

< VOLTAR



Exemplificando a criação da estrutura das tabelas, regras e eliminação da estrutura

Apresentação da criação, eliminação da estrutura das tabelas, regras, a partir exemplos em que são utilizados comandos DDL

NESTE TÓPICO
[> Referências](#)

Marcar
tópico



Os comandos do subconjunto DDL (Data Definition Language), ou Linguagem de Definição de Dados, trabalham a estrutura de um banco de dados, possibilitando criar novas tabelas, alterar o que já foi criado e eliminar qualquer estrutura que já exista. Lembrando que estamos usando o SGDBR Oracle 11g, versão disponível para download no site da Oracle (<http://www.oracle.com> (<http://www.oracle.com>)).



De acordo com modelo físico, implementar a tabela em SQL:

Tabela:Tipo_Produto			
Nome da Coluna	Tipo de Dados	Tamanho	Regra – Constraint
Codigo_Tipo	Numérico	4	Chave Primária
Descricao_Tipo	Alfanumérico	20	Não nulo e Único
Tabela Produto			
Nome da Coluna	Tipo de Dados	Tamanho	Regra – Constraint
Codigo_produto	Numérico	4	Chave Primária

Nome_Tipo	Alfanumérico	20	Não nulo e Único
Preco_produto	Numérico	7,2	

Utilizando a versão que insere o nome da regra ao mesmo tempo em que se cria a coluna.

1. create table tipo_produto
2. (codigo_tipo number(4) constraint tp_cod_pk primary key,
3. descricao_tipo varchar2(20) constraint tp_ds_nn not null
4. constraint tp_ds_uk unique)

Resultado da criação tabela tipo_produto:



A digitação dos comandos no ambiente SQL PLUS obedece às convenções de uso, por exemplo, do WORD, ou seja, não há diferença entre maiúsculas e minúsculas, entre a digitação em uma única linha ou em várias linhas, é obrigatório apenas o uso do ponto e vírgula ao final da instrução, somente assim o SQL PLUS entende que o comando está finalizado e pode ser executado.

Caso ocorra erro na digitação, deve-se pressionar a tecla até voltar o prompt (SQL>) ou usar os comandos para edição da instrução vistos na aula anterior.

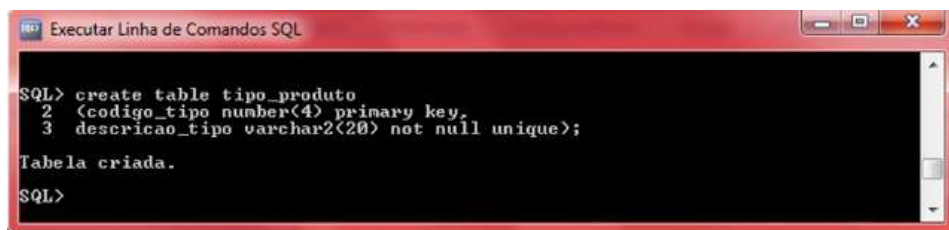
Quando utilizamos a versão que insere o nome da regra depois que se cria a coluna, a única regra que não tem permissão de ser criada desta forma é a regra de preenchimento obrigatório (not null).



```
Executar Linha de Comandos SQL
SQL> create table tipo_produto
2 (codigo_tipo number(4),
3 descricao_tipo varchar2(20) constraint tp_ds_nn not null,
4 constraint tp_cod_pk primary key (codigo_tipo),
5 constraint tp_ds_uk unique(descricao_tipo));
Tabela criada.
SQL>
```

1. create table tipo_produto
2. (codigo_tipo number(4),
3. descricao_tipo varchar2(20) constraint tp_ds_nn not null,
4. constraint tp_cod_pk primary key (codigo_tipo),
5. constraint tp_ds_uk unique(descricao_tipo));

Resultado da criação da tabela produto nomeando as regras após a criação da coluna:



```
SQL> create table tipo_produto
2 (codigo_tipo number(4) primary key,
3 descricao_tipo varchar2(20) not null unique);
Tabela criada.
SQL>
```

Ao utilizar este estilo de desenvolvimento você deverá, na linha em que está criando a regra, indicar a coluna que vai recebê-la, utilizando a versão que cria a regra, mas não insere o nome da regra depois que se cria a coluna.

