Usando DDL Statements para Criar e Gerenciar Tabelas

Um schema é uma coleção de múltiplos objetos de bases de dados, que são conhecidos como objetos do schema. O dono do schema tem acesso direto a esses objetos. Abaixo a lista de objetos de um schema.

* Table – tabela para armazenar dados
* View – para projetar os dados no formato desejado, proveniente de uma ou mais tabelas.
* Sequence – para gerar valores numéricos
* Index – índice para melhorar a performance das consultas a uma tabela.
* Synonym - sinônimo, nome alternativo para um objeto

Um dos primeiros passos da criação de uma base de dados é criar as tabelas que irão armazenar os dados de uma organização. O design da base de dados envolve identificar os requisites do usuário para os vários sistemas organizacionais, como por exemplo pedido, gerenciamento do inventário, finanças. Independentemente do tamanho, cada base de dados é composta por tabelas.

Criação de Tabelas – DDL – Data Definition Language

Para criar uma tabela em uma base de dados, o DBA (Database Administrator) deve ter algumas informações – o nome da tabela, nome das colunas, tipos de dados das colunas e tamanhos das colunas. Todas essas informações podem ser modificadas mais tarde usando comandos DDL.

Convenções de Nomes de Tabelas -

* O nome escolhido para uma tabela deve seguir as seguintes regras:
* O nome da tabela deve iniciar com uma letra: A-Z ou a-z
* Pode conter números e underscores( \_ )
* Pode conter **UPPER CASE** (maiúsculas) ou **lower case** (minúsculas)
* O comprimento (**length**) deve ser de até 30 caracteres.
* O nome não deve ser utilizado por nenhum outro objeto do schema.
* O nome não pode ser uma palavra reservada do **SQL (Structured Query Language)**

De acordo com as regras, **'EMP85'** é um nome de tabela válido, mas **‘85EMP’** não é.

Da mesma forma, **UPDATE** não pode ser escolhido como nome de tabela, pois é uma palavra reservada do **SQL**.

CREATE TABLE statement (um statement é uma linha de comando, é bom se familiarizar com os termos)

O **CREATE TABLE** é um comando **DDL** para criar tabelas na base de dados. A tabela é criada assim que o script que contém o **CREATE TABLE** é executado, e está pronta para armazenar dados. O usuário deve ter o **privilégio** no sistema para criar tabelas no seu próprio esquema. Porém, para criar uma tabela no esquema de qualquer outro usuário, deve ter o **privilégio** **CREATE ANY TABLE**.

Veremos concessão de privilégio (permissão) mais adiante.

Essa é a sintaxe do comando CREATE TABLE (veremos constraints mais adiante)

CREATE TABLE [schema.]table

( { column datatype [DEFAULT expr] [column\_constraint] ...

| table\_constraint}

[, { column datatype [DEFAULT expr] [column\_constraint] ...

| table\_constraint} ]...)

[AS subquery]

Na sintaxe acima, **DEFAULT** especifica um valor que é usado, quando o comando **INSERT** não contém essa coluna. O valor default não pode conter referências a colunas de outras tabelas ou a pseudo-colunas (CURRVAL, NEXTVAL, LEVEL, and ROWNUM) excetp SYSDATE(data do sistema) e USER (usuário corrente), ou ainda constantes de datas não especificadas.

**Constraints** são as regras definidas opcionalmente para a coluna ou para a tabelal (cobertas mais adiante).These rules are checked during any data action (Insert, update) on the table and raise error to abort the action upon its violation. Essas regras são verificadas em cada ação sobre a tabela (**INSERT** - comando SQL para inserir dados, ou **UPDATE** – comando SQL para atualizar uma tabela), e lançam um erro caso sejam violadas.

No exemplo abaixo, **CREATE TABLE** cria a tabela **EMP\_TEST**, que pertence ao esquema **RESGATANDO**. **Note a especificação das colunas, o tipo de dados e precisão.**

CREATE TABLE **RESGATANDO**.**EMP\_TEST**

(IDEMP NUMBER,

NOMEEMP VARCHAR2(100),

DEPARTMENTO\_ID NUMBER,

SALARIO NUMBER,

CODIGOFUNC VARCHAR2(3),

ADMISSAO DATE);

Exemplo de uma query SELECT

**SELECT** NOMEEMP, SALARIO,

**FROM** **RESGATANDO**.**EMP\_TEST**;

Exemplo de criação da tabela com valor Default: é criada a tabela DEPARTAMENTO, que possui as colunas ID\_DEPARTAMENTO, NOME\_DEP e ID\_LOCALIDADE. Para ID\_LOCALIDADE, é atribuído o valor DEFAULT 100, ou seja, quando for inserido um registro na tabela DEPARTAMENTO e não for passado o valor para a coluna ID\_LOCALIDADE, como está definido o DEFAULT para essa coluna, o registro será criado contendo o valor default par aa coluna ID\_LOCALIDADE.

CREATE TABLE RESGATANDO.DEPARTMENTO

(ID\_DEPARTAMENTO NUMBER,

NOME\_DEP VARCHAR2 (100),

ID\_LOCALIDADE NUMBER DEFAULT 100);

Tipos de Dados

Os tipos de dados são usados para especificar o comportamento básico de uma coluna da tabela, ou seja, o tipo de valores que ela deve conter, NUMBER (números inteiros), caracter, STRING (palavra ou sequência de caracteres), DATE, ...Há vários SUBTIPOS que pertencem a esses tipos de dados.

Number

O tipo NUMBER datatype engloba tanto valores numéricos inteiros, decimais, float.

Escolha o tipo NUMBER quando uma coluna deve armazenar dados numéricos que podem ser usados em cálculos matemáticos.

NUMBER (p, s), onde p é a precisão de até 38 dígitos and s é a escala (número de dígitos à direita do ponto decimal). A escala pode variar de -84 a 127.

NUMBER (p) é um número com escala zero e precisão de valor p.

FLOAT [(p)], onde p é a precisão binária que pode variar de 1 to 126. Note que p está entre [ ], indicando que é opcional.

Date

Para cada tipo DATE, Century (século), Year (ano), Month (mês), Day (dia) , Hour (hora), Minute (minuto), Second (segundo) são armazenados na base de dados. Cada Sistema de base de dados tem um formato default definido pela inicialização do parâmetro NSL\_DATE\_FORMAT. Esse parâmetro é normalmente definido para DD-MM-YY. Se um horário não for especificado, a hora default é 12:00:00 a.m.

Character

Oracle suporta 3 tipos de caracteres : CHAR (um caracter único), VARCHAR(tamanho definido, ex. VARCHAR(10)), e VARCHAR2, sendo que VARCHAR2 suporta até 9000 caracteres.

### INSERT

O comando INSERT serve para inserir dados no banco de dados.

Com o comando é possível fazer isto de maneira mais dinâmica e dentro da própria aplicação, sem precisar recorrer a códigos PHP para inserir dados.

### UPDATE

O comando UPDATE serve para facilitar a atualização de dados nas tabelas.

Esse recurso é bastante simples e de grande utilidade, já que torna possível a alteração de diversos registros com poucos cliques.

### DELETE

Como o próprio nome já denuncia, o comando DELETE serve para excluir um ou mais registros de uma base de dados.

É possível remover vários registros simultaneamente com o uso do DELETE, o que faz o desenvolvedor economizar tempo na checagem do banco de dados.

### CREATE TABLE

Esse comando serve para criar novas tabelas em uma base de dados.

O CREAT TABLE cria novas tabelas, conseguindo dividi-las em colunas, onde é possível salvar e referenciar especificações de produtos, etc.

Com a criação de uma tabela, é preciso especificar todas as colunas e campos criados e este comando consegue dar o tom para toda essa parte importante para a inserção de novos registros num banco de dados.

## Por que devo conhecer os comandos SQL listados acima?

Quem trabalha com a parte que foca em tecnologias Server-side (lado do servidor) sabe que aprender a utilizar comandos SQL em banco de dados pode otimizar muito o trabalho e aumentar a produtividade de toda uma equipe de desenvolvedores.

Para entender a importância de conhecer os comandos, basta relacionar o número de registros que uma base de dados geralmente possui.

Quem lida com variados bancos de dados, deve utilizar os comandos SQL para conseguir manter tudo organizado e saber onde está cada registro sem tanta dificuldade.

Esses comandos são essenciais para quem precisa fazer muitas consultas durante o dia ou precisa cadastrar novos produtos com frequência.

Quem trabalha com lista de produtos, de endereços, entre outros tipos, sabe que usar os comandos em seus determinados fins pode agilizar muito o trabalho.

Aprender a usar os comandos SQL que listamos acima é fundamental na vida de qualquer desenvolvedor back-end.