# Base de Datos I

II. DISEÑO LOGICO DE BASE DE DATOS Y MODELAJE CONCEPTUAL

# 4.1. Componentes de una Entidad en BD.

El **modelo de base de datos relacional** fue creado en 1970 por Edgar Frank Codd desde los laboratorios de IBM, para mejorar la forma en que se consultaban los datos. Para comprender mejor cómo funciona y por qué es tan importante en la era digital, te explicaremos algunos conceptos fundamentales.

### ¿Qué es una base de datos relacional?

Una **base de datos relacional** es un conjunto de información que se organiza en tablas de columnas y filas (registros). Se llama relacional, justamente, porque los datos se ordenan en relaciones predefinidas. Esas relaciones son **conexiones lógicas** entre las tablas que interactúan al conectar diferentes aspectos de ellas.

Si bien era muy complejo para la época, este modelo se basa en el principio de guardar los datos como **relaciones (tablas)**.
Las tablas están formadas por *filas*, que puedes ver en forma vertical (registros), y también por *columnas*, que puedes ver en dirección horizontal.

Las B**ases de Datos relacionales** están diseñadas para recopilar grandes volúmenes de datos. permiten que muchos usuarios accedan y consulten los datos de forma rápida y segura, esto se debe a que cumplen con una serie de **Restricciones** que garantizan una robustez de la Base de Datos y es fundamental conocer cuales son y como se aplican.

# 4.1. Componentes de una Entidad en BD.

Restricciones SQL: NOT NULL; UNIQUE; PRIMARY KEY; FOREIGN KEY; CHECK; DEFAULT

1. NOT NULL: Se usa una restricción NOT NULL en SQL para evitar insertar valores NULL en la columna especificada

```
USE SQLShackDemo
GO
CREATE TABLE ConstraintDemo1
(
        ID INT NOT NULL,
    Name VARCHAR(50) NULL
)

INSERT INTO ConstraintDemo1 ([ID],[NAME]) VALUES (1,'Ali')
GO
INSERT INTO ConstraintDemo1 ([ID]) VALUES (2)
GO
INSERT INTO ConstraintDemo1 ([NAME]) VALUES ('Fadi')
GO
```

Pequeño caso practico: Suponga que necesitamos evitar que la columna Nombre de la tabla anterior acepte valores NULL después de crear la tabla, hay dos caminos que podríamos tomar.

El primero:

Utilizando la instrucción T-SQL ALTER TABLE, tendríamos que borrar todos los registro para realizar la alteración de campo.

La segunda:

Hacer una actualización dela restricción y luego ejecutar T-SQL ALTER TABLE, no tendríamos que borrar los registros.

# 4.1. Componentes de una Entidad en BD.

Restricciones SQL: NOT NULL; UNIQUE; PRIMARY KEY; FOREIGN KEY; CHECK; DEFAULT

1. NOT NULL

a)

```
ALTER TABLE ConstraintDemo1 ALTER COLUMN [Name] VARCHAR(50) NOT NULL
```

b)

```
UPDATE ConstraintDemo1 SET [Name]='' WHERE [Name] IS NULL
```

+

```
ALTER TABLE ConstraintDemo1 ALTER COLUMN [Name] VARCHAR(50) NOT NULL
```

A continuación validaremos que e efecto lo explicado se puede concretar desde SQL, apoyándonos en las instrucciones previamente elaboradoras

# 4.1. Componentes de una Entidad en BD.

Restricciones SQL: NOT NULL; UNIQUE; PRIMARY KEY; FOREIGN KEY; CHECK; DEFAULT

2. UNIQUE : La restricción UNIQUE en SQL se utiliza para garantizar que no se inserten valores duplicados en una columna específica o combinación de columnas que participen en la restricción UNIQUE y no formen parte de la CLAVE PRIMARIA

