Sección 01:

Que veremos en esta sección:

* Conceptos básicos en base de datos
* Tipos de bases de datos
* Que es un gerenciador de bases de datos (DBMS) y un administrador de base de datos (DBA)
* Ventajas y desventajas de usar bases de datos

Conceptos básicos en base de datos:

Datos: Son símbolos que se caracterizan por carecer de valor, por ejemplo 100, Calle Araoz

No tienen capacidad de comunicar un significado

Información: Conjunto de datos pertinentes y significativos, por Ejemplo: Yo vivo en la calle Araoz al 100

Para ser útiles los datos deben convertirse en información.

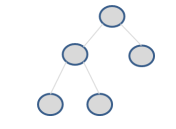
Las bases de datos almacenan datos que aislados no me dicen nada

Tipos de bases de datos:

Vamos a ver las bases de datos relacionales, existen otros tipos de bases de datos como:

* Las jerárquicas: En este modelo los datos se organizan en una forma similar a un árbol (visto al revés), en donde un nodo padre de datos puede tener varios hijos. El nodo que no tiene padres

es llamado raíz, y a los nodos que no tienen hijos se los conoce como hojas.

Raíz

Nodo

Hoja

* Objetos: Este modelo, bastante reciente, y propio de los modelos informáticos orientados a objetos, trata de almacenar en la base de datos los objetos completos (estado y comportamiento).
* Relacional: Es el que vamos a estudiar, Éste es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Los datos pueden ser recuperados o almacenados mediante «consultas» que ofrecen una amplia flexibilidad y poder para administrar los datos. Se llama Relacional porque los datos están relacionados de alguna forma

Sistemas de gestión de base de datos (DBMS)

Son programas informáticos que permiten crear, almacenar y posteriormente acceder a datos de forma rápida y estructurada. Los DBMS constan de ciertos componentes tales como:

* Data definition language (DDL)
* Data manipulation language (DML)
* Structured query language (SQL)

Ejemplo: Microsoft SQL, MySQL, Oracle

No vamos a implementar una base de datos, vamos a diseñar una base de datos

Administrador de Base de Datos (DBA)

Es la persona o equipo de personas profesionales responsables del control y manejo del sistema de base de datos, generalmente tiene(n) experiencia en DBMS, diseño de bases de datos, Sistemas operativos, comunicación de datos, hardware y programación.

El DBA normalmente puede estar involucrado desde el mismo inicio del desarrollo de la

base de datos definiendo la estructura interna, compra de equipos, desarrollo,

seguridad, permisologia a los usuarios finales, etc

Ventajas de usar una base de datos:

1. Centraliza de una manera segura los datos de una organización.

2. Automatización de procesos.

3. Acceso a la información de una manera mucho más rápida.

4. Aumenta la calidad de respuesta de la organización.

5. Crea nuevos puestos de trabajo.

Desventajas de usar una base de datos:

1. Documentación de los procesos de la organización (si no los posee).

2. Costos adicionales por hardware y software especializado.

3. Nuevo puestos de trabajo (incremento de los gastos de la organización).

4. Cambio de paradigma en la forma de trabajo.

5. Si son pocos datos, es mejor utilizar otras opciones tales como una hoja de cálculo.

Sección 02 – Herramientas online recomendadas.

Hay muchas herramientas online y de escritorio que pueden perfectamente ayudar en el proceso de diseño de base de datos relacionales. Sin embargo, aquí te coloco algunas para que puedas de una vez empezar a probar a ver si alguna de ellas cumple con tus requerimientos. En el curso utilizaremos LUCIDCHART.

Articulo completo:

<https://www.holistics.io/blog/top-5-free-database-diagram-design-tools/>

DBDiagram.io:

<https://dbdiagram.io/home>

Draw.io:

<https://www.draw.io/>

Lucidchart:

<https://www.lucidchart.com/pages/home>

QuickDBD:

<https://www.quickdatabasediagrams.com/>

Vamos a crear dos tipos de diagramas

* El modelo Entidad – Relacion
* El modelo Relacional

Sección 03: Modelos de Datos y Niveles de Diseño

Que vemos en esta sección:

* ¿Qué es un modelo de datos?
* Dominio de datos
* Los tres niveles fundamentales en el diseño de una BD
* Pasos para diseñar una BD

Que es un modelo de Datos:

Un modelo de datos es un conjunto de conceptos utilizados para organizar los datos de interés y describir su estructura en forma comprensible para un sistema informático

Si los modelos no son definidos apropiadamente podemos tener muchos problemas el momento de ejecutar consultas a la base de datos para tratar de obtener alguna información.

