TEMA 3: SQL

1. INTRODUCCIÓN

La mayoría de funciones SQL pueden englobarse en dos categorías:

a) **Lenguaje de definición de datos (DDL)**: creación. eliminación y modificación de objetos a través de CREATE, ALTER, DROP, TRUNCATE, RENAME
b) **Lenguaje de manipulación de datos (DML)**: consulta, inserción, modificación y borrado de datos a través de SELECT, INSERT, UPDATE, DELETE
c) Lenguaje de control de datos (CDL): GRANT, REVOKE
d) Lenguaje de control de transacción (TCL): COMMIT, ROLLBACK, SAVEPOINT

2. FUNCIONES DDL

Creamos el esquema sobre el que vamos a trabajar y la BDs

```
CREATE SCHEMA nombre_esquema;
CREATE DATABASE nombre_db;
```

Indicamos el esquema (o BDs) sobre el que vamos a trabajar

```
USE database;
```

NOTA: todos los comandos acaban con ; y los comentarios van precedidos por #

Usamos **CREATE** para crear tablas (y otras estructuras). Se debe especificar su nombre, atributos, tipos (dominios) de las columnas (atributos) y restricciones de seguridad:

```
CREATE TABLE nombre_tabla (
  columna_1 tipo [UNIQUE] [NOT NULL] [DEFAULT valor],
  …
  columna_n tipo [UNIQUE]  [NOT NULL] [DEFAULT valor],
  PRIMARY KEY (columna_i),
  CONSTRAINT fk_nombre_1 FOREIGN KEY (columna_j) REFERENCES tabla_a (PK tabla_a)
        ON [DELETE | UPDATE] {CASCADE | SET DEFAULT | SET NULL}
  …
  CONSTRAINT fk_nombre_n FOREIGN KEY (columna_k) REFERENCES tabla_n (PK tabla_n)
        ON [DELETE | UPDATE] {CASCADE | SET DEFAULT | SET NULL}
);
```

donde…

TIPOS DE DATOS

INT → número entero	DATE → fecha en formato AAAA-MM-DD
FLOAT → número real (coma flotante)	TIME → hora en formato HH:MM:SS
CHAR(n) → cadena de n caracteres fijos	BOOL / BOOLEAN → valores TRUE / FALSE por defecto
VARCHAR(n) → cadena de hasta n caracteres variables	TINYINT, BIGINT, NUMERIC, DOUBLE, BIT, BINARY, ENUM, JSON…

RESTRICCIONES

i) UNIQUE: no puede haber dos filas con el mismo valor en esa columna. Se llaman claves alternativas ya que son alternativas a la PK.
ii) NOT NULL: no permite valores nulos en esa columna. Si se inserta una fila sin especificar un valor en una columna el valor por defecto es NULL.
iii) DEFAULT *valor*: inserta *valor* en lugar de NULL cuando no se especifica un valor

INTEGRIDAD REFERENCIAL

Hay varias opciones al borrar o modificar los datos a los que apuntan (FOREIGN KEY):
i) CASCADE: se eliminan todos los datos que dependían de la FK
ii) SET DEFAULT: se establece el valor por defecto de la columna (si se ha indicado o NULL en caso contrario)
iii) SET NULL: se establece el valor a NULL

Usamos **DROP** para eliminar todo lo que se ha creado con el comando **CREATE**:

```
DROP SCHEMA nombre_esquema;
DROP DATABASE nombre_BD;
DROP TABLE nombre_tabla [{RESTRICT | CASCADE}];
```

Se usa la opción RESTRICT por defecto ya que impide borrar una tabla si hace referencia a otra. Con CASCADE se fuerza el borrado de una tabla y por consecuencia de todas las que apuntan a ella.

Usamos **ALTER** para añadir, modificar o eliminar columnas de una tabla:

```
ALTER TABLE tabla,
ADD COLUMN columna tipo [restricciones];
DROP COLUMN columna {RESTRICT | CASCADE};
CHANGE COLUMN nombre_col_antigua nombre_col_nueva nuevo_tipo [nuevas_restricciones];
```

3. FUNCIONES DML

Insertar filas en una tabla

```
INSERT INTO tabla (columna_1, columna_2, …, columna_n)
VALUES (valor_1, valor_2, …, valor_n)
```

* Si se especifican todos los valores en el mismo orden que se insertaron en la DB, entonces podemos prescindir de esta parte.

Realizar consultas

```
SELECT lista_de_atributos   → atributos que se desean recuperar
FROM lista_de_tablas        → nombre de las tablas requeridas para la consulta (puede ser 1)
[WHERE condiciones];        → expresión condicional (booleana) que identifica las filas que
                              se quieren recuperar
```

- ➢ Si en lista_de_atributos especificamos * nos devolverá todos los atributos de la tabla
- ➢ Se puede renombrar una columna con **AS**, aunque es opcional y se puede prescindir de él
  Ejemplo: SELECT nombre AS name FROM actor;
  SELECT nombre name FROM actor;
- ➢ Los resultados pueden ordenarse usando **ORDER BY**. Es posible especificar una o varias columnas para ordenar los resultados. Hay dos ordenaciones diferentes:
  (1) ascendente (*asc*) - que es la que se utiliza por defecto - y
  (2) descendente (*desc*)
- ➢ Se puede restringir el número de filas devueltas con **LIMIT**
- ➢ Se pueden usar operadores lógicos de varios tipos:
  - **BOOLEANOS**: **AND**, **OR**, **NOT**
  - **COMPARACIONES**: **=, <>** (distinto)**, <, <=, >, >=, IS NULL, IS NOT NULL**
  - **PATRONES**: columna **LIKE** patrón, columna **NOT LIKE** patrón
    donde _ representa un carácter cualquiera (uno y solo uno); mientras que
    % representa un número indeterminado de caracteres (desde 0 a infinito)
- ➢ Con el operador **IN** se puede dar una lista de valores que puede tomar una columna.
  Es equivalente al comando **OR**
- ➢ Con el operador **BETWEEN** se puede especificar el rango por el que filtrar una columna
- ➢ Se puede hacer que no haya elementos repetidos usando **SELECT DISTINCT**
- ➢ Si se especifican varias tablas entonces se crea una única tabla con el producto cartesiano ("*join*") de todas las tablas seleccionadas
- ➢ Si hay ambigüedad, los atributos con el mismo nombre pueden distinguirse con **tabla.columna**
- ➢ Una misma tabla puede usarse dos veces en la cláusula FROM, en este caso es OBLIGATORIO el uso de un **alias** para cada tabla repetida
- ➢ Con la sentencia **GROUP BY** se agrupan los datos según los valores de una columna. Después se aplican las funciones de agregación** sobre cada uno de los grupos. Los grupos obtenidos pueden filtrarse mediante la cláusula **HAVING** (es como WHERE pero en este caso afecta a grupos. **OJO: ¡¡¡NO CONFUNDIR !!!**)

** Funciones de agregación:

- **COUNT**: número de filas seleccionadas por la consulta
- **MIN (column)**: valor mínimo en la columna de la consulta especificada
- **MAX (column)**: valor máximo en la columna de la consulta especificada
- **SUM (column)**: suma de los valores de la columna de la consulta especificada
- **AVG (column)**: promedio de los valores de la columna de la consulta especificada

NOTA: toda columna de la consulta debe ser agrupada o se debe usar una función de agrupación sobre ella.

Borrar filas de una tabla

```
DELETE FROM nombre_tabla
[WHERE condiciones];
```

→ si no se especifican, se borran todas filas de la tabla

NOTA: con **TRUNCATE** se borran todos los elementos de la tabla sin modificar su estructura

Actualizar filas de una tabla

```
UPDATE nombre_tabla
SET columna_1=valor_1, …, columna_i=valor_i
[WHERE condiciones];
```

→ si no se especifican, se actualizan todas las filas de la tabla

WARNING: cuidado al borrar (o actualizar) datos que otras tablas referencian. Los efectos se propagan según las restricciones de integridad referencial elegidas.

Subconsultas:

- ➢ Se trata de consultas anidadas en otra sentencia DML de SQL que DEBE ir entre paréntesis.
- ➢ Algunos *join* pueden hacerse mediante subconsultas.
- ➢ Pueden anidarse tantas subconsultas como se necesiten.
- ➢ También se pueden incluir en la cláusula FROM.

