Bases de Datos

Carrera Programador full-stack

Lenguaje de DDL - Creación de Esquemas

Introducción a SQL

- Para acceder y manipular los datos dentro de la base de datos se necesita un lenguaje declarativo
- **SQL** significa Lenguaje de consulta estructurado (Structured Query Language)
- **SQL** es un lenguaje de bases de datos global: cuenta con sentencias para definir datos, consultas y actualizaciones
- Sintaxis para:
  - Lenguaje de definición de datos: **DDL** (Data Definition Language)
  - Lenguaje de manipulación de datos: **DML** (Data Manipulation Language)

Lenguaje de definición de datos - DDL

- Permite crear y definir nuevas bases de datos, campos e índices:
  - **CREATE:** Crea nuevas tablas, campos e índices
  - **DROP:** Elimina tablas e índices
  - **ALTER:** Modifica las tablas agregando campos o cambiando la definición de los campos

Lenguaje de definición de datos - DDL

Comando SQL	Descripción
CREATE TABLE	Crea una nueva tabla en la base de datos.
ALTER TABLE	Modifica una tabla existente (estructura y propiedades).
DROP TABLE	Elimina una tabla de la base de datos.
CREATE INDEX	Crea un índice en una o más columnas de una tabla.
DROP INDEX	Elimina un índice existente en una tabla.

Sintaxis para sentencias SQL

- ***{Alternativas}***, entre llaves se colocarán las palabras que tienen opciones o alternativas en la sentencia a la que pertenecen

[Opcional], entre corchetes se colocarán las palabras que son opcionales en la sentencia, es decir que pueden colocarse o pueden obviarse

Creación de Base de Datos

- CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] ***nombre_base_datos***
  - Esta sentencia sirve para crear una base de datos con un nombre específico
  - Para poder crear una base de datos, el usuario que la crea debe tener privilegios de creación asignados
- **IF NOT EXISTS** significa: SI NO EXISTE, por lo tanto, esto es útil para validar que la base de datos sea creada en caso de que no exista, si la base de datos existe y se ejecuta esta sentencia, se genera error
- CREATE SCHEMA o CREATE DATABASE son sinónimos

CREATE DATABASE IF NOT EXISTS complejo_de_cine

Creación de Tablas

- CREATE [TEMPORARY] TABLE [IF NOT EXISTS] nombre_de_tabla
- nombre_de_columna **tipo_de_dato** [NOT NULL | NULL] [DEFAULT valor_por_defecto][AUTO_INCREMENT] [UNIQUE | [PRIMARY] KEY
- [CONSTRAINT [nombre_relación] FOREIGN KEY (nombre_columna) REFERENCES nombre_de_tabla (nombre_columna)]
- [ON DELETE **opciones_de_referencia**]
- [ON UPDATE **opciones_de_referencia**]

Tipo de Datos Más Comunes

- BIT[(longitud)] → indica un valor booleano
- INT[(longitud)] → indica un número entero
- BIGINT[(longitud)]→ indica un número entero grande
- DOUBLE[(longitud,decimales)] → indica un número en punto flotante con precisión
- DATE → almacenar solo el día, mes y año
- TIME → almacena una hora con minutos y segundos
- DATETIME→ almacenar día, mes, año, hora, minuto y segundo
- CHAR[(longitud)] → indica un caracter o cadena de caracteres de longitud fija
- VARCHAR(longitud) → indica una cadena de caracteres de longitud variable según contenido.

