da coluna

```

onde

( + ) é o símbolo do outer join, que pode ser colocado em quaisquer dos lados da cláusula *where* mas não em ambos os lados. Este símbolo deve ser colocado seguindo o nome da coluna que pode não ter correspondente.

- Ex05\_10 - mostrar os nomes de todos os clientes e a identificação e o último nome do representante de vendas que atende cada um, ordenados pelo nome do cliente. Existem clientes que não têm um representante de vendas os atendendo.

- Dica: para solucionar problemas desse tipo imaginar a existência de uma linha fictícia totalmente nula na tabela Empregado, para que ela seja associada às linhas da tabela Cliente que não tenham representantes de vendas.

```

• SELECT c_cliente.nome, c_empr.id,c_empr.ult_nome
FROM c_cliente,c_empr
WHERE c_cliente.id_repr_vendas = c_empr.id (+)
ORDER BY c_cliente.nome;

```

- Ex05\_11 - mostrar os nomes de todos os clientes e os números de suas faturas. Existem clientes que não têm faturas.

```

• SELECT c_cliente.nome, c_fatura.id
FROM c_fatura,c_cliente
WHERE c_fatura.id_cliente (+) = c_cliente.id
ORDER BY c_cliente.nome

```

- Ex05\_12 - mostrar os nomes e as situações de crédito dos clientes que não têm faturas.