EJERCICIOS: SQL con Sakila

<u>Nota importante:</u> Las soluciones no tienen por qué ser únicas para todos los ejercicios.

1. Obtener todos los datos de la tabla actor.

2. Obtener cuántos actores hay en la base de datos.

3. ¿Cuántos actores se llaman NICK?

4. Obtener todos los datos del actor NICK WAHLBERG.

5. Hacer una consulta que devuelva todos los actores ordenados por su: apellido como primer criterio, nombre como segundo.

6. ¿Cómo hacer para que la consulta anterior nos devuelva solamente el primer actor que resulte de esa ordenación?

7. ¿Cuántas películas tienen un precio de alquiler superior a 4?

8. ¿Cuántas películas su título empieza por W?

9. ¿Y cuántas acaban en W? ¿Cuáles son?

10. Obtener el título y la duración de la película más larga.

11. ¿Cuál es la duración media y coste medio de las películas?

12. Obtener todas las películas (todos los datos) con rating NC-17 o PG. Hacerlo de dos modos.

13. Obtener todas las películas (título) con una duración superior o igual a 100 minutos e inferior o igual a 180. Hacerlo de dos modos.

14. Obtener todos los datos del actor NICK WAHLBERG con dos consultas encadenadas usando como criterio de la primera consulta el ID.

15. En base a la anterior, ¿qué ocurre si la segunda subconsulta solo ponemos el nombre y no el apellido? ¿Por qué?

16. ¿Cuántos actores han participado en la película 'ACADEMY DINOSAUR'?

17. Obtener los nombres de dichos actores.

18. ¿Cuántas películas pertenecen a la categoría de acción (Action)?

19. ¿Cuáles son los títulos de esas películas?

20. ¿Cuál es la duración media de las películas de Terror (Horror)?

21. ¿Cuál es la duración media de las películas en función de su categoría? Devolverlas ordenadas de forma ascendente.

22. Las películas están en un determinado idioma, ¿cuántos idiomas diferentes tenemos en nuestra lista de películas?

23. ¿Qué idiomas son?

24. ¿Cuál es el país predominante de la tabla “city”?

25. ¿Quién es el cliente que ha alquilado más películas?

26. ¿Cuál es la película de las que se tienen más copias según el inventario y cuántas copias? Obtener ambos datos en una sola query.

27. Obtener una relación de los identificadores de las tiendas y el número de películas que tienen inventariadas.

28. Obtener la dirección de la tienda que tiene más películas en su inventario.

#con subconsulta

#sin subconsulta

29. ¿En qué ciudad trabaja MIKE?

#sin subconsultas

#con subconsultas

30. Borrar todos los pagos que haya realizado MARY SMITH.

31. Hemos contratado a la novia de MIKE, que vive y trabajará con él, añadirla a la tabla staff:

- ID empleado: 3
- Nombre: Laura Martínez
- Dirección: Misma que Mike
- Email: laura@laura.com
- Tienda: Misma que Mike
- Usuario: Laura
- Contraseña: laura@sakila (cifrada con SHA1) → SHA1('laura@sakila')
- Última actualización: CURRENT_TIMESTAMP

32. Mike se ha echado novia en ‘Woodbridge’ y ha pedido el traslado a dicha ciudad. Actualizar la base de datos para que Mike pueda irse allí. **Nota:** La dirección de la tienda nueva contiene ‘MySQL’ en la calle.

33. Mike ha fallecido haciendo paracaidismo. La empresa ha contratado a Laura para suplirlo y le ha cedido el piso donde vivía Mike (que es de la empresa). Por lo tanto:
    a. Añade a Laura a la base de datos con estos datos:
        - ID empleado: 78
        - Nombre: Laura Martinez
        - Dirección: Misma que Mike (ID 3)
        - Email: laura@laura.com
        - Tienda: Misma que Mike (ID 2)
        - Usuario: Laura
        - Contraseña: laura@sakila (cifrada con SHA1)

    b. Actualiza la base de datos para que todo lo que estaba relacionado con Mike ahora sea Laura la persona que se ocupe.

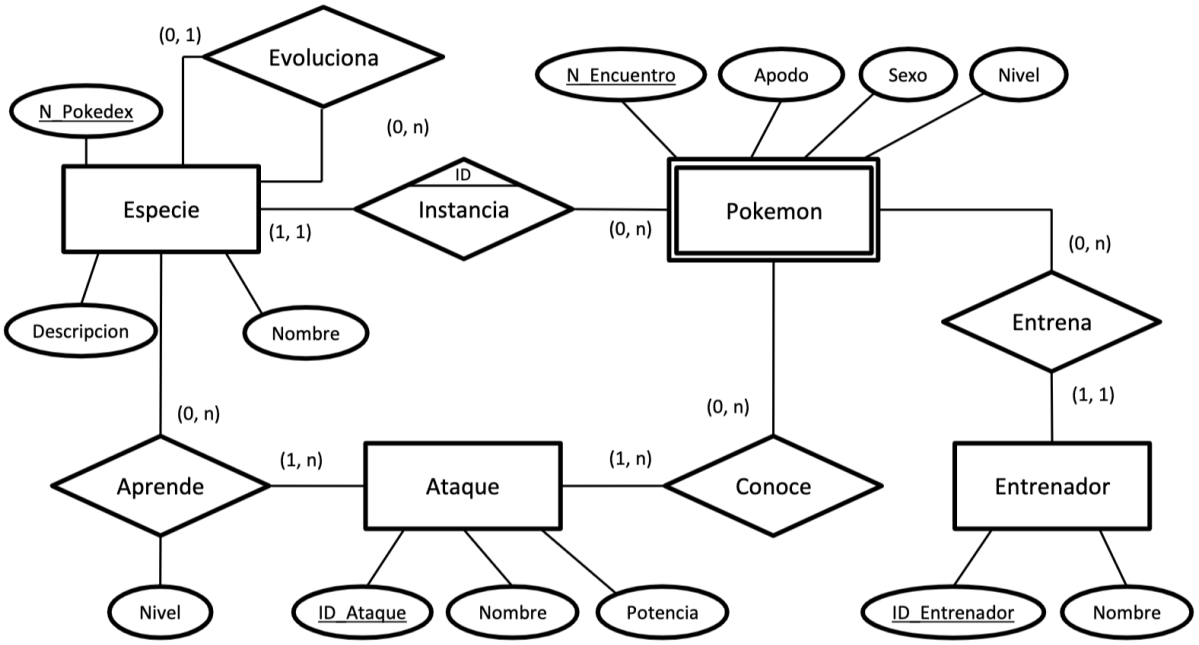
    c. Borra a Mike de la base de datos.

EXÁMENES SQL

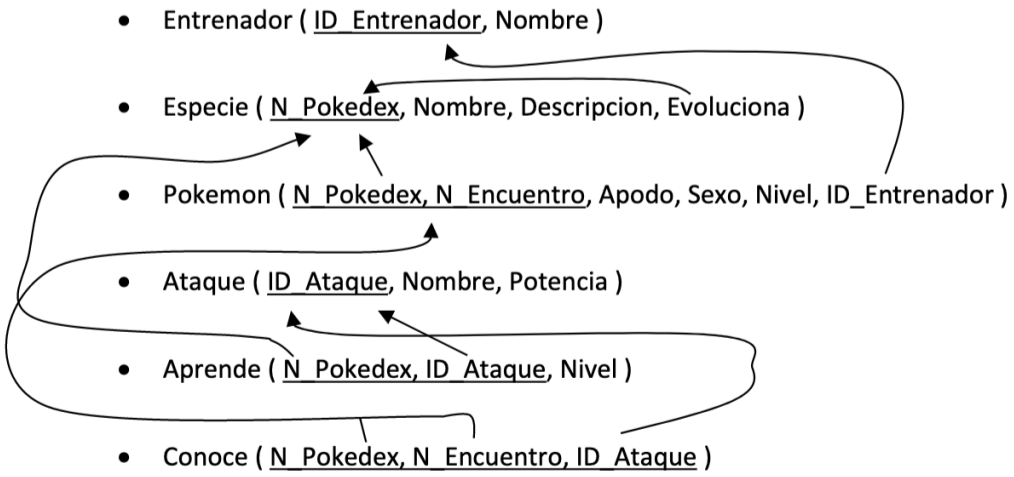
<u>Nota importante:</u> Las soluciones no tienen por qué ser únicas para todos los ejercicios.

