UD2

IMPLEMENTACIÓN DE BASES DE DATOS

MP_0484

Bases de Datos

2.1 Introducción y tipos de datos

Introducción

Es el momento de crear la base de datos (BD) y las tablas que conforman esta. Es importante disponer de un buen diseño lógico y tener bien definidas las características de los campos, el rango de valores de cada campo y las restricciones entre tablas. Para implementar todo esto, crearemos las tablas y el resto de los elementos que son los que van a ser el soporte de todas esas características y consideraciones de nuestra BD.

Conocer todo lo que se puede se definir e implementar lo necesario es lo que nos va a permitir crear una BD eficiente y con datos consistentes. El objetivo de esta unidad es crear todo lo necesario para almacenar los datos, lo que se denomina el diseño físico.

Creación, modificación y eliminación de la BD

En esta unidad crearemos nuestra base de datos y la cargaremos con datos. Las estructuras vistas hasta ahora son el esquema conceptual (diagrama entidad/relación) y el esquema lógico relacional (diagrama relacional). En este tema se va a crear el diseño físico de la base de datos, tal y como se describe en la figura 1.

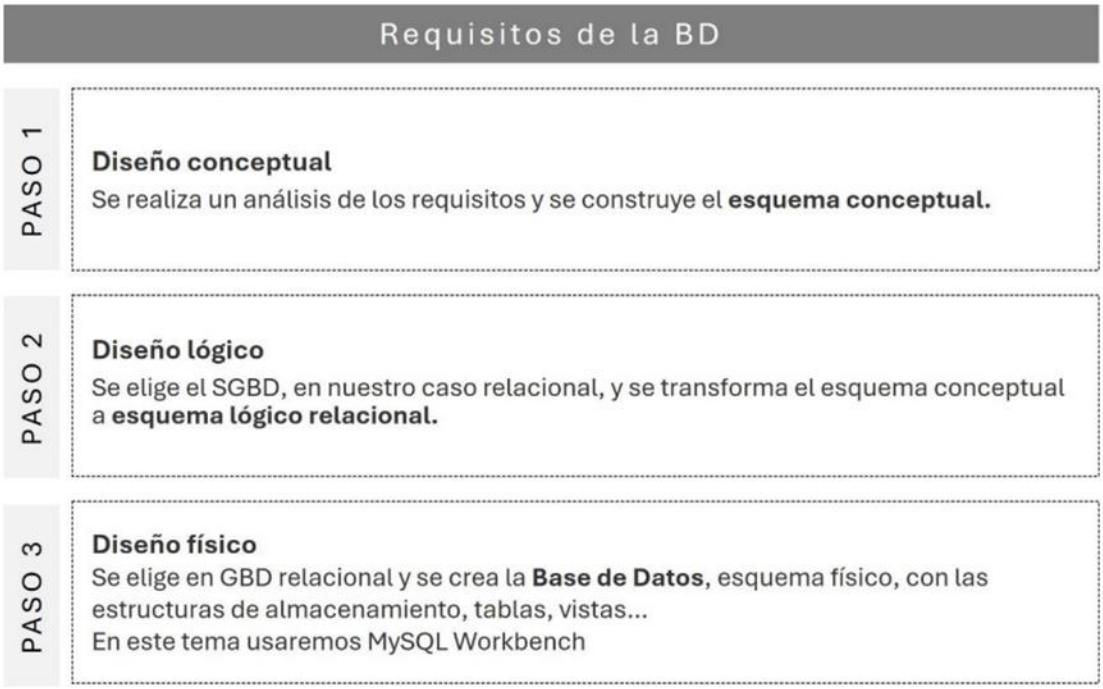

Figura 1. Etapas de diseño de la base de datos

Creación de tablas

El diseño físico de una base de datos es la etapa en la que se determina cómo se almacenarán físicamente los datos en los dispositivos de almacenamiento. Este proceso implica tomar decisiones sobre la estructura de almacenamiento, las tablas, el diseño de los índices, vistas y otras optimizaciones que pueden afectar significativamente el rendimiento y la eficiencia de la base de datos.

El elemento principal en las BD relacionales son las tablas, cada fila se denomina registro, las columnas o campos de la tabla se corresponden con los atributos, el conjunto finito de posibles valores que puede tener un campo se llama dominio. El número de filas de la tabla se denomina cardinalidad y el número de columnas, grado.

Tipos de datos

Antes de comenzar a crear las tablas, debemos tener claro los tipos de datos con los que se trabaja en MySQL WorkBench. Los tipos de datos son categorías que definen la naturaleza de los datos que se pueden almacenar y manipular en un programa o base de datos. Estos tipos determinan el tamaño, el rango y el tipo de operaciones que se pueden realizar con los datos.

En el siguiente listado encontrarás señalados en **negrita** los tipos de datos más comúnmente empleados en MySQL WorkBench, pero ten en cuenta que puedes utilizar el que mejor se adecúe a tus necesidades.

