

EXPORTANDO A BASE DE DATOS DESDE XML CON PHP

Antonio Ferrer Lopez.

INDICE

1. Introducción.
2. Análisis de datos.
3. Diseño de base de datos.
4. Implementación de servidor.
5. Creación de la base de datos en el gestor de bases de datos.
6. Creación del software mediante php.
7. Comprobación de resultados (CONSULTAS A LA BBDD CREADA).

1. Introducción.

Se pretende crear una aplicación del lado del servidor mediante lenguaje de programación PHP que extraiga automáticamente los datos de un archivo xml y los inserte correctamente en una base de datos.

Dicho programa debe realizar las siguientes tareas:

- Creación de la estructura de la base de datos de acuerdo al diseño previo de esta.
- Creación de las relaciones entre las tablas de la base de datos.
- Lectura y serialización del archivo xml proporcionado.
- Insertión de los datos en la base de datos.

Puesto que se trata de un backend, posteriormente realizaremos una serie de consultas a la base de datos para demostrar el correcto funcionamiento de la misma.

Posteriormente, sería posible diseñar un front-end para el consumo de dicha base de datos, la cual nos permitiría ofrecer dichos datos.

2. Análisis de datos.

Se nos proporciona un archivo de datos en formato XML compuesto por un gran número de registros, a continuación, mostramos la estructura de dicho archivo.

```
Casos.xml  <?xml version="1.0" encoding="UTF-8"?> Untitled-1
1  <?xml version="1.0" encoding="UTF-8"?>                                //Declaración de archivo xml , su codificación y versión
2  <records>                                                                //INICIO de los registros (root) , contiene todos los registros individuales
3    <record>                                                                //Comienzo de registro individual , se repelirá por cada registro
4      <dateRep>28/03/2020</dateRep>                                        //Fecha en la que se toma el registro
5      <day>28</day>                                                        //Dia del registro
6      <month>3</month>                                                    //Mes de l registro
7      <year>2020</year>                                                    //Año del registro
8      <cases>16</cases>                                                    // Número de casos confirmados con Coronavirus en dicha fecha
9      <deaths>1</deaths>                                                  // Número de fallecidos en dicha fecha por Coronavirus
10     <countriesAndTerritories>Afghanistan</countriesAndTerritories>      // Territorio que comprende los datos de dicho registros
11     <geoId>AF</geoId>                                                    // Referencia del territorio (identificador corto)
12     <countryterritoryCode>AFG</countryterritoryCode>                    //Codigo de territorio referencia del pais
13     <popData2018>37172386</popData2018>                                // Población total de dicho pais
14   </record>                                                                // FIN del registro
15   <record>                                                                // A PARTIR DE ESTE PUNTO LOS REGISTROS SE REPITEN CON diferente contenido.
16     <dateRep>27/03/2020</dateRep>
17     <day>27</day>
18     <month>3</month>
19     <year>2020</year>
20     <cases>0</cases>
21     <deaths>0</deaths>
22     <countriesAndTerritories>Afghanistan</countriesAndTerritories>
23     <geoId>AF</geoId>
24     <countryterritoryCode>AFG</countryterritoryCode>
25     <popData2018>37172386</popData2018>
26   </record>
27
```

Estos datos, expresan los casos positivos, fallecimientos y la población total en varios países del mundo a la largo de unas fechas determinadas.

Dichos datos nos permitirían obtener estadísticas sobre la evolución del virus a nivel global.

