CURSO DE PROGRAMACIÓN FULL STACK

BASES DE DATOS CON MYSQL

GUÍA DE BASE DE DATOS

BASE DE DATOS

Una base de datos es un conjunto organizado de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. Las bases de datos son utilizadas para modelar un aspecto relevante de la realidad. En las bases de datos los datos se almacenan de manera permanente, para luego ser procesados y transformados en información.

DBMS (DATA BASE MANAGER SYSTEM)

Existen programas denominados sistemas gestores de bases de datos (DBMS por sus siglas en inglés), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Un DBMS es una colección de software muy específico, cuya función es servir de interfaz entre la base de datos, el usuario y las distintas aplicaciones utilizadas.

FUNCIONES DEL DBMS

- Metadatos: la definición o información descriptiva de una base de datos, relacionada con su estructura y los datos que contiene, también se almacenan; y es lo que se conoce como metadatos.

- Construcción de la base de datos: es el proceso consistente en almacenar los datos en algún medio de almacenamiento controlado por el DBMS.

- Manipulación de una base de datos: son aquellas funciones como la consulta de la base de datos para recuperar datos específicos, actualizar la base de datos para reflejar los cambios introducidos y generar informes a partir de los datos.

- Compartir una base de datos: permite que varios usuarios y programas accedan a la base de datos de forma simultánea.

- Consultas: una aplicación accede a la base de datos enviando consultas o solicitudes de datos al DBMS. Una consulta normalmente provoca la recuperación de algunos datos.

- Transacciones: una transacción puede provocar la lectura o la escritura de algunos datos en la base de datos.

- Protección: incluye la protección del sistema contra el funcionamiento defectuoso del hardware o el software (caídas) y la protección de la seguridad contra el acceso no autorizado o malintencionado.

- Mantenimiento: una gran base de datos típica puede tener un ciclo de vida de muchos años, por lo que el DBMS debe ser capaz de mantener el sistema de bases de datos permitiendo que el sistema evolucione según cambian los requisitos con el tiempo.

¿POR QUÉ INTERESA USAR UNA BASE DE DATOS?

- Mayor independencia. Los datos son independientes de las aplicaciones que los usan, así como de los usuarios.

- Mayor disponibilidad. Se facilita el acceso a los datos desde contextos, aplicaciones y medios distintos, haciéndolos útiles para un mayor número de usuarios.

- Mayor seguridad (protección de los datos). Por ejemplo, resulta más fácil replicar una base de datos para mantener una copia de seguridad que hacerlo con un conjunto de ficheros almacenados de forma no estructurada. Además, al estar centralizado el acceso a los datos, existe una verdadera sincronización de todo el trabajo que se haya podido hacer sobre estos (modificaciones), con lo que esa copia de seguridad servirá a todos los usuarios.

- Menor redundancia. Un mismo dato no se encuentra almacenado en múltiples archivos o con múltiples esquemas distintos, sino en una única instancia en la base de datos. Esto redunda en menor volumen de datos y mayor rapidez de acceso.

- Mayor eficiencia en la captura, codificación y entrada de datos.

CLASIFICACIÓN DE LAS BASES DE DATOS

1. De acuerdo a cómo se modelan los datos:

    - Relacional: representa a la base de datos como una colección de tablas. Estas bases de datos suelen utilizar SQL como lenguaje de consultas de alto nivel.
    - Orientado a objetos: define a la base de datos en términos de objetos, sus propiedades y sus operaciones. Todos los objetos que tienen la misma estructura y comportamiento pertenecen a una clase y las clases se organizan en jerarquías.
    - Objeto-relacional o relacional extendido: son los sistemas relacionales con características de los orientado a objetos.
    - Jerárquico: representa los datos como estructuras jerárquicas de árbol

2. De acuerdo al número de usuarios a los que da servicio:

    - Monousuario
    - Multiusuario

MODELO RELACIONAL

El modelo relacional, para el modelado y la gestión de bases de datos, es un modelo de datos basado en la lógica de predicados y en la teoría de conjuntos. Su idea fundamental es el uso de relaciones. Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados tuplas. En general, se puede pensar en cada relación como si fuese una tabla que está compuesta por registros (cada fila de la tabla sería un registro o "tupla") y columnas (también llamadas "campos"). El modelo relacional es el más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente.

ENTIDAD

El elemento básico representado por el modelo entidad relación es una entidad, que es un objeto del mundo real con una existencia independiente. Una entidad puede ser un elemento con una existencia física (por ejemplo, una persona en particular, un coche, una casa o un empleado) o puede ser un elemento con una existencia conceptual (por ejemplo, una venta, un trabajo o un curso universitario). Cada entidad tiene atributos (propiedades particulares que la describen).

ATRIBUTOS

Un atributo es una abstracción que identifica características, propiedades que posee una entidad. Los atributos de una entidad deben ser:

- Completos: capturar toda la información que interesa del objeto, desde el punto de vista del sistema.
- Plenamente elaborados: cada atributo captura un aspecto separado de la entidad.
- Mutuamente independientes: cada atributo debe tomar un valor independientemente de los valores asumidos por otros atributos.

IDENTIFICADOR ÚNICO

Se denomina identificador a uno o más atributos que identifican unívocamente cada instancia de una entidad; es conocido también como "clave candidata". Es decir, nunca puede existir dos instancias de una entidad con el mismo valor de su atributo identificador.

El o los atributos identificadores se señalan con el símbolo "@"(arroba), o de lo contrario con la sigla PK (clave primaria).

Para mejorar el desempeño de la base de datos se recomienda utilizar identificadores numéricos; por lo tanto, si una entidad no posee un atributo identificador numérico, se debería agregar un atributo, comúnmente llamado id (abreviación de identificador) seguido por el nombre de la entidad.

RELACIONES

Una relación es la abstracción de un conjunto de asociaciones que existen entre las instancias de dos entidades, por ejemplo, existe una relación entre Película y PaisDeOrigen.

- Las relaciones tienen sentido bidireccional.

- Las relaciones existen ya que las entidades representan aspectos del mundo real y en este mundo los componentes no están aislados, sino que se relacionan entre sí; es por esto que es necesario que existan las relaciones entre las entidades.

LENGUAJE DE CONSULTA ESTRUCTURADO SQL

SQL es un lenguaje que se usa para operar su base de datos. SQL es el lenguaje básico utilizado para todas las bases de datos. De acuerdo con ANSI (American National Standards Institute), SQL es el lenguaje estándar para operar un sistema de administración de bases de datos relacionales.

SQL se usa para acceder, actualizar y manipular datos en una base de datos. Su diseño permite la gestión de datos en un RDBMS, como MYSQL. El lenguaje SQL también se usa para controlar el acceso a datos y para la creación y modificación de esquemas de Base de datos. SQL utiliza los términos tabla, fila y columna para los términos relación, tupla y atributo del modelo relacional formal, respectivamente. Por lo tanto, es posible utilizar todos estos términos indistintamente.

Existen dos tipos de comandos SQL:

1. Lenguaje de Definición de Datos (DDL): permite crear y definir nuevas bases de datos, campos e índices.

    - CREATE: Crea nuevas tablas, campos e índices.
    - DROP: Elimina tablas e índices.
    - ALTER: Modifica las tablas agregando campos o cambiando la definición de los campos.

2. Lenguaje de Manipulación de Datos (DML): permiten generar consultas para ordenar, filtrar y extraer datos de la base de datos.

    - SELECT: Consulta registros de la base de datos que satisfagan un criterio determinado.
    - INSERT: Carga lotes de datos en la base de datos en una única operación.
    - UPDATE: Modifica los valores de los campos y registros especificados.
    - DELETE: Elimina registros de una tabla de una base de datos.