- **Datos de tipo numérico**
  - BIT[(M)]. Un tipo de valor de bit. M indica el número de bits por valor, de 1 a 64. El valor predeterminado es 1 si M se omite.
  - TINYINT[(M)] [UNSIGNED] [ZEROFILL]. Un número entero muy pequeño. El rango con signo va de -128 a 127. El rango sin signo va de 0 a 255.
  - **BOOL, BOOLEAN**. Estos tipos son sinónimos de TINYINT(1). Un valor de cero se considera falso. Los valores distintos de cero se consideran verdaderos.
  - SMALLINT[(M)] [UNSIGNED] [ZEROFILL]. Un número entero pequeño. El rango con signo va de -32768 a 32767. El rango sin signo va de 0 a 65535.
  - MEDIUMINT[(M)] [UNSIGNED] [ZEROFILL]. Un número entero de tamaño mediano. El rango con signo va de -8388608 a 8388607. El rango sin signo va de 0 a 16777215.

- **INT[(M)] [UNSIGNED] [ZEROFILL]**. Un número entero de tamaño normal. El rango con signo va de -2147483648 a 2147483647. El rango sin signo va de 0 a 4294967295.
- INTEGER[(M)] [UNSIGNED] [ZEROFILL]. Este tipo es sinónimo de INT.
- BIGINT[(M)] [UNSIGNED] [ZEROFILL]. Un número entero grande. El rango con signo va de -9223372036854775808 a 9223372036854775807. El rango sin signo va de 0 a 18446744073709551615.
- **DECIMAL[(M[,D])] [UNSIGNED] [ZEROFILL]**. Un número de punto fijo «exacto». M es el número total de dígitos (la precisión) y D es el número de dígitos después del punto decimal (la escala). El punto decimal y (para números negativos) el - signo no se cuentan M. Si D es 0, los valores no tienen punto decimal ni parte fraccionaria. El número máximo de dígitos (M) decimal es 65. El número máximo de decimales admitidos (D) es 30. Si D se omite, el valor predeterminado es 0. Si M se omite, el valor predeterminado es 10.
- FLOAT[(M,D)] [UNSIGNED] [ZEROFILL]. Un número pequeño (de precisión simple) de punto flotante. Los valores permitidos van de -1.175494351E-38 a -1.175494351E-38, mientras que sin signo va de 0 a 3.402823466E+38.
- FLOAT(p) [UNSIGNED] [ZEROFILL]. Un número de punto flotante. La precisión en bits la representa p, pero MySQL usa este valor solo para determinar si se usa FLOAT o DOUBLE para el tipo de datos resultante. Si p es de 0 a 24, el tipo de datos pasa FLOAT a tener ningún valor M o D. Si p es de 25 a 53, el tipo de datos pasa a DOUBLE tener ningún valor M o D.
- DOUBLE[(M,D)] [UNSIGNED] [ZEROFILL]. Un número de punto flotante de tamaño normal (doble precisión). Los valores permitidos van de -2.2250738585072014E-308 a -2.2250738585072014E-308.

- **Datos de tipo fecha**. Los campos de tipo fecha utilizan valores de fecha en el formato YYYY-MM-DD (año, mes, día). Para convertir la fecha a otros formatos una función muy útil sería SRT_TO_DATE(). Las fechas que contienen valores de años de dos dígitos son ambiguas porque se desconoce el siglo. Los valores de año en el rango 70 - 99 se convierten a 1970 -1999 y el rango 00-69 se convierte en 2000 – 2069.
  - **DATE()**. Se utiliza para valores con una parte de fecha, pero sin parte de hora. El rango admitido es de (YYYY-MM-DD) 1000-01-01 a 9999-12-31.
  - **DATETIME()**. Se utiliza para valores que contienen partes de fecha y hora. El rango admitido es de (YYYY-MM-DD hh:mm:ss) 1000-01-01 00:00:00 a 9999-12-31 23:59:59.

- **TIMESTAMP()**. Se utiliza para valores que contienen partes de fecha y hora. Tiene un rango de 1970-01-01 00:00:01 UTC a 2038-01-19 03:14:07 UTC.
- TIME(). Recupera y muestra valores hora en formato hh:mm:ss.
- YEAR(). Muestra valores de año (YYYY), con un rango de 1901 a 2155 y 0000.
- NOW(). Podemos usar esta función si queremos guardar la fecha y hora actuales del sistema, devolviéndonos un tipo de dato **DATETIME()**.

- **Datos de tipo alfanuméricos**
  - **CHAR(n) o VARCHAR(n)**. Para almacenar datos de tipo carácter, siendo n el número de posiciones entre 0 y 255 en el caso de CHAR y hasta 65535 en VARCHAR. VARCHAR(n) requiere de espacio de almacenamiento solo lo que ocupa, mientras que con CHAR(n), se guardan las n posiciones, aunque no se utilicen. En general, CHAR es más eficiente en los casos en los que la longitud siempre vaya a ser la misma.
  - BINARY y VARBINARY son similares a CHAR y VARCHAR, excepto que almacenan cadenas binarias.