



**Facultad de Ciencias Exactas y Naturales Agrimensura**

**Catedra: Base de Datos I**

**Año: 2024**

**Informe:**

**Proyecto Integrador de Bases de Datos I**

Alumnos:

-Almada, Tomas Emanuel

DNI: 44876943

-Gimeno, Daniel Eduardo

DNI:37327946

-Rodriguez, Maria Agustina

DNI: 40565365

-Trangoni, Diego Gerardo

DNI: 34899698

# I – Introducción

En este proyecto se abordara el desarrollo de diversos métodos de gestión de base de datos basado en una escenario en el que se puedan ilustrar las practicas de dichos métodos.

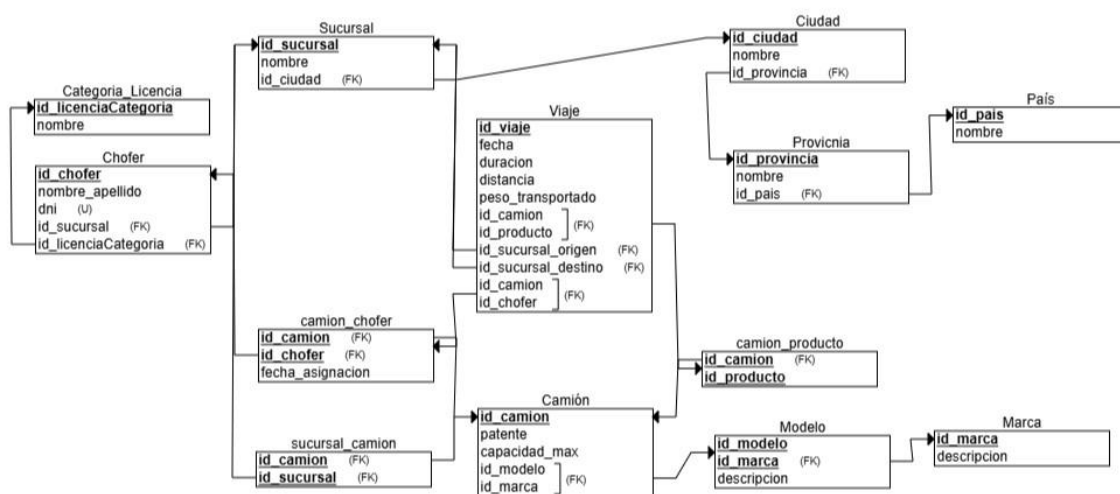
## Caso de Estudio: Empresa Nacional de Transporte

Una empresa que se especializa en transporte que tiene su sede central en Corrientes, Argentina, pero posee múltiples sucursales en los distintos países de América del Sur. Esta gestiona y genera informes de todos los movimientos que realizan sus transportes, desde el chofer, el camión designado al mismo hasta los materiales que transporta, hacia donde y su tiempo de viaje.

En estos informes que realiza la empresa se detallan varios datos personales de los choferes, y también la sucursal a la que pertenecen. De los vehículos se guardan su modelo, marcas y tipo de carga que puede llevar, esto porque normalmente cada camión transporta un solo tipo de material por cada viaje que realiza.

Tras cada viaje, se detallan la cantidad de kilómetros que recorrió el transporte, su tiempo tardado desde un punto a otro y el peso total que estuvo llevando durante su viaje. Además de eso, también se conoce su capacidad máxima de carga y los kilómetros totales recorridos hasta la fecha.

A continuación se mostrara el diagrama relacional de la situación:



## Diccionario de datos:

REPORT		ServerName	DatabaseName	TableName	SchemaName	ColumnName	DataType	MaxLength	IsNull	IsIdentity	Description
1	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	camion	dbo	id_camion	int	4	NO	NO	-- add description here
2	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	camion	dbo	patente	varchar	10	NO	NO	-- add description here
3	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	camion	dbo	capacidad_max	int	4	NO	NO	-- add description here
4	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	camion	dbo	id_modelo	int	4	NO	NO	-- add description here
5	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	camion	dbo	id_marca	int	4	NO	NO	-- add description here
6	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	camion_chofer	dbo	fecha_asignacion	date	3	NO	NO	-- add description here
7	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	camion_chofer	dbo	id_camion	int	4	NO	NO	-- add description here
8	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	camion_chofer	dbo	id_chofer	int	4	NO	NO	-- add description here
9	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	camion_producto	dbo	id_producto	int	4	NO	NO	-- add description here
10	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	camion_producto	dbo	id_camion	int	4	NO	NO	-- add description here
11	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	camion_sucursal	dbo	id_camion	int	4	NO	NO	-- add description here
12	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	camion_sucursal	dbo	id_sucursal	int	4	NO	NO	-- add description here
13	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	categoria_licen...	dbo	id_categoria_lic	int	4	NO	NO	-- add description here
14	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	categoria_licen...	dbo	nombre	varchar	10	NO	NO	-- add description here
15	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	chofer	dbo	id_chofer	int	4	NO	NO	-- add description here
16	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	chofer	dbo	nombre	varchar	100	NO	NO	-- add description here
17	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	chofer	dbo	apellido	varchar	100	NO	NO	-- add description here
18	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	chofer	dbo	dni	int	4	NO	NO	-- add description here
19	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	chofer	dbo	id_sucursal	int	4	NO	NO	-- add description here
20	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	chofer	dbo	id_categoria_lic	int	4	NO	NO	-- add description here
21	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	ciudad	dbo	id_ciudad	int	4	NO	NO	-- add description here
22	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	ciudad	dbo	nombre	varchar	100	NO	NO	-- add description here
23	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	ciudad	dbo	id_provincia	int	4	NO	NO	-- add description here
24	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	marca	dbo	id_marca	int	4	NO	NO	-- add description here
25	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	marca	dbo	descripcion	varchar	200	NO	NO	-- add description here
26	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	modelo	dbo	id_modelo	int	4	NO	NO	-- add description here
27	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	modelo	dbo	descripcion	varchar	100	NO	NO	-- add description here
28	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	modelo	dbo	id_marca	int	4	NO	NO	-- add description here
29	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	pais	dbo	id_pais	int	4	NO	NO	-- add description here
30	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	pais	dbo	nombre	varchar	100	NO	NO	-- add description here
31	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	provincia	dbo	id_provincia	int	4	NO	NO	-- add description here
32	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	provincia	dbo	nombre	varchar	100	NO	NO	-- add description here
33	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	provincia	dbo	id_pais	int	4	NO	NO	-- add description here
34	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	sucursal	dbo	id_sucursal	int	4	NO	NO	-- add description here
35	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	sucursal	dbo	domicilio	varchar	200	NO	NO	-- add description here
36	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	sucursal	dbo	id_ciudad	int	4	NO	NO	-- add description here
37	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	viaje	dbo	fecha	int	4	NO	NO	-- add description here
38	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	viaje	dbo	id_viaje	int	4	NO	NO	-- add description here
39	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	viaje	dbo	duracion	int	4	NO	NO	-- add description here
40	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	viaje	dbo	distancia	int	4	NO	NO	-- add description here
41	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	viaje	dbo	peso_transporta...	int	4	NO	NO	-- add description here
42	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	viaje	dbo	id_producto	int	4	NO	NO	-- add description here
43	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	viaje	dbo	id_camion	int	4	NO	NO	-- add description here
44	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	viaje	dbo	id_sucursal_orig...	int	4	NO	NO	-- add description here
45	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	viaje	dbo	id_sucursal_des...	int	4	NO	NO	-- add description here
46	DATADictionary	DESKTOP-AQMEDK\SQLXPRESS	caso_camiones	viaje	dbo	id_chofer	int	4	NO	NO	-- add description here

Consulta ejecutada correctamente. DESKTOP-AQMEDK\SQLXPRESS ... DESKTOP-AQMEDK\Franke... caso\_camiones 00:00:00 46 filas

## Objetivo de este proyecto

El objetivo de este proyecto es facilitar el manejo de esta base de datos sobre la empresa de transporte, el como facilitar las gestión de los datos de los viajes que camiones todos los días viajan y derivan en muchos informes diarios, que en el propenso caso genere confusión a la hora de comprobar quienes conducían que camiones y que su dichosa carga este dentro de los reglamentos, sin que se transportan objetos no deseados o no mencionados, que a la hora de un camión realice un viaje no tenga accidentes o problemas por cargas no adecuadas o peligrosas para su modelo.

## II – Marco Conceptual O Referencial

Durante la elaboración de este modelo de datos se espera aplicar distintos procedimientos que ayuden a la gestión de volúmenes grandes de datos que generan diariamente la empresa, entre los viajes, la asignación de choferes viejos y nuevos, renovación de los vehículos de transporte y mas.

También se trabajara en la seguridad, sobre quienes tendrán acceso a la base de datos, que son los principales encargados de cargar la información anteriormente recibida. Quienes pueden modificar las tablas, cargar los informes o en caso de errores, quienes podrán abordarlos o solucionarlos. Definir el control de cada parte de la base de datos.

Así mismo, la creación de procedimientos que faciliten la carga de datos monótonos que se saben desde un principio que serán iguales al menos de forma temporal hasta que dicte necesario un cambio. También para realizar otras funciones que podrían llevar mucho mas tiempo del esperado si no se planificaron de antemano.

## III – Metodología Utilizada

La presente investigación de los temas aquí expuestos, fueron llevados a cabo utilizando una metodología de desarrollo (SCRUM).

### **a) Descripción de cómo se realizó el trabajo practico:**

El grupo se dividió las tareas en diferentes partes que realizaran por separado pero contando con el apoyo de los demás en caso de necesidad, una vez cada parte hecha, unificar cada una y verificar que todo funcione de forma correcta.

### **b) Herramientas:**

El grupo trabajo por medio de grupos de Whatsapp, donde se designaron las tareas y nos pusimos en contacto para las consultas de cada uno. Para la organización de los archivos, se utilizo Github donde cada miembro participaría en subir sus tareas para su posterior unificación.

## IV - Desarrollo de Temas

### Manejo de Permisos a Nivel de usuarios de base datos

Este tema consiste crear usuarios para la base de datos a los que puede otorgar distintos roles que definen las funciones a las que tienen acceso o que pueden usar para navegar o trabajar en la base de datos.

SQL Management Studio por ejemplo, ya comienza con un “super administrador” el cual tiene acceso a todas las funciones del programa y desde esta misma es posible crear, organizar y controlar los permisos de los futuros usuarios.

Esto es importante para mantener un orden en la base de datos de una empresa por ejemplo, limitar quienes tienen acceso a que funciones y evitar accidentes o modificaciones no esperadas que podrían llevar a un serio problema de organización y un gasto de tiempo que pudo haberse evitado. Por ejemplo:

```
--Se crean los usuarios a nivel servidor, sin roles asignados.

create login manuel with password='Password123';
create login juan with password='Password123';

--Se crea los usuarios con los login anteriores para una base de datos ESPECIFICA

-USE caso_camiones; ----- <-Verificar en que base de datos crear estos usuarios.
GO

] CREATE USER manuel FOR LOGIN manuel;
CREATE USER juan FOR LOGIN juan;
```

```

]--- Se asignan distintos permisos a los usuarios creados para esta base de datos (caso_camiones)

--- Por ejemplo, manuel solo puede leer las tablas, es decir esta restringido a usar solo la sentencia SELECT
-EXEC sp_addrolemember 'db_datareader', 'manuel';
go

--Probamos si los permisos funcionan.
]EXECUTE AS USER = 'manuel';

SELECT * FROM pais --Debe ver las tablas.
INSERT INTO pais(id_pais, nombre) VALUES (4, 'Venezuela') ---No deberia poder insertar.

]REVERT ---Regresas al estado inicial, sin el user manuel activo.

---- Para el usuario Juan, se le otorga permisos de escritura, es decir, puedo usar unicamente sentencias INSERT
EXEC sp_addrolemember 'db_datawriter', 'juan';

--- Probamos...
EXECUTE AS USER = 'juan';
SELECT * FROM pais --No deberia poder ver las tablas.
INSERT INTO pais(id_pais, nombre) VALUES (4, 'Venezuela') ---Deberia poder insertar.

REVERT

```

Otros tipos de roles que existen (Conocidos como roles fijos):

- **db\_owner:** Los miembros del rol fijo de base de datos db\_owner pueden realizar todas las actividades de configuración y mantenimiento en la base de datos, y también pueden drop la base de datos en SQL Server. (En SQL Database y Azure Synapse, algunas actividades de mantenimiento requieren permisos a nivel de servidor y no se pueden realizar por db\_owners).
- **db\_securityadmin:** Los miembros del rol fijo de base de datos db\_securityadmin pueden modificar la pertenencia a roles únicamente para roles personalizados y administrar permisos. Los miembros de este rol pueden elevar potencialmente sus privilegios y se deben supervisar sus acciones.
- **db\_accessadmin:** Los miembros del rol fijo de base de datos db\_accessadmin pueden agregar o eliminar el acceso a la base de datos para inicios de sesión de Windows, grupos de Windows e inicios de sesión de SQL Server.
- **db\_backupoperator:** Los miembros del rol fijo de base de datos db\_backupoperator pueden crear copias de seguridad de la base de datos.
- **db\_ddladmin:** Los miembros del rol fijo de base de datos db\_ddladmin pueden ejecutar cualquier comando del lenguaje de definición de datos (DDL) en una base de datos. Los miembros de este rol pueden potencialmente aumentar sus privilegios manipulando código que puede ser ejecutado bajo altos privilegios y sus acciones se deben supervisar.

- **db\_datawriter:** Los miembros del rol fijo de base de datos db\_datawriter pueden agregar, eliminar o cambiar datos en todas las tablas de usuario. En la mayoría de los casos de uso, este rol se combinará con la membresía db\_datareader para permitir la lectura de los datos que se van a modificar.
- **db\_datareader:** Los miembros del rol fijo de base de datos db\_datareader pueden leer todos los datos de todas las tablas y vistas de usuario. Los objetos de usuario pueden existir en cualquier esquema, excepto sys e INFORMATION\_SCHEMA.
- **db\_denydatawriter:** Los miembros del rol fijo de base de datos db\_denydatawriter no pueden agregar, modificar ni eliminar datos de tablas de usuario de una base de datos.
- **db\_denydatareader:** Los miembros del rol fijo de base de datos db\_denydatareader no pueden leer datos de las tablas y vistas de usuario dentro de una base de datos.

#### Luego tenemos un par de roles mas especiales:

Estos roles y permisos se aplican principalmente a la base de datos "master" y suelen estar relacionados con la administración y configuración del servidor.

- **dbmanager:** Puede crear y eliminar bases de datos, convirtiéndose en el propietario. Tiene todos los permisos en las bases de datos que crea, pero no en otras.
- **db\_exporter:** Aplicable a grupos de SQL dedicados de Azure Synapse Analytics. Permite actividades de exportación de datos con permisos como CREATE TABLE, ALTER ANY SCHEMA, ALTER ANY EXTERNAL DATA SOURCE, y ALTER ANY EXTERNAL FILE FORMAT.
- **loginmanager:** Puede crear y eliminar inicios de sesión en la base de datos "master" virtual.

## **Procedimientos y Funciones Almacenadas en Bases de Datos**

### **1. Definición**

#### **1.1 Procedimientos Almacenados**

En términos generales en bases de datos, un procedimiento almacenado es un conjunto de instrucciones SQL que se almacena asociado a una base de datos. Un procedimiento puede tener cero o muchos parámetros de entrada y cero o muchos parámetros de salida. Se pueden utilizar para validar datos, controlar el acceso o reducir el tráfico de red.

En SQL Server es un grupo de una o varias instrucciones Transact-SQL o una referencia a un método de Common Runtime Language (CLR) de Microsoft .NET Framework. Los procedimientos se asemejan a las construcciones de otros lenguajes de programación, porque pueden:

- Aceptar parámetros de entrada y devolver varios valores en forma de parámetros de salida al programa que realiza la llamada.
- Contener instrucciones de programación que realicen operaciones en la base de datos. Entre otras, pueden contener llamadas a otros procedimientos.
- Devolver un valor de estado a un programa que realiza una llamada para indicar si la operación se ha realizado correctamente o se han producido errores, y el motivo de estos.

## 1.2 Funciones Almacenadas

En términos generales en bases de datos, una función almacenada es un conjunto de instrucciones SQL que se almacena asociado a una base de datos. Es un objeto que se crea con la sentencia CREATE FUNCTION y se invoca con la sentencia SELECT o dentro de una expresión. Una función puede tener cero o muchos parámetros de entrada y siempre devuelve un valor, asociado al nombre de la función. Son usadas principalmente para realizar cálculos o transformaciones, se limitan más que los procedimientos porque suelen estar orientadas a consultas de datos y no a la modificación de los mismos.

## 2. Características de Procedimientos y Funciones Almacenadas

Característica	Procedimiento Almacenado	Función Almacenada
Retorno de valor	Puede devolver múltiples valores mediante parámetros de salida o un conjunto de resultados.	Devuelve un solo valor (excepto en bases que soportan tablas como retorno).
Uso de SQL DML	Puede usar instrucciones INSERT, UPDATE, DELETE, etc.	Principalmente SELECT; algunos sistemas no permiten INSERT o DELETE.
Llamada desde SQL	Se invoca mediante CALL o EXECUTE.	Se invoca en consultas SQL normales, como en un SELECT o en cláusulas WHERE.
Emisión de transacciones	Puede iniciar, confirmar, o revertir transacciones.	No puede gestionar transacciones.
Uso como funciones escalares	No se puede usar en cláusulas SELECT.	Se puede usar en cláusulas SELECT y WHERE.

## 3. Ventajas de usar procedimientos almacenados

- **Mejor rendimiento:** Un procedimiento se compila la primera vez que se ejecuta, permitiendo que en posteriores ejecuciones pueda usarse. Esto permite que el procesador necesite menos tiempo para procesar el procedimiento.



- **Tráfico de red reducido entre el cliente y el servidor:** Los comandos de un procedimiento se ejecutan en un único lote de código, lo que puede reducir significativamente el tráfico de red entre el servidor y el cliente ya que únicamente se envía a través de la red la llamada que va a ejecutar el procedimiento. Sin la encapsulación de código que proporciona un procedimiento, cada una de las líneas de código tendría que enviarse a través de la red.
- **Reutilización de código:** Permiten centralizar la lógica de negocio, evitando duplicación de código, reducen las inconsistencias de código y se permite que cualquier usuario o aplicación que cuente con los permisos necesarios pueda acceder al código y ejecutarlo.
- **Seguridad y control:** Pueden restringir el acceso a los datos mediante permisos, ya que los usuarios no necesitan acceso directo a las tablas para ejecutar procedimientos o funciones.  
En SQL Server varios usuarios y programas cliente pueden realizar operaciones en los objetos de base de datos subyacentes a través de un procedimiento, aunque los usuarios y los programas no tengan permisos directos sobre esos objetos subyacentes. El procedimiento controla qué procesos y actividades se llevan a cabo y protege los objetos de base de datos subyacentes. Esto elimina la necesidad de conceder permisos en cada nivel de objetos y simplifica los niveles de seguridad.
- **Mantenimiento y modularidad:** Si los procedimientos y funciones se mantienen en la capa de datos, solo deben actualizarse los cambios de los procesos en la base de datos subyacente. A nivel aplicación no se tiene conocimiento de los cambios que se realizan en los diseños, en las relaciones o los procesos de la base de datos

#### 4. Tipos de Parámetros en Procedimientos Almacenados

Los procedimientos almacenados pueden aceptar varios tipos de parámetros, que permiten controlar el flujo de datos entre el procedimiento y la aplicación o los llamadores:

- **IN:** Reciben datos desde la aplicación hacia el procedimiento.
- **OUT:** Envían datos desde el procedimiento hacia la aplicación.
- **INOUT:** Funcionan en ambas direcciones; el valor se envía al procedimiento y se actualiza al llamador al finalizar.

#### 5. Tipos de Procedimientos Almacenados

- **Definidos por el Usuario:** Este procedimiento se crea en una base de datos definida. En SQL Server se pueden crear en todas las bases de datos del sistema, excepto en la base de datos **Resource**. El procedimiento se puede crear en Transact-SQL o como referencia a un método Common Language Routine (CLR) de Microsoft .NET Framework.
- **Sistema:** Estos procedimientos se incluyen con el motor de la base de datos. Se encuentran almacenados en la base de datos interna de un sistema. En SQL Server se encuentran en Resources y se muestran de forma lógica en el esquema sys de cada base de datos definida por el sistema y por el usuario.

- **Temporales:** Son una forma de procedimientos definidos por el usuario y permanentes. Existen dos tipos:
  - **Locales:** tienen como primer carácter de sus nombres un solo signo numeral (#). Solo son visibles en la conexión actual de usuario y se eliminan cuando se cierra la conexión.
  - **Globales:** estos presentan dos signos numeral (##) antes del nombre, son visibles para cualquier usuario después de su creación y se eliminan al final de la última sesión en la que se usa el procedimiento,

## Optimización de consultas a través de índices

La optimización de consultas es esencial en bases de datos con grandes volúmenes de datos para mejorar los tiempos de respuesta y eficiencia general del sistema. Los índices permiten al motor de base de datos acceder más rápidamente a los datos, evitando búsquedas completas en las tablas. Este informe explora cómo los índices afectan las consultas en términos de rendimiento, especialmente cuando se trabaja con tablas de gran tamaño.

### Desarrollo:

Para evaluar el impacto de los índices en las consultas, se realizaron varias pruebas sobre una tabla de base de datos con un millón de registros y una columna de tipo fecha. A través de consultas que filtran datos por rango de fechas, se midieron los tiempos de respuesta y los planes de ejecución en diferentes escenarios: sin índices, con un índice agrupado en la columna de fecha y con un índice agrupado que incluye varias columnas.

1. **Prueba sin índices:** Inicialmente, se ejecutó una consulta de búsqueda por periodo sin ningún índice aplicado. Esta búsqueda exigió una lectura completa de la tabla (full scan), resultando en tiempos de respuesta prolongados. Resultado: CPU time = 4292 ms, elapsed time = 5688 ms. (ver imágenes *exec\_plan\_1*)
2. **Índice agrupado en columna de fecha:** Al aplicar un índice agrupado sobre la columna de fecha, se observó una mejora significativa en el tiempo de respuesta. El motor de base de datos pudo acceder a los datos más eficientemente al filtrar directamente sobre el índice sin recorrer la tabla entera. Resultado: CPU time = 893 ms, elapsed time = 4697 ms. (ver imágenes *exec\_plan\_2*)
3. **Índice agrupado en fecha e inclusión de columnas:** Posteriormente, se definió un índice agrupado en la columna de fecha, incluyendo además las columnas seleccionadas en la consulta. Este enfoque permitió al motor de base de datos satisfacer la consulta directamente desde el índice, reduciendo aún más los tiempos de acceso. Resultado: CPU time = 530 ms, elapsed time = 737 ms. (ver imágenes *exec\_plan\_3*)

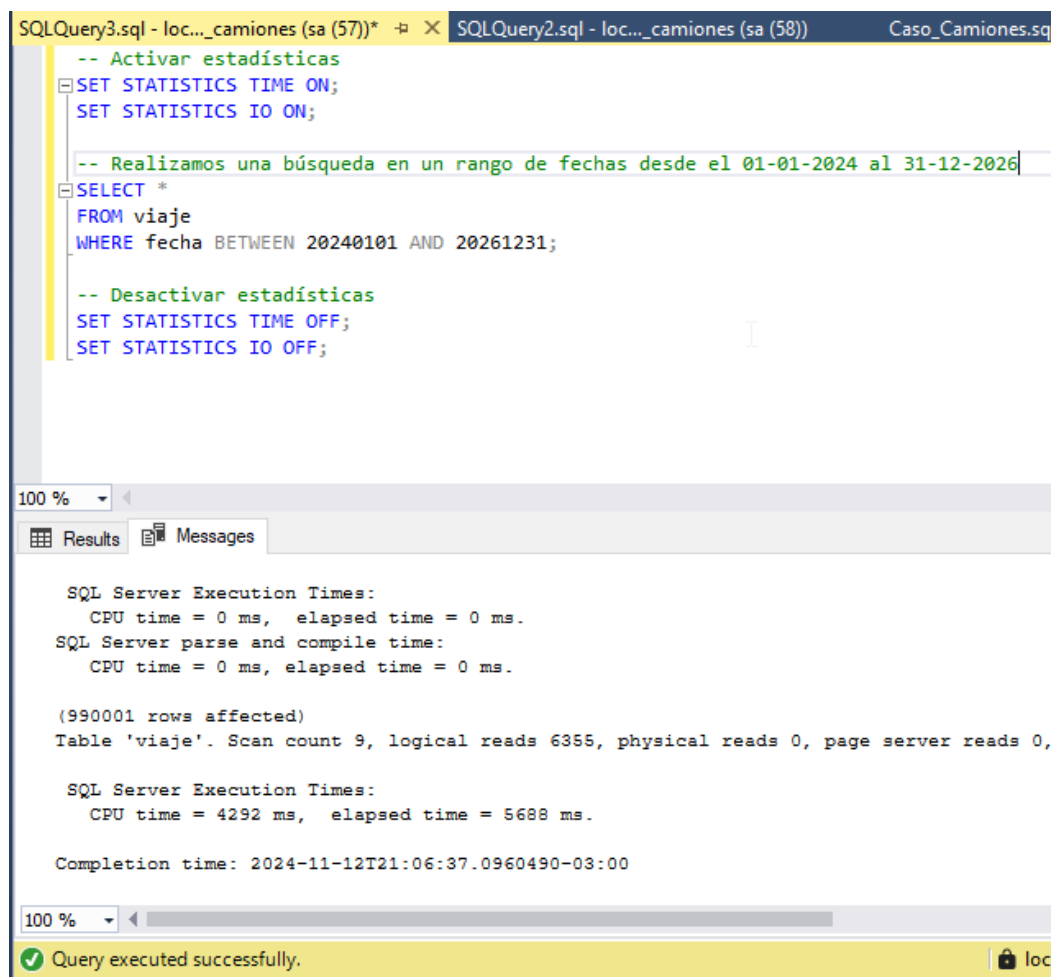
### Análisis de los Planes de Ejecución:

Los planes de ejecución mostraron una notable disminución en el costo de las consultas conforme se aplicaban índices. En el caso de las pruebas con el índice agrupado, el acceso a los datos fue directo, sin necesidad de realizar operaciones de lectura adicionales. Al incluir columnas en el índice, el plan de ejecución evidenció aún menos operaciones de E/S, logrando una consulta más rápida y eficiente.

### Conclusiones:

El uso de índices agrupados en consultas con filtrado por fechas resultó en una mejora notable en el rendimiento. Adicionalmente, incluir las columnas seleccionadas dentro del índice agrupado optimizó aún más la consulta, eliminando lecturas innecesarias. Por lo tanto, en entornos con gran volumen de datos y consultas frecuentes por periodos, es recomendable utilizar índices específicos que incluyan tanto la columna de filtrado como las columnas de selección.

Este análisis demuestra que los índices bien diseñados pueden ser un recurso poderoso para mejorar el rendimiento de consultas, pero su implementación debe ser evaluada cuidadosamente para evitar efectos adversos en otras operaciones.



The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a SQL query in a text editor. The query is as follows:

```
-- Activar estadísticas
SET STATISTICS TIME ON;
SET STATISTICS IO ON;

-- Realizamos una búsqueda en un rango de fechas desde el 01-01-2024 al 31-12-2026
SELECT *
FROM viaje
WHERE fecha BETWEEN 20240101 AND 20261231;

-- Desactivar estadísticas
SET STATISTICS TIME OFF;
SET STATISTICS IO OFF;
```

The bottom pane shows the execution plan and results. The execution plan is a simple table scan. The results pane shows the following output:

```
SQL Server Execution Times:
    CPU time = 0 ms,  elapsed time = 0 ms.
SQL Server parse and compile time:
    CPU time = 0 ms,  elapsed time = 0 ms.

(990001 rows affected)
Table 'viaje'. Scan count 9, logical reads 6355, physical reads 0, page server reads 0,

SQL Server Execution Times:
    CPU time = 4292 ms,  elapsed time = 5688 ms.

Completion time: 2024-11-12T21:06:37.0960490-03:00
```

The status bar at the bottom indicates that the query was executed successfully.

SQLQuery3.sql - loc...camiones (sa (57)) \* X SQLQuery2.sql - loc...camiones (sa (58)) Indices(Ejemplo).sq...camiones (sa (53))

```
-- Activar estadísticas
SET STATISTICS TIME ON;
SET STATISTICS IO ON;

-- Realizamos una búsqueda en un rango de fechas desde el 01-01-2024 al 31-12-2026
SELECT *
FROM viaje
WHERE fecha BETWEEN 20240101 AND 20261231;

-- Desactivar estadísticas
SET STATISTICS TIME OFF;
SET STATISTICS IO OFF;

-- Eliminar el índice de clave primaria que la tabla ya tenía
ALTER TABLE viaje DROP CONSTRAINT PK_viaje_nueva;
```

100 %

Results Messages Execution plan

Query 1: Query cost (relative to the batch): 100%

SELECT \* FROM [viaje] WHERE [fecha]>=@1 AND [fecha]<=@2

Missing Index (Impact 73.5852): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[viaje] ([fech...

Table Scan  
[viaje]  
Cost: 100 %  
0.753s  
990001 of  
990001 (100%)

Activar Windows

Query executed successfully. localhost (16.0 RTM) sa (57) caso\_camiones 00:00:08 | 990,001 rows

*exec\_plan\_1*

SQLQuery3.sql - loc...\_camiones (sa (57))\* X SQLQuery2.sql - loc...\_camiones (sa (58)) Caso\_Cami

```
-- Eliminar el índice de clave primaria que la tabla ya tenía
ALTER TABLE viaje DROP CONSTRAINT PK_viaje_nueva;

-- Crear el índice agrupado en la columna fecha
CREATE CLUSTERED INDEX idx_fecha ON viaje(fecha);

-- Repetir la búsqueda por periodo
SET STATISTICS TIME ON;
SET STATISTICS IO ON;

SELECT *
FROM viaje
WHERE fecha BETWEEN 20240101 AND 20261231;

SET STATISTICS TIME OFF;
SET STATISTICS IO OFF;
```

100 %

Results Messages

CPU time = 0 ms, elapsed time = 0 ms.  
 SQL Server parse and compile time:  
 CPU time = 15 ms, elapsed time = 17 ms.  
 SQL Server parse and compile time:  
 CPU time = 0 ms, elapsed time = 0 ms.

(990001 rows affected)  
 Table 'viaje'. Scan count 1, logical reads 7152, physical reads 0, page server re.

SQL Server Execution Times:  
 CPU time = 853 ms, elapsed time = 4697 ms.

Completion time: 2024-11-12T21:23:49.0508531-03:00

100 %

Query executed successfully.

SQLQuery3.sql - loc...\_camiones (sa (57))\* X SQLQuery2.sql - loc...\_camiones (sa (58)) Indices(Eje

```
-- Eliminar el índice de clave primaria que la tabla ya tenía
ALTER TABLE viaje DROP CONSTRAINT PK_viaje_nueva;

-- Crear el índice agrupado en la columna fecha
CREATE CLUSTERED INDEX idx_fecha ON viaje(fecha);

-- Repetir la búsqueda por periodo
SET STATISTICS TIME ON;
SET STATISTICS IO ON;

SELECT *
FROM viaje
WHERE fecha BETWEEN 20240101 AND 20261231;

SET STATISTICS TIME OFF;
SET STATISTICS IO OFF;
```

100 %

Results Messages Execution plan

Query 1: Query cost (relative to the batch): 100%  
 SELECT \* FROM [viaje] WHERE [fecha]>=@1 AND [fecha]<=@2

Clustered Index Seek (Clustered)  
 [viaje].[idx\_fecha]  
 Cost: 100 %  
 0.660s  
 990001 of  
 990001 (100%)

SELECT  
 Cost: 0 %

Query executed successfully.

*Exec\_plan\_2*



# Triggers

## Definición:

Un trigger o disparador es un tipo especial de procedimiento almacenado que se ejecuta automáticamente cuando se produce un evento en el servidor de bases de datos. Los desencadenadores DML se ejecutan cuando un usuario intenta modificar datos mediante un evento de lenguaje de manipulación de datos (DML). Los eventos DML son instrucciones INSERT, UPDATE o DELETE de una tabla o vista. Estos desencadenadores se activan cuando se desencadena cualquier evento válido, con independencia de que las filas de la tabla se vean o no afectadas.

## Tipos de triggers:

Los triggers o disparadores en SQL se pueden clasificar en varios tipos, entre ellos:

**Activadores DML:** Son los más comunes y se activan cuando se modifica un dato, como al insertar, actualizar o eliminar una tabla o vista. Algunos ejemplos de activadores DML son AFTER, BEFORE, e INSTEAD OF.

**Activadores DDL:** Se activan cuando se modifica la estructura de la base de datos, como al crear, modificar o eliminar una tabla. También se activan en eventos relacionados con el servidor, como cambios de seguridad o actualización de eventos estadísticos.

**Activadores de inicio de sesión:** Son otro tipo de activador en SQL Server.

**Row Triggers:** Se ejecutan cada vez que se llama al disparador desde la tabla asociada.

**Statement Triggers:** Se ejecutan una sola vez, sin importar la cantidad de veces que se cumpla con la condición.

Practica de Triggers.

Se adjunta la tabla que se creo para la auditoria de la tabla inmueble.

```
CREATE TABLE AuditLogInmueble (  
    AuditID INT IDENTITY PRIMARY KEY,  
    TableName NVARCHAR(100),  
    Operation NVARCHAR(10),  
    PrimaryKeyValue INT,  
    nro_piso NVARCHAR(MAX),  
    dpto NVARCHAR(MAX),  
    sup_Cubierta NVARCHAR(MAX),  
    frente NVARCHAR(MAX),  
    balcon NVARCHAR(MAX),  
    idprovincia NVARCHAR(MAX),  
    idlocalidad NVARCHAR(MAX),  
    idconsorcio NVARCHAR(MAX),  
    ModifiedBy NVARCHAR(100),  
    ModifiedDate DATETIME  
);
```

La tabla anteriormente creada permitirá obtener todos los valores antes de su modificación incluyendo el usuario que realizo la misma y la fecha hora del momento.



Ahora se muestran los triggers creado para update y delete

UPDATE:

```
CREATE TRIGGER trg_Audit_Update_Inmueble
ON dbo.inmueble
AFTER UPDATE
AS
BEGIN
    DECLARE @userName NVARCHAR(100) = SYSTEM_USER;

    INSERT INTO AuditLogInmueble (
        TableName, Operation, PrimaryKeyValue, nro_piso, dpto, sup_Cubierta,
        frente, balcon, idprovincia, idlocalidad, idconsorcio, ModifiedBy, ModifiedDate
    )
    SELECT
        'inmueble' AS TableName,
        'UPDATE' AS Operation,
        d.idinmueble AS PrimaryKeyValue,
        CONVERT(NVARCHAR(MAX), d.nro_piso),
        CONVERT(NVARCHAR(MAX), d.dpto),
        CONVERT(NVARCHAR(MAX), d.sup_Cubierta),
        CONVERT(NVARCHAR(MAX), d.frente),
        CONVERT(NVARCHAR(MAX), d.balcon),
        CONVERT(NVARCHAR(MAX), d.idprovincia),
        CONVERT(NVARCHAR(MAX), d.idlocalidad),
        CONVERT(NVARCHAR(MAX), d.idconsorcio),
        @userName AS ModifiedBy,
        GETDATE() AS ModifiedDate
    FROM
        deleted d;
END;
GO
```

Explicacion en la imagen anterior se puede ver que una de las primeras sentencias se usa para obtener el usuario del sistema mediante SYSTEM\_USER

El insert into es para guardar en la tabla de auditoria el registro que esta siendo modificado, con algunos campos extra para la auditoria.

El select que continua vemos que obtiene todos los valores de inmueble, pero de una tabla deleted, la tabla deleted es donde se almacenaran temporalmente los valores antes de ser actualizados, por eso se lee de ahí.

## Trigger DELETE

```
CREATE TRIGGER trg_Audit_Delete_Inmueble
ON dbo.inmueble
AFTER DELETE
AS
BEGIN
    DECLARE @userName NVARCHAR(100) = SYSTEM_USER;

    INSERT INTO AuditLogInmueble (
        TableName, Operation, PrimaryKeyValue, nro_piso, dpto, sup_Cubierta,
        frente, balcon, idprovincia, idlocalidad, idconsorcio, ModifiedBy, ModifiedDate
    )
    SELECT
        'inmueble' AS TableName,
        'DELETE' AS Operation,
        d.idinmueble AS PrimaryKeyValue,
        CONVERT(NVARCHAR(MAX), d.nro_piso),
        CONVERT(NVARCHAR(MAX), d.dpto),
        CONVERT(NVARCHAR(MAX), d.sup_Cubierta),
        CONVERT(NVARCHAR(MAX), d.frente),
        CONVERT(NVARCHAR(MAX), d.balcon),
        CONVERT(NVARCHAR(MAX), d.idprovincia),
        CONVERT(NVARCHAR(MAX), d.idlocalidad),
        CONVERT(NVARCHAR(MAX), d.idconsorcio),
        @userName AS ModifiedBy,
        GETDATE() AS ModifiedDate
    FROM
        deleted d;
END;
GO
```

Este trigger para la eliminación es bastante similar al anterior, con la diferencia que la tercer línea indica que se ejecutara después de un delete sobre algún registro.

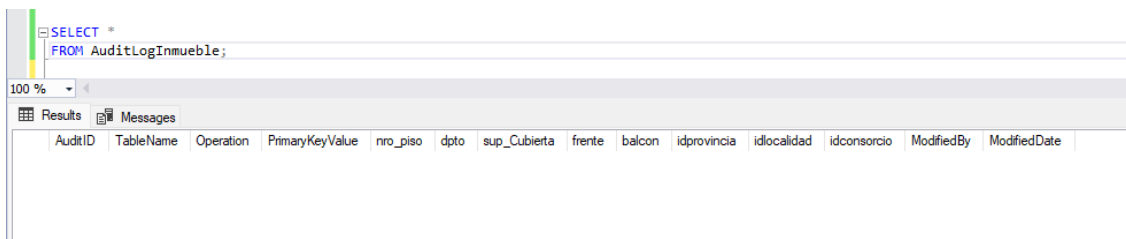
Con estos triggers se hicieron pruebas, donde se muestra el valor del registro con id = 1 antes de un update.

```
SELECT *
FROM dbo.inmueble i
WHERE i.idinmueble = 1;
```

100 %

	idInmueble	nro_piso	dpto	sup_Cubierta	frente	balcon	idprovincia	idlocalidad	idconsorcio
1	1	0	A	85.00	0	0	1	1	1

Tabla de auditoria antes del update.



```
SELECT *
FROM AuditLogInmueble;
```

AuditID	TableName	Operation	PrimaryKeyValue	nro_piso	dpto	sup_Cubierta	frente	balcon	idprovincia	idlocalidad	idconsorcio	ModifiedBy	ModifiedDate
---------	-----------	-----------	-----------------	----------	------	--------------	--------	--------	-------------	-------------	-------------	------------	--------------

Luego de ejecutar un update al registro con id = 1.



```
update dbo.inmueble
SET sup_Cubierta = 110
WHERE idinmueble = 1;
```

(1 row affected)

(1 row affected)

Completion time: 2024-11-12T23:22:03.1438523-03:00

Se observa que la pantalla muestra 2 veces , que 1 fila fue afectada, esto indica que uno fue sobre el registro en cuestión y la otra fila afectada es la que inserto el trigger en la tabla de auditoria.

Tabla de auditoria luego de ejecutar el update.

SELECT \*

FROM

AuditLogInmueble;

100 %

Results

Messages

	AuditID	TableName	Operation	PrimaryKeyValue	nro_piso	dpto	sup_Cubierta	frente	balcon	idprovincia	idlocalidad	idconsorcio	ModifiedBy	ModifiedDate
1	1	inmueble	UPDATE	1	0	A	85.00	0	0	1	1	1	sa	2024-11-13 02:22:03.123

Como prueba adicional luego de realizar el update, se elimino el registro, donde se puede ver que el mismo también indica todos los valores que tenia al momento de ser borrado.



The screenshot shows a SQL query editor with the following query:

```
SELECT *  
FROM AuditLogInmueble;
```

Below the query, the results are displayed in a table with 14 columns: AuditID, TableName, Operation, PrimaryKeyValue, nro\_piso, dpto, sup\_Cubierta, frente, balcon, idprovincia, idlocalidad, idconsorcio, ModifiedBy, and ModifiedDate. The table contains two rows of data.

AuditID	TableName	Operation	PrimaryKeyValue	nro_piso	dpto	sup_Cubierta	frente	balcon	idprovincia	idlocalidad	idconsorcio	ModifiedBy	ModifiedDate
1	inmueble	UPDATE	1	0	A	85.00	0	0	1	1	1	sa	2024-11-13 02:22:03.123
2	inmueble	DELETE	1	0	A	110.00	0	0	1	1	1	sa	2024-11-13 02:24:52.640

## Limitaciones

Si bien los triggers son herramientas poderosas, deben usarse con cuidado:

- **Recursividad:** Si un trigger modifica una tabla que a su vez activa otro trigger, podría entrar en un ciclo infinito.
- **Desempeño:** Los triggers pueden afectar negativamente el rendimiento si no se implementan correctamente, especialmente en bases de datos con un alto volumen de transacciones.
- **Debugging y Mantenimiento:** A diferencia de los procedimientos almacenados estándar o el código en un lenguaje de programación como Python o Java, el código dentro de un trigger puede ser más difícil de depurar y mantener.

## Conclusion

Los triggers son una muy buena forma para obtener la auditoria de la base de datos, de realizar comprobación lógicas de negocio antes de realizar alguna operación (INSERT, UPDATED, DELETE) pero utilizadas de forma incorrecta pueden provocar mas problemas que no se esperaban, siendo algunos de los problemas mencionados anteriormente en la parte de limitaciones.

## **V – Conclusión**

Terminando este informe, hemos visto, aprendido y manipulado diferentes métodos u herramientas que permiten un manejo mas optimo de la bases de datos de varias perspectivas, que no dudamos que serán de utilidad y serán aplicadas en futuros proyectos o en la propia vida laboral manejando en una gran base de datos que requeriría de todos estos métodos para asegurar la integridad y proporcionar respuestas rápidas a nuestra empresa