Pequeño caso practico: Creemos una tabla con dos columnas, ID y Nombre. La columna de ID no puede contener valores duplicados debido a la restricción ÚNICA especificada con la definición de columna. No hay restricciones definidas en la columna Nombre.

```
USE SQLShackDemo
GO
CREATE TABLE ConstraintDemo2
(
     ID INT UNIQUE,
   Name VARCHAR(50) NULL
)

INSERT INTO ConstraintDemo2 ([ID],[NAME]) VALUES (1,'Ali')
GO
INSERT INTO ConstraintDemo2 ([ID],[NAME]) VALUES (2,'Ali')
GO
INSERT INTO ConstraintDemo2 ([ID],[NAME]) VALUES (NULL,'Adel')
GO
INSERT INTO ConstraintDemo2 ([ID],[NAME]) VALUES (1,'Faris')
GO
```

- Los dos primeros registros se insertarán correctamente, sin las restricciones que impidan los valores duplicados de la columna Nombre.
- El tercer registro también se insertará correctamente, ya que la única columna de ID solo permite un valor NULL.
- La última instrucción INSERT fallará, ya que la columna ID no permite valores duplicados y el valor ID proporcionado ya está insertado en esa columna

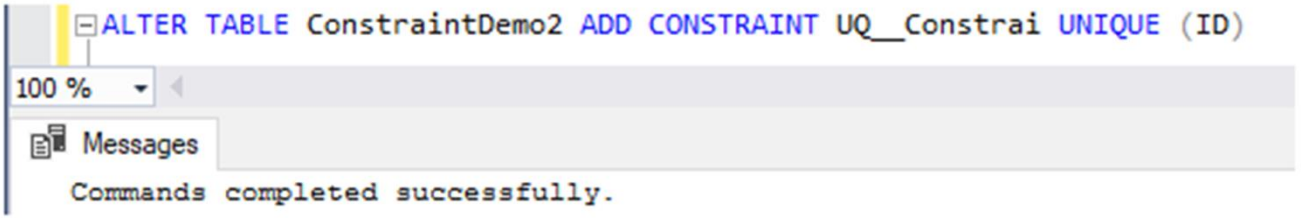
# 4.1. Componentes de una Entidad en BD.

Restricciones SQL: NOT NULL; UNIQUE; PRIMARY KEY; FOREIGN KEY; CHECK; DEFAULT

Para añadir la restricción ÚNICA, se tiene la opción de eliminar o modificar los valores duplicados. En nuestro caso, actualizaremos el segundo valor de ID duplicado usando la siguiente instrucción de ACTUALIZAR:

```
UPDATE [SQLShackDemo].[dbo].[ConstraintDemo2] SET ID =3 WHERE NAME='FARIS'
```

Ahora, la restricción UNIQUE en SQL se puede agregar sin error a la columna ID como se ve líneas abajo:




La clave ÚNICA se puede ver utilizando SQL Server Management Studio, expandiendo el nodo de las Claves debajo de la tabla seleccionada. Usted también puede ver el índice creado automáticamente que se utiliza para garantizar la unicidad de los valores de columna. Tenga en cuenta que no podrá eliminar ese índice sin eliminar primero la restricción ÚNICA:

# 4.1. Componentes de una Entidad en BD.

Restricciones SQL: NOT NULL; UNIQUE; PRIMARY KEY; FOREIGN KEY; CHECK; DEFAULT

1. PRIMARY KEY : La restricción PRIMARY KEY de SQL se combina entre las restricciones UNIQUE y SQL NOT NULL, donde la columna o el conjunto de columnas que participan en PRIMARY KEY no pueden aceptar el valor NULL.

Pequeño caso practico: la tabla tiene la identificación y el nombre. El ID se define como una CLAVE PRIMARIA para esa tabla, la que se utiliza para identificar cada fila en esa tabla, no sepueden inserten valores NULL o duplicados en esa columna ID.

```
USE SQLShackDemo
GO
CREATE TABLE ConstraintDemo3
(
     ID INT PRIMARY KEY,
   Name VARCHAR(50) NULL
)

INSERT INTO ConstraintDemo3 ([ID],[NAME]) VALUES (1,'John')
GO
INSERT INTO ConstraintDemo3 ([NAME]) VALUES ('Fadi')
GO
INSERT INTO ConstraintDemo3 ([ID],[NAME]) VALUES (1,'Saeed')
GO
```

- Usted Verá que el primer registro se insertará exitosamente y correctamente ya que los valores de ID y Nombre son válidos
- La segunda operación de inserción fallará, ya que la columna ID es obligatoria y no puede ser NULL, ya que la columna ID es la CLAVE PRIMARIA SQL.
- La última instrucción INSERT también fallará ya que el valor de ID proporcionado ya existe y los valores duplicados ya no están permitidos en la CLAVE PRIMARIA,

# 4.1. Componentes de una Entidad en BD.

Restricciones SQL: NOT NULL; UNIQUE; PRIMARY KEY; FOREIGN KEY; CHECK; DEFAULT

1. PRIMARY KEY : La restricción PRIMARY KEY de SQL se combina entre las restricciones UNIQUE y SQL NOT NULL, donde la columna o el conjunto de columnas que participan en PRIMARY KEY no pueden aceptar el valor NULL.

Pequeño caso practico: la tabla tiene la identificación y el nombre. El ID se define como una CLAVE PRIMARIA para esa tabla, la que se utiliza para identificar cada fila en esa tabla, no se pueden inserten valores NULL o duplicados en esa columna ID.

```
USE SQLShackDemo
GO
CREATE TABLE ConstraintDemo3
(
     ID INT PRIMARY KEY,
   Name VARCHAR(50) NULL
)

INSERT INTO ConstraintDemo3 ([ID],[NAME]) VALUES (1,'John')
GO
INSERT INTO ConstraintDemo3 ([NAME]) VALUES ('Fadi')
GO
INSERT INTO ConstraintDemo3 ([ID],[NAME]) VALUES (1,'Saeed')
GO
```

- Usted Verá que el primer registro se insertará exitosamente y correctamente ya que los valores de ID y Nombre son válidos
- La segunda operación de inserción fallará, ya que la columna ID es obligatoria y no puede ser NULL, ya que la columna ID es la CLAVE PRIMARIA SQL.
- La última instrucción INSERT también fallará ya que el valor de ID proporcionado ya existe y los valores duplicados ya no están permitidos en la CLAVE PRIMARIA,

# 4.1. Componentes de una Entidad en BD.

Restricciones SQL: NOT NULL; UNIQUE; PRIMARY KEY; FOREIGN KEY; CHECK; DEFAULT

La restricción CHECK en SQL se utiliza para asegurar que los valores de una columna cumplan con una condición específica. Esta restricción permite definir una expresión lógica que los datos

```
CREATE TABLE VENDEDOR (
    Id INT IDENTITY(1,1) PRIMARY KEY,                -- Campo Id como Primary Key con incremento automático (Identity)
    Nombre NVARCHAR(100) NOT NULL,                   -- Campo Nombre como NOT NULL
    Salario DECIMAL(10, 2) CHECK (Salario > 550 AND Salario < 5500)  --Rerestricción CHECK para que esté entre 550 y
5500
);
INSERT INTO VENDEDOR (Nombre, Salario) VALUES ('Juan Perez', 1200.00);

INSERT INTO VENDEDOR (Nombre, Salario) VALUES ('Maria Lopez', 4500.50);

INSERT INTO VENDEDOR (Nombre, Salario) VALUES ('Carlos Sanchez', 550.00);  -- Esto fallará porque 550 no es mayor que
550
```

Id: Es la clave primaria (PRIMARY KEY), y utiliza la función IDENTITY(1,1) para que se incremente automáticamente con cada nuevo registro.
Nombre: Es un campo de texto (NVARCHAR(100)) que no puede ser nulo (NOT NULL).
Salario: Es un campo numérico (DECIMAL(10, 2)) que debe ser mayor a 0, gracias a la restricción CHECK.

# 4.1. Componentes de una Entidad en BD.

Restricciones SQL: NOT NULL; UNIQUE; PRIMARY KEY; FOREIGN KEY; CHECK; DEFAULT

La restricción DEFAULT en SQL se utiliza para asignar un valor predeterminado a una columna cuando no se proporciona uno explícito al insertar un nuevo registro. Esto asegura que la columna siempre tenga un valor

```
CREATE TABLE ESTUDIANTES (
    Id INT IDENTITY(1,1) PRIMARY KEY,          -- Campo Id como Primary Key con incremento
automático (Identity)
    Nombre NVARCHAR(100) NOT NULL,             -- Campo Nombre como NOT NULL
    Carrera NVARCHAR(100) NULL,                -- Campo Carrera como NULL
    Activo BIT DEFAULT 1                       -- Campo Activo con restricción DEFAULT que
se establece en true (1)
);
```

•**Id**: Es la clave primaria (PRIMARY KEY), y utiliza la función IDENTITY(1,1) para que se incremente automáticamente con cada nuevo registro.
•**Nombre**: Es un campo de texto (NVARCHAR(100)) que no puede ser nulo (NOT NULL).
•**Carrera**: Es un campo de texto (NVARCHAR(100)) que puede aceptar valores nulos (NULL).
•**Activo**: Es un campo de tipo BIT que tiene un valor por defecto de 1 (DEFAULT 1), lo que indica que el estudiante está activo por defecto.

# 4.2. Lenguaje de Manipulación de Datos

| id | first_name | last_name | department | salary |
|---|---|---|---|---|
| 1 | Paul | Garrix | Corporate | 3,547.25 |
| 2 | Astrid | Fox | Private Individuals | 2,845.56 |
| 3 | Matthias | Johnson | Private Individuals | 3,009.41 |
| 4 | Lucy | Patterson | Private Individuals | 3,547.25 |
| 5 | Tom | Page | Corporate | 5,974.41 |
| 6 | Claudia | Conte | Corporate | 4,714.12 |
| 7 | Walter | Deer | Private Individuals | 3,547.25 |
| 8 | Stephanie | Marx | Corporate | 2,894.51 |
| 9 | Luca | Pavarotti | Private Individuals | 4,123.45 |
| 10 | Victoria | Pollock | Corporate | 4,789.53 |

Como cualquier tabla, tiene un nombre **Empleados**. Cada tabla tiene columnas que también tienen nombres. Y describen los datos contiene. Consulta que la crea Aquí

Las columnas y los datos de la tabla anterior son:

- id - El ID único del empleado y la clave primaria de la tabla.
- first_name - El nombre del empleado.
- last_name - El apellido del empleado.
- department - El departamento del empleado.
- salary - El salario mensual del empleado, en USD.

Esto esto nos indica que esta tabla es una lista de los empleados de una empresa y sus salarios. También hay datos sobre los departamentos de los empleados.

# 4.2. Lenguaje de Manipulación de Datos

| employee_id | q1_2022 | q2_2022 | q3_2022 | q4_2022 |
|---|---|---|---|---|
| 8 | 3,471.41 | 14,789.25 | 3,478.34 | 1,254.23 |
| 4 | 5,417.81 | 12,846.23 | 8,741.54 | 3,589.99 |
| 10 | 1,547.52 | 1,269.66 | 1,478.65 | 2,474.33 |
| 1 | 8,715.55 | 8,465.65 | 24,747.82 | 3,514.36 |
| 3 | 12,774.51 | 24,784.31 | 12,223.34 | 8,451.51 |
| 2 | 4,989.23 | 5,103.22 | 4,897.98 | 5,322.05 |
| 7 | 18,415.66 | 15,279.37 | 14,634.44 | 14,445.12 |
| 6 | 2,498.63 | 8,741.45 | 3,997.65 | 2,497.21 |
| 5 | 6,349.74 | 7,555.55 | 6,944.35 | 7,788.01 |
| 9 | 4,485.36 | 4,101.50 | 8,787.45 | 7,648.90 |

La tabla se llama **VentasCuatriMes**. Se muestra a continuación, y la consulta para crearla está aquí.

Las columnas son:

- employee_id - El ID único del empleado. Además, una clave externa que hace referencia a la columna id de la tabla employees.
- q1_2022 - Las ventas realizadas por ese empleado en el primer trimestre de 2022.
- q2_2022 - Las ventas realizadas por ese empleado en el segundo trimestre de 2022.
- q3_2022 - Las ventas realizadas por ese empleado en el tercer trimestre de 2022.
- q4_2022 - Las ventas realizadas por ese empleado en el cuarto trimestre de 2022.

Esta tabla es una lista de las ventas de cada trimestre realizadas por cada empleado mostrado en la primera tabla.

# 4.2. Lenguaje de Manipulación de Datos

```sql
--(1)
-- Creación de la tabla VENDEDORES
CREATE TABLE VENDEDORES (
    id INT IDENTITY(1,1) PRIMARY KEY,
    first_name NVARCHAR(50),
    last_name NVARCHAR(50),
    departament NVARCHAR(50),
    salary MONEY
);

-- Inserción de 12 vendedores
INSERT INTO VENDEDORES (first_name, last_name, departament, salary)
VALUES
('John', 'Doe', 'Electronics', 50000.00),
('Jane', 'Smith', 'Clothing', 45000.00),
('Mike', 'Johnson', 'Furniture', 48000.00),
('Sara', 'Williams', 'Toys', 42000.00),
('David', 'Brown', 'Sports', 53000.00),
('Emily', 'Davis', 'Automotive', 49000.00),
('Chris', 'Miller', 'Home Appliances', 47000.00),
('Anna', 'Wilson', 'Books', 43000.00),
('James', 'Moore', 'Music', 44000.00),
('Linda', 'Taylor', 'Garden', 46000.00),
('Robert', 'Anderson', 'Grocery', 52000.00),
('Karen', 'Thomas', 'Beauty', 51000.00);
```

```sql
-- (3)
-- Los valores son aleatorios para simular las ventas cuatrimestrales
SELECT * FROM VENTASCUATRIMES
ORDER BY NEWID();


--(4)
-- Crear la relación entre id de VENDEDORES y employee_Id de VENTASCUATRIMES
ALTER TABLE VENTASCUATRIMES
ADD CONSTRAINT FK_VentasCuatrimes_EmployeeId
FOREIGN KEY (employee_Id)
REFERENCES VENDEDORES(id)
ON UPDATE CASCADE;
```

```sql
---(2)
-- Creación de la tabla VENTASCUATRIMES
CREATE TABLE VENTASCUATRIMES (
    employee_Id INT,
    Q1_2023 MONEY,
    Q2_2023 MONEY,
    Q3_2023 MONEY,
    Q4_2023 MONEY,
    CONSTRAINT FK_VentasCuatrimes_EmployeeId FOREIGN KEY (employee_Id)
    REFERENCES VENDEDORES(id)
    ON UPDATE CASCADE
);
-- Inserción de 24 registros en la tabla VENTASCUATRIMES (2 por cada vendedor)

INSERT INTO VENTASCUATRIMES (employee_Id, Q1_2023, Q2_2023, Q3_2023, Q4_2023)
VALUES
(1, 15000.00, 20000.00, 18000.00, 21000.00),
(1, 16000.00, 21000.00, 19000.00, 22000.00),
(2, 14000.00, 19000.00, 17000.00, 20000.00),
(2, 13000.00, 18000.00, 16000.00, 21000.00),
(3, 18000.00, 22000.00, 20000.00, 23000.00),
(3, 17000.00, 21000.00, 19000.00, 22000.00),
(4, 12000.00, 17000.00, 15000.00, 18000.00),
(4, 13000.00, 18000.00, 16000.00, 19000.00),
(5, 19000.00, 24000.00, 22000.00, 25000.00),
(5, 20000.00, 25000.00, 23000.00, 26000.00),
(6, 15000.00, 20000.00, 18000.00, 21000.00),
(6, 16000.00, 21000.00, 19000.00, 22000.00),
(7, 14000.00, 19000.00, 17000.00, 20000.00),
(7, 13000.00, 18000.00, 16000.00, 19000.00),
(8, 18000.00, 23000.00, 21000.00, 24000.00),
(8, 19000.00, 24000.00, 22000.00, 25000.00),
(9, 15000.00, 20000.00, 18000.00, 21000.00),
(9, 16000.00, 21000.00, 19000.00, 22000.00),
(10, 12000.00, 17000.00, 15000.00, 18000.00),
(10, 13000.00, 18000.00, 16000.00, 19000.00),
(11, 19000.00, 24000.00, 22000.00, 25000.00),
(11, 20000.00, 25000.00, 23000.00, 26000.00),
(12, 14000.00, 19000.00, 17000.00, 20000.00),
(12, 13000.00, 18000.00, 16000.00, 19000.00);
```

# 4.2. Lenguaje de Manipulación de Datos

Practica en clase: Construya la Tabla a la que hace referencia cada restricción de FOREIGN KEY y cerciorese en el SSMS que se puede ver la correcta relacion entre las llaves de ambas Tablas

```
--Relación entre la tabla aeropuerto país
ALTER TABLE aeropuerto
ADD CONSTRAINT FK_aeropuerto_pais
FOREIGN KEY(idpais) REFERENCES pais (idpais)
go
```

```
--Relación entre la tabla pago y pasajero
ALTER TABLE pago
ADD CONSTRAINT FK_pago_pasajero
FOREIGN KEY (idpasajero) REFERENCES pasajero (idpasajero)
go
```

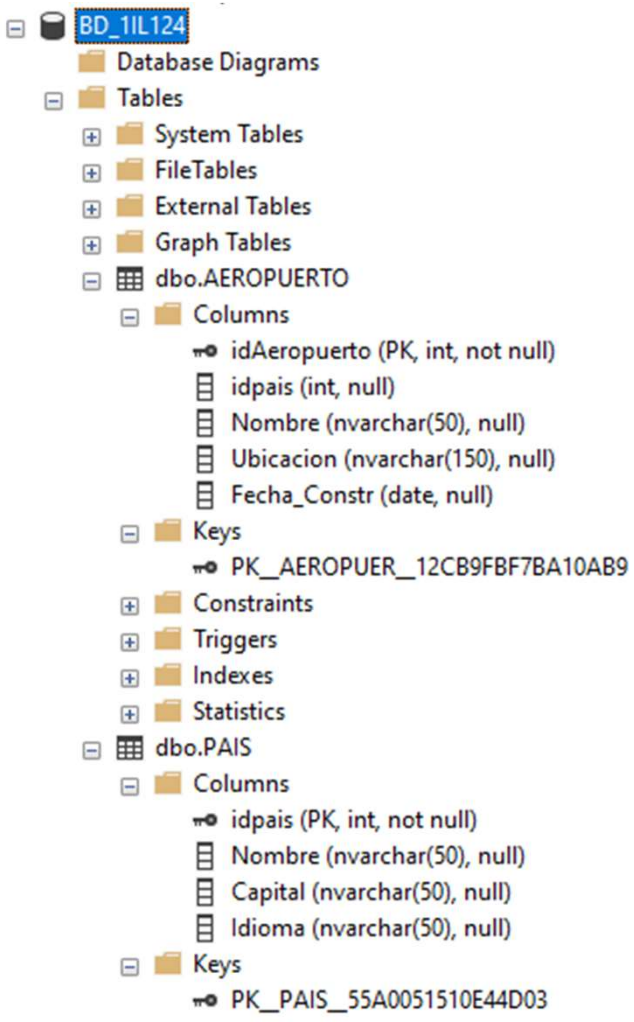
```
--Relación entre la tabla pago y reserva
ALTER TABLE pago
ADD CONSTRAINT fk_pago_reserva
FOREIGN KEY (idreserva) REFERENCES reserva(idreserva)
go
```

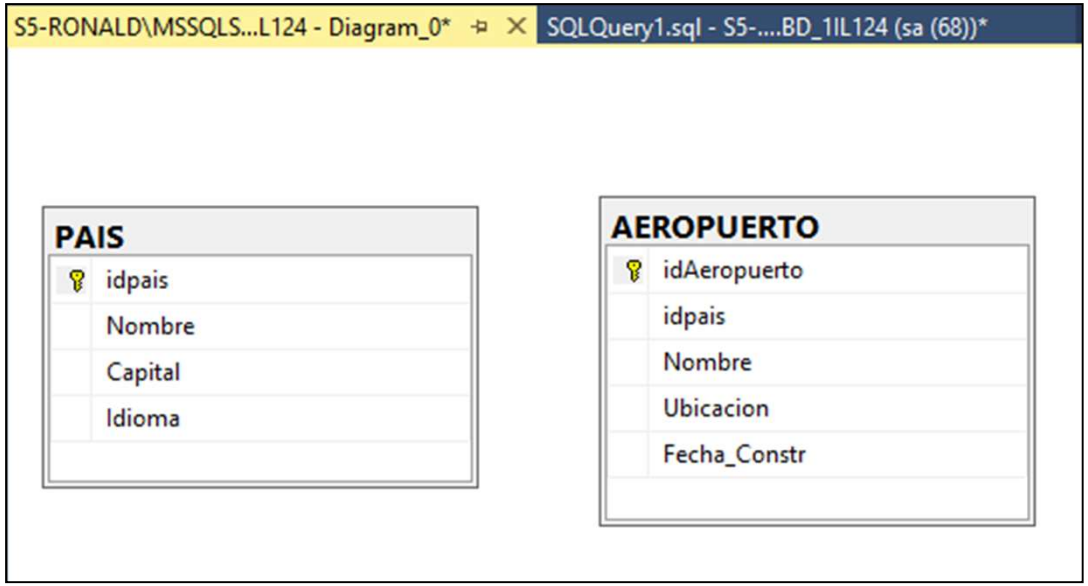
```
USE BD_1IL124
CREATE TABLE PAIS (
    idpais INT IDENTITY(1,1) PRIMARY KEY,
    Nombre NVARCHAR(50),
    Capital NVARCHAR(50),
    Idioma NVARCHAR(50),
 );
```

```
 USE BD_1IL124
CREATE TABLE AEROPUERTO (
    idAeropuerto INT PRIMARY KEY,
    idpais INT,
Nombre NVARCHAR(50),
    Ubicacion NVARCHAR(150),
    Fecha_Constr DATE,
 );

 ALTER TABLE AEROPUERTO
 ADD CONSTRAINT FK_aeropuerto_pais
 FOREIGN KEY(idpais) REFERENCES PAIS (idpais)
 GO
```

# 4.2. Lenguaje de Manipulación de Datos



# 4.2. Lenguaje de Manipulación de Datos

- dbo.AEROPUERTO
  - Columns
    - idAeropuerto (PK, int, not null)
    - idpais (FK, int, null)
    - Nombre (nvarchar(50), null)
    - Ubicacion (nvarchar(150), null)
    - Fecha_Constr (date, null)
  - Keys
    - PK__AEROPUER__12CB9FBF7BA10AB9
    - FK_aeropuerto_pais
  - Constraints
  - Triggers
  - Indexes
  - Statistics
- dbo.PAIS
  - Columns
    - idpais (PK, int, not null)
    - Nombre (nvarchar(50), null)
    - Capital (nvarchar(50), null)
    - Idioma (nvarchar(50), null)
  - Keys
    - PK__PAIS__55A0051510E44D03