

TEMA 4
DISEÑO CONCEPTUAL Y LÓGICO DE UNA BASE DE DATOS



4.1. FORMATOS DE REPRESENTACIÓN.

- El siguiente paso de la metodología para realizar el diseño de la BD después de la definición de requerimientos es el diseño conceptual empleando para ello los conceptos del llamado Modelo Entidad relación E/R.
- Para poder iniciar con el diseño conceptual, es importante que en la etapa de análisis se hayan identificado todos los elementos del modelo de datos: Entidades, atributos, relaciones entre entidades, restricciones.
- Una vez que se ha desarrollado el diseño conceptual, se procede con la transformación del Modelo entidad relación ER en el llamado Modelo relacional empleado los conceptos vistos en el tema anterior. A este proceso se le conoce como Diseño lógico.
- En este capítulo se ilustra el uso de distintas notaciones empleadas para realizar el diseño tanto conceptual como lógico.
- Se revisan ambos modelos en paralelo debido a la correspondencia que existe entre cada uno de ellos, de una visión global y general que ofrece el modelo conceptual a un modelo particular, en este caso el relacional.

4.1.1. Principales formatos.



Para el diseño conceptual:

- ** Formato de Chen (Chen's Format) - Creador **Peter Pin-Shan Chen** en 1976.

Para el diseño lógico:

- Formato Relacional (Relational Format)
- ** Formato IE (International Engineering Format) / Formato de Martín (Martin's Format) / Modelo Pata de gallo (Crow's foot Format).
- ** Formato IDEF1X. Originalmente desarrollado por "The Computer System Laboratory of the National Institute of Standards and Technology" en diciembre del 1993.

** Estos 3 formatos son los que se emplearán en este capítulo.

4.2. REPRESENTACIÓN DE ENTIDADES Y ATRIBUTOS.



4.2.1. Representación de entidades.

Diseño conceptual	Diseño lógico	Crow's Foot e IDEF1X
Entidad <div style="border: 1px solid black; padding: 5px; text-align: center;">Cliente</div>		Tabla CLIENTE <div style="border: 1px solid black; padding: 5px; text-align: center;"></div>

Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X
<ul style="list-style-type: none"> Se emplea un rectángulo con el nombre de la entidad en su interior iniciando en mayúsculas, sustantivo en singular Los nombres de las entidades pueden contener espacios, acentos, etc. 	<ul style="list-style-type: none"> En el modelo relacional, se emplea el concepto de tabla para representar a una entidad. El nombre se escribe arriba del rectángulo en mayúsculas. Si el nombre de la tabla es compuesto, se emplea “_” (guion bajo) para separar palabras. En el primer rectángulo se enlistan los atributos que formarán parte de la PK En el segundo rectángulo se enlistan los atributos que no forman parte de la PK.

4.2.2. Representación general de atributos.

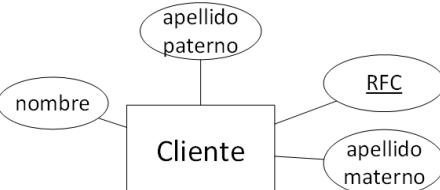
Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X									
 <ul style="list-style-type: none"> Los atributos se representan por óvalos. Los nombres pueden contener espacios 	<p>CLIENTE</p> <table border="1"> <tr> <td>◆ NOMBRE</td> <td>VARCHAR(50)</td> <td>NOT NULL</td> </tr> <tr> <td>◆ AP_PATERNO</td> <td>VARCHAR(50)</td> <td>NOT NULL</td> </tr> <tr> <td>◆ AP_MATERO</td> <td>VARCHAR(50)</td> <td>NOT NULL</td> </tr> </table>	◆ NOMBRE	VARCHAR(50)	NOT NULL	◆ AP_PATERNO	VARCHAR(50)	NOT NULL	◆ AP_MATERO	VARCHAR(50)	NOT NULL
◆ NOMBRE	VARCHAR(50)	NOT NULL								
◆ AP_PATERNO	VARCHAR(50)	NOT NULL								
◆ AP_MATERO	VARCHAR(50)	NOT NULL								

4.2.3. Clasificación de atributos.

- Clave principal y llave primaria natural.
- Clave candidata y llaves primaria candidata.
- Clave artificial y llave primaria artificial o subrogada
- Atributos obligatorios y opcionales
- Atributos simples y compuestos.
- Atributo de valores múltiples y de valor simple.
- Atributos derivados.



4.2.3.1. Clave principal y llave primaria natural

Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X												
 <ul style="list-style-type: none"> Una clave principal es un atributo cuyo valor es único para cada instancia de la entidad. Permite identificar de forma única a cada una de las instancias de la entidad. El atributo se subraya. 	<p>CLIENTE</p> <table border="1"> <tr> <td>◆ RFC</td> <td>VARCHAR(13)</td> <td>NOT NULL</td> </tr> <tr> <td>◆ NOMBRE</td> <td>VARCHAR(50)</td> <td>NOT NULL</td> </tr> <tr> <td>◆ AP_PATERNO</td> <td>VARCHAR(50)</td> <td>NOT NULL</td> </tr> <tr> <td>◆ AP_MATERO</td> <td>VARCHAR(50)</td> <td>NOT NULL</td> </tr> </table>	◆ RFC	VARCHAR(13)	NOT NULL	◆ NOMBRE	VARCHAR(50)	NOT NULL	◆ AP_PATERNO	VARCHAR(50)	NOT NULL	◆ AP_MATERO	VARCHAR(50)	NOT NULL
◆ RFC	VARCHAR(13)	NOT NULL											
◆ NOMBRE	VARCHAR(50)	NOT NULL											
◆ AP_PATERNO	VARCHAR(50)	NOT NULL											
◆ AP_MATERO	VARCHAR(50)	NOT NULL											

Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X
<ul style="list-style-type: none"> La clave principal es seleccionada de la lista de los atributos 'naturales' de la entidad Todas las entidades deben contar con su clave principal. 	<ul style="list-style-type: none"> En la práctica, generalmente se reemplaza a la llave primaria natural por una llave primaria artificial por las razones vistas en el capítulo anterior.

4.2.3.2. Clave candidata y llave primaria candidata.

Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X																					
	<p>CLIENTE</p> <table border="1"> <tr> <td>NUM_CLIENTE</td> <td>NUMERIC(10,0)</td> <td>NOT NULL</td> </tr> <tr> <td>RFC</td> <td>VARCHAR(13)</td> <td>NOT NULL</td> </tr> <tr> <td>CURP</td> <td>VARCHAR(18)</td> <td>NOT NULL</td> </tr> <tr> <td>NUM</td> <td>VARCHAR(30)</td> <td>NOT NULL</td> </tr> <tr> <td>NOMBRE</td> <td>VARCHAR(50)</td> <td>NOT NULL</td> </tr> <tr> <td>AP_PATERNO</td> <td>VARCHAR(50)</td> <td>NOT NULL</td> </tr> <tr> <td>AP_MATERNO</td> <td>VARCHAR(50)</td> <td>NOT NULL</td> </tr> </table>	NUM_CLIENTE	NUMERIC(10,0)	NOT NULL	RFC	VARCHAR(13)	NOT NULL	CURP	VARCHAR(18)	NOT NULL	NUM	VARCHAR(30)	NOT NULL	NOMBRE	VARCHAR(50)	NOT NULL	AP_PATERNO	VARCHAR(50)	NOT NULL	AP_MATERNO	VARCHAR(50)	NOT NULL
NUM_CLIENTE	NUMERIC(10,0)	NOT NULL																				
RFC	VARCHAR(13)	NOT NULL																				
CURP	VARCHAR(18)	NOT NULL																				
NUM	VARCHAR(30)	NOT NULL																				
NOMBRE	VARCHAR(50)	NOT NULL																				
AP_PATERNO	VARCHAR(50)	NOT NULL																				
AP_MATERNO	VARCHAR(50)	NOT NULL																				

- Una entidad puede contener varias claves principales. En este caso se selecciona a uno de estos atributos y a los demás se les llama **claves candidatas**.
- Observar en uso de una X para identificar a las claves candidatas que no fueron seleccionadas como **claves principales**.

- En el modelo relacional el concepto de **clave candidata** corresponde con el concepto de **llave primaria candidata**.
- No existe una notación en particular para identificar a las PK candidatas.
- Puede asignarse una restricción de integridad UNIQUE.

4.2.3.3. Clave artificial

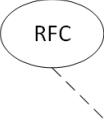
Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X												
	<p>CLIENTE</p> <table border="1"> <tr> <td>CLIENTE_ID</td> <td>NUMERIC(10,0)</td> <td>NOT NULL</td> </tr> <tr> <td>NOMBRE</td> <td>VARCHAR(50)</td> <td>NOT NULL</td> </tr> <tr> <td>AP_PATERNO</td> <td>VARCHAR(50)</td> <td>NOT NULL</td> </tr> <tr> <td>AP_MATERNO</td> <td>VARCHAR(50)</td> <td>NOT NULL</td> </tr> </table>	CLIENTE_ID	NUMERIC(10,0)	NOT NULL	NOMBRE	VARCHAR(50)	NOT NULL	AP_PATERNO	VARCHAR(50)	NOT NULL	AP_MATERNO	VARCHAR(50)	NOT NULL
CLIENTE_ID	NUMERIC(10,0)	NOT NULL											
NOMBRE	VARCHAR(50)	NOT NULL											
AP_PATERNO	VARCHAR(50)	NOT NULL											
AP_MATERNO	VARCHAR(50)	NOT NULL											

- Solo para aquellos casos donde la clave principal no es tan evidente o no se especifica claramente el enunciado del problema se puede emplear una **clave artificial**.
- Por ejemplo, solo se hace mención de los atributos como son nombre, apellido paterno y apellido materno.
- De ser así es posible crear una **clave principal** empleando la notación `id <nombre entidad>`.

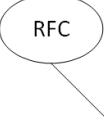
- En el modelo relacional el concepto de **clave artificial** corresponde con el concepto de **llave primaria artificial o subrogada**. Revisada en el capítulo anterior. Recordar la notación: `<nombre_tabla>_id`

4.2.3.4. Atributos opcionales y obligatorios.

Atributo opcional: Atributo cuyos valores pueden ser nulos.

Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X									
	<p>CLIENTE</p> <table border="1"> <tr> <td>CLIENTE_ID</td> <td>NUMERIC(10,0)</td> <td>NOT NULL</td> </tr> <tr> <td>RFC</td> <td>VARCHAR(13)</td> <td>NULL</td> </tr> <tr> <td>CURP</td> <td>VARCHAR(18)</td> <td>NULL</td> </tr> </table>	CLIENTE_ID	NUMERIC(10,0)	NOT NULL	RFC	VARCHAR(13)	NULL	CURP	VARCHAR(18)	NULL
CLIENTE_ID	NUMERIC(10,0)	NOT NULL								
RFC	VARCHAR(13)	NULL								
CURP	VARCHAR(18)	NULL								
• La línea de conexión es punteada	• El atributo aparece con la palabra NULL.									

Atributo requerido: Atributo cuyos valores no pueden ser nulos.

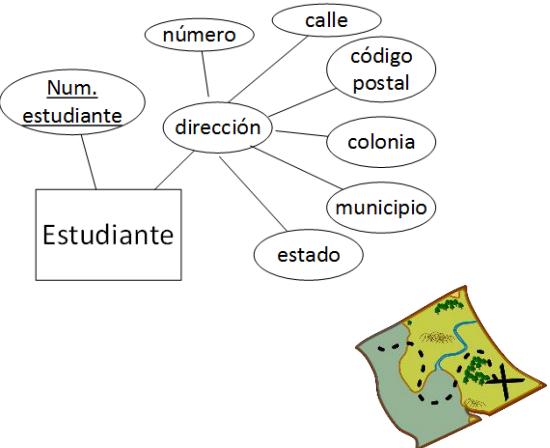
Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X									
	<p>CLIENTE</p> <table border="1"> <tr> <td>CLIENTE_ID</td> <td>NUMERIC(10,0)</td> <td>NOT NULL</td> </tr> <tr> <td>RFC</td> <td>VARCHAR(13)</td> <td>NOT NULL</td> </tr> <tr> <td>CURP</td> <td>VARCHAR(18)</td> <td>NOT NULL</td> </tr> </table>	CLIENTE_ID	NUMERIC(10,0)	NOT NULL	RFC	VARCHAR(13)	NOT NULL	CURP	VARCHAR(18)	NOT NULL
CLIENTE_ID	NUMERIC(10,0)	NOT NULL								
RFC	VARCHAR(13)	NOT NULL								
CURP	VARCHAR(18)	NOT NULL								
• La línea de conexión es continua.	• El atributo aparece con la palabra NOT NULL.									

4.2.3.5. Atributos simples y compuestos.

Atributo simple: Atributo que ya no puede ser dividido en sub atributos

Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X						
	<p>CLIENTE</p> <table border="1"> <tr> <td>CLIENTE_ID</td> <td>NUMERIC(10,0)</td> <td>NOT NULL</td> </tr> <tr> <td>COLONIA</td> <td>VARCHAR(100)</td> <td>NOT NULL</td> </tr> </table>	CLIENTE_ID	NUMERIC(10,0)	NOT NULL	COLONIA	VARCHAR(100)	NOT NULL
CLIENTE_ID	NUMERIC(10,0)	NOT NULL					
COLONIA	VARCHAR(100)	NOT NULL					
• En el ejemplo, la colonia ya no puede dividirse en sub atributos. • No existe notación el particular para este tipo de atributos en ambos diseños.							

Atributo compuesto: Es aquel que puede ser descompuesto en al menos 2 sub atributos relevantes para las reglas de negocio o caso de estudio.

Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X																																													
	<p>Opción 1:</p> <p>ESTUDIANTE</p> <table border="1"> <tr> <td>NUM_ESTUDIANTE</td> <td>NUMERIC(10,0)</td> <td>NOT NULL</td> </tr> <tr> <td>NOMBRE</td> <td>VARCHAR(30)</td> <td>NOT NULL</td> </tr> <tr> <td>APELLIDO_PATERNO</td> <td>VARCHAR(30)</td> <td>NOT NULL</td> </tr> <tr> <td>APELLIDO_MATERNO</td> <td>VARCHAR(30)</td> <td>NOT NULL</td> </tr> <tr> <td>DIRECCION</td> <td>VARCHAR(30)</td> <td>NOT NULL</td> </tr> </table> <p>Opción 2:</p> <p>ESTUDIANTE</p> <table border="1"> <tr> <td>NUM_ESTUDIANTE</td> <td>NUMERIC(10,0)</td> <td>NOT NULL</td> </tr> <tr> <td>NOMBRE</td> <td>VARCHAR(30)</td> <td>NOT NULL</td> </tr> <tr> <td>APELLIDO_PATERNO</td> <td>VARCHAR(30)</td> <td>NOT NULL</td> </tr> <tr> <td>APELLIDO_MATERNO</td> <td>VARCHAR(30)</td> <td>NOT NULL</td> </tr> <tr> <td>CALLE</td> <td>VARCHAR(30)</td> <td>NOT NULL</td> </tr> <tr> <td>NUM_EXTERIOR</td> <td>NUMERIC(5,0)</td> <td>NOT NULL</td> </tr> <tr> <td>COLONIA</td> <td>VARCHAR(30)</td> <td>NOT NULL</td> </tr> <tr> <td>CODIGO_POSTAL</td> <td>VARCHAR(30)</td> <td>NOT NULL</td> </tr> <tr> <td>MUNICIPIO_DELEGACION</td> <td>VARCHAR(30)</td> <td>NOT NULL</td> </tr> <tr> <td>ESTADO</td> <td>VARCHAR(30)</td> <td>NOT NULL</td> </tr> </table>	NUM_ESTUDIANTE	NUMERIC(10,0)	NOT NULL	NOMBRE	VARCHAR(30)	NOT NULL	APELLIDO_PATERNO	VARCHAR(30)	NOT NULL	APELLIDO_MATERNO	VARCHAR(30)	NOT NULL	DIRECCION	VARCHAR(30)	NOT NULL	NUM_ESTUDIANTE	NUMERIC(10,0)	NOT NULL	NOMBRE	VARCHAR(30)	NOT NULL	APELLIDO_PATERNO	VARCHAR(30)	NOT NULL	APELLIDO_MATERNO	VARCHAR(30)	NOT NULL	CALLE	VARCHAR(30)	NOT NULL	NUM_EXTERIOR	NUMERIC(5,0)	NOT NULL	COLONIA	VARCHAR(30)	NOT NULL	CODIGO_POSTAL	VARCHAR(30)	NOT NULL	MUNICIPIO_DELEGACION	VARCHAR(30)	NOT NULL	ESTADO	VARCHAR(30)	NOT NULL
NUM_ESTUDIANTE	NUMERIC(10,0)	NOT NULL																																												
NOMBRE	VARCHAR(30)	NOT NULL																																												
APELLIDO_PATERNO	VARCHAR(30)	NOT NULL																																												
APELLIDO_MATERNO	VARCHAR(30)	NOT NULL																																												
DIRECCION	VARCHAR(30)	NOT NULL																																												
NUM_ESTUDIANTE	NUMERIC(10,0)	NOT NULL																																												
NOMBRE	VARCHAR(30)	NOT NULL																																												
APELLIDO_PATERNO	VARCHAR(30)	NOT NULL																																												
APELLIDO_MATERNO	VARCHAR(30)	NOT NULL																																												
CALLE	VARCHAR(30)	NOT NULL																																												
NUM_EXTERIOR	NUMERIC(5,0)	NOT NULL																																												
COLONIA	VARCHAR(30)	NOT NULL																																												
CODIGO_POSTAL	VARCHAR(30)	NOT NULL																																												
MUNICIPIO_DELEGACION	VARCHAR(30)	NOT NULL																																												
ESTADO	VARCHAR(30)	NOT NULL																																												

Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X
<ul style="list-style-type: none"> • Observar en el ejemplo, los sub atributos se modelan como sub atributos del atributo compuesto. Es decir, la dirección se puede descomponer en calle, número, colonia, etc. 	<ul style="list-style-type: none"> • Se tienen 2 opciones: Opción 1: Especificar solo el atributo compuesto Opción 2: Eliminar el atributo compuesto y especificar solo los atributos simples. • La decisión depende de las reglas de negocio, lo que resulte más adecuado para la solución. <p><u>Ejemplo:</u> Si se requiere procesar o consultar atributos simples por separado, lo adecuado es la opción 2.</p>

4.2.3.6. Atributos derivados.

- Es un atributo cuyo valor es generado a partir de algún cálculo o algoritmo empleando los valores de otros campos de la tabla.
- El valor de un atributo derivado siempre se puede calcular a partir de los valores de los otros campos.

Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X									
	<p>ESTUDIANTE</p> <table border="1"> <tr> <td>ESTUDIANTE_ID</td> <td>NUMERIC(10,0)</td> <td>NOT NULL</td> </tr> <tr> <td>FECHA_NACIMIENTO</td> <td>DATE</td> <td>NOT NULL</td> </tr> <tr> <td>EDAD</td> <td>NUMERIC(3,0)</td> <td>NOT NULL</td> </tr> </table>	ESTUDIANTE_ID	NUMERIC(10,0)	NOT NULL	FECHA_NACIMIENTO	DATE	NOT NULL	EDAD	NUMERIC(3,0)	NOT NULL
ESTUDIANTE_ID	NUMERIC(10,0)	NOT NULL								
FECHA_NACIMIENTO	DATE	NOT NULL								
EDAD	NUMERIC(3,0)	NOT NULL								

- En el ejemplo, el campo edad es derivado ya que su valor puede ser calculado a partir de la fecha de nacimiento.
- Se representa por un óvalo punteado.

- No existe representación en particular. Si se decide conservar al atributo derivado, aparecerá como un atributo más.

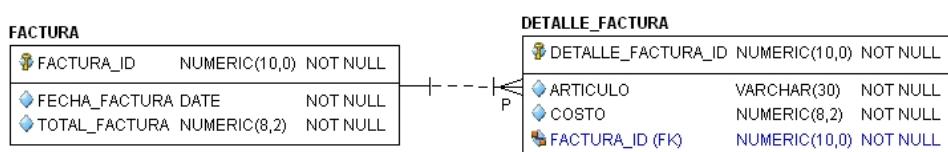
Ventajas de conservar el atributo derivado:

- Ahorro del cálculo requerido para obtener su valor, en especial si este se considera como costoso o complejo.
- Útil si el cálculo es costoso o complejo.

Desventajas de conservar el atributo derivado:

- En sentido estricto, un campo derivado puede considerarse como redundante.
- Al ser redundante puede causar inconsistencias. ¿Qué sucede si se actualiza la fecha de nacimiento, y no se actualiza la edad?
- Posible impacto en almacenamiento. Se requiere espacio para almacenar los valores del atributo derivado.

Algunos manejadores ofrecen algunas opciones para eliminar las desventajas de un atributo derivado. Oracle ofrece el concepto de **columna virtual**. Esta idea permite eliminar los problemas de posibles inconsistencias y posibles impactos en el almacenamiento. (Ver la parte de SQL para mayores detalles).

Ejemplo:

El campo `total_factura` es derivado, ya que en la tabla `detalle_factura` se guarda el costo de cada artículo que contiene la factura. Su valor se obtendría sumando los costos de cada artículo.

4.2.3.7. Atributo de valores múltiples y de valor simple.

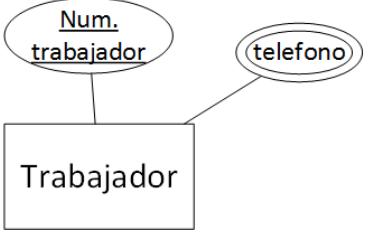
Atributo de valor simple.

- Es un atributo que solo puede tener un único valor por cada registro o instancia de una entidad.
- Observar que un atributo simple no es necesariamente un atributo de valor simple.

Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X
Atributo de valor simple 	Atributo de valor simple
<ul style="list-style-type: none"> • En el ejemplo, la fecha de nacimiento de un estudiante es un atributo de valor simple, ya que por cada estudiante (instancia) solo se tiene un valor para el campo fecha de nacimiento. • Visto de otra forma: el estudiante tiene una sola fecha de nacimiento. • No existe notación en particular para distinguir a un atributo de valor simple en ambos diseños. 	

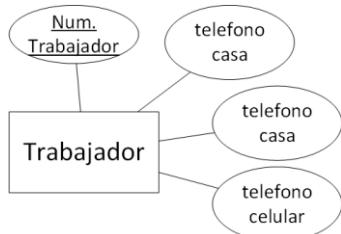
Atributo de valores múltiples.