Uno de los pasos cruciales en la construcción de una aplicación que maneje la base de datos es el diseño de la base de datos, en donde lo mas importante son los modelos de datos.

Entre los objetivos en el diseño de la base de datos está en tener un acceso eficiente a la información en cuanto a la velocidad y facilidad de acceso.

Dominios de bases de datos:

En una función son todos los valores para los cuales está definida la función.

En bases de datos son los valores que pueden tomar los datos o sea que también hay restricciones.

Por ejemplo, los meses son 12, no hay un mes 13 si los numero del 1 al 12.

Una persona no puede tener una edad negativa.

Esas son las restricciones que se aplican en cada caso.

Niveles de diseño:

¿Qué es un diseño de BD?

No es otra cosa que definir la estructura que le daremos a los datos que se van a manipular en el sistema informático que se desea desarrollar.

Es un proceso que se puede dominar de forma intuitiva con facilidad basado en la experiencia.

Es importante conocer el negocio para empezar a modelar adecuadamente.

Los pasos serían:

1. Entender el negocio.
2. Entrevistas con los dueños del negocio, se debe pedir la aprobación de los mismos para evitar que haya quedado en el camino algo sin contemplar.
3. Generar una lista de requerimientos, conviene que esté aprobada por los dueños del negocio.
4. Documentación técnica. Hay modelos pre determinados.
5. Posibles prototipos (opcional)

Niveles básicos de diseño:

Se compone de los siguientes niveles

1. Nivel Conceptual
2. Nivel Lógico
3. Nivel Físico
4. Nivel Conceptual: -> Salida es el Esquema Conceptual

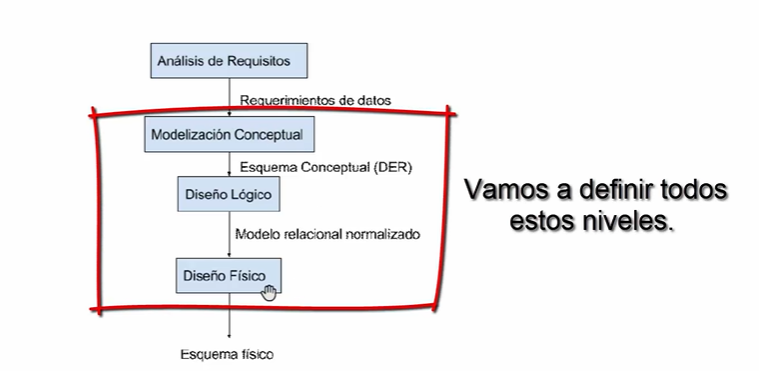
Se definen los datos específicos que se van a manejar en la base de datos. Estos datos se agrupan en componentes y se define la relación entre los componentes y su cardinalidad (La cantidad de relaciones entre un componente A y otro B)

En esta fase no importa qué tipo de DBMS vamos a utilizar

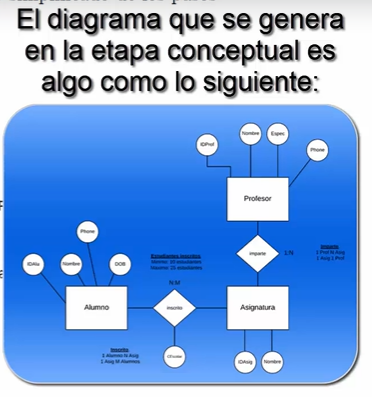
El objetivo del diseño conceptual es describir los datos de la base de datos y NO las estructuras de almacenamiento que se necesitarán para manejar estos datos.