PASOS PARA IMPLEMENTAR UNA BASE DE DATOS

PASO	Descripción
1	Definir en el disco duro, el área física que contendrá las tablas de la base de datos. Sentencia SQL --> CREATE DATABASE.
2	Crear las diferentes tablas de la base de dato. Sentencia SQL --> CREATE TABLE
3	Insertar las filas de las diferentes tablas, sin violar la integridad de datos. Sentencia SQL --> INSERT INTO .
4	Actualizar los datos que cambien con el tiempo en las diferentes tablas. Sentencia SQL --> UPDATE.
5	Eliminar las filas que ya no se requieran en las diferentes tablas. Sentencia SQL --> DELETE
6	Realizar las consultas deseadas a las tablas de la base de datos a través de la poderosa sentencia de consultas del SQL, llamada SELECT.
7	Dar nombre a las consultas. elaboradas en el paso No.6 cuando se requiera ocultar el diseño y columnas de las tablas a través de la creación de vistas lógicas. Sentencia SQL ----> CREATE VIEW.

1. CREATE DATABASE

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] nombre_base_datos
```

- Esta sentencia sirve para crear una base de datos con un nombre específico.
- Para poder crear una base de datos, el usuario que la crea debe tener privilegios de creación asignados.
- IF NOT EXISTS significa: SI NO EXISTE, por lo tanto, esto es útil para validar que la base de datos sea creada en caso de que no exista, si la base de datos existe y se ejecuta esta sentencia, se genera un error.
- CREATE SCHEMA o CREATE DATABASE son sinónimos.

2. CREATE TABLE

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] nombre_de_tabla
(
campo1 tipo dato [NULL/NOT NULL] | CHECK (expresiónLógica) | [
DEFAULT expresiónConstante],
campo2 tipo dato [NULL/NOT NULL] | CHECK (expresiónLógica) | [
DEFAULT expresiónConstante ],
campo-N,
PRIMARY KEY(campo_llave),
FOREIGN KEY (campo_llave) REFERENCES tabla2 (campo_llave-tabla2)
)
```

Ligaduras

Tipo: Integridad de Dominio o Columna

Especifica un conjunto de valores que son válidos a ingresar sobre una columna específica para una tabla de la base de datos. Esta integridad se verifica a través de una la validación de los valores de datos que se ingresan y el tipo de los datos a introducir (numérico, alfanumérico, alfabético, etc.).

- DEFAULT: Esta restricción asigna un valor específico a una columna cuando el valor para ello no haya sido explícitamente proporcionado para tal columna en una sentencia "INSERT" o de adición de un nuevo registro en la tabla.

- CHECK: Específica los valores de datos que el DBMS acepta le sean ingresados para una columna.

- REFERENCES: Específica los valores de datos que el DBMS acepta le sean ingresados para una columna.

Tipo: Integridad de Entidad o Tabla

Específica que, en una tabla o entidad, todas sus filas tengan un identificador único que diferencie a una fila de otra y también que se establezcan columnas cuyo contenido es un valor único que las hace llaves candidatas para un futuro como por ejemplo: número de cédula, número de seguro social o cuenta de email.

- PRIMARY KEY: Este tipo de restricción se aplica a todas las filas permitiendo que exista un identificador, que se conoce como llave primaria y que se asegura que los usuarios no introduzcan valores duplicados. Además, asegura que se cree un índice para mejorar el desempeño. Los valores nulos no están permitidos para este tipo de restricción.

- UNIQUE: Con esta restricción se previene la duplicación de valores en columnas que tienen valor único y que no son llave primaria pero que pueden ser una llave alternativa o candidata para el futuro. Asegura que se cree (Por parte del DBMS) un índice para mejorar el desempeño. Y al igual que las llaves primarias, no se le está permitido que se introduzcan valores nulos.

Tipo: Integridad Referencial

La Integridad Referencial asegura que las relaciones que existe entre llave primaria (en la tabla referenciada) y la llave foránea (en las tablas referenciantes) serán siempre mantenidas. Una fila o registro en la tabla referenciada (tabla donde reside la llave primaria) no puede ser borrada o su llave primaria cambiada si existe una fila o registro con una llave foránea (en la tabla referenciante) que se refiere a esa llave primaria.

- FOREIGN KEY: En esta restricción se define una columna o combinación de columnas en las cuales su valor debe corresponder al valor de la llave primaria en la misma u en otra tabla.

OTRAS SENTENCIAS RELACIONADAS CON TABLAS:

Eliminación de Tablas:

La sentencia para eliminar una tabla y por ende todos los objetos asociados con esa tabla es DROP TABLE, donde r es el nombre de una tabla existente.

```
DROP TABLE r
```

Modificación de Tablas

Después que una tabla ha sido utilizada durante algún tiempo, los usuarios suelen descubrir que desean almacenar información adicional con respecto a las tablas. La sentencia ALTER TABLE se utiliza sobre tablas que ya poseen desde cientos a miles de filas por ser tablas de un sistema de

Base de Datos que ya está en producción.

```
ALTER TABLE nombre_tabla acción
```

Siendo acción una de las siguientes:

- RENAME TO nuevo_nombre
- ADD [COLUMN] nombre_atributo definición_atributo
- DROP [COLUMN] nombre_atributo
- MODIFY nombre_atributo definición_atributo
- CHANGE nombre_atributo nuevo_nombre nueva_definición
- ALTER COLUMN nombre_atributo nuevo_nombre nueva_definición

Los cambios que se pueden realizar con la sentencia SQL ALTER TABLE son:

- Añadir una definición de columna a una tabla. Puede crearse con valores nulos o con valores.

- Eliminar una columna de la tabla. Pero antes de su eliminación deben ser eliminados por ALTER TABLE todas las restricciones que estén definidas sobre esta columna.

- Eliminar la definición de: llave primaria, foránea o restricciones de ligaduras de integridad (check), existentes para una tabla. Esta acción no elimina a la columna con sus valores, ella permanece tal cual como está, solo se elimina su definición.

- Definir una llave primaria para una tabla. La columna(s) a la cual se le dará esta responsabilidad debe contener previamente valores únicos por fila.

- Definir una nueva llave foránea para una tabla. La columna a definir como llave foránea debe contener previamente valores que corresponden a la llave primaria de otra tabla.

3. INSERT INTO

En su formato más sencillo, INSERT se utiliza para añadir una sola fila a una tabla. Debemos
especificar el nombre de la tabla y una lista de valores para la fila. Los valores deben suministrarse en el mismo orden en el que se especificaron los atributos correspondientes en el comando CREATE TABLE.

```
INSERT INTO nombre_tabla (columna1, columna2, columna3,...) VALUES
(valor, valor2, valor3,...);
```

4. UPDATE

El comando UPDATE se utiliza para modificar los valores de atributo de una o más filas

seleccionadas. Una cláusula WHERE en el comando UPDATE selecciona las filas de una

tabla que se van a modificar. La sentencia UPDATE tiene la siguiente forma:

```
UPDATE nombre_tabla

SET col_nombre_1={valor1|DEFAULT} [,

col_nombre_2={valor2|DEFAULT}]

[ORDER BY ...] [WHERE condicion]
```

5. DELETE

El comando DELETE elimina filas de una tabla. Incluye una cláusula WHERE, para seleccionar las filas que se van a eliminar.

- Las filas se eliminan explícitamente sólo de una tabla a la vez. Sin embargo, la eliminación se puede propagar a filas de otras tablas si se han especificado opciones de acciones referenciales en las restricciones de integridad referencial del DDL.

- En función del número de filas seleccionadas por la condición de la cláusula WHERE, ninguna, una o varias filas pueden ser eliminadas por un solo comando DELETE. La ausencia de una cláusula WHERE significa que se borrarán todas las filas de la relación; sin embargo, la tabla permanece en la base de datos, pero vacía. Debemos utilizar el comando DROP TABLE para eliminar la definición de la tabla.

```
DELETE FROM nombre_tabla [WHERE condicion] [ORDER BY ...] [LIMIT
cantidad_filas]
```

6. SELECT

La sentencia SELECT es muy poderosa y ampliamente rica en sus cláusulas y variantes permitiendo la capacidad de atender en poco tiempo a consultas complejas sobre la base de datos. Está en el especialista desarrollador de aplicaciones conocerlo a profundidad para explotar las bondades y virtudes.

La sentencia SELECT, obtiene filas de la base de datos y permite realizar la selección de una o varias filas o columnas de una o varias tablas.

```
SELECT nombres de las columnas
[INTO nueva tablaDestino para resultados del select]
FROM tablaOrigen
[WHERE condición de Búsqueda]
[GROUP BY nombres de columnas por la cual Agrupar]
[HAVING condiciónBúsqueda para Group By ]
[ORDER BY nombre de columnas [ASC | DESC] ]
```

Cláusulas:

- SELECT: Se usa para listar las columnas de las tablas que se desean ver en el resultado de una consulta. Además de las columnas se pueden listas columnas a calcular por el SQL cuando actué la sentencia. Esta cláusula no puede omitirse.

- FROM: Lista las tablas que deben ser analizadas en la evaluación de la expresión de la cláusula WHERE y de donde se listarán las columnas enunciadas en el SELECT. Esta cláusula no puede omitirse.

- WHERE: establece criterios de selección de ciertas filas en el resultado de la consulta gracias a las condiciones de búsqueda. Si no se requiere condiciones de búsqueda puede omitirse y el resultado de la consulta serán todas las filas de las tablas enunciadas en el FROM.

- GROUP BY: especifica una consulta sumaria. En vez de producir una fila de resultados por cada fila de datos de la base de datos, una consulta sumaria agrupa todas las filas similares y luego produce una fila sumaria de resultados para cada grupo de los nombres de columnas enunciado en esta cláusula. En otras palabras, esta cláusula permitirá agrupar un conjunto de columnas con valores repetidos y utilizar las funciones de agregación sobre las columnas con valores no repetidas. Esta cláusula puede omitirse.

- HAVING: le dice al SQL que incluya sólo ciertos grupos producidos por la cláusula GROUP BY en los resultados de la consulta. Al igual que la cláusula WHERE, utiliza una condición de búsqueda para especificar los grupos deseados. La cláusula HAVING es la encargada de condicionar la selección de los grupos en base a los valores resultantes en las funciones agregadas utilizadas debidas que la cláusula WHERE condiciona solo para la selección de filas individuales. Esta cláusula puede omitirse.

- ORDER BY: permitirá establecer la columna o columnas sobre las cuales las filas resultantes de la consulta deberán ser ordenadas. Esta cláusula puede omitirse.

7. CREATE VIEW

Las vistas pueden considerarse como tablas virtuales. Generalmente hablando, una tabla tiene un conjunto de definiciones, y almacena datos físicamente. Una vista también tiene un conjunto de definiciones, que se construye en la patrte superior de la(s) tabla(s) u otra(s) vista(s), y no almacena datos físicamente. La sintaxis para la creación de una vista es la siguiente:

```
CREATE [OR REPLACE] VIEW nombre_vista AS

SELECT columna1, columna2, ...

FROM nombre_tabla

WHERE condiciones;
```

- OR REPLACE: es opcional y si no se especifica esta cláusula y la vista ya existe, la sentencia CREATE VIEW producirá un error.

- view_name: el nombre de la vista que se desea crear en MySQL.

- WHERE condiciones: es opcional. Las condiciones deben ser realizadas para los registros a ser incluidos en la vista.

PREGUNTAS DE APRENDIZAJE

1. Responda Verdadero (V) o Falso (F)

	V	F
Una primary key es la columna (columnas) que tiene datos completamente únicos a lo largo de la tabla.	( )	( )
La función principal de una clave primaria en una tabla es mantener su integridad.	( )	( )
Las foreign keys son campos que vinculan una tabla con la clave primaria o externa de otra tabla.	( )	( )
Una tabla no puede tener más de una clave foránea definida.	( )	( )
INSERT, UPDATE, GRANT, TRUNCATE y CREATE son commandos DDL.	( )	( )
El comando DROP se utiliza para eliminar todas las filas de una tabla.	( )	( )

2. En una cláusula LIKE, ¿cómo se obtienen todos los nombres de personas que tienen exactamente cuatro caracteres?

   a) LIKE "????"
   b) LIKE "____"
   c) LIKE "...."
   d) Las anteriores respuestas no son correctas

3. Una sentencia SELECT sin la cláusula WHERE devuelve

   a) Todos los registros existentes en la tabla que no estén relacionados con otra tabla
   b) Todos los registros existentes en la tabla
   c) No se puede ejecutar una sentencia SELECT sin la cláusula WHERE
   d) Las anteriores respuestas no son correctas

4. ¿Cuál de las siguientes no es una función de agregación?

   a) AVG()
   b) FLOOR()
   c) SUM()
   d) Las anteriores respuestas no son correctas

5. En una cláusula LIKE, ¿cómo se obtienen todos los nombres de personas que comienzan con "Juan"?

   a) LIKE "Juan%"
   b) LIKE "Juan*"
   c) LIKE "Juan$"
   d) LIKE "Juan&"

6. ¿Qué instrucción se emplea para eliminar todo el contenido de una tabla, pero conservando la tabla?

   a) DELETE TABLE
   b) DROP TABLE
   c) TRUNCATE TABLE
   d) Las anteriores respuestas no son correctas

7. En SQL, para ordenar los datos devueltos por una sentencia SELECT se emplea la cláusula

   a) ORDER BY
   b) ORDERED BY
   c) SORT BY
   d) SORTED BY

8. ¿En cuál de las siguientes sentencias del lenguaje SQL se emplea la cláusula SET?

   a) DELETE
   b) DROP
   c) SELECT
   d) UPDATE

9. En SQL, para modificar la estructura de una tabla de una base de datos se emplea la instrucción

   a) ALTER TABLE
   b) CHANGE TABLE
   c) MODIFY TABLE
   d) Las anteriores respuestas no son correctas

10. ¿Cuál de las siguientes no es una función de agregación?

    a) COUNT()
    b) LIMIT()
    c) MAX()
    d) MIN()

11. En SQL, ¿cuál de estas sentencias añade una fila a una tabla en una base de datos?

    a) ADD
    b) INSERT
    c) UPDATE
    d) INCLUDE

12. ¿Cómo se borra toda una base de datos con SQL?

    a) DELETE DATABASE
    b) DROP DATABASE
    c) ERASE DATABASE
    d) Las anteriores respuestas no son correctas

13. En SQL, para eliminar las filas duplicadas del resultado de una sentencia SELECT se emplea:

    a) NO DUPLICATE
    b) UNIQUE
    c) DISTINCT
    d) Las anteriores respuestas no son correctas

14. ¿Cuáles de las siguientes sentencias son ciertas sobre las vistas?

    a) Una vista representa un subconjunto de los atributos de una tabla y que puede ser diseñado para facilitar un caso en particular.
    b) El manejo de permisos y otras tareas administrativas es mucho más fácil a través de vistas que a través de tablas.
    c) Una vista es utilizada para recuperación rápida de datos.
    d) Una vista es una rápida descripción de una base de datos.

15. ¿Qué establece un primary key de una tabla?

    a) Integridad Referencial
    b) Integridad de los registros
    c) Integridad de las columnas
    d) Constrainsts de identidad

16. ¿Cuál de las siguientes sentencias es cierta sobre las relaciones?

    a) Las relaciones son entidades
    b) Las relaciones son enlaces lógicos entre las tablas implementadas a través de primary y foreign keys.
    c) Las relaciones son almacenadas como atributos en la base de datos.
    d) Las relaciones explícitamente definen una asociación entre 2 tablas.

EJERCICIOS DE APRENDIZAJE

PRIMEROS PASOS

1. Para llevar adelante la resolución de esta guía es necesario tener instalado el motor de base datos MySQL, junto con una herramienta visual de diseño de bases de datos llamada Workbench. Esta herramienta permite integrar el desarrollo de software, la administración de bases de datos, el diseño de bases de datos, y la gestión y mantenimiento para el sistema de base de datos MySQL.

   Los pasos a llevar a cabo son los siguientes:

   A. Instalar el paquete de Visual C++ Redistributable. Este paquete es necesario para poder instalar luego MySQL y se descarga a través del siguiente link:

      Visual C++ Redistributable

   B. Descargar e instalar MySQL y MySQL Workbench a través de los siguientes links:

      MySQL Installer

      MySQL Workbench

   Tenés problemas con la instalación?? Mirá los siguientes links:

   https://www.profesionalreview.com/2018/12/13/mysql-windows-10/

   https://www.youtube.com/watch?v=myo0ZhJXw1k

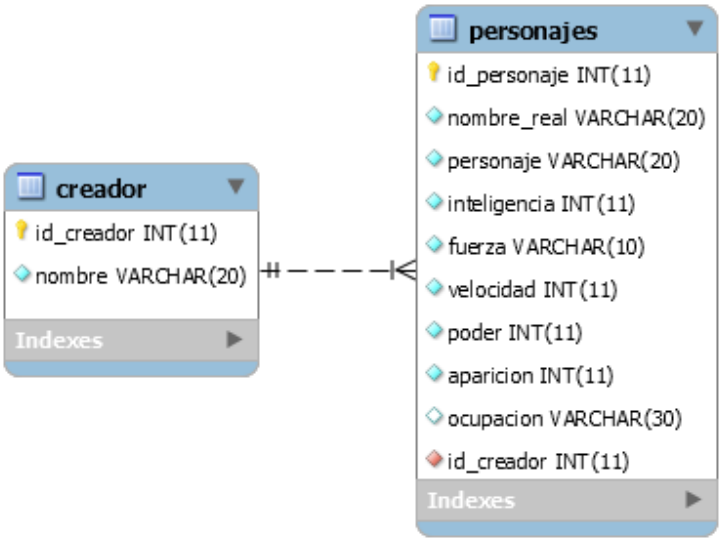
   https://www.youtube.com/watch?v=DxZfYDziQGE

   https://www.youtube.com/watch?v=EVdUYI6Wxw4

   Para la realización de los ejercicios que se describen a continuación, es necesario descargar el archivo scriptsBD.zip que contiene algunos scripts con las bases de datos sobre las cuales se va a trabajar. En cada ejercicio se indica el nombre del script que se debe utilizar. Ante alguna duda sobre las sentencias de MySQL se recomienda leer en el "**Apéndice C**".

VIDEO: Introducción Base de Datos

2. Abrir el script llamado "superheroes" y ejecutarlo de modo tal que se cree la base de datos y todas sus tablas. Después hacer las siguientes tareas:



VIDEO: Insertar, Modificar y Eliminar

a) Insertar en las tablas creadas los siguientes datos:

Tabla creador

id_creador	creador
1	Marvel
2	DC Comics

Tabla personajes

id_personaje	nombre_real	personaje	Inteligencia	fuerza	velocidad	poder	aparicion	ocupación	id_creador
1	Bruce Banner	Hulk	160	600 mil	75	98	1962	Físico Nuclear	1
2	Tony Stark	Iron Man	170	200 mil	70	123	1963	Inventor Industrial	1
3	Thor Odinson	Thor	145	infinita	100	235	1962	Rey de Asgard	1
4	Wanda Maximoff	Bruja Escarlata	170	100 mil	90	345	1964	Bruja	1
5	Carol Danvers	Capitana Marvel	157	250 mil	85	128	1968	Oficial de inteligencia	1
6	Thanos	Thanos	170	infinita	40	306	1973	Adorador de la muerte	1
7	Peter Parker	Spiderman	165	25 mil	80	74	1962	Fotógrafo	1
8	Steve Rogers	Capitan America	145	600	45	60	1941	Oficial Federal	1
9	Bobby Drake	Ice Man	140	2 mil	64	122	1963	Contador	1
10	Barry Allen	Flash	160	10 mil	120	168	1956	Científico forense	2
11	Bruce Wayne	Batman	170	500	32	47	1939	Hombre de negocios	2
12	Clarck Kent	Superman	165	infinita	120	182	1948	Reportero	2
13	Diana Prince	Mujer Maravilla	160	infinita	95	127	1949	Princesa guerrera	2

Una vez insertados todos los registros realizar una selección de todos los atributos para corroborar que la tablas se encuentren completas.

b) Cambiar en la tabla personajes el año de aparición a 1938 del personaje Superman. A continuación, realizar un listado de toda la tabla para verificar que el personaje haya sido actualizado.

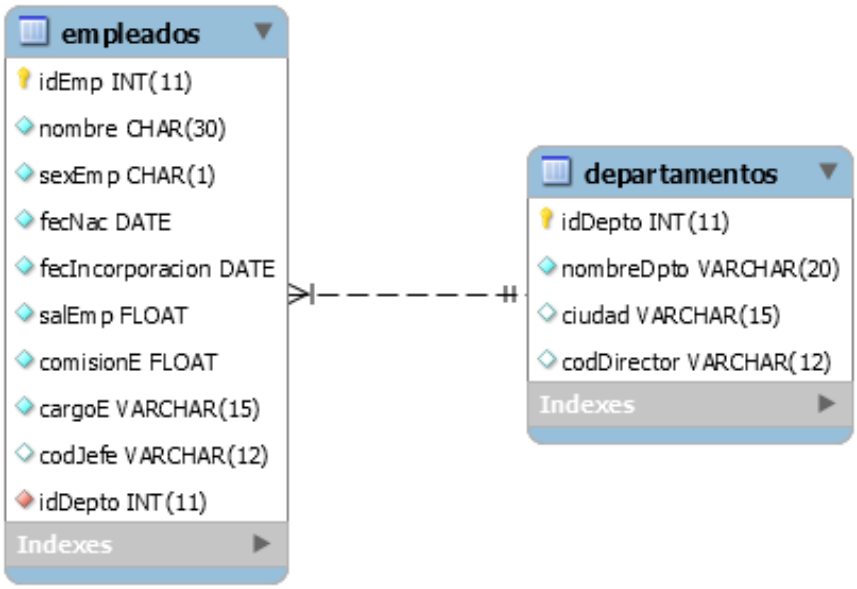
c) El registro que contiene al personaje Flash. A continuación, mostrar toda la tabla para verificar que el registro haya sido eliminado.

d) Eliminar la base de datos superhéroes.

VIDEO: Select. From. Where

Para realizar los siguientes ejercicios se recomienda leer en el Apéndice C el apartado: Tipos de JOIN representados a través de diagramas de Venn.

3. Abrir el script llamado “personal-inserts” y ejecutarlo de modo tal que se inserten todos los datos en las tablas creadas.



a) A continuación, realizar las siguientes consultas sobre la base de datos personal:

1. Obtener los datos completos de los empleados.
2. Obtener los datos completos de los departamentos.
3. Obtener los datos de los empleados cuyo cargo sea ‘Secretaria’.
4. Obtener el nombre y salario de todos los empleados.
5. Obtener los datos de los empleados vendedores, ordenados por nombre alfabéticamente.
6. Listar el nombre de los departamentos.

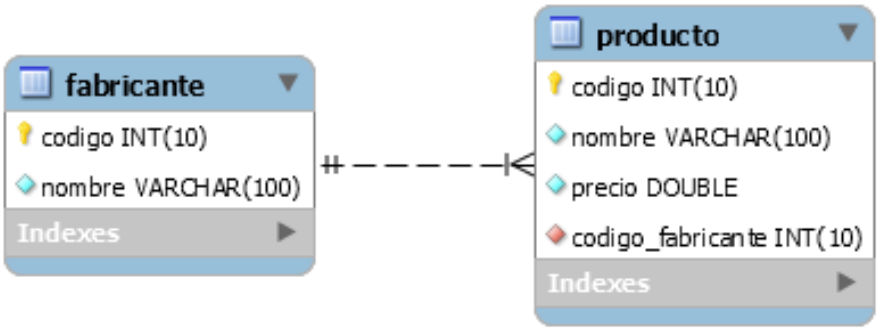
VIDEOS:

A) Tablas Relacionadas
B) Join Avanzados
C) Order by, Group by, Having

7. Obtener el nombre y cargo de todos los empleados, ordenados por salario de menor a mayor.
8. Listar los salarios y comisiones de los empleados del departamento 2000, ordenado por comisión de menor a mayor.
9. Listar todas las comisiones.
10. Obtener el valor total a pagar que resulta de sumar a los empleados del departamento 3000 una bonificación de 500, en orden alfabético del empleado.
11. Obtener la lista de los empleados que ganan una comisión superior a su sueldo.
12. Listar los empleados cuya comisión es menor o igual que el 30% de su sueldo.
13. Elabore un listado donde para cada fila, figure 'Nombre' y 'Cargo' antes del valor respectivo para cada empleado.
14. Muestra los empleados cuyo nombre empiece entre las letras J y Z (rango). Liste estos empleados y su cargo por orden alfabético.
15. Listar el salario, la comisión, el salario total (salario + comisión) y nombre, de aquellos empleados que tienen comisión superior a 1000.
16. Obtener un listado similar al anterior, pero de aquellos empleados que NO tienen comisión
17. Hallar los empleados cuyo nombre no contiene la cadena "MA"
18. Obtener los nombres de los departamentos que no sean "Ventas" ni "Investigación" ni 'Mantenimiento.
19. Obtener el nombre y el departamento de los empleados con cargo 'Secretaria' o 'Vendedor', que no trabajan en el departamento de "PRODUCCION", cuyo salario es superior a $1000, ordenados por fecha de incorporación.
20. Obtener información de los empleados cuyo nombre tiene exactamente 11 caracteres
21. Obtener información de los empleados cuyo nombre tiene al menos 11 caracteres
22. Listar los datos de los empleados cuyo nombre inicia por la letra 'M', su salario es mayor a $800 o reciben comisión y trabajan para el departamento de 'VENTAS'
23. Mostrar el salario más alto de la empresa.
24. Mostrar el nombre del último empleado de la lista por orden alfabético.
25. Hallar el salario más alto, el más bajo y la diferencia entre ellos.
26. Mostrar el número de empleados de sexo femenino y de sexo masculino.
27. Hallar el salario promedio por departamento.
28. Mostrar la lista de los empleados cuyo salario es mayor o igual que el promedio de la empresa. Ordenarlo por departamento.
29. Hallar los departamentos que tienen más de tres empleados. Mostrar el número de empleados de esos departamentos.

30. Mostrar el código y nombre de cada jefe, junto al número de empleados que dirige. Solo los que tengan más de dos empleados (2 incluido).
31. Hallar los departamentos que no tienen empleados
32. Mostrar el nombre del departamento cuya suma de salarios sea la más alta, indicando el valor de la suma de los salarios.

4. Abrir el script de la base de datos llamada "tienda.sql" y ejecutarlo para crear sus tablas e insertar datos en las mismas. A continuación, generar el modelo de entidad relación. Deberá obtener un diagrama de entidad relación igual al que se muestra a continuación:



A continuación, se deben realizar las siguientes consultas sobre la base de datos:

1. Lista el nombre de todos los productos que hay en la tabla producto.
2. Lista los nombres y los precios de todos los productos de la tabla producto.
3. Lista todas las columnas de la tabla producto.
4. Lista el nombre de los productos, el precio en euros y el precio en dólares estadounidenses (USD).
5. Lista los nombres y los precios de todos los productos de la tabla producto, convirtiendo los nombres a mayúscula.
6. Lista el nombre de todos los fabricantes en una columna, y en otra columna obtenga en mayúsculas los dos primeros caracteres del nombre del fabricante.
7. Lista los nombres y los precios de todos los productos de la tabla producto, redondeando el valor del precio.
8. Lista los nombres y los precios de todos los productos de la tabla producto, truncando el valor del precio para mostrarlo sin ninguna cifra decimal.
9. Lista el código de los fabricantes que tienen productos en la tabla producto.
10. Lista el código de los fabricantes que tienen productos en la tabla producto, sin mostrar los repetidos.
11. Lista los nombres de los fabricantes ordenados de forma ascendente.
12. Lista los nombres de los productos ordenados en primer lugar por el nombre de forma ascendente y en segundo lugar por el precio de forma descendente.
13. Devuelve una lista con las 5 primeras filas de la tabla fabricante.

14. Lista el nombre y el precio del producto más barato. (Utilice solamente las cláusulas ORDER BY y LIMIT)
15. Lista el nombre y el precio del producto más caro. (Utilice solamente las cláusulas ORDER BY y LIMIT)
16. Lista el nombre de todos los productos del fabricante cuyo código de fabricante es igual a 2.
17. Lista el nombre de los productos que tienen un precio menor o igual a $120.
18. Lista todos los productos que tengan un precio entre $80 y $300. Sin utilizar el operador BETWEEN.
19. Lista todos los productos que tengan un precio entre $60 y $200. Utilizando el operador BETWEEN.
20. Lista todos los productos donde el código de fabricante sea 1, 3 o 5. Sin utilizar el operador IN.
21. Lista todos los productos donde el código de fabricante sea 1, 3 o 5. Utilizando el operador IN.
22. Lista el nombre y el precio de los productos en céntimos (Habrá que multiplicar por 100 el valor del precio). Cree un alias para la columna que contiene el precio que se llame céntimos.
23. Lista los nombres de los fabricantes cuyo nombre sea de 4 caracteres.
24. Devuelve una lista con el nombre de todos los productos que contienen la cadena Portátil en el nombre.

Consultas multitabla (Composición interna)

1. Devuelve una lista con el código del producto, nombre del producto, código del fabricante y nombre del fabricante, de todos los productos de la base de datos.
2. Devuelve una lista con el nombre del producto, precio y nombre de fabricante de todos los productos de la base de datos. Ordene el resultado por el nombre del fabricante, por orden alfabético.
3. Devuelve el nombre del producto, su precio y el nombre de su fabricante, del producto más barato.
4. Devuelve una lista de todos los productos del fabricante Lenovo.
5. Devuelve una lista de todos los productos del fabricante Crucial que tengan un precio mayor que $200.
6. Devuelve un listado con todos los productos de los fabricantes Asus, Hewlett-Packard. Sin utilizar el operador IN.
7. Devuelve un listado con todos los productos de los fabricantes Asus, Hewlett-Packard. Utilizando el operador IN.

8. Devuelve un listado con el nombre de producto, precio y nombre de fabricante, de todos los productos que tengan un precio mayor o igual a $180. Ordene el resultado en primer lugar por el precio (en orden descendente) y en segundo lugar por el nombre (en orden ascendente)

Consultas multitabla (Composición externa)

Resuelva todas las consultas utilizando las cláusulas LEFT JOIN y RIGHT JOIN.

1. Devuelve un listado de todos los fabricantes que existen en la base de datos, junto con los productos que tiene cada uno de ellos. El listado deberá mostrar también aquellos fabricantes que no tienen productos asociados.
2. Devuelve un listado donde sólo aparezcan aquellos fabricantes que no tienen ningún producto asociado.

Subconsultas (En la cláusula WHERE)

Con operadores básicos de comparación

1. Devuelve todos los productos del fabricante Lenovo. (Sin utilizar INNER JOIN).
2. Devuelve todos los datos de los productos que tienen el mismo precio que el producto más caro del fabricante Lenovo. (Sin utilizar INNER JOIN).
3. Lista el nombre del producto más caro del fabricante Lenovo.
4. Lista todos los productos del fabricante Asus que tienen un precio superior al precio medio de todos sus productos.

Subconsultas con ALL y ANY

1. Devuelve el producto más caro que existe en la tabla producto sin hacer uso de MAX, ORDER BY ni LIMIT.
2. Devuelve el producto más barato que existe en la tabla producto sin hacer uso de MIN, ORDER BY ni LIMIT.
3. Devuelve los nombres de los fabricantes que tienen productos asociados. (Utilizando ALL o ANY).
4. Devuelve los nombres de los fabricantes que no tienen productos asociados. (Utilizando ALL o ANY).

Subconsultas con IN y NOT IN

1. Devuelve los nombres de los fabricantes que tienen productos asociados. (Utilizando IN o NOT IN).
2. Devuelve los nombres de los fabricantes que no tienen productos asociados. (Utilizando IN o NOT IN).

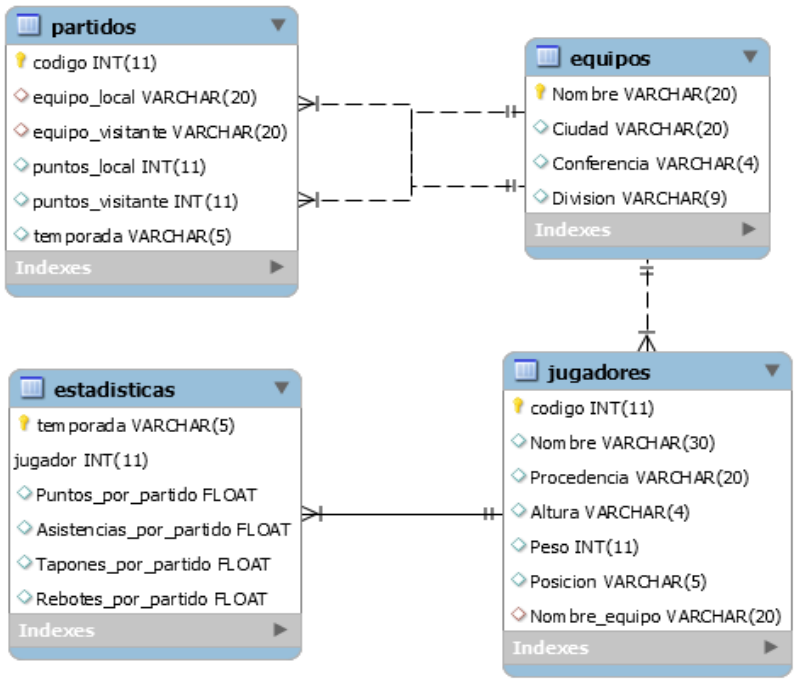
Subconsultas con EXISTS y NOT EXISTS

1. Devuelve los nombres de los fabricantes que tienen productos asociados. (Utilizando EXISTS o NOT EXISTS).
2. Devuelve los nombres de los fabricantes que no tienen productos asociados. (Utilizando EXISTS o NOT EXISTS).

Subconsultas (En la cláusula HAVING)

1. Devuelve un listado con todos los nombres de los fabricantes que tienen el mismo número de productos que el fabricante Lenovo.

5. Abrir el script de la base de datos llamada "nba.sql" y ejecutarlo para crear todas las tablas e insertar datos en las mismas. A continuación, generar el modelo de entidad relación. Deberá obtener un diagrama de entidad relación igual al que se muestra a continuación:



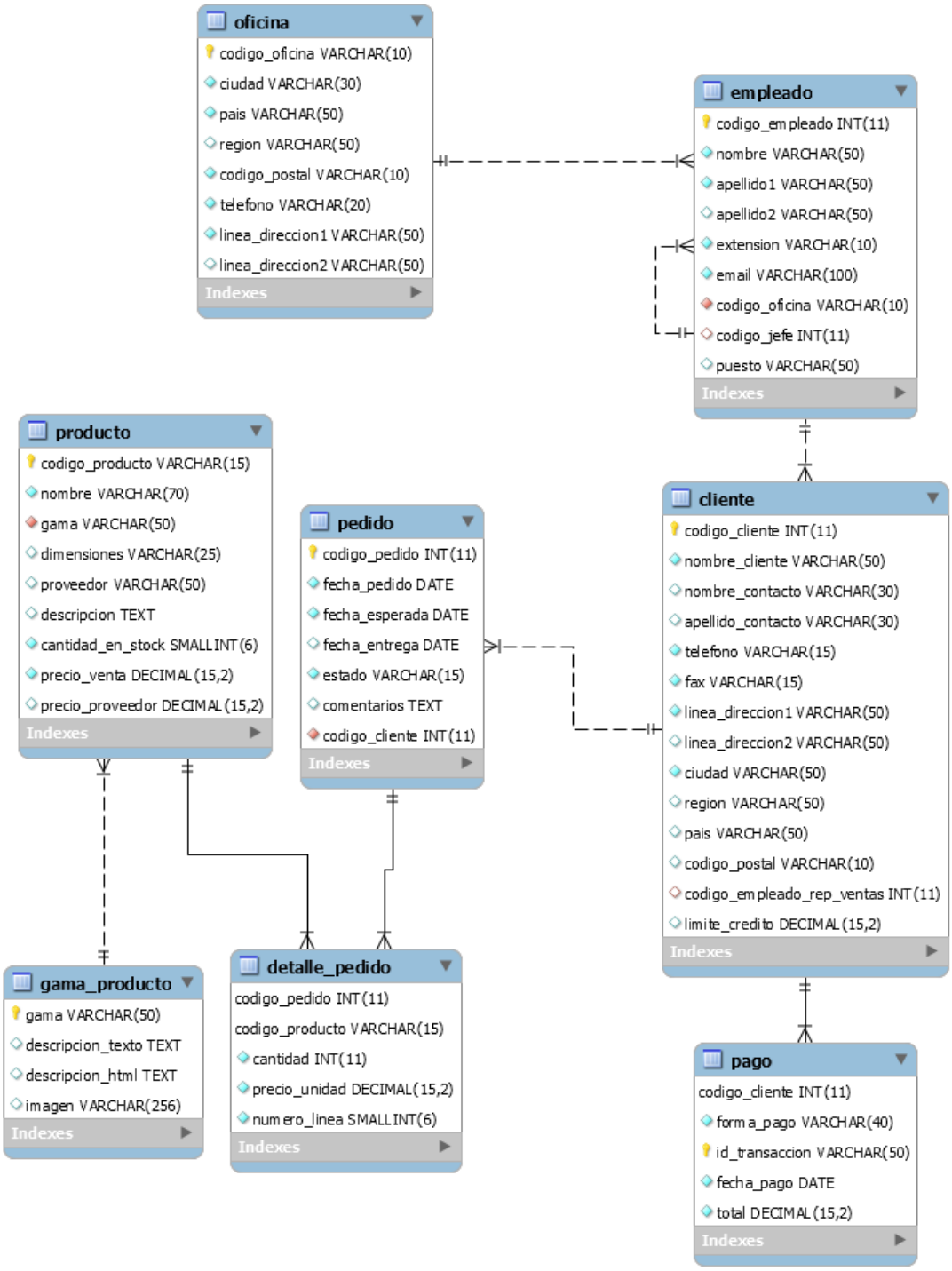
A continuación, se deben realizar las siguientes consultas sobre la base de datos:

1. Mostrar el nombre de todos los jugadores ordenados alfabéticamente.
2. Mostrar el nombre de los jugadores que sean pivots ('C') y que pesen más de 200 libras, ordenados por nombre alfabéticamente.
3. Mostrar el nombre de todos los equipos ordenados alfabéticamente.
4. Mostrar el nombre de los equipos del este (East).
5. Mostrar los equipos donde su ciudad empieza con la letra 'c', ordenados por nombre.
6. Mostrar todos los jugadores y su equipo ordenados por nombre del equipo.

7. Mostrar todos los jugadores del equipo “Raptors” ordenados por nombre.
8. Mostrar los puntos por partido del jugador ‘Pau Gasol’.
9. Mostrar los puntos por partido del jugador ‘Pau Gasol’ en la temporada ‘04/05’.
10. Mostrar el número de puntos de cada jugador en toda su carrera.
11. Mostrar el número de jugadores de cada equipo.
12. Mostrar el jugador que más puntos ha realizado en toda su carrera.
13. Mostrar el nombre del equipo, conferencia y división del jugador más alto de la NBA.
14. Mostrar la suma de los puntos por partido de todos los jugadores españoles donde el equipo donde juegan este en ‘Los Angeles’.
15. Mostrar la media de puntos en partidos de los equipos de la división Pacific.
16. Mostrar el partido o partidos (equipo_local, equipo_visitante y diferencia) con mayor diferencia de puntos.
17. Mostrar la media de puntos en partidos de los equipos de la división Pacific.
18. Mostrar los puntos de cada equipo en los partidos, tanto de local como de visitante.
19. Mostrar quien gana en cada partido (codigo, equipo_local, equipo_visitante, equipo_ganador), en caso de empate sera null.

EJERCICIOS COMPLEMENTARIOS

6. Abrir el script de la base de datos llamada "jardineria.sql" y ejecutarlo para crear todas las tablas e insertar datos en las mismas. Deberá obtener un diagrama de entidad relación igual al que se muestra a continuación:



A continuación, se deben realizar las siguientes consultas sobre la base de datos:

Consultas sobre una tabla

1. Devuelve un listado con el código de oficina y la ciudad donde hay oficinas.
2. Devuelve un listado con la ciudad y el teléfono de las oficinas de España.
3. Devuelve un listado con el nombre, apellidos y email de los empleados cuyo jefe tiene un código de jefe igual a 7.
4. Devuelve el nombre del puesto, nombre, apellidos y email del jefe de la empresa.
5. Devuelve un listado con el nombre, apellidos y puesto de aquellos empleados que no sean representantes de ventas.
6. Devuelve un listado con el nombre de los todos los clientes españoles.
7. Devuelve un listado con los distintos estados por los que puede pasar un pedido.
8. Devuelve un listado con el código de cliente de aquellos clientes que realizaron algún pago en 2008. Tenga en cuenta que deberá eliminar aquellos códigos de cliente que aparezcan repetidos. Resuelva la consulta:
    - Utilizando la función YEAR de MySQL.
    - Utilizando la función DATE_FORMAT de MySQL.
    - Sin utilizar ninguna de las funciones anteriores.
9. Devuelve un listado con el código de pedido, código de cliente, fecha esperada y fecha de entrega de los pedidos que no han sido entregados a tiempo.
10. Devuelve un listado con el código de pedido, código de cliente, fecha esperada y fecha de entrega de los pedidos cuya fecha de entrega ha sido al menos dos días antes de la fecha esperada.
    - Utilizando la función ADDDATE de MySQL.
    - Utilizando la función DATEDIFF de MySQL.
11. Devuelve un listado de todos los pedidos que fueron rechazados en 2009.
12. Devuelve un listado de todos los pedidos que han sido entregados en el mes de enero de cualquier año.
13. Devuelve un listado con todos los pagos que se realizaron en el año 2008 mediante Paypal. Ordene el resultado de mayor a menor.
14. Devuelve un listado con todas las formas de pago que aparecen en la tabla pago. Tenga en cuenta que no deben aparecer formas de pago repetidas.
15. Devuelve un listado con todos los productos que pertenecen a la gama Ornamentales y que tienen más de 100 unidades en stock. El listado deberá estar ordenado por su precio de venta, mostrando en primer lugar los de mayor precio.
16. Devuelve un listado con todos los clientes que sean de la ciudad de Madrid y cuyo representante de ventas tenga el código de empleado 11 o 30.

Consultas multitabla (Composición interna)

Las consultas se deben resolver con INNER JOIN.

1. Obtén un listado con el nombre de cada cliente y el nombre y apellido de su representante de ventas.
2. Muestra el nombre de los clientes que hayan realizado pagos junto con el nombre de sus representantes de ventas.
3. Muestra el nombre de los clientes que no hayan realizado pagos junto con el nombre de sus representantes de ventas.
4. Devuelve el nombre de los clientes que han hecho pagos y el nombre de sus representantes junto con la ciudad de la oficina a la que pertenece el representante.
5. Devuelve el nombre de los clientes que no hayan hecho pagos y el nombre de sus representantes junto con la ciudad de la oficina a la que pertenece el representante.
6. Lista la dirección de las oficinas que tengan clientes en Fuenlabrada.
7. Devuelve el nombre de los clientes y el nombre de sus representantes junto con la ciudad de la oficina a la que pertenece el representante.
8. Devuelve un listado con el nombre de los empleados junto con el nombre de sus jefes.
9. Devuelve el nombre de los clientes a los que no se les ha entregado a tiempo un pedido.
10. Devuelve un listado de las diferentes gamas de producto que ha comprado cada cliente.

Consultas multitabla (Composición externa)

Resuelva todas las consultas utilizando las cláusulas LEFT JOIN, RIGHT JOIN, JOIN.

1. Devuelve un listado que muestre solamente los clientes que no han realizado ningún pago.
2. Devuelve un listado que muestre solamente los clientes que no han realizado ningún pedido.
3. Devuelve un listado que muestre los clientes que no han realizado ningún pago y los que no han realizado ningún pedido.
4. Devuelve un listado que muestre solamente los empleados que no tienen una oficina asociada.
5. Devuelve un listado que muestre solamente los empleados que no tienen un cliente asociado.
6. Devuelve un listado que muestre los empleados que no tienen una oficina asociada y los que no tienen un cliente asociado.
7. Devuelve un listado de los productos que nunca han aparecido en un pedido.

8. Devuelve las oficinas donde no trabajan ninguno de los empleados que hayan sido los representantes de ventas de algún cliente que haya realizado la compra de algún producto de la gama Frutales.
9. Devuelve un listado con los clientes que han realizado algún pedido pero no han realizado ningún pago.
10. Devuelve un listado con los datos de los empleados que no tienen clientes asociados y el nombre de su jefe asociado.

Consultas resumen

1. ¿Cuántos empleados hay en la compañía?
2. ¿Cuántos clientes tiene cada país?
3. ¿Cuál fue el pago medio en 2009?
4. ¿Cuántos pedidos hay en cada estado? Ordena el resultado de forma descendente por el número de pedidos.
5. Calcula el precio de venta del producto más caro y más barato en una misma consulta.
6. Calcula el número de clientes que tiene la empresa.
7. ¿Cuántos clientes tiene la ciudad de Madrid?
8. ¿Calcula cuántos clientes tiene cada una de las ciudades que empiezan por M?
9. Devuelve el nombre de los representantes de ventas y el número de clientes al que atiende cada uno.
10. Calcula el número de clientes que no tiene asignado representante de ventas.
11. Calcula la fecha del primer y último pago realizado por cada uno de los clientes. El listado deberá mostrar el nombre y los apellidos de cada cliente.
12. Calcula el número de productos diferentes que hay en cada uno de los pedidos.
13. Calcula la suma de la cantidad total de todos los productos que aparecen en cada uno de los pedidos.
14. Devuelve un listado de los 20 productos más vendidos y el número total de unidades que se han vendido de cada uno. El listado deberá estar ordenado por el número total de unidades vendidas.
15. La facturación que ha tenido la empresa en toda la historia, indicando la base imponible, el IVA y el total facturado. La base imponible se calcula sumando el coste del producto por el número de unidades vendidas de la tabla detalle_pedido. El IVA es el 21 % de la base imponible, y el total la suma de los dos campos anteriores.
16. La misma información que en la pregunta anterior, pero agrupada por código de producto.
17. La misma información que en la pregunta anterior, pero agrupada por código de producto filtrada por los códigos que empiecen por OR.

18. Lista las ventas totales de los productos que hayan facturado más de 3000 euros. Se mostrará el nombre, unidades vendidas, total facturado y total facturado con impuestos (21% IVA)

Subconsultas con operadores básicos de comparación

1. Devuelve el nombre del cliente con mayor límite de crédito.
2. Devuelve el nombre del producto que tenga el precio de venta más caro.
3. Devuelve el nombre del producto del que se han vendido más unidades. (Tenga en cuenta que tendrá que calcular cuál es el número total de unidades que se han vendido de cada producto a partir de los datos de la tabla detalle_pedido. Una vez que sepa cuál es el código del producto, puede obtener su nombre fácilmente.)
4. Los clientes cuyo límite de crédito sea mayor que los pagos que haya realizado. (Sin utilizar INNER JOIN).
5. Devuelve el producto que más unidades tiene en stock.
6. Devuelve el producto que menos unidades tiene en stock.
7. Devuelve el nombre, los apellidos y el email de los empleados que están a cargo de Alberto Soria.

Subconsultas con ALL y ANY

1. Devuelve el nombre del cliente con mayor límite de crédito.
2. Devuelve el nombre del producto que tenga el precio de venta más caro.
3. Devuelve el producto que menos unidades tiene en stock.

Subconsultas con IN y NOT IN

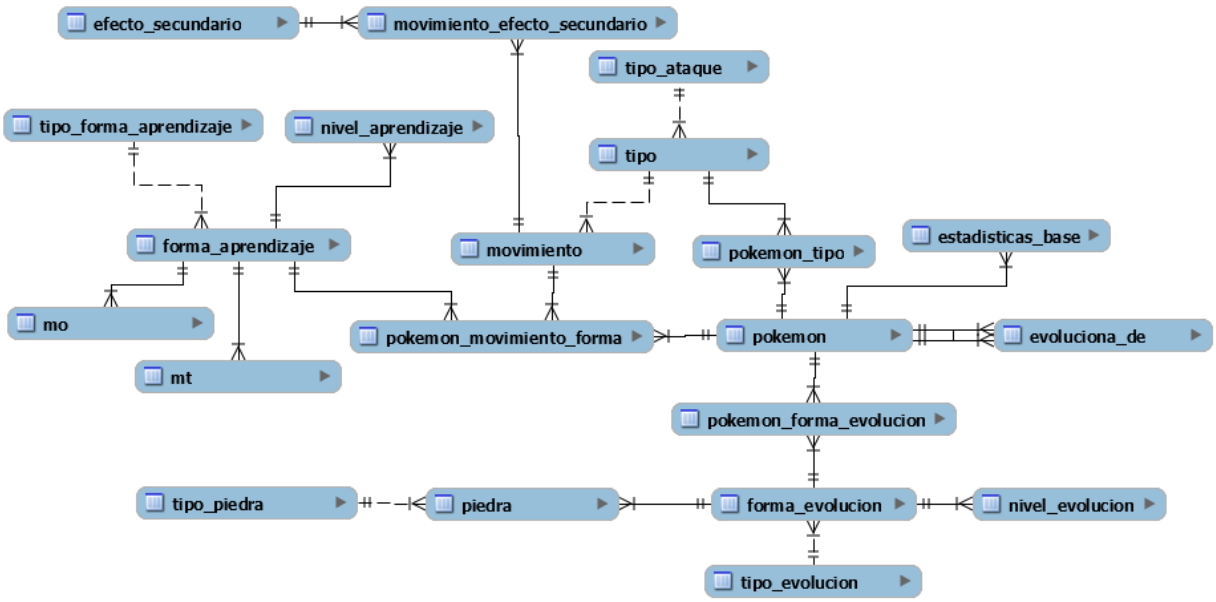
1. Devuelve el nombre, apellido1 y cargo de los empleados que no representen a ningún cliente.
2. Devuelve un listado que muestre solamente los clientes que no han realizado ningún pago.
3. Devuelve un listado que muestre solamente los clientes que sí han realizado ningún pago.
4. Devuelve un listado de los productos que nunca han aparecido en un pedido.
5. Devuelve el nombre, apellidos, puesto y teléfono de la oficina de aquellos empleados que no sean representante de ventas de ningún cliente.

Subconsultas con EXISTS y NOT EXISTS

1. Devuelve un listado que muestre solamente los clientes que no han realizado ningún pago.
2. Devuelve un listado que muestre solamente los clientes que sí han realizado ningún pago.

3. Devuelve un listado de los productos que nunca han aparecido en un pedido.
4. Devuelve un listado de los productos que han aparecido en un pedido alguna vez.

7. Importar el script de la base de datos llamada “pokemondb.sql” y ejecutarlo para crear todas las tablas e insertar los registros en las mismas. A continuación, generar el modelo de entidad relación y reorganizar las tablas para mayor claridad de sus relaciones. Deberá obtener un diagrama de entidad de relación similar al que se muestra a continuación:



A continuación, se deben realizar las siguientes consultas:

1. Mostrar el nombre de todos los pokemon.
2. Mostrar los pokemon que pesen menos de 10k.
3. Mostrar los pokemon de tipo agua.
4. Mostrar los pokemon de tipo agua, fuego o tierra ordenados por tipo.
5. Mostrar los pokemon que son de tipo fuego y volador.
6. Mostrar los pokemon con una estadística base de ps mayor que 200.
7. Mostrar los datos (nombre, peso, altura) de la prevolución de Arbok.
8. Mostrar aquellos pokemon que evolucionan por intercambio.
9. Mostrar el nombre del movimiento con más prioridad.
10. Mostrar el pokemon más pesado.
11. Mostrar el nombre y tipo del ataque con más potencia.
12. Mostrar el número de movimientos de cada tipo.
13. Mostrar todos los movimientos que puedan envenenar.

14. Mostrar todos los movimientos que causan daño, ordenados alfabéticamente por nombre.
15. Mostrar todos los movimientos que aprende pikachu.
16. Mostrar todos los movimientos que aprende pikachu por MT (tipo de aprendizaje).
17. Mostrar todos los movimientos de tipo normal que aprende pikachu por nivel.
18. Mostrar todos los movimientos de efecto secundario cuya probabilidad sea mayor al 30%.
19. Mostrar todos los pokemon que evolucionan por piedra.
20. Mostrar todos los pokemon que no pueden evolucionar.
21. Mostrar la cantidad de los pokemon de cada tipo.