BD2

Trabajo práctico

Fabiola Ramos

Facundo Méndez

Gino Tubaro

8 Mayo 2025

Trabajo Práctico: Lenguaje de Consulta SQL

OBJETIVOS: Realizar un Trabajo Practico Integrador, utilizando todos los conocimientos brindados tanto en BD1 como en BD2

Basándose en la Base de Datos de Prestamos de Libros vista en BD1

- LIBRO (NRO_LIBRO, TITULO, AUTOR, TIPO, PRECIO_ORI, PRECIO_ACT, EDICION, ESTADO)
- TIPOLIBRO (TIPO, DESCTIPO)
- COPIAS (NRO_LIBRO, NRO_COPIA, ESTADO)
- LECTOR (NRO_LECTOR, NOMBRE, DIRECCION, TRABAJO, SALARIO, ESTADO)
- PRESTAMO (NRO-LECTOR, NRO-LIBRO, NRO-COPIA, F_PREST, F_DEVOL)

Y considerando las siguientes restricciones:

El campo ESTADO de la tabla Copias puede ser:

- P- Prestado
- D- Devuelto
- N- No Disponible

El campo ESTADO de la tabla Libros puede ser:

- D- Disponible
- N- No Disponible

El campo ESTADO de la tabla Lector

- H- Habilitado
- I- InHabilitado

Lo que se presta no es el Libro, sino la Copia del Libro, y un Lector puede sacar prestada la misma copia, pero no el mismo día.

Ni la copia ni el Libro se puede prestar si esta No Disponible.

Un Lector Inhabilitado no puede realizar Prestamos

Parte 1A : Creación del esquema y carga de Datos

Asignarle las Restricciones de Integridad basándose en la definición previa

- PK
- FK
- Check
- Default
- En todos los casos el campo Estado no puede ser Nulo
- Mejorar la Tabla LECTOR
    - Poner Campos Compuestos donde considere
    - Validar que el Campo Trabajo cumpla sólo con Valores Válidos
    - Validar que el Salario sea mayor que 0(Cero)

Formato de Entrega:

Archivo 1

- DDL
- Poner Encabezado con Integrantes

Archivo 2

- DML con los Inserts

Respuesta:

Archivo 1 se entrega como DDL_Libros.sql

Archivo 2 se entrega como DML_Libros.sql

Parte 1B: Normalización

Basándose en el modelo origen del TP, generar un modelo alternativo, no son necesarias las restricciones de integridad, solo las PK, en el que:

- Alguna tabla No este en ninguna FN
- Alguna tabla que este en 1FN pero no en 2FN
- Alguna tabla que este en 2FN pero no en 3FN
- Alguna tabla que este en 3FN pero no en 2FN

Justificar cada Caso, o sea porque esta en nFN pero no en n+1FN

De considerarlo necesario, se pueden crear campos que tengan relación con el modelo inicial.

Se pueden utilizar variantes de la misma Tabla para dar los ejemplos

Formato de Entrega:

- Word
    - Tabla, en formato DLR (pegar imagen) o DDL
    - Justificación de cada caso debajo de la tabla

RESPUESTA: Parte 1B – Normalización (Formato DDL)

Las siguientes tres tablas en formato DDL buscarán dar respuesta a las consignas, cada una correspondiente a un nivel específico de normalización y su justificación. La cuarta tabla, como se explica más abajo, no puede existir según la teoría de normalización.

Tabla 1 – No está en ninguna FN

DDL:

```
CREATE TABLE LibroConAutorCompuesto (
    NRO_LIBRO INT PRIMARY KEY,
    TITULO VARCHAR(100),
    AUTOR VARCHAR(255)
);
```

Justificación:

No cumple 1FN porque el campo AUTOR puede contener múltiples valores (multivaluados), lo cual viola la atomicidad requerida por la primera forma normal.

Por ejemplo, si el campo AUTOR almacenara una cadena con varios nombres de autores separados por comas o punto y coma (ej: 'Borges, J.L.; Bioy Casares, A.'), se consideraría multivaluado.

Tabla 2 – En 1FN pero no en 2FN

DDL:

```
CREATE TABLE Prestamo1FN (
    NRO_LECTOR INT,
    NRO_LIBRO INT,
    NOMBRE_LECTOR VARCHAR(100),
    PRIMARY KEY (NRO_LECTOR, NRO_LIBRO)
);
```

Justificación:

Cumple 1FN porque todos los campos contienen valores atómicos. Sin embargo, no cumple 2FN porque el atributo NOMBRE_LECTOR depende únicamente de NRO_LECTOR, que es solo parte

de la clave primaria compuesta (NRO_LECTOR, NRO_LIBRO). Esto representa una dependencia parcial, lo que viola la segunda forma normal.