Tipo de Datos Numéricos

- TINYINT:

Rango: -128 a 127 (signed), 0 a 255 (unsigned)

- SMALLINT:

Rango: -32,768 a 32,767 (signed), 0 a 65,535 (unsigned)

- MEDIUMINT:

Rango: -8,388,608 a 8,388,607 (signed), 0 a 16,777,215 (unsigned)

- INT (o INTEGER):

Rango: -2,147,483,648 a 2,147,483,647 (signed), 0 a 4,294,967,295 (unsigned)

Tipo de Datos Numéricos

- BIGINT:

Rango: -9,223,372,036,854,775,808 a 9,223,372,036,854,775,807 (signed), 0 a 18,446,744,073,709,551,615 (unsigned)

- DECIMAL o NUMERIC:

Precisión máxima: 65 dígitos decimales

Tamaño: Dependiente de la precisión

- FLOAT:

Precisión: Hasta 7 dígitos

- DOUBLE:

Precisión: Hasta 15 dígitos

Creación de Tablas

Ejemplo de Película

```
CREATE TABLE IF NOT EXISTS `PELICULA` (
 `id_pelicula` INT NOT NULL,
 `anio_estreno` INT,
 `disponible` BIT,
 `duracion` VARCHAR(45),
 `fecha_ingreso` DATETIME,
 PRIMARY KEY (`id_pelicula`))
```

Opciones de Referencias

- Las opciones de referencia sirven para establecer que se hará en casos de que se elimine o se actualice una fila de la tabla primaria que está siendo referenciada por una fila de la tabla secundaria
- **CASCADE**: Eliminar o actualizar la fila de la tabla primaria, y automáticamente eliminar o actualizar las filas coincidentes en la tabla secundaria
- **SET NULL**: Eliminar o actualizar la fila de la tabla primaria, y establecer la columna de clave externa (Foreign key) de la tabla secundaria a NULL. Si se especifica una SET NULL, hay que asegurarse que no se haya declarado la columna de la tabla secundaria como NOT NULL
- **RESTRICT:** Rechaza la operación de eliminación o actualización en la tabla primaria
- **SET DEFAULT:** Para una operación de eliminación o actualización en la tabla primaria se establecerá un valor por defecto para la tabla secundaria

Definición de Opcionalidades

- Las opciones **NOT NULL | NULL**, sirven para especificar si dicha columna puede aceptar valores nulos: NULL, o si no puede guardar valores nulos: NOT NULL
- También se puede de manera opcional, indicar un valor por defecto **DEFAULT** para una columna
- **AUTO_INCREMENT:** se refiere a un valor AUTO INCREMENTAL; sirve para aquellas columnas con valores que numéricos enteros donde se necesita que dicho valor se incremente en uno por cada fila insertada en la tabla. Se utiliza muy frecuentemente en las claves primarias
- **UNIQUE:** sirve para indicar que una columna en la tabla no puede tener valores repetidos, debe ser UNICA. No pueden existir dos filas en la tabla que tengan el mismo valor para un atributo definido como UNIQUE

Definición de Opcionalidades

- **PRIMARY KEY:** sirve para especificar que una columna es clave primaria de una tabla. Si una columna es clave primaria implica que sus valores no deben repetirse (UNIQUE)
- **CONSTRAINT:** significa RESTRICCIÓN y se le asigna un nombre para identificarla; dicho nombre funciona como clave que identifica unívocamente a cada CONSTRAINT que exista en la base de datos, por lo que toda CONSTRAINT debe tener un nombre no repetido; luego se indica con la palabra FOREIGN KEY, cuál es la columna de la tabla que funciona como clave foránea, indicando a continuación a cuál tabla y a cuál columna de dicha tabla hace referencia la clave foránea con la palabra REFERENCES

Alteración de Tabla

- Las acciones mas comunes que se realizan sobre una **Tabla** mediante la alteración son:
- **ADD:** COLUMN, INDEX, KEY, CONSTRAINT
- **CHANGE:** COLUMN
- **MODIFY:** COLUMN
- **DROP:** COLUMN. PRIMARY KEY, INDEX, KEY, FOREING KEY

