


	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	BASES DE DATOS
 Escuela Superior de Ingeniería y Tecnología Universidad de La Laguna	PROYECTO: Librería Crossing	Generación de código
	<i>Autores:</i> Anabel Díaz Labrador (alu0101206011) Alejandro Martín de León (alu0101015941) Jaime Pablo Pérez Moro (alu0101278919) Andrea Calero Caro (alu0101202952) Saúl Pérez García (alu0101129785) Sheyla Ruiz-Gómez Ferreira (alu0101124445)	
Versión: 1.0		Fecha : 04/02/22

Script inicial generado del modelo relacional

Tras realizar el modelo relacional, lo primero que se ha hecho es generar el script en formato **‘.sql’**, de tal forma que a posteriori, podamos implementar las distintas comprobaciones, inserciones, disparadores...

Se puede encontrar el script original en el siguiente [link](#):

CÓDIGO DE COMPONENTES


Posteriormente, se procede a transformar el script que se ha generado previamente, con el objetivo de hacer uso del mismo en el entorno de PostgreSQL. A continuación, se pretende detallar los diferentes bloques por los que se compone el [script](#):

En primer lugar, las tablas se generan a través de las siguientes sentencias:

```
DROP TABLE IF EXISTS "mydb"."LIBROS" CASCADE;

CREATE TABLE "mydb"."LIBROS" (
  "ISBN" INT NOT NULL,
  "Título" VARCHAR(45) NOT NULL,
  "Autor" VARCHAR(45) NULL,
  "Género" VARCHAR(45) NULL,
  "Editorial" VARCHAR(45) NULL,
  "Precio" DOUBLE PRECISION NULL,
  "FechaPublicacion" DATE NULL,
  PRIMARY KEY ("ISBN"));
```

Donde inicialmente, se elimina la tabla si existía previamente, y luego, se crea la tabla con todos sus atributos. En el ejemplo anterior se puede ver como ha de crearse la tabla LIBROS.

	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	BASES DE DATOS
 Escuela Superior de Ingeniería y Tecnología Universidad de La Laguna	PROYECTO: Librería Crossing	Generación de código
	<i>Autores:</i> Anabel Díaz Labrador (alu0101206011) Alejandro Martín de León (alu0101015941) Jaime Pablo Pérez Moro (alu0101278919) Andrea Calero Caro (alu0101202952) Saúl Pérez García (alu0101129785) Sheyla Ruiz-Gómez Ferreira (alu0101124445)	
Versión: 1.0	Tiempo invertido:	Fecha : 04/02/22

Posteriormente, una vez han sido creadas todas las tablas, se procede a determinar los diferentes triggers. Para ello, en primer lugar se establece la función que va a ejecutar el trigger, tal y como se muestra a continuación:

```
CREATE OR REPLACE FUNCTION actualizar_stock_edicion_entrega() RETURNS trigger AS $trigger_actualizar_stock_edicion_entrega$
BEGIN
    UPDATE "mydb"."EDICION" SET
        "Stock" = "Stock" + 1
    WHERE NEW."EDICION_NumEdicion" = "mydb"."EDICION"."NumEdicion";
    UPDATE "mydb"."CLIENTE" SET
        "LibrosExtraibles" = "LibrosExtraibles" + 1
    WHERE NEW."CLIENTE_DNI" = "mydb"."CLIENTE"."DNI";
    RETURN NEW;
END;
$trigger_actualizar_stock_edicion_entrega$ LANGUAGE plpgsql;
```


En este caso, se trata de la función “actualizar_stock_edicion_entrega()”, la cuál permitirá realizar una actualización en la base de datos del stock de las diferentes ediciones. Esta función no será ejecutada hasta que se active el trigger que se encuentra relacionado con la misma. En este caso, se trata del siguiente:

```
CREATE TRIGGER trigger_actualizar_stock_edicion_entrega AFTER INSERT ON "mydb"."ENTREGA"
FOR EACH ROW EXECUTE PROCEDURE actualizar_stock_edicion_entrega();
```

Después de cada inserción en la tabla entrega, se procede a comprobar y ejecutar la función a la que se relaciona, actualizando así el stock en las tablas edición y clientes.

