APELLIDOS, NOMBRE / COGNOMS, NOM **DNI / NIE**

Moreno Bolivar, Jose Manuel 52659570N

A

Gracias por empezar cada ejercicio en una nueva cara

[5 PUNTOS] EJERCICIO 1: MODELADO LÓGICO RELACIONAL

Realiza el paso a tablas en tercera forma normal (3FN). ***Encuadrado las tablas en 3FN***

//Relacción BASE + HANGAR.
//Cardinalidad 1:N
//Atención: Participación (1,1) de todo Hangar tiene asignada una Base como maximo y como minimo por eso NOT NULL de cod_base en HANGAR.

BASE (cod_base, nombre, fecha_ini, fecha_fin)
PK: {cod_base}

PERDIDA SEMÁNTICA: No existen perdidas semánticas.

1FN: No existen atributos derivados, multievaluados ni compuestos.
2FN: Está en 1FN, tiene PK simple, por lo que todos sus atributos dependen de la PK completa.
3FN: Está en 2FN y no existen atributos con dependencias transitivas.

HANGAR (nombre, descripción, cod_base → Base)
PK: {nombre}
FK: {cod_base -> BASE(cod_base)}
NOT NULL: {cod_base -> BASE(cod_base)}

PERDIDA SEMÁNTICA: No existen perdidas semánticas.

****

//Relacción HANGAR + NAVE.
//Cardinalidad 1:N
//Atención: Participación (1,1) de toda NAVE tiene asignada un Hangar como máximo y como mínimo por eso NOT NULL de nombre_hangar en NAVE.

HANGAR (nombre, descripción, cod_base → Base)
PK: {nombre}
FK: {cod_base -> BASE(cod_base)}
NOT NULL: {cod_base -> BASE(cod_base)}

PERDIDA SEMANTICA: Habrá que verificar que todo hangar tenga como mínimo una nave a guardar.

1FN: No existen atributos derivados, multievaluados ni compuestos.
2FN: Está en 1FN, tiene PK simple, por lo que todos sus atributos dependen de la PK completa.
3FN: Está en 2FN y no existen atributos con dependencias transitivas.

NAVE (matricula, nombre, operativa, nombre_hangar → HANGAR(nombre))
PK: {matricula}
FK: {nombre_hangar -> HANGAR(nombre)}
NOT NULL: {nombre_hangar -> HANGAR(nombre)}

PERDIDA SEMÁNTICA: No existen perdidas semánticas.

****

//Relacción NAVE+TIE FIGHTER+CRUCERO.
//Especialización Parcial Disjunta
//Atención: Se trata de una especialización. La PK será la misma para las tres.

NAVE (matricula, nombre, operativa, nombre_hangar → HANGAR(nombre))
PK: {matricula}
FK: {nombre_hangar -> HANGAR(nombre)}
NOT NULL: {nombre_hangar -> HANGAR(nombre)}

PERDIDA SEMÁNTICA: Si la nave es de un tipo de especificación no puede ser de otro tipo.

1FN: No existen atributos derivados, multievaluados ni compuestos.
2FN: Está en 1FN, tiene PK simple, por lo que todos sus atributos dependen de la PK completa.
3FN: Está en 2FN y no existen atributos con dependencias transitivas.

TIE FIGHTER (matricula → NAVE(matricula), biplaza, armamento)
PK: {matricula -> NAVE(matricula)}
FK: {matricula -> NAVE(matricula)}

1FN: Existe el atributo multievaluado ARMAMENTO que nos obliga a crear otra tabla y borrar el atributo en la tabla TIE FIGHTER.

TIE FIGHTER (matricula → NAVE(matricula), biplaza)
PK: {matricula -> NAVE(matricula)}
FK: {matricula -> NAVE(matricual)}

//El atributo armamento es multievaluado por lo que nos oblica a crear otra tabla. Y borrar el atributo de la tabla TIE FIGHTER.
2FN: Está en 1FN, tiene PK simple, por lo que todos sus atributos dependen de la PK completa.
3FN: Está en 2FN y no existen atributos con dependencias transitivas.

ARMAMENTO (armamento, matricula → TIE FIGHTER(matricula))
PK: armamento, matricula → TIE FIGHTER(matricula)}
FK: ´{matricula → TIE FIGHTER(matricula)}

1FN: No existen atributos derivados, multievaluados ni compuestos.
2FN: Está en 1FN, tiene PK simple, por lo que todos sus atributos dependen de la PK completa.
3FN: Está en 2FN y no existen atributos con dependencias transitivas.

CRUCERO (matricula → NAVE(matricula), escudo, carga)
PK: {matricula -> NAVE(matricula)}
FK: {matricula -> NAVE(matricula)}

PERDIDA SEMÁNTICA: No existen perdidas semánticas.

****

//Relacción PILOTO+PILOTO.
//Cardinalidad 1:N
//Atención: Al ser una relación reflexiva aparece un nuevo atributo, en este caso supervisado_por.

PILOTO (licencia, fecha_li, nom_completo_nombre, nom_completo_apellido, supervisado_por → PILOTO(licencia))
PK: {licencia}
FK: {supervisado_por → PILOTO(licencia)}

PERDIDA SEMÁNTICA: No existen perdidas semánticas.

****

//Relacción PILOTO+CRUCERO.
//Cardinalidad 1:N
//Atención: Participación (1,1) de todo CRUCERO tiene asignado un PILOTO como máximo y como mínimo por eso NOT NULL de licencia → PILOTO en CRUCERO.

PILOTO (licencia, fecha_li, nom_completo_nombre, nom_completo_apellido, supervisado_por → PILOTO(licencia))
PK: {licencia}
FK: {supervisado_por → PILOTO(licencia)}

PERDIDA SEMÁNTICA: No existen perdidas semánticas.

1FN: No existen atributos derivados, multievaluados pero si compuestos por lo que hay que se eliminar **nom_compuesto** y se dejar los atributos, **nombre** y **apellidos**.

PILOTO (licencia, fecha_li, nombre, apellido, supervisado_por → PILOTO(licencia))
PK: {licencia}
FK: {supervisado_por → PILOTO(licencia)}

PERDIDA SEMÁNTICA: No existen perdidas semánticas.

2FN: Está en 1FN, tiene PK simple, por lo que todos sus atributos dependen de la PK completa.
3FN: Está en 2FN y no existen atributos con dependencias transitivas.

CRUCERO (matricula → NAVE, escudo, carga, licencia → PILOTO(licencia))
PK: {matricula -> NAVE(matricula)}
FK: {matricula -> NAVE(matricula)}
FK: {licencia -> PILOTO(licencia)}
NOT NULL: {licencia -> PILOTO(licencia)}

PERDIDA SEMÁNTICA: No existen perdidas semánticas.

****

//Relacción CRUCERO+TRANSPORTE.
//Cardinalidad N:M nos obliga a hacer una tercera tabla que en este caso será REALIZAR
//Atención: Participación (1,n) de todo TRANSPORTE tiene asignado un CRUCERO como mínimo es una perdida semántica.

CRUCERO (matricula → NAVE, escudo, carga, licencia → PILOTO(licencia))
PK: {matricula -> NAVE(matricula)}
FK: {matricula -> NAVE(matricula)}
FK: {licencia -> PILOTO(licencia)}
NOT NULL: {licencia -> PILOTO(licencia)}

PERDIDA SEMÁNTICA: No existen perdidas semánticas.

1FN: No existen atributos derivados, multievaluados ni compuestos.
2FN: Está en 1FN, tiene PK simple, por lo que todos sus atributos dependen de la PK completa.
3FN: Está en 2FN y no existen atributos con dependencias transitivas.

TRANSPORTE (codigo, objetivo)
PK: {codigo}

PERDIDA SEMANTICA: Todo transporte debe tener un crucero asociado.

1FN: No existen atributos derivados, multievaluados ni compuestos.
2FN: Está en 1FN, tiene PK simple, por lo que todos sus atributos dependen de la PK completa.
3FN: Está en 2FN y no existen atributos con dependencias transitivas.

REALIZAR (código → TRANSPORTE, matricula → CRUCERO, fecha_salida, fecha_llegada)
PK: {codigo -> TRANSPORTE(codigo), matricula -> CRUCERO(matricula)}
FK: {codigo -> TRANSPORTE(codigo)}
FK: {matricula -> CRUCERO(matricula)}

PERDIDA SEMÁNTICA: No existen perdidas semánticas.

1FN: No existen atributos derivados, multievaluados ni compuestos.
2FN: Está en 1FN, tiene PK compuesta, pero todos los atributos apuntan a la PK completa por lo que no es necesario crear otra tabla.
3FN: Está en 2FN y no existen atributos con dependencias transitivas.

****

//Relacción TRANSPORTE+COSTE.
//Cardinalidad 1:N
//Atención: Participación (1,n) de todo TRANSPORTE tiene asignado un COSTE como mínimo es una perdida semántica.

TRANSPORTE (codigo, objetivo)
PK: {codigo}

PERDIDA SEMANTICA: Revisar que todo transporte tenga un coste

1FN: No existen atributos derivados, multievaluados ni compuestos.
2FN: Está en 1FN, tiene PK simple, por lo que todos sus atributos dependen de la PK completa.
3FN: Está en 2FN y no existen atributos con dependencias transitivas.

COSTE (codigo → TRANSPORTE(codigo), linea, concepto, importe)
PK: {linea, codigo → TRANSPORTE(codigo)}
FK: {codigo → TRANSPORTE(codigo)}

PERDIDA SEMÁNTICA: No existen perdidas semánticas.

1FN: No existen atributos derivados, multievaluados ni compuestos.
2FN: Está en 1FN, tiene PK compuesta, pero todos los atributos apuntan a la PK completa por lo que no es necesario crear otra tabla.
3FN: Está en 2FN y no existen atributos con dependencias transitivas.
Paso a fomato fisico DDL

[2 PUNTOS] EJERCICIO 2: MODELADO FÍSICO DDL (CREACIÓN DE METADATOS)

Indica las sentencias necesarias para incorporar las tablas del ejercicio anterior en MySQL.

BASE (cod_base, nombre, fecha_ini, fecha_fin)
PK: {cod_base}

```
CREATE TABLE base (
cod_base CHAR(6),
nombre VARCHAR(30),
fecha_ini DATE,
fecha_fin DATE,
CONSTRAINT ba_cod_pk PRIMARY KEY (cod_base)
);
```

HANGAR (nombre, descripción, cod_base → Base)
PK: {nombre}
FK: {cod_base -> BASE(cod_base)}
NOT NULL: {cod_base -> BASE(cod_base)}

```
CREATE TABLE hangar (
nombre CHAR(6),
descripcion VARCHAR(30),
cod_base CHAR(6) NOT NULL,
CONSTRAINT ha_nom_pk  PRIMARY KEY(nombre),
CONSTRAINT ha_cod_fk FOREIGN KEY (cod_base) REFERENCES base (cod_base)
);
```

NAVE (matricula, nombre, operativa, nombre_hangar → HANGAR(nombre))
PK: {matricula}
FK: {nombre_hangar -> HANGAR(nombre)}
NOT NULL: {nombre_hangar -> HANGAR(nombre)}

```
CREATE TABLE nave (
matricula CHAR(10),
nombre VARCHAR(30),
operativa VARCHAR(2),
nombre_hangar CHAR(6) NOT NULL,
CONSTRAINT na_ma_pk PRIMARY KEY(matricula),
CONSTRAINT na_nom_fk FOREIGN KEY(nombre_hangar) REFERENCES hangar (nombre),
CONSTRAINT na_op_ck CHECK (operativa IN ('si','no'))
);
```

TIE FIGHTER (matricula → NAVE(matricula), biplaza)
PK: {matricula -> NAVE(matricula)}
FK: {matricula -> NAVE(matricual)}

```
CREATE TABLE tie_fighter (
matricula CHAR(10),
biplaza VARCHAR(2),
CONSTRAINT tie_ma_pk PRIMARY KEY(matricula),
CONSTRAINT tie_ma_fk FOREIGN KEY (matricula) REFERENCES nave(matricula),
CONSTRAINT na_op_ck CHECK (biplaza IN ('si','no'))
);
```

ARMAMENTO (armamento, matricula → TIE FIGHTER(matricula))
PK: armamento, matricula → TIE FIGHTER(matricula)}
FK: ´{matricula → TIE FIGHTER(matricula)}

```
CREATE TABLE armamento (
armamento VARCHAR(30),
matricula CHAR(10),
CONSTRAINT ar_ma_pk PRIMARY KEY(armamento, matricula),
CONSTRAINT ar_ma_fk FOREIGN KEY (matricula) REFERENCES tie_fighter(matricula),
);
```

PILOTO (licencia, fecha_li, nombre, apellido, supervisado_por → PILOTO(licencia))
PK: {licencia}
FK: {supervisado_por → PILOTO(licencia)}

```
CREATE TABLE piloto (
licencia CHAR(5),
fecha_li DATE,
nombre VARCHAR(15),
apellido VARCHAR(15),
supervisado_por CHAR(5),
CONSTRAINT pi_li_pk PRIMARY KEY (licencia),
CONSTRAINT pi_su_fk FOREIGN KEY (supervisado_por) REFERENCES piloto(licencia)
);
```

CRUCERO (matricula → NAVE, escudo, carga, licencia → PILOTO(licencia))
PK: {matricula -> NAVE(matricula)}
FK: {matricula -> NAVE(matricula)}
FK: {licencia -> PILOTO(licencia)}
NOT NULL: {licencia -> PILOTO(licencia)}

```
CREATE TABLE crucero (
matricula CHAR(10),
escudo VARCHAR(2),
carga INTEGER,
licencia CHAR(5) NOT NULL,
CONSTRAINT cru_ma_pk PRIMARY KEY (matricula),
CONSTRAINT cru_li_fk FOREIGN KEY (licencia) REFERENCES piloto(licencia)
CONSTRAINT cru_es_ck CHECK (escudo IN ('si','no'))
);
```

TRANSPORTE (codigo, objetivo)
PK: {codigo}

```
CREATE TABLE transporte (
codigo CHAR(9),
objetivo VARCHAR(30),
CONSTREINT tr_co_pk PRIMARY KEY (codigo),
);
```

REALIZAR (<u>código</u> → TRANSPORTE, <u>matricula</u> → CRUCERO, fecha_salida, fecha_llegada)
PK: {codigo -> TRANSPORTE(codigo), matricula -> CRUCERO(matricula)}
FK: {codigo -> TRANSPORTE(codigo)}
FK: {matricula -> CRUCERO(matricula)}

```
CREATE TABLE realizar (
codigo CHAR(9),
matricula CHAR(10),
fecha_salida DATE,
fecha_llegada DATE,
CONSTRAINT rea_cod_ma_pk PRIMARY KEY (codigo, matricula),
CONSTRAINT rea_cod_fk FOREIGN KEY (codigo) REFERENCES transporte(codigo),
CONSTRAINT rea_mat_fk FOREIGN KEY (matricula) REFERENCES crucero(matricula),
);
```

COSTE (<u>codigo</u> → TRANSPORTE(codigo), <u>linea</u>, concepto, importe)
PK: {linea, codigo → TRANSPORTE(codigo)}
FK: {codigo → TRANSPORTE(codigo)}

```
CREATE TABLE coste (
codigo CHAR(9),
linea INTEGER,
concepto VARCHAR(20),
importe DOUBLE,
CONSTRAINT cos_co_li_pk PRIMARY KEY(codigo, linea),
CONSTRAINT cos_cod_fk FOREIGN KEY(codigo) REFERENCES transporte(codigo)
);
```

Restricciones:
(1) - Habrá que verificar que todo hangar tenga como mínimo una nave a guardar.
(2) - Si la nave es de un tipo de especificación no puede ser de otro tipo.
(3) - Todo transporte debe tener un crucero asociado.
(4) - Revisar que todo transporte tenga un coste.

[1 PUNTO] EJERCICIO 3: MODELADO FÍSICO DDL (MODIFICACIÓN DE METADATOS)

Indica las sentencias necesarias para realizar estas modificaciones en MySQL:

- Modificar la tabla PILOTO para que los campos “nombre” y “apellidos” sean únicos (en conjunto), es decir, que se pueda repetir el nombre y los apellidos por separado pero no puedan insertarse dos filas con los dos datos idénticos.

// Añadir restricción a tabla PILOTO para que los campos nombre y apellidos sean unicos.
SENTENCIA 1; ALTER TABLE piloto ADD CONSTRAINT pi_nom_ape_uk UNIQUE KEY(nombre, apellidos);

- Cambiar la participación de la entidad nave en la relación GUARDAR de (1,1) a (0,1).

// Quitar la restricción de NOT NULL a nombre_hangar de la entidad NAVE
SENTENCIA 1; ALTER TABLE nave MODIFY nombre_hangar CHAR(6);

- Permitir que se pueda modificar el ID de la tabla TRANSPORTE, propagando este cambio a las tablas dependientes de este dato.

// Quitar los antiguos constraint para poner unos nuevos que permitan hacer cambios en el codigo de la entidad TRANSPORTE.
SENTENCIA 1; ALTER TABLE realizar DROP CONSTRAINT rea_cod_fk;

SENTENCIA 2; ALTER TABLE coste DROP CONSTRAINT cos_cod_fk;

SENTENCIA 3; ALTER TABLE realizar ADD CONSTRAINT rea_cod_fk FOREIGN KEY (codigo)
REFERENCES transporte(codigo) ON UPDATE CASCADE;

SENTENCIA 4; ALTER TABLE coste ADD CONSTRAINT cos_cod_fk FOREIGN KEY (codigo)
REFERENCES transporte(codigo) ON UPDATE CASCADE;

- Modificar la tabla BASE para que por defecto la fecha de inicio sea la fecha del sistema.

// Poner que la fecha de inicio sea la del sistema.
SENTENCIA 1; ALTER TABLE base MODIFY fecha_ini TIMESTAMP;

[2 PUNTOS] EJERCICIO 4: MODELADO FÍSICO DML (MANIPULACIÓN DE DATOS)

Indica las sentencias necesarias para realizar estas modificaciones en MySQL:

- Crea una base con código "BALDS1", nombre "Base Alderaan sistema 1".

```
// Insertar codigo y nombre en la tabla base
SENTENCIA 1;    INSERT INTO base (cod_base, nombre)
                VALUES ('BALDS1','Base Alderaan sistema 1');
```

- Crea un hangar con nombre "HALDS1", descripción "Hangar Alderaan sistema 1".

```
// Insertar nombre y descripción en la tabla hangar
SENTENCIA 1;    INSERT INTO hangar (nombre, descripcion)
                VALUES ('HALDS1','Hangar Alderaan sistema 1');
```

- Crea 3 pilotos con los siguientes datos:
  - Licencia "LI001", nombre "Trooper1", apellidos "Comandante".
  - Licencia "LI002", nombre "Trooper2", apellidos "Teniente".
  - Licencia "LI003", nombre "Trooper3", apellidos "Dummie".

```
// Insertar licencia, nombre y apellidos en la tabla piloto
SENTENCIA 1;    INSERT INTO piloto (licencia, nombre, apellidos)
                VALUES ('LI001','Trooper1','Comandante'),
                ('LI002','Trooper2','Teniente'),
                ('LI003','Trooper3','Dummie');
```

- Crea 2 naves operativas con los siguientes datos (nos referiremos como nave 1 y 2):
  - Matrícula "CIALDS1001", nombre "Crucero Imperial carga pesada".
  - Matrícula "CIALDS1002", nombre "Crucero Imperial carga media".
  - La nave 1 es de tipo crucero con carga 10000 toneladas y tiene escudo protector, siendo pilotada por trooper1.
  - La nave 2 es de tipo crucero con carga 5000 toneladas y no tiene escudo protector, siendo pilotada por trooper2

```
// Insertar matricula, nombre y operativa en la tabla nave, ademas insertar matricula, escudo, carga y licencia en tabla crucero
SENTENCIA 1;    INSERT INTO nave (matricula, nombre, operativa)
                VALUES ('CIALDS1001','Crucero Imperial carga pesada','si'),
                ('CIALDS1002','Crucero Imperial carga media','si');

SENTENCIA 2;    INSERT INTO crucero (matricula, escudo, carga, licencia)
                VALUES('CIALDS1001','si',10000,'LI001'),
                ('CIALDS1002','no',5000,'LI002');
```

- El piloto Trooper3 es supervisado por Trooper1.

```
// Actualizar que piloto con licencia LI003 es supervisado por Ll001
SENTENCIA 1;    UPDATE piloto
                SET supervisado_por = 'LI001'
                WHERE licencia = 'LI003';
```

- La base realiza un TRANSPORTE con código "TC_ALDEND" y fecha de salida 1/12/2022 y:
  - Las 2 naves crucero se encargan del transporte de coaxium.
  - El transporte genera 2 costes: "Dietas" con importe 10 créditos imperiales y "Tasas aeropuerto" con importe 0,5 créditos imperiales, respectivamente.

// Insertar codigo y objetivo en transporte, insertar codigo, matricula y fecha de salida a realizar, e insertar codigo, linea, concepto e importe a coste.

```
SENTENCIA 1;    INSERT INTO transporte (codigo, objetivo)
                VALUES('TC_ALDEND','transportar coaxium');

SENTENCIA 2;    INSERT INTO realizar (código, matricula, fecha_salida)
                VALUES('TC_ALDEND','CIALDS1001','2022/12/01'),
                ('TC_ALDEND','CIALDS1002','2022/12/01');

SENTENCIA 3;    INSERT INTO coste (codigo, linea, concepto, importe)
                VALUES('TC_ALDEND',1,'Dietas',10),
                ('TC_ALDEND',2,'Tasas',0.5);
```

- Actualiza la fecha de llegada del transporte anterior a 5/12/2022.

// Actualizar fecha de llegada de transporte TC_ALDEND a 5/12/2022.

```
SENTENCIA 1;    UPDATE realizar
                SET fecha_llegada = '2022/12/05'
                WHERE codigo = 'TC_ALDEND';
```

- Borra el piloto Trooper3.

// Borrar piloto Trooper3

```
SENTENCIA 1;    DELETE FROM piloto
                WHERE nombre = 'Trooper3';
```