Bases de Datos

Unidad 3:

Diseño Físico de Bases de Datos

UNIDAD 3: DISEÑO ´FÍSICO DE BASES DE DATOS

1. Características físicas del almacenamiento de la base de datos
2. Herramientas gráficas para la implementación de la base de datos.
3. El lenguaje de definición de datos.
4. Creación modificación y eliminación de bases de datos.
5. Tipos de datos. Valores. Operadores
6. **Administración de tablas**
    - **6.1 Sintaxis de la instrucción CREATE TABLE**
    - **6.2 Propiedades de tablas**
    - 6.3 Modificación de tablas
        - 6.3.1 ALTER TABLE
        - 6.3.2 CREATE INDEX
        - 6.3.3 DROP INDEX
        - 6.3.4 RENAME TABLE
    - 6.4 Eliminación de tablas
7. Vistas
8. El lenguaje de control de datos.
9. Usuarios y privilegios

6. Administración de tablas

Las instrucciones DDL para administrar tablas permiten:

- Crear un tabla con un nombre, definiendo, además, las columnas que contiene, sus tipos y las restricciones.
- Establecer propiedades de una tabla.
- Modificar la estructura de una tabla (añadir una columna, eliminar una columna, modificar las columnas PRIMARY KEY, añadir una FOREIGN KEY, etc.)
- Eliminar una tabla.
- Crear un índice.
- Eliminar un índice.
- Renombrar una tabla.

6.1 Sintaxis de la instrucción CREATE TABLE

La instrucción SQL para crear una tabla es **CREATE TABLE**. La sintaxis completa de esta instrucción es bastante compleja. La puedes ver dentro de la documentación oficial de MySQL.

https://dev.mysql.com/doc/refman/5.7/en/create-table.html

Vamos a explicar la sintaxis de una forma más simple:

```
CREATE  [TEMPORARY] TABLE  [IF NOT EXISTS] nombre_tabla
(
        nombre_columna1   tipo   [restricciones_tipo_1],
        nombre_columna2   tipo   [restricciones_tipo_1],
        ……….
        [restricción_tipo_2 ],
        [restricción_tipo_2 ],
        ……..
)  [opciones_tabla];
```

6.1 Sintaxis de la instrucción CREATE TABLE

Interpretación de la sintaxis:

1.- La cláusula TEMPORARY hace que la tabla creada es temporal para la sesión cliente en ejecución. Al salir de la sesión la tabla se elimina automáticamente.

2.- La cláusula **IF NOT EXISTS** hace que el servidor no devuelva un error cuando se intenta crear una tabla y ya existe.

3.- Se abre y cierra un paréntesis. Entre el paréntesis se escriben separadas por comas todas las definiciones de columnas. Después de las definiciones de las columnas se escriben separadas por comas, si las hay, las definiciones de restricciones en la tabla.

6.1 Sintaxis de la instrucción CREATE TABLE

Interpretación de la sintaxis:

4.- Al finalizar la definición de columnas se escriben todas las opciones o propiedades de la tabla. Si no se escribe ninguna se establecen las propiedades por defecto.

5.- En la definición de cada columna se indica su nombre, su tipo (con los modificadores) y restricciones que se le aplican:

PRIMARY KEY no es recomendable usarla en la definición de columnas
UNIQUE
NOT NULL
DEFAULT valor
AUTO_INCREMENT
GENERATED ALWAYS AS (expresión)

Ejemplo 1:

*Crear una tabla **familiasprof** que almacenará las familias profesionales de FP. La tabla tiene una columna código de la familia que se representa con tres letras y un nombre de la familia profesional. Esas columnas no admiten nulos.*

```
CREATE TABLE familiasprof (
  codfamilia CHAR(3) NOT NULL,
  nomfamilia VARCHAR(50) NOT NULL);
```

*Crear la tabla **familiasprof** para que reciba en nomfamilia el valor "desconocida" cuando no se introduzca el nombre de una familia al insertar una fila.*

```
CREATE TABLE familiasprof (
  codfamilia CHAR(3) NOT NULL,
  nomfamilia VARCHAR(50) NOT NULL DEFAULT 'desconocida' );
```

6.1 Sintaxis de la instrucción CREATE TABLE

Interpretación de la sintaxis:

6.- Restricciones tipo 2. Se pueden definir las siguientes restricciones que se aplican a una sola columna o a varias columnas de la tabla:

```
a.- [CONSTRAINT [simbolo]] PRIMARY KEY (columna1,...)

b.- INDEX [nom_indice] (columna 1,...)

c.- [CONSTRAINT [simbolo]] UNIQUE [nom_indice] (columna
1,...)

d.- FULLTEXT[nom_indice] (columna 1,...)

e.- [CONSTRAINT [simbolo]] FOREIGN KEY (colAjena1,...)
REFERENCES tblReferenciada (colReferenciada 1,...)
[ON DELETE opcion_integridad][ON UPDATE opcion_integridad]
```

Interpretación de la sintaxis:

7.- Restricciones de comportamiento de FOREIGN KEY para modificación (ON UPDATE).

ON UPDATE RESTRICT : No se puede modificar un valor de la clave primaria en la tabla referenciada si se tienen filas con ese valor en la clave ajena de la tabla hija. Por ejemplo, si en una tabla paises se intenta modificar el identificador de España, no se podría hacer la modificación si en una tabla ciudades hubiera ciudades con ese código de país en su clave ajena.
ON UPDATE NO ACTION : Comportamiento por defecto, idéntico a RESTRICT.
ON UPDATE CASCADE : *Lo más adecuado normalmente.* Si se modifica el valor de la clave primaria en la tabla referenciada, se modifican los valores en la clave ajena de todas las filas de la tabla hija que tuvieran el valor modificado. Por ejemplo, si en la tabla paises se modifica el código de España, en todas las filas de la tabla de ciudades cuyo código de país fuera El de España, se modificaría ese valor.
ON UPDATE SET NUL : Si se modifica el valor de la clave primaria en la tabla referenciada, se ponen a NULL o valor nulo los valores en la clave ajena de todas las filas de la tabla hija que tuvieran el valor modificado.

Interpretación de la sintaxis:

7.- Restricciones de comportamiento de FOREIGN KEY para borrado (ON DELETE).

ON DELETE RESTRICT : No se puede eliminar una fila en la tabla referenciada si se tienen filas relacionadas por clave ajena de la tabla hija. Por ejemplo, si en una tabla paises se intenta eliminar la fila de España, no se podría hacer la eliminación si en una tabla ciudades hubiera ciudades de españa (con la clave ajena de pais al valor de España).
ON DELETE NO ACTION : Comportamiento por defecto, idéntico a RESTRICT.
ON DELETE CASCADE : *Es peligroso usarlo porque puede eliminarnos mucha información.* Si se elimina una fila en la tabla referenciada, se eliminan todas las filas relacionadas en la tabla hija. Por ejemplo, si en la tabla paises se elimina la fila de España, se eliminarían en la tabla ciudades todas las ciudades de España.
ON DELETE SET NUL : Si se elimina una fila en la tabla referenciada, se pone a nulo el valor de la clave ajena de todas las filas relacionadas en la tabla hija. Por ejemplo, si en la tabla paises se elimina la fila de España, se establecerá que el país de todas las ciudades que pertenecían a España es NULL.

6.1.1 Tipos de índices

Los índices son referencias o punteros que apuntan a las filas que contienen un valor en una o varias columnas. Su función es mejorar el rendimiento en tablas muy grandes haciendo que las consultas se hagan más rápido.

*Por ejemplo, si tenemos en un tabla los datos de todos los alumnos de Cantabria y tenemos en esa tabla la columna localidad de residencia, ésta columna se podría establecer como índice. De esta forma, una consulta que busca los alumnos de "**Santillana del Mar**" se haría más rápido que si se hiciera sin ser localidad un índice.*

Pero no se debe abusar de los índices ya que:

- Ralentizan las operaciones de inserción y modificación de datos.
- La base de datos ocupa mayor espacio en disco.

Tipos de índices:

- **PRIMARY KEY**

- **UNIQUE:** Índice que no admite valores repetidos. Se pueden declarar varios índices UNIQUE en una tabla y un índice UNIQUE se puede aplicar a una sola columna o a varias columnas.

- **INDEX:** Es un índice normal que admite valores repetidos.

- **FULLTEXT**: Estos índices se emplean para realizar búsquedas sobre texto (CHAR, VARCHAR y TEXT). Estos índices permiten buscar palabras dentro de los contenidos de las columnas.

- *Cuando se crea una clave ajena (FOREIGN KEY) en una columna, se establece que esa columna sea INDEX salvo que ya estuviera declarada con algún tipo de índice.*

6.1.2 Ejemplos de instrucción CREATE TABLE

Ejemplo 2:

*Crear una tabla **familiasprof** que almacenará las familias profesionales de FP. La tabla tiene una columna código de la familia que se representa con tres letras y es PRIMARY KEY y un nombre de la familia que no se puede repetir,*

```
CREATE TABLE familiasprof (
  codfamilia CHAR(3) NOT NULL,
  nomfamilia VARCHAR(50) NOT NULL,
  PRIMARY KEY(codfamilia),
  UNIQUE(nomfamilia));
```

6.1.2 Ejemplos de instrucción CREATE TABLE

Ejemplo 3:

*Crear una tabla **centros** que tiene las columnas código del centro, nombre del centro, localidad y unidades que tiene el centro. El código de centro es un entero sin signo que se genera automáticamente por autoincremento. El código se representa siempre con tres cifras. El nombre de centro no admite valores repetidos en la tabla En unidades se carga por defecto 1. Ninguna columna admite nulos.*

```
CREATE TABLE centros (
  codcentro INT(3) UNSIGNED ZEROFILL NOT NULL AUTO_INCREMENT,
  nomcentro VARCHAR(45) NOT NULL,
  localidad VARCHAR(30) NOT NULL,
  unidades TINYINT NOT NULL DEFAULT 1,
  PRIMARY KEY (codcentro),
  UNIQUE uk_nomcentro (nomcentro)
 ) ;
```

6.1.2 Ejemplos de instrucción CREATE TABLE

Ejemplo 4:

*Crear una tabla **ciclos** que tiene información de todos los ciclos formativos de FP. Cada ciclo tiene un código que es clave primaria y que está formado por el código de la familia a la que pertenece y un número de tres cifras que se rellena con ceros para las no significativas. Además un ciclo tiene un nombre de hasta 100 caracteres y con una letra se indica si es de grado superior, medio o de FP básica. El grado de ciclo admite nulos.*

```
CREATE TABLE ciclos (
  familia char(3) NOT NULL,
  numero int(3) unsigned zerofill NOT NULL,
  nombreciclo varchar(100) NOT NULL,
  grado char(1),
 PRIMARY KEY (familia,numero),
  CONSTRAINT fk_ciclos_familias FOREIGN KEY (familia) REFERENCES
familiasprof (codfamilia) ON DELETE NO ACTION ON UPDATE CASCADE
 ) ;
```

Opciones o propiedades de tabla I:

Como hemos visto en la sintaxis de CREATE TABLE, después de la definición de columnas y restricciones, podemos establecer las opciones o propiedades de tabla:

```
CREATE  [TEMPORARY] TABLE  [IF NOT EXISTS] nombre_tabla
(
        nombre_columna1         tipo restricciones_tipo_1,
        nombre_columna2         tipo restricciones_tipo_1,
        ……….
        restricción_tipo_2                  columnas_a_las_que_se_aplica,
        restricción_tipo_2                  columnas_a_las_que_se_aplica,
        ……..
) [opciones_tabla];
```

Si se establecen varias opciones, éstas se separan simplemente con espacio.

Opciones o propiedades de tabla II:

ENGINE = {BDB | HEAP | ISAM | InnoDB | MERGE UNION=| MRG_MYISAM | MYISAM }

Sirve para indicar el motor de almacenamiento de la tabla. Si no se indica será de tipo InnoDB.

Tablas InnoDB: Son tablas de transacción segura, es decir, sobre ellas se pueden usar las sentencias COMMIT o ROLLBACK para confirmar o anular un proceso de transacción que se ha iniciado anteriormente. Admiten control de integridad referencial.
Tablas MyISAM: Son tablas que usan pocos recursos para el almacenamiento. Los procesos son más rápidos. No admiten transacciones seguras ni integridad referencial.
Tablas MERGE: Son tablas que se definen como resultado de una unión entre dos tablas que tienen las mismas columnas con el mismo formato.
Tablas HEAP: Son tablas temporales que quedan almacenadas en memoria, cuando se crean y que, por tanto, dejan de existir cuando se cierra la sesión. Son muy eficientes cuando se desea acceder de forma muy rápida a los datos que contienen.

Opciones o propiedades de tabla III:

AUTO_INCREMENT = valor Indicaría el primer valor que recibiría la columna tipo autoincremento de la tabla.

COMMENT = 'string' Un comentario que se va a mostrar cuando se muestre la estructura de la tabla.

MAX_ROWS = num Indicaría el máximo número de filas que puede llegar a tener la tabla.

SELECT ... Indica que la tabla va a recibir inicialmente en sus columnas los datos resultado de la consulta SELECT indicada sobre otras tablas ya creadas y con datos.

DATA DIRECTORY = 'path absoluto para tabla de datos' Indica la carpeta donde se almacenará el archivo de contenido de la tabla.

CHARACTER SET character_set_name [COLLATE collation_name]

Indica el sistema de codificación de los datos de tipo carácter en la tabla y la colación o criterios de comparación y ordenación alfabética.

Actividad 3-1