1. create table tipo_produto
2. (codigo_tipo number(4) primary key,
3. descricao_tipo varchar2(20) not null unique);

Resultado da criação da tabela sem nome nas regras:



```
SQL> create table tipo_produto
2 (codigo_tipo number(4) primary key,
3 descricao_tipo varchar2(20) not null unique);
Tabela criada.
SQL>
```

Existe um motivo para inserir o nome na regra, pois, para realizar qualquer alteração na estrutura da tabela, usamos o nome da tabela, o nome da coluna e também o nome da regra. Se no desenvolvimento já tenha sido realizado este processo, não será necessário procurar o nome, caso contrário você terá um pouco mais de trabalho.

Uma tabela também pode ser criada sem regras, veja o exemplo a seguir:

1. create table tipo_produto
2. (codigo_tipo number(4),
3. descricao_tipo varchar2(20));

Resultado da criação da tabela sem regras:



```
SQL> create table tipo_produto
2 (codigo_tipo number(4),
3 descricao_tipo varchar2(20));
Tabela criada.
SQL> _
```

Para visualizar a estrutura criada de uma tabela qualquer, deve-se usar o seguinte comando:

1. Desc nome_tabela;
2. Exemplo: desc tipo_produto;

Neste comando o uso do ponto e vírgula no final da instrução é opcional.
Resultado de visualização de uma tabela:



The first screenshot shows the result of the command `SQL> desc tipo_produto`. It displays the table structure with columns `CODIGO_TIPO` and `DESCRICAO_TIPO`, their data types (`NUMBER(4)` and `VARCHAR2(20)`), and whether they are nullable.

```

SQL> desc tipo_produto
None                                     Nulo?      Tipo
-----
CODIGO_TIPO                             NUMBER(4)
DESCRICAO_TIPO                          VARCHAR2(20)
SQL>

```

The second screenshot shows the result of the command `SQL> Select constraint_name, constraint_type, table_name 2 from user_constraints where table_name = 'TIPO_PRODUTO';`. It lists the constraints for the `TIPO_PRODUTO` table.

```

SQL> Select constraint_name, constraint_type, table_name
2 from user_constraints where table_name = 'TIPO_PRODUTO';
CONSTRAINT_NAME      C TABLE_NAME
-----
SYS_C004003          C TIPO_PRODUTO
SYS_C004004          P TIPO_PRODUTO
SYS_C004005          U TIPO_PRODUTO
SQL>

```

Observe que esta tabela não possui regras. Entretanto, como saber se não há regras?

As colunas demonstram: nome da coluna – nome, null? – Chave primária ou not null e tipo – tipo de dados e tamanho.

Para verificar se existem outras regras, deve-se usar o seguinte comando:

1. `Select constraint_name, constraint_type, table_name`
2. `from user_constraints where table_name = 'TIPO_PRODUTO';`

Resultado da visualização das regras de uma tabela:

Esta instrução mostra a coluna `constraint_name` (nome da constraint), `constraint_type` (tipo da constraint) que foram ou não criadas na tabela, observe que o nome das regras começam com `sys`, que é abreviação de sistema, e `C`, abreviação de constraint, e logo após um código, isto mostra que a tabela foi criada sem o recurso para nomear a regra, ou seja, o próprio SGBDR nomeou.

Agora você conhece a estrutura da mesma tabela usando a instrução que dá nome às regras.



The screenshot shows the result of the command `SQL> Select constraint_name, constraint_type, table_name 2 from user_constraints where table_name = 'TIPO_PRODUTO';`. It lists the constraints for the `TIPO_PRODUTO` table with meaningful names.

```

SQL> Select constraint_name, constraint_type, table_name
2 from user_constraints where table_name = 'TIPO_PRODUTO';
CONSTRAINT_NAME      C TABLE_NAME
-----
TP_DS_MN             C TIPO_PRODUTO
TP_COD_PK            P TIPO_PRODUTO
TP_DS_UK             U TIPO_PRODUTO
SQL>

```

Pode-se perceber que agora não há mais códigos e, sim, nomes que são dados pelo programador.

Exemplos:

TP – é a abreviação do nome da tabela: `Tipo_Produto`

DS – é a abreviação do nome da coluna: `Descrição`

NN – é a abreviação do tipo da regra: Not Null

Há uma diferença em relação ao que é criado por um usuário de banco de dados e o que é criado pelo sistema gerenciador, pois as estruturas que criamos são chamadas de "tabelas de usuários" e as estruturas que já são nativas do SGDBR, por exemplo, user_objects, no Oracle são chamadas de "dicionários de dados". São estas estruturas que controlam o ambiente de trabalho, gerenciam a execução dos comandos. Por exemplo, caso tente criar uma tabela que já exista ou que tenha o mesmo nome, o SGBDR não permitirá a criação.

Para saber quais as tabelas que já foram criadas em um usuário, basta executar o comando a seguir:

```
1. select table_name from tabs;
```

O comando select cria um relatório, table_name é o nome de uma coluna na tabela tabs, ou seja, será criado um relatório que mostrará o conteúdo da coluna table_name que pertence à tabela tabs.

Outro exemplo, agora misturando as versões:

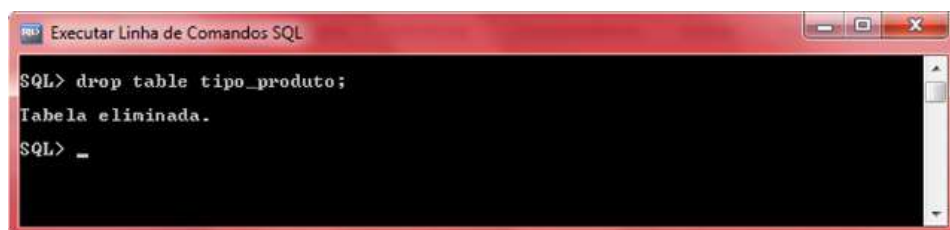
```
1. create table produto
2. (codigo_produto number(4) primary key,
3. nome_produto varchar2(20) not null,
4. preco_produto number(9,2),
5. constraint pro_nome_uk unique (nome_produto));
```

Note que não há problema em misturar as versões de criação, esta versatilidade do SQL permite que cada desenvolvedor crie seu próprio método de trabalho.

Caso você queira eliminar a estrutura de uma tabela, digite o seguinte comando:

```
1. drop table nome_tabela;
```

Exemplo: drop table tipo_produto;



Neste caso, como ainda não citamos o relacionamento, o comando será executado sem problemas. Entretanto, quando houver o relacionamento, algumas medidas devem ser observadas.

Referências

BEIGHLEY, Lynn. *Use a Cabeça SQL*. Rio de Janeiro: Alta Books, 2008.

FANDERUFF, Damaris. *Dominando o Oracle 9i: Modelagem e Desenvolvimento*, São Paulo: Makron, 2003.

GRAVES, Mark. *Projeto de banco de dados com XML*. São Paulo: Pearson, 2003.

MORELLI, Eduardo Terra. *Oracle 9i Fundamental: SQL, PL/SQL e Administração*, São Paulo, Editora Érica, 2002.

PRICE, Jason. *Oracle Database 11g SQL*. (tradução: João Eduardo Nóbrega Tortello). Porto Alegre: Bookman, 2009.

SILVA, Robson. *Oracle Database 10g Express Edition*. São Paulo: Editora Érica, 2007.



Avalie este tópico



ANTERIOR

Trabalhando com a estrutura de tabelas

Biblioteca

([https://www.uninove.br/conheca-](https://www.uninove.br/conheca-a-uninove/biblioteca/sobre-a-biblioteca/apresentacao/)

a-

uninove/biblioteca/sobre-

a-

biblioteca/apresentacao/)

Portal Uninove

(<http://www.uninove.br>)

Mapa do Site



Índice

Trabalhando a estrutura das tabelas, relacionam

relacionam

© Todos os direitos reservados

Ajuda?

PRÓXIMO

([https://ava.un](https://ava.uninove.br/seu/AVA/topico/topico.php)

https://ava.uninove.br/seu/AVA/topico/topico.php

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam

relacionam