<u>JUNIO 2019</u>

Considérese el siguiente diagrama que representa, de forma simplificada, una base de datos de un juego de Pokémon. La entidad "Especie" representa cada especie Pokémon (por ejemplo, la especie "Pidgey"), mientras que la entidad "Pokemon" representa un ejemplar concreto de una especie (por ejemplo, el primer ejemplar de Pidgey que aparece). A cada ejemplar de Pokémon se le asigna un "N_Encuentro" correlativo para su especie (es decir, puede haber un ejemplar de la especie Pidgey cuyo atributo "N_Encuentro" tenga un valor de 1 y un ejemplar de la especie Rattata cuyo atributo "N_Encuentro" también valga 1).



A continuación, se da también el paso a tablas del diagrama anterior:



- Entrenador (ID_Entrenador, Nombre)
- Especie (N_Pokedex, Nombre, Descripcion, Evoluciona)
- Pokemon (N_Pokedex, N_Encuentro, Apodo, Sexo, Nivel, ID_Entrenador)
- Ataque (ID_Ataque, Nombre, Potencia)
- Aprende (N_Pokedex, ID_Ataque, Nivel)
- Conoce (N_Pokedex, N_Encuentro, ID_Ataque)

(*) Nótese que el atributo "Evoluciona" de la tabla Especie indica el N_Pokédex de la especia que evoluciona a dicha especie, ya que en algunos casos un mismo Pokémon puede evolucionar en varios distintos y, por tanto, no es posible guardar en un atributo la relación en sentido inverso.

Se pide escribir los comandos SQL para realizar las operaciones en un SGBD MySQL que se detallan a continuación. No deben suponerse restricciones adicionales (en particular, cualquier atributo que pueda ser nulo sin violar restricciones de integridad, debe poder tomar ese valor).

1. Comando para crear la tabla "Especie". La columna "N_Pokedex" contiene valores enteros, "Nombre" es texto de longitud variable con un máximo de 10 caracteres y "Descripción" un texto de longitud variable con hasta 200 caracteres. Debe inferirse el tipo de las claves foráneas si las hubiera, se debe permitir eliminar las que la referencian y actualizar las referencias si se produce una actualización en la clave. *(2pt)*

2. Comandos para insertar las siguientes especies (una celda vacía implica que ese atributo debe ser nulo): *(1pt)*

N_Pokedex	Nombre	Descripción	Evoluciona
1	Bulbasaur		
2	Ivysaur		1
3	Venusaur		2

3. Consulta que retorne la cantidad total de ejemplares Pokémon (de cualquier especie y entrenados por cualquier entrenador) que se han encontrado: *(1 pt)*

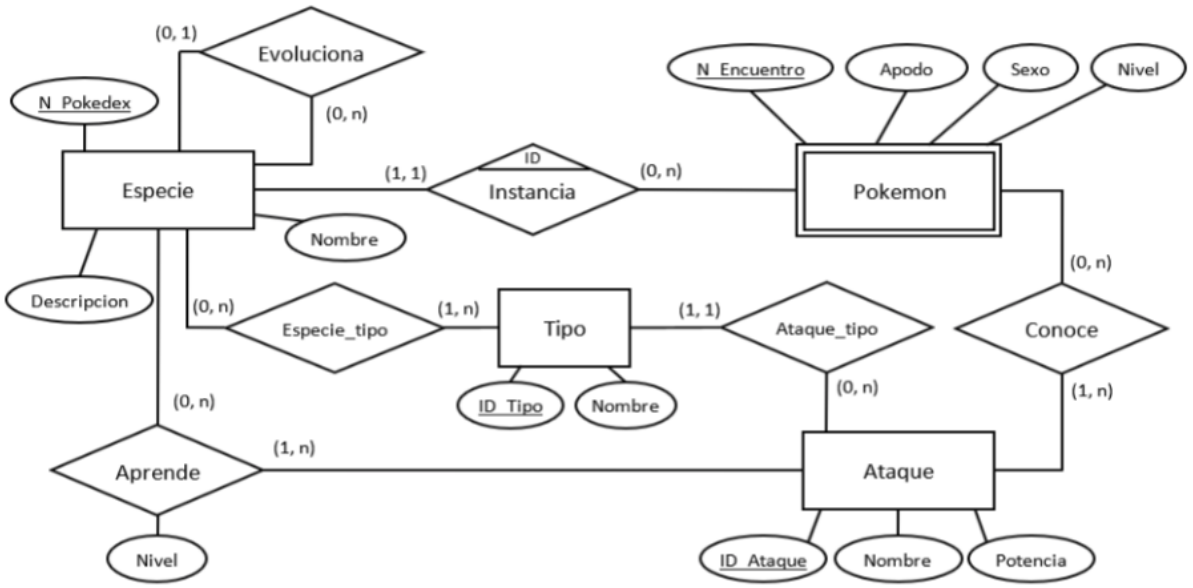
4. Consulta que retorne todos los ID de entrenador junto al nivel medio de los ejemplares de Pokémon que entrena cada entrenador, siempre que dicho nivel medio sea superior a 20. *(2 pt)*

5. Consulta que retorne la lista de los nombres de los ataques que puede aprender la especie “Pikachu” y el nivel al que aprende cada uno. *(2 pt)*

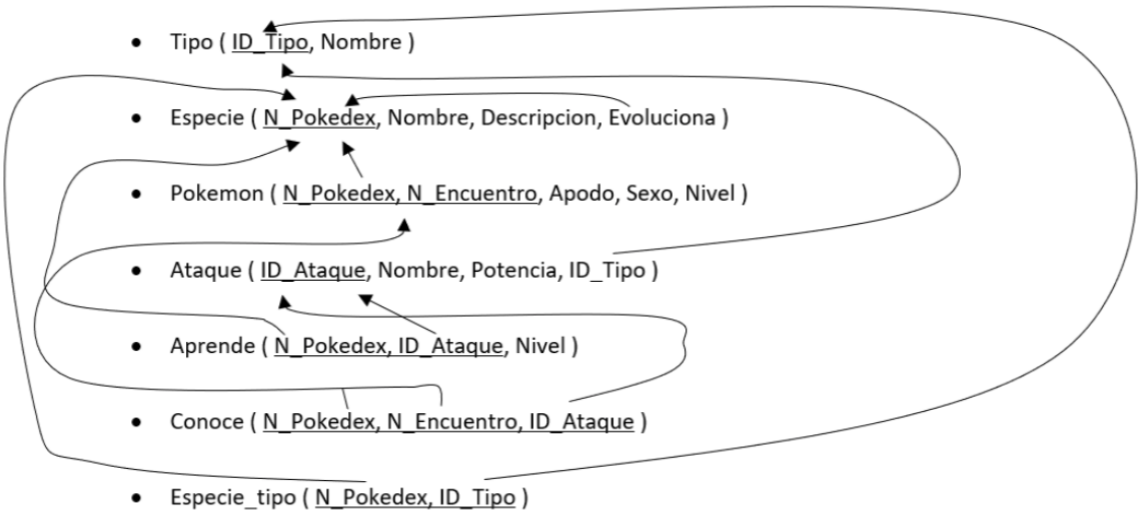
6. Queremos borrar a los jugadores que hacen trampas y explotan bugs del juego, por lo que se pide borrar a todos los entrenadores que entrenen un Pokémon de la especie “Mew”. *(2 pt)*

JULIO 2019

Considérese el siguiente diagrama que representa, de forma simplificada, una base de datos de un juego de Pokémon. La entidad "Especie" representa cada especie Pokémon (por ejemplo, la especie "Pidgey"), mientras que la entidad "Pokemon" representa un ejemplar concreto de una especie (por ejemplo, el primer ejemplar de Pidgey que aparece). A cada ejemplar de Pokémon se le asigna un "N_Encuentro" correlativo para su especie (es decir, puede haber un ejemplar de la especie Pidgey cuyo atributo "N_Encuentro" tenga un valor de 1 y un ejemplar de la especie Rattata cuyo atributo "N_Encuentro" también valga 1).



