**CAPÍTULO 1 – Introdução**

**1. CRIANDO TABELA**

CREATE TABLE [nome\_da\_tabela] ([nome\_do\_campo] [tipo] [tamanho] NOT NULL ,PRIMARY KEY[nome\_do\_campo]);

**Exemplo:**

***CREATE TABLE*** *TALUNO*

*(*

*COD\_ALUNO* ***INTEGER NOT NULL****,*

*NOME* ***VARCHAR****(30),*

*CIDADE* ***VARCHAR2****(30),*

*CEP* ***VARCHAR****(10),*

***PRIMARY******KEY*** *(COD\_ALUNO)*

*);*

ou:

CREATE TABLE [nome\_da\_tabela] ([nome\_do\_campo] NOT NULL PRIMARY KEY [tipo] [tamanho]);

Exemplo:

***CREATE TABLE*** *TCURSO(*

*COD\_CURSO I****NTEGER NOT NULL PRIMARY KEY****,*

*NOME* ***VARCHAR****(30),*

*VALOR* ***NUMBER****(8,2),*

*CARGA\_HORARIA* ***INTEGER***

*);*

2. INSERINDO DADOS NA TABELA

INSERT INTO [NOME\_DA\_TABELA]([campo1],[campo2],[campo3]) VALUES('dado1','dado2',dado3);

Exemplo:

***INSERT INTO*** *TALUNO(COD\_ALUNO,NOME,CIDADE,CEP)*

***VALUES*** *(1,'EMERSON','NOVO HAMBURGO','93004900');*

ou:

INSERT INTO NOME\_TABELA VALUES('dado1','dado2',dado3);

Exemplo:

***INSERT INTO*** *TCURSO* ***VALUES****(1,'ORACLE SQL e PL/SQL',500,25);*

2.1 Inserindo valor padrão em uma coluna:

***ALTER TABLE*** *TALUNO* ***ADD*** *ESTADO* ***CHAR(2) DEFAULT*** *'RS'****;***

A coluna será inserida e preenchida com o valor específico padrão.

3. INSERINDO COLUNA, DROPANDO E RENOMEANDO COLUNA (ALTER TABLE)

3.1. Inserindo Coluna:

ALTER TABLE [nome\_da\_table] ADD [nomde\_da\_coluna][atributo1][atributo2];

Exemplo:

*ALTER TABLE TCONTRATO ADD COL\_TESTE VARCHAR(50);*

3.2. Dropando Coluna:

ALTER TABLE [nome\_da\_tabela] DROP COLUMN [nome\_da\_coluna];

Exemplo:

*ALTER TABLE TCONTRATO DROP COLUMN COL\_TESTE;*

3.3. Renomeando Coluna:

ALTER TABLE [nome\_da\_tabela] RENAME COLUMN [nome\_da\_coluna] TO [novo\_nome\_col];

Exemplo:

*ALTER TABLE TCONTRATO RENAME COLUMN COL\_TESTE TO COLUNA\_TESTE;*

3.4. Renomeando Tabela:

ALTER TABLE [nome\_da\_tabela] RENAME TO [novo\_nome\_da\_tabela];

*ALTER TABLE TESTADI RENAME TO TESTADO;*

4. SEQUENCE

Usado para criar uma sequencia de valores a serem inseridos automaticamente a partir do valor específico.

CREATE SEQUENCE SEQ\_[nome\_tabela] START WITH [valor]

Exemplo:

*CREATE SEQUENCE SEQ\_ALUNO START WITH 4;*

4.1. INSERINDO O VALOR COM O SEQUENCE CRIADO:

INSERT INTO [nome\_tabela] VALUE([nome\_sequence].NEXTVAL,[valores]);

Exemplo:

*INSERT INTO TALUNO(COD\_ALUNO,NOME,CIDADE,CEP)*

*VALUES (SEQ\_ALUNO.NEXTVAL,’EMILLY’,’IVOTI’,’599388388’);*

**5. SETANDO DATA ATUAL E VALOR ATUAL DO SEQUENCIA**

Data/Hora Atual

*SELECT SYSDATE FROM DUAL;*

Valor Atual da Sequencia

*SELECT SEQ\_ALUNO.CURRVAL FROM DUAL;*

6. EXCLUIR TABELAS, DESFAZER TUDO, CONFIRMAR TUDO

Excluir Tabela

*DROP TABLE [nome\_da\_tabela];*

Desfazer todas as ações ou comandos executados

*ROLLBACK;*

Confirmar todos os comandos executados

*COMMIT;*

7. EXCLUIR UM E TODOS OS REGISTROS DA TABELA

7.1. Excluir um registro:

*DELETE FROM TALUNO*

*WHERE COD\_ALUNO = 1;*

7.1. Excluir todos os registros:

*DELETE FROM TALUNO;*

8. SELECIONANDO POR VALOR ESPECÍFICO

*SELECT \* FROM TALUNO WHERE NOME = 'EMERSON';*

9. ALTERAR REGISTRO DA TABLEA

Exemplo:

Alterando um valor:

***UPDATE*** *TCONTRATO* ***SET*** *DESCONTO = 5* ***WHERE*** *COD\_CONTRATO = 1;*

ou

Alterando vários valores:

***UPDATE*** *TCONTRATO* ***SET*** *DESCONTO = 20****,*** *TOTAL = 750* ***WHERE*** *COD\_CONTRATO = 2****;***

Alterando valor com especificação maiúscula ou minúscula:

***UPDATE*** *TALUNO* ***SET***

*CIDADE = 'CANOAS'*

***WHERE Upper****(CIDADE) = 'BULGARIA'****;***

**CAPÍTULO 2 – EXECUTANDO COMANDOS SQL BÁSICOS**

SELECT [DISTINCT] {\*,column [alias], ...}

FROM table;

SELECT - é uma lista de uma ou mais colunas.

DISTINCT – suprime duplicidades.

\* - seleciona todas as colunas.

column – seleciona a coluna nomeada.

alias – fornece para as colunas selecionadas títulos diferentes.

FROM table – especifica a tabela que contém colunas.

1. Escrevendo comandos SQL

- **Comandos** SQL não fazem distinção entre maiúscula e minúscula.

- Comandos SQL podem estar em uma ou mais linhas.

- **Palavras chaves** (keywords) não podem ser abreviadas ou divididas em mais de uma linha.

- **Cláusulas** são normalmente colocadas em linhas separadas.

- **Tabulações** e identações são utilizadas para melhorar a visualização do comando.

Exemplos:

- Trazer todos os dados da tabela:

***SELECT*** *\** ***FROM*** *TALUNO****;***

- Trazer somente os dados de uma coluna:

***SELECT*** *COD\_ALUNO* ***FROM*** *TALUNO****;***

- Trazer dados de mais de uma coluna:

***SELECT*** *COD\_ALUNO, NOME* ***FROM*** *TALUNO****;***

1. Padrões de Cabeçalho de Colunas

Alinhamento padrão:

- Esquerda: Dados tipo caracteres e data.

- Direita: Dados numéricos

Exibição padrão nome coluna: Maiúsculas (se quiser minúscula, tem que forçar).

1. Expressões aritméticas

- Crie expressões em dados tipo NUMBER e DATE utilizando operadores aritméticos.

|  |  |
| --- | --- |
| Operador | Descrição |
| + | Soma |
| - | Subtração |
| \* | Multiplicação |
| / | Divisão |

1. Definindo um valor Nulo

- Um nulo é um valor que é indispensável, não atribuído, desconhecido ou inaplicável.

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

1. Definindo um Alias de Coluna

- Altera o cabeçalho de uma coluna.

- É útil em cálculos.

- Segue imediatamente o nome da coluna: palavra chave AS entre o nome da coluna e o alias é opcional.

- Requer aspas duplas se ele possui espaços ou caracteres especiais ou deve diferenciar maiúsculas de minúsculas.

Exemplo:

Utilizando Alias de Coluna

***SELECT*** *cod\_aluno codigo****,*** *nome* ***AS*** *aluno* ***FROM*** *taluno;*

CODIGO ALUNO

...

***SELECT*** *nome “Aluno”* ***FROM*** *taluno;*

Aluno preferencial

...

1. Operador de concatenação

- Concatena colunas ou strings de caracteres a outras colunas

- É representado através de duas barras verticais (||)

- Cria uma coluna resultante que é uma expressão caractere.

Exemplo:

***SELECT*** *cod\_aluno* ***||*** *nome* ***AS*** *“Aluno”* ***FROM*** *taluno;*

1. Strings de caracteres Literais

- Um literal é qualquer caractere, expressão ou números incluídos na lista da cláusula SELECT.

- Valores literais do tipo data e caractere devem ser colocados entre aspas simples.

- Cada string de caractere é exibida uma vez para cada linha retornada.

Exemplo:

***SELECT*** *nome* ***|| ‘*** *nasceu em* ***‘ ||*** *nascimento* ***AS “****Nascimento do Aluno****” FROM*** *taluno****;***

1. Linhas Duplicadas

E exibição padrão de consultas, lista todas as linhas, incluindo linhas duplicadas.

***SELECT*** *cidade**CIDADE* ***FROM*** *taluno****;***

CIDADE

São Paulo

Rio de Janeiro

Novo Hamburgo

Novo Hamburgo

São Paulo

São Paulo

Novo Hamburgo

1. Eliminando linhas duplicadas

Elimine linhas duplicadas utilizando a palavra chave DISTINCT na cláusula SELECT.

***SELECT DISTINCT*** *cidade**CIDADE* ***FROM*** *taluno****;***

CIDADE

São Paulo

Rio de Janeiro

Novo Hamburgo

1. Pesquisa por Ordem

Exemplo:

***SELECT*** *NOME* ***AS***

*CURSO,*

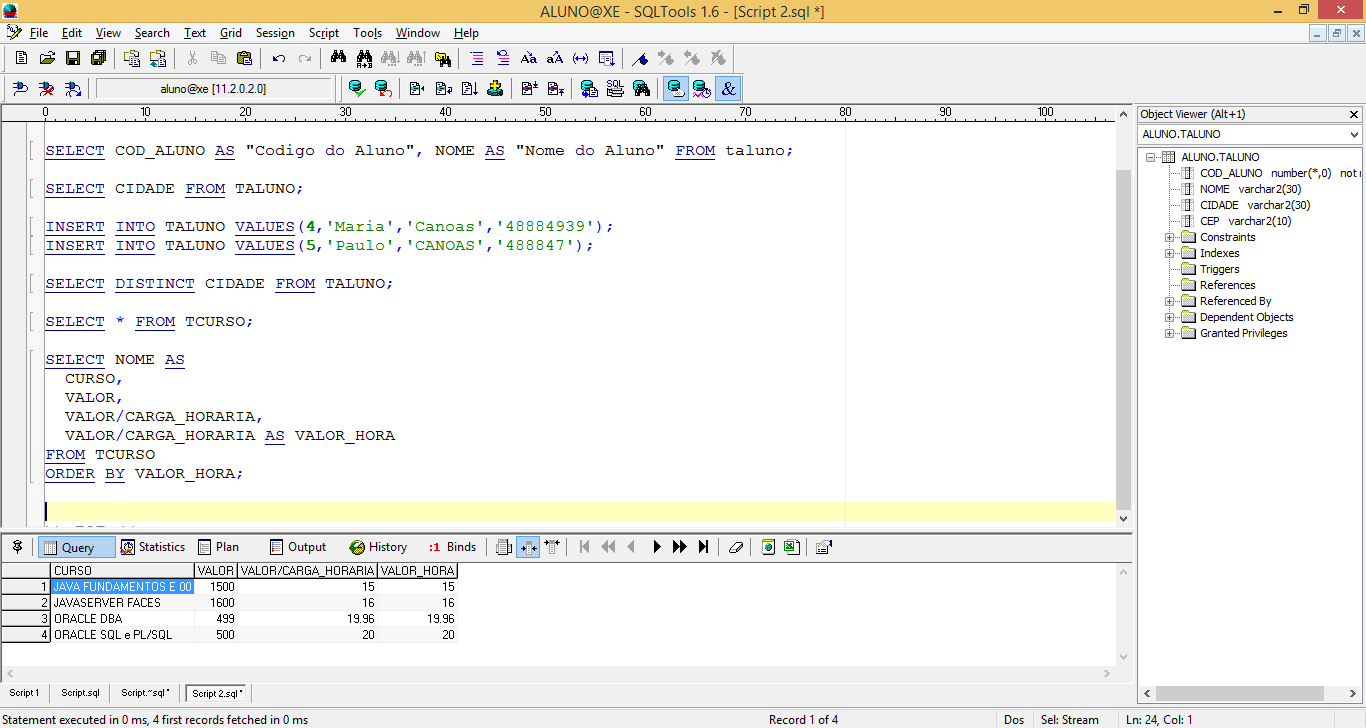
*VALOR,*

*VALOR/CARGA\_HORARIA,*

*VALOR/CARGA\_HORARIA* ***AS*** *VALOR\_HORA*

***FROM*** *TCURSO*

***ORDER BY*** *VALOR\_HORA****;***



1. Arredondado o Valor para casas decimais específicas

Duas casas decimais:

***Round(****VALOR/CARGA\_HORARIA,****2) AS*** *VALOR\_HORA*

Sem casas decimais:

***Round(****VALOR/CARGA\_HORARIA****) AS*** *VALOR\_HORA*

1. Trabalhando com valores nulos

***SELECT*** *COD\_CONTRATO****,***

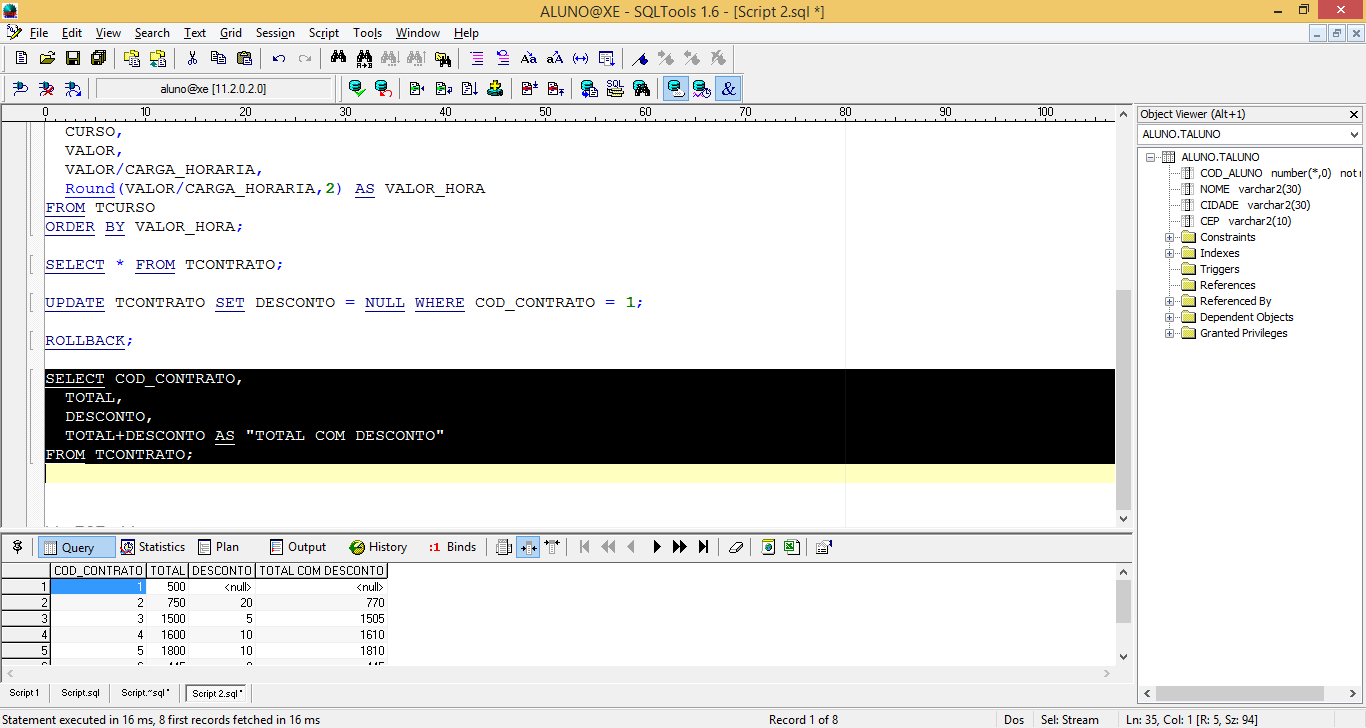
*TOTAL,*

*DESCONTO,*

*TOTAL+DESCONTO* ***AS*** *"TOTAL COM DESCONTO"*

***FROM*** *TCONTRATO****;***

O resultado abaixo mostra que a soma de valores ‘null’ retorna null.



Para tratar esse tipo de resultado, usamos o NVL no campo que desejamos que retorne o valor 0:

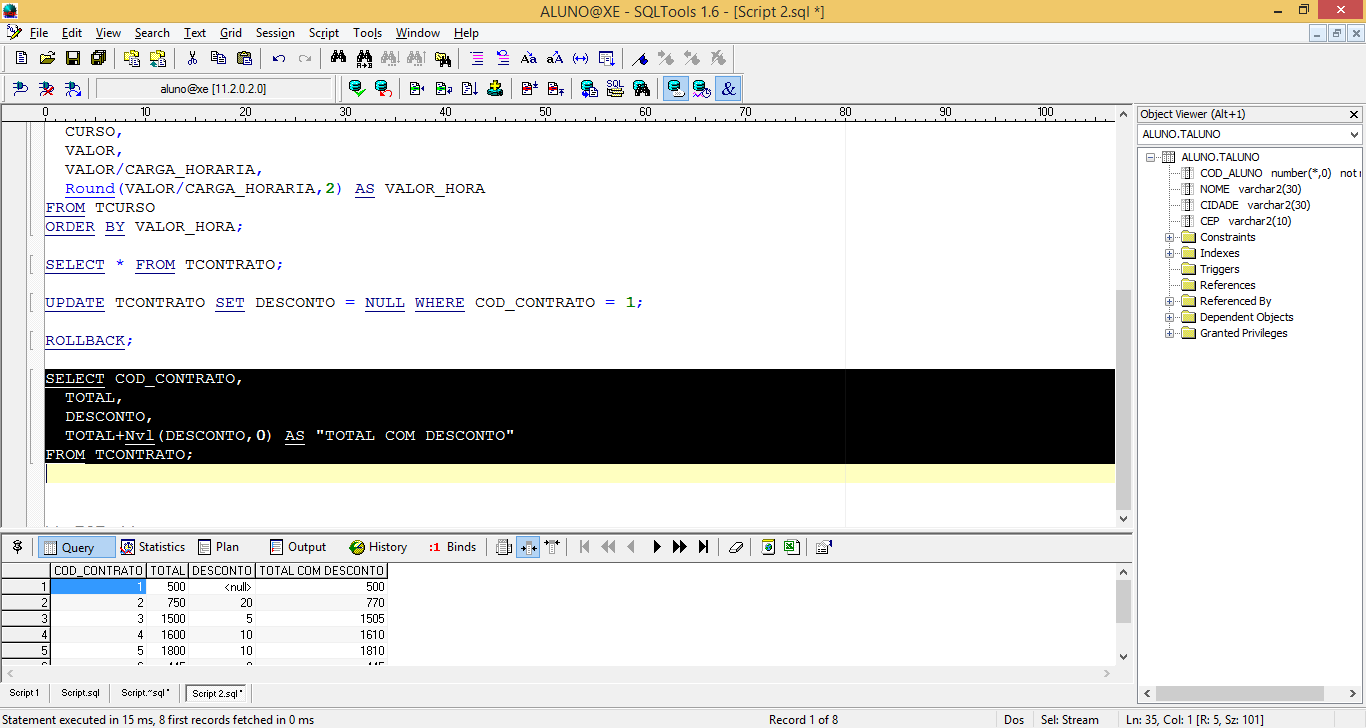
***SELECT*** *COD\_CONTRATO****,***

*TOTAL,*

*DESCONTO,*

*TOTAL****+Nvl(****DESCONTO****,0) AS*** *"TOTAL COM DESCONTO"*

***FROM*** *TCONTRATO****;***



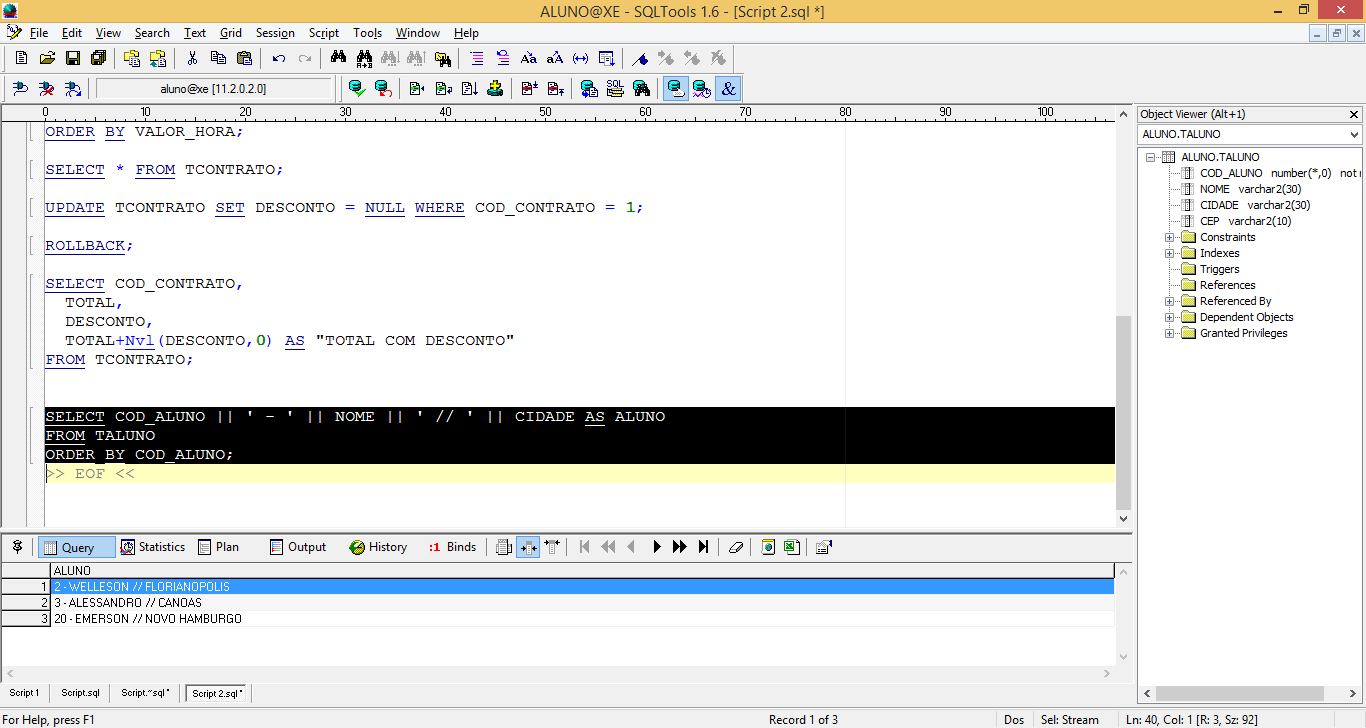
1. Exemplo de pesquisa usando Concatenação

***SELECT*** *COD\_ALUNO* ***||*** *' - '* ***||*** *NOME* ***||*** *' // '* ***||*** *CIDADE* ***AS*** *ALUNO*

***FROM*** *TALUNO*

***ORDER BY*** *COD\_ALUNO****;***

O resultado será:



**CAPÍTULO 3 – Restringindo e Ordenando dados**

- Limitar as linhas recuperadas por uma consulta  
- Ordenar as linhas recuperadas por uma consulta

3.1 Limitando as Linhas Selecionadas

***SELECT [DISTINCT]*** *{\*, column [alias], ...}*

***FROM*** *table*

***[WHERE*** *condition(s)****];***

Na sintaxe:

WHERE - restringe a consulta para as linhas que satisfazem a condição.

condition – é composta de nomes de coluna expressões, constantes, e operadores de comparação.

Exemplo:

***SELECT*** *cod\_aluno, nome*

***FROM*** *taluno*

***WHERE*** *estado = ‘RS’****;***

Strings de caractere e valores tipo data são inclusos entre aspas simples.

Valores caractere fazem distinção entre maiúsculas e minúsculas e valores tipo data são sensíveis ao formato.