Esta etapa ayuda al analista y desarrollador de bases de datos a entender el problema, los datos, restricciones, etc. El resultado es el Esquema Conceptual

Un flujo simplificado de los pasos que se siguen en el diseño tradicional son:

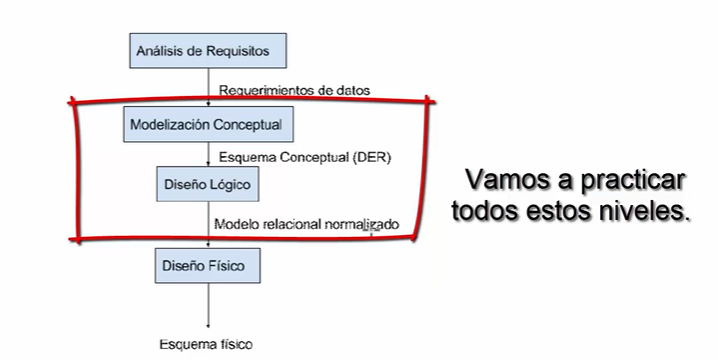
****

En el curso no se hace al análisis de requisitos, ya vamos a tener los requerimientos para implementar la base de datos. Partimos de un análisis de requisitos ya hecho.



A partir de este diagrama vamos a:

1. Creamos el Modelo Conceptual
2. El Esquema Conceptual
3. El Diseño Lógico
4. El Modelo Relacional Normalizado



1. Nivel lógico: -> Salida es el Esquema Lógico

En esta fase ya están definidos todos los componentes debido al diseño anterior (Nivel Conceptual).

En esta etapa estructuramos cada componente principal como una tabla que está relacionada con otras tablas del modelo de datos.

En esta fase ya puede haber una relación directa con algún modelo de DBMS como Oracle o SQL

En el Nivel Conceptual entendemos el problema, entendemos que datos vamos a utilizar. Al final generamos el Esquema Conceptual. Cada componente de ese esquema lo transformamos en tablas y cada tabla va a estar relacionada entre sí.

Un modelo lógico, por tanto, es un lenguaje usado para especificar esquemas lógicos (Modelo relacional, Modelo de objetos,). El diseño lógico depende del tipo de DBMS (Oracle, SQL) que se vaya a utilizar, no depende del producto concreto.

La salida del nivel lógico es el Esquema Lógico.

1. Nivel Físico

El diseño físico parte del Esquema Lógico y da como resultado un Esquema Físico.

Un Esquema Físico es una descripción de una implementación de una base de datos:

* Las estructuras de almacenamiento
* Métodos utilizados para tener un acceso eficiente a los datos

Depende del DBMS utilizado y se expresa a través de su lenguaje de definición de datos.

En esta etapa debe haber una relación con el DBMS que vayamos a utilizar. Para este nivel ya tienen que haber una decisión sobre el DBMS que vayamos a utilizar.

QUIZ:

Question 1:

Definir un modelo de datos es importante ya que ayuda a los empleados de una organización a entender la importancia de los datos dentro del negocio.

Falso

Correcto!! El modelo de datos es fundamental para que un sistema informático pueda organizar los datos y pueda, más adelante, obtener información relevante. Los empleados no necesariamente deben entender todos los datos que están involucrados en una organización para que puedan realizar su trabajo de manera eficiente.

Sección 4: Anomalías en el diseño de una Base de datos Relacional

En esta sección veremos:

* Cuáles son los problemas más comunes
* Insertar datos
* Modificar datos
* Borrar datos

Anomalías:

Se denominan así a los problemas que aparecen con frecuencia en el manejo de las mismas cuando el diseño no ha sido realizado en forma “Normalizada” Vamos a definir normalización en la sección 7.

Se denominan anomalías básicas a:

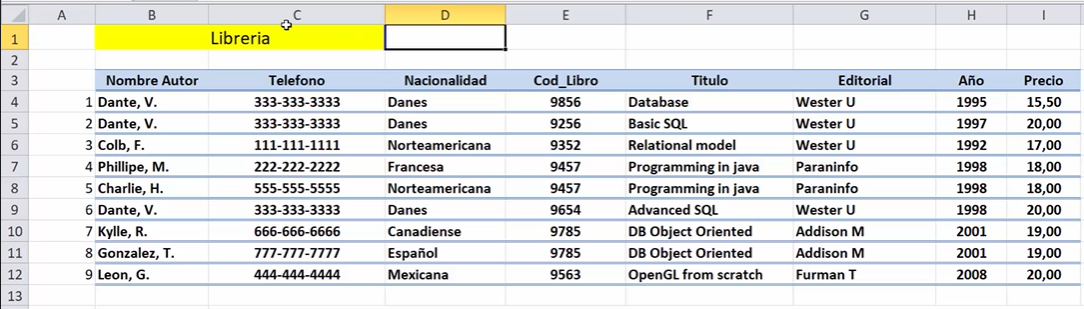
* Inserción
* Modificación
* Borrado

Adicionalmente hay que tener cuidado con la redundancia de datos.