A continuación se da también el paso a tablas del diagrama anterior:

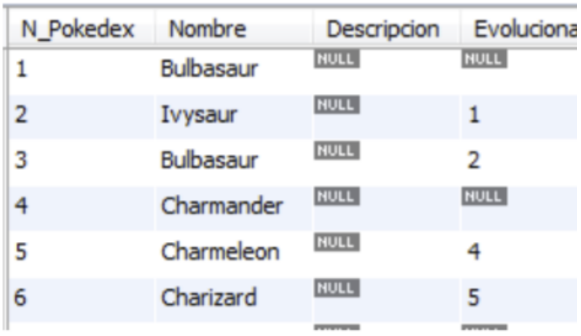


(*) Nótese que el atributo "Evoluciona" de la tabla Especie indica el N_Pokédex de la especia que evoluciona a dicha especie, ya que en algunos casos un mismo Pokémon puede evolucionar en varios distintos y, por tanto, no es posible guardar en un atributo la relación en sentido inverso.

Se pide escribir los comandos SQL para realizar las operaciones en un SGBD MySQL que se detallan a continuación. No deben suponerse restricciones adicionales (en particular, cualquier atributo que pueda ser nulo sin violar restricciones de integridad, debe poder tomar ese valor).

1. Comando para crear la tabla “Pokemon”. Todas las claves primarias de todas las tablas son valores enteros, la columna “Nivel” contiene valores enteros, “Apodo” es texto de longitud variable con un máximo de 10 caracteres y “Sexo” un solo carácter. Debe inferirse el tipo de las claves foráneas si las hubiera, se debe permitir eliminar una especie a la que se haga referencia (eliminando a su vez todos los ejemplares de dicha especie) y se deben actualizar las referencias si se produce una actualización de la clave. *(2 pt)*

2. Comandos para insertar los siguientes “Pokemon” (ejemplares) de la especie “Charmander”, teniendo en cuenta los datos ya existentes en la tabla “Especie” (se muestran en la imagen). Puede suponerse que no hay otros ejemplares de la especie “Charmander” ya introducidos. *(1 pt)*



N_Pokedex	Nombre	Descripcion	Evoluciona
1	Bulbasaur	NULL	NULL
2	Ivysaur	NULL	1
3	Bulbasaur	NULL	2
4	Charmander	NULL	NULL
5	Charmeleon	NULL	4
6	Charizard	NULL	5

Apodo	**Sexo**	**Nivel**
Llamita	H	5
Lagarto	M	10
CornFlames	M	15

3. Consulta que retorne cuántos ataques de tipo “Fuego” hay. No se permite el uso de subconsultas. *(1 pt)*

4. La especie “Eevee” es muy particular, ya que puede evolucionar en muchas especies distintas (evoluciones alternativas). Escribe una consulta que retorne los nombres de todas las especies que son evoluciones de la especie de nombre “Eevee”. Deben realizarse dos versiones: una que haga uso de una subconsulta y otra que no use ninguna subconsulta. (*Con subconsulta: 1pt/Sin subconsulta: 1.5pt*)

   CON SUBCONSULTA:

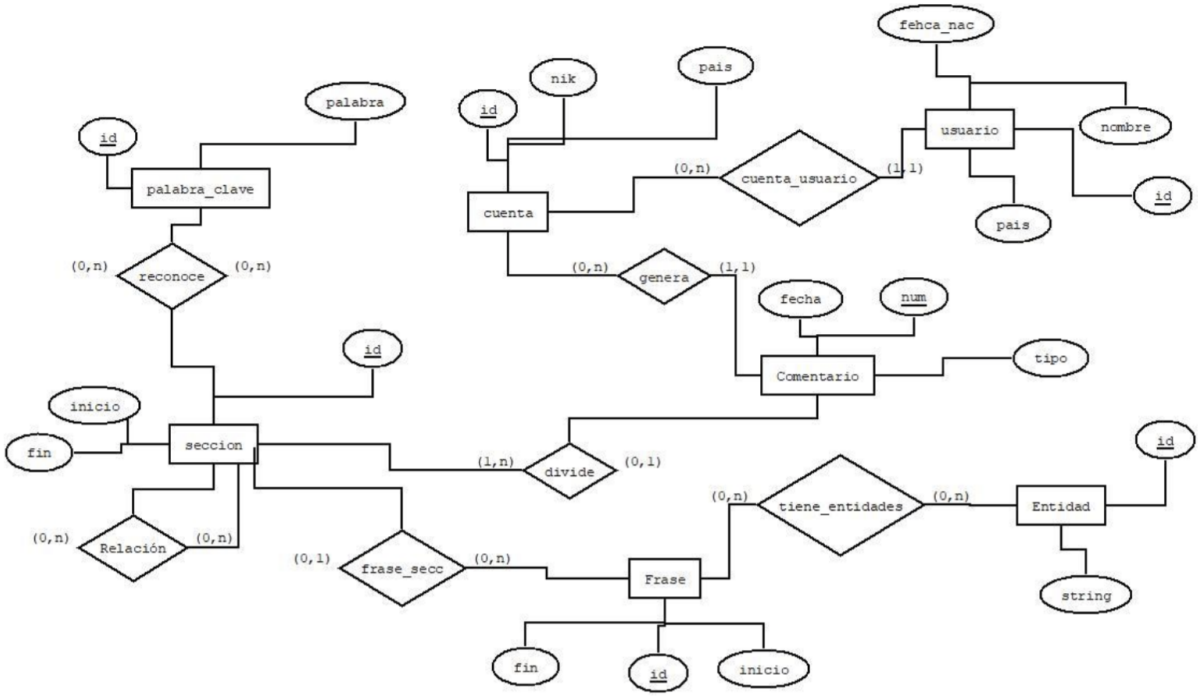
   SIN SUBCONSULTA:

5. Retornar el nombre de la especie que más ataques aprende y cuántos son esos ataques. No se permite el uso de subconsultas. (*2 pt*)

6. Se ha corrompido nuestra partida, resultando en ataques cuyo nombre de tipo es “Error”. Se pide borrar todos los ataques con este nombre de tipo. Este ejercicio debe realizarse obligatoriamente usando una subconsulta en la cláusula WHERE del comando para borrar. (*1.5 pt*)

JUNIO 2018

Dado el siguiente diagrama E/R, se pide:



Paso a tablas de la parte del diseño necesaria para realizar las siguientes consultas:

1. Listado que muestre el id de cada frase y el número de entidades que aparece en cada una de ellas, pero solo para frases que tengan más de 3 entidades.

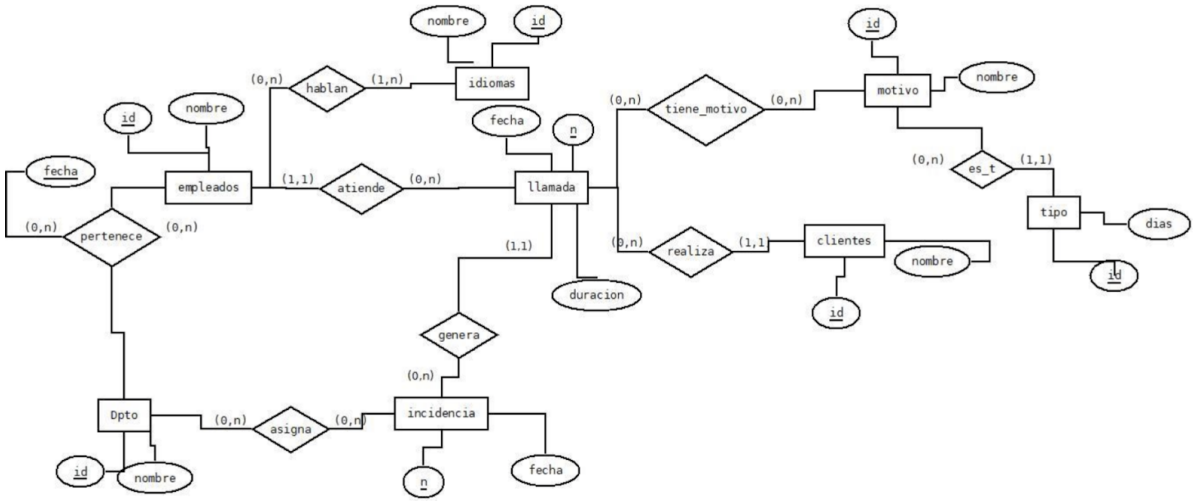
2. Listado de entidades (id y palabras) que contienen la palabra negativo.