- Llamado también atributo multi-valorado.
- Es un atributo que puede tener más de un valor para cada registro de la tabla
- En el ejemplo: Un trabajador puede tener varios teléfonos: teléfono de casa, celular, teléfono de oficina, etc.
- En este caso, cada registro o instancia de `Trabajador` puede tener varios valores para el campo `teléfono`. Se trata de un atributo multi-valor.

Diseño conceptual	Diseño lógico Crow's Foot e IDEF1X
Atributo multivalor 	Atributo multivalor <ul style="list-style-type: none"> En el modelo relacional un atributo no puede tener múltiples valores para un mismo registro. Para resolver este caso se emplean estrategias alternas que se muestran a continuación, sin embargo, algunos manejadores permiten la definición de colecciones como tipo de dato y así poder asociar una lista de valores a un registro.
<ul style="list-style-type: none"> Observar que un atributo multi – valor se distingue por un doble ovalo. 	

Estrategia 1.

Agregar un nuevo campo para cada tipo de teléfono. ¿Qué ventajas y desventajas presenta?



TRABAJADOR		
TRABAJADOR_ID	NUMERIC(10,0)	NOT NULL
NOMBRE	VARCHAR(30)	NOT NULL
APELLIDO_PATERNO	VARCHAR(30)	NOT NULL
APELLIDO_MATERNO	VARCHAR(30)	NOT NULL
TELEFONO_CASA	VARCHAR(30)	NULL
TELEFONO_CELULAR	VARCHAR(30)	NULL
TELEFONO_OFICINA	VARCHAR(30)	NULL

Desventajas:

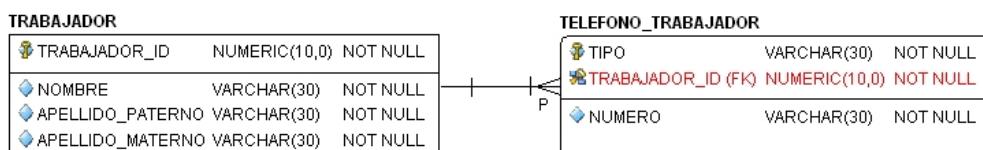
- ¿Qué pasa si el trabajador agrega un nuevo teléfono, por ejemplo, `telefono_dept`? Se tendría que modificar la estructura de la tabla.
- Observar que los 3 campos de teléfono se tienen que definir como nulos ya que no todos los trabajadores cuentan con los mismos tipos de teléfono.

Ventajas:

- Los 3 campos están en la misma tabla lo que permite una explotación y /o recuperación de los 3 valores de forma directa.

Estrategia 2.

Agregar una tabla que permita almacenar un registro por cada tipo de teléfono. ¿Qué ventajas y desventajas presenta?



Ventajas:

- El modelo puede guardar N números telefónicos, no solo 3. Observar el uso de una nueva tabla formada por una PK compuesta. Cada trabajador puede tener varios teléfonos, clasificados por tipo.
- Para cada trabajador y para cada tipo se almacena el número telefónico.
- El número telefónico ya no tiene que definirse como null.

- ¿Qué pasa si un trabajador no cuenta con algún número telefónico?: Simplemente, no existirán registros en la tabla `telefono_trabajador` para un trabajador en particular.
- Los datos se verían así:

TRABAJADOR_ID	NOMBRE	APELLIDO_PATERNO	APELLIDO_MATERNO
1	JUAN	LOPEZ	LARA
2	PABLO	LUNA	MONTES
3	MARIA	LARA	MARISCAL

TIPO	TRABAJADOR_ID	NUMERO
CASA	1	55909023902
CELULAR	1	55029309233
OFICINA	1	33892898323
CASA	2	55902930212
CELULAR	2	55902899323

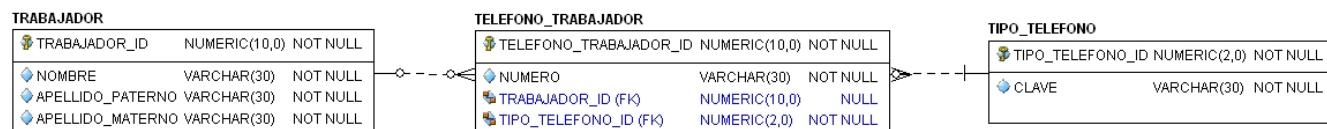
Desventajas:

- La recuperación de la información acerca de los teléfonos de un trabajador implica hacer una operación y lectura extra con la tabla nueva, es decir, se requiere hacer una operación “join” entre ambas tablas para recuperar los datos. Esto pudiera generar problemas de desempeño sobre todo con tablas con una gran cantidad de registros. Sin embargo, este detalle en términos generales se prefiere a la estrategia anterior.
- Observar que la PK es compuesta y está formada por una cadena de texto. ¿Qué pasa si en lugar de guardar la palabra CELULAR, se guarda “CEL”? Este problema se resuelve con la siguiente estrategia:

Estrategia 3.

Optimización con una PK artificial y creando catálogo de tipos de teléfonos.

- Ahora tenemos 2 entidades: `trabajador` y `tipo_telefono`. Por lo anterior, podemos escribir lo siguiente:
- Un trabajador puede tener varios tipos de teléfonos.
- Un tipo de teléfono puede ser asociado con varios trabajadores



Los datos se verían así:

TRABAJADOR

TRABAJADOR_ID	NOMBRE	APELLIDO_PATERNO	APELLIDO_MATERNO
1	JUAN	LOPEZ	LARA
2	PABLO	LUNA	MONTES
3	MARIA	LARA	MARISCAL

TELEFONO_TRABAJADOR

TELEFONO_TRABAJADOR_ID	TRABAJADOR_ID	TIPO_TELEFONO_ID	NUMERO
1	1	1	55909023902
2	1	2	55029309233
3	1	3	33892898323
4	2	1	55902930212
5	2	2	55902899323

TIPO_TELEFONO

TIPO_TELEFONO_ID	NOMBRE
1	CASA
2	CELULAR
3	OFICINA

- Observar, se agrega una PK artificial para mejorar el desempeño al momento de recuperar los datos.
- En esta última estrategia, se tiene el mejor nivel de integridad, pero la explotación se ve afectada.

Ejercicio en clase 1

Para el siguiente enunciado realizar:

- Propuesta de diseño conceptual empleando un modelo ER
- Propuesta de un diseño lógico empleando un modelo relacional.



Se desea construir una pequeña base de datos para almacenar el registro de solicitudes de profesores candidatos para ocupar las vacantes de un colegio privado. Como parte inicial del proceso se requiere almacenar nombre y apellidos del candidato, número de cédula profesional, descripción de su título, año de titulación. Para establecer una comunicación segura, a cada aspirante se le permite registrar hasta 3 correos electrónicos.

Se requiere registrar la lista de asignaturas que el profesor puede o desea impartir (álgebra, cálculo, etc.). Finalmente, en caso que el aspirante haya trabajado anteriormente en el colegio, se registran los números de contrato (folios de 5 dígitos) realizados en el pasado.

4.3. REPRESENTACIÓN DE RELACIONES.

Resumen:

- Representación general.
- Niveles de dependencia (débil y fuerte).
- Cardinalidad
- Dependencia de existencia.
- Participación de una entidad en una relación.
- Entidades débiles.
- Grado de una relación



4.3.1. Representación general de relaciones.

Diseño conceptual.

- Asociación o correspondencia existente entre entidades.
- Representación a través de un rombo con el nombre de la relación (verbo en 3^a persona).
- El nombre que se especifica en el rombo debe permitir leer la relación en 2 posibles sentidos dependiendo la forma en la que ambas entidades están organizadas en el diagrama.
 - De izquierda hacia la derecha
 - De arriba hacia abajo

Ejemplo:



4.3.1.1. Papel o rol

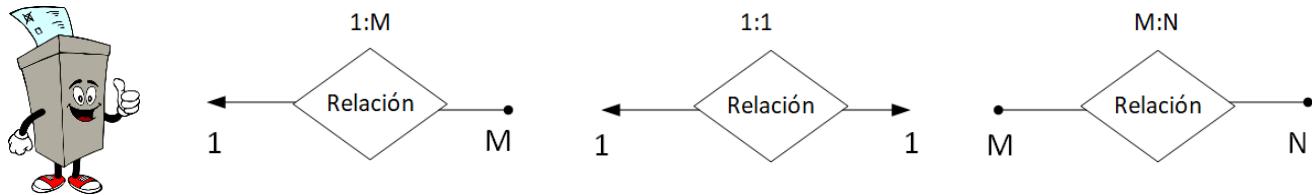
- Es la función que realiza cada tipo de entidad en una relación
- Se representa con un pequeño texto que se coloca sobre la línea que une a las entidades.

- En la práctica es poco común agregar el rol.



- En este caso, la relación se lee de izquierda a derecha: Un empleado “Administra” una agencia.

4.3.1.2. Representación de tipos de relaciones.



- En una relación 1:M la flecha apunta a la tabla padre
- En una relación M:N se especifican 2 puntos finales en lugar de flechas.
- En una relación 1:1 Se especifican ambas flechas.

Ejemplos:

Relación	Representación diseño conceptual
Sucursal y Ciudad (1:M)	<pre> classDiagram class Sucursal class Ciudad Sucursal "1" -->{ Ciudad : pertenece </pre>
Factura y Renta (1:1)	<pre> classDiagram class Renta class Factura Renta "1" -->{ Factura : genera Factura "1" -->{ Renta : genera </pre>
Sucursal y Empleado (Agente) (M:N)	<pre> classDiagram class Empleado class Sucursal Empleado "M:N" -->{ Sucursal : pertenece Agente de Ventas "Agente de Ventas" -->{ Empleado Agente de Ventas -->{ Sucursal </pre>

Diseño lógico.

Recordando del tema anterior, para relacionar 2 entidades se emplea el concepto de llave foránea (FK). Para asociar a las 2 entidades es necesario conocer el tipo de relación:

- Relación uno a uno (1:1)
- Relación uno a muchos (1:M)
- Relación muchos a muchos (M:N)

El siguiente paso es identificar la tabla que contendrá a la FK. Este proceso se realiza en la siguiente sección.

4.3.2. Niveles de dependencia.

El nivel de dependencia indica que tan fuerte es la relación entre 2 entidades. Existen 2 niveles representados por 2 tipos de relaciones:

- Relaciones suaves, débiles o no identificativas. -----
- Relaciones fuertes, duras o identificativas. _____

4.3.2.1. Relaciones no identificativas, suaves o débiles.

- Consiste en relacionar 2 entidades a través de una **línea punteada**. Normalmente se emplea para representar relaciones 1:M aunque también puede emplearse para representar relaciones 1:1, y en algunos casos, relaciones M:N.
- Al asociar 2 tablas con una relación no identificativa (línea punteada) la llave primaria (PK) de la tabla padre pasa como un campo más de la tabla hija como llave foránea (FK).

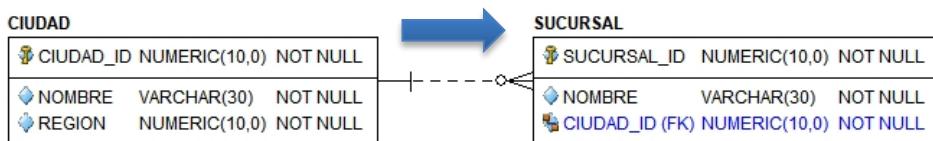
Ejemplos:

Considerar las parejas de entidades que se revisaron en temas anteriores.

i. Sucursal y Ciudad (1:M)

- **Una** sucursal se ubica en **una** ciudad. (1:1)
- En **una** ciudad pueden existir **varias** sucursales. (1:M)

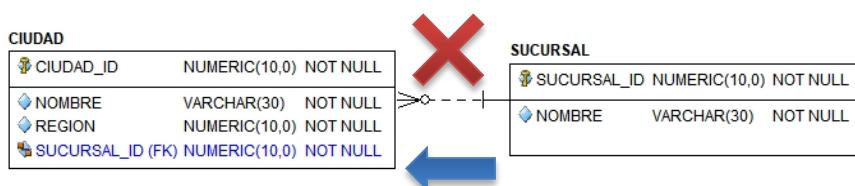
En una relación 1:M, la FK debe quedar del lado “Muchos”. Observando el ejemplo, la FK debe quedar del lado de la entidad Sucursal.



Para verificar lo anterior, basta con imaginar los datos:

- Cada registro de **sucursal**, se asocia con un registro de **ciudad**.
- Cada registro de **ciudad** puede estar asociado con varios registros de **sucursal**.

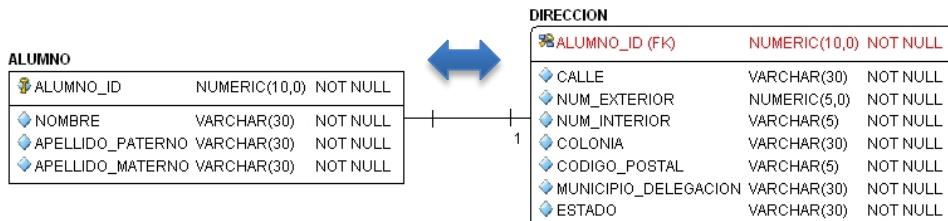
¿Qué sucedería si la FK estuviera del lado de la tabla **ciudad**?



- Como se puede observar, a cada registro de **ciudad** solo se le puede asociar un registro de **sucursal**, pero, ¿Qué sucede si 2 sucursales están en la misma ciudad?
- Esta condición puede ocurrir, pero con la FK de esta forma, no es posible registrar los datos correspondientes.

4.3.2.2. Relaciones identificativas, fuertes o duras.

- Consiste en relacionar 2 entidades a través de una *línea continua*. Se emplea para representar relaciones 1:1 y para modelar relaciones M:N
- Al asociar 2 tablas con una relación identificativa (línea continua) la llave primaria (PK) de la tabla padre pasa como llave primaria (PK) y también como llave foránea (FK) en la tabla hija.
- La diferencia con la relación no identificativa es que la (FK) forma parte de la PK de la tabla hija.
- Si la tabla hija no tiene una PK propia, la PK de la tabla padre y de la tabla hija es **compartida**.
- Por lo anterior y por la definición de PK, la relación entre ambas tablas es 1:1, un registro de la tabla padre, **solo** puede asociarse con un registro de la tabla hija.
- Al compartir la PK con la tabla hija, está también **identificará** a cada registro de la tabla hija. Por tal razón a este tipo de relaciones se les conoce como **identificativas**.

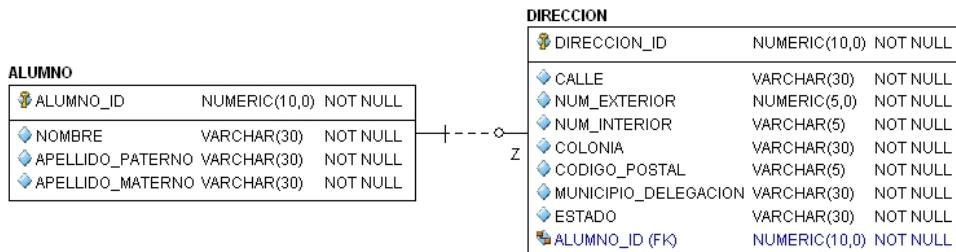


Ejemplos:

- i. Alumno y Dirección (1:1)
 - **Un** alumno tiene **una** dirección. (1:1)
 - **Una** dirección le pertenece a **un** alumno. (1:1)
- Observar que la PK se comparte. Para cada registro de `alumno` le corresponde un registro de `direccion`. No es posible, por ejemplo, registrar 2 direcciones asociadas al mismo alumno. Por tal razón, este tipo de relaciones se emplea para representar relaciones 1:1
- Observar que la PK de `direccion` también es FK.
- ¿Qué pasa si invertimos la PK?, es decir, ahora la PK es `direccion_id` Y es PK Y FK en `alumno`. En realidad, no pasa nada, es equivalente. La relación sigue siendo 1:1
- Observar que en el diseño lógico la tabla "hija" que recibe la PK tiene las esquinas redondeadas.

Estrategia 2:

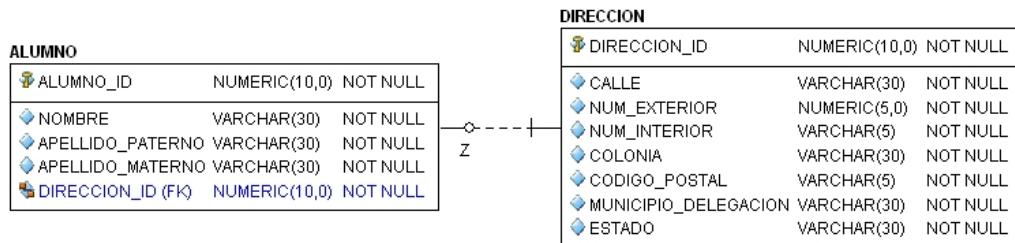
- El hecho de que la tabla dirección tenga una PK y FK `alumno_id` pareciera verse un poco rara. Lo común es ver su propio identificador: `direccion_id`. Para respetar este estilo, se puede aplicar la siguiente variante:



- En sentido estricto, la relación anterior es un 1:M, al estar la PK en `direccion`, podríamos tener que el campo `alumno_id` se repitiera ya que este ya no es PK: Un alumno puede tener varias direcciones.

direccion_id(pk)	alumno_id (fk)
1	1001
2	1001
3	1002
4	1002

- Para respetar la relación 1:1, se puede agregar un índice tipo **unique** a la FK, y con esta técnica el diseño es equivalente al anterior.
- ¿Qué pasa si la FK la escribimos del lado de alumno?



- Ahora podríamos decir que el campo `direccion_id` se puede duplicar, es decir, en una dirección pueden vivir varios alumnos.

alumno_id (pk)	direccion_id (fk)
1001	1
1002	1
1003	2
1004	2

- Si se desea respetar la relación 1:1, podemos emplear la misma técnica, agregar un índice tipo **unique** a la FK, y, por lo tanto, la FK puede quedar en **cualquiera** de las 2 tablas, esto, *siempre y cuando se trate de una relación 1:1*

Estrategia 3:

- Otra solución es eliminar la tabla `direccion` y agregar todos sus campos a `alumno`. Esto es válido al tratarse de una relación 1:1. En realidad se decide separar en 2 tablas, cuando se deseé tener una clara definición y separación de entidades. Si para el caso de estudio es relevante manejar ambas entidades por separado, se recomienda separarlas, de otra forma pueden fusionarse.

ALUMNO
ALUMNO_ID NUMERIC(10,0) NOT NULL
NOMBRE VARCHAR(30) NOT NULL
APELLIDO_PATERNO VARCHAR(30) NOT NULL
APELLIDO_MATERNO VARCHAR(30) NOT NULL
CALLE VARCHAR(30) NOT NULL
NUM_EXTERIOR NUMERIC(5,0) NOT NULL
NUM_INTERIOR VARCHAR(5) NOT NULL
COLONIA VARCHAR(30) NOT NULL
CODIGO_POSTAL VARCHAR(5) NOT NULL
MUNICIPIO_DELEGACION VARCHAR(30) NOT NULL
ESTADO VARCHAR(30) NOT NULL

Ejercicio en clase 2

Considerar las siguientes entidades:

Sucursal y Gerente (1:1)

- Una** sucursal es administrada por **un** gerente. (1:1)
- Un** gerente solo puede administrar **una** sucursal. (1:1)

- ¿Cuál de las 3 estrategias anteriores podría implementar esta relación?
- Construir el modelo relacional correspondiente.



4.3.3. Cardinalidad.

- La cardinalidad expresa el número mínimo y máximo de ocurrencias de una entidad (instancias o registros) asociados con una ocurrencia (instancia o registro) de otra entidad.
- Se emplea el formato (X,Y) y se escribe a un costado de cada entidad.
 - X: valor mínimo
 - Y: valor máximo. Si el valor máximo no se conoce con exactitud se escribe "*" (muchos).
- Los usos de estos valores son de mayor relevancia para la aplicación. La cardinalidad no se implementa en la base de datos, pero es importante que se exprese en el modelo. A nivel de desarrollo de la aplicación es útil contar con esta información.

Ejemplo:

- Un** profesor imparte como máximo **4** cursos.
- Un** curso es impartido por **un** profesor.

Para cada entidad se realizan las siguientes preguntas:

Entidad **profesor**:

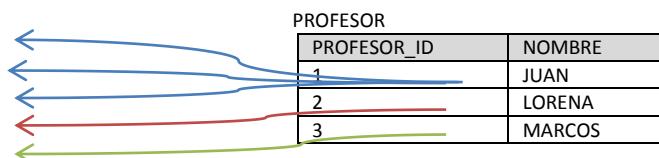
- ¿Cuántas instancias de la entidad **curso** (min y máx.) se asocian con una instancia de **profesor**?
Respuesta: **(1,4)**
 - Un profesor imparte como mínimo 1 curso, máximo 4.*
- El resultado se escribe del lado de la entidad contraria, en este caso al lado de la entidad **curso**.

Entidad **curso**:

- ¿Cuántas instancias de la entidad **profesor** (min y máx.) se asocian con una instancia de **CURSO**?
Respuesta: **(1,1)**.
 - Un curso es impartido mínimo por un profesor, máximo por 1.*
- El resultado se escribe del lado de la entidad contraria, en este caso al lado de la entidad **profesor**.

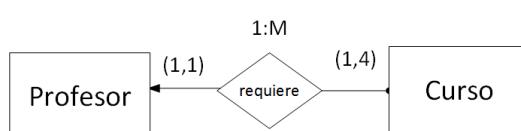
Los datos se verían así:

CURSO		
CURSO_ID	NOMBRE	PROFESOR_ID
1	ALGEBRA	1
2	CALCULO	1
3	GEOMETRIA	1
4	ESTADISTICA	3
5	ESTATICA	2



Observar que para determinar de forma correcta la cardinalidad, es importante conocer las reglas de negocio.

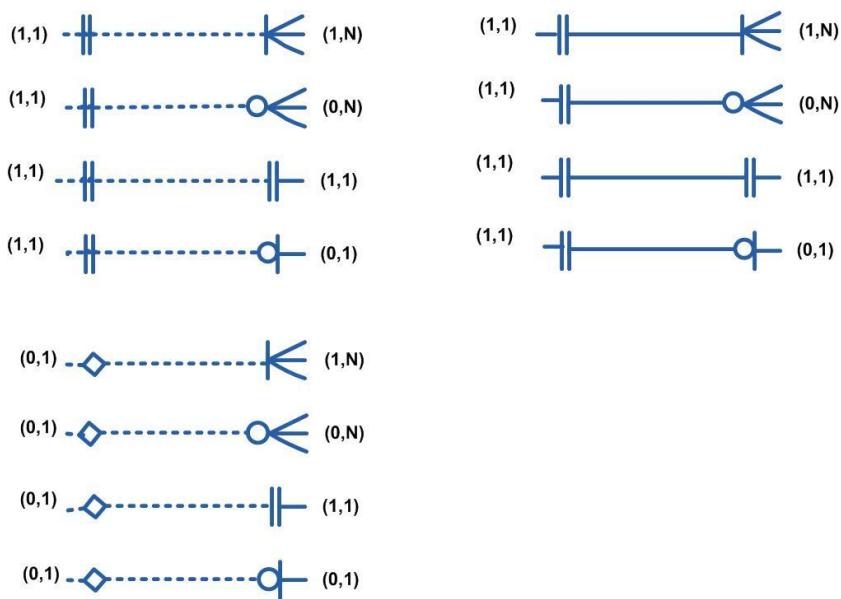
Diseño conceptual



Diseño lógico.

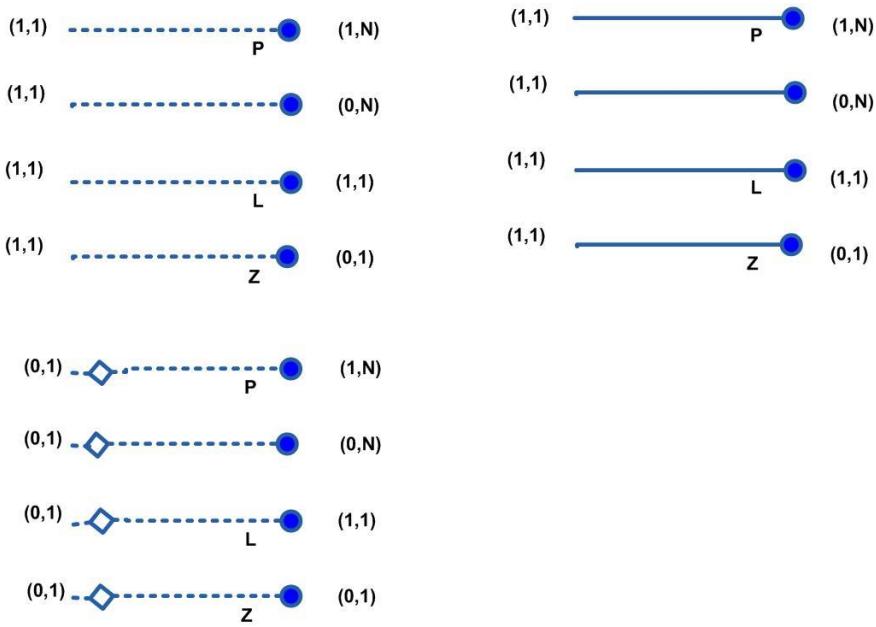
- Similar al diseño conceptual, se emplea la notación (min, max) en ambos lados de las entidades.
- Adicionalmente las relaciones identificativas y no identificativas se complementan con las siguientes notaciones:

4.3.3.1. Cardinalidad con notación crow's Foot



- Observar que no existen relaciones identificativas con cardinalidad (0,1) del lado izquierdo. **¿Cuál es la razón?** Respuesta: la cardinalidad (0,1) del lado de la tabla padre implica una FK que puede tener valores nulos, cosas que no ocurre con una identificativa, ya que la FK forma parte de la PK de la tabla hija, y las llaves primarias no deben tener valores nulos.

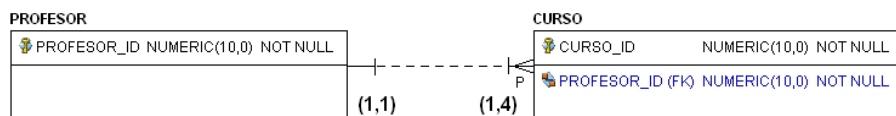
4.3.3.2. Cardinalidad con notación IDEF1X



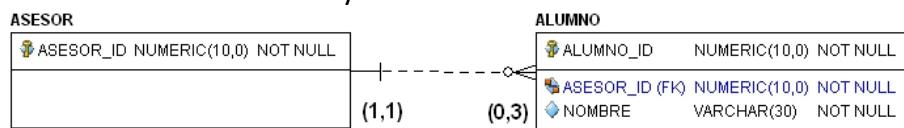
Ejemplo:

Retomando el ejemplo anterior:

- **Un** profesor imparte como máximo **4** cursos.
- **Un** curso es impartido por **un** profesor.

**Ejemplo:**

- Un asesor si lo desea, puede ser asesor de hasta 3 alumnos.
- Un alumno **debe** contar con su asesor y es uno solo durante su estancia en la escuela.

**Ejemplo:**

Suponer que se modifica la regla anterior:

- Un asesor si lo desea puede ser asesor de hasta 3 alumnos.
- Un alumno **puede** solicitar a un único asesor durante su estancia en la escuela.

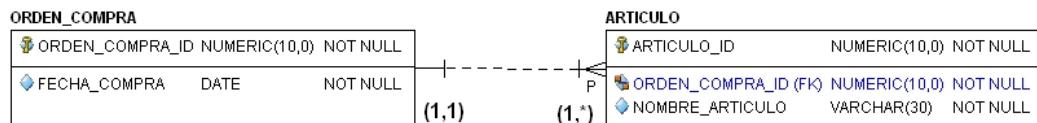


Observar las diferencias en los 2 diagramas anteriores:

- La cardinalidad de alumno hacia asesor es (0,1), ya que el alumno puede o no tener asesor.
- Observar que, en este caso, la FK debe ser declarada como null. De lo anterior, siempre que el valor mínimo de la cardinalidad sea 0, la FK se declara como opcional (null).

Ejemplo:

- Una orden de compra está integrada por varios artículos.
- Un artículo pertenece a una orden de compra.



- Los valores de los lados izquierdos de la cardinalidad, solamente deben tener los valores (0,1), o (1,1)

4.3.4. Dependencia de existencia.

- Una entidad se considera **dependiente de existencia** cuando sus instancias o registros requieren la existencia de una instancia de la entidad con la que se relaciona.

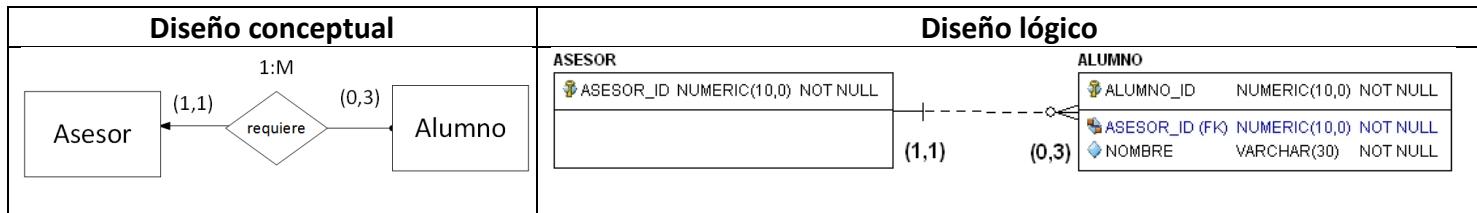


- La dependencia de existencia siempre se verifica del lado de la entidad “**hija**”, la cual puede ser dependiente o independiente de existencia con respecto a la tabla padre.

Ejemplo:

Considerando nuevamente el ejemplo de las entidades **asesor** y **alumno**.

- Un profesor si lo desea, puede ser asesor de hasta 3 alumnos.
- Un alumno **debe** contar con su asesor y es uno solo durante su estancia en la escuela.



Si observamos las reglas de negocio con respecto a la tabla hija: **alumno**, podemos decir, que **alumno** es **dependiente de existencia**, ya que la regla nos dice que todo alumno debe contar con su asesor. Es decir, para poder registrar un alumno es necesario que se le asigne su asesor, por lo tanto, requiere que exista una instancia de **asesor**.

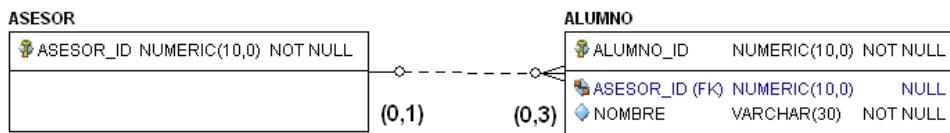
La base de datos nos permite implementar esta condición definiendo la FK como `not null`. Esta restricción impide que se registre un alumno sin su asesor.

Por otro lado, observar la cardinalidad de la tabla **alumno** hacia **asesor** (lado izquierdo). Cuando una entidad es dependiente de existencia, el valor de la cardinalidad siempre es (1,1). Adicionalmente, observar la notación. En Crow's foot, del lado **izquierdo** se emplean 2 líneas para indicar la relación (1,1).

Ejemplo:

Retomando la variante de las entidades **asesor** y **alumno**

- Un profesor si lo desea puede ser asesor de hasta 3 alumnos.
- Un alumno **puede** solicitar a un único asesor durante su estancia en la escuela.



Ahora observamos las reglas de negocio, podemos concluir que la tabla **alumno** es **independiente de existencia** con respecto a **asesor**.

- El alumno puede ser registrado sin su asesor, tal y como lo dicen la regla de negocio.
- La FK está declarada como `null`.
- La cardinalidad del lado de **asesor**, toma el valor de (0,1). Esto siempre ocurre con una entidad independiente de existencia.
- Observar la notación, tanto en Crow's foot como en IDEF1X, se emplea un “rombo” para representar la independencia de existencia.



4.3.5. Participación de una entidad en una relación

A diferencia de la dependencia de existencia, este concepto se verifica observando la cardinalidad en la tabla padre. La participación de una entidad en una relación puede ser obligatoria u opcional:

4.3.5.1. Participación obligatoria.

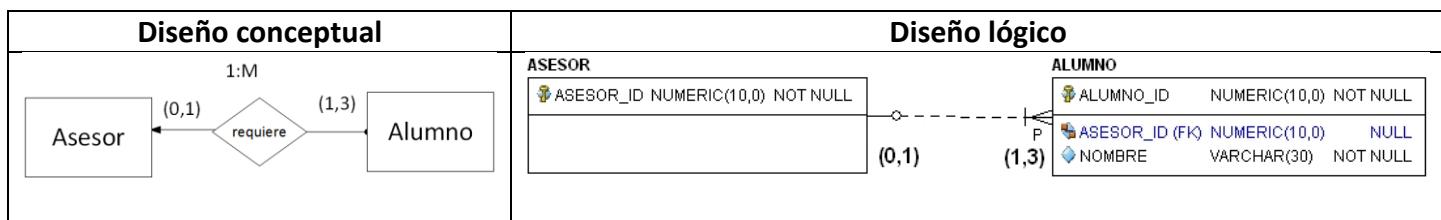
Una entidad “padre” tiene participación obligatoria en la relación con su entidad hija, cuando cada una de las instancias de la entidad padre se asocia con al menos una instancia de la entidad hija.

Ejemplo:



Considerando la siguiente variante entre asesor y alumno.

- Un profesor **debe** ser asesor de uno o hasta 3 alumnos.
 - Un alumno **puede** solicitar a un único asesor durante su estancia en la escuela.



- Ahora nos concentraremos en revisar las reglas de negocio con respecto a la entidad padre. Para determinar el tipo de participación podemos formular la siguiente pregunta:

¿Puede existir una instancia de asesor sin asociarse con alumno?

- En este caso la respuesta es NO, ya que cada profesor debe asesorar al menos a un alumno. Por lo tanto, la entidad ASESOR tiene **participación obligatoria**.
 - Observar la cardinalidad del lado derecho, cuando una entidad padre tiene participación obligatoria, el valor mínimo de cardinalidad deber ser **1**. Adicionalmente, observar la notación, en crow's foot tiene el símbolo  y en IDEF1X, solo se escribe la “P”.

Es importante no confundir con el concepto de dependencia de existencia, es decir, no importa si la FK está definida como null o not null.

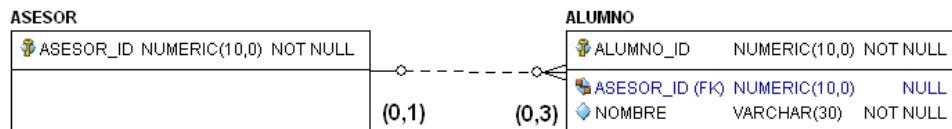


La base de datos **no tiene** forma de restringir esta condición, solo se representa empleando las notaciones y observando el valor mínimo de la cardinalidad del lado derecho.

Ejemplo:

Considerando la siguiente variante entre asesor y alumno.

- Un profesor **si así lo desea**, puede ser asesor máximo de 3 alumnos.
 - Un alumno **puede** solicitar a un único asesor durante su estancia en la escuela.



Aplicando la misma pregunta:

¿Puede existir una instancia de asesor sin asociarse con alumno?

- En este caso la respuesta es SI, ya que las reglas nos indican que, si un profesor no lo desea, no asesora, pueden existir instancias de `asesor` que no se relacionen con `alumno`. Por lo tanto, la entidad `asesor` tiene **participación opcional**.
- Observar que el valor mínimo de la cardinalidad del lado derecho en una participación opcional es 0. Nuevamente, no confundir con el concepto de dependencia de existencia, no importa la forma en la que se define la FK (`not null` o `null`).

Ejercicio en clase 3:



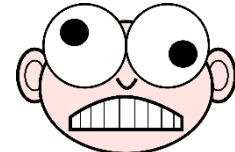
Considerar nuevamente el enunciado del ejercicio en clase 1 que hace referencia a una base de datos empleada para guardar los datos de los profesores aspirantes de un proyecto.

- Asignar las cardinalidades a cada una de las relaciones obtenidas.
- Para cada relación, aplicar los conceptos de dependencia e independencia de existencia, participación opcional y obligatoria.

4.3.6. Dependencia de identificación

- Representa una extensión del concepto de dependencia de existencia.
- Este tipo de relación se verifica en la tabla hija al igual que en dependencia de existencia.
- La tabla hija define su propia PK, pero sus valores no son suficientes para poder garantizar unicidad, es decir, la tabla hija requiere de la PK de la tabla padre para funcionar correctamente, formando una **PK compuesta**.
- Esta situación se produce generalmente cuando los valores de la PK original de la tabla hija se reinician o se repiten por cada valor de la PK de la tabla padre.

Ejemplo:



- Una librería cuenta con un catálogo de libros.
- Un libro se identifica de manera única con su ISBN
- Por cada ejemplar que se compra de dicho libro, se guarda un registro en base de datos identificándolo por su número iniciando en 1, por ejemplo, el ejemplar 1, del libro con ISBN 12, el ejemplar 2 del libro con ISBN 12, el ejemplar 1 del libro con el ISBN 13, etc.
- Se observa que el número de ejemplar se puede repetir por cada libro, es decir, ejemplar 1 del libro 12, el ejemplar 1 del libro 13, etc.
- Por lo anterior, la llave primaria de la entidad `Ejemplar` estará formada por la pareja <ISBN, numEjemplar> para garantizar combinaciones únicas en `ejemplar`.

Diseño conceptual	Diseño lógico
<p>Diseño conceptual</p> <pre> classDiagram class Libro { ISBN titulo } class Ejemplar { Num. ejemplar fecha entrega } Libro "1:m" -- "(1,*)" Ejemplar : cuenta con </pre> <p>Para hacer énfasis en que la llave primaria de la entidad <code>Libro</code> debe propagarse hacia la entidad <code>Ejemplar</code>, el rombo y el rectángulo de la tabla hija se marca con doble línea.</p>	<p>Diseño lógico</p> <pre> classDiagram class LIBRO { ISBN VARCHAR(30) NOT NULL TITULO VARCHAR(30) NOT NULL } class EJEMPLAR { NUM_EJEMPLAR NUMERIC(10,0) NOT NULL ISBN(FK) VARCHAR(30) NOT NULL FECHA_ENTREGA DATE NOT NULL } LIBRO "1,1" -- "0,1" EJEMPLAR : </pre> <p>Observar que la PK de <code>LIBRO</code> se propaga a <code>EJEMPLAR</code> empleando una relación identificativa. De esta forma es posible identificar de forma única a cada ejemplar</p>

- Los datos se verían así:

ISBN (PK)	NUM_EJEMPLAR_ID (PK)(FK)
1001	1
1001	2
1001	3
1002	1
1002	2
1003	1

Ejercicio en clase 4:

Generar el modelo ER y el modelo relacional para el siguiente enunciado:



- En una sala de teatro se reparten boletos foliados por cada función iniciando en 001.
- Adicional al folio, se almacena el importe del boleto, el nombre de la función y el número de asiento.
- Un cliente puede comprar varios boletos, se almacena el nombre, apellidos y el email (único por cliente).
- Se desea almacenar en la base de datos, los boletos que ha comprado cada cliente.

4.3.7. Grado de una relación.

Consiste en clasificar a las relaciones por el número de entidades (no instancias) que participan en una relación:

- Relaciones unarias
- Relaciones binarias
- Relaciones ternarias.

4.3.7.1. Relaciones unarias.

Ocurre al existir una relación empleando una sola entidad, es decir, una instancia de una entidad se relaciona con otra instancia de la **misma** entidad. A este tipo de relaciones también se les conoce como relaciones **recursivas**.



Ejemplo:

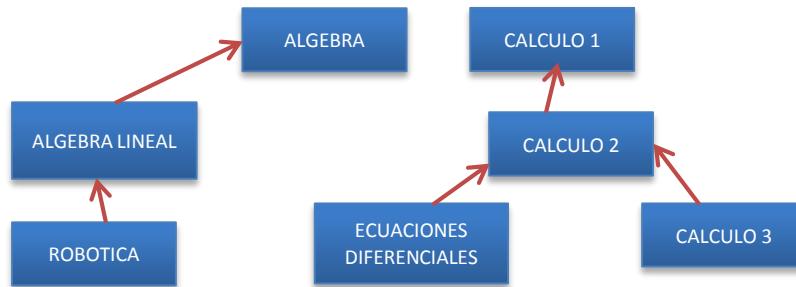
Considerar las siguientes reglas de negocio.

Asignatura y Asignatura antecedente. (1:M)

- Una asignatura puede requerir de una asignatura antecedente para poder cursarse. (1:1)
- Una asignatura puede ser antecedente de varias asignaturas. (1:M)

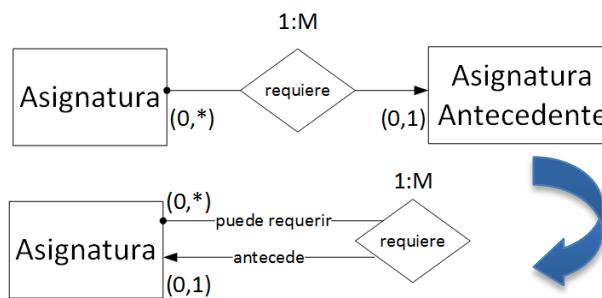
Estas 2 reglas de negocio se pueden reflejar en estos escenarios:

- La asignatura Álgebra lineal requiere de la asignatura Álgebra para que un alumno la pueda cursar.
- La asignatura Robótica requiere de la asignatura Álgebra lineal para poder cursarse.
- La asignatura Calculo 2 debe cursarse antes de cursar Calculo 3 y ecuaciones diferenciales



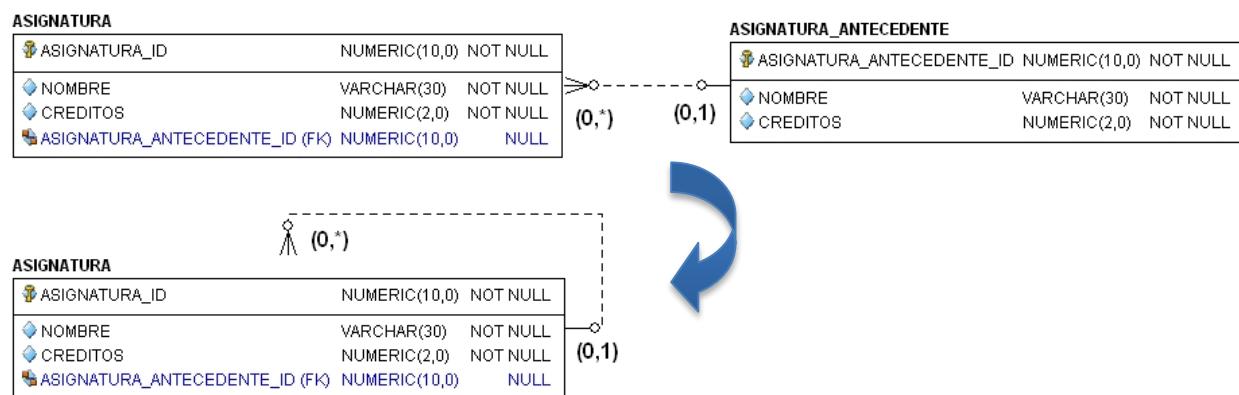
- Observar que este tipo de relación permite asociar instancias de una entidad en una estructura jerárquica, en donde la instancia raíz no se asocia con alguna otra instancia.
- La forma más sencilla es imaginar a las 2 entidades como si existieran por separado:

Diseño conceptual.



Una asignatura puede requerir de una asignatura antecedente para poder cursarse.
 Una asignatura puede ser antecedente de varias asignaturas.

Diseño lógico.



- Dependencia de existencia: Independiente. No todas las asignaturas (tabla hija) tienen asignatura antecedente (tabla padre). La FK es null
 - ¿Qué pasaría si la FK se declara como not null?
- Participación de una entidad: obligatoria. Toda asignatura antecedente (tabla padre) debe asociarse con al menos una asignatura subsecuente.

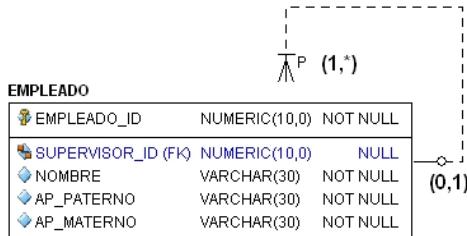
Los datos se verían así:

ASIGNATURA_ID	NOMBRE	CREDITOS	ASIGNATURA_REQUERIDA_ID
1	ALGEBRA	6	NULL
2	ALGEBRA LINEAL	7	1
3	CALCULO 1	6	NULL
4	CALCULO 2	7	3
5	CALCULO 3	7	4
6	ECUACIONES DIFERENCIALES	7	4
7	ROBOTICA	8	2

Ejemplo:

Entidad Empleado y Supervisor.

- A cada empleado se le asigna su supervisor el cual también es un empleado.
- Un supervisor puede tener asignados a varios empleados.



- En este caso se tiene una independencia de existencia. No todos los empleados tienen supervisor, es decir, los supervisores en si no tienen supervisor.
- Participación: obligatoria. Un supervisor debe tener al menos un empleado asignado.

Los datos se verían así:

EMPLEADO_ID	NOMBRE	AP_PATERNO	AP_MATERNO	SUPERVISOR_ID
1	JUAN	MARTINEZ	AGUIRRE	4
2	LILIANA	LOPEZ	HURTADO	NULL
3	GERARDO	JIMENEZ	BENITEZ	2
4	MARIA	MONDRAGON	MENDEZ	NULL
5	JORGE	LUNA	AGUILAR	2

4.3.7.2. Relaciones binarias.

Ocurren entre 2 entidades diferentes. Este tipo de relaciones son las que se revisaron en la sección 4.5

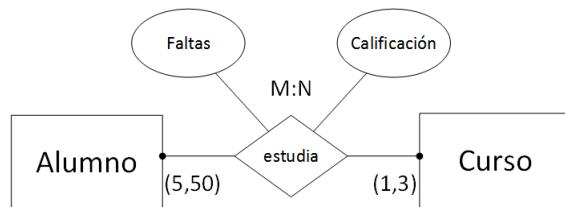
4.3.7.3. Relaciones ternarias.

Ocurren cuando se relacionan 3 entidades entre si para poder cumplir una regla de negocio. ¿En qué casos se podrían relacionar 3 entidades? Suponer el siguiente escenario:

Entidades Curso y Alumno (M:N).

- Un curso está integrado mínimo de 5 alumnos, máximo de 50 (1:M)
- Un alumno puede tomar uno o hasta 3 cursos. (1:M)
- Se requiere registrar la calificación y el número de faltas que obtuvieron los alumnos.

Diseño conceptual:



- Observar que los atributos que dependen de la combinación de ambas instancias se asocian al rombo y no a las entidades.
- Tanto la calificación como las faltas dependen de la combinación de los valores de cada alumno y de cada curso.

Diseño lógico

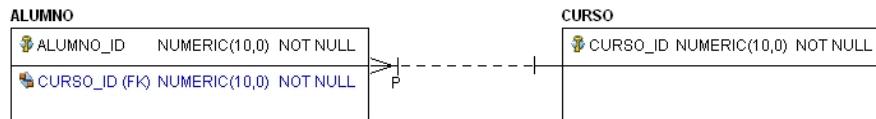
Si se intenta modelar una relación M:N asignando la FK en alguna de las 2 tablas ocurren las siguientes situaciones:

- FK en curso



En esta estrategia, se cumple la regla “un alumno puede tomar varios cursos” ya que un mismo `alumno_id` puede aparecer en cursos diferentes. Sin embargo, la regla “un curso está formado por varios alumnos” no se cumple. Como se puede observar en el diagrama, por cada curso únicamente se puede registrar a un alumno.

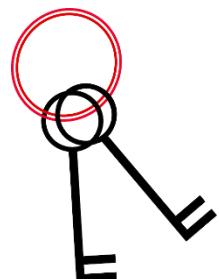
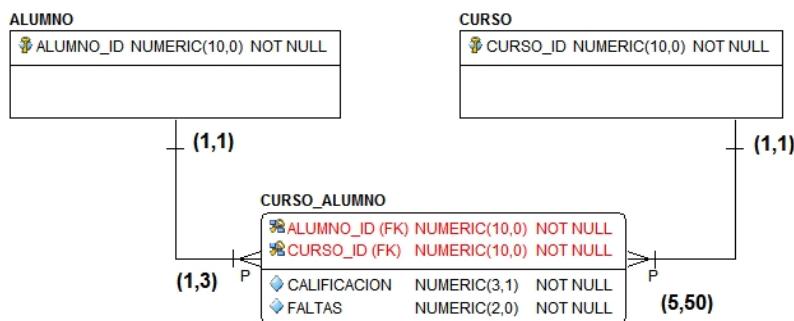
- FK en alumno



En esta estrategia, se cumple la regla “un curso está formado por varios alumnos”. El mismo `curso_id` puede aparecer en varios registros de la tabla `alumno`. Sin embargo, la regla “Un alumno puede tomar varios cursos” no se cumple, por alumno solo se puede registrar a un alumno.

Para resolver el problema anterior, en una relación M:N, se debe incorporar una tabla intermedia. El modelo relacional no soporta la implementación de relaciones M:N de forma directa.

Estrategia 1: PK compuesta.



- Observar en este caso la cardinalidad en ambos lados de la tabla intermedia:
 - Un alumno toma como mínimo un curso (participación obligatoria), y como máximo 3.
 - Un curso está formado por mínimo 5 alumnos (participación obligatoria), máximo 50.
- Observar que la dependencia de existencia no tiene sentido verificarla, las FKs forman parte de la llave primaria de la tabla intermedia, por lo tanto, no pueden ser definidas como NULL.
- En una relación M:N siempre existirá dependencia de existencia.
- La participación puede ser opcional u obligatoria.
- Los datos se verían así:

ALUMNO

ALUMNO_ID	NOMBRE
1	JUAN
2	MARIO
3	LAURA
4	LORENA

CURSO

CURSO_ID	NOMBRE
1000	PROGRAMACIÓN
1001	ALGORITMOS
1002	INTEGRACION
1003	COCINA

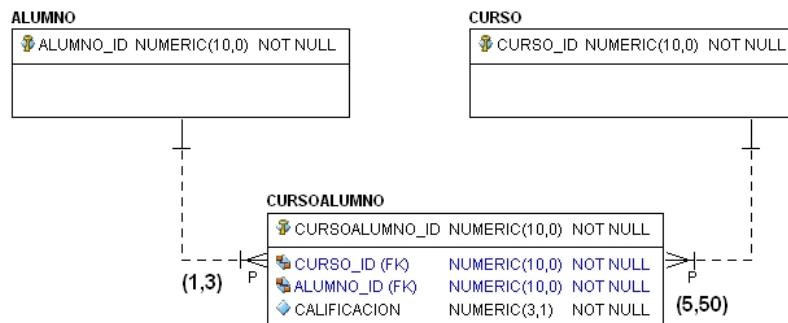


CURSO_ALUMNO

CURSO_ID(FK)	ALUMNO_ID(FK)	CALIFICACION
1000	1	10
1001	1	9
1002	2	8
1002	3	7
1003	4	9

- Observar que la tabla intermedia permite implementar correctamente las reglas de negocio al incorporar una PK compuesta.
 - Un alumno puede tomar varios cursos (primeros 2 registros)
 - Un curso está formado por varios alumnos (registro 3 y 4)
- Observar que los campos que pueden aparecer en la tabla intermedia dependen de la combinación de los valores de la PK. Por ejemplo, la calificación, es un atributo que depende tanto del curso como del alumno (para cada curso y para cada alumno se asigna una calificación).

Estrategia 2: PK artificial.



En esta estrategia se emplea una PK simple y artificial, y las 2 tablas se relacionan a través de relaciones no identificativas. Aunque la BD lo permite, las FKs no deben declararse como null por lo siguiente:

- ¿Qué sentido tendría un registro en la tabla intermedia en donde un alumno tuviera como valor de curso_id un valor nulo?
- ¿Qué sentido tendría un registro en la tabla intermedia en donde un curso tuviera un alumno_id con un valor nulo?
- ¿Qué pasa si un alumno no se inscribe a algún curso? Las FKs aun así no deben ser marcadas como null. En este caso, simplemente no existiría un registro en la tabla intermedia, por lo tanto, estaríamos hablando de una participación opcional, y la cardinalidad del lado izquierdo cambiaría a (0,3).
- Lo mismo ocurre con el curso. Si las reglas nos dijieran que pudiera existir un curso sin alumnos, no habría registros en la tabla intermedia asociados a dicho curso, las FKs siguen siendo definidas como not null y la cardinalidad cambiaría a (0,50).

Ventajas de esta estrategia:

- Al tener una PK artificial el desempeño tiende a mejorar ya que al realizar consultas y relacionar tablas, el procesamiento con llaves compuestas siempre es mayor al requerido para ligar tablas con PKs simples.
- Las consultas SQL requeridas para explotar los datos de esta tabla intermedia se simplifican con PKs simples (esto se verá en la parte de SQL).

Desventajas de esta estrategia:

- Se agrega un nuevo campo (la PK artificial), sin embargo, no representa un problema mayor en cuanto a espacio de almacenamiento.
- Posibilidad de insertar datos inconsistentes, situación que no ocurre con la estrategia anterior. En este caso, se le delega a la aplicación o al usuario insertar datos consistentes.

Ejemplo:

Los siguientes registros son inconsistentes. La base de datos no tiene forma de detectar o restringir la inconsistencia, a menos que se creara un trigger para verificar.

CURSO_ALUMNO

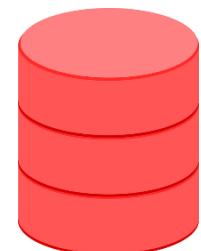
CURSO_ALUMNO_ID(PK)	CURSO_ID(FK)	ALUMNO_ID(FK)	CALIFICACION
1	1000	1	10
2	1000	1	9

- En este caso, el registro es duplicado, y lo peor es que se tienen 2 calificaciones distintas para el mismo alumno y curso.
- Una solución adicional, podría ser, crear un UNIQUE index que aplique a ambos campos, tanto a curso_id como a alumno_id

En conclusión, se recomienda más la segunda estrategia, a menos que hubiera algún requerimiento muy particular que lo impida.

4.4. TIPOS DE DATOS EMPLEADOS PARA REALIZAR MODELOS RELACIONALES.

Un punto importante dentro del diseño lógico es la selección del tipo de dato. Cuando se agrega cada una de las columnas de una tabla es indispensable indicar el tipo de dato que debe contener la columna. En esta sección se revisan los diferentes tipos de datos SQL que pueden ser empleados durante el proceso de diagramado de un caso de estudio.



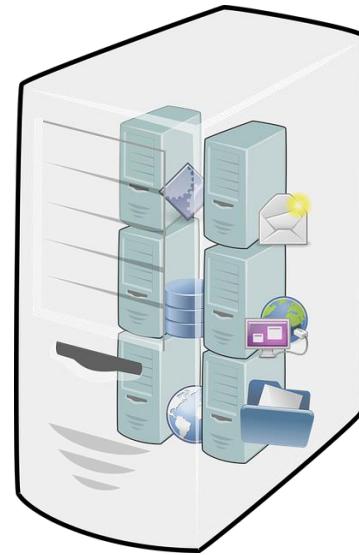
Adicional a las restricciones que pudiera tener una columna en una tabla, es importante definir el tipo de dato y su dominio. Dicha definición debe reflejar de forma correcta la naturaleza de los datos:

- Valores monetarios
- Fechas
- Cantidades
- Nombres, etc.

Para implementar este requerimiento, el estándar SQL ha definido una serie de tipos de datos que todo manejador debe implementar. La realidad es que no todos los manejadores cumplen al 100% con el estándar. Algunos definen variantes o sus propios tipos de datos.

Clasificación de los tipos de datos SQL:

- Cadenas
 - Cadenas de caracteres
 - De longitud fija
 - De longitud variable
 - Cadenas binarias
- Numéricos
 - Exactos
 - Aproximados
- Tiempo y fechas
- Otros:
 - boolean (Estándar, MySQL, PostgreSQL)
 - ROWID (Oracle)
 - UROWID (Oracle)
 - BFILE (Oracle)
 - XML



4.4.1. Cadenas de caracteres

Conjunto o secuencia de caracteres que pertenecen a un “Juego de caracteres” predefinido. La longitud de la cadena está definida por en número de caracteres que contiene. El número de bytes de espacio requerido para almacenar una cadena depende del **juego de caracteres** con el que se represente. Internamente, cada carácter es representado generalmente por un valor numérico que se obtiene a través de la aplicación de algún algoritmo de codificación. Algunas normas de codificación (juego de caracteres):

- ISO 646
- ASCII
- UNICODE
- ISO 8859
- Juego de caracteres de Windows
- UTF8

Ejemplos:

Carácter	ISO-8859-1	UTF-8	UTF-16
a	0x61	0x61	0x00 0x61
b	0x62	0x62	0x00 0x62

Uno de los puntos importantes al momento de instalar una base de datos, es la asignación del **juego de caracteres** que se empleará para almacenar los datos. Esto permitirá la correcta codificación, representación y almacenamiento de las cadenas de caracteres en la base de datos. Dependiendo de la diversidad de símbolos y caracteres que puedan ser enviados a la base de datos, se deberá elegir el juego de caracteres a configurar. Por default, se selecciona el juego de caracteres que emplea el sistema operativo.

4.4.1.1. Cadenas de caracteres de longitud fija:

Si un atributo en la BD se define como una cadena de longitud fija N, como máximo se podrán insertar N caracteres. Si la cadena es menor a N, esta se rellena con espacios.

Ejemplo: cadena de caracteres de longitud 8 (fija)

A	R	P	A				
M	U	S	I	C	A		
P	I	A	N	O			



El tipo de dato que representa a una cadena de longitud fija es CHAR. (Ver tabla siguiente).

Sintaxis:

CHAR [ACTER] [(n)]

EL contenido que esta entre [], es opcional. El valor de “n” representa al número de caracteres de la cadena. Si no se especifica, es equivalente a CHAR(1), un carácter.

Ejemplos:

Ejemplo	Descripción
CHAR(10)	Cadena de caracteres de longitud 10
CHAR	Cadena de caracteres de longitud 1

4.4.1.2. Cadenas de caracteres de longitud variable.

A diferencia de las cadenas de longitud fija, solo se almacena el número real de caracteres que contiene la cadena. A nivel de definición del campo solo se define la capacidad máxima.

Ejemplo: cadena de caracteres de longitud 8 (variable)

A	R	P	A				
M	U	S	I	C	A		
P	I	A	N	O			

El tipo de dato que representa a una cadena de longitud fija es VARCHAR. (Ver tabla siguiente).

Sintaxis:

VARCHAR (n)

En este caso, “n” define el número máximo de caracteres que puede tener la cadena: VARCHAR(1), VARCHAR(100), etc.

Para el caso particular de Oracle, se recomienda emplear VARCHAR2 (este será el tipo de dato a emplear en el curso para las prácticas de SQL).

4.4.1.3. Cadenas de caracteres de gran tamaño.



En algunas ocasiones no se conoce con precisión el tamaño máximo que pudiera tener una cadena a insertar en la base de datos. Lo único que se conoce es que pueden ser cadenas muy grandes. Para estos casos, existe un tipo de dato CLOB (Character Large Object).

Se emplea para almacenar grandes cantidades de caracteres, textos, artículos, contenido de un libro, etc.

4.4.1.4. Cadenas de caracteres en diferente juego de caracteres.

Los tipos de datos NCHAR Y NCHAR VARING (N) , NVARCHAR2 (para Oracle) se emplean para guardar cadenas representadas en un juego de caracteres diferente al configurado en la base de datos. Para el caso de Oracle, se emplea como juego de caracteres UNICODE, aunque el estándar define CHARACTER VARING (n) CHARACTER SET <name> en el que se especifica el nombre del juego de caracteres a emplear, Oracle siempre emplea UNICODE. La principal razón: UNICODE es el juego de caracteres universal, por lo que se puede representar cualquier carácter.

4.4.1.5. Clasificación de los tipos de datos para representar cadenas de caracteres

La siguiente tabla muestra la clasificación de las cadenas de caracteres. Del lado izquierdo, representa al tipo de dato definido por el estándar SQL, y las demás columnas representan la forma en la que cada manejador implementa al tipo de dato.

SQL Estándar (especificación)	ORACLE 11g	DB2 9.5	SQL Server 2008	PostgreSQL 8.x	MySQL 5.x
CHAR[ACTER] [(n)]	CHAR[ACTER] [(n)]	CHAR[ACTER] [(n)]	CHAR[ACTER] [(n)]	CHAR[ACTER] [(n)]	CHAR [(M)]
CHAR[ACTER] VARING(n) ó VARCHAR(n)	CHAR[ACTER] VARING(n) ó VARCHAR(n) ó VARCHAR2 (n)	CHAR[ACTER] VARING(n) ó VARCHAR(n) ó LONGVARCHAR	CHAR[ACTER] VARING(n) ó VARCHAR [(n)] TEXT	CHAR[ACTER] VARING(n) ó VARCHAR(n)	VARCHAR(M) ó TINYTEXT ó TEXT ó MEDIUMTEXT ó LONGTEXT
CLOB ó CHARACTER LARGE OBJECT	CLOB ó LONG[VARCHAR]	CLOB [(n) [K M G]]	VARCHAR(MAX)	TEXT	BINARY [(M)] VARBINARY(M)
NATIONAL CHAR[ACTER] [(n)] ó NCHAR[(n)] ó CHARACTER [(n)] CHARACTER SET <charset-name>	NATIONAL CHAR[ACTER] [(n)] ó NCHAR [(n)]	GRAPHIC [(n)]	NATIONAL CHAR[ACTER] [(n)] ó NCHAR [(n)] ó NTEXT	-----	NATIONAL CHARACTER(n) ó CHAR(n) CHARACTER SET <name> ó NCHAR(n)
NATIONAL CHAR[ACTER] VARING(n) ó NCHAR VARING(n) ó CHARACTER VARING(n) CHARACTER SET <name>	NATIONAL CHAR[ACTER] VARING(n) ó NCHAR VARING(n) ó NVARCHAR2 (n)	VARGRAPHIC(n) ó LONG VARGRAPHIC(n)	NATIONAL CHARACTER VARING [(n)] ó NTEXT	-----	VARCHAR(n) CHARACTER SET <name> ó NATIONAL VARCHAR(n) ó NCHAR VARCHAR(n) ó NATIONAL CHAR[ACTER] VARING(n)
NATIONAL CHARACTER LARGE OBJECT	NCLOB	DBCLOB [(n) [K M G]]	NVARCHAR(MAX)	-----	-----

- Las palabras entre [] son opcionales, las palabras entre <> se sustituyen por un valor.

4.4.1.6. Rangos soportados por los principales manejadores para cadenas de caracteres

Tipo de dato	Oracle
CHAR	1-2000
VARCHAR2	1-4000
NCHAR (unicode)	1-2000
NVARCHAR (unicode)	1-4000
CLOB	8 TB
NCLOB	8 TB

Tipo de dato	DB2
CHAR	1-254
VARCHAR	1-32672
LONG VARCHAR	1-32700
CLOB	2 GB
GRAPHIC	1-127
VARGRAPHIC	1-16, 336
LONG VARGRAPHIC	16, 350

Tipo de dato	SQL Server
CHAR	1-8000
VARCHAR	1-8000
TEXT	2 GB
NCHAR NVARCHAR	1-4000
NTEXT	1 GB
VARCHAR (MAX) NVARCHAR	2 GB

Tipo de dato	MySQL
CHAR	1-255
VARCHAR	1-65535
TINYTEXT	0-255
TEXT	0-65535
MEDIUM TEXT	0-16777215
LONGTEXT	0-4294967295

Tipo de dato	PostgreSQL
CHAR	1 GB
VARCHAR	1-GB
TEXT	1-GB



4.4.2. Cadenas binarias

Se emplean para almacenar secuencia de bytes en la base de datos, es decir, se almacenan archivos binarios: fotos, música, videos, huellas, documentos, etc. El tipo de dato más común empleado es BLOB (Binary Large Object), de forma similar a CLOB, no es necesario escribir la longitud máxima que puede almacenarse en la BD.

Clasificación:

SQL Estándar (especificación)	ORACLE 11g	DB2 9.5	SQL Server 2008	PostgreSQL 8.x	MySQL 5.x
BLOB ó BINARY OBJECT LARGE	BLOB ó LONGRAW ó RAW(n)	BLOB[(n) [K M G]]	VARBINARY (MAX) ó VARBINARY[(n)] ó IMAGE	BYTEA	BINARY[(n)] ó VARBINARY(M) ó TINYBLOB ó BLOB ó MEDIUM BLOB ó LONG BLOB

4.4.2.1. Rangos cadenas binarias.

Tipo de dato	Oracle
BLOB	8 TB
RAW	4000 BYTES

Tipo de dato	DB2
BLOB	2 GB

Tipo de dato	SQL Server
BINARY	8000
VARBINARY	8000
IMAGE	2147483647
VARBINARY (MAX)	2 GB

¡Observar que en el caso de Oracle es posible almacenar hasta un archivo de 8 TB!

4.4.3. Tipos de datos numéricos.

Existen 2 clasificaciones de los tipos de datos numéricos:

- Tipos de datos numéricos exactos
- Tipos de datos numéricos aproximados.



4.4.3.1. Tipos de datos numéricos exactos

Los tipos de datos numéricos exactos pueden ser números enteros finitos o números con parte decimal finita.

Al definir un atributo de una tabla con un tipo de dato numérico exacto se puede definir 2 elementos:

- Precisión: Número total de dígitos: Parte entera + parte decimal
- Escala: Número de dígitos que formarán a la parte decimal.

Por lo anterior, el número máximo de dígitos que podrá tener la parte entera de un valor numérico será:

```
#digitosParteEntera = precisión - escala
```

Ejemplo.

¿Cuál será el dominio de la siguiente columna? precio number (10, 4)

Precision = 10

Escala = 4

Parte entera = 10-4 = 6

Dominio: [0, 999999.9999]



4.4.3.2. Clasificación de los tipos de datos numéricos exactos.

SQL Estándar (especificación)	ORACLE 11g	DB2 9.5	SQL Server 2008	PostgreSQL 8.x	MySQL 5.x
INT [TEGER]	NUMBER (n)	INT [TEGER] ó BIGINT	INT [TEGER] ó BIGINT	INTEGER ó BIGSERIAL ó SERIAL ó BIGINT	INT [(M)] ó MEDIUMINT [(M)] ó BIGINT [(M)]
SMALLINT	SMALLINT NUMBER (n)	SMALLINT	SMALLINT ó TINYINT	SMALLINT	SMALLINT [(M)] ó TINYINT [(M)]
NUMERIC [(P[,S])] ó DECIMAL [(P[,S])]	NUMERIC [(P[,S])] ó DECIMAL [(P[,S])] ó NUMBER [(P[,S])]	NUMERIC [(P[,S])] ó DECIMAL [(P[,S])]	NUMERIC [(P[,S])] ó DECIMAL [(P[,S])] MONEY SMALLMONEY	NUMERIC [(P[,S])] ó DECIMAL [(P[,S])]	

4.4.3.3. Tipos de datos numéricos aproximados.

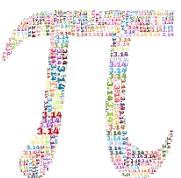
Se emplean cuando no se conoce con exactitud los valores de la precisión y/o escala que puede tener los valores de una columna.

4.4.3.4. Clasificación de los tipos de datos numéricos aproximados.

SQL Estándar (especificación)	ORACLE 11g	DB2 9.5	SQL Server 2008	PostgreSQL 8.x	MySQL 5.x
FLOAT [(P)]	FLOAT [(P)] ó NUMBER	FLOAT [(P)]	FLOAT [(P)]	-----	FLOAT [(P[,S])]
REAL	REAL NUMBER	REAL	REAL	REAL	REAL [(P[,S])]
DOUBLE PRECISION	DOUBLE PRECISION NUMBER	DOUBLE [PRECISION]	DOUBLE PRECISION	DOUBLE PRECISION	DOUBLE [PRECISION]

4.4.3.5. Rangos tipos de datos numéricos para Oracle:

- INTEGER, SMALLINT se convierten a NUMBER (38)
- NUMERIC, DECIMAL se convierten a NUMBER



- Rango: 1×10^{-130} a 9×10^{125} (38 nueves).
- Oracle permite la definición de escalas negativas empleadas para redondear. Por ejemplo:
 - NUMBER (10, 2), 6,345,768,903.678 será redondeado a 6,345,768,904

Observar que, para el caso de Oracle, en todos los tipos que define el estándar, Oracle lo representa a través del tipo de dato NUMBER. Por lo anterior, se recomienda que todos los tipos de datos numéricos (tanto exactos como aproximados) sean representados empleando NUMBER.

4.4.4. Tipos de datos para el manejo de fechas.

Esta clasificación de tipos de dato es la más complicada en el sentido de que prácticamente ningún manejador sigue el estándar SQL:

A nivel del estándar:

- DATE representa una fecha hasta el nivel de días (Año, Mes, Dia)
- TIME Representa tiempo a nivel de HH:MM:SS
- TIMESTAMP Representa una combinación: Año, Mes, Día, Hora, Minuto, Segundo, Milisegundo



Es importante recalcar que, en la mayoría de las aplicaciones o clientes gráficos empleados para consultar los datos, estos emplean un formato predefinido para representar a una fecha. Ejemplos:

- 02/03/08
- 02-03-2008
- 02 de marzo de 2008, etc.

Lo anterior no significa que en la base de datos una fecha se almacene como una cadena con un determinado formato. Normalmente las fechas se almacenan internamente como datos numéricos.

4.4.5. Otros tipos de datos.

4.4.5.1. Clasificación de los tipos de datos para manejo de fechas.

SQL Estándar (especificación)	ORACLE 11g	DB2 9.5	SQL Server 2008	PostgreSQL 8.x	MySQL 5.x
DATE	DATE	DATE	DATETIME ó SMALLDATETIME	DATE	DATE
TIME [WITH TIME ZONE]	DATE	TIME	DATETIME ó SMALLDATETIME	TIME [(P)] [WITH [OUT] TIMEZONE]	TIME
TIMESTAMP [(P)] [WITH TIME ZONE]	DATE ó TIMESTAMP [(P)] [WITH TIME ZONE]	TIMESTAMP	DATETIME SMALLDATETIME	TIMESTAMP [(P)] [WITH [OUT] TIMEZONE]	DATETIME TIMESTAMP
INTERVAL	INTERVAL DAY [(P)] TO SECOND [(P)] ó INTERVALYEAR [(P)] TO MONTH	-----	-----	INTERVAL [fields] [(p)] Fields: YEAR,MONTH,DAY,HOUR, etc.	YEAR

Observar nuevamente, para el caso de Oracle, se emplea el tipo de dato DATE para representar fechas, tiempo o ambas.

4.4.5.2. BOOLEAN

- Se agrega al estándar SQL 2003. Posibles valores para este tipo de dato: TRUE, FALSE.
- Oracle, DB2, SQL server no tienen este tipo de dato. Para el caso de Oracle se emplea NUMBER (1, 0). Típicamente 1 significa TRUE, 0 significa FALSE.

- En MySQL: BOOL, BOOLEAN, TINYINT (1)
- En PostgreSQL: BOOLEAN

4.4.5.3. ROWID

Como se revisó en temas anteriores, ROWID es exclusivo de Oracle, se emplea para almacenar una dirección única para cada registro de una tabla que se crea para localizar los datos de una forma directa.

4.4.5.4. UROWID

Similar a ROWID, solo que este se emplea para tablas indexadas.

4.4.5.5. BFILE

Tipo dato exclusivo de Oracle, se emplea para leer datos binarios que están almacenados fuera de la base de datos (se considera una especie de puntero hacia la ubicación física del archivo).

4.4.5.6. XML:

Tipo de dato empleado para almacenar documentos XML. Permite el manejo y explotación directa del documento a través de SQL en combinación de lenguaje empleado para navegar sobre los elementos de un documento XML.

4.5. MODELADO DE CASOS DE ESTUDIO

En esta sección se revisará el modelado de casos de estudio empleando los conceptos vistos en este capítulo. El objetivo es la creación de un modelo de datos que cumpla con las reglas de negocio especificadas en cada caso.

4.5.1. Base de datos para una empresa que administra colegios.



Las siguientes reglas de negocio resultaron de las entrevistas con los dueños de una empresa que administra colegios particulares la cual ha solicitado el diseño de una base de datos.

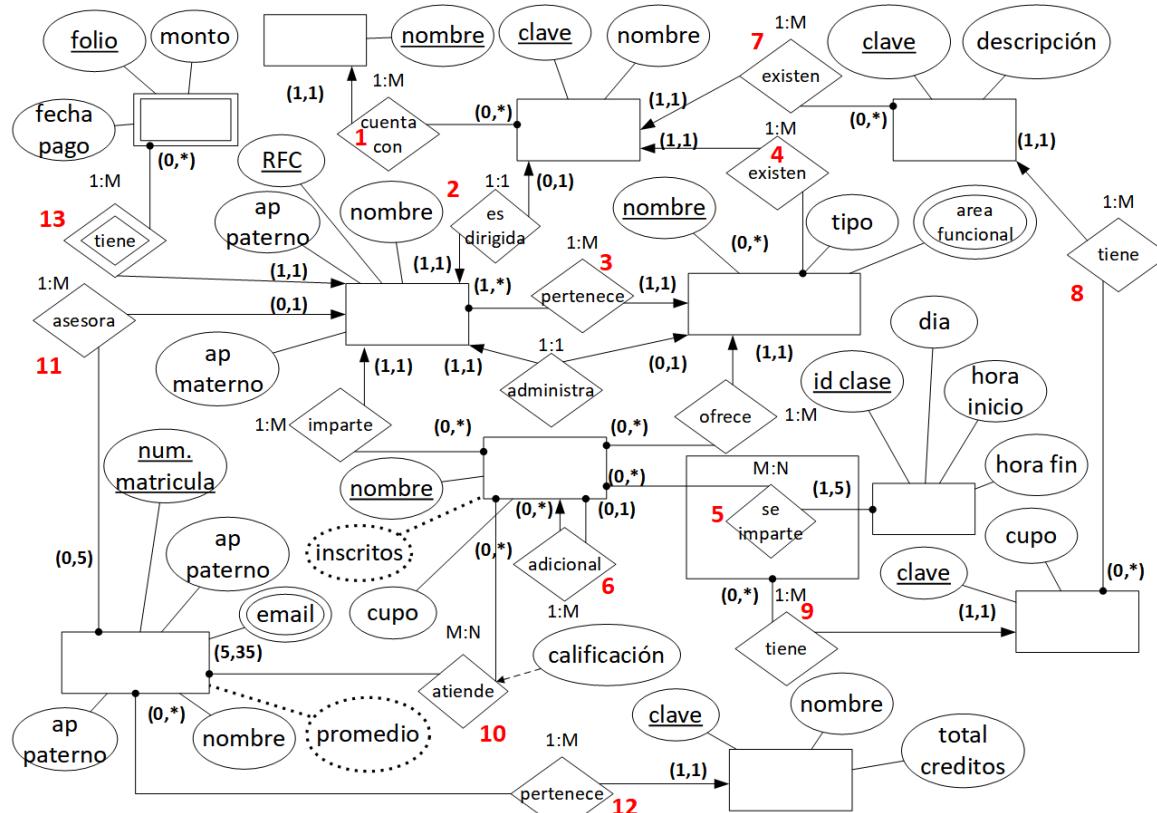
Ejercicio en clase 5:



- Generar una lista de entidades candidatas, realizar el análisis correspondiente para descartar falsas entidades.
 - Considerar el modelo ER mostrado a final del enunciado. Analizar los elementos de diseño, escribir los nombres de las entidades según corresponda.
 - Realizar el diseño lógico empleando para ello la transformación del modelo E/R obtenido en el punto anterior a un modelo relacional con notación crow's foot.
1. Cada colegio está formado por varias escuelas. De cada colegio se requiere almacenar su nombre.
 2. Para cada escuela se requiere almacenar el nombre y su clave. Cada escuela es administrada por un director el cual a su vez es un profesor. Cada director puede administrar una sola escuela. Para un profesor no es requisito que administre una escuela.
 3. Cada escuela está formada por varios departamentos. Se requiere almacenar el nombre del departamento.
 4. Cada departamento está dividido en un conjunto de áreas funcionales. Se desea almacenar los nombres de las áreas funcionales del departamento.
 5. Cada departamento ofrece cursos; otros departamentos se clasifican, por ejemplo, como departamentos de investigación por lo que no ofrecen cursos. Se debe almacenar el tipo de departamento ("de investigación", o "regular"). Cuando se registra el curso se debe indicar en

nombre del curso, el cupo máximo, número de alumnos inscritos y el departamento al que pertenece.

6. Cada curso se imparte en varias clases a la semana. Para cada clase se indica la hora inicio, hora fin, el día de la semana y el salón (un mismo curso puede impartirse en varios salones).
 7. Para cada curso, en una semana se pueden dar como mínimo una clase, y como máximo 5 clases. El curso lo imparte un único profesor.
 8. El colegio requiere guardar los siguientes datos de cada salón: clave de 5 caracteres, y cupo máximo. No en todos los salones se imparten cursos.
 9. Para contar con una mejor comprensión de un curso, en algunos casos se recomienda tomar un curso previo. Para dichos casos se requiere asociar el curso recomendado.
 10. Cada profesor pertenece a un departamento. Uno de estos profesores es el encargado de administrarlo (Jefe de departamento). Los datos que se registran de un profesor son: Nombre, apellidos, y RFC. Algunos profesores no imparten cursos.
 11. Un estudiante puede estar inscrito en 1 y hasta en 6 cursos en un mismo periodo. Cada curso puede tener como mínimo 5 estudiantes, y como máximo 35. Al final del periodo se registra la calificación del estudiante. Se requiere almacenar nombre, apellidos del estudiante y número de matrícula. El estudiante debe proporcionar al menos un correo electrónico como medio de contacto.
 12. Cada estudiante pertenece a una carrera. El colegio cuenta con un catálogo de carreras en el que se almacena clave de la carrera, nombre de la carrera, y total de créditos.
 13. Cada estudiante puede tener un asesor, el cual es un profesor. Solo aquellos profesores que así lo deseen pueden asesorar hasta 5 estudiantes.
 14. Cada clase se imparte en un salón, y cada salón se encuentra localizado en un edificio perteneciente a la escuela. Se requiere almacenar los datos del edificio: Clave de 2 letras, y descripción. No en todos los edificios se imparten cursos.
 15. Cada año un asesor puede recibir varios bonos por su buen desempeño. Se requiere registrar la fecha de pago, el monto del bono y un número consecutivo (folio de pago) que indica el número de pago. El folio está formado por 5 dígitos. Ejemplo: 00001 para el primer pago, 0002 para el segundo pago, etc.



TEMA 5
MODELO DE DATOS EXTENDIDO



5.1. ENTIDADES SUPERTIPO Y SUBTIPO

Para entender estos 2 conceptos dentro del modelado de bases de datos, considerar el siguiente escenario. Suponer que se tiene la siguiente lista de entidades con sus respectivos atributos:

- Profesor (**nombre, apellido paterno, apellido materno, edad, email, RFC, tipo profesor**)
- Investigador (**nombre, apellido paterno, apellido materno, edad, email, cédula, total_articulos**)
- Administrativo (**nombre, apellido paterno, apellido materno, edad, departamento, horas_extras**)

Observaciones de las 3 entidades anteriores:

- Las 3 entidades tienen atributos en común: **nombre, apellido paterno, apellido materno y edad**.
- Cada entidad tiene atributos que son particulares, por ejemplo, el RFC solo aparece en la entidad **profesor**.

Para este escenario, existe una forma particular de realizar el modelado de datos aplicando una especie de “factorización” de los atributos comunes, similar al concepto de herencia dentro de la programación orientada a objetos (POO).

- Los atributos comunes se extraen y se asocian a una entidad llamada “**supertipo**”. En este caso, es posible crear una nueva entidad llamada **persona** que tendrá el rol de **supertipo**.
- Las entidades **profesor, investigador y administrativo**, se les conoce como “**subtipos**” de la entidad **persona**.

5.2. IDENTIFICACIÓN DE UNA RELACIÓN DE SUPERTIPO - SUBTIPOS

Existen 2 estrategias para realizar la identificación:



5.2.1. Generalización: Proceso Bottom – Up

PROFESOR

◆ NOMBRE	VARCHAR(30)	NOT NULL
◆ AP_PATERNO	VARCHAR(30)	NOT NULL
◆ AP_MATERNO	VARCHAR(30)	NOT NULL
◆ EDAD	NUMERIC(2,0)	NOT NULL
◆ EMAIL	VARCHAR(150)	NOT NULL
◆ RFC	VARCHAR(13)	NOT NULL
◆ ESPECIALIDAD	VARCHAR(30)	NOT NULL

INVESTIGADOR

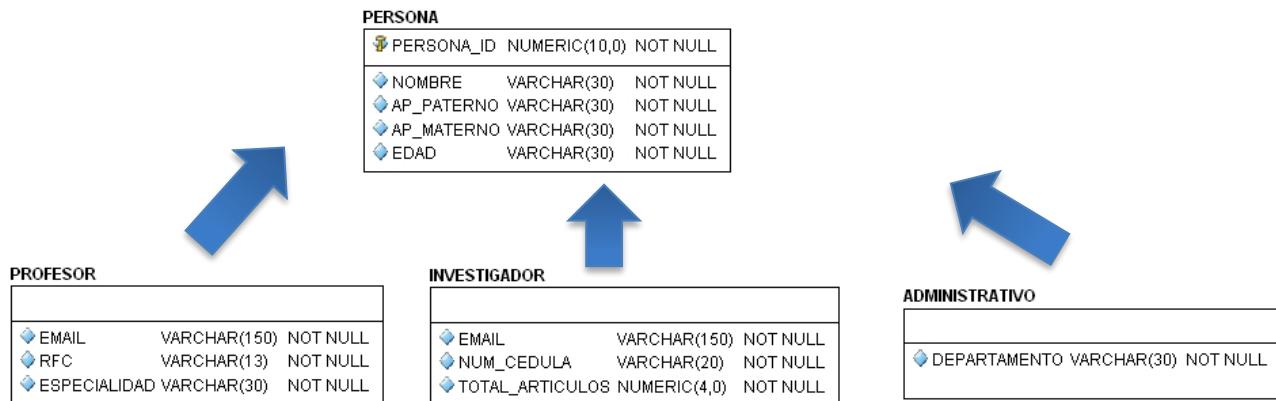
◆ NOMBRE	VARCHAR(30)	NOT NULL
◆ AP_PATERNO	VARCHAR(30)	NOT NULL
◆ AP_MATERNO	VARCHAR(30)	NOT NULL
◆ EDAD	NUMERIC(2,0)	NOT NULL
◆ EMAIL	VARCHAR(150)	NOT NULL
◆ NUM_CEDULA	VARCHAR(20)	NOT NULL
◆ TOTAL_ARTICULOS	NUMERIC(4,0)	NOT NULL

ADMINISTRATIVO

◆ NOMBRE	VARCHAR(30)	NOT NULL
◆ AP_PATERNO	VARCHAR(30)	NOT NULL
◆ AP_MATERNO	VARCHAR(30)	NOT NULL
◆ EDAD	NUMERIC(30,0)	NOT NULL
◆ DEPARTAMENTO	VARCHAR(30)	NOT NULL

- En este caso, se cuenta con la existencia de 2 o más entidades y se identifica la existencia de atributos en común.
- En este proceso, los atributos en común se extraen o se separan de las entidades y se agregan a una nueva entidad “general” que actuará como supertipo.

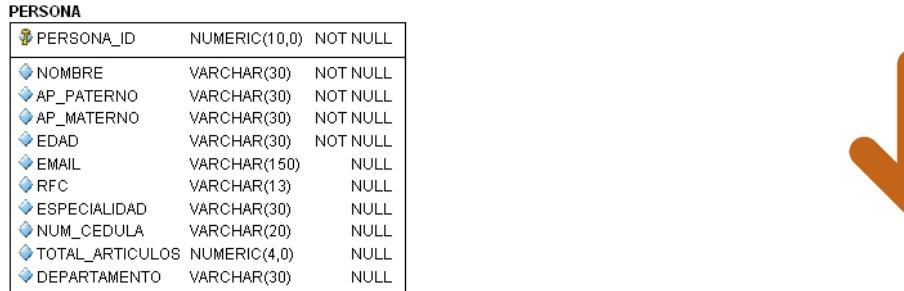
- La forma más sencilla es identificando los campos que se repiten en las 3 tablas. Al aplicar un proceso de generalización, se obtiene lo siguiente:



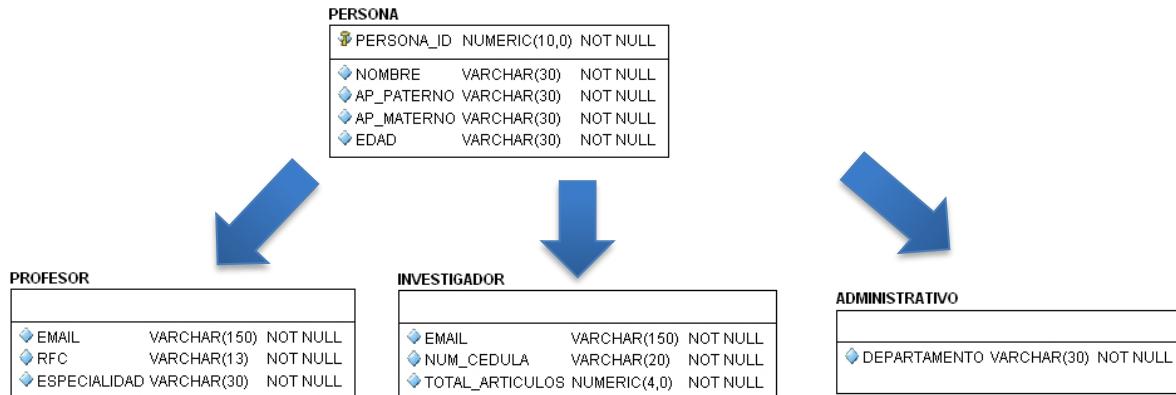
5.2.2. Especialización: Proceso Top - Down

- En este caso, se parte de la existencia de una sola entidad.
- Se observa que una entidad tiene ciertos atributos y/o interrelaciones que tienen sentido para ciertas instancias, pero no para otras, por lo que es conveniente definir uno o varios subtipos.

Ejemplo:



- Observar que los campos particulares de cada subtipo se declaran como NULL. Al crear un registro, algunos campos pueden carecer de valor.
- En este proceso los campos se extraen y se crea una nueva tabla subtipo. Debe existir al menos un campo para que la tabla tenga sentido:

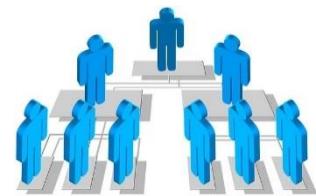


Para cualquiera de las 2 estrategias, verificar la correcta construcción del supertipo y sus subtipos, se deberán cumplir las siguientes condiciones:

- Todos los subtipos deberán contar con al menos un campo particular. No deben contener campos comunes.
- Al igual que en herencia, la existencia del subtipo se justifica aplicando la palabra “ES”. Por ejemplo, un profesor es una persona, un investigador es una persona, etc.

5.3. JERARQUÍAS DE ESPECIALIZACIÓN.

Observar que esta estrategia de diseño lleva a la construcción de una Jerarquía en este caso, jerarquía de tablas. Teóricamente es posible tener N niveles de Jerarquía, por ejemplo:



Con base a las 2 estrategias de identificación mencionadas anteriormente, se tiene un total de 3 posibles opciones de diseño:

5.3.1. opción 1: Nivel alto de Jerarquía.

En esta estrategia, se crean tantos niveles y subtipos como sean necesarios. En el ejemplo anterior, se tienen 3 niveles: 2 supertipos (**persona** y **administrativo**) con sus respectivos subtipos.

Desventajas:

- A mayor nivel de Jerarquía, el desempeño de la base de datos puede verse afectado, ya que se requiere un mayor número de operaciones `join` para ligar (relacionar) tablas. Por ejemplo, para obtener los datos de un contador, se requieren realizar operaciones `join` entre 3 tablas: contador, administrador y persona.
- Típicamente un nivel máximo de 2 podría considerarse como razonable en una base de datos productiva o transaccional.

Ventajas:

- A nivel de diseño, esta estrategia representa la mejor opción. Los atributos particulares a cada subtipo se definen como `not null`, existe un mayor control de la integridad de los datos.
- Adecuada cuando se tenga un número de atributos aproximadamente similar entre el supertipo y cada uno de sus subtipos.
- Para aplicaciones que hacen uso de la programación orientada a objetos que acceden a la base de datos, un nivel alto de jerarquía permite establecer una relación 1 – 1 con respecto al modelo de clases. Por ejemplo, a nivel de la aplicación puede existir esta misma jerarquía de clases, por lo que habrá una correspondencia directa entre una clase y una tabla: A la clase **Persona** le corresponde la tabla **persona**, y así sucesivamente.

5.3.2. Opción 2: Tabla simple sin subtipos:

En esta técnica se eliminan los subtipos y sus atributos son incorporados al supertipo como atributos opcionales, es decir, se aplica un proceso de **generalización**.

- Los atributos deben ser incorporados como nulos ya que no todas las instancias del supertipo estarán asociadas con cada subtipo: No todas las personas son profesores, o administradores, o investigadores.

Desventajas:

- Los campos particulares a cada subtipo se deberán definir como null. Esto puede generar problemas, por ejemplo, Un profesor se puede registrar sin su RFC. ¡La base de datos lo permite!

Ventajas:

- Se tiene el mejor resultado en cuanto a desempeño, no requiere ligar tablas.
- Adecuada cuando se tienen muy pocos atributos particulares por cada subtipo.

5.3.3. Opción 3: Conjunto de tablas sin su supertipo:

En esta técnica se elimina el supertipo, sus atributos son incorporados a cada uno de los subtipos. Esto significa que existirá cierta redundancia de atributos en cada subtipo. En este escenario se aplica el concepto de **especialización**.

PROFESOR

PROFESOR		
◊ NOMBRE	VARCHAR(30)	NOT NULL
◊ AP_PATERNO	VARCHAR(30)	NOT NULL
◊ AP_MATERNO	VARCHAR(30)	NOT NULL
◊ EDAD	NUMERIC(2,0)	NOT NULL
◊ EMAIL	VARCHAR(150)	NOT NULL
◊ RFC	VARCHAR(13)	NOT NULL
◊ ESPECIALIDAD	VARCHAR(30)	NOT NULL

INVESTIGADOR

INVESTIGADOR		
◊ NOMBRE	VARCHAR(30)	NOT NULL
◊ AP_PATERNO	VARCHAR(30)	NOT NULL
◊ AP_MATERNO	VARCHAR(30)	NOT NULL
◊ EDAD	NUMERIC(2,0)	NOT NULL
◊ EMAIL	VARCHAR(150)	NOT NULL
◊ NUM_CEDULA	VARCHAR(20)	NOT NULL
◊ TOTAL_ARTICULOS	NUMERIC(4,0)	NOT NULL

ADMINISTRATIVO

ADMINISTRATIVO		
◊ NOMBRE	VARCHAR(30)	NOT NULL
◊ AP_PATERNO	VARCHAR(30)	NOT NULL
◊ AP_MATERNO	VARCHAR(30)	NOT NULL
◊ EDAD	NUMERIC(30,0)	NOT NULL
◊ DEPARTAMENTO	VARCHAR(30)	NOT NULL

Ventajas:

- El desempeño no se afecta. Se cuenta únicamente con subtipos (un solo nivel).
- Los campos particulares se pueden definir como not null
- Adecuada para casos donde se tienen muy pocos atributos comunes.
- Adecuada cuando se tienen una relación de exclusión. Por ejemplo, una persona cuenta con un solo rol. Esto implica que no existirá redundancia de datos para los atributos en común, los cuales se encuentran en cada uno de los subtipos.

Desventajas:

- Los valores de los atributos comunes en cada tabla se repiten. Por ejemplo, suponer que una persona puede contar con varios roles a la vez, administrativo y profesor. Esto significa que existirá una instancia en la entidad administrativo y otra en profesor. Los valores de los atributos en común deberán repetirse en cada subtipo, por ejemplo, el nombre, apellidos, etc.

5.4. RELACIONES ENTRE UN SUPERTIPO Y SUS SUBTIPOS.

¿Qué tipo de relación existirá entre un supertipo y cada uno de sus subtipos?

- Una instancia de la tabla supertipo se asocia con 0 o máximo con un registro de alguno de sus subtipos. Por ejemplo, la persona con ID =1, no puede asociarse con 2 registros de la tabla profesor, ya que eso significaría que 2 profesores tienen la misma identidad. Por lo anterior, la relación de supertipo y subtipo es 1:1.

Existen 4 variantes para representar una relación entre un supertipo y su subtipo, se expresan a través de 2 tipos de restricciones:

5.4.1. Restricción Excluyente/traslape (*disjoint / overlapping*).

- **Excluyente:** Cada instancia del supertipo se asocia a lo más con una instancia de alguno de sus subtipos.
- **Traslape:** Cada instancia del supertipo puede asociarse con más de una instancia de sus subtipos

5.4.2. Restricciones Parciales o totales (*partial / complete*).

- **Total:** Cada instancia del supertipo se asocia por lo menos con una instancia de sus subtipos.
- **Parcial:** Cada instancia del supertipo puede o no asociarse con las instancias de sus subtipos.

Independiente a las restricciones anteriores, la cardinalidad entre un supertipo y cada uno de sus subtipos siempre será (0,1). Esto se revisa a detalle más adelante.

5.4.3. Discriminante de subtipo.

Considerar el siguiente escenario:

Se desea obtener todos los datos de una persona empleando como único criterio su identificador. Al emplear un diseño Supertipo -Subtipos surgen los siguientes puntos:

- ¿Cómo podríamos determinar con cuál de sus subtipos se relaciona cada instancia del supertipo? A partir del identificador no es posible determinar el subtipo(s) asociado(s) con la instancia del supertipo en cuestión. Se tendría que realizar una búsqueda del registro asociado en cada subtipo lo que puede generar inconvenientes de desempeño.
- Para resolver la pregunta anterior se emplea el concepto de “**discriminante**”.

El discriminante de subtipo puede ser implementado por uno o varios atributos, e inclusive por una entidad. Su estructura depende de las combinaciones entre las restricciones mencionadas anteriormente.



Ejemplos:

Restricción excluyente / total (*disjoint / complete*)

Una persona debe contar con un único rol: profesor, administrador o administrativo

Diseño conceptual	Diseño lógico
<ul style="list-style-type: none"> Se emplea el arco para indicar una restricción excluyente. Se asocia al triángulo invertido. Se emplea un círculo para indicar una restricción total. El discriminante se representa por un atributo. Tipo = P, A, I 	<ul style="list-style-type: none"> En IDEF1X Se emplea un círculo y al interior una letra "D" para indicar una restricción excluyente ("Disjoint"). Se emplea doble línea horizontal justo abajo del círculo para indicar una resticcion total. El discriminante se incorpora al supertipo.
	<ul style="list-style-type: none"> En Crow's foot se emplea un semi-círculo con una cruz al centro para representar una restricción excluyente. No existe notación para representar una restricción total.

Restricción excluyente / parcial (disjoint / partial)

El rol de una persona puede ser opcional. En caso de contar con rol, debe ser solo uno.

Diseño conceptual	Diseño lógico
<ul style="list-style-type: none"> En una restricción parcial se omite el círculo arriba del triángulo. Observar que el atributo que actúa como discriminante se vuelve opcional para los casos en los que la persona no cuente con roles asignados. 	<ul style="list-style-type: none"> En IDEF1X una restriccion parcial se representa por una sola línea que se ubica justo abajo del círculo. Observar que el discriminante es opcional.

Diseño conceptual	Diseño lógico

Traslape / Total (Overlapping / complete):

Una persona puede tener varios roles, mínimo uno.

Diseño conceptual	Diseño lógico
<ul style="list-style-type: none"> Se elimina el arco para representar una restricción de traslape. Se agrega un atributo multi – valorado ya que un mismo empleado puede tener más de un rol (tipo). 	<ul style="list-style-type: none"> En IDEF1X se emplea la letra "O" dentro del círculo para indicar una restricción de traslape. Observar que se debe agregar una nueva tabla para implementar el atributo multivalorado. Esta técnica funciona, pero requiere de una tabla adicional: tipo_persona. Para saber los roles de una persona se tendrá que hacer una operación join. Existe otra técnica que evita la creación de una tabla adicional la cual se muestra a continuación.

Opción 2: Eliminación de la tabla tipo_persona.

Diseño conceptual	Diseño lógico
<ul style="list-style-type: none"> Observar el uso de 3 nuevos atributos (uno por cada subtipo) que actúan como "banderas" para indicar los roles de una persona. 	<ul style="list-style-type: none"> Notar el uso de las 3 banderas, su tipo de dato es booleano.

Representación en Crow's foot

Diseño conceptual	Diseño lógico
	<p>PERSONA</p> <pre> PERSONA_ID NUMERIC(10,0) NOT NULL NOMBRE VARCHAR(30) NOT NULL AP_PATERNO VARCHAR(30) NOT NULL AP_MATERNO VARCHAR(30) NOT NULL EDAD VARCHAR(30) NOT NULL ES_PROFESOR BIT NOT NULL ES_ADMIN BIT NOT NULL ES_INVESTIGADOR BIT NOT NULL </pre> <p>PROFESOR</p> <pre> PERSONA_ID (FK) NUMERIC(10,0) NOT NULL EMAIL VARCHAR(150) NOT NULL RFC VARCHAR(13) NOT NULL ESPECIALIDAD VARCHAR(30) NOT NULL </pre> <p>ADMINISTRATIVO</p> <pre> PERSONA_ID (FK) NUMERIC(10,0) NOT NULL DEPARTAMENTO VARCHAR(30) NOT NULL </pre> <p>INVESTIGADOR</p> <pre> PERSONA_ID (FK) NUMERIC(10,0) NOT NULL EMAIL VARCHAR(150) NOT NULL NUM_CEDULA VARCHAR(20) NOT NULL TOTAL_ARTICULOS NUMERIC(4,0) NOT NULL </pre> <ul style="list-style-type: none"> En Crow's foot una restricción de traslape se representa por el semi -círculo sin la cruz al centro.

Traslape / parcial (Overlapping / partial):

Una persona puede o no tener roles. Pueden ser 0 o varios.

Diseño conceptual	Diseño lógico
<p>Empleado</p> <p>profesor</p> <p>Administrativo</p> <p>Investigador</p>	<p>PERSONA</p> <pre> PERSONA_ID NUMERIC(10,0) NOT NULL NOMBRE VARCHAR(30) NOT NULL AP_PATERNO VARCHAR(30) NOT NULL AP_MATERNO VARCHAR(30) NOT NULL EDAD VARCHAR(30) NOT NULL ES_PROFESOR BIT NOT NULL ES_ADMIN BIT NOT NULL ES_INVESTIGADOR BIT NOT NULL </pre> <p>PROFESOR</p> <pre> PERSONA_ID (FK) NUMERIC(10,0) NOT NULL EMAIL VARCHAR(150) NOT NULL RFC VARCHAR(13) NOT NULL ESPECIALIDAD VARCHAR(30) NOT NULL </pre> <p>ADMINISTRATIVO</p> <pre> PERSONA_ID (FK) NUMERIC(10,0) NOT NULL DEPARTAMENTO VARCHAR(30) NOT NULL </pre> <p>INVESTIGADOR</p> <pre> PERSONA_ID (FK) NUMERIC(10,0) NOT NULL EMAIL VARCHAR(150) NOT NULL NUM_CEDULA VARCHAR(20) NOT NULL TOTAL_ARTICULOS NUMERIC(4,0) NOT NULL </pre>

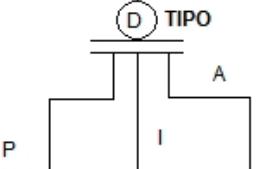
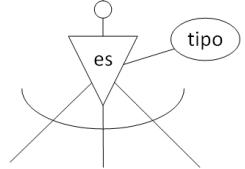
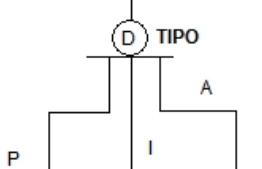
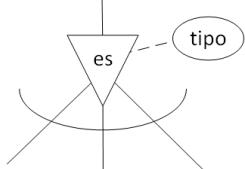
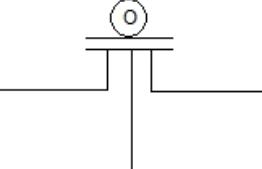
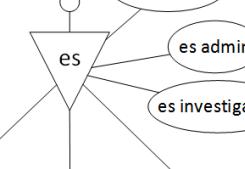
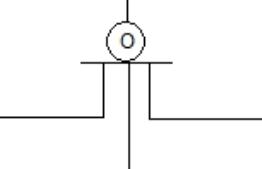
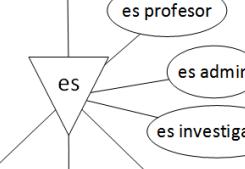
Diseño conceptual	Diseño lógico
<ul style="list-style-type: none"> Se elimina el círculo y el arco para indicar una restricción de traslape y parcial. Notar que las banderas siguen siendo obligatorias a pesar de ser una restricción parcial. Si se desea emplear la técnica con el atributo multivalorado, dicho atributo deberá ser opcional. <p>En este caso, se omite el círculo.</p>	<ul style="list-style-type: none"> En IDEF1X se emplea una sola línea paralela para indicar una relación parcial. Se emplea la letra “O” para indicar una restricción de traslape. <pre> erDiagram person --o{ professor : o} person --o{ administrativo : o} person --o{ investigador : o} } </pre>

- Observar que, en todos los casos, la PK del supertipo se propaga hacia los subtipos empleando una relación identificativa.
- La cardinalidad entre una instancia y una instancia de sus subtipos es siempre (0,1) ¿por qué razón?



5.4.4. Resumen de notaciones.

La siguiente tabla muestra el resumen de las diferentes combinaciones con sus respectivas notaciones y características del discriminante para cada caso.

Tipo de relación	Diseño lógico	Diseño conceptual	Discriminante.
Exclusiva/total			<p>tipo char (1) not null</p> <ul style="list-style-type: none"> • Observar las letras P, I, A, se suele especificar una letra (CHAR) como valor del campo para identificar a cada subtipo, se define como not null al ser una relación Total. • Observar la etiqueta “TIPO” que corresponde con el nombre del campo que actúa como discriminante.
Exclusiva/parcial			<p>tipo_persona char (1) null</p> <p>Observar que el campo debe ser NULL ya que, en este caso, la persona puede o no tener un rol asignado (relación parcial).</p>
Traslape/total			<p>es_admin bool not null es_profesor bool not null es_investigador bool not null</p> <ul style="list-style-type: none"> • Observar que en un tipo de relación de traslape se requiere un atributo booleano, uno por cada subtipo. • Si una persona tiene los 3 roles asignados, el valor de los 3 campos será true.
Traslape/parcial			<p>es_admin bool not null es_profesor bool not null es_investigador bool not null</p> <ul style="list-style-type: none"> • Observar que, a pesar de ser una relación parcial, los 3 campos se definen como not null. Si una persona no tiene rol asignado, el valor de los 3 campos será false.

5.5. MODELADO DE CATÁLOGOS Y DATOS CON HISTÓRICO.

En algunos casos, existen campos en una tabla que se actualizan con cierta frecuencia, y por cuestiones de control, se desea que se almacene cada uno de estos cambios, la fecha en la que se modificó, etc.

Ejercicio en clase 1:



Empleando el concepto de manejo y diseño de entidades e históricos realizar el diseño conceptual y lógico para el siguiente enunciado.

Una agencia de viajes aéreos desea llevar el control de los viajes que son adquiridos por sus clientes. Los datos del viaje son los siguientes:

- Lugar de salida (origen)
- Lugar de llegada (destino)
- Fecha y hora de salida

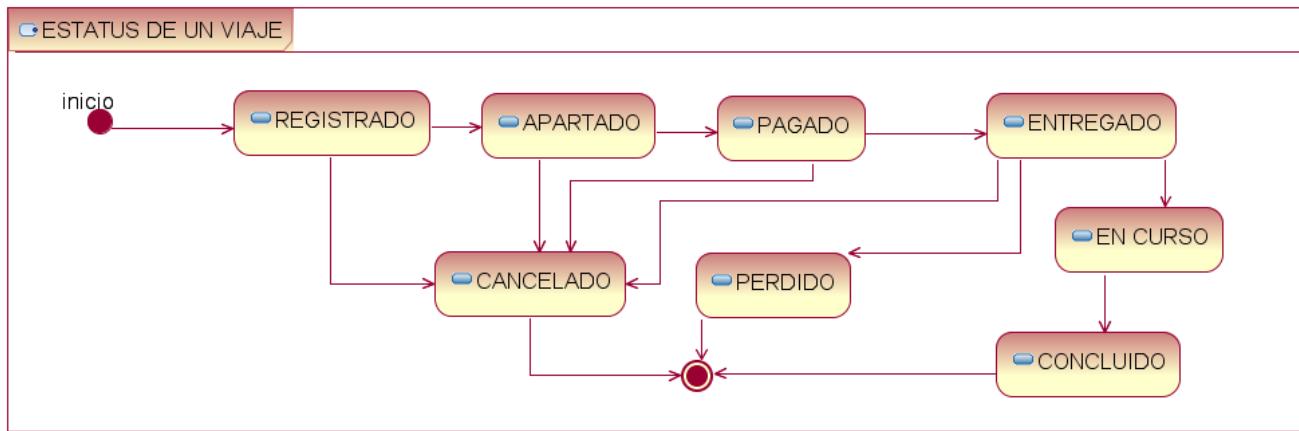
Cada destino y origen contiene una clave única de 3 caracteres, así como el nombre y descripción del lugar.

Ejemplo:



lugar_id	clave_lugar	descripción
1	AGS	AGUASCALIENTES
2	LCB	LOS CABOS
3	CAN	CANCUN

Adicional a esta información, para que un viaje se realice, debe pasar por las siguientes etapas:



1. REGISTRADO: El cliente captura en el sistema los datos del viaje: origen, destino, nombre y apellidos de cada pasajero y la fecha de salida. El viaje se considera como REGISTRADO.
2. APARTADO: El sistema le da como tiempo máximo 1hr para que el cliente seleccione los números de asientos del avión para cada pasajero y para que el cliente capture los datos de su tarjeta de crédito. Mientras el cliente no proporcione los datos de su tarjeta, y no expire el tiempo antes mencionado, el viaje permanece como APARTADO.
3. PAGADO: Un viaje se considera PAGADO, cuando el cliente capture su número de tarjeta de crédito en el sistema, y es validado contra el banco correspondiente. Si el banco autoriza el pago, el viaje se considera como PAGADO. En caso contrario, se le notifica al cliente, y si el cliente no corrige el error, el viaje se considera como CANCELADO.
4. ENTREGADO: El cliente deberá acudir como máximo, un día antes a la agencia para recoger sus boletos. En este momento el viaje se considera como ENTREGADO. Si no acude el viaje se CANCELA.
5. PERDIDO: Ocurre cuando el cliente ha recogido sus boletos, pero este no se presenta al aeropuerto a tomar su vuelo.
6. EN CURSO: El viaje toma este valor cuando se le notifica a la agencia que el cliente ha tomado el vuelo. Permanece en este estado hasta que el viaje termina.
7. CONCLUIDO. Un viaje se considera concluido cuando:

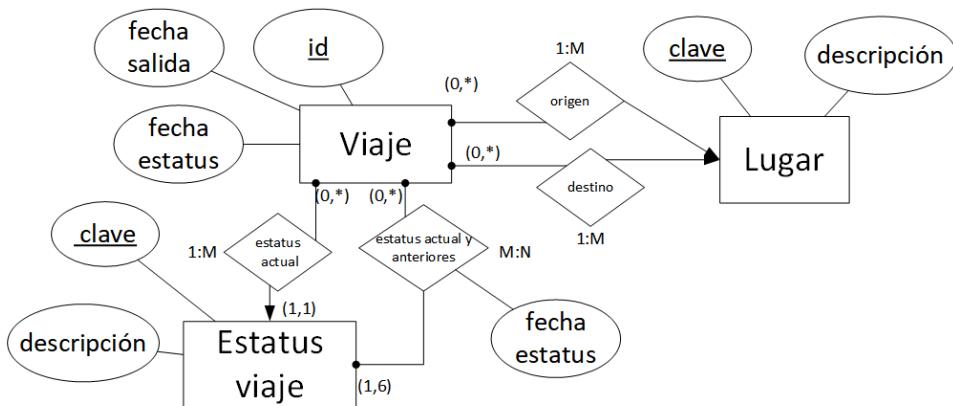
- a. El pasajero ha descendido del avión en el lugar destino para viajes sencillos.
 - b. El pasajero ha descendido del avión en el lugar origen para viajes redondos.
8. CANCELADO. Finalmente, el viaje se puede cancelar por las siguientes causas:
- a. El cliente no captura los datos de su tarjeta dentro del tiempo máximo permitido
 - b. El cliente no pudo comprobar su pago vía tarjeta de crédito.
 - c. La agencia cancela el viaje una vez que este ha sido ENTREGADO por alguna causa de fuerza mayor (fenómeno natural, fallas mecánicas, etc.)

La agencia requiere que se almacene en la base de datos, el estado actual que tiene cada viaje, así como la fecha en la que adquirió dicho estado.

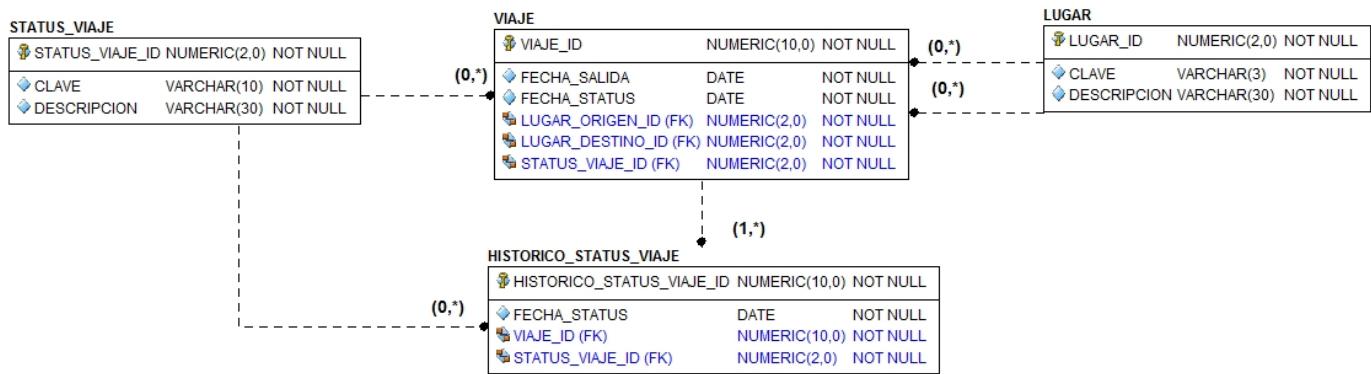
Se requiere adicionalmente almacenar toda la secuencia de estados que ha tenido un viaje, desde su inicio hasta que llegue a su estado terminal.

Respuesta:

Modelo ER:



El modelo relacional para este ejemplo se muestra a continuación:



5.5.1. Acerca de los catálogos.

- Observar el modelado de la tabla `lugar`. Se trata de una tabla que contiene los datos del catálogo de lugares. Este catálogo se considera como estático ya que sus valores no son modificados y tampoco se agregan nuevos.
- El uso de catálogos es adecuado cuando sus valores se emplean en varias tablas para mejorar consistencia, adicional a que cada valor del catálogo tiene sus propios atributos: `clave` y `descripción`.

- En cuanto a la cardinalidad, observar la participación opcional. Generalmente en una relación entre una entidad con un catálogo, la participación es opcional: la existencia de un elemento del catálogo no requiere la existencia de un registro en la tabla hija.

5.5.2. Acerca del histórico.

- Observar que en la tabla viaje se guarda el valor del “status” (`status_viaje_id`) actual del viaje y su fecha (`fecha_status`).
- Observar que estos 2 campos se duplican en `historico_status_viaje`
- Se emplea un catálogo estático para representar a cada uno de los estados del diagrama. A cada valor se le asigna su clave y su descripción. Se prefiere el uso de un catálogo ya que se emplea en 2 tablas.
- La forma en la que trabaja este modelo es la siguiente:
 - Cuando se crea un nuevo viaje, a este se le asocia su primer status y su fecha.
 - Una vez que el viaje ha sido creado, se agrega un nuevo registro en su histórico de status con los mismos valores asignados en la tabla VIAJE.
 - Cuando el viaje cambia de status, se realiza una actualización de los campos `status_viaje_id` y `fecha_status`.
 - Para no perder los valores de los campos actualizados, se agrega una nueva entrada al histórico. De esta forma se guardan los cambios generados para el campo `status_viaje_id`.
- En sentido estricto, **existe redundancia** entre los campos `status_viaje_id` y `fecha_status` de la tabla `viaje` con el último registro agregado en el histórico. A pesar de este detalle, el tener los valores en la tabla `viaje`, permite conocer de forma directa el valor actual del status del viaje y la fecha en la que se modificó.
- Notar que en cualquier requerimiento que implique la generación de un histórico, **se debe considerar un campo de fecha** para almacenar el instante en el tiempo en el que el campo o campos que requieren histórico se actualiza.

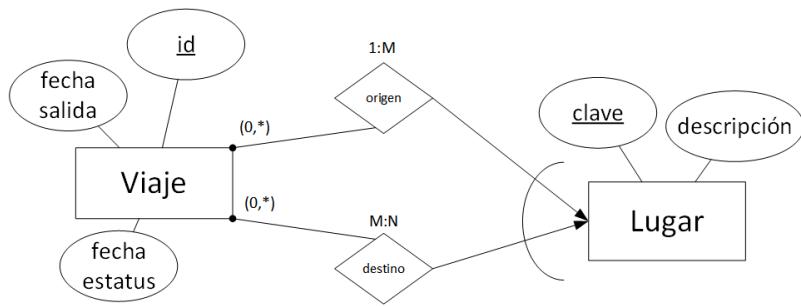
5.6. RELACIONES EXCLUSIVAS

- Suponer una entidad A que se relaciona con otras 2 entidades B y C.
- Una relación de exclusión se presenta cuando las instancias de A se deberán asociar con B o con C, pero no con ambas a la vez.
- Visto de otra forma, cada instancia de A participa en una relación con B o con C.

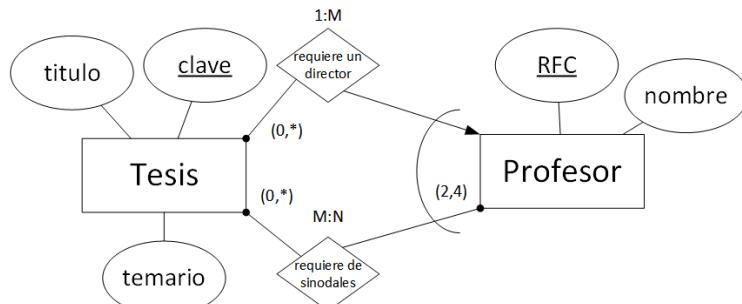
Ejemplo:

Considerando el ejemplo de la agencia de viajes anterior, la entidad Viaje se asocia 2 veces con la entidad Lugar. Un viaje tiene un destino y un origen. Cada instancia de Viaje se asocia con una instancia de Lugar para definir el origen del viaje, y con otra distinta para el destino. Lo anterior implica una relación de exclusividad ya que un viaje no puede tener el mismo origen y destino.

En el diseño conceptual la relación de exclusividad se representa el Arco (el mismo empleado en una relación supertipo -Subtipos).

**Ejemplo:**

- Un profesor puede participar como director en una Tesis. De serlo, no podrá participar como sinodal. Es decir, un profesor puede ser sinodal o director de una tesis, pero no ambos.



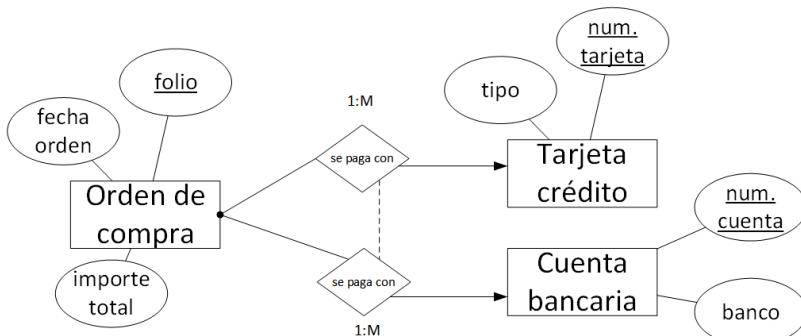
Observar que, en este tipo de relación, la exclusión se revisa del lado de la tabla padres. El arco se dibuja del lado de la tabla padres.

5.7. RELACIONES DE EXCLUSIVIDAD.

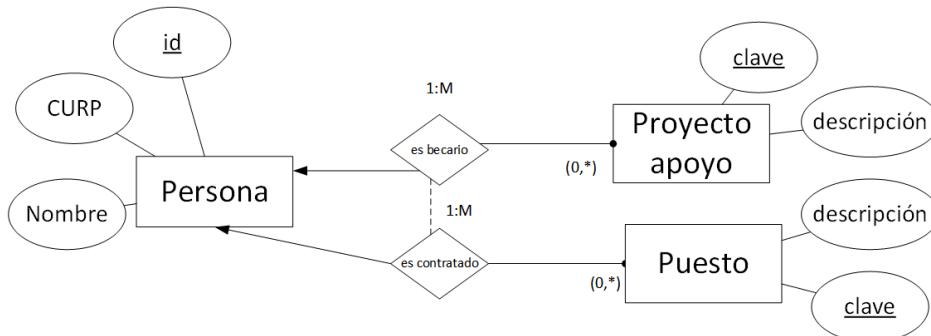
- En este caso, la relación de exclusividad se revisa del lado de la tabla hija.
- En estas situaciones, una entidad hija C se asocia con 2 entidades A y B. Cada instancia de C se asocia solo con una de las 2 instancias A o B, pero no ambas.
- En el diseño lógico, esto se refleja a través de la ocurrencia de 2 FKs, en donde para cada registro, solo una de ellas tendrá un valor asignado, por lo que ambas FKs deberán estar definidas como NULL

Ejemplo:

Una orden de compra puede ser pagada con tarjeta de crédito o a través de una cuenta bancaria, pero no ambas.

**Ejemplo:**

Cuando una persona entra a una empresa, puede ser contratada y por lo tanto se le asigna un puesto, o puede ser considerada como becario(a) por lo que, en lugar de asignarle un puesto, se le asigna a un programa social.



Observar que en este caso se traza una **línea punteada** entre las 2 relaciones para indicar la relación de **exclusividad**.

5.8. AGREGACIÓN

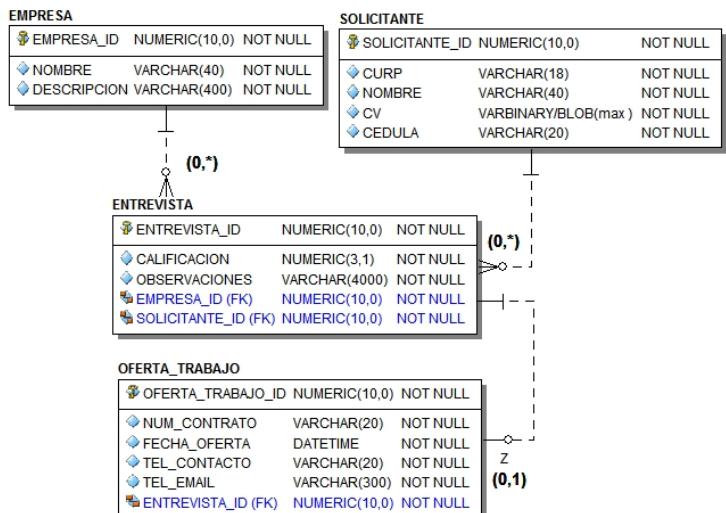
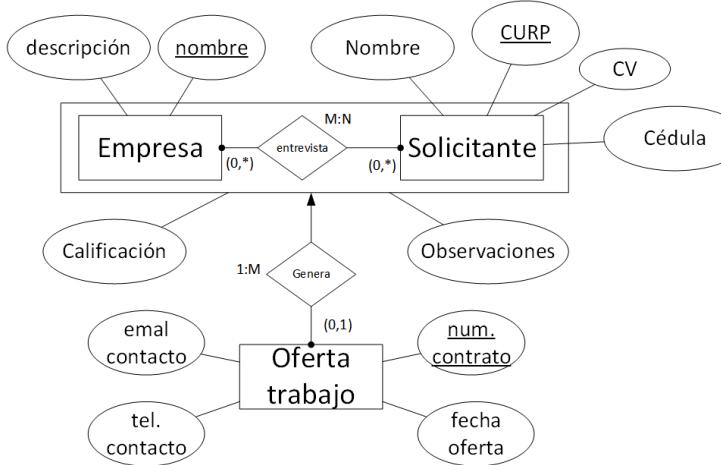
Este tipo de relación se presenta en especial con situaciones como la siguiente:

- Se tiene una relación M:N entre 2 entidades A y B.
- Se tiene una entidad C y se desea relacionarla con el resultado de relacionar a las entidades A y B.
- La condición anterior se puede ver de forma más clara en el diseño lógico: La entidad C se relaciona con la tabla intermedia que se produce al relacionar a las entidades A y B.

Ejemplo:

- En una feria de empleo las empresas realizan diversas entrevistas para reclutar a personas llamados solicitantes.
- Cada empresa registra sus datos en la feria: nombre, descripción y tipo.
- Cada solicitante registra sus datos: CURP, nombre, cedula profesional, CV. El solicitante puede participar en diversas entrevistas.
- Del resultado de las entrevistas se almacena la calificación de su examen y un texto que describe las observaciones que cada empresa encontró en el solicitante.
- Cuando el solicitante es aceptado en una empresa, se registra una nueva oferta de trabajo: fecha de la oferta, numero de contrato, email y teléfono del contacto para continuar con el proceso.
- Adicional a lo anterior, a la oferta de trabajo se le asocia el resultado de la entrevista que acredita al solicitante como apto o aprobado para cubrir dicha oferta.

Modelos:



Observar que la relación M:N se trata como una nueva entidad empleando el cuadro que las agrupa. A partir de ella se pueden crear nuevas relaciones con otras entidades.

TEMA 6

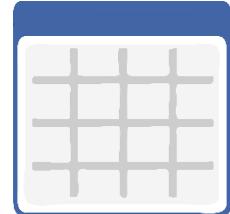
NORMALIZACIÓN

6.1. PROCESO DE NORMALIZACIÓN.

Normalización: Proceso empleado para evaluar y corregir la estructura de tablas con la finalidad de minimizar redundancia reduciendo así la posibilidad de existencia de anomalías en los datos.

El proceso de normalización se desarrolla a través de la aplicación de una serie de pasos o estados llamados formas normales:

- Primera forma normal (1FN).
- Segunda forma normal (2FN)
- Tercera forma normal (3FN)
- Forma normal de Boyce –Codd (BCNF)
- Cuarta forma normal (4FN).



La aplicación de un mayor o menor nivel de normalización puede generar las 2 siguientes situaciones:

Mayor nivel de Normalización

- Posibles problemas de desempeño y eficiencia
- Menor o prácticamente ausencia de redundancia de datos.



Menor nivel de Normalización

- Prácticamente sin problemas de desempeño y eficiencia.
- Mayor nivel de redundancia de datos.

- Típicamente, para la mayoría de los modelos, se emplea hasta la 3FN.
- Desde el punto de vista estructural, una forma normal de mayor nivel es mejor que una de menor nivel.
- El proceso de normalización es importante pero un alto nivel *no siempre es lo deseado o lo mejor*.
- A mayor nivel de normalización la información se almacena en un mayor número de tablas y por lo tanto se incrementará el número de operaciones *join* entre ellas para satisfacer una consulta.
- Antes de decidir el nivel de normalización a aplicar se deberán considerar aspectos *de rapidez y desempeño*.
- Al proceso inverso de normalización se le conoce como *denormalización*.

Para realizar el proceso de normalización, considerar la siguiente tabla de datos en la que se almacenan las faltas y las calificaciones de los alumnos de una universidad, en donde cada alumno pertenece a una carrera la cual también se especifica en la tabla de datos.

num_estudiante	nombre	ap_paterno	ap_materno	clave_asignatura	nombre_asignatura	créditos_asignatura	faltas	calificación	clave_nacimiento	Lugar_nacimiento	clave_carrera	nombre_carrera
1001	Juan	Méndez	Kim	1763	Algebra	10	1	9	COL	Colima	110	I. Civil
				3411	Calculo 2	8	0	7	COL	Colima	110	I. Civil
1002	Mario	Luna	Ubaldo	1890	Calculo 1	10	3	7	CHIH	Chihuahua	110	I. Civil
1003	Eva	Aguirre	Salas	3411	Calculo 2	8	5	8	NL	Nuevo León	111	I. Electro
1004	Lucía	Juárez	Aldama	1763	Algebra	10	0	10	MICH	Michoacán	111	I. Electro
1005	Alonso	Lugo	López	1890	Calculo 1	10	2	5	SON	Sonora	111	I. Electro
1002	Mario	Luna	Ubaldo	1763	Algebra	10	2	8	CHIH	Chihuahua	110	I. Civil
1006	Eva	Lugo	Macías	1790	Estadística	8	9	6	QRO	Querétaro	111	I. Electro

¿Qué anomalías existen en esta tabla de datos que podrían ser eliminadas al aplicar un proceso de normalización?



- Al existir varios registros por alumno, su información se repite en los campos nombre, ap_paterno, ap_materno, clave_nacimiento, lugar_nacimiento, clave_carrera y nombre_carrera.
- Los nombres de las asignaturas en el campo nombre_asignatura se repiten por alumno, lo mismo ocurre para los campos lugar_nacimiento, nombre_carrera.
- En caso de aplicar una actualización, si no se actualizan los campos con valores repetidos, genera inconsistencia (anomalías de actualización).
- ¿Qué pasaría si se registra a un alumno que no está inscrito en alguna asignatura? Todos los demás campos deberán declararse como null (anomalías de inserción).
- ¿Qué pasa si se elimina el registro del alumno 1006?, Los datos de la asignatura estadística desaparecen ya que solo este alumno hace referencia a dicha asignatura (anomalías de eliminación).

6.2. APLICACIÓN DE LA 1FN

Objetivo:

Una tabla estará en su 1FN cuando no existen **grupos de repetición**, la PK, **dependencias parciales** y **dependencias transitivas** se han identificado.



6.2.1. Grupos de repetición.

Ocurre al realizar una agrupación de 2 o más registros para una o varias columnas que tienen el mismo valor. Por ejemplo, para la tabla de datos anterior, los primeros 2 registros son agrupados en los campos nombre, ap_paterno y ap_materno. En la 1FN, estas agrupaciones se deberán eliminar repitiendo los datos en cada registro:

num_estudiante	nombre	ap_paterno	ap_materno	clave_asignatura	nombre_asignatura	créditos_asignatura	faltas	calificación	clave_nacimiento	Lugar_nacimiento	clave_carrera	nombre_carrera
1001	Juan	Méndez	Kim	1763	Algebra	10	1	9	COL	Colima	110	I. Civil
1001	Juan	Méndez	Kim	3411	Calculo 2	8	0	7	COL	Colima	110	I. Civil

6.2.2. Dependencia funcional.

Recordando este concepto visto en el tema 3:

Consiste en determinar y verificar el o los campos que actuarán o tendrán el papel de PK.

Definición: El atributo “B” es funcionalmente dependiente de un atributo “A” si cada valor de la columna “A” determina uno y solo un valor de la columna “B”. Se expresa:

$$A \rightarrow B$$

De lo anterior, se puede concluir que “A” puede tomar el rol de llave primaria, siempre y cuando A, también pueda determinar el valor de las demás columnas. Es decir:

$$A \rightarrow B, C, D, E \dots$$

Ejemplo:

Considerando los campos de la tabla anterior, iniciando en C_1, C_2 , hasta C_{13} de izquierda a derecha, determinar si las siguientes expresiones son verdaderas

- $\{C_4 \rightarrow C_3\}$? (C_3 será funcionalmente dependiente de C_4 ?), Respuesta: FALSE.
- $\{C_4 \rightarrow C_1\}$? (C_1 será funcionalmente dependiente de C_4 ?), Respuesta: FALSE
- $\{C_1 \rightarrow C_4\}$? (C_4 será funcionalmente dependiente de C_1 ?), Respuesta: TRUE
- $\{C_5 \rightarrow C_7\}$? (C_7 será funcionalmente dependiente de C_5 ?), Respuesta: TRUE
- $\{C_1 \rightarrow C_8\}$? Respuesta FALSE
- $\{C_5 \rightarrow C_8\}$? Respuesta: FALSE
- $\{C_1, C_5 \rightarrow C_8, C_9\}$? Respuesta: TRUE

6.2.3. Identificación de la PK.

Empleando el concepto de **dependencia funcional** se determina la PK. El proceso consiste en encontrar el(s) campo(s) que determinen de manera única a cada uno de los registros de la tabla:

$$A \rightarrow B$$

- Todos los atributos de la tabla representados por “B” son conocidos como **atributos dependientes** de A.
- Todos los atributos de la tabla representados por “A” son conocidos como **atributos determinantes** debido a que en su conjunto pueden identificar o determinar de manera única a cada registro de la tabla. En este sentido, todos los atributos formados por A representan a la PK de la tabla.

Para la tabla anterior, la PK estará definida por los siguientes atributos:

`num_estudiante, clave_asignatura ->`
 nombre, ap_paterno, ap_materno, nombre_asignatura,
 créditos_asignatura, faltas, calificación, clave_nacimiento,
 lugar_nacimiento, clave_carrera, nombre_carrera.



A través de la combinación de ambos atributos, es posible determinar de forma única a los demás.

6.2.4. Dependencias parciales.

- La dependencia parcial puede existir únicamente en tablas con una llave primaria compuesta.
- En una dependencia parcial, alguno de los atributos que forma parte de la PK puede por si solo actuar como **atributo determinante** de uno o más atributos de la tabla.

Ejemplo:

Para la tabla de datos, la primera condición se cumple, existe una PK compuesta. Para verificar si existen dependencias parciales, se verifica si los campos que integran a la PK de forma individual pueden determinar a otros campos:

```
num_estudiante ->
nombre, ap_paterno, ap_materno, clave_nacimiento, lugar_nacimiento,
clave_carrera, nombre_carrera

clave_asignatura -> nombre_asignatura, créditos_asignatura
```

- Observar que existen 2 dependencias parciales. A partir del número de estudiante se pueden determinar los campos que aparecen del lado derecho, y lo mismo sucede con la clave de la asignatura.
- Observar que podría ocurrir lo mismo con la clave de la carrera, sin embargo, esta no es una dependencia parcial, ya que el atributo determinante debe formar parte de la PK.

6.2.5. Dependencias transitivas.

- Una dependencia transitiva ocurre cuando se detecta una dependencia funcional en donde el atributo determinante no forma parte de la PK (a diferencia de las dependencias parciales).
- En este caso, no importa si la tabla tiene PK simple o compuesta.

Ejemplo:

Para la tabla de datos mostrada anteriormente, se tienen las siguientes relaciones transitivas:

clave_nacimiento -> lugar_nacimiento

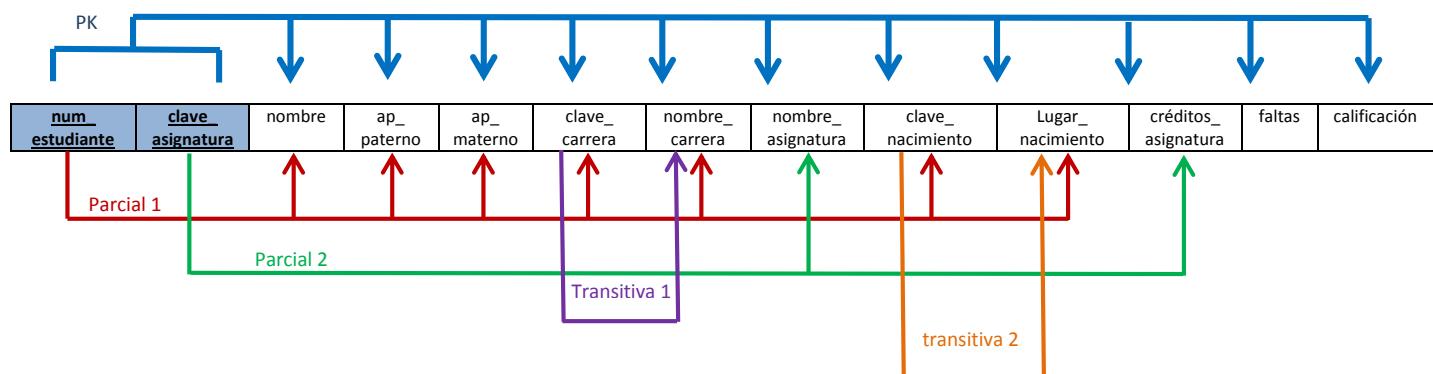
clave_carrera -> nombre_carrera

Recordando, para que una tabla este en 1FN:

- Paso 1: Se identifica la PK
- Paso 2: Se eliminan grupos de repetición
- Paso 3: Se identifican dependencias parciales.
- Paso 4: Se identifican dependencias transitivas.

Este resultado se describe en los llamados diagramas **de dependencias**:

6.2.6. Diagrama de dependencias para la 1F



6.3. APLICACIÓN DE LA 2FN



Objetivo:

Una tabla está en su 2FN cuando:

- La tabla está en su 1FN
- Se han eliminado las dependencias parciales.



tip Si la tabla no tiene una PK compuesta, en automático esta se encuentra en su 2FN.

6.3.1. Paso 1: Eliminación de dependencias parciales.

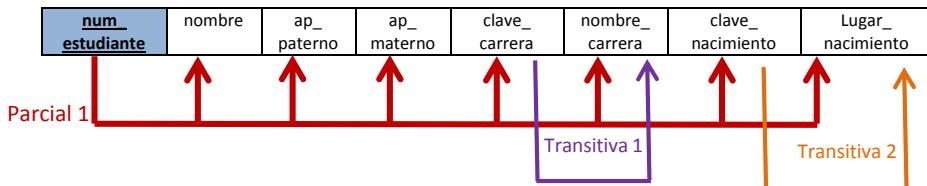
- Por cada dependencia parcial identificada se crea una tabla nueva asignando como PK al atributo determinante y con sus atributos dependientes como atributos de la tabla.
- Se le asigna un nombre a la tabla nueva.
- De la tabla original en su 1FN se eliminan únicamente los atributos **dependientes** que integran a las nuevas tablas.

Ejemplo:

Para la tabla de datos, se tienen 2 nuevas tablas:

- Nombre de la tabla: estudiante.

num_estudiante \rightarrow
nombre, ap_paterno, ap_materno, clave_nacimiento, lugar_nacimiento,
clave_carrera, nombre_carrera



Observar que las relaciones transitivas permanecen aún.

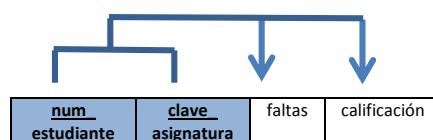
- Nombre de la tabla: asignatura.

clave_asignatura \rightarrow nombre_asignatura, créditos_asignatura



- Tabla Original: estudiante_asignatura

num_estudiante, clave_asignatura \rightarrow faltas, calificación



- Observar que ahora la tabla original solo contiene los campos que dependen de la PK compuesta.
- Observar que las PKs de las tablas nuevas, son ahora PKs y FKs de la tabla original.
- Hasta el momento se tienen 3 tablas:
 - estudiante, asignatura, estudiante_asignatura

6.4. APLICACIÓN DE LA 3FN.

Objetivo.

Una tabla está en su 3FN cuando:

- La tabla está en su 2FN
- Se han eliminado las dependencias transitivas.



6.4.1. Eliminación de dependencias transitivas.

- El procedimiento es similar al de la 2FN. Para cada dependencia transitiva se crea una tabla nueva.
- De la tabla original en su 2FN se eliminan únicamente los atributos dependientes que integran a las nuevas tablas.

Ejemplo:

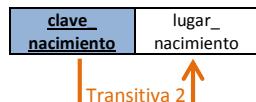
- La tabla asignatura se queda en su 2FN ya que no cuenta con dependencias transitivas.
- Para la tabla estudiante, que es la que contiene las 2 dependencias transitivas se tiene:
- Nombre de la tabla: carrera.

clave_carrera → nombre_carrera



- Nombre de la tabla: lugar_nacimiento.

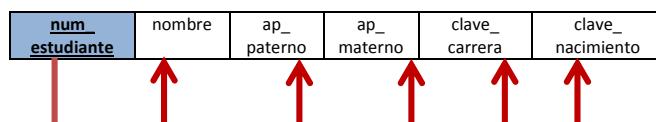
clave_nacimiento → lugar_nacimiento



- Tabla original: estudiante

num_estudiante →

nombre, ap_paterno, ap_materno, clave_nacimiento, clave_carrera

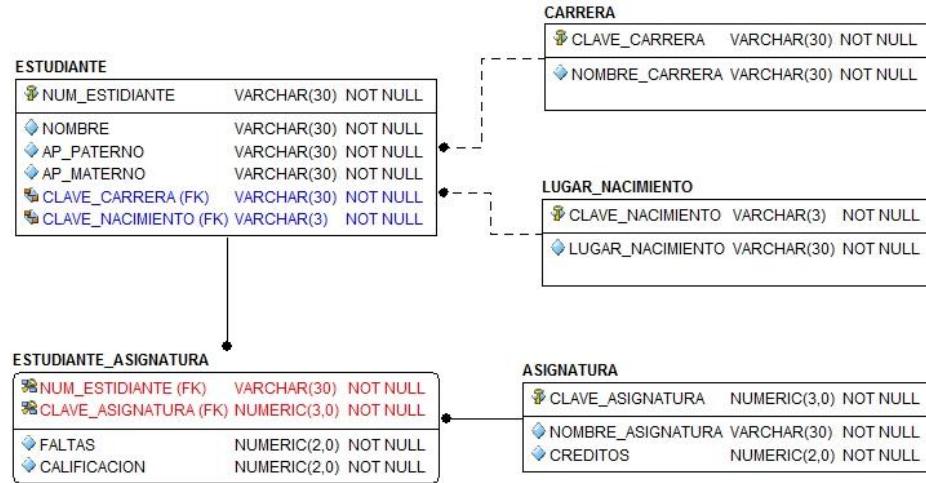


- Observar nuevamente, los campos clave_carrera y clave_nacimiento, ahora son FKs de las tablas carrera y lugar_nacimiento respectivamente.
- En tercera forma normal se tienen las siguientes tablas:

- o estudiante, asignatura, estudiante_asignatura, carrera, lugar_nacimiento.

6.4.2. Construcción del modelo relacional.

Con base a los diagramas de dependencias que se obtuvieron durante el proceso anterior, es posible generar un modelo relacional:



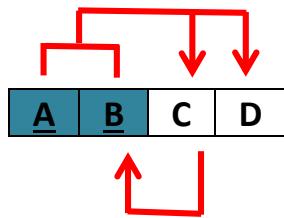
- Observar que el diagrama se genera con las tablas y atributos que se determinaron en el proceso de normalización.
- Un siguiente paso consiste en realizar el refinamiento del diseño, en el que se pueden aplicar algunas mejoras. Para el ejemplo se pueden realizar los siguientes campos:
 - o Agregar una PK artificial numérica como PK para mejorar desempeño y/o evitar el problema de posibilidad de cambios en las PKs. Hay que recordar que las PKs que se determinan en este proceso son llaves primarias naturales (campos que forman parte de las reglas de negocio).
 - o Agregar una PK artificial a la tabla intermedia (similar a la técnica que se revisó en el capítulo 4).
- Observar que la tabla original de datos fue fragmentada en 5 tablas, y al final del proceso, esta tabla resultó ser una tabla intermedia que implementa una relación M:N entre estudiante y asignatura.

6.5. APLICACIÓN DE FORMAS NORMALES DE ORDEN SUPERIOR.

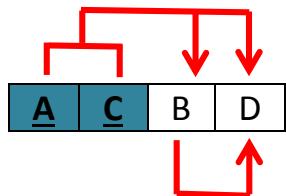
6.5.1. Forma normal de Boyce – Codd

Esta forma normal es una variante de la 3FN, y fue desarrollada en 1974 por Raymond F. Boyce y Edgar F. Codd en la cual se resuelven algunas anomalías que no resuelve la 3FN.

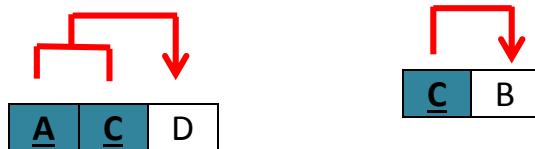
- Ocurre en tablas con PK compuesta en donde existen atributos que son determinantes y que por alguna razón no fueron seleccionados para formar parte de la PK. A nivel general se tiene la siguiente condición:



- Observar que el campo C determina al campo B, pero el campo C no es parte de la PK.
- Dado que $C \rightarrow B$, resulta completamente válido modificar la PK de la tabla intercambiando a B con C:



- La PK sigue cumpliendo con su función ya que C es un atributo determinante.
- Observar que al hacer el intercambio de estos s atributos, se forma una dependencia parcial: $C \rightarrow B$
- Recordando el proceso para normalizar una tabla en su 2FN, la tabla en BCNF ocurre al eliminar la dependencia parcial que se acaba de formar. Por lo tanto, la tabla en su BCNF se divide en 2:



Las 2 tablas anteriores ahora es tan en su forma normal Boyce – Codd (BCNF)

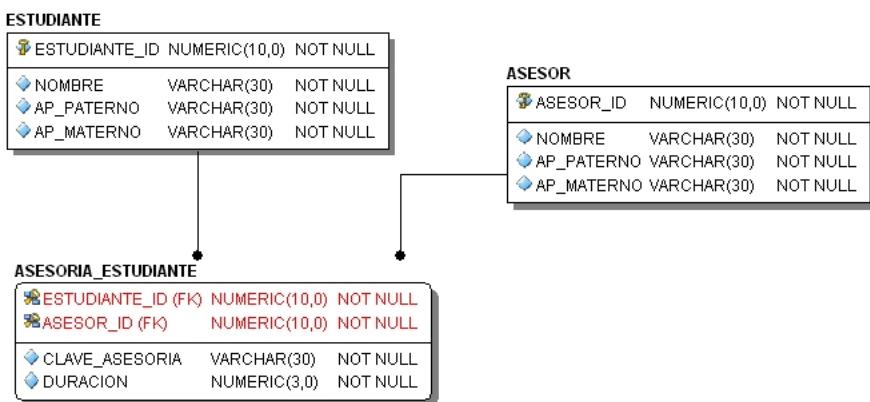
Para comprender el concepto y las anomalías que se generan al no realizar este proceso se considerar el siguiente escenario:

Ejemplo:

Registro de las asesorías y de los profesores asignados a un alumno.

- Un estudiante puede tener a varios asesores asignados durante un semestre y un asesor puede tener asignados a varios alumnos.
- Un asesor puede dar varias asesorías. Cada asesoría se identifica por una clave. Una asesoría la ofrece un solo asesor.
- El estudiante puede solicitar asesoría por parte de uno de sus asesores y al acudir se guarda el tiempo que dura dicha asesoría.

El modelo relacional que se ha generado para modelar estas reglas de negocio es el siguiente:



La siguiente tabla muestra el comportamiento de los datos para la tabla `asesoria_estudiante`

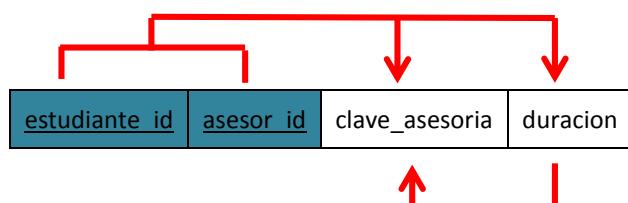
estudiante_id	asesor_id	clave_asesoria	duración (min)
1	1	ALG-001	15
2	1	ALG-001	22
3	2	BD-022	19
4	3	MATH-089	34
4	4	CALC-020	32
5	4	CALC-020	27
6	5	JAVA-093	21

¿Qué anomalías presenta este diseño?

- La tabla `asesoria_estudiante` intenta describir 2 reglas:
 - Las asesorías a las que acudió el estudiante.
 - Las asesorías asignadas a un asesor.
- Lo anterior tiene como resultado anomalías de actualización y de eliminación:
 - Anomalía de actualización: Si al asesor con Id = 1 se le cambia su asesoría a ALG-020, se deberán actualizar 2 registros en la tabla, la asignación de la asesoría para el asesor 1 se duplica.
 - Anomalía de eliminación: Si se elimina el registro del estudiante con Id =6, ¿cómo sabríamos la asesoría asignada al profesor con Id = 5?

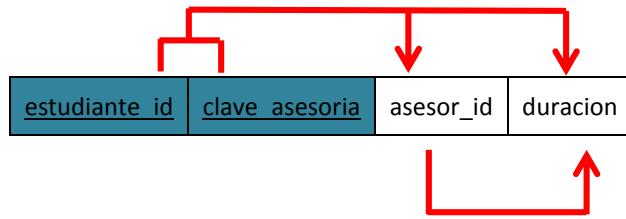
Observar que el atributo `clave_asesoria` es un atributo determinante de `asesor_id`, es decir, a partir de la clave de la asesoría es posible determinar al asesor que la ofrece.

`clave_asesoria` → `asesor_id`

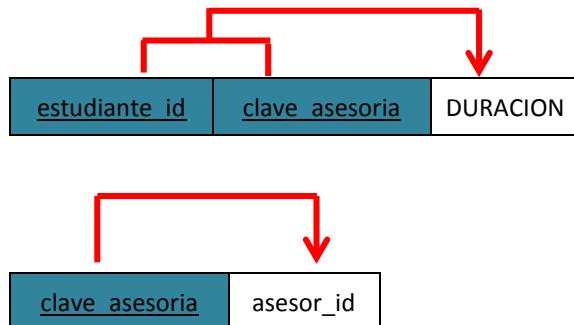


Si se realizara el intercambio para generar una dependencia parcial, la tabla quedaría de esta forma:

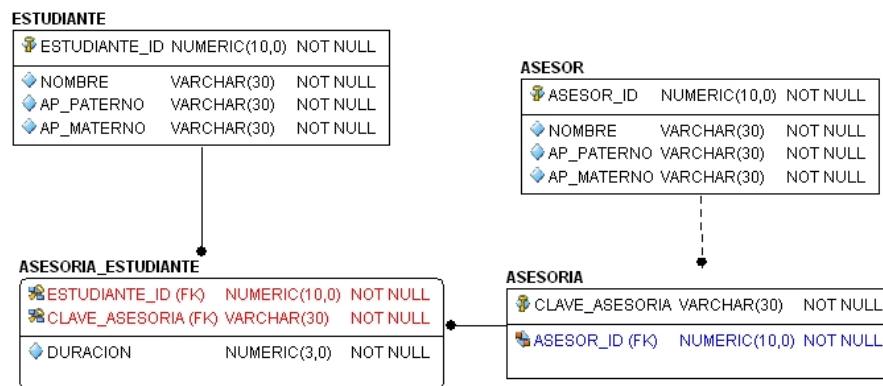
`estudiante_id,clave_asesoria` → `asesor_id,duracion`



Observar que la tabla sigue siendo equivalente a la original, pero ahora con una dependencia parcial, la cual al eliminarla se tiene a la tabla en su BCNF:



El modelo relacional resultante será:



Con este nivel de normalización la tabla `asesoria_estudiante` ahora solo describe las asesorías a las que acude un estudiante. La definición de las asesorías que ofrece cada asesor se encuentra en la tabla nueva `asesoria`.

6.5.2. Aplicación de la cuarta forma normal (4FN)

Una tabla estará en su cuarta forma normal (4FN) cuando:

- Está en su tercera forma normal 3FN
- Se han eliminado posibles **dependencias multivalor**.



6.5.2.1. Dependencias multivalor

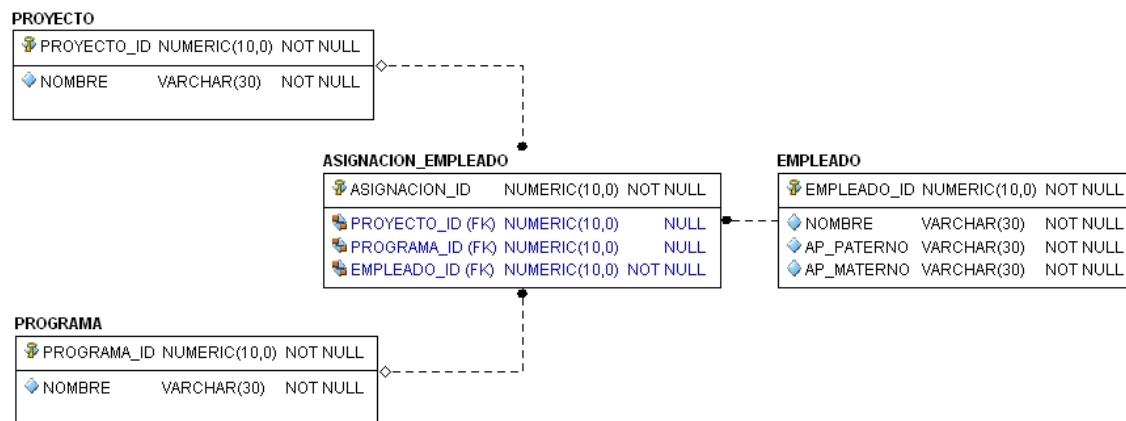
Ocurre cuando la PK de una tabla determina múltiples valores de 2 o más campos de la tabla y cada uno de estos campos son totalmente independientes entre sí, es decir, no hay relación alguna entre ellos.

Ejemplo:

Programas y proyectos en los que puede estar inscrito un empleado:

- Un empleado puede estar asignado a uno o a mas proyectos
- Un proyecto está formado por varios empleados.
- Un empleado puede participar en uno o más programas voluntarios de apoyo.
- Un programa de apoyo está integrado por varios empleados.

El modelo relacional que se ha generado para modelar estas reglas de negocio es el siguiente:



La siguiente tabla muestra el comportamiento de los datos para la tabla `asignacion_empleado`.

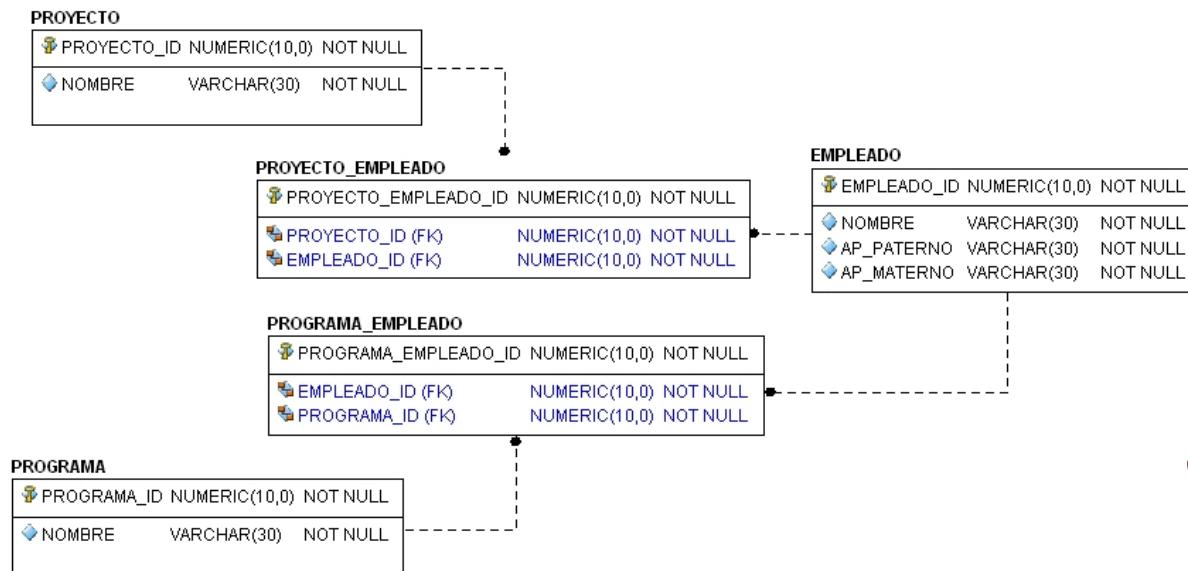
asignacion_id	proyecto_id	programa_id	empleado_id
1	4	NULL	10
2	5	NULL	10
3	NULL	2	10
4	NULL	3	10

¿Qué anomalías presenta este diseño?

- Observar la dependencia multivalor: `proyecto_id` y `programa_id` son 2 campos totalmente independientes, y sin embargo están asociados a un mismo empleado.
- Para un mismo empleado se pueden generar múltiples valores de `proyecto_id` y `programa_id`. En otras palabras, estas anomalías se pueden generar cuando se emplea una misma tabla intermedia para representar más de una relación M:N, en este caso son 2 relaciones M:N `empleado - proyecto`, y `empleado - programa`.
- Estas 2 condiciones provocan que ambos campos se tengan que definir como NULL como se muestra en la tabla de datos.

Para eliminar la dependencia multivalor, se crea una tabla nueva por cada campo que genera múltiples valores, es decir, una tabla por cada relación M:N

El diagrama en su 4FN es el siguiente:



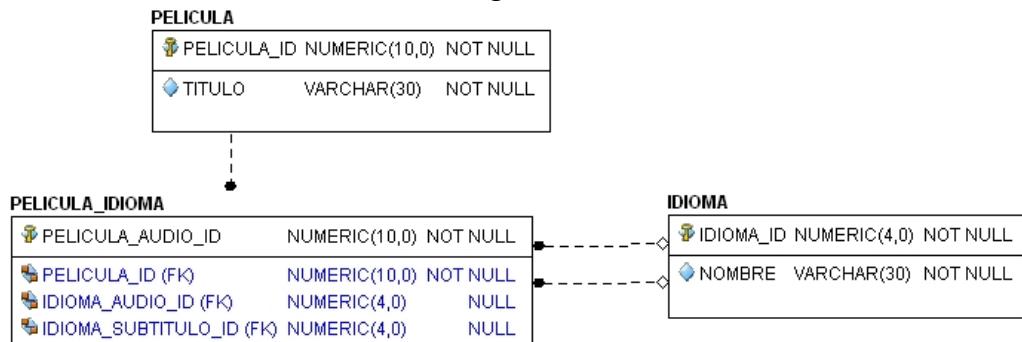
Ejemplo:

Registro de películas:

- Una película tiene varios idiomas de audio disponibles. Un idioma puede estar asociado a varias películas.
- Una película tiene varios idiomas de subtítulos disponibles. Un idioma puede estar asociado a varias películas.

Como se puede observar, se trata de 2 relaciones M:N. Los subtítulos y los audios en sentido estricto son independientes, es decir, se pueden seleccionar de forma independiente, por lo tanto, es posible generar 2 tablas intermedias, una para cada relación M:N (4FN).

Sin este nivel de normalización, se tendrían los siguientes inconvenientes:



asignacion_idioma_id	idioma_audio_id	idioma_subtitulo_id	pelicula_id
	NULL	1	1
	NULL	2	1
	3	NULL	1
	4	NULL	1

- Observar que se deben definir los campos como NULL, o en su defecto, se tendrían que agregar todas las posibles combinaciones entre idiomas de audio y de subtítulos. Lo anterior obliga a relacionar los idiomas de los audios con los idiomas de los subtítulos cuando en sentido estricto son independientes.
- Aplicando la 4FN, el modelo relacional es el siguiente:

