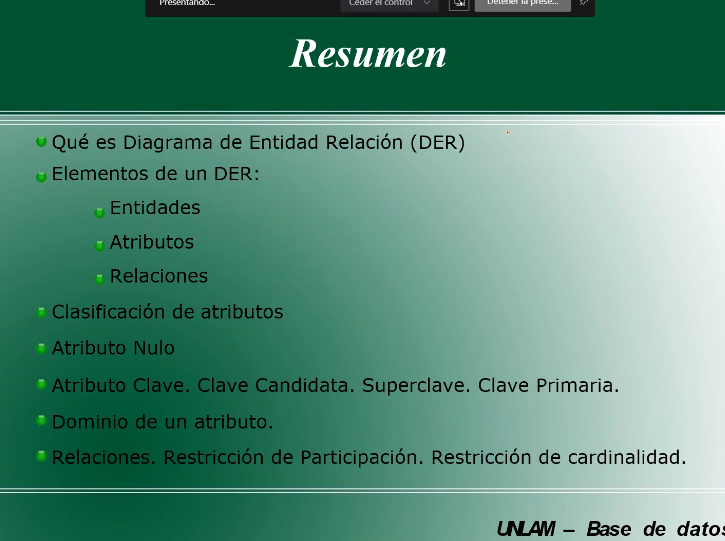
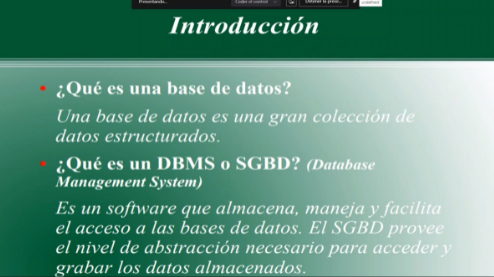
**BASE DE DATOS**





están almacenados en una estructura, ordenados.



dos grupos: relacionales (SQL), no relacionales (no SQL). las más popus son las relacionales. Ejemplo: MySql, Microsoft SQL, etc.

Relacionales: su principal objeto es relación, su principal lenguaje es SQL

existe otras bases de (las no SQL); tienen dif clasificaciones -> tabulares, orientadas a obj, etc. Ya que su principal objeto no es la relación. Ese conjunto tiene dif caract: act específicas.

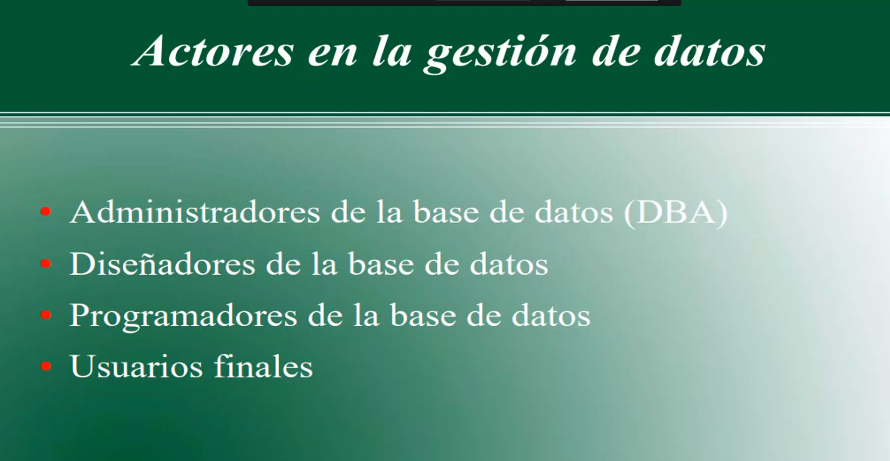
**- Sistema de gestión, tienen funciones principales:**



1- Tengo que saber qué grabé por si cambié la dirección.

2 – Mantenimiento de esquemas: las bases de datos tienen una estructura donde se almacenan. Defino la estructura para ese cliente, donde sus datos están básicamente estructurados.

3 – Manejo de transacciones: Recuperación: Vamos a tener muchas operaciones que van a estar corriendo al mismo tiempo. Ya que el motor, tiene que poder controlar esas transacciones. Voy a tener sesiones recurrentes que van a permitir a muchos, puedan visualizar esa información, no tiene que tener inconsistencias.



**Actores:**

**DBA**: Los que intervienen en la base datos (DBA): Los actores principales de los cuales se van a encargar el mantenimiento de la base de datos. (Seguridad, backups, performance adecuada, etc.)

**Diseñadores de las bases de datos:** Qué y cómo se van a guardar los datos. Comienza a entender el flujo de datos que va a tener esa bd con los datos que se van teniendo.

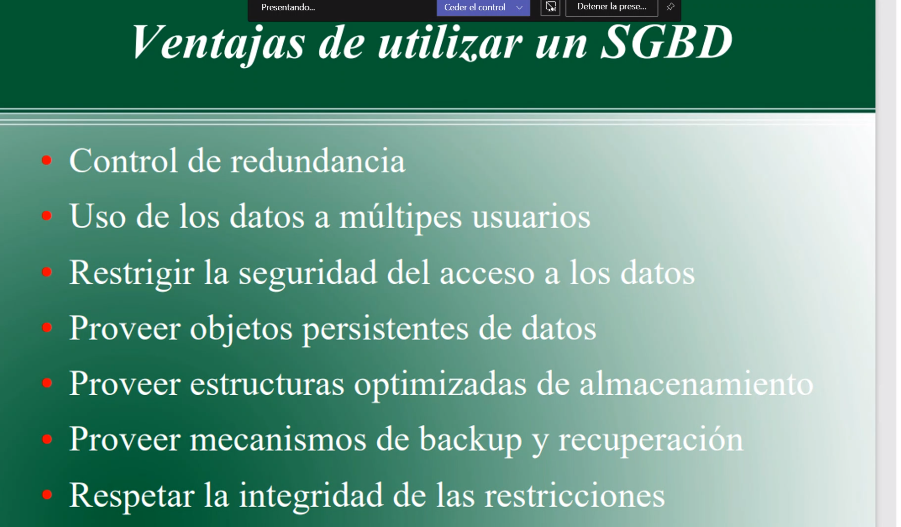
**Programadores:** son los que codifican los procesos, funciones que van a tener la bd para su ejecución. Ej: baja de un cliente. No estamos suplantando LA CAPA DE LA SUPERFICIE DE LA APP.

- Hasta donde desarrollo en la base de datos: Siempre conviene generar los procesos muyyy simples.

- Hasta donde desarrollo la app: si es algo más complejo, se utiliza un servidor intermedio, y no insuma recursos de la bd.

**Usuarios finales:** los que usan la bd. Son nomás de consultas, no van a modificar nada. No pueden modificar los datos, sería una infracción de seguridad. App gestión de clientes: la única manera que puede cambiar los datos.

**¿Qué pasa si se cambias los datos en general?**



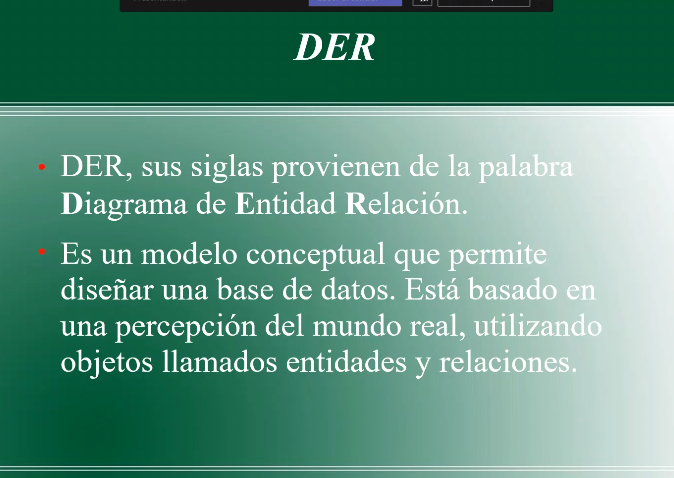
- **Control de redundancia**: Va a tener datos estructurados, para no buscar mil veces las cosas. Hay que tener backups para no perder los datos.

- **Respetar la int**: A través de los datos, se puede controlar la info

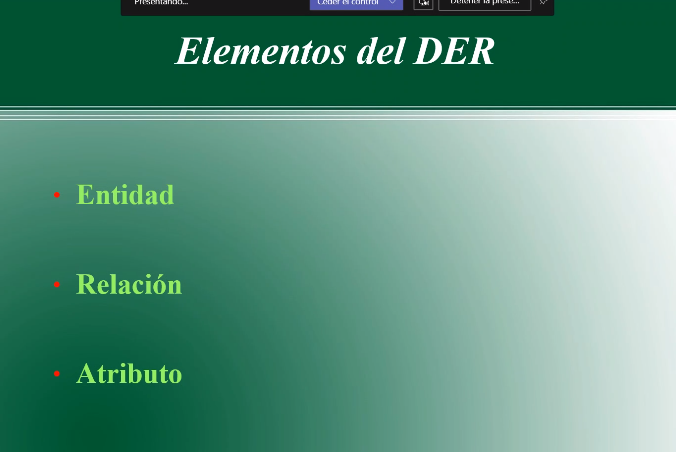
**Diseño**

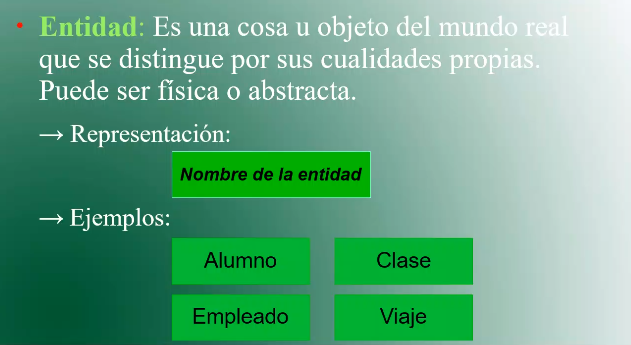
- Cada motor tiene una métrica estrella por la cual se caracteriza.





**Modelo conceptual:** no hace falta tener en cuenta el motor que voy a tener que implementar. No importa en que plataforma digital la voy a hacer. Existen dif estrategias, es para entender el flujo de datos.





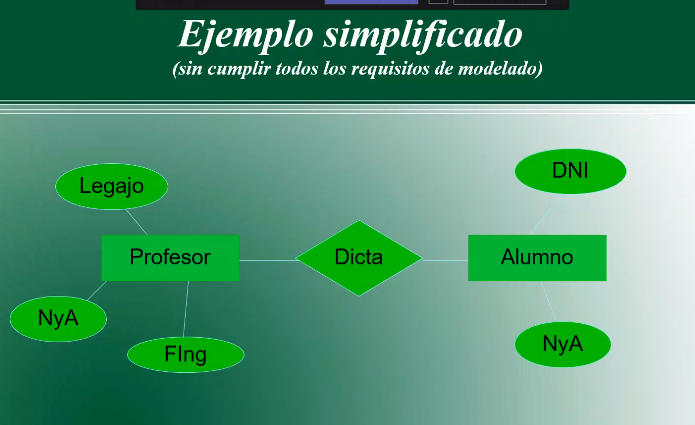
ES LO QUE ME INTERESA ALMACENAR. NO ES QUE NO SE PUEDE PONER MÁS DE UNA PALABRA, SE TIENE UNA TENDENCIA A QUE LOS NOMBRES **NO SON PLURALES**.



El que describe a una cualidad de la entidad. Se pone en un óvalo. SOLO QUE NOS INTERESE MODELAR Y REPRESENTE EN MI SISTEMA.



QUIÉN VA A UNIR LOS DIF OBJETOS DE NUESTRO DIAGRAMA. Cómo se relacionan los dif objetos.



Se va a interpretar qué hacemos con esto.



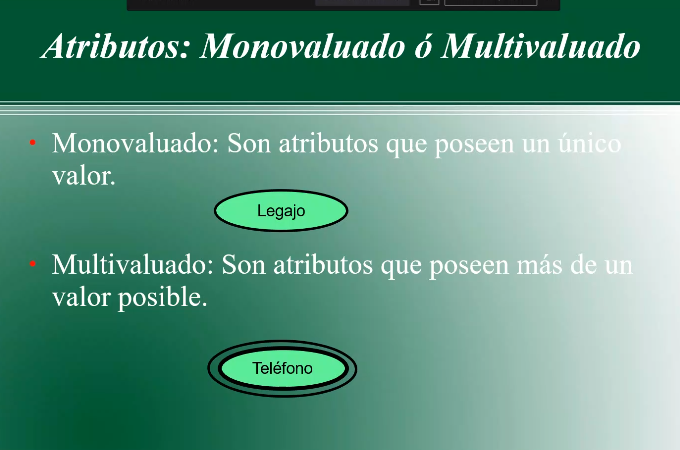
Ej: Solo nros, solo fechas, cualquier carácter.

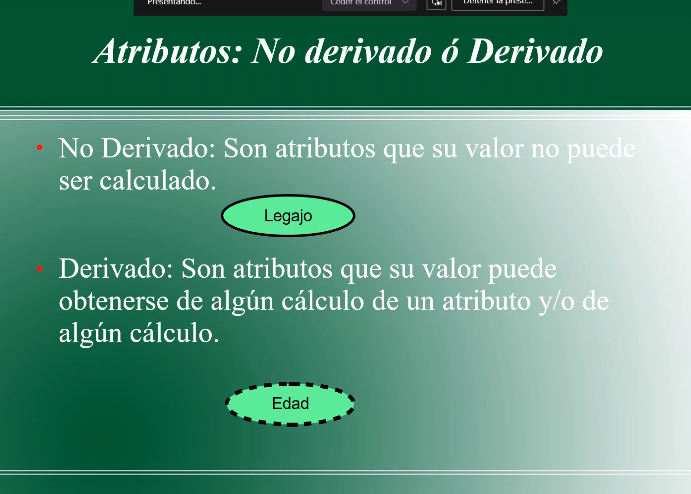




**Ejemplos:**

- Simple: Fecha de ingreso, nro de legajo

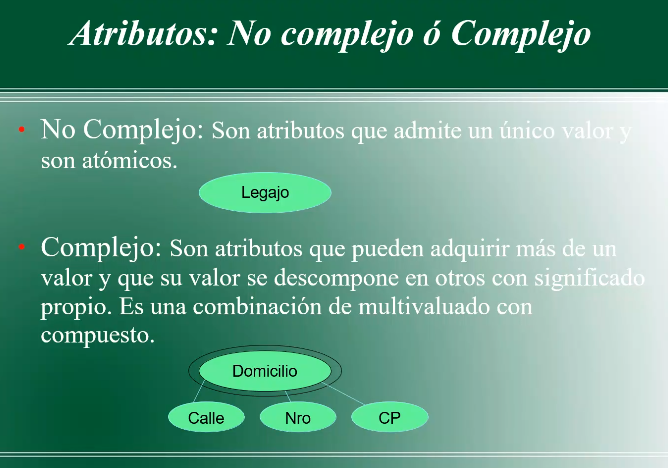




**Ejemplo**:

- No derivado: el legajo 12345 no puedo adivinarlo

- Derivado: La edad (si tengo la fecha de nac. Como dato, es un dato que se puede calcular). Linea puntuada



****

**No se dejan en blanco los datos**

****

**-** En el contexto de bases de datos, las claves son elementos fundamentales para la organización de los datos. Las tres claves más importantes son las claves primarias, las claves foráneas y las claves candidatas. A continuación, te explico en qué consiste cada una de ellas:

1. **Clave primaria**: es un campo o conjunto de campos que identifica de forma única cada registro en una tabla de una base de datos. La clave primaria se utiliza para garantizar la integridad de los datos y evitar duplicidades o inconsistencias. Por lo general, la clave primaria se define como un campo que no admite valores nulos y que es único para cada registro de la tabla.

Ejemplo: En una tabla de clientes, la clave primaria podría ser el número de identificación de cada cliente, ya que este valor es único para cada uno de ellos y permite identificarlos sin ambigüedad.

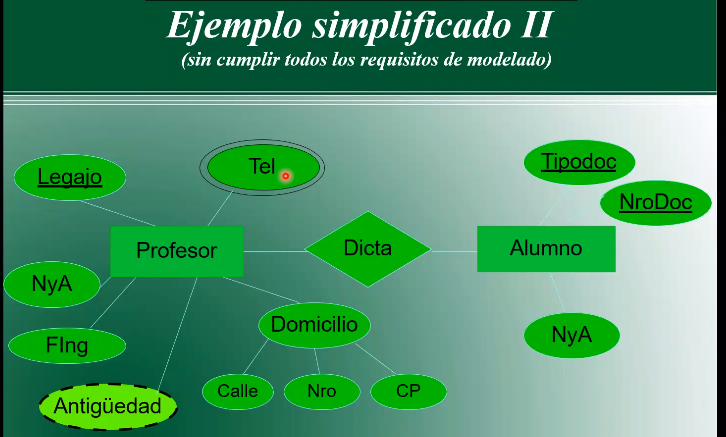
1. **Clave foránea**: es un campo o conjunto de campos que establece una relación entre dos tablas de una base de datos. La clave foránea se utiliza para asegurar la integridad referencial entre las tablas y garantizar que los datos sean coherentes. Por lo general, la clave foránea se define como el campo de una tabla que coincide con la clave primaria de otra tabla.

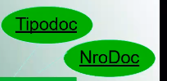
Ejemplo: Si tenemos una tabla de pedidos y otra tabla de clientes, la clave foránea en la tabla de pedidos podría ser el número de identificación del cliente que realizó el pedido. De esta forma, podemos relacionar cada pedido con su correspondiente cliente.

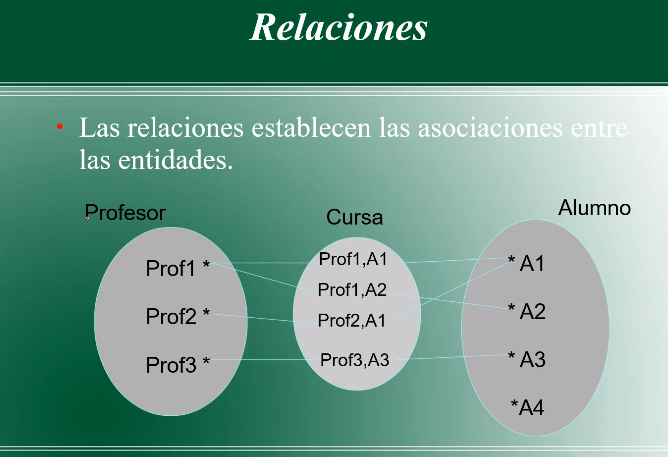
1. **Clave candidata:** es un campo o conjunto de campos que podría utilizarse como clave primaria de una tabla, pero que aún no ha sido seleccionado como tal. La clave candidata se utiliza para identificar registros de forma única, pero aún no se ha establecido como clave primaria definitiva.

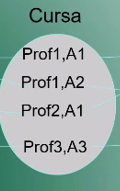
Ejemplo: En una tabla de empleados, tanto el número de identificación como el número de seguridad social podrían ser claves candidatas, ya que ambos campos identifican de forma única a cada empleado. Sin embargo, solo uno de ellos se seleccionaría como clave primaria definitiva.

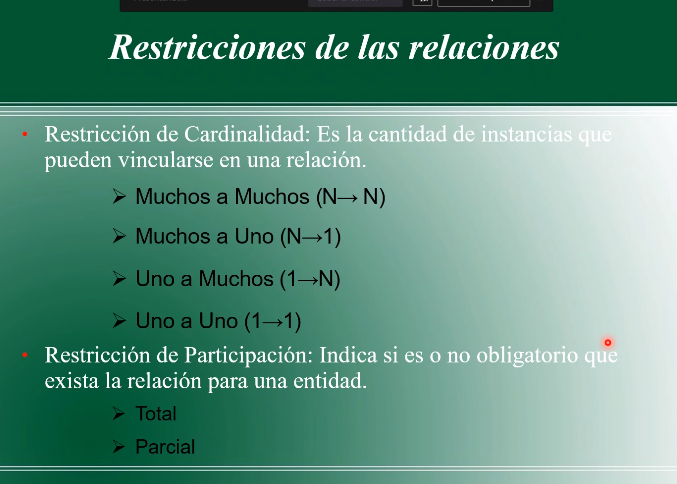
Principio del formulario

****

**- Los subrayados: Es una clave primaria, puede ser simple o compuesta. Si yo tengo el nro. de doc. solo, no es suficiente para identificar al alumno. (Puedo tener dos alumnos que tengan el mismo nro. de doc.)**

****

**¿Cómo se quiénes son los que están ahí?: Con la clave primaria para que más nos convenga para nuestra relación. La vinculación son las claves primarias, no va a permitir vincular.**

****

Para identificar las restricciones de cardinalidad en una base de datos, es importante tener en cuenta los siguientes factores clave:

1. Entender la relación entre las tablas: Para determinar la cardinalidad entre dos tablas, es importante entender cómo se relacionan entre sí. Por ejemplo, una tabla de clientes puede estar relacionada con una tabla de pedidos, donde cada cliente puede tener muchos pedidos.
2. Identificar el tipo de relación: Las restricciones de cardinalidad pueden variar según el tipo de relación que exista entre las tablas. Por ejemplo, si una tabla tiene una relación uno a uno, entonces solo habrá una instancia de cada registro en ambas tablas.
3. Observar los valores de los campos: Para determinar la cardinalidad entre dos tablas, es importante observar los valores de los campos que se utilizan para relacionar las tablas. Por ejemplo, si un campo tiene valores repetidos, entonces puede ser una relación de muchos a uno.

Algunos ejemplos de restricciones de cardinalidad son:

* Uno a uno (1:1): Una restricción de cardinalidad 1:1 se produce cuando cada registro en una tabla solo puede estar relacionado con un registro en otra tabla, y viceversa. Por ejemplo, en una tabla de empleados, cada empleado puede tener solo un número de seguridad social, y cada número de seguridad social solo puede estar relacionado con un empleado.
* Uno a muchos (1:N): Una restricción de cardinalidad 1:N se produce cuando cada registro en una tabla solo puede estar relacionado con uno o muchos registros en otra tabla, pero cada registro en la segunda tabla solo puede estar relacionado con un registro en la primera tabla. Por ejemplo, en una tabla de pedidos, cada pedido puede estar relacionado con un solo cliente, pero cada cliente puede tener muchos pedidos.
* Muchos a muchos (N:N): Una restricción de cardinalidad N:N se produce cuando muchos registros en una tabla pueden estar relacionados con muchos registros en otra tabla. Por ejemplo, en una tabla que relaciona estudiantes con cursos, cada estudiante puede estar matriculado en muchos cursos, y cada curso puede tener muchos estudiantes matriculados.

Es importante tener en cuenta que la cardinalidad entre dos tablas puede cambiar con el tiempo, y es posible que se requieran ajustes en las restricciones de cardinalidad para adaptarse a los cambios en los datos.

****

**EL N : Lo establezco**

****

**Derecha a izq:**

**- 1 profesor dicta a muchos alumnos**

**Izq a derecha:**

**- 1 alumno tiene muchos profesores. (Tiene muchas materias)**

**\* Es relativo al modelo de requerimientos que tengamos \***

**\* Tener en cuenta que tiene que ser de arriba abajo la lectura \***

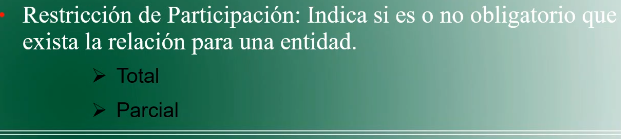
La cardinalidad puede cambiar en base al contexto en el que se desenvuelve una base de datos. Por ejemplo, si una tabla de una base de datos está diseñada para almacenar información de clientes y se decide agregar un campo que permita a los clientes registrarse en un programa de lealtad, la cardinalidad entre la tabla de clientes y la tabla de programas de lealtad podría cambiar. Antes, la relación podría haber sido 1:N (un cliente tiene muchos pedidos) y después del cambio podría ser N:M (muchos clientes pueden registrarse en muchos programas de lealtad).

También es posible que las restricciones de cardinalidad cambien si se agregan o eliminan tablas en la base de datos o si se modifican las relaciones entre ellas. Por ejemplo, si se elimina una tabla de productos de una base de datos de pedidos en línea, la cardinalidad entre la tabla de pedidos y la tabla de productos cambiaría.

En resumen, la cardinalidad puede cambiar en cualquier momento en base a los cambios en el contexto en el que se utiliza la base de datos. Es importante revisar y actualizar las restricciones de cardinalidad en caso de cambios en la estructura de la base de datos para garantizar la integridad y la coherencia de los datos almacenados.

****

****

****

**- Total: Se marca con línea doble. Sí o sí tiene que existir. Ej: Estoy embarazado y pido licencia. No tengo relación con mis alumnos, pero no me borran.**

**- Parcial: Marca una opcionalidad, puede existir o no. Puede tener relación o no. Se marca con un circulito (¿?)**

****

**Derecha a izq: total.**

**Izq a derecha: parcial.**

**GRADO DE UNA RELACIÓN**

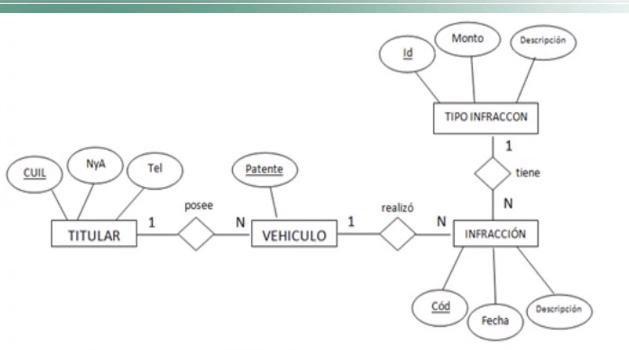
****

**Intervienen sí o sí 3 entidades (Ternaria)**

**Primer N(Izquierda): Una empresa, para un proyecto. ¿Cuántos empleados tiene?**

**Segundo N (Abajo): Un empleado para un proyectos ¿ Cuántas empresas**

**Tercer N(Derecha):**

****

**Dice que un titular puede tener muchos autos.**

**Primer cuadro:**

**- Al titular lo identifico por el CUIT**

**- Posee una relación, tiene un solo titular.**

**Segundo cuadro:**

**- Lo identifico por la patente**

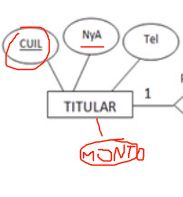
**Tercer cuadro:**

**- Puede tener muchas infracciones con una descripción.**

**TIPO DE ENTIDAD: REQUIERE INDIVIDUALIZARLO**

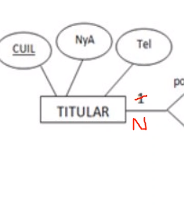
**(NO ESTÁ PUESTO EL TIPO PORQUE TENGO UN CÓDIGO) à ESTÁ TIPIFICADO**

*Al estar relacionado, se puede llegar con el tipo de infracción al titular que tiene que abonar la multa.*

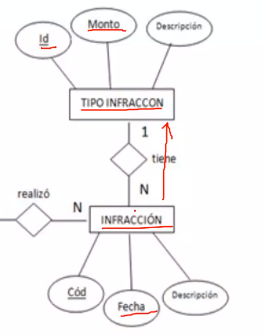
Lo incorrecto de poner MONTO ahí, es que no es un atributo propio del titular:

- NO ES UN DATO ESTÁTICO, ya que mañana me pueden hacer una infracción, y aumenta el monto.

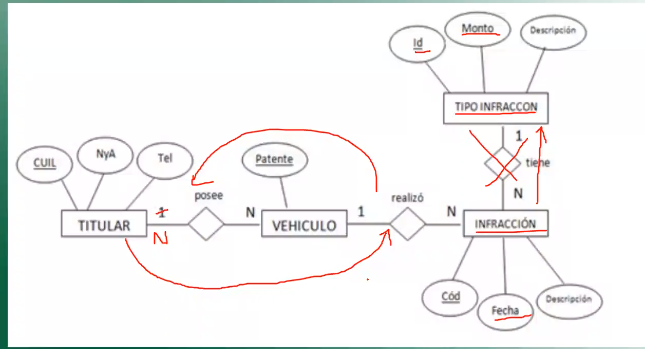
- Tienen un camino que respetar por la vinculación de los datos.



Si yo le quiero cobrar A LA PERSONA RESPONSABLE la multa, el DER está mal hecho (Un auto puede ser compartido por una pareja por ej) --> Por eso le tacha el 1



Lo que me permiten las vinculaciones, es tener la info que tiene cada entidad por separada.



Si doy vuelta las cosas, el monto lo tiene el titular, no el vehículo. La complicación es cuando es N a N. ¿Cómo se sabe qué persona fue la responsable?

Por esto:

- ES IMPORTANTE SABER EL REQUERIMIENTO DE NUESTRO DER

- Se puede modificar.