Sentencia completa ver:
https://dev.mysql.com/doc/refman/8.0/en/alter-table.html

ALTERACION

Operación	Sintaxis SQL	Descripción
Agregar una columna	`` `ALTER TABLE table_name ADD column_name datatype;` ``	Añade una nueva columna a la tabla.
Eliminar una columna	`` `ALTER TABLE table_name DROP COLUMN column_name;` ``	Elimina una columna de la tabla.
Cambiar el nombre de una columna	`` `ALTER TABLE table_name CHANGE old_column_name new_column_name datatype;` ``	Cambia el nombre de una columna.
Modificar una columna (utilizando MODIFY)	`` `ALTER TABLE table_name MODIFY column_name new_datatype;` ``	Modifica el tipo de datos de una columna.
Modificar una columna (utilizando ALTER)	`` `ALTER TABLE table_name ALTER COLUMN column_name SET DATA TYPE new_datatype;` ``	Modifica el tipo de datos de una columna.
Cambiar la definición de una columna	`` `ALTER TABLE table_name ALTER COLUMN column_name SET DEFAULT default_value;` ``	Cambia la definición predeterminada de una columna.
Agregar un índice	`` `ALTER TABLE table_name ADD INDEX index_name (column_name);` ``	Añade un índice a la columna especificada.
Eliminar un índice	`` `ALTER TABLE table_name DROP INDEX index_name;` ``	Elimina un índice de la tabla.

Alteración de Tabla

Ejemplos

- Para agregar una nueva columna ***d***, de tipo DATETIME

```
ALTER TABLE t2
        ADD d DATETIME;
```

- Para cambiar los tipos de datos de sus atributos, suponiendo que tenemos dos atributos ***a*** es INTEGER y ***b*** es CHAR con longitud 10: CHAR (10), cambiaremos ***a*** como TINYINT sin aceptar nulos y cambiaremos ***b*** por el nombre ***c*** y con longitud 20:

```
ALTER TABLE t2
        MODIFY a TINYINT NOT NULL,
        CHANGE b c CHAR(20);
```

Alteración de Tabla

Ejemplos

- Para cambiar el nombre de una tabla de ***t1*** a ***t2***:

```
ALTER TABLE t1
        RENAME t2;
```

- Para agregar un nuevo índice en una columna ***d***:

```
ALTER TABLE t2
        ADD INDEX (d);
```

- Para eliminar la columna ***c***:

```
ALTER TABLE t2
        DROP COLUMN c;
```

Alteración de Tabla

Ejemplos

- Para agregar una nueva columna auto-incremental y clave primaria, de tipo entera llamada ***e***:

```
ALTER TABLE t2
        ADD e INT NOT NULL AUTO_INCREMENT,
        ADD PRIMARY KEY (e);
```

Borrado de Tabla

- **DROP TABLE** remueve una o más tablas. Para poder eliminar tablas, el usuario que ejecuta la sentencia debe tener privilegios de **DROP** para cada tabla que quiera eliminar

Con este comando se eliminan todos los datos de la tabla, tanto la definición como las filas de datos de la misma. Si no existe la tabla que se desea borrar MySQL devuelve error, por eso es útil utilizar la opción IF EXISTS en caso de que exista la tabla la elimina

DROP [TEMPORARY] TABLE [IF EXISTS]

nombre_tabla [,nombre_tabla] ...

Creación de índices

- Un índice es una estructura de disco asociada con una tabla o una vista que acelera la recuperación de filas de la tabla
- Un índice contiene claves generadas a partir de una o varias columnas de la tabla. Dichas claves están almacenadas en una estructura que permite que SQL Server busque de forma rápida y eficiente la fila o filas asociadas a los valores de cada clave

CREATE INDEX *nombre_indice*

ON *nombre_tabla* (*col_name* [(*length*)] [ASC | DESC]) ...

