ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

CARRERA DE INGENIERÍA DE SOFTWARE

AUDITORÍA INFORMÁTICA GR1SW

Manual Técnico y de Usuario de la Tarea 3

INTEGRANTES GRUPO 3:

Anderson Cárdenas
Diana López
Mahatma Quijano

FECHA DE ENTREGA

21 de febrero del 2022

Contenido

1. Objetivos del programa

El principal objetivo de implementar el siguiente programa es la ejecución de varias actividades de Auditoria de Bases de Datos para SQL Server, las cuales se especifican a continuación:

1) Identificación automática de las relaciones que requieren integridad referencial, que existen y que deberían existir. Esto se realizará mediante procedimientos almacenados que verifiquen las relaciones existentes a partir de las restricciones de clave foráneas, y la inspección de aquellas tablas con columnas repetidas en otras, indicando la posibilidad de una relación que debería existir.
2) Chequeo automático de anomalías en la definición de la integridad referencial para inserción, eliminación y actualización de información. Esto mediante la inspección de los disparadores existentes en la base de datos y la acción a los que se aplican.
3) Chequeo automático de las anomalías de los datos, lo cual se realizará a partir de la verificación de las restricciones existentes en la base de datos.
4) Generación de log personalizado para cada caso.

2. Requisitos del sistema

2.1. Requerimientos de hardware

A continuación, se detallan los requerimientos de hardware que se necesitan para utilizar el sistema:

- Computadora de escritorio o portátil
- Mouse o ratón

2.2. Requerimientos de software

A continuación, se detallan los requerimientos de software que se necesitan para utiliza el sistema:

- Sistema operativo Unix, Linux, macOS o Windows.
- Python 3.10.1
- Librerías de Python: tkinter, pyodbc, pillow, pyhamcrest

3. Herramientas de desarrollo

3.1. Python

Es un lenguaje de programación multiparadigma (parcialmente orientado a objetos, programación imperativa y programación funcional) que se puede utilizar en múltiples sistemas operativos. Por otro lado, permite diseño de la interfaz y manejo de base de datos de manera directa y sencilla.

3.2. Base de datos (SQL Server)

Es un sistema de gestión de base de datos relacionales que permite manejar y manipular datos en base al lenguaje Transact-SQL y el uso del estándar ANSI. Además, este puede ser trabajado en un entornó gráfico que permite usar comandos DDL y DML gráficamente.

3.3. Visual Studio Code

Es un editor de código fuente para multiples lenguajes de programación, el cual incluye soporte de depuración, manejo con Git, resaltado de sintaxis, refactorización inteligente de código, entre otras opciones.

4. Instalación

4.1. Repositorio Github

El programa desarrollado se puede descargar en el respectivo repositorio en GitHub.

Enlace: https://github.com/AndersCFR/AplicacionAuditoriaBases.git

4.1. Librerías de Python

Para el correcto uso del programa desarrollado, es necesario la instalación de las librerías indicadas en la sección 2.2. Esto se puede realizar mediante los siguientes comandos:

```
pip install pyodbc
pip install tk
pip install pillow
pip install PyHamcrest==2.0.2
```

5. Estructura del proyecto

El código del repositorio mencionado previamente tiene la siguiente estructura:

1. Carpeta img: Contiene dos logos y un ícono que son usados dentro de la interfaz del programa desarrollado.
2. Carpeta Interfaz: Contiene cuatro archivos:
    a. baseDatos.py: Manejo de la conexión con la base de datos y la ejecución de los procedimientos almacenados para realizar los análisis pertinentes en base a las anomalías (tanto las que posee como las que podrían ser).
    b. interfaz.py: Lógica del diseño de la interfaz gráfica a ser mostrada y la distribución de la información obtenida.
    c. main.py: Inicialización de los objetos de la interfaz gráfica y base de datos.
    d. log.txt: Archivo de texto generado al momento de presionar el botón “Generar Log”, el cual aparece una vez realizado el análisis de la base de datos seleccionada.
3. Carpeta SQL_script: Contine 5 archivos SQL, los cuales tienen el código para la generación y eliminación de los procedimientos almacenados para el análisis de anomalías.

6. Estructura de las consultas SQL

A continuación, se mostrarán las diferentes consultas SQL que se muestran en la carpeta SQL_script del proyecto realizado:

6.1. DROP_EXISTING_SP.sql

Permite eliminar los procedimientos almacenados en caso de que ya existan:

```
IF OBJECT_ID('RelacionesDeshabilitadas') IS NOT NULL
	DROP PROCEDURE RelacionesDeshabilitadas
IF OBJECT_ID('PosiblesRelaciones') IS NOT NULL
	DROP PROCEDURE PosiblesRelaciones
IF OBJECT_ID('TriggersDeshabilitados') IS NOT NULL
	DROP PROCEDURE TriggersDeshabilitados
IF OBJECT_ID('ChequeoAutomatico') IS NOT NULL
	DROP PROCEDURE ChequeoAutomatico
```

6.2. SP_CHEQUEO_AUTOMÁTICO.sql

Permite crear el procedimiento almacenado denominado *ChequeoAutomático* que se encarga de revisar todas las restricciones existentes en la base de datos en busca de datos anómalos.

```
CREATE PROCEDURE ChequeoAutomatico
AS
BEGIN
	SET NOCOUNT ON
	CREATE TABLE DatosAnomalos (
		Tabla VARCHAR(100),
		[Restriccion de FK] VARCHAR(100),
		[Valor anomalo] VARCHAR(100)
	)
	INSERT INTO DatosAnomalos EXEC('DBCC CHECKCONSTRAINTS WITH
ALL_CONSTRAINTS')
	SELECT * FROM DatosAnomalos
	DROP TABLE DatosAnomalos
END
```

6.3. SP_POSIBLES_RELACIONES.sql

Permite crear el procedimiento almacenado denominado *PosiblesRelaciones* que inspecciona las columnas de cada tabla existente en la base de datos con el fin de encontrar aquellas que tengan nombres repetidos en diferentes tablas que no tengan relaciones entre sí. Esto con el fin de determinar posibles relaciones que deberían existir, y que al no estar presentes pueden contar con posibles anomalías de datos. Igualmente, indica las posibles tablas con las que debería tener una relación.

```
CREATE PROCEDURE PosiblesRelaciones
AS
BEGIN
	SET NOCOUNT ON
	/* Tabla temporal: Encontrar PK */
	SELECT
		tablas.name as 'Tabla',
		col.name as 'Columna PK'
	INTO Claves_Primarias
	FROM sys.tables tablas JOIN sys.indexes pk
		ON tablas.object_id = pk.object_id
		AND pk.is_primary_key = 1
	JOIN sys.index_columns ic
		ON ic.object_id = pk.object_id
		AND ic.index_id = pk.index_id
	JOIN sys.columns col
		ON pk.object_id = col.object_id
		AND col.column_id = ic.column_id
	WHERE tablas.name != 'sysdiagrams'
	AND tablas.name != 'ClavesPrimarias'

	/* Tabla temporal: Columnas existentes en cada tabla */
	SELECT
		tab.name as table_name,
		col.name as column_name,
```

```
		t.name as data_type
	INTO columnas
	FROM sys.tables tab
		JOIN sys.columns as col
			ON tab.object_id = col.object_id
		LEFT JOIN sys.types as t
			ON col.user_type_id = t.user_type_id
	WHERE tab.name != 'sysdiagrams'


	/* Obtener columnas que se repiten en otras tablas y sus posibles
relaciones inexistentes */
	SELECT
		table_name as 'Tabla',
		column_name as 'Columna',
		Tablas as 'Tablas donde la columna es PK'
	FROM (
		/* Columnas que no apuntan a un padre y al tener nombres
similares pueden implicar que falta una relación */
		SELECT
			table_name, column_name
		FROM (
			/* Columnas que tienen más de 1 ocurrencia con su
respectiva tabla */
			SELECT
				columnas.column_name,
				columnas.table_name
			FROM (
				/* Columnas que tienen más de 1 ocurrencia */
				SELECT column_name FROM columnas
				GROUP BY column_name
				HAVING COUNT(*) > 1
			) columnas_repetidas
			JOIN (
				SELECT * FROM columnas
			) columnas
			ON columnas_repetidas.column_name = columnas.column_name
			) posibles_relaciones
		FULL JOIN (
			/* Restricciones de clave foránea con sus respectivas
columnas */
			SELECT
				fk_tab.name as 'Tabla Hijo',
				pk_tab.name as 'Tabla Padre',
				fk_col.name as 'Columna FK',
				pk_col.name as 'Columna PK',
				fk.name as 'Restriccion'
			FROM sys.foreign_keys fk
				JOIN sys.tables fk_tab
					ON fk_tab.object_id = fk.parent_object_id
				JOIN sys.tables pk_tab
					ON pk_tab.object_id = fk.referenced_object_id
				JOIN sys.foreign_key_columns fk_cols
					ON fk_cols.constraint_object_id =
fk.object_id
				JOIN sys.columns fk_col
					ON fk_col.column_id =
fk_cols.parent_column_id
					AND fk_col.object_id = fk_tab.object_id
				JOIN sys.columns pk_col
					ON pk_col.column_id =
fk_cols.referenced_column_id
```

```
                                    AND pk_col.object_id = pk_tab.object_id
                ) claves_foraneas
                ON posibles_relaciones.column_name = claves_foraneas.[Columna FK]
                AND (
                        posibles_relaciones.table_name = claves_foraneas.[Tabla
Hijo]
                        OR
                        posibles_relaciones.table_name = claves_foraneas.[Tabla
Padre]
                )
                WHERE Restriccion is NULL
        ) columnas_posibles
        LEFT JOIN (
                /* Claves primarias y sus tablas */
                SELECT
                        [Columna PK],
                        /* Concatenación de los nombres de las tablas */
                        STUFF((
                                SELECT ', ' + Tabla
                                FROM Claves_Primarias
                                WHERE [Columna PK] = claves.[Columna PK]
                                FOR XML PATH(''),
TYPE).value('(./text())[1]','VARCHAR(MAX)')
                        ,1,2,'') as Tablas
                FROM Claves_Primarias claves
                GROUP BY [Columna PK]
        ) pk_tablas
        /* Se buscan tablas con las que podría necesitarse una relación */
        ON columnas_posibles.column_name = pk_tablas.[Columna PK]
        ORDER BY column_name

        /* Eliminar tablas temporales */
        DROP TABLE columnas
        DROP TABLE Claves_Primarias
END
```

6.4. SP_RELACIONES_DESHABILITADAS.sql

Permite crear el procedimiento almacenado denominado *RelacionesDeshabilitadas* que se encarga de inspeccionar las restricciones de claves foráneas existentes en la base de datos que se encuentren deshabilitadas y, por ende, faciliten la creación de datos anómalos.

```
CREATE PROCEDURE RelacionesDeshabilitadas
AS
BEGIN
        SET NOCOUNT ON
        SELECT
                name as 'Relacion',
                OBJECT_NAME(parent_object_id) as 'Tabla hijo',
                OBJECT_NAME(referenced_object_id) as 'Tabla padre',
                is_disabled as '�Deshabilitado?'
        FROM sys.foreign_keys
        WHERE is_disabled = 1
END
```

6.5. SP_TRIGGERS_DESHABILITADOS.sql

Permite crear el procedimiento almacenado denominado *TriggersDeshabilitados* que se encarga de inspeccionar los disparadores existentes en la base de datos que se encuentren deshabilitados y, por ende, faciliten la inserción, actualización o eliminación no controlada de datos, posiblemente resultando en anomalías.

```
CREATE PROCEDURE TriggersDeshabilitados
AS
BEGIN
	SET NOCOUNT ON
	SELECT
		OBJECT_NAME(te.object_id) as 'Trigger',
		te.type_desc as 'On',
		OBJECT_NAME(t.parent_id) as 'Tabla',
		t.is_disabled as '¿Deshabilitado?'
	FROM sys.trigger_events te
	JOIN sys.triggers t
		ON te.object_id = t.object_id
	WHERE is_disabled = 1
END
```

7. Prototipos del software

Se realizaron prototipos de alta fidelidad para la previsualización de la estructura gráfica básica que se tiene en el producto final desarrollado. A continuación, se presenta cada una de las pantallas indicando los datos obtenidos para el análisis de la integridad de la base de datos a analizar.

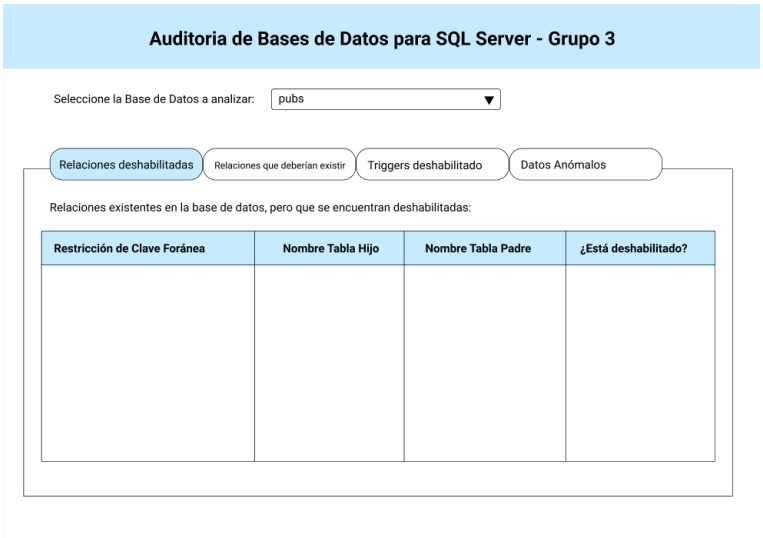


Figura 1. Sección de relaciones deshabilitadas en la base de datos.

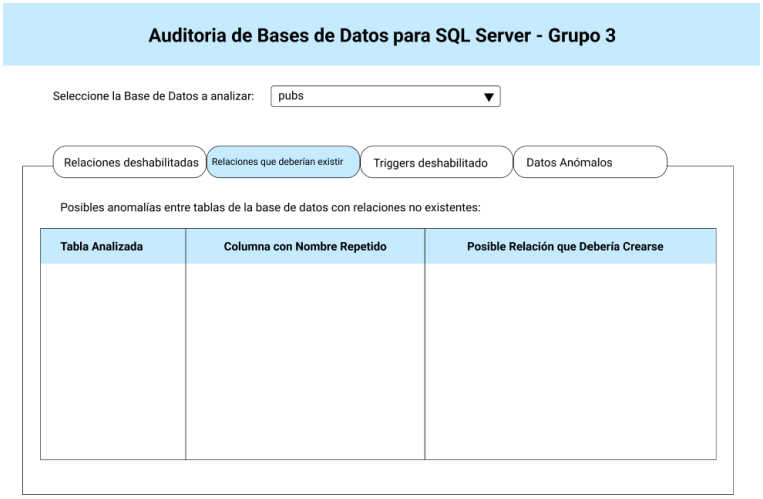


Figura 2. Sección de las relaciones que deberían existir en la base de datos.

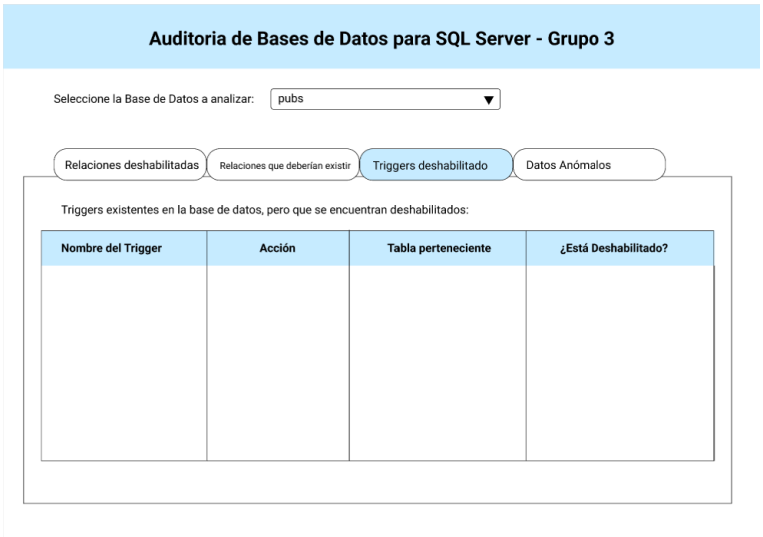


Figura 3. Sección de los triggers/disparadores deshabilitados en la base de datos.

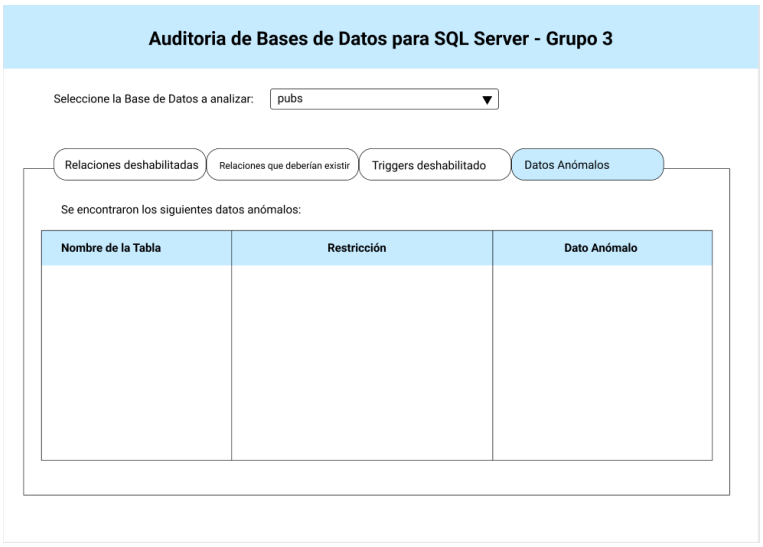


Figura 4. Sección de los datos anómalos en la base de datos.

8. Manual de uso

Para iniciar el programa se puede hacer doble clic sobre el archivo *main.py* que se encuentra en la carpeta *Interfaz* dentro de la carpeta del programa. Igualmente, es posible ejecutar el programa mediante el comando *"py main.py"* dentro de la misma carpeta.

Inmediatamente se abrirá el programa mostrando la pantalla principal:

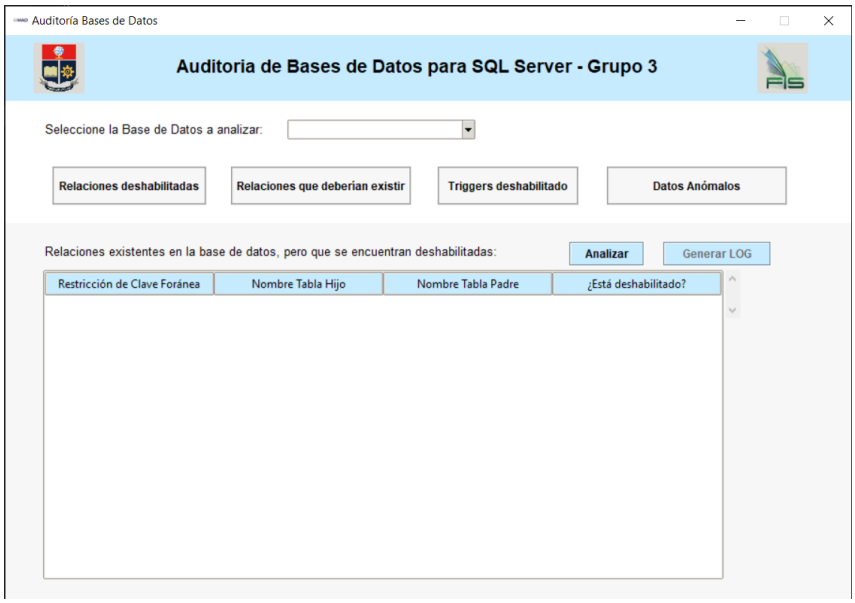


Figura 5. Apertura del programa.

A continuación, se debe seleccionar la base de datos de SQL Server que se desea analizar.

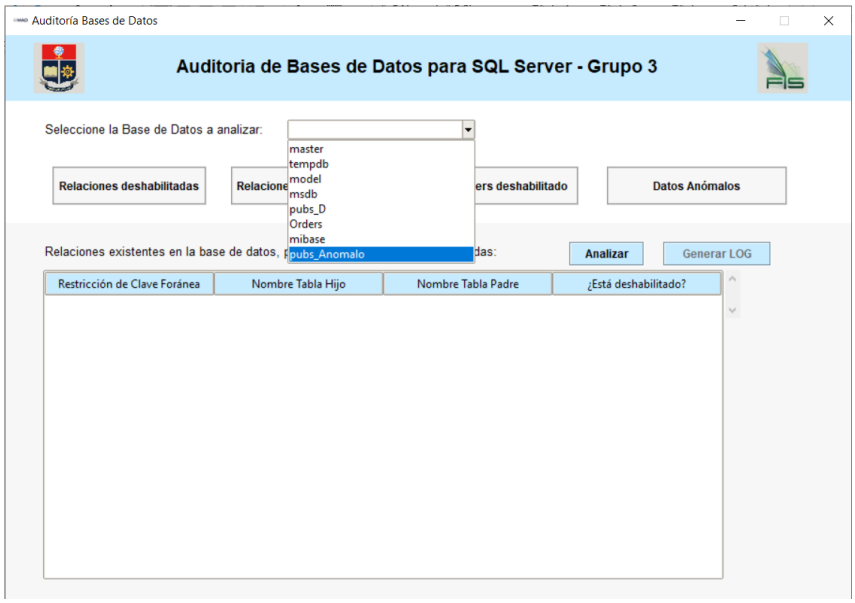


Figura 6. Selección de la base de datos a analizar.

Luego, para comenzar el proceso de auditoría se debe considerar el tipo de análisis a realizar.

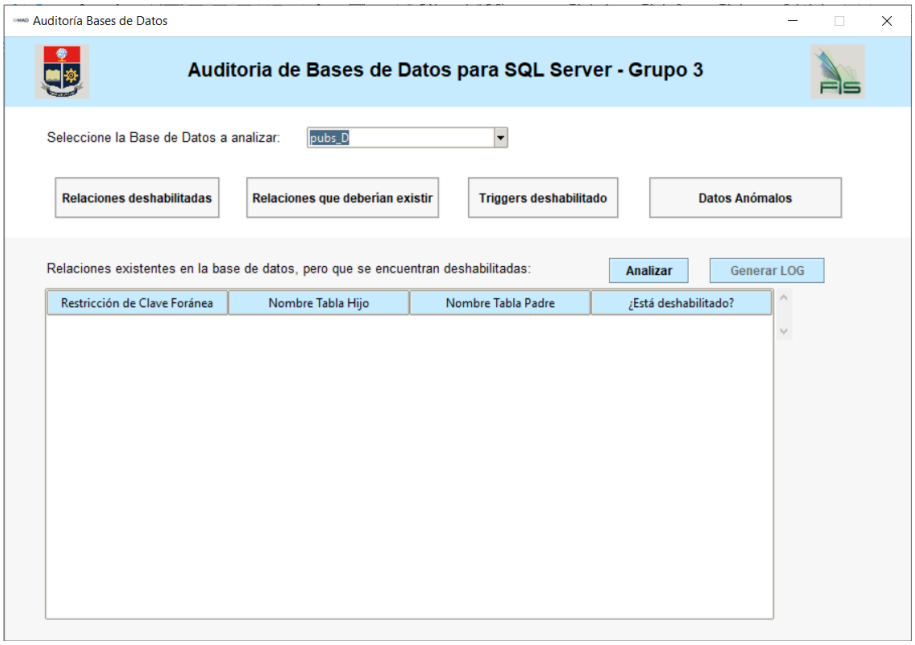


Figura 7. Tipos de análisis a ejecutarse.

Primero seleccionamos la pestaña “Relaciones deshabilitadas” y presionamos el botón “Analizar”. Inmediatamente, se actualiza la tabla indicando cuales son las claves foráneas que se encuentran deshabilitadas, el nombre de las tablas hijo y padre de la relación.

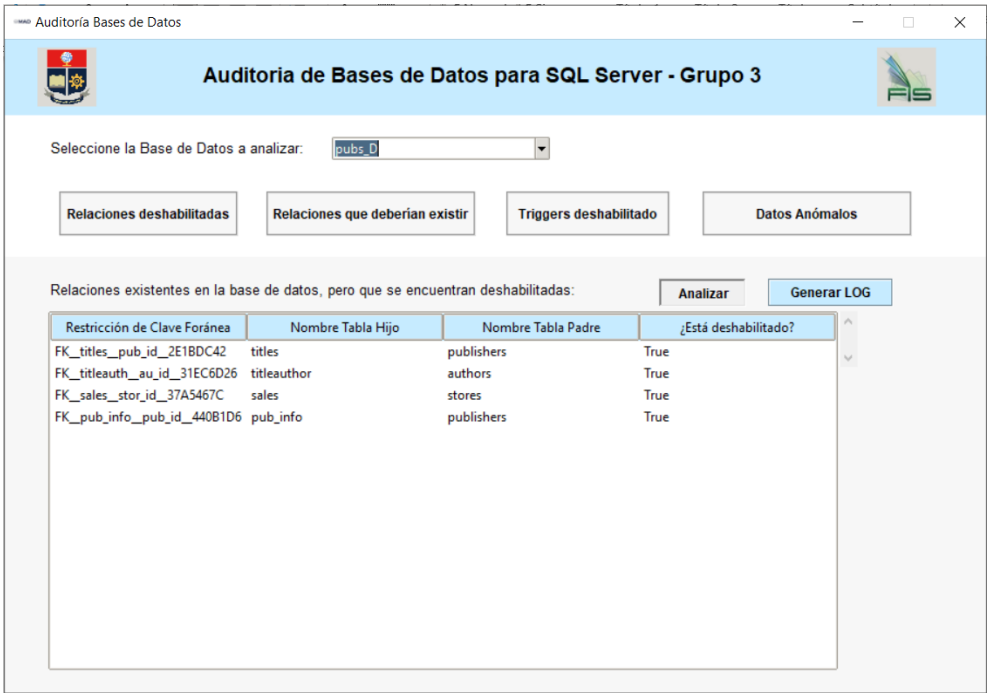


Figura 8. Análisis de relaciones deshabilitadas.

El siguiente análisis corresponde a las “Relaciones que deberían existir”, el cual muestra las tablas que tienen columnas con nombres repetidos y podrían ser relaciones que deberían estar creadas. En la tercera columna se muestran sugerencias de tablas con las que se podrían crear las relaciones.

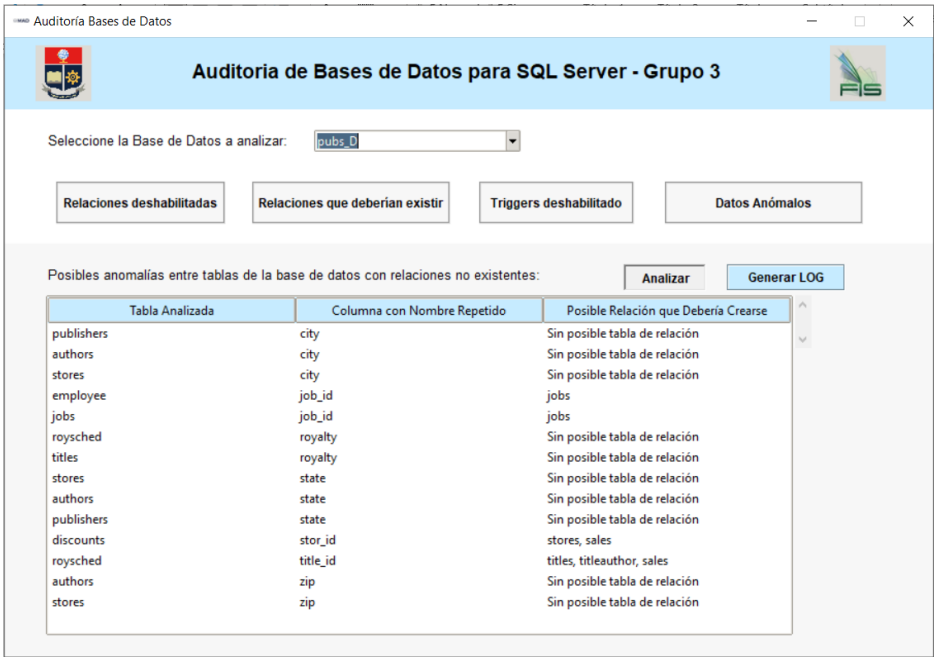


Figura 9. Análisis de relaciones que deberían existir.

El análisis de “Triggers deshabilitados” muestra los disparadores que se encuentran deshabilitados en las diferentes tablas de la base de datos seleccionada.

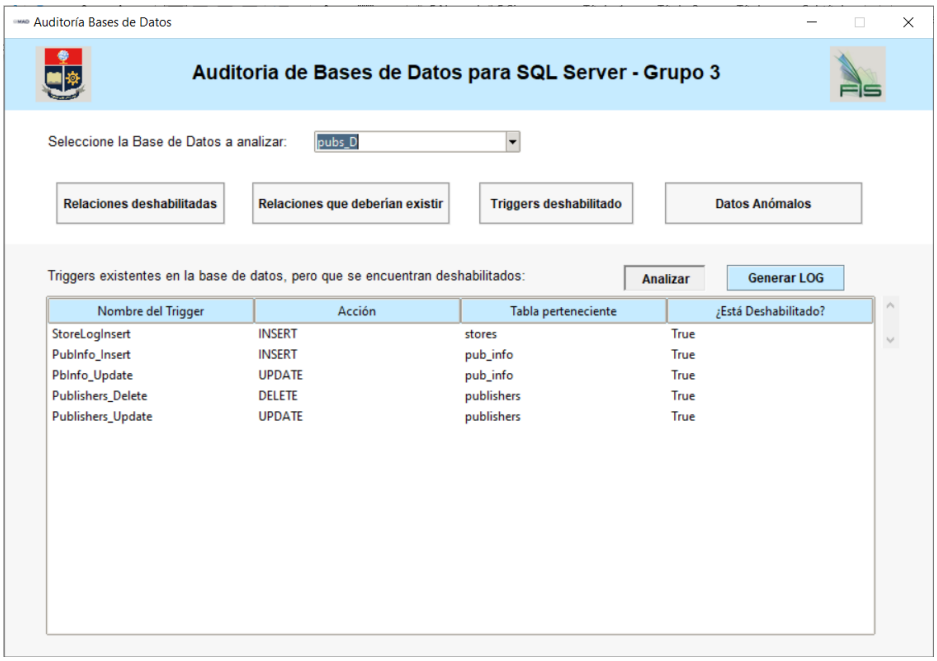


Figura 10. Análisis de disparadores deshabilitados.

Por último, la sección “Datos Anómalos” muestra cuáles son todos los problemas en datos específicos por el no cumplimiento de las restricciones respectivas:

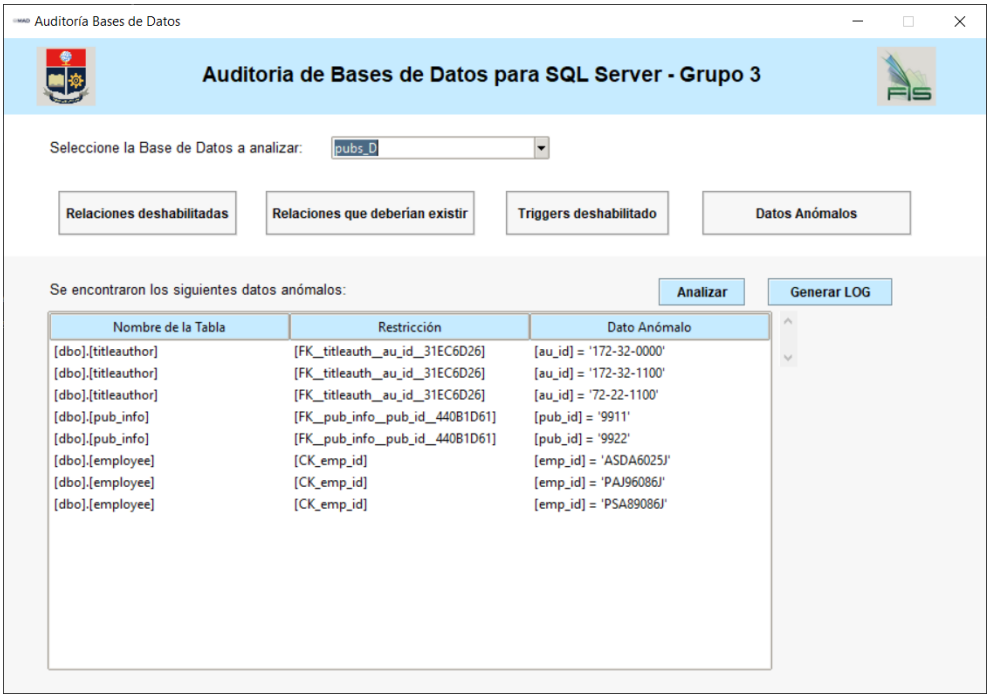


Figura 11. Análisis de datos anómalos.

En adición a las acciones realizadas previamente, se puede añadir a un archivo *log* los resultados obtenidos en cada sección. Para ello, se debe acceder a la sección deseada y presionar el botón “Generar LOG”.

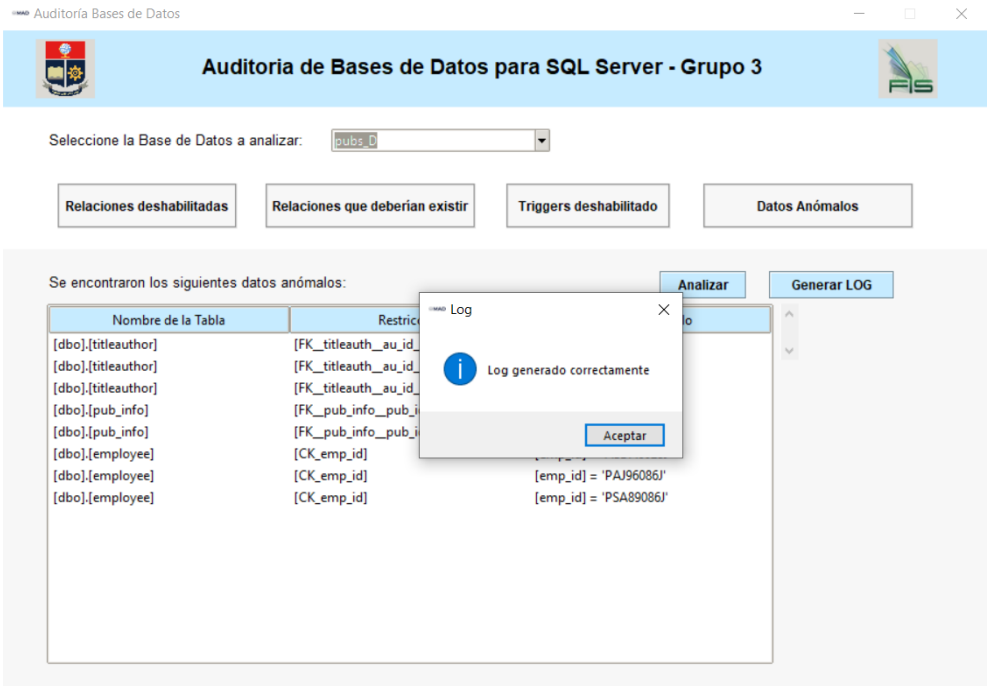
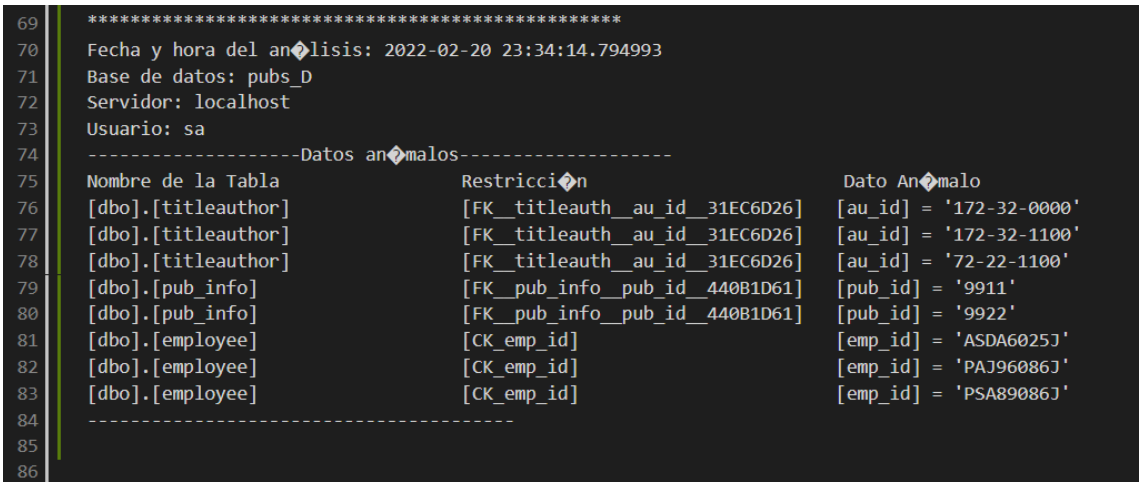


Figura 12. Generar un registro en el archivo de texto.

Como podemos ver, se añadieron las filas correspondiente a la sección de “Datos Anómalos” indicando la fecha y hora, la base de datos, el servidor y el nombre de usuario al momento de realizar la auditoría:



```
69  ************************************************
70  Fecha y hora del an□lisis: 2022-02-20 23:34:14.794993
71  Base de datos: pubs_D
72  Servidor: localhost
73  Usuario: sa
74  --------------------Datos an□malos--------------------
75  Nombre de la Tabla               Restricci□n                      Dato An□malo
76  [dbo].[titleauthor]              [FK__titleauth__au_id__31EC6D26]   [au_id] = '172-32-0000'
77  [dbo].[titleauthor]              [FK__titleauth__au_id__31EC6D26]   [au_id] = '172-32-1100'
78  [dbo].[titleauthor]              [FK__titleauth__au_id__31EC6D26]   [au_id] = '72-22-1100'
79  [dbo].[pub_info]                 [FK__pub_info__pub_id__440B1D61]   [pub_id] = '9911'
80  [dbo].[pub_info]                 [FK__pub_info__pub_id__440B1D61]   [pub_id] = '9922'
81  [dbo].[employee]                 [CK_emp_id]                      [emp_id] = 'ASDA6025J'
82  [dbo].[employee]                 [CK_emp_id]                      [emp_id] = 'PAJ96086J'
83  [dbo].[employee]                 [CK_emp_id]                      [emp_id] = 'PSA89086J'
84  ---------------------------------------
85
86
```

Figura 13. Datos guardados en el archivo de texto.