* Redundancia: Un mismo dato aparece varias veces, no es siempre malo, pero si aparece muchas veces hay que ver si no se trata de un error. Puede causar que la búsqueda de información sea muy lenta.
* Anomalía de Inserción: Imposibilidad de adicionar un dato a la base por ausencia de otro dato en la misma.
* Anomalía de Actualización: Inconsistencias en los datos como resultado de datos redundantes y actualizaciones parciales.
* Anomalía de borrado: Perdidas no intencionales de datos debido a que se han borrado otros datos. Borramos un dato A pero como este está de alguna manera unido a un dato B borramos de forma no intencional este último.

La técnica de Normalización reduce las anomalías. Algunas se pueden eliminar por programación, pero la idea es que el DBMS se encargue de forma automática de estas restricciones al crear tablas y sus relaciones.

Vamos a trabajar con la siguiente tabla como ejemplo:



Problemas de diseño: Redundancia de datos

En el caso del autor Dante su nombre, TE y nacionalidad se repite por cada libro publicado.

En el caso de DB Object Oriented el mismo está escrito por dos autores, Kylle y Gonzalez, y se repite la editorial.

Los datos de la editorial se repiten si los libros tienen más de un autor.

Problemas de diseño: Inserción

En esta tabla la clave primaria es el Cod\_Libro. La clave primaria es como el DNI de una persona, sin ese dato no podemos ingresar una fila nueva. Por ejemplo, si quiero ingresar a un autor nuevo que no haya escrito aún un libro no puedo hacerlo ya que la clave primaria está asociada a un libro.

La clave primaria va a diferenciar cada fila de cada tabla de la base de datos. En esta tabla vemos que se repiten las claves primarias (Fila 7 y 8) por lo que podemos decir que la tabla está mal diseñada.

Problemas de diseño: Modificación

En esta tabla si yo modifico el nombre de la editorial en la primera fila no se modifica el mismo en las demás filas. Si modifico en la primera fila Western U por Western A la modificación no se va a ver reflejada en las demás filas.

Lo mismo va a pasar si modifico Dante V por Dante A.

Esto da lugar a incoherencias en los datos.

Problema de diseño: Borrado

Si quiero eliminar un libro de esta base, por ejemplo, Relational Model, lo elimino también a Colb que es su autor. Si elimino OpenGL from Scratch también elimino la editorial que es Furman T.

Principio Básico:

Hechos distintos deben almacenarse en objetos distintos.

En esta base de datos tengo 3 hechos distintos según veo en este gráfico:



* Autores
* Libros
* Editoriales

Debería separar esta tabla gigante en 3 tablas y relacionarlas entre sí.

En la sección 5 vamos a aprender a modelar los datos en base a los requerimientos.

Sección 5: Modelo Entidad-Relación

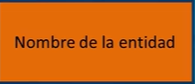
En esta sección vamos a ver:

* Que es una entidad, una relación o un atributo
* Estudiaremos la cardinalidad entre dos atributos
* Desarrollaremos nuestro diagrama de E-R
* Hacemos las primeras prácticas.

Que es una Entidad:

Es un objeto que existe, de forma tangible o intangible, y que se distingue de otros por sus características particulares. Ej., Un alumno, una transacción bancaria, una materia en curso.

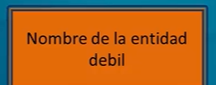
Se simboliza con un rectángulo con su nombre adentro.



Entidad Debil:

Cuando una entidad participa de una relación puede ser una entidad fuerte o débil. Una entidad débil es la que no puede existir sin participar de una relación. No puede ser identificada solo por sus atributos.

El símbolo que se utiliza para representar una entidad débil es un rectángulo con doble borde



Ej:



Para que exista el pago debe haber un préstamo.