3. Realizar la creación de la tabla resultante del paso a tablas que tenga restricciones de integridad referencial.

4. Modificar todas las palabras de entidades que las palabras estén vacías por la palabra “desconocido”.

JULIO 2018

Dado el siguiente diagrama E/R, se pide:



Transformación a modelo relacional (paso a tablas) de la parte del diagrama planteado que sea necesario para realizar las siguientes consultas: (*1 pt*)

1. Listar el número y la fecha de todas las incidencias asignadas al departamento con identificador = 123.

2. Listar el número de llamada junto con el número de partes de incidencia que ha generado.

3. Insertar el parte de incidencia número 27 de fecha 3 de noviembre de 1970 en la base de datos.

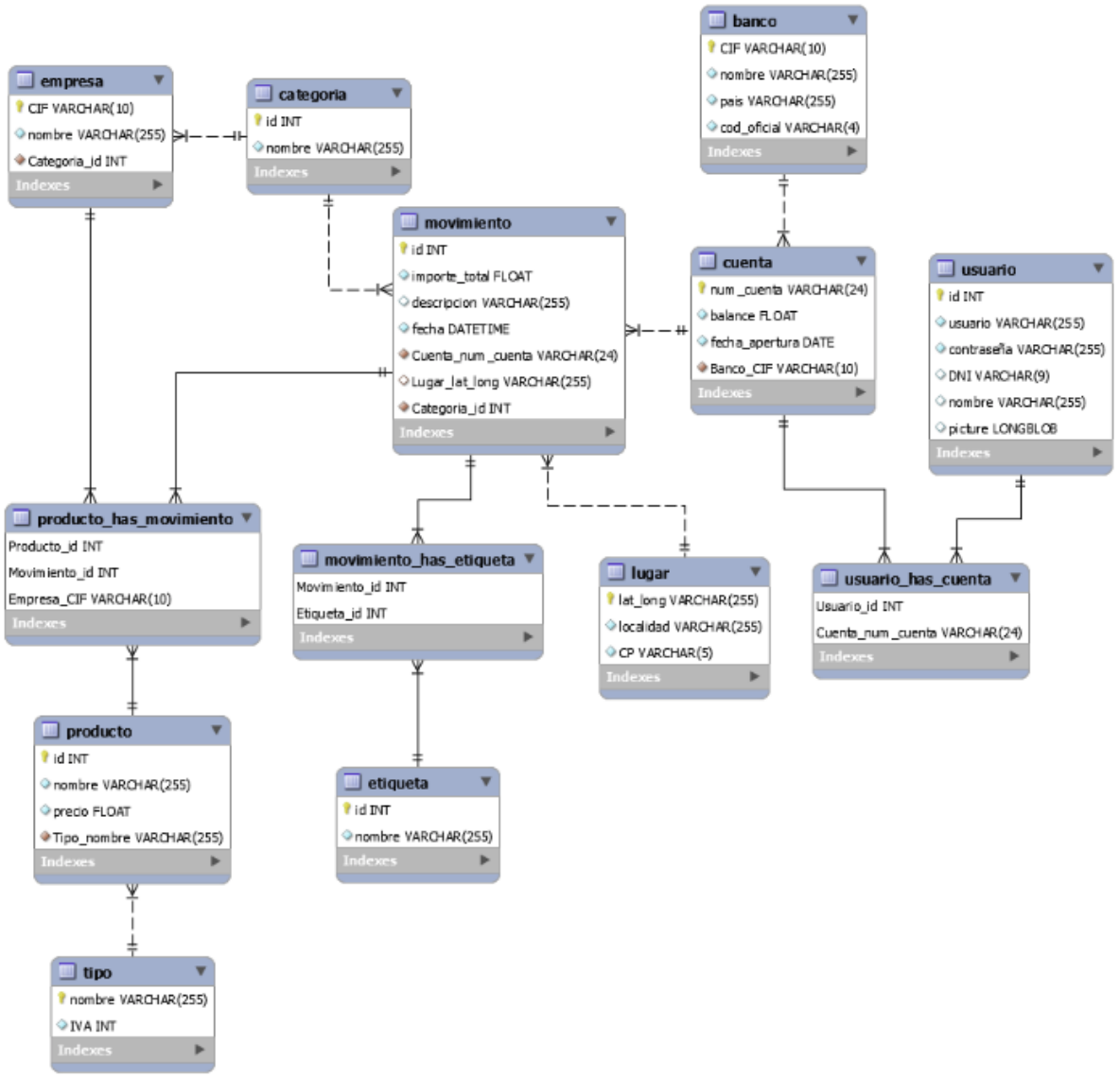
4. Crear la tabla de incidencias.

5. Visualizar el código de los departamentos que resolvieron incidencias asignadas entre 2016 y 2018.

6. Aumenta la duración en dos minutos de todas las llamadas que se recibieron en los meses de julio, agosto, septiembre, octubre y diciembre.

JUNIO 2020 - FINAL

Dado el diagrama de tablas adjunto, escoger la respuesta más correcta o eficiente para cada pregunta



Bloque 1: INSERT/DELETE/UPDATE

Bloque 1.1: Coger 1 del bloque

1. Si queremos añadir un nuevo banco situado en España con CIF A981341 y cuyo nombre es Bancofi con código ES33, ¿Cuál es la opción correcta?

A. insert into banco values('A981341', 'Bancofi', 'España', 'ES33')
B. insert into banco(CIF, nombre, pais, cod_oficial) values('A981341', 'Bancofi', 'España', 'ES33')
C. No es posible insertar en banco porque existe dependencia con cuenta
D. Las dos posibilidades de insert son correctas

2. Se quiere agregar un nuevo titular, cuyo ID es 7 a una cuenta existente (cuyo número de cuenta es 0078978012). ¿Cuál es la sentencia correcta?

A. insert into usuario_has_cuenta(id_usuario, num_cuenta) values (7, 0078978012)
B. insert into usuario_has_cuenta(Usuario_id, Cuenta_num_cuenta) values (7, 0078978012)
C. insert into usuario_has_cuenta(Usuario_id, Cuenta_num_cuenta) values (7, '0078978012')
D. insert into usuario_has_cuenta(Cuenta_num_cuenta, Usuario_id) values (0078978012, 7)

3. Si en la tabla cuenta la columna Banco_CIF fue definida como NOT NULL, ¿podemos agregar una cuenta sin asociarle un banco?

A.Si
B.No
C.Si,siempre que id de la tabla banco también sea NOT NULL.
D.Si, siempre que el id de la tabla banco sea del mismo tipo.

Bloque 1.2: Coger 1 del bloque

1. Tenemos un banco con nombre ‘Bancofi’ y CIF ‘A981341’. ¿Cuál es la forma más adecuada para borrarlo?

A.DELETE FROM banco WHERE CIF = 'A981341'
B.DELETE FROM banco WHERE CIF LIKE '%A981341%'
C.DELETE FROM banco WHERE nombre = 'Bancofi'
D.DELETE FROM banco WHERE nombre LIKE '%Bancofi%'

2. Se quiere desasociar todas las cuentas que tiene el usuario con ID 8. ¿Cuál es la sentencia correcta?

A.delete from usuario_has_cuenta where Usuario_id = 8
B.delete from usuario where id = 8
C.delete from cuenta Usuario_id = 8
D.delete from cuenta where id_usuario = 8

3. Por un error de programación en el acceso a la base de datos al no usar el tipo de Statement apropiado, se ha producido un ataque por inyección SQL y se han podido obtener las contraseñas de todos los DNIs que empiezan por '612'. Queremos cambiar todas sus contraseñas y poner de contraseña temporal la cadena 'newPwdAvoidSteal18.x2fQs3'. ¿Qué consulta se debe ejecutar para realizar esa actualización?

A.delete update usuario set contraseña = 'newPwdAvoidSteal18.x2fQs3' where id like '%612%'
B.update usuario set contraseña = 'newPwdAvoidSteal18.x2fQs3' where id like '612%'
C.update usuario set contraseña = 'newPwdAvoidSteal18.x2fQs3' where id like '%612'
D.Ninguna es correcta

Bloque 2: OPERADORES O TEORÍA

Bloque 2.1: Coger 1 del bloque

1. ¿Qué tipo de operación es DELETE?

A.DML B.DDL C.DTL D.DHL

2. ¿Qué tipo de operación es ALTER TABLE?

A.DML B.DDL C.DHL D.DSL

3. ¿Con qué operador puedo calcular la media de una columna?

A.AVERAGE B.MEAN C.MN D.AVG

Bloque 2.2: Coger 1 del bloque
1. ¿Con qué operador podríamos quitar los duplicados derivados de una consulta?