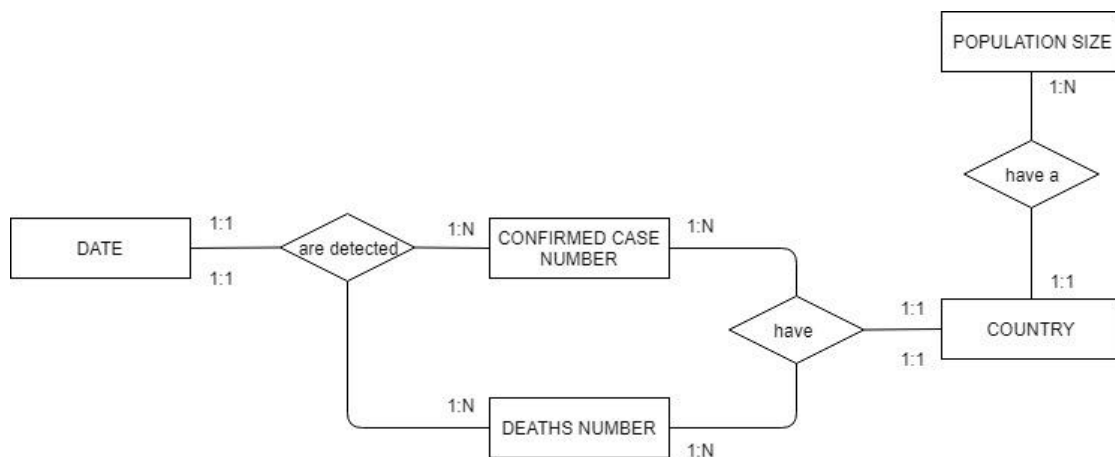
3. Diseño de la base de datos.

Una vez comprendidos los datos de debemos almacenar, procederemos al diseño de la base de datos para su implantación.

Para lo cual, partiremos del diseño del modelo entidad-relación, posteriormente crearemos el modelo relacional y aplicaremos las FN hasta la 3FN.

De esta forma obtendremos el esquema que nos permitirá crear nuestras consultas dentro de la aplicación que generarán las tablas y las relaciones entre ellas.

MODELO ENTIDAD RELACIÓN.

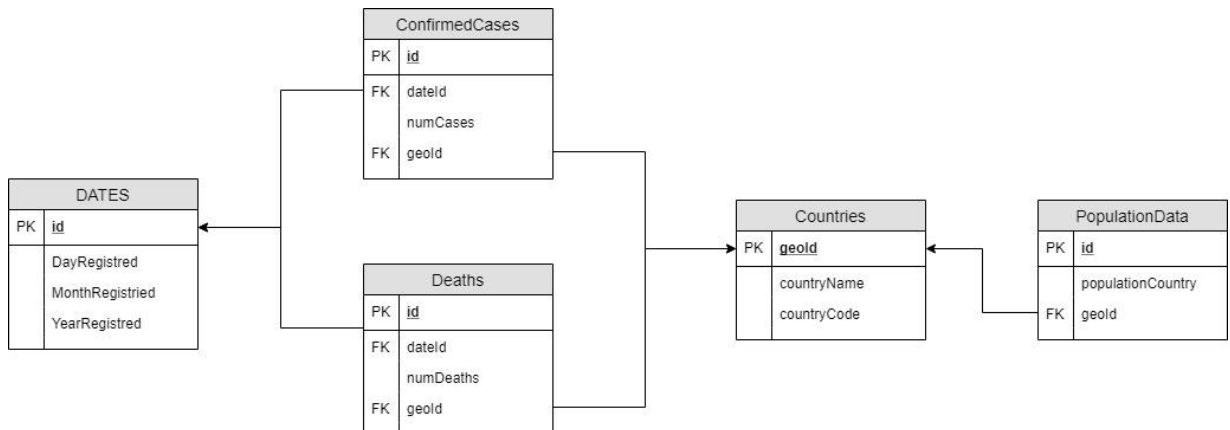


Pasemos a revisar las relaciones que se establecen en el modelo:

- En una fecha determinada (DATE) habrá una cantidad de casos confirmados y de fallecimientos (solo un número total por cada fecha).
- Un número de fallecidos o confirmados puede repetirse en varias fechas (si no hay nuevos contagios o fallecimientos).
- Por cada país hay un número total de fallecidos y de casos confirmados, dicho número de casos puede ser el mismo en diferentes países.
- Un país tiene una densidad de población concreta, puede haber países que posean la misma densidad de población.

Una vez establecidas las entidades que conforman el modelo, así como sus relaciones, pasamos dicho modelo a un nivel más de abstracción obteniendo el modelo relacional.

MODELO RELACIONAL



A partir de este modelo relacional podemos analizar si respeta la normalización buscada.

1FN

Podemos asegurar que la primera forma normal es respetada puesto que no hay atributos NO atómicos en las tablas, por lo que podemos pasar a analizar si cumple la segunda forma normal

2FN

Puesto que no hay claves primarias compuestas en ninguna tabla, este diseño cumple con la segunda forma normal.