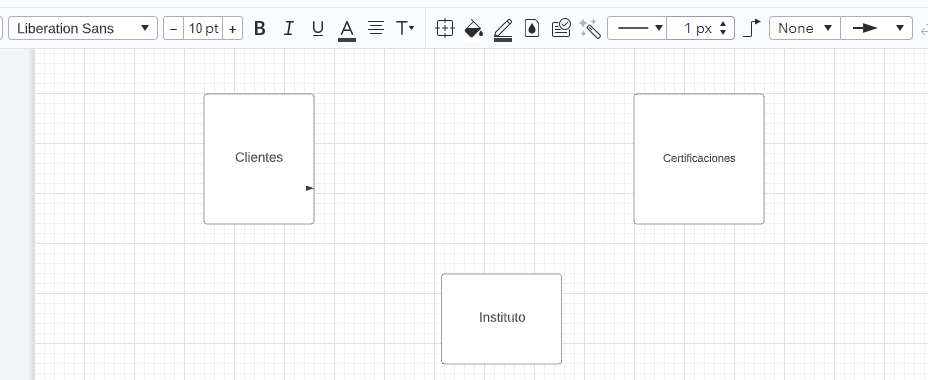
Supongamos el siguiente enunciado:

Un profesional puede obtener diferentes certificaciones en nuestro instituto

Las entidades serían:

* Profesionales (Clientes)
* Certificaciones (Oracle, Microsoft)
* Instituto

En Lucidchart lo podemos dibujar así: Siempre se puede modificar.



Que es un atributo:

Las características de las entidades se conocen como atributos. Dos objetos de una misma entidad se van a diferenciar gracias a los atributos.

Ej: Dos alumnos de un curso

El símbolo que se usa para representar un atributo es un círculo o un óvalo.



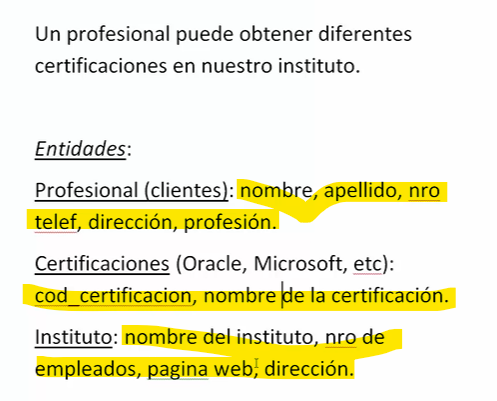
Cada atributo tiene un conjunto de valores que se conoce como Dominio o Dominios. Define todos los valores que puede tomar el atributo.

Si un atributo es multivalorado su símbolo es igual que antes, pero se usa una línea doble

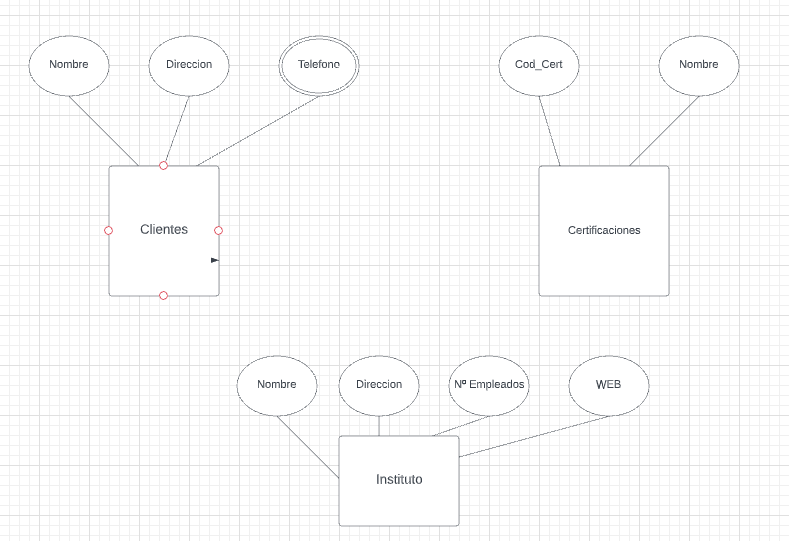


Por ejemplo, una entidad alumno puede tiene un atributo teléfono que es multivalorado ya que puede tener más de un número telefónico.

En el ejemplo anterior del instituto tenemos las entidades (subrayadas) y los atributos resaltados en amarillo:



En Lucidchart se vería así:



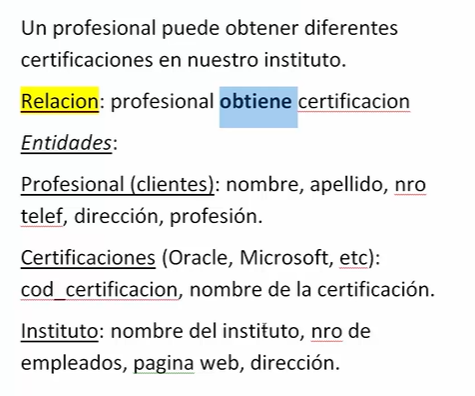
Que es una Relación:

Es la asociación que existe entre una o más entidades. Un alumno puede estar relacionado con una materia a través de la relación <está inscripto>.

