Decroly SANTANDER

UD2

IMPLEMENTACIÓN DE BASES DE DATOS

MP_0484
Bases de Datos

2.2 Creación de bases de datos

Introducción

Tal y como vimos en la anterior presentación, vamos a enfocar nuestra implementación hacia MySQL. Esto significa que ya deberíamos haber creado un diseño sólido, no redundante y robusto con la integridad referencial necesaria. Ten en cuenta que incluso si los errores se pueden corregir en esta etapa, esto nos llevará más tiempo.

Tipos de lenguajes en el contexto de BBDD

El lenguaje va a ser la herramienta que utilicemos en nuestra interacción con la base de datos. Pero, en primer lugar, ¿por qué necesitamos un idioma a la hora de crear una base de datos?

- **Control**. Los lenguajes de base de datos proporcionan comandos para controlar el acceso a los datos y administrar otros aspectos de la base de datos.
- **Comunicación**. El lenguaje actúa como un medio para que los humanos instruyan al DBMS sobre qué acciones realizar.
- **Estandarización**. Un lenguaje estandarizado garantiza la coherencia y la comprensión entre los diferentes sistemas y usuarios.
- **Automatización.** La automatización permite la automatización de tareas y la capacidad de programar operaciones complejas que se pueden repetir según sea necesario

Dentro de los lenguajes utilizados en la implementación de bases de datos, podemos establecer dos grandes divisiones:

- **DDL (Data Definition Language)**. Su utilidad pasa por definir y modificar la estructura de la base de datos, incluyendo las tablas, vistas, índices y otros objetos. Estos comandos nos permiten:
    - CREATE establece el esquema inicial de la base de datos
    - ALTER Modifica las estructuras existentes
    - DROP elimina las estructuras cuando ya no sean necesarias
- **DML (Data Manipulation Language)**. DML nos permite introducir y manipular los datos de la base de datos. It includes commands like:
    - **SELECT** recuperar datos de la base de datos
    - **INSERT** agrega nuevos datos a las tablas
    - **UPDATE** modifica los datos existentes
    - **DELETE** elimina datos

Cabe destacar que SQL cuenta con ambas vertientes. En esta primera aproximación a la implementación de bases de datos vamos a trabajar con la dimensión DDL del lenguaje, pues es quien define y modifica la estructura de la base de datos.

¿Por qué SQL?

SQL (Structured Query Language) es el lenguaje de programación más popular utilizado para interactuar con bases de datos relacionales. Desarrollado a principios de la década de 1970 por IBM, su diseño ofrecía una sintaxis más legible, similar al inglés, que permitía a los usuarios escribir consultas que podían entenderse y utilizarse en varios sistemas de bases de datos.

El aspecto DDL de SQL permite a los usuarios definir y modificar la estructura de los objetos de la base de datos a través de comandos como CREATE, ALTER y DROP. Estos comandos se utilizan para configurar tablas y otras estructuras, modificarlas a medida que los sistemas evolucionan y eliminarlas cuando ya no son necesarias.

Por otro lado, la funcionalidad DML de SQL se centra en la manipulación de los propios datos dentro de las estructuras de base de datos definidas por DDL. Incluye comandos como SELECT, INSERT, UPDATE y DELETE, que se utilizan para leer y modificar los datos.

Respaldado como el lenguaje estándar para los sistemas de gestión de bases de datos relacionales por ANSI e ISO, la amplia adopción de SQL es un testimonio de su versatilidad. Integra a la perfección las capacidades de DDL y DML, ofreciendo un conjunto de herramientas completo para la configuración, manipulación y administración de bases de datos

Creación de la base de datos

En primer lugar, usaremos estas sentencias para crear la estructura de la base de datos:

```
DROP DATABASE IF EXISTS nombreBD;
CREATE DATABASE IF NOT EXISTS nombreBD;
USE nombreBD;
```

A continuación, se lleva a cabo la creación de las diferentes tablas de la base de datos. Aunque puede parecer un poco intimidante replicar nuestro modelo relacional, pues hay muchas partes de a especificar y muchas opciones para cada elemento, en la práctica, crear una nueva tabla es sencillo. Veamos cómo se hace.

Creación de una tabla

Veamos a continuación la sentencia SQL más básica para crear tablas:

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] nombre_tabla (
definicion_col [tipo_dato] [restricción_col]);

Y ahora, cada uno de los componentes:

- **nombre_tabla**: aquí se indica el nombre con el que queremos que se conozca la nueva tabla. Si la BD en que queremos crear la tabla no está activa o es una BD externa, debemos indicar el nombre de la siguiente forma nombreBD.nombreTabla.
- **definición_col**: permite definir las columnas que compondrán la tabla; tiene una sintaxis especial para definir los tipos de campos que se verán a continuación.
- **restricción_col**: en esta parte colocamos las restricciones que queremos que contenga nuestra tabla. Esto es opcional, puede no contener restricciones. Lo más habitual es que las tablas tengas las restricciones de primary key, foreing key, check, etc.

Definir columnas

nombre_columna tipo_dato [(dimensión[, precisión])] [restric_col],

Formato nombre_columna tipo:dato [(dimensión, tam)] [DEFAULT def_defecto] [restric_col]

- **nombre_columna**: indica el nombre con el que queremos conocer la columna, debe ser único dentro de la tabla.
- **tipo_dato**: debe indicar el tipo de datos que contendrá la columna. Es obligatorio ponerlo. Aunque depende del tipo de dato, generalmente, puede ponerse el tamaño del dato entre paréntesis (dimensión, precisión).
- **restric_col**: se establecen restricciones que queremos que contenga la columna, entre otros, están:
    - **NOT NULL**: no puede tener un valor vacío.
    - **UNIQUE**: todos los valores de esa columna son distintos.
    - **CHECK**: la restricción CHECK permite establecer condiciones de integridad que debe cumplir ese campo. Será imposible asegurar que los usuarios introducen bien los datos, pero se pueden establecer controles en algún campo para que, en caso de que no se cumpla, no deje dar de alta

Creación de claves primarias

La clave primaria es la columna o columnas de la tabla que identifican a un registro de forma única. Por lo tanto, no puede ser NULL ni puede haber dos registros con el mismo valor UNIQUE. Para denotar a una columna como PK, podemos hacerlo tras haber definido todas las columnas de la tabla de la siguiente manera:

CONSTRAINT nombrePK PRIMARY KEY (nombre_col),

Creación de claves externas

La clave ajena o foránea es la columna de la tabla que se asocia a una clave principal de otra tabla. Por lo tanto, el valor que se encuentra en esa columna identifica unívocamente a un registro de la otra tabla. Podemos denotar esta circunstancia en restricciones de la tabla:

CONSTRAINT nombreFK FOREIGN KEY (nombre_col) REFERENCES tabla(PK_tabla)

Ejemplos

Veamos a continuación una base de datos simple con dos tablas.

```
DROP DATABASE IF EXISTS Geography;
CREATE DATABASE IF NOT EXISTS Geography;
USE Geography;

CREATE TABLE province(
    province_code CHAR(2),
    name VARCHAR(50),
    CONSTRAINT province_code PRIMARY KEY (province_code)
);

CREATE TABLE city(
    province_code CHAR(2),
    city_code CHAR(4),
    name VARCHAR(50),
    CONSTRAINT city_pk PRIMARY KEY (province_code,city_code),
    CONSTRAINT province_code FOREIGN KEY (province_code) REFERENCES     province
(province_code)
);
```