A.DISTINCT B.NOT DUPLICATES
C.ONLY ONE D.Ninguno de los anteriores

2. ¿Cómo podemos calcular el balance medio de las cuentas del banco cuyo CIF es '0126X'?

A.select average(balance) from cuenta where Banco_CIF = '0126X'
B.select avg(balance) from cuenta where Banco_CIF = '0126X'
C.select avg(balance) from banco where Banco_CIF = '0126X'
D.Ninguna de las anteriores

3. ¿Cuál es el símbolo que se usa en la consulta SELECT para obtener todas las columnas?
A.* B.& C.% D.@

Bloque 3: CONSULTAS SIMPLES

Bloque 3.1: Coger 1 del bloque

1. ¿Qué consulta te permite obtener todos los datos de todos los bancos (tabla banco)?

A.select all from banco
B.select all() from banco
C.select * from banco
D.select all(*) from banco

2. ¿Qué consulta permite saber cuántas empresas pertenecen al sector (categoría) de la alimentación?

A. select count(*) from empresa, categoria where empresa.Categoria_id = categoria.id and categoria.nombre = 'Alimentación'
B. select count(*) from empresa where categoria.nombre = 'Alimentación'
C. select count(*) from categoria where categoria.nombre = 'Alimentación'
D. Todas son válidas

3. ¿Qué consulta nos permite saber qué movimientos (id, importe total, descripción y fecha) tuvieron lugar en Oviedo, ordenados por fecha de forma descendiente?

A. select id, importe_total, descripcion, fecha from movimiento, lugar where lugar.localidad = 'Oviedo' order by fecha desc
B. select id, importe_total, descripcion, fecha from movimiento, lugar where movimiento.Lugar_lat_long = lugar.lat_long and lugar.localidad = 'Oviedo' order by fecha asc
C. Ninguna de las anteriores
D. select id, importe_total, descripcion, fecha from movimiento, lugar where movimiento.Lugar_lat_long = lugar.lat_long and lugar.localidad = 'Oviedo' order by fecha desc

Bloque 3.2: Coger 1 del bloque

1. ¿Con qué consulta podemos obtener los números de cuentas que han tenido movimientos en Madrid?

A. select num_cuenta from cuenta, movimiento, lugar where cuenta.num_cuenta = movimiento.Cuenta_num_cuenta and movimiento.Lugar_lat_long = lugar.lat_long and localidad = 'Madrid'
B. select num_cuenta from cuenta, movimiento where cuenta.num_cuenta = movimiento.Cuenta_num_cuenta and movimiento.Lugar_lat_long = lugar.lat_long and localidad = 'Madrid'
C. select num_cuenta from cuenta, movimiento, lugar where cuenta.num_cuenta = movimiento.Cuenta_num_cuenta and lugar.Lugar_lat_long = movimiento.lat_long and localidad = 'Madrid'

D. Todas son correctas

2. ¿Con que consulta podemos sacar de la forma más eficiente cuantos movimientos de productos ha tenido la empresa cuyo nombre es DIA?

A. select count(*) from empresa, movimiento where empresa.nombre = 'DIA' and movimiento.id= empresa.CIF
B. select count(*) from empresa, producto_has_movimiento where empresa.nombre = 'DIA' and empresa.CIF = producto_has_movimiento.Empresa_CIF
C. select count(*) from empresa, producto_has_movimiento, movimiento where empresa.nombre = 'DIA' and empresa.CIF = producto_has_movimiento.Empresa_CIF and producto_has_movimiento.Movimiento_id = movimiento.id
D. Ninguna es correcta

3. ¿Con qué consulta sacamos todos los movimientos (todos los datos) cuyo importe está entre 10 y 20 euros (ambos incluidos)?

A. select * from movimiento where importe_total > 10 or importe_total < 20
B. select * from movimiento where importe_total > 10 and importe_total < 20
C. select * from movimiento where importe_total between 10 and 20
D. select * from movimiento where importe_total between 10 or 20

Bloque 4: CONSULTAS COMPLEJAS

Bloque 4.1: Coger 1 del bloque

1. ¿Con que consulta podemos obtener que banco (nombre) tiene un mayor número de cuentas (quedarse con el primero usando como criterio de ordenación descendiente el nombre del banco)?

A.select nombre from banco, cuenta where banco.CIF = cuenta.Banco_CIF group by banco.CIF order by count(banco.CIF) desc, banco.nombre desc limit 1
B.select nombre from banco, cuenta where banco.CIF = cuenta.Banco_CIF group by count(banco.CIF) order by banco.nombre limit 1
C.select nombre from banco, cuenta where banco.CIF = cuenta.Banco_CIF group by banco.CIF order by count(banco.CIF) desc limit 1
D.Ninguna es válida

2. ¿Con qué consulta podemos obtener la cantidad de movimientos que tienen las empresas de una determinada categoría (obtener nombre de categoría y número de productos)?

A.select categoria.nombre, count(producto_has_movimiento.Producto_id) as cantidad from categoria, empresa, producto_has_movimiento where categoria.id = empresa.Categoria_id and empresa.CIF = producto_has_movimiento.Empresa_CIF group by categoria.id
B.select categoria.nombre, count(producto_has_movimiento.Producto_id) as cantidad from categoria, empresa, producto_has_movimiento where categoria.id = empresa.Categoria_id and empresa.CIF = producto_has_movimiento.Empresa_CIF group by count(producto_has_movimiento.Producto_id)
C.select categoria.nombre, count(producto_has_movimiento.Producto_id) as cantidad from categoria, empresa, producto_has_movimiento where categoria.id = empresa.Categoria_id and empresa.CIF = producto_has_movimiento.Empresa_CIF group by producto_has_movimiento.Producto_id
D.select categoria.nombre, count(producto_has_movimiento.Producto_id) as cantidad from categoria, empresa, producto_has_movimiento where categoria.id = empresa.Categoria_id and empresa.CIF = producto_has_movimiento.Empresa_CIF group by count(categoria.id)

3. Dada esta consulta: *select categoria.nombre, count(producto_has_movimiento.Producto_id) as cantidad from categoria, empresa, producto_has_movimiento where categoria.id = empresa.Categoria_id and empresa.CIF = producto_has_movimiento.Empresa_CIF group by categoria.id.* ¿Qué estamos queriendo obtener?

A.La cantidad de categorías que tienen los productos de las empresas (obteniendo nombre de categoría y cantidad).
B.La cantidad de productos asociados que tienen las empresas de una determinada categoría (obteniendo nombre de productos y cantidad).
C.La cantidad de movimientos que tienen las empresas de una determinada categoría (obteniendo nombre de categoría y número de movimientos).
D.La cantidad de productos con movimientos que tienen las empresas de una determinada categoría (obtener número de categorías y nombre de productos).

Bloque 4.2: Coger 1 del bloque

1. ¿Con qué consulta podemos obtener el número de movimientos del Banco BBVA en base al lugar (localidad) donde se produjeron ordenados de forma descendente por el número de movimientos de forma descendente?

A.select lugar.localidad, count(localidad) from lugar, movimiento, banco, cuenta where banco.nombre = 'BBVA' and banco.CIF = cuenta.Banco_CIF and cuenta.num_cuenta = movimiento.Cuenta_num_cuenta and movimiento.Lugar_lat_long = lugar.lat_long group by (localidad) order by count(localidad) desc
B.select lugar.localidad, count(localidad) from lugar, movimiento, banco, cuenta where banco.nombre = 'BBVA' and banco.CIF = cuenta.Banco_CIF and cuenta.num_cuenta = movimiento.Cuenta_num_cuenta and movimiento.Lugar_lat_long = lugar.lat_long group by (count(localidad)) order by count(localidad) desc
C.select lugar.localidad, group(localidad) from lugar, movimiento, banco, cuenta where banco.nombre = 'BBVA' and banco.CIF = cuenta.Banco_CIF and cuenta.num_cuenta = movimiento.Cuenta_num_cuenta and movimiento.Lugar_lat_long = lugar.lat_long countby (localidad) order by count(localidad) desc
D.select lugar.localidad, count(localidad) from lugar, movimiento, banco, cuenta where banco.nombre = 'BBVA' and banco.CIF = cuenta.Banco_CIF and and movimiento.Lugar_lat_long = lugar.lat_long group by (localidad) order by count(localidad) desc