Por otro lado, se encuentra otro bloque donde se muestran varios ejemplos de inserciones que se han realizado sobre la base de datos de forma que se muestre el funcionamiento del mismo. Un ejemplo de este tipo de sentencia es la siguiente:

```
START TRANSACTION;
INSERT INTO "mydb"."EDICION" ("LIBROS_ISBN", "NumEdicion", "Stock", "CLIENTE_DNI_recibe") VALUES (1234, 1, 10, '79063836');
INSERT INTO "mydb"."EDICION" ("LIBROS_ISBN", "NumEdicion", "Stock", "CLIENTE_DNI_recibe") VALUES (1234, 5, 30, '79063836');
INSERT INTO "mydb"."EDICION" ("LIBROS_ISBN", "NumEdicion", "Stock", "CLIENTE_DNI_recibe") VALUES (2468, 4, 15, '04576453');
COMMIT;
```

	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	BASES DE DATOS
 Escuela Superior de Ingeniería y Tecnología Universidad de La Laguna	PROYECTO: Librería Crossing	Generación de código
	<i>Autores:</i> Anabel Díaz Labrador (alu0101206011) Alejandro Martín de León (alu0101015941) Jaime Pablo Pérez Moro (alu0101278919) Andrea Calero Caro (alu0101202952) Saúl Pérez García (alu0101129785) Sheyla Ruiz-Gómez Ferreira (alu0101124445)	
Versión: 1.0	Tiempo invertido:	Fecha : 04/02/22

Por último, al final del script se han adjuntado una serie de pruebas sobre la base de datos, donde se muestran el contenido de las diferentes tablas, así como de algunas inserciones concretas para comprobar el buen funcionamiento de los diferentes disparadores. Un ejemplo de esto es el siguiente:

```
SELECT * FROM "mydb"."EDICION";
START TRANSACTION;
INSERT INTO "mydb"."RECIBE" ("CLIENTE_DNI", "EDICION_LIBROS_ISBN", "EDICION_NumEdicion") VALUES ('04576453', 1234, 1);
COMMIT;
SELECT * FROM "mydb"."EDICION";
```


CÓDIGO DE PROCEDIMIENTOS DE OPERACIÓN Y SEGURIDAD

A continuación, se pretende exponer las diferentes comprobaciones y disparadores que se encuentran presentes en la base de datos.

En primer lugar, se muestra en la siguiente imagen el código generado para el primer trigger:

```
CREATE TRIGGER trigger_actualizar_stock_edicion_entrega AFTER INSERT ON "mydb"."ENTREGA"
FOR EACH ROW EXECUTE PROCEDURE actualizar_stock_edicion_entrega();
```

En este caso, el trigger se activa tras cada inserción en la tabla **ENTREGA**, donde se ha de realizar una actualización en el stock (tanto en la tabla **EDICIÓN** como en el número de libros extraíbles para la tabla **CLIENTE**). Con el objetivo de llevar esto a cabo, se ha enlazado a este trigger la siguiente función:

	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	BASES DE DATOS
 Escuela Superior de Ingeniería y Tecnología Universidad de La Laguna	PROYECTO: Librería Crossing	Generación de código
	<i>Autores:</i> Anabel Díaz Labrador (alu0101206011) Alejandro Martín de León (alu0101015941) Jaime Pablo Pérez Moro (alu0101278919) Andrea Calero Caro (alu0101202952) Saúl Pérez García (alu0101129785) Sheyla Ruiz-Gómez Ferreira (alu0101124445)	
Versión: 1.0	Tiempo invertido:	Fecha : 04/02/22

```
CREATE OR REPLACE FUNCTION actualizar_stock_edicion_entrega() RETURNS trigger AS $trigger_actualizar_stock_edicion_entrega$
BEGIN
    UPDATE "mydb"."EDICION" SET
        "Stock" = "Stock" + 1
    WHERE NEW."EDICION_NumEdicion" = "mydb"."EDICION"."NumEdicion";
    UPDATE "mydb"."CLIENTE" SET
        "LibrosExtraibles" = "LibrosExtraibles" + 1
    WHERE NEW."CLIENTE_DNI" = "mydb"."CLIENTE"."DNI";
    RETURN NEW;
END;
$trigger_actualizar_stock_edicion_entrega$ LANGUAGE plpgsql;
```

Por otro lado, se pretende actualizar el stock en el momento en el que los **CLIENTES** realicen una RESERVA de alguna edición de un libro. Para ello, y de forma similar al caso anterior, se crea el siguiente trigger:


```
CREATE TRIGGER trigger_actualizar_stock_edicion BEFORE INSERT ON "mydb"."RESERVA"
FOR EACH ROW EXECUTE PROCEDURE actualizar_stock_edicion();
```

Al mismo tiempo, se le ha sido asignada la siguiente función, encargada de comprobar el stock previo de esa edición, donde ha de ser mayor que cero. En caso afirmativo, se procede a realizar la actualización sobre el stock en la tabla **EDICIÓN**.

```
CREATE OR REPLACE FUNCTION actualizar_stock_edicion() RETURNS trigger AS $trigger_actualizar_stock_edicion$
BEGIN
    IF (SELECT "mydb"."EDICION"."Stock" FROM "mydb"."EDICION" WHERE "mydb"."EDICION"."NumEdicion" = NEW."EDICION_NumEdicion") > 0 THEN
        UPDATE "mydb"."EDICION" SET
            "Stock" = "Stock" - 1
        WHERE NEW."EDICION_NumEdicion" = "mydb"."EDICION"."NumEdicion";
        RETURN NEW;
    ELSE
        RETURN 'ERR_NO_STOCK';
    END IF;
END;
$trigger_actualizar_stock_edicion$ LANGUAGE plpgsql;
```

Reutilizando la función anterior, se crea un nuevo trigger. El cometido de este, es el de actualizar el stock en **EDICIÓN** antes de una nueva entrada de datos en la tabla **PERTENECE**.

Por último, tras cada inserción en la tabla **RECIBE**, se pretende también tener un control en el stock de **EDICIÓN** y **CLIENTE**.

	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	BASES DE DATOS
 Escuela Superior de Ingeniería y Tecnología Universidad de La Laguna	PROYECTO: Librería Crossing	Generación de código
	<i>Autores:</i> Anabel Díaz Labrador (alu0101206011) Alejandro Martín de León (alu0101015941) Jaime Pablo Pérez Moro (alu0101278919) Andrea Calero Caro (alu0101202952) Saúl Pérez García (alu0101129785) Sheyla Ruiz-Gómez Ferreira (alu0101124445)	
Versión: 1.0	Tiempo invertido:	Fecha : 04/02/22

```
CREATE TRIGGER trigger_actualizar_stock_edicion_recibe BEFORE INSERT ON "mydb"."RECIBE"
FOR EACH ROW EXECUTE PROCEDURE actualizar_stock_edicion_recibe();
```

De esta manera, en caso de que se cumplan las siguientes condiciones:

- Hay al menos un libro en stock de la edición que desea el cliente (Stock).
- El cliente tiene como mínimo, la posibilidad de retirar un libro (LibrosExtraibles).

Una vez verificado, se actualizan ambos atributos.


```
CREATE OR REPLACE FUNCTION actualizar_stock_edicion_recibe() RETURNS trigger AS $trigger_actualizar_stock_edicion_recibe$
BEGIN
  IF (SELECT "mydb"."EDICION"."Stock" FROM "mydb"."EDICION" WHERE "mydb"."EDICION"."NumEdicion" = NEW."EDICION_NumEdicion") > 0
  AND (SELECT "mydb"."CLIENTE"."LibrosExtraibles" FROM "mydb"."CLIENTE" WHERE "mydb"."CLIENTE"."DNI" = NEW."CLIENTE_DNI") > 0 THEN
    UPDATE "mydb"."EDICION" SET
      "Stock" = "Stock" - 1
    WHERE NEW."EDICION_NumEdicion" = "mydb"."EDICION"."NumEdicion";
    UPDATE "mydb"."CLIENTE" SET
      "LibrosExtraibles" = "LibrosExtraibles" - 1
    WHERE NEW."CLIENTE_DNI" = "mydb"."CLIENTE"."DNI";
    RETURN NEW;
  ELSE
    RETURN 'ERR_NO_STOCK_OR_NO_NUM_RECEIVE';
  END IF;
END;
$trigger_actualizar_stock_edicion_recibe$ LANGUAGE plpgsql;
```

Por otro lado, respecto a la propia seguridad de la base de datos se han establecido controles a través de **CHECKS**. Se pretende por tanto, mostrar algunos de los ejemplos de control:

- En el caso de la tabla **EDICION**, la intención es que nunca se pueda introducir un stock negativo en la tabla.

```
CREATE TABLE "mydb"."EDICION" (
  "LIBROS_ISBN" INT NOT NULL,
  "NumEdicion" INT NOT NULL,
  "Stock" INT NOT NULL CHECK("Stock" >= 0),
```

- En cuanto a la fecha límite de la tabla **RESERVA** nunca podrá ser anterior a la fecha en la que se creó dicha reserva.

	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	BASES DE DATOS
 Escuela Superior de Ingeniería y Tecnología Universidad de La Laguna	PROYECTO: Librería Crossing	Generación de código
	<i>Autores:</i> Anabel Díaz Labrador (alu0101206011) Alejandro Martín de León (alu0101015941) Jaime Pablo Pérez Moro (alu0101278919) Andrea Calero Caro (alu0101202952) Saúl Pérez García (alu0101129785) Sheyla Ruiz-Gómez Ferreira (alu0101124445)	
Versión: 1.0		Fecha : 04/02/22

```
CREATE TABLE "mydb"."RESERVA" (
  "CLIENTE_DNI" VARCHAR(45) NOT NULL,
  "FISICO_EDICION_LIBROS_ISBN" INT NOT NULL,
  "EDICION_NumEdicion" INT NOT NULL,
  "ID" VARCHAR(45) NOT NULL,
  "FECHA_LIMITE" DATE NOT NULL CHECK("FECHA_LIMITE" >= current_date),
```