3.2 Operadores de Comparação

|  |  |
| --- | --- |
| Operador | Significado |
| = | Igual a |
| > | Maior que |
| >= | Maior que ou igual a |
| < | Menor que |
| <= | Menor que ou igual a |
| <> ou != | Diferente de |
|  | |
| *...WHERE exp operador value* | |

Exemplo:

***SELEC*** *cod\_contrato, data, total, desconto + 10*

***FROM*** *tcontrato*

***WHERE*** *total <= desconto + 10****;***

3.3 Outros Operadores de Comparação

|  |  |
| --- | --- |
| Operador | Significado |
| BETWEEN  ...AND... | Entre dois valores (inclusive) |
| IN(list) | Igual a um dos valores da lista |
| LIKE | Igual a um padrão de caracteres |
| IS NULL | Possui um valor nulo |

Exemplo:

1. Selecionar todos os alunos de estados diferentes e que recebem igual ou menor o salário de 800 e ordenar em ordem decrescente:

***SELECT*** *\** ***FROM*** *TALUNO*

***WHERE*** *ESTADO* ***<>*** *'RS'*

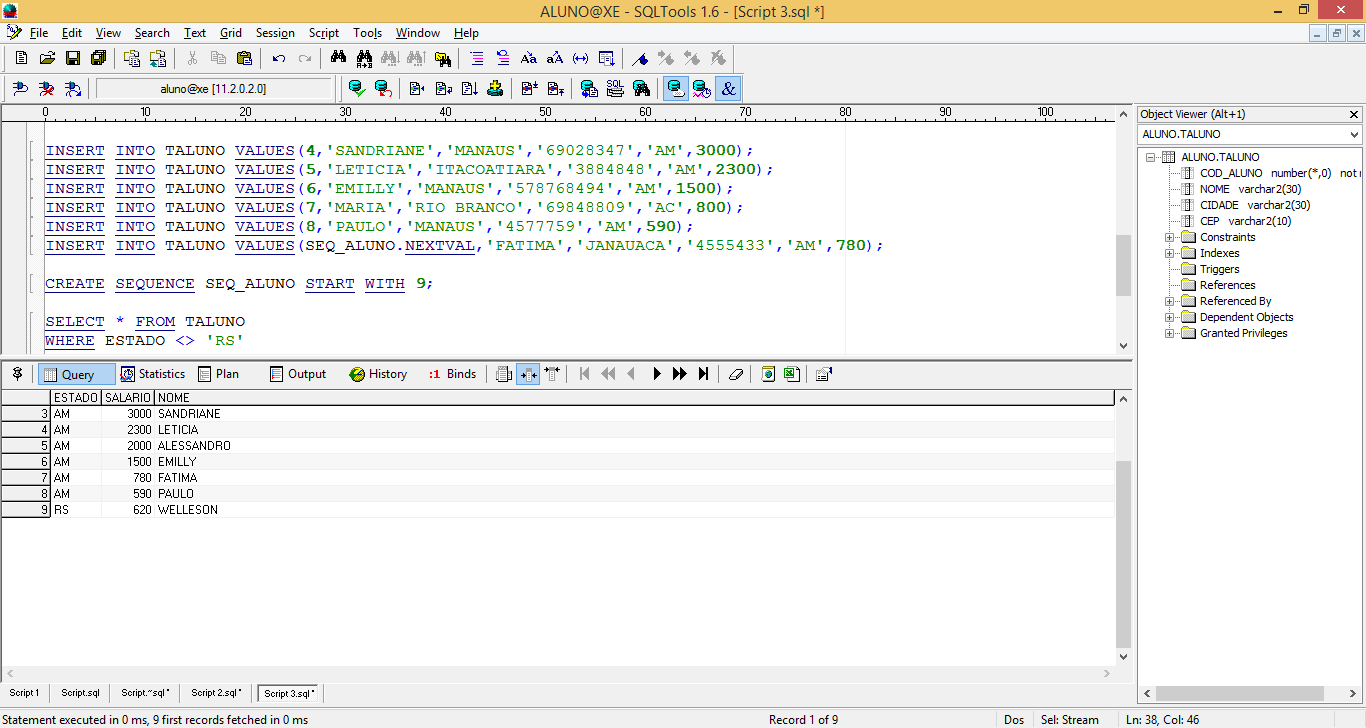
***AND*** *SALARIO* ***<=*** *800*

***ORDER BY*** *SALARIO* ***DESC;***

1. Ordem alfabética segue um critério de organização por grupo. Por exemplo, vamos listar em ordem alfabética todos os estados e em cada grupo de estado, organizar em ordem decrescente os salários:

***SELECT*** *ESTADO, SALARIO, NOME* ***FROM*** *TALUNO*

***ORDER BY*** *ESTADO, SALARIO* ***DESC;***



1. Filtrando valores com data/hora

Trunc = ignora as horas:

***SELECT*** *COD\_ALUNO, NASCIMENTO,* ***Trunc(****NASCIMENTO****),*** *NOME*

***FROM*** *TALUNO*

***WHERE Trunc(****NASCIMENTO****) =*** *'05/08/2015'****;***

1. Selecionando Datas com Between

***SELECT*** *COD\_ALUNO, NASCIMENTO,* ***Trunc(****NASCIMENTO****),*** *NOME*

***FROM*** *TALUNO*

***WHERE*** *NASCIMENTO*

***BETWEEN To\_Date(****'05/08/2015 14:33'****,'DD/MM/YYYY HH24:MI')***

***AND To\_Date(****'05/08/2015 23:33'****,'DD/MM/YYYY HH24:MI');***

1. Selecionar contratos que calcule + 1000 e listar TOTAIS que sejam menor ou igual ao desconto com + 1000;

***SELECT*** *COD\_CONTRATO, DATA, TOTAL, DESCONTO, DESCONTO* ***+ 1000 AS*** *CALCULO*

***FROM*** *TCONTRATO*

***WHERE TOTAL <=*** *DESCONTO* ***+ 1000;***

1. Listando valores NULL e diferentes de NULL e Intervalos de valores

6.1. Valores Not Null

***SELECT*** *DESCONTO*

***FROM*** *TCONTRATO*

***WHERE*** *DESCONTO* ***IS NOT NULL;***

6.2. Valores Null

***SELECT*** *DESCONTO*

***FROM*** *TCONTRATO*

***WHERE*** *DESCONTO* ***IS NULL;***

6.3. Intervalo de Valores

***SELECT*** *DESCONTO*

***FROM*** *TCONTRATO*

***WHERE*** *DESCONTO* ***BETWEEN 0 AND 10;***

1. Considerando o valor Null

***SELECT*** *DESCONTO****, Nvl(****DESCONTO****,0)***

***FROM*** *TCONTRATO*

***WHERE Nvl(****DESCONTO****,0) BETWEEN*** *0* ***AND*** *10****;***

1. Between e equivalente:

8.1 Selecionar descontos entre 0 e 10:

***SELECT*** *\** ***FROM*** *TCONTRATO*

***WHERE*** *DESCONTO* ***>=*** *0*

***AND*** *DESCONTO* ***<=*** *10****;***

Ou:

***SELECT*** *\** ***FROM*** *TCONTRATO*

***WHERE*** *DESCONTO* ***BETWEEN*** *0* ***AND*** *10****;***

8.2 Selecionar descontos entre 0 e 10 e Null

***SELECT*** *COD\_CONTRATO, TOTAL, DESCONTO,* ***Nvl(****DESCONTO****,0)***

***FROM*** *TCONTRATO*

***WHERE Nvl(****DESCONTO****,0) BETWEEN*** *0* ***AND*** *10****;***

Ou:

***SELECT*** *\** ***FROM*** *TCONTRATO*

***WHERE*** *DESCONTO* ***>=*** *0*

***AND*** *DESCONTO* ***<=*** *10*

***OR*** *DESCONTO* ***=*** *NULL****;***

* 1. Selecionar registros que contenham apenas os valores específicos

***SELECT*** *\** ***FROM*** *TCURSO*

***WHERE*** *COD\_CURSO* ***IN (****1,2,4****);***

* 1. Selecionar apenas os cursos não vendidos

***SELECT*** *\** ***FROM*** *TCURSO*

***WHERE*** *COD\_CURSO* ***NOT IN (SELECT*** *COD\_CURSO* ***FROM*** *TITEM****);***

8.5 Selecionar apenas os cursos vendidos

***SELECT*** *\** ***FROM*** *TCURSO*

***WHERE*** *COD\_CURSO* ***IN (SELECT*** *COD\_CURSO* ***FROM*** *TITEM****);***

* 1. Pesquisa por string

- Selecionar os cursos que começam com O:

***SELECT*** *\** ***FROM*** *TCURSO* ***WHERE*** *NOME* ***LIKE '****O%****';***

- Selecionar os cursos que terminam com O:

***SELECT*** *\** ***FROM*** *TCURSO* ***WHERE*** *NOME* ***LIKE '%****O****';***

- Selecionar os cursos que contenham O em qualquer parte do nome:

***SELECT*** *\** ***FROM*** *TCURSO* ***WHERE*** *NOME* ***LIKE '%****O****';***

* 1. Precedência de Operadores

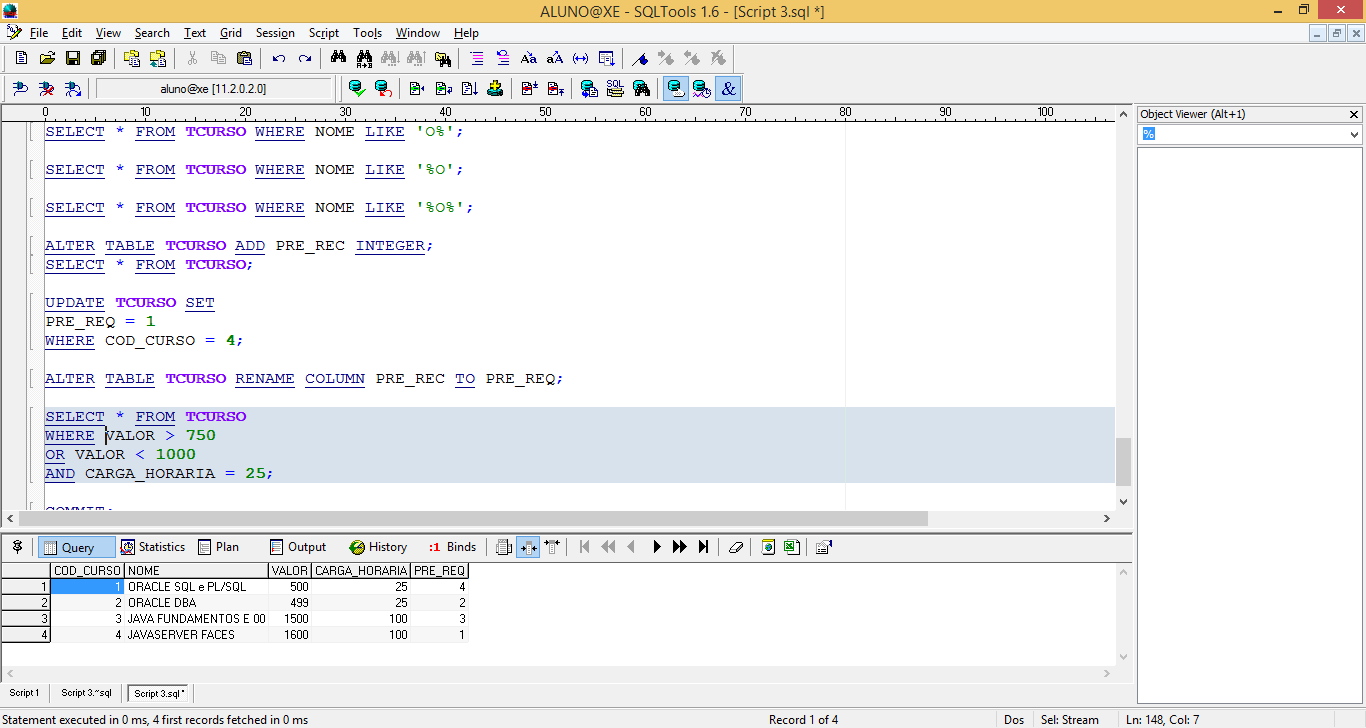
Primeiro será executado o AND, em seguida o OR, antes os parênteses:

***SELECT*** *\** ***FROM*** *TCURSO*

***WHERE*** *VALOR* ***>*** *750*

***OR*** *VALOR* ***<*** *1000*

***AND*** *CARGA\_HORARIA* ***=*** *25****;***

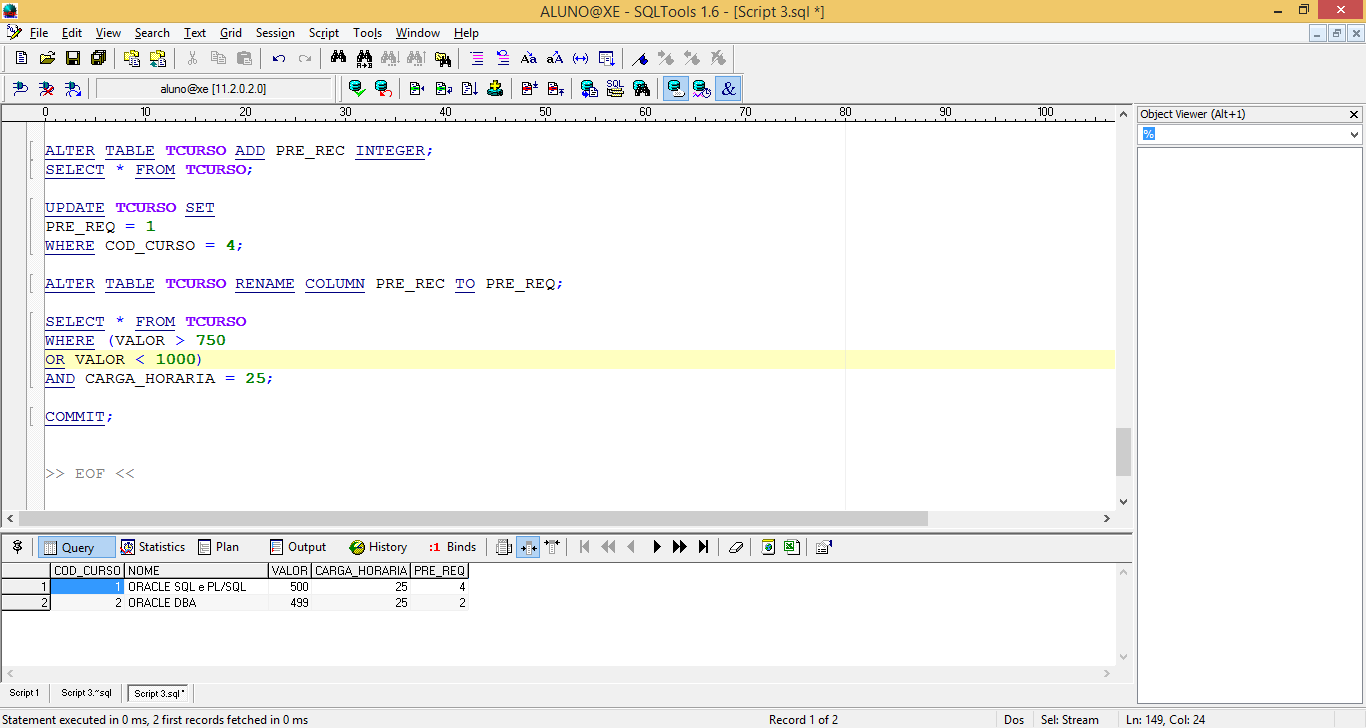


***SELECT*** *\** ***FROM*** *TCURSO*

***WHERE*** *(VALOR* ***>*** *750*

***OR*** *VALOR* ***<*** *1000)*

***AND*** *CARGA\_HORARIA* ***=*** *25****;***



**CAPÍTULO 4 – Introdução as Funcões Básicas**

- Descrever os vários tipos de funções disponíveis em SQL.

- Utilizar funções do tipo caractere, numéricas e de datas em comandos SELECT.

- Descrever o uso de funções de conversão.

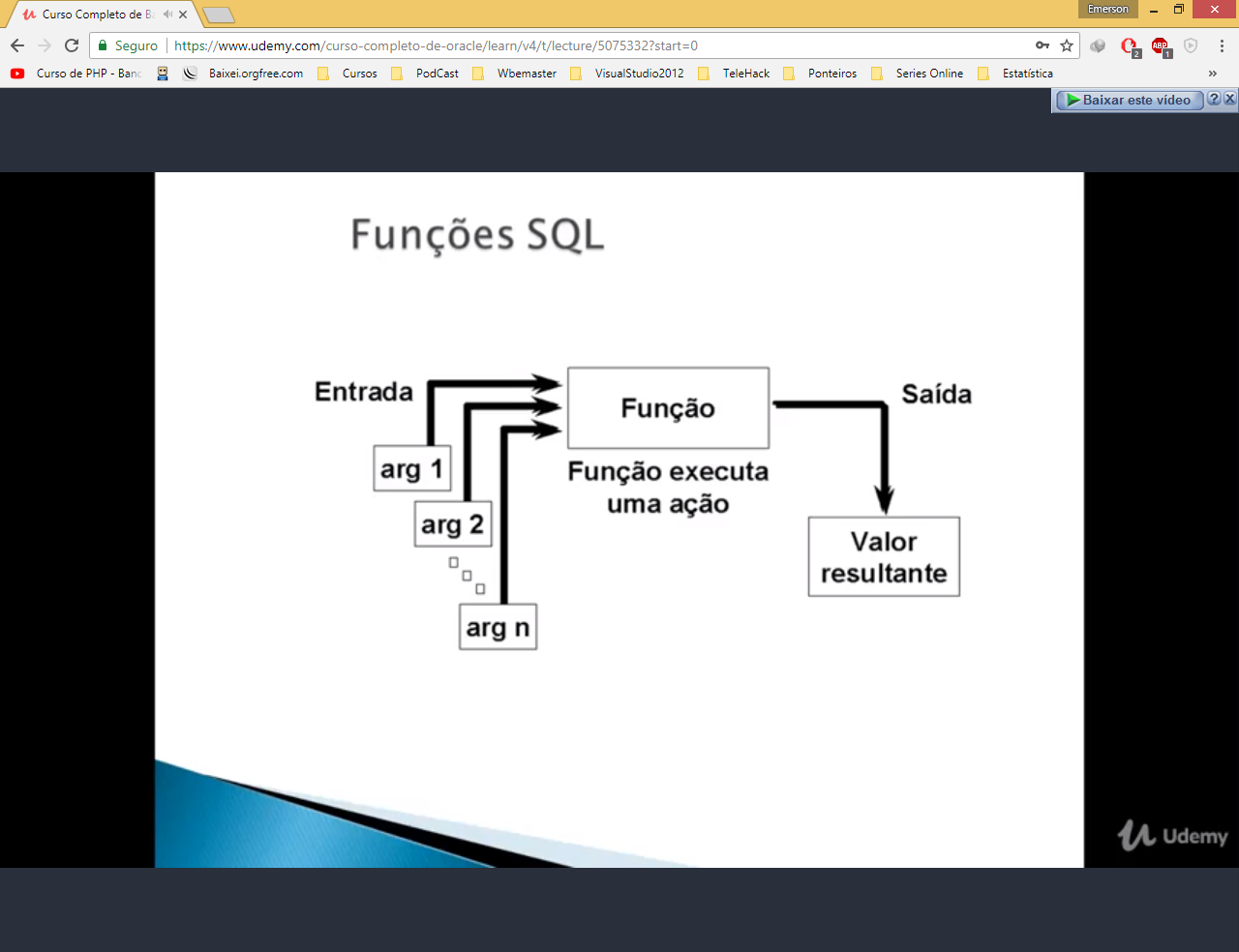


Figura 1 - Função de entrada do tipo single row.

Tipos de Funções:

**SINGLE ROW** – Estas funções operam em linhas únicas retornando um resultado para cada linha processada.

**MULTIPLE-ROW** – Estas funções manipulam grupos de linhas para obter um resultado para cada grupo processado.

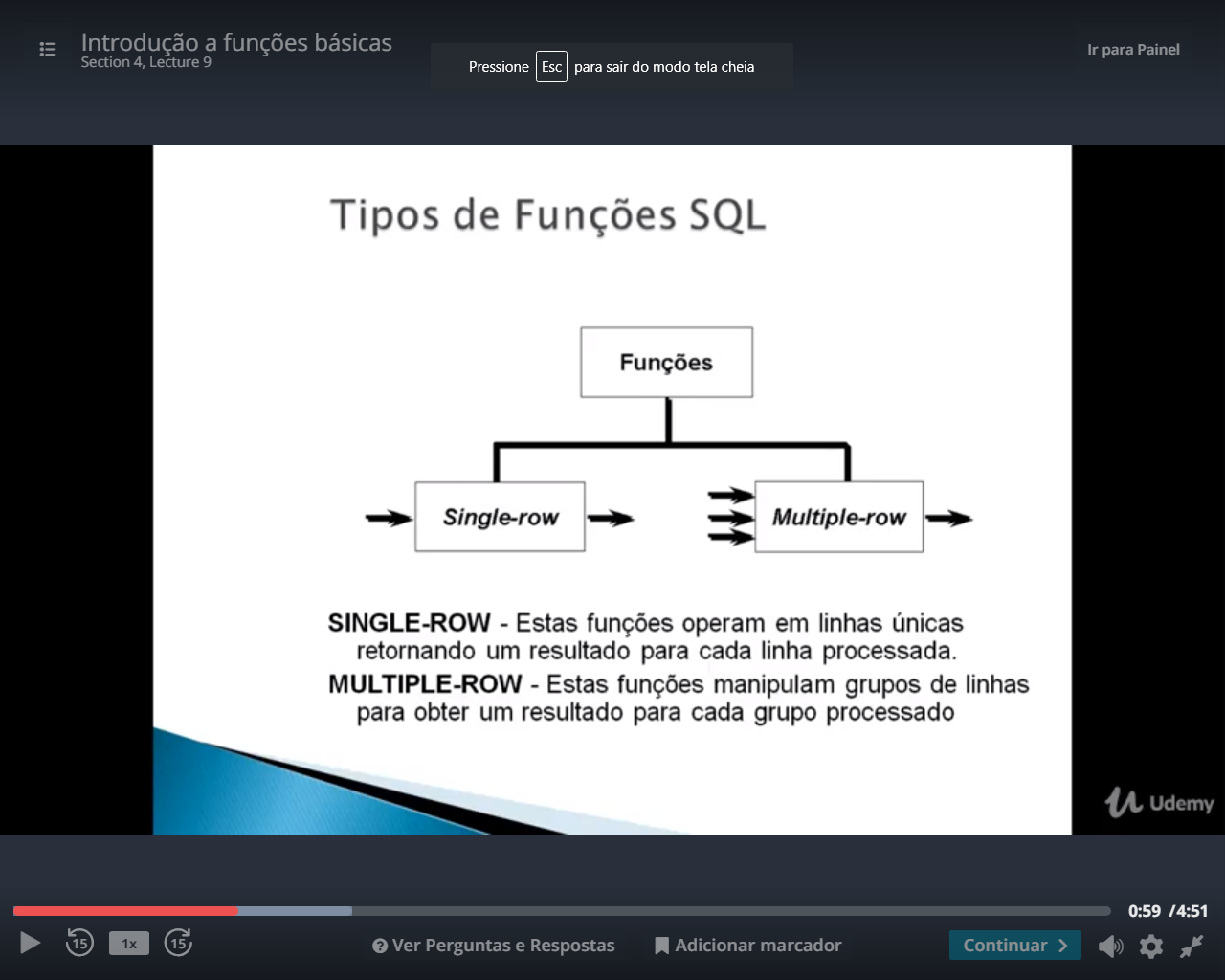


Figura 2 - Tipos de funções.

4.1 **Funções do tipo Single-Row:**

- Manipulam itens de dados.

- Recebem argumentos e retornam um valor.

- Atuam sobre cada linha recuperada.

- Retornam um resultado por linha.

- Podem modificar o tipo de um lado.

- Podem ser aninhadas.

function\_name (column | expression, [arg1, arg2, ...])

