RESUMEN:

DDL, DML y DCL

Lenguaje de Definición de Datos (DDL)

Es un lenguaje de programación para definir estructuras de datos, proporcionado por los sistemas gestores de bases de datos, en este caso MySql.

Para definir las estructura disponemos de tres sentencias:

- **CREATE**, se usa para crear una base de datos, tabla, vistas, etc.
- **ALTER**, se utiliza para modificar la estructura, por ejemplo añadir o borrar columnas de una tabla.
- **DROP**, con esta sentencia, podemos eliminar los objetos de la estructura, por ejemplo un índice o una secuencia.

Lenguaje de Manipulación de Datos (DML)

Utilizando instrucciones de SQL, se le permite a los usuarios introducir datos para posteriormente realizar tareas de consultas o realizar modificaciones en la base.

Los elementos que se utilizan para manipular los datos, son los siguientes:

- **SELECT**, esta sentencia se utiliza para realizar consultas sobre los datos.
- **INSERT**, con esta instrucción podemos insertar los valores en una base de datos.
- **UPDATE**, sirve para modificar los valores de uno o varios registros.
- **DELETE**, se utiliza para eliminar las finas de una tabla

Lenguaje de Control de Datos (DCL)

Estos comandos permiten al Administrador del sistema gestor de base de datos, controlar el acceso a los objetos, es decir, podemos otorgar o denegar permisos a uno o más roles para realizar determinadas tareas.

Los comandos para controlar los permisos son los siguientes:

- **GRANT**, permite otorgar permisos.
- **REVOKE**, elimina los permisos que previamente se han concedido.

Crear una base de datos

```
CREATE DATABASE mi_negocio;
```

#	Time	Action	Message	Duration / Fetch
1	01:12:18	CREATE DATABASE mi_negocio	1 row(s) affected	0,00062 sec

Usar una base de datos

```
USE mi_negocio;
```

#	Time	Action	Message	Duration / Fetch
1	01:13:08	USE mi_negocio	0 row(s) affected	0,00030 sec

Creo una tabla empezando por las que no tienen clave foránea

```
CREATE TABLE categoria(

id_categoria int not null auto_increment,

detalle_categoria varchar(50) not null,

PRIMARY KEY (id_categoria)

)
```

- Creo la tabla con sus columnas.
- Creo un identificador único y lo hago auto_increment (es un número que se va a cargar solo aumentando a medida que se insertan más registros)
- Y por último con el comando primary key hago que id_categoria sea la clave primaria de la tabla.

#	Time	Action	Message	Duration / Fetch
1	01:24:50	CREATE TABLE categoria(id_categoria int not null auto_incre...	0 row(s) affected	0,269 sec

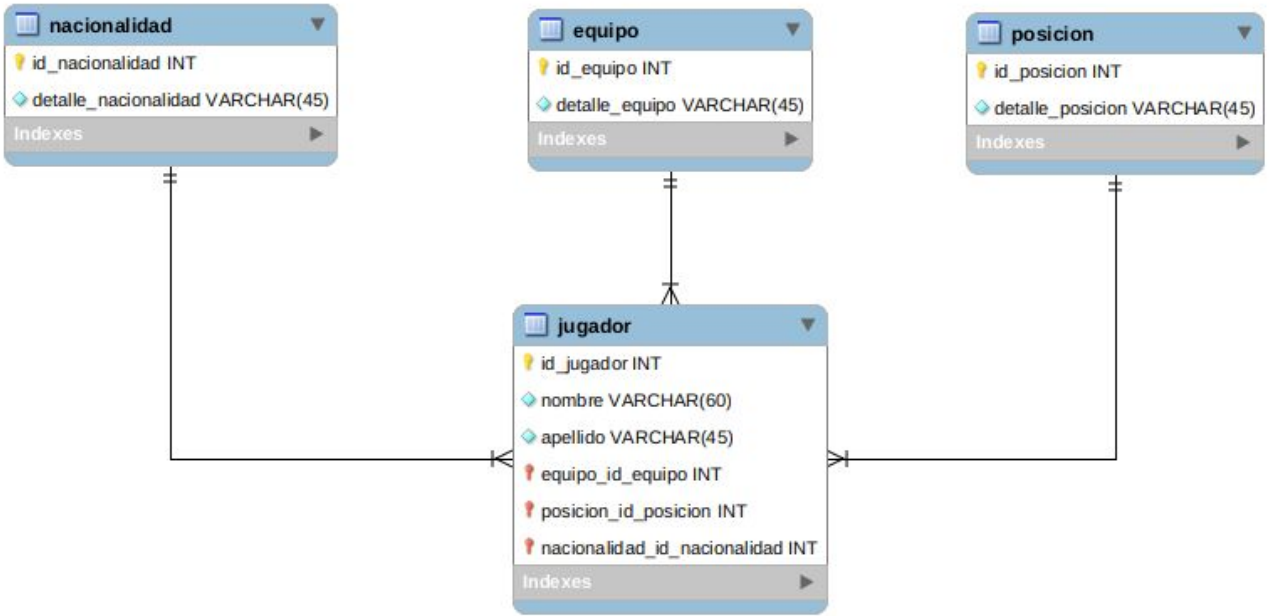
Creo las demás tablas y hago la referencias con sus claves foráneas

```
CREATE TABLE producto(

id_producto int not null auto_increment,

detalle_producto varchar(50) not null,

id_categoria int,

PRIMARY KEY(id_producto),

FOREIGN KEY (id_categoria) REFERENCES categoria(id_categoria)

)
```

- Creo la tabla con sus columnas.
- Creo un identificador único y lo hago auto_increment (es un número que se va a cargar solo aumentando a medida que se insertan más registros)
- Con el comando primary key hago que id_producto sea la clave primaria de la tabla.
- Y por último con el comando FOREIGN KEY conecto las dos tablas y hago que id_categoria en la tabla producto sea la clave foránea y que haga referencia a la tabla categoría al campo id_categoria

#	Time	Action	Message	Duration / Fetch
1	01:25:40	CREATE TABLE producto(id_producto int not null auto_increm...	0 row(s) affected	0,269 sec

Crear las tablas definidas en el problema planteado la semana anterior



Insertar datos tabla posición

Primero tenemos que insertar datos en las tablas que no tienen claves foráneas (Porque para que esa clave foránea exista primero tiene que estar cargada en otra tabla:

Sintaxis:

INSERT INTO <nombre_de_tabla> (campos de la tabla) VALUES (los valores que queremos insertar)
INSERT INTO posicion (detalle_posicion) VALUES ('Extremo derecho')
INSERT INTO posicion (detalle_posicion) VALUES ('Extremo izquierdo'), ('Arquero'), ('Defensa central'), ('Centro delantero')

id_posicion	detalle_posicion
1	Extremo derecho
2	Extremo izquierdo
3	Arquero
4	Defensa central
5	Centro delantero

Consultar datos

Consultar datos

Vamos a empezar con consultas simple, en la próxima clase vamos a agregar mas cosas a las querys (consultas)

Sintaxis:

SELECT * FROM <nombre_de_tabla>

SELECT * FROM equipo

Actualizar datos

Actualizar datos

Vamos a empezar con consultas simple, en la próxima clase vamos a agregar mas cosas a las querys (consultas)

Sintaxis:

UPDATE <nombre_tabla>
SET <columna> = <valor>, <columna> = <valor>, ...
WHERE <condición para que encuentre la fila que necesitamos, en este caso vamos a usar el id_jugador>;

UPDATE jugador SET nombre = 'Neymar', apellido= 'Jr' WHERE id_jugador = 3;

Eliminar datos

Eliminar datos

Vamos a poder eliminar registros si no están relacionados con otras tablas, por ejemplo podemos eliminar jugadores pero no podemos eliminar nacionalidades que están siendo usadas en algún jugador. Para eliminar una nacionalidad tendría que cambiarle la nacionalidad al jugador primero o eliminar directamente al jugador.

Sintaxis:

DELETE FROM <nombre_de_tabla> where <condición, en este caso el id_jugador> = <número de id>

DELETE FROM jugador WHERE id_jugador = 8

Modificar el auto_increment

Ahora si insertamos un nuevo jugador se va a insertar con el id_jugador n° 9 porque nosotros pusimos que ese campo sea auto_increment, si analizando el código vemos que no nos va a generar ningún problema podemos modificar ese auto_increment para que empiece desde el id que tenia Aguero para que no tengamos ese salto en los los id de 7 a 9, esto lo hacemos con este comando:

ALTER TABLE <nombre_de_la_tabla> AUTO_INCREMENT = <n°_que_queremos_que_tenga_el_proximo_id_jugador>

ALTER TABLE jugador AUTO_INCREMENT = 8

#	Time	Action	Message	Duration / Fetch
1	02:53:10	ALTER TABLE jugador AUTO_INCREMENT = 8	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0,104 sec

Agregar y eliminar columnas de una tabla

Agregar columnas

Sintaxis:

ALTER TABLE <nombre_de_tabla> ADD COLUMN <nombre_de_columna> <tipo_de_dato> AFTER <nombre_de_la_columna>;

ALTER TABLE jugador ADD COLUMN sueldo_mensual int(20) AFTER apellido;

ALTER TABLE jugador ADD COLUMN valor_mercado int(20) AFTER sueldo_mensual;

Agregar columnas

Como ven se creo la nueva columna con los valores vacíos, ahora habría que hacer un update por cada id_jugador para agregarle el sueldo a cada uno.

id_jugador	nombre	apellido	sueldo_mensual	valor_mercado	equipo_id_equipo	posicion_id_posicion	nacionalidad_id_nacionalidad
1	Lionel Andres	Messi	NULL	NULL	1	1	1
2	Cristiano	Ronaldo	NULL	NULL	2	2	2
3	Neymar	Junior	NULL	NULL	3	2	3
4	Jan	Oblak	NULL	NULL	4	3	4
5	Eden	Hazard	NULL	NULL	5	2	5
6	Marc-Andre	ter Stegen	NULL	NULL	1	3	6
7	Virgil	Van Dijk	NULL	NULL	6	4	7
8	Paulo	Dybala	NULL	NULL	2	6	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Eliminar columnas

Supongamos que la columna valor_mercado no nos sirve, la podemos eliminar de la siguiente forma:

Sintaxis:

ALTER TABLE <nombre_de_la_tabla> DROP COLUMN <nombre_de_la_columna>

ALTER TABLE jugador DROP COLUMN valor_mercado