Referencia SQL: Tablas

En esta sección se presenta una referencia de las sentencias de SQL para crear, modificar y eliminar una tabla de la base de datos.

Creación de tablas

La sentencia CREATE TABLE se utiliza para crear una tabla nueva en la base de datos.

La sintaxis básica es:

```
CREATE TABLE table_name (
   column1 datatype,
   column2 datatype,
   column3 datatype,
  ....
);
```

Donde *column1*, *column2*, *column3*, etc. son los nombres de las columnas y *datatype* es el tipo de datos de la columna. Los tipos de datos soportados dependerá del motor de base de datos utilizando. En SQLite son:

- INTEGER
- TEXT
- NUMERIC
- REAL
- BLOB

Ejemplo:

```
CREATE TABLE `autor` (
        `ID`    INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        `NOMBRE`        TEXT NOT NULL,
        `APELLIDO`      TEXT NOT NULL
);
```

Do Not Copy Post

Material de la Universidad Austral, preparado por el profesor Agustín Olmedo, Buenos Aires, Argentina, en junio de 2019 para su uso en el Programa Especializado de "Aprende a programar con Python". Prohibida la reproducción, total o parcial sin previa autorización escrita por parte del autor.

Restricciones SQL

Además, al definir las columnas en la creación de una tabla se pueden especificar algunas restricciones de esas columnas:

- **NOT NULL:** Asegura que la columna no tenga el valor NULL. Con lo cual, la columna tendrá que tener siempre definido algún valor.
- **UNIQUE:** Asegura que todos los valores en una columna sean distintos.
- **PRIMARY KEY:** Una combinación de NOT NULL y UNIQUE. Identifica unívocamente cada registro en una tabla.
- **FOREIGN KEY:** Se identifica de forma única un registro en otra tabla.
- **CHECK:** Asegura que todos los valores en una columna satisfacen una condición específica. Por ejemplo, que sean valores mayores a cero.
- **DEFAULT:** Pone un valor por defecto para una columna cuando no se especifica un valor.
- **INDEX:** Utilizados para crear y devolver datos de forma eficiente. No soportado, en la creación de la tabla, por todas las versiones de SQL.
- **AUTOINCREMENT:** Genera automáticamente un número único (incremental) cuando se inserta un nuevo registro a la tabla. Muchas veces se utiliza en los campos que son la clave primaria de la tabla.

A continuación, se muestra un ejemplo que utiliza algunas de las restricciones mencionadas anteriormente:

```
CREATE TABLE `libro` (
        `ID`        INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        `ISBN`    TEXT NOT NULL,
        `TITULO`TEXT NOT NULL,
        `DESCRIPCION`    TEXT NOT NULL DEFAULT "",
        `AUTOR`INTEGER NOT NULL DEFAULT 1,
        `precio`  INTEGER,
        `categoria_id`      INTEGER,
        FOREIGN KEY(`AUTOR`) REFERENCES `autor`(`ID`),
        FOREIGN KEY(`categoria_id`) REFERENCES `categoria_libro`(`CODIGO`)
);
```

Copy or Post

Creación de una tabla a partir del resultado de una consulta

Además, con la sentencia CREATE TABLE se puede crear una copia de una tabla existente, como resultado de una consulta.

La nueva tabla tendrá la misma definición de columnas. Se podrán seleccionar todas las columnas o algunas específicas.

Si se crea una tabla nueva a partir de una tabla existente, la nueva tabla se llenará con los valores existentes de la tabla vieja.

La sintaxis básica es:

```
CREATE TABLE new_table_name AS
   SELECT column1, column2,...
   FROM existing_table_name
   WHERE ....;
```

Por ejemplo:

```
CREATE TABLE libro_copy AS
   SELECT ISBN, TITULO FROM libro;
```

Do Not Copy or Post

Modificación de tablas

La sentencia ALTER TABLE se utiliza para agregar, borrar o modificar columnas en una tabla existente. Además, esta sentencia se utiliza para agregar o quitar restricciones sobre una tabla existente.

La sintaxis básica es:

```
-- Se agrega columna column_name
ALTER TABLE table_name
ADD column_name datatype;

-- Se borra columna column_name
ALTER TABLE table_name
DROP COLUMN column_name;

-- Actualizo columna column_name
ALTER TABLE table_name
ALTER COLUMN column_name datatype;
```

A continuación, veremos algunos ejemplos sobre las diversa formas de modificar una tabla:

```
ALTER TABLE libro
ADD categoria_id INTEGER REFERENCES categoria_libro(ID);

ALTER TABLE libro
DROP COLUMN categoria_id;

ALTER TABLE libro
ALTER COLUMN  precio REAL;

-- Agregar una restricción
ALTER TABLE libro
ADD CONSTRAINT chk_price CHECK (precio>=0);
```

La primera consulta agrega la columna *categoria_id* de tipo INTEGER que referencia a la tabla categoria_libro a través de la columna ID. Con lo cual, se está creando una columna a la vez que se define una clave foránea.

La segunda consulta borra la columna categoria_id. La tercera consulta actualiza el tipo de datos de la columna precio al tipo REAL. Y la cuarta consulta agrega una restricción que verifica que la columna precio contenga valores mayores o iguales a cero.

Borrado de tablas

La sentencia DROP TABLE se utiliza para borrar una tabla existente en la base de datos.

La sintaxis básica es:

```
DROP TABLE table_name;
```

Donde table_name es el nombre de una tabla existente en la base de datos.

A continuación, se muestra un ejemplo donde se borra la tabla libro_copy:

```
DROP TABLE libro_copy;
```

La sentencia TRUNCATE TABLE se utiliza para borrar los datos de una tabla sin borrar la misma.

La sintaxis es:

```
TRUNCATE TABLE table_name;
```

Cabe aclarar que no todos los motores de base de datos soportan esta sentencia.

Do Not Copy or Post