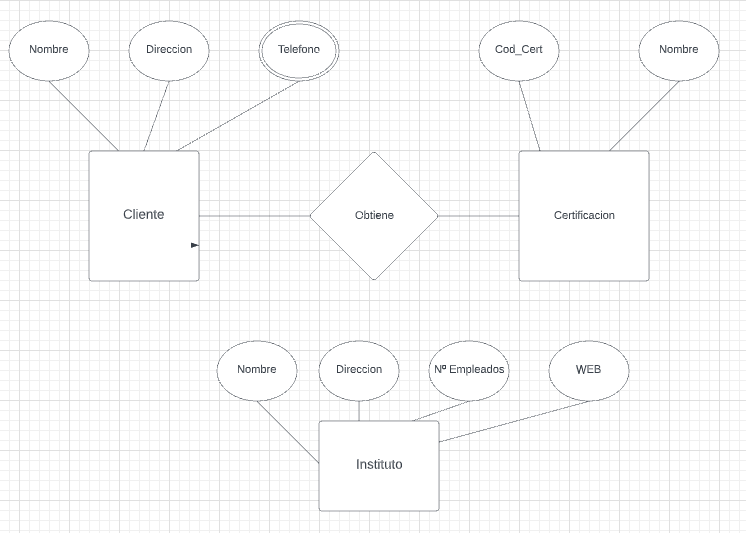
El símbolo que se usa para representar una relación es un rombo



Se puede decir que la relación es un verbo que vincula dos entidades.



En Lucidchart se vería sí



Cardinalidad:

La cardinalidad se puede definir como el número de entidades que está relacionada con otra entidad.

Ej: Alumno y Curso, un alumno puede estar registrado en N cursos.

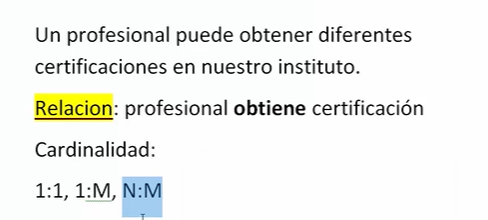
Existen varios tipos de relaciones que pueden establecerse entre las entidades.

* Cardinalidad 1 a 1. Se denota como 1:1 y significa que la entidad A solo puede relacionarse con un elemento de la entidad B. Ej: Una pareja de esposos.
* Cardinalidad 1 a M. Se denota como 1:M y significa que la entidad A se puede relacionar con M elementos de la entidad B. Ej: Una factura puede ser emitida a una sola persona, pero una persona puede tener varias facturas
* Cardinalidad N a M. Se denota como N:M y significa que la entidad A se puede relacionar con M elementos de la entidad B y la entidad B se puede relacionar con N elementos de la entidad A.

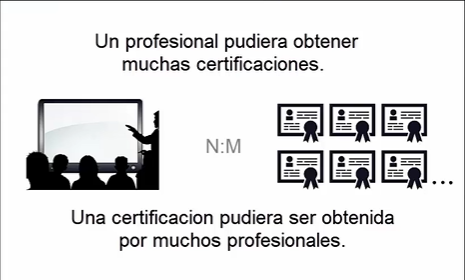
Ej: Un cliente puede adquirir varios servicios (M) de una compañía y ese servicio puede ser adquiridos por varios clientes (N)

* Cardinalidad o a 1 o 0 a M. Se denota como 0:1 o 0:M. La entidad B no tiene relación con la entidad A. Ej: Todo policía tiene registrado un arma, pero si esta de franco no lo tiene en su poder.