```

• SELECT c_cliente.id,c_cliente.nome, c_cliente.sit_cred
FROM c_cliente, c_fatura
WHERE c_cliente.id = c_fatura.id_cliente (+)
AND c_fatura.id IS NULL;

```

- Ex05\_02 - mostrar código do departamento, código da região e nome da região de todos os departamentos. Chamar as colunas de Departamento, Região e Nome da região.
  - ```
SELECT c_depto.id "Departamento",c_regiao.id "Região",
c_regiao.nome "Nome da região"
FROM c_depto,c_regiao
WHERE c_depto.id_regiao = c_regiao.id
```
- Mostrar linhas específicas de tabelas relacionadas, especificando condições de pesquisa e condições de união entre as tabelas na cláusula *where*.
 - Ex05_03 - mostrar identificador, último nome, código e nome do departamento do empregado com último nome igual a Pires.
 - ```
SELECT c_empr.id, c_empr.ult_nome, c_empr.id_depto, c_depto.nome
FROM c_empr,c_depto
WHERE c_empr.id_depto = c_depto.id
AND c_empr.ult_nome = 'Pires'
```
  - Ex05\_04 - mostrar identificador e nome de todos os departamentos situados na região de nome North America.
    - ```
SELECT c_depto.id,c_depto.nome
FROM c_regiao,c_depto
WHERE c_regiao.nome = 'North America'
AND c_regiao.id = c_depto.id_regiao
```
- Usar apelidos para as tabelas referenciadas em um *select*.
 - As colunas são qualificadas com os apelidos dados às tabelas referenciadas no *select*.
 - Os apelidos são válidos apenas naquele *select*.
 - Ex05_05 - mostrar nome do cliente, código da região e nome da região de todos os clientes das regiões 4 e 5. Atribuir apelidos às tabelas referenciadas.
 - ```
SELECT cl.nome, cl.id_regiao, r.nome
FROM c_cliente cl, c_regiao r
WHERE cl.id_regiao IN (4,5)
AND cl.id_regiao = r.id
```
  - Ex05\_06 - mostrar o último nome, nome do departamento e nome da região de todos os empregados que recebem comissão,
    - ```
SELECT e.ult_nome,d.nome DEPARTAMENTO, r.nome REGIÃO
FROM c_empr e,c_depto d, c_regiao r
WHERE e.perc_comissao IS NOT NULL
AND e.id_depto = d.id AND d.id_regiao = r.id
```
 - Ex05_07 - mostrar nome do produto, código do produto e quantidade faturada dos itens da fatura de número 101.
 - ```
SELECT p.nome,p.id,i.qtde
FROM c_item_fat i, c_produto p
WHERE i.id_fat = 101
AND i.id_produto = p.id
```

## 5 - MOSTRAR DADOS DE MÚLTIPLAS TABELAS

- Objetivos
  - Recuperar dados a partir de várias tabelas
  - Combinar dados de tabelas relacionadas onde algumas linhas de uma das tabelas não têm correspondência direta com linhas da outra tabela.
  - Fazer uma junção de uma tabela com ela própria
  - Criar uma visão de tabelas múltiplas.
- Quando são necessários dados de várias tabelas são feitas junções destas tabelas. As linhas de uma tabela são juntadas com as linhas de outra tabela através de valores comuns existentes em colunas correspondentes, tipicamente chaves primárias e chaves estrangeiras.
- Tipos principais de junções:
  - Equijoin
    - Quando uma coluna de uma tabela corresponde diretamente a uma coluna em uma outra tabela.
  - Non-equijoin
    - Quando nenhuma coluna de uma tabela corresponde diretamente a uma coluna em uma outra tabela.
  - Outer join
    - Quando algumas linhas de uma das tabelas não têm correspondência direta com linhas da outra tabela.
  - Self join
- Equijoin
- Mostrar dados de tabelas relacionadas, fazendo *join* (união) de tabelas através do comando *select* com a cláusula *where*.

• **SELECT** *nome da tabela . nome da coluna, nome da tabela . nome da coluna ....*  
**FROM** *nome da tabela 1, nome da tabela 2*  
**WHERE** *nome da tabela 1. nome da coluna = nome da tabela 2. nome da coluna*

onde

- *nome da tabela 1. nome da coluna = nome da tabela 2. nome da coluna* é a condição que junta as tabelas.
- Ao escrever um comando *select* que junta tabelas, o nome da coluna deve ser prefixado pelo nome da tabela, por questões de clareza e desempenho da query.
- Ex05\_01 - mostrar identificador, último nome, código do departamento e nome do departamento de todos os empregados.
  - **SELECT** *c\_empr.id, c\_empr.ult\_nome, c\_empr.id\_depto, c\_depto.nome*  
**FROM** *c\_empr, c\_depto*  
**WHERE** *c\_empr.id\_depto = c\_depto.id*

- Criação de índice
  - **CREATE INDEX** *nome do índice*  
**ON** *nome da tabela* ( *nome da coluna* [ , *nome da coluna* ] .... )
  - O Oracle cria automaticamente índices para as colunas que têm restrições PRIMARY KEY ou UNIQUE.
- Ex04\_29 - criar um índice para a coluna *ult\_nome* da tabela *c\_empr* .
  - **CREATE INDEX** *c\_empr\_ult\_nome\_x*  
**ON** *c\_empr* (*ult\_nome*)
- Queries ao dicionário de dados do Oracle
  - A tabela USER\_INDEXES armazena os índices criados pelos usuários. As colunas dessa tabela são, entre outras:
    - INDEX\_NAME
      - Mostra o nome do índice.
    - TABLE\_OWNER
      - Mostra o nome do dono do objeto indexado.
    - TABLE\_NAME
      - Mostra o nome do objeto indexado.
    - TABLE\_TYPE
      - Mostra o tipo do objeto indexado.
    - UNIQUENESS
      - Indica se os valores indexados são únicos ou não..
- Ex04\_30- mostrar os índices da tabela *c\_empr* e se os seus valores são únicos.
  - **SELECT** *index\_name, uniqueness*  
**FROM** *user\_indexes*  
**WHERE** *table\_name = 'C\_EMPR'*
- Excluir um índice.
  - **DROP INDEX** *nome do índice*
- Ex04\_31 - excluir o índice da coluna *ult\_nome* da tabela *c\_empr* .
  - **DROP INDEX** *c\_empr\_ult\_nome\_x*

- Ex04\_27 - mostrar identificador e nome das regiões existentes na tabela *x\_regiao*.
  - ```
SELECT id,nome
FROM x_regiao;
```
- Remover uma seqüência.
 - **DROP SEQUENCE** *nome da seqüência*
- Ex04_28 - excluir a seqüência da tabela *x_regiao*.
 - ```
DROP SEQUENCE x_regiao_id_sq
```
- Aumentar a performance de queries
  - Criação de um ou mais índices nas tabelas.
    - Os índices são objetos criados explicitamente.
    - Os índices são armazenados independentemente das tabelas que eles indexam.
    - Os índices são usados automaticamente e mantidos pelo Oracle.
    - Os índices apontam diretamente para as linhas das tabelas.
    - O uso dos índices pode reduzir as operações de entrada e saída nos discos magnéticos.
  - O Oracle escolhe o método de acesso aos dados baseado na existência de índices e da estrutura do comando *SELECT*.
    - Usando ROWID
      - Acesso direto usando o endereço que indica o local exato do dado.
      - É o método de acesso mais rápido.
    - Browse
      - Pesquisa seqüencial através de todas as linhas da tabela.
    - Usando índice
      - Pesquisa usando uma estrutura de árvore *B\** com os valores das colunas.
  - Vantagens e desvantagens da criação de índices.
    - Criar um índice em uma coluna se:
      - a coluna for usada freqüentemente em uma cláusula *where* ou em uma condição de junção entre tabelas.
      - cada valor dentro de uma coluna é único.
      - a coluna contém uma variedade grande de valores.
      - a coluna é esparsamente populada, isto é, ela tem uma grande quantidade de valores nulos.
      - a tabela é grande e a maioria das queries espera recuperar menos que 10-15% das linhas. Neste contexto uma tabela é grande se tiver algumas centenas de milhares de linhas.
    - Não criar índices se:
      - a tabela for pequena, isto é, se toda a tabela puder ser armazenada em alguns blocos.
      - a maioria das queries espera recuperar mais que 10-15% das linhas.
      - a tabela é atualizada freqüentemente.
    - O Oracle automaticamente atualiza os índices. Não há impacto na sintaxe do **SQL**; entretanto, pode haver impacto na performance se o dado é freqüentemente atualizado.

- **START WITH**
    - especifica o primeiro número da seqüência. O default é 1.
  - **MAXVALUE**
    - especifica o valor máximo que a seqüência pode gerar.
- 
- Ex04\_24- criar a seqüência *x\_regiao\_id\_sq* para a chave primária da tabela *x\_regiao*, começando com 12, com incremento de 3 e valor máximo de 9.999.
    - ```
CREATE SEQUENCE x_regiao_id_sq  
INCREMENT BY 3  
START WITH 12  
MAXVALUE 9999
```
 - Queries ao dicionário de dados do Oracle
 - A visão **USER_SEQUENCES** recupera as seqüências criadas pelos usuários. As colunas dessa tabela são, entre outras:
 - **SEQUENCE_NAME**
 - **MIN_VALUE**
 - **MAX_VALUE**
 - **INCREMENT_BY**
 - **LAST_NUMBER**
 - Mostra o último número usado naquela seqüência.
 - Ex04_25- mostrar valor máximo, valor mínimo, incremento e último número usado da seqüência *x_regiao_id_sq*.
 - ```
SELECT sequence_name,min_value,max_value,increment_by,last_number
FROM user_sequences
WHERE sequence_name = 'X_REGIAO_ID_SQ'
```
  - Uso de seqüências para criação automática de valores de chaves primárias.
    - **nome da seqüência . NEXTVAL**
      - Retorna automaticamente o próximo valor disponível da seqüência; retorna um valor único toda vez que a seqüência é referenciada, mesmo que esta referência seja feita por usuários diferentes.
  - Ex04\_26 - criar duas novas regiões na tabela *x\_regiao*: Brasil e Austrália. A designação dos códigos destas duas regiões deve ser automática, usando a seqüência *x\_regiao\_id\_sq*. O script destas criações deve ser salvo no arquivo texto *ex04\_26.sql*.
    - ```
INSERT INTO x_regiao  
VALUES (x_regiao_id_sq.nextval,'BRASIL',NULL);  
INSERT INTO x_regiao  
VALUES (x_regiao_id_sq.nextval,'AUSTRÁLIA',NULL);
```
 - Ex04_26a – executar o script salvo no arquivo *ex04_26.sql*.
 - @ a: *ex04_26.sql*

WHERE id = 16

- 0 rows updated.
- Nenhuma linha foi atualizada porque os dados do empregado não são sequer recuperados, uma vez que ele está lotado no departamento 41 e essa visão só encontra os empregados do departamento 42.
- Ex04_22 - alterar a senha do empregado de identificação 18, passando-a para *pimenta*, usando a visão *v_empr_depto42*.
 - UPDATE v_empr_depto42
SET senha = 'pimenta'
WHERE id = 18
- Queries ao dicionário de dados do Oracle
 - A visão USER_VIEWS armazena as visões dos usuários. As colunas dessa tabela são:
 - VIEW_NAME
 - Nome da visão
 - TEXT_LENGTH
 - Tamanho do texto
 - TEXT
 - Texto da visão
 - Ex04_23 - mostrar as visões definidas.
 - SELECT view_name
FROM user_views
- Exclusão de uma visão
 - **DROP VIEW *nome da visão***
- Criação de seqüências.

Seqüência é um objeto Oracle usado para facilitar o processo de criação de identificadores únicos de uma linha de uma tabela. Uma seqüência nada mais é que um contador automático incrementado toda vez que é acessado. O comando SQL usado para a criação de uma seqüência é

- **CREATE SEQUENCE *nome da seqüência***
[INCREMENTED BY *n*]
[START WITH *n*]
[{ MAXVALUE *n* | NOMAXVALUE }]

onde

- *nome da seqüência*
 - identifica a seqüência.
- INCREMENTED BY
 - especifica o incremento da seqüência. O default é 1.

FROM v_empr_dep41

- Criação de visão com uma subquery embutida e com opção de check.
 - **CREATE VIEW** *nome da visão* (*apelido 1, apelido 2, ..., apelido n*)
AS subquery
WITH CHECK OPTION

onde

 - **WITH CHECK OPTION**
 - especifica que inclusões e atualizações executadas através de uma visão não podem gerar linhas que não são mostradas através de query com esta mesma visão.
- Ex04_18 - criar a visão *v_empr_depto42* contendo todos os dados dos empregados do departamento 42. Esta visão deve ter **WITH CHECK OPTION**.
 - **CREATE VIEW** *v_empr_depto42*
AS SELECT *
FROM c_empr
WHERE id_depto = 42
WITH CHECK OPTION
- Ex04_19 - mostrar os empregados do departamento 42.
 - **SELECT** *id, ult_nome, id_depto*
FROM v_empr_depto42
- Resultado do comando

ID	ULT_NOME	ID_DEPTO
7	Machado	42
18	Nozaki	42
19	Pires	42

- Ex04_20 - passar o empregado de identificação 18 ,que está lotado no departamento 42 conforme mostrado no exercício anterior, para o departamento 41, usando a visão *v_empr_depto42*.
 - **UPDATE** *v_empr_depto42*
SET id_depto = 41
WHERE id = 18
 - **ERROR** at line 1:ORA-01402: view WITH CHECK OPTION where-clause violation
 - O erro foi causado porque se o departamento mudar para 41, a visão *v_empr_depto42* não será mais capaz de mostrar os dados desse empregado.
- Ex04_21 - passar o empregado de identificação 16, que está lotado no departamento 41, para o departamento 42, usando a visão *v_empr_depto42*.
 - **UPDATE** *v_empr_depto42*
SET id_depto = 42

[(*apelido* [, *apelido*],)]

AS *subquery*

onde

- *apelido*
 - especifica os nomes para expressões selecionadas pela visão.
 - a quantidade de apelido deve ser igual à quantidade de expressões selecionadas pela visão.
 - *subquery*
 - é um comando SELECT completo.
-
- Ex04_12 - criar a visão *v_empr_depto10* contendo identificação, último nome e cargo dos empregados do departamento 10.
 - CREATE VIEW *v_empr_depto10*
AS SELECT id,ult_nome,cargo
FROM *c_empr*
WHERE id_depto = 10
 - Ex04_13 - mostrar identificação, último nome e cargo dos empregados do departamento 10, usando a visão *v_empr_depto10*.
 - SELECT *
FROM *v_empr_depto10*
 - Ex04_14 - criar a visão *v_empr_depto41* contendo identificação, último nome e cargo dos empregados do departamento 41. A coluna da identificação deve ter o cabeçalho “Matrícula” a do último nome o cabeçalho “Sobrenome” e a de cargo o cabeçalho “Função”. Esses cabeçalhos deverão ser “apelidos” definidos na visão.
 - CREATE VIEW *v_empr_depto41* (*matricula, sobrenome, função*)
AS SELECT id, ult_nome, cargo
FROM *c_empr*
WHERE id_depto = 41
 - Ex04_15- mostrar identificação, último nome e cargo dos empregados do departamento 41, usando a visão *v_empr_depto41*.
 - SELECT *
FROM *v_empr_depto41*
 - Ex04_16 - criar a visão *v_empr_dep41* contendo identificação, primeiro nome e salário dos empregados do departamento 41. A coluna da identificação deve ter o cabeçalho “Matrícula”, a do primeiro nome o cabeçalho “Prenome” e a de salário o cabeçalho “Salário mensal”. Esses cabeçalhos deverão ser “apelidos” definidos na query.
 - CREATE VIEW *v_empr_dep41*
AS SELECT id Matrícula, prim_nome Prenome, salario "Salário mensal"
FROM *c_empr*
WHERE id_depto = 41
 - Ex04_17 - mostrar identificação, primeiro nome e salário dos empregados do departamento 41, usando a visão *v_empr_dep41*
 - SELECT *

- TABLE_NAME
 - COLUMN_NAME
 - Mostra o nome da coluna especificada na definição da restrição.
 - POSITION
 - Posição original da coluna na restrição.
- Ex04_11- mostrar os nomes das restrições definidas para a tabela X_EMPR e as colunas nelas envolvidas.
- `SELECT constraint_name,column_name,position`
`FROM user_cons_columns`
`WHERE table_name = 'X_EMPR'`

- Resultado do comando

CONSTRAINT_NAME	COLUMN_NAME	POSITION
SYS_C00946	SALARIO	
X_EMPR_ID_PK	ID	1
X_EMPR_PERC_COMISSAO_CK	PERC_COMISSAO	
X_EMPR_SENHA_NN	SENHA	
X_EMPR_SENHA_UK	SENHA	1
X_EMPR_ULT_NOME_NN	ULT_NOME	

- Remoção de tabelas

- **DROP TABLE** *nome da tabela*

- Criação de visões.

- Uma visão é um subconjunto lógico de dados de uma ou mais tabelas.
- Uma visão não armazena dados; todos os dados por ela mostrados provêm das tabelas por ela referenciadas.
- Vantagens do uso de uma visão.
 - Segurança
 - Restringe o acesso ao banco de dados porque uma visão pode mostrar apenas uma parte selecionada do banco de dados.
 - Simplificação das queries
 - Permite ao usuário fazer queries simples para recuperar informações a partir de queries complicadas.
 - Independência de dados
 - Para o usuário de uma visão não importa em que tabelas os dados estão armazenados.
 - Perspectiva
 - Proporciona aos diferentes grupos de usuários acesso aos dados de acordo com critérios particulares de cada grupo.
- Criação de visão com uma subquery embutida.
 - **CREATE VIEW** *nome da visão*

- Mostra o tipo da restrição
 - C - check ou not null
 - P - primary key
 - U - unique key
 - R - foreign key
 - TABLE_NAME
 - Mostra o nome da tabela associada com a restrição.
 - SEARCH_CONDITION
 - Mostra o texto da condição de pesquisa para uma restrição de CHECK.
 - R_OWNER
 - Mostra o nome do usuário da tabela referenciada em uma restrição de integridade referencial (chave estrangeira).
 - R_CONSTRAINT_NAME
 - Mostra o nome da restrição de chave primária na coluna referenciada de uma restrição de integridade referencial.
 - DELETE_RULE
 - Mostra a ação a ser tomada com as chaves estrangeiras que se referenciam a uma PRIMARY KEY (ou a uma chave UNIQUE) que vai ser excluída; as opções atualmente suportadas são CASCADE e NO ACTION (default).
 - STATUS
 - Usado internamente pelo Oracle.
- Ex04_09 - mostrar os nomes e os tipos das restrições criadas para a tabela X_EMPR (digitar o nome da tabela com letras maiúsculas). Dar novos nomes as colunas mostradas.
 - `SELECT constraint_name NOME, constraint_type TIPO
FROM user_constraints
WHERE table_name = 'X_EMPR'`
- Ex04_10- mostrar os nomes e o texto de validação para as restrições do tipo C da tabela X_EMPR. Dar novos nomes às colunas mostradas.
 - `SELECT constraint_name NOME, search_condition CONDIÇÃO
FROM user_constraints
WHERE table_name = 'X_EMPR'
AND constraint_type = 'C'`
 - Resultado do comando

NOME	CONDIÇÃO
x_empr_ult_nome_nn	ult_nome is not null
x_empr_senha_nn	senha is not null
x_empr_perc_comissao_ck	perc_comissao in (10,12.5,15,17.5,20)
sys_c00660	salario is not null
- Obter os nomes das colunas envolvidas em uma restrição.
 - A tabela USER_CONS_COLUMNS armazena os nomes das colunas envolvidas nas restrições. As colunas dessa tabela são:
 - OWNER
 - CONSTRAINT_NAME

- Ex04_04 - incluir a coluna Comentarios, com tamanho máximo de 255, na tabela X_REGIAO.
 - ALTER TABLE x_regiao
ADD comentarios VARCHAR2(255)
- Modificar uma coluna em uma tabela.
 - ALTER TABLE *nome da tabela*
MODIFY (*nome da coluna tipo de dado, ...*)
- Ex04_05 - aumentar o tamanho da coluna ULT_NOME da tabela X_EMPR para 30 posições.
 - ALTER TABLE x_empr
MODIFY (ult_nome VARCHAR2(30))
- Ex04_06 - modificar a coluna SALARIO da tabela X_EMPR para que ela passe a ser obrigatória.
 - ALTER TABLE x_empr
MODIFY (salario NOT NULL)
- Incluir uma restrição em uma tabela
 - ALTER TABLE *nome da tabela*
ADD CONSTRAINT *nome da restrição restrição*
- Ex04_07 - incluir a restrição na tabela X_EMPR para indicar que um gerente já deve existir como um empregado na tabela X_EMPR.
 - ALTER TABLE x_empr
ADD CONSTRAINT x_empr_id_gerente_fk
FOREIGN KEY (id_gerente) REFERENCES x_empr (id)
- Excluir uma restrição de uma tabela
 - ALTER TABLE *nome da tabela*
DROP CONSTRAINT *nome da restrição*
- Ex04_08 - excluir a restrição da tabela X_EMPR para indicar que um gerente já deve existir como um empregado na tabela X_EMPR.
 - ALTER TABLE x_empr
DROP CONSTRAINT x_empr_id_gerente_fk
- Queries ao dicionário de dados do Oracle
 - Verificar as restrições que foram criadas para uma tabela.
 - A visão USER_CONSTRAINTS armazena as restrições criadas para uma tabela de usuário. As colunas dessa tabela são:
 - OWNER
 - Mostra o nome do usuário que criou a restrição para uma dada tabela.
 - CONSTRAINT_NAME
 - Mostra o nome da restrição
 - CONSTRAINT_TYPE

Nome da coluna	Id	Ult nome	Prim nome	Senha	Dt_adm	Salario	Perc comissao	Id gerente	Id depto
Tipo de chave	PK							FK1	FK2
Nulo e/ou único		NN		NN, U					
Tabela referencia								x_empr	c_depto
Coluna referencia								id	id
Valores a verificar							10,12.5,15,17.5, 20		
Tipo de dado	Number	Varchar2	Varchar2	Varchar2	Date	Number	Number	Number	Number
Tamanho máximo	7	25	25	8		11,2	4,2	7	7
EXEMPLOS									

- CREATE TABLE x_empr
 - (id NUMBER(7),
 - ult_nome VARCHAR2(25)
 - CONSTRAINT x_empr_ult_nome_nn NOT NULL,
 - prim_nome VARCHAR2(25),
 - senha VARCHAR2(8)
 - CONSTRAINT x_empr_senha_nn NOT NULL,
 - dt_admissao DATE,
 - salario NUMBER(11,2),
 - id_gerente NUMBER(7),
 - id_depto NUMBER(7),
 - perc_comissao NUMBER(4,2),
 - CONSTRAINT x_empr_id_pk PRIMARY KEY (id),
 - CONSTRAINT x_empr_senha_uk UNIQUE (senha),
 - CONSTRAINT x_empr_perc_comissao_ck CHECK (perc_comissao IN (10,12.5,15,17.5,20)))
- Ex04_03 - mostrar a estrutura da tabela X_EMPR.
 - DESC x_empr
- Alteração de tabelas.
 - Incluir uma coluna em uma tabela.
 - ALTER TABLE nome da tabela
 - ADD (nome da coluna tipo de dado [DEFAULT expressão] [NOT NULL],
 -)

- Nome da coluna envolvida na restrição
- Abreviatura da restrição:
 - NN - not null
 - PK - primary key
 - FK - foreign key
 - UK - unique key
 - CK - check
 - Exemplo: c_regiao_id_pk

- Ex04_01 - criar a tabela X_REGIAO de acordo com o seguinte quadro de instâncias:

NOME DA TABELA: x_regiao

Nome da coluna	id	nome
Tipo de chave	PK	
Nulo e/ou único		NN, U
Tabela refer FK		
Coluna refer FK		
Valores a verificar		
Tipo de dado	Number	Varchar2
Tamanho máximo	7	50
EXEMPLOS	1	América do Norte
	2	América do Sul
	3	África e Oriente Médio
	4	Ásia
	5	Europa

- CREATE TABLE x_regiao
 (id NUMBER(7),
 nome VARCHAR2(50)
 CONSTRAINT x_regiao_nome_nn NOT NULL,
 CONSTRAINT x_regiao_id_pk PRIMARY KEY (id),
 CONSTRAINT x_regiao_nome_uk UNIQUE (nome))

- Ex04_02 - criar a tabela X_EMPR de acordo com o seguinte quadro de instâncias. **As chaves estrangeiras da tabela serão criadas posteriormente.**

NOME DA TABELA: x_empr

- restrição da coluna
 - é uma restrição de integridade como parte da definição da coluna
- restrição da tabela
 - é uma restrição de integridade como parte da definição da tabela

- Tipos de dados:
 - CHAR (n)
 - Coluna de tamanho fixo, de caracteres, com *n* tendo o valor máximo de 255.
 - VARCHAR2 (n)
 - Coluna de tamanho variável, de caracteres, com *n* tendo o valor máximo de 2000.
 - DATE
 - Coluna com valores representando datas.
 - LONG
 - Coluna de tamanho variável, de caracteres, com o valor máximo de 2 gigas.
 - NUMBER (w,d)
 - Coluna de tamanho variável, numérica, onde *w* é a quantidade total de dígitos e *d* é a quantidade de dígitos à direita do vírgula decimal.

- Restrições
 - NOT NULL
 - Especifica que a coluna tem preenchimento obrigatório.
 - É definida a nível de coluna.
 - UNIQUE
 - Especifica a coluna ou a combinação de colunas cujo valor deve ser único na tabela.
 - É definida a nível de coluna ou a nível de tabela.
 - Automaticamente cria um índice.
 - PRIMARY KEY
 - Especifica a chave primária da tabela.
 - É definida a nível de coluna ou a nível de tabela.
 - Automaticamente cria um índice.
 - Automaticamente cria uma restrição de NOT NULL.
 - FOREIGN KEY nome da coluna REFERENCES nome da tabela (nome da coluna)
 - Designa uma coluna ou combinação de colunas como uma chave estrangeira
 - Especifica um relacionamento entre a coluna (chave estrangeira) e a chave primária da tabela referenciada.
 - É definida a nível de coluna ou a nível de tabela.
 - CHECK
 - Especifica uma condição que deve ser verdadeira.
 - É definida a nível de coluna ou a nível de tabela.

- Nome da restrição
 - Identifica a restrição no dicionário de dados do Oracle.
 - Tamanho máximo de 30 caracteres (A-Z, a-z, 0-9, _, \$, #), começando com uma letra.
 - Deve ser assim formado:
 - Nome da tabela

4 - CRIAR TABELAS E ESTRUTURAS DE DADOS

- Objetivos
 - Criar tabelas contendo restrições para integridade
 - Descrever os tipos de dados que podem ser usados ao especificar as definições das colunas
 - Reconhecer os índices que são criados automaticamente pelas restrições
 - Criar uma tabela e preenchê-la com as linhas de outra tabela
 - Modificar a estrutura de uma tabela existente
 - Reforçar a integridade dos dados da empresa ao adicionar e remover restrições à definição de uma tabela
 - Criar visões lógicas de uma tabela
 - Gerar valores de chave primária usando seqüências
 - Determinar quando aumentar a performance de algumas queries através da criação de índices.
- Estruturas de dados existentes no Oracle e seus objetivos:
 - Tabelas
 - Armazenam dados.
 - Visões
 - Representam subconjuntos lógicos de dados de uma ou mais tabelas.
 - Seqüências
 - Geram valores para chaves primárias.
 - Índices
 - Aumentam o desempenho na execução de algumas queries.
- Documentar as tabelas, colunas, restrições e exemplos de linhas através de um quadro de instâncias, usando as seguintes convenções:
 - PK - Indica a coluna que é chave primária.
 - FK - Indica a coluna que é chave estrangeira.
 - FK1, FK2 - Indicam duas chaves estrangeiras na mesma tabela.
 - FK1, FK1 - Indicam duas colunas formando uma chave estrangeira composta.
 - NN - Indica uma coluna cujo conteúdo é obrigatório.
 - U - Indica uma coluna cujo conteúdo é único dentro da tabela.
 - U1,U1 - Indicam duas colunas cujos conteúdos, combinados, são únicos dentro da tabela.
- Criação de tabelas para armazenar dados.
 - **CREATE TABLE** nome da tabela
(nome da coluna tipo de dado [**DEFAULT** expressão] [restrição da coluna],
.....
[restrição da tabela]);

onde

- **DEFAULT** expressão
 - especifica o valor padrão se um valor for omitido no comando INSERT
- tipo de dado
 - é o tipo de dado armazenado na coluna, e o seu tamanho

