De las fichas de archivo a las bases de datos

Jairo Antonio Melo Flórez
jairom@colmich.edu.mx

Contenido

Introducción

Al día de hoy, las fichas bibliográficas y documentales constituyen aún la herramienta fundamental dentro del arsenal metodológico del historiador. Los investigadores más apegados a sus costumbres todavía visitan el archivo armados con una ordenada pila de tarjetas de cartón delgado en la que apuntan los datos principales de cada documento, su origen, productor, personajes, fechas, lugares, palabras claves y algunos, los más sistemáticos, crean un código personal para recuperar el orden de las fichas en caso de sufrir una desastrosa caída que las desparrame por el suelo. Algunos menos ordenados recurren al tradicional cuaderno, al bloc de notas, incluso a hojas sueltas. Los más anarquistas simplemente toman hojas recicladas que reutilizan el lado blanco de las fotocopias desechadas.

Los archivos históricos son espacios bastante apegados a sus costumbres. Antes de permitir el traspaso de la puerta que da a la sala de consulta, se tienen que dejar los efectos personales innecesarios para la consulta en los casilleros de la entrada, donde un guardián se encarga de resguardar nuestras precarias posesiones. Los archivos más rigurosos apenas dejarán entrar con un lápiz y algunas hojas a media carta, incluso asignarán a un vigilante que estará pendiente del uso adecuado del documento. Los más laxos, simplemente te dejarán una silla y una mesa, en la que el historiador se sentirá a sus anchas para hacer su trabajo. La buena noticia es que cada vez son más los archivos que permiten ingresar con una computadora portátil para hacer la consulta; son más incluso que los que dejan tomar fotografías. Por lo que ya es común encontrar salas llenas de pesquisadores tomando notas directamente en sus ordenadores.

De esta manera, cada quien crea un estilo. Para muchos las fichas de cartón han pasado a ser tablas dentro de un procesador de texto. Muchos simplemente pasan directamente a una hoja de cálculo, pero no para hacer operaciones matemáticas sino para rellenarlas de texto y anotaciones varias. Los menos, usan un sistema de gestión de información cualitativa, llámese Atlas.ti, MAXQDA, o gestores de bases de datos como Access o Base. Incluso, hay dos o tres *nerds* que hacen sus propios sistemas de gestión de información y los exhiben presuntuosamente a los compañeros de sala; recibiendo por ello miradas descalificadoras por parte de los investigadores apegados a las fichas en cartón delgado.

Independientemente del soporte en el cual se realice la captura de la información documental, nos remitimos a un asunto propio de la técnica de investigación antes que de su metodología. La investigación puede estar precedida por la base de datos, nadie obliga al historiador a construir la suya. Si tuviéramos a la mano todas las fuentes que favorecen la resolución de nuestro problema de investigación no la haría menos relevante que aquella que tiene que partir de documentos desconocidos, ocultos en el baúl de un antiguo monasterio. Tal vez la segunda sería una investigación más curiosa, que traería un poco de misterio a la narración, pero nada asegura que esos documentos “perdidos” representen un aporte sustancial al conocimiento historiográfico.

En ese sentido, tampoco construir una base de datos hace al historiador mejor que sus pares, ni tampoco peor. Algunos investigadores consolidados consideran que recurrir a las fuentes digitalizadas y las bases de datos es un fomento a la pereza y la lectura afanosa del documento; un empobrecimiento del saber histórico. Pero sería lo mismo que juzgar que es un acto vicioso el que una de las primeras tareas al llegar a un archivo sea revisar los catálogos. Nadie inicia su investigación desde los anaqueles, toda investigación histórica empieza por el archivista y el bibliotecario.

Hoy día, el historiador va al archivo, al museo, al repositorio de audios o videos, y tiene que hacer una tarea manual previa a su análisis. Durante mucho tiempo las recopilaciones de fuentes primarias eran una publicación importante para la hoja de vida de un historiador. Aquel que tenía los recursos o el prestigio necesario, podía desplazarse junto con unos cuantos ayudantes a paleografiar y transcribir cientos de documentos de una colección documental, la cual luego podría imprimirse en varios volúmenes que son todavía la dicha de muchos de nosotros. También se acostumbra, incluso hoy, a dedicar unas cuantas páginas de las tesis para un anexo documental que transcribe unos cuantos folios relevantes o curiosos para el lector. Esto se debe a que la transcripción es una tarea de cierta manera gratificante, pero que toma mucho tiempo, esfuerzo físico y mental, para dar con un resultado que no siempre es publicable.

Pero hay otra tarea que consume inclusive más tiempo: la pesquisa. En los cafés que se encuentran en las inmediaciones de cualquier buen archivo, los investigadores se reúnen en una o varias sentadas a hablar de los expolios de la jornada. Los más exitosos comentan cómo avanzan en un interminable legajo, mientras los más miserables apenas podrán comentar si encontraron una escueta epístola con la que van llenando su botín de testimonios del pasado. Esta tarea es socialmente enriquecedora, pero no favorece a la investigación histórica; menos aún en estos tiempos de eficacias administrativas, cortas estancias y escasos recursos. No son pocos los historiadores que tienen que explorar frenéticamente un fondo durante los pocos días que pueden aprovechar durante un congreso académico o incluso de sus vacaciones.

Así que tenemos dos aspectos con los cuales relacionar las bases de datos y la labor de los historiadores: la recolección sistemática y la búsqueda automatizada de información histórica relevante para construir y falsear hipótesis sobre el pasado. En el estado actual de disponibilidad de las fuentes históricas, es factible que ambas tareas tengan que ser realizadas por el propio investigador; aunque el más afortunado pueda contar con la ayuda de un pequeño ejército de colaboradores. En un mundo ideal, los archivos y bibliotecas se harán cargo de esta tarea, como lo ha hecho, por ejemplo, la Biblioteca Nacional de España; teniendo el pesquisador como actividad principal la formulación de preguntas, la consulta en los sistemas de acceso a los datos y las demás tareas propias de la metodología de la historia. Aunque en el plano de la utopía, el esfuerzo colaborativo de cientos de historiadores podría poner en común una inmensa base de datos colectiva que sirviera como semilla para los investigadores nóveles y no tan novatos.

En este texto, exploraremos los conceptos y elementos básicos que debe conocer un historiador para construir y consultar las bases de datos. Como advertencia, no haremos

uso de una tecnología o aplicación específica, puesto que nuestro objetivo consiste en que se comprenda la lógica de una base de datos antes que el uso de una herramienta. Consideramos que será más fácil partir de estos fundamentos y luego llegar a sistema de gestión de datos como Atlas.ti, MAXQDA, FileMaker, Access, o el que tenga cada quien a su disposición. Asimismo, será la posibilidad de que desde ya se plantee la posibilidad de trascender la aplicación y se consideren las ventajas de usar sistemas libres para la manipulación de datos, en especial lenguajes de programación como Python o R. Finalmente, este texto no es un tutorial sino una guía o documentación para iniciar la construcción de una base de datos de investigación.

El dato y los datos[1]

Empecemos por la unidad mínima de una base de datos: el dato. Un dato en sí mismo no tiene mucho sentido. Por ejemplo, ‘1’ es un dato, también ‘p’ o la palabra ‘perro’. Un dato es un valor al que puede asignársele sentido de acuerdo al contexto. De este modo si digo “1 manzana” el numeral adquiere un sentido como cuantificador de un objeto, en este caso la fruta. Pero si digo “Juan es el número 1” el sentido cambia completamente, en este caso no estoy contando cuántos “juanes” existen, sino el lugar que ocupa Juan en un grupo. A mayor contexto el dato adquiere mayor sentido. El dato, “1”, puede ser parte de una oración completa: “Juan es el número 1 porque estudió mucho para el exámen”; en este caso, el número 1 adquiere otro sentido gracias a la acción “estudiar”. Pero si la oración fuese “Juan es el número 1 porque sobornó a su maestro” la valoración es completamente diferente.

El nombre “Juan”, el número “1” y el verbo “estudiar” o “sobornar” pueden ser tratados como datos. Es la oración la que les da un sentido del que carecen por separado. Ahora bien, si tenemos una cantidad de variables muy limitada es sencillo realizar el ejercicio mental y asignarles sentido. En el caso de ser un director de un curso, podría ver que Juan es el número 1 en matemáticas, pero su desempeño es muy mediocre en el resto de asignaturas, incluso las que tienen como base el pensamiento matemático. Podría entonces sospechar que hubo alguna acción subrepticia que hizo que el maestro lo pusiese en primer lugar. Pero a menos que tenga el dato específico, no podría saberlo con seguridad. En este sentido, la investigación se iniciaría por los datos, no por la observación directa.

El problema es que unos pocos datos no tienen mucha utilidad. Una tabla de nombres apenas serviría para ordenar alfabéticamente el listado, no importa cuál sea su tamaño. Si los nombres se acompañan de edades, puedo ordenarlos por edades o alfabéticamente, pues la relación entre nombres y edades no tiene mucho sentido. Podríamos ver cuántos hombres se llaman “Juan” en cierto rango de edad, pero no más allá de eso. Al ir asignando más criterios los datos se hacen más complejos y la información que pueden brindar es más representativa. Así, podría ver la tendencia en las calificaciones de los estudiantes de acuerdo con sus edades, condiciones socioeconómicas, étnicas, familiares, entre otras. Estas tendencias me permitirían construir hipótesis y métodos de investigación para saber si existen correlaciones entre condiciones económicas y calificaciones.

[1] En inglés, la palabra *datum* expresa el singular de *data*. En español no existe tal distinción.

En ese sentido, un dato es un elemento mínimo de información, que adquiere mayor sentido en tanto se correlaciona con otros datos. De esta manera, la relevancia de la información está dada en tanto sea más relevante para conocer sobre una realidad general o concreta. Si tenemos esto en mente, podremos entender que un dato histórico es cualquier unidad de información sobre el pasado: una fecha, un lugar, un nombre, un evento, un parentesco, entre otros. No debe confundirse dato con hecho histórico. “1527” es un dato histórico, pero “1527 es el año de nacimiento de Felipe II” es una afirmación que tiene que sustentarse en evidencia empírica, de comprobable autenticidad. Es decir, el dato histórico solamente da cuenta de esa unidad mínima de información, pero el hecho histórico proviene de las preguntas que el mismo historiador le hace al documento. El dato histórico puede carecer de relevancia histórica y eso no le desprende su carácter de información. Miles de cartas en los archivos son acuses de recibido de otras comunicaciones, muchas de ellas apenas expresan algo más que la cotidianeidad de la vida administrativa del pasado. Estos documentos están llenos de datos, pero no será hasta que el historiador las someta a crítica que podrán considerarse como testimonios de hechos históricos.[2]

Por lo anterior, podemos saldar este asunto desde ahora, una mayor cantidad de datos no representa mayor objetividad histórica. Significa, eso sí, un incremento en la cantidad de información disponible para emprender cuestionamientos y exploraciones sobre el pasado. Por otra parte, la extracción del dato es una tarea que ya va cargada de subjetividad. Cierta información no afecta el sentido del documento, como la ubicación en el catálogo del archivo, los folios, el estado del legajo. Pero el simple hecho de *observar* (en el sentido de prestar atención a) remite a una identificación de las huellas por parte del historiador. Tal como lo afirmó Paul Ricoeur a mediados del siglo pasado: “observar no significa nunca registrar un hecho bruto”.[3] Por más baladí que parezca un dato histórico, el hecho de haberlo identificado como un elemento que merece la atención para la historia ya lo está dotando de subjetividad. Esa es la razón por la cual en el oficio del historiador se hace necesario ejercitar diariamente la capacidad de separar datos simples de datos históricos, y esa es una tarea que va más allá de la técnica,[4] como trataremos de explicar a continuación.

Datos y fuentes

Hemos visto ya que el dato es una unidad mínima de información y que al ponerla en contexto con un conjunto de datos puede ser dotada de sentido. Ahora pasemos de la abstracción a la práctica con un documento simple, el cual puede ser consultado a través del siguiente enlace: http://pares.mcu.es/ParesBusquedas20/catalogo/show/62474?nm . Este fue tomado del Portal de Archivos Españoles (PARES) y proviene del Archivo General de Indias (AGI), de la sección Estado, legajo 15, número 25. La descripción del documento brindada por el catalogador es la siguiente: “Carta nº 364 del Administrador de Correos D. José Fuertes al Duque de la Alcudia; anuncia la salida del 'Pinzón', que se ha dilatado por el

[2] Ricoeur, *Historia y verdad*, 31-32.
[3] Ricoeur, 31
[4] Nuevamente recurro a Ricoeur cuando dice que “el oficio del historiador hace la historia *y* al historiador”. Ricoeur, 41.

mal tiempo, y da noticia de los útiles auxilios que para esta expedición ha debido al General de Marina D. Juan de Araoz. Duplicado. Con el índice de remisión común con la nº 363”.

La catalogación y la descripción del documento ya han brindado datos relevantes: Origen, ubicación, nombre del remitente, nombre del destinatario, consecutivo de la carta, nombre del navío, originalidad del documento (copia), índice de remisión común. Esta información podríamos trasladarla a una lista:

```
Origen del documento: Archivo General de Indias
Sección: Estado
Legajo: 15
Número: 25
Tipo de documento: Carta
Número original del documento: 364
Originalidad: Copia
Índice de remisión común: 363
Nombre del remitente: José Fuertes
Nombre del destinatario: Duque de la Alcudia
Nombre del navío: Pinzón
```

Sin necesidad de abrir el documento, ya encontramos 11 *campos* en los cuales incorporar información.

Como habrás observado, hubo una pieza de información que dejamos de lado: “da noticia de los útiles auxilios que para esta expedición ha debido al General de Marina D. Juan de Araoz” ¿Cómo incorporar esta información a un campo? ¿Qué nombre le darías a ese campo? Una solución simple consiste en aplicar una técnica de la archivística que sintetiza la carta en un campo llamado “asunto”. En este caso la fila quedaría así:

```
Origen del documento: Archivo General de Indias
Sección: Estado
Legajo: 15
Número: 25
Tipo de documento: Carta
Número original del documento: 364
Originalidad: Copia
Índice de remisión común: 363
Nombre del remitente: José Fuertes
Nombre del destinatario: Duque de la Alcudia
Nombre del navío: Pinzón
Asunto: da noticia de los útiles auxilios que para esta expedición ha
debido al General de Marina D. Juan de Araoz.
```

Esta es una solución sencilla. Si fuera de interés, podría crearse un campo para incluir al general Juan de Araoz, pero este criterio depende fundamentalmente de la pesquisa a la que estemos sometiendo a esa carta. Si estuviéramos siguiendo la carrera del general Araoz, o la protección de las naves españolas contra los corsarios del Caribe, sin duda tendríamos que crear un campo que incluyera a este personaje.

Ahora es el momento de abrir el documento. En el verso del primer folio podemos rescatar dos datos fundamentales: lugar y fecha. La siguiente labor consistirá en leer el documento e identificar información adicional que pueda ser útil para nuestra investigación. Supongamos que nuestro interés consiste en identificar el transporte de correos españoles en el Caribe. En este caso, podríamos ampliar la información con el campo “tipo de navío”, para incluir que este era un bergantín. Añadamos el puerto de origen y el puerto de destino, La Habana y Cádiz respectivamente. También podemos incluir el nombre del capitán del navío, Juan Bautista de Aránaga. El listado quedaría entonces de la siguiente manera:

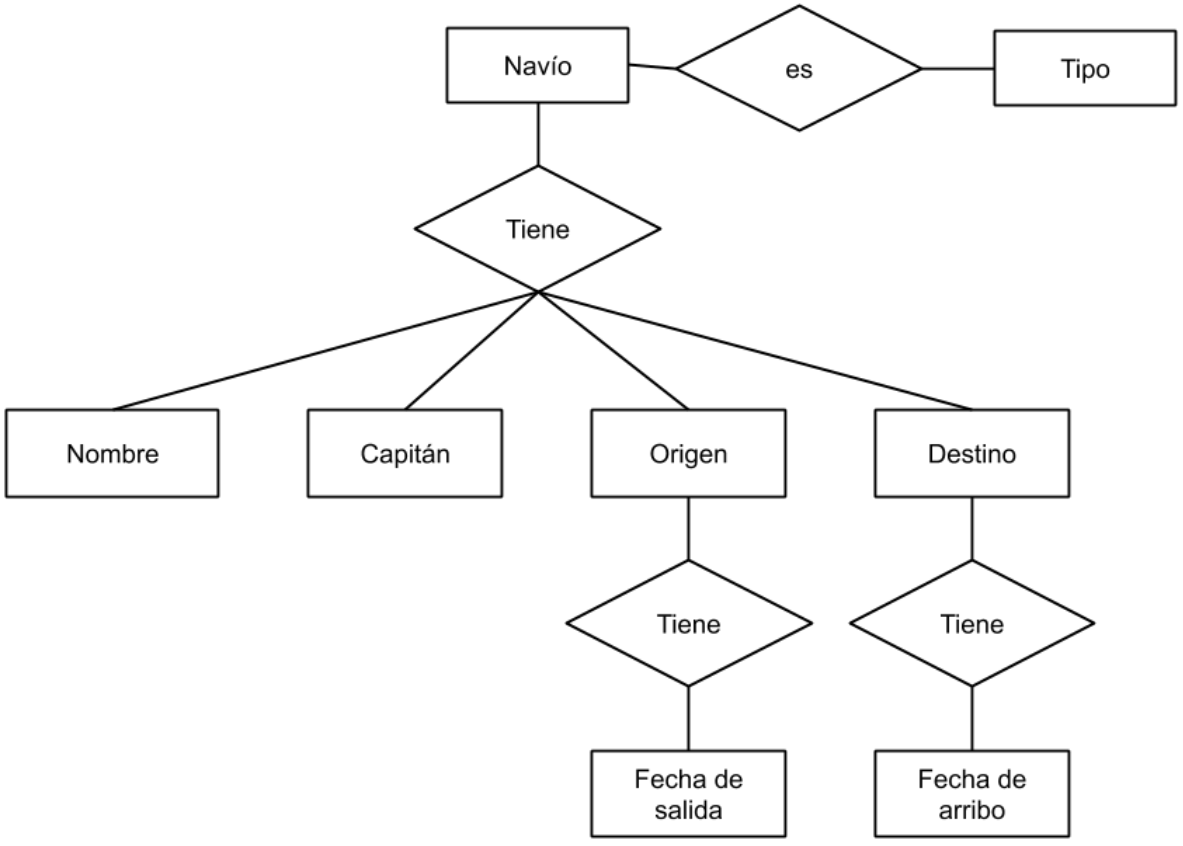
```
Origen del documento: Archivo General de Indias
Sección: Estado
Legajo: 15
Número: 25
Lugar: La Habana
Fecha: 1795-01-26
Tipo de documento: Carta
Número original del documento: 364
Originalidad: Copia
Índice de remisión común: 363
Nombre del remitente: José Fuertes
Nombre del destinatario: Duque de la Alcudia
Nombre del navío: Pinzón
Capitán de navío: Juan Bautista de Aránaga
Puerto de origen: La Habana
Puerto de destino: Cádiz
Asunto: da noticia de los útiles auxilios que para esta expedición ha
debido al General de Marina D. Juan de Araoz.
```

En este momento, nuestro listado se ha ampliado a 17 campos, los cuales podemos ampliar o reducir dependiendo de nuestros intereses. Por lo pronto, evitemos la angustia que produce abandonar el documento sin realizar una transcripción completa y añadamos un campo adicional con el enlace a la carta digitalizada en un campo que llamaremos “url” y que incluirá el enlace completo
http://pares.mcu.es/ParesBusquedas20/catalogo/show/62474?nm

Lo que hicimos en este ejercicio fue construir una *entidad* que es descrita por medio de *atributos*. Dependiendo de los intereses de nuestra investigación las entidades podrán ser personas, lugares, objetos, acciones, eventos, entre otros. Si observaste atentamente, habrás notado que la entidad solamente fue identificada después de decidir el objeto de

nuestra pesquisa: el transporte de correos españoles en el Caribe. Las fuentes que escojamos para esta pesquisa darán más o menos información, por lo que su utilidad estará condicionada a la profundidad del problema que se quiera resolver. Si simplemente queremos identificar el flujo de correo durante un determinado tiempo a través de los puertos del Caribe, la extensión de los campos y el tipo de fuentes a consultar varía significativamente con respecto a un problema que, por ejemplo, quisiera analizar las reformas administrativas que fortalecieron el sistema postal en Indias durante los siglos XVI al XVIII,[5] los campos se ampliarían a tal nivel que difícilmente se podrían incluir en una sola lista o tabla.

Lo ideal es que cada entidad pueda ser graficada con sus atributos y relaciones. Por ejemplo:



La idea básica que se debe entender en este momento es que cada entidad se construye a partir de atributos que pueden ser descritos en forma de relaciones. De este modo, la red puede ampliarse por el campo del “Capitán”, generando así una relación entre dos entidades: navío y capitán. Más adelante cuando construyamos el modelo de la base de datos profundizaremos en este asunto. Lo importante es entender aquí la relevancia de identificar los datos dentro de un tipo documental asociado a una serie, fondo o archivo; así

[5] Lo cual abordó Nelson González en su tesis doctoral de 2018, dirigida por Frédérique Lange, *Communiquer l’empire : l’administration du courrier dans le monde atlantique hispano-américain (1501-1768)*.

como el ejercicio intelectual que debe anteponerse a la extracción de datos de las fuentes históricas.

Por otra parte, tratamos aquí de una fuente escrita bastante simple. Pero la investigación histórica implica tratar con una multiplicidad de fuentes de diversa complejidad. Por esta razón, la base de datos no se lleva al archivo, se construye *en* el archivo. Parte de la experticia del pesquisador consiste en saber cuándo incluir un campo o relación, así como identificar cuándo es necesario eliminar uno de estos. Lo importante será actuar con la suficiente prudencia que permita evitar hacer bases de datos pantagruélicas, entendiendo que con los datos se buscan patrones, no excepciones. Una posible solución consiste en aplicar alguno de estos dos principios:

- Si pocos elementos requieren de un campo específico considera la posibilidad de eliminarlo.
- Si un campo puede combinarse con otro, hazlo.

Ahora bien, hasta el momento hemos tratado los datos como si fuesen texto. Esto no nos permitiría hacer operaciones como combinar, calcular, determinar rangos, entre otras. Para mejorar nuestra base de datos, y que no sea únicamente una tabla, requerimos entender dos conceptos básicos: los tipos de datos y sus niveles de medición.

Tipos de datos

Para nosotros un dato es un dato. Entendemos que “1795-01-26” es una fecha y que “José Fuertes” es un texto, no tenemos ningún problema con ello. Pero las computadoras no tienen ese nivel de comprensión. Tal como hicimos nuestra lista, todos los elementos que en ella están contenidos son textos. Esto no puede parecer un problema en principio, pero supongamos que queremos delimitar cuántos navíos salieron de la Habana entre 1790 y 1800. Al ser texto, sería lo mismo que preguntar cuántos navíos salieron del puerto habanero entre El Pinzón y El Gallego. Por este motivo, necesitamos determinar qué tipos de datos estaremos incluyendo en nuestra base de datos con el propósito de usarlos correctamente.

Es necesario partir del hecho que al asignar un tipo de dato específico le estamos “diciendo” de antemano al sistema cómo queremos usar los datos. Afortunadamente para los historiadores y los humanistas en general, los datos que usamos no son necesariamente complejos. Básicamente usamos números y palabras, pero existen algunos datos que no son inherentes a los lenguajes de programación como las fechas, los valores monetarios o las coordenadas geográficas.

Los tipos de datos más comunes, compartidos por casi todos los lenguajes de programación y sistemas son:

Enteros (*integer*[6]): Este tipo incluye a los números no tienen decimales pero sí valores positivos y negativos. Para el uso de este tipo se debe comprender que el número no se representa como una fracción, de tal manera que es un buen tipo para contabilizar sujetos y objetos no fragmentables. Este es un problema común en la demografía histórica, porque si tenemos siete personas que viven en dos casas el promedio sería 3,5 personas por casa, lo que es un resultado absurdo (no podemos dividir a una persona y dar una mitad a cada casa); pero si se dividen enteros el resultado será 3 personas por casa, lo que deja a un sujeto sin hogar. Por esta razón, el tipo entero es útil precisamente cuando se quieren conservar valores naturales, por ejemplo rangos (1 a 10, 25 a 50, 1750 a 1800).

Coma flotante (*float*): Incluye los números reales, es decir, que tienen decimales. Este tipo es el preferido por los sistemas de programación puesto que permite ejecutar cualquier operación matemática con precisión. En este caso, los números no se expresan como enteros (ej. 1, 2, 3) sino como decimales (ej. 1.0, 2.0, 3.0). Algunos lenguajes de programación como Python favorecen este tipo de datos, de tal manera que si en un campo se combinan números enteros y decimales (ej. 1 y 2.8) el sistema tratará a todos los números como de coma flotante.

Carácter (*character*): Como su nombre lo indica, corresponde a un grafema o símbolo que representa una letra, un dígito, un signo de puntuación común (ej. “.” o “-”) o un espacio en blanco. Hay otros “caracteres no imprimibles” (como el símbolo de párrafo, el tabulador, entre otros) que sirven para que la máquina presente el texto de una manera determinada. Aunque no sea común almacenar un campo con caracteres (o al menos no pensados en términos de caracteres), es importante entender que dependiendo del sistema de codificación de la base de datos (trataremos de ello más adelante) la letra “á” puede verse en pantalla como ‘\u00E1’.

Cadena de caracteres (*string*): En términos simples, este tipo corresponde a los textos. En este tipo de datos se pueden incluir palabras,oraciones, párrafos, incluso un libro podría estar en un campo bajo este tipo. Esta flexibilidad también conlleva una dificultad para gestionar la información. Podríamos estar tentados en incluir simplemente las transcripciones de documentos completos y ahorrar así el aumentar los campos con datos, pero esto conlleva dos problemas fundamentales: la recuperación de información es más lenta (llegando al borde del colapso en bases de datos muy grandes) y las operaciones de análisis de la información son más limitadas. Otro aspecto a tener en cuenta es que muchas cosas se pueden almacenar en forma de cadenas, una url, un código doi, una ruta a un archivo dentro de la computadora, por lo que es definitivamente un tipo de dato dominante en la base de datos, especialmente de los historiadores y humanistas.

Lógico o booleano (*boolean*): Con este tipo de datos se representan valores lógicos binarios, es decir, verdaderos y falsos. Cuando se hacen operaciones con este tipo de valores no se obtiene un número o un carácter sino un valor verdadero o falso.

[6] Incluyo la denominación en inglés por ser bastante común en los sistemas de bases de datos y lenguajes de programación.

Esto permite crear condiciones para proceder o evitar la ejecución de un programa. En términos de una base de datos no se almacenan tipos de dato booleano, aunque puede representarse en forma de entero siendo 0 = falso y 1 = verdadero. Más adelante veremos como utilizar valores booleanos para interactuar con las bases de datos.

Los anteriores son los tipos de datos básicos, comunes a casi todos los lenguajes de programación. Pero para las bases de datos tenemos ciertos tipos que tenemos que tener en cuenta, en particular los tipos de datos de fecha y tiempo, así como los espaciales. Es importante mencionar aquí que dependiendo del sistema de gestión de bases de datos los tipos pueden variar, así que es una tarea del investigador el familiarizarse con estos sistemas para conocer sus limitaciones y ventajas. Esto es relevante en cuanto si queremos pasar una base de datos, por ejemplo, de ACCESS a MySQL, tendremos que tener en cuenta cuáles tipos de datos aceptará la nueva plataforma.

Fecha y tiempo: Este tipo de dato es ampliamente utilizado en la informática, sobre todo para dejar registro del momento en que sucede una interacción con la base de datos (p. ej: publicar un mensaje en Twitter). El tipo de dato fecha almacena esta información con el formato ‘0000-00-00’ o ‘AAAA-MM-DD’. De esta manera, el sistema entiende que los cuatro primeros dígitos corresponden al año, los siguientes al mes y los últimos al día. En caso del tiempo, este se guarda con el formato ‘00:00:00’ o ‘HH:MM:SS’ para horas, minutos y segundos. En MySQL también existe la opción de almacenar solamente el año (muy útil para aquellos que trabajamos con fechas incompletas) en el tipo de dato YEAR con el formato ‘0000’. La diferencia entre usar el tipo de dato fecha y una cadena con el formato ‘AAAA-MM-DD’ consiste en que es posible hacer búsquedas por rango de fechas, mientras en formato de texto habría que buscar coincidencias de año por año.

Datos espaciales: Estos datos permiten almacenar representaciones geométricas asociadas con objetos espaciales. La información contenida allí indica una serie de coordenadas cartesianas que representarán en pantalla puntos, líneas y polígonos. Es posible utilizar coordenadas geográficas para, por ejemplo, indicar un punto en el espacio. Por ejemplo, si quisiera indicar las coordenadas de la “Plaza de Bolívar” de Bogotá podría hacerlo indicando en el campo el dato POINT(4.598270, -74.076001). Para aquellos interesados en el análisis espacial o las representaciones geográficas será de mucha utilidad iniciar la exploración de este tipo de datos.

Existen otros tipos de datos, como el binario, que dependerán de usos avanzados y en general excepcionales dentro de una investigación histórica, por lo que no serán tratados aquí.

Finalmente, si aplicamos la teoría de los tipos de datos a nuestro listado quedaría algo como esto:

```
Origen del documento: "Archivo General de Indias"
```

```
Sección: "Estado"
Legajo: "15"
Número: "25"
Lugar: "La Habana"
Fecha: DATE(1795-01-26)
Tipo de documento: "Carta"
Número original del documento: "364"
Originalidad: "Copia"
Índice de remisión común: "363"
Nombre del remitente: "José Fuertes"
Nombre del destinatario: "Duque de la Alcudia"
Nombre del navío: "Pinzón"
Capitán de navío: "Juan Bautista de Aránaga"
Puerto de origen: "La Habana"
Coordenadas del puerto de origen: POINT(23.136769, -82.347652)
Puerto de destino: "Cádiz"
Coordenadas del puerto de destino: POINT(36.537040, -6.282699)
Asunto: "da noticia de los útiles auxilios que para esta expedición ha
debido al General de Marina D. Juan de Araoz."
URL: "http://pares.mcu.es/ParesBusquedas20/catalogo/show/62474?nm"
```

Todos los elementos que están entre comillas son del tipo cadena de caracteres. Como notarás, los números no son incluidos en el tipo “entero” o “flotante”, y esto se debe a que no haremos cómputos con ellos; no sumaremos legajos, ni multiplicaremos números de documentos, o dividiremos los consecutivos de las cartas. De hecho, solamente aprovechamos dos tipos de datos: fecha y punto. Vale la pena reiterar sobre este punto: el tipo de dato se asigna por lo que se piensa hacer con esta información, no por la “naturaleza” del dato. Un número no requiere ser asignado como un tipo de dato numérico, el tipo de dato numérico se asigna a aquellos datos que requieren algún tipo de cómputo u operación aritmética. Nuevamente, si tuviéramos que trabajar con valores de monedas, cantidad de personas, pesos, inventarios, entre otros datos con valor cuantitativo, en ese caso el tipo de dato numérico será relevante.

Niveles de medida de los datos

Otro aspecto importante para definir los tipos de datos que contendrá cada campo viene definido por sus niveles de medida, es decir, cómo pueden ser usados de manera estadística. No nos extenderemos en este punto puesto que podemos agrupara todos los tipos de datos en cuatro categorías que suelen denominarse según sus siglas en inglés: NOIR.

Nominales: Como su nombre lo indica, esta categoría se refiere básicamente a elementos que pueden ser nombrados. Por ejemplo, el nombre de una persona, el tipo de navío, un lugar. Como podrás suponer, la mayoría de los datos con los que trabajamos serán de tipo nominal. Las operaciones que podremos hacer con estos datos son sencillas, como ordenar alfabéticamente y determinar frecuencias

(cuántas veces aparece un mismo nombre en un rango de datos). Este es el mejor lugar para las cadenas de caracteres.

Ordinales u ordenados: Estos son datos que tienen un orden natural, es decir que pueden ser ubicados en un rango. Un ejemplo sencillo son las posiciones en una carrera o las calificaciones de los estudiantes, ambos son datos que se valoran en escala (p. ej: 1 a 5 = insuficiente, 6 a 7 = aceptable, 8 a 9 = bueno y 10 = excelente). Con estos tipos de datos se suelen hacer operaciones estadísticas como el cálculo de medianas (el valor de la variable de posición central en un conjunto de datos ordenados) y modas (el valor con mayor frecuencia en una distribución de datos)

Intervalos o cardinales: En este tipo de datos los intervalos entre cada valor son iguales sin importar su magnitud. Otra característica importante es que son datos que no tienen valor cero absoluto, es decir, aceptan cantidades negativas. El ejemplo más común es la temperatura. La diferencia de magnitud entre 10 y 11 grados centígrados es la misma que entre 25 y 26 grados. Otros valores que entran en esta categoría son el peso, el tiempo, la altura, las coordenadas geográficas, entre otros. Con estos datos es posible calcular, además de la mediana y la moda, la media o promedio de los datos, así como el cálculo de máximos y mínimos dentro de un cierto rango.

Ratio o de razón: Corresponde a los datos que cuentan con un cero absoluto. El ejemplo más sencillo para explicar estos datos es la edad. Una persona puede tener un valor de 0 años, pero no de -1 años. En general, las medidas temporales que tienen un inicio y un final pueden ser incluidas en este orden de medida, con la clara excepción de las fechas, que son datos nominales. También podemos incluir aquí la cantidad de personas. Si en un lugar no hay personas es posible decir que hay “0 personas”, pero no es factible afirmar que existen “-1” personas. Además de las operaciones estadísticas que se realizan con los datos ordinales y cardinales, es posible someter estos datos a operaciones más complejas como desviaciones estándar o pruebas t.

La relevancia de saber cómo medir estadísticamente los tipos de datos se fundamenta en un error frecuente en los análisis de información. No son pocas las ocasiones en que los datos se analizan en escalas aritméticas o se interpretan valores estadísticos como absolutos. Es importante conocer qué tipo de operaciones básicas se pueden hacer con los datos con los que contamos para de esa manera determinar qué tipo de datos vamos a incluir en nuestra base de datos. Lo más común será que utilicemos datos nominales y de razón, pero sabemos que una deuda significa tener menos dinero, por lo que el intervalo será una forma útil para medir la información; también la medida entre dos puntos dará cuenta de la distancia y será posible calcular promedios entre varias rutas en una escala de tiempo. Las posibilidades son amplias, pero, hay que reiterar, depende específicamente de lo que queramos hacer con la información, no de lo que los datos representan por sí mismos.

¿Y los archivos?

Es bastante probable que te estés haciendo esa pregunta. Si mi información está guardada en PDF, o en imágenes, ¿dónde la voy a “guardar”? El asunto está precisamente en el “guardar” la información. Un archivo no es un dato, ni un conjunto de datos, no en el sentido que pueda manejarlo una base de datos por lo menos. Un documento en PDF no puede ser “guardado” en una base de datos porque ésta no guarda *objetos*. Pero esto no quiere decir que la base de datos no pueda incluir datos relacionados con el archivo, de hecho estos tienen un nombre específico: *metadatos*.

En las bases de datos en lugar de guardar un archivo (o su código binario), se incorporan metadatos que describen e identifican al objeto. Algunas tienen que ver simplemente con sus características físicas, sean analógicas o digitales. Por ejemplo, un metadato puede ser el formato de archivo (pdf, doc, txt, jpg…), su tamaño digital (Kb, Mb, Gb) e incluso físico (peso, largo, ancho), asimismo, información sobre su soporte (digital, papel, pergamino…) o su estado de conservación. Otros metadatos pueden incluir información propia del contenido del documento como la fecha de elaboración, el autor, las páginas, el lenguaje, la cobertura espacial y temporal, entre otros. También se pueden incluir datos propios de la propiedad y uso del documento como el archivo o biblioteca en el que se encuentra resguardado, el código de ubicación, o los derechos de uso, copia y redistribución.

Esta información puede ser relevante o irrelevante, dependiendo del propósito de la base de datos. En un inventario de archivo o biblioteca los metadatos que describen las características físicas del objeto son fundamentales, pero para una investigación histórica no lo pueden ser tanto; aunque hay que decirlo, es siempre útil incluir un campo que describa el estado de conservación del documento, pues esta es una variable que en ocasiones permite o impide el acceso a la información completa de un folio o legajo.

En el listado de datos que elaboramos para el documento incluimos algunos metadatos, proveídos por el propio Archivo General de Indias, como son la ubicación física (signatura) y la dirección del recurso digital (URL). Si los archivos están guardados en un disco duro u otro soporte de almacenamiento digital es una buena práctica incluir la ruta que conduce al objeto. El inevitable problema con *apuntar* hacia rutas fijas es que si cambiamos de computadora o el sitio Web modifica sus URL (p. ej. cambia su dominio) tendremos que modificar todas las rutas que escribimos en nuestra base de datos. Por esta razón, si un sitio Web pone a disposición identificadores de objetos digitales (como los *permalink* y *doi*), deberíamos priorizar su uso antes que las tradicionales URL.

Como ejemplo, la siguiente imagen muestra la forma en que la Biblioteca Estatal de Bavaria gestiona sus direcciones Web para cada objeto digital. En este caso, una copia digitalizada de *De indianum iure* de Juan de Solórzano Pereira del año 1672. Nótese que la ruta del *permalink* es totalmente diferente a la de la url, pero si hacemos clic en el enlace permanente (http://mdz-nbn-resolving.de/urn:nbn.de.bvb:12-bsb11396247-0) nos enviará al mismo lugar que la URL. Esto se debe a que cada objeto digital tiene una dirección absoluta (urn:) y si hay cambios en la url un *script* adaptará la ruta para que no haya un “enlace roto”.

Los metadatos, como cualquier dato, no corresponde a una estructura rígida. Las bases de datos nos permitirán incluir como metadatos cualquier campo que consideremos que es útil para describir el documento, imagen, escultura, edificio, persona, animal, lugar, o lo que sea que queramos identificar. Esto representa un problema para ciertos proyectos porque si cada quien decide a su arbitrio qué metadatos incluir en su base de datos prácticamente habrá tantos modelos de metadatos como bases de datos en existencia. Por esta razón se han desarrollado esquemas de metadatos, que funcionan como estándares para la creación de sistemas que puedan ser compartidos y entendidos por otros (usuarios y sistemas).

No existe un único o mejor sistema de metadatos. Como tantos aspectos de las ciencias de la computación, todo depende de lo que queramos hacer. En realidad, la cantidad de esquemas y estándares pueden ser abrumadores. Pero en términos de una base de datos de investigación, alguno de los más comunes y documentados puede ser de gran utilidad.

Dublin Core es un esquema ampliamente utilizado a nivel mundial, especialmente entre los humanistas. La versión 1.1, hoy considerada “medianamente” obsoleta, constaba de quince elementos para describir un objeto físico o digital: contribuidor (entidad), cobertura (espacial o temporal), creador o autor, fecha, descripción, formato, identificador, lenguaje, editor, relación (con otro objeto), derechos, fuente (de la cual se derivó el producto), tema, título y tipo (de recurso). La última versión del Dublin Core incluye una serie de términos enfocados en la Web Semántica y en el uso de Datos Enlazados. Un listado de estos términos puede encontrarse en la siguiente entrada de Wikipedia, https://en.wikipedia.org/wiki/Dublin_Core#DCMI_Metadata_Terms, y la descripción de cada uno puede consultarse en su sitio oficial, https://www.dublincore.org/specifications/dublin-core/dcmi-terms/.

Los estándares de metadatos constituyen una regla pero no una obligación. Reitero, podremos construir una base de datos sin requerir incluir un esquema de metadatos o de términos relacionales; pero esto no implica que siempre realicemos bases de datos exclusivamente para nuestras investigaciones personales. Proyectos de difusión histórica,

de preservación y digitalización documental, de promoción del patrimonio, de educación, entre varios otros campos en los que se puede vincular un historiador profesional, se ven enriquecidos si tenemos la capacidad de por lo menos entender de qué significan los metadatos y cómo pueden utilizarse. Es precisamente en nuestras investigaciones personales donde tenemos mayor libertad para experimentar y equivocarnos,[7] así que no estaría de más destinar un poco de tiempo en aplicar estos esquemas en nuestras bases de datos para aprovechar estas habilidades en el futuro.

Bases de datos

La base de datos consiste simplemente en una colección organizada de datos, que pueden ser analógicas o digitales. Un catálogo bibliográfico puede ser considerado una base de datos en tanto contiene información organizada de todos los elementos dentro de una biblioteca. De esta manera, nuestros colegas que visitan el archivo y diligencian ordenadamente las fichas en cartón están haciendo, de cierta manera, una base de datos analógica. El problema obvio que surge de una base de datos en papel es que es muy difícil de consultar y extraer información de ella consume mucho tiempo, además de ser susceptible a una mayor cantidad de errores de lectura. Las bases de datos digitales son fundamentales hoy en día porque permiten gestionar la información de una manera más rápida y eficiente.

En ciertas ocasiones suelen equipararse los términos “base de datos” y “banco de datos”, pero no debería asumirse que son sinónimos, menos que una es una traducción “más ajustada al castellano” que la otra. El término base de datos hace referencia a las colecciones en general, independientemente de su tipo, diseño o tecnología. Con banco de datos nos referimos a un *repositorio* de información sobre uno o varios temas específicos, organizada de tal manera que permita la recuperación remota de esta información. En inglés, el término *data bank* se considera obsoleto y se denomina como *database* todas las colecciones. En el idioma español, tenemos el término repositorio (derivado del término *content repository*), que se ha popularizado gracias a las instituciones académicas que han adoptado sistemas de depósito para los trabajos que producen sus investigadores, profesores y estudiantes.

Otra advertencia importante antes de iniciar con este tema: una base de datos no es una tecnología específica. Es tan válido hacer una base de datos en archivos de texto como hacerlo con LibreOffice Base o cualquier otro software de gestión de bases de datos. La forma como gestiones tu base de datos dependerá del tiempo, conocimientos y recursos disponibles. Una base de datos en archivos .csv requiere menos recursos de computadora, pero exige mayor conocimiento para interactuar con los datos que están contenidos en esos archivos. Por otra parte, aplicaciones comerciales como Atlas.ti o MAXQDA requieren

[7] De hecho, el fracaso es en sí un tema de las Humanidades Digitales. Jasmine Simone Kirby, “How NOT to Create a Digital Media Scholarship Platform: The History of the Sophie 2.0 Project”, *IASSIST Quarterly* 42, núm. 4 (2018): 1–16, https://doi.org/10.29173/iq926; Quinn Dombrowski, “Towards a Taxonomy of Failure”, *Quinn Dombrowski* (blog), el 30 de enero de 2019, http://quinndombrowski.com/?q=blog/2019/01/30/towards-taxonomy-failure; Max Kemman, “DH Failures vs Findings”, *Max Kemman* (blog), el 28 de febrero de 2019, https://www.maxkemman.nl/2019/02/dh-failures-vs-findings/.

mejores recursos de computadora y un presupuesto para adquirir el programa, pero evita que tengas que diseñar y gestionar tu base de datos. Es decir, no importa la tecnología, metodología o sistema que uses; lo que debes tener en cuenta es que los resultados deben ser iguales sin importar la plataforma. En ese sentido, no hay mayor mérito en usar archivos de texto o software libre que en recurrir a programas de pago y código propietario. Los datos son los que deben hablar y las preguntas que hagamos deben ser relevantes, todo lo demás es técnica y práctica.

Modelar los datos

Ahora, pensemos en los datos que haremos en nuestra base de datos. En este momento tenemos conformado un listado con 20 campos, si le agregamos los metadatos de cada objeto el listado podría ampliarse más de lo que deseamos. Lo ideal es que las bases de datos se compongan de varias tablas de unas cuantas columnas. ¿Recuerdas que anteriormente hablamos de las *entidades*? Bueno, lo ideal es que cada tabla represente la información adecuada para una entidad. En este caso, en lugar de ampliar el listado con los metadatos del documento deberíamos hacer una lista aparte, similar a la siguiente:

```
`documentos`
id: 1
Origen del documento: "Archivo General de Indias"
Sección: "Estado"
Legajo: "15"
Número: "25"
Lugar: "La Habana"
Fecha: DATE(1795-01-26)
Tipo de documento: "Carta"
Número original del documento: "364"
Originalidad: "Copia"
Índice de remisión común: "363"
Nombre del remitente: "José Fuertes"
Nombre del destinatario: "Duque de la Alcudia"
Asunto: "da noticia de los útiles auxilios que para esta expedición ha debido al
General de Marina D. Juan de Araoz."
url: "http://pares.mcu.es/ParesBusquedas20/catalogo/show/62474?nm"
```

```
`navios`
id: 1
id_fuente: 1
Nombre del navío: "Pinzón"
Capitán de navío: "Juan Bautista de Aránaga"
Puerto de origen: "La Habana"
Coordenadas del puerto de origen: POINT(23.136769, -82.347652)
Puerto de destino: "Cádiz"
Coordenadas del puerto de destino: POINT(36.537040, -6.282699)
```

¿Por qué nos sería útil este esquema? Sencillamente porque todo viaje tiene un tornaviaje. Sería muy extraño si tomáramos un diario de viaje y anotáramos la misma información para

el origen y el destino. Con este sistema podemos anotar varios puntos del viaje del “Pinzón” sin necesidad de repetir una y otra vez la información del documento. Por ejemplo, si quisiéramos hacer referencia al tornaviaje (suponiendo que el mismo documento lo señala) escribiríamos algo así:

```
`navios`
id: 2
id_fuente: 1
Nombre del navío: "Pinzón"
Capitán de navío: "Juan Bautista de Aránaga"
Puerto de origen: "Cádiz"
Coordenadas del puerto de origen: POINT(36.537040, -6.282699)
Puerto de destino: "La Habana"
Coordenadas del puerto de destino: POINT(23.136769, -82.347652)
```

Aquí introdujimos otro concepto importante: el identificador. Este es un número único que se asocia con cada entidad. En este caso, cada viaje tiene un identificador (1 y 2) y asimismo lo tiene el documento fuente (1). Cuando relacionamos entre varias tablas lo hacemos a través de identificadores que suelen ser números consecutivos pero no jerárquicos (son datos de razón).

Lo que estamos haciendo consiste en modelar la base de datos, que no es otra cosa sino establecer qué relaciones van a tener nuestros datos entre sí. En nuestro esquema las relaciones van a ser bastante sencillas pero, mientras más complejos sean nuestros datos o mayores sean las operaciones que necesitemos hacer con ellos, podremos requerir esquemas más complicados. Una recomendación es que hagamos algunos esquemas preliminares siguiendo la lógica de los datos que nos brindan las fuentes y el propósito de la investigación que estemos llevando a cabo, para luego iniciar con el modelado y la creación de la base de datos.

Relaciones

Las relaciones son uno de los aspectos más interesantes dentro de la metodología de las bases de datos. Por medio de estas, podemos crear sistemas complejos de datos que pueden ser vinculados entre sí sin necesidad de estar repitiendo información.

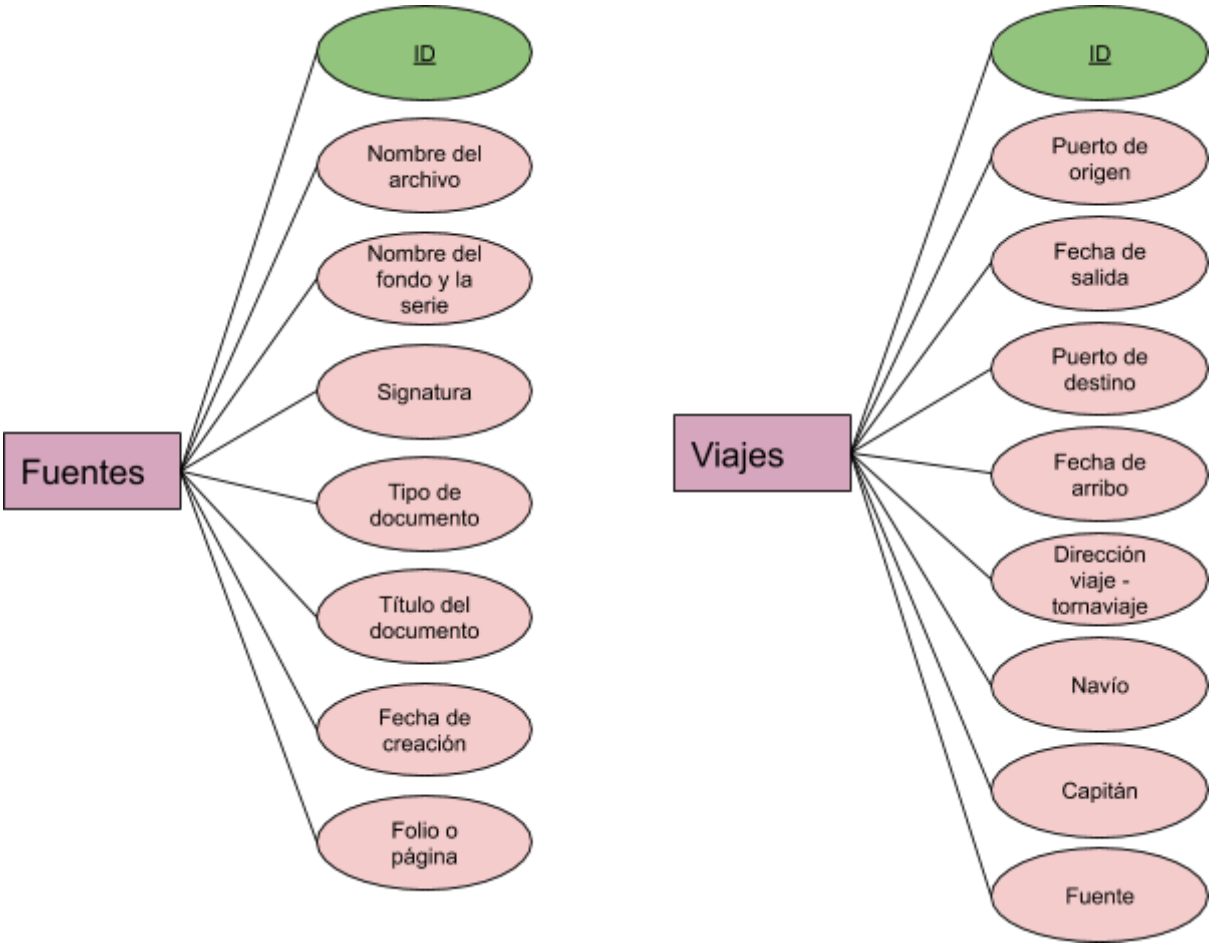
La lógica para indicar las relaciones está marcada por la idea de *cardinalidad*, es decir, por la relación numérica entre columnas de dos tablas. Por lo general, estas cardinalidades se enuncian como relaciones de uno-a-uno, uno-a-muchos y muchos-a-muchos. Vamos a ver cómo funciona esto con nuestro ejemplo.

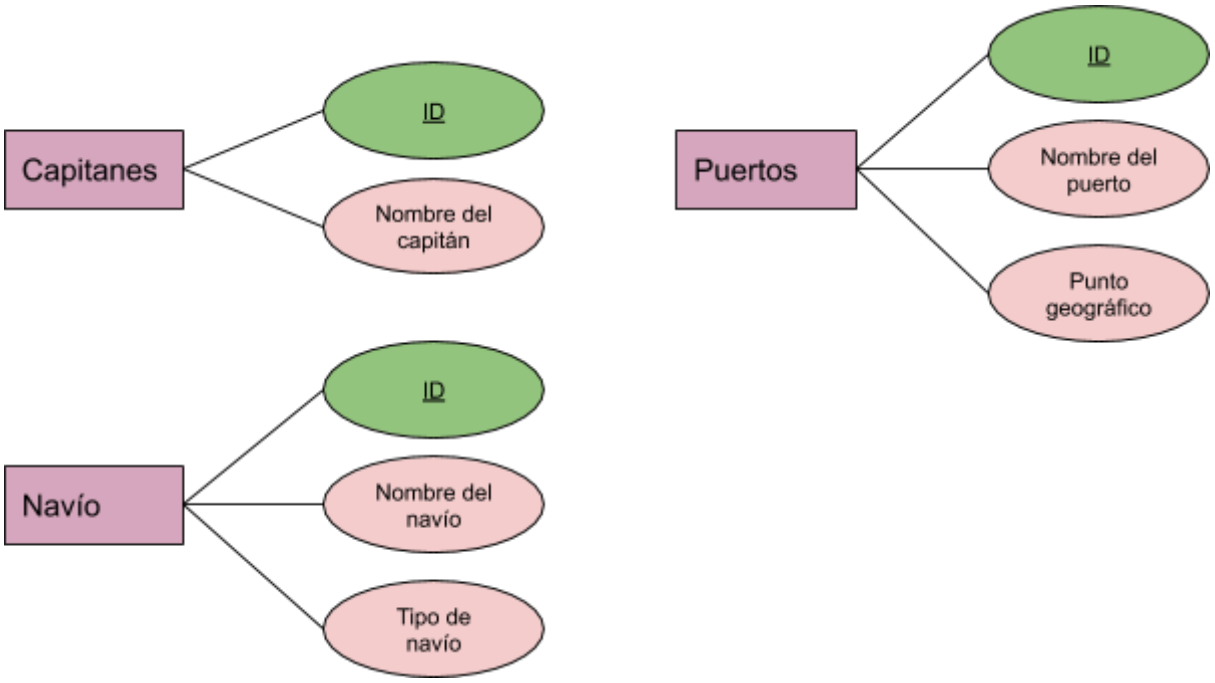
Podemos considerar que nuestra base de datos consta de cinco entidades:

- Fuentes: Metadatos del documento que da testimonio del viaje.
- Viajes: Información relativa al trasegar del navío entre varios puertos.
- Capitanes: Nombre y otros datos relacionados con el capitán del navío.
- Navío: Nombre y tipo de navío que llevó a cabo el viaje.

- Puertos: Datos georreferenciados de los puertos donde ancló el navío durante el viaje.

También podemos establecer que nuestras entidades pueden ser descritas con los siguientes atributos:



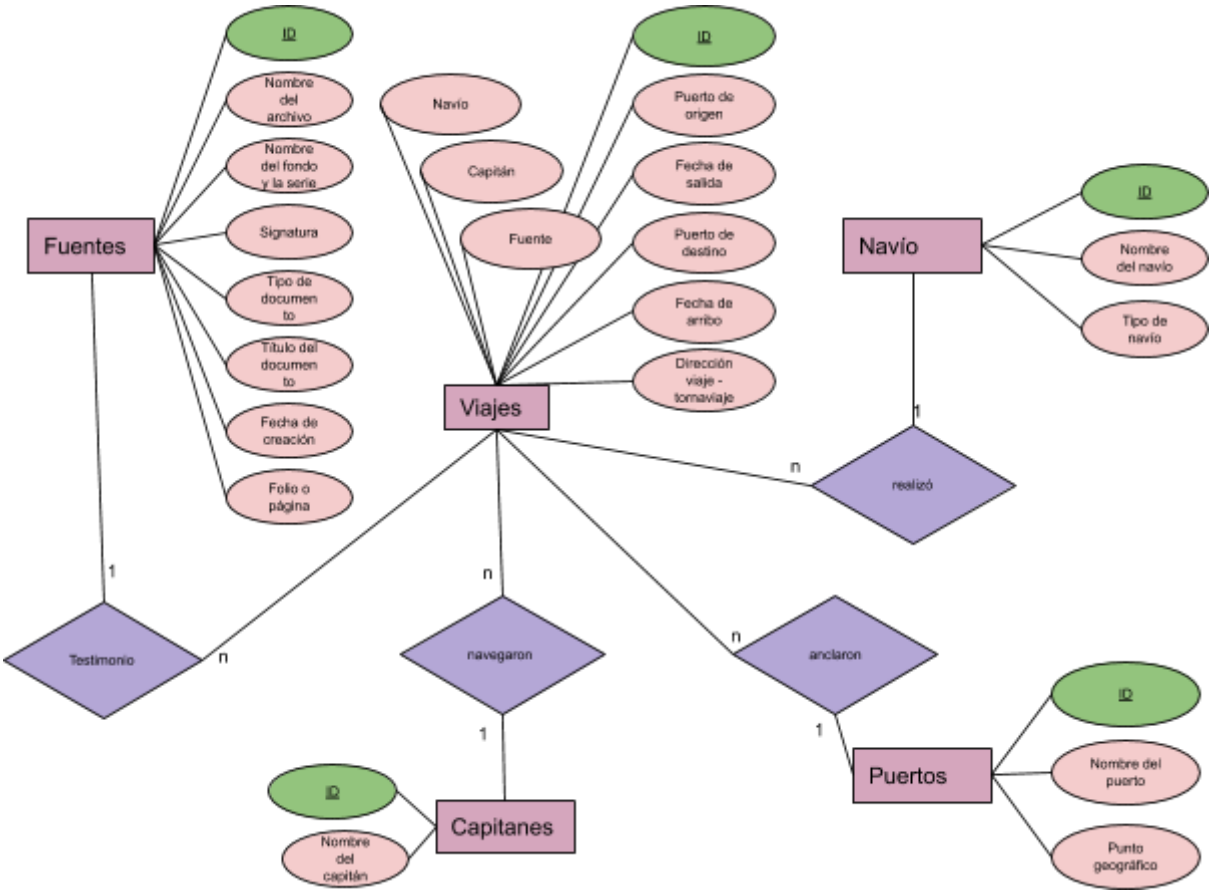


Con esta información, ya podemos hacer nuestras tablas. Pero si queremos que el conjunto de estas tablas conformen una base de datos tendremos que relacionarlas. Como vemos en la entidad `viajes` tenemos unos atributos que indican relación con las otras tablas: `puerto de origen`, `puerto de destino`, `navío`, `capitán` y `fuente`. Para hacer la relación necesitamos saber la cardinalidad de la relación, las cuales podemos describir así:

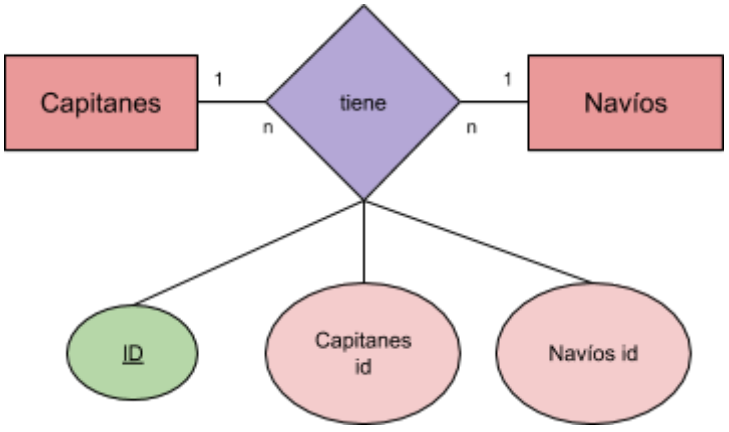
- Una fuente puede dar cuenta de uno o muchos viajes.
- Un capitán puede realizar uno o muchos viajes.
- Un puerto puede ser origen o destino de uno o muchos viajes.
- Un navío puede realizar uno o muchos viajes.

Vemos que todas nuestras relaciones son de uno a muchos. Una relación de uno a uno sería, por ejemplo, el certificado de nacimiento de una persona, ya que no es posible que una misma acta de nacimiento se relaciones con diversas personas. Por otra parte, una relación de muchos a muchos sería, por ejemplo, los cursos en los que están registrados los estudiantes, ya que muchos estudiantes están incluidos en muchos cursos y viceversa.

Gráficamente podemos representar estas relaciones de la siguiente manera:



Vemos como se establecen las relaciones entre las diferentes entidades. En este caso, todo está centrado en la entidad `viajes` porque nuestra información es bastante simple. No obstante, podemos crear otras relaciones, por ejemplo, entre capitanes y navíos pues un navío tuvo varios capitanes y un capitán lideró varios navíos. La relación quedaría entonces de la siguiente manera:



En este caso los atributos se asignan a la relación, por lo que se creará una tabla con los tres campos: id, capitanes id y navíos id.

Creación de la base de datos

Como ya conocemos los tipos de datos que vamos a utilizar, así como las relaciones que se establecerán entre ellos, podemos pasar a crear la base de datos. Como dijimos anteriormente, fácilmente podrías crear una base de datos simplemente guardando archivos de texto en un directorio de tu computadora, pero esto haría más complicado el establecer relaciones y hacer tareas complejas con varias tablas. Lo que haremos será utilizar una herramienta para la gestión de una base de datos, lo cual tiene varias ventajas, entre ellas proveer de un entorno gráfico para manejar la información y familiarizar al usuario con los comandos de SQL.

Instalación del sistema de gestión de base de datos y las herramientas adicionales

Para nuestra base de datos vamos a utilizar un lenguaje de programación, un sistema administrador de bases de datos relacionales, y una herramienta de base de datos.

- **Lenguaje de programación:** Para nuestra tarea recurriremos a SQL (siglas para *Structured Query Language*), un lenguaje diseñado para administrar y consultar información de bases de datos relacionales. No obstante, hay que hacer una advertencia: no todos los sistemas de gestión de bases de datos manejan el mismo estándar del lenguaje. MySQL y PostgreSQL están basados en SQL, pero una consulta programada para un sistema puede que no funcione en el otro. Programas como LibreOffice BASE y Microsoft ACCESS incluyen extensiones para hacer consultas en SQL, así que es en general un lenguaje bastante amplio para usarse en un buen número de sistemas de bases de datos relacionales.
- **Sistema administrador de bases de datos relacionales:** En nuestro caso usaremos MariaDB, la versión gratuita y comunitaria de MySQL[8], el sistema más utilizado a nivel mundial (junto a Oracle y Microsoft SQL Server). Wikipedia, Facebook, Twitter, Youtube y otras grandes compañías de la Web utilizan este sistema para gestionar sus bases de datos. De igual manera, sistemas de gestión de contenido como WordPress, Joomla, Omeka, DSpace, entre otras más, están estructuradas en MySQL. Por esta razón es apenas lógico que utilicemos este sistema para realizar nuestra base de datos. Otros sistemas bastante populares y de código abierto son PostgreSQL y SQLite[9].
- **Herramienta de base de datos:** Podríamos interactuar directamente con MySQL desde la terminal o línea de comandos[10], pero para facilitar un poco la gestión de la base de datos utilizaremos la herramienta PhpMyAdmin, la cual se ejecuta desde el

[8] MySQL es en buena medida gratuito, pero después de haber sido adquirido por Oracle en 2009 se creó MariaDB como una bifurcación con una alta compatibilidad con MySQL. La versión actual de MariaDB (10.5.5) es compatible casi totalmente con MySQL 5.7. La última versión de MySQL (8.0) no es compatible con MariaDB.

[9] Véase Anexo:Comparación de sistemas administradores de bases de datos relacionales

[10] Al respecto, recomiendo la consulta de los tutoriales de The Programming Historian dedicados a la línea de comandos: Ian Milligan y James Baker, "Introducción a la línea de comandos en Bash", traducido por Víctor Gayol, *The Programming Historian en español* 1 (2017), https://doi.org/10.46430/phes0013; Ted Dawson, "Introducción a la línea de comandos de Windows con PowerShell", traducido por Victor Gayol, *The Programming Historian en español* 2 (2018), https://doi.org/10.46430/phes0037.

navegador. Ambas son herramientas de uso extendido en la gestión de bases de datos, así que no estará de más conocer su funcionamiento básico. Ahora que si deseas, puedes darle un vistazo al amplísimo universo de herramientas de base de datos y escoger una que se adapte mejor a tus necesidades[11].

Para instalar MySQL podemos hacerlo directamente en la computadora siguiendo las instrucciones de instalación para Windows[12], macOS,[13] o Linux[14]. Estos instaladores por lo general traen un sistema de servidor integrado que permitiría hacer consultas a los archivos. Dependiendo de tu gusto por el uso de la línea de comandos, puedes hacer cualquier tarea de las aquí propuestas. No obstante, preferimos realizar una instalación completa que incluya la herramienta de base de datos.

Utilizaremos en adelante el entorno de desarrollo de aplicaciones XAMPP[15], una distribución del servidor Apache que contiene además MariaDB 10.4 y phpMyAdmin 5.0[16]. Descarga la versión adecuada para tu sistema operativo desde el sitio oficial de XAMPP y procede a la instalación del software. Los instaladores son bastante sencillos de utilizar, pero en caso de requerir instrucciones puedes consultarlas para Windows, Mac OSX y Linux[17]. Después de instalado, el sistema de base de datos se activará automáticamente.

Reitero que más que enfocarnos en las herramientas debemos centrarnos en los conceptos. Las versiones de una aplicación cambian continuamente y un tutorial fundamentado en cierto tipo de software puede resultar obsoleto en cuestión de meses. De hecho, hacemos la invitación para replicar los pasos que seguiremos adelante en otros programas como LibreOffice BASE o MySQL Workbench.

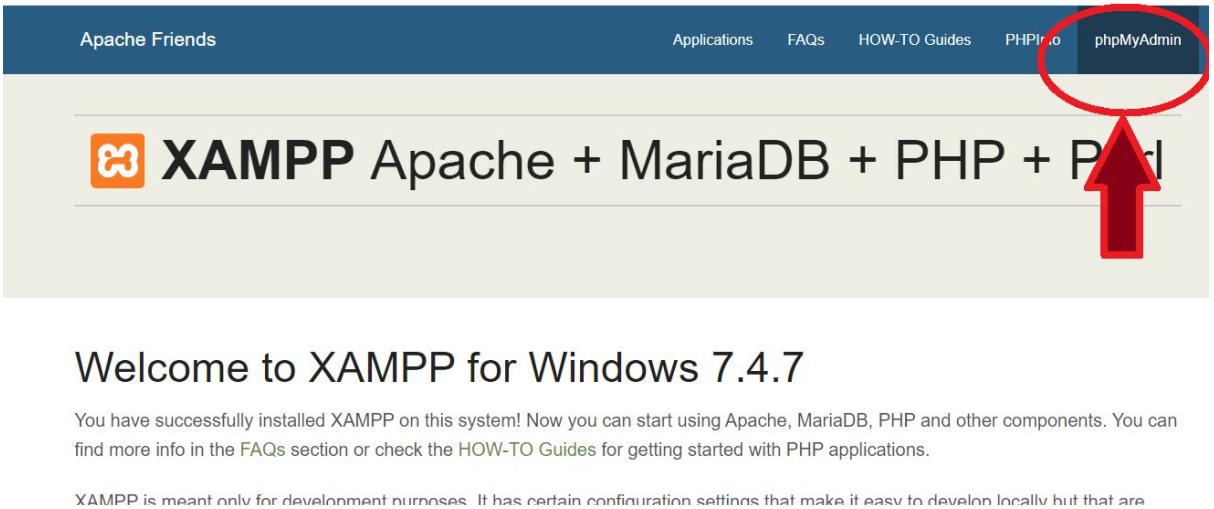
Los pasos que seguiremos a continuación serán bastante básicos, así que no nos detendremos mucho en su explicación.

Crear la base de datos

Para acceder al sistema phpMyAdmin deberás abrir una ventana de un navegador Web (Chrome, Mozilla, Opera o cualquier otro) e ingresar a la dirección http://localhost o http://127.0.0.1. Si todo funciona correctamente[18], veremos un enlace a phpMyAdmin en la

[11] Véase Anexo:Comparativa de herramientas de bases de datos
[12] MySQL 8.0 Reference Manual :: 2.3 Installing MySQL on Microsoft Windows
[13] MySQL 8.0 Reference Manual :: 2.4 Installing MySQL on macOS
[14] MySQL 8.0 Reference Manual :: 2.5 Installing MySQL on Linux
[15] XAMPP Installers and Downloads for Apache Friends
[16] La información sobre otros componentes incluidos en la versión que vamos a utilizar está disponible en New XAMPP release 7.2.33 , 7.3.21 , 7.4.9
[17] También puedes seguir las instrucciones disponibles en el tutorial Go Sugimoto, "Introduction to Populating a Website with API Data," *The Programming Historian* 8 (2019), https://programminghistorian.org/en/lessons/introduction-to-populating-a-website-with-api-data#xampp-installation.
[18] Partimos del supuesto que tu computadora no tiene instalado XAMPP u otra distribución similar (LAMPP o WAMPP) y que no hay conflictos con otras aplicaciones. Si después de instalar XAMPP no se carga el escritorio en http://localhost puede ser un problema con los puertos. Puedes revisar algunas maneras de corregirlo en esta entrada de Stackoverflow en español: Apache con Xampp no inicia por puertos bloqueados

barra de navegación superior. También podemos acceder directamente al programa desde la dirección http://localhost/phpmyadmin/.



En la página de inicio de phpMyAdmin, en la columna izquierda, haz clic en “Nueva”. Se abrirá un formulario sencillo con una casilla en blanco para poner el nombre de la base de datos y una lista desplegable para el cotejamiento. Vamos a usar el nombre `bdd_navios` para nuestra base de datos y el cotejamiento `utf8mb4_general_ci`. El cotejamiento (*collation*) se refiere a una serie de reglas para interpretar un conjunto de caracteres. Necesitaremos un cotejamiento que pueda interpretar caracteres como la \ñ\, las vocales tildadas y las diéresis[19].

Si quisieras crear la base de datos desde una línea de comandos o desde la consola de phpMyAdmin (la abres desde la pestaña “SQL”) lo puedes hacer con el siguiente código:

```
CREATE SCHEMA IF NOT EXISTS `bdd_navios` DEFAULT CHARACTER SET utf8mb4
```

Puedes hacer el ejercicio, haz correr este código en la consola de phpMyAdmin, cambia ciertas partes del código para ver qué sucede (por ejemplo, elimina la condición IF NOT EXISTS e interpreta el mensaje de error). Lo ideal es que pruebes y experimentes con este código, no que simplemente sigas instrucciones.

[19] Este video del canal Computerphile es bastante interesante para comprender el sistema de cotejamiento utf-8 Characters, Symbols and the Unicode Miracle - Computerphile

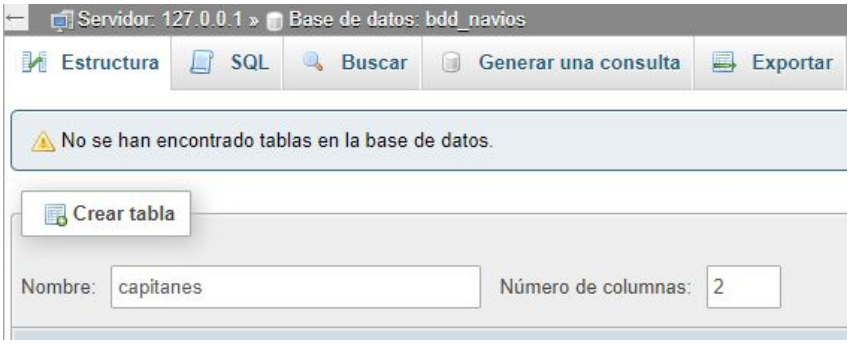
Crear las tablas del esquema

Crearemos seis tablas de acuerdo con las relaciones que establecimos anteriormente, para lo cual usaremos los siguientes nombres:

- capitanes
- capitanes_de_navio
- fuente
- navio
- puertos_georreferencia
- viajes

Nota que no usamos tildes, diéresis ni espacios; no porque no sea posible, sino porque es una convención que evita ciertos errores que se podrían presentar por programas que no usen el esquema utf-8.

Para crear las tablas simplemente haremos clic en el nombre de la base de datos que aparece en la columna izquierda y aparecerá un formulario en la pestaña “Estructura” que dirá “Crear tabla” con una casilla para el nombre y otra para la cantidad de columnas. Iniciemos con la tabla `capitanes` que tiene dos atributos.



Al darle continuar aparecerá un formulario dividido en dos partes. La primera está destinada a crear las columnas, la segunda a la estructura de la tabla. Llenaremos la primera parte con las siguientes instrucciones:

- La primera columna deberá estar dedicada a la clave principal. La llamaremos `idcapitanes`, el tipo de datos será INT, la longitud será 11, en índice seleccionaremos PRIMARY (esto desplegará una ventana, solamente dale continuar). Desliza la ventana a la derecha y selecciona A_I, que significa incremento automático. Los demás campos los dejamos sin modificar.
- La segunda columna la dedicaremos al nombre del capitán. Vamos a ingresar el nombre completo en una sola columna, pero podrías definir una para nombre y otra para apellidos. Llamaremos esta columna `nombre_capitan`, seleccionaremos el tipo VARCHAR, que es la manera como MySQL nombra a los conjuntos de caracteres. La longitud será de 100 caracteres, que sería suficiente para la mayoría de los

nombres. En caso de requerir una mayor cantidad, solamente deberás ampliar ese valor al crear la tabla o posteriormente actualizarla.

Ahora podremos dar clic al botón “Guardar” y se habrá creado nuestra tabla.

Nuevamente, podríamos crear la tabla directamente desde la consola de phpMyAdmin simplemente escribiendo el siguiente código:

```
CREATE TABLE IF NOT EXISTS `capitanes` (
  `idcapitanes` int(11) NOT NULL AUTO_INCREMENT,
  `nombre_capitan` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`idcapitanes`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
```

Nota que están en este código todas las instrucciones que incorporamos en la tabla: crear la tabla si no existe, el nombre de la tabla, cada característica de las columnas, la asignación de la clave primaria, el motor de almacenamiento (InnoDB) y el cotejamiento predeterminado.

Repite el procedimiento y crea la tabla `navio` para la entidad “Navíos”. Usa el formulario o la consola para crearla. Asigna el nombre de las columnas como `nombre_navio` y `tipo_navio`. Asegúrate que el resultado sea la siguiente estructura.

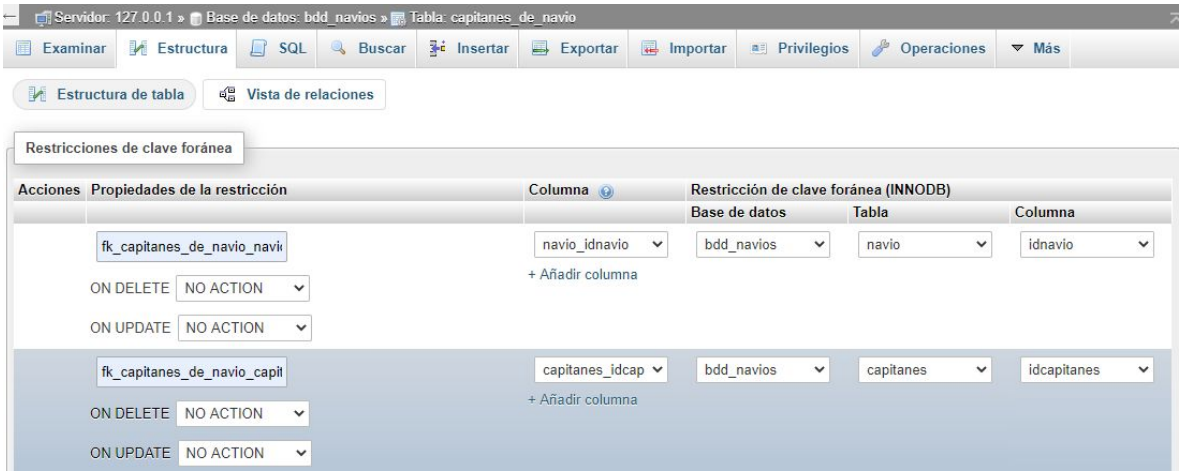
Ahora vamos a hacer algo más complejo, la tabla de relaciones “capitanes - tiene - navíos” que tendrá como atributos las claves de las tablas `capitanes` y `navios`. La nombraremos `capitanes_de_navio`. La clave primaria de esta tabla será llamada `idcapitanes_de_navio` (recuerda seleccionar la opción AUTO_INCREMENT). Para la clave de capitanes llamaremos la columna `capitanes_idcapitanes` y para navíos `navio_idnavio`. Todas las columnas serán del tipo INT y tendrán una longitud de 11.

Después de crear la tabla con esos valores tendremos que hacer que todos los índices sean considerados como claves primarias. Para ello iremos a “estructura” y en la sección “índices”, en la columna “Acción”, haremos clic en “Editar” la clave PRIMARY. En la ventana que se abre agregaremos una columna para `capitanes_idcapitanes` y otra para `navio_idnavio`. Dale continuar y estamos listos.

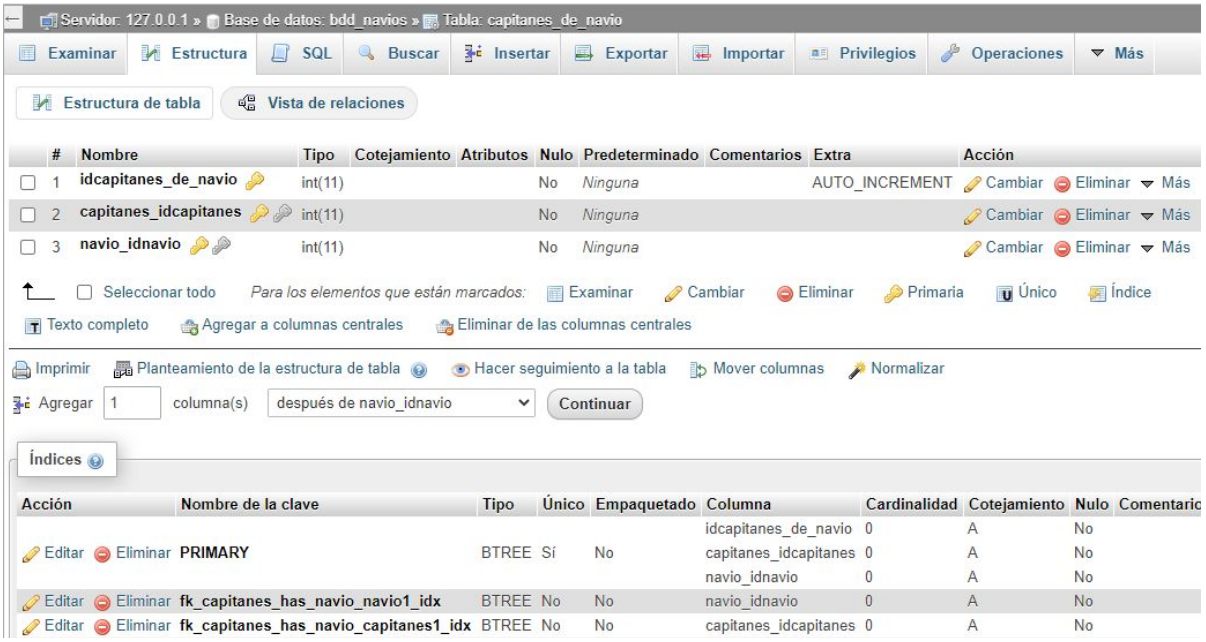
También tendremos que crear unos índices para indicar las relaciones entre tablas. En la misma sección “Índices” daremos clic en el formulario que dice “Crear un índice en [1] columna(s)”. En la ventana que se despliega escribimos como Nombre del índice `fk_capitanes_de_navio_navio1_idx`, como opción del índice seleccionamos INDEX, y escogeremos la columna `navio_idnavio [int(11)]`. Repetiremos la operación para la columna `capitanes_idcapitanes`, a la cual le crearemos el índice `fk_capitanes_de_navio_capitanes1_idx`.

El siguiente paso consistirá en crear relaciones. phpMyAdmin tiene una opción para hacer estas relaciones de una manera sencilla. Para ello, en la pestaña “Estructura” seleccionaremos la opción “Vista de relaciones” y aparecerá un formulario de restricción de clave foránea. Crearemos dos restricciones, una para la tabla capitanes y otra para navíos:

- La primera restricción tendrá como nombre `fk_capitanes_de_navio_navio1_idx`, en columna escogeremos la `navio_idnavio`, en base de datos `bdd_navios`, en tabla `navio`, en columna `idnavio`. En las opciones ON DELETE y ON UPDATE escogeremos NO ACTION.
- La segunda restricción tendrá como nombre `fk_capitanes_de_navio_capitanes1_idx` en columna escogeremos `capitanes_idcapitanes`, la base de datos será `bdd_navios`, tabla `capitanes` y columna `idcapitanes`. Las opciones serán NO ACTION.



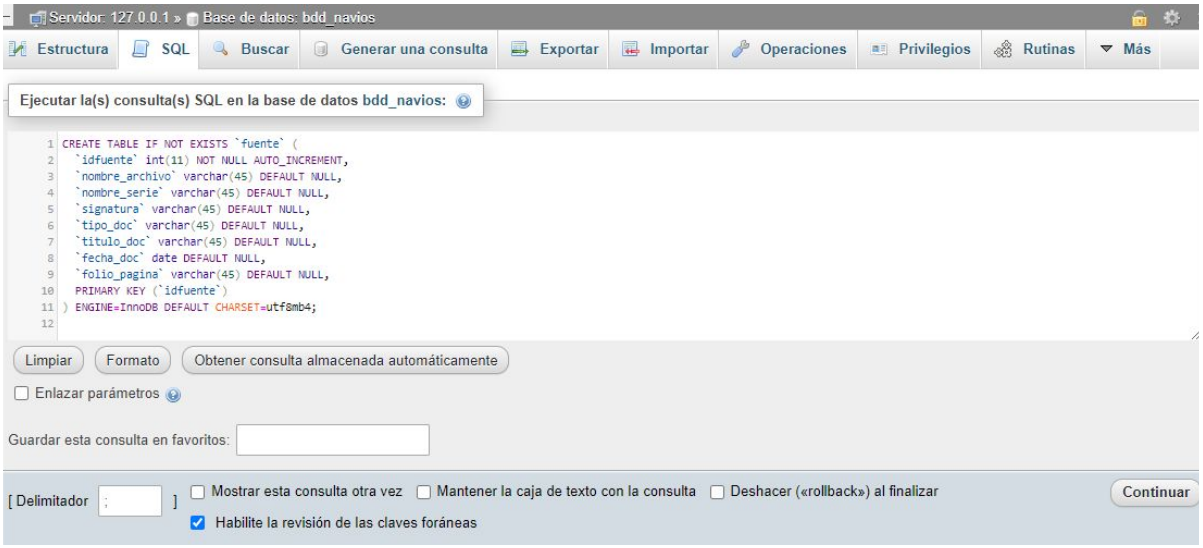
Después de crear la relación, en la estructura de la tabla deberá verse de la siguiente manera:



Para visualizar la relación podemos ir al inicio de la base de datos y en la pestaña “Más” seleccionaremos “Diseñador”. Se desplegará una estructura similar a la siguiente:

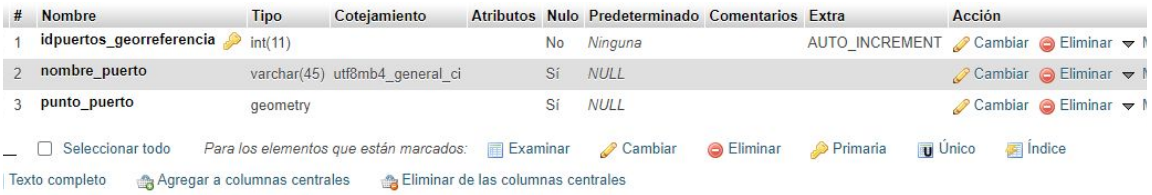
Ahora, crea la tabla `fuente`. Toma como guía el siguiente código o ejecútalo en la consola de phpMyAdmin:

```
CREATE TABLE IF NOT EXISTS `fuente` (
  `idfuente` int(11) NOT NULL AUTO_INCREMENT,
  `nombre_archivo` varchar(45) DEFAULT NULL,
  `nombre_serie` varchar(45) DEFAULT NULL,
  `signatura` varchar(45) DEFAULT NULL,
  `tipo_doc` varchar(45) DEFAULT NULL,
  `titulo_doc` varchar(200) DEFAULT NULL,
  `fecha_doc` date DEFAULT NULL,
  `folio_pagina` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`idfuente`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```



Ahora, vamos a crear la tabla para los datos de los puertos que se llamará `puertos_georreferencia`. Esta incluirá un campo para el atributo geográfico. La clave primaria se llamará `idpuertos_georreferencia`. Crearemos un campo para el nombre del puerto que se llamará `nombre_puerto`, será del tipo VARCHAR y 45 de longitud. Para la

variable geográfica utilizaremos el nombre `punto_puerto` y utilizaremos el tipo espacial GEOMETRY.



#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	idpuertos_georreferencia	int(11)			No	*Ninguna*		AUTO_INCREMENT	Cambiar Eliminar
2	nombre_puerto	varchar(45)	utf8mb4_general_ci		Sí	*NULL*			Cambiar Eliminar
3	punto_puerto	geometry			Sí	*NULL*			Cambiar Eliminar

La última tabla de nuestra base de datos la titularemos `viajes`. Esta será la que entrelazará toda la base de datos. Para crearla tendremos que tener incluir las siguientes columnas:

- `idviajes`, INT, 11, AUTO_INCREMENT
- `id_puerto_salida`, INT, 11
- `fecha_salida`, DATE, predeterminado NULL
- `id_puerto_arribo`, INT, 11
- `fecha_arribo`, DARE, predeterminado NULL
- `direccion_viaje`, VARCHAR, 45, predeterminado NULL
- `navioid`, INT, 11
- `capitanesid`, INT, 11
- `fuenteid`, INT, 11

Verás que hay varios tipos INT, estos serán los que relacionarán nuestros datos con los puertos, los navíos, los capitanes y las fuentes. Todas esas columnas deberán ser asignadas como claves primarias.

También deberemos asociar cada una de esas columnas (con excepción de `idviajes`) con un índice. Los asociaremos de la siguiente manera:

Nombre del índice	Opción de índice	Columna
idnavio_idx	INDEX	navioid
idcapitanes_idx	INDEX	capitanesid
idfuente_idx	INDEX	fuenteid
idsalida_idx	INDEX	id_puerto_salida
idllegada_idx	INDEX	id_puerto_arribo

Crea la tabla con la información proveída o con el siguiente código. Antes de copiarlo y pegarlo, mira cómo está escrito y de qué manera podrías replicarlo en otras tablas que quisieras crear.

```
DROP TABLE IF EXISTS `viajes`;
```

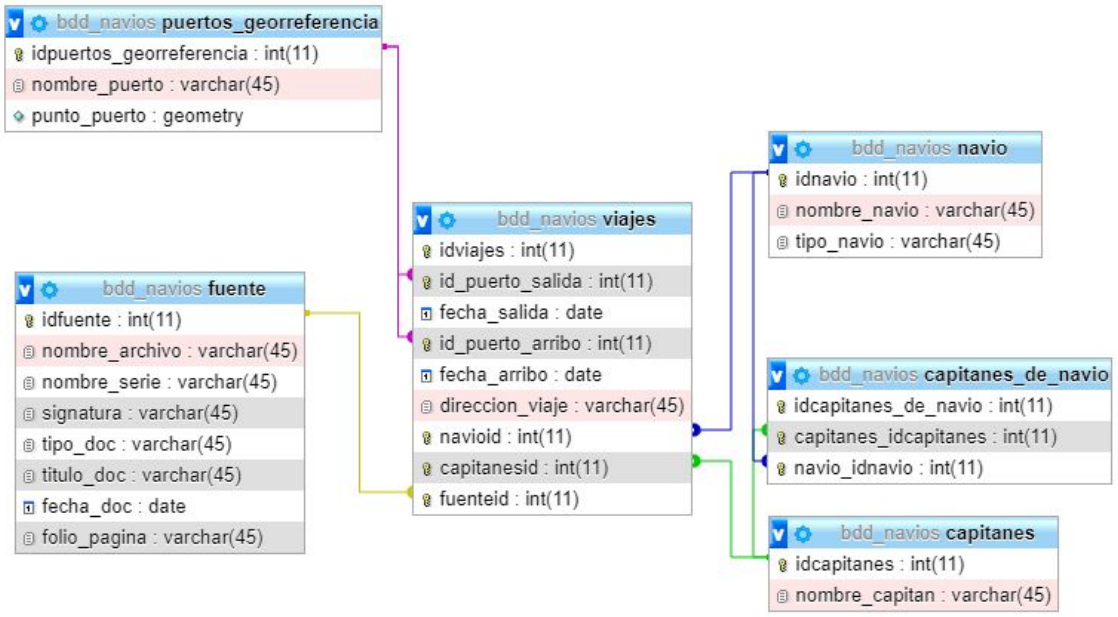
```
CREATE TABLE IF NOT EXISTS `viajes` (
  `idviajes` int(11) NOT NULL AUTO_INCREMENT,
  `id_puerto_salida` int(11) NOT NULL,
  `fecha_salida` date DEFAULT NULL,
  `id_puerto_arribo` int(11) NOT NULL,
  `fecha_arribo` date DEFAULT NULL,
  `direccion_viaje` varchar(45) DEFAULT NULL,
  `navioid` int(11) NOT NULL,
  `capitanesid` int(11) NOT NULL,
  `fuenteid` int(11) NOT NULL,
  PRIMARY KEY
(`idviajes`,`navioid`,`capitanesid`,`fuenteid`,`id_puerto_salida`,`id_pu
erto_arribo`),
  KEY `idnavio_idx` (`navioid`),
  KEY `idcapitanes_idx` (`capitanesid`),
  KEY `idfuente_idx` (`fuenteid`),
  KEY `idsalida_idx` (`id_puerto_salida`),
  KEY `idllegada_idx` (`id_puerto_arribo`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Después de creada la tabla, podemos crear relaciones. Revisa las redes que hicimos previamente y diseña los enlaces entre las tablas siguiendo el mismo modelo que utilizamos para `capitanes_de_navio`. Para cada uno de los índices debe existir una relación, así que trata de identificar correctamente con qué tabla se vincula cada uno de ellos. Presta atención principalmente a la relación entre `id_puerto_salida` y `id_puerto_arribo` y la tabla puertos.

Si te enfrentas a muchos problemas para realizar esta tarea desde los formularios, puedes correr el siguiente código desde la consola de phpMyAdmin:

```
ALTER TABLE `viajes`
  ADD CONSTRAINT `idcapitanes` FOREIGN KEY (`capitanesid`) REFERENCES
`capitanes` (`idcapitanes`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  ADD CONSTRAINT `idfuente` FOREIGN KEY (`fuenteid`) REFERENCES `fuente`
(`idfuente`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  ADD CONSTRAINT `idllegada` FOREIGN KEY (`id_puerto_arribo`) REFERENCES
`puertos_georreferencia` (`idpuertos_georreferencia`) ON DELETE CASCADE
ON UPDATE CASCADE,
  ADD CONSTRAINT `idnavio` FOREIGN KEY (`navioid`) REFERENCES `navio`
(`idnavio`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  ADD CONSTRAINT `idsalida` FOREIGN KEY (`id_puerto_salida`) REFERENCES
`puertos_georreferencia` (`idpuertos_georreferencia`) ON DELETE CASCADE
ON UPDATE CASCADE;
```

Para comprobar que todo funcione correctamente, ve al “Diseñador” para visualizar la red, que debe verse similar a la siguiente:



Si todas las tablas están relacionadas significa que la base de datos ha quedado correctamente estructurada. A partir de este momento, podemos empezar a guardar información.

Creación de bases de datos desde un archivo sql

Obviamente, los desarrolladores y diseñadores de bases de datos no crean las bases de datos de la manera como lo hicimos. Esto les llevaría muchísimo tiempo, sobre todo en grandes estructuras con muchas tablas y relaciones. Lo que se hace comúnmente consiste en crear las bases de datos a partir de una serie de instrucciones que se escriben en un archivo `.sql`. No vamos a profundizar sobre este tema porque nos llevaría por otros caminos.

Si así lo deseas, puedes revisar el archivo `bd_viajes.sql` que se encuentra en el siguiente enlace: https://github.com/jairomelo/sesion_bd_UNIANDES/blob/master/assets/bd_viajes.sql Trata de interpretar cómo está estructurado el programa, qué instrucciones vienen primero, cuales después y cómo se cierra el programa. Nota que el programa no lo escribí, sino fué generado automáticamente a través de la pestaña “Exportar” de phpMyAdmin. Puedes probar creando bases de datos y exportar el código de esa manera. También puede crear una base de datos desde la pantalla de inicio de phpMyAdmin a través del botón “Importar”. Nuevamente, prueba con el código, modifica secciones y encuentra cuáles generan mensajes de error y trata de interpretarlos.

Lo ideal es que comprendas que las tareas automatizadas que parecen “arte de magia” responden a instrucciones precisas hechas por una o varias personas. Los sistemas tienen

varias estrategias para filtrar errores, pero esto no impide que se cometan errores involuntarios o deliberados que puedan resultar en información incorrecta. Por esta razón, no te sientas satisfecho simplemente con que el programa corra, tienes que tratar siempre de encontrar los errores para luego poder regocijarte con los resultados.

Gestión de la bases de datos

Después de haber construido la estructura de nuestra base de datos tenemos que llenarla de contenidos y saber cómo modificarlos, consultarlos y realizar operaciones con ellos. Las posibilidades son amplisimas y dependen en buena medida de los tipos de datos y sus niveles de medida. En este apartado les presentaré algunas operaciones básicas para que puedan iniciar con el análisis de sus datos y escojan algunas rutas posibles para continuar con la exploración de posibles operaciones con estos.

“Poblar” [ingresar datos en] la base de datos

Gracias al dominio lingüístico del inglés en la informática, la acción de ingresar datos en una base o sistema se ha denominado “poblar”, una traducción literal de *populate*[20]. Hago explícita esta “curiosidad” del lenguaje por razones prácticas, puesto que se encontrarán frecuentemente con el término *populate a database*, incluso “poblar” una base de datos, cuando exploren por formas de ingresar datos a una base.

Capturar la información

La diferencia entre “poblar” y “capturar” datos es sutil pero debemos tenerla en mente antes de empezar. “Capturar” datos, que viene del inglés *data capture*, se refiere al proceso manual o automatizado en el que se extrae información de un documento u objeto material, para hacerlo legible por la computadora. Básicamente es el ejercicio que hicimos con la carta del administrador de correos de La Habana al duque de la Alcudia. “Poblar” la base de datos hacer referencia a los procedimientos que hacemos para pasar esos datos capturados a una base de datos.

Por lo general, hacemos una combinación de ambos procesos, como cuando elaboramos una base de datos en LibreOffice base y llenamos filas una por una; también lo podríamos hacer con un formulario que guarde la información capturada inmediatamente a la base de datos, lo cual es un procedimiento regular en la Web. También podemos hacer el ejercicio en dos pasos, capturando la información en un archivo y luego importar los datos a una base. Esta segunda forma es particularmente conveniente cuando necesitamos revisar los datos recopilados, cuando no tenemos la posibilidad de acceder a una aplicación que sirva de intermediador entre los datos y la base, o cuando hacemos una recuperación automatizada de información.

[20] En la lengua española el verbo “poblar” puede tener el sentido de llenar (p. ej., “el lugar estaba poblado de niños”) pero no es un significado común. Pero, para ser justos, en el idioma inglés, *populate* en el sentido de llenar con datos es realmente parte de la jerga informática antes que del lenguaje cotidiano.

Lo importante es que conozcas que hay diferentes caminos para resolver un mismo problema, y que si en ocasiones puede ser útil tener un formulario que permita capturar la información de cada elemento, habrá ocasiones en las que requeriremos incluir grandes cantidades de información y la forma más sencilla consistirá en automatizar los procesos.

Como hacer la captura manual de los diarios de navegación disponibles en el Archivo General de Indias involucra una gran cantidad de tiempo, decidí confiar en el catálogo disponible en PARES y recuperar la información directamente desde los metadatos de cada archivo. La búsqueda la pueden encontrar en la siguiente ruta: http://pares.mcu.es/ParesBusquedas20/catalogo/find?nm=&fraseExacta=Diario+de+navegaci%C3%B3n&archivo=10&digitalizado=S

No obstante, recabar la información abriendo cada registro y copiando los datos en una tabla es todavía una tarea bastante dispendiosa. Es por esta razón que la mejor estrategia consiste en automatizar el proceso mediante un lenguaje de programación. Para este ejercicio utilicé Python. El código se los comparto en el directorio /scripts del repositorio de esta lección y dejo algunas instrucciones para su utilización en el archivo README.md. Básicamente, este programa crea una tabla .csv con ocho columnas:

- Identificador
- Título
- Fecha creación del documento
- Lugar de creación del documento
- Signatura
- Archivo
- Alcance y contenido
- Enlace URL

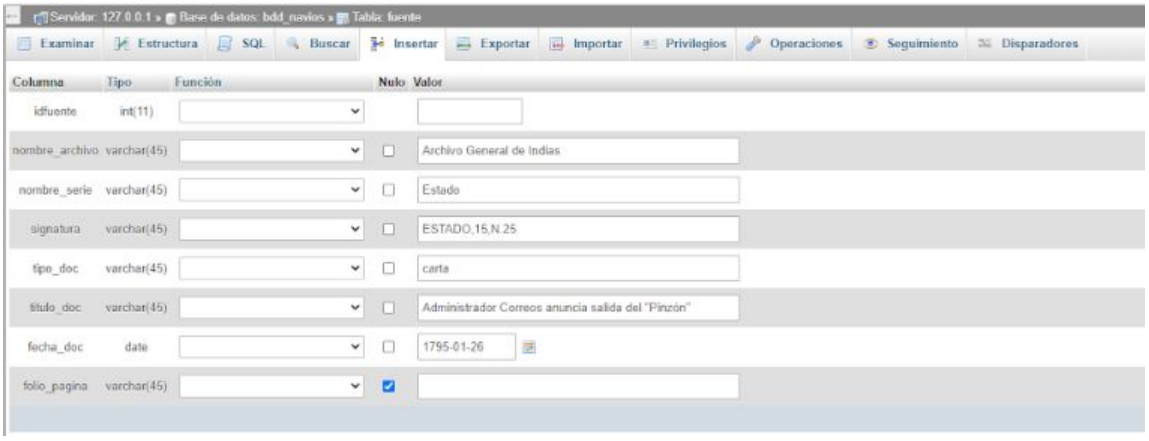
Estos datos nos servirán para ir “poblando” nuestra base. El archivo .csv está disponible en el directorio /data para que lo puedas descargar y consultar. Como verás, esta tabla todavía necesita adaptarse a la estructura de la base de datos, pero nos permitirá construir fácilmente otras tablas y automatizar ciertos procesos.

Ingresar la información

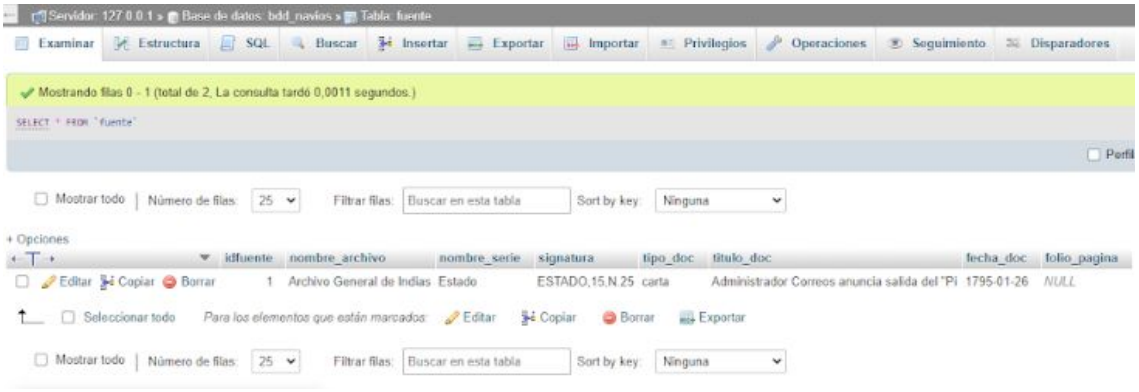
Ahora veremos tres formas de ingresar la información a nuestra base de datos. La primera la haremos directamente desde phpMyAdmin, la siguiente a través de un script SQL y la última desde un archivo csv.

Ingresar desde un formulario phpMyAdmin

Esta es la manera más sencilla de insertar información. Simplemente tendremos que entrar al menú de la tabla que deseemos llenar, por ejemplo `fuente` y hacer clic en la pestaña “Insertar”. En el formulario que se despliega podemos ingresar la información de acuerdo con el tipo de dato que corresponda a la columna.



Después de enviar el formulario verás en la pestaña “Examinar” que el elemento ha sido ingresado exitosamente.



Esta es una manera bastante segura de ingresar información, aunque un poco lenta, ya que tendremos que repetir el proceso con cada tabla.

Ingresar información con SQL

Un problema al que nos vamos a enfrentar continuamente en los formularios de phpMyAdmin serán los errores que se derivan por problemas en la ejecución de ciertas funciones. En términos particulares, he sufrido bastantes inconvenientes para ingresar información geo-espacial desde los formularios de phpMyAdmin[21]. Además, debido a que no es una práctica extendida el ingresar información desde los formularios de esa plataforma, la documentación, ayuda y tutoriales se enfocan en la ejecución de instrucciones en SQL.

[21] Aunque, para ser justos, MariaDB y MySQL no son sistemas de bases de datos utilizados para la gestión de datos espaciales. Programas de gestión de información espacial como QGIS se conectan fácilmente a bases de datos en PostGIS, SpatiaLite y Oracle Spatial; pero tienen muchos problemas para gestionar datos de MySQL. Problemas similares los vamos a encontrar con otros programas de representación de datos espaciales. Mi recomendación es que, en caso de desear trabajar con datos espaciales, lo hagas directamente con cualquiera de los sistemas compatibles.

SQL es un lenguaje de programación bastante sencillo (lo cual puede ser tanto una ventaja como un problema)[22]. La lógica de este lenguaje es simple: básicamente das una instrucción para que se busque, ingrese o modifique la información dentro de una base de datos. Hay cuestiones más complejas de hacer, como condicionales del tipo if-else[23]. Por esta razón, los programadores generalmente recurren a otro lenguaje que sirva para ejecutar estas tareas complejas, siendo uno de los más comunes (amado y odiado) php. Otros lenguajes como JavaScript y Python también pueden servir para este propósito, por lo que sería importante que, si te interesa ahondar en la gestión de bases de datos, los explores.

Vamos a insertar información en la tabla capitanes. Para ello sencillamente debemos dar la instrucción INSERT INTO el nombre de nuestra tabla y los valores (VALUES) que queremos ingresar. Esta tabla es bastante simple pues tiene dos campos y uno de ellos es automático. Ingresemos entonces la información del capitán Juan Bautista de Aránaga con el siguiente código en la casilla de texto que se encuentra en la pestaña “SQL” de phpMyAdmin:

```
INSERT INTO `capitanes`(`idcapitanes`, `nombre_capitan`)
VALUES(NULL, 'Juan Bautista de Aránaga')
```

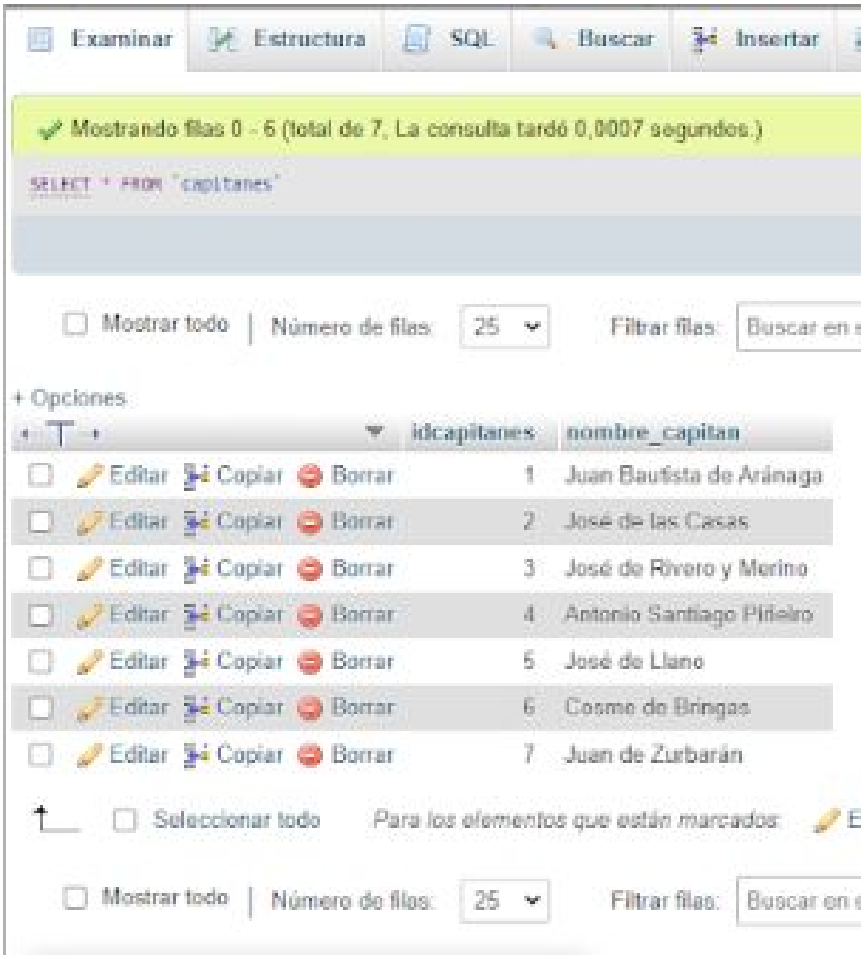
Al ejecutar esta consulta se almacenará inmediatamente la información en las columnas respectivas.

Como podrás notar, este proceso también será sumamente tedioso si se tiene que repetir una y otra vez. No obstante, podemos incluir varios elementos en una misma consulta de la siguiente manera:

```
INSERT INTO `capitanes`(`idcapitanes`, `nombre_capitan`)
VALUES(NULL, "José de las Casas"),
(NULL, "José de Rivero y Merino"),
(NULL, "Antonio Santiago Piñeiro"),
(NULL, "José de Llano"),
(NULL, "Cosme de Bringas"),
(NULL, "Juan de Zurbarán")
```

Después de ejecutar ambas consultas tendremos un pequeño listado de capitanes:

[22] Existen diversos tutoriales y cursos para aprender SQL. Un tutorial básico es ofrecido por W3Schools SQL Tutorial. Un curso un poco más completo es desarrollado por Codecademy Learn SQL.
[23] Expresiones condicionales en SQL estándar | BigQuery



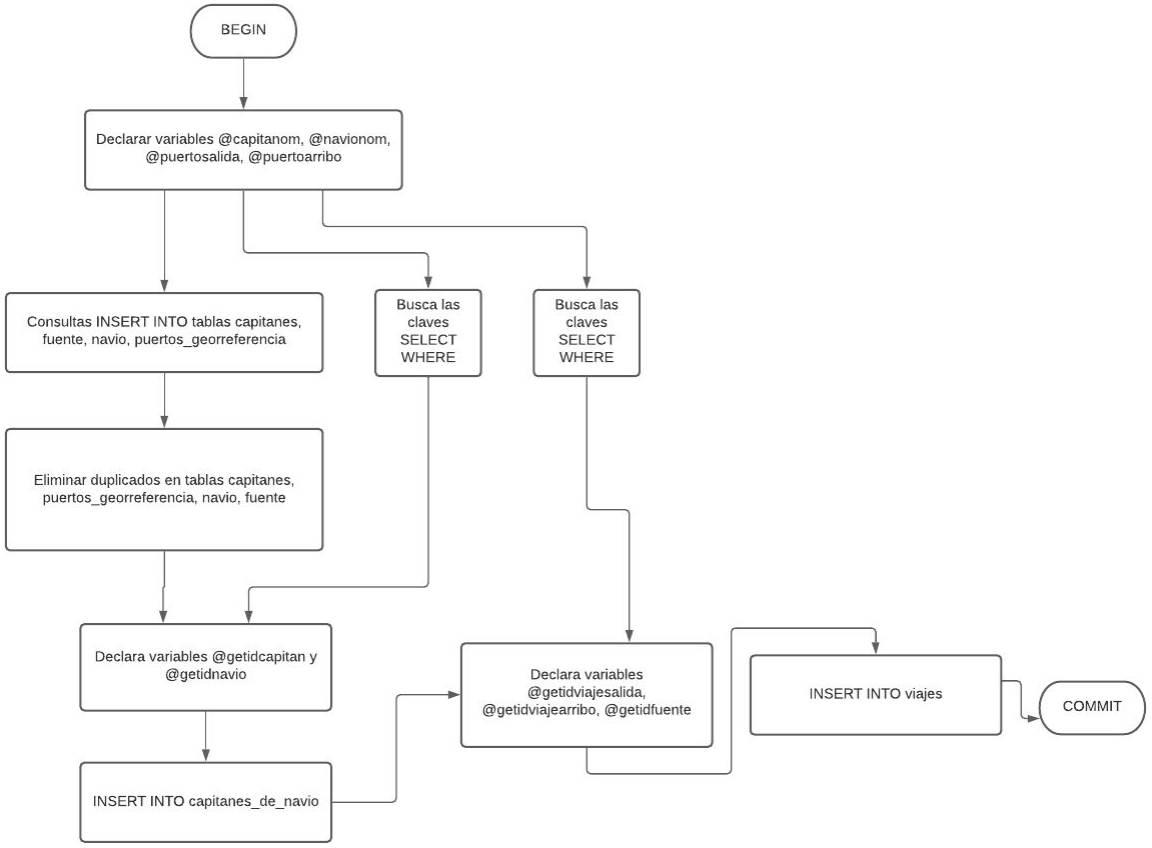
Otra posibilidad que brinda un código SQL consiste en insertar información en varias tablas al mismo tiempo. Para ello necesitaremos utilizar una instrucción que le indique a SQL donde inicia el proceso y donde termina. Vamos a ingresar en una sola consulta la información de un elemento completo para que veas cómo funciona. Este código es un poco más complicado que el anterior pues tenemos que tener en cuenta algunas cosas como:

- Las relaciones en las tablas `puertos_georreferencia` y `viajes` se hacen con claves nuevas, que desconocemos al momento de ingresar la información.
- Es necesario eliminar posibles duplicados para evitar confusiones con las claves únicas asociadas.

El archivo del programa lo podrás encontrar en nuestro repositorio, en el directorio /assets con el nombre poblar_bdd_navios.sql. Voy a explicar escuetamente en qué consiste:

En primer lugar, antes de ejecutar una secuencia de acciones debemos indicarlo a SQL, para ello usamos dos *estamentos*: BEGIN y COMMIT. Con el primero estamos indicándole al compilador de código que vienen a continuación una serie de consultas, cada una separada con un punto y coma. Lo que hace el compilador consiste en revisar la sintaxis, simular una consulta y, si no hay errores, ejecuta las consultas secuencialmente.

Esa secuencialidad es un algoritmo bastante sencillo que paso por paso llena la información de las tablas, ejecuta consultas y almacena variables. Una representación sintética de este algoritmo es la siguiente:



Como ves, todo es bastante jerárquico. El paso da paso al siguiente y así sucesivamente. Las variables son todo lo que está declarado con el estamento SET y se representan con una @ al inicio del nombre. Estos objetos permiten guardar y reutilizar la información. Así, en lugar de escribir en diferentes partes del código el nombre del capitán, del navío o de los puertos (con el riesgo de escribirlo mal en alguna parte), guardamos esta información en cuatro variables:

```
SET @capitanom = "Manuel Antonio de la Villa" ;
SET @navionom = "El Colón" ;
SET @puertosalida = "Cádiz" ;
SET @puertoarribo = "La Habana" ;
```

De esta manera, podemos reutilizar la información sin preocuparnos en otras partes del código, como en el ingreso de información en las tablas capitanes, fuente, navío y puertos:

```
-- Insertar información en tabla capitanes
INSERT INTO `capitanes` (`idcapitanes`, `nombre_capitan`)
VALUES(NULL, @capitanom) ;
-- Insertar información fuente
```

```
INSERT INTO `fuente` (`idfuente`, `nombre_archivo`, `nombre_serie`,
`signatura`, `tipo_doc`, `titulo_doc`, `fecha_doc`, `folio_pagina`)
VALUES(NULL, "Archivo General de Indias", "Correos",
"CORREOS,275A,R.10", "Diario de navegación", "Diario de navegación del
paquebot \"El Colón\"", "1770-09-28", NULL) ;
-- Insertar información navío
INSERT INTO `navio` (`idnavio`, `nombre_navio`, `tipo_navio`)
VALUES(NULL, @navionom, "paquebot") ;
-- Insertar puertos
INSERT INTO `puertos_georreferencia` (`idpuertos_georreferencia`,
`nombre_puerto`, `punto_puerto`)
VALUES(NULL, @puertoarribo, ST_PointFromText('POINT(23.137085
-82.347309', 4326)),(NULL, @puertosalida,
ST_PointFromText('POINT(36.542729 -6.282635', 4326)) ;
```

Dos cosas a señalar aquí. En primer lugar, la información que va entre comillas (como el nombre del paquebot en el título) deberá señalarse con un caracter de escape, que para SQL es la barra invertida: `\"El Colón\"`. En segundo lugar, nótese que la información relativa al punto geográfico se ingresa mediante la función `ST_PointFromText()`. Con ella, se convierten las coordenadas geográficas señaladas en texto en un punto binario que puede ser interpretado por MySQL. El último atributo (el número 4326) señala el código SRID, que da cuenta de la geometría y coordenadas de un punto, línea o polígono sobre una proyección espacial[24].

Después de llenar la información de estas tablas, debemos asegurarnos de eliminar los duplicados en las tablas capitanes, navíos, puertos_georreferencia y fuente. Esto debido a que requerimos las claves únicas de los elementos de esas tablas, algunos de los cuales acabamos de ingresar y podría estar repetido. Para ello ejecutamos el siguiente código:

```
-- Eliminar duplicados para conservar ID única en relaciones
-- Eliminar duplicados de capitanes
DELETE t1 FROM `capitanes` t1
INNER JOIN `capitanes` t2
WHERE
    t1.`idcapitanes` > t2.`idcapitanes` AND
    t1.`nombre_capitan` = t2.`nombre_capitan` ;
-- Eliminar duplicados de navíos
DELETE t3 FROM `navio` t3
INNER JOIN `navio` t4
WHERE
    t3.`idnavio` > t4.`idnavio` AND
    t3.`nombre_navio` = t4.`nombre_navio` ;
-- Eliminar duplicados de puertos
```

[24] ¿Qué es un SRID?—Ayuda | ArcGIS Desktop

```
DELETE t5 FROM `puertos_georreferencia` t5
INNER JOIN `puertos_georreferencia` t6
WHERE
    t5.`idpuertos_georreferencia` > t6.`idpuertos_georreferencia` AND
    t5.`nombre_puerto` = t6.`nombre_puerto` ;
-- Eliminar duplicados de fuente
DELETE t7 FROM `fuente` t7
INNER JOIN `fuente` t8
WHERE
    t7.`idfuente` > t8.`idfuente` AND
    t7.`signatura` = t8.`signatura` ;
```

Básicamente le estamos diciendo a SQL que de la tabla señalada elimine las filas en las que la clave sea mayor que aquella donde coincida una palabra clave (p. ej., la signatura o el nombre del capitán). Esta es una solución simple, pero habría que tener cuidado en caso de homónimos.

Para llenar la tabla capitanes_de_navio necesitaremos la clave del nombre del capitán y del nombre del navío, para ello, guardaremos una consulta en las variables @getidcapitan y @getidnavio, que coincida con la búsqueda exacta del nombre del capitán y del navío.

```
SET @getidcapitan = (SELECT `idcapitanes` FROM `capitanes` WHERE
`nombre_capitan` LIKE @capitanom) ;
SET @getidnavio = (SELECT `idnavio` FROM `navio` WHERE `nombre_navio`
LIKE @navionom) ;
```

Con estas variables ya podemos llenar la tabla capitanes_de_navio

```
INSERT INTO `capitanes_de_navio` (`idcapitanes_de_navio`,
`capitanes_idcapitanes`, `navio_idnavio`)
VALUES(NULL, @getidcapitan, @getidnavio) ;
```

La tabla viajes es la última pues es la que involucra más relaciones entre tablas. Igualmente, necesitamos que las claves que existen y las que ahora se hayan creado sean leídas por el programa, pues de otra manera tendríamos que adivinar cuál sería la clave que le correspondería a cada elemento, o averiguar exactamente cuál tiene ya asignada en la tabla. Entre menores sean los procesos manuales que tengamos que hacer, mejores resultados vamos a obtener. Las variables las asignaremos siguiendo el siguiente código:

```
SET @getidviajesalida = (SELECT `idpuertos_georreferencia` FROM
`puertos_georreferencia` WHERE `nombre_puerto` LIKE @puertosalida) ;
SET @getidviajearribo = (SELECT `idpuertos_georreferencia` FROM
`puertos_georreferencia` WHERE `nombre_puerto` LIKE @puertoarribo) ;
```

```
SET @getidfuente = (SELECT `idfuente` FROM `fuente` WHERE `signatura`
LIKE "CORREOS,275A,R.10") ;
```

Terminado este paso, lo único que faltaría sería completar la tabla viajes:

```
INSERT INTO `viajes`(`idviajes`, `id_puerto_salida`, `fecha_salida`,
`id_puerto_arribo`, `fecha_arribo`, `direccion_viaje`, `navioid`,
`capitanesid`, `fuenteid`)
VALUES(NULL, @getidviajesalida, "1770-09-28", @getidviajearribo,
"1770-11-28", "viaje", @getidnavio, @getidcapitan, @getidfuente) ;
```

Si copias el código de poblar_bdd_navios.sql, en la pestaña SQL de phpMyAdmin y lo ejecutas, verás como la información queda guardada en cada una de las tablas y no se repite ningún nombre de capitán, navío o puerto.

Habrás notado que el código SQL hace las cosas más sencillas que el formulario de phpMyAdmin. Obviamente, un formulario en un lenguaje de programación como php podría lidiar con todas las variables anteriores de una manera más amigable. Pero por lo pronto basta con que comprendas parte de la lógica de la programación de la base de datos para que luego la puedas aplicar a tus propios experimentos. Por esta razón, intenta hacer una nueva entrada reutilizando el código, cambiando valores e incluso variables. ¿Habrá una manera de hacer la operación más rápida y sencilla? Seguramente, pero eso depende de la capacidad de experimentación de cada uno.

Ingresar información desde un archivo csv

Obviamente, cualquiera de las dos estrategias anteriores es bastante inadecuada para incluir información en grandes cantidades. ¿Recuerdas la información que capturamos? Sería bastante tedioso incluirla una por una siguiendo cualquiera de los dos métodos. Lo que haremos consistirá en incluir la información de nuestro csv en la base de datos que acabamos de crear para “poblarla” con muchos más datos.

En el directorio /data de nuestro repositorio, dejé tres archivos csv en los que adapté la información recuperada automáticamente de PARES para que pueda ser incorporada automáticamente a las tablas capitanes, fuente y navio. Puedes abrir los archivos desde una hoja de cálculo y ver la manera en que están estructurados. La recomendación es que los archivos csv no tengan encabezados, ni adaptaciones de formato como celdas combinadas. Si abres el archivo desde Microsoft Excel es muy probable que no puedas ver los caracteres especiales (tildes, diéresis y virgulillas). No guardes el archivo desde este programa porque modificarás la codificación y tendrás problemas para importarlo a MySQL. Es recomendable usar un programa como LibreOffice calc para crear y manipular archivos csv, teniendo cuidado de abrirlos y guardarlos con una codificación utf-8.

Para importar la información desde un archivo csv solamente deberás entrar al menú de la tabla en phpMyAdmin y escoger la pestaña “Importar”. En esa pantalla deberás seleccionar el archivo. La configuración posterior se adaptará a las características del archivo csv, por lo

que solamente será necesario Continuar. Dependiendo de la velocidad de tu computadora y del tamaño del archivo, el proceso de importación puede tardar incluso unos minutos. Si todo funcionó correctamente, se mostrará una pantalla donde se confirmarán la importación exitosa de los datos. Podrás ver las filas importadas en la pestaña Examinar.

Intenta replicar el ejercicio con las tablas navio y fuente. De esa manera, tendremos una base de datos medianamente poblada.

Finalmente, habrás notado que esta es una forma ágil de incorporar muchos datos a una base. Solamente tendrás que tener cuidado con la codificación del archivo csv y con formar las columnas correctamente para que correspondan con la estructura de la respectiva tabla.

Operaciones con bases de datos

Después de “poblar” la base de datos es importante tener la capacidad de hacer algunas operaciones con ellos. Vamos a revisar algunas operaciones básicas siguiendo una serie de problemas.

Primero, vamos a modificar la estructura de una tabla que ya contiene datos. En la tabla fuente quiero agregar dos columnas: contenido y URL. Para eso utilizaremos el estamento ALTER TABLE. La sintaxis para hacer esta modificación es bastante sencilla, solamente requerimos ejecutar la siguiente consulta:

```
BEGIN;
```

```
ALTER TABLE `fuente`
ADD `contenido` VARCHAR(500);

ALTER TABLE `fuente`
ADD `URL` VARCHAR(250);

COMMIT;
```

Verás que la sintaxis es bastante simple. Solamente requerimos declarar el nombre de la tabla que queremos modificar e indicar, mediante el estamento ADD, qué nombre tendrá la nueva columna y cual será el tipo de dato que contendrá y su longitud. Como el contenido puede ser bastante amplio será prudente establecer una longitud de 500 y para la URL de 250.

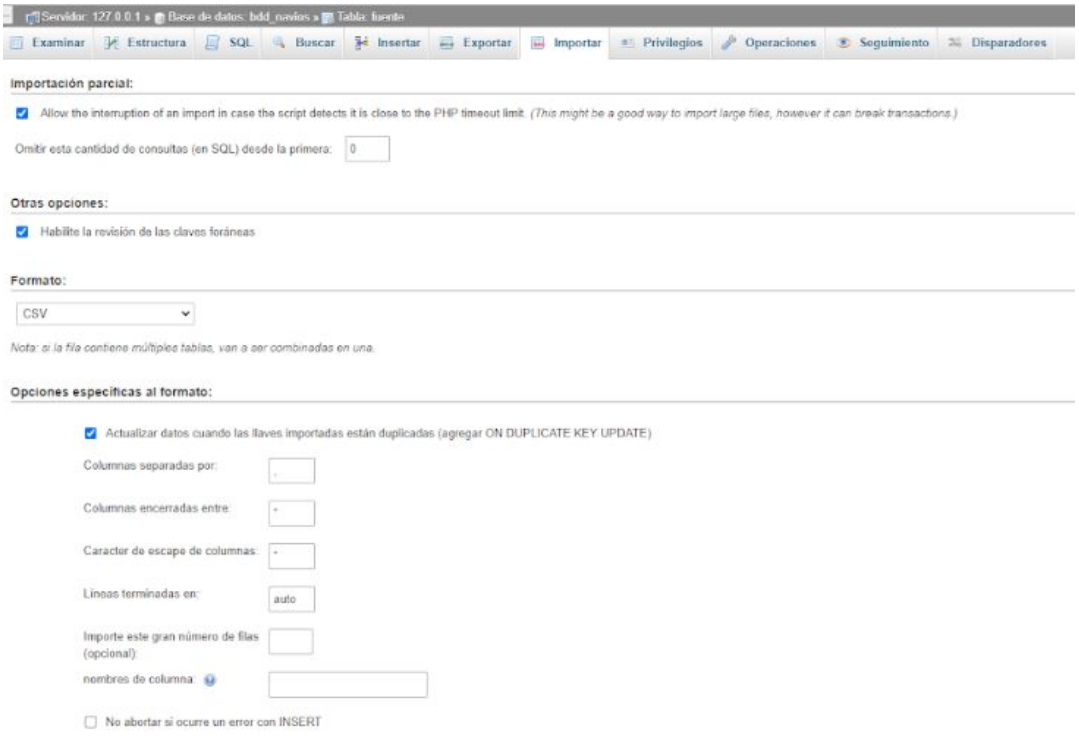
Después de crear las dos columnas será necesario llenarlas de contenido. Para ello tendremos que utilizar el estamento UPDATE. Con este podremos incluir la información requerida únicamente en la columna que necesitamos:

```
BEGIN;

UPDATE `fuente` SET `contenido` = 'Carta nº 364 del Administrador de
Correos D. José Fuertes al Duque de la Alcudia; anuncia la salida del
\'Pinzón\', que se ha dilatado por el mal tiempo, y da noticia de los
útiles auxilios que para esta expedición ha debido al General de Marina
D. Juan de Araoz. Duplicado. Con el índice de remisión común con la nº
363. [V. Estado, 15-24]. 2 hoj. fol' WHERE `fuente`.`idfuente` = 1;

UPDATE `fuente` SET `URL` =
'http://pares.mcu.es/ParesBusquedas20/catalogo/show/62474?nm' WHERE
`fuente`.`idfuente` = 1;

COMMIT;
```

Obviamente, querremos modificar todos los elementos y no hacerlo individualmente. Para esto, requeriremos recurrir a un método que se denomina INSERT… ON DUPLICATE KEY UPDATE, con el cual no tendremos que preocuparnos de modificar la clave principal de la tabla. Afortunadamente, phpMyAdmin incluye esta opción en el menú de importación de archivos csv. La mejor manera de hacer esta modificación consiste en guardar la tabla fuente en un archivo csv desde la pestaña Exportar de phpMyAdmin. Este archivo lo podremos editar con la información adicional. Para facilitar este ejercicio, incluí una versión modificada de la tabla en el directorio /data con el nombre fuente_altertable.csv.

Finalmente, desde el menú Importar seleccionamos el archivo modificado, y nos aseguramos de seleccionar la opción “Actualizar datos cuando las llaves importadas están

duplicadas (agregar ON DUPLICATE KEY UPDATE)". Con este simple procedimiento nuestra tabla estará actualizada sin que se modifiquen los valores originales.



Si realizaste el ejercicio con el archivo fuente_altertable.csv notarán que en la tabla `fuente` hay una fila sin contenido ni url. Esto se debe a que es un elemento duplicado (elemento 16). Vamos a eliminar esta fila de manera segura. Recordemos que la fila 2 está vinculada con la tabla viajes. Entonces, debemos primero actualizar esta tabla para que remita al ítem 16 y posteriormente eliminar la fila 2.

```
BEGIN;

UPDATE `viajes` SET `fuenteid` = 16 WHERE `idviajes` = 1;

DELETE FROM `fuente` WHERE `fuente`.`idfuente` = 2 ;

COMMIT;
```

Con este simple código, hemos eliminado la fila duplicada sin perder el enlace en la tabla viajes.

Los procedimientos y operaciones que se han mostrado en este apartado pueden ser utilizadas en muchos otros aspectos de la gestión de una base de datos. Con estos conceptos básicos y algo de práctica, sin duda podrás construir tu propia base de datos de investigación con muchísima más información relevante para la pesquisa histórica.

¿Es necesario hacer bases de datos para toda investigación histórica?

Como lo expresé en los primeros párrafos de este texto, las investigaciones históricas se han llevado a cabo exitosamente durante muchas décadas siguiendo un método anclado en lo "analógico". Entre 1960 y 1990 hubo una tendencia bastante fuerte que privilegiaba las investigaciones cuantitativas sobre las cualitativas. Hacer historia económica y demográfica se consideraba una tarea imprescindible para el conocimiento del pasado, no porque fuese una "moda",[25] sino porque respondía a un paradigma de la época que consideraba que una de las formas más apropiadas de acercarse a la evolución humana era a través del estudio de sus estructuras. La economía y la demografía se fundamentan en la mensurabilidad de los hallazgos, es posible medir, promediar, proyectar, identificar tendencias y patrones. Por esa razón, los métodos cuantitativos tuvieron un particular auge durante esas décadas. Obviamente hubo ciertos excesos, como la consideración de que los métodos cuantitativos eran "mejores" que los cualitativos, un prejuicio que se le ha endilgado a la ahora difamada escuela cliométrica.

El incremento de la digitalización, la necesidad de recurrir a técnicas para el manejo de datos masivos y la cada vez mayor facilidad para aprender ciertos lenguajes de programación como Python y R; son algunos de los factores que hacen temer que la historia digital conlleve un renacer de la cliometría.[26] No obstante, la historia digital no enfrenta tanto un reto de cuantificación como de "datificación",[27] es decir, de construir hipótesis no a partir de la observación o lectura de fuentes sino del análisis de datos. En su forma más avanzada, la historia digital sería el puente hacia un nuevo paradigma científico en el que la investigación no está fundamentada en la observación y elaboración de hipótesis, sino en el impulso de los datos.[28]

El resultado cualitativo de una investigación histórica puede depender de los métodos con los que se aborden las fuentes históricas, no hay duda de ello; pero elaborar o no una base de datos no es garantía ni de éxito ni de fracaso en el abordaje del pasado. No obstante, tampoco es una labor superflua. Los historiadores deberíamos tener las habilidades necesarias para hacer un análisis crítico de una base de datos, algo que todavía se deja al arbitrio del estudiante o investigador. Esperamos que en el futuro cercano en las propuestas

---

[25] Véase la objeción de Rugiero Romano al respecto en "Historia cuantitativa, historia económica e historia", párr. 3, https://books.openedition.org/cemca/634?lang=en

[26] C. Annemieke Romein et al., "State of the Field: Digital History", *History* 105, núm. 365 (2020): 292, https://doi.org/10.1111/1468-229X.12969.

[27] M. Lynne Markus, "Datification, Organizational Strategy, and IS Research: What's the Score?", *The Journal of Strategic Information Systems* 26, núm. 3 (2017): 233–41, https://doi.org/10.1016/j.jsis.2017.08.003.

[28] Frédéric Clavert, "Une histoire par les données ? Le futur très proche de l'histoire des relations internationales", *Bulletin de l'Institut Pierre Renouvin* N° 44, núm. 2 (2016): 119–30; Serge ter Braake et al., "Digital History: Towards New Methodologies", en *Computational History and Data-Driven Humanities*, ed. Bojan Bozic et al., IFIP Advances in Information and Communication Technology (Cham: Springer International Publishing, 2016), 23–32, https://doi.org/10.1007/978-3-319-46224-0_3.

de investigación se evite apuntar vagamente “elaborar una base de datos” y en su lugar se presenten modelos y posibles estrategias para el análisis de datos, las cuales sirvan para decidir en cada evaluación. Igualmente, incluir en la descripción de la metodología el uso de tal o cual programa o aplicación es irrelevante. Se supone que diferentes aplicaciones deben brindar soluciones equivalentes y el historiador debería saber exactamente el por qué de sus resultados. Si los resultados se aceptan acríticamente solamente porque se desconoce cómo el programa está computando los datos la culpa se debe achacar completamente al usuario.

No es necesario hacer una base de datos y de ninguna manera debería ser una exigencia académica, pero tampoco está prohibido hacerlo, ni afecta el trabajo de investigación. Si se opta por usar una base de datos se debe establecer previamente el para qué de su elaboración y el qué tareas va a facilitar. La base de datos que elaboramos para este ejercicio fue realizada en el transcurso de unas horas (incluyendo el código con el que se creó el programa para descargar los elementos básicos), los resultados no son “sorprendentes”, pero pueden ser útiles precisamente para incluir esos aspectos en principio “superfluos” que pueden conllevar mucho tiempo de trabajo que mejor podría dedicarse al análisis de otros textos y documentos.