Autor: **Nicola Strappazzon**
Web: http://nicola.strappazzon.me
Revisión: 17/11/11

Lección 9

Tablas

Este material se encuentra basado en el curso de Fundamentos a SQL de ORACLE, el cual es adaptado para el producto PostgreSQL, todos los ejemplos, códigos fuentes y la Base de Datos HR es propiedad de ORACLE.

Objetivos

Al completar esta lección usted podrá entender los siguientes puntos:

- Conocer la estructura de una tabla.
- Listar los diversos tipos de datos.
- Entender como funcionan las restricciones de los datos.
- Mantener la integridad de los datos con las relaciones entre tablas.
- Mejorar el tiempo de acceso a los datos mediante los Indices.

Las Tablas

- Es un objeto que contiene datos de forma permanente.
- Cada tabla es una entidad que agrupa conceptualmente una serie de datos en común y estos se encuentran segmentados en varias columnas.
- Cada tabla tiene reglas que definen una serie de restricciones para evitar redundancia y mantener la integridad de los mismos.
- Las tablas dependen entre ellas para mantener la relación de los datos.

Definir nombres

Los nombres de las tablas y las columnas deben respetar las siguientes reglas:

- Deben empezar con letras.
- Debe tener entre 1 y 63 caracteres.
- Solamente se permiten los siguientes caracteres;

  0-9, a-z, A-Z, _ y $.
- No se permiten nombres duplicados, solo en Case Sensitive.
- Para mantener nombres en Case Sensitive debe colocarlos entre comillas dobles.

Usando la cláusula CREATE TABLE

Considere lo siguiente:

- Debe tener privilegios para crear tablas.
- Espacio en disco.
- Nombre de la tabla.
- Nombre de las columnas, tipo de dato a utilizar y su longitud.

```
CREATE TABLE table_name ([
  column_name data_type [ DEFAULT expr ], ...
]);
```

Tipo de Datos para números

Nombre	Tamaño	Rango
smallint	2 bytes	-32768 hasta +32767
integer	4 bytes	-2147483648 hasta +2147483647
bigint	8 bytes	-9223372036854775808 hasta 9223372036854775807
decimal	variable	No existe limite, definir la expresión exacta.
numeric	variable	No existe limite, definir la expresión exacta.
real	4 bytes	6 decimales
double precision	8 bytes	15 decimales
serial	4 bytes	1 hasta 2147483647
bigserial	8 bytes	1 hasta 9223372036854775807

Tipo de Datos para carácteres

Nombre	Descripción	Ejemplo	Resultado
character(l)	Se define la longitud de caracteres (l), completa con espacios en blanco.	character(2)	VE
character varing(l)	Se define la longitud de caracteres (l), no completa con espacios en blanco.	character varing(10)	Caracas
text	De longitud ilimitada.	text	abcdefghijk lmnu...

Tipo de Datos para fechas

Nombre	Tamaño	Descripción	Min.	Max.
timestamp	8 bytes	Fecha y hora, permite la zona horaria.	4713 AC	5874987 DC
interval	12 bytes	Intervalo de tiempo, contempla cualquier unidad.	-178000000 años	178000000 años
date	4 bytes	Solo fecha.	4713 AC	5874987 DC
time	8-12 bytes	Solo hora, permite la zona horaria.	00:00:00+1359	24:00:00-1359

Crear tablas

- Crear una tabla:

```
CREATE TABLE films (
    code        char(5),
    title       varchar(40),
    summary     text,
);
```

- Visualizar la estructura de la tabla desde psql:

```
HR=# \d films
              Table "public.films"
  Column    |          Type           | Modifiers
------------+-------------------------+-----------
 code       | character(5)            |
 title      | character varying(40)   |
 summary    | text                    |
HR=#
```

Valores por defecto

Agregar valores por defecto cuando son omitidos en su inserción.

```
CREATE TABLE products (
    pk_product integer NOT NULL,
    name       text    NOT NULL,
    summary    text,
    price      numeric NOT NULL,
    discounted numeric DEFAULT 0 NOT NULL,
    company    integer NOT NULL
);
```

Consultas de Ejemplo

Insertamos un nuevo registro omitiendo las columnas **summary** y **discounted**, observando el valor **0** por defecto que es incorporado de forma automática.

```
HR=# INSERT INTO products (pk_product, name, price, company)
HR-# VALUES (1, 'DVD', 12, 1);
INSERT 0 1
HR=# SELECT * FROM products;
 pk_product | name | summary | price | discounted | company
------------+------+---------+-------+------------+---------
          1 | DVD  |         |    12 |          0 |       1
```

Restricciones

- Define reglas dentro de una tabla o en columnas.
- Evita eliminar datos dentro de una tabla que tiene dependencias.
- Se pueden aplicar mientras crea una tabla o después.
- Existen los siguientes tipos de restricciones:
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FORAIGN KEY
  - CHECK

Restricción NOT NULL

Evita que una columna en especifico contenga valores de tipo NULL, en caso de que esta no contenga la restricción, se permite la inserción del valor NULL.

```
CREATE TABLE films (
    code        char(5)      NOT NULL,
    title       varchar(40)  NOT NULL,
    summary     text
);
```

Restricción UNIQUE

Restringe un grupo de columnas donde se requieren que sus valores siempre sean únicos, evitando que existan valores duplicados en dichas columnas.

```
CREATE TABLE films (
    code        char(5)      NOT NULL,
    title       varchar(40)  NOT NULL,
    summary     text,
    CONSTRAINT un_film UNIQUE (code, title)
);
```

Restricción PRIMARY KEY

Define una o mas columnas como clave primaria de la tabla, haciendo ésta única y permitir la relación de un registro con otra tabla para crear la dependencia.

```
CREATE TABLE films (
    code        char(5)      NOT NULL,
    title       varchar(40)  NOT NULL,
    summary     text,
    CONSTRAINT pk_film PRIMARY KEY (code)
);
```

Restricción FORAING KEY

Define una o mas columnas como clave primaria de la tabla, haciendo ésta única y permitir la relación de un registro con otra tabla para crear la dependencia.

```
CREATE TABLE films (
    code        char(5)      NOT NULL,
    title       varchar(40)  NOT NULL,
    summary     text,
    company     integer      NOT NULL,
    CONSTRAINT fk_company FOREIGN KEY (company)
    REFERENCES product (pk_product)
);
```

Restricción CHECK

Define una o varias condiciones que debe satisfacer a cada uno de los registro existentes de forma boleana.

- No se permiten consultas.
- Solo se utilizan operadores de condición y de conjunto.

```
CREATE TABLE products (
    pk_product integer,
    name       text,
    price      numeric,
    discounted numeric,
    CONSTRAINT cn_price CHECK (price > 0 AND
                               price > discounted)
);
```

Código de Ejemplo

```
CREATE TABLE companies (
    pk_company integer NOT NULL,
    name       text    NOT NULL,
    CONSTRAINT un_company UNIQUE (name),
    CONSTRAINT pk_company PRIMARY KEY (pk_company)
);

CREATE TABLE products (
    pk_product integer NOT NULL,
    fk_company integer NOT NULL,
    name       text    NOT NULL,
    summary    text,
    price      numeric NOT NULL,
    discounted numeric DEFAULT 0,
    CONSTRAINT un_product UNIQUE (name),
    CONSTRAINT ck_product CHECK (discounted > 0 AND price > discounted),
    CONSTRAINT pk_product PRIMARY KEY (pk_product),
    CONSTRAINT fk_company FOREIGN KEY (fk_company)
    REFERENCES companies (pk_company)
);
```

Violando una restricción

No existe el la clave primaria:

```
HR=# INSERT INTO products (pk_product, fk_company, name, price, discounted)
HR-# VALUES (1, 1, 'DVD', 10, 0.5);
ERROR:  insert or update on table "products" violates foreign key constraint
"fk_company"
DETAIL:  Key (fk_company)=(1) is not present in table "companies".
HR=#
```

No existe la clave primaria con el valor 1 en la tabla companies.

Violando una restricción

No cumple con una restricción

```
HR=# INSERT INTO products (pk_product, fk_company, name, price)
HR-# VALUES (1, 1, 'DVD', 10);
ERROR:  new row for relation "products" violates check constraint
"ck_product"
HR=#
```

Los datos a insertar no cumplen con la restricción ck_product.

Violando una restricción

Visualizar la estructura de la tabla para identificar la restricción:

```
HR=# \d products
     Table "public.products"
   Column    |  Type   | Modifiers
-------------+---------+-----------
 pk_product  | integer | not null
 fk_company  | integer | not null
 name        | text    | not null
 summary     | text    |
 price       | numeric | not null
 discounted  | numeric | default 0
Indexes:
    "pk_product" PRIMARY KEY, btree (pk_product)
    "un_product" UNIQUE, btree (name)
Check constraints:
    "ck_product" CHECK (discounted > 0::numeric AND price > discounted)
Foreign-key constraints:
    "fk_company" FOREIGN KEY (fk_company) REFERENCES companies(pk_company)

HR=#
```

Violando una restricción

No puede actualizar o eliminar un registro por que una de las claves foráneas se encuentra relacionada con otra tabla.

```
HR=# DELETE FROM companies WHERE pk_company = 1;
ERROR:  update or delete on table "companies" violates foreign key
constraint "fk_company" on table "products"
DETAIL:  Key (pk_company)=(1) is still referenced from table "products".
HR=#
```

La clave foránea de valor 1 se encuentra en uso dentro de la tabla products.

Sentencia `ALTER TABLE`

Modifica la estructura física de una tabla, contemplando:

- Columnas.
- Tipo de dato sobre una columna.
- Valores por defecto.
- Restricciones.
- Relaciones.
- Triggers.
- Usuario Dueño.
- Permisos.
- Comentarios.

Eliminar una tabla

- Toda la data es eliminada.
- Toda su estructura se elimina.
- Todo tipo de restricción, relación, indices y triggers son eliminados.
- No se puede hacer rollback una ves que la tabla se haya eliminado.

```
HR=# DROP TABLE products;
DROP TABLE
```

Orden de las culumnas

Para mejorar el rendimiento levemente de las consultas, debe tener en cuenta el siguiente criterio para el orden mientras crea las columnas.

- Clave primarias.
- Columnas con la restricción de tipo único.
- Clave foráneas.
- Columnas obligatorias.
- Columnas no obligatorias.

Indices

Estructura física de datos que permiten ubicar un determinado valor solicitado utilizando un metodo de acceso.

- Son estructuras que no se encuentran alojadas dentro de una tabla pero dependen de ella.
- Incrementa la velocidad de búsqueda.
- Se definen por una o varias columnas en especifico donde son utilizadas para realizar una búsqueda.
- El algoritmo definido por defecto es el B-Trees.
- Se pueden ordenar los indices de forma ascendente o descendente y existe un tratamiento especial para los valores de tipo NULL.

Desventajas de los Indices

- Consumen más espacio en disco.
- El uso consecutivo de sentencias de tipo **`INSERT`**, **`UPDATE`** y/o **`DELETE`** disminuyen el rendimiento cuando existen indices, por lo que se requiere reconstruirlos periódicamente como una actividad de mantenimiento. Se utiliza el comando: **reindexdb**

Uso de los Indices

Debe crean indices obligatoriamente en las siguientes columnas, las cuales son utilizadas constantemente:

- Claves primarias.
- Claves foráneas.
- Columnas que se utilizan con operadores lógicos.
- Campos de búsqueda.

Algoritmo B-Trees

Existen muchos tipos, por defecto se considera que el algoritmo B-Trees, es suficiente debido a que puede manejar perfectamente las condiciones de igualdad.

- <
- <=
- =
- >=
- >

Parámetros básicos

Adicionalmente el algoritmo se apoya de parámetros en su creación, con el fin de ampliar más su rendimiento.

- **ASC:** Ordena de forma ascendente (Por defecto).
- **DESC:** Ordena de forma descendente.
- **NULL FIRST:** Especifica que los valores de tipo NULL estarán de primero con los valores que no son de tipo NULL, este valor es asignado por defecto cuando se ordena de forma DESC.
- **NULL LAST:** Especifica que los valores de tipo NULL estarán de ultimo con los valores que no son de tipo NULL, este valor es asignado por defecto cuando se ordena de forma ASC.
- **Unique:** El sistema arroja un error al verificar si existen valores duplicados en la tabla cada vez que el indice es creado o actualizado.

Ejemplo

Sintaxis básica:

```
CREATE [ UNIQUE ] INDEX name ON table [ USING method ]
( column [ ASC | DESC ] [ NULLS { FIRST | LAST } ]
[, ...] )
```

Ejemplos:

```
CREATE UNIQUE INDEX title_idx ON films (title);
```

```
CREATE INDEX title_idx_nulls ON films (title NULLS FIRST);
```

Resumen

En esta lección, usted debió entender como:

- Crear, modificar y eliminar tablas.
- Visualizar la estructura de la tabla.
- Definir restricciones para el contenido de una tabla.
- Establecer la relación entre una o mas tablas.
- Incrementar la velocidad de acceso a los registros mediante los Indices.