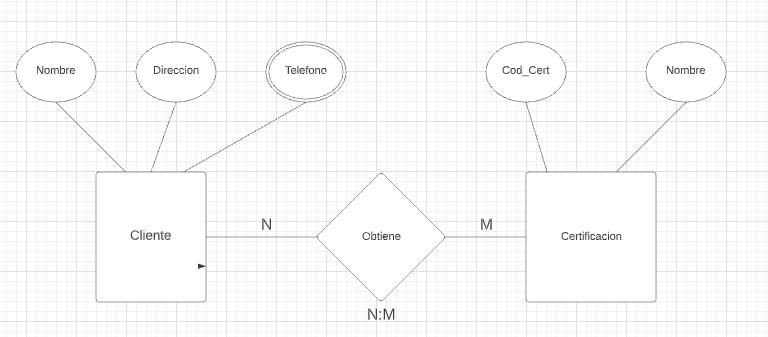
En nuestro ejemplo del instituto sería:



Porque N profesionales pueden obtener M certificaciones



En el diagrama de Lucidchart sería:



Puede ser debajo del símbolo de Relación o a los costados donde muestra con que entidades está relacionado

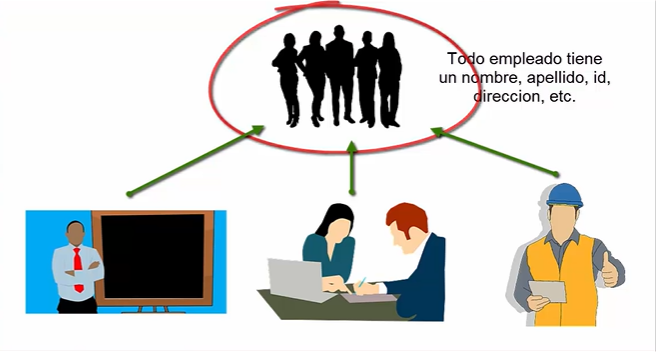
Generalización:

Una entidad E es una generalización de un grupo de Sub Entidades E1, E2, ….E(n) si cada ocurrencia de una entidad E(n) es también una ocurrencia de la entidad E.

Se representa generalmente así:



Todo empleado del instituto tiene un ID, Teléfono, fecha de nacimiento



Pero hay empleados que son docentes, otros son administrativos y otros son de mantenimiento.

El docente tiene a su cargo cursos, el administrativo gestiones y el de mantenimiento tomacorrientes.

El docente no tiene nada que ver con tomacorrientes.

Hay datos generales que tiene cada empleado y que son comunes a todos pero después cada empleado tiene características propias de sus responsabilidades.

Las entidades “Hijas” heredan las características de la entidad “Padre”

Diagrama Entidad / Relación:

El diagrama que estamos haciendo el Lucidchart es el diagrama Entidad / Relación.

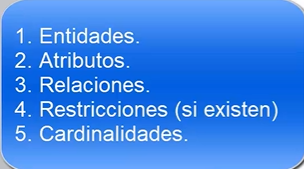
Es un diagrama donde se pueden ver las entidades con sus atributos y las relaciones entre ellas.

1. Se elabora el diagrama E/R con sus entidades, nos reunimos con el cliente y el cliente nos cuenta sobre su negocio.
2. Se completa el diagrama con los atributos, si no se pueden colocar todos los óvalos se confecciona una lista de atributos.
3. Se describen las restricciones que no se pueden reflejar en el diagrama. Por ejemplo, un atributo solo puede ser positivo, o mayor que 100.
4. Se agrega la cardinalidad a cada relación en el diagrama.

Este diagrama es la salida del Modelo Conceptual y este a su vez es la entrada al diseño lógico.

Aquí todavía no elegimos el DBMS, si va a ser Oracle, si va a ser móvil. Estamos entendiendo los datos, como están relacionados ,estamos entendiendo el modelo de datos.

El orden para hacer el diagrama E/S es:



Sección 6: Modelo Relacional

Que vemos en esta sección:

* Veremos los componentes del modelo relacional
* Estudiaremos el algebra relacional (básico)
* Repasamos el nivel de diseño deseado en esta etapa
* Practicaremos para formar un diagrama relacional

Al momento lo que tenemos es el diagrama Entidad / Relacion (E/R) con sus atributos.



Decimos que en el paso que vamos a ver ahora la entrada es el “Esquema Conceptual”

La salida va a ser un Modelo Relacional Ineficiente



Después vamos a aplicar dos procesos de análisis:

* Cardinalidad
* Normalización

Cuya salida va a ser un Diagrama Relacional Mejorado:



Redundancia de datos:

Repetición excesiva de una palabra o concepto.

En el diseño de bases de datos siempre habrá algún tipo de redundancia pero debemos tratar de eliminar las repeticiones innecesarias que deberían dañar el sistema.