Tabla 3 – En 2FN pero no en 3FN

DDL:

```
CREATE TABLE Lector2FN (
    NRO_LECTOR INT PRIMARY KEY,
    NOMBRE VARCHAR(100),
    TRABAJO VARCHAR(50),
    SALARIO_ESTIMADO MONEY
);
```

Justificación:

Cumple 2FN porque la clave primaria NRO_LECTOR es simple, lo que significa que no pueden existir dependencias parciales de atributos no clave sobre una parte de la clave primaria. Todos los atributos no clave (NOMBRE, TRABAJO, SALARIO_ESTIMADO) dependen funcionalmente de la totalidad de la clave primaria NRO_LECTOR.

Cumple 2FN ya que todos los atributos no clave dependen completamente de la clave primaria. Sin embargo, no cumple 3FN debido a la dependencia transitiva: TRABAJO -> SALARIO_ESTIMADO.

Tabla 4 – En 3FN pero no en 2FN

Justificación:

Una tabla que esté en 3FN pero no en 2FN es teóricamente imposible. La normalización es un proceso jerárquico, por lo que para que una tabla esté en 3FN debe necesariamente haber pasado por 1FN y 2FN. Por lo tanto, no se puede construir una tabla válida que cumpla esta condición sin violar la lógica fundamental de la teoría de normalización.

Parte 2: Desarrollo

Crear por lo menos:

- Un Trigger
- Un Stored Procedure
- Una Función
- Una Vista con las Inconsistencias
- Una Transacción
    - Hacer un SQL Script/SP que tenga un Begin Tran y Commit/Rollback
    - Se recomienda usar el concepto de Try y Catch

Ejemplos:

Como ejemplo de Trigger SP podría ser que se cambie el campo Estado de la tabla Copias cuando el Libro se preste, y/o se devuelva.

Como ejemplo de Transacción se podría grabar un préstamo y en caso de que el libro tenga mal el Estado, no lo deje Prestar/Devolver.

Para la generación de la Vista con las Inconsistencias, se puede tener en cuenta la relación entre las tablas Copias y los Préstamos. Por ejemplo, si una copia está como Disponible y hay un Préstamo de esa copia que no tiene Fecha de Devolución, sería una inconsistencia, lo mismo si ocurre lo contrario, una copia que está como Prestada, pero no haya ningún registro de Préstamo con Fecha de devolución Nula

NOTA

Se puede utilizar otra base alternativa, en vez de la de Libros, por ejemplo si están usando alguna BD en otra materia.

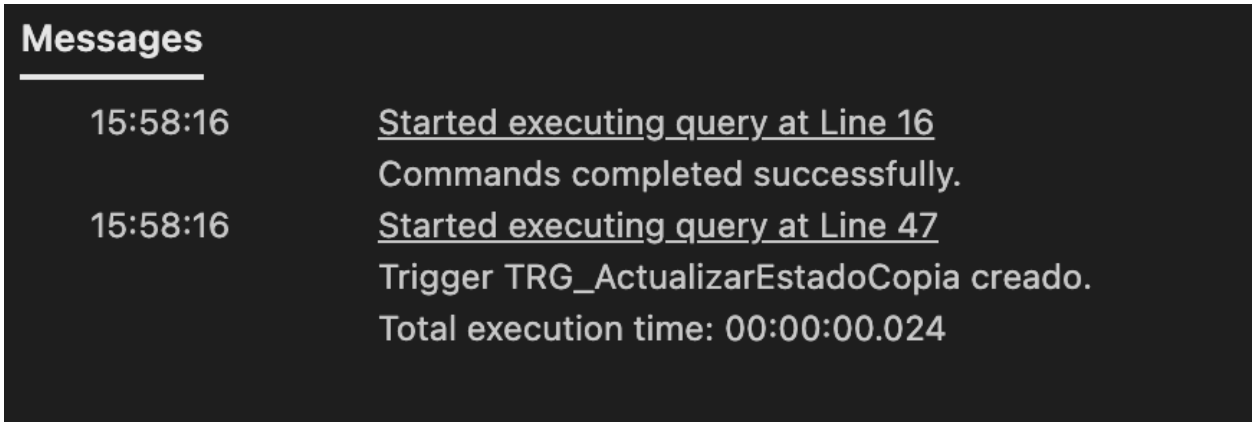
Las tareas a realizar son similares.

Respuesta:

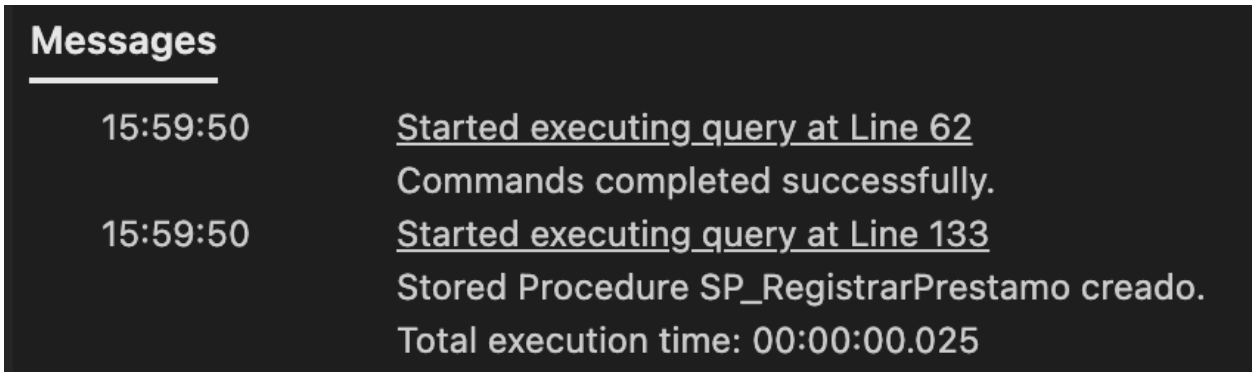
Se entrega un archivo llamado Parte2_Desarrollo.sql con los scripts solicitados.

Capturas de ejecución en Azure Data Studio:

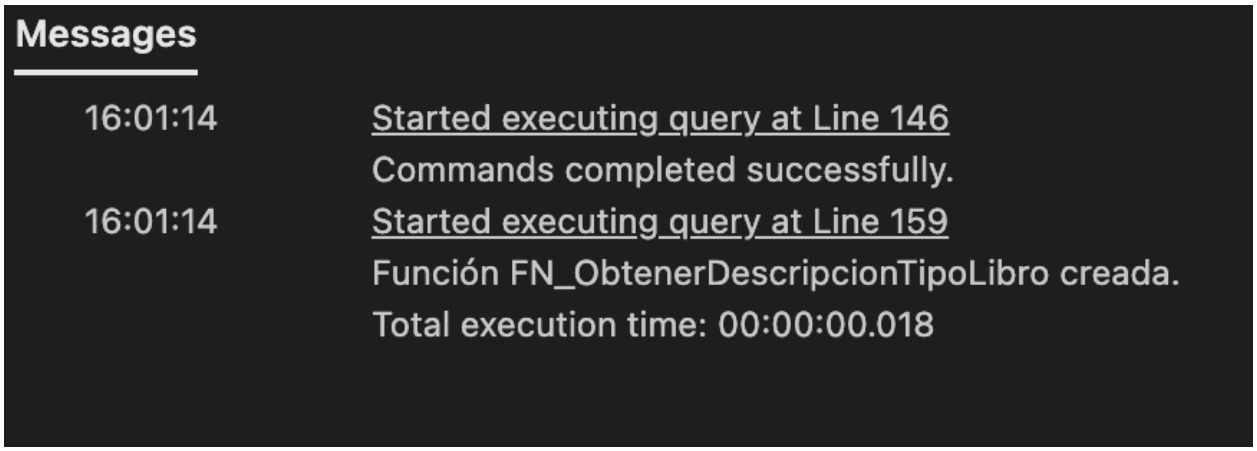
1. Trigger



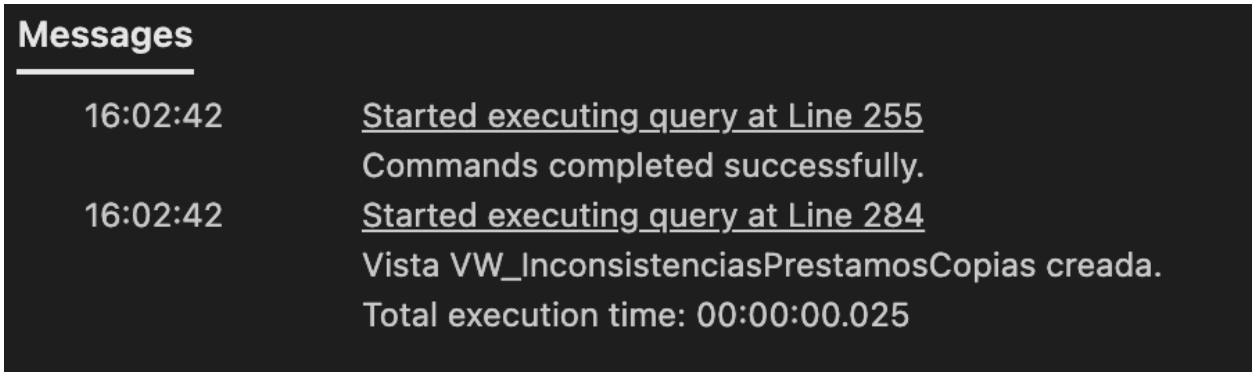
2. Stored Procedure:



3. Función



4. Vista con inconsistencias:



5. Transacción