UNIVERSIDAD NACIONAL
DEL NORDESTE

Base de Datos I

Nombre del proyecto: “Estación de Ómnibus”

Año: 2022.

Grupo Numero 14.

INTEGRANTES:

Méndez, Santiago Alfredo.

Olivos Battestin, Santiago Nicolas.

Parodi, Lucas Ivan.

Pelegrin, Karen Ivana.

Piñeyro, Juan Martin.

INDICE

CAPITULO I:

Introducción:

En este proyecto vamos a tratar la venta de pasajes de colectivos de corta, media y larga distancia que se da en una terminal de ómnibus ubicado en la ciudad de Corrientes Capital.

La idea principal es unificar las boleterías y dejar de lado la venta individual de las distintas empresas, para que de este modo se puedan alivianar los costos referentes a empleados comerciales, internet, luz y alquiler de las distintas empresas asociadas, también ayudara a que esté mejor organizada y se pueda brindar mayor información a los clientes con respecto a los distintos horarios, precios y servicios que existen, ya que al haber tantos locales no se contemplan todas las opciones de viaje que se posee.

También dará espacio para que en un futuro se pueda incorporar un servicio de venta a través de una aplicación web ya que actualmente no se cuenta con una.

Para esto realizaremos una base de dato en Microsoft SQL Server y gestionado con Management Studio.

El principal problema sería generar un pasaje que contenga información acerca del cliente que lo compra, el vendedor, los datos de la empresa seleccionada para llevar a cabo el servicio de transporte, el destino y fecha de salida.

Se crearán perfiles para los empleados con sus respectivas contraseñas y rangos para llevar un mejor control del sistema, también usaremos distintas técnicas de negocio y seguridad como disparadores, vistas, transacciones y demás.

Alcance:

Este sistema de ventas funcionara a través de una aplicación de escritorio de forma local y permitirá que un cliente pueda comprar un pasaje de colectivo de manera presencial donde se tomaran los datos del mismo y se ofrecerá distintas empresas para llevar a cabo el servicio de transporte dependiendo del destino elegido.

Usaremos distintas técnicas de negocio como así también disparadores, transacciones, vistas, etc. que darán un mejor aspecto y desarrollo del sistema.

Esta aplicación solucionara distintos problemas como ser:

- Vender pasajes de manera fácil y ordenada.
- Facilitar la información sobre distintas empresas y destinos.
- Mejorar la contabilidad total de pasajes vendidos.
- Tener datos sobre la cantidad de personas que viajan, (esto dará una mejor estadística para la información turística).
- Los destinos más concurrentes.
- Mejor contabilidad de ganancias, etc.

CAPITULO II:

FUNCIONES

Las funciones integradas de SQL Server son deterministas o no deterministas. Las funciones son deterministas cuando devuelven siempre el mismo resultado cada vez que se llaman con un conjunto específico de valores de entrada. Las funciones son no deterministas cuando es posible que devuelvan distintos resultados cada vez que se llaman con un mismo conjunto específico de valores de entrada.

A la hora de escribir un script complejo para SQL que realiza una determinada tarea, se puede crear una función para que se encargue de ello. Así, en caso de volver a necesitar nuevamente esa tarea, no habrá que volver a crear el script, siendo suficiente volver a llamar a esa función proporcionándole los parámetros necesarios.

PROCEDIMIENTOS ALMACENADOS

Los procedimientos almacenados de SQL pueden ser instrucciones de tipo Transact-SQL o referencias a un método de Common Runtime Language (CLR) de .NET. Dichas instrucciones se encuentran almacenadas de forma física con un nombre dentro de la base de datos.

Ventajas:

Tráfico de red reducido entre el cliente y el servidor:

Los comandos de un procedimiento se ejecutan en un único lote de código. Esto puede reducir significativamente el tráfico de red entre el servidor y el cliente porque únicamente se envía a través de la red la llamada que va a ejecutar el procedimiento.

Mayor seguridad

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

Reutilización del código

El código de cualquier operación de base de datos redundante resulta un candidato perfecto para la encapsulación de procedimientos. De este modo, se elimina la necesidad de escribir de nuevo el mismo código, se 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.

Tipos de procedimientos almacenados

Sistema

Los procedimientos del sistema se incluyen con SQL Server. Están almacenados físicamente en la base de datos interna y oculta Resource y se muestran de forma lógica en el esquema sys de cada base de datos definida por el sistema y por el usuario.

Definidos por el usuario

Un procedimiento definido por el usuario se puede crear en una base de datos definida por el usuario o en todas las bases de datos del sistema excepto en la base de datos Resource.

Temporales

Los procedimientos temporales son una forma de procedimientos definidos por el usuario. Los procedimientos temporales son iguales que los procedimientos permanentes salvo porque se almacenan en tempdb. Hay dos tipos de procedimientos temporales: locales y globales.

```
/*** Este procedimiento almacenado nos permite cargar un nuevo cliente ***/

CREATE PROC PA_CargarCliente
       @dni INT,
       @nombre VARCHAR (30),
       @apellido VARCHAR (30),
       @numero_telefono VARCHAR (11),
       @correo VARCHAR (50),
       @fecha_nacimiento DATE
AS
       INSERT INTO cliente
       VALUES (@dni,
                     @nombre,
                     @apellido,
                     @numero_telefono,
                     @correo,
                     @fecha_nacimiento)
       SELECT @dni = dni,
                  @nombre = nombre,
                  @apellido = apellido,
                  @numero_telefono = celular,
                  @correo = correo,
                  @fecha_nacimiento = fechaNacimiento
       FROM cliente;
       SELECT * FROM cliente;
GO

     /*** Este PA nos permite leer los datos de los clientes solo con el DNI ***/

CREATE PROC PA_Leer_Datos_Clientes
              @dni INT
AS
       SELECT dni'DNI',
                  nombre'NOMBRE',
                  apellido'APELLIDO',
                  celular'TELEFONO',
                  correo'CORREO',
                  DATEDIFF (YEAR, fechaNacimiento, GETDATE())'EDAD'
       FROM cliente
       WHERE (dni = @dni);

GO
```

VISTAS

Una vista es una tabla virtual cuyo contenido está definido por una consulta. Al igual que una tabla, una vista consta de un conjunto de columnas y filas de datos con un nombre. Sin embargo, a menos que esté indizada, una vista no existe como conjunto de valores de datos almacenados en una base de datos. Las filas y las columnas de datos proceden de tablas a las que se hace referencia en la consulta que define la vista y se producen de forma dinámica cuando se hace referencia a la vista. Las vistas suelen usarse para centrar, simplificar y personalizar la percepción de la base de datos para cada usuario. Las vistas pueden emplearse como mecanismos de seguridad, que permiten a los usuarios obtener acceso a los datos por medio de la vista, pero no les conceden el permiso de obtener acceso directo a las tablas base subyacentes de la vista.

Vistas indizadas: Una vista indizada es una vista que se ha materializado. Esto significa que se ha calculado la definición de la vista y que los datos resultantes se han almacenado como una tabla.

Vistas con particiones: Una vista con particiones combina datos horizontales con particiones de un conjunto de tablas miembro en uno o más servidores. Esto hace que los datos aparezcan como si fueran de una tabla.

Vistas del sistema: Las vistas de sistema exponen metadatos de catálogo. Puede usar las vistas del sistema para devolver información acerca de la instancia de SQL Server u objetos definidos en la instancia.

```
              /*** Esta vista nos permite listar los pasajes ***/
CREATE VIEW VW_ListaPasajes WITH ENCRYPTION
AS
      SELECT
            cod_pasaje'Nº',
            terminal.nombre'TERMINAL',
            opciones_pago.tipo_pago'TIPO DE PAGO',
            (localidad.localidad + ' - ' + ' - ' + provincia.provincia + ' -
' + pais.pais) 'DESTINO',
            empresa.nombre'EMPRESA',
            colectivo.numero_colectivo'Nº COLECTIVO',
            cliente.dni'DNI',
            (cliente.nombre + ' ' + cliente.apellido)'CLIENTE',
            (empleado.nombre + ' ' + empleado.apellido)'VENDEDOR',
            fecha_emicion'FECHA COMPRA',
            fecha_salida'SALIDA',
            destino.precio'PRECIO'
      FROM pasaje
JOIN terminal ON terminal.cod_terminal = pasaje.cod_terminal
JOIN opciones_pago ON opciones_pago.cod_pago = pasaje.cod_pago
JOIN empresa ON empresa.cod_empresa = pasaje.cod_empresa
JOIN colectivo ON colectivo.numero_colectivo = pasaje.numero_colectivo AND
              colectivo.cod_empresa = pasaje.cod_empresa
JOIN cliente ON cliente.dni = pasaje.dni_cliente
JOIN empleado ON empleado.dni = pasaje.dni_vendedor
JOIN destino ON destino.cod_destino = pasaje.cod_destino
JOIN localidad ON localidad.cod_localidad = destino.cod_localidad AND
              localidad.cod_provincia = destino.cod_provincia AND
              localidad.cod_pais = destino.cod_pais
JOIN provincia ON provincia.cod_provincia = destino.cod_provincia AND
              provincia.cod_pais = destino.cod_pais
JOIN pais ON pais.cod_pais = destino.cod_pais
```

TRIGGERS

Un desencadenador 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.

FOR o AFTER especifica que el desencadenador DML solo se activa cuando todas las operaciones especificadas en la instrucción SQL desencadenadora se han iniciado correctamente. Además, todas las acciones referenciales en cascada y las comprobaciones de restricciones también deben ser correctas para que este desencadenador se active.

No puede definir desencadenadores AFTER en vistas.

```
 /***TRIGGER para saber que usuario inserto un nuevo cliente junto con la fecha ***/

CREATE TRIGGER TR_RegistroInsercionUsuario_BI
       ON cliente FOR INSERT
 AS
 SET NOCOUNT ON
 DECLARE @dni_cliente INT
 SELECT @dni_cliente = dni FROM inserted
      INSERT INTO registro_cliente
      (dni_cliente, dato ,usuario, fecha)
VALUES
(@dni_cliente, 'SE AGREGO CLIENTE', SYSTEM_USER, GETDATE());
```

USUARIOS

Los permisos de Motor de base de datos se administran en el nivel de servidor asignados a los inicios de sesión y roles de servidor, y en el nivel de base de datos asignados a usuarios de base de datos y roles base de datos. Una vez que comprenda los permisos, aplique permisos de nivel de servidor a los inicios de sesión o roles de servidor, y permisos de nivel de base de datos a los usuarios o roles de base de datos con las instrucciones GRANT, REVOKE y DENY.

Para administrar con facilidad los permisos de las bases de datos, SQL Server proporciona varios roles* que son las entidades de seguridad que agrupan a otras entidades de seguridad. Son como grupos en el sistema operativo Microsoft Windows. Los roles de nivel de base de datos se aplican a toda la base de datos en lo que respecta a su ámbito de permisos.

Para agregar y quitar usuarios en un rol de base de datos, use las opciones ADD MEMBER y DROP MEMBER de la instrucción ALTER ROLE.

Los roles fijos de base de datos se definen en el nivel de base de datos y existen en cada una de ellas. Los miembros de los roles de base de datos db_owner pueden administrar la pertenencia a roles fijos de base de datos. También hay algunos roles de base de datos con fines especiales en la base de datos msdb.

```
/*** Creación de usuarios y permisos ***/
CREATE LOGIN vendedor WITH PASSWORD = '1234', DEFAULT_DATABASE = terminal
GO
USE terminal
GO
CREATE USER vendedor FOR LOGIN vendedor
GO
GRANT SELECT ON OBJECT::destino TO vendedor
GO
GRANT SELECT ON OBJECT::empresa TO vendedor
GO
GRANT SELECT ON OBJECT::asientos TO vendedor
GO
GRANT SELECT ON OBJECT::opciones_pago TO vendedor
GO

DENY SELECT ON OBJECT::tipo_empleado TO vendedor
GO
DENY SELECT ON OBJECT::tipo_terminal TO vendedor
GO
DENY SELECT ON OBJECT::empleado TO vendedor
GO

DENY DELETE ON OBJECT::tipo_empleado TO vendedor
GO
DENY DELETE ON OBJECT::tipo_terminal TO vendedor
GO
DENY DELETE ON OBJECT::empleado TO vendedor
GO
```

INDICES

Un índice es una estructura de disco asociada con una tabla o una vista que acelera la recuperación de filas de la tabla o de la vista. Un índice contiene claves generadas a partir de una o varias columnas de la tabla o la vista. Dichas claves están almacenadas en una estructura (árbol b) que permite que SQL Server busque de forma rápida y eficiente la fila o filas asociadas a los valores de cada clave.

Los índices se crean automáticamente cuando las restricciones PRIMARY KEY y UNIQUE se definen en las columnas de tabla. Por ejemplo, cuando crea una tabla con una restricción UNIQUE, Motor de base de datos crea automáticamente un índice no agrupado. Si configura una restricción PRIMARY KEY, Motor de base de datos crea automáticamente un índice agrupado, a menos que ya exista uno. Cuando intenta aplicar una restricción PRIMARY KEY en una tabla existente y ya existe un índice agrupado en esa tabla, SQL Server aplica la clave principal mediante un índice no agrupado.

Los índices bien diseñados pueden reducir las operaciones de E/S de disco y consumen menos recursos del sistema, con lo que mejoran el rendimiento de la consulta. Los índices pueden ser útiles para diversas consultas que contienen instrucciones SELECT, UPDATE, DELETE o MERGE.

Tipos de índices:

Índices agrupados

Un índice agrupado define el orden en el cual los datos son físicamente almacenados en una tabla. Los datos de las tablas pueden ser ordenados sólo en una forma, por lo tanto, sólo puede haber un índice agrupado por tabla. En SQL Server, la restricción de llave primaria crea automáticamente un índice agrupado en esa columna en particular.

Índices no agrupados

Un índice no agrupado no ordena los datos físicos dentro de la tabla. De hecho, un índice no agrupado es agrupado en un solo lugar y los datos de la tabla son almacenados en otro lugar. Esto es similar a un libro de texto donde el contenido del libro está localizado en un lugar y el índice está localizado en otro. Esto permite tener más de un índice no agrupado por tabla.

Es importante mencionar que dentro de la tabla los datos serán ordenados por un índice agrupado. De todos modos, por dentro los datos del índice no agrupado son almacenados en un orden específico. El índice contiene valores de columna en los cuales el índice es creado y la dirección del registro a la que el valor de la columna pertenece

Diferencias

1. Puede haber sólo un índice agrupado por tabla. De todos modos, usted puede crear múltiples índices no agrupados en una sola tabla.
2. Los índices agrupados sólo ordenan tablas. Por lo tanto, no consumen almacenaje extra. Los índices no agrupados son almacenados en un lugar separado de la tabla real. Reclamando más espacio de almacenamiento.
3. Los índices agrupados son más rápidos que los índices no agrupados, ya que no involucran ningún paso extra de búsqueda.

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 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'. Los puntos de recuperación (SavePoints) permiten manejar las transacciones por pasos, pudiendo hacer rollbacks hasta un punto marcado por el savepoint y no por toda la transacción.

También se pueden manejar tomadas como excepciones mediante un try o un catch, como también con un operador condicional if.

```
/*** Esta transacción cancela la operación si no encuentra asientos disponibles para esa fecha ***/

CREATE PROC comprarPasaje (
	@cod_terminal INT,
	@cod_pago INT,
	@cod_destino INT,
	@cod_empresa INT,
	@numero_colectivo INT,
	@dni_cliente INT,
	@dni_vendedor INT,
	@fecha_salida DATE
	)
AS
	BEGIN
		BEGIN TRY
			BEGIN TRANSACTION
			declare @error date
			DECLARE @fecha_emicion DATETIME
			SELECT @fecha_emicion = GETDATE()
			INSERT INTO pasaje (cod_terminal, cod_pago, cod_destino,
cod_empresa, numero_colectivo, dni_cliente, dni_vendedor, fecha_emicion,
fecha_salida)
			VALUES (@cod_terminal, @cod_pago, @cod_destino, @cod_empresa,
@numero_colectivo, @dni_cliente, @dni_vendedor, @fecha_emicion, @error)

			IF  @fecha_salida in (select fecha_salida from asientos)
				BEGIN
				UPDATE asientos SET num_asientos = num_asientos - 1 WHERE
@cod_empresa = empresa AND @numero_colectivo = num_colectivo AND @error =
fecha_salida
				END
			COMMIT TRANSACTION
		END TRY
		BEGIN CATCH
			ROLLBACK TRANSACTION
			PRINT 'No hay asientos disponibles para esa fecha';
			THROW;
		END CATCH
	END
GO
```

RESPALDO Y RESTAURACION

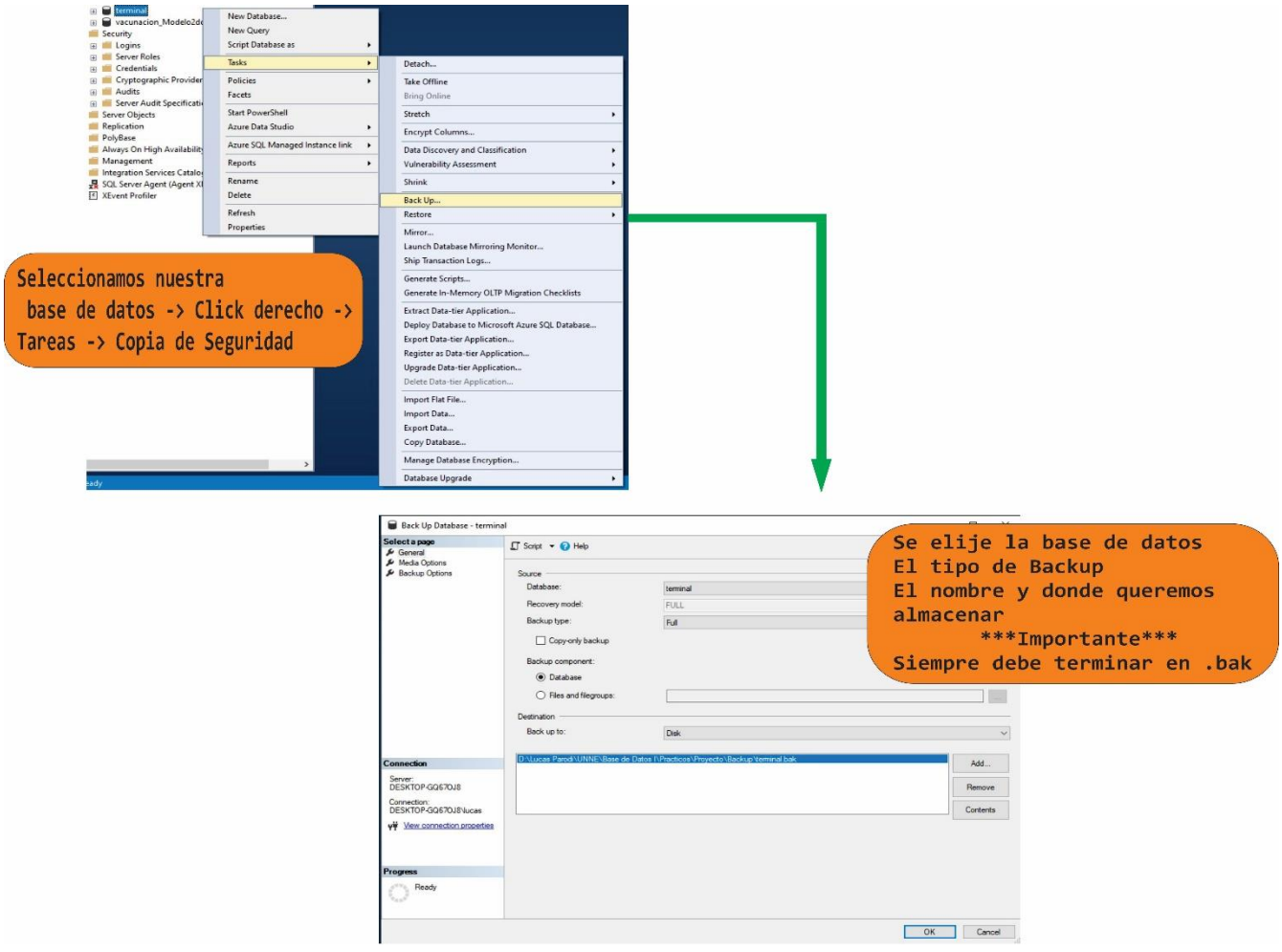
La palabra backup, viene a traducirse como respaldo, y es eso, un respaldo de información. Es decir, es la copia de información de los datos importantes de un dispositivo primario en uno o varios dispositivos secundarios, esto para que en caso de que ocurra alguna falla con el primero, sea posible disponer con la mayor parte de la información necesaria para continuar con las actividades rutinarias y evitar pérdida de datos.

¿Por qué es tan importante un backup?

La importancia principal de un backup radica en que todos los dispositivos de almacenamiento masivo de información tienen –aunque sea mínima- la posibilidad de fallar, por lo tanto, tener una copia de seguridad de la información es necesario, ya que la probabilidad de que dos dispositivos presentes fallas, es mucho menos probable.

Con SQL Server podemos realizar copias de seguridad de una forma segura y confiable, podemos elegir el periodo con el cual queremos que se realice, el tamaño máximo permitido, la ubicación donde se va a almacenar (esto es importante ya que, si estamos en un hosting de servicio web, podemos elegir que se realice en otro dispositivo así nos ahorramos espacio en el servidor) e incluso la frecuencia con la que se eliminan las copias viejas.

A continuación, se muestra cómo realizar una copia de seguridad en SQL Server Management Studio.



AUDITORIA

La auditoría de base de datos, finalmente, es un proceso implementado por los auditores de sistemas con el fin de auditar los accesos a los datos, por lo general siguiendo bien una metodología basada en un checklist que contempla los puntos que se quieren comprobar o mediante la evaluación de riesgos potenciales.

En concreto, se realiza un examen de los accesos a los datos almacenados en las bases de datos con el fin de poder medir, monitorear y tener constancia de los accesos a la información almacenada en las mismas. Si bien el objetivo puede variar en función de la casuística, en todos los casos el fin último persigue, de uno u otro modo, la seguridad corporativa.

Una auditoría de base de datos, por lo tanto, facilita herramientas eficaces para conocer de forma exacta cuál es la relación de los usuarios a la hora de acceder a las bases de datos, incluyendo las actuaciones que deriven en una generación, modificación o eliminación de datos.

En la práctica, permite responder a muchas preguntas que pueden resultar relevantes a la hora de controlar y auditar. Desde determinar quién accede a los datos, cuándo se accedió o cuál es su ubicación en la Red y desde qué dispositivo o aplicación lo hizo, hasta qué sentencia SQL fue ejecutada, así como el resultado del acceso.

En este sentido, la auditoría de base de datos es un control necesario, cuya dificultad aumenta de forma paralela a la creciente complejidad de las tecnologías de bases de datos. Así mismo, las amenazas de seguridad se han multiplicado, apareciendo nuevos riesgos e incrementándose los ya existentes, al tiempo que se amplía su alcance a través de la disciplina conocida como Gestión de Recursos de Información.

```
-- Auditoria para mostrar fecha y hora que se realizo un backup
/****** Object:  Audit [Audit-Backup]    Script Date: 12/11/2022 12:13:46 ******/
CREATE SERVER AUDIT [Audit-Backup]
TO FILE
(   FILEPATH = N'D:\Lucas Parodi\UNNE\Base de Datos I\Practicos\Proyecto\Auditorias\'
    ,MAXSIZE = 0 MB
    ,MAX_ROLLOVER_FILES = 2147483647
    ,RESERVE_DISK_SPACE = OFF
) WITH (QUEUE_DELAY = 1000, ON_FAILURE = CONTINUE, AUDIT_GUID = 'f1b51878-f284-4248-835a-ae2669e2918c')
ALTER SERVER AUDIT [Audit-Backup] WITH (STATE = ON)
GO

CREATE SERVER AUDIT SPECIFICATION [ServerAuditSpecification-Backup]
FOR SERVER AUDIT [Audit-Backup]
ADD (BACKUP_RESTORE_GROUP)
WITH (STATE = ON)
GO
```

CAPITULO III: METODOLOGIA

El proyecto lo llevamos a cabo mediante la búsqueda individual de información respecto al funcionamiento de distintas terminales de ómnibus para así tener un mejor panorama a la hora de realizar las reuniones tanto virtuales como presenciales donde expusimos nuestras ideas y posibles problemas que pudieran surgir a medida que avanzara el proyecto.

Realizamos el diagrama de relación-entidad y nos dividimos en partes equitativa la investigación acerca de las diferentes técnicas de trabajo (disparadores, transacciones, vistas, etc.) y el desarrollo de las mismas.

Fuimos proponiendo distintas fechas de reuniones para ir compartiendo lo desarrollado y acoplando las distintas partes del proyecto, eso lo hicimos de manera virtual para tener una mejor visión del código y la búsqueda de información.

El informe lo realizamos entre todos ya que fuimos aportando ideas y uniendo lo investigado, así como la carga de datos ya que contiene muchas claves foráneas y se hacía imposible ir viendo los errores ya que algunas tablas dependen de otras.

Herramientas utilizadas

WhatsApp: Fue unos de los medios más utilizado para ir coordinando las reuniones, respondiendo consultas, y pasar información no solo del proyecto si no de la materia en sí.

Meet: Fue el medio que utilizamos para llevar a cabo algunas reuniones virtuales e ir compartiendo el proceso en la creación del proyecto ya que es una plataforma gratuita y que no requiere una conexión a internet muy buena.

Discord: Se utilizo también para llevar a cabo reuniones virtuales, pero sobre todo para ir almacenando información en los distintos canales de texto. Se realizaron varios roles y permisos para simular un verdadero proyecto ya que el servidor lo comparten muchas personas que no son de la carrera ni la facultad.

GitHub: fue la última herramienta en incorporar, porque fue donde cargamos los códigos para compartirnos y dimos permisos de colaborador a los profesores que desarrollan la materia.

Como herramienta principal y primordial, utilizamos **SQL SERVER MANAGEMENT STUDIO** donde llevamos a cabo el desarrollo de la base de datos. Decidimos utilizar esta herramienta ya que es muy completa y brinda una versión gratuita.

A continuación, detallaremos algunas descripciones y ventajas de utilizarla.

Es un sistema de gestión de bases de datos relacionales (SGBD) basado en el lenguaje TransactSQL, y específicamente en Sybase IQ, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea. Así de tener unas ventajas que a continuación se pueden describir.

Microsoft SQL Server constituye la alternativa de Microsoft a otros potentes sistemas gestores de bases de datos como son Oracle, Sybase ASE, PostgreSQL o MySQL.

Ventajas:

1. Soporte de transacciones.
2. Escalabilidad, estabilidad y seguridad.
3. Soporta procedimientos almacenados.
4. Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
5. Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información.
6. Permite administrar información de otros servidores de datos.

Desventajas:

1. Costo de las licencias comparadas con otros competidores.

CAPITULO IV:

DIAGRAMA ENTIDAD-RELACION

DICCIONARIO DE DATOS:

FORMA DE PAGOS				
NOMBRE DEL CAMPO	CLAVE	TIPO DE DATO	DESCRIPCION	TAMAÑO DEL CAMPO
COD_PAGO	PK	ENTERO	REPRESENTA LAS DISTINTAS FORMAS DE PAGO	AUTOINCREMENTAL
OPCION		VARCHAR	NOMBRE DE LA FORMA DE PAGO	30
CLIENTE				
NOMBRE DEL CAMPO	CLAVE	TIPO DE DATO	DESCRIPCION	TAMAÑO DEL CAMPO
DNI_CLIENTE	PK	ENTERO	DNI DEL CLIENTE QUE VA A COMPRAR EL PASAJE	8
NOMBRE		VARCHAR	NOMBRE DEL CLIENTE	30
APELLIDO		VARCHAR	APELLIDO DEL CLIENTE	30
CELULAR		VARCHAR	CELULAR DEL CLIENTE	12
CORREO		VARCHAR	CORREO DEL CLIENTE	50
FECHA_NACIMIENTO		DATE	FECHA DE NACIMIENTO DEL CLIENTE	-
EMPLEADO				
NOMBRE DEL CAMPO	CLAVE	TIPO DE DATO	DESCRIPCION	TAMAÑO DEL CAMPO
DNI	PK	ENTERO	DNI DE LOS EMPLEADOS	8
NOMBRE		VARCHAR	NOMBRE DEL EMPLEADO	30
APELLIDO		VARCHAR	APELLIDO DEL EMPLEADO	30
CELULAR		VARCHAR	CELULAR DEL EMPLEADO	12
CORREO		VARCHAR	CORREO DEL EMPLEADO	50
FECHA_NACIMIENTO		DATE	FECHA NACIMIENTO DEL EMPLEADO	-
COD_TIPO_EMPLEADO		ENTERO	ASOCIA CON EL TIPO EMPLEADO	-
DIRECCION		VARCHAR	DIRECCION DEL EMPLEADO	100
CHOFER				
NOMBRE DEL CAMPO	CLAVE	TIPO DE DATO	DESCRIPCION	TAMAÑO DEL CAMPO
DNI	PK	INT	DNI CHOFER	8
COD_EMPRESA	FK	INT	EMPRESA CON LA CUAL SE ENCUENTRA CONTRATADO	0
NOMBRE		VARCHAR	NOMBRE DEL CHOFER	30
APELLIDO		VARCHAR	APELLIDO DEL CHOFER	30
CELULAR		VARCHAR	CELULAR DEL CHOFER	12
CORREO		VARCHAR	CORREO DEL CHOFER	50
FECHA_NACIMIENTO		TIME	FECHA DE NACIMIENTO DEL CHOFER	-
TIPO_LICENCIA	FK	VARCHAR	CATEGORIA DE LA LICENCIA DEL CHOFER	6
COLECTIVO				
NOMBRE DEL CAMPO	CLAVE	TIPO DE DATO	DESCRIPCION	TAMAÑO DEL CAMPO
COD_EMPRESA	FK-PK	ENTERO	EMPRESA DUEÑA DEL COLECTIVO	5
NUM_COLECTIVO	PK	ENTERO	NUMERO DEL COLECTIVO	5
NUM_PATENTE		VARCHAR	PATENTE DEL COLECTIVO	7
NUM_ASIENTOS		ENTERO	CANTIDAD DE ASIENTOS QUE TRAE EL COLECTIVO	2
DESCRIPCION		VARCHAR	DESCRIPCION DEL COLECTIVO	70
TERMINAL				
NOMBRE DEL CAMPO	CLAVE	TIPO DE DATO	DESCRIPCION	TAMAÑO DEL CAMPO
COD_TERMINAL	PK	ENTERO	NUMERO DE LA TERMINAL	7
TIPO_TERMINAL	FK	ENTERO	TIPO DE TERMINAL (LOCAL, INTERPROVINCIAL,ETC.)	3
NOMBRE		VARCHAR	NOMBRE DE LA TERMINAL	30
CANT_ESTACIONES		ENTERO	CANTIDAD DE ESTACIONES PARA COLECTIVOS DISPONIBLES	3
TELEFONO		VARCHAR	TELEFONO DE LA TERMINAL	12
DIRECCION		VARCHAR	DIRECCION DE LA TERMINAL	100
CORREO		VARCHAR	CORREO ELECTRONICO DE LA TERMINAL	50
TIPO_TERMINAL				
NOMBRE DEL CAMPO	CLAVE	TIPO DE DATO	DESCRIPCION	TAMAÑO DEL CAMPO
COD_TIPO	PK	ENTERO	CATEGORIA DE LA TERMINAL(NACIONAL, PROVINCIAL, ETC,)	5
TIPO		VARCHAR	CATEGORIA	50

FORMA DE PAGOS				
NOMBRE DEL CAMPO	**CLAVE**	**TIPO DE DATO**	**DESCRIPCION**	**TAMAÑO DEL CAMPO**
COD_PAGO	PK	ENTERO	REPRESENTA LAS DISTINTAS FORMAS DE PAGO	AUTOINCREMENTAL
OPCION		VARCHAR	NOMBRE DE LA FORMA DE PAGO	30
CLIENTE				
NOMBRE DEL CAMPO	**CLAVE**	**TIPO DE DATO**	**DESCRIPCION**	**TAMAÑO DEL CAMPO**
DNI_CLIENTE	PK	ENTERO	DNI DEL CLIENTE QUE VA A COMPRAR EL PASAJE	8
NOMBRE		VARCHAR	NOMBRE DEL CLIENTE	30
APELLIDO		VARCHAR	APELLIDO DEL CLIENTE	30
CELULAR		VARCHAR	CELULAR DEL CLIENTE	12
CORREO		VARCHAR	CORREO DEL CLIENTE	50
FECHA_NACIMIENTO		DATE	FECHA DE NACIMIENTO DEL CLIENTE	-
EMPLEADO				
NOMBRE DEL CAMPO	**CLAVE**	**TIPO DE DATO**	**DESCRIPCION**	**TAMAÑO DEL CAMPO**
DNI	PK	ENTERO	DNI DE LOS EMPLEADOS	8
NOMBRE		VARCHAR	NOMBRE DEL EMPLEADO	30
APELLIDO		VARCHAR	APELLIDO DEL EMPLEADO	30
CELULAR		VARCHAR	CELULAR DEL EMPLEADO	12
CORREO		VARCHAR	CORREO DEL EMPLEADO	50
FECHA_NACIMIENTO		DATE	FECHA NACIMIENTO DEL EMPLEADO	-
COD_TIPO_EMPLEADO		ENTERO	ASOCIA CON EL TIPO EMPLEADO	-
DIRECCION		VARCHAR	DIRECCION DEL EMPLEADO	100
CHOFER				
NOMBRE DEL CAMPO	**CLAVE**	**TIPO DE DATO**	**DESCRIPCION**	**TAMAÑO DEL CAMPO**
DNI	PK	INT	DNI CHOFER	8
COD_EMPRESA	FK	INT	EMPRESA CON LA CUAL SE ENCUENTRA CONTRATADO	0
NOMBRE		VARCHAR	NOMBRE DEL CHOFER	30
APELLIDO		VARCHAR	APELLIDO DEL CHOFER	30
CELULAR		VARCHAR	CELULAR DEL CHOFER	12
CORREO		VARCHAR	CORREO DEL CHOFER	50
FECHA_NACIMIENTO		TIME	FECHA DE NACIMIENTO DEL CHOFER	-
TIPO_LICENCIA	FK	VARCHAR	CATEGORIA DE LA LICENCIA DEL CHOFER	6
COLECTIVO				
NOMBRE DEL CAMPO	**CLAVE**	**TIPO DE DATO**	**DESCRIPCION**	**TAMAÑO DEL CAMPO**
COD_EMPRESA	FK-PK	ENTERO	EMPRESA DUEÑA DEL COLECTIVO	5
NUM_COLECTIVO	PK	ENTERO	NUMERO DEL COLECTIVO	5
NUM_PATENTE		VARCHAR	PATENTE DEL COLECTIVO	7
NUM_ASIENTOS		ENTERO	CANTIDAD DE ASIENTOS QUE TRAE EL COLECTIVO	2
DESCRIPCION		VARCHAR	DESCRIPCION DEL COLECTIVO	70

DESTINO				
NOMBRE DEL CAMPO	**CLAVE**	**TIPO DE DATO**	**DESCRIPCION**	**TAMAÑO DEL CAMPO**
COD_DESTINO	PK	ENTERO	CODIGO DE DESTINO	5
COD_PAIS		ENTERO	PAIS	5
COD_PROVINCIA		ENTERO	PROVINCIA	5
COD_LOCALIDAD		ENTERO	LOCALIDAD	5
PRECIO		NUMERIC	PRECIO DEL PASAJE	9.2
DETALLE		VARCHAR	DETALLE DEL DESTINO	50
TIPO_EMPLEADO				
NOMBRE DEL CAMPO	**CLAVE**	**TIPO DE DATO**	**DESCRIPCION**	**TAMAÑO DEL CAMPO**
COD_TIPO_EMPLEADO	PK	ENTERO	CODIGO DEL TIPO EMPLEADO	5
TIPO_EMPLEADO		VARCHAR	DESCRIPCION	30

TIPOS DE RESTRICCIONES

TABLA	TIPO RESTRICCION	NOMBRE	REFERENCIA
asientos	CHECK	CK_CheckAsientos	-
asientos	FOREIGN KEY	FK_Asientos_Colectivo	COLECTIVO
destino	FOREIGN KEY	FK_destino_localidad	LOCALIDAD
colectivo	FOREIGN KEY	FK_colectivo_empresa	EMPRESA
empleado	FOREIGN KEY	FK_empleado_tipo_empleado	TIPO_EMPLEADO
pasaje	FOREIGN KEY	FK_pasaje_terminal	TERMINAL
pasaje	FOREIGN KEY	FK_pasaje_tipo_pago	TIPO_PAGO
pasaje	FOREIGN KEY	FK_pasaje_destino	DESTINO
pasaje	FOREIGN KEY	FK_pasaje_empresa	EMPRESA
pasaje	FOREIGN KEY	FK_pasaje_cliente	CLIENTE
pasaje	FOREIGN KEY	FK_pasaje_empleado	EMPLEADO
localidad	FOREIGN KEY	FK_localidad_provincia	PROVINCIA
provincia	FOREIGN KEY	FK_provincia_pais	PAIS
chofer	FOREIGN KEY	FK_chofer_empresa	EMPRESA
terminal	FOREIGN KEY	FK_terminal_tipo_terminal	TIPO_TERMINAL
tipo_empleado	PRIMARY KEY	PK__tipo_emp__5923F315A33BC276	-
empleado	PRIMARY KEY	PK__empleado__D87608A6615083AA	-
colectivo	PRIMARY KEY	PK__colectiv__8F45AFB9C83ED81E	-
localidad	PRIMARY KEY	PK__localida__65553356141C1AC5	-
empresa	PRIMARY KEY	PK__empresa__C1DD514DE1566B5A	-
chofer	PRIMARY KEY	PK__chofer__D87608A6DB7ADA15	-
cliente	PRIMARY KEY	PK__cliente__D87608A6D1DF6C13	-
pais	PRIMARY KEY	PK__pais__CB1379BB3A76A3FD	-
provincia	PRIMARY KEY	PK__provinci__5FB30042183B1B41	-
asientos	PRIMARY KEY	PK__asientos__3B2D547C0B5A8766	-
opciones_pago	PRIMARY KEY	PK__opciones__CB1571389261FA77	-
destino	PRIMARY KEY	PK__destino__121A4CF51E66973C	-
tipo_terminal	PRIMARY KEY	PK__tipo_ter__7C06900D4FFA39C7	-
terminal	PRIMARY KEY	PK__terminal__191BF49EFD171961	-
pasaje	PRIMARY KEY	PK__pasaje__AADE5FD6832F0F52	-
cliente	UNIQUE	UQ__cliente__D87608A7EE59A44C	-
chofer	UNIQUE	UQ__chofer__D87608A796F9D1DE	-
empleado	UNIQUE	UQ__empleado__D87608A7D76BA1BD	-

CAPITULO V:

Conclusiones

Como conclusión podemos decir que se alcanzaron los objetivos planeados realizando una buena estructuración y modelado de la base de datos, se utilizaron diferentes técnicas aplicando buenas prácticas de programación y normalización de tablas, esto también da la posibilidad para que en un futuro se pueda implementar un servicio de aplicación web.

De este modo pudimos solucionar el problema principal que era generar un pasaje a un destino en concreto seleccionando distintas empresas que hay a disposición. También sirve para generar datos, como, por ejemplo, la cantidad de viajantes en distintas fechas, destinos más concurrentes, listas de empleados con sus respectivos rubros etc.

Se generaron diferentes usuarios con sus respectivas contraseñas y rango para llevar a cabo la utilización y el control de la base de datos. Se implementaron disparadores, procedimientos almacenados, transacciones para tener un mejor nivel de integridad y seguridad junto con las vistas para evitar la manipulación directa de los datos.

Se utilizaron muchas herramientas de comunicación para trabajar en equipo e ir coordinando y solucionando los problemas que surgían a medida que avanzaba el proyecto ya que fueron varias etapas las que se llevaron a cabo, como ser la **planificación** donde se recopilo información sobre distintas herramientas, el funcionamiento de las terminales ya existentes, etc. El **análisis** donde decidimos que haría el software, el **diseño** donde se realizo un diagrama entidad-relación para de este modo realizar el **desarrollo** y las distintas **pruebas** de control del mismo.

BIBLIOGRAFIA

Estos son algunos de los sitios utilizados para la búsqueda de información sobre SQL en general.

- https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver16
- SQL Server Ya (tutorialesprogramacionya.com)
- https://www.ibm.com/docs/en/i/7.2?topic=programming-data-manipulation-language
- https://www.youtube.com/watch?v=mLosHjJdc54&list=PLU8oAlHdN5Bmx-LChV4K3MbHrpZKefNwn&index=18