Bases de datos relacionales y lenguaje SQL

Introducción

Introducción

Objetivos y contenidos

Objetivos

- Utilizar el modelo entidad-relación para representar el modelo conceptual de datos.
- Conocer los conceptos fundamentales del modelo relacional.
- Transformar el modelo conceptual al modelo relacional.
- Obtener el script SQL del modelo relacional.
- Diseñar consultas en lenguaje SQL mediante el lenguaje de manipulación de datos (DML).

Contenidos

- Introducción a las bases de datos.
- Diagrama Entidad-Relación.
- El modelo relacional: conceptos importantes.
- Transformación del diagrama Entidad-Relación al modelo relacional.
- Obtención del script SQL del modelo relacional.
- Consultas SQL básicas.

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

Introducción

Herramientas para trabajar con bases de datos relacionales

Los sistemas gestores de bases de datos más importantes son MySQL y Oracle. No obstante, estos sistemas requieren de instalación en el sistema operativo, por lo que se ha optado por utilizar herramientas online para el desarrollo de todo el curso y así evitar problemas de instalación en diferentes entornos.

A continuación se van a nombrar las herramientas que se pueden utilizar para realizar las actividades de todo el curso. Puede que algunos términos no los entendáis o no sepáis que son ahora mismo, pero a lo largo de todo el curso se explica todo más detalle.

Diagrama Entidad-Relación

Existen muchas herramientas online que permiten crear diagramas de todo tipo, pero la que más se acerca a la notación que se utiliza en este curso es la herramienta **Dia**.

Dia es una aplicación que hay que instalar en el sistema operativo, se puede hacer desde aquí, pero desde la web rollApp se puede utilizar la aplicación de manera online sin necesidad de instalarla en el ordenador. Simplemente hay que registrarse en la web y una vez iniciada la sesión, ofrece multitud de aplicaciones para utilizar de manera online, lo que podemos llamar un escritorio virtual.

Modelo relacional

Para la creación del modelo relacional se ha elegido la aplicación online **ERD Plus**, que también permite realizar diagramas Entidad-Relación, pero algo más complicado que Dia.

Para utilizar ERD Plus es necesario también registrarse, y una vez hecho el registro su uso es muy intuitivo.

Lenguaje SQL

A la hora de ejecutar sentencias SQL, es necesario disponer de un sistema gestor de base de datos instalado en el sistema operativo. La instalación de un sistema gestor de base de datos no entra

dentro del alcance de este curso, por lo que se hará uso de la web Coding Rooms, que dispone de una interfaz sencilla en la que se escriben las sentencias SQL y nos devuelve el resultado de las mismas de manera sencilla.

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

Introducción

Relación de los contenidos con la LOMLOE

En la nueva normativa educativa publicada el año 2022 se impulsa de forma decisiva la enseñanza de contenidos relacionados con las bases de datos relacionales y las consultas mediante SQL, en Bachillerato.

En concreto los contenidos abordados en este curso corresponden a la Competencia específica nº 4 de la materia Informática desarrollada en la **Orden ECD/1173/2022, de 3 de agosto, por la que se aprueban el currículo y las características de la evaluación del Bachillerato**. En concreto esta competencia consiste en "*Utilizar un software de hoja de cálculo para el manejo sencillo de información, realizar el diseño completo de una base de datos relacional sencilla plasmado en un sistema gestor de bases de datos relacional en entorno ofimático, y conocer y comprender la noción de datos masivos, así como las oportunidades y riesgos, tanto sociales como personales, de su tratamiento*"

Específicamente en los criterios de evaluación de dicha competencia, para el curso de 1º de Bachillerato, se habla de:

- *4.2. Utilizar el diagrama entidad-interrelación para representar el modelo conceptual de datos de una situación sencilla del mundo real descrita en lenguaje natural.*
- *4.3. Conocer los conceptos fundamentales del modelo de datos relacional.*
- *4.4. Transformar el modelo conceptual de datos a un modelo de datos relacional.*
- *4.5. Utilizar un sistema gestor de bases de datos relacionales en entorno ofimático para implementar el modelo relacional obtenido, incluyendo la creación de formularios, informes y consultas.*
- *4.6. Diseñar consultas en lenguaje SQL para la manipulación de datos.*

Asimismo el currículo de dicha materia establece como uno de los bloques de saberes básicos de esta materia aquellos concernientes a **Datos**, estableciendo como conocimientos, destrezas y actitudes a desarrollar con el alumnado en 1º de Bachillerato las siguientes:

- Introducción a los modelos de datos: del modelo entidad-interrelación al modelo relacional.
- Conceptos básicos del modelo de datos relacional: relación, atributo, tupla, clave primaria y clave ajena.

- Sistemas Gestores de Bases de Datos Relacionales: definición de tablas, relaciones entre tablas, consultas.
- Lenguaje SQL como lenguaje de manipulación de datos.

Por lo tanto, tanto los ejercicios planteados en el curso así como la metodología encajan perfectamente en la programación de esta materia en 1º de Bachillerato.

Una vez que el alumnado se ha familiarizado con los conceptos del diseño de las bases de datos y del lenguaje SQL, es tiempo de plantearle situaciones de aprendizaje en los que aplicarlos, preferentemente en la resolución de problemas reales y aplicando metodologías de trabajo en equipo.

La Competencia arriba descrita también se encuentra en el Currículo de la materia optativa de 3º de ESO de Programación y Robótica como Competencia Específica nº 4 de esa materia, pudiendo aplicarse todo lo dicho anteriormente también en el desarrollo de dicha materia.

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

Introducción

Herramientas de Bases de Datos para uso en clase

A la hora de dar nuestras clases, puede ser mucho más cómodo e interesante utilizar software instalado en nuestro ordenador, para no tener que depender de las conexiones a Internet y porque para trabajar con los alumnos es más cómodo.

Diseño de Bases de Datos.

Para el diseño de las bases de datos, recomiendo utilizar la misma herramienta que se ha utilizado online, **Dia**, pero en la versión para instalar. Que la podéis encontrar aquí.

Lenguaje SQL

Para trabajar a nivel de Bachiller, que es el público objetivo de este curso, lo más práctico puede ser utilizar un servidor de MySQL que, además, instale un cliente MySQL para poder conectarse a ese servidor, en el que lanzar las consultas SQL.

Con Vitalinux

En el caso de que vuestro centro disponga de Vitalinux, el play tiene un paquete que instala servidor y cliente con un solo click.

Se abre el play, se busca MySQL y se instala:

Instalar paquete MySQL Vitalinux IsardVDI | noVNC Viewer - Mozilla Firefox_122.png

Con esto ya está instalado el servidor y el cliente. El cliente que instala se llama DBeaver y es muy intuitivo.

Una vez instalado el paquete aparecerá un mensaje que da varias opciones, "Arrancar BD", "Eliminar BD" y "Cerrar".

IsardVDI | noVNC Viewer - Mozilla Firefox_123.png

- **Arrancar BD**: arranca el servicio de MySQL para que el cliente pueda conectarse a él.
- **Eliminar BD**: elimina todo lo que se ha hecho en el servidor. Esto es útil si se quiere empezar de 0 de manera rápida sin tener que ir buscando cada base de datos que se ha creado para eliminarla.
- **Cerrar**: simplemente cierra la ventana del mensaje.

Una vez instalado y arrancado, sale un mensaje que informa de que el servidor está activo. Además informa también de la contraseña del usuario root en el servidor MySQL. Esto es muy importante, ya que esa contraseña será la que haya que poner en el cliente DBeaver al crear la conexión con el servidor, así que hay que copiarla para ponerla luego.

IsardVDI | noVNC Viewer - Mozilla Firefox_124.png

Para abrir el cliente instalado, DBeaver, basta con pulsar ctrl+espacio y poner en el buscador "Dbeaver" o ir al menú "Programación > dbeaver-ce". De cualquier manera se abrirá la siguiente aplicación:

IsardVDI | noVNC Viewer - Mozilla Firefox_126.png

Ahora hay que crear la conexión con nuestro servidor.

Se selecciona el botón de "Nueva conexión > MySQL":

IsardVDI | noVNC Viewer - Mozilla Firefox_130.png

Se rellena la ventana de la conexión que se abre con la información que aparece en la imagen siguiente. La contraseña es la que se ha tenido que copiar de la ventana que se ha abierto al instalar el paquete.

IsardVDI | noVNC Viewer - Mozilla Firefox_131.png

Una vez hecho esto ya se puede trabajar. Pinchando en "**localhost**", se abre todo lo que hay en el servidor MySQL a lo que tiene acceso el usuario con el que nos hemos conectado.

Pinchando sobre **Database,** se abren todas las bases de datos que hay creadas. Se puede crear una base de datos haciendo click derecho encima de **Database** y seleccionando "**Crear Nuevo Database**".

IsardVDI | noVNC Viewer - Mozilla Firefox_127.png

Para trabajar con el lenguaje SQL, hay que seleccionar la opción "**SQL**" del menú principal y ahí elegir si se desea crear un nuevo fichero o abrir uno existente. Se abrirá la opción elegida en la parte derecha de la aplicación para poder empezar a trabajar.

IsardVDI | noVNC Viewer - Mozilla Firefox_129.png

Si no se tiene Vitalinux

Una opción fácil es instalar el paquete **XAMPP**, que solo hay que descargarlo e instalarlo y ya tiene el servidor MySQL, Apache, PHP e incluye el cliente phpMyAdmin, que se abre en el navegador, para trabajar con las bases de datos.

Hay que descargar el ejecutable para el Sistema Operativo de nuestro ordenador (disponibles: Windows, Linux y OS X), desde el enlace puesto antes, luego se ejecuta (doble click) y se lanza el instalador. Es posible que en Linux, antes haya que hacer ejecutable el fichero descargado.

La primera ventana es la de Bienvenida:

XAMPP Bienvenida

En la segunda hay que seleccionar lo que se desea instalar. Para utilizar solo MySQL, basta con seleccionar: **Apache**, **MySQL**, **PHP** y **phpMyAdmin**:

XAMPP seleccionar

En la siguiente hay que introducir la ruta en la que se desea instalar XAMPP:

XAMPP Ruta

Con esto ya comienza la instalación:

XAMPP instalando

Una vez terminada se abre el panel de control y se puede seleccionar qué servicios queremos arrancar. Para poder trabajar con el phpMyAdmin es necesario arrancar **Apache** y **MySQL**. Para eso basta con pulsar sobre el botón "Start" de cada uno de ellos y esperar a que arranquen.

XAMPP servicios

Ahora, para acceder al phpMyAdmin, hay que abrir un navegador y escribir en la barra de navegación "localhost". Si se ha instalado correctamente, se abrirá la página de bienvenida de XAMPP.

Instalación y uso de XAMPP en Windows. PHP. Bartolomé Sintes Marco. www.mclibre.org - Mozilla Fi

Seleccionando en el menú principal "phpMyAdmin", se abrirá el cliente para trabajar con MySQL desde el navegador.

Acceder_phpMyAdmin.png

phpMyAdmin tiene la siguiente pinta:

mov-wewordpress-01-ges.catedu.aragon.local - bov-mysqlwordpress-01-ges.catedu.aragon.local | p

A la izquierda están las bases de datos y arriba el menú para trabajar con la base de datos seleccionada.

Para lanzar consultas SQL, hay que seleccionar la pestaña SQL

Con esto ya podéis trabajar en clase con vuestros alumnos todo lo que aprendáis en este curso.

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

Módulo 1. Las bases de datos

Introducción a las bases de datos

Módulo 1. Las bases de datos

Qué es una base de datos

Una base de datos es un sistema informático orientado a los datos que pretende recuperar y almacenar la información de manera eficiente y cómoda. Surge en un intento de resolver las dificultades del procesamiento tradicional de los datos, teniendo en cuenta que los datos suelen ser independientes de las aplicaciones.

Las bases de datos nacen como solución al problema que se plantea a los programadores a la hora de diseñar programas eficientes y capaces de manejar grandes cantidades de información.

Estas bases de datos responden a un modelo. Modelar consiste en definir un mundo abstracto y teórico tal que las conclusiones que se puedan sacar de él coinciden con el comportamiento del mundo real.

Un **modelo de datos** es un conjunto de **conceptos**, **reglas** y **conversiones** que permiten describir los datos del problema que se plantea solucionar y su posterior manipulación.

Los modelos de datos que han contado en la historia con un mayor grado de aceptación son el **jerárquico**, **de red** y **relacional**. Aunque la complejidad de las aplicaciones informáticas ha hecho que las **BD NoSQL** estén ganando cada vez más importancia.

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

Módulo 1. Las bases de datos

Bases de datos relacionales

El modelo relacional nació como consecuencia de los trabajos publicados por Codd en 1970, y su duración en el tiempo es debido en gran medida a que es un modelo bien fundado en bases matemáticas que puede representarse fácilmente usando algoritmos computacionales.

Los **objetivos** fundamentales que buscaba el trabajo de Codd se pueden resumir en:

- **Independencia física**: el modo de almacenamiento de los datos no influye en su manipulación lógica.
- **Independencia lógica**: el añadir, eliminar o modificar objetos de la BD no repercute en el acceso a subconjuntos de los mismos.
- **Flexibilidad**: se puede presentar a cada usuario los datos de una forma distinta.
- **Uniformidad**: las estructuras lógicas de los datos presentan un aspecto uniforme.
- **Sencillez**: es fácil de utilizar y comprender para el usuario final.

En 1985 Codd publicó sus 12 reglas analizando algunos productos comerciales. Éstas son:

1. **Regla de información**: la información está representada explícitamente a nivel lógico y exactamente de un modo, mediante valores en tablas.
2. **Regla de acceso garantizado**: cada uno de los datos se garantiza que sea lógicamente accesible mediante una combinación de nombre de tabla, valor de clave primaria y nombre de columna.
3. **Tratamiento sistemático de valores nulos**: los valores nulos se soportan en los SGBD completamente relacionales para representar la falta de información.
4. **Catálogo en línea** dinámico basado en el modelo relacional: la descripción de la base de datos se representa a nivel lógico del mismo modo que los datos ordinarios.
5. **Regla de sublenguaje completo de datos**: Existen varios tipos de lenguaje. El modelo relacional debe soportar por lo menos un lenguaje relacional que:
   - Tenga una sintaxis lineal.
   - Puede ser utilizado dentro de programas de uso.
   - Soporte operaciones de definición de datos, de manipulación de datos, seguridad e integridad y operaciones de administración de transacciones.
6. **Regla de actualización de vista**: toda vista teóricamente actualizable es también actualizable por el sistema.
7. **Inserción, actualización y supresión de alto nivel**: los datos se pueden recuperar de una base de datos relacional en los sistemas construidos de datos de filas múltiples y/o de tablas múltiples

8. **Independencia física de los datos**.
9. **Independencia lógica de los datos**.
10. **Independencia de integridad**: las limitaciones de la integridad se deben especificar por separado de los programas de la aplicación y se almacenan en la base de datos.
11. **Independencia de distribución:** la distribución de las porciones de la base de datos a las varias localizaciones debe ser invisible a los usuarios de la base de datos.
12. **Regla de no subversión**: si el sistema proporciona una interfaz de bajo nivel de registro, a parte de una interfaz relacional, esa interfaz de bajo nivel no se puede utilizar para subvertir el sistema

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

Módulo 1. Las bases de datos

Sistemas Gestores de Bases de Datos

Las bases de datos requieren básica y fundamentalmente un software de gestión que facilite las operaciones y las interfaces con los usuarios. Esto es el Sistema Gestor de Bases de Datos, SGBD, o en inglés DBMS, Data Base Management System.

Las **operaciones típicas** que debe realizar un SGBD son las siguientes:

- Aquéllas que afectan a la totalidad de los datos.
  - Creación
  - Reestructuración
  - Consultas

- Las que tienen lugar sobre registros concretos, que suelen llamarse operaciones de actualización:
  - Inserciones
  - Borrados
  - Modificaciones
  - Consultas de selección.

Componentes de los SGBD

Para realizar todas las operaciones nombradas anteriormente es necesario que el SGBD cuente con una serie de componentes:

- **Lenguajes de la base de datos**: lenguaje de definición de datos (LDD), lenguaje de manipulación de datos (LMD) y lenguaje de control de datos (LCD).
- **El diccionario de datos**: es un conjunto de archivos que contienen información acerca de los datos que se almacenan en la base de datos. Se trata de una "metabase de datos", es decir, una base de datos que contienen información sobre la base de datos (datos acerca de los datos)
- **El gestor de la base de datos**: Este componente es el encargado de proporcionar una interfaz entre los datos almacenados y los programas de aplicación que los manejan. Puede verse el gestor de la base de datos como un intérprete entre el usuario y los datos. Toda operación que se quiere realizar "contra" la base de datos debe ser previamente

autorizada por el gestor de la misma, el cual, una vez interpretada y validada, o bien realiza la operación devolviendo el resultado de la misma al programa que lo solicitó o bien lo rechaza.

- **El administrador de la base de datos**: Es una persona o grupo de personas encargadas de la función de administración de la base de datos
- **Usuarios de la base de datos**.

Sistemas Gestores de Bases de Datos en el mercado

Actualmente las bases de datos relacionales más conocidas son **MySQL** y **Oracle**, ambas pertenecientes a ORACLE y que requieren instalar el sistema gestor de bases de datos en un sistema operativo.

logo_mysql.png logo_oracle.jpg

MySQL es **gratis** y de **código abierto**, mientras que Oracle es de **pago** y **privado**. Tanto MySQL como Oracle ofrecen soporte tanto de comunidad como técnico.

Aunque ambos son sistemas basados en modelos relacionales, se pueden encontrar bastantes diferencias entre ellos, siendo ambas opciones muy válidas y potentes para trabajar con bases de datos relacionales.

Para evitar tener que andar con instalaciones en este curso, nos vamos a centrar en herramientas que permiten realizar el diseño de las bases de datos online, sin necesidad de instalar ningún gestor en nuestros sistemas operativos,

Para realizar el diseño de los modelos Entidad-Relación, así como el relacional, se pueden utilizar distintas herramientas que permitan crear diagramas, hay muchas en el mercado. Entre ellas podemos encontrar tanto de escritorio como online, como **ERD Plus** o **Creatly**, que son online o **Dia**, que es una aplicación de escritorio pero también se puede utilizar online a través de la web https://www.rollapp.com/app/dia y es la que yo recomiendo utilizar por su semejanza con la notación explicada a lo largo de este curso. Todas las herramientas tienen alguna limitación a la hora de crear los diagramas E/R, como no permitir crear relaciones que no sean binarias, o poner cardinalidades en las relaciones, por ejemplo. Así que elijáis la que elijáis, en algún momento tendréis que idear la manera de representar lo que no permite esa herramienta en concreto. Durante este curso se va a utilizar la notación de base de datos Chen, y la aplicación Dia la permite.

Hay dos opciones, o bien instalar la aplicación Dia en vuestro ordenador, descargándola desde su web aquí (fijarse que hay versión para Windows, Mac y Linux, hay que elegir la correcta) o bien registrarse en rollApp (https://www.rollapp.com) y luego buscar la aplicación Dia y lanzarla online.

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

Módulo 1. Las bases de datos

Diseño de una base de datos

El diseño de bases de datos requiere de una metodología lo suficientemente potente para realizar el proceso de modelado de una forma metódica y eficiente, ya que a la hora de realizar este trabajo pueden aparecer problemas difíciles de solucionar más adelante.

La manera óptima de diseñar una base de datos es realizando el proceso en fases. Las fases a seguir para modelar un sistema de bases de datos relacional son:

Fases diseño.png

- **Diseño conceptual**: se obtiene a través del estudio del problema y seleccionando qué elementos se van a modelar. En el caso del modelo relacional, se utiliza el Modelo Entidad-Relación.
- **Diseño lógico**: El objetivo del diseño lógico es convertir el esquema conceptual en un esquema lógico que se ajuste al modelo de SGBD sobre el que se vaya a implementar el sistema. El proceso a seguir para realizar el diseño lógico consiste en tomar el diagrama Entidad-Relación obtenido en el diseño conceptual y obtener las relaciones o tablas propias del modelo relacional, siguiendo unas determinadas reglas de transformación. El diagrama obtenido en este paso puede presentar algunos problemas derivados de fallos a a hora de interpretar del problema real, fallos arrastrados del diseño del diagrama Entidad-Relación o fallos del paso al modelo relacional. Entre otros problemas destacan:
  - Incapacidad para almacenar ciertos hechos.
  - Redundancias, y por tanto, posibilidad de incoherencias.
  - Pérdida de información.
  - Pérdida de dependencias funcionales, es decir, de ciertas restricciones de integridad que dan lugar a interdependencias entre los datos.
  - Aparición, en la base de datos, de estados que no son válidos en el mundo real, es decir, anomalías de inserción, borrado y modificación.

  Es por esto que el esquema relacional debe ser analizado para comprobar que no presenta estos problemas. Y esto se hace por medio de la **normalización** de las tablas que, aunque es algo que no abordamos en este curso, es importante nombrarlo para saber que es un paso importante en el diseño de bases de datos relacionales.

- **Diseño físico**: se realiza la implementación propiamente dicha de la base de datos y está estrechamente relacionada al SGBD. Los pasos a seguir para la realización de un buen diseño físico son:

- Traducir el esquema lógico para el SGBD específico.
- Diseñar la representación física.
- Diseñar los mecanismos de seguridad.
- Monitorizar y afinar el sistema.

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

Módulo 2. Diseño de una base de datos

Explicación del diagrama Entidad-Relación, el modelo relacional y la transformación del diagrama Entidad-Relación al modelo relacional.

Módulo 2. Diseño de una base de datos

Diagrama Entidad-Relación

Gracias al modelo conceptual Entidad-Relación, creado por Peter Chen en los años setenta, se puede representar el mundo real a través de una serie de símbolos y expresiones determinados. El objetivo de este modelo es obtener una representación de la realidad que capture sus propiedades.

Al tratarse de un modelo conceptual, no está orientado a ningún sistema físico concreto: tipo de ordenador, SGBD, sistema operativo... Tampoco tiene un objetivo informático concreto, podría perfectamente utilizarse para explicarle a una persona el funcionamiento de cualquier proceso, por lo que debe ser fácil de comprender.

Lo primero que se debe que hacer cuando se va a crear una base de datos es:

- Analizar el problema
- Pensar qué tipo de información se necesita almacenar, o dicho de otra forma, qué tipo de información se necesitará obtener de la base de datos.

Antes de seguir con el diseño de diagramas, hay que aclarar que distintos autores utilizan distintos símbolos o maneras de representar algunos elementos y características del modelo E/R, por lo que mientras el diseño esté bien realizado, el modo de expresarlo no es lo más importante, mientras todos los que trabajen en el problema utilicen y entiendan los mismos elementos gráficos. En Internet encontraréis diferentes maneras de representar un diagrama entidad relación. Lo importante es que plasmen el problema que se estudia de manera correcta.

Entidades y relaciones

El primer paso para elaborar el diagrama E-R para una base de datos es:

- Encontrar las **entidades** que intervienen en el problema.
- Encontrar las **relaciones** existentes entre esas entidades.
- Encontrar los **atributos** que tienen esas entidades y esas relaciones.

Entidades

Al observar el mundo que rodea al problema que se quiere plasmar en el diagrama, se puede detectar el conjunto de objetos de los que se desea almacenar información. Todos los objetos de una misma clase se representan con un tipo de entidad concreto (por simplicidad, se les nombra simplemente entidades), que se diferencia de otra entidad porque posee ciertas características que

la hacen única.

Cada entidad tendrá una serie de instancias, que no son más que ocurrencias concretas de ese tipo de entidad.

En resumen: **una entidad representa cualquier persona, suceso, evento o concepto (cualquier "cosa") sobre el que se desea almacenar información**.

En el modelo E/R una entidad se representa con un rectángulo en cuyo interior aparece el nombre de la entidad.

Entidad asignatura.png

Es importante tener en cuenta que el diseño E/R acabará plasmado en un sistema gestor de bases de datos (SGBD) físico en un ordenador, por lo que es interesante respetar los nombres utilizados en el diseño, y por esa razón es conveniente no utilizar tildes en los nombres, ya que hay SGBD que no aceptan este tipo de caracteres.

> Existen tipos de entidades, pero no se va a entrar en ese detalle en este curso, para el que desee profundizar en el contenido del curso, existen apuntes de la UOC (Universitat Oberta de Cataluña) con licencia Creative Commons sobre el modelo conceptual en este enlace.

Relaciones

Se entiende por relación aquella **asociación o correspondencia existente entre entidades**.

Se representa mediante un rombo con el nombre de la relación en el interior.

En el siguiente ejemplo, se representa la relación "trabaja" que se establece entre un empleado y una sucursal bancaria, de forma que represente que un empleado trabaja en una sucursal bancaria, y que la sucursal bancaria es el lugar de trabajo del empleado.

Relacion.png

En las líneas que unen las entidades con las relaciones se puede escribir el rol o papel que desempeña una entidad en la relación en caso de que dicho papel no quede claro.

Para definir una relación se tienen en cuenta los siguientes elementos:

- **Nombre de la relación**: cada relación tiene un nombre que la distingue del resto y mediante el cual se hace referencia a ella. Habitualmente se utiliza un verbo en forma singular. (Trabaja, tiene, produce, etc)

- **Grado de la relación**: es el número de entidades que participan en una relación.
- **Cardinalidad de la relación**: Es el número máximo de instancias de cada entidad que pueden intervenir en una instancia de la relación que se está tratando. En la representación gráfica aparece como una etiqueta con **1:1**, **1:N,** o **N:M**, que se leen respectivamente como uno a uno, uno a muchos y muchos a muchos.

  Ejemplos de cardinalidades:

- ◦ **Uno a uno**, es el caso de las entidades EMPLEADO, DEPARTAMENTO, y la relación 'es_jefe' en el que un departamento solo pueda tener un jefe y un empleado solo pueda ser jefe de un departamentp, y viceversa.
  ◦ **Uno a muchos**, es el caso de las entidades EMPRESA, EMPLEADO y la relación 'trabaja'. Considerando que no se permita el pluriempleo, en una empresa concreta trabajan muchos empleados, pero un empleado sólo trabaja en una empresa concreta.
  ◦ **Muchos a muchos**, es el caso de las entidades CLIENTE, ARTÍCULO y la relación 'compra'. Un cliente puede comprar diferentes artículos y un artículo puede ser comprado por diferentes clientes.

- **Cardinalidades de las entidades**: hace falta un apartado entero para esto. Vayamos a ello.

Cardinalidades de las entidades

Las cardinalidades de las entidades se definen como el número mínimo y máximo de instancias de una entidad que pueden estar relacionadas con una instancia de otra u otras entidades que participan en la relación. Su representación gráfica es una etiqueta del tipo **(0,1)**, **(1,1)**, **(0,n)** o **(1,n)**, según corresponda.

Por ejemplo, el siguiente diagrama representa el hecho de que un departamento de una empresa puede tener varios empleados trabajando en él (lo indica la cardinalidad máxima n) o ningún empleado trabajando en él (lo indica la cardinalidad mínima 0) y un empleado debe trabajar como mínimo y como máximo en un solo departamento.

Cardinalidades.png

Cardinalidades de las relaciones

En el caso de las **relaciones**, la cardinalidad indica el número de instancias de una entidad que se relacionan con un ejemplar de la otra entidad que forma parte de la relación, y viceversa.

La cardinalidad de las relaciones se obtiene de considerar el máximo número de instancias con las que puede participar cada una de las entidades en la relación, es decir con el máximo de las

cardinalidades de cada una de las entidades que participan en la relación.

Dependiendo del número de instancias que aparezcan, podemos tener:

- **Relación uno a uno.** En la notación se pone **1:1**. Con ejemplos se verá más fácil. Veamos el caso de las entidades EMPLEADO y PUESTO_DE_TRABAJO, y la relación “ocupa”. Suponiendo que un determinado puesto de trabajo puede estar ocupado por un solo empleado, y al mismo tiempo, un empleado puede ocupar simultáneamente un único puesto de trabajo. El diagrama sería el siguiente:

Cardinalidad 11.png

- **Relación uno a muchos**. En la notación se pone **1:N**. Por ejemplo, teniendo las entidades ASIGNATURA y PROFESOR, y la relación "imparte" para un curso concreto. En el caso de que una asignatura pueda ser impartida por un único profesor (no contemplando desdobles), pero cada profesor pueda impartir muchas asignaturas. El diagrama sería:

Cardinalidad 1N.png

- **Relación muchos a uno**. Es el mismo concepto que el de una relación uno a muchos (1:N).
- **Relación muchos a muchos**. En la notación se pone **N:M**. En el caso de una empresa de autobuses, si consideramos las entidades CONDUCTOR y AUTOBÚS, y la relación "conduce", lo normal es que cada autobús pueda ser conducido por distintos conductores, en diferentes turnos, y al mismo tiempo, que cada conductor pueda conducir varios autobuses en distintos turnos, de forma que cada autobús se relaciona con muchos conductores, y cada conductor se relaciona con muchos autobuses, formando una relación muchos a muchos.

Cardinalidad NM.png

Aunque en este momento pueda parecer que los conceptos **cardinalidad de una relación** y **cardinalidad de una entidad** son muy similares, ambos son necesarios para la transformación del diagrama E/R al modelo relacional, como se verá más adelante.

Grado de una relación

El grado de una relación es el número de entidades que participan en la relación.

Se pueden encontrar los siguientes tipos de relaciones según su grado:

- **Reflexiva:** participa una única entidad.

Relacion reflexiva.png

- **Binaria**: Es aquella relación en la que participan dos entidades, es el tipo más habitual de relación.

Cardinalidad 1N.png

- **Ternaria**: Es aquella relación en la que participan tres entidades al mismo tiempo.

Ternaria.png

Este diagrama representa que una película se relaciona con un actor que ha interpretado un determinado personaje de los que forman parte del guión. O que un personaje se relaciona con la película de la que forma parte y con el actor que lo interpreta. O que un actor se relaciona con el personaje que interpreta y con la película en la que interviene... distintas formas de decir lo mismo.

- **N-aria**: Es aquella relación en la que participan n conjuntos de entidades. Es muy poco frecuente su aparición y es importante intentar disminuir el grado de la relación para hacer más intuitivo el modelado del sistema.

N-aria.png

Imaginemos una relación de orden 4, como la de la imagen, ¿cómo podemos disminuir su orden?

Las relaciones que expresa el diagrama son: un actor se relaciona con una película en la que interviene, que es producida por un estudio, y lo hace a cambio de un determinado salario de la tabla salarial que tienen establecida en ese estudio, y todas esas relaciones son en realidad contractuales, es decir, derivadas de contratos.

n-aria 2.png

Sustituyendo la relación 'tiene_contrato' por una entidad nueva llamada CONTRATO y convirtiendo todas las relaciones en binarias se elimina la relación de grado 4.

Ahora, cada entidad de PELÍCULA, SALARIO, ESTUDIO y ACTOR se relaciona con las demás entidades sóla a través de la entidad CONTRATO.

Atributos

Un atributo es cualquier **característica que sirve para calificar, identificar, clasificar, cuantificar o expresar el estado de algo**. Por ejemplo, las propiedades, cualidades, identificadores o características de entidades o relaciones

También hay que tener en cuenta si los atributos son simples o compuestos, o si son obligatorios u opcionales.

Los atributos de una entidad se representan mediante elipses o círculos etiquetados, que se conectan por una línea recta a la entidad o relación que califican, cada uno de los cuales tiene que tener un nombre único y que haga referencia a su contenido. Los nombres de los atributos deben ir en minúsculas.

Atributos.png

Cada atributo tiene un conjunto de valores asociados denominado **dominio**. El dominio define t **odos los valores posibles que puede tomar un atributo**.

Tipos de atributos

Los atributos pueden ser **obligatorios** u **opcionales**.

Un atributo obligatorio es aquél que siempre debe estar definido, como por ejemplo, el que identifica a una entidad. En una entidad EMPLEADO, un atributo obligatorio de esa entidad podría ser 'DNI', que no se puede dejar vacío porque gracias a él se identifican todas y cada una de las instancias de esa entidad.

Un atributo opcional, en cambio, puede quedar sin definir para algunas de las instancias de la entidad. En el caso de la entidad EMPLEADO un atributo opcional podría ser 'edad', que es un atributo que no es imprescindible para la identificación de las instancias de la entidad.

Otra clasificación de los atributos es: **simples** o **compuestos**.

Un atributo simple es un atributo que tiene un solo componente, que no se puede dividir en partes más pequeñas que tengan un significado propio, un ejemplo podría ser el DNI de una persona.

Un atributo compuesto es un atributo con varios componentes, cada uno con un significado por sí mismo. Por ejemplo, considerando la dirección de una persona como la unión de la calle donde vive, el número y la población.

En la siguiente figura se observa la forma de representar los atributos simples y los compuestos, Dirección es un atributo compuesto, que consta de varios componentes simples (calle, numero, poblacion).

Tipos atributos.png

Las relaciones también pueden tener atributos asociados. Se representan igual que los atributos de las entidades.

Imaginar que es necesario guardar la fecha de emisión de una factura a un cliente, y que es posible emitir duplicados de la factura (con distinta fecha). En tal caso, el atributo "Fecha de emisión" de la factura debería colocarse en la relación "se emite".

Atributo relacion.png

Clave primaria

Es muy importante poder distinguir cada instancia de una entidad o de una relación. Para esto tenemos la clave primaria.

Cada instancia de una entidad se puede distinguir de cualquier otra por todos sus atributos, y la mayoría de las veces no son necesarios todos, sino un subconjunto de ellos. Pero puede ocurrir que un subconjunto sea igual para varias entidades, por lo que no nos vale cualquier subconjunto. Lo importante es que el conjunto de todos los atributos que se seleccionan no se repita con los mismo valores para distintas instancias. Teniendo en cuenta esto se definen diferentes conceptos:

- **Clave**: es el atributo o conjunto de atributos que identifican a una entidad. Por ejemplo el DNI identifica una instancia de cualquier otra dentro de la entidad EMPLEADO, por lo que puede considerarse una clave de EMPLEADO. A veces es necesario más de un atributo para conseguir la identificación de las instancias. En ese caso la clave está constituida por el conjunto de atributos que garantice la identificación de cada una de las instancias.
- **Clave candidata**: Se trata de uno o más atributos cuyos valores son únicos para cada instancia de una entidad, pero que no son elegidos para identificar la entidad. Si tenemos la entidad EMPLEADO con los siguientes atributos: DNI, codigo_empleado, nombre, apellidos, direccion, fecha_nacimiento, etc... Dos claves candidatas son DNI y codigo_empleado, ya que ambas identifican de manera única una instancia de EMPLEADO.
- **Clave primaria**: Es la clave candidata elegida para identificar la entidad. Debe cumplir además que ningún subconjunto de ella sea clave candidata. En el caso anterior de la entidad EMPLEADO, pueden ser clave primaria tanto DNI, como codigo_empleado, y depende del criterio del diseñador de la base de datos que elija una u otra. Pero una vez que el diseñador elige una, sólo ese atributo (o conjunto de atributos) es clave primaria.

A continuación hay un diagrama en el que se representa el caso de la entidad Empleado y su clave primaria.

image-1661434932607.png

Restricciones avanzadas del modelo Entidad-Relación

> En este curso no se va a entrar en estos contenidos por ser avanzados, pero podéis encontrar apuntes de la UOC con licencia Creative Commons en esta web.

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

Módulo 2. Diseño de una base de datos

Creación de un diagrama Entidad-Relación

Se va a realizar el diagrama E/R que representa la información de una Universidad sobre los grados y sus asignaturas que se pueden estudiar en ella y se va a incluir, además, la información de los alumnos matriculados en las asignaturas junto con las calificaciones que obtienen en ellas. Para este caso se va a suponer que una asignatura solo puede pertenecer a una grado.

Lo primero es buscar las **entidades**, que son **Asignatura**, **Grado** y **Alumno**.

Después las **relaciones** entre las entidades. Asignatura se relaciona con Grado a través de la relación "**pertenece**", y Alumno se relaciona con Asignatura a través de "**esta_matriculado**".

Creacion diagrama 1.png

Ahora se ponen los atributos de las entidades y las relaciones (si los tuvieran). El enunciado del problema es bastante escueto y no nombra qué información se desea guardar. Normalmente los enunciados deben ser completos para no dejar nada a la imaginación del diseñador y que todo quede lo más aproximado posible al problema real.

En este caso no dicen nada, así que lo primero que debemos pensar es que las entidades deben estar identificadas y guardar la información más común de estas entidades. Por ejemplo, de Grado podemos guardar el **nombre** y un **código interno** dentro de la universidad que lo distingue en la base de datos. De las asignaturas, el **nombre**, **código de la asignatura** (que será la clave primaria), **duración** y **horas semanales**. De los alumnos, **DNI** (que será la clave primaria), **nombre**, **apellidos**, **fecha de nacimiento** e **email**.

Con todo esto el diagrama quedaría así (a falta de estudiar las cardinalidades).

Creacion ER final.png

Hay que fijarse que el atributo **nota** está en la relación porque es la nota que el alumno obtiene en una asignatura en concreto y para cada asignatura tendrá una nota distinta, por lo tanto, no puede ser un atributo de la entidad.

Ahora falta añadir las cardinalidades.

Empezamos con la relación "**esta_matriculado**". Un alumno puede estar matriculado en 1 o varias asignaturas, así que en el lado de la relación más próximo a la entidad Asignatura, se pone la cardinalidad **(1,n)**. Una asignatura puede tener a ninguno o a varios alumnos matriculados, así que en el lado de la relación más próximo a la entidad Alumno se escribe la cardinalidad **(0,n)** y de estas dos cardinalidades, se deriva la cardinalidad de la relación, **N:M**.

La relación "**pertenece**". Una asignatura pertenece a un y solo un grado, con lo que la cardinalidad de la relación más próxima a la entidad grado, será **(1,1)**. Y un grado se compone de mínimo 1 y máximo varias asignaturas, así que en el lado más próximo a la entidad Asignatura, se pone la cardinalidad **(1,n)**. De estas cardinalidades, la cardinalidad de la relación es **1:N**.

El diagrama final del problema es el siguiente:

Diagrama finall.png

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

Módulo 2. Diseño de una base de datos

Crear diagrama ER con DIA

Vamos a crear el diagrama de la universidad que se ha explicado en la página Creación de un diagrama Entidad-Relación, utilizando el programa Dia.

Registrarse en rollApp

Por pasos, primero hay que registrarse en rollApp:

rolApp signup.png

Una vez finalizado el proceso de registro, al acceder se ve una página en donde se muestran unas sugerencias de aplicaciones y se puede buscar la que deseamos utilizar. Para buscar la que queremos hay que pinchar en Apps del menú principal y luego buscar en el buscador de la derecha, como se muestra en la imagen:

rollApp Apps.png

Al introducir en el buscador la aplicacion "Dia", nos aparecerán las sugerencias del buscador y hay que seleccionar la aplicación Dia, como muestra la imagen siguiente:

rollApp select Dia.png

Abrimos esa aplicación y podemos empezara a utilizarla seleccionando "Launch online":

rollApp Dia LaunchOnline.png

Esto abrirá la aplicación en una nueva ventana:

Dia.png

Una vez abierta la aplicación hay que seleccionar el diagrama que vamos a utilizar del desplegable de diagramas disponibles: ER

Dia ER.png

Ahora los elementos que aparecen son los del diagrama ER

Dia ER Elementos.png

Entidad

Para crear una entidad se selecciona el primer elemento, el rectángulo con la E dentro de él, y se pincha sobre la parte del lienzo en la que se desea crear la entidad. Esto crea un cuadro vacío en el que tendremos que poner el nombre de la entidad e ir añadiéndole atributos y relaciones.

Dia Entidad.png

Si solo queremos cambiar el nombre de la entidad, basta con pulsar F2 teniendo la entidad seleccionada y nos dejará editar el nombre. Si lo que se desea es abrir todas las propiedades de la entidad, hay que hacer doble clic sobre ella y se abre la ventana de las propiedades:

Dia propiedades entidad.png

Atributos

Para añadir atributos, hay que seleccionar el elemento que es un círculo con una A dentro de él y pinchar en la parte del lienzo en la que se desea crear el atributo. Esto crea una elipse a la que hay que cambiar el nombre y ponerle el nombre del atributo correspondiente. Las propiedades se abren igual que la entidad.

Dia propiedades atributo.png

Para indicar que un atributo es clave primaria, bastaría con clicar sobre el botón con etiqueta "Clave".

Para unir los atributos a su entidad, hay que utilizar el elemento que son como dos líneas paralelas. Clicar en el atributo y arrastrar hasta la entidad y entonces soltar. También se puede utilizar una linea normal del los elementos generales, que queda más bonito.

Así tendríamos ya:

Dia ER Entidad-atributos.png

Relaciones

Para añadir relaciones hay que crearlas y luego unirlas, igual que se hace con los atributos.

Una relación se añade con el elemento que es un rombo con una R en él. Se selecciona y se clica sobre la parte del lienzo en la que se desea poner la relación. Y se une a las entidades igual que los atributos. Así, entre Asignatura y Grado, tendremos:

DIa relacion dos entidades.png

Cardinalidades

Ahora hay que añadir las cardinalidades. Las de las entidades se añaden desde el cuadro de propiedades de la relación.

Dia Propiedades relacioni.png

Ahora ya tenemos dos entidades con sus atributos y su relación, pero falta la cardinalidad de la relación que, en el caso de la App Dia hay que ponerlo como una etiqueta de texto del cuadro general de elementos:

Dia etiqueta texto.png

Con todo esto ya tendríamos toda la relación completa:

Dia relacioin completa.png

Siguiendo los mismo pasos se van añadiendo el resto de entidades y relaciones del diagrama.

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

Módulo 2. Diseño de una base de datos

Conceptos fundamentales del modelo relacional

El modelo relacional se basa en el concepto matemático de relación, que se representa gráficamente mediante una tabla. Codd utilizó terminología matemática para definir el modelo relacional, en concreto la de la teoría de conjuntos y de la lógica de predicados.

El concepto de relación

La **relación** es el elemento básico del modelo relacional y está compuesta por dos partes:

- **Cabecera**. Formada por un conjunto fijo de atributos. Es la parte fija de la relación. Está constituida por:
  - El nombre del conjunto: la tabla.
  - El nombre de los atributos: las columnas de la tabla.
  - Los dominios de los que toman valores los atributos.
- **Cuerpo**. El cuerpo está formado por un conjunto de filas.

El número de columnas que tiene una relación recibe el nombre de **grado** de la relación, y el número de filas recibe el nombre de **cardinalidad** de la relación.

Como ejemplo, la siguiente tabla representa la relación PERSONA.

NOMBRE	APELLIDOS	EDAD	TELÉFONO
Alfonso	Gutiérrez Pérez	38	698 569 854
Lucía	López García	37	666 999 888
Jorge	Juan Bonilla	38	632 458 785
Ana	García García	29	654 987 321
Diego	Rodríguez Gracia	36	632 985 632
Marta	Pérez Martínez	34	678 521 456
Alberto	Vega Domínguez	31	698 584 521
Manuela	Fernández Hernández	35	636 696 898
Silvia	Gracia Barrós	36	654 654 654

La cabecera de esta relación es: PERSONA (NOMBRE, APELLIDOS, EDAD, TELEFONO)

El cuerpo es el conjunto de 9 filas con los datos concretos de personas, el grado de la relación es 4 y la cardinalidad 9.

Las tablas del modelo relacional cumplen las siguientes propiedades:

- No existen filas repetidas: el cuerpo de la relación es un conjunto matemático y en matemáticas, por definición, los conjuntos no incluyen elementos repetidos. Esto se traduce en que dos registros de una misma relación deben diferir, al menos, en el valor de un campo.
- Las filas no están ordenadas: esta propiedad muestra la diferencia entre una relación y una tabla, porque las filas de una tabla tienen un orden obvio de arriba hacia abajo, mientras que las filas de una relación no tienen orden.
- Los atributos no están ordenados: esto proviene del hecho de que la cabecera de una relación se define también como conjunto. Las columnas de una tabla tienen un orden evidente de izquierda a derecha, pero los atributos de una relación no tienen orden.
- Todos los valores de los atributos son atómicos: esto quiere decir que un atributo sólo puede tomar un valor en cada fila.

Clave primaria y claves ajenas

Sea va a recordar lo que era una clave primaria y se va a añadir un concepto nuevo, el de clave ajena.

En el modelo relacional tenemos los siguientes tipos de claves:

- **Clave candidata**: es un conjunto mínimo y no vacío de atributos que identifica unívocamente cada registro de una relación.
- **Clave primaria**: es la clave candidata que elige el usuario para identificar los registros de una relación. Una clave primaria es compuesta cuando está formada por más de un atributo.
- **Clave alternativa**: es cualquiera de las claves candidatas que no han sido elegidas como clave primaria.
- **Clave ajena**: Es un conjunto no vacío de atributos de una relación cuyos valores han de coincidir con los valores de la clave primaria de otra relación. Las dos relaciones no tienen que ser necesariamente distintas, podrían ser la misma relación (es el caso de las relaciones reflexivas).

Un ejemplo a continuación:

Oficina					
NUM_OFICINA	**CALLE**	**AREA**	**POBLACION**	**TELEFONO**	**FAX**

001	Rúa Seco, 19	Sur	Olite	948 222222	948 658745
002	Larraga, 21	Norte	Olite	948 121212	948 123465
003	Tudela, 15	Sur	Pamplona	948 323232	948 236589
004	Italia, 12	Centro	Zaragoza	976 658745	976 548721
005	de la Parra, 16	Centro	Teruel	978 225588	978 159732

El atributo NUM_OFICINA es una clave candidata ya que identifica de manera única cada registro de la tabla, y en este caso además es la clave primaria porque no existe en la tabla ninguna otra clave candidata.

Empleado					
NUM_EMPLEADO	**DNI**	**OFICINA**	**TELEF_FIJO**	**TELEF_MOVIL**	**FAX**
12340	25369874	005	978 225588	655 191919	978 159732
12350	72658412	002	948 121212	655 212019	948 123465
12360	72658965	003	948 323232	655 242563	948 236589
12370	25814796	001	976 456985	655 256985	948 658745
12380	25369854	004	976 658745	655 658965	976 548721

En esta tabla hay dos claves candidatas, el atributo NUM_EMPLEADO y el atributo DNI, ya que ambos identifican de manera única a cada registro de la tabla. Si se considera el atributo DNI como clave primaria, el atributo NUM_EMPLEADO pasa a ser una clave alternativa de la tabla.

Si la relación OFICINA tuviera la siguiente estructura:

Oficina					
NUM_OFICINA	**CALLE**	**AREA**	**POBLACION**	**TELEFONO**	**DIRECTOR**
001	Rúa Seco, 19	Sur	Olite	948 222222	25369874
002	Larraga, 21	Norte	Olite	948 121212	72658412
003	Tudela, 15	Sur	Pamplona	948 323232	72658965
004	Italia, 12	Centro	Zaragoza	976 658745	72658412
005	de la Parra, 16	Centro	Teruel	978 225588	72658965

Se puede ver que el atributo DIRECTOR hace referencia al atributo DNI de la tabla EMPLEADO, que además es la clave primaria de dicha tabla, por lo que DIRECTOR es la clave ajena de la tabla OFICINA y referencia la tabla EMPLEADO.

Las claves primarias y ajenas cumplen una serie de propiedades:

- Una clave ajena y la clave primaria de la tabla referenciada asociada han de estar definidas sobre los mismos dominios.
- Una tabla puede poseer más de una clave ajena. Tendrá una clave ajena por cada tabla referenciada de la cual dependa.
- Una tabla puede no tener ninguna clave ajena.
- Una clave ajena puede relacionar una tabla consigo misma (relaciones reflexivas).

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

Módulo 2. Diseño de una base de datos

Transformación del modelo Entidad-Relación al modelo relacional

Principios de transformación

La transformación de un diagrama E/R al modelo relacional está basado en los siguientes principios:

- Toda entidad se convierte en una tabla.
- Toda relación N:M se transforma en una tabla.
- Toda relación 1:N se traduce en el fenómeno de "propagación de clave" (se crea una clave ajena).

Transformación de las entidades y sus atributos

Entidades: cada entidad que aparece en el diagrama E/R se convierte en una tabla.

Atributos de las entidades: Cada atributo de una entidad se transforma en una columna de la tabla a la que ha dado lugar la entidad.

Ahora vamos a ver cómo se definen cada uno de los tipos de atributos:

- Todos los atributos pasan a ser columnas de la tabla.
- Los atributos que forman parte de la clave primaria de una entidad pasan a ser la clave primaria de la tabla. Se debe especificar que no son nulos, esto es, que no pueden quedarse esos campos vacíos al insertar filas nuevas en la tabla.

Siguiendo con el ejemplo del diagrama E/R de la universidad, se tendrían las siguientes tablas con sus atributos. Las claves primarias están marcadas en negrita y subrayadas.

Paso al relacional 1.png

Transformación de las relaciones y sus atributos

Transformación de las relaciones entre entidades: dependiendo del tipo de relación y de la cardinalidad que tenga, existen diversas maneras de transformarlas:

- **Relaciones N:M**. Se crea una nueva tabla que incluye los atributos de la relación (si los tiene) y las claves primarias de las dos entidades, que forman la clave primaria de la nueva tabla.
- **Relaciones 1:N**. Estas relaciones se pueden transformar de dos maneras diferentes:
  - Propagar la clave primaria de la entidad que tiene cardinalidad máxima 1 a la que tiene cardinalidad máxima N, y hacer desaparecer la tabla de la relación como tal. Esto quiere decir que el atributo que es clave primaria en la entidad con cárdinalidad máxima 1 se añade como columna a la tabla que surge de la entidad que tiene cardinalidad máxima N. Además esta columna sería también clave ajena de la tabla, referenciando a la otra tabla de la relación. Si la relación tuviera atributos asociados, estos atributos pasan a formar parte de la tabla correspondiente al tipo de entidad que participa con cardinalidad máxima N.
  - Transformarla en una nueva tabla como si fuese de una relación de tipo N:M, es decir, incluyendo los atributos de la relación y las claves primarias de las dos entidades. Esta acción es recomendable sólo en los siguientes casos:
    - Cuando es posible que aparezcan muchos nulos (campos vacíos en las filas) porque existen pocos elementos relacionados.
    - Cuando se prevé que dicha relación pasará en un futuro a ser de tipo N:M,
    - Cuando la relación tiene atributos propios.
- **Relaciones 1:1**. Este es un caso particular de cualquiera de los dos casos anteriores, por lo que se podrían aplicar las reglas anteriores. Es recomendable tener en cuenta las siguientes recomendaciones:
  - Si la relación es entre entidades con cardinalidades (0,1) y (0,1), es mejor crear una relación para evitar tener muchos nulos como propagación de alguna de las claves a la otra.
  - Si la relación es entre entidades con cardinalidades (0,1) y (1,1), es mejor propagar la clave de la entidad (1,1) a la (0,1).
  - Si la relación es entre entidades con cardinalidades (1,1) y (1,1), la propagación es indiferente, y se hará atendiendo a los criterios de frecuencia de acceso (consulta, modificación, inserción, etc.) a cada una de las tablas en cuestión.
- **Transformación de relaciones ternarias (grado 3)**.
  - **Relaciones muchos a muchos a muchos**. Este tipo de relación se transforma en una tabla cuya clave primaria es la concatenación de las claves primarias de las tablas surgidas al transformar las entidades que forman parte de la relación. Junto a estos atributos se incluyen los atributos propios de la relación. Cada uno de los atributos que forman la clave primaria de esta tabla son a la vez claves ajenas respecto a cada una de las tablas donde dicho atributo es clave primaria.
  - **Relaciones muchos a muchos a uno**. Este tipo de relación se transforma en una tabla cuya clave primaria es la concatenación de las claves primarias de las tablas

que corresponden a la cardinalidad N y M, surgidas al transformar las entidades que forman parte de la relación. Junto a estos atributos se incluyen los atributos propios de la relación más la clave primaria de la tabla que corresponde a la cardinalidad 1. Cada uno de los atributos que forman la clave primaria de esta tabla y los atributos añadidos de la relación de cardinalidad 1 son claves ajenas respecto a cada una de las tablas donde dicho atributo es clave primaria.

Transformación de los atributos de relaciones

Si la relación se transforma en una tabla, todos sus atributos pasan a ser columnas de la tabla. En el caso en que alguno de los atributos de la relación sea clave primaria, deberá ser incluido como parte de la clave primaria en dicha tabla.

En el ejemplo de la Universidad, tenemos una relación 1:N y otra N:M. De la relación 1:N tendremos una propagación de clave, es decir, la clave primaria de la tabla con cardinalidad máxima 1 (Grado), pasa como atributo a la otra tabla (Asignatura) y además como clave ajena que referencia a la clave primaria de Grado. De la relación N:M se obtiene una nueva tabla con los atributos que tiene la relación (Nota) y la clave primaria la forma la unión de las claves primarias de las entidades que intervienen en la relación (Codigo de la asignatura y DNI del Alumno).

Paso al relacional 2.png

> Para realizar estos diagramas, os recomiendo la herramienta **ERD Plus**, que solo requiere registro en la misma y además permite luego exportar el código SQL de los diagramas relacionales, lo que es muy útil para luego generar la base de datos en un sistema físico.

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

Módulo 2. Diseño de una base de datos

Crear diagrama relacional con ERD Plus

Vamos a ver cómo se crearía un diagrama relacional con la herramienta **ERD Plus**.

Imaginad que tenemos el diagrama ER siguiente sobre una base de datos de recetas:

Recetas ER.png

Si diagrama relacional correspondiente es el siguiente:

Recetas relacional.png

Voy a indicar cómo se crea cada uno de los elementos del diagrama.

Tablas

Hay que clicar sobre el botón "table":

Boton table.png

A continuación clicar sobre cualquier punto del lienzo de dibujo y se creará una tabla vacía:

Tabla vacia.png

Seleccionando la tabla, a la derecha aparecen todas las opciones de la tabla:

Propiedades tabla2.png

Vamos a crear la tabla Receta.

Tiene 5 columnas, pero una es una clave ajena, así que esa se añade de otra manera que se explica más abajo.

Para añadir las columnas, hay que clicar sobre el botón "ADD" y se abre debajo la ventana con las columnas:

columnas receta.png

Se van rellenando los nombres de las columnas y el tipo de dato como en la imagen de arriba.

Ahora, para indicar la clave primaria, se clica en "PRIMARY KEY" y se abre una ventana con todas las columnas para seleccionar cuáles forman parte de la clave primaria:

clave primaria recetas.png

También hay que indicar que la columna notas es opcional (no obligatoria). Para eso clicar sobre "OPTIONAL" y hacer lo mismo que al seleccionar la clave primaria:

opcionales receta.png

Falta solo añadir la clave ajena. Para eso es necesario tener previamente creadas las dos tablas que intervienen en la clave ajena, en este caso la tabla "TIPO_RECETA" y la tabla "RECETA".

Se selecciona el botón "CONNECT" del menú principal y hay que clicar en la tabla cuya clave primaria va a ser clave ajena (TIPO_RECETA) y arrastrar hasta la tabla en la que va a añadirse la columna como clave ajena (RECETA):

Clave ajena receta.png

Y la columna se añade sola a la tabla RECETA.

Siguiendo estos mismos pasos para el resto de tablas, se crea el diagrama completo.

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

Módulo 3. Implementación del modelo relacional en un SGBD

Módulo 3. Implementación del modelo relacional en un SGBD

Lenguaje DDL

Para almacenar y tratar la información esquematizada en los modelos E/R y relacional por medio de un Sistema Gestor de Bases de Datos se utiliza el **lenguaje SQL**. SQL son las siglas de Structured Query Language (Lenguaje de Consulta Estructurado).

Es un lenguaje que permite interactuar con los SGBD Relacionales para especificar las operaciones que se desean realizar sobre los datos y su estructura. Es un lenguaje declarativo, lo cual quiere decir que en él se especifica al Sistema qué se quiere obtener pero no cómo conseguirlo.

Lenguaje DDL

Para la creación de las tablas de la base de datos, se utiliza el lenguaje de descripción de datos (DDL, Data Definition Language).

A continuación se explica la sintaxis de cada sentencia para llegar a crear una base de datos SQL.

> Hay que tener en cuenta que en este curso nos centramos en el lenguaje SQL del SGBD MySQL, y cada uno tiene sus particularidades y sus diferencias, pero en lineas generales todos deben seguir el estándar SQL y, por lo tanto, tener todos más o menos la misma sintaxis.

Toda la sintaxis detallada del lenguaje MySQL la podéis encontrar en la web de documentación de MySQL, aquí. A continuación se dan las nociones básicas de las sentencias básicas del lenguaje DDL.

Creación de tablas

```
CREATE TABLE nombreTabla (
    columna1 tipoDato,
    columna2 tipoDato,
    ...
    columnaN tipoDato)
```

Para crear la tabla Alumno:

Tabla Alumno.png

```
CREATE TABLE Alumno (
  DNI CHAR(9),
  Nombre VARCHAR(20),
  Apellidos VARCHAR(50),
  Fecha_nacimiento DATE,
  Email VARCHAR(50)
);
```

Claves primarias

Siguiendo con el ejemplo de la tabla Alumno, la clave primaria está formada por un solo campo, por lo que se puede poner directamente en la columna que es la clave primaria la palabra clave **PRIMARY KEY**:

```
CREATE TABLE Alumno (
  DNI CHAR(9) primary key,
  Nombre VARCHAR(20),
  Apellidos VARCHAR(50),
  Fecha_nacimiento DATE,
  Email VARCHAR(50)
);
```

Si la clave primaria está formada por varias columnas, se debe poner al final, después de la declaración de todas las columnas:

Tabla Matricula.png

```
CREATE TABLE Matricula (
  Nota FLOAT,
  Codigo CHAR(5),
  DNI CHAR(9),
  PRIMARY KEY (Codigo, DNI)
);
```

Claves ajenas

La tabla Matricula tiene dos claves ajenas, en la columna Codigo y en la columna DNI.
El diagrama de las tablas Alumno, Matricula y Asignatura:

DDL Claves ajenas.png

Se crea con las sentencias SQL siguientes:

```
CREATE TABLE Alumno (
  DNI CHAR(9),
  Nombre VARCHAR(20),
  Apellidos VARCHAR(50),
  Fecha_nacimiento DATE,
  Email VARCHAR(50),
  PRIMARY KEY (DNI)
);

CREATE TABLE Asignatura (
  Codigo CHAR(5),
  Nombre VARCHAR(30),
  Duracion VARCHAR(9),
  Horas_semanales INT,
  Codigo_grado CHAR(5),
  PRIMARY KEY (Codigo)
);

CREATE TABLE Matricula (
  Nota FLOAT,
  Codigo CHAR(5),
  DNI CHAR(9),
  PRIMARY KEY (Codigo, DNI),
  FOREIGN KEY (Codigo) REFERENCES Asignatura(Codigo),
  FOREIGN KEY (DNI) REFERENCES Alumno(DNI)
);
```

La clave ajena se indica después de todas las columnas, se pone FOREIGN KEY, a continuación entre paréntesis el nombre de la columna o columnas que forman parte de la clave ajena, luego la palabra REFERENCES y el nombre de la tabla con el de la columna a la que hace referencia como clave primaria, entre paréntesis.

Algunas restricciones de integridad

- **Prohibir nulos**. Esto obliga a que la columna tenga que tener obligatoriamente un valor para que sea almacenado el registro. Para esto, basta con añadir en la columna que no se permite dejar vacía en ninguna fila, después del tipo de dato, las palabras claves NOT NULL.

```
CREATE TABLE Alumno(
  DNI CHAR(9) NOT NULL,
  Nombre VARCHAR(20) NOT NULL,
  Apellidos VARCHAR(50) NOT NULL,
  Fecha_nacimiento DATE NOT NULL,
  Email VARCHAR(50) NOT NULL,
  PRIMARY KEY (DNI)
);
```

- **Valores no repetidos.** Esto obliga a que el contenido de una o más columnas no puedan repetir valores en la tabla. Basta con añadir al final de la columna la palabra clave UNIQUE.

  Si no permitimos que dos grados tengan el mismo nombre:

```
CREATE TABLE Grado(
  Codigo CHAR(5),
  Nombre VARCHAR(50) UNIQUE,
  PRIMARY KEY (Codigo)
);
```

  Puede ser que los valores que no se permiten repetir sea la combinación de varios campos. En ese caso se debe poner al final de la declaración de todas las columnas, igual que se indicaba la clave primaria.

Eliminación de tablas

La orden **DROP TABLE** seguida del nombre de una tabla, permite eliminar la tabla en cuestión.

```
DROP TABLE Grado;
```

Normalmente, el borrado de una tabla es irreversible y no hay ninguna petición de confirmación, por lo que conviene ser muy cuidadoso con esta operación.

Modificación de tablas

- Cambiar el nombre a una tabla.

```
ALTER TABLE nombre_viejo RENAME nombre_nuevo;
```

- Añadir columnas.

```
ALTER TABLE Alumno ADD (Direccion VARCHAR(100));
```

- Eliminar columnas.

```
ALTER TABLE Alumno DROP (Direccion);
```

- Modificar una columna. Basta con volver a definirla mediante la siguiente sintaxis:

```
ALTER TABLE Alumno Modify (Email VARCHAR(100));
```

Se ha cambiado el tipo de dato de la columna email y además ahora se permite dejar en blanco.

> Se pueden también añadir y eliminar restricciones, pero esto y más se puede ver en la documentación oficial de SQL.

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

Módulo 3. Implementación del modelo relacional en un SGBD

Lenguaje DML

El lenguaje de manipulación de datos (Data Manipulation Language) permite indicar al sistema las operaciones que se quieren realizar con los datos almacenados en las estructuras creadas por medio de las sentencias DDL. Por ejemplo son las sentencias que permiten: generar consultas, ordenar, filtrar, añadir, modificar, borrar, etc.

Inserción de datos

La inserción de datos a una tabla se realiza mediante la instrucción **INSERT**. Su sintaxis fundamental es:

INSERT INTO tabla (listaDeColumnas) VALUES (valor1,valor2, ...)

La tabla representa la tabla a la que se quiere añadir el registro y los valores que siguen a VALUES son los valores que se dan a los distintos campos del registro. Si no se especifica la lista de campos, la lista de valores debe seguir el orden de las columnas según fueron creados.

La lista de campos a rellenar se indica si no se quieren rellenar todos los campos. Los campos no rellenados explícitamente con la orden INSERT, se rellenan con su valor por defecto o bien se dejan vacíos (valor nulo) si no se indica valor alguno. Si algún campo tiene restricción de obligatoriedad (NOT NULL), ocurrirá un error si no rellenamos el campo con algún valor.

Inserción de un registro en la tabla Alumno:

```
INSERT INTO ALUMNO VALUES ('25458982A','Juan','Pérez
Gómez','01/06/2003','jperezgomez@gmail.com');
```

Actualización de registros

La modificación de los datos de los registros lo implementa la instrucción UPDATE. Sintaxis:

UPDATE tabla SET columna1=valor1 ,columna2=valor2,... WHERE condición

Se modifican las columnas indicadas en el apartado SET con los valores indicados. La cláusula WHERE permite especificar qué filas serán modificados, aquellas que cumplan la condición indicada.

Modificación de la fecha de nacimiento del registro insertado en el ejemplo anterior:

```
UPDATE ALUMNO SET fecha_nacimiento='01/06/1999' where DNI LIKE '25458982A';
```

Borrado de registros

Se realiza mediante la instrucción DELETE:

DELETE FROM tabla WHERE condición

Elimina todas las filas de la tabla que cumplan la condición indicada.

Borrado del registro insertado antes:

```
DELETE FROM ALUMNOS WHERE DNI LIKE '25458982A';
```

> Con la sentencia DELETE hay que tener cuidado. Si se ejecuta la sentencia sin la cláusula WHERE (sin ninguna condición), se elimina todo el contenido de la tabla, es decir, se vacía la tabla.

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

Módulo 3. Implementación del modelo relacional en un SGBD

Generación del script SQL mediante ERD Plus

Teniendo el diagrama relacional en la herramienta ERD Plus, se puede generar el SQL de creación de las tablas, con sus restricciones.

Hay que clicar en el elemento “Documents” del menú principal y se abren todos los diseños que tenemos creados en nuestra cuenta: ERD Plus Documents.png

Clicando en los 3 puntos de la derecha del modelo que se desea exportar en SQL, se despliega un menú en el que hay que seleccionar la opción “Generate SQL”.

ERD Plus Generate SQL 2.png

Pinchando en “Generate SQL” se abre una ventana con las sentencias SQL que al ejecutarlas en un sistema gestor de base de datos, creará las tablas con las restricciones del modelo diseñado.

ERD Plus SQL.png

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

Módulo 4. Consultas básicas SQL

Módulo 4. Consultas básicas SQL

Consultas básicas de selección

La sentencia SELECT

El comando SELECT permite:

- Obtener datos de ciertas columnas de una tabla.
- Obtener filas de una tabla de acuerdo con ciertos criterios.
- Realizar cálculos sobre los datos.
- Mezclar datos de tablas diferentes.
- Agrupar datos.

En este curso, vamos a quedarnos en lo más sencillo, que son las opciones 1, 2 y 3.

En su forma más simple, la sintaxis para la expresión SELECT es:

```
SELECT campos FROM tablas [WHERE condiciones]
```

- **campos:** son las columnas (separadas por comas) o cálculos que se desea recuperar de la base de datos. Se puede utilizar * si se desean recuperar todas las columnas.
- **tablas:** indica las tablas de las que se desean recuperar los registros. Mínimo debe haber una tabla para poder recuperar registros.
- **condiciones:** las condiciones que deben cumplir las filas que se desean recuperar.

La sentencia más sencilla sería la de recuperar todos los campos de una tabla:

```
SELECT *
FROM alumnos;
```

Si se desea recuperar solo DNI, nombre y apellidos de los alumnos, entonces se pondría:

```
SELECT DNI, nombre, apellidos
FROM alumnos;
```

Condiciones

La cláusula **WHERE** sirve para restringir los datos de salida. Permite poner una o varias condiciones que han de cumplir las filas que se desean recuperar, las que no cumplan esas condiciones no aparecerán en el resultado de la consulta.

Por ejemplo, si solo se desea mostrar los datos de las asignaturas que tengan una duración mayor de 2 horas a la semana. Sería como sigue:

```
SELECT *
FROM Asignatura
WHERE Horas_semanales > 2;
```

Se van a explicar a continuación los operadores básicos que se pueden utilizar en la cláusula WHERE.

Operadores de comparación

Operador	Significado
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
=	Igual
<>	Distinto
!=	Distinto

Valores lógicos

Operador	Significado
AND	Devuelve verdadero si las expresiones de su izquierda y derecha son ambas verdaderas.
OR	Devuelve verdadero si cualquiera de las dos expresiones a izquierda y derecha del OR, son verdaderas.
NOT	Invierte la lógica de la expresión que está a su derecha. Si es verdadera, mediante NOT pasa a ser falsa.

Por ejemplo, si se desea mostrar los datos de las asignaturas que tengan una duración mayor de 2 horas a la semana pero menor de 6. Sería:

```
SELECT *
FROM Asignatura
WHERE Horas_semanales>2 AND Horas_semanales<6;
```

LIKE

Se usa sobre todo con textos, permite obtener registros cuyo valor en un campo cumpla una condición textual. LIKE utiliza una cadena que puede contener estos símbolos:

Símbolo	Significado
%	Una serie cualquiera de caracteres
_	Un carácter cualquiera

Por ejemplo, mostrar los nombres de los alumnos que empiecen por S:

```
SELECT nombre
FROM alumnos
WHERE nombre LIKE 'S%';
```

IS NULL

Devuelve verdadero si el valor que examina es nulo. Los campos tienen valor nulo cuando al insertar una fila, ese campo se deja vacío.

La siguiente sentencia devuelve el nombre y apellidos de los alumnos que no tienen email.

```
SELECT nombre, apellidos
FROM alumno
WHERE email IS NULL
```

> Hay otros operadores como **BETWEEN** o **IN**, que no se van a explicar en este curso.

ORDENACIÓN

El orden inicial de los registros obtenidos por una SELECT es el orden en el que fueron introducidos. Para ordenar basándose en otros criterios, se utiliza la cláusula **ORDER BY**.

En esa cláusula se coloca una lista de campos que indica la forma de ordenar. Se ordena primero por el primer campo de la lista, si hay coincidencias por el segundo, si ahí también las hay por el tercero, y así sucesivamente.

Se puede colocar la palabra **ASC** O **DESC** (por defecto, si se indica nada, utiliza ASC).

Por ejemplo, la siguiente sentencia muestra todos los datos de los alumnos que no tienen email, ordenados alfabéticamente por el nombre:

```
SELECT nombre, apellidos FROM alumno
WHERE email IS NULL
ORDER BY nombre ASC;
```

Contar filas

Existe una función que sirve para contar filas, es la función **count**. Sirve para contar filas. Vamos a verlo con algún ejemplo.

La siguiente consulta cuenta cuántos alumnos hay en la tabla alumno:

```
SELECT count(*)
from alumno;
```

Si queremos contar cuántos alumnos hay cuyo apellido empiece por A:

```
SELECT count(*)
from alumno
where apellidos like 'A%';
```

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

Módulo 4. Consultas básicas SQL

Consultas calculadas

CONSULTAS CALCULADAS

Cálculos aritméticos

En las consultas se pueden utilizar los operadores + (suma), - (resta), * (multiplicación) y / (división) para hacer cálculos. Cuando se utilizan como expresión en una consulta SELECT, no modifican los datos originales sino que como resultado de la vista generada por la consulta, aparece una nueva columna.

Ejemplo:

```
SELECT dni, nota, nota*2
FROM alumnos;
```

Esa consulta obtiene tres columnas: el dni, la nota y la nota multiplicada por 2 . La tercera tendrá como nombre la expresión utilizada, para poner un alias basta utilizar dicho alias tras la expresión **as**:

```
SELECT dni, nota, nota*2 as nota_doble
FROM alumnos;
```

La prioridad de esos operadores es la normal: tienen más prioridad la multiplicación y división, después la suma y la resta. En caso de igualdad de prioridad, se realiza primero la operación que esté más a la izquierda. Se puede evitar cumplir esa prioridad usando paréntesis; el interior de los paréntesis es lo que se ejecuta primero.

Cuando una expresión aritmética se calcula sobre valores NULL, el resultado es el mismo valor NULL.

Concatenación de textos

Todas las bases de datos incluyen algún operador para encadenar textos. En SQLSERVER es el signo + en Oracle son los signos || y en MySQL se hace con la función **CONCAT** .

Ejemplo:

```
SELECT dni, concat(apellidos,', ',nombre), email
FROM alumnos;
```

El resultado de esa consulta es una tabla con tres columnas: la primera el DNI, la segunda columna es la concatenación de los apellidos con una coma y un espacio y luego el nombre, y la tercera columna el email.

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

Módulo 4. Consultas básicas SQL

Consultas básicas sobre una base de datos en Coding Rooms

Aquí van unos pocos ejemplos de consultas básicas sobre una base de datos, para practicar sobre la herramienta Coding rooms.

Antes de nada, aquí puedes descarar la base de datos para ejecutarla en la herramienta que más abajo se explica paso a paso cómo hacerlo.

La base de datos es sobre recetas, tiene el siguiente diagrama relacional:

recetas relacional.png

Coding Rooms

Para utilizar Coding Rooms no se necesario registrarse si se accede directamente al workspace desde este enlace. Pero si te registras se guardan los workspace que vayas utilizando, por lo que yo lo recomiendo.

Para iniciar el compilador solo hay que pulsar sobre el botón "Start coding MySQL for free now".

Coding Rooms start.png

Esto abre el Workspace de MySQL siguiente.

Workspace codingrooms mysql.png

Lo primero que tenemos que hacer es ejecutar el script descargado para crear la base de datos de recetas. Para eso, en el área de navegación de archivos, clicar sobre la flecha hacia arriba para subir el archivo.

coding rooms uploadfile.png

El archivo se carga y haciendo clic sobre él se abre en el escritorio. Ahora solo hay que desplegar los archivos del botón de ejecución y seleccionar el archivo que se desea ejecutar.

Ejecutar script.png

El resultado de la ejecución se carga en la consola de debajo. En el caso del script de creación de la base de datos de recetas, como no son consultas, no muestra nada por pantalla, así que si no muestra ningún error, es que todo ha ido bien.

image-1662366980077.png

Con esto ya deberíamos tener la base de datos creada en el servidor de coding rooms para empezar a hacer consultas.

Consultas SQL sencillas

En el archivo que podéis descargar aquí, hay ejemplos de consultas que a continuación se explican.

Ejecutar consultas en Coding Rooms

Para ejecutar cada consulta en Coding Rooms, hay que poner la consulta en un archivo y ejecutar ese archivo.

Para crear un archivo nuevo hay que pulsar en el + del navegador de ficheros:

image-1662367796199.png

En ese nuevo archivo podéis ir escribiendo cada consulta, y el resultado se mostrará en la consola.

Ejemplos del archivo

1. Mostrar todos los tipos de recetas que hay en la base de datos:

```
select * from tipo_receta;
```

La consola muestra el resultado:

image-1662368642075.png

2. Mostrar el nombre de todas las recetas que no tienen anotaciones (en el campo notas)

```
select nombre from receta where notas is null;
```

Resultado:

Consulta ejemplo 2.png

3. Queremos saber cómo se prepara la "Ensalada de verano"

```
select preparacion from receta where nombre like 'Ensalada de verano';
```

Resultado:

Consulta ejemplo 3.png

Ahora vamos a ver cómo sacaríamos información que hay en varias tablas sin haber visto las consultas multitabla.

4. Queremos saber qué recetas llevan 'Ternera'.

Los ingredientes están en la tabla **ingrediente**, la unión de recetas con ingredientes está en la tabla **receta_tiene_ingrediente**, pero el nombre de las recetas está en la tabla **receta**.

Primero hay que buscar cuál es el identificador del ingrediente 'Ternera' para buscarlo en la tabla **receta_tiene_ingrediente**. Así esta tabla nos dará los identificadores de las recetas que llevan ternera y con ese identificador podremos ir a buscar las recetas en la tabla **receta**.

Identificador del ingrediente 'Ternera':

```
select id_ingrediente from ingrediente where nombre like 'Ternera';
```

Resultado:

Consulta ejemploe 4_1.png

Identificador de las recetas que llevan 'Ternera':

```
select id_receta from receta_tiene_ingrediente where id_ingrediente=1;
```

Resultado:

Consulta ejemplo 4_2.png

Recetas que llevan 'Ternera':

```
select nombre from receta where id_receta=9 or id_receta=1;
```

Resultado:

Consulta ejemplo 4_3.png

5. Contar cuántas recetas tienen anotaciones (en el campo notas).

```
select count(*) from receta where notas is not null;
```

Resultado:

Consulta ejemplo 5.png

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

Módulo 4. Consultas básicas SQL

Un ejemplo de consulta multitabla

Vamos a ver cómo sacaríamos los nombres de las recetas que llevan ternera usando solo una consulta sobre arias tablas a la vez.

La manera de hacerlo consultando las tablas por separado

Los ingredientes están en la tabla **ingrediente**, la unión de recetas con ingredientes está en la tabla **receta_tiene_ingrediente**, pero el nombre de las recetas está en la tabla **receta**.

Primero hay que buscar cuál es el identificador del ingrediente 'Ternera' para buscarlo en la tabla **receta_tiene_ingrediente**. Así esta tabla nos dará los identificadores de las recetas que llevan ternera y con ese identificador podremos ir a buscar las recetas en la tabla **receta**.

Identificador del ingrediente 'Ternera':

```
select id_ingrediente from ingrediente where nombre like 'Ternera';
```

Identificador de las recetas que llevan 'Ternera':

```
select id_receta from receta_tiene_ingrediente where id_ingrediente=1;
```

Recetas que llevan 'Ternera':

```
select nombre from receta where id_receta=9 or id_receta=1;
```

Consulta multitabla

El modo básico de hacerlo es poniendo en la cláusula **from** las tablas separadas por comas, y en la cláusula **where** indicar cómo se relacionan esas tablas, es decir, indicando qué columnas son las que deben ser iguales en cada fila.

En el ejemplo anterior sería:

```
SELECT receta.nombre
FROM ingrediente, receta_tiene_ingrediente, receta
WHERE ingrediente.id_ingrediente = receta_tiene_ingrediente.id_ingrediente
and receta_tiene_ingrediente.id_receta = receta.id_receta
    and ingrediente.nombre like 'Ternera';
```

> Fíjate que se están indicando las columnas con el nombre de la tabla delante seguidos por un punto. Esto es necesario cuando en una consulta multitabla hay columnas en diferentes tablas con el mismo nombre, como es este caso, para saber a la columna de qué tabla se está haciendo referencia.

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

Créditos

Curso creado en Octubre de 2022 por:

Berta Besteiro

Cualquier observación o detección de error en soporte.catedu.es

image-1648462225402.gif

image-1648462299882.png

image-1648462361893.png

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png