```
INSERT INTO c_regiao (id,nome)
VALUES (52,'Italia');
INSERT INTO c_regiao (id,nome)
VALUES (53,'Canada');
```

- Ex03_43 - executar este script.
 - @ a:ex03_42.sql
- Criar um arquivo texto para carregar dados de forma interativa, embutindo parâmetros de substituição.
 - Ex03_44 - usando o Bloco de Notas do Windows 95, criar um arquivo sql (Ex03_44.sql) para criar uma nova região, baseado em valores especificados pelo usuário de forma interativa.
 - SET ECHO OFF

```
INSERT INTO c_regiao (id,nome)
VALUES (&id_regiao, '&nome_região')
```

 - Observação: o parâmetro *&nome_região* está entre aspas simples para evitar que o nome da região seja digitado entre aspas.
 - Ex03_45 – criar a região de identificador 20 e nome Eldorado, executando o script criado no exercício anterior.
 - @ a:ex03_44.sql
 - Ex03_46 - criar um arquivo (Ex03_46.sql) para criar uma nova região baseada em valores especificados pelo usuário de forma interativa, colocando as seguintes mensagens para substituição dos parâmetros: "Informar identificador da região:" e "Informar nome da região:". Usar o comando SET VERIFY OFF para que não apareçam na tela as mensagens referentes às substituições dos parâmetros.
 - SET ECHO OFF
 - SET VERIFY OFF

```
ACCEPT id_regiao PROMPT 'INFORMAR IDENTIFICADOR DA REGIÃO : '
```

```
ACCEPT nome_regiao PROMPT 'INFORMAR NOME DA REGIÃO : '
```

```
INSERT INTO c_regiao (id,nome)
VALUES (&id_regiao, '&nome_regiao')
```
- Ex03_47 criar a região de identificador 25 e nome Maravilha, executando o script criado no exercício anterior.
 - @ a:ex0346.txt

- Ex03_35 - criar o departamento de Treinamento 4 na região 4.
 - INSERT INTO c_depto
VALUES (c_id_depto.NEXTVAL,'Treinamento 4',4)
- Ex03_36 - criar o ponto de salvamento chamado *PONTO2*.
 - SAVEPOINT ponto2
- Ex03_37 - criar o departamento de Treinamento 5 na região 5.
 - INSERT INTO c_depto
VALUES (c_id_depto.NEXTVAL,'Treinamento 5',5)
- Ex03_38 - mostrar os departamentos de Treinamento.
 - SELECT *
FROM c_depto
WHERE nome LIKE 'Treinamento%'
- Ex03_39 - desfazer as duas últimas criações.
 - ROLLBACK TO SAVEPOINT ponto1
- Ex03_40 - concretizar as alterações.
 - COMMIT
- Ex03_41 - mostrar os departamentos de Treinamento.
 - SELECT *
FROM c_depto
WHERE nome LIKE 'Treinamento%'
- Situações nas quais o COMMIT e o ROLLBACK são implícitos.
 - Execução de um comando DDL, como um CREATE TABLE
 - COMMIT automático
 - Saída normal do SQL*Plus, sem que tenha sido explicitado COMMIT ou ROLLBACK
 - COMMIT automático
 - Término anormal do SQL*Plus ou queda do sistema
 - ROLLBACK automático
 - Recomendação
 - Sempre explicitar o COMMIT e o ROLLBACK
- Criar um arquivo texto para carregar uma grande quantidade de dados de uma vez só, através da inclusão de vários comandos INSERT
 - Ex03_42 - usando o Bloco de Notas do Windows 95, criar um arquivo (Ex03_42.sql) contendo um script para incluir três novas regiões. Usar SET ECHO OFF para que os comandos não apareçam na tela quando forem executados. Salvar este arquivo (em um disquete, por exemplo) com o nome de Ex03_42.sql.
 - SET ECHO OFF
INSERT INTO c_regiao (id,nome)
VALUES (51,'Brasil');

WHERE id = 2

- Ex03_28 - concretizar essas modificações.
 - COMMIT
- Situação do dado depois do ROLLBACK.
 - As mudanças são desfeitas. O conteúdo anterior do dado é restabelecido.
 - Os bloqueios nas linhas afetadas são desfeitos; as linhas ficam disponíveis para que outros usuários possam executar novas alterações.
- Ex03_29 - criar o departamento de Relações Trabalhistas, na região 3.
 - INSERT INTO c_depto
(id,nome,id_regiao)
VALUES (c_id_depto.NEXTVAL,'Relações Trabalhistas',3)
- Ex03_30 - mostrar os departamentos da região 3.
 - SELECT *
FROM c_depto
WHERE id_regiao = 3
- Ex03_31 - desfazer essa modificação.
 - ROLLBACK
- Ex03_32 - mostrar os departamentos da região 3.
 - SELECT *
FROM c_depto
WHERE id_regiao = 3
 - Verifica-se que o departamento de Relações Trabalhistas não é mostrado.
- Alterar a lógica das transações com savepoints.
 - Pode-se criar um ponto de salvamento dentro de uma transação corrente através do uso do comando SAVEPOINT. Assim a transação é dividida em partes menores. Pode-se então descartar-se as alterações pendentes até o ponto de salvamento especificado, através do uso do comando ROLLBACK TO SAVEPOINT.
 - Um savepoint marca um ponto intermediário no processamento de uma transação.
 - **SAVEPOINT nome do ponto de salvamento**
 - Marca um ponto de salvamento dentro da transação.
 - **ROLLBACK TO SAVEPOINT nome do ponto de salvamento**
 - Descarta as alterações pendentes feitas depois de que um ponto de salvamento foi marcado.
- Ex03_33 - criar o departamento de Treinamento 3 na região 3.
 - INSERT INTO c_depto
VALUES (c_id_depto.NEXTVAL,'Treinamento 3',3)
- Ex03_34 - criar o ponto de salvamento chamado *PONTO1*.
 - SAVEPOINT ponto1

- Um comando DDL (CREATE, ALTER, DROP, RENAME) ou um comando DCL (GRANT, REVOKE, AUDIT, DROP, RENAME) aparece
 - Certos erros são identificados, como um deadlock
 - O usuário sai do SQL*Plus
 - O computador é desligado
- Após uma transação terminar, o próximo comando SQL executável automaticamente inicia uma nova transação.
- COMMIT
 - Encerra a transação corrente fazendo com que todas as modificações pendentes passem a ser definitivas.
 - ROLLBACK
 - Encerra a transação corrente fazendo com que todas as modificações pendentes sejam desprezadas.
 - Todas as modificações feitas durante a transação são temporárias até que a transação seja “committed” (concretizada).
 - Situação do dado antes do COMMIT ou do ROLLBACK
 - As operações de manipulação de dados primeiramente afetam o buffer do banco de dados.
 - O usuário corrente pode rever os resultados das operações de manipulação de dados usando o comando SELECT.
 - Outros usuários NÃO podem ver os resultados das operações de manipulação de dados do usuário corrente.
 - As linhas afetadas ficam bloqueadas; outros usuários não podem modificar os dados existentes nas linhas afetadas.
 - Situação do dado depois do COMMIT.
 - As modificações são concretizadas no banco de dados. O conteúdo anterior do dado é definitivamente perdido.
 - Todos os usuários podem ver os resultados da transação.
 - Os bloqueios nas linhas afetadas são desfeitos; as linhas ficam disponíveis para que outros usuários possam executar novas alterações.
 - Ex03_25 - recriar o departamento de Treinamento 2, na região 2
 - INSERT INTO c_depto
(id,nome,id_regiao)
VALUES (c_id_depto.NEXTVAL,'Treinamento 2',2)
 - Ex03_26 - mostrar os departamentos de Treinamento.
 - SELECT *
FROM c_depto
WHERE nome LIKE 'Treinamento%'
 - Ex03_27 - transferir o empregado de identificador 2 para o departamento 53
 - UPDATE c_empr
SET id_depto = 53

- Modificar todas as linhas de uma tabela omitindo a cláusula WHERE.
 - Ex03_19 - estabelecer uma comissão de 10% para todos os empregados.
 - UPDATE c_empr
SET perc_comissao = 10
 - Ex03_20 - selecionar dados dos empregados para confirmar as modificações.
 - SELECT id,ult_nome,perc_comissao
FROM c_empr
- Excluir uma linha de uma tabela.
 - **DELETE FROM *nome da tabela***
WHERE *condição*
 - Ex03_21 - excluir todas as informações do departamento 52.
 - DELETE FROM c_depto
WHERE ID = 52
 - Ex03_22 - selecionar dados do departamento 52 para confirmar a exclusão.
 - SELECT id
FROM c_depto
WHERE id = 52
- Excluir várias linhas de uma tabela usando a cláusula WHERE para identificar um conjunto de linhas.
 - Ex03_23 - excluir os empregados do departamento 50
 - DELETE FROM c_empr
WHERE id_depto = 50
 - Ex03_24 - mostrar a exclusão.
 - SELECT *
FROM c_empr
WHERE id_depto = 50
- Controlar a lógica das transações usando os comandos COMMIT e ROLLBACK
 - O Oracle assegura a consistência dos dados baseado nas transações. As transações dão mais flexibilidade e controle quando da mudança do conteúdo das tabelas e asseguram a consistência dos dados em caso de falhas nos processos do usuário ou falhas no sistema.
 - Uma transação consiste de comandos DML (insert, update,delete,commit,rollback) que fazem uma mudança consistente nos dados. Por exemplo, uma transferência de valores entre duas contas bancárias implica no débito em uma conta e no crédito em outra no mesmo montante. Ambas as ações ou são realizadas ou são anuladas. O crédito não pode ser concretizado sem o correspondente débito.
 - Uma transação começa quando o primeiro comando SQL executável é encontrado e termina quando:
 - Um comando COMMIT ou ROLLBACK aparece

```
(id,nome,id_regiao)
VALUES (c_id_depto.NEXTVAL,'Treinamento 1',1)
```

- Ex03_12 - criar um novo departamento chamado Treinamento 2, na região 2. O identificador de cada departamento deve ser criado automaticamente.
 - INSERT INTO c_depto
(id,nome,id_regiao)
VALUES (c_id_depto.NEXTVAL,'Treinamento 2',2)
- Ex03_13 - mostrar os novos departamentos
 - SELECT *
FROM c_depto
WHERE nome LIKE 'Treinamento%'
- Modificar dados de uma linha de uma tabela
 - UPDATE *nome da tabela*
SET *nome da coluna 1 = valor 1, ..., nome da coluna 2 = valor 2*
WHERE *condição*
- Ex03_14 - transferir o empregado de identificador 2 para o departamento 10
 - UPDATE c_empr
SET id_depto = 10
WHERE id = 2
- Ex03_15 - transferir o empregado de identificador 1 para o departamento 32 e alterar seu salário para \$ 2.550
 - UPDATE c_empr
SET id_depto = 32, salario = 2550
WHERE id = 1
- Ex03_16 - confirmar as duas modificações, mostrando identificação, primeiro nome, último nome, salário e código do departamento.
 - SELECT id, prim_nome, ult_nome, salario, id_depto
FROM c_empr
WHERE id IN (1,2)
- Modificar várias linhas de uma tabela usando a cláusula WHERE para identificar um conjunto de linhas.
 - Ex03_17 – colocar o comentário "em observação" em todos os empregados do departamento 41.
 - UPDATE c_empr
SET comentarios = 'em observação'
WHERE id_depto = 41;
 - Ex03_18 – selecionar último nome do empregado, identificação do seu departamento e comentários para confirmar as modificações.
 - SELECT id, ult_nome, id_depto, comentarios
FROM c_empr
WHERE id_depto = 41;

- Implicitamente: omitindo o nome da coluna na lista de colunas
- Evidentemente, a coluna em questão deve permitir valores nulos
- Ex03_05 - incluir o novo cliente Esportes Sobre Rodas, identificador 216, situação de crédito boa, tendo como representante de vendas o empregado de identificador 12 e pertencente à região 2. As demais colunas devem ter valores nulos.
 - INSERT INTO c_cliente
VALUES (216,'Esportes sobre Rodas', NULL, NULL, NULL, NULL, NULL, NULL, 'BOA',12,2, NULL)
- Ex03_06- mostrar identificação, nome e situação de crédito do novo cliente.
 - SELECT id, nome, sit_cred
FROM c_cliente
WHERE id=216;
- Ex03_07 - incluir o novo cliente Superball , identificador 217, situação de crédito boa, tendo como representante de vendas o empregado de identificador 12 e pertencente à região 2. Inserir os valores nulos com um string vazio.
 - INSERT INTO c_cliente
VALUES (217,'Superball', ' ', ' ', ' ', ' ', ' ', ' ', 'BOA',12,2, ' ')
- Ex03_08 - incluir o novo cliente Academia do Corpo, identificador 218, situação de crédito boa, tendo como representante de vendas o empregado de identificador 12 e pertencente à região 2. Não sabemos, no momento, telefone, endereço, cidade, estado, país e CEP. Inserir os valores nulos de forma implícita.
 - INSERT INTO c_cliente
(id,nome, sit_cred, id_repr_vendas, id_regiao)
VALUES (218,'Academia do Corpo', 'BOA',12,2)
- Ex03_09 - mostrar identificação, nome, situação de crédito dos clientes incluídos(id 216, 217,218).
 - SELECT *
FROM c_cliente
WHERE id IN (216, 217, 218)
- Geração automática de valores de uma chave primária.
 - **sequence. NEXTVAL**
 - Gera o próximo valor disponível de uma seqüência
 - Antes de gerar um valor de uma chave primária, confirme o nome da seqüência usando a visão USER_OBJECTS do dicionário de dados e cujas colunas foram mostradas no exercício 01_46.
- Ex03_10 - mostrar os nomes de todas as seqüências autorizadas ao usuário desta sessão.
 - SELECT object_name
FROM user_objects
WHERE object_type='SEQUENCE'
- Ex03_11 - criar um novo departamento chamado Treinamento 1, na região 1. O identificador de cada departamento deve ser criado automaticamente.
 - INSERT INTO c_depto

3 - MANIPULAÇÃO DE DADOS

- Objetivos
 - Adicionar novas linhas a uma tabela
 - Inserir um valor nulo numa coluna de uma nova linha
 - Gerar automaticamente valores de chave primária para as novas linhas
 - Modificar linhas já existentes em uma tabela
 - Modificar várias linhas de uma tabela usando a cláusula WHERE para identificar um conjunto de linhas.
 - Modificar várias linhas de uma tabela omitindo a cláusula WHERE.
 - Modificar todas as linhas de uma tabela omitindo a cláusula WHERE.
 - Remover linhas existentes em uma tabela
 - Controlar a lógica das transações usando os comandos COMMIT e ROLLBACK
 - Alterar a lógica das transações com savepoints.
 - Criar um arquivo texto para carregar uma grande quantidade de dados de uma vez só.
 - Criar um arquivo texto para carregar dados de forma interativa, embutindo parâmetros de substituição.
 - Executar o mesmo comando SQL repetidamente com valores diferentes
- Incluir dados em uma tabela
 - **INSERT INTO *nome da tabela* (*nome da coluna 1*, *nome da coluna 2*, ..., *nome da coluna n*)
VALUES (*valor da coluna 1*, *valor da coluna 2*, ..., *valor da coluna n*)**
 - Ex03_01 - incluir na tabela Departamento o departamento Financeiro, código 11, da região 2
 - INSERT INTO c_depto
VALUES (11,'Financeiro',2)
 - Ex03_02 - mostrar a inclusão do novo departamento
 - SELECT * FROM c_depto
 - Ex03_03 - incluir a novo empregado Elizabeth Hering, identificador 26, código de departamento 32, senha ehering, admitido hoje (a data de hoje é obtida pela função SYSDATE) e gerenciada pelo empregado de identificador 2.
 - INSERT INTO c_empr
(id, ult_nome,prim_nome,senha,dt_admissao,id_gerente,id_depto)
VALUES (26,'Hering','Elizabeth','ehering',SYSDATE,2,32)
 - Ex03_04 - mostrar identificação, último nome, primeiro nome e data de admissão do novo empregado
 - SELECT id, ult_nome,prim_nome,dt_admissao
FROM c_empr
WHERE id=26;
- Inserir um valor nulo numa coluna de uma nova linha
 - Explicitamente:
 - especificando a palavra chave NULL na lista de valores
 - para atributos tipo CHAR ou DATE, especificando um string vazio entre aspas simples na lista de valores

```
FROM c_fatura
ORDER BY id_repr_vendas,id_cliente,dt_fatura;
```

```
SET PAGESIZE 14
SET LINESIZE 80
SET SPACE 1
TTITLE OFF
BTITLE OFF
CLEAR COLUMNS
CLEAR BREAK
CLEAR COMPUTES
SET FEEDBACK ON
```

- Ex02_22 - Salvar o relatório no arquivo spool vendas.lst.

- SET FEEDBACK OFF
 - SET PAGESIZE 35
 - SET LINESIZE 60
 - SET SPACE 2

```
TTITLE 'Relatório|de|Vendas'
BTITLE 'Em ordem de representantes de vendas'
COLUMN id_repr_vendas HEADING 'Representante|de Vendas'
COLUMN id_cliente HEADING 'Ident|Cliente'
COLUMN dt_fatura HEADING 'Data da|Fatura'
COLUMN total HEADING 'Total' FORMAT $9,999,999.99
BREAK ON id_repr_vendas SKIP 1 ON id_cliente SKIP 1 ON REPORT
COMPUTE SUM OF total ON id_repr_vendas REPORT
```

SPOOL vendas.lst

```
SELECT id_repr_vendas,id_cliente,dt_fatura,total
FROM c_fatura
ORDER BY id_repr_vendas,id_cliente,dt_fatura;
SET PAGESIZE 14
SET LINESIZE 80
SET SPACE 1
TTITLE OFF
BTITLE OFF
CLEAR COLUMNS
CLEAR BREAK
CLEAR COMPUTES
SET FEEDBACK ON
```

Fri Aug 08

page 2

Relatório
de
Vendas

Representante de Vendas	Ident Cliente	Data da Fatura	Total
-----	-----	-----	-----
14	202	31/08/92	\$595.00
	203	31/08/92	\$7,707.00
	205	31/08/92	\$8,056.60
*****			-----
sum			\$16,358.60
15	206	01/09/92	\$8,335.00
	208	02/09/92	\$377.00
		03/09/92	\$32,430.00
	211	07/09/92	\$142,171.00
*****			-----
sum			\$183,313.00
*****			-----
sum			\$2,078,491.97

Em ordem de representantes de vendas

- SET FEEDBACK OFF
- SET PAGESIZE 35
- SET LINESIZE 60
- SET SPACE 2

```
TTITLE 'Relatório|de|Vendas'
BTITLE 'Em ordem de representantes de vendas'
COLUMN id_repr_vendas HEADING 'Representante|de Vendas'
COLUMN id_cliente HEADING 'Ident|Cliente'
COLUMN dt_fatura HEADING 'Data da Fatura'
COLUMN total HEADING 'Total' FORMAT $9,999,999.99
BREAK ON id_repr_vendas SKIP 1 ON id_cliente SKIP 1 ON REPORT
```

- COMPUTE SUM OF total ON id_repr_vendas REPORT

- SELECT id_repr_vendas,id_cliente,dt_fatura,total

• **Modelo do relatório**

Fri Aug 08

page 1

Relatório
de
Vendas

Representante de Vendas	Ident Cliente	Data da Fatura	Total
-----	-----	-----	-----
11	204	31/08/92	\$601,100.00
		09/09/92	\$2,770.00
	209	04/09/92	\$2,722.24
	213	08/09/92	\$1,020,935.00
	214	09/09/92	\$1,539.13
*****	*****		-----
sum			\$1,629,066.37
12	201	28/08/92	\$84,000.00
	210	31/08/92	\$550.00
		07/09/92	\$15,634.00
*****	*****		-----
sum			\$100,184.00
13	212	07/09/92	\$149,570.00
*****	*****		-----
sum			\$149,570.00

Em ordem de representantes de vendas

EXERCÍCIO DE REVISÃO

- Ex02_21 - Preparar um relatório com as seguintes características
 - Cabeçalho: Relatório de Vendas Empregados, em 3 linhas
 - Rodapé: Em ordem de representante de vendas
 - Cada página com 35 linhas, 60 caracteres por linha.
 - Supressão da mensagem que informa a quantidade de registros recuperados.
 - Conteúdo: identificação do representante de vendas, identificação do cliente, data da fatura e total da fatura.
 - Ordenação: crescente de identificação do representante de vendas, de identificação do cliente e da data da fatura.
 - Quebra por representante de vendas e por cliente.
 - Total das faturas por representante de vendas e total geral.
 - Após a impressão do relatório, restabelecer os valores defaults.

Fri Aug 08

page 1

Relatório
dos
Empregados

FUNÇÃO	SOBRENOME	ID	SALÁRIO ATUAL
-----	-----	-----	-----
Almoxarife	Madeira	16	\$1,400.00
	Nozaki	18	\$1,200.00
	Schwartz	25	\$1,100.00
	Silva	17	\$940.00
	Dantas	24	\$860.00
	Martins	21	\$850.00
	Chang	22	\$800.00
	Pires	19	\$795.00
	Pires	23	\$795.00
	Newman	20	\$750.00
*****			-----
sum			\$9,490.00
Presidente	Velasquez	1	\$2,500.00
*****			-----
sum			\$2,500.00
Representante Vendas Nice		14	\$1,525.00
	Sedeghi	13	\$1,515.00
	Guimarães	12	\$1,490.00
	Dumas	15	\$1,450.00
	Margarida	11	\$1,400.00
*****			-----
sum			\$7,380.00
*****			-----
sum			\$19,370.00
	Confidencial		

ORDER BY cargo,salario DESC,ult_nome;

- Ex02_19 - Voltar aos valores defaults.

- SET PAGESIZE 14
SET LINESIZE 80
SET FEEDBACK ON
TTITLE OFF
BTITLE OFF
CLEAR COLUMNS
CLEAR BREAKS
CLEAR COMPUTE

SPOOL OFF

- Ex02_20 - Usando qualquer sistema operacional listar o arquivo c:\orawin\bin\rel_sala.lst.

```
COLUMN ult_nome HEADING 'SOBRENOME' FORMAT A15  
COLUMN salario HEADING 'SALÁRIO|ATUAL' FORMAT $99,999.99
```

```
BREAK ON cargo SKIP 1 ON REPORT  
COMPUTE SUM OF salario ON cargo REPORT
```

```
SELECT cargo,ult_nome,id,salario  
FROM c_empr  
WHERE cargo NOT LIKE 'VP%' AND  
       cargo NOT LIKE 'Gerente de Almoxarifado'  
ORDER BY cargo,salario DESC,ult_nome;
```

- Ex02_17 - Voltar aos valores defaults.
 - SET PAGESIZE 14
SET LINESIZE 80
SET FEEDBACK ON
TTITLE OFF
BTITLE OFF
CLEAR COLUMNS
CLEAR BREAKS
CLEAR COMPUTE
- Salvar os comandos de geração do relatório.
 - **SPOOL *nome do arquivo*.LST**
 - **SPOOL**
 - mostra o que está estabelecido no momento para este comando.
 - **SPOOL OFF**
 - fecha o arquivo de spool.
- Ex02_18 - Salvar o relatório no arquivo spool rel_sala.lst.
 - SET PAGESIZE 37
SET LINESIZE 75
SET FEEDBACK OFF
TTITLE 'Relatório|dos|Empregados'
BTITLE 'Confidencial'
COLUMN cargo HEADING 'FUNÇÃO' FORMAT A25
COLUMN ult_nome HEADING 'SOBRENOME' FORMAT A15
COLUMN salario HEADING 'SALÁRIO|ATUAL' FORMAT \$99,999.99

```
SPOOL rel_sala.lst
```

```
BREAK ON cargo skip 1 ON REPORT  
COMPUTE SUM OF salario ON cargo REPORT
```

```
SELECT cargo,ult_nome,id,salario  
FROM c_empr  
WHERE cargo NOT LIKE 'VP%' AND  
       cargo NOT LIKE 'Gerente de Almoxarifado'
```

- **COMPUTE**
 - mostra o que está estabelecido no momento para este comando.
- **CLEAR COMPUTE**
 - limpa o que foi estabelecido para este comando.
- Ex02_14- No relatório do Ex02_12,incluir subtotais por cargo.
 - SET PAGESIZE 37
SET LINESIZE 75
SET FEEDBACK OFF
TTITLE 'Relatório|dos|Empregados'
BTITLE 'Confidencial'
COLUMN cargo HEADING 'FUNÇÃO' FORMAT A25
COLUMN ult_nome HEADING 'SOBRENOME' FORMAT A15
COLUMN salario HEADING 'SALÁRIO|ATUAL' FORMAT \$99,999.99
BREAK ON cargo SKIP 1

COMPUTE SUM OF salario ON cargo

```
SELECT cargo,ult_nome,id,salario
FROM c_empr
WHERE cargo NOT LIKE 'VP%' AND
      cargo NOT LIKE 'Gerente de Almoxarifado'
ORDER BY cargo,salario DESC,ult_nome;
```

- Ex02_15 - Voltar aos valores defaults.
 - SET PAGESIZE 14
SET LINESIZE 80
SET FEEDBACK ON
TTITLE OFF
BTITLE OFF
CLEAR COLUMNS
CLEAR BREAKS

CLEAR COMPUTE

- Calcular e imprimir linhas de totais finais baseadas nos valores de uma coluna usando BREAK e COMPUTE juntos.

- **BREAK ON COLUMN | REPORT SKIP n | PAGE**
- **COMPUTE *função* OF nome da coluna de cálculo
ON nome da coluna de quebra ... REPORT**

- Ex02_16 - No relatório do Ex02_14 imprimir o total final dos salários.
 - SET PAGESIZE 37
SET LINESIZE 75
SET FEEDBACK OFF
TTITLE 'Relatório|dos|Empregados'
BTITLE 'Confidencial'
COLUMN cargo HEADING 'FUNÇÃO' FORMAT A25

```
SET FEEDBACK OFF
TTITLE 'Relatório|dos|Empregados'
BTITLE 'Confidencial'
COLUMN cargo HEADING 'FUNÇÃO' FORMAT A25
COLUMN ult_nome HEADING 'SOBRENOME' FORMAT A15
COLUMN salario HEADING 'SALÁRIO|ATUAL' FORMAT $99,999.99
BREAK ON cargo SKIP 1
```

```
SELECT cargo,ult_nome,id,salario
FROM c_empr
WHERE cargo NOT LIKE 'VP%' AND
      cargo NOT LIKE 'Gerente de Almojarifado'
ORDER BY cargo,salario DESC,ult_nome;
```

- Ex02_13 - Voltar aos valores defaults.

- SET PAGESIZE 14
- SET LINESIZE 80
- SET FEEDBACK ON
- TTITLE OFF
- BTITLE OFF
- CLEAR COLUMNS

CLEAR BREAKS

- Incluir cálculos sumarizados.

- **COMPUTE** *função* **OF** *nome da coluna do cálculo*
ON *nome da coluna da quebra*

- onde *função* pode ser
 - COUNT
 - contagem de valores não-nulos
 - NUM
 - contagem de linhas
 - MAX
 - valor máximo
 - MIN
 - valor mínimo
 - SUM
 - total de valores não-nulos
 - AVG
 - média de valores não-nulos
 - STD
 - desvio padrão de valores não-nulos
 - VAR
 - variância de valores não-nulos
- Exemplo
 - COMPUTE AVG OF salario ON id_depto

```
WHERE cargo NOT LIKE 'VP%' AND
      cargo NOT LIKE 'Gerente de Almoxarifado'
ORDER BY cargo,salario DESC,ult_nome;
```

- Ex02_10 - No relatório do Ex02_09

Na coluna SALÁRIO ATUAL usar máscara de edição que suprima todos os zeros não significativos pelo cifrão flutuante.

```
• SET PAGESIZE 37
SET LINESIZE 75
SET FEEDBACK OFF
TTITLE 'Relatório|dos|Empregados'
BTITLE 'Confidencial'
COLUMN cargo HEADING 'FUNÇÃO' FORMAT A25
COLUMN ult_nome HEADING 'SOBRENOME' FORMAT A15
COLUMN salario HEADING 'SALÁRIO|ATUAL' FORMAT $99,999.99
```

```
SELECT cargo,ult_nome,id,salario
FROM c_empr
WHERE cargo NOT LIKE 'VP%' AND
      cargo NOT LIKE 'Gerente de almoxarifado'
ORDER BY cargo,salario DESC,ult_nome;
```

- Ex02_11 - Voltar aos valores defaults.

```
SET PAGESIZE 14
SET LINESIZE 80
SET FEEDBACK ON
TTITLE OFF
BTITLE OFF
CLEAR COLUMNS
```

- Construir uma quebra

- **BREAK ON *nome da coluna***
[ON *nome da coluna ...*] | **SKIP *n*** | **PAGE**

onde

nome da coluna é a coluna da quebra.

SKIP *n* é a quantidade de linhas a ser pulada entre cada quebra

PAGE salta para a próxima página antes de imprimir a linha onde a quebra ocorreu. A quantidade de linhas por página pode ser estabelecida com SET PAGESIZE

- supprime a exibição de valores duplicatas para a coluna especificada.
 - pula uma linha cada vez que o conteúdo da coluna muda.
 - imprime subtotais cada vez que o conteúdo da coluna muda no final do relatório.
- Ex02_12- No relatório do Ex02_10, não repetir o nome do cargo do empregado.
 - SET PAGESIZE 37
 - SET LINESIZE 75

```

• SET PAGESIZE 14
  SET LINESIZE 80
  SET FEEDBACK ON
  TTITLE OFF
  BTITLE OFF

```

- Controle da exibição das colunas

- **COLUMN** *nome da coluna* **HEADING** *cabeçalho*
FORMAT *máscara de edição*

onde

máscara de edição especifica como o dado da coluna será mostrado.

- 9 representa supressão do 0 (zero).
- ,
- .
- \$ representa o cifrão flutuante.
- An estabelece a largura *n* para colunas com dados tipo CHAR ou DATE.

- **COLUMN** *nome da coluna*
 - mostra o que foi estabelecido para uma determinada coluna.
- **COLUMN**
 - mostra o que foi estabelecido para todas as colunas.
- **COLUMN** *nome da coluna* **CLEAR**
 - limpa o que foi estabelecido para uma determinada coluna.
- **CLEAR COLUMNS**
 - limpa o que foi estabelecido para todas as colunas.
- Ex02_09 - No relatório do Ex02_07
 - Na coluna Cargo usar o cabeçalho FUNÇÃO, com tamanho de 25 posições.
 - Na coluna Ult_nome usar o cabeçalho SOBRENOME, com tamanho de 15 posições.
 - Na coluna Salario usar o cabeçalho SALÁRIO ATUAL, em duas linhas e com máscara de edição que suprima todos os zeros não significativos.

```

• SET PAGESIZE 37
  SET LINESIZE 75
  SET FEEDBACK OFF
  TTITLE 'Relatório|dos|Empregados'
  BTITLE 'Confidencial'
  COLUMN cargo HEADING 'FUNÇÃO'
  COLUMN ult_nome HEADING 'SOBRENOME' FORMAT A15
  COLUMN salario HEADING 'SALÁRIO|ATUAL' FORMAT 99,999.99

```

```

SELECT cargo,ult_nome,id,salario
FROM c_empr

```

Fri Aug 08

page 1

Relatório
dos
Empregados

FUNÇÃO	SOBRENOME	ID	SALÁRIO ATUAL
Almoxarife	Madeira	16	\$1,400.00
	Nozaki	18	\$1,200.00
	Schwartz	25	\$1,100.00
	Silva	17	\$940.00
	Dantas	24	\$860.00
	Martins	21	\$850.00
	Chang	22	\$800.00
	Pires	19	\$795.00
	Pires	23	\$795.00
	Newman	20	\$750.00
*****			-----
sum			\$9,490.00
Presidente	Velasquez	1	\$2,500.00
*****			-----
sum			\$2,500.00
Representante Vendas Nice		14	\$1,525.00
	Sedeghi	13	\$1,515.00
	Guimarães	12	\$1,490.00
	Dumas	15	\$1,450.00
	Margarida	11	\$1,400.00
*****			-----
sum			\$7,380.00
sum		-----	\$19,370.00

- SET PAGESIZE 37
SET LINESIZE 75
SET FEEDBACK OFF
TTITLE 'Relatório|dos|Empregados'
BTITLE 'Confidencial'

```
SELECT cargo,salario,ult_nome,id
FROM c_empr
WHERE cargo NOT LIKE 'VP%' AND
      cargo NOT LIKE 'Gerente de Almoxarifado'
ORDER BY cargo,salario DESC,ult_nome;
```

- Ex02_08 - Voltar aos valores defaults.

- Comandos de formato
 - **TTITLE** *título*
 - coloca um *título* no cabeçalho de cada página.
 - **TTITLE OFF**
 - **BTITLE** *título*
 - coloca um *título* no rodapé de cada página.
 - **BTITLE OFF**
 - **COLUMN** *nome da coluna* **HEADING** *cabeçalho* **FORMAT** *máscara de edição*
 - controla os atributos de exibição de cada coluna.
 - **BREAK**
 - grupa subconjuntos de dados
 - **COMPUTE**
 - produz cálculos sumarizados.

- Ex02_07 - Preparar um relatório com as seguintes características
 - Cabeçalho: Relatório dos Empregados, em 3 linhas
 - Rodapé: Confidencial
 - Cada página com 37 linhas, 75 caracteres por linha.
 - Supressão da mensagem que informa a quantidade de registros recuperados.
 - Conteúdo: cargo, salário, sobrenome e identificação dos empregados que não tenham o cargo de vice-presidente nem sejam gerente de almoxarifado.
 - Ordenação: crescente de cargo, decrescente de salário e crescente de sobrenome.

- ```
FROM c_regiao;
```
- Não aparece a mensagem porque a quantidade de registros selecionada é 5 e o default é 6.
  - Ex02\_03 - mudar o FEEDBACK para 2 e mostrar as regiões.
    - SET FEEDBACK 2

```
SELECT *
FROM c_regiao;
```
  - Ex02\_04 - Voltar aos valores defaults.
    - SET FEEDBACK 6
  - Ex02\_05 - Alterar o ambiente para exibir 10 linhas de cada vez, para exibir mensagem para continuar a cada 10 linhas. Mostrar sobrenome e primeiro nome de todos os empregados.
    - SET PAGESIZE 10
    - SET PAUSE 'Pressione qualquer tecla para continuar'
    - SET PAUSE ON

```
SELECT ult_nome,prim_nome
FROM c_empr;
```
  - Ex02\_06 - Voltar aos valores defaults.
    - SET PAGESIZE 14
    - SET PAUSE OFF

## 2 - CONSTRUIR RELATÓRIOS EMERGENCIAIS USANDO O SQL\*Plus

### • OBJETIVOS

- Controlar o ambiente de geração de relatórios de um sessão do SQL\*Plus.
- Formatar um relatório.
- Salvar os comandos para gerar um relatório.
- Salvar o relatório em um arquivo texto.
- Criar um relatório simples usando comandos SQL\*Plus
- Comandos de ambiente - afetam o comportamento geral da sessão.
- Comandos de formato - controlam o aspecto dos relatórios
- Comandos de arquivo - armazenam informações em arquivos textos.

### • Comandos de ambiente

- **SET *variável-de-sistema valor***

onde

*variável-de-sistema* é a variável que controla um dos aspectos do ambiente.  
*valor* é o valor para a variável-de-sistema.

- **FEEDBACK *n***
  - mostra a quantidade de registros recuperados por uma query quando esta query seleciona pelo menos *n* linhas; o default é 6
- **FEEDBACK ON | OFF**
- **PAUSE *mensagem***
  - mostra uma linha em branco seguida da mensagem e aguarda o usuário pressionar ENTER. Exemplo:
    - SET PAUSE 'Pressionar qualquer tecla para continuar'
- **PAUSE ON | OFF**
- **HEADING ON | OFF**
  - determina se o cabeçalho de cada coluna é exibido no relatório; o default é ON
- **PAGESIZE *n***
  - estabelece o número de linhas por página para *n*; o default é 14.
- **LINESIZE *n***
  - estabelece o número de caracteres por linha para *n*; o default é 80.
- **SPACE *n***
  - estabelece o número de espaços entre colunas para *n*; o default é 10.
- **ECHO ON | OFF**
  - determina se cada comando em um arquivo de texto é listado quando é executado; o default é OFF.
- **VERIFY ON | OFF**
- Ex02\_01 - Mostrar sobrenome e primeiro nome de todos os empregados.
  - SELECT ult\_nome,prim\_nome  
FROM c\_empr;
  - Como o default de FEEDBACK é 6, aparece a mensagem "25 rows selected".
- Ex02\_02 - Mostrar as regiões.
  - SELECT \*

- Criadas e mantidas pelo usuário, tais como C\_EMPR, C\_CLIENTE, C\_DEPTO, que contêm informações do usuário.
- Tabelas do Oracle
  - Criadas e mantidas pelo Oracle, tal como USER\_OBJECTS, que contêm informações sobre o banco de dados Oracle.
  - Exemplos de conteúdos do dicionário de dados
    - Nomes dos usuários do Oracle
    - Privilégios concedidos aos usuários
    - Nomes dos objetos do banco de dados (tabelas, índices, visões, etc.)
    - Restrições das tabelas
    - Informações para auditoria, tais como quem acessou ou atualizou determinados objetos do banco de dados.
- Ex01\_46 - Mostrar a estrutura da visão User\_objects
  - DESC user\_objects
  - O resultado deste comando mostra essa visão é constituída pelas colunas:
    - Object\_name
      - Contém o nome do objeto.
    - Object\_id
    - Object\_type
      - Contém o tipo do objeto, como: table, index, sequence, etc.
    - Created
    - Last\_ddl\_time
    - Timestamp
    - Status
- Ex01\_47 - Mostrar os nomes das tabelas disponíveis, e que comecem com 'C\_'.
  - ```
SELECT object_name
FROM user_objects
WHERE object_type = 'TABLE'
AND object_name LIKE 'C\_%' ESCAPE '\';
```

- Mostrar linhas específicas de uma tabela, em determinada ordenação.

```
• SELECT { * | nome da coluna [, nome da coluna ...] }  
FROM nome da tabela  
WHERE condição {AND | OR} condição  
ORDER BY nome da coluna [ ASC | DESC ]  
      [, nome da coluna [ ASC | DESC] ... ]
```

onde

ASC ordena as linhas de forma ascendente; é a ordenação default.

DESC ordena as linhas de forma descendente.

- Ex01_42 - Mostrar sobrenome, salário e código de departamento dos empregados lotados no departamento 45 em ordem crescente de salário.
 - SELECT ult_nome, salario
FROM c_empr
WHERE id_depto = 45
ORDER BY salario;
- Ex01_43 - Mostrar sobrenome, salário e departamento dos empregados lotados no departamento 45 em ordem decrescente de sobrenome.
 - SELECT ult_nome,salario
FROM c_empr
WHERE id_depto = 45
ORDER BY ult_nome DESC;
- Ex01_44 - Mostrar identificação da região e identificação e nome do departamento, em ordem decrescente de identificação da região e de identificação do departamento.
 - SELECT id_regiao,id,nome
FROM c_depto
ORDER BY id_regiao DESC, id DESC;
- Ex01_45 - Mostrar identificação do departamento, sobrenome e salário dos empregados que são almoxarifes, em ordem crescente de identificação do departamento e decrescente de salário.
 - SELECT id_depto,salario,ult_nome
FROM c_empr
WHERE cargo = 'Almoxarife'
ORDER BY id_depto,salario DESC;
- Recuperar nomes de tabelas, de colunas, de usuários e a estrutura de objetos de banco de dados, através de pesquisas ao dicionário de dados.
 - O dicionário de dados do Oracle contém informações sobre os objetos de banco de dados, usuários e eventos. O acesso a essas informações é feito através de queries às tabelas do dicionário de dados, como a USER-OBJECTS, que mostra os objetos de banco de dados do usuário que abriu a sessão SQL*Plus, no caso o usuário Scott.
 - Tabelas do banco de dados Oracle
 - Tabelas do usuário

WHERE perc_comissao IS NOT NULL;

- Mostrar linhas específicas de uma tabela, usando condições compostas

- **SELECT** { * | *nome da coluna* [, *nome da coluna* ...]}

FROM *nome da tabela*

WHERE *condição* { **AND** | **OR** } *condição*

onde

AND mostra as linhas somente se ambas as condições são verdadeiras.

OR mostra as linhas se pelo menos uma das condições for verdadeira.

- Ex01_36 - Mostrar sobrenome, primeiro nome dos empregados com sobrenome começando com "S" e terminando com "z".
 - **SELECT** ult_nome,prim_nome
FROM c_empr
WHERE ult_nome LIKE 'S%' AND ult_nome LIKE '%z';
- Ex01_37 - Mostrar sobrenome, salário e código do departamento dos empregados lotados no departamento 42 e que recebam pelo menos \$1.200
 - **SELECT** ult_nome, salario,id_depto
FROM c_empr
WHERE id_depto = 42 AND salario >= 1200;
- Ex01_38 - Mostrar sobrenome, salário e cargo dos empregados lotados no departamento 41 e que sejam almoxarifes.
 - **SELECT** ult_nome, salario,cargo
FROM c_empr
WHERE id_depto = 41 AND cargo = 'Almoxarife';
- Ex01_39 - Mostrar sobrenome, salário e cargo dos empregados lotados no departamento 41 ou que sejam almoxarifes.
 - **SELECT** ult_nome, salario,cargo,id_depto
FROM c_empr
WHERE id_depto = 41 OR cargo = 'Almoxarife';
- Ex01_40 - Mostrar sobrenome, salário e código de departamento dos empregados lotados no departamento 44 e que ganham pelo menos \$1.000, e dos empregados lotados no departamento 42.
 - **SELECT** ult_nome, salario,id_depto
FROM c_empr
WHERE (id_depto = 44 AND salario >= 1000) OR
id_depto = 42;
- Ex01_41 - Mostrar sobrenome, salário e código de departamento dos empregados lotados no departamento 42 ou no departamento 44 e que ganham pelo menos \$1.000.
 - **SELECT** ult_nome, salario,id_depto
FROM c_empr
WHERE (id_depto = 42 OR id_depto = 44) AND
salario >= 1000;

```
WHERE dt_admissao LIKE '%91';
```

- Ex01_28 - Mostrar sobrenome e senha dos empregados que tenham a letra "a" na senha.
 - ```
SELECT ult_nome,senha
FROM c_empr
WHERE senha LIKE '%a%';
```
- Ex01\_29 - Mostrar sobrenome, primeiro nome e data de admissão dos empregados admitidos no mês de janeiro, de qualquer ano.
  - ```
SELECT ult_nome,prim_nome,dt_admissao
FROM c_empr
WHERE dt_admissao LIKE '%01%';
```
- Ex01_30 - Mostrar sobrenome, primeiro nome e data de admissão dos empregados admitidos no dia 18, de qualquer mês e ano.
 - ```
SELECT ult_nome,prim_nome,dt_admissao
FROM c_empr
WHERE dt_admissao LIKE '18%';
```
- Ex01\_31 - Mostrar sobrenome, primeiro nome dos empregados com sobrenome começando com "P" e terminando com "s".
  - ```
SELECT ult_nome,prim_nome
FROM c_empr
WHERE ult_nome LIKE 'P%s';
```
- Ex01_32 - Mostrar sobrenome e primeiro nome dos empregados com sobrenome com 6 letras começando com "D" e terminando com "s".
 - ```
SELECT ult_nome,prim_nome
FROM c_empr
WHERE ult_nome LIKE 'D_ _ _ _ s';
```
  - Observação: o caracter "\_" substitui um caracter.
- Ex01\_33 - Mostrar sobrenome e primeiro nome dos empregados com sobrenome cuja segunda letra seja "a"
  - ```
SELECT ult_nome,prim_nome
FROM c_empr
WHERE ult_nome LIKE '_a%';
```
- Ex01_34 - Mostrar identificação, nome e situação de crédito dos clientes que não têm um representante de vendas.
 - ```
SELECT id, nome,sit_cred
FROM c_cliente
WHERE id_repr_vendas IS NULL;
```
- Ex01\_35 - Mostrar sobrenome, cargo e percentual de comissão dos empregados dos empregados comissionados.
  - ```
SELECT ult_nome, cargo,perc_comissao
FROM c_empr
```

- Ex01_20 - Mostrar identificação, sobrenome, salário e data de admissão dos empregados que foram admitidos antes de 31/DEC/90
 - SELECT ident,ult_nome, salario,dt_admissao
FROM c_empr
WHERE dt_admissao < '31/DEC/90';
- Ex01_21 - Mostrar primeiro nome, sobrenome e data de admissão dos empregados admitidos entre 9/MAY/91 e 17/JUN/91.
 - SELECT prim_nome,ult_nome,dt_admissao
FROM c_empr
WHERE dt_admissao BETWEEN '9/MAY/91' AND '17/JUN/91';
- Ex01_22 - Mostrar o primeiro nome, sobrenome e salário dos empregados que tenham salário entre \$ 1.500 e \$ 3.000.
 - SELECT prim_nome,ult_nome,salario
FROM c_empr
WHERE salario BETWEEN 1500 AND 3000;
- Ex01_23 - Mostrar o primeiro nome, sobrenome, salário e data de admissão dos empregados que não foram contratados em 1991.
 - SELECT prim_nome,ult_nome,salario,dt_admissao
FROM c_empr
WHERE dt_admissao NOT BETWEEN '1/JAN/91' AND
'31/DEC/91';
- Ex01_24 - Mostrar sobrenome, cargo e código do departamento dos empregados que sejam gerentes de almoxarifado ou almoxarifes.
 - SELECT ult_nome,cargo,id_depto
FROM c_empr
WHERE cargo IN ('Gerente de Almoxarifado', 'Almoxarife');
- Ex01_25 - Mostrar identificação, nome e código da região dos departamentos que não estão nem na região de código 4 nem na região de código 5.
 - SELECT id,nome,id_regiao
FROM c_depto
WHERE id_regiao NOT IN (4,5);
- Ex01_26 - Mostrar sobrenome e primeiro nome dos empregados que têm sobrenome começando com M.
 - SELECT ult_nome,prim_nome
FROM c_empr
WHERE ult_nome LIKE 'M%';
 - Observação: o caracter "%" substitui qualquer quantidade de caracteres.
- Ex01_27 - Mostrar sobrenome, primeiro nome e data de admissão dos empregados admitidos em 1991.
 - SELECT ult_nome,prim_nome,dt_admissao
FROM c_empr

```
WHERE id_depto = 41;
```

- Ex01_12 - Mostrar primeiro nome e cargo do empregado com sobrenome NICE
 - ```
SELECT prim_nome,cargo
FROM c_empr
WHERE ult_nome = 'NICE';
```
  - Observações:
    - Constantes tipo CHAR devem vir entre aspas simples
    - Não foi selecionada linha alguma porque as constantes tipo CHAR são "case sensitives".
- Ex01\_13 - Mostrar primeiro nome e cargo do empregado com sobrenome Nice
  - ```
SELECT prim_nome,cargo
FROM c_empr
WHERE ult_nome = 'Nice';
```
- Ex01_14 - Mostrar sobrenome e senha dos empregados admitidos em 4/MAR/90
 - ```
SELECT ult_nome,senha
FROM c_empr
WHERE dt_admissao = '4/MAR/90';
```
- Ex01\_15 - Mostrar sobrenome e senha dos empregados admitidos após 4/MAR/90
  - ```
SELECT ult_nome,senha
FROM c_empr
WHERE dt_admissao > '8/MAR/90';
```
- Ex01_16 - Mostrar sobrenome, cargo e salário dos empregados que não sejam almoxarifes
 - ```
SELECT ult_nome, cargo,salario
FROM c_empr
WHERE cargo <> 'Almoxarife';
```
- Ex01\_17 - Mostrar sobrenome, salário,dt\_admissão dos empregados que foram admitidos depois 1/APR/91
  - ```
SELECT ult_nome,salario,dt_admissao
FROM c_empr
WHERE dt_admissao > '1/APR/91';
```
- Ex01_18 - Mostrar sobrenome, cargo e salário dos empregados que ganham mais que \$1.400
 - ```
SELECT ult_nome, cargo,salario
FROM c_empr
WHERE salario > 1400;
```
- Ex01\_19 - Mostrar primeiro nome e sobrenome dos empregados cujo sobrenome vem alfabeticamente depois de "Pires".
  - ```
SELECT ult_nome, prim_nome
FROM c_empr
WHERE ult_nome > 'Pires';
```

- SELECT DISTINCT
- Ex01_06 - Mostrar os nomes de todos os departamentos
 - SELECT nome
FROM c_depto;
 - Os nomes aparecem na ordem em que os registros foram armazenados.
- Ex01_07 - Mostrar os nomes de todos os departamentos, sem repetição
 - SELECT DISTINCT nome
FROM c_depto;
 - Os nomes aparecem em ordem alfabética.
- Ex01_08 - Mostrar os diferentes cargos dos empregados
 - SELECT DISTINCT cargo
FROM c_empr;
- Mostrar o conteúdo de algumas colunas de uma tabela, especificando um cabeçalho para a coluna
 - Ex01_09 - Mostrar os últimos nomes dos empregados , com o cabeçalho Sobrenome
 - SELECT ult_nome Sobrenome
FROM c_empr ;
 - Se o novo cabeçalho contiver brancos, caracteres especiais (por exemplo, #,_, etc.), ele deve vir entre aspas duplas.
 - Ex01_10 - Mostrar os nomes de todos os departamentos, mas com o cabeçalho DEPS DIFERENTES
 - SELECT DISTINCT nome " DEPS DIFERENTES"
FROM c_depto
- Mostrar linhas específicas de uma tabela
 - SELECT { * | nome da coluna [, nome da coluna ...] }
FROM nome da tabela
WHERE condição
onde
 - condição* é constituída de nomes de colunas, expressões, constantes e operadores para comparações.
 - Operadores para comparações:
=, <>, >, >=, <, <=,
BETWEEN ... AND ...
NOT BETWEEN ... AND ...
IN (...)
NOT IN (...)
LIKE
IS NULL
IS NOT NULL
- Ex01_11 - Mostrar primeiro nome e sobrenome dos empregados lotados no departamento cuja identificação é 41
 - SELECT prim_nome,ult_nome
FROM c_empr

- NAME - nome de cada coluna da tabela
- NULL? - se a coluna é de preenchimento obrigatório
 - Not null significa que a coluna é de preenchimento obrigatório
- TYPE - o tipo do dado armazenado na coluna
 - NUMBER (w,d)
 - VARCHAR2(n)
 - DATE
 - CHAR(n)
- Ex01_01: mostrar a estrutura da tabela de Empregado
 - DESC c_empr;
- Ex01_02: mostrar a estrutura da tabela de Cliente
 - DESC c_cliente;
- Mostrar o conteúdo de uma tabela

• **SELECT [DISTINCT] { * | *nome da coluna* [, *nome da coluna* ...]}**
FROM *nome da tabela*
WHERE *condição*
ORDER BY *nome da coluna*

onde

select especifica as colunas, expressões ou constantes que devem ser recuperadas; esta cláusula é obrigatória;

distinct suprime linhas repetidas;

from especifica as tabelas que contêm os dados; esta cláusula é obrigatória;

where especifica os critérios para recuperar as linhas; esta cláusula é opcional;

order by especifica a ordem na qual as linhas recuperadas vão ser mostradas; esta cláusula é opcional;

- Padrões: letras maiúsculas para as palavras chaves
- Ex01_03 - Mostrar o conteúdo da tabela de Região
 - SELECT *
 - FROM c_regiao;
- Ex01_04 - Mostrar o conteúdo da tabela de Departamento
 - SELECT *
 - FROM c_depto;
 - Dados tipo CHAR alinhados à esquerda
 - Dados tipo NUMBER alinhados à direita
- Mostrar o conteúdo de algumas colunas de uma tabela
 - Ex01_05 - Mostrar identificação e nome de cada produto
 - SELECT id,nome
 - FROM c_produto;
- Mostrar o conteúdo de algumas colunas de uma tabela, sem repetição

1 - RECUPERAR DADOS

• OBJETIVOS

- Entrar e sair em uma sessão do SQL*Plus
 - Mostrar a estrutura de uma tabela
 - Recuperar dados de uma tabela
 - Especificar a ordem na qual as linhas são mostradas
 - Recuperar informações do dicionário de dados do Oracle
-
- SQL é uma linguagem para se comunicar com o Oracle Server a partir de qualquer ferramenta ou aplicação.
 - O SQL*Plus é um ambiente de “front end” da Oracle que, através de comandos SQL, possibilita o gerenciamento do banco de dados Oracle.
 - O buffer SQL é uma parte da memória do computador gerenciada pelo SQL*Plus e onde são armazenados os comandos SQL.
 - Um comando SQL permanece no buffer até que um novo comando SQL seja digitado.
 - Para executar um comando SQL digitar ; (ponto e vírgula) e pressionar a tecla ENTER.
 - Para executar o comando armazenado no buffer digitar / (barra) e pressionar a tecla ENTER.
 - Para carregar o SQL*Plus no Windows 95:
 - Clicar no botão INICIAR
 - No menu INICIAR, apontar para PROGRAMAS e clicar em ORACLE FOR WINDOWS 95
 - No menu ORACLE FOR WINDOWS 95 dar um clique em SQL*Plus 3.3
 - Para iniciar uma sessão SQL*Plus:
 - Na caixa USER NAME da janela LOG ON digitar **Scott**
 - Na caixa PASSWORD da janela LOG ON digitar **tiger**
 - Pressionar a tecla ENTER
 - Para terminar uma sessão SQL*Plus:
 - No menu FILE dar um clique duplo em EXIT
 - Carregar o banco de dados da SuperSports, usando o SQL*Plus
 - Colocar o disquete no drive A
 - Digitar o comando @ **a:carga_ss.sql**
 - Pressionar a tecla / (barra)
 - Uso do Bloco de Notas
 - Mostrar a estrutura de uma tabela usando o comando DESCRIBE
 - DESC nome da tabela
 - O objetivo deste comando é mostrar a estrutura de uma tabela

5 - MOSTRAR DADOS DE MÚLTIPLAS TABELAS	52
• Recuperar dados a partir de várias tabelas	
• Combinar dados de tabelas relacionadas onde algumas linhas de uma das tabelas não têm correspondência direta com linhas da outra tabela.	
• Fazer uma junção de uma tabela com ela própria	
• Criar uma visão de tabelas múltiplas.	
6 - UTILIZAR SUBQUERIES	56
• Passar uma linha de dados recuperada por uma subquery para a query principal.	
• Passar várias linhas de dados recuperadas por uma subquery para a query principal.	
• Encadear múltiplas subqueries dentro da query principal.	
7 - EXECUTAR OPERAÇÕES COM DADOS	60
• Executar cálculos com números usando operadores aritméticos e funções.	
• Executar cálculos com datas usando operadores aritméticos e funções.	
• Reformatar números e datas.	
• Manipular strings de caracteres e usá-los com funções.	
• Executar cálculos sumarizados com grupos de linhas usando funções de grupo.	
ANEXOS	
• Estrutura das tabelas	70
• Diagrama de estrutura de dados	73

ÍNDICE

1 - RECUPERAR DADOS	4
• Entrar e sair em uma sessão do SQL*Plus	
• Mostrar a estrutura de uma tabela	
• Recuperar dados de uma tabela	
• Especificar a ordem na qual as linhas são mostradas	
• Recuperar informações do dicionário de dados do Oracle	
2 - CONSTRUIR RELATÓRIOS EMERGENCIAIS USANDO O SQL*Plus	14
• Controlar o ambiente de geração de relatórios de um sessão do SQL*Plus.	
• Formatar um relatório.	
• Salvar os comandos para gerar um relatório.	
• Salvar o relatório em um arquivo texto.	
• Criar um relatório simples usando comandos SQL*Plus	
• Comandos de ambiente - afetam o comportamento geral da sessão.	
• Comandos de formato - controlam o aspecto dos relatórios	
• Comandos de arquivo - armazenam informações em arquivos textos.	
3 - MANIPULAÇÃO DE DADOS	29
• Adicionar novas linhas a uma tabela	
• Inserir um valor nulo numa coluna de uma nova linha	
• Gerar automaticamente valores de chave primária para as novas linhas	
• Modificar linhas já existentes em uma tabela	
• Modificar várias linhas de uma tabela usando a cláusula WHERE para identificar um conjunto de linhas.	
• Modificar várias linhas de uma tabela omitindo a cláusula WHERE.	
• Modificar todas as linhas de uma tabela omitindo a cláusula WHERE.	
• Remover linhas existentes em uma tabela	
• Controlar a lógica das transações usando os comandos COMMIT e ROLLBACK	
• Alterar a lógica das transações com savepoints.	
• Criar um arquivo texto para carregar uma grande quantidade de dados de uma vez só.	
• Criar um arquivo texto para carregar dados de forma interativa, embutindo parâmetros de substituição.	
• Executar o mesmo comando SQL repetidamente com valores diferentes	
4 - CRIAR TABELAS E ESTRUTURAS DE DADOS	38
• Criar tabelas contendo restrições para integridade	
• Descrever os tipos de dados que podem ser usados ao especificar as definições das colunas	
• Reconhecer os índices que são criados automaticamente pelas restrições	
• Criar uma tabela e preenchê-la com as linhas de outra tabela	
• Modificar a estrutura de uma tabela existente	
• Reforçar a integridade dos dados da empresa ao adicionar e remover restrições à definição de uma tabela	
• Criar visões lógicas de uma tabela	
• Gerar valores de chave primária usando seqüências	
• Determinar quando aumentar a performance de algumas queries através da criação de índices.	

INTRODUÇÃO

A parte prática da cadeira Laboratório de Banco de Dados é constituída de uma introdução à linguagem SQL, voltada para o ambiente Oracle. Os comandos SQL serão executados usando um banco de dados previamente criado e carregado com dados. Esse banco de dados refere-se a uma empresa hipotética, cuja descrição é apresentada a seguir. Deverão ser elaborados o Diagrama de Entidade e Relacionamento e o Diagrama de Estrutura de Dados dessa empresa, para o completo entendimento do uso do Oracle.

DESCRIÇÃO SUCINTA DA EMPRESA HIPOTÉTICA

A SuperSports é uma empresa multinacional distribuidora de artigos esportivos. Seus clientes são grandes firmas varejistas localizadas em diversos países.

A área de atuação da SuperSports é mundial, estando no momento dividida em cinco regiões.

Atualmente existem cinco almoxarifados, um em cada região; em um futuro próximo poderão ser criados outros almoxarifados em cada região. Um almoxarifado armazena cada um dos produtos da linha de distribuição da SuperSports.

As regiões têm vários clientes, cada um vinculado a uma só região. As vendas aos clientes são efetivadas através de faturas. Uma fatura abrange vários produtos.

Na empresa existem cinco cargos: presidente, vice-presidente, gerente de almoxarifado, representante de vendas e almoxarife. Os recursos humanos da SuperSports são constituídos de um presidente, quatro vice-presidentes (administrativo, financeiro, operações e vendas), alguns gerentes de almoxarifado, alguns representantes de vendas e vários almoxarifes. Um empregado só desempenha um cargo. Existe uma hierarquia entre os empregados, a saber: os vice-presidentes são subordinados ao presidente; os gerentes de almoxarifado ao vice-presidente de operações, os representantes de vendas ao vice-presidente de vendas e os almoxarifes ao gerente do almoxarifado da região.

A SuperSports possui 12 departamentos: um financeiro, localizado na região 1, cinco de vendas, um em cada região, cinco de operações, um em cada região, e um administrativo, localizado na região 1.

Cada empregado está lotado em um só departamento. Os representantes de vendas têm na sua “conta” vários clientes e um cliente está vinculado a um só representante de vendas.