Universidad Nacional del Nordeste

Facultad de Ciencias Exactas y Naturales y Agrimensura

Proyecto de Estudio e Investigación

Tema: “Actualización y gestión de una base de datos de empleados de una empresa forestal”

Carrera: Licenciatura en Sistemas de Información

Materia: Base de Datos I

Profesor: Villegas, Darío Oscar

Fecha: 14/11/2022

Integrantes:

- BARTOLUCCI, Gastón Leonel Lu: 55.425
- OJEDA, Lidia Mariangeles Lu: 56.100
- RAMIREZ, Alfredo Agustín Lu: 46.911
- ZAPATA, Sergio Ariel Lu: 46.789

Índice:

Índice de imágenes

Capítulo 1: Introducción

Tema:

Actualización y gestión de una base de datos sobre los empleados forestales de la empresa “Servicios Forestales de Lorenzo Zimmermann” de la provincia de Misiones

Definición o planteamiento del problema:

Descripción

La empresa “Servicios Forestales de Lorenzo Zimmermann” tiene actualmente sus datos almacenados en un sistema ya obsoleto, se pretende remodelar la base de datos hacia una forma más adecuada y funcional para el manejo de los mismos. Dicha empresa se dedica a la gestión integral de servicios forestales de producción de cortes de madera y pasta de celulosa para papel.

Se desea conocer las producciones que realizan cada empleado de la empresa con el fin de realizar las liquidaciones de los sueldos que están sujetas a estas producciones que serán sumados al sueldo básico mensual y descontado los insumos y combustibles que se hubieran utilizado para la labor que fueron suministrados por el stock de la empresa pero que corre a cuenta de los empleados. A su vez, se puede realizar descuento por los vales de supermercados o anticipos que pudieron haber solicitado en cierto periodo. Por cada servicio de forestación se desea conocer el tipo de corte realizado que tienen diferentes precios, quienes lo realizaron y qué insumos fueron utilizados.

Para desarrollar este proyecto tomamos la idea de la migración de la base de datos creada con un gestor de base de datos para el sistema operativo DOS llamada dbase, con el software construido con un lenguaje llamado Clipper. Y debido a que las nuevas versiones de Windows ya no brindan el soporte para ese tipo de software es necesario esta actualización, para esto tratamos de combinar todos los datos de esas tablas y volcarlos a una base de datos.

Este proyecto tendrá las siguientes restricciones para su funcionamiento:

1 - Cada legajo del empleado es único

2 - El registro de producciones y remito son únicos

Alcance del Proyecto

El proyecto que se desea realizar está basado en un software que lleve a cabo la gestión y control de las producciones que realizan los empleados de la empresa forestal de Misiones orientado a calcular el sueldo final de cada empleado luego de realizar las deducciones correspondientes, como ser los insumos que utilizaron y los anticipos y vales de supermercados que hayan solicitado. En él se puede encontrar los remitos de las entregas de los diferentes cortes de madera realizados en una producción.

Objetivos del trabajo práctico

Objetivos generales

Realizar la actualización de la base de datos sobre los empleados forestales de la empresa logrando la normalización de la misma.

Generar las funcionalidades necesarias para llevar a cabo la gestión adecuada de la base de datos en cuestión.

Objetivos específicos

- Registrar las producciones que se realizan, que empleados las realizaron y los tipos de corte que se produjeron.
- Registrar los diferentes remitos de cada entrega realizada por los empleados que corresponde en determinada fecha detallando la cantidad.
- Mostrar los insumos que utilizan cada uno de los empleados ya que los mismos luego deberán ser deducidos del total a recibir en su recibo de sueldo.

- Registrar los anticipos de sueldo que sean solicitados como así también los vales de supermercado que pueden ser usados para el retiro de mercadería en distintos negocios de la ciudad, ambos con sus correspondiente fecha y monto total.

CAPÍTULO II: Metodología seguida

Descripción de cómo se realizó el Trabajo Práctico.

En este trabajo, al ser una base de dato ya existente y en funcionamiento el primer paso fue el entendimiento general de los componentes y entidades necesarias para lograr el objetivo con el cual fue creada. Para esto decidimos plantear el modelo de datos desde el inicio tomado los componentes necesarios y descartando los que no aportan información meritoria. Ya con el modelo completo procedimos a crear las tablas y sus respectivas relaciones en el motor de base de datos SQL Server. Luego volcamos un lote de datos representativos a cada tabla, algunos reales tomados de la base de datos original y otros ficticios que en su conjunto nos permitiera realizar las consultar necesarias para la prueba del funcionamiento y sus consecuentes correcciones derivadas de la prueba y error que surgieron. Finalmente nos focalizamos en el desarrollo de los temas técnicos correspondientes que serán tratados más adelante.

Herramientas (Instrumentos y procedimientos)

La metodología de trabajo, además de algunas reuniones personales, se desarrolló mayormente en entorno distribuido para lo cual se usa reuniones vía web con Google Meet y usando distintas herramientas para su desarrollo. Para el desarrollo del documento se utilizó Google Docs como procesador de texto para la escritura en conjunto de los miembros del grupo y Google Sheets cuando se requirió el uso de una hoja de cálculo. La principal herramienta fue le motor de base de datos SQL Server para el tratamiento de los datos y tablas relacionales usadas. A su vez, usamos el sistema de control de versiones GitHub para alojar el proyecto y poder de esta forma trabajar en conjunto en una sola versión del

proyecto y tener siempre acceso a los cambios que realice cualquier miembro del grupo. Para la organización y distribución de tareas de uso el software Trello creado para la administración de proyectos. Para el modelado de datos y el diagrama entidad relación usamos Lucidchart y Vertabelo como herramienta de diagramación. La extracción de los datos para el lote de prueba se usó la planilla de cálculos Excel y el programa ESF Database Migration Toolkit.

CAPÍTULO III: Desarrollo del tema / presentación de resultados

Diagrama Entidad Relación

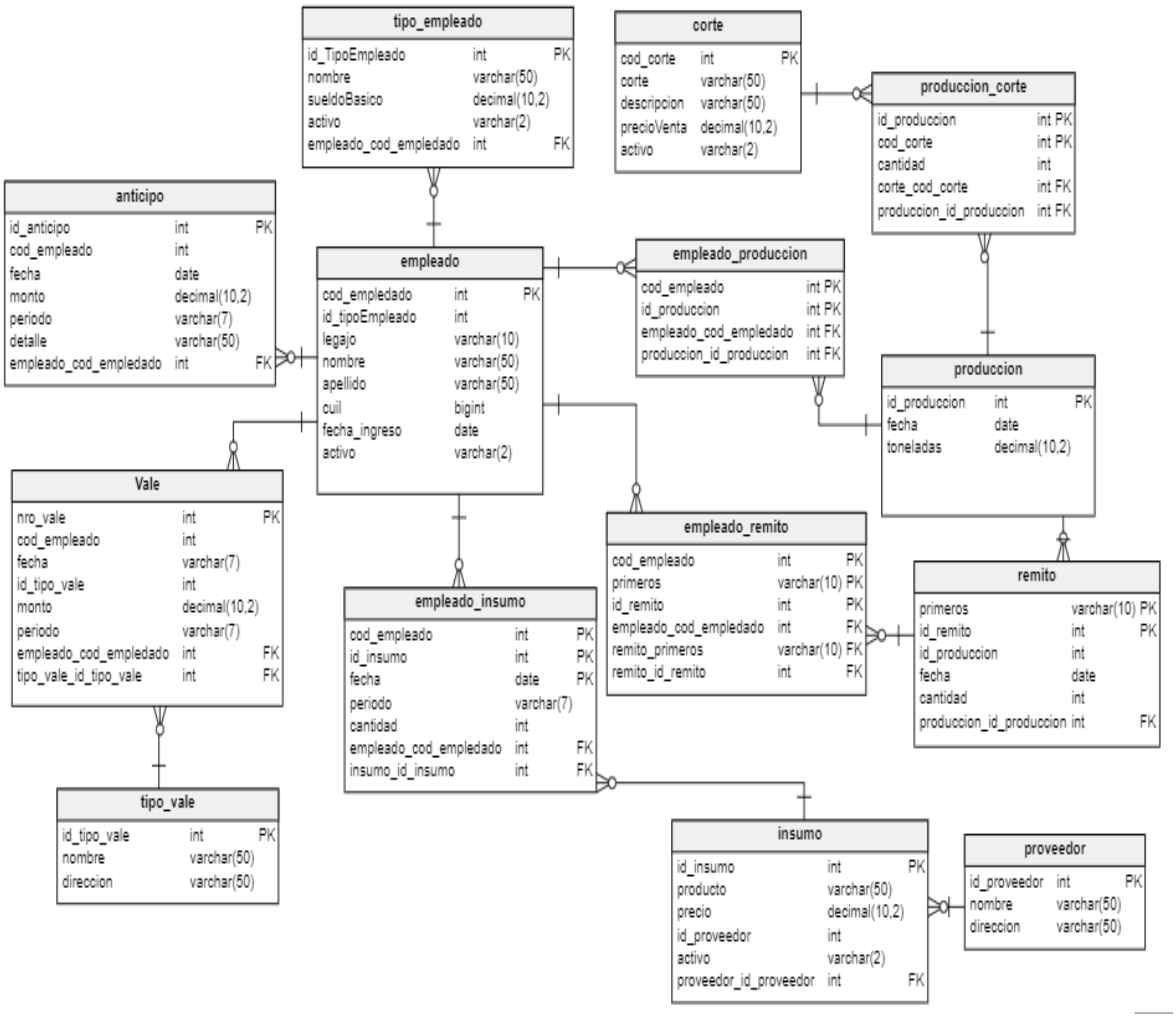

Imagen 1: Diagrama Entidad Relación. Elaboración propia en Vertabelo.

Diccionario de datos:

A - Empleados							
La tabla contendrá los datos de los empleados de la empresa							
Nombre	Tipo de Dato	Longitud	Nulo	Clave Primaria	Clave Foránea	Tabla Referenciada	Descripción
Cod_empleado	INT	4	NO	SI	NO	-	Código que identifica a cada empleado
Legajo	INT	4	NO	NO	NO	-	Número de legajo del empleado
Nombre	VARCHAR	50	NO	NO	NO	-	Nombre del empleado
Apellido	VARCHAR	50	NO	NO	NO	-	Apellido del empleado
Cuil	BIGINT	11	NO	NO	NO	-	Cuil del empleado
Fecha_ingreso	DATE	8	NO	NO	NO	-	Fecha de ingreso a la empresa del empleado
Id_TipoEmpleado	INT	4	NO	NO	SI	TipoEmpleado	Identificación del tipo de empleado
Activo	CHAR	2	NO	NO	NO	-	Identifica si el empleado esta activo o no

B - TipoEmpleado							
La tabla contendrá los distintos tipos de empleados con sus respectivas funciones que realiza							
Nombre	Tipo de Dato	Longitud	Nulo	Clave Primaria	Clave Foránea	Tabla Referenciada	Descripción
Id_TipoEmpledado	INT	4	NO	SI	NO	-	Identificación del tipo de empleado
Nombre	VARCHAR	50	NO	NO	NO	-	Nombre del tipo de empleado
SueldoBasico	FLOAT	10.2	NO	NO	NO	-	Sueldo básico del tipo de empelado
Activo	CHAR	2	NO	NO	NO	-	Identifica si el tipo de empleado esta activo o no

C - Producciones							
La tabla contendrá los datos de las producciones o servicios que se realizan en determinadas áreas							
Nombre	Tipo de Dato	Longitud	Nulo	Clave Primaria	Clave Foránea	Tabla Referenciada	Descripción
Id_Producción	INT	4	NO	SI	NO	-	Identificación de la producción
Cod_Corte	INT	4	NO	NO	SI	Corte	Código del corte
toneladas	FLOAT	(10,2)	NO	NO	NO	-	Representa las toneladas de cada producción

D - Empleados-Producciones							
La tabla contendrá los datos de los empleados que trabajaron en cada producción							
Nombre	Tipo de Dato	Longitud	Nulo	Clave Primaria	Clave Foránea	Tabla Referenciada	Descripción
Cod_Empleado	INT	4	NO	SI	SI	Empleado	Código que identifica a cada empleado
Id_Producción	INT	4	NO	SI	SI	Producciones	Identificación de la producción

E - Corte							
La tabla contendrá los datos sobre los cortes que se realizan en las producciones							
Nombre	Tipo de Dato	Longitud	Nulo	Clave Primaria	Clave Foránea	Tabla Referenciada	Descripción
Cod_Corte	INT	4	NO	SI	NO	-	Número que identifica al registro del corte.
Corte	VARCHAR	50	NO	NO	NO	-	Nombre del corte.
Descripción	VARCHAR	200	NO	NO	NO	-	Descripción ampliada del corte.
PrecioVenta	FLOAT	(10,2)	NO	NO	NO	-	Precio de cada corte para cada empleado.
Activo	CHAR	2	NO	NO	NO	-	Identifica si el corte está activo o no.

F - Remito							
La tabla contendrá los datos de los remitos que representas las entregas físicas de las producciones							
Nombre	Tipo de Dato	Longitud	Nulo	Clave Primaria	Clave Foránea	Tabla Referenciada	Descripción
Primeros	INT	4	NO	SI	NO	-	Representa los primeros cuatros dígitos del número de remito.
Id_Remito	INT	8	NO	SI	NO	-	Código que identifica al remito.
Fecha	DATETIME	8	NO	NO	NO	-	Fecha del remito.
UnidadMedida	VARCHAR	50	NO	NO	NO	-	Unidad de medida de la producción entregada.
Cantidad	INT	10	NO	NO	NO	-	Cantidad de la producción.
Id_Produccción	INT	4	NO	NO	SI	Producciones	Identificación de la producción.

G - Empleado-Remito							
La tabla contendrá los datos de los empleados que entregaron determinado remito							
Nombre	Tipo de Dato	Longitud	Nulo	Clave Primaria	Clave Foránea	Tabla Referenciada	Descripción
Cod-Empleado	INT	4	NO	SI	SI	Empleado	Código que identifica a cada empleado
Primeros	INT	4	NO	SI	SI	Remito	Representa los primeros cuatros dígitos del número de remito.
Id_Remito	INT	10	NO	SI	SI	Remito	Código que identifica al remito.

H - Insumo							
La tabla contendrá los datos de los insumos que se usen en la empresa							
Nombre	Tipo de Dato	Longitud	Nulo	Clave Primaria	Clave Foránea	Tabla Referenciada	Descripción
Id_Insumo	INT	4	NO	SI	NO	-	Código que identifica al insumo.
Producto	VARCHAR	50	NO	NO	NO	-	Nombre del insumo.
Precio	FLOAT	(10,2)	NO	NO	NO	-	Precio del insumo.
Proveedor	VARCHAR	50	NO	NO	NO	-	Nombre del proveedor del insumo.
Activo	CHAR	2	NO	NO	NO	-	Identifica si el insumo está activo o no.

I - Empleado-Insumo							
La tabla contendrá los datos de los insumos que usen cada uno de los empleados en un determinado momento							
Nombre	Tipo de Dato	Longitud	Nulo	Clave Primaria	Clave Foránea	Tabla Referenciada	Descripción
Cod_Empleado	INT	4	NO	SI	SI	Empleado	Código que representa a un empleado
Id_Insumo	INT	4	NO	SI	SI	Insumo	Código que identifica al insumo.
Fecha	DATETIME	8	NO	NO	NO	-	Fecha en que fue consumido el insumo
periodo	VARCHAR	10	NO	NO	NO	-	Mes que el empleado usa el insumo
cantidad	FLOAT	(10,2)	NO	NO	NO	-	Cantidad de cada insumo

J - Anticipo							
La tabla contendrá los datos de los anticipos de sueldo que solicitan los empleados							
Nombre	Tipo de Dato	Longitud	Nulo	Clave Primaria	Clave Foránea	Tabla Referenciada	Descripción
Id_Anticipo	INT	8	NO	SI	NO	-	Código que representa anticipo otorgado al empleado
Cod_Empleado	INT	4	NO	NO	SI	Empleado	Código que representa a un empleado
Fecha	DATETIME	8	NO	NO	NO	-	Fecha en la que se otorgó el anticipo
Monto	FLOAT	(10,2)	NO	NO	NO	-	Monto del anticipo
Periodo	VARCHAR	7	NO	NO	NO	-	Mes al que corresponde el anticipo
Detalle	VARCHAR	200	NO	NO	NO	-	Detalle que incorpora motivo del anticipo

K - Vale							
La tabla contendrá los datos de los vales de supermercado que solicitan los empleados							
Nombre	Tipo de Dato	Longitud	Nulo	Clave Primaria	Clave Foránea	Tabla Referenciada	Descripción
Id_Vale	INT	8	NO	SI	NO	-	Código que representa vale de mercadería otorgado al empleado
Cod_Empleado	INT	4	NO	NO	SI	Empleado	Código que representa a un empleado
fecha	DATE	8	NO	NO	NO	-	Fecha en la que otorgó el Vale
Monto	FLOAT	(10,2)	NO	NO	NO	-	Monto del vale
Id_Supermercado	INT	4	NO	NO	SI	Supermercado	Código que representa al supermercado para consumo del vale
Periodo	VARCHAR	7	NO	NO	NO	.	Mes al que corresponde el vale

L – Tipo_vale							
La tabla contendrá los datos de los supermercados y otros negocios en los cuales los empleados pueden cambiar los vales							
Nombre	Tipo de Dato	Longitud	Nulo	Clave Primaria	Clave Foránea	Tabla Referenciada	Descripción
Id_tipo_vale	INT	4	NO	SI	NO	-	Código que representa al negocio para consumo de vales
Nombre	VARCHAR	50	NO	NO	NO	-	Nombre del supermercado
Dirección	VARCHAR	50	NO	NO	NO	-	Dirección del supermercado

Para entender mejor la situación de una producción realizamos un cuadro que resume la situación de las producciones 1 y 2 del lote de datos creado

Producción 1 – 100 tn			
Cortes	4		25 tn
	8		30 tn
	2		45 tn
Empleados	3		
	2		
Remitos	ID	Tn	Empleados
	1	30	3 - 2
	2	70	3 - 2

Producción 2 – 200 tn			
Cortes	8		70 tn
	4		20 tn
	3		40 tn
	1		70 tn
Empleados	3		
	6		
	1		
	7		
	10		
Remitos	ID	Tn	Empleados
	3	20	1-10
	4	90	3-6-1-7-10
	5	30	6-3
	6	20	1-3
	7	40	10

Temas técnicos

Funciones y procedimientos almacenados

Procedimientos Almacenados: Un procedimiento almacenado (store procedure) es un programa (o procedimiento) el cual es almacenado físicamente en una base de datos. Generalmente son escritos en un lenguaje de datos como SQL. La ventaja de un procedimiento almacenado es que, al ser ejecutado, en respuesta de una petición de

usuario, es ejecutado directamente en el motor de la base de datos en el cual usualmente corre en un servidor separado. Como tal posee acceso directo a los datos que necesita manipular y sólo necesita enviar sus resultados de regreso al usuario, deshaciéndose de la sobrecarga resultante de comunicar grandes cantidades de datos salientes y entrantes. Los procedimientos se usan para realizar consultas SQL sobre los objetos de la base de datos de una manera abstracta. Los procedimientos almacenados pueden recibir y devolver información, para ello se emplea parámetros, de entrada y salida, respectivamente. Los procedimientos almacenados pueden hacer referencias a tablas, vistas, funciones definidas por el usuario y otros procedimientos almacenados, así como a tablas temporales. Si un procedimiento almacenado crea una tabla local temporal, la tabla temporal sólo existe para atender al procedimiento almacenado y desaparece cuando finaliza la ejecución del mismo.

Creación de un procedimiento almacenado: Para crear un procedimiento almacenado usaremos la siguiente instrucción CREATE PROCEDURE.

```
CREATE PROCEDURE [dbo].[AltaRemito]
    @primeros INT,
    @id_produccion INT,
    @fecha DATE,
    @cantidad INT
AS
BEGIN
    BEGIN TRY --manejador de errores
            INSERT INTO remito(
                        primeros,
                        id_produccion,
                        fecha,
                        cantidad)
            VALUES (
                    @primeros,
                    @id_produccion,
                    @fecha,
                    @cantidad)
                PRINT 'ALTA DE REMITO GUARDADO CORRECTAMENTE'
    END TRY
    BEGIN CATCH
                PRINT 'OCURRIO UN ERROR '+'EN LA LINEA '+CONVERT(VARCHAR(255), ERROR_LINE())+'.'
    END CATCH
END
```

Imagen 2: Ejemplo creación de un procedimiento almacenado

Modificación de una base de datos: Para modificar un procedimiento almacenado existente y conservar la asignación de los permisos, use la instrucción ALTER PROCEDURE. SQL Server sustituye la definición anterior al procedimiento almacenado cuando se modifica con ALTER PROCEDURE. En la siguiente figura podemos ver cómo modificamos un procedimiento almacenado

```
ALTER PROCEDURE [dbo].[AltaRemito]
    @primeros INT,
    @id_produccion INT,
    @fecha DATE,
    @cantidad INT
AS
BEGIN
    BEGIN TRY --manejador de errores
            INSERT INTO remito(
                        primeros,
                        id_produccion,
                        fecha,
                        cantidad)
            VALUES (
                    @primeros,
                    @id_produccion,
                    @fecha,
                    @cantidad)
                PRINT 'ALTA DE REMITO GUARDADO CORRECTAMENTE'
    END TRY
    BEGIN CATCH
                PRINT 'OCURRIO UN ERROR '+'EN LA LINEA '+CONVERT(VARCHAR(255), ERROR_LINE())+'.'
    END CATCH
END
```

Imagen 3: Ejemplo modificación de un procedimiento almacenado

Eliminación de un procedimiento Almacenado: La instrucción DROP PROCEDURE se utiliza para quitar procedimientos almacenados definidos por el usuario de la base de datos actual.

```
DROP PROCEDURE AltaRemito
```

Funciones: Las funciones es un conjunto de instrucciones SQL que realizan una tarea específica de manera automática. Las funciones fomentan la reutilización del código.

Una función es una rutina almacenada que recibe unos parámetros escalares de entrada, los procesa según la definición de la función y finalmente retorna un resultado de un tipo específico que permitirá su utilización con un objetivo.

Existen tres tipos de funciones:

- Funciones escalares

Las funciones escalares son aquellas que reciben parámetros de entrada para ser procesados y al final retornar en un valor con un tipo de dato sencillo. Es decir un tipo de dato elemental como INT, FLOAT, VARCHAR, etc. Pues SQL Server no permite que este tipo de funciones retorna valores de tipo text, ntext, image, cursor y timestamp. Utilizaremos la palabra reservada Returns para indicar el tipo de dato en el cual retornará la función. El cuerpo de una función escalar estará contenido en un bloque de instrucciones como en los procedimientos almacenados.

- Funciones con valores de tabla en línea

Este tipo de función tiene la misma sintaxis que una función escalar, la única diferencia es que devuelve tipo de dato TABLE (una tabla compuesta de registros).

Funciones con valores de tabla y múltiples instrucciones

Este tipo de funciones es similar a las funciones de tabla en línea, pero ahora incluyen un bloque de sentencias para manipular la información antes de retornar la tabla.

Creación de una función: Para crear una usaremos la siguiente instrucción CREATE FUNCTION, de esta manera hemos creado lo mencionado antes.

```
CREATE FUNCTION  [dbo].[F_InsumpoPorEmpleado] (
@empleado INT
)
RETURNS FLOAT
AS
BEGIN
DECLARE @total  DECIMAL(10,2)
    SELECT @total  =  SUM(precio * cantidad)
        FROM empleado e
        INNER JOIN empleado_insumo ei ON (e.cod_empleado = ei.cod_empleado)
        INNER JOIN insumo i ON (ei.id_insumo = i.id_insumo)
        WHERE ei.cod_empleado = @empleado
    RETURN @total
END
```

Imagen 4: creación de una función

Modificación de una función: Para modificar una función existente y conservar la asignación de los permisos, use la instrucción ALTER FUNCTION. SQL Server sustituye la definición anterior a la FUNCIÓN, cuando se modifica con ALTER FUNCIÓN. En la siguiente figura podemos ver cómo modificamos un procedimiento almacenado

```
ALTER FUNCTION  [dbo].[F_InsumpoPorEmpleado] (
@empleado INT
)
RETURNS FLOAT
AS
BEGIN
DECLARE @total  DECIMAL(10,2)
    SELECT @total  =  SUM(precio * cantidad)
        FROM empleado e
        INNER JOIN empleado_insumo ei ON (e.cod_empleado = ei.cod_empleado)
        INNER JOIN insumo i ON (ei.id_insumo = i.id_insumo)
        WHERE ei.cod_empleado = @empleado
    RETURN @total
END
```

Imagen 5: modificación de una función

Eliminación de un procedimiento Almacenado: La instrucción DROP FUNCTION se utiliza para quitar una función definida por el usuario de la base de datos actual.

```
DROP FUNCTION dbo.F_InsumpoPorEmpleado
```

Disparadores

Disparador o trigger: Consiste en una serie de procedimientos que se ejecutan, según instrucciones definidas, cuando se lleven a cabo determinadas operaciones, sobre la información que contiene una base de datos. Los disparadores o triggers, sirven para gestionar de mejor manera la base datos de manera automática, sin que una persona tenga que intervenir. Generalmente se utilizan para garantizar la integridad de la información, a través de restricciones, requerimientos o acciones de verificación, para evitar errores y facilitar la sincronización de la información.

Los trigger se puede ejecutar cuando el usuario realiza alguna acción relacionada con añadir, actualizar o eliminar información de una tabla. Es decir, al usar los comandos INSERT, UPDATE o DELETE. Por lo tanto, para poder usar un trigger es necesario que el usuario posea permisos INSERT y DELETE en dicha base de datos.

TIPOS DE TRIGGERS

Existen diferentes tipos de disparadores, en función de las ejecuciones que realizan.

- Disparadores de fila: también llamados row triggers, son aquellos cuya ejecución se realiza a través de llamadas desde una tabla asociada al trigger.
- Disparadores de secuencia. también llamados statement triggers, son aquellos que se ejecutan solo una vez, independientemente de la cantidad de veces que se cumplan las condiciones para su ejecución.

Los triggers pueden ejecutar su acción en diferentes momentos.

- Antes de ejecutar la sentencia (before statement)
- Después de ejecutar la sentencia y de comprobar las restricciones y condiciones aplicables (after statement)

- Antes de modificar la fila de la tabla afectada por la sentencia del trigger, y de comprobar las restricciones y condiciones de ejecución (before row)
- Después de modificar la fila de la tabla afectada por la sentencia del trigger, y de comprobar las restricciones y condiciones de ejecución (after row)

En este se usaron triggers principalmente para cuando se hace alguna modificación en alguna tabla se realicen los cambios necesarios en las tablas asociadas. Como ejemplo podemos plantear la situación en que un tipo de empleado deje de estar activo todos los empleados de ese tipo también pasaran al estado de inactivo:

```
--Cambia el activo a 'NO' de un empleado cuando el tipo de empleado asociado cambia a 'NO'
CREATE TRIGGER TR_Tipoempleado ON tipo_empleado FOR UPDATE
AS SET NOCOUNT ON
DECLARE @id int = (SELECT id_TipoEmpleado FROM inserted)
UPDATE empleado SET activo = (SELECT activo FROM tipo_empleado
                              WHERE id_TipoEmpleado = @id)
WHERE id_TipoEmpleado = @id

UPDATE tipo_empleado SET activo = 'NO' WHERE id_TipoEmpleado = 8
```

Imagen 6: Ejemplo de trigger

Transacciones

Una transacción es un conjunto de operaciones Transact SQL que se ejecutan como un único bloque, es decir, si falla una operación Transact SQL fallan todas. Si una transacción tiene éxito, todas las modificaciones de los datos realizadas durante la transacción se confirman y se convierten en una parte permanente de la base de datos. Si una transacción encuentra errores y debe cancelarse o revertirse, se borran todas las modificaciones de los datos.

La transacción más simple en SQL Server es una única sentencia SQL. una transacción 'autocommit', una transacción autocompletada.

```
UPDATE clientes SET sexo='F' WHERE sexo ='FEMENINO'
```

Cuando enviamos esta sentencia al SQL Server se escribe en el fichero de transacciones lo que va a ocurrir y a continuación realiza los cambios necesarios en la base de datos. Si hay

algún tipo de problema al hacer esta operación el SQL Server puede leer en el fichero de transacciones lo que se estaba haciendo y si es necesario puede devolver la base de datos al estado en el que se encontraba antes de recibir la sentencia.

Por supuesto este tipo de transacciones no requieren de nuestra intervención puesto que el sistema se encarga de todo. Sin embargo, si hay que realizar varias operaciones y queremos que sean tratadas como una unidad tenemos que crear esas transacciones de manera explícita.

La sentencia que se utiliza para indicar el comienzo de una transacción es 'BEGIN TRAN'. Si alguna de las operaciones de una transacción falla hay que deshacer la transacción en su totalidad para volver al estado inicial en el que estaba la base de datos antes de empezar. Esto se consigue con la sentencia 'ROLLBACK TRAN'.

Si todas las operaciones de una transacción se completan con éxito hay que marcar el fin de una transacción para que la base de datos vuelva a estar en un estado consistente con la sentencia 'COMMIT TRAN'.

La transacción sigue activa hasta que se emita una instrucción COMMIT o ROLLBACK. Una vez que la primera transacción se ha confirmado o revertido, se inicia automáticamente una nueva transacción la siguiente vez que la conexión ejecuta una instrucción para modificar datos.

- BEGIN TRANSACTION o BEGIN TRAN: marca el inicio de una transacción. TRAN es un sinónimo de TRANSACTION y se suele usar más a menudo por abreviar.
- ROLLBACK TRANSATION o ROLLBACK TRAN: fuerza que se deshaga la transacción en caso de haber un problema o querer abandonarla. Cierra la transacción.
- COMMIT TRANSACTION O COMMIT TRAN: confirma el conjunto de operaciones convirtiendo los datos en definitivos. Marca el éxito de la operación de bloque y cierra la transacción.

En el presente trabajo se usaron transacciones en las operaciones en que se requieren la inserción conjunta de los datos, por ejemplo, cuando se carga un remito a los empleados implicados debe cargar a todos y si hay un error no cargarle a ninguno.

```
DECLARE @alta VARCHAR(50);
SELECT @alta = 'AltaEmpleados_Remitos';

    BEGIN TRANSACTION @Alta;
    USE Forestal;

    BEGIN TRY
        INSERT INTO empleado_remito (cod_empleado, primeros, id_remito) VALUES (9,6,14);
        INSERT INTO empleado_remito (cod_empleado, primeros, id_remito) VALUES (2,6,14);
        INSERT INTO empleado_remito (cod_empleado, primeros, id_remito) VALUES (8,6,14);
        INSERT INTO empleado_remito (cod_empleado, primeros, id_remito) VALUES (14,6,14);
        COMMIT TRANSACTION @Alta
        PRINT 'EL REMITO SE HA AGREGADO CORRECTAMENTE A CADA EMPLEADO'
    END TRY
    BEGIN CATCH
        /*SI OCCURRE UN ERROR EN EL ALTA*/
        ROLLBACK TRANSACTION @Alta
        RAISERROR('HUBO UN ERROR INESPERADO', 16, 1)
    END CATCH
```

Imagen 7: Ejemplo de transacción

Vistas

Una vista es una tabla virtual que su contenido está definido por una consulta, que al igual que una tabla, consta con un conjunto de columnas y filas de datos con un nombre, a menos que esta esté indizada. Una vista no existe como conjunto de valores de datos en una base de datos, mientras sus filas y columnas proceden de tablas a las que se hace referencia en la consulta que define la vista y procede de forma dinámica cuando se hace la referencia.

Además de los roles estándar de las vistas básicas definidas por el usuario, esta proporciona varios tipos de vistas que permiten llevar a cabo objetivos especiales en una base de datos:

Vista indizadas: es una vista que se ha materializado. Lo que significa que se ha definido la vista y que los datos resultantes se han almacenado como una tabla. "se pueden indizar una vista creando un índice clúster único en ella"

Estas vistas pueden dar una mejora considerable al rendimiento de algunos tipos de consultas. Estas también funcionan mejor para consultas que agregan muchas filas, pero no

son adecuadas para los conjuntos de datos subyacentes que se actualizan con mucha frecuencia.

vista con particiones: éstas combinan los datos horizontales con particiones de conjuntos de tablas miembros de uno o más servidores, loque hace que los datos parecen como si fueran de una tabla.

vista del sistema: estas exponen los metadatos de catálogo, estas pueden usarse para que devolver información acerca de las instancias de SQL server y objetos definidos en la instancia

Las vistas poseen distintos tipos de tareas comunes como: creación de vista, creación de vista indizadas, modificar vistas, modificar datos mediante una vista, eliminar vista, obtener información acerca de una vista y cambiar nombre de la misma.

En este trabajo realizamos vistas que nos permiten visualizar tablas que por si solas no aportan mucha información, pero que presentando de tal forma obtenemos lo que buscamos mostrar, algunos ejemplos entre tantos que se pueden realizar son:

```
SELECT * FROM VW_REMITO_PRODUCCION
SELECT * FROM VW_NOMINA_EMPLEADO
SELECT * FROM VW_INSUMOS_UTILIZADOS
SELECT * FROM VW_VALES_UTILIZADOS
```

```
/*CONSULTA DE LA NOMINA DE EMPLEADOS*/
CREATE VIEW [dbo].[VW_NOMINA_EMPLEADO]
 AS
SELECT  e.legajo AS 'Legajo',
        CONCAT(e.apellido, ' ', e.nombre) AS 'Nombre y Apellido',
        e.cuil AS 'Cuil',
        e.fecha_ingreso AS 'Fecha de Ingreso',
        te.nombre AS 'Función'

FROM    empleado e
        INNER JOIN tipo_empleado te ON e.id_TipoEmpleado = te.id_TipoEmpleado
WHERE   e.activo = 'SI'
go
```

Imagen 8: Ejemplo de vista

Permisos

SQL Server define 4 conceptos básicos:

- Inicios de sesión (Login): Un login es la habilidad de utilizar una instancia del Servidor SQL, está asociado con un usuario de Windows o con un usuario de SQL. Son autenticados contra SQL Server por lo tanto son los accesos al servidor, pero esto no quiere decir que puedan acceder a las bases de datos o a otros objetos. Para poder acceder a cada una de las bases de datos se necesita de un usuario (user).

- Usuario de la base de datos (User): El usuario de la base de datos es la identidad del inicio de sesión cuando está conectado a una base de datos.

  El usuario de la base de datos puede utilizar el mismo nombre que el inicio de sesión, pero no es necesario.

  - Los Logins son asignados a los usuarios
  - Los grants se les asignan a los usuarios.
  - A los usuarios se le asignan sus propios Esquemas(schemas)

Usuarios por defecto en una BD

dbo: Propietario. No puede ser borrado de la BD

Guest: Permite a usuarios que no tienen cuenta en la BD, que accedan a ella, pero hay que hacerle permiso explícitamente

Information_schema Permite ver los metadatos de SQL Server.

sys Permite consultar las tablas y vistas del sistema, procedimientos extendidos y otros objetos del catálogo del sistema.

Los usuarios pueden pertenecer a Roles.

- Todos los usuarios son miembros del Role “Public”
- El login “sa” está asignado al usuario dbo en todas las bases de datos.

Da acceso a la base de datos, pero esto tampoco quiere decir que pueda hacer cualquier operación sobre la base de datos, en principio no puede hacer casi nada, salvo que se le

vaya asignando roles y otros privilegios para hacerle permisos de acceso a los objetos de esa base de datos.

- Roles: Los Roles pueden existir a nivel de instancia o base de datos.

  A nivel de Instancia:

  - Los logins pueden ser otorgados roles llamados “server roles”.
  - No se pueden crear Roles nuevos

  A nivel de Base de Datos

  - Los usuarios de base de datos pueden ser otorgados roles.
  - Se pueden crear roles nuevos.

  Roles de una Aplicación

  Un rol de aplicación sirve para asignarle permisos a una aplicación:

  - Tiene un password
  - No contiene usuarios

Estos roles existen en todas las bases de datos. A excepción del rol de base de datos public, no se pueden cambiar los permisos asignados a los roles fijos de base de datos:
SysAdmin: es el Administrador del sistema de base de datos, tiene permiso para todos los procesos como eliminar, agregar y modificar tablas, como también la modificación los permisos de demás usuarios. En el proyecto es el administrador de la base de datos quien contara con este rol.
ABM: ABM significa Altas, Bajas y Modificaciones, y se refiere al sistema mediante el cual las aplicaciones de bases de datos se mantienen actualizadas. Los sistemas ABM suelen colocarse en un directorio sin acceso al resto de los usuarios, en general protegido con un nombre de usuario y una contraseña. Este rol está destinado al sector administrativo quien se encargará mayormente de la interacción con la base.

Consulta: son el tipo de usuario que puede ver solamente las tablas que el SysAdmin permite. En este rol entran los niveles más bajos de la organización quienes solo podrán consultar ciertos datos.

Los permisos de los roles de base de datos definidos por el usuario se pueden personalizar con las funciones GRANT, DENY y REVOKE:

GRANT: Concede permisos sobre un elemento protegible a una entidad de seguridad

DENY: Evita que la entidad de seguridad herede permisos por su pertenencia a grupos o roles. DENY tiene prioridad sobre todos los permisos, salvo que DENY no se aplique a los propietarios de objetos o miembros del rol fijo de servidor sysadmin.

REVOKE: Quita un permiso concedido o denegado previamente.

Ejemplo para el rol de administrador:

```
--CREACION  DE INICIO DE SESION ADMINISTRADOR y ROL SYSADMIN--
CREATE LOGIN [Administrador] WITH PASSWORD = '1234',
CHECK_EXPIRATION = OFF,
CHECK_POLICY = OFF;
CREATE USER Gerente FOR LOGIN Administrador;
EXEC sys.sp_addsrvrolemember @loginame = N'Administrador', @rolename = N'sysadmin';
```

Imagen 9: Ejemplo de permiso para usuario con rol de administrador

Otros temas

Migración de una base de datos

Al tratarse de una base de datos que actualmente está funcionando pero que necesitaba una actualización y reestructuración el paso final será migrar todos los datos cargados a lo largo de varios años atrás a esta nueva base de datos. En las siguientes imágenes mostramos cómo se encontraba antes de trabajarla y aplicar todas las técnicas y formas recientemente estudiadas en esta materia.

Hemos aplicado la técnica de normalización ya que la base de datos no contaba con ninguna de las tres formas, una vez normalizado la base de datos pudimos aplicar un lote de datos ficticio para probar su funcionamiento, para en un futuro poder hacer una migración exitosa

de todos los datos. La herramienta utilizada para hacer la migración se llama ESF Database Migration Toolkit, la cual nos permitió obtener las tablas viejas en nuestro motor y listo para el siguiente tratamiento de los datos que requerirá.

Proceso de extracción de datos

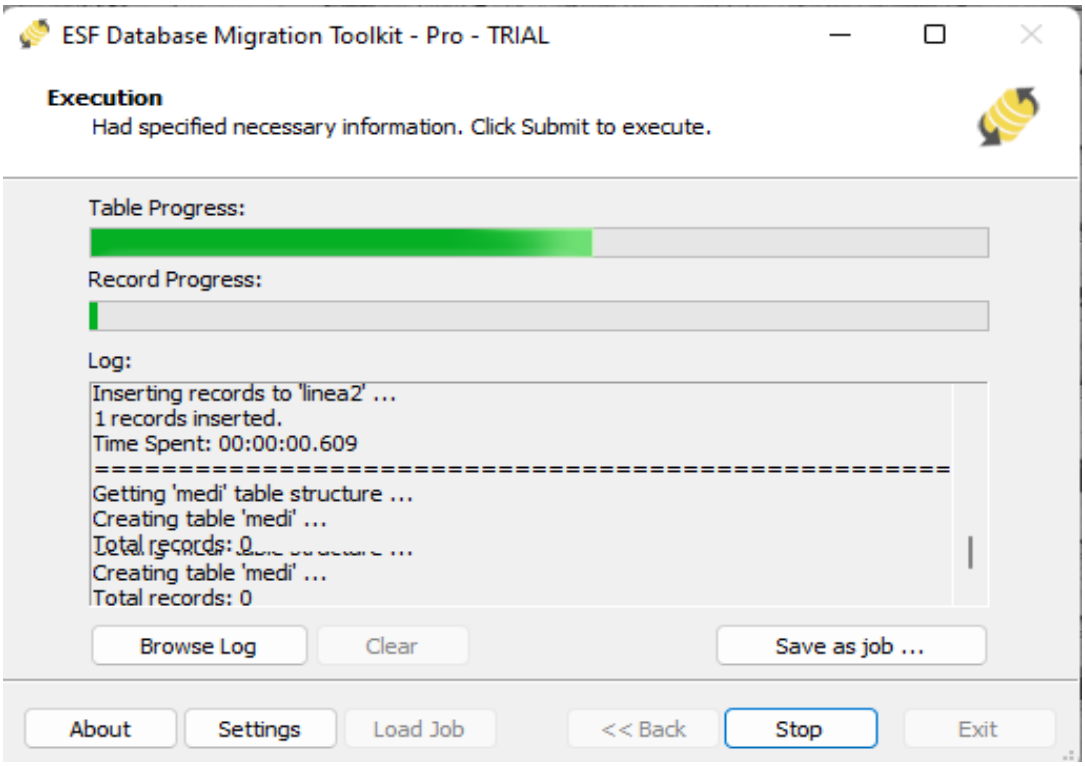

Imagen 10: Proceso de extracción de datos

Las tablas extraídas fueron las siguientes:

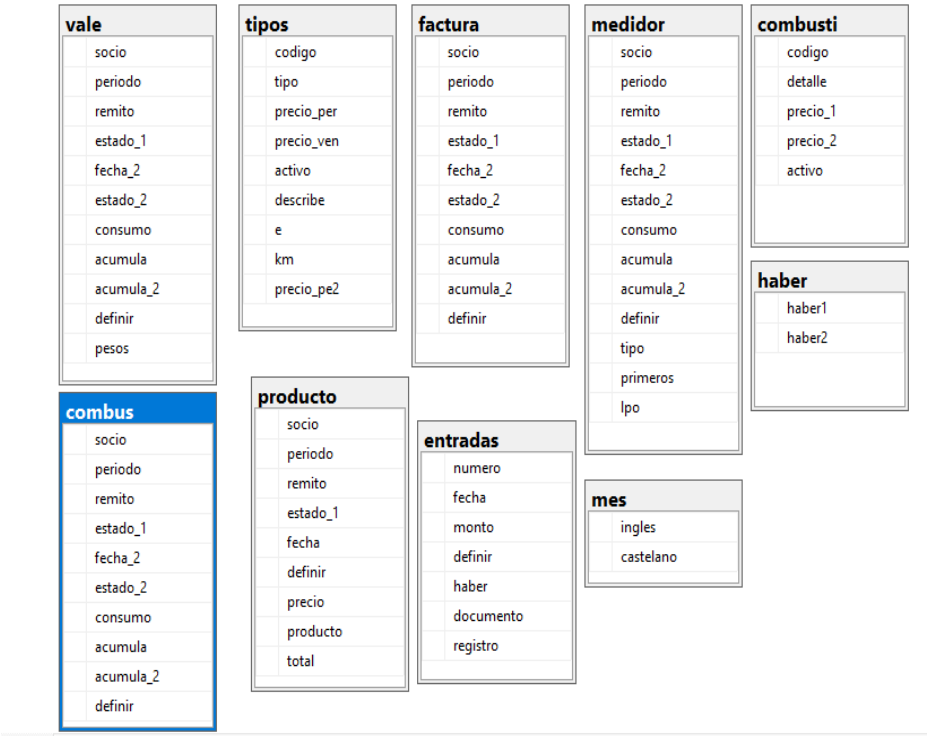

Imagen 11: tablas de la base de datos original

	socio	periodo	remito	estado_1	fecha_2	estado_2	consumo	acumula	acumula_2	definir
10	35	11/1997	26642	0.00	1997-11-19	210.00	0	0	0.00	2
11	35	11/1997	26642	0.00	1997-11-19	2996.00	0	0	0.00	1
12	35	11/1997	26757	0.00	1997-11-26	40.00	0	0	0.00	2
13	35	11/1997	26757	0.00	1997-11-26	2751.00	0	0	0.00	1
14	35	11/1997	26868	0.00	1997-11-30	70.00	0	0	0.00	2
15	35	11/1997	26868	0.00	1997-11-30	1078.00	0	0	0.00	1
16	35	12/1997	27098	0.00	1997-12-10	470.00	0	0	0.00	2
17	35	12/1997	27098	0.00	1997-12-10	5558.00	0	0	0.00	1
18	35	12/1997	27228	0.00	1997-12-17	250.00	0	0	0.00	2
19	35	12/1997	27228	0.00	1997-12-17	2813.00	0	0	0.00	1
20	35	12/1997	27342	0.00	1997-12-24	120.00	0	0	0.00	2
21	35	12/1997	27342	0.00	1997-12-24	1576.00	0	0	0.00	1
22	35	12/1997	27456	0.00	1997-12-31	498.00	0	0	0.00	1
23	35	01/1998	27651	0.00	1998-01-07	460.00	0	0	0.00	2
24	35	01/1998	27651	0.00	1998-01-07	2233.00	0	0	0.00	1
25	35	01/1997	27753	0.00	1998-01-14	290.00	0	0	0.00	2
26	35	01/1997	27753	290.00	1998-01-14	3116.00	0	0	0.00	1

Query executed successfully. DESKTOP-2715

Imagen 12: Ejemplo de tabla de la base original

CAPÍTULO IV: Conclusión

A lo largo del desarrollo del trabajo se tuvo que investigar sobre el proyecto a desarrollar, aclarar los objetivos y restricciones del mismo, emplear diagramas además de herramientas para bosquejar el sistema que queríamos desarrollar. Para ello fue fundamental entender los temas dictados en la materia y poder volcarlos al proyecto. Esos conocimientos ayudaron a desarrollar no solo el aspecto del desarrollo del cuerpo del trabajo en todos sus contenidos sino también todo lo que conlleva a su programación, creación de base de datos, tablas, restricciones, relaciones entre tablas, creación de usuarios y sus permisos, utilización de transacciones, disparadores, visualizaciones de vistas, entre otros.

Como resultado de todo ello, se logró un sistema que atienda a la gestión de base de datos de la empresa forestal donde podrá realizar altas modificaciones y bajas de empleados, insumos, producciones, tipos de cortes que realizan, los remitos de las toneladas entregadas entre otros, logrando así llegar a la consulta principal que es la del calculo del sueldo de un determinado empleado en el mes requerido, para esto se requirió el uso de casi todas las tablas sumando y restando las columnas correspondientes.

Finalmente, creemos que logramos el objetivo propuesto al inicio el de realizar la actualización de la base de datos sobre los empleados forestales de la empresa logrando la normalización de la misma y generar las funcionalidades necesarias para llevar a cabo la gestión adecuada, sabiendo que a futuro y con el uso surgirán modificaciones y/o mejoras que agregar.

Bibliografía

Fortiz, O. L. (2022). *Transacciones en SQL Server*. Obtenido de https://ofortiz.space/olgafortiz/unidad-4-transacciones-en-sql-server/

Microsoft. (2022). *Documentación técnica de SQL Server*. Obtenido de https://learn.microsoft.com/es-es/sql/sql-server/?view=sql-server-ver16

Neoattack. (27 de Agosto de 2022). Obtenido de https://neoattack.com/neowiki/trigger/#:~:text=Consiste%20en%20una%20serie%20de,los%20datos%20de%20una%20tabla.

W3Schools. (2022). *SQL Tutorial*. Obtenido de https://www.w3schools.com/sql/default.asp