Desarrollo de aplicaciones web / multiplataforma

M02A: Bases de datos

UF2b. Lenguaje DDL

Índice

ILERNA Online

1. Bases de datos

Creación de bases de datos.

Para definir una nueva base de datos usaremos el comando CREATE DATABASE - o create SCHEMA, en MySQL son equivalentes- con la siguiente sintaxis:

```
CREATE {DATABASE | SCHEMA} nombreBaseDatos
[CHARACTER SET juegoDeCaracteres]
[COLLATE       collateName]
```

```
CREATE DATABASE Ilerna
CHARACTER SET Latin1
COLLATE latin1_spanish_ci;

USE Ilerna;
```

Como vemos, para definir una base de datos debemos definir:

- Nombre de la base de datos
- Juego de caracteres → Este parámetro es opcional, en caso de no indicarlo, tomará el valor por defecto.
- Collate → Este parámetro es igualmente opcional, hace referencia al modo en que la base de datos debe tratar las comparaciones, cosas como distinción entre mayúsculas y minúsculas o el peso de cada elemento (Si a>A o a<A).

1. Bases de datos

Modificación de bases de datos.

Para modificar una base de datos previamente definida podemos hacer uso del comando ALTER DATABASE con la siguiente sintaxis:

```
ALTER{DATABASE | SCHEMA} nombreBaseDatos
[CHARACTER SET juegoDeCaracteres]
[COLLATE       collateName]
```

```
ALTER SCHEMA Ilerna
CHARACTER SET Latin1
COLLATE latin1_spanish_ci;
```

ILERNA Online

1. Bases de datos

Borrado de bases de datos.

Para poder eliminar una base de datos hacemos uso del comando DROP DATABASE con la siguiente sintaxis:

```
DROP{DATABASE | SCHEMA} nombreBaseDatos
```

```
DROP SCHEMA Ilerna;
```

2. Tablas

Creación de tablas.

Para la creación de tablas se usa el comando CREATE TABLE con la siguiente sintaxis:

```
CREATE TABLE nombreTabla(
columna1 tipo(N) [UNIQUE|NOT NULL],
columna2 tipo(N) [UNIQUE|NOT NULL],
columna3 tipo(N) [UNIQUE|NOT NULL],
columna4 tipo(N) [UNIQUE|NOT NULL],
[Primary Key(...)],
[Foreign Key(...) References …],
)
[Collate=...,
Engine=...];
```

Por tanto, para crear una base de datos debemos indicar:

- El nombre de la base de datos
- Las definiciones de sus columnas
- Las restricciones (Claves primarias y ajenas, etc)
- Las opciones de la tabla (Opcional)

ILERNA Online

2. Tablas

Creación de tablas.

Para definir una columna debemos indicar:

- Su nombre
- Su tipo
- Su longitud

```
Nombre TIPO_DE_DATO(LONGITUD)
```

En MySQL contamos con los siguientes tipos de datos
http://dev.mysql.com/doc/refman/5.7/en/data-types.html

```
CREATE TABLE profesores(
    NIF Varchar(10),
    Nombre Varchar(50),
    Apellidos Varchar(100),
    Fecha_Nacimiento datetime,
    Primary Key(NIF)
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB;
```

ILERNA Online

2. Tablas

Creación de tablas.

Para definir las restricciones de una tabla debemos indicar:

- Nombre
- Tipo
- Columnas a las que afecta
- En su caso, tabla de referencia (Para claves ajenas)

```
Constraint nombre [Primary Key(...)|Foreign Key()]
```

```
create table asignaturas(
   Codigo INT(3),
   Nombre Varchar(12),
   Descripcion Varchar(30),
   Profesor_Titular Varchar(10),

   Constraint PK_Asignatura Primary Key(Codigo),
   Constraint FK_Asignatura_Profesor
      Foreign Key(Profesor_Titular) references profesores(NIF)
      On delete Restrict on update cascade
```

ILERNA Online

2. Tablas

Modificación de tablas.

SQL permite modificar la estructura de una tabla anteriormente definida, para ello se emplea el comando ALTER TABLE con la siguiente sintaxis:

```
ALTER TABLE nombreTabla
MODIFICACIÓN.
```

MODIFICACIÓN puede ser:

- **ADD**:
  - columnaX tipo(N) [FIRST|AFTER]
  - (columnaX tipo(N) [FIRST|AFTER], columnaY tipo(N) [FIRST|AFTER],...)
  - Constraint [Primary KEY| Foreign Key | UNIQUE]

  Add permite añadir diferentes elementos en la tabla, en el caso de las columnas, es posible definir la posición en la que queremos que se añada, mediante FIRST (En primer lugar) o AFTER (Después de otra columna ya existente).
- **CHANGE** nombre_columna Nueva Definición Columna

  Permite cambiar el nombre de una columna
- **RENAME COLUMN** nombreColumnaAnterior TO nombreColumnaNuevo.

  Permite cambiar el nombre de una columna
- **MODIFY** definicionColumna

  Modify se emplea para modificar la definición de una columna y/o añadir restricciones a la misma.
- **DROP COLUMN** NombreColumna
- **DROP Primary Key**
- **DROP Foreign Key** NombreFK

  Drop elimina elementos de nuestra tabla.

2. Tablas

Borrado de tablas.

Para borrar una tabla en SQL empleamos el comando DROP TABLE con la siguiente sintaxis:

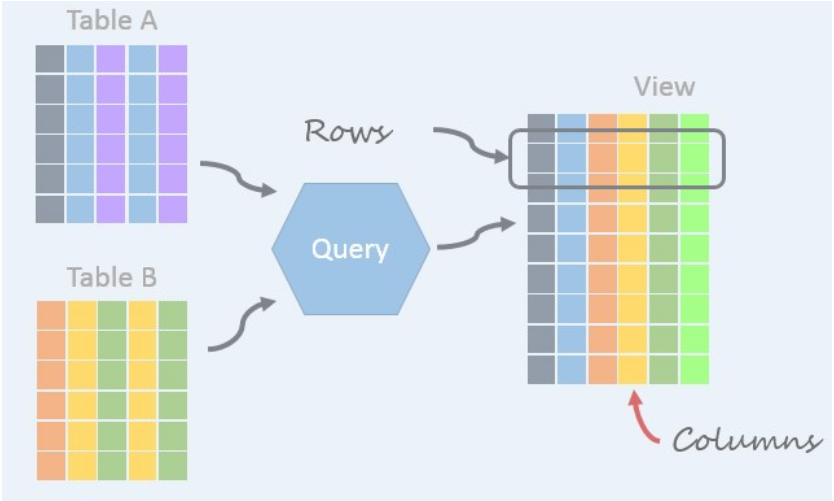
```
DROP TABLE NombreDeLaTabla
```

```
DROP TABLE asignaturas;
```

3. Vistas

Introducción.

- Tabla virtual que muestra datos definidos a través de una consulta.
- No almacena información per se.
- Su principal cometido es el de proporcionar una forma diferente de mostrar la información.



ILERNA Online

3. Vistas

Creación de Vistas.

Para la creación de las vistas, se usa el comando CREATE VIEW con la siguiente sintaxis.

```
CREATE VIEW nombreVista(columna1, columna2, columna3, columna4) AS SENTENCIA_SELECT.
```

```
CREATE VIEW ProfesoresAsignaturas (Nombre_Profesor, Apellido_Profesor, Nombre_Asignatura, Descripción) AS
SELECT profesores.nombre, profesores.apellidos, asignaturas.Nombre, asignaturas.Descripcion
from profesores, asignaturas
where profesores.NIF=asignaturas.Profesor_Titular;
```

ILERNA Online

3. Vistas

Modificación de Vistas.

Para la modificación de las vistas, se usa el comando REPLACE VIEW con la siguiente sintaxis.

```
CREATE OR REPLACE VIEW nombreVista(columna1, columna2, columna3, columna4) AS
SENTENCIA_SELECT.
```

```
CREATE OR REPLACE VIEW ProfesoresAsignaturas (Nombre_Profesor, Nombre_Asignatura) AS
SELECT profesores.nombre, asignaturas.Nombre
from profesores, asignaturas
where profesores.NIF=asignaturas.Profesor_Titular;
```

3. Vistas

Eliminación de Vistas.

Para eliminar una vista hacemos uso del comando DROP VIEW con la siguiente sintaxis:

```
DROP VIEW NombreDeLaVista
```

```
DROP VIEW ProfesoresAsignaturas;
```

4. Índices

Introducción

En SQL, es posible crear índices sobre las tablas para agilizar la consulta de registros.

Los índices cumplen las siguientes funciones:

- Encontrar las filas que cumplen la condición WHERE de la consulta cuyas columnas están indexadas.
- Para recuperar las filas de otras tablas cuando se emplean operaciones de tipo JOIN. Para ello, es importante que los índices sean del mismo tipo y tamaño ya que aumentará la eficiencia de la búsqueda.
- Disminuir el tiempo de ejecución de las consultas con ordenación (ORDER BY) o agrupamiento (GROUP BY) si todas las columnas presentes en los criterios forman parte de un índice.
- Si la consulta emplea una condición simple cuya columna de la condición está indexada, las filas serán recuperadas directamente a partir del índice, sin pasar a consultar la tabla.

4. Índices

Creación de índices

Para la creación de índices debemos usar la sentencia CREATE INDEX con la siguiente sintaxis:

```
CREATE INDEX nombreIndice ON tabla(Campo)
```

Esta es una sintaxis básica, en el caso de MySQL existen multitud de opciones, para saber más: http://dev.mysql.com/doc/refman/5.7/en/create-index.html

```
CREATE INDEX ApellidoProfes ON Profesores(Apellidos);
```

ILERNA Online

4. Índices

Borrado de índices

Para borrar un índice debemos usar el comando DROP INDEX con la siguiente sintaxis:

```
DROP INDEX nombreIndice ON tabla
```

```
DROP INDEX ApellidoProfes ON Profesores
```

5. Transacciones

Introducción

Una transacción puede definirse como un conjunto de sentencias SQL que se tratan como una sola instrucción, es decir, una operación atómica. Esto quiere decir que sólo se confirmará (Se hará COMMIT en la base de datos) en caso de que todas las operaciones que forman la transacción se hayan completado correctamente. En caso contrario, los cambios se desharán (Se hará ROLLBACK en la base de datos). Es decir, o se ejecutan correctamente todas las sentencias, o no se ejecuta ninguna. Esto resulta especialmente útil para mantener la integridad referencial de los datos y evitar inconsistencias.

```
START TRANSACTION;
   Insert into profesores values ('000000A', 'Jose', 'Martinez Martinez', '1987-12-11');
   Insert into asignaturas values ('0', 'BBDD', 'Base de datos', '000000A');
COMMIT WORK;
```

ILERNA Online

5. Transacciones

Acceso concurrente a los datos

Puede suceder que dos transacciones quieran acceder al mismo dato en el mismo instante de tiempo. Esto puede ocasionar una serie de problemas que vienen descritos en el estándar SQL.

- **Dirty Read (Lectura Sucia)** → Se produce cuando una transacción lee datos introducidos por otra antes de hacer COMMIT.

- **Non Repeatable Read (Lectura no repetible)** → Se produce cuando una transacción realiza dos lecturas de los mismos datos, obteniendo resultados diferentes. Esto indica que los datos han sido modificados en el lapso de tiempo entre ambas.

- **Phantom Read (Lectura Fantasma)** → Una transacción lee unos datos que no existían al comienzo de la misma.

ILERNA Online

5. Transacciones

Acceso concurrente a los datos

SQL permite bloquear conjuntos de datos para evitar que sucedan estos problemas. Según el nivel de concurrencia deseado se pueden establecer cuatro niveles de aislamiento.

- **Read Uncommitted (Lectura no acometida)** → Este nivel permite que se produzcan cualquiera de las problemáticas anteriormente descritas.

- **Read Committed (Lectura acometida)** → Una transacción no puede leer datos introducidos por otra hasta que esta última no realice COMMIT.

- **Repeatable Read (Lectura repetible)** → Una vez que una transacción realiza una consulta sobre un registro, este no puede ser modificado por otra.

- **Serializable** → En este nivel, las transacciones se ejecutan de forma totalmente aislada del resto. Es decir, se bloquean las transacciones de modo que la ejecución concurrente de las mismas sobre un conjunto de datos se deshabilita, pasando a ejecutarse secuencialmente.

ILERNA Online

```
SET TRANSACTION ISOLATION LEVEL {READ UNCOMMITTED|
                                 READ COMMITTED |
                                 REPEATABLE READ |
                                 SERIALIZABLE}
```

6. Seguridad de la información

Introducción

Cuando hablamos de la seguridad en el acceso a la información debemos diferenciar entre:

- **Seguridad integrada en el sistema operativo** → La proporcionada por un sistema de dominio o un servicio de directorio (LDAP).

- **Seguridad nativa del SGBD** → En este caso es el propio sistema gestor el que se gestionar el acceso a los elementos de la base de datos mediante el uso de usuarios y privilegios.

ILERNA Online

6. Seguridad de la información

Usuarios

- Crear usuarios

  ```
  CREATE USER nombre_usuario IDENTIFIED BY password
  ```

- Modificar usuarios
  - Nombre de usuario

    ```
    RENAME USER nombre_usuario@dominio TO nombre_usuario@dominio
    ```

  - Contraseña

    ```
    SET PASSWORD for nombre_usuario@dominio =PASSWORD('nuevaPassword')
    ```

- Eliminar usuario

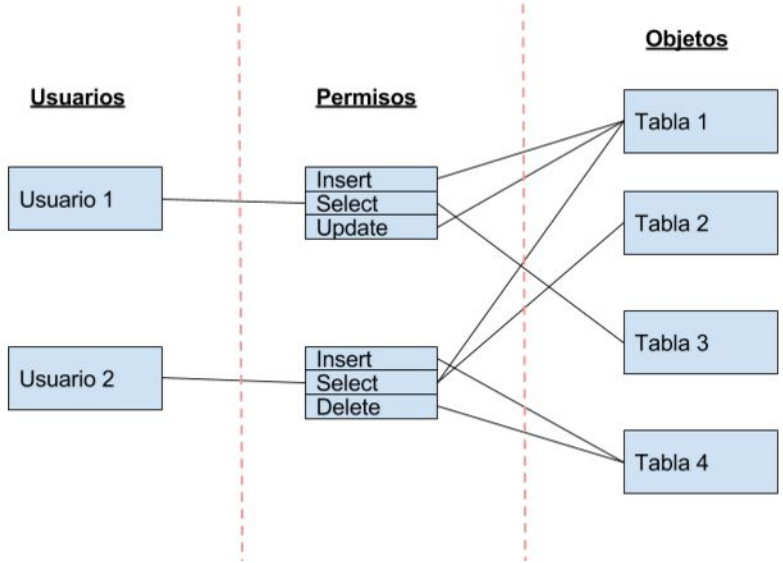
  ```
  DROP USER nombre_usuario
  ```

ILERNA Online

6. Seguridad de la información

Privilegios

- Privilegio ➔ permiso que tiene (o no) un determinado usuario para realizar acciones sobre un objeto del modelo concreto.
- Un privilegio está asociado a una acción y un objeto concreto de la base de datos.



6. Seguridad de la información

Privilegios

- Asignar permisos

  GRANT tipo_privilegio [(columnas)] ON nombre_tabla TO usuario [IDENTIFIED BY ‘password’]

- Del mismo modo que los otorgamos, es posible retirar los permisos a un usuario mediante la sentencia REVOKE con la siguiente sintaxis:

  REVOKE tipo_privilegio [(columnas)] ON nombre_tabla FROM usuario [IDENTIFIED BY ‘password’]

```
CREATE USER usuario1 IDENTIFIED BY 'contrasena1';

GRANT insert ON ilerna.asignaturas  TO usuario1;
GRANT select ON ilerna.asignaturas  TO usuario1;
GRANT update ON ilerna.asignaturas  TO usuario1;
GRANT delete ON ilerna.asignaturas  TO usuario1;
```

ILERNA Online

Esta es la estructura básica del GRANT, para ver más detalles podemos acceder al manual de MySQL,
http://dev.mysql.com/doc/refman/5.7/en/grant.html