HINTS

COMIENZA CON SENTENCIAS SIMPLES

Uno de los errores más comunes son los de sintaxis, es por eso que primero debes estudiar sobre el motor de bases de datos, para así entender cómo probar y depurar tus consultas SQL. Es importante comenzar con sentencias simples.

ESCRIBE EL CÓDIGO DE MANERA SIMPLE Y ORDENADA

Hay que recordar siempre que este código puede ser modificado a futuro. Un código complejo tarda más tiempo en ser modificado, y es más factible cometer errores en él. Escribe el código SQL lo más ordenado y simple posible, y así será más fácil de entender.

IMPORTANCIA ÁLGEBRA RELACIONAL

Es adecuado ver a los datos como un conjunto de objetos que se pueden relacionar para obtener información, por lo que las operaciones de algebra relacional básicas deben ser aprendidas a la perfección, ya que el objetivo de los motores de datos relacionales se sustenta en esto.

DEFINICIÓN DE DATOS

Es importante que aprendas a asociar el álgebra relacional para dar solución a problemas cotidianos en la obtención o requerimiento de información.

GENERACIÓN DE IDENTIFICADORES CON SECUENCIAS

En PostgreSQL, una secuencia es un tipo especial de objeto que genera una secuencia de números enteros. Éstas se usan a menudo como la columna de la primary key en una tabla.

Al crear una nueva tabla, la secuencia se puede generar a través del pseudotipo **SERIAL**. Al asignarle éste a alguna columna que almacena los id, PostgreSQL realiza lo siguiente:

- Primero, crea un objeto de secuencia, y establece el siguiente valor generado como el predeterminado para la columna.
- En segundo lugar, la secuencia implementa una restricción **NOT NULL** a la columna id, porque una secuencia siempre genera un número entero, que es un valor no nulo.
- Tercero, la secuencia es asignada al objeto propietario de la columna id; así el objeto de secuencia desaparece cuando se elimina la columna o tabla de id.

Es importante tener en cuenta que **SERIAL** no crea implícitamente un índice en la columna, ni hace que ésta sea la columna de llave primaria. Sin embargo, esto se puede lograr especificando la restricción **PRIMARY KEY** para la columna **SERIAL**.

La siguiente declaración crea la tabla de frutas con la columna id como la columna **SERIAL**:

```
CREATE TABLE frutas(
   id SERIAL PRIMARY KEY,
   name VARCHAR NOT NULL
);
```

Para asignarle un valor predeterminado a una columna con el objeto **SERIAL**, al ingresar un registro en la tabla podemos ignorar el nombre de la columna entre los paréntesis, o usar la palabra clave **DEFAULT** en el **INSERT**:

```
INSERT INTO frutas(id, nombre)
VALUES(DEFAULT,'Manzana');
```

Ya que usamos el pseudotipo **SERIAL**, al momento de ingresar un nuevo registro en la tabla no necesitamos ingresar un valor, y podemos omitir esa columna por completo.

```
INSERT INTO frutas(nombre)
VALUES('Naranja');

INSERT INTO frutas(nombre)
VALUES('Uvas');
```

Si ahora consultamos los registros de la tabla **FRUTAS**, podremos ver que cada uno tiene su id generado mediante una secuencia en Postgre.

```sql
SELECT * FROM FRUTAS;
```

id	name
1	Naranja
2	Manzana
3	Uvas