Deusdeth Mariano
E-mail: deusdeth.mariano@ceub.edu.br

UniCEUB
Centro Universitário de Brasília

# SQL - DDL - Tabelas

# Conceitos da Abordagem Relacional

- Objetivos:
  - Entender e utilizar a sintaxe básica dos comandos de DDL para criar, alterar e eliminar tabelas do BD.

# Objetos do banco de dados

| Objeto | Descrição |
|---|---|
| Tabela | Unidade básica de armazenamento, composta de linhas ou tuplas. |
| Visão | Representa logicamente subconjunto de dados de uma ou mais tabelas. |
| Seqüência | Gera valores de chave primária. |
| Índice | Agiliza o processamento de algumas consultas. |
| Sinônimo | Atribui nomes alternativos a objetos. |

# Sintaxe SQL

- A sintaxe de um comando SQL se refere à estrutura utilizada para representar este comando tal como descrito no padrão SQL:1999.
- Por exemplo, a seguinte sintaxe representa a definição de um comando para criação de tabela (CREATE TABLE):

```
table definition:: =
    CREATE [(GLOBAL | LOCAL) TEMPORARY] TABLE table name
    (table Element [(, table element)...])
    [ON COMMIT (CONSERVAR | DELETE) ROWS]
```

# Nomes de objetos em ambiente SQL

- Todo objeto criado em um BD recebe um nome identificador.
- Um nome de objeto no ORACLE deve ser construído com até 30 caracteres e deve seguir algumas convenções:
  - Os nomes não são case-sensitive.
  - Somente letras, dígitos e sublinhados são permitidos.
  - Nenhuma palavra reservada para o SQL deve ser utilizada para nomear objetos.

# Recomendações para nomeações de objetos

- O nome deve ser sugestivo, identificando claramente sua finalidade. Por exemplo,se a tabela irá armazenar dados de funcionários ela deverá ser nomeada funcionario.
- O nome deve estar sempre no singular. Por exemplo, Cliente no lugar de Clientes.
- Evite usar abreviações, se necessário utilização padrões listados em um dicionário de dados.
- Não usar preposições.
- Não utilizar o nome de um outro objeto de banco de dados utilizado pelo sistema.
- Caso o nome do objeto ultrapasse 30 caracteres, abreviar os nomes partindo do menos determinante até o mais importante.

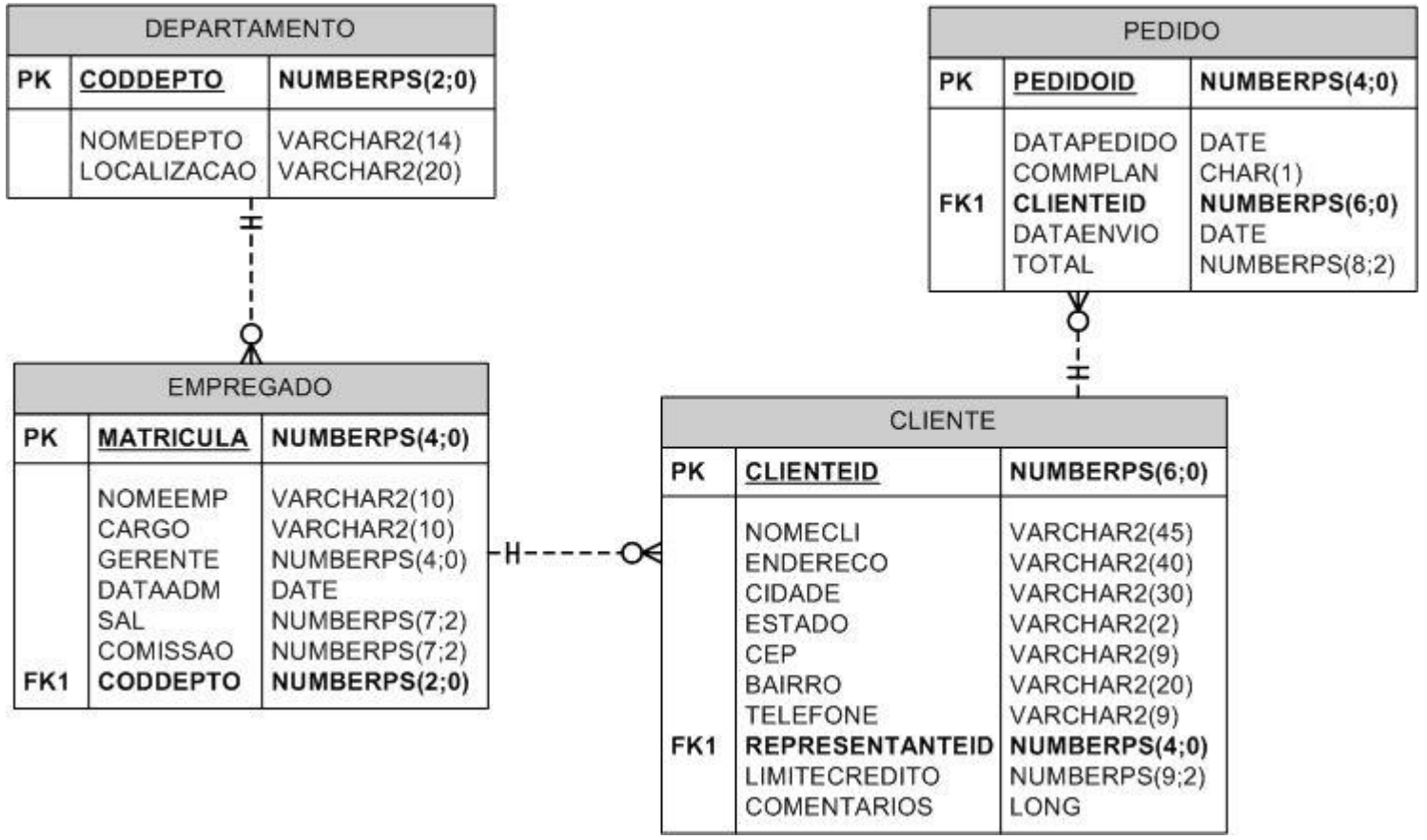
# Elementos de sintaxe

- Palavras reservadas em MAIÚSCULAS
  - Não se trata de exigência do SQL – é mais uma forma de identificá-las claramente em uma declaração.
- [ ] - indica que a sintaxe entre colchetes é opcional.

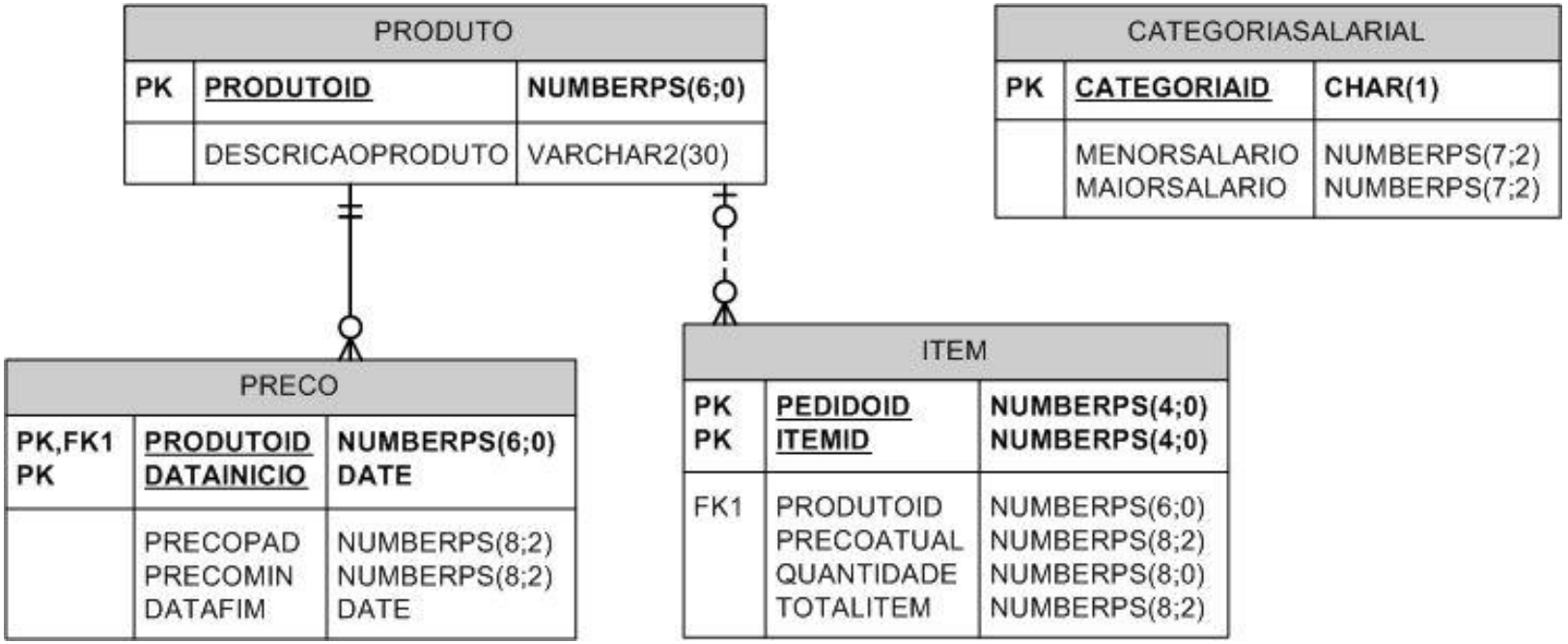
# Elementos de sintaxe

- | - pode ser lido como "ou". Entre dois elementos de sintaxe opte por um deles.
  - Em PRESERVE|DELETE. Escolha a opção PRESERVE ou DELETE.
- { } – são utilizados para agrupar elementos de sintaxe.
  - Deve-se escolher uma opção da sintaxe que se encontra entre chaves e adequá-la ao restante da definição SQL a opção escolhida.
  - Exemplo: se escolher DELETE da lista {PRESERVE | DELETE} a linha de código poderá continuar com ON COMMIT DELETE ROWS ou ON COMMIT PRESERVE ROWS, caso contrário

# Modelo das tabelas do ambiente[1]



# Modelo das tabelas do ambiente[2]



# Criando Tabelas

- Você deve ter:
  - Privilégio para criar tabelas (Atribuição RESOURCE ou CREATE TABLE).
  - Uma área de armazenamento (tablespace  com quota disponível para o usuário).

# Sintaxe de criação de tabelas

- Sintaxe:

```
CREATE TABLE [schema.]table
(column datatype [DEFAULT expr] [{,column
  datatype [,…]}] );
```

Onde,

- schema (esquema) - o nome do proprietário.
- table é o nome da tabela
- column – é o nome do campo/atributo da tabela.
- datatype – é o tipo de dados do campo.
- DEFAULT expr: especifica um valor default para o campo através de uma constante ou expressão para obter o valor.

# Tipos de Dados Oracle

| Tipo de dados | Descrição |
|---|---|
| CHAR(x) | Representa uma string de tamanho x. o valor default de x é 1. |
| VARCHAR2(X) | Representa uma string de tamanho variável. |
| CLOB | Representa strings de 1 byte até 4 gigabytes. |
| BLOB | Representa dados binários de até 4 gigabytes. Ex. Imagens. |
| NUMBER(p,s) | Dado numérico de comprimento variável, com uma precisão (p) e uma escala(s). |
| INTEGER | Para representar inteiros |
| REAL | Ponto flutuante de precisão simples |
| DOUBLE | Ponto flutuante de precisão dupla |
| DATE | Valores de data e hora. |
| RAW E LONG RAW | Dados binários brutos, não interpretados pelo banco. |
| BFILE | Dados binários armazenado em arquivo externo de até 4 gigabytes. |

# Exemplo: criando uma tabela

- Crie a tabela funcionario, com a sintaxe abaixo:

```
CREATE TABLE funcionario (
    codfuncionario  NUMBER,
    cpf             CHAR(11),
    nomefunc        VARCHAR2(40)
);
```

- Informação sobre tabelas existentes:
  - DESCRIBE tabela
  - Comando do programa SQL*Plus, não da linguagem SQL

# Alterando uma tabela

- Use o comando ALTER TABLE para:
  - Adicionar uma nova coluna.
  - Modificar uma coluna existente.
  - Definir um valor default para a nova coluna.
  - Eliminar uma coluna existente

# Sintaxe de alteração de tabela

- **Sintaxe:**

```
ALTER TABLE [schema.]table
    column_clauses,...
```

Onde,

*column_clauses*, pode ser uma das seguintes claúsulas :

```
ADD ( column datatype [DEFAULT expr])
MODIFY column datatype [DEFAULT expr]
DROP COLUMN column
SET UNUSED (column,...)
RENAME COLUMN column TO new_name
```

# Adicionando,modificando e renomeando coluna

- Use a cláusula ADD para adicionar colunas.

```
ALTER TABLE funcionario
ADD (salario   NUMBER(10,2));
```

- Modificando uma coluna:

```
ALTER TABLE funcionario
MODIFY (nomefunc VARCHAR2(60));
```

Ou

```
ALTER TABLE funcionario
RENAME COLUMN salario TO salariofunc;
```

# Eliminando uma coluna

- Utilizar a cláusula DROP COLUMN para eliminar colunas que não são mais necessárias.

```
ALTER TABLE funcionario
DROP COLUMN salariofunc;
```

- Ao invés de eliminar a coluna, pode-se colocar a mesma como "Não Usada"

```
ALTER TABLE funcionario
SET UNUSED COLUMN cpf;
```

  - Vantagem: Mais rápido que a eliminação de colunas, já que não tenta recuperar o espaço em disco, pois os dados não são removidos.
  - Pode-se eliminá-las depois com o sistema com menos atividade:

```
ALTER TABLE funcionario
DROP UNUSED COLUMNS;
```

# Eliminando uma tabela

- Sintaxe:

```
DROP TABLE [schema.]table
```

- Considerações
  - Todos os dados e estrutura de dados serão excluídos.
  - Todas as transações pendentes sofrerão *commit*.
  - Todos os índices serão eliminados.
  - Não se pode fazer *rollback* deste comando.

- Exemplo:

```
DROP TABLE funcionario
```

# O comando TRUNCATE

- O comando TRUNCATE TABLE:
  - Remove todas as linhas de uma tabela.
  - Libera o espaço de armazenamento usado por esta tabela;
- Não pode ser feito rollback da remoção da linha ao usar TRUNCATE.
- Como alternativa, você pode remover as linhas usando a instrução DELETE.

# “Limpando” uma tabela

- Verifique que existem linhas armazenadas na tabela item:

```
SELECT * FROM item;
```

- Execute um comando de truncate na tabela item:

```
TRUNCATE TABLE item;
```

- Verifique que após o comando já não existem mais linhas armazenadas na tabela item:

```
SELECT * FROM item;
```

# Usando uma subconsulta para criar uma tabela

- Sintaxe:

```
CREATE TABLE [schema.]table
[(column, column,…)]
as subconsulta;
```

- Cria uma tabela e carregar os dados a partir de outra tabela, através da opção de subconsulta.
- Faz a correspondência do número de colunas especificadas com o número de colunas da subconsulta.
- Define colunas com nomes de colunas da subconsulta e valores default.

# Criando uma tabela usando uma subconsulta

- Criando uma tabela com toda a estrutura e valores de outra

```
CREATE TABLE emp2
   AS SELECT * FROM empregado;
```

- Criando uma tabela com alguns atributos de outra tabela

```
CREATE TABLE emp3 (numemp, nomeemp)
AS SELECT matricula, nomeemp
FROM empregado;
```

# Incluindo restrições

- As restrições impõem regras no nível da tabela.
- As restrições evitam que uma tabela seja apagada se houver dependências.
- Os seguintes tipos de restrições são válidos no oracle:
  - NOT NULL : a coluna não pode ser nula.
  - UNIQUE: todos os valores de uma linha deve ser único.
  - PRIMARY KEY: identifica a coluna chave da tabela.
  - FOREIGN KEY: especifica que a coluna é chave primária em outra tabela.
  - CHECK: especifica que uma determinada condição deve ser verdadeira.

# Diretrizes sobre restrições

- Nomeie uma restrição ou o servidor Oracle gerará um nome usando o formato SYS_Cn.
  - n é um número inteiro para criar um nome de restrição exclusivo.
- Cria-se a restrição:
  - No momento que a tabela é criada.
    - No nível da coluna (restrição em linha)
    - No nível da tabela (restrição fora de linha).
  - Depois que a tabela tiver sido criada.

# Definindo restrições na criação da tabela

- Sintaxe:

```
CREATE TABLE [schema.]table
((column datatype [DEFAULT expr]
                  [column_constraint]
   [,column datatype [,…]] )
 [table_constraint] [,...]);
```

  - Onde:
    - column_constraint: é uma restrição de integridade como parte da definição da coluna.
    - table_constraint : é uma restrição de integridade como parte da definição da tabela.

# Definindo restrições em nível de restrição de coluna

- Sintaxe de column_constraint :

```
column datatype][CONSTRAINT constraint]
[NOT] NULL
|PRIMARY KEY
|UNIQUE
|REFERENCES [schema.]table [(column)] [ON DELETE
  CASCADE]
|CHECK (condition)
[ENABLE|DISABLE]
```

Onde,

*constraint* - nome dado a restrição.

DISABLE - desativa a restrição de integridade.

# Definindo restrições em nível de restrição de coluna

- Exemplo usando column_constraint:

```
CREATE TABLE empregado (
   matricula  NUMBER(4)
      CONSTRAINT empregado_matricula_pk
         PRIMARY KEY,
   nomeemp    VARCHAR2(10),
   cod_depto  NUMBER(2) NOT NULL);
```

- Não é possível utilizar a claúsula column_constraint para adicionar uma restrição sobre duas ou mais colunas, neste caso utiliza-se a claúsula table_constraint.

# Definindo restrições em nível de restrição de tabela

- Sintaxe de table_constraint :

```
[CONSTRAINT constraint]
|PRIMARY KEY (column [, column] ...)
|UNIQUE (column [, column] ...)
|FOREIGN KEY (column [, column] ...)
   REFERENCES[schema.]table (column [,
  column] ...)
   [ON DELETE CASCADE]
|CHECK (condition)
[ENABLE|DISABLE]
```

# Definindo restrições na criação da tabela (5)

- Exemplo:

```
CREATE TABLE empregado (
    matricula NUMBER(4),
    nomeemp   VARCHAR2(10),
    coddepto  NUMBER(2) NOT NULL,
CONSTRAINT empregado_matricula_pk
     PRIMARY KEY(matricula));
```

# A restrição NOT NULL

- Assegura que valores nulos não sejam permitidos para a coluna.

```
matricula       NUMBER(4),
nomefunc    VARCHAR2(10) NOT NULL,
cargo           VARCHAR2(9),
matgerente      NUMBER(4),
dataadmissao       DATE,
salario         NUMBER(7,2),
comissao        NUMBER(7,2),
coddepto           NUMBER(7,2) NOT NULL
```

# A restrição PRIMARY KEY

- Cria uma chave primária para a tabela. E somente uma chave primária pode ser criada para cada tabela.

```
CREATE TABLE departamento (
    coddepto      NUMBER(2),
    nomedepto     VARCHAR2(14),
    locacao       VARCHAR2(13),
    CONSTRAINT departamento_coddepto_pk
                  PRIMARY KEY(coddepto);
```

# A restrição UNIQUE KEY

- Requer que cada valor em uma coluna ou conjunto de colunas seja exclusivo.
- Pode ser definida no nível da coluna ou da tabela.

```
CREATE TABLE departamento (
   coddepto      NUMBER(2),
   nomedepto     VARCHAR2(14),
   locacao       VARCHAR2(13),
   CONSTRAINT departamento_nomedepto_uk
           UNIQUE(nomedepto),
   CONSTRAINT departamento_coddepto_pk
           PRIMARY KEY(coddepto));
```

# A restrição FOREIGN KEY

- Restrição de integridade referencial, designa uma coluna ou combinação de colunas como sendo chave primária em outra tabela.

```
CREATE TABLE empregado (
   matricula     NUMBER(4) NOT NULL,
   nomeemp    VARCHAR2(10),
   coddeptoNUMBER(2) NOT NULL,
   CONSTRAINT empregado_matricula_pk
      PRIMARY KEY (matricula),
   CONSTRAINT empregado_coddepto_fk
      FOREIGN KEY (coddepto)
      REFERENCES DEPT (coddepto));
```

# Palavras-chave da restrição FOREIGN KEY

- FOREIGN KEY: define a coluna na tabela filha no nível de restrição da tabela.
- REFERENCES: identifica a tabela e a coluna na tabela mãe.
- ON DELETE CASCADE: permite a exclusão na tabela mãe e das linhas dependentes na tabela filha.

# A restrição CHECK

- Define uma condição que cada linha deve satisfazer.
- Não são permitidas consultas que se referem a outros valores em outras linhas.
- Exemplo:

```
...,
  coddepto NUMBER(2),
  CONSTRAINT empregado_coddepto_ck
     CHECK (coddepto BETWEEN 10 AND 99),
  ...
```

# Adicionando Restrições

- Sintaxe

```
ALTER TABLE [schema.]table
constraint_clause [,…]
[ENABLE [[NO]VALIDATE] [UNIQUE]
  (column [,...] )]
[ENABLE [[NO]VALIDATE] PRIMARY KEY]
[{ENABLE|DISABLE} TABLE LOCK]
[{ENABLE|DISABLE} ALL TRIGGERS];
```

# Adicionando Restrições (cont.)

- Sintaxe

```
constraint_clause:
ADD table_constraint(s)
DROP PRIMARY KEY [CASCADE]
DROP UNIQUE (column [,…])
DROP CONSTRAINT constraint [CASCADE]
MODIFY CONSTRAINT constraint [ENABLE|DISABLE]
   [VALIDATE|NOVALIDATE]
MODIFY PRIMARY KEY [ENABLE|DISABLE] [VALIDATE|NOVALIDATE]
MODIFY UNIQUE (column [,…]) [ENABLE|DISABLE]
   [VALIDATE|NOVALIDATE]
RENAME CONSTRAINT constraint TO new_name
```

Onde,

VALIDADE/NOVALIDATE – Ativa uma restrição [ENABLE] sem validade [NOVALIDADE] não verificando se há violação de restrição nos dados já existentes na tabela.

# Mantendo Restrições

- Adicione ou elimine, mas não "altere" uma restrição.
- Ative ou desative restrições.
- Adicione uma restrição NOT NULL usando a cláusula MODIFY.
- Adicionando uma restrição FOREIGN KEY à tabela EMP indicando que um gerente deve existir como um funcionário válido na tabela EMP.

```
ALTER TABLE empregado
 ADD CONSTRAINT empregado_matgerente_fk
 FOREIGN KEY (matgerente)
   REFERENCES empregado(matricula);
```

# Eliminando uma restrição

- Remova a restrição do gerente da tabela EMP.

```
ALTER TABLE empregado
   DROP CONSTRAINT emp_mgr_fk;
```

- Remova a restrição PRIMARY KEY na tabela DEPARTMENTO e elimine a restrição FOREIGN KEY associada a coluna EMPREGADO.CODDEPTO.

```
ALTER TABLE departamento
   DROP PRIMARY KEY CASCADE;
```

- Observação: para identificar o nome da restrição a partir das visões do dicionário de dados USER_CONSTRAINTS e USER_CONS_COLUMNS.

# Desativando Restrições

- Execute a cláusula DISABLE do comando ALTER TABLE para desativar uma restrição de integridade.
- Aplique a opção CASCADE para desativar restrições de integridade dependentes.

```
ALTER TABLE empregado
  DISABLE CONSTRAINT
   empregado_matricula_pk CASCADE;
```

# Ativando restrições

- Ative uma restrição de integridade atualmente desativada na definição da tabela usando a cláusula ENABLE.

```
ALTER TABLE empregado
ENABLE CONSTRAINT empregado_matricula_pk;
```

- Um índice UNIQUE ou PRIMARY KEY é automaticamente criado se a restrição UNIQUE KEY ou PRIMARY KEY for ativada.

# Restrições em cascata

- A cláusula CASCADE CONSTRAINTS é usada junto com a cláusula DROP COLUMN.
- A cláusula CASCADE CONSTRAINTS elimina todas as restrições de integridade referenciais que se referem às chaves exclusiva e primária definidas nas colunas eliminadas.
- Observação:
  - Consulte a tabela USER_CONSTRAINTS para ver todos os nomes e definições de restrições.
  - Visualize as colunas associadas aos nomes de restrições na visão USER|_CONS_COLUMNS.

# FIM