En esta tabla



Vemos que se repiten los nombres (una persona puede tener mas de un auto) la marca (hay muchos autos d una marca) y el número de placa (un auto puede tener más de un dueño).

En este caso hablamos de redundancias fuertes que son innecesarias.

Vamos a ver procesos para evitar estas redundancias.

Tuplas:

Las tuplas en bases de datos asocian unívocamente los campos de una relación con sus respectivos valores, es decir son una fila de una tabla relacional.

Cada una de las tuplas en una relación es distinta a las demás, es decir no hay tuplas ( filas ) duplicadas.

En la tabla anterior tenemos 6 tuplas.

Relación Codd:

La base del modelo relacional son las tablas sin embargo Codd lo llamó relación.

En la relación Codd las filas se llaman tuplas y las columnas se llaman atributos

Hay una relación de Chen que es la relación de los gráficos E/R y no tienen que ver con Codd.

Vemos que en el diagrama E/S las entidades se transforman en tablas donde los atributos pasan a ser las columnas.

Algebra relacional:

Es un conjunto de operaciones que describen paso a paso como operar (obtener) una respuesta en base a relaciones dadas. Describe el aspecto de la manipulación de datos.

Estas operaciones se usan como una representación intermedia de una consulta de bases de datos. Por tanto ayuda a que dichas consultas sean eficientes y optimizadas.

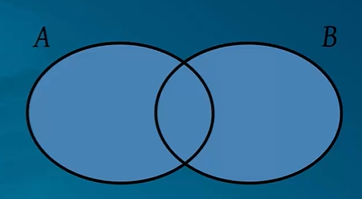
Las operaciones básicas son:

1. Selección / Restricción(A): Ayuda a seleccionar todas las tuplas que cumplan una cierta condición
2. Proyección(A): Permite extraer un número finito de atributos de una relación

El tabla anterior si quiero obtener las tuplas con el ID 1 voy a obtener los datos de Carlos Gomez.

Algebra relacional:

Para el siguiente diagrama de Venn:

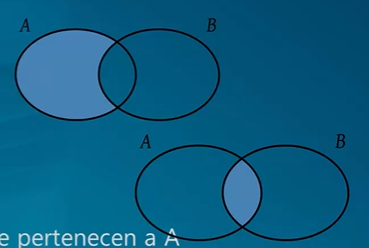


Las operaciones básicas son:

1. Producto Cartesiano (A,B): es la combinación de todas las tuplas de una relación A con una relación B

Son las filas de una tabla A que se relacionan con filas de la tabla B

1. Union(A,B): Es la unión de las tuplas que pertenecen a A o B o ambas
2. Diferencia(A,B): Es una nueva relación de tuplas que pertenecen a A pero no a B
3. Intersección(A,B): Corresponde a todas las tuplas que están en A y en B



Clave Primaria:

Es un atributo o subconjunto de atributos que permite distinguir unívocamente cada uno de los elementos que pertenecen a una entidad.

Es OBLIGATORIO rellenar los datos de una clave primaria en cada fila, no se pueden tener valores nulos.

En la siguiente tabla



Puedo tener muchos conductores que se llamen José Perez pero no puedo tener dos ID iguales, ID es una clave primaria, no acepta repeticiones.

Lo mismo para vehículos, puede haber muchos Ford iguales pero no puede haber ID iguales.

La ID de Vehículos es lo que se llama una clave foránea.

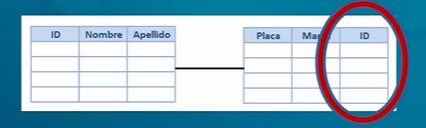
Las ID no pueden tener valores nulos

Clave Foranea:

Es un atributo o subconjunto de atributos que permite distinguir unívocamente a un elemento de una entidad externa que debe ser referenciado en la entidad actual.

La clave foránea es una clave primaria en otra entidad.

Si quiero saber quien es el dueño de un vehículo en esta tabla:



Tomo el ID del vehículo y voy a la tabla de nombres y busco allí el mismo ID. Así se quién es el dueño.

Componentes básicos del sistema relacional:

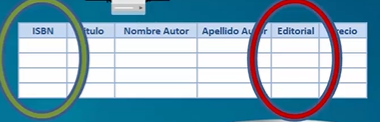
* Un grupo de tablas (relaciones Codd)



* Filