Universidad Nacional del Nordeste

Facultad de Ciencias Exactas y Naturales y Agrimensura

Licenciatura en Sistemas de Información

BASE DE DATOS I

Grupo N°7

Gestión de Vuelos en Aeropuerto

Integrantes:

- *Coutinho, Martin.* *LU: 53327*
- *Coronel, Hipólito Ismael.* *LU: 53880*
- *Garrido, Santiago.* *LU: 46248*
- *Gómez, Kevin David.* *LU: 53555*
- *Kryvenki, Nicolás Emiliano.* *LU: 53235*

2020

Índice:

Proyecto: Gestión de vuelos en aeropuerto

Descripción del proyecto:

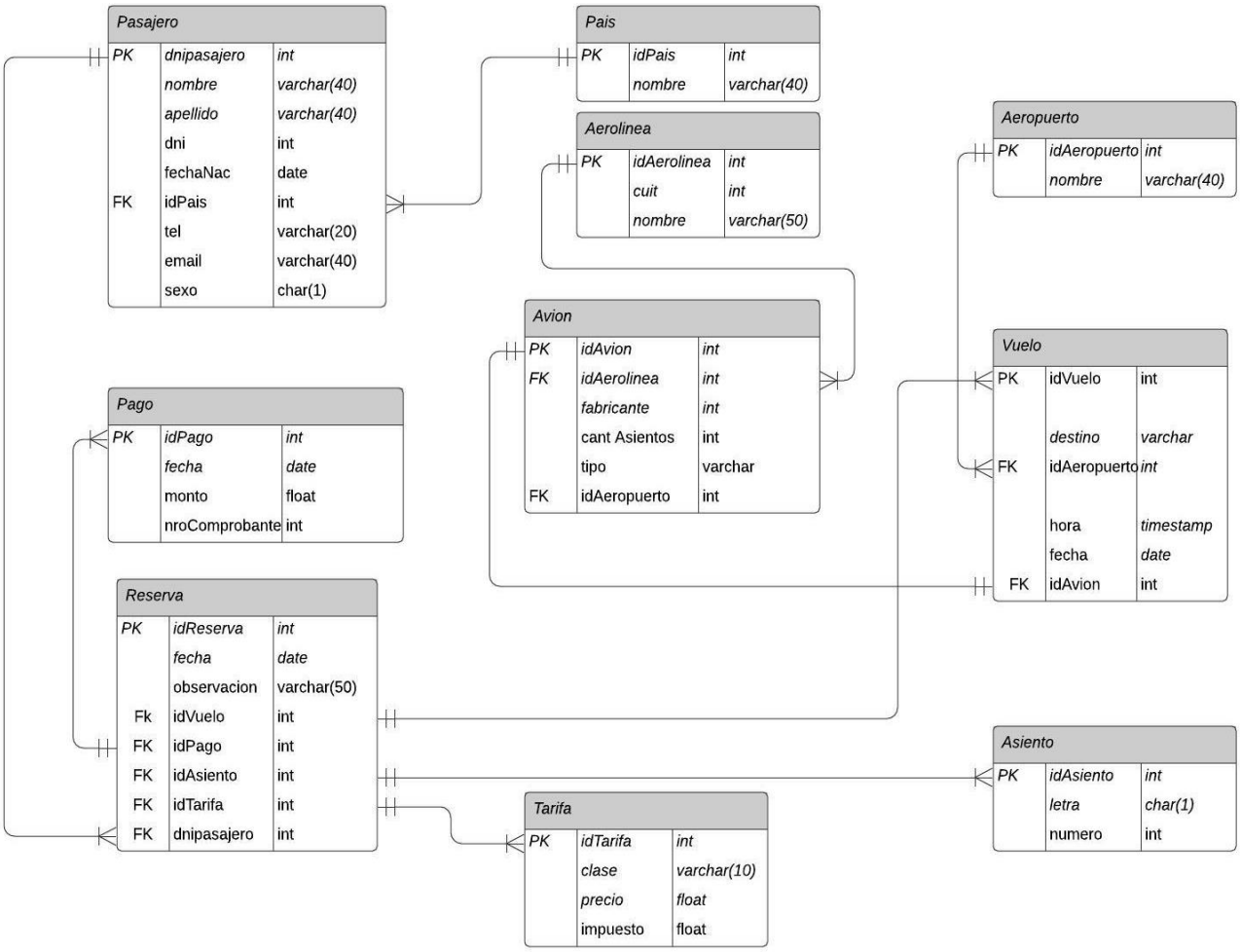
En el presente informe, se desarrollará un sistema de Base de Datos para la gestión de vuelos en un aeropuerto, la cual permitirá agilizar y simplificar varios de los trámites que hay que realizar a la hora de querer utilizar este medio de transporte.

Alcance del proyecto:

La base de datos permitirá gestionar los vuelos con todos los datos correspondientes para contar con la información completa sobre dicho vuelo, el cual ya incluirá las reservas por parte de los pasajeros junto a sus datos, las tarifas finales, asientos, conocer la aerolínea que brindará el servicio entre otros.

Esto resulta fundamental ya que facilitará la manera de poder controlar los vuelos realizados, servirá como información de respaldo ante la aparición de problemas relacionados con algún vuelo en particular, por otro lado, permitirá emitir informes con estadísticas mensuales o anuales con la información recaudada a diario, como ser: la cantidad de vuelos, aerolíneas más utilizadas, entre otras.

Diagrama Entidad Relación:



Restricciones:

Nombre entidad	Tipo restricción	Nombre de la restricción	Columna
aeropuerto	Primary key	PK__aeropuer__12CB9FBFC83FE810	idaeropuerto
aerolinea	Primary key	PK__aeroline__BB152E88DBC687F2	idaerolinea
avion	Primary key	PK__avion__7564B8CA36DFBC36	idavion
avion	Foreign key	FK_aerolinea	ideaerolinea
vuelo	Primary key	PK__vuelo__3B128B9888F71914	idvuelo
vuelo	Foreign key	FK_avion	Idavion
vuelo	Foreign key	FK_idaeropuerto2	idaeropuerto
tarifa	Primary key	PK__tarifa__72C0F72581D28DEF	Idtarifa
asiento	Primary key	PK__asiento__A76D3DC460FA7802	idasiento
pais	Primary key	PK__pais__BD2285E34E40709A	idpais
pago	Primary key	PK__pago__BD2295AD7CCBDE21	Idpago
pasajero	Primary key	PK__pasajero__62CCD3CF6BBD7A06	dnipasajero
pasajero	Foreign key	FK_pasajero_pais	idpais
pasajero	check	CK_sexo	sexo
reserva	Primary key	PK__Reserva__94D104C8E4043726	idreserva
reserva	Foreign key	FK_reserva_vuelo	idvuelo
reserva	Foreign key	FK_reserva_pago	idpago
reserva	Foreign key	FK_reserva_asiento	idasiento
reserva	Foreign key	FK_reserva_tarifa	idtarifa
reserva	Unique	UQ_idpago	Idpago
reserva	Foreign key	FK_reserva_pasajero	dnipasajero

Índices:

Nombre entidad: aeropuerto
Nombre índice: PK__aeropuer__12CB9FBFC83FE810
Descripción: clustered, unique, primary key located on PRIMARY
Campo clave: idaeropuerto

Nombre entidad: aerolinea
Nombre índice: PK__aeroline__BB152E88DBC687F2
Descripción: clustered, unique, primary key located on PRIMARY
Campo clave: idaerolinea

Nombre entidad: avion
Nombre índice: PK__avion__7564B8CA36DFBC36
Descripción: clustered, unique, primary key located on PRIMARY
Campo clave: idavion

Nombre entidad: vuelo
Nombre índice: PK__vuelo__3B128B9888F71914
Descripción: clustered, unique, identity, primary key located on PRIMARY
Campo clave: idvuelo

Nombre entidad: tarifa
Nombre índice: PK__tarifa__72C0F72581D28DEF
Descripción: clustered, unique, primary key located on PRIMARY
Campo clave: idtarifa

Nombre entidad: asiento
Nombre índice: PK__asiento__A76D3DC460FA7802
Descripción: clustered, unique, primary key located on PRIMARY
Campo clave: idasiento

Nombre entidad: pais
Nombre índice: PK__pais__BD2285E34E40709A
Descripción: clustered, unique, identity, primary key located on PRIMARY
Campo clave: idpais

Nombre entidad: pago
Nombre índice: PK__pago__BD2295AD7CCBDE21
Descripción: clustered, unique, identity, primary key located on PRIMARY
Campo clave: idpago

Nombre entidad: pasajero
Nombre índice: PK__pasajero__62CCD3CF6BBD7A06
Descripción: clustered, unique, primary key located on PRIMARY
Campo clave: idpasajero

Nombre entidad: reserva
Nombre índice: PK__Reserva__94D104C8E4043726
Descripción: clustered, unique, identity, primary key located on PRIMARY
Campo clave: idreserva

Scripts o sentencias SQL de borrado y creación

Borrado:

```
/* ---------------------
*  -   DELETE
*  ---------------------
*/
--tablas
/*DELETE FROM reserva;
DELETE FROM aeropuerto;
DELETE FROM aerolinea;
DELETE FROM asiento;
DELETE FROM avion;
DELETE FROM pago;
DELETE FROM pais;
DELETE FROM pasajero;
DELETE FROM tarifa;
DELETE FROM vuelo;
--registros
DELETE FROM reg_reserva;
*/
/* ---------------------
```

```
*  -   DROP
*  ---------------------
*/
--tablas
/*DROP TABLE reserva;
DROP TABLE aeropuerto;
DROP TABLE aerolinea;
DROP TABLE asiento;
DROP TABLE avion;
DROP TABLE pago;
DROP TABLE pais;
DROP TABLE pasajero;
DROP TABLE tarifa;
DROP TABLE vuelo;
--registros
DROP TABLE reg_reserva;
*/
```

Creación:

```
CREATE TABLE aeropuerto(
    idAeropuerto int identity primary key not null,
    nombre VARCHAR(150) not null,

)

Create table aerolinea (idaerolinea int primary key not null,
                                        cuit int not null,
                                    nombre varchar(50)

)

create table avion(
idavion int primary key not null,
idaerolinea int not null,
fabricante int null,
cantasientos int not null ,
tipo varchar(50) null
Constraint FK_aerolinea FOREIGN KEY (idaerolinea)  REFERENCES aerolinea(idaerolinea)

)

create table vuelo (
idvuelo int identity primary key,
destino varchar(50) not null,
idaeropuerto int,
hora int,
fecha datetime,
idavion int,
CONSTRAINT FK_avion FOREIGN KEY (idavion) REFERENCES avion(idavion),
CONSTRAINT FK_idaeropuerto2 FOREIGN KEY (idaeropuerto) REFERENCES aeropuerto (idAeropuerto)

)

CREATE TABLE tarifa(
    idtarifa int primary key,
    clase varchar(10),
    precio float,
    impuesto float

    );
```

```sql
CREATE TABLE asiento(
    idasiento int primary key,
    letra char(1),
    numero int

    );



CREATE TABLE pais(
    idPais INT PRIMARY KEY IDENTITY,
    nombre VARCHAR(40) NOT NULL
)

CREATE TABLE pago(
    idPago INT PRIMARY KEY IDENTITY,
    dniPasajero INT NOT NULL,
    fecha DATETIME NOT NULL,
    nroComprobante INT NOT NULL,
    monto FLOAT NOT NULL
)

CREATE TABLE pasajero(
    dniPasajero INT primary key not null,
    nombre VARCHAR(40) NOT NULL,
    apellido VARCHAR(40) NOT NULL,
    fechaNac DATETIME,
    idPais INT NOT NULL,
    tel BIGINT NULL,
    sexo VARCHAR(1),
    email VARCHAR(80),
    CONSTRAINT CK_sexo CHECK (sexo='F' OR sexo='M'),
    CONSTRAINT FK_pasajero_pais FOREIGN KEY (idPais) REFERENCES pais(idPais)
)

CREATE TABLE Reserva(
    idReserva int identity primary key not null,
    fecha date not null,
    observacion varchar(150),
    idVuelo int not null,
    idPago int not null,
    idAsiento int not null,
    idTarifa int not null,
       dnipasajero int not null,
       CONSTRAINT FK_reserva_pasajero FOREIGN KEY (dnipasajero) REFERENCES
pasajero(dnipasajero),
    CONSTRAINT FK_reserva_vuelo FOREIGN KEY (idVuelo) REFERENCES vuelo(idVuelo),
    CONSTRAINT FK_reserva_pago FOREIGN KEY (idPago) REFERENCES pago(idPago),
    CONSTRAINT FK_reserva_asiento FOREIGN KEY (idAsiento) REFERENCES asiento(idAsiento),
    CONSTRAINT FK_reserva_tarifa FOREIGN KEY (idTarifa) REFERENCES tarifa(idTarifa),
    CONSTRAINT UQ_idpago UNIQUE(idpago)
)
```

Lote de datos con sus respectivas sentencias de inserción en las tablas:

```
-----INSERCION DE DATOS
GO
--TABLA AEROPUERTO
--creamos un proceimiento para cargar datos a la tabla aeropuerto
create procedure inserta_aeropuerto3(
--drop procedure inserta_aeropuerto3
--no creo el atributo idaeropuerto porque es autoincremental pero en los casos que no son
autoincremental hay que crearlos todos
@nombre varchar(150)
)
as
insert into aeropuerto(nombre) values (@nombre);
--cargamos los datos al procedimeinto
--el primer campo no ingreso nada porque es autoincremental
exec inserta_aeropuerto3 'belgrano';
exec inserta_aeropuerto3 'san martin';
exec inserta_aeropuerto3 'Artigas';
exec inserta_aeropuerto3 'Rosas';
select * from aeropuerto;



--TABLA AEROLINEAS
create procedure inserta_aerolinea(
@idaerolinea int,
@cuit int ,
@nombre varchar(50))
as
insert into aerolinea(idaerolinea,cuit,nombre) values (@idaerolinea, @cuit,@nombre);
--drop procedure inserta_aerolinea

exec inserta_aerolinea '1','000000001','Aerolínea argentina' ;
exec inserta_aerolinea '2','000000010','Air Flight' ;
exec inserta_aerolinea '3','000001110','Fly Bondi' ;
exec inserta_aerolinea '4','020300010','Latam airlines' ;
select * from aerolinea;

-- TABLA AVION
create procedure inserta_avion2(
@idavion int,
@idaerolinea int ,
@fabricante int ,
@cantasientos int,
@tipo varchar(50)
)
as
insert into avion(idavion, idaerolinea,fabricante,cantasientos,tipo) values (@idavion,
@idaerolinea,@fabricante,@cantasientos,@tipo);
-----------------------------------------------------------------------------------------------
----------------------
exec inserta_avion2 '1','3' ,'2','50','vip';
exec inserta_avion2 '2','3' ,'2','100','super vip`';
exec inserta_avion2 '3','2' ,'2','100',' vip';
exec inserta_avion2 '4','2' ,'2','100','super vip';
exec inserta_avion2 '5','1' ,'2','40','express';
exec inserta_avion2 '6','4' ,'2','100','particular';
select *from avion
order by idavion asc;

--TABLA VUELO
--tampoco se ingresa idvuelo ya que es autoincremental
insert into vuelo (destino,idaeropuerto,hora,fecha,idavion) values ('chile' ,
2,'20','20200910',1);
```

```
insert into vuelo (destino,idaeropuerto,hora,fecha,idavion) values ('brazil' ,
3,'08','20200911',2);
insert into vuelo (destino,idaeropuerto,hora,fecha,idavion) values ('paraguay' ,
4,'19','20200912',3);
insert into vuelo (destino,idaeropuerto,hora,fecha,idavion) values ('uruguay' ,
4,'21','20200913',4);
insert into vuelo (destino,idaeropuerto,hora,fecha,idavion) values ('japón' ,
2,'24','20200914',1);
insert into vuelo (destino,idaeropuerto,hora,fecha,idavion) values ('angola' ,
3,'10','20200905',2);
insert into vuelo (destino,idaeropuerto,hora,fecha,idavion) values ('eeuu' ,
3,'12','20200606',3);
insert into vuelo (destino,idaeropuerto,hora,fecha,idavion) values ('mexico' ,
4,'14','20200907',4);
insert into vuelo (destino,idaeropuerto,hora,fecha,idavion) values ('españa' ,
2,'16','20201010',4);
select * from  vuelo;

--TABLA TARIFA
insert into tarifa(idtarifa,clase,precio,impuesto) values (1,'turista' , 2000,21);
insert into tarifa(idtarifa,clase,precio,impuesto) values (2,'normal' , 1000,21);
insert into tarifa(idtarifa,clase,precio,impuesto) values (3,'vip' , 5000,30);
select * from tarifa;
--TABLA PAIS
insert into pais (nombre) values ('chile');
insert into pais (nombre) values ('brazil');
insert into pais (nombre) values ('paraguay');
insert into pais (nombre) values ('uruguay');
insert into pais (nombre) values ('angola');
insert into pais (nombre) values ('eeuu');
insert into pais (nombre) values ('mexico');
insert into pais (nombre) values ('españa');
select * from pais;
---TABLA ASIENTO
insert into asiento (idasiento,letra,numero) values (1,'A',1);
insert into asiento (idasiento,letra,numero) values (2,'A',2);
insert into asiento (idasiento,letra,numero) values (3,'A',3);
insert into asiento (idasiento,letra,numero) values (4,'A',4);
insert into asiento (idasiento,letra,numero) values (5,'B',1);
insert into asiento (idasiento,letra,numero) values (6,'B',2);
insert into asiento (idasiento,letra,numero) values (7,'B',3);
insert into asiento (idasiento,letra,numero) values (8,'B',4);
insert into asiento (idasiento,letra,numero) values (9,'C',1);
insert into asiento (idasiento,letra,numero) values (10,'C',2);
insert into asiento (idasiento,letra,numero) values (11,'C',3);
insert into asiento (idasiento,letra,numero) values (12,'C',4);
insert into asiento (idasiento,letra,numero) values (13,'D',1);
insert into asiento (idasiento,letra,numero) values (14,'D',2);
insert into asiento (idasiento,letra,numero) values (15,'D',3);
insert into asiento (idasiento,letra,numero) values (16,'D',4);
select *from asiento;

--TABLA PASAJERO
insert into pasajero (dniPasajero,nombre,apellido,fechaNac,idPais,tel,sexo,email) values
(23532341,'santiago','garrido','19850701',1,379400000,'M','santi@hotmail.com');
insert into pasajero (dniPasajero,nombre,apellido,fechaNac,idPais,tel,sexo,email) values
(33412341,'kevin','gomez','19900701',2,3794236745,'M','kevin@hotmail.com');
insert into pasajero (dniPasajero,nombre,apellido,fechaNac,idPais,tel,sexo,email) values
(23464341,'nico','kryvenki','19900701',3,3794341212,'M','nico@hotmail.com');
insert into pasajero (dniPasajero,nombre,apellido,fechaNac,idPais,tel,sexo,email) values
(23232341,'martin','coutinho','19900701',6,379498766234,'M','martin@hotmail.com');
insert into pasajero (dniPasajero,nombre,apellido,fechaNac,idPais,tel,sexo,email) values
(43412341,'ismael','coronel','19900701',7,3794457643,'M','coronel@hotmail.com');
insert into pasajero (dniPasajero,nombre,apellido,fechaNac,idPais,tel,sexo,email) values
(10532332,'maria','gimenez','20001012',8,3794437192,'F','maria@hotmail.com');
```

```
select *from pasajero;

--Uso de transacciones, si se produce un error se volverá a un estado anterior mediante un
roolback, sino terminar la transacción con un commit
begin try
begin tran
--declare @idpago int
--TABLA PAGO
insert into pago (dniPasajero,fecha,nroComprobante,monto) values
(23532341,'21210101',00001,1000);
insert into pago (dniPasajero,fecha,nroComprobante,monto) values
(33412341,'21210101',00002,3000);
insert into pago (dniPasajero,fecha,nroComprobante,monto) values
(31883884,'21210101',00003,5000);
insert into pago (dniPasajero,fecha,nroComprobante,monto) values
(23464341,'21210101',00004,1000);
insert into pago (dniPasajero,fecha,nroComprobante,monto) values
(23232341,'21210101',00005,3000);
insert into pago (dniPasajero,fecha,nroComprobante,monto) values
(43412341,'21210101',00006,5000);
--set @idpago = SCOPE_IDENTITY()
--select * from pago;

--TABLA RESERVA
insert into reserva(fecha,observacion,idVuelo,idPago,idAsiento,idTarifa,dniPasajero)
values('20200101','',2,1,1,1,23532341);
insert into reserva(fecha,observacion,idVuelo,idPago,idAsiento,idTarifa,dniPasajero)
values('20200101','',1,2,2,2,33412341);
insert into reserva(fecha,observacion,idVuelo,idPago,idAsiento,idTarifa,dniPasajero)
values('20200101','',3,3,3,3,23464341);
insert into reserva(fecha,observacion,idVuelo,idPago,idAsiento,idTarifa,dniPasajero)
values('20200101','',4,4,4,3,23232341);
insert into reserva(fecha,observacion,idVuelo,idPago,idAsiento,idTarifa,dniPasajero)
values('20200101','',1,5,5,2,43412341);
insert into reserva(fecha,observacion,idVuelo,idPago,idAsiento,idTarifa,dniPasajero)
values('20200101','',4,6,6,1,10532332);

--select * from reserva;

commit tran
end try
begin catch
  rollback tran
  print 'ERROR: NO SE LOGRÓ CONCRETAR LA TRANSACCIÓN'
end catch
```

Temas investigados y aplicados en el proyecto

Funciones:

Una función es un conjunto de instrucciones SQL que realizan una tarea específica de manera automática. Favorecen a la reutilización del código, acepta entradas en forma de parámetros y devuelve un valor.

Ejemplo:

```
--esta función  sirve para saber cuantos pasajeros hay por país
CREATE FUNCTION cantidadporpais(
	@Valor int)
RETURNS TABLE
AS
RETURN(
	SELECT * FROM pasajero WHERE idPais = @Valor;
select * from cantidadporpais(1);
--muestra la cantidad de pasajeros en el idPais 1
```

	dniPasajero	nombre	apellido	fechaNac	idPais	tel	sexo	email	idReserva
1	23532341	santiago	garrido	1985-07-01 00:00:00.000	1	379400000	M	santi@hotmail.com	1

Procedimientos almacenados:

Un procedimiento almacenado está formado por un conjunto de instrucciones que definen un determinado proceso a ejecutar, puede aceptar parámetros de entrada y devolver un valor o conjunto de resultados

Ejemplo:

```
--este procedimiento realiza la misma búsqueda que la función anterior y es para
demostrar las diferencias que poseen entre si
create procedure cantidadporpais1(
	@valor int) as
SELECT * FROM pasajero WHERE idPais = @Valor;
exec cantidadporpais1 1;
```

Disparadores:

Los TRIGGERS de SQL Server son procedimientos almacenados especiales que se ejecutan automáticamente en respuesta al objeto de la base de datos, la base de datos y los eventos del servidor. SQL Server proporciona tres tipos de disparadores.

Existe 3 tipos de triggers:

1. DML triggers
2. DDL triggers
3. Logon trigger

Los TRIGGERS DML se activan automáticamente en respuesta a eventos DML (INSERT, UPDATE & DELETE).

Los desencadenantes DML se pueden clasificar nuevamente en 2 tipos:
1. FOR (se desencadenan después)
2. INSTEAD OF (se desencadenan antes)
Generalmente son llamados After y Before.

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.

Ejemplo:

```
--Uso de transacciones, si se produce un error se volverá a un estado anterior mediante un
roolback, sino terminar la transacción con un commit
begin try
begin tran
--declare @idpago int

--TABLA PAGO
insert into pago (dniPasajero,fecha,nroComprobante,monto) values
(23532341,'21210101',00001,1000);
insert into pago (dniPasajero,fecha,nroComprobante,monto) values
(33412341,'21210101',00002,3000);
insert into pago (dniPasajero,fecha,nroComprobante,monto) values
(31883884,'21210101',00003,5000);
--set @idpago = SCOPE_IDENTITY()
--select * from pago;

--TABLA RESERVA
insert into reserva(fecha,observacion,idVuelo,idPago,idAsiento,idTarifa,dniPasajero)
values('20200101','',2,1,1,1,23532341);
insert into reserva(fecha,observacion,idVuelo,idPago,idAsiento,idTarifa,dniPasajero)
values('20200101','',1,2,2,2,33412341);
insert into reserva(fecha,observacion,idVuelo,idPago,idAsiento,idTarifa,dniPasajero)
values('20200101','',3,3,3,3,23464341);

--select * from reserva;

commit tran
end try
begin catch
  rollback tran
  print 'ERROR: NO SE LOGRÓ CONCRETAR LA TRANSACCIÓN'
end catch
```

Lo que ocurriría en caso de un error:

```
(1 row affected)

(0 rows affected)
ERROR: NO SE LOGRO CONCRETAR LA TRANSACCIÓN
```

Vistas:

Las vistas nos sirven para proporcionar información a los usuarios de una forma simplificada y personalizada. También las podemos usar como mecanismos de seguridad, ya que de esta forma no permitimos el acceso directo hacia la tabla base. Y además, para proporcionar una interfaz compatibles con versiones anteriores para emular una tabla que haya cambiado.

Ejemplo:

```
--VISTAS
--Vista de Aviones con su cantidad de asientos, su tipo y su aerolínea.
create view Aviones with encryption
as
select avion.idavion,avion.cantasientos,avion.tipo,aerolinea.nombre as 'Aerolinea'
from avion
INNER JOIN aerolinea ON avion.idaerolinea = aerolinea.idaerolinea
select * from Aviones
```

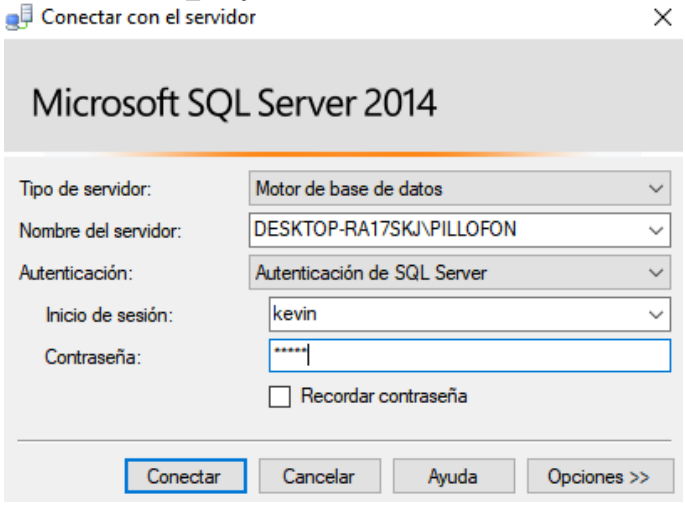
	idavion	cantasientos	tipo	Aerolinea
1	1	50	vip	Fly Bondi
2	2	100	super vip`	Fly Bondi
3	3	100	vip	Air Flight
4	4	100	super vip	Air Flight
5	5	40	express	Aerolínea argentina
6	6	100	particular	Latam airlines

PERMISOS:

Los permisos se organizan de una manera jerarquica con relación con los usuarios intentando dar una mayor seguridad al servidor o base de datos. Éstos usuarios interfieren de una forma mayor o menor en la base datos según los privilegios o roles que éstos cumplen.

Por ej: El usuario Kevin tiene el rol de solo lectura y solamente puede trabajar en la base de datos aeropuerto.

```
create login kevin with password= 'admin'
use aeropuerto
create user kevin for login kevin
alter role db_datareader add member kevin
alter role db_denydatawriter add member kevin
```



INDICES:

Los índices facilitan la recuperación de datos, permitiendo el acceso directo y acelerando las búsquedas, consultas y otras operaciones que optimizan el rendimiento general.SQL Server permite crear dos tipos de índices: 1) agrupados (clustered) y 2) no agrupados (nonclustered).

1) Un INDICE AGRUPADO es similar a una guía telefónica, los registros con el mismo valor de campo se agrupan juntos. Modifican el orden físico de los registros, ordenándolos secuencialmente. Una tabla sólo puede tener UN índice agrupado.

2) Un INDICE NO AGRUPADO es como el índice de un libro, los datos se almacenan en un lugar diferente al del índice, los punteros indican el lugar de almacenamiento de los elementos indizados en la tabla.

Resultados antes de aplicar un nonclustered al atributo nombre de la tabla pasajero

Resultados | Mensajes

	dniPasajero	nombre
1	10532332	maria
2	23232341	martin
3	23464341	nico
4	23532341	santiago
5	33412341	kevin
6	43412341	ismael

Después de aplicar nonclustered

Resultados | Mensajes

	dniPasajero	nombre
1	43412341	ismael
2	33412341	kevin
3	10532332	maria
4	23232341	martin
5	23464341	nico
6	23532341	santiago

Backup y restauración:

Las copias de seguridad de SQL Server proveen una importante solución para proteger datos críticos que están almacenados en bases de datos SQL. Y para minimizar el riego de pérdida de datos, usted necesita asegurarse de que respalda sus bases de datos regularmente tomando en consideración los cambios aplicados a sus datos. Es una buena práctica probar sus copias de seguridad restaurando archivos de copias de seguridad al azar a un ambiente de pruebas y verificar que los archivos no estén corruptos.

Ejemplo:

```
/** BACKUP y RESTORE */

--Backup
BACKUP DATABASE [Aeropuerto]
TO DISK = N'C:\Program Files\Microsoft SQL
Server\MSSQL15.SQLEXPRESS\MSSQL\Backup\Aeropuerto.bak'
WITH NOFORMAT, NOINIT,
NAME = N'Aeropuerto-Full Database Backup', SKIP, NOREWIND, NOUNLOAD,  STATS = 10
GO

--Restore
USE [master]
RESTORE DATABASE [Aeropuerto]
FROM DISK = N'C:\Program Files\Microsoft SQL
Server\MSSQL15.SQLEXPRESS\MSSQL\Backup\Aeropuerto.bak' WITH  FILE = 1,  NOUNLOAD,  STATS = 5
GO
```

Consultas:

```
-- Cantidad de pasajeros por país
select pa.nombre as Pais, count(p.dnipasajero) as 'Cantidad pasajeros'
from pasajero p
inner join pais pa
on p.idpais = pa.idpais
group by pa.nombre
```

	Pais	Cantidad pasajeros
1	brazil	1
2	chile	1
3	eeuu	1
4	españa	1
5	mexico	1
6	paraguay	1

```
-- Detalle de la reserva
select r.idreserva, r.fecha as 'Fecha reserva', concat(p.apellido, ' ' ,p.nombre) as
Pasajero, v.destino, v.fecha as 'Fecha del vuelo', v.hora as 'Hora del vuelvo',
concat(a.letra,' ' ,a.numero) as Asiento, (pa.monto * t.impuesto)/100 + pa.monto + t.precio
as
'Total a pagar'
from reserva r
inner join pasajero p on r.dnipasajero = p.dniPasajero
inner join vuelo v on r.idvuelo = v.idvuelo
inner join asiento a on r.idAsiento = a.idasiento
inner join pago pa on r.idpago = pa.idpago
inner join tarifa t on r.idtarifa = t.idtarifa
order by r.idReserva
```

	idreserva	Fecha reserva	Pasajero	destino	Fecha del vuelo	Hora del vuelvo	Asiento	Total a pagar
1	1	2020-01-01	garrido santiago	brazil	2020-09-11 00:00:00.000	8	A 1	3210
2	2	2020-01-01	gomez kevin	chile	2020-09-10 00:00:00.000	20	A 2	4630
3	3	2020-01-01	krivensky nico	paraguay	2020-09-12 00:00:00.000	19	A 3	11500
4	4	2020-01-01	couti martin	uruguay	2020-09-13 00:00:00.000	21	A 4	6300
5	5	2020-01-01	coronel ismael	chile	2020-09-10 00:00:00.000	20	B 1	4630
6	6	2020-01-01	gimenez maria	uruguay	2020-09-13 00:00:00.000	21	B 2	8050

```
--Detalle de vuelos disponibles
 select v.idvuelo, v.destino, v.fecha, v.hora, a.tipo, ae.nombre as Aerolinea
 from vuelo v
 inner join avion a
 on v.idavion = a.idavion
 inner join aerolinea ae
 on a.idaerolinea = ae.idaeroline
```

	idvuelo	destino	fecha	hora	tipo	Aerolinea
2	2	brazil	2020-09-11 00:00:00.000	8	super vip`	Fly Bondi
3	3	paraguay	2020-09-12 00:00:00.000	19	vip	Air Flight
4	4	uruguay	2020-09-13 00:00:00.000	21	super vip	Air Flight
5	5	japón	2020-09-14 00:00:00.000	24	vip	Fly Bondi
6	6	angola	2020-09-05 00:00:00.000	10	super vip`	Fly Bondi
7	7	eeuu	2020-06-06 00:00:00.000	12	vip	Air Flight
8	8	mexico	2020-09-07 00:00:00.000	14	super vip	Air Flight
9	10	españa	2020-10-10 00:00:00.000	16	super vip	Air Flight