4.1 **Funções de Caracteres**

|  |  |
| --- | --- |
| **Função** | **Propósito** |
| CONCAT(column1|expression1,column2|expression2) | Concatena a primeira string de caracateres com a segunda. Equivalente ao operador de concatenação (||). |

|  |  |
| --- | --- |
| INITCAP (column|expression) | Converte strings de caracteres deixando a primeira letra de cada palavra em maiúscula e as demais em minúsculas. |
| INSTR (column|expression,m) | Retorna a posição numérica do caracter dentro da string. |
| LENGTH (column|expression) | Retorna o número de caracteres da string. |
| LOWER (column|expression) | Converte strings de caracteres para minúscula. |
| LPAD (column|expression, n, ‘string’) | Retorna uma string com tamanho total de *n* alinhada à direita. |
| REPLACE (x,search\_string,replace\_string) | Procura a string *search\_string* em *x* e substitui por *replace\_string*. |
| RPAD (column|expression, n, ‘string’ | Retorna uma string com tamanho total de *n* alinhada à esquerda. |
| SUBTR(column|expression, m[,n]) | Retorna os caracteres especificados a partir da string de caracteres, começando na posição *m*, com tamanho de *n* caracteres. Se *m* for negativo, a contagem inicia a partir do final da string. Se *n* for omitido, são retornados todos os caracteres até o final da string. |
| UPPER(column|expression) | Converte strings de caracteres para maiúsculas. |

4.2 Funções de conversão entre Maiúsculas/Minúsculas (Exemplos)

|  |  |
| --- | --- |
| Função | Resultado |
| LOWER(‘Introdução ORACLE 11g’) | introdução oracle 11g |
| UPPER(‘Introdução ORACLE 11g’) | INTRODUÇÃO ORACLE 11G |
| INITCAP(‘Introdução ORACLE 11g’) | Introdução Oracle 11g |

SELECT ‘ O Cliente ’ || INITCAP (nome) || ‘ mora em ’ || UPPER (cidade) || ‘ - ’ || LOWER(estado) AS “Informações” FROM taluno;

4.3 Exemplos de Funções:

***SELECT*** *' O(A) Cliente ' || INITCAP (nome) || ' mora em ' || UPPER (cidade) || ' - ' ||* ***LOWER****(estado)* ***AS*** *"Informações"* ***FROM*** *taluno;*

***SELECT*** *\** ***FROM*** *TALUNO;*

***SELECT Concat****(COD\_ALUNO, NOME)* ***FROM*** *TALUNO;*

***SELECT*** *COD\_ALUNO* ***||*** *' - '* ***||*** *NOME* ***FROM*** *TALUNO;*

***SELECT******InitCap****(NOME)* ***FROM*** *TALUNO;*

***SELECT******InStr****(NOME, 'R')* ***FROM*** *TALUNO;*

***SELECT*** *nome, Length(NOME)* ***FROM*** *TALUNO;*

***SELECT*** *nome, Lower(NOME)* ***FROM*** *TALUNO;*

***SELECT*** *nome, Upper(NOME)* ***FROM*** *TALUNO;*

*-- L = left - esquerda*

***SELECT*** *nome,* ***LPad****(COD\_ALUNO, 5, '0')* ***FROM*** *TALUNO;*

*-- R = direita - esquerda - GERALMENTE USADO PARA TEXTOS.*

***SELECT*** *nome,* ***RPad****(COD\_ALUNO, 5, '0')* ***FROM*** *TALUNO;*

***SELECT*** *nome,* ***LPad****(COD\_ALUNO, 5, ' R$')* ***FROM*** *TALUNO;*

*--copia parte de um texto*

***SELECT*** *SubStr(NOME, 3,1)* ***FROM*** *TALUNO;*

***SELECT*** *NOME, SubStr(NOME, 1,3)* ***FROM*** *TALUNO;*

***SELECT*** *REPLACE(NOME, 'E', '3')* ***FROM*** *TALUNO;*

***SELECT*** *NOME,* ***SubStr****(NOME,* ***Length****(nome)-1, 2)* ***FROM*** *TALUNO;*

***SELECT*** *nome, salario,* ***RPad****(SALARIO, 8,'0')* ***FROM*** *TALUNO;*

***SELECT*** *nome, RPad****(****NOME,10,'$')* ***FROM*** *TALUNO;*

***SELECT*** *nome,* ***SubStr****(NOME, 1,3)* ***FROM*** *TALUNO;*

***SELECT******SubStr****(NOME,* ***Length****(nome)-1, 2)* ***FROM*** *TALUNO;*

***SELECT*** *NOME,* ***SubStr****(NOME,6,1)* ***FROM*** *TALUNO;*

***SELECT*** *\** ***FROM*** *TALUNO;*

***SELECT*** *\** ***FROM*** *TALUNO*

***WHERE******Upper****(****SubStr****(CIDADE,1,3)) = 'NOV';*

***UPDATE*** *TALUNO* ***SET***

*SALARIO = 633.47*

***WHERE*** *COD\_ALUNO = 1;*

***UPDATE*** *TALUNO* ***SET***

*COD\_ALUNO = 1*

***WHERE*** *COD\_ALUNO = 20;*

***SELECT***

*SALARIO,*

***REPLACE****(SALARIO, ',' , ''),*

***RPad****(SALARIO, 10, '0'),*

***LPad****(SALARIO, 10, '0'),*

***LPad****(****REPLACE****(SALARIO, ',',''), 10, '0')*

***FROM*** *TALUNO;*

4.4 Exemplos de funções com Data

*--DUAL é uma tabela vazia - Data*

***SELECT*** *\** ***FROM*** *DUAL;*

***SELECT******SYSDATE******FROM******DUAL****;*

***SELECT******Round****(45.925,2), --45,93*

***Trunc****(45.929,2), --45.92*

***Mod****(10, 2) AS RESTO\_DIVISÃO,*

***Trunc****(1.99),*

***Trunc****(1.99, 2)*

***FROM******DUAL****;*

*--Funções de Data/Hora*

***SELECT*** *DATA,* ***SYSDATE****, DATA + 5* ***FROM*** *TCONTRATO;*

***SELECT******SYSDATE*** *- DATA* ***AS*** *DIF\_DIAS* ***FROM*** *TCONTRATO;*

***SELECT******Trunc****(****SYSDATE*** *- DATA)* ***AS*** *DIAS* ***FROM*** *TCONTRATO;*

*--Somando horas em uma tabela*

***SELECT******SYSDATE****,* ***SYSDATE*** *+ 2 / 24* ***AS*** *ADD\_HORAS* ***FROM*** *TCONTRATO;*

*--Somar minutos (1440 minutos tem 1 dia)*

***SELECT******SYSDATE****,* ***SYSDATE*** *+ 15 / 1440* ***AS*** *ADD\_MINUTOS* ***FROM*** *TCONTRATO;*

*--Somar segundo (3600 segundos tem um dia)*

***SELECT******SYSDATE****,* ***SYSDATE*** *+ 30 / (3600 \* 24)* ***AS*** *ADD\_SEGUNDOS* ***FROM*** *TCONTRATO;*

*--Hora fica 00:00:00*

***SELECT******Trunc****(****SYSDATE****)* ***FROM******DUAL****;*

*--Diferença de meses entre datas.*

***SELECT******Trunc****(****Months\_Between****(****SYSDATE****,* ***SYSDATE****-90))* ***AS*** *DIF\_MES* ***FROM*** *DUAL;*

*--Adiciona Meses*

***SELECT******Trunc****(****Add\_Months****(****SYSDATE****, 2))* ***AS*** *ADICIONA\_MES\_DATA* ***FROM*** *DUAL;*

*--Proxima data a partir de um dia da semana*

***SELECT******Next\_Day****(****SYSDATE****, 'FRIDAY')* ***AS*** *PROXIMA\_QUARTA\_DATA* ***FROM*** *DUAL;*

*--Ultimo dia do Mês corrente*

***SELECT******Last\_Day****(****SYSDATE****)* ***AS*** *ULTIMO\_DIA\_MES* ***FROM*** *DUAL;*

*--Primeiro dia do proximo mês*

*--Até 15 dias do mês, pega o primeiro dia do mês atual*

*--A partir do dia 16 do mês atual retorna o primeiro dia do próximo mês*

***SELECT******Round****(****SYSDATE****,* ***'MONTH'****)* ***AS*** *PRIMEIRO\_DIA\_PROXIMO\_MES* ***FROM*** *DUAL;*

*--Trunc sempre mostra o primeiro doa do mes corrente*

***SELECT******Trunc****(SYSDATE,* ***'MONTH'****)* ***AS*** *PRIMEIR\_DIA\_MES\_CORRENTE* ***FROM*** *DUAL;*

4.5 Exemplos de funções com Data usando To\_Char

*--DD -> dia do mês*

***SELECT*** *SYSDATE,* ***To\_Char****(SYSDATE,* ***'DD'****)* ***FROM*** *DUAL;*

*-- Mostra somente a data com 2 ou 4 digitos do ano*

***SELECT To\_Char****(SYSDATE,* ***'DD/MM/YY****')* ***FROM*** *DUAL;*

***SELECT******To\_Char****(SYSDATE,* ***'DD/MM/YYYY'****)* ***FROM*** *DUAL;*

*--Mostra o Dia e o Mês*

***SELECT To\_Char****(SYSDATE,* ***'DD/MM'****)* ***FROM*** *DUAL;*

*--Mostra somento o dia, o mês ou o ano*

***SELECT******To\_Char****(SYSDATE,* ***'DD'****)* ***FROM*** *DUAL;*

***SELECT******To\_Char****(SYSDATE,* ***'MM'****)* ***FROM*** *DUAL;*

***SELECT******To\_Char****(SYSDATE,* ***'YYYY'****)* ***FROM*** *DUAL;*

*--Mostra o mês corrente*

***SELECT******To\_Char****(SYSDATE,****'MONTH'****) MES1* ***FROM*** *DUAL;*

*--Dia da semana*

***SELECT******To\_Char****(****SYSDATE****,'D') DIA\_SEMANA* ***FROM*** *DUAL;*

*--Dia da semana - nome*

***SELECT******To\_Char****(****SYSDATE****,'DY') DIA\_NOME* ***FROM*** *DUAL;*

*--Mostra o nome do Ano*

***SELECT******To\_Char****(SYSDATE,'YEAR') ANO\_NOME* ***FROM*** *DUAL;*

*--Data por extenso*

***SELECT******To\_Char****(SYSDATE,' "NOVO HAMBURGO,"* ***fmDAY*** *","* ***DD*** *"de"* ***fmMonth*** *"de"* ***YYYY'****)* ***FROM*** *DUAL;*

4.6 Exemplos de funções com Hora usando To\_Char

*--Hora e Minuto*

***SELECT******To\_Char****(****SYSDATE****, 'HH24:MI') HORA\_MIN* ***FROM*** *DUAL;*

*--Hora Completa*

***SELECT******To\_Char****(****SYSDATE****, 'HH24:MI:SS') HORA\_MIN* ***FROM*** *DUAL;*

*--Dia, Mês, Hora e minutos*

***SELECT******To\_Char****(****SYSDATE****, 'DD/MM HH24:MI') DATA\_HORA* ***FROM*** *DUAL;*

*--Dia, Mês, Hora, minutos e segundos*

***SELECT******To\_Char****(****SYSDATE****, 'DD/MM HH24:MI:SS') DATA\_HORA* ***FROM*** *DUAL;*

*--Dia, Mês, Ano, Hora, minutos e segundos*

***SELECT******To\_Char****(****SYSDATE****, 'DD/MM/YYYY HH24:MI:SS') DATA\_HORA* ***FROM*** *DUAL;*

4.7 Outras funções com To\_Char

**L -> R$**

**G -> ponto**

**D -> casas decimais**

***SELECT*** *'R$ ' || (****Round****(Salario,2))* ***AS*** *salario* ***FROM*** *TALUNO;*

*--Trim - Tira os espaços em branco*

***SELECT******Trim****(****To\_Char****(Salario, 'L9999.99'))* ***salario*** *,* ***To\_Char****(Salario, 'L99G999D99')* ***salario2******FROM*** *TALUNO;*

*--NVL e NVL2 (Quando queremos retornar Null como 0.*

*--NVL2 é como if,else - Se for Null retorna -1, se não retorna TOTAL*

*--NVL2 Geralmente usa-se quando é utilizado duas colunas, do contrario, caso queira o resultado na mesma coluna, usa o NVL.*

***SELECT*** *\** ***FROM*** *TCONTRATO;*

***SELECT*** *Total,*

*Desconto,*

*Desconto+Total,*

***Nvl****(Desconto,0) desconto1,*

***Nvl****(Desconto,0) + TOTAL,*

***Nvl2****(DESCONTO, TOTAL, -1)*

***FROM*** *TContrato;*

4.8. Selecionando Nome e Substituindo o Valor NULL por OUTRO VALOR Usando NVL

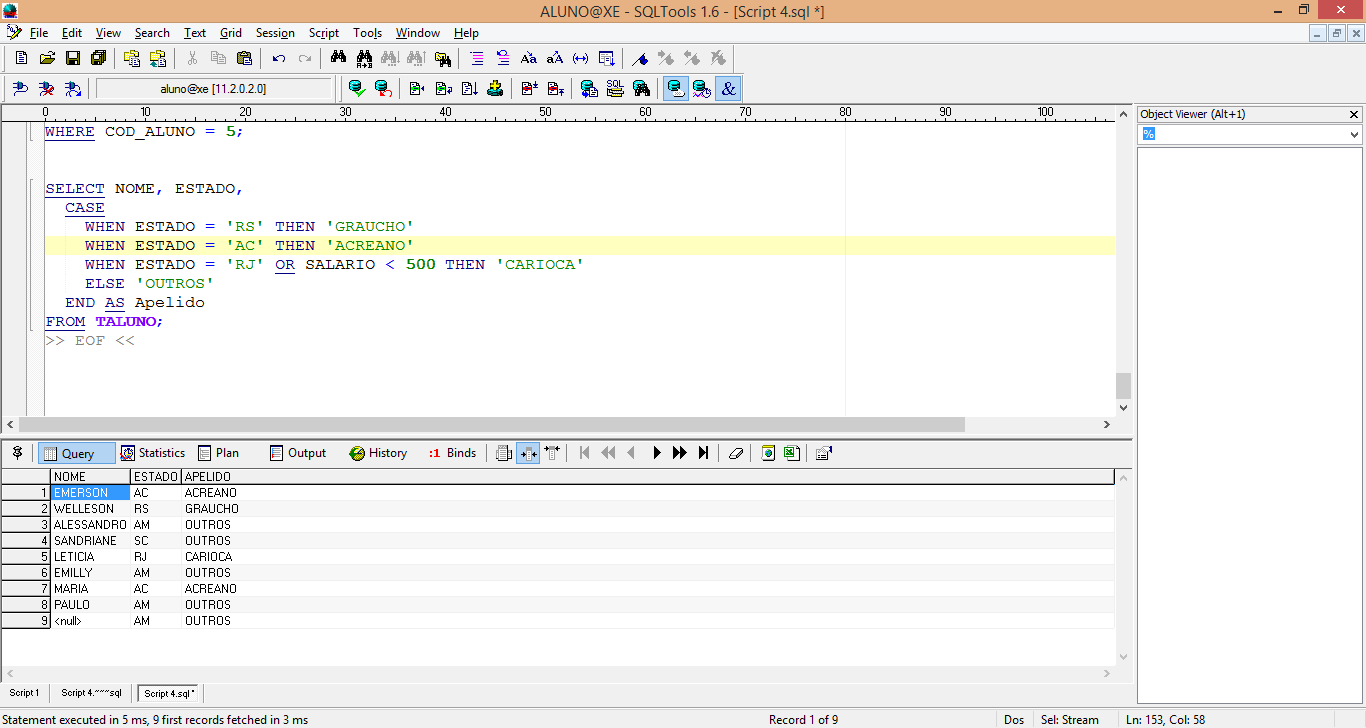
***UPDATE*** *TALUNO* ***SET***

*NOME = NULL*

***WHERE*** *COD\_ALUNO = 9;*

***SELECT*** *COD\_ALUNO,* ***Nvl****(Nome, 'SEM NOME')* ***FROM*** *TALUNO;*

5. CASE

***SELECT*** *NOME, ESTADO,*

***CASE***

***WHEN*** *ESTADO = 'RS'* ***THEN*** *'GRAUCHO'*

***WHEN*** *ESTADO = 'AC'* ***THEN*** *'ACREANO'*

***WHEN*** *ESTADO = 'RJ'* ***OR*** *SALARIO < 500* ***THEN*** *'CARIOCA'*

***ELSE*** *'OUTROS'*

***END AS*** *Apelido*

***FROM*** *TALUNO;*

6. Decode (faz a mesma coisa que o CASA)

***SELECT*** *NOME, ESTADO,*

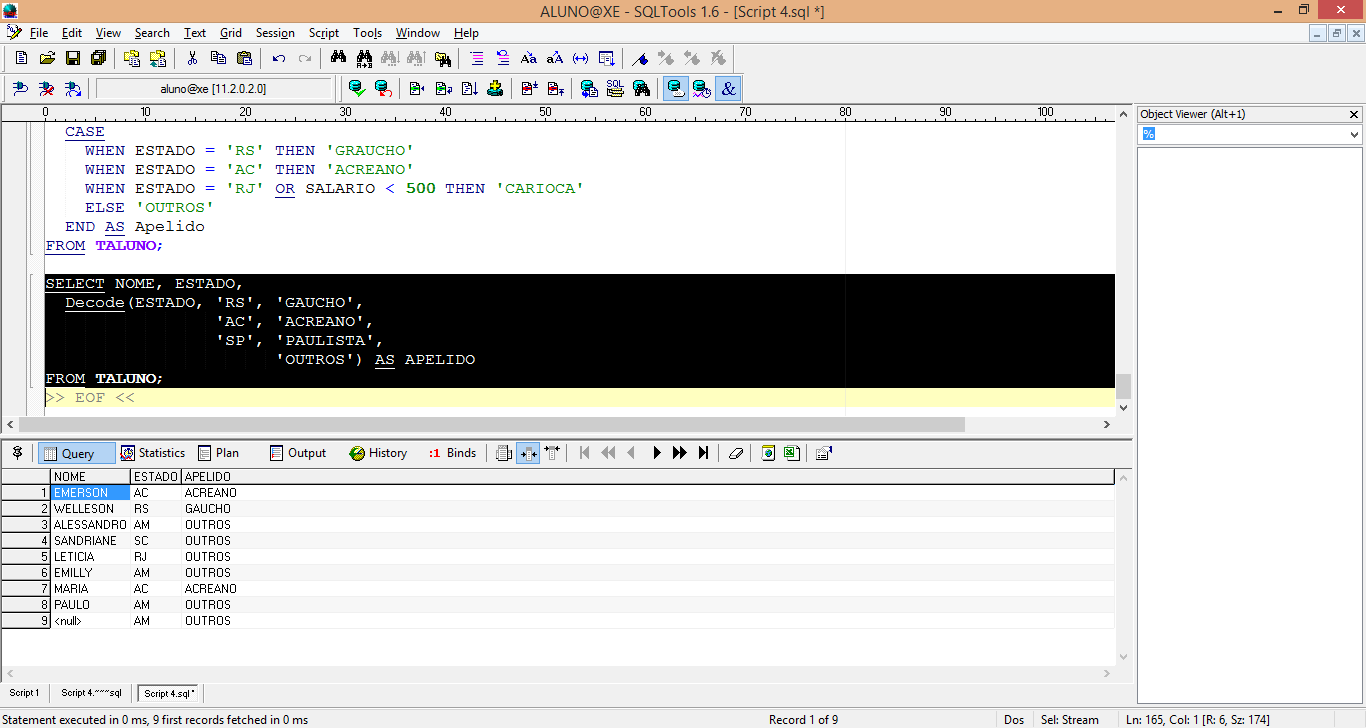
***Decode****(ESTADO, 'RS', 'GAUCHO',*

*'AC', 'ACREANO',*

*'SP', 'PAULISTA',*

*'OUTROS')* ***AS*** *APELIDO*

***FROM*** *TALUNO;*



**CAPÍTULO 5 – EXIBINDO DADOS A PARTIR DE MÚLTIPLAS TABELAS**

1. Objetivo

Escrever comandos **SELECT** para acessar dados a partir de mais de uma tabela utilizando diversos tipos de **JOINS**.

Visualizar dados que geralmente não correspondem a condição de **JOIN** utilizando **OUTER JOIN**.

Executar um **JOIN** de uma tabela com ela mesma (**SELF JOIN**).

1. Obtendo dados a partir de Múltiplas Tabelas

- Às vezes você precisa utilizar dados de mais de uma tabela.

- Precisamos unir campos de uma tabela com campos de outra.

1. O que é um JOIN?

*SELECT table1.column, table2.column*

*FROM table1, table2*

*WHERE table1.column1 = table2.column2;*

- Utilize um JOIN para consultar dados de mais de uma tabela.

Escreva a condição de JOIN da cláusula WHERE.

**Prefixe** o **nome da coluna** com o **nome** **da tabela** quando o mesmo nome de coluna existir em mais de uma tabela.

1. Produto Cartesiano

Um produto cartesiano é formado quando:

- Uma condição de JOIN é **omitida**.

- Uma condição de JOIN é **inválida**.

- Todas as linhas da primeira tabela são unidas à todas as linhas da segunda tabela.

Para evitar um produto cartesiano, sempre inclua uma condição de **JOIN** válida na cláusula **WHERE**.

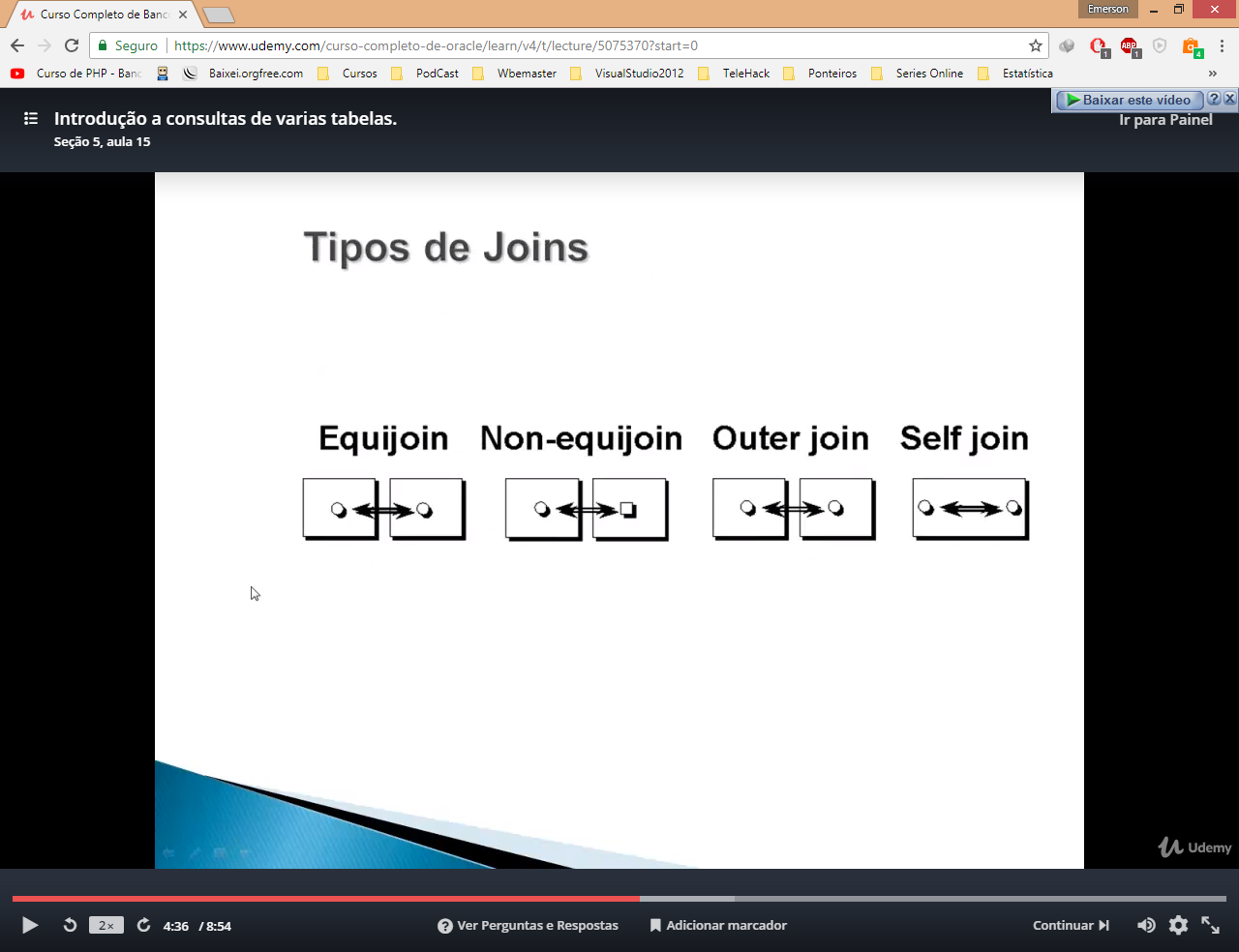
1. Qualificando Nomes de Colunas Ambíguos

Utilize prefixos de tabela para qualificar nomes de colunas que estão em múltiplas tabelas;

Aumente a performance utilizando prefixos de tabela;

Faça distinção de colunas que possuem nomes idênticos mas residem em tabelas diferentes utilizando alias de coluna.

1. Tipos de JOINS



* 1. O que é um EQUIJOIN?

A junção equijoin conecta duas ou mais tabelas de acordo com dados que são comuns a elas, resumindo, ela procura por dados idênticos nas tabelas envolvidas.

* 1. O que é um NON-EQUIJOIN

A junção non-equijoin procura por relacionamentos que não correspondam a uma condição de igualdade, são geralmente utilizadas para verificar valores dentro de um certo conjunto de dados.

* 1. OUTER JOINS

Neste tipo de junção temos como resultado dados existentes em uma tabela que não possuem uma condição de igualdade com outra, ou seja, os registros da tabela que não tem registros de ligação são incluídos no resultado.

Você pode utilizar um outer join para também visualizar as linhas que não correspondem a condição de join.

O operador de outer join é o sinal de adição

+

* 1. SELF JOINS

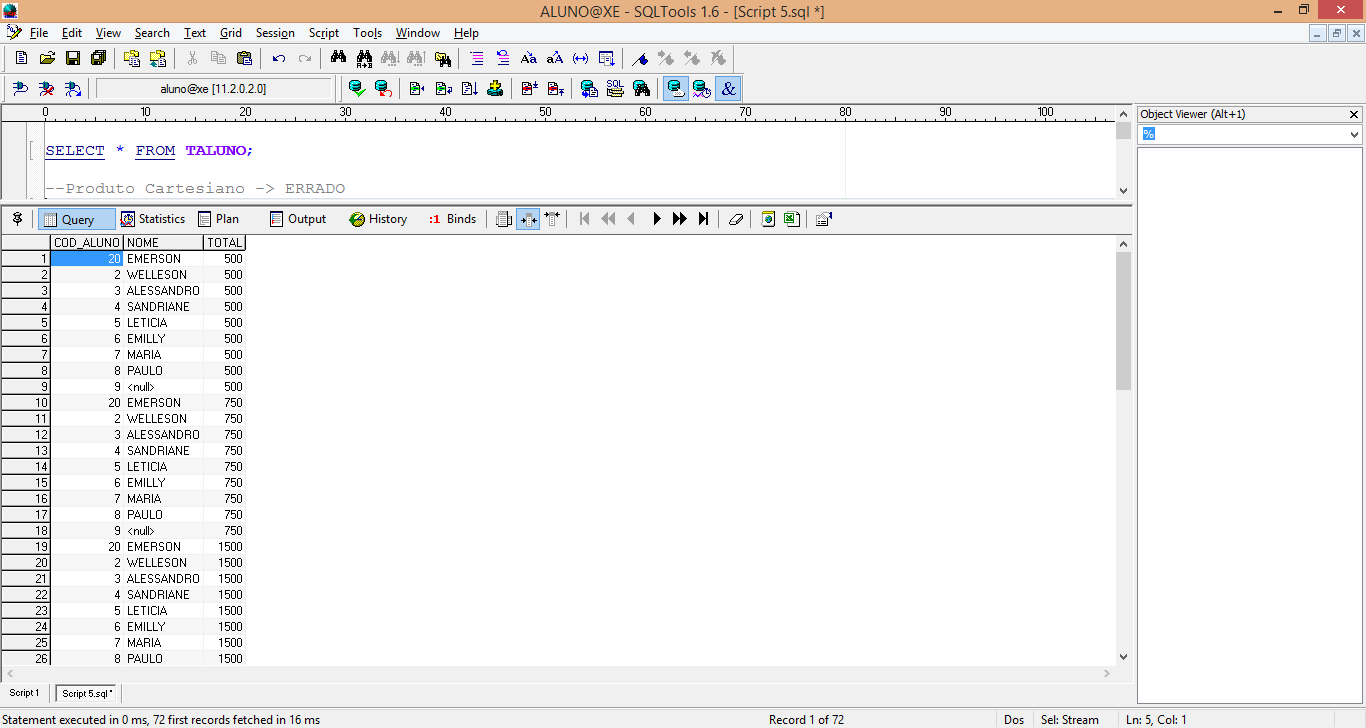
Nas junções deste tipo, são relacionados os dados de uma mesma tabela mais de uma vez, usando o mesmo raciocínio de uma Equijoins, só que nesse caso com a mesma tabela.

1. PRODUTO CARTESIANO (FORMA INCORRETA DE CONSULTAR)

*--Produto Cartesiano -> ERRADO*

***SELECT*** *TALUNO.COD\_ALUNO, TALUNO.NOME, TCONTRATO.TOTAL*

***FROM*** *TALUNO, TCONTRATO;*



1. CRITÉRIO DE UNIÃO

*--Falta Critério de União das duas Tabelas.*

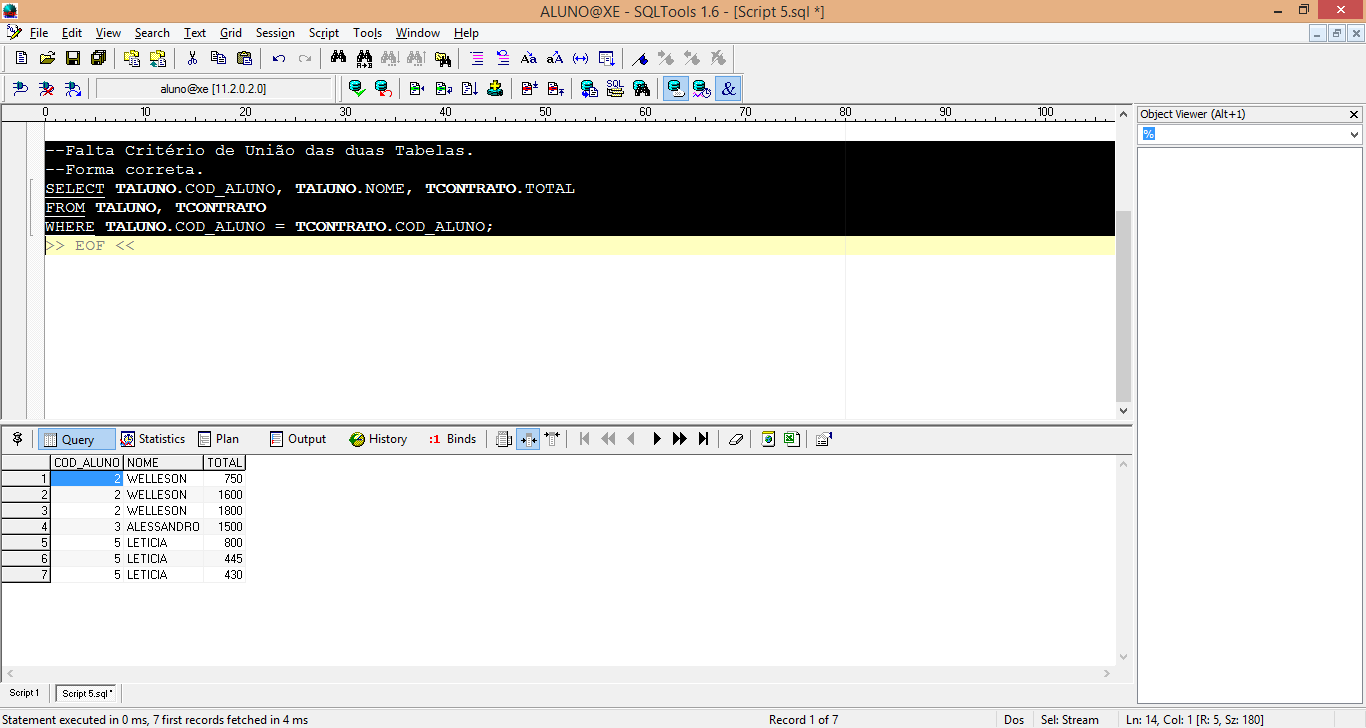
*--Forma correta.*

***SELECT*** *TALUNO.COD\_ALUNO, TALUNO.NOME, TCONTRATO.TOTAL*

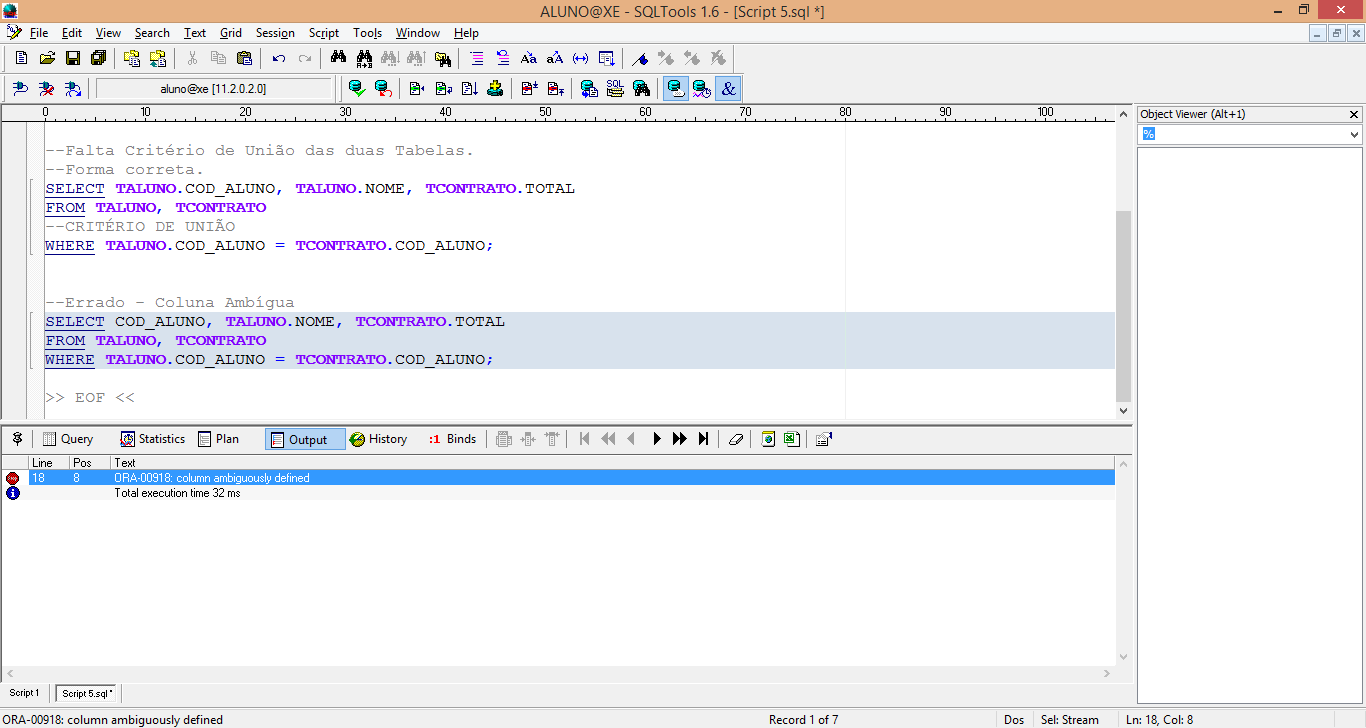
***FROM*** *TALUNO, TCONTRATO*

*--Critério de União*

***WHERE*** *TALUNO.COD\_ALUNO = TCONTRATO.COD\_ALUNO;*

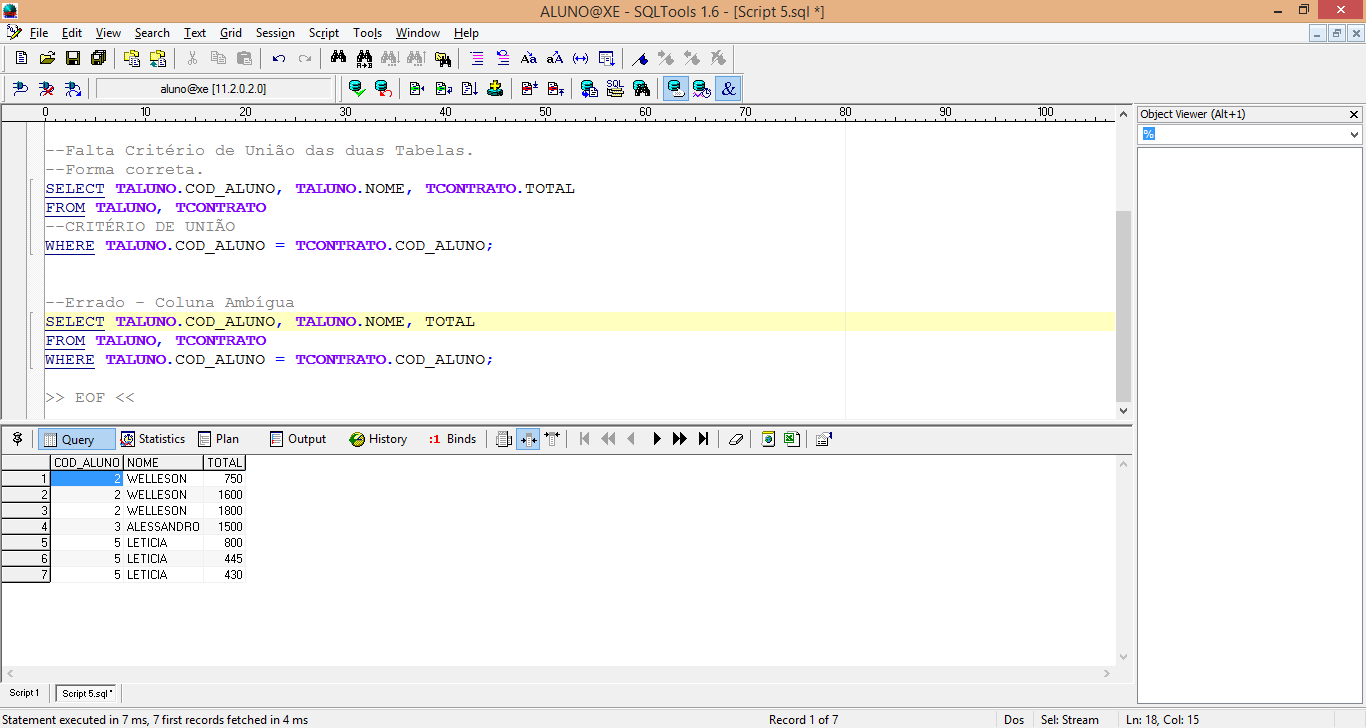


1. COLUNAS AMBIGUAS
   1. Quando duas colunas em tabelas diferentes possuem o mesmo nome.



* 1. TOTAL

Se tirarmos o nome da tabela como prefixo da coluna total, observaremos que a pesquisa funcionar, pois a coluna TOTAL só existe na tabela TCONTRATO.



Obs.: Mas como boas práticas de programação, melhor colocar o nome da tabela como prefixo antes do nome da coluna.

***SELECT*** *TALUNO.COD\_ALUNO, TALUNO.NOME, TCONTRATO.TOTAL*

***FROM*** *TALUNO, TCONTRATO*

***WHERE*** *TALUNO.COD\_ALUNO = TCONTRATO.COD\_ALUNO;*

Correto: quando uma coluna existe com o mesmo nome em uma tabela, colocar prefixo na coluna.

* 1. Dando “apelido às tabelas”

Ao invés de digitarmos o nome inteiro da tabela como prefixo, podemos dar um apelido a tabela e usa-lo como prefixo nos campos. Dessa forma, o código fica bem mais reduzido e elegante.

*--Dando apelido a tabela*

***SELECT*** *ALU.COD\_ALUNO, ALU.NOME* ***AS*** *ALUNO,*

*CON.COD\_CONTRATO, CON.DATA, CON.TOTAL*

***FROM*** *TALUNO ALU, TCONTRATO CON*

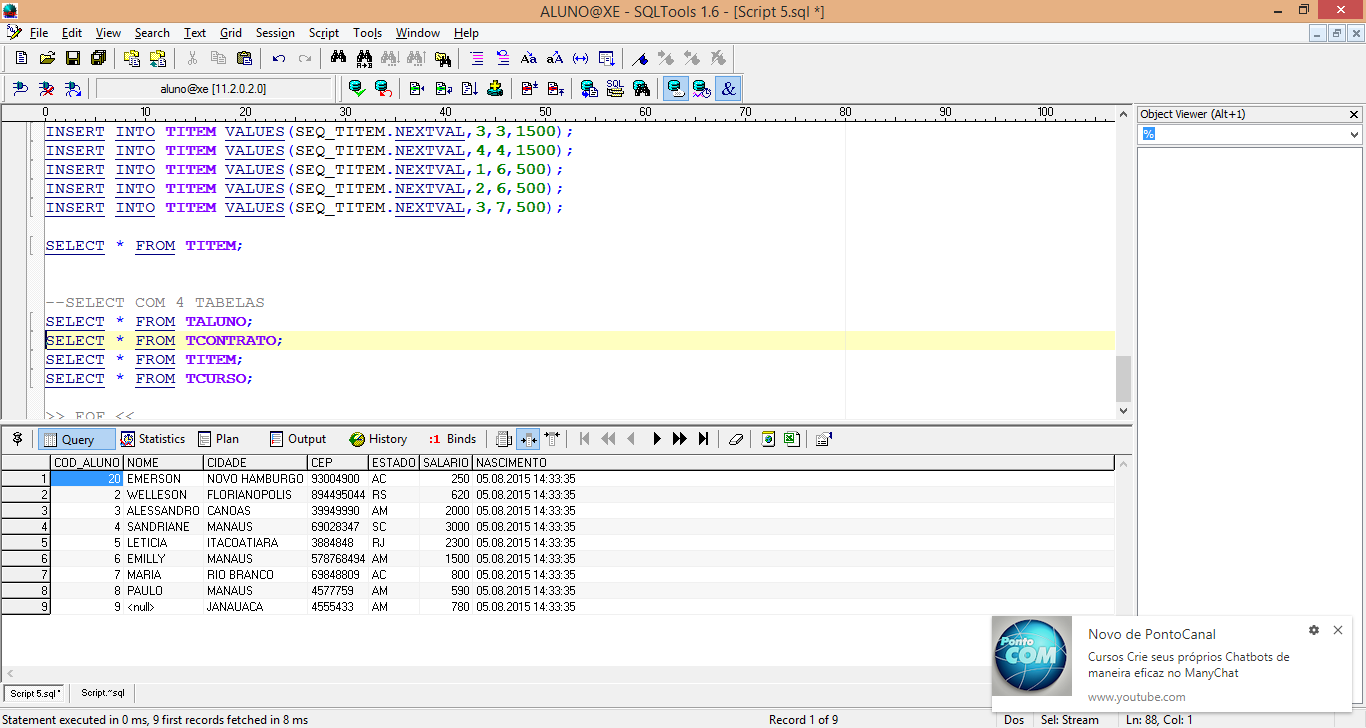
***WHERE*** *CON.COD\_ALUNO* ***=*** *ALU.COD\_ALUNO --Critério de união*

***AND Upper****(ALU.NOME)* ***LIKE*** *'****%****' --Filtro*

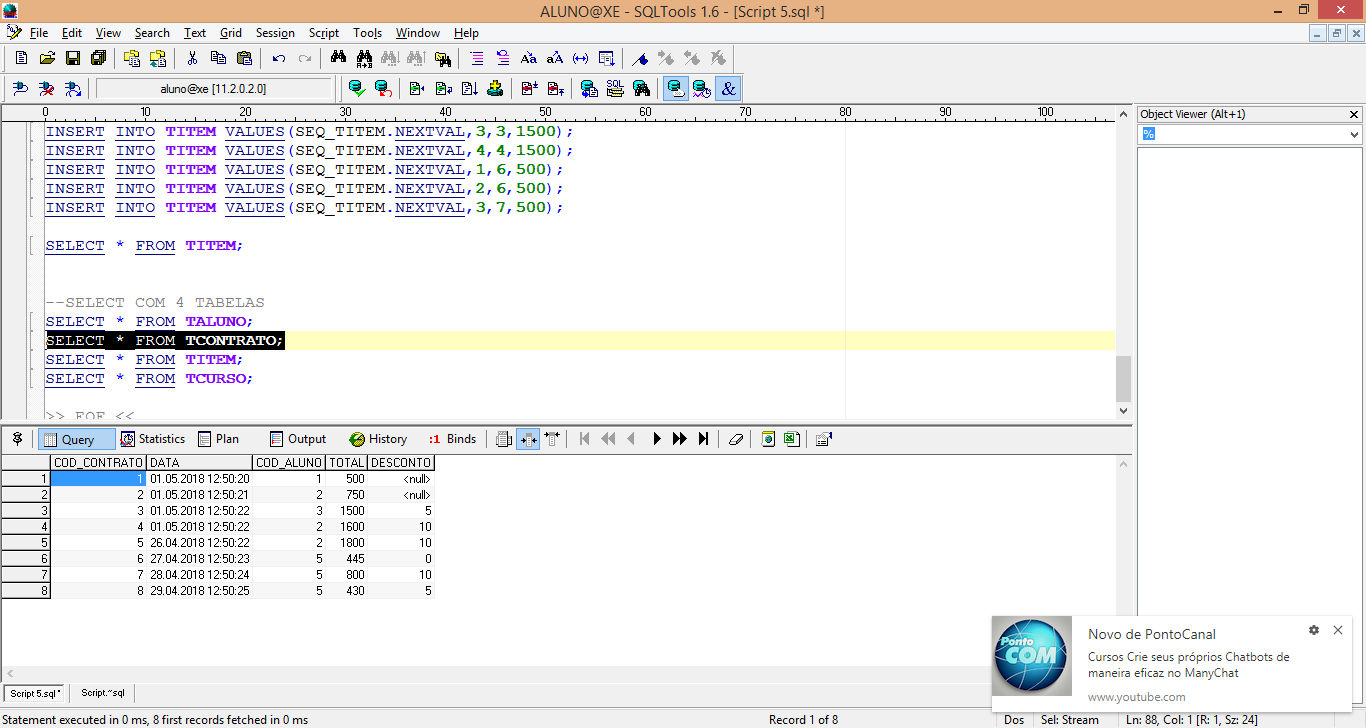
***ORDER BY*** *ALU.NOME; --Ordenar por nome*

* 1. Consulta de 4 tabelas

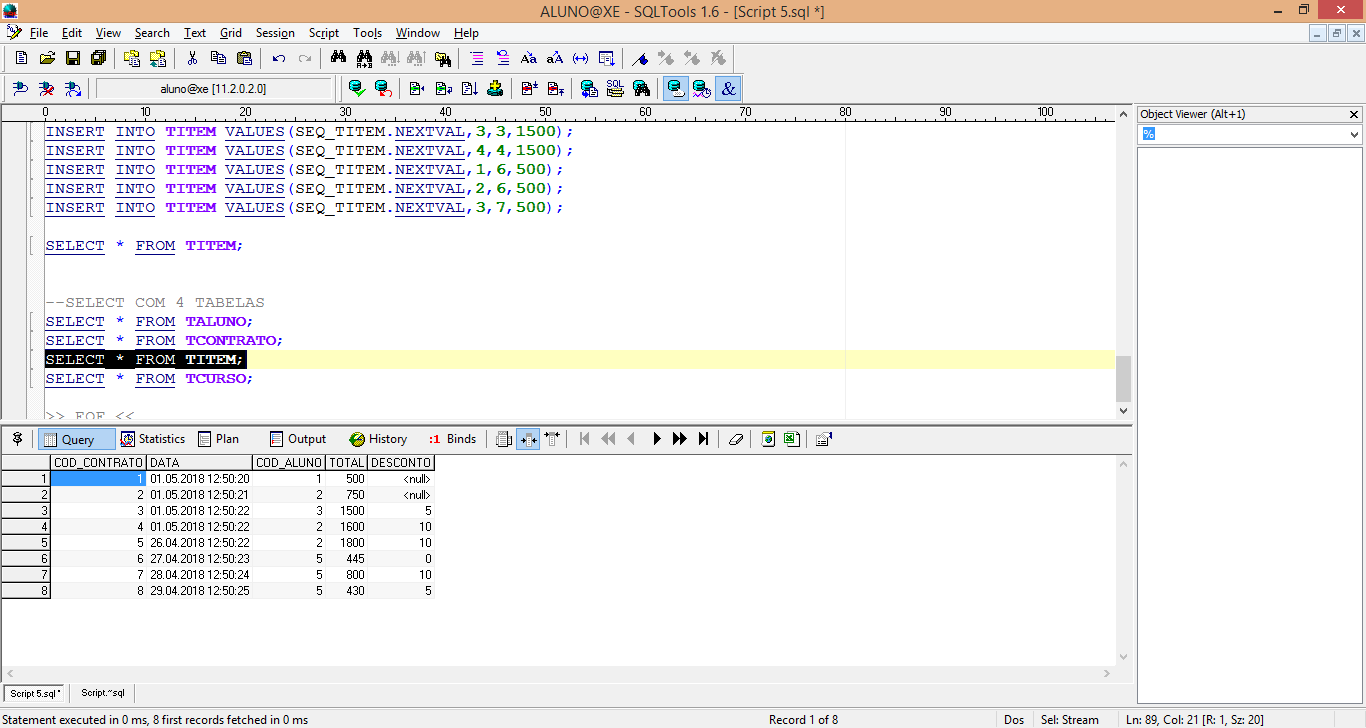
***SELECT*** *\** ***FROM*** *TALUNO*



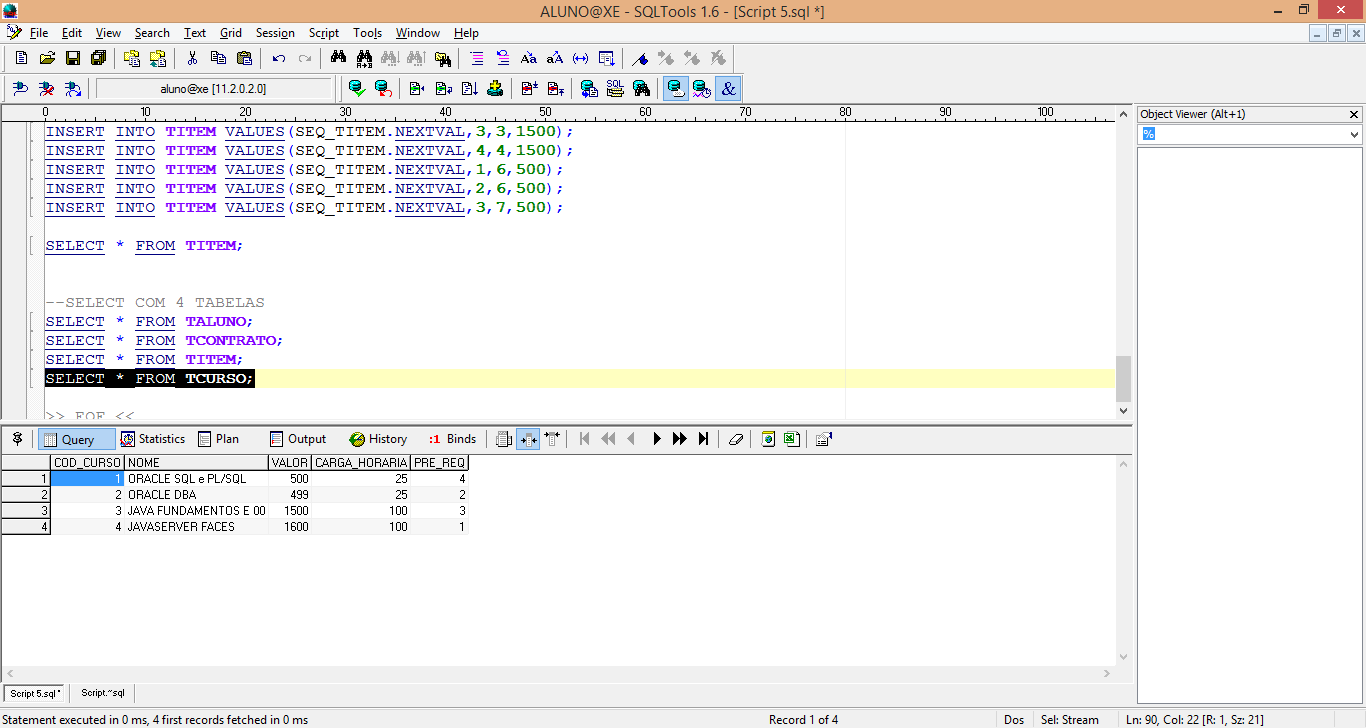
***SELECT*** *\** ***FROM*** *TCONTRATO;*



***SELECT*** *\** ***FROM*** *TITEM;*



***SELECT*** *\** ***FROM*** *TCURSO;*



***SELECT*** *ALU.COD\_ALUNO, ALU.NOME* ***AS*** *ALUNO,*

*CON.COD\_CONTRATO, CON.DATA, CON.TOTAL,*

*ITE.COD\_CURSO, CUR.NOME* ***AS*** *CURSO,*

*ITE.VALOR*

***FROM*** *TALUNO ALU, TCONTRATO CON,*

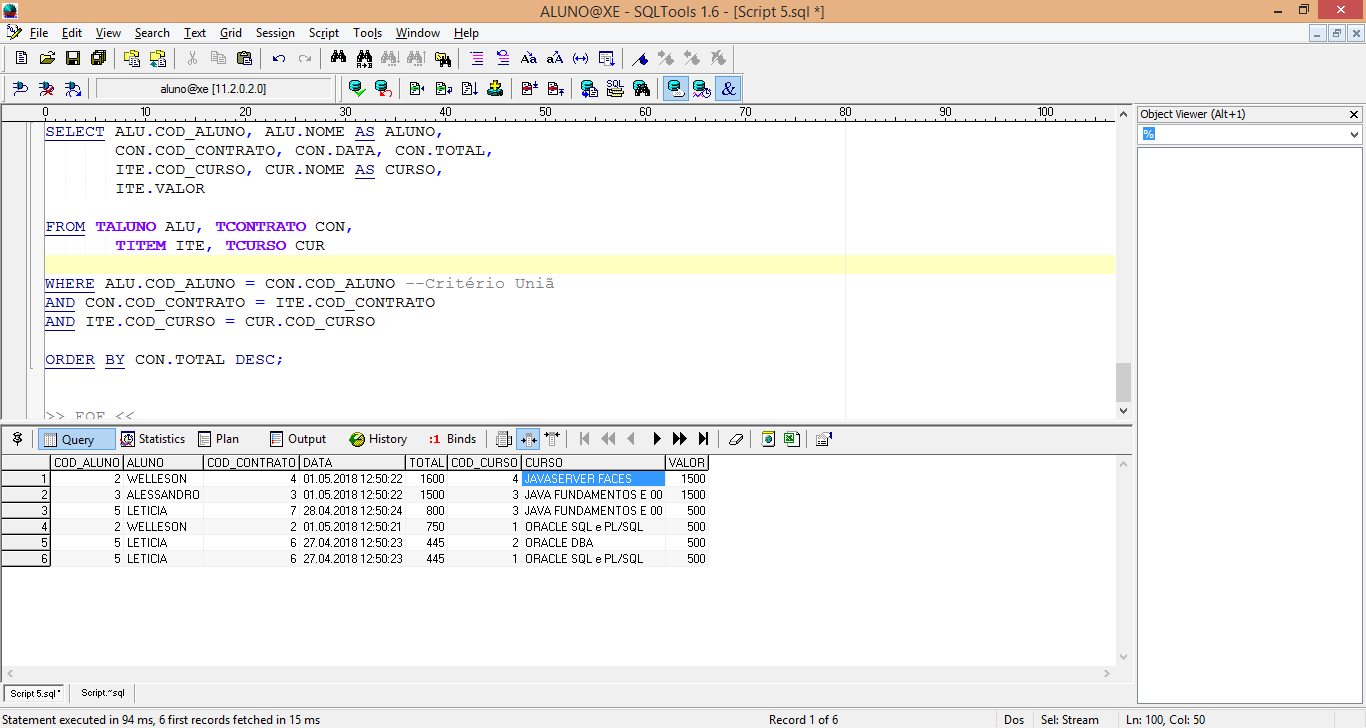
*TITEM ITE, TCURSO CUR*

***WHERE*** *ALU.COD\_ALUNO* ***=*** *CON.COD\_ALUNO*

***AND*** *CON.COD\_CONTRATO* ***=*** *ITE.COD\_CONTRATO*

***AND*** *ITE.COD\_CURSO* ***=*** *CUR.COD\_CURSO*

***ORDER BY*** *CON.TOTAL* ***DESC****;*



* 1. Critério União - (+) Mostra todos os alunos, mesmo não tendo correspondência nas outras tabelas.

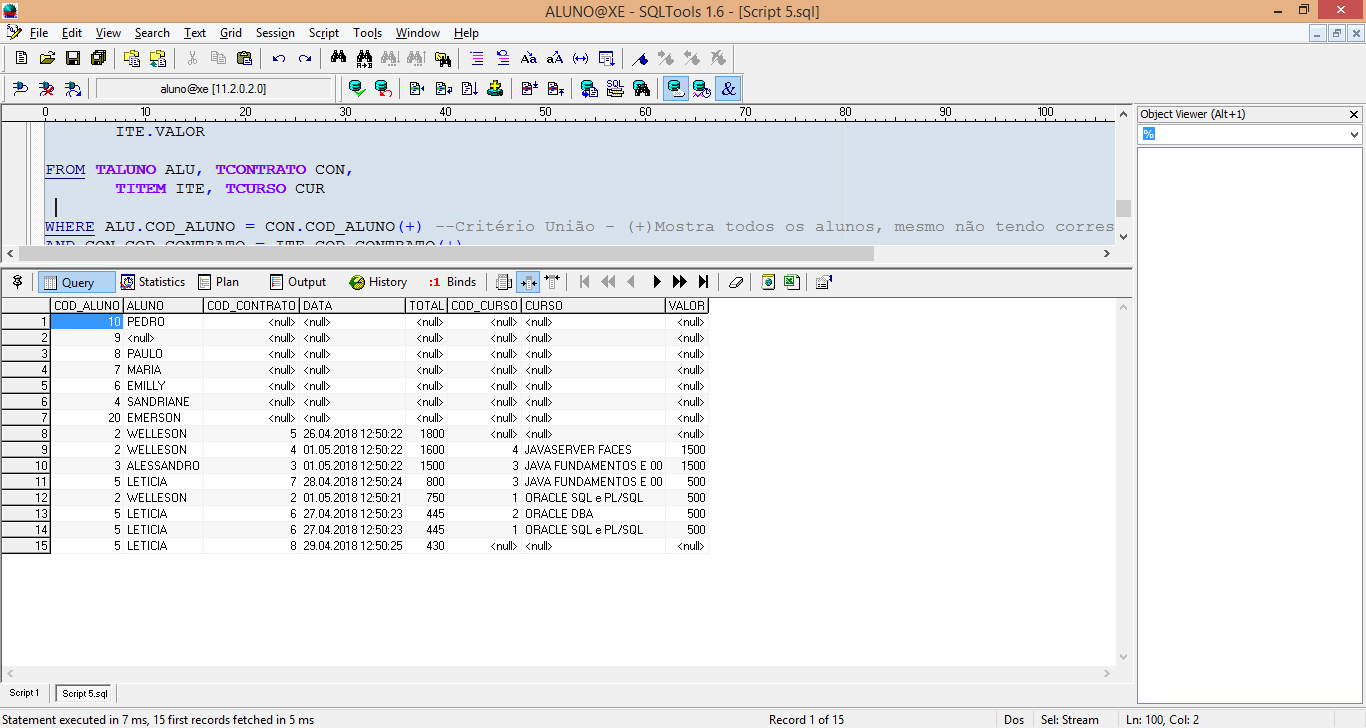
*...*

***WHERE*** *ALU.COD\_ALUNO* ***=*** *CON.COD\_ALUNO(+)*

*AND CON.COD\_CONTRATO = ITE.COD\_CONTRATO(+)*

*AND ITE.COD\_CURSO = CUR.COD\_CURSO(+)*

*...*



Obs.:

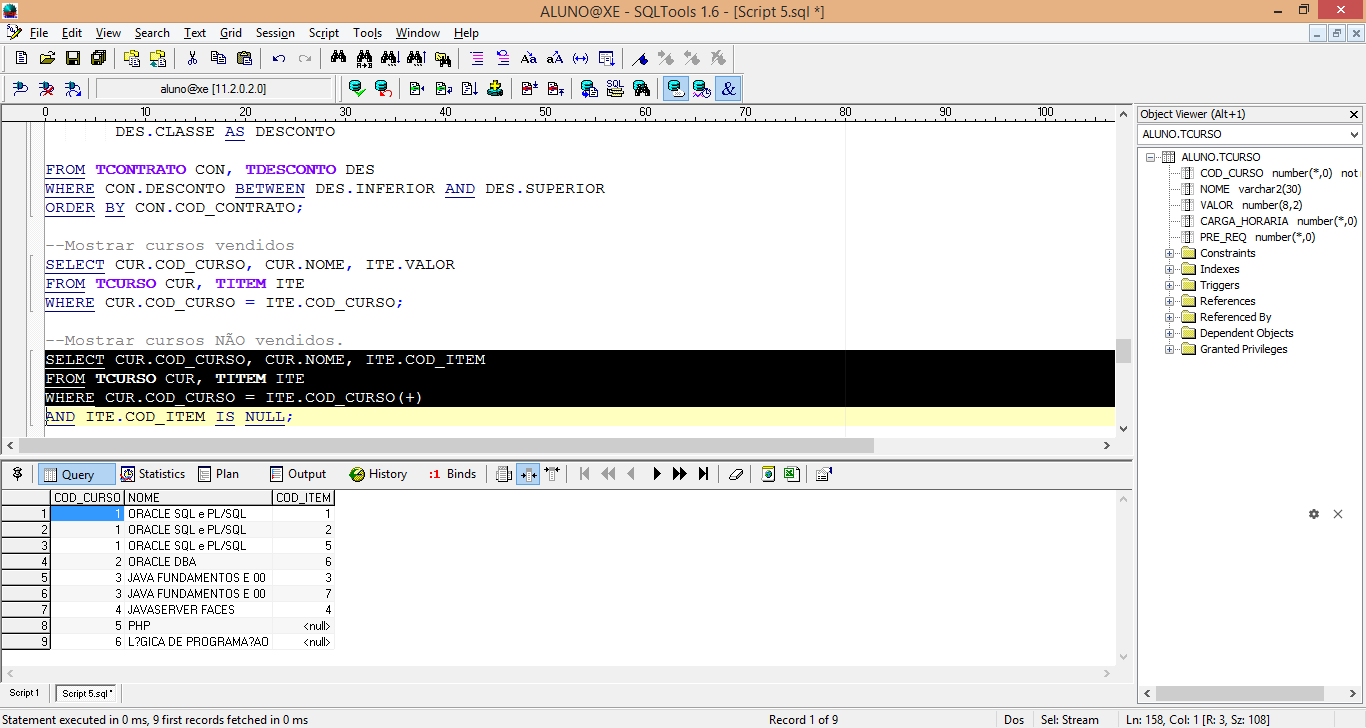
**(+) JOIN** = Só mostra registros que existem nas duas tabelas.

* 1. Mostrar todos os cursos

***SELECT*** *CUR.COD\_CURSO, CUR.NOME, ITE.COD\_ITEM*

***FROM*** *TCURSO CUR, TITEM ITE*

***WHERE*** *CUR.COD\_CURSO* ***=*** *ITE.COD\_CURSO****(+)***

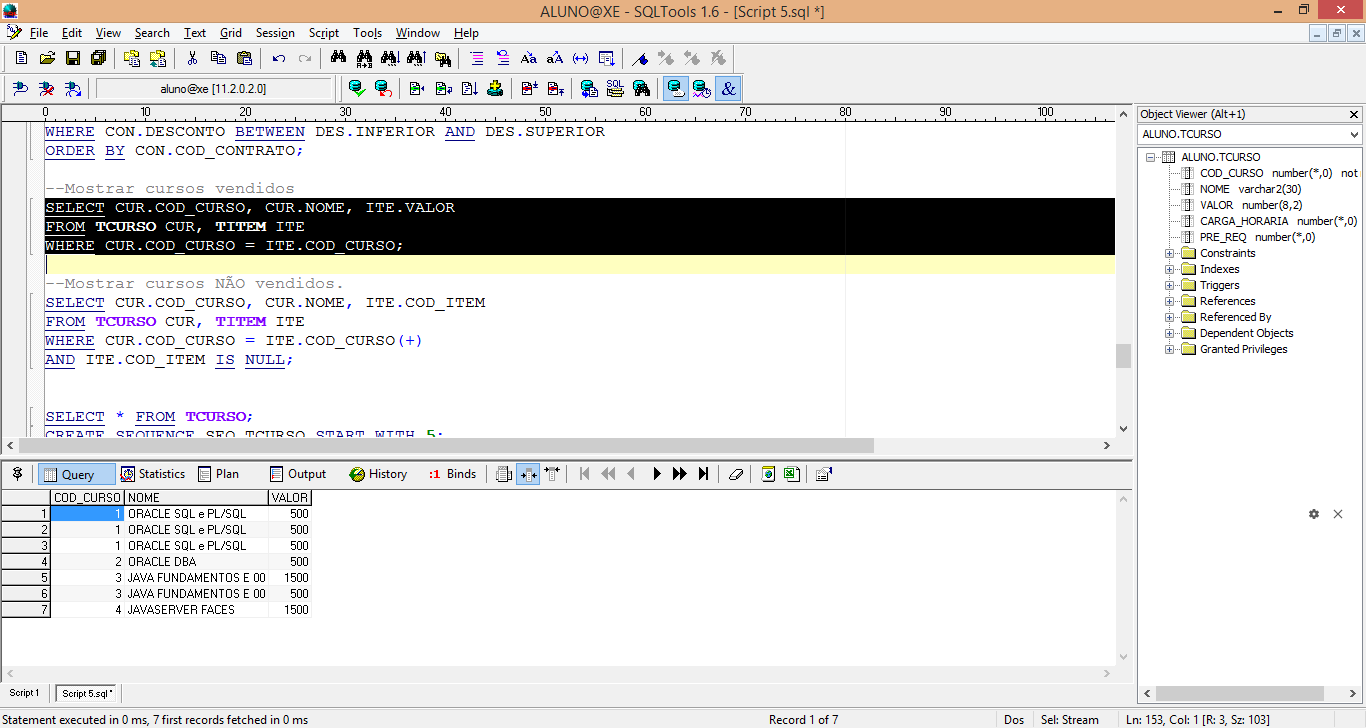


* 1. Mostrar cursos vendidos

***SELECT*** *CUR.COD\_CURSO, CUR.NOME, ITE.VALOR*

***FROM*** *TCURSO CUR, TITEM ITE*

***WHERE*** *CUR.COD\_CURSO* ***=*** *ITE.COD\_CURSO;*



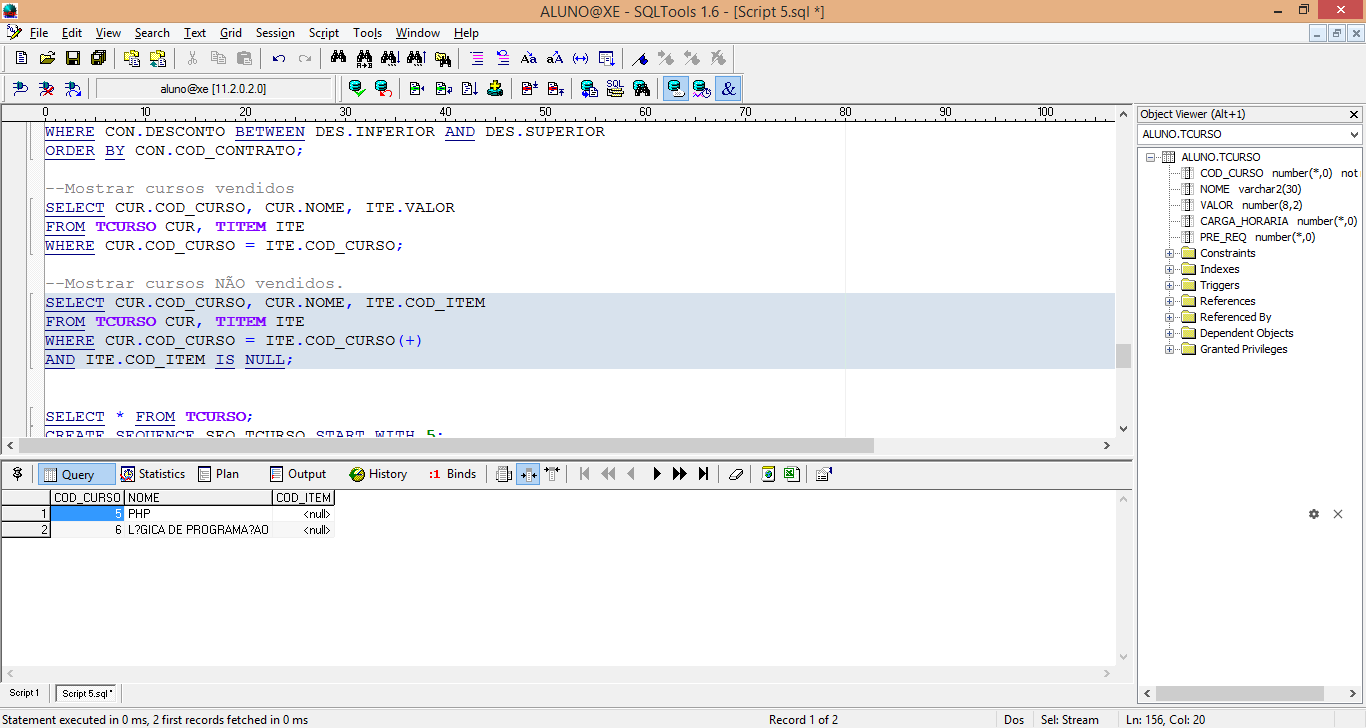
* 1. Mostrar cursos NÃO vendidos.

***SELECT*** *CUR.COD\_CURSO, CUR.NOME, ITE.COD\_ITEM*

***FROM*** *TCURSO CUR, TITEM ITE*

***WHERE*** *CUR.COD\_CURSO* ***=*** *ITE.COD\_CURSO****(+)***

***AND*** *ITE.COD\_ITEM IS NULL;*



* 1. Select de duas Tabelas (com ela mesma)

*--Para fazer o curso de Lógica, não precisa de pré-requisito;*

*--Para fazer o curso de PHP, é necessário ter Lógica como Pré-requesito;*

*--Para fazer o curso de Oracle, é necessário ter Lógica como Pré-requesito;*

*--Para fazer o curso de Java, é necessário ter Lógica como Pré-requesito;*

*--Para fazer o curso de DBA, é necessário ter SQL e PL/SQL como Pré-requesito;*

***UPDATE*** *TCURSO* ***SET*** *PRE\_REQ* ***=*** *6* ***WHERE*** *COD\_CURSO* ***=*** *5;*

***UPDATE*** *TCURSO* ***SET*** *PRE\_REQ* ***=*** *6* ***WHERE*** *COD\_CURSO* ***=*** *1;*

***UPDATE*** *TCURSO* ***SET*** *PRE\_REQ* ***=*** *6* ***WHERE*** *COD\_CURSO* ***=*** *3;*

***UPDATE*** *TCURSO* ***SET*** *PRE\_REQ* ***=*** *1* ***WHERE*** *COD\_CURSO* ***=*** *2;*

***UPDATE*** *TCURSO* ***SET*** *PRE\_REQ* ***=*** *3* ***WHERE*** *COD\_CURSO* ***=*** *4;*

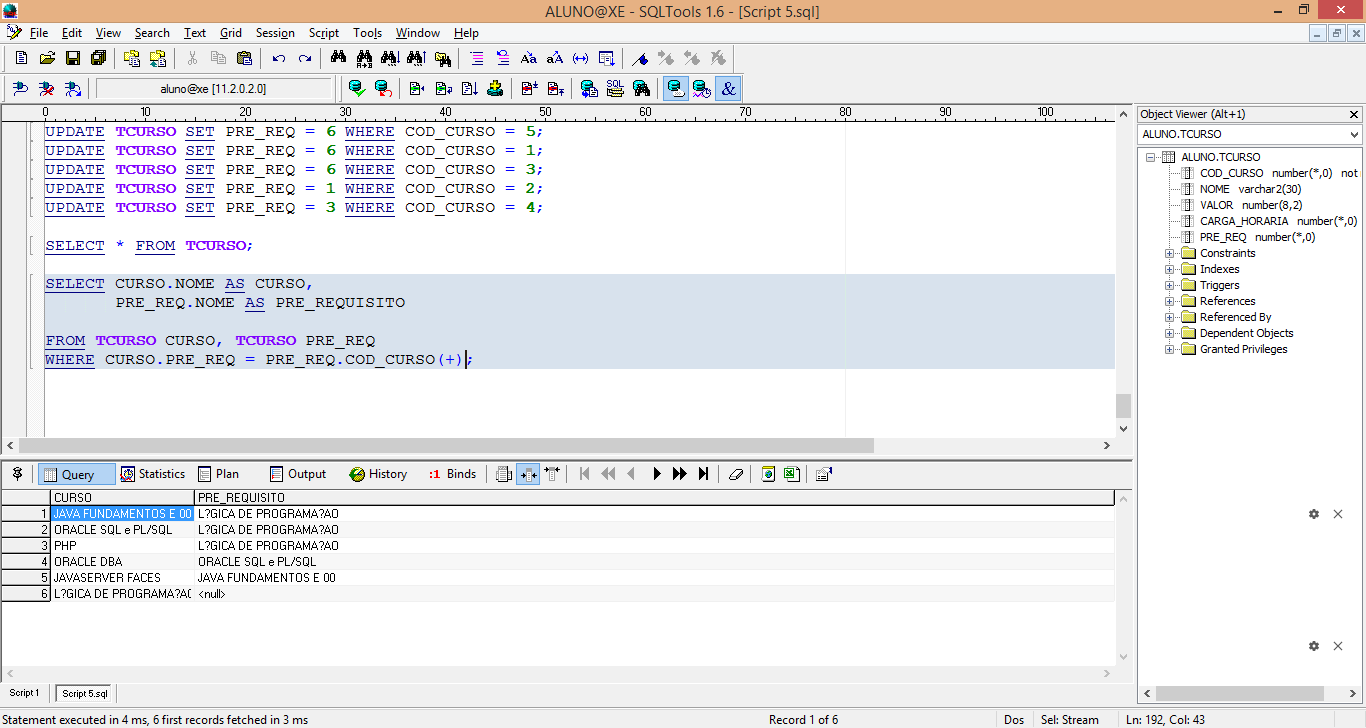
***SELECT*** *\** ***FROM*** *TCURSO;*

***SELECT*** *CURSO.NOME* ***AS*** *CURSO,*

*PRE\_REQ.NOME* ***AS*** *PRE\_REQUISITO*

***FROM*** *TCURSO CURSO, TCURSO PRE\_REQ*

***WHERE*** *CURSO.PRE\_REQ* ***=*** *PRE\_REQ.COD\_CURSO****(+);***

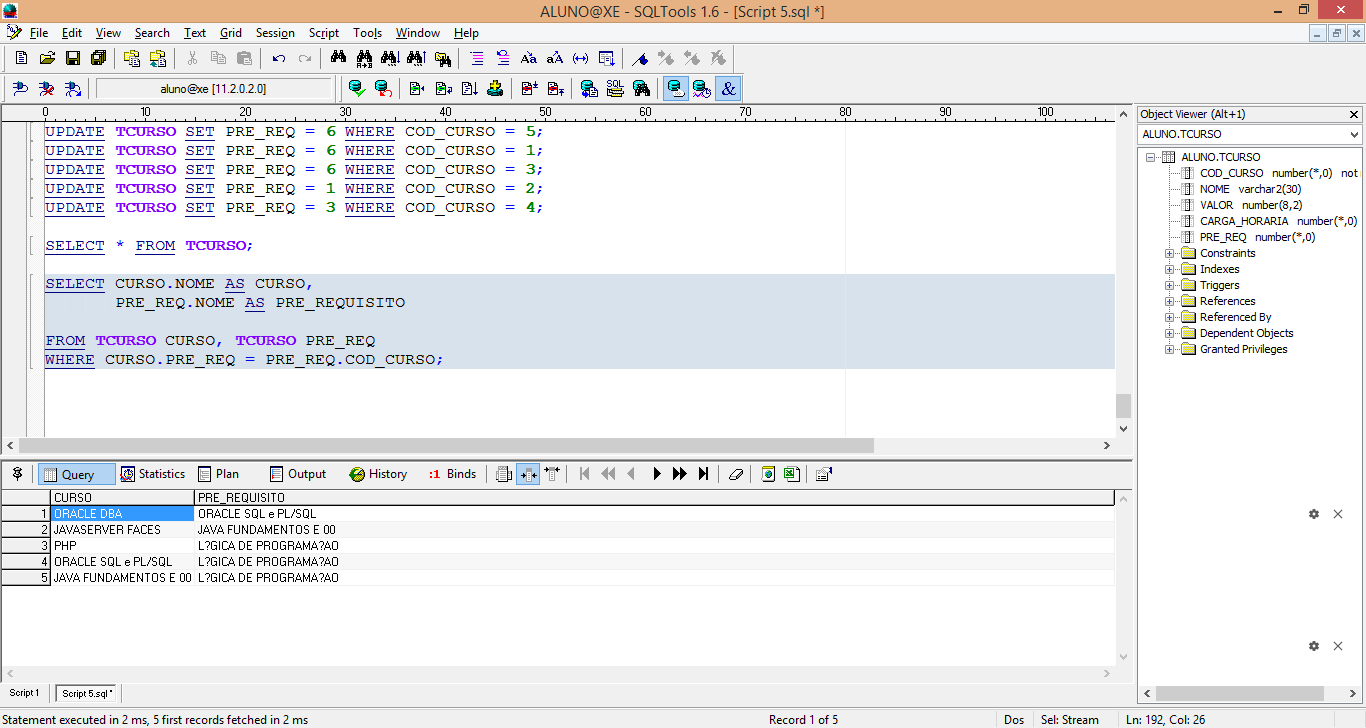


***SELECT*** *CURSO.NOME* ***AS*** *CURSO,*

*PRE\_REQ.NOME* ***AS*** *PRE\_REQUISITO*

***FROM*** *TCURSO CURSO, TCURSO PRE\_REQ*

***WHERE*** *CURSO.PRE\_REQ* ***=*** *PRE\_REQ.COD\_CURSO;*



**CAPÍTULO 6 – UTILIZANDO FUNÇÕES DE GRUPO**

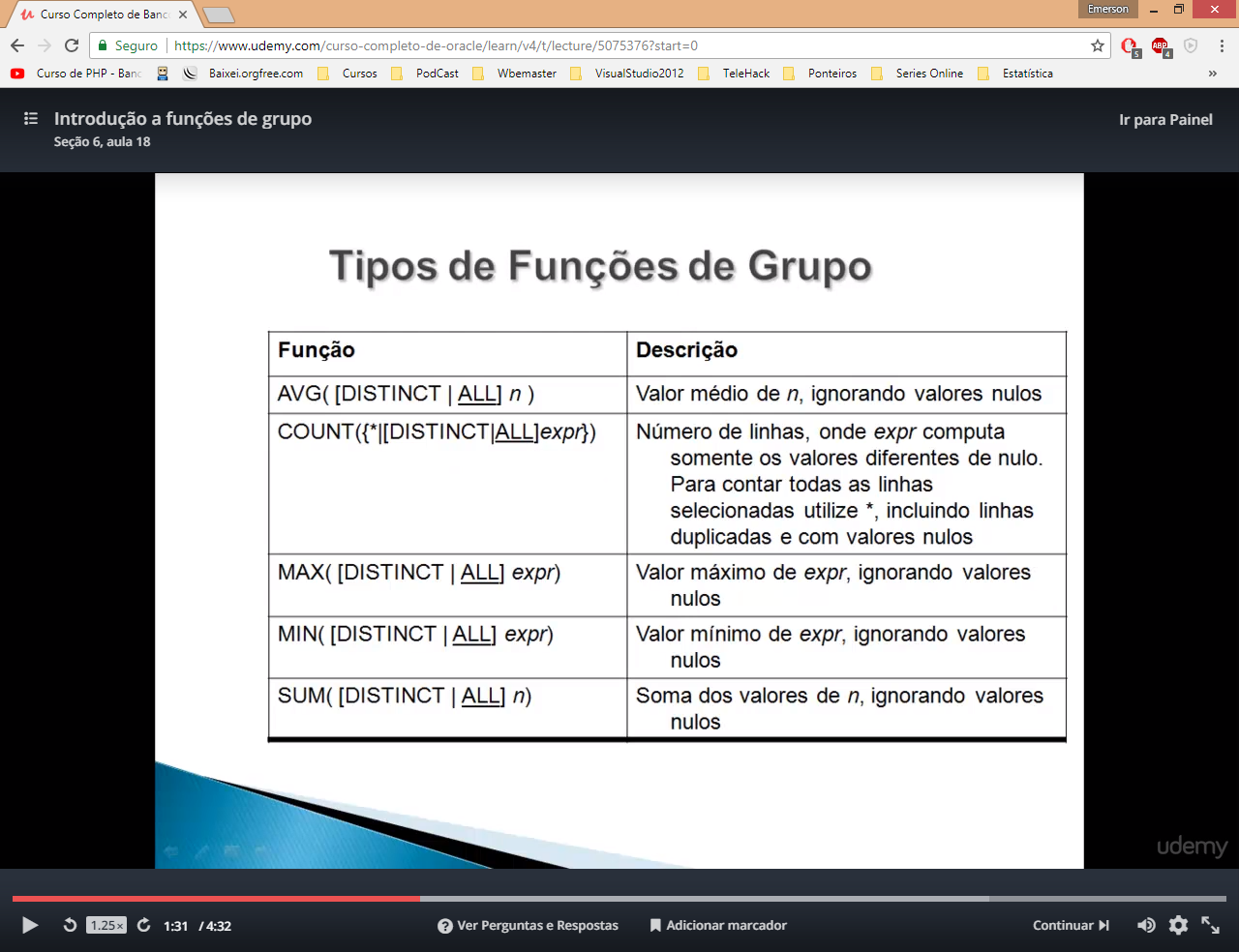
**Objetivo:**

- Identificar as funções de grupo disponíveis;  
- Descrever o uso de funções de grupo;  
- Agrupar dados Utilizando a cláusula GROUP BY;  
- Incluir ou excluir linhas agrupada utilizando a cláusula HAVING.

**O que são funções de grupo?**

* Diferente das funções básicas (single-row), as funções de grupo atuam em conjuntos de linhas para obter um resultado por grupo;
* Estes conjuntos podem ser a tabela inteira ou a tabela dividida em grupos.

**Tipos de Funções de Grupo:**



**Utilizando Funções de Grupo:**

***SELECT******group\_function****(column)*

***FROM*** *table*

*[****WHERE*** *condition]*

*[****ORDER BY*** *expr];*

6.1. Count, Avg, Min, Max, Sum

***SELECT Count****(****\*****)* ***AS*** *QTDE\_REGISTROS,*

***Avg****(****TOTAL****)* ***AS*** *MEDIA,*

***Round****(****Avg****(TOTAL))* ***AS*** *MEDIA,*

***Max****(TOTAL)* ***AS*** *MAXIMO,*

***Min****(TOTAL)* ***AS*** *MINIMO,*

***Sum****(TOTAL)* ***AS*** *SOMA,*

***Max****(DATA)* ***AS*** *DATA\_MAIOR,*

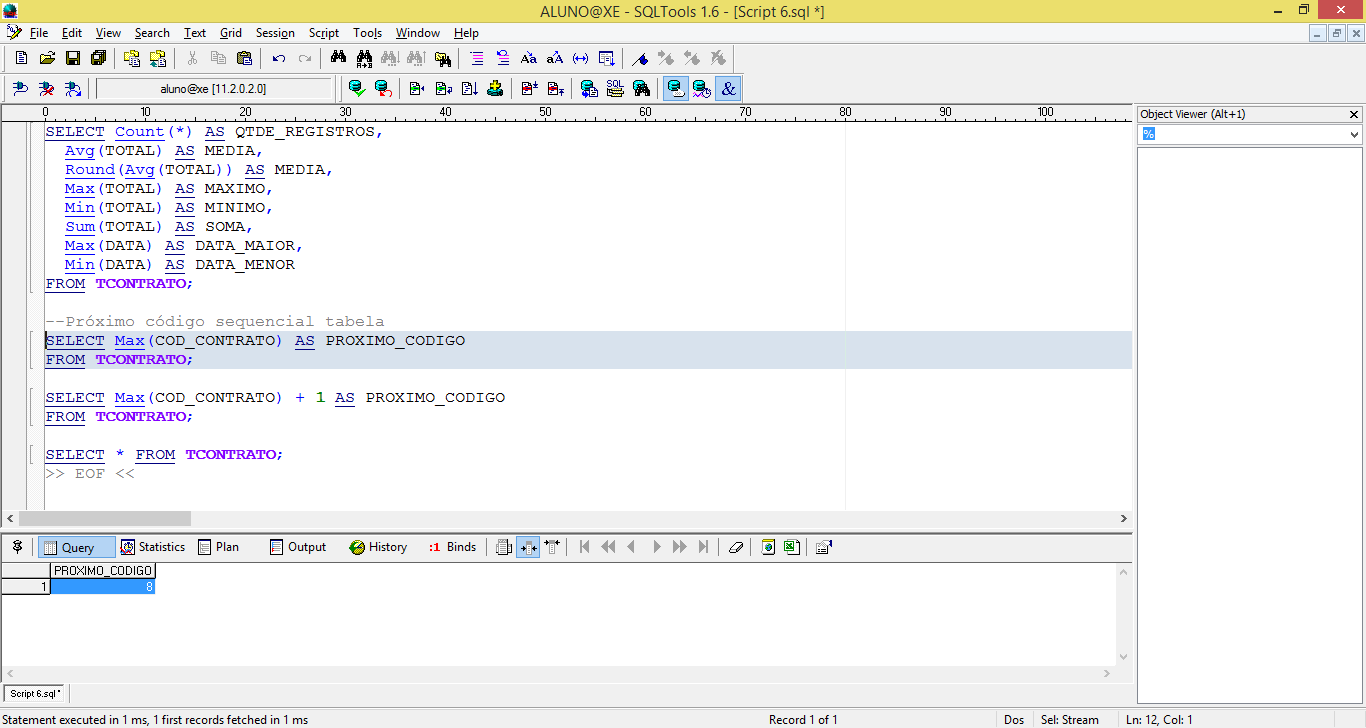
***Min****(DATA)* ***AS*** *DATA\_MENOR*

***FROM*** *TCONTRATO;*

6.2. Próximo código sequencial tabela

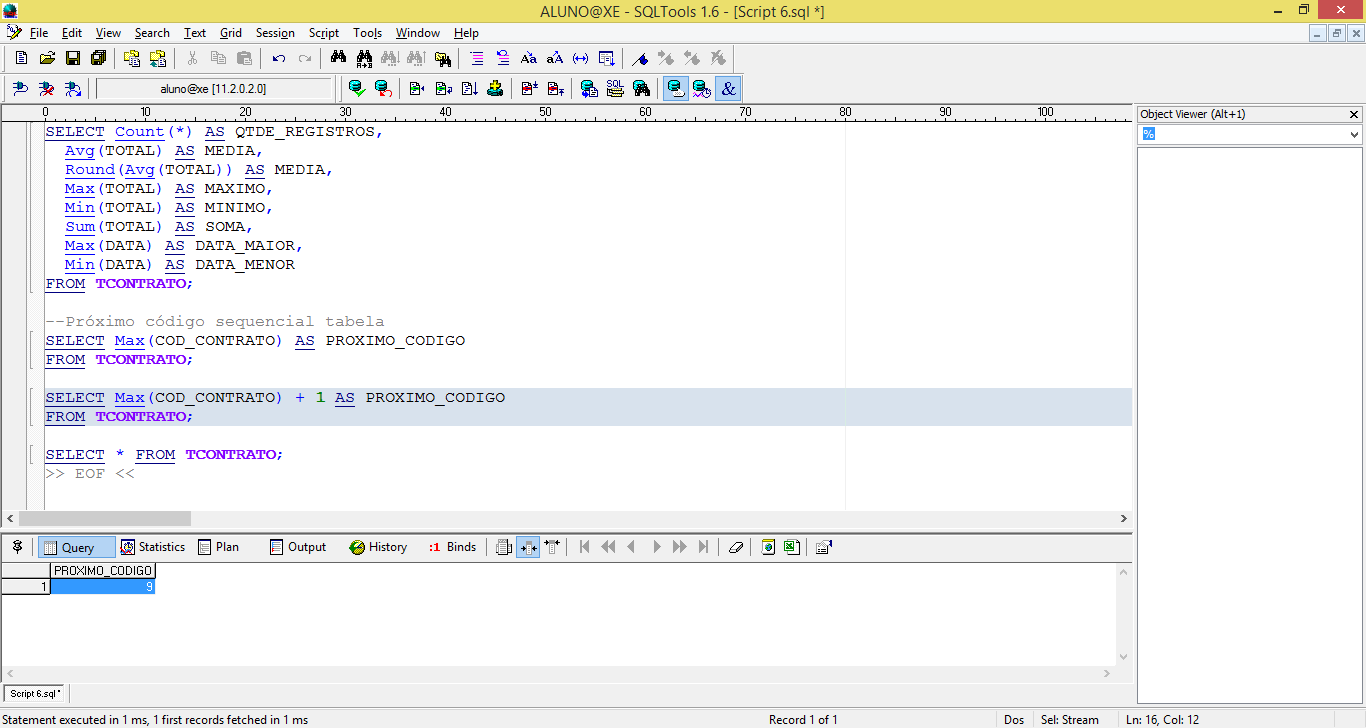
***SELECT******Max****(COD\_CONTRATO)* ***AS*** *PROXIMO\_CODIGO*

***FROM*** *TCONTRATO;*



***SELECT Max****(COD\_CONTRATO)* ***+*** *1* ***AS*** *PROXIMO\_CODIGO*

***FROM*** *TCONTRATO;*



6.3. Total de Contrato por Aluno

COLUNAS COM FUNÇÃO DE GRUPO DA TABELA COM COLUNAS SEM FUNÇÃO DE GRUPO, PRECISA ESPECIFICAR O GRUPO BY COM A COLUNA SEM FUNÇÃO NO GROUP BY

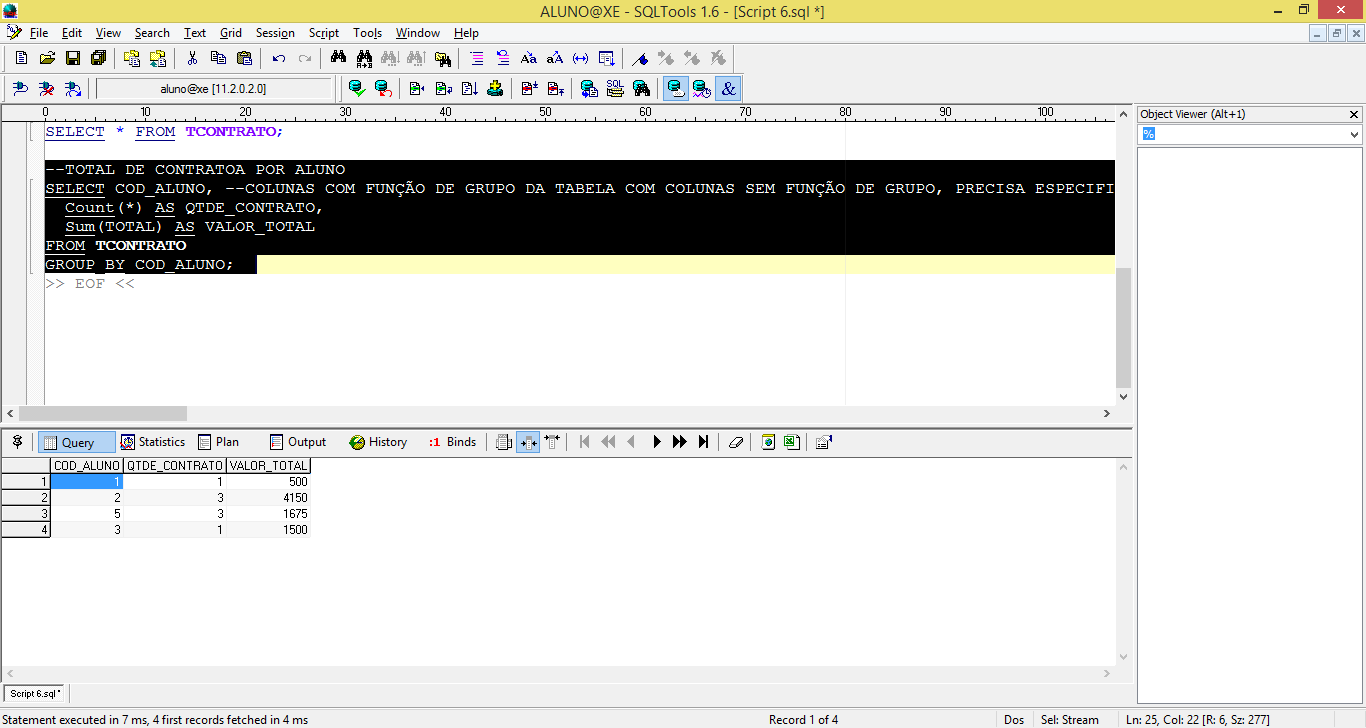
***SELECT*** *COD\_ALUNO,*

***Count(****\*****)******AS*** *QTDE\_CONTRATO,*

***Sum****(TOTAL)* ***AS*** *VALOR\_TOTAL*

***FROM*** *TCONTRATO*

***GROUP BY*** *COD\_ALUNO;*



6.4. Selecionando e Agrupando Datas

REVISÃO

***SELECT******To\_Char****(****Trunc****(SYSDATE), 'DD/MM/YYYY')* ***AS*** *DATA* ***FROM*** *DUAL;*

*--Trunc - TIRA A HORA*

*--To\_Char - COLOCA A DATA NO FORMATO QUE QUEREMOS DD/MM/YYYY*

*--TOTAL DE CONTRATO POR DATA*

***SELECT******To\_Char****(****Trunc****(DATA), 'DD/MM/YYYY')* ***AS*** *DATA,*

***Sum****(TOTAL)* ***AS*** *SOMA, --soma do total vendido*

***Avg****(TOTAL)* ***AS*** *MEDIA, --media do total vendido*

***Count****(\*)* ***AS*** *QTDE --quantidade do total vendido*

***FROM*** *TCONTRATO*

***GROUP BY Trunc****(DATA) --Trunc pra ordernar por data e não por hora*

***ORDER BY*** *DATA* ***DESC***

6.5. Exercícios

***SELECT******Count****(****\*****)* ***AS*** *QTDE\_REGISTROS*

***FROM*** *TCONTRATO;*

6.6. COUNT -> IGNORA NULOS

***SELECT Count****(DESCONTO)*

***FROM*** *TCONTRATO;*

6.7. QTDE DE REGISTROS COM ESTADO INFORMADO

***SELECT Count****(ESTADO)* ***FROM*** *TALUNO;*

6.8. QTDE DE ESTADOS DIFERENTES

***SELECT Count****(****DISTINCT****(ESTADO))* ***FROM*** *TALUNO;*

6.9. QDTE DE REGISTROS POR ESTADO

***SELECT*** *ESTADO,* ***Count****(****\*****)*

***FROM*** *TALUNO*

***GROUP BY*** *ESTADO;*

6.10. SOMA, MEDIA, QUANTIDADE E MEDIA COM NULOS DE DESCONTO

***SELECT Sum****(DESCONTO)* ***AS*** *SOMA,*

***Avg****(DESCONTO)* ***AS*** *MEDIA, --SOMA E DIVIDE PELA QUANTIDADE DE REGISTROS SEM OS NULOS*

***Count****(DESCONTO)* ***AS*** *QUANTIDADE,*

***Round****(****Avg****(****Nvl****(DESCONTO,0)),2)* ***AS*** *MEDIA\_ARREDOND --SOMA E DIVIDE PELA QUANTIDADE DE REGISTROS COMOS NULOS*

***FROM*** *TCONTRATO;*

6.11. TOTAL DE CONTRATO POR ESTADO E DATA

***SELECT*** *ALU.ESTADO,* ***Trunc****(CON.DATA)* ***AS*** *DATA,*

***Sum****(CON.TOTAL) TOTAL,* ***Count****(****\*****) QTDE*

***FROM*** *TALUNO ALU, TCONTRATO CON*

***WHERE*** *ALU.COD\_ALUNO* ***=*** *CON.COD\_ALUNO --UNIÃO*

***GROUP BY*** *ALU.ESTADO,* ***Trunc****(CON.DATA)*

***ORDER BY*** *ALU.ESTADO, DATA* ***DESC****;*

6.12. TOTAL DE CONTRATO POR ESTADO

***SELECT*** *ALU.ESTADO,*

***Sum****(CON.TOTAL)* ***TOTAL****, Count(****\*****) QTDE*

***FROM*** *TALUNO ALU, TCONTRATO CON*

***WHERE*** *ALU.COD\_ALUNO* ***=*** *CON.COD\_ALUNO --UNIÃO*

***GROUP BY*** *ALU.ESTADO*

***ORDER BY*** *ALU.ESTADO;*

6.13. HAVING

*--Having - filtrar coluna com função de grupo*

*--Having é uma cláusula tipo where usado para função de grupos*

***SELECT*** *COD\_ALUNO,* ***Round****(****Avg****(TOTAL)) MEDIA*

***FROM*** *TCONTRATO*

***WHERE*** *COD\_ALUNO* ***>*** *0*

***HAVING Avg****(TOTAL)* ***>*** *500 --PODE SER USADO ANTES DO GROUP BY*

***GROUP BY*** *COD\_ALUNO*

***ORDER BY*** *COD\_ALUNO;*

*--Having antes do where*

***SELECT*** *COD\_ALUNO,* ***Round****(****Avg****(TOTAL)) MEDIA*

***FROM*** *TCONTRATO*

***WHERE*** *COD\_ALUNO* ***>*** *0*

***GROUP BY*** *COD\_ALUNO*

***HAVING Avg****(TOTAL)* ***>*** *500 --PODE SER USADO DEPOIS DO GROUP BY*

***ORDER BY*** *COD\_ALUNO;*

6.14. MEDIA MAIS ALTA POR ALUNO

***SELECT Max****(****Avg****(TOTAL))*

***FROM*** *TCONTRATO*

***GROUP BY*** *COD\_ALUNO;*

6.15. SOMA DOS SALARIOS POR ESTADO

***SELECT*** *ESTADO,* ***Sum****(SALARIO)* ***AS*** *TOTAL*

***FROM*** *TALUNO*

***GROUP BY*** *ESTADO*

***ORDER BY*** *TOTAL DESC;*

**CAPÍTULO 7 – Introdução as Sub-consultas**

***SELECT*** *COD\_CONTRATO, DATA, TOTAL*

***FROM*** *TCONTRATO*

***WHERE*** *TOTAL <=*

*(****SELECT*** *VALOR* ***FROM*** *TCURSO*

***WHERE*** *COD\_CURSO* ***=*** *1);*

7.1. Errado (Só pode retornar 1 Linha na subconsulta)

***SELECT*** *COD\_CONTRATO, DATA, TOTAL*

***FROM*** *TCONTRATO*

***WHERE*** *TOTAL* ***>=***

*(****SELECT*** *VALOR* ***FROM*** *TCURSO*

***WHERE*** *VALOR* ***>*** *500);*

7.2. Consultar os Alunos da Mesma cidade do Aluno 1 Menos o Aluno 1

***SELECT*** *COD\_ALUNO, NOME, CIDADE*

***FROM*** *TALUNO*

***WHERE*** *CIDADE* ***=***

*(****SELECT*** *CIDADE* ***FROM*** *TALUNO*

***WHERE*** *COD\_ALUNO* ***=*** *1)*

***AND*** *COD\_ALUNO* ***<> 1****;*

7.3. Ao comparar colunas em subconsulta, devemos usar a mesma quantidade de coluna consultaremos a cidade e o estado nas duas tabelas, menos o aluno de codigo 1