3FN

Este diseño cumple la 3FN puesto que todos los atributos de las tablas que no son clave primaria no dependen de forma transitiva de la clave primaria.

A partir de este punto, y tras el análisis previo, podemos asegurar que estamos ante un diseño correcto por lo que podemos pasar a realizar los preparativos de nuestro sistema para correr aplicaciones diseñadas en php para, posteriormente desarrollar el programa propiamente dicho.

4. Implementación mediante php.

Para la preparación del entorno de trabajo precisamos un servidor web, en este caso emplearemos Apache, php (Emplearemos la versión 7.4) instalado en el sistema y un motor de bases de datos, en este caso emplearemos MariaDb.

El sistema operativo sobre el que trabajaremos es Linux Mint en su versión 19. Dispongo de un motor de bases de datos en servidor local (192.168.1.52:3306) por lo que haremos uso de ella.

Comenzaremos por instalar todo lo necesario en nuestro sistema.

Primero actualizaremos nuestros repositorios y paquetes mediante:

```
apt update
```

```
apt upgrade
```

Posteriormente instalaremos el servidor web Apache

```
apt install apache2
```

Posteriormente instalamos los paquetes de php y alguna de sus extensiones que pueden sernos útiles.

```
apt install php-7.4 php7.4 libapache2-mod-php7.4 php7.4-cli php7.4-mysql
```

Posteriormente reiniciamos nuestro servidor web mediante el comando

```
service apache2 restart
```

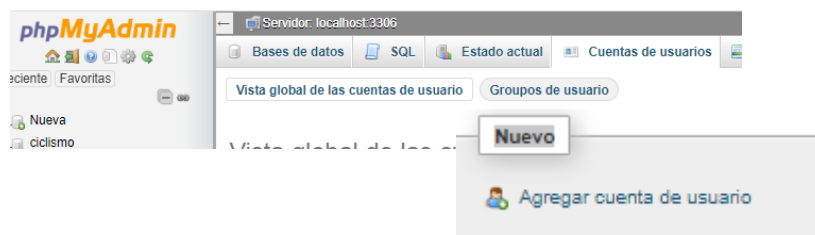
Y ya dispondremos de nuestro sistema listo para desarrollar nuestro software en PHP. Cabe indicar que nuestra web debe estar en el directorio `var/www/html` para poder ser visto en el navegador.

5. Creación de la base de datos en el gestor de bases de datos.

A continuación, debemos crear la base de datos que va a contener todos los datos, así como el usuario con el que configuraremos nuestra aplicación, el cual debe tener permisos para poder manipular dicha base de datos en concreto.

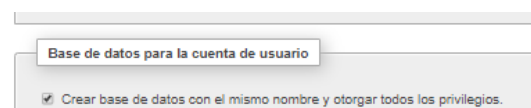
Se ha decidido nombrar a la base de datos `covidxml` y al usuario de la misma forma. La contraseña será autogenerada de forma aleatoria en el momento de la creación del usuario y la debemos guardar puesto que la requeriremos para hacer la conexión a la misma.

Una de las formas más rápidas para crear un usuario con su base de datos asociada es mediante phpMyadmin.



Una vez logeados, seleccionamos la opción Cuentas de usuario y añadimos un nuevo usuario.

En el nuevo panel rellenamos el formulario de acuerdo a las preferencias que se requieran, y en el momento de insertar contraseña, dejamos que el sistema genere una aleatoria (debemos guardarla para su uso posterior). Nos aseguraremos de indicar que cree una base de datos para dicho usuario.

This image shows a close-up of a form within the phpMyAdmin interface. The form is titled 'Base de datos para la cuenta de usuario'. It contains a single checkbox with the label 'Crear base de datos con el mismo nombre y otorgar todos los privilegios.' which is currently checked.

Con esto último ya dispondremos de una base de datos con la que trabajar.

6. Creación del software mediante php.

```
index.php <
1 <?php
2 generateTablesIfNotExists();
3 $dbIsPrepared = createTableReferences();
4 if($dbIsPrepared){
5     echo "Insertando datos en la base de datos-->>>";
6     insertValuesOnDataBase();
7     echo "DATOS INSERTADOS CORRECTAMENTE";
8 }else{
9     echo "LA BASE DE DATOS NO ESTA LISTA.";
10 }
11 function generateTablesIfNotExists(){...}
12
13 function createTableReferences(){...}
14
15 function insertValuesOnDataBase(){
16     $conn = new mysqli( host: "192.168.1.52:3306", username: "covidxml", passwd: "AindyCVCUCGmvee0", dbname: "covidxml");
17     $xml = simplexml_load_file( filename: "Casos.xml") or die("Error: No se puede cargar el fichero xml");
18
19     foreach ($xml->children() as $fila) {
20         $day = $fila->day;
21         $month = $fila->month;
22         $year = $fila->year;
23         $confirmedCases = $fila->cases;
24         $deaths = $fila->deaths;
25         $countriesName = $fila->countriesAndTerritories;
26         $geoId = $fila->geoId;
27         $countryCode = $fila->countryterritoryCode;
28         $population = $fila->popData2018;
29
30         $dateId = $year . $month . $day;
31
32         $stmtDates = $conn->prepare( query: "INSERT INTO dates(id,dayRegistered,monthRegistered,yearRegistered) VALUES(?,?,?)");
33         if ($stmtDates) {
34             $stmtDates->bind_param( types: 'sii', &var1: $dateId, &var2: $day, &var3: $month, &var4: $year);
35             $stmtDates->execute();
36         } else {
37             echo mysqli_error($stmtDates);
38         }
39
40         $stmtCountries = $conn->prepare( query: "INSERT INTO countries(geoId,countryName,countryCode) VALUES(?,?,?)");
41         if ($stmtCountries) {
42             $stmtCountries->bind_param( types: 'sss', &var1: $geoId, &var2: $countriesName, &var3: $countryCode);
43             $stmtCountries->execute();
44         } else {
45             echo mysqli_error($stmtCountries);
46         }
47
48         $stmtPopulation = $conn->prepare( query: "INSERT INTO populationData(populationCountry,geoId) VALUES(?,?)");
49         if ($stmtPopulation) {
50             $stmtPopulation->bind_param( types: 'is', &var1: $population, &var2: $geoId);
51             $stmtPopulation->execute();
52         } else {
53             echo mysqli_error($stmtPopulation);
54         }
55
56         $stmtConfirmedCases = $conn->prepare( query: "INSERT INTO confirmedCases(dateId,numCases,geoId) VALUES(?,?,?)");
57         if ($stmtConfirmedCases) {
58             $stmtConfirmedCases->bind_param( types: 'sis', &var1: $dateId, &var2: $confirmedCases, &var3: $geoId);
59             $stmtConfirmedCases->execute();
60         } else {
61             echo mysqli_error($stmtConfirmedCases);
62         }
63
64         $stmtDeaths = $conn->prepare( query: "INSERT INTO deaths(dateId,numDeaths,geoId) VALUES(?,?,?)");
65         if ($stmtDeaths) {
66             $stmtDeaths->bind_param( types: 'sis', &var1: $dateId, &var2: $deaths, &var3: $geoId);
67             $stmtDeaths->execute();
68         } else {
69             echo mysqli_error($stmtDeaths);
70         }
71     }
72
73     $conn->close();
74 }
75
76 }
77
78 }
```

Pasemos a crear nuestra aplicación en php, abajo se muestra le código de la aplicación que se ha realizado y posteriormente a este, se detalla el flujo del programa y las funciones que lo componen.

Comencemos a analizar el código propuesto parte por parte.

```
<?php
generateTablesIfNotExists();
$dbIsPrepared = createTableReferences();
if($dbIsPrepared){
    echo "Insertando datos en la base de datos-->>>";
    insertValuesOnDataBase();
    echo "DATOS INSERTADOS CORRECTAMENTE";
}else{
    echo "LA BASE DE DATOS NO ESTA LISTA.";
}
```

Al comienzo de la ejecución del código, lanzamos la función *generateTablesIfNotExists()* que nos generará las tablas de la base de datos.

Al terminar de ejecutarse esta función, ejecutaremos la función que se encarga de crear las referencias entre las tablas

denominada *createTableReferences()* , esta función devuelve TRUE si se ha conseguido ejecutar correctamente, con lo que conseguimos saber si se ha creado correctamente la estructura de la base de datos correctamente, en caso contrario , nos muestra un mensaje de error . Si la respuesta fuera correcta continua con el flujo de ejecución ejecutando la función *insertValuesOnDataBase()* que se encarga de serializar los datos xml e insertarlos en la base de datos.

Por motivos de simplicidad se ha optado por no separar las dos tareas que realiza esta última función pese a romper el principio de responsabilidad única ([Principios SOLID](#)).

Una vez analizado el flujo de ejecución pasemos a revisar por encima las funciones que componen el programa.

generateTablesIfNotExists()

En 12 crea una conexión con la base de datos con la que realizaremos las consultas sql necesarias.

Posteriormente almacenamos en variables las consultas de creación de cada una de las tablas (String).

Emplearemos el helper *mysqli_multi_query* que nos permite realizar varias consultas seguidas, por lo que concatenaremos todas las consultas para formar un String multiconsulta (47) y posteriormente realizaremos la consulta a la base de datos. Puesto que las consultas a la base

```
11 function generateTablesIfNotExists(){
12     $conn = new mysqli("192.168.1.52:3306", "root", "covidmi1", "root", "AindyCVCUCGavee0", "root", "covidmi1");
13     $tableDates = "CREATE TABLE IF NOT EXISTS dates(
14         id DATE,
15         dayRegistered INT ,
16         monthRegistered INT ,
17         yearRegistered INT ,
18         PRIMARY KEY (id)
19     );";
20     $tableConfirmedCases = "CREATE TABLE IF NOT EXISTS confirmedCases(
21         id INT AUTO_INCREMENT ,
22         dateId DATE NOT NULL ,
23         numCases INT ,
24         geoid VARCHAR(3) NOT NULL ,
25         PRIMARY KEY (id)
26     );";
27     $tableDeaths = "CREATE TABLE IF NOT EXISTS deaths(
28         id INT AUTO_INCREMENT ,
29         dateId DATE NOT NULL ,
30         numDeaths INT ,
31         geoid VARCHAR(3) NOT NULL ,
32         PRIMARY KEY (id)
33     );";
34     $tableCountries = "CREATE TABLE IF NOT EXISTS countries(
35         geoid VARCHAR(3) NOT NULL,
36         countryName VARCHAR(20) ,
37         countryCode VARCHAR(4),
38         PRIMARY KEY (geoid)
39     );";
40     $tablePopulationData = "CREATE TABLE IF NOT EXISTS populationData(
41         id INT AUTO_INCREMENT ,
42         populationCountry INT ,
43         geoid VARCHAR(3) NOT NULL,
44         PRIMARY KEY (id)
45     );";
46
47     $allTables = $tableDates . $tableConfirmedCases . $tableDeaths . $tableCountries . $tablePopulationData;
48     try{
49         mysqli_multi_query($conn , $allTables);
50     }catch(mysqli_sql_exception $e){
51         echo "Error al crear las tablas : " . $e;
52     }finally {
53         mysqli_close($conn);
54     }
55 }
```

de datos pueden generar un error, debemos capturarlo mediante un try-catch y actuar en consecuencia. Finalmente cerramos la conexión a la BBDD para liberar recursos.

CreateTableReferences()

Después de realizar una conexión a la base de datos en 58, almacenamos en la variable `tableReferences` el String de la consulta que nos creará las referencias entre las tablas de la base de datos, para , posteriormente realizar la consulta de la misma manera que podemos observar en la función `generateTablesIfNotExists()` , en una refactorización de código posterior sería interesante extraer esta parte del código en una función privada para poder utilizarla en las dos funciones sin repetir código.

```
57 function createTableReferences(){
58     $conn = new mysqli( "host: '192.168.1.52:3306'", "username: 'covidxml'", "passwd: 'AindyCVCUCGveee0'", "dbname: 'covidxml'");
59     $tableReferences = "ALTER TABLE confirmedCases ADD CONSTRAINT confirmedCases_Date_PK FOREIGN KEY (dateId) REFERENCES dates(id);
60
61     ALTER TABLE confirmedCases ADD CONSTRAINT confirmedCases_Country_PK FOREIGN KEY (geoId) REFERENCES countries(geoId);
62     ALTER TABLE deaths ADD CONSTRAINT deaths_Date_PK FOREIGN KEY (dateId) REFERENCES dates(id);
63     ALTER TABLE deaths ADD CONSTRAINT deaths_Country_PK FOREIGN KEY (geoId) REFERENCES countries(geoId);
64     ALTER TABLE populationData ADD CONSTRAINT population_Country_PK FOREIGN KEY (geoId) REFERENCES countries(geoId);
65 ";
66
67     try {
68         mysqli_multi_query($conn, $tableReferences);
69     } catch (mysqli_sql_exception $e){
70         echo "falló al crear las referencias : " . $e;
71     } finally {
72         return mysqli_close($conn);
73     }
74 }
```

insertValuesOnDatabase()

Después de crear nuestra conexión a la base de datos, Cargamos nuestro archivo xml mediante el método `simple_load_file`.

Mediante un `foreach` recorreremos todos los registros del archivo, accediendo a cada uno de los hijos (Estos son cada uno de los registros que queremos procesar)

Por cada uno de los registros, realizaremos tres tareas:

Primero extraemos y guardamos en una variable cada uno de los datos que nos interesan del registro (de 79 a 87).

```
74 function insertValuesOnDatabase(){
75     $conn = new mysqli( "host: '192.168.1.52:3306'", "username: 'covidxml'", "passwd: 'AindyCVCUCGveee0'", "dbname: 'covidxml'");
76     $xml = simplexml_load_file( "localhost: 'Cargos.xml'" ) or die("Error: No se puede cargar el fichero xml");
77
78     foreach ($xml->children() as $fila) {
79         $day = $fila->day;
80         $month = $fila->month;
81         $year = $fila->year;
82         $confirmedCases = $fila->cases;
83         $deaths = $fila->deaths;
84         $countriesName = $fila->countriesAndTerritories;
85         $geoId = $fila->geoId;
86         $countryCode = $fila->countryTerritoryCode;
87         $population = $fila->popData2018;
88
89         $dateFormatted = $year."-".$month."-".$day;
90         if($month < 10 ){
91             $month = str_pad($month, 2, '0', STR_PAD_LEFT);
92         }
93         $stmtDates = $conn->prepare( "INSERT INTO dates(id,dateRegistered,monthRegistered,yearRegistered) VALUES(?,?,?,?)");
94         if ($stmtDates) {
95             $stmtDates->bind_param( "ssss", $geoId, $dateFormatted, $month, $year);
96             $stmtDates->execute();
97         } else {
98             echo mysqli_error($stmtDates);
99         }
100
101         $stmtCountries = $conn->prepare( "INSERT INTO countries(geoId,countryName,countryCode) VALUES(?,?,?)");
102         if ($stmtCountries) {
103             $stmtCountries->bind_param( "sss", $geoId, $countriesName, $countryCode);
104             $stmtCountries->execute();
105         } else {
106             echo mysqli_error($stmtCountries);
107         }
108
109         $stmtPopulation = $conn->prepare( "INSERT INTO populationData(populationCountry,geoId) VALUES(?,?)");
110         if ($stmtPopulation) {
111             $stmtPopulation->bind_param( "si", $population, $geoId);
112             $stmtPopulation->execute();
113         } else {
114             echo mysqli_error($stmtPopulation);
115         }
116     }
117 }
```

Segundo creamos un String con la fecha para generar el id de la fecha (ya que debe almacenarse en formato date, se crea una variable que contendrá la cadena formateada de la siguiente forma =

YYYY – MM – DD

De esta forma quedará almacenado en formato fecha.)

```
114
115 $stmtConfirmedCases = $conn->prepare( mysqli_stmt_prepare($conn, "INSERT INTO confirmedCases(dateId,numCases,geoId) VALUES(?,?.?)");
116
117 if ($stmtConfirmedCases) {
118     $stmtConfirmedCases->bind_param( 'sis', $dateFormatted, $numConfirmedCases, $geoId);
119     $stmtConfirmedCases->execute();
120 } else {
121     echo mysqli_error($stmtConfirmedCases);
122 }
123
124 $stmtDeaths = $conn->prepare( mysqli_stmt_prepare($conn, "INSERT INTO deaths(dateId,numDeaths,geoId) VALUES(?,?.?)");
125
126 if ($stmtDeaths) {
127     $stmtDeaths->bind_param( 'sis', $dateFormatted, $numDeaths, $geoId);
128     $stmtDeaths->execute();
129 } else {
130     echo mysqli_error($stmtDeaths);
131 }
132
133 $conn->close();
134 }
```

El tercer paso vamos a crear las consultas SQL, pero, esta vez, ya que vamos a introducir datos, y por motivos de seguridad, realizaremos la creación de la consulta mediante un preparedStatement, con el que tenemos mayor control sobre los datos que vamos a introducir. Para ello:

Primero creamos la consulta preparada y en la posición en la que deben ir los valores a introducir ponemos “?”. Ya que puede haber fallos al generar la consulta preparada, mediante un condicional permitiremos que continúe con el Binding y la ejecución siempre que este correcto, en caso contrario imprimiremos el error para poder tracearlo.

Segundo, “bindeamos” los parámetros, es decir le indicamos en cada uno de los “?” que tipo de valor va a contener y la variable que contiene dicho valor.

Tercer paso, ejecutaremos la consulta preparada mediante ->execute().

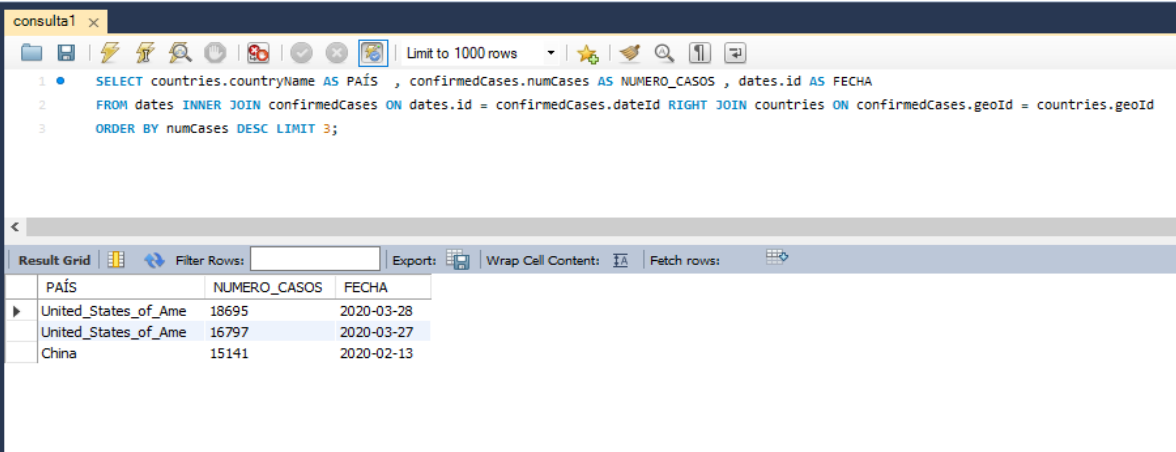
Estos pasos se repiten con cada una de las tablas y los datos, insertando los datos que correspondan en cada uno de las tablas, a su vez, este proceso se repite por cada uno de los registros del archivo xml hasta llegar al final, llenando todos los datos que nos interesa tener almacenados.

Una vez tenemos todos los datos cargados cerramos la conexión (132).

7. Comprobación de resultados (CONSULTAS A LA BBDD CREADA).

A modo de muestra, se detallan una serie de 4 consultas a la base de datos y su resultado, así mismo se adjuntas las 4 en archivos .sql para su revisión.

- a. Indica per ordre els 3 primers països amb més casos confirmats i la data del mateix



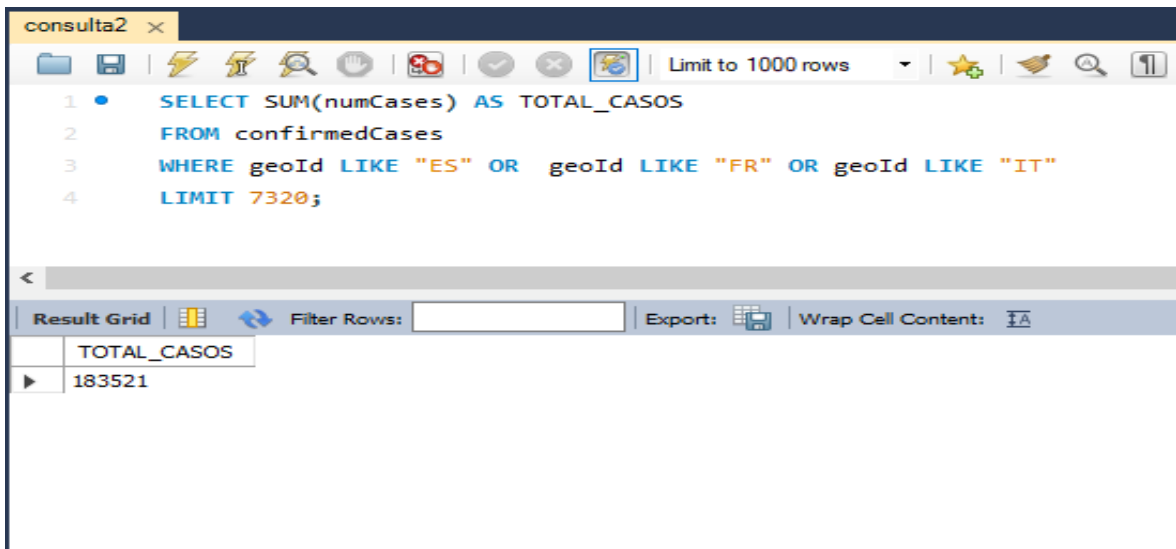
The screenshot shows a database query interface with a tab labeled 'consulta1'. The SQL query is as follows:

```
1 SELECT countries.countryName AS PAÍS , confirmedCases.numCases AS NUMERO_CASOS , dates.id AS FECHA
2 FROM dates INNER JOIN confirmedCases ON dates.id = confirmedCases.dateId RIGHT JOIN countries ON confirmedCases.geoId = countries.geoId
3 ORDER BY numCases DESC LIMIT 3;
```

The results are displayed in a table with the following data:

PAÍS	NUMERO_CASOS	FECHA
United_States_of_Ame	18695	2020-03-28
United_States_of_Ame	16797	2020-03-27
China	15141	2020-02-13

- b. Quants casos tindrien Espanya, França e Italia si sumem tots els dies que han tingut casos. Es a dir dia 1 100 casos dia 2 125 casos el total seria 225 casos.



The screenshot shows a database query interface with a tab labeled 'consulta2'. The SQL query is as follows:

```
1 SELECT SUM(numCases) AS TOTAL_CASOS
2 FROM confirmedCases
3 WHERE geoId LIKE "ES" OR geoId LIKE "FR" OR geoId LIKE "IT"
4 LIMIT 7320;
```

The results are displayed in a table with the following data:

TOTAL_CASOS
183521

c. Quin és el país que més dies ha obtés 0 casos.

The screenshot shows a SQL query editor window titled 'consulta3'. The query is as follows:

```
1 • SELECT COUNT(dates.id) AS Dias0Casos , countries.countryName
2 FROM dates INNER JOIN confirmedCases ON dates.id = confirmedCases.dateId
3 INNER JOIN countries ON countries.geoId = confirmedCases.geoId
4 WHERE confirmedCases.numCases = 0 GROUP BY countries.countryName ORDER BY Dias0Casos DESC LIMIT 1;
5
```

Below the query, the 'Result Grid' shows the following data:

Dias0Casos	countryName
73	Nepal

d. Indica els països que han obtés més de 500 morts en un dia.

The screenshot shows a SQL query editor window titled 'consulta4'. The query is as follows:

```
1 • SELECT countries.countryName
2 FROM countries INNER JOIN deaths ON countries.geoId = deaths.geoId
3 WHERE deaths.numDeaths > 500
4 GROUP BY countryName;
```

Below the query, the 'Result Grid' shows the following data:

countryName
Italy
Spain