Bases de Datos I

Curso 2022-2023

Práctica 1

"Bases de datos de Lego"

Participantes:

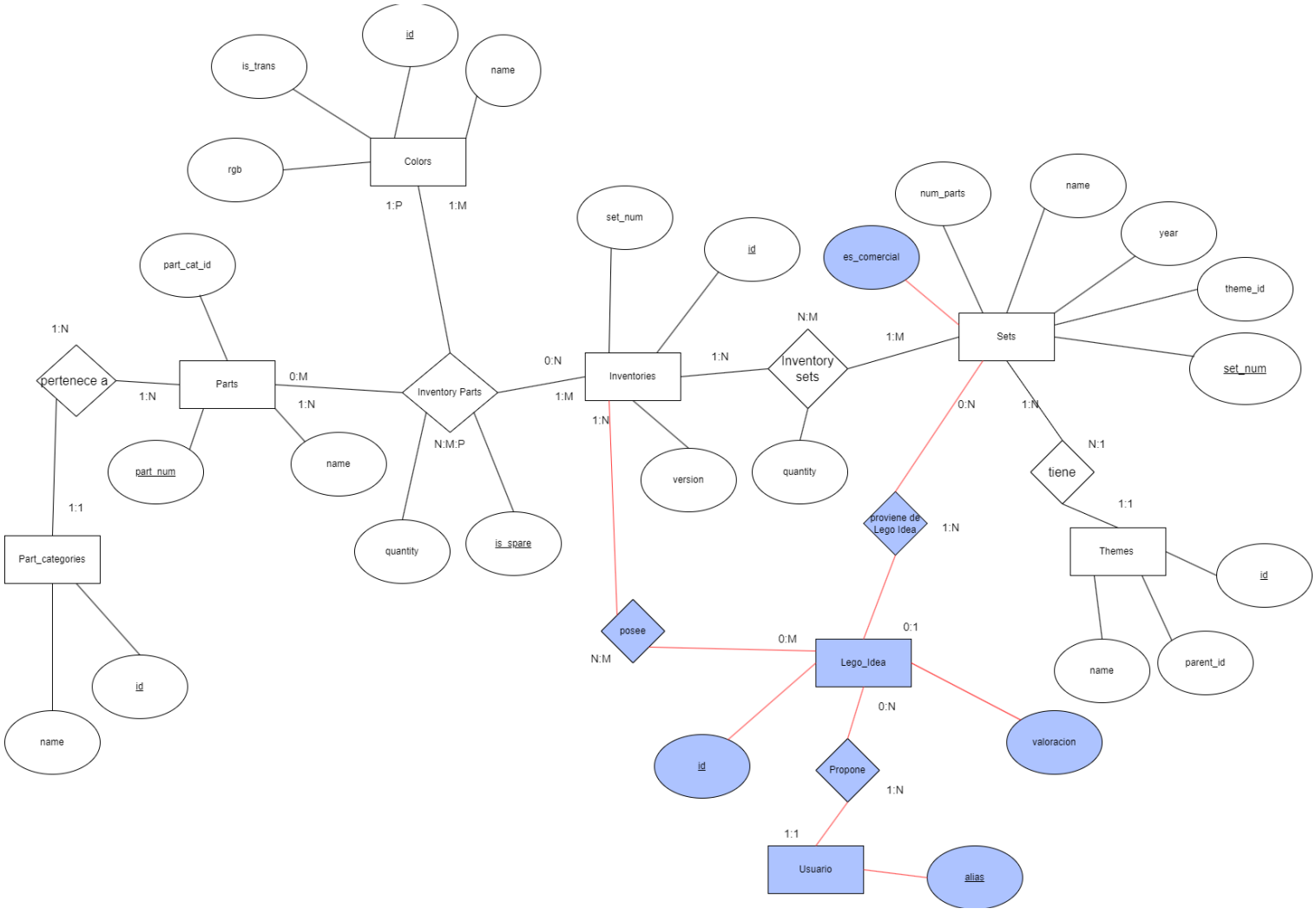
Copado Cantarero, Sergio

Puche García, Alejandro

Ejercicio 1. y 2.

Analiza los ficheros.csv anexos a esta práctica y realiza un diagrama Entidad-Relación empleando notación Chen que represente el modelo de datos contenido en los mismos.

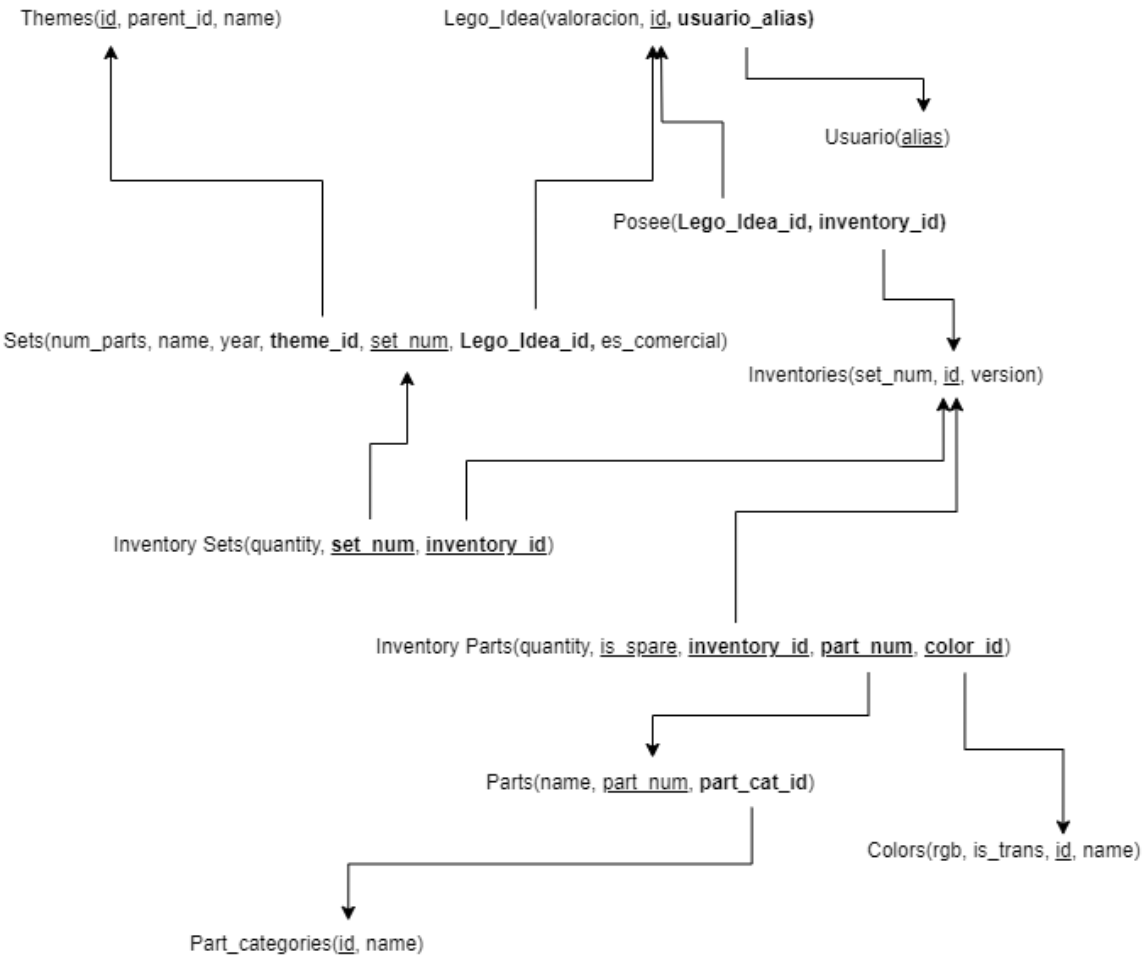
Modifica el diagrama Entidad-Relación para añadir los nuevos requisitos indicados por LEGO en lo referente al proyecto LEGO Ideas. Muestra los cambios en un color diferente en el diagrama para facilitar la corrección.



Para hacer el modelo entidad relación hemos hecho uso tanto de los datos dados en los archivos CSV como de la información dada en la descripción del trabajo.
En el caso del ejercicio 1, hemos realizado el modelo entidad relación para que las tablas que se obtengan posean los atributos de las columnas de los CSV. Mientras que, para realizar el segundo ejercicio, hemos creado los elementos que están en azul en el diagrama. Además, para ver qué atributo era la clave primaria hemos visto si los datos dados en los csv eran únicos y no nulos.

Ejercicio 3.

Realiza el proceso de paso a tablas a partir del modelo Entidad-Relación final.



Analizando las cardinalidades de las distintas relaciones planteadas en el modelo entidad-relación de los ejercicios anteriores, hemos realizado las distintas tablas. En el caso de sets, pasa a tener el atributo Lego_Idea_id el cual será nulo siempre que los sets no provengan de lego ideas, de esta manera se da por hecho si un set propuesto por un usuario de Lego Ideas es comercial, la columna de Lego_idea_id no estará vacía.

Ejercicio 4.

Codifica los scripts de creación de tablas para que la base de datos pueda ser creada en MySQL.

```
CREATE SCHEMA proyecto_lego
DEFAULT CHARACTER SET utf8
COLLATE utf8_spanish2_ci;
USE proyecto_lego;

CREATE TABLE Themes (
        id INT NOT NULL PRIMARY KEY,
        name VARCHAR(250) DEFAULT NULL,
        parent_id INT DEFAULT NULL,
        set_num VARCHAR(250) DEFAULT NULL,
        FOREIGN KEY (parent_id) REFERENCES Themes(id));

CREATE TABLE Usuario (
        alias VARCHAR(250),
        PRIMARY KEY (alias)
);

CREATE TABLE Lego_Idea (
         valoracion INT,
         id INT,
         usuario_alias VARCHAR(250),
         FOREIGN KEY (usuario_alias) REFERENCES Usuario(alias),
         PRIMARY KEY (id)
);

CREATE TABLE Sets (
         set_num VARCHAR(250) NOT NULL PRIMARY KEY,
         name VARCHAR(250),
         year INT,
         theme_id INT,
         num_parts INT,
         Lego_Idea_id INT DEFAULT NULL,
         es_comercial VARCHAR(1) DEFAULT 't',
         FOREIGN KEY (theme_id) REFERENCES Themes(id),
         FOREIGN KEY (Lego_Idea_id) REFERENCES Lego_Idea(id)
);

CREATE TABLE Inventories (
         id INT NOT NULL,
         version INT,
         set_num VARCHAR(250),
         PRIMARY KEY (id)
```

```
);
CREATE TABLE Posee (
        Lego_Idea_id INT,
        inventory_id INT,
        FOREIGN KEY (Lego_Idea_id) REFERENCES Lego_Idea(id),
        FOREIGN KEY (inventory_id) REFERENCES Inventories(id),
        PRIMARY KEY (Lego_Idea_id, inventory_id));

CREATE TABLE Inventory_Sets (
        inventory_id INT NOT NULL,
        set_num VARCHAR(250) NOT NULL,
        quantity INT DEFAULT NULL,
        PRIMARY KEY (inventory_id, set_num),
        FOREIGN KEY (set_num) REFERENCES Sets(set_num),
        FOREIGN KEY (inventory_id) REFERENCES Inventories(id)
);

CREATE TABLE Part_categories (
        id INT NOT NULL,
        name VARCHAR(250),
        PRIMARY KEY (id)
);

CREATE TABLE Parts (
        part_num VARCHAR(250) NOT NULL,
        name VARCHAR(250),
        part_cat_id INT,
        PRIMARY KEY (part_num),
        FOREIGN KEY (part_cat_id) REFERENCES Part_categories(id)
);

CREATE TABLE Colors (
        id INT NOT NULL,
        name VARCHAR(250),
        rgb VARCHAR(250),
        is_trans VARCHAR(1),
        PRIMARY KEY (id)
);

CREATE TABLE Inventory_Parts (
        inventory_id INT NOT NULL,
        part_num VARCHAR(250) NOT NULL,
        color_id INT NOT NULL,
        quantity INT DEFAULT NULL,
        is_spare VARCHAR(1) NOT NULL,
        FOREIGN KEY (inventory_id) REFERENCES Inventories(id),
        FOREIGN KEY (part_num) REFERENCES Parts(part_num),
        FOREIGN KEY (color_id) REFERENCES Colors(id),
```

```
    PRIMARY KEY (inventory_id, part_num, color_id, is_spare));
```

Para definir los tipos de datos de cada atributo, hemos visto los datos dados dentro de los CSVs. En el caso de los atributos is_spare e is_trans los hemos definido como una cadena de caracteres de tamaño 1 puesto que solo toman los valores “f” y “t”. En los otros casos en los que hay una cadena de caracteres, le hemos asignado un tamaño de 250, puesto que, era una tamaño que permitía introducir los datos dados sin dar ningún problema. Además, hemos puesto como condición que las claves primarias sean no nulas.

Ejercicio 5.

Puebla las tablas de la base de datos con la información de los ficheros .csv.

```
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/themes.csv'
    INTO TABLE Themes
    FIELDS TERMINATED BY ','
    ENCLOSED BY  “ ”
    LINES TERMINATED BY  '\n'
    IGNORE 1 ROWS
    (id, name, @parent_id)
    SET parent_id = NULLIF(@parent_id,' ');

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/sets.csv'
    INTO TABLE Sets
    FIELDS TERMINATED BY  ','
    ENCLOSED BY  “ ”
    LINES TERMINATED BY  '\n'
    IGNORE 1 ROWS
    (@num, @name, @year, @theme, @parts)
    SET set_num=@num, name=@name, year=@year, theme_id = @theme, num_parts
    = @parts;

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/inventories.csv'
    INTO TABLE Inventories
    FIELDS TERMINATED BY  ','
    ENCLOSED BY  “”
    LINES TERMINATED BY  '\n'
    IGNORE 1 ROWS;

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/inventory_sets.csv'
    INTO TABLE Inventory_Sets
    FIELDS TERMINATED BY  ','
    ENCLOSED BY  “”
```

```
        LINES TERMINATED BY  '\n'
        IGNORE 1 ROWS;

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/part_categories.csv'
        INTO TABLE Part_categories
        FIELDS TERMINATED BY  ','
        ENCLOSED BY  “”
        LINES TERMINATED BY  '\n'
        IGNORE 1 ROWS;

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/parts.csv'
        INTO TABLE Parts
        FIELDS TERMINATED BY  ','
        ENCLOSED BY  “”
        LINES TERMINATED BY  '\n'
        IGNORE 1 ROWS
        (@part_num, @name, @categories_id)
        SET part_num = @part_num, name = @name, part_cat_id = @categories_id;

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/colors.csv'
        INTO TABLE Colors
        FIELDS TERMINATED BY  ','
        ENCLOSED BY  “”
        LINES TERMINATED BY  '\n'
        IGNORE 1 ROWS;

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/inventory_parts.csv'
        IGNORE INTO TABLE Inventory_Parts
        FIELDS TERMINATED BY  ','
        ENCLOSED BY  “”
        LINES TERMINATED BY  '\n'
        IGNORE 1 ROWS;
```

Para introducir los datos, hemos usado la versión de MySQL 8.0 presentada en el Moodle de la asignatura como una alternativa a Docker, puesto que, al crear la base de datos en local permitía subir los archivos CSV permitiéndonos introducir los datos desde esos archivos.

Ejercicio 6.

Resuelve las siguientes consultas empleando el lenguaje SQL.

A. Nombre y número de piezas de los sets del año 2012 del tema ‘Ferrari’.

```
SELECT sets.name, sets.num_parts
FROM sets
	INNER JOIN themes ON sets.theme_id=themes.id
WHERE sets.year=2012 AND themes.name="Ferrari";
```

B. Número de sets de cada tema ordenados de más sets a menos sets.

```
SELECT themes.name, COUNT(DISTINCT sets.name) AS number_of_sets
FROM sets
	INNER JOIN themes ON sets.theme_id = themes.id
GROUP BY themes.name
ORDER BY number_of_sets DESC;
```

C. Sets que contienen piezas transparentes y no transparentes.

```
SELECT DISTINCT sets.name
FROM sets
	INNER JOIN inventory_sets ON inventory_sets.set_num = sets.set_num
	INNER JOIN inventories ON inventories.id = inventory_sets.inventory_id
	INNER JOIN inventory_parts ON inventories.id = inventory_parts.inventory_id
	INNER JOIN colors ON inventory_parts.color_id = colors.id
WHERE colors.is_trans IS NOT NULL
GROUP BY sets.set_num
HAVING COUNT(DISTINCT colors.is_trans) = 2;
```

D. Temas con sets publicados en los años 2013, 2014 y 2015.

```
SELECT DISTINCT themes.id, themes.name
FROM themes
	INNER JOIN sets ON themes.id = sets.theme_id
WHERE sets.year BETWEEN 2013 AND 2015
GROUP BY themes.id
HAVING COUNT(DISTINCT sets.year) = 3;
```

Para hacer este apartado, hemos entendido que los temas tenían que tener sets publicados en los 3 años pedidos.

E. Usuarios de LEGO Ideas a los que les han comercializado todos sus sets.

```
SELECT DISTINCT lego_idea.usuario_alias
FROM lego_idea
WHERE lego_idea.usuario_alias NOT IN (SELECT lego_idea.usuario_alias
                                      FROM lego_idea
                                        LEFT JOIN sets ON sets.Lego_Idea_id=lego_idea.id
                                     WHERE sets.es_comercial='f' OR sets.set_num IS NULL);
```

-Explicación: al hacer el LEFT JOIN en la subconsulta nos quedamos únicamente con los sets que provienen de lego idea. Además, pedimos que se hayan comercializado todos sus sets así que a los usuarios de lego ideas les quitamos los que tienen un set no comercializado. También restamos los que tienen set_num nulo tras hacer el left join, por si hubiese una lego idea sin un set asociado.

F. Sets que no contienen ninguna pieza de color rgb 'FFA70B' (no transparente).

```
SELECT DISTINCT sets.name, colors.rgb, colors.is_trans
FROM sets
       INNER JOIN inventory_sets on sets.set_num = inventory_sets.set_num
       INNER JOIN inventories on inventory_sets.inventory_id = inventories.id
       INNER JOIN inventory_parts on inventories.id = inventory_parts.inventory_id
       INNER JOIN colors on inventory_parts.color_id = colors.id
WHERE sets.set_num NOT IN(SELECT sets.set_num
              FROM sets
              INNER JOIN inventory_sets ON sets.set_num = inventory_sets.set_num
              INNER JOIN inventories ON inventory_sets.inventory_id= inventories.id
              INNER JOIN inventory_parts ON inventories.id = inventory_parts.inventory_id
              INNER JOIN colors on inventory_parts.color_id = colors.id
              WHERE colors.rgb = 'FFA70B' AND colors.is_trans = 'f')

GROUP BY sets.name, colors.rgb, colors.is_trans
ORDER BY sets.name;
```

G. Piezas contenidas en todos los sets independientemente de su color.

```
SELECT parts.part_num
FROM sets
       INNER JOIN inventory_sets ON inventory_sets.set_num=sets.set_num
       INNER JOIN inventories ON inventories.id=inventory_sets.inventory_id
       INNER JOIN inventory_parts ON inventories.id=inventory_parts.inventory_id
       INNER JOIN parts ON inventory_parts.part_num=parts.part_num
GROUP BY parts.part_num
HAVING COUNT(DISTINCT sets.set_num) = (SELECT COUNT(DISTINCT sets.set_num)
                                       FROM sets);
```

H. Sets con todas las piezas que contengan ‘Green’ en su nombre.

```
SELECT DISTINCT sets.set_num
FROM sets
       INNER JOIN inventory_sets ON sets.set_num=inventory_sets.set_num
       INNER JOIN inventories ON inventory_sets.inventory_id=inventories.id
       INNER JOIN inventory_parts ON inventories.id=inventory_parts.inventory_id
       INNER JOIN parts ON parts.part_num=inventory_parts.part_num
WHERE parts.name LIKE '%Green%'
GROUP BY sets.set_num
HAVING COUNT(DISTINCT parts.name) = (SELECT COUNT(DISTINCT parts.name)
                                     FROM parts
                                     WHERE parts.name LIKE '%Green%');
```

I. Nombre del set con mayor cantidad de piezas.

```
SELECT sets.name, num_parts
FROM sets
WHERE num_parts = (SELECT MAX(num_parts)
                   FROM sets);
```

J. Usuario de LEGO Ideas con más sets comercializados.

```
SELECT lego_idea.usuario_alias, COUNT(DISTINCT sets.Lego_Idea_id) AS count_set
FROM lego_idea
       INNER JOIN sets ON sets.Lego_Idea_id = lego_idea.id
WHERE sets.es_comercial=’t’
GROUP BY lego_idea.usuario_alias
ORDER BY count_set DESC
LIMIT 1;
```

Ejercicio 7.

Codifica un procedimiento que reciba como parámetro de entrada el alias de un usuario del proyecto LEGO Ideas y devuelva todos los sets propuestos ordenados de forma descendente por el número de piezas que contienen.

```
DELIMITER $$
CREATE PROCEDURE sets_propuestos_por(IN user_alias VARCHAR(250))
BEGIN
SELECT sets.set_num, sets.num_parts
FROM lego_idea
       INNER JOIN sets ON lego_idea.id = sets.Lego_Idea_id
WHERE lego_idea.usuario_alias = user_alias
ORDER BY sets.num_parts DESC;
END$$
DELIMITER ;
```

Ejercicio 8.

Codifica un trigger en el que, cada vez que se inserte un nuevo set proveniente del proyecto LEGO Ideas, proporcione a su creador un crédito de 1000€ que quedará registrado en la base de datos. Modifica las tablas de la base de datos si es necesario.

```
ALTER TABLE proyecto_lego.usuario
ADD credito FLOAT DEFAULT 0.0;

DELIMITER $$
CREATE FUNCTION ret_alias(ID INTEGER)
RETURNS VARCHAR(250)
DETERMINISTIC
BEGIN
DECLARE user_alias VARCHAR(250);
SELECT usuario.alias INTO user_alias
FROM sets
	INNER JOIN lego_idea ON lego_idea.id = sets.Lego_Idea_id
	INNER JOIN usuario ON usuario.alias = lego_idea.usuario_alias
WHERE sets.Lego_Idea_id = ID;
RETURN (user_alias);
END$$
DELIMITER ;


DELIMITER $$
CREATE TRIGGER dar_credito
AFTER INSERT ON sets
FOR EACH ROW
BEGIN
	IF NEW.Lego_Idea_id IS NOT NULL AND NEW.es_comercial = “t”
	THEN
	UPDATE usuario
	SET usuario.credito = usuario.credito + 1000.0
	WHERE usuario.alias = ret_alias(NEW.Lego_Idea_id);
	END IF;
END$$
DELIMITER ;
```

Para hacer este ejercicio hemos tenido que modificar la tabla usuario añadiendo un nuevo atributo al que hemos llamado “crédito” y lo hemos definido como un float. Además, hemos puesto que por defecto se asigne un crédito de 0 al usuario si no se especifica otro al introducir el usuario.
A continuación, hemos definido la función ret_alias que toma como parámetro el id de una Lego idea y devuelve el alias del usuario que lo ha creado. Por último, hemos definido el trigger llamado dar_credito que, en caso de que se inserte un set con el atributo Lego_Idea_id no nulo y que tenga es_cormercial=’t’, aumenta en 1000 el crédito del usuario que lo ha creado. Para llegar hasta el usuario que ha creado la lego idea el trigger hace uso de la función ret_alias definida antes.