***SELECT*** *COD\_ALUNO, NOME, CIDADE, ESTADO*

***FROM*** *TALUNO*

***WHERE*** *(CIDADE,ESTADO)* ***=***

*(****SELECT*** *CIDADE, ESTADO* ***FROM*** *TALUNO*

***WHERE*** *COD\_ALUNO* ***= 1****)*

***AND*** *COD\_ALUNO* ***<> 1****;*

7.4. Consultaremos a cidade e o estado e o cep, porém não retornar resultado pois nenhuma tem o mesmo cep, menos o aluno de codigo 1

***SELECT*** *COD\_ALUNO, NOME, CIDADE, ESTADO*

***FROM*** *TALUNO*

***WHERE*** *(CIDADE,ESTADO, CEP)* ***=***

*(****SELECT*** *CIDADE, ESTADO, CEP* ***FROM*** *TALUNO*

***WHERE*** *COD\_ALUNO* ***= 1****)*

***AND*** *COD\_ALUNO* ***<> 1****;*

7.5. USANDO HAVING EM SUBCONSULTA PARA FUNÇAO

***SELECT*** *COD\_CURSO,* ***Min****(VALOR),* ***Sum****(VALOR),* ***Count****(****\*****) QTDE*

***FROM*** *TITEM*

***WHERE*** *COD\_CURSO* ***> 0***

***GROUP******BY*** *COD\_CURSO*

***HAVING******Min****(VALOR)* ***>=*** *--ESSE CRITÉRIO*

*(****SELECT******Avg****(VALOR)* ***FROM*** *TCURSO) --DEPENDE DESSE CRITÉRIO E VICE-VERSA.*

***ORDER******BY*** *COD\_CURSO;*

7.6. Somar o total de contrato por aluno e motrar somente cujo o menor contrato seja maior que o valor do custo médio

***SELECT*** *COD\_ALUNO,* ***Min****(****TOTAL****),* ***Sum****(****TOTAL****)*

***FROM*** *TCONTRATO*

***GROUP******BY*** *COD\_ALUNO*

***HAVING******Min****(TOTAL) >*

*(****SELECT******Avg****(VALOR)* ***FROM*** *TCURSO);*

7.7. Todos os cursos que estão na tabela de item (vendidos)

***SELECT*** *COD\_CURSO, NOME, VALOR*

***FROM*** *TCURSO*

***WHERE*** *COD\_CURSO* ***IN***

*(****SELECT*** *COD\_CURSO* ***FROM*** *TITEM);*

7.8. Todos os cursos que não estão na tabela de item (não vendidos) agrupando os cursos

***SELECT*** *COD\_CURSO, NOME, VALOR*

***FROM*** *TCURSO*

***WHERE*** *COD\_CURSO* ***IN***

*(****SELECT*** *COD\_CURSO* ***FROM*** *TITEM)*

7.9. Código equivalente a subselect (se os valores sao conhecidos) - não é a maneira eficiente ou mais correta.

***SELECT*** *COD\_CURSO, NOME, VALOR*

***FROM*** *TCURSO* ***WHERE*** *COD\_CURSO* ***IN*** *(1,2,3,4);*

OU:

***SELECT*** *COD\_CURSO, NOME, VALOR*

***FROM*** *TCURSO*

***WHERE*** *COD\_CURSO* ***=*** *1* ***OR*** *COD\_CURSO* ***=*** *2* ***OR*** *COD\_CURSO* ***=*** *3;*

7.10. Todos os cursos que foram vendidos, pelo valor padrão.

***SELECT*** *\** ***FROM*** *TITEM*

***WHERE*** *(COD\_CURSO, VALOR)* ***IN***

*(****SELECT*** *COD\_CURSO, VALOR* ***FROM*** *TCURSO);*

7.11. Subconsulta na cláusula from

***SELECT*** *ITE.COD\_CONTRATO, ITE.VALOR,*

*ITE.COD\_CURSO,*

*CUR.COD\_CURSO* ***AS*** *CODIGO, CUR.VALOR*

***FROM*** *TITEM ITE,*

*(****SELECT*** *COD\_CURSO, VALOR --SEGUNDA TABELA É UMA SUBCONSULTA*

***FROM*** *TCURSO* ***WHERE*** *VALOR > 500) CUR*

***WHERE*** *CUR.COD\_CURSO* ***=*** *ITE.COD\_CURSO;*

**CAPÍTULO 8 – Manipulando Dados (Insert – Update - Delete)**

1. Objetivos:

* Descrever cada comando DML
* Inserir linhas em uma tabela
* Atualizar linhas de uma tabela
* Remover linhas de uma tabela
* Controlar transações

1. Linguagem de Manipulação de Dados

* Um comando DML é executado quando você:

- Adiciona novas linhas para uma tabela

- Modifica linhas existentes em uma tabela

- Remove linhas de uma tabela

Uma ***transação*** consiste de um conjunto de comandos DML que formam uma unidade lógica de trabalho.

1. Comando INSERT

***INSERT INTO table [(column [, column...])]***

***VALUES (value [, value...]);***

* Adicione novas linhas para uma tabela utilizando o comando INSERT.
* Somente uma linha é inserida de cada vez com esta sintaxe.

8.1. CAPITULO 8 - DML

* Fazer copia dos dados de uma tabela existente.

***CREATE*** *TABLE TDESCONTO2*

***AS*** *SELECT* ***\**** *FROM TDESCONTO;*

* FAZER COPIA DE UMA TABELA - APENAS A ESTRUTURA.

***CREATE******TABLE*** *TDESCONTO2*

***AS******SELECT*** *\** ***FROM*** *TDESCONTO* ***WHERE******ROWID IS NULL****;*

***SELECT*** *\** ***FROM*** *TDESCONTO2;*

***INSERT INTO*** *TDESCONTO (CLASSE, INFERIOR, SUPERIOR)*

***VALUES*** *('&cla',&inf,&sup);*

***SELECT*** *\** ***FROM*** *TDESCONTO*

***WHERE*** *CLASSE = '&cla';*

***UPDATE*** *TDESCONTO* ***SET***

*INFERIOR = &inf,*

*SUPERIOR = &sup*

***WHERE*** *CLASSE = '&cla';*

***DELETE FROM*** *TDESCONTO*

***WHERE*** *CLASSE* ***=*** *'&cla';*

***DELETE******FROM*** *TDESCONTO2;*

* TRANSIÇÃO (COMMIT/ROLLBACK)

***ROLLBACK****; --SOMENTE PARA INSERT, UPDATE E DELETE*

***COMMIT****;*

*--DELETA TODOS OS REGISTRO DA TABELA*

*--NÃO TEM CLÁUSULA WHERE*

***TRUNCATE******TABLE*** *TDESCONTO2; --NÃO GERA LOG, É COMO SHIFT + DELETE*

***SELECT******\*******FROM*** *TDESCONTO2;*

* SAVEPOINT

Serve para criar um ponto de restauração.

*SAVEPOINT upd\_b;*

***UPDATE*** *TDESCONTO* ***SET***

*SUPERIOR* ***=*** *88*

***WHERE*** *CLASSE* ***=*** *'B';*

***SAVEPOINT*** *upd\_a;*

***UPDATE*** *TDESCONTO* ***SET***

*SUPERIOR = 88*

***WHERE*** *CLASSE* ***=*** *'A';*

*--PONTO DE RESTAURAÇÃO*

***SAVEPOINT*** *ins\_ok;*

***INSERT******INTO*** *TDESCONTO (CLASSE, INFERIOR, SUPERIOR)*

***VALUES*** *('&cla', &inf, &sup);*

***ROLLBACK TO SAVEPOINT*** *upd\_b;*

***ROLLBACK TO SAVEPOINT*** *upd\_a;*

***ROLLBACK TO SAVEPOINT*** *ins\_ok;*

***SELECT******\*******FROM*** *TDESCONTO;*

**CAPÍTULO 9 – Criando e Gerenciando Tabelas**

Objetivos:

* Descrever os principais objetos do banco de dados;
* Criar Tabelas;
* Descrever os tipos de dados que podem ser utilizados na definição de colunas;
* Alterar a definição de tabelas;
* Remover, renomear e truncar tabelas

9.1. Convenções de nomes

* Devem começar com uma letra
* Podem possuir de 1 até 30 caracteres de tamanho
* Devem conter somente A-Z, a-z, 0-9,\_,$, e #
* Não pode possuir o mesmo nome do outro objeto criado pelo mesmo usuário
* Não pode ser uma palavra reservada do Servidor Oracle

9.2. Comando CREATE TABLE

**CREATE TABLE [schema.] table**

**(column datatype [DEFAULT expr1]);**

Você especifica:

* Nome da tabela;
* Nome da coluna, tipo de dado e tamanho da coluna;

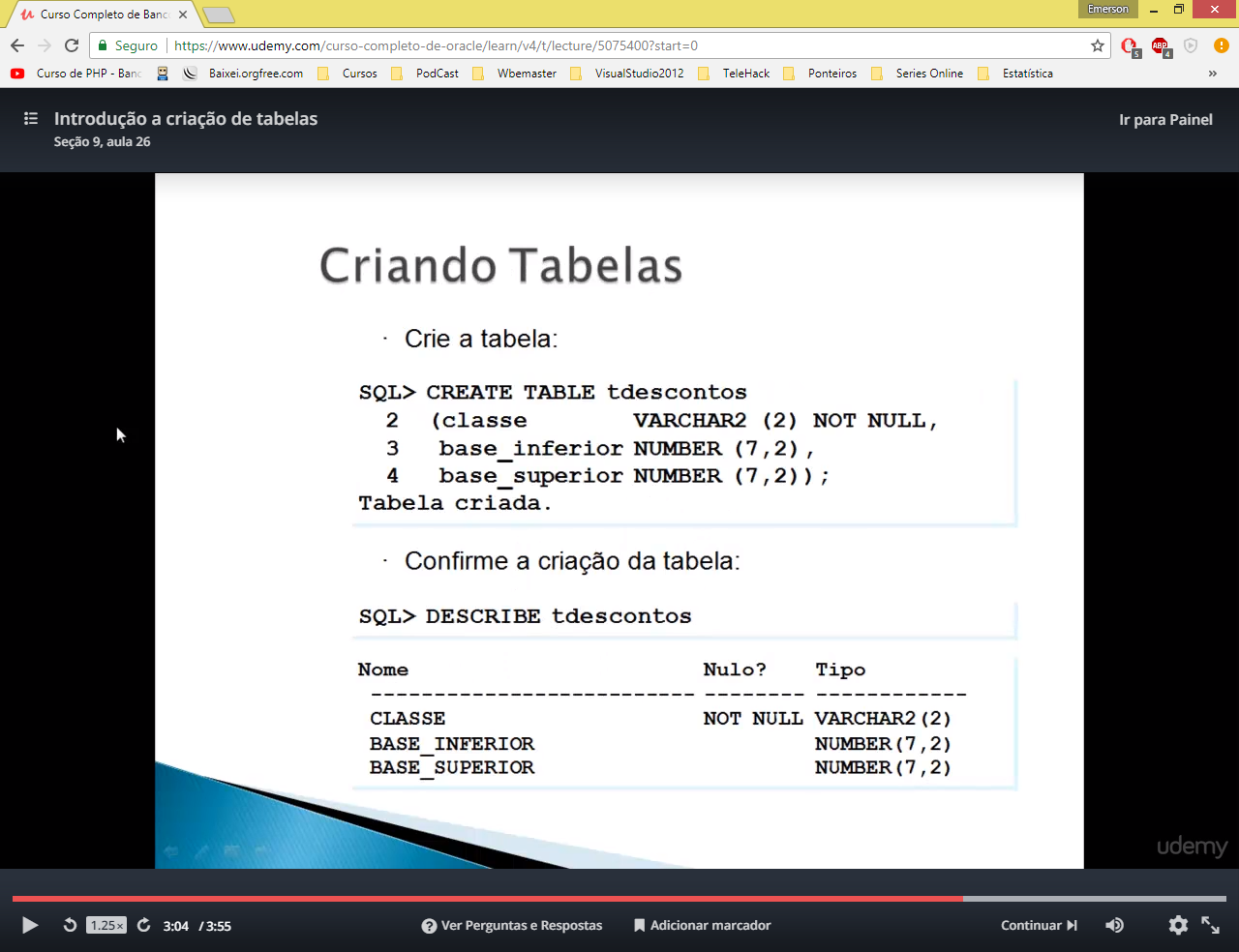
9.3. Opção DEFAULT

... dt\_compra DATE DEFAULT SYSDATE, ...

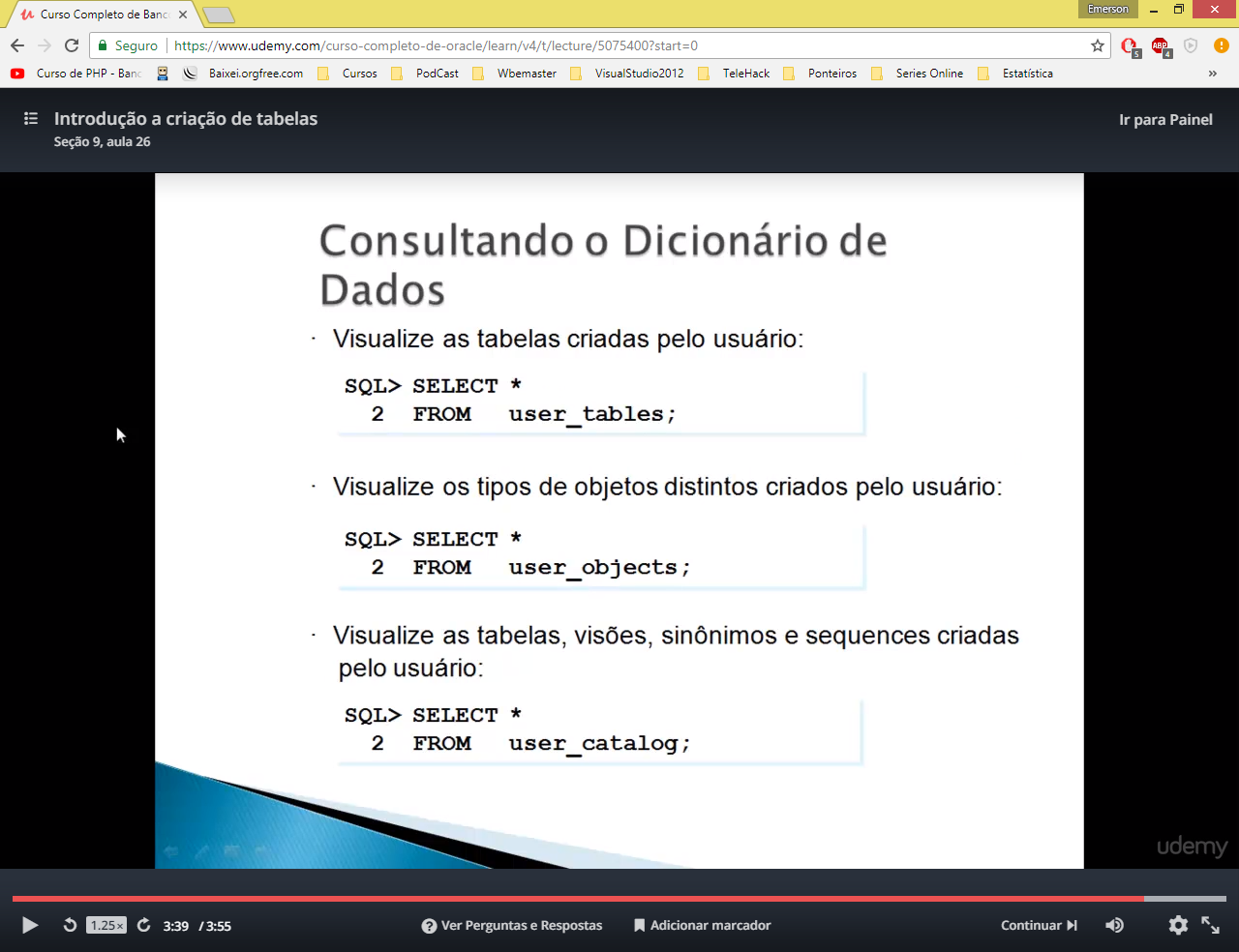
* Valores válidos são valores literais, expressões ou funções SQL
* Valores ilegais são nomes de outras colunas ou pseudocolunas
* O tipo de dado da opção default deve corresponder ao tipo de dado da coluna

9.4. Criando Tabelas

Crie a tabela:



9.5. Consultando o Dicionário de Dados



9.6. Criando, Renomeando e Apagando Colunas e Tabelas:

***CREATE TABLE*** *TESTE*

*(*

*CODIGO INTEGER* ***NOT NULL PRIMARY KEY****,*

*DATA* ***DATE******DEFAULT SYSDATE***

*);*

***INSERT INTO*** *TESTE (CODIGO)* ***VALUES****(1);*

***INSERT INTO*** *TESTE (CODIGO)* ***VALUES****(2);*

***SELECT******\*******FROM*** *TESTE;*

9.7. TABELAS CRIADAS PELO USUARIO - USER\_TABLES (View)

***SELECT \*******FROM*** *USER\_TABLES;*

***SELECT \* FROM*** *ALL\_TABLES;*

9.8. CRIAR UMA TABELAS A PARTIR DE UMA OUTRA TABELA DE DE UM RESULTADO

***CREATE TABLE*** *TCONTRATO\_VIP*

***AS***

***SELECT \* FROM*** *TCONTRATO WHERE TOTAL* ***>*** *500;*

***SELECT \* FROM*** *TCONTRATO\_VIP;*

9.9. ADD COLUNA NA TABELA

***ALTER TABLE*** *TCONTRATO\_VIP* ***ADD*** *VALOR* ***NUMBER****(5,2); -- 999,99*

9.10. ALTERAR COLUNA

***ALTER TABLE*** *TCONTRATO\_VIP* ***MODIFY*** *VALOR* ***NUMBER****(8,2); -- 999999,99*

9.11. ALTERAR COLUNA

***ALTER TABLE*** *TCONTRATO\_VIP* ***MODIFY*** *VALOR* ***NUMBER****(12,2)* ***DEFAULT*** *0;*

9.12. RENOMEAR COLUNA

***ALTER TABLE*** *TCONTRATO\_VIP* ***RENAME COLUMN*** *VALOR* ***TO*** *VALOR2;*

9.13. EXCLUIR COLUNA

***ALTER TABLE*** *TCONTRATO\_VIP* ***DROP COLUMN*** *VALOR2;*

9.14. EXCLUIR TABELA

***DROP TABLE*** *TCONTRATO\_VIP;*

9.15. COMENTÁRIOS NA TABELA

***COMMENT ON TABLE*** *TCONTRATO* ***IS*** *'Informações de Contratos'; --É POSSIVEL VER NA ESTRUTURA DE EXPORTAÇÃO DA TABELA*

9.16. COMENTÁRIO NA COLUNA DA TABLEA

***COMMENT ON COLUMN*** *TCONTRATO.COD\_CONTRATO* ***IS*** *'Código do Contrato'; --SELECIONA A TABELA E DA F12*

***COMMENT ON COLUMN*** *TCONTRATO.DATA* ***IS*** *'Data de emissão do Contrato';*

***SELECT******\* FROM*** *USER\_COL\_COMMENTS* ***WHERE******TABLE\_NAME******=*** *'TCONTRATO';*

***SELECT******\* FROM*** *USER\_TAB\_COMMENTS* ***WHERE******TABLE\_NAME******=*** *'TCONTRATO';*

9.17. DESABILITA A COLUNA

***ALTER TABLE*** *TCONTRATO* ***ADD*** *TOTAL2* ***NUMERIC****(8,2);*

***ALTER TABLE*** *TCONTRATO* ***SET UNUSED*** *(TOTAL2); --NÃO TEM COMO HABILITAR, APÓS ISSO, SÓ DROPANDO A COLUNA.*

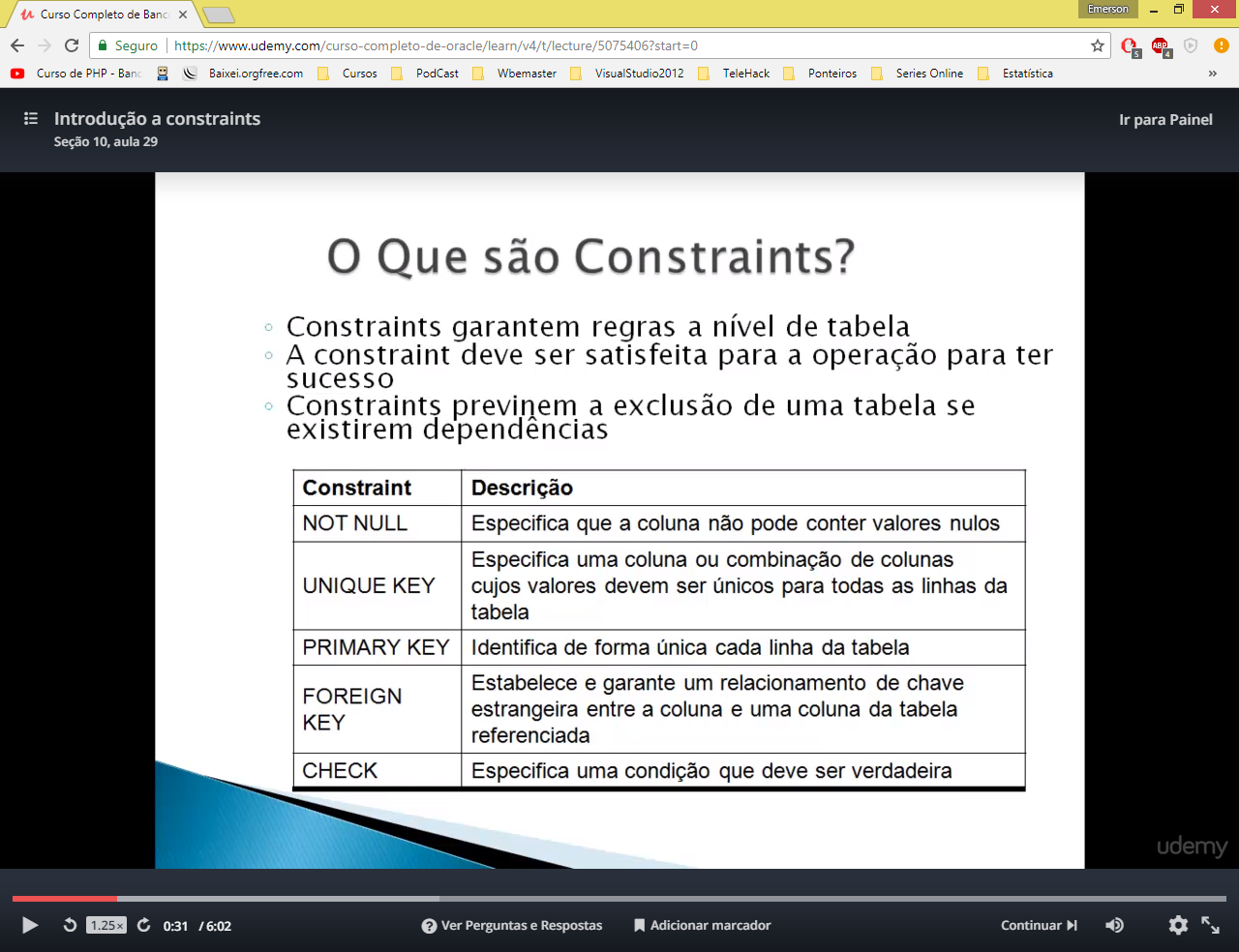
**CAPÍTULO 9 – Implementando Constraints**

9.1. Objetivo:

* Descrever constrainsts de integridade
* Criar e administrar constraints

O Que são Constraints?

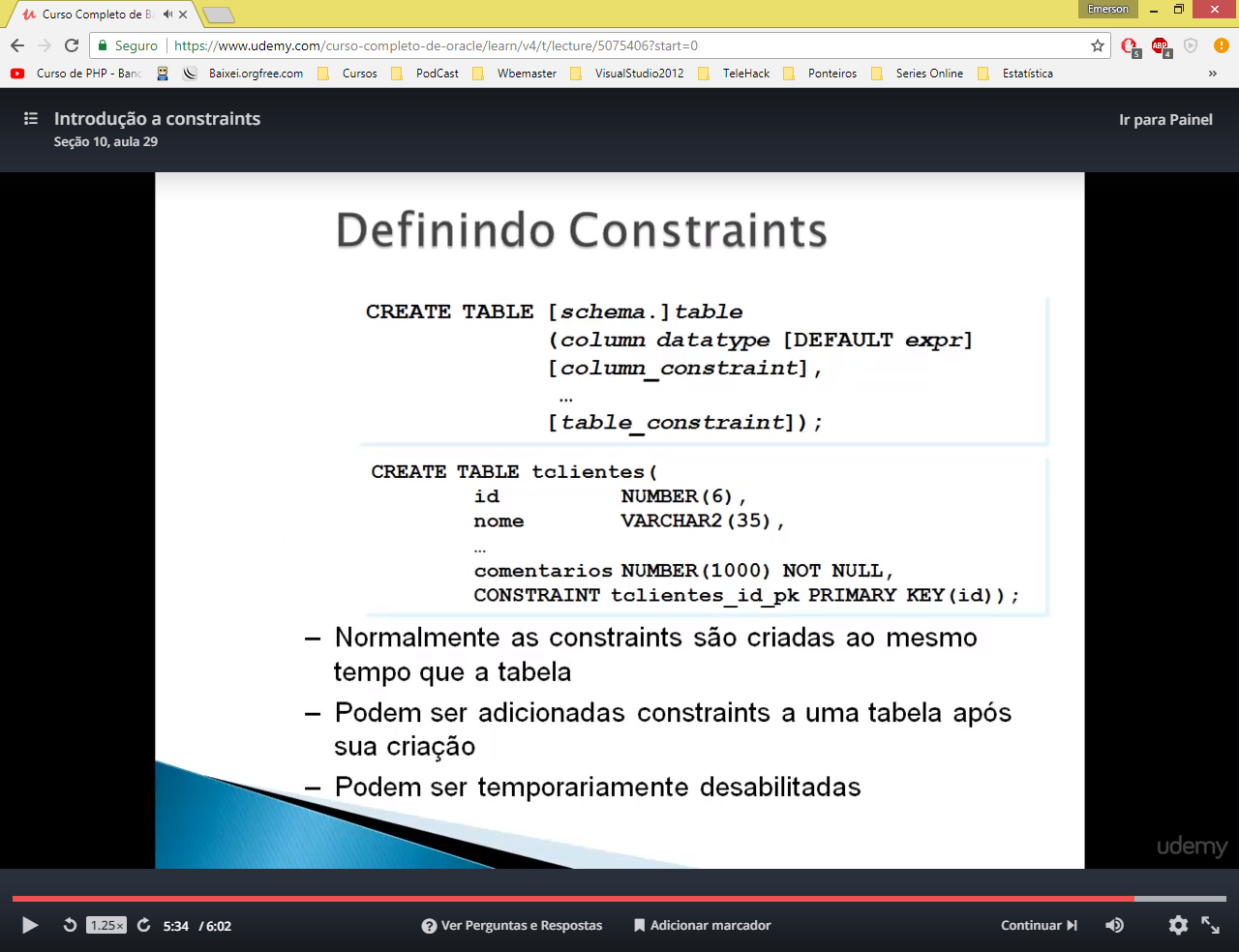
* Constraints garantem regras a nível de tabela
* A constraints deve ser satisfeita para a operação para ter acesso
* Constraints previnem a exclusão de uma tabela se existirem dependências



9.2. Diretrizes para Constraints

* Forneça um nome para a constraint ou o Servidor Oracle gerará um nome utilizando o formato SYS\_Cn
* Crie a constraint:
  + No mesmo momento em que a tabela é criada
  + Após a tabela ter sido criada
* Defina a constraint no nível da coluna ou tabela
* Visualize a Constraint no dicionário de dados – USER\_CONSTRAINTS

9.3. Definindo Constraints



9.4. CONSTRAINTS

*TALUNO*

*COD\_ALUNO - PK -> Chave Primária -> PRIMARY KEY*

*TCONTRATO*

*COD\_CONTRATO - PK -> Chave Primária -> PRIMARY KEY*

*COD\_ALUNO - FK -> Chave primária que vem de outra tabela*

9.5. SELECIONANDO CONSTRAINTS DO USUÁRIO E DO BANCO

***SELECT*** *\** ***FROM*** *USER\_CONSTRAINTS; --CONSTRAINTS DO USUÁRIO*

***SELECT*** *\** ***FROM*** *ALL\_CONSTRAINTS; --CONSTRAINTS DO BD*

***CREATE TABLE*** *TCIDADE(*

*COD\_CIDADE INTEGER* ***NOT NULL****, --PRIMARY KEY - FAREMOS NA CONSTRAINT*

*NOME* ***VARCHAR****(40),*

***CONSTRAINT*** *PK\_CIDADE* ***PRIMARY KEY****(COD\_CIDADE)*

*);*

***SELECT*** *\** ***FROM*** *TCIDADE;*

9.6. CRIAMOS UMA CHAVE PRIMÁRIA MULTIPLA

***CREATE TABLE*** *TBAIRRO(*

*COD\_CIDADE* ***INTEGER NOT NULL****,*

*COD\_BAIRRO* ***INTEGER NOT NULL****,*

*NOME* ***VARCHAR2****(40),*

***CONSTRAINT*** *PK\_BAIRRO* ***PRIMARY KEY****(COD\_CIDADE, COD\_BAIRRO)*

*);*

Exemplo:

CIDADE BAIRRO

1 1

1 2

2 1

2 2

2 2 - NÃO PODERÁ SER ACEITO POIS JÁ EXISTE ESSES VALORES NAS DUAS COLUNAS

***SELECT*** *\** ***FROM*** *TBAIRRO;*

***SELECT*** *\** ***FROM*** *TCIDADE;*

9.7. LIGANDO AS DUAS TABELAS E ADICIONANDO UMA CHAVE ESTRANGEIRA

***ALTER TABLE*** *TBAIRRO* ***ADD******CONSTRAINT*** *FK\_COD\_CIDADE --ALTERAR A TABELA FILHA*

***FOREIGN KEY*** *(COD\_CIDADE) --COD\_CIDADE DA TABELA TBAIRRO*

***REFERENCES*** *TCIDADE(COD\_CIDADE); --FAZ REFERÊNCIA A COD\_CIDADE DA TABELA TCIDADE.*

Obs.: Pode fazer essa constraint junto ou separado, como fizemos.

9.8. Tabela TRUA

***CREATE TABLE*** *TRUA(*

*COD\_RUA* ***INTEGER NOT NULL****,*

*COD\_CIDADE* ***INTEGER****,*

*COD\_BAIRRO* ***INTEGER****,*

*NOME* ***VARCHAR****(40),*

***CONSTRAINT*** *PK\_RUA PRIMARY* ***KEY****(COD\_RUA)*

*);*

***SELECT*** *\** ***FROM*** *TRUA;*

9.9. Restringindo cod\_cidade e cod\_bairro (Foreign Key)

***ALTER TABLE*** *TRUA* ***ADD CONSTRAINT*** *FK\_CIDADEBAIRRO*

***FOREIGN KEY****(COD\_CIDADE, COD\_BAIRRO)*

***REFERENCES*** *TBAIRRO(COD\_CIDADE, COD\_BAIRRO);*

Obs.: Qualquer tabela que seja ligada a tbairro, tem que ser de duas colunas, pois a tbairro possui uma chave primária de duas colunas

9.10. Criando tabela TPESSOA

***CREATE TABLE*** *TPESSOA(*

*COD\_PESSOA* ***INTEGER NOT NULL****,*

*TIPO* ***VARCHAR2****(1)* ***NOT NULL****,*

*NOME* ***VARCHAR2****(30)* ***NOT NULL****,*

*PESSOA* ***VARCHAR2****(1)* ***NOT NULL****,*

*COD\_RUA* ***INTEGER NOT NULL****,*

*CPF* ***VARCHAR2****(15),*

***CONSTRAINT*** *PK\_PESSOA* ***PRIMARY KEY*** *(COD\_PESSOA)*

*);*

* 1. Chave Única (UNIQUE KEY)

***ALTER TABLE*** *TPESSOA* ***ADD CONSTRAINT*** *UK\_CPF UNIQUE(CPF); --RESTRINÇÃO PARA DENTRO DA TABELA*

* 1. Chave Estrangeira para cod\_rua

***ALTER TABLE*** *TPESSOA* ***ADD CONSTRAINT*** *FK\_PESSOA\_RUA --RESTRINÇÃO QUE SE REFERE A VALORES FORA DA TABELA*

***FOREIGN KEY*** *(COD\_RUA)*

***REFERENCES*** *TRUA; --NÃO PRECISA ESPECIFICAR CAMPO, POIS SO PODE FAZER REFERENCIA AO CAMPO COM CHAVE PRIMARIA (DARIA ERRO SE NÃO TIVESSE CHAVE PRIMARIA NA TABELA PAI)*

9.13. Inserindo dados nas Tabelas

*--CIDADE*

***INSERT INTO*** *TCIDADE* ***VALUES****(1,'NOVO HAMBURGO');*

***INSERT INTO*** *TCIDADE* ***VALUES****(2,'IVOTI');*

***INSERT INTO*** *TCIDADE* ***VALUES****(3,'SAPIRANGA');*

***INSERT INTO*** *TCIDADE* ***VALUES****(4,'TAQUARA');*

***SELECT*** *\** ***FROM*** *TCIDADE;*

*--BAIRRO*

***INSERT INTO*** *TBAIRRO* ***VALUES****(1,1,'CENTRO');*

***INSERT INTO*** *TBAIRRO* ***VALUES****(2,1,'RIO BRANCO');*

***INSERT INTO*** *TBAIRRO* ***VALUES****(3,1,'CENTRO');*

***INSERT INTO*** *TBAIRRO* ***VALUES****(4,1,'FRITZ');*

***INSERT INTO*** *TBAIRRO* ***VALUES****(5,1,'AMARAL'); --FK NAO DEIXA INSERIR FILHO SEM PAI*

***INSERT INTO*** *TBAIRRO* ***VALUES****(6,1,'EMPRESA');*

*--SE TENTAR EXCLUIR CIDADE, NAO VAI DEIXAR, POIS EXISTE FILHO AMARRADO A CIDADE*

***SELECT*** *\** ***FROM*** *TBAIRRO;*

*--RUA*

***INSERT INTO*** *TRUA* ***VALUES****(1,1,1,'MARCILIO DIAS');*

***INSERT INTO*** *TRUA* ***VALUES****(2,2,1,'FRITZ');*

***INSERT INTO*** *TRUA* ***VALUES****(3,3,1,'JACOBINA');*

***INSERT INTO*** *TRUA* ***VALUES****(4,3,1,'JOAO DA SILVA');*

***SELECT*** *\** ***FROM*** *TRUA;*

*--INSERIR PESSOAS*

***INSERT INTO*** *TPESSOA* ***VALUES****(1,'C','MARCIO','F',1,'1234');*

***INSERT INTO*** *TPESSOA* ***VALUES****(2,'F','BEATRIZ','F',2,'123');*

***INSERT INTO*** *TPESSOA* ***VALUES****(3,'F','PEDRO','F',4,'12345');*

***INSERT INTO*** *TPESSOA* ***VALUES****(4,'C','MARIA','J',4,'123456');*

***SELECT*** *\** ***FROM*** *TPESSOA;*

9.15. TIPO DE RESTRIÇÃO CHECK

***ALTER TABLE*** *TPESSOA* ***ADD CONSTRAINT*** *CK\_PESSOA\_TIPO*

***CHECK*** *(TIPO* ***IN****('C','F'));*

***ALTER TABLE*** *TPESSOA* ***ADD CONSTRAINT*** *CK\_PESSOA\_JS*

***CHECK*** *(PESSOA IN('J','F'));*

9.16. EXCLUIR CONSTRAINT

***ALTER TABLE******DROP******CONSTRAINT*** *NOME\_CONSTRAINT;*

9.17. APAGAR UMA CONSTRAINT

***ALTER TABLE*** *TPESSOA* ***DROP CONSTRAINT*** *NOME\_DA\_CONSTRAINT*

***CASCADE CONSTRAINT****; --CASO DE APAGAR CONSTRAINT RELACIONADAS COM OUTRAS TABELAS.*

9.18. CHECK – LIMITANDO VALOR

***ALTER TABLE*** *TCONTRATO*

***ADD CONSTRAINT*** *CK\_CONTRATO\_DESCONTO*

***CHECK*** *(DESCONTO* ***BETWEEN*** *0* ***AND*** *30);*

***SELECT*** *\** ***FROM*** *TCONTRATO;*

***UPDATE*** *TCONTRATO* ***SET*** *DESCONTO = 31* ***WHERE*** *COD\_CONTRATO = 1; --NAO PERMITE ALTERAR O DESCONTO MAIOR QUE 30.*

***UPDATE*** *TCONTRATO* ***SET*** *DESCONTO = 30* ***WHERE*** *COD\_CONTRATO = 1; --PERMITE ALTERAR O DESCONTO ATÉ 30.*

9.19. DESABILITANDO E HABILITANDO CONSTRAINT

***ALTER TABLE*** *TPESSOA* ***DISABLE*** *CONSTRAINT UK\_CPF;*

***ALTER TABLE*** *TPESSOA* ***ENABLE*** *CONSTRAINT UK\_CPF;*

9.19.1. USANDO ALT+F3

É sÉ clicar com o botão direito sobre a constraint (uks, pks, fks, chks) e selecionar habilitar ou desabilitar.

9.20. SELECIONAR AS CONSTRAINTS

***SELECT*** *\** ***FROM*** *USER\_CONSTRAINTS*

***WHERE******TABLE\_NAME*** *= 'TPESSOA';*

**Obs:**

- CONSTRAINT\_TYPE C=CHECK, P=PRIMARY KEY, U=UNIQUE KEY, R=RELACIONAMENTO.

- VERIFICAR A COLUNA CONSTRAINT\_NAME, A COLUNA TABLE\_NAME E STATUS PARA VERIFICAR A CONSTRAINT QUE ESTA ATIVADA OU DESATIVADA.

- COLUNA SEARCH\_CONDITION = MOSTRA A CONDIÇÃO DA CONSTRAINT.

9.21. CONSTRAINT E AS COLUNAS ASSOCIADAS

***SELECT******CONSTRAINT\_NAME****,* ***COLUMN\_NAME***

***FROM******USER\_CONS\_COLUMNS***

***WHERE******TABLE\_NAME*** *= 'TPESSOA';*

***SELECT OBJECT\_NAME, OBJECT\_TYPE***

***FROM USER\_OBJECTS***

***WHERE OBJECT\_NAME IN*** *('TPESSOA');*

**CAPÍTULO 10 – Criando Visões**

**1. Objetivos:**

- Descrever uma visão.

- Criar uma visão.

- Recuperar dados através de uma visão.

- Alterar a definição de uma visão.

- Inserir, Atualizar e Remover dados através de uma visão.

- Remover uma visão.

**2. O que é uma visão?**

* É uma tabela baseada em uma tabela ou outra visão.
* As tabelas nas quais uma visão é baseada são chamadas de tabelas básicas.
* A visão é armazenada como um comando SELECT no dicionário de dados.

**3. Por que criar visões?**

- Para restringir o acesso ao banco de dados

- Para tornar simples consultas complexas

- Para permitir independências de dados

- Para apresentar visões diferentes do mesmo dado

**4. As Views são divididas em dois grupos:**

**Visões Simples e Visões Complexas:**

|  |  |  |
| --- | --- | --- |
| **Características** | **Visão Simples** | **Visão Complexa** |
| Número de tabelas | Uma | Uma ou mais |
| Contém funções | Não | Sim |
| Possui grupos de dados | Não | Sim |
| DML através da visão | Sim | Talvez |

**5. Criando uma visão:**

***CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW view [(alias[, alias]...)]***

***AS subquery***

***[WITH CHECK OPTION [CONSTRAINT constraint]]***

***[WITH READ ONLY]***

* Você insere uma subconsulta dentro do comando CREATE VIEW.
* A subconsulta pode conter uma sintaxe complexa do comando SELECT.
* Você pode utilizar a opção OR REPLACE para modificar a definição da visão.

6. CRIANDO UMA VIEW

***CREATE OR REPLACE VIEW*** *V\_ALUNO*

***AS***

***SELECT*** *COD\_ALUNO* ***AS*** *CODIGO,*

*SALARIO,*

*NOME* ***AS*** *ALUNO,*

*ESTADO,*

*CIDADE*

***FROM*** *TALUNO*

***WHERE*** *ESTADO* ***=*** *'RS';*

*--SELECT QUE REPRESENTA ESSA VIEW*

6.1. USANDO A VIEW

***SELECT*** *\** ***FROM*** *V\_ALUNO*

***ORDER BY*** *ALUNO;*

6.2. VIEW CONTRATO\_TOP

***CREATE OR REPLACE VIEW*** *V\_CONTRATO\_TOP*

***AS***

***SELECT*** *COD\_CONTRATO, DESCONTO*

***FROM*** *TCONTRATO*

***WHERE*** *DESCONTO* ***>=*** *10;*

*--USANDO A VIEW CONTRATO\_TOP*

***SELECT*** *\** ***FROM*** *V\_CONTRATO\_TOP;*

6.3. NOME E CONTEÚDO DAS VIEWS

***SELECT VIEW\_NAME****,* ***TEXT***

***FROM******USER\_VIEWS****;--VIEW QUE MOSTRA OUTRAS VIEWS.*

6.4. -VIEW COM PARÂMETRO DE SAÍDA

***CREATE OR REPLACE VIEW*** *V\_ALUNO2(COD, ALUNO, SAL)*

***AS***

***SELECT*** *COD\_ALUNO, NOME, SALARIO*

***FROM*** *TALUNO;*

***SELECT*** *\** ***FROM*** *V\_ALUNO2;*

6.5. VIEW COMPLEXA

Obs.:A caracteristica dessa view que a faz ser complexa, nao que seja complexa

***CREATE OR REPLACE VIEW*** *V\_CONTRATO*

***AS***

***SELECT Trunc****(DATA)* ***AS*** *DATA,*

***Max****(DESCONTO) MAXIMO,*

***Avg****(DESCONTO) MEDIA,*

***Count****(\*) QTDE*

***FROM*** *TCONTRATO*

***GROUP BY******Trunc****(DATA);*

***SELECT*** *\** ***FROM*** *V\_CONTRATO;*

Obs:

> PODEMOS APAGAR OS SELECTS OU SCRIPTS, POIS AS VIEWS FICAM SALVAS NO BANCO (F3 - VIEWS).

> CLICAR NA VIEW DA LISTA COM O BOTÃO DIREITO EM SEGUIDA CLICA EM LOAD DLL.

> OU SELECIONA O NOME DA VIEW E CTRL+F12 OU CTRL+SHIFT+F12.

6.6. VIEW SIMPLES - CRIANDO UMA VIEW PARA LISTAR APENAS PESSOAS TIPO FÍSICAS

***CREATE OR REPLACE VIEW*** *V\_PESSOA\_F*

***AS***

***SELECT*** *COD\_PESSOA, TIPO, NOME, COD\_RUA* ***AS*** *RUA*

***FROM*** *TPESSOA*

***WHERE*** *TIPO* ***=*** *'F';*

***SELECT*** *\** ***FROM*** *V\_PESSOA\_F;*

6.7. EXEMPLO DE CONSULTA USANDO VIEW E TABELA

*--ENVOLVENDO TABELA TRUA E TCIDADE.*

***SELECT*** *PES.COD\_PESSOA* ***AS*** *CODIGO,*

*PES.NOME* ***AS*** *PESSOA,*

*CID.NOME* ***AS*** *CIDADE,*

*RUA.NOME* ***AS*** *RUA*

***FROM*** *V\_PESSOA\_F PES, TRUA RUA, TCIDADE CID*

***WHERE*** *PES.RUA* ***=*** *RUA.COD\_RUA(+)*

***AND*** *CID.COD\_CIDADE* ***=*** *RUA.COD\_CIDADE*

***ORDER BY*** *PES.NOME;*

6.8. OPERAÇÃO DML NA VIEW

*--INSERIR REGISTROS COM VALORES = 1000*

***CREATE OR REPLACE VIEW*** *VCURSOS1000CK*

***AS***

***SELECT*** *COD\_CURSO, NOME, VALOR*

***FROM*** *TCURSO*

***WHERE*** *VALOR = 1000*

***WITH CHECK OPTION CONSTRAINT*** *VCURSO1000\_CK;*

*-- VALOR NAO SERA INSERIDO*

***INSERT INTO*** *VCURSOS1000CK*

*(COD\_CURSO, NOME, VALOR)*

***VALUES****(52,'TESTE Y',900);*

OBS.: ATRAVÉS DA VIEW OS DADOS SÃO INSERIDOS NA TABELA, NÃO SÃO INSERIDOS NA TABELA VIEW.

***INSERT INTO*** *VCURSOS1000CK*

*(COD\_CURSO, NOME, VALOR)*

***VALUES****(52,'TESTE Y',1000);*

***SELECT \* FROM*** *TCURSO;*

6.9. DELETE VIEW

OBS.: OS DADOS SERÃO EXCLUÍDOS NA TABELA TALUNO E NA TABELA VIEW\_ALUNO

***DELETE FROM*** *V\_ALUNO* ***WHERE*** *CODIGO* ***=*** *3;*

***SELECT \* FROM*** *V\_ALUNO;*

***SELECT \* FROM*** *TALUNO;*

6.10. INSERT EM VIEW

***INSERT INTO*** *V\_ALUNO*

***VALUES****(50,500,'MARIA','RS','NH');*

6.11. DELETE UMA VIEW

*--NÃO É PERMITIDO DELETAR UMA VIEW COMPLEXA, APENAS VIEW SIMPLES*

***DELETE FROM*** *V\_CONTRATO;*

6.12. VIEW SOMENTE LEITURA (NÃO PERMITE DML)- PORQUE USAREMOS A CLÁUSULA WITH COM READ ONLY

***CREATE OR REPLACE VIEW*** *V\_ALUNO3*

***AS***

***SELECT*** *COD\_ALUNO CODIGO,*

*NOME ALUNO, CIDADE*

***FROM*** *TALUNO*

***WHERE*** *ESTADO* ***=*** *'RS'*

***WITH*** *READ ONLY;*

6.13. NÃO PODE EXECUTAR DELETE EM VIEW SOMENTE LEITURA

***DELETE FROM*** *V\_ALUNO3;*

6.14. EXCLUINDO VISÃO

***DROP VIEW*** *V\_ALUNO3;*

**CAPÍTULO 11 – OUTROS OBJETOS DO BANCO**

1. OBJETIVOS

* Criar, Alterar e utilizar sequences
* Criar e alterar índices

1. O que é uma sequencia?

* Automaticamente gera números únicos
* É um objeto compartilhado
* É normalmente utilizado para criar valores para chaves primárias
* Substitui código de aplicação

1. Comando CREATE SEQUENCE

CREATE SEQUENCE sequence  
[INCREMENT BY n]  
[START WITH n]  
[{MAXVALUE n | NOMAXVALUE}]  
[{MINVALUE n | NOMINVALUE}]  
[{CYCLE | NOCYCLE}]  
[{CACHE n | NOCACHE];

4. Criando uma sequencia:

* 1. Defina uma sequencia para gerar números sequenciais automaticamente.

*SLQ>* ***CREATE SEQUENCE*** *sclientes\_id* ***INCREMENTE BY*** *1* ***START WITH*** *201* ***MAXVALUE*** *300* ***NOCACHE******NOCYCLE****;*

*Sequencia criada.*

4.2. Crie uma sequence chamada CLIENTES\_ID para ser utilizada para a chave primária da tabela TCLIENTES. Utilize a opção NOCYCLE

***CREATE SEQUENCE*** *SEQ\_ALUNO1*

***START WITH*** *60*

*--valor inicial*

***INCREMENT BY*** *5*

*--Qtde a Incrementar*

***MINVALUE*** *60*

*--Valor Mínimo*

***MAXVALUE*** *250*

*--Valor Máximo*

***NOCACHE***

*--Não guarda em cache faixa de valores -- PADRÃO É 20*

***NOCYCLE****;*

*--QUANDO CHEGAR EM 250 ELE VOLTARIA PRA 60, MAS NÃO FARÁ.*

*--USO RARO*

4.3. INSERINDO VALOR COM SEQUENCE

***INSERT INTO*** *TALUNO (COD\_ALUNO, NOME)*

***VALUES*** *(SEQ\_ALUNO1.NEXTVAL,'MASTER TRAINING');*

***INSERT INTO*** *TALUNO (COD\_ALUNO, NOME)*

***VALUES*** *(SEQ\_ALUNO1.NEXTVAL,'MASTER TRAINING 2');*

***SELECT*** *\** ***FROM*** *TALUNO;*

***SELECT \* FROM*** *USER\_SEQUENCES;*

*--EXIBE A TABELA COM SEQUENCES DO USUARIO*

4.5. VALOR ATUAL DA SEQUENCE

***SELECT*** *SEQ\_ALUNO1.****CURRVAL FROM DUAL****;*

4.6. ALTERAR SEQUENCE

***ALTER SEQUENCE*** *SEQ\_ALUNO1* ***MAXVALUE*** *500;*

***ALTER SEQUENCE*** *SEQ\_ALUNO1* ***INCREMENT BY*** *2;*

4.7. ALTERAR VALOR DA SEQUENCIA

*--APAGA*

***DROP SEQUENCE*** *SEQ\_ALUNO1;*

*--CRIA NOVAMENTE*

***CREATE SEQUENCE*** *SEQ\_ALUNO1* ***START WITH*** *80;*