2. ¿Qué queremos obtener al ejecutar esta consulta?: *select lugar.localidad, count(localidad) from lugar, movimiento, banco, cuenta where banco.nombre = 'BBVA' and banco.CIF = cuenta.Banco_CIF and cuenta.num_cuenta = movimiento.Cuenta_num_cuenta and movimiento.Lugar_lat_long = lugar.lat_long group by (localidad) order by count(localidad) desc*

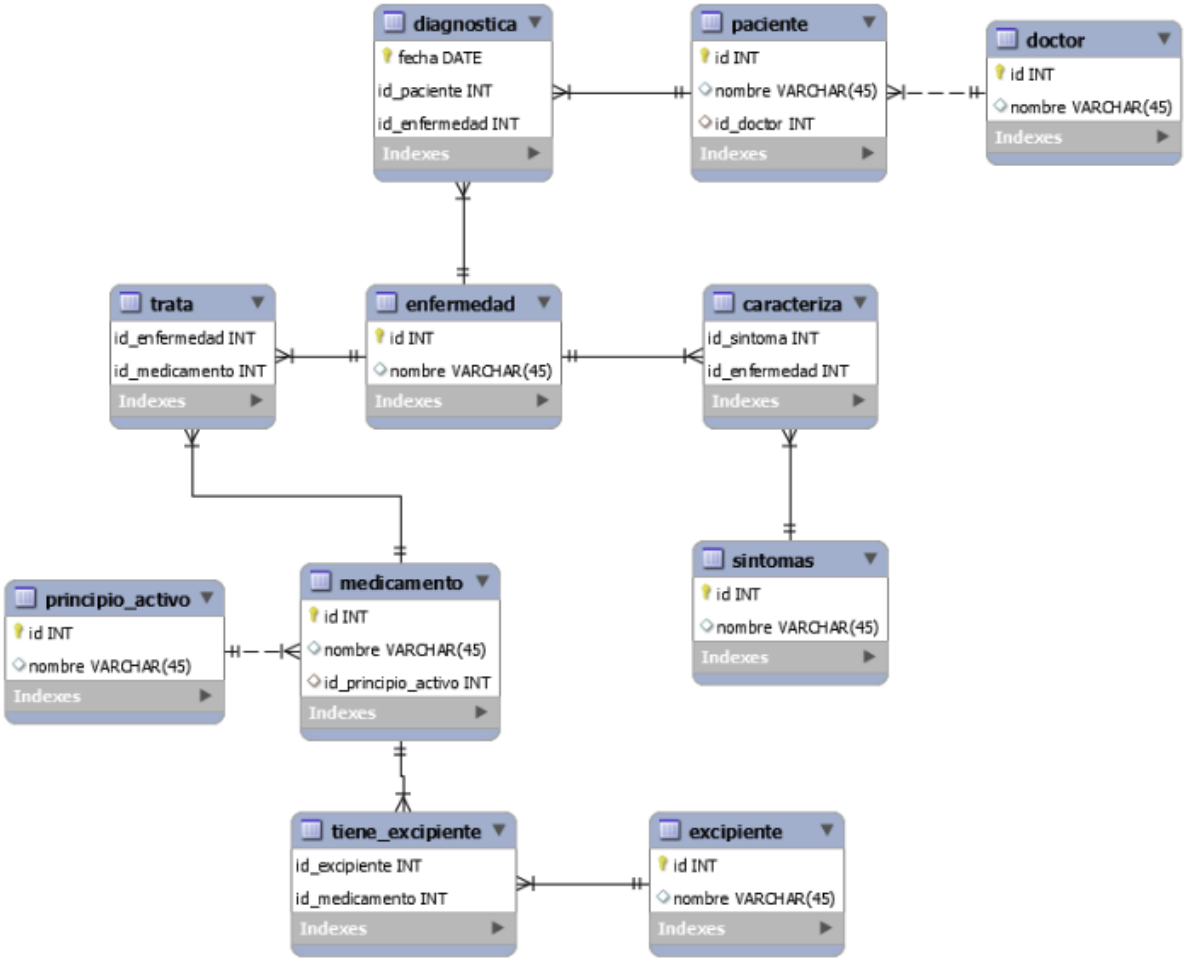
A.El número de cuentas del banco BBVA que pertenecen a una determinada localidad ordenado de forma descendente por el número.
B.El número de movimientos del banco BBVA que se efectuaron en una determinada localidad ordenado de forma descendente por el número.
C.El número de localidades donde hay una oficina del BBVA ordenado de forma descendente por el número.
D.Ninguna de las anteriores.

3. ¿Con que consulta puedo obtener el número de movimientos de cada cuenta que pertenece al usuario cuyo DNI es "5523X" (obtener número de cuenta y movimientos)?

A.Ninguna es correcta
B.select cuenta.num_cuenta, count(movimiento.Cuenta_num_cuenta) from usuario, usuario_has_cuenta, cuenta, movimiento where usuario.DNI = '5523X' and usuario.id = usuario_has_cuenta.Usuario_id and usuario_has_cuenta.Cuenta_num_cuenta = cuenta.num_cuenta and cuenta.num_cuenta = movimiento.Cuenta_num_cuenta group by (count(movimiento.Cuenta_num_cuenta))
C.select cuenta.num_cuenta, count(movimiento.Cuenta_num_cuenta) from usuario,usuario_has_cuenta, cuenta, movimiento where usuario.DNI = '5523X' and usuario.id = usuario_has_cuenta.Usuario_id and usuario_has_cuenta.Cuenta_num_cuenta = cuenta.num_cuenta and cuenta.num_cuenta = movimiento.Cuenta_num_cuenta count by (movimiento.Cuenta_num_cuenta)
D.select cuenta.num_cuenta, count(movimiento.Cuenta_num_cuenta) from usuario, usuario_has_cuenta, cuenta, movimiento where usuario.DNI = '5523X' and usuario.id = usuario_has_cuenta.Usuario_id and usuario_has_cuenta.Cuenta_num_cuenta = cuenta.num_cuenta and cuenta.num_cuenta = movimiento.Cuenta_num_cuenta group by (movimiento.Cuenta_num_cuenta)

JUNIO 2020

Dado el diagrama de tablas adjunto, escoger la respuesta más correcta o eficiente para cada pregunta



Bloque 1: INSERT/DELETE/UPDATE

Bloque 1.1: Coger 1 del bloque

1: Si queremos añadir un nuevo doctor con ID 20 y nombre Roberto, ¿cuál es la sentencia correcta?

A.INSERT INTO DOCTOR(id, nombre) VALUES (20, 'Roberto')
B.INSERT INTO DOCTOR(id, nombre) VALUES (20, Roberto)
C.INSERT INTO DOCTOR(id_doctor, nombre) VALUES (20, 'Roberto')
D.Ninguna de las anteriores

2: Si queremos añadir un nuevo doctor con ID 20 y nombre Roberto, ¿cuál es la sentencia correcta?

A.INSERT INTO DOCTOR(nombre,id) VALUES (20, 'Roberto')
B.INSERT INTO DOCTOR VALUES (20, Roberto)
C.INSERT INTO DOCTOR(nombre,id) VALUES (20, Roberto)
D.Ninguna de las anteriores

3: Si en la tabla paciente la columna id_doctor fue definida como NOT NULL, ¿podemos agregar un paciente sin asociarle un doctor?

A.Si
B.No
C.Si, siempre que id de la tabla doctor también sea NOT NULL.
D.Si, siempre que el id de la tabla doctor sea del mismo tipo.

Bloque 1.2: Coger 1 del bloque

1: Los pacientes de la BD contienen un solo apellido. Por un error en la BD, se introdujeron pacientes con apellido Perez asumiendo que el formato de la variable "nombre" es 'Nombre Apellido'. ¿Con qué sentencia podemos borrarlos?

A.DELETE FROM paciente where nombre like '%Perez';
B.DELETE FROM paciente where nombre = '%Perez';
C.DELETE FROM paciente where nombre like 'Perez%';
D.DELETE FROM paciente where nombre = 'Perez%';

2: Si queremos borrar todos los diagnósticos de todos los pacientes cuyo nombre es Amaia(asumiendo que los pacientes no tienen apellido), ¿qué sentencia debemos usar?

A.delete from diagnostica where id_paciente IN (select *from paciente where nombre = 'Amaia')
B.delete from diagnostica where id_paciente = (select id from paciente where nombre = 'Amaia')
C.delete from diagnostica where id_paciente IN (select id from paciente where nombre = 'Amaia')
D.Ninguna de las anteriores

3: ¿Qué operación debemos usar para borrar cualquier tipo de síntoma que conlleve fiebre en la definición/caracterización de las enfermedades?

A.delete from enfermedad where id_sintoma IN (select id from sintoma where nombre like '%fiebre%')
B.delete from caracteriza where id_sintoma IN (select id from sintomas where nombre like '%fiebre%')
C.delete from caracteriza where id_sintoma IN (select id from sintomas where nombre like 'fiebre%')
D.delete from enfermedad where id_sintoma =(select id from sintomas where nombre like 'fiebre%')

Bloque 2: OPERADORES O TEORÍA

Bloque 2.1: Coger 1 del bloque

1: ¿Qué tipo de operación es UPDATE?

A.DML B.DDL C.DTL D.DHL

2: ¿Qué tipo de operación es CREATE TABLE?

A.DML B.DDL C.DHL D.DSL

3: ¿Con qué operador puedo calcular la media de una columna?

A.AVERAGE B.MEAN C.MN D.AVG

Bloque 2.2: Coger 1 del bloque

1: ¿Con qué operador podríamos quedarnos solamente con un resultado?

A.FORCE B.CONSTRAINT C.LIMIT D.Ninguno de los anteriores

2: ¿Cómo podemos contar cuántos pacientes tenemos en la base de datos(tabla paciente)?

A.select count(*) from paciente
B.select count(*) from paciente where id is null
C.select enum(*) from paciente
D.Ninguna de las anteriores

3: ¿Cuál es el símbolo que se usa como 'wildcard' o comodín en las expresiones regulares de texto según lo visto en clase/transparencias en MySQL?

A.* B.& C.% D.@

Bloque 3: CONSULTAS SIMPLES

Bloque 3.1: Coger 1 del bloque

1: ¿Qué consulta te permite obtener todos los datos de todas las enfermedades(tabla enfermedad)?

A.select all from enfermedad
B.select all() from enfermedad
C.select * from enfermedad
D.select all(*) from enfermedad

2: ¿Qué consulta permite saber cuántos pacientes se llaman Pepe (asumiendo que el formato de la columna nombre es ‘Nombre Apellidos’)?

A.select * from paciente where nombre = ‘Pepe’
B.select count(*) from paciente where nombre like ‘Pepe %’
C.select count(*) from paciente where nombre like ‘%Pepe’
D.select * from paciente where nombre like ‘Pepe %’

3: ¿Qué consulta nos permite saber cuántos medicamentos tienen como principio activo el Paracetamol?

A.select count(*) from principio_activo where id_principio_activo = (select id from principio_activo where nombre = 'Paracetamol')
B.select * from medicamento where id_principio_activo = (selectid from principio_activo where nombre = 'Paracetamol')
C.Ninguna de las anteriores
D.select count(*) from principio_activo P, medicamento M where P.id=M. id_principio_activo and P.nombre='Paracetamol'

Bloque 3.2: Coger 1 del bloque

1: ¿Con qué consulta podemos obtener el nombre de los medicamentos de la base de datos ordenados por su nombre de la Z a la A?

A.select nombre from medicamento order by nombre desc
B.select nombre from medicamento order by nombre asc
C.select nombre from medicamento order by desc
D.select nombre from medicamento and order by nombre desc

2: ¿Con qué consulta podemos obtener el nombre de los medicamentos de la base de datos ordenados por su nombre de la Z a la A y quedarnos con el primer resultado?

A.select nombre from medicamento order by nombre desc limit 1
B.select nombre from medicamento order by nombre asc limit 0,1
C.select nombre from medicamento order by desc limit 1
D.select nombre from medicamento and order by nombre desc limit 0,1

3: ¿Con qué consulta podríamos sacar todos los datos de los excipientes Glucosa o Talco?

A.select * from excipiente where nombre = 'Glucosa' or 'Talco'
B.select * from excipiente where nombre = 'Glucosa' or nombre = 'Talco'
C.select * from excipiente where nombre = 'Glucosa' and nombre = 'Talco'
D.select * from excipiente where nombre = 'Glucosa' and 'Talco'

Bloque 4: CONSULTAS COMPLEJAS

Bloque 4.1: Coger 1 del bloque

1: (ANULADA)

2: ¿Con qué consulta podemos saber que enfermedades sufrió Amaia Perez y cuántas veces?

A.select enfermedad.nombre, count(enfermedad.id) as cuantas from enfermedad, paciente, diagnostica where paciente.nombre = 'Amaia Perez' and paciente.id = diagnostica.id_paciente and diagnostica.id_enfermedad = enfermedad.id group by enfermedad.id, enfermedad.nombre
B.select enfermedad.nombre, count(enfermedad.id) as cuantas from enfermedad, paciente, diagnostica where paciente.nombre = 'Amaia Perez' and paciente.id = diagnostica.id_paciente and diagnostica.id_enfermedad = enfermedad.id order by enfermedad.id, enfermedad.nombre
C.select enfermedad.nombre, count(paciente.id) as cuantas from enfermedad, paciente, diagnostica where paciente.nombre = 'Amaia Perez' and paciente.id = diagnostica.id_paciente and diagnostica.id_enfermedad = enfermedad.id group by enfermedad.id, enfermedad.nombre
D.Ninguna de las anteriores

3: ¿Con qué consulta podemos sacar los nombres de los medicamentos y sus principios activos ordenando los resultados en base al nombre del medicamento?

A.Las dos opciones son correctas
B.Ninguna es correcta
C.select medicamento.nombre as medicamento, principio_activo.nombre as pactivo from medicamento, principio_activo order by principio_activo.nombre
D.select medicamento.nombre as medicamento, principio_activo.nombre as pactivo from medicamento, principio_activo where medicamento.id_principio_activo = principio_activo.id order by medicamento.nombre

Bloque 4.2: Coger 1 del bloque

1: ¿Con qué consulta podemos sacar los nombres de las enfermedades y sus síntomas ordenando los resultados en base al nombre de la enfermedad?
A.select enfermedad.nombre as enfermedad, sintomas.nombre as sintoma from enfermedad, sintomas, caracteriza where caracteriza.id_enfermedad =enfermedad.id and sintomas.id =caracteriza.id_sintoma order by enfermedad.nombre
B.select enfermedad.nombre as enfermedad, sintomas.nombre as sintoma from enfermedad, sintomas group by enfermedad.nombre
C.select enfermedad.nombre as enfermedad, sintomas.nombre as sintoma from enfermedad, sintomas, caracteriza where caracteriza.id_enfermedad = enfermedad.id and sintomas.id = caracteriza.id_sintoma

D.select enfermedad.nombre as enfermedad, sintomas.nombre as sintoma from enfermedad, sintomas order by enfermedad.id and group by enfermedad.nombre

2: ¿Con qué consulta podemos sacar las enfermedades que tienen como síntoma “Fotofobia”?

A.select enfermedad.nombre from enfermedad, caracteriza where caracteriza.id_sintoma = (select id from sintomas where nombre = 'Fotofobia') and caracteriza.id_enfermedad = enfermedad.id_enfermedad
B.select enfermedad.nombre from enfermedad, caracteriza where caracteriza.id_sintoma = (select nombre from sintomas where nombre = 'Fotofobia') and caracteriza.id_enfermedad = enfermedad.id
C.select enfermedad.nombre from enfermedad, caracteriza where caracteriza.id_sintoma IN (select id from sintomas where nombre = 'Fotofobia') and caracteriza.id_enfermedad = enfermedad.id
D.Ninguna de las anteriores

3: ¿Con qué consulta puedo sacar el número de síntomas que caracteriza cada enfermedad?

A.select enfermedad.nombre, count(caracteriza.id_enfermedad) as numSintomas from enfermedad, caracteriza order by caracteriza.id_enfermedad
B.select enfermedad.nombre, count(caracteriza.id_enfermedad) as numSintomas from enfermedad, caracteriza where enfermedad.id = caracteriza.id_enfermedad order by caracteriza.id_enfermedad
C.select enfermedad.nombre, count(caracteriza.id_enfermedad) as numSintomas from enfermedad, caracteriza group by caracteriza.id_enfermedad
D.select enfermedad.nombre, count(caracteriza.id_enfermedad) as numSintomas from enfermedad, caracteriza where enfermedad.id = caracteriza.id_enfermedad group by caracteriza.id_enfermedad