Anotaciones de Repaso – SQL

SQL : Conceptos Basicos – Partes de una tabla - Tipos de Lenguajes para tablas (DDL, DML, etc) – Sentencias / Sintaxis / Comandos Simples – Sintaxis de JOIN

(tratar de prestar atención, mayoritariamente a las llaves, que hacen los tipos de lenguaje, y la practica base y join)

Conceptos Basicos:

*¿Qué entendemos por una **Base de Datos**? ¿Qué es?

Se puede definir a una base de datos, en su forma mas simple, como un conjunto de información relacionada que se encuentra agrupada o estructurada.

Basicamente, una base de datos es una serie de datos organizados y relacionados entre sí

o también, puede ser

un programa capaz de almacenar una gran cantidad de información (datos), relacionados y estructurados en orden.

Las bases de datos pueden ser consultadas, modificadas, procesadas, analizadas, etc, a través de un Sistema De Gestión (o Gestor) de Base de Datos (Por ejemplo, SQL server)

*¿Qué es una **entidad**?

Dentro de una base de datos, una entidad es un objeto que representa una "cosa" u "objeto" del mundo real. Pueden ser concretos o abstractos, como una persona o una fecha respectivamente. Se distinguen de otros objetos mediante **atributos**

*¿Qué son los **atributos**?

Los atributos, en una base de datos, son características, propiedades o valores de las entidades las describen en detalle (son los valores de las entidades, por ejemplo: Id, Nombre, edad, domicilio, etc.

Partes de una tabla

*¿Cuáles son las entidades principales que componen una tabla de una base de datos?

Debemos recordar que una tabla siempre va a tener una Clave o Llave Primaria (PK o Primary Key), y una Clave o Llave Foránea (FK o Foreign Key)

-Una clave o llave primaria es un campo único, que no se repite, que sirve para identificar los registros de una tabla de manera unívoca, y además para hacer relaciones (uno a uno o 1-1, uno a muchos o 1-N, Y muchos a muchos o N-N, entre tablas)

Sin estas llaves la información de las tablas seria siempre repetida, y no tendría sentido. Debe haber una PK para que una tabla califique como relacional

Solo puede haber 1 llave primaria por tabla

Por otro lado, una Clave o llave foránea es, simplemente, un campo de una tabla que se corresponde con la clave primaria de otra tabla. Basicamente, una clave foránea sirve para indicar como están relacionadas las tablas.

Si una clave foránea contiene un valor, ese valor se refiere a un registro que ya existe en la tabla con la que se relaciona

Una tabla puede tener relaciones con varias tablas a través de distintas claves foráneas.

*Que es la **cardinalidad** de una tabla?

La cardinalidad es la cantidad de elementos de una entidad que puede asociarse a un elemento de la otra entidad relacionada (básicamente, la cantidad de "cosas" con las que interactua una entidad con otra)

Por ejemplo

Un profesor puede dar n materias y, una materia es dada por 1 profesor.

Un profesor puede dar n materias y, una materia es dada por n profesores.

Un país tiene 1 capital y, una capital pertenece a 1 país.

Tipos de lenguajes

Sigla	Nombre	Descripción	Instrucciones básicas
DDL	Data Definition Language	Permite crear y administrar las bases de datos y sus objetos	CREATE DROP ALTER
DML	Data Manipulation Language	Permite obtener, insertar, modificar y eliminar información de la BD	SELECT INSERT UPDATE DELETE
DCL	Data Control Language	Permite manejar la seguridad de los objetos de la BD	GRANT DENY REVOKE

(Tipicamente, en las consultas con **select, from, group by, order by,** etc, estamos usando lenguaje DML)

Sentencias – Sintaxis Simple

Empezemos por lo básico : **Para crear una tabla**, debemos usar el nombre de la tabla entre paréntesis, y después, todo entre paréntesis, el nombre del campo, tipo de dato, opcional si puede ser not null, etc

Por ejemplo

```
--1--Crear la Tabla T_Pais con los siguientes atributos:
-- Id_Pais    Entero, Obligatorio, Autoincremental
-- NombrePais Cadena de 30, Obligatorio
-- FechaAlta  Fecha y Hora, Obligatorio
-- Activo     Permite dos estados 1 o 0, Obligatorio

create table T_Pais (
      Id_Pais int not null identity(1,1),
    NombrePais varchar(30) not null,
    FechaAlta datetime not null,
    Activo bit not null
)
```

Para **modificar una tabla**, usamos el comando **alter**

Ejemplo:

```
--2--Modificar el campo Id_Pais para que sea Clave primaria.

alter table T_Pais
add Primary Key (Id_Pais)
```

Para **eliminar una tabla**, usamos el comando **drop**

Ejemplo:

```
--3--Eliminar la tabla T_Pais de la Base de Datos.
drop table T_Pais
```

Sintaxis JOIN

La clausula/comando JOIN sirve para combinar registros de una o mas tablas de la base de datos.

Teniamos 4 tipos principales de join:

- join: devuelve los registros que están en las dos tablas relacionadas por el criterio de combinación

```
select *
from Customers c
join Orders o
on c.CustomerID = o.CustomerID
```

(RECORDAR : Para usar cualquier join, SI O SI tengo que usarlo con el comando **on** con las claves foráneas o primarias de las tablas que quiera relacionar, en este caso tanto Customers como Orders tienen una Clave (primaria para Customers y foránea para Orders) que permiten relacionarlas entre las 2

- left outer join: devuelve los registros que están en la tabla a la izquierda del join , y la de la derecha, solo los registros que están dentro del criterio de combinación (la clave). Nos trae todos los registros compatibles de la tabla izquierda, y si la derecha no los tiene, los trae como NULL

```
select *
from Customers c
left outer join Orders o
on c.CustomerID = o.CustomerID
```

- right outer join: exactamente lo mismo que el anterior, pero priorizando a la tabla que esta a la derecha del join

- full outer join/full join: devuelve **todos** los campos de la tabla de la izquierda y de la derecha, mas los que están en null de ambos incluso.

```
select *
from Products p
full outer join Suppliers s
on p.SupplierID = s.SupplierID
```

Ejemplos y Anotaciones útiles

```
select  c.CustomerID, count (c.CustomerID) -- Cada vez que tenga alguno de estos operandos
from dbo.Customers c                        -- (sum(), count(), avg(), Debe haber un GROUP BY)
group by c.CustomerID --
```

```sql
---- mostrar un listado con los productos mas vendidos por vendedor

select e.FirstName, e.LastName
from dbo.Employees e   ----- las e, que aca puse en la tabla son alias,
----me ayudan a conseguir rapidamente los campos de dicha tabla llamando a la letra, que en este caso estan arriba
```

```sql
----cual es el nombre del cliente que mas ha gastado en envìo de ordenes

select ContactName, o.Freight as valor
from dbo.Customers c
join dbo.Orders o
on o.CustomerID = c.CustomerID
order by valor desc
```

```sql
----listar los nombres de los clientes que han gastado mas de 1000 en envio de ordenes

select c.CustomerID, sum(v.Freight) as valor
from Customers c
join Orders v
on v.CustomerID = c.CustomerID
group by c.CustomerID
having sum(v.Freight) > 1000
```

```sql
--7 Listar cuanto ha vendido cada empleado

select e.EmployeeID,e.LastName apellido, sum(od.Quantity * od.UnitPrice) Ventas
from Employees e join Orders o
on e.EmployeeID = o.EmployeeID
join [Order Details] od
on o.OrderID = od.OrderID
group by e.EmployeeID,e.LastName
```