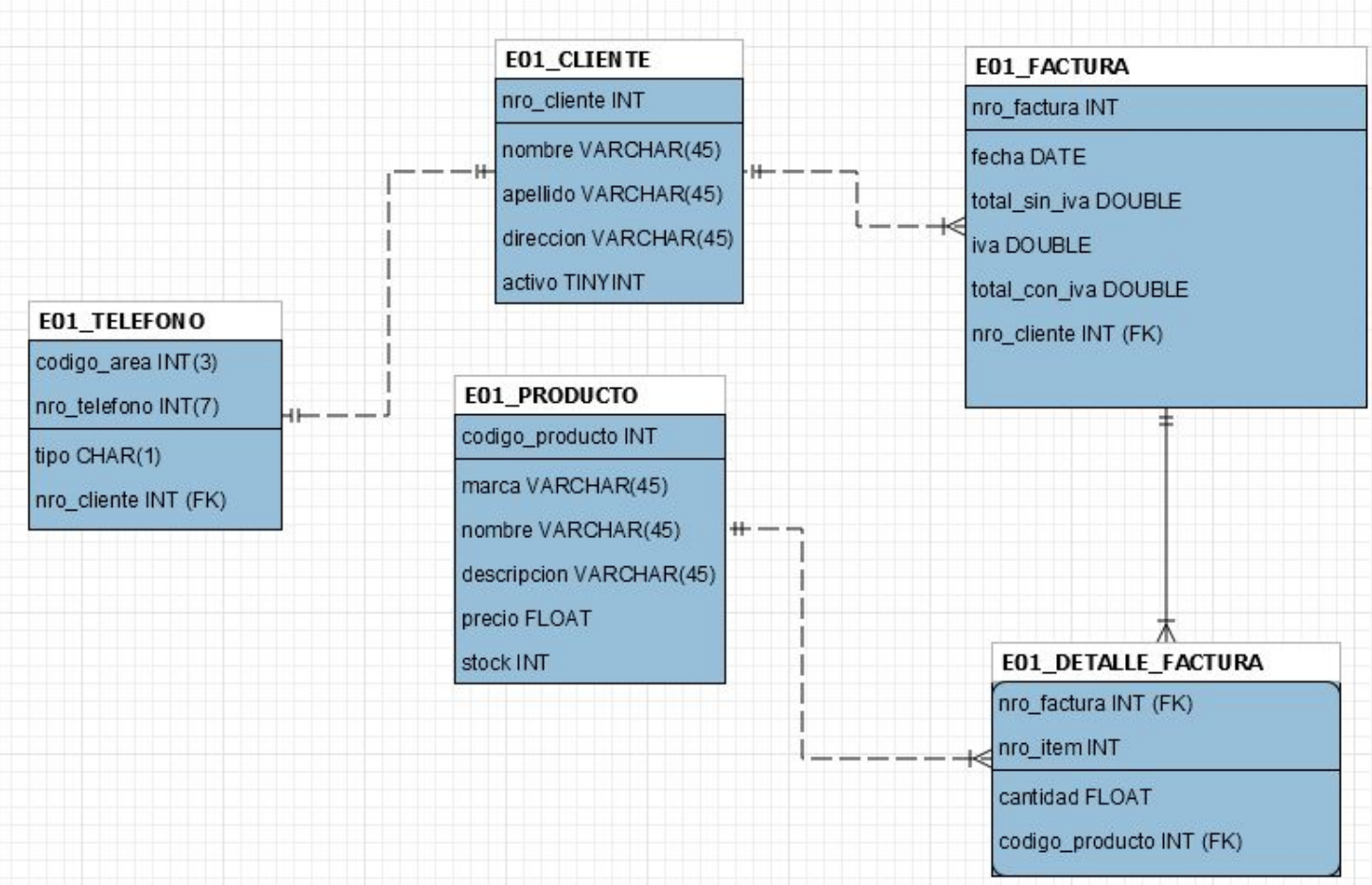
Bases de Datos

Carrera Programador full-stack

Lenguaje de DDL (Ejercicios)

Creación de Esquemas

Ejemplo Facturación



Creación de Esquemas

Ejemplo Facturación

- Dado el Modelo de Entidad-Relación de la figura anterior, crear las siguientes tablas con DDL:
  - E01_TELEFONO
  - E01_CLIENTE
  - E01_PRODUCTO
  - E01_FACTURA
  - E01_DETALLE_FACTURA

Bases de Datos

Carrera Programador full-stack

Lenguaje de DDL (Resolución)

Creación de Tablas

```
CREATE TABLE IF NOT EXISTS E01_CLIENTE` (
 `nro_cliente` INT NOT NULL,
 `nombre` VARCHAR(45) NOT NULL,
 `apellido` VARCHAR(45) NOT NULL,
 `direccion` VARCHAR(45) NOT NULL,
 `activo` TINYINT NOT NULL,
 PRIMARY KEY (`nro_cliente`))
```

Creación de Tablas

```
CREATE TABLE IF NOT EXISTS `bd111mil`.`E01_TELEFONO` (
 `codigo_area` INT(3) NOT NULL,
 `nro_telefono` INT(7) NOT NULL,
 `tipo` CHAR(1) NOT NULL,
 `nro_cliente` INT NOT NULL,
 PRIMARY KEY (`codigo_area`, `nro_telefono`),
 FOREIGN KEY (`nro_cliente`)  REFERENCES
   `bd111mil`.`E01_CLIENTE` (`nro_cliente`) ON DELETE NO
ACTION ON UPDATE NO ACTION)
```

Creación de Tablas

```
CREATE TABLE IF NOT EXISTS `bd111mil`.`E01_FACTURA` (
 `nro_factura` INT NOT NULL,
 `fecha` DATE NOT NULL,
 `total_sin_iva` DOUBLE NOT NULL,
 `iva` DOUBLE NOT NULL,
 `total_con_iva` DOUBLE GENERATED ALWAYS AS (total_sin_iva+iva)
   VIRTUAL,
 `nro_cliente` INT NOT NULL,
 PRIMARY KEY (`nro_factura`),
 FOREIGN KEY (`nro_cliente`)  REFERENCES `bd111mil`.`E01_CLIENTE`
   (`nro_cliente`)  ON  DELETE  NO ACTION  ON UPDATE NO ACTION)
```

Creación de Tablas

```
CREATE TABLE IF NOT EXISTS
  `bd111mil`.`E01_PRODUCTO` (
 `codigo_producto` INT NOT NULL,
 `marca` VARCHAR(45) NOT NULL,
 `nombre` VARCHAR(45) NOT NULL,
 `descripcion` VARCHAR(45) NOT NULL,
 `precio` FLOAT NOT NULL,
 `stock` INT NOT NULL,
 PRIMARY KEY (`codigo_producto`))
```

Creación de Tablas

```
CREATE TABLE IF NOT EXISTS `bd111mil`.`E01_DETALLE_FACTURA` (
 `nro_factura` INT NOT NULL,
 `nro_item` INT NOT NULL,
 `cantidad` FLOAT NOT NULL,
 `codigo_producto` INT NOT NULL,
 PRIMARY KEY (`nro_factura`, `nro_item`),
 FOREIGN KEY (`codigo_producto`)
   REFERENCES `bd111mil`.`E01_PRODUCTO` (`codigo_producto`)
   ON DELETE NO ACTION
   ON UPDATE NO ACTION,
   FOREIGN KEY (`nro_factura`)
   REFERENCES `bd111mil`.`E01_FACTURA` (`nro_factura`)
   ON DELETE NO ACTION   ON UPDATE NO ACTION)
```

Bases de Datos

Carrera Programador full-stack

Lenguaje de DLL (Repaso)

Lenguaje de Consulta – Creación de Esquemas

Repaso

- La definición de los datos se realiza a través de las sentencias de DDL (Data Definition Language) del SQL
- Sus comandos permiten definir la semántica del esquema relacional: qué tablas o relaciones se establecen, sus dominios, asociaciones, restricciones, etc
- Las tablas son indentificadas unívocamente por sus nombres y contienen filas y columnas
- Una tabla en una base de datos relacional es similar a una tabla en papel, posee filas y columnas

Lenguaje SQL

Algunas funciones del estándar SQL son:

- DDL
  - **Definición de datos:**
    - **Creación de tablas (CREATE)**
    - **Modificación de tablas (ALTER)**
    - **Eliminación de tablas (DROP)**
- DML
  - Consulta de datos
    - Selección (SELECT)
  - Actualización de los datos
    - Inserción (INSERT)
    - Actualización (UPDATE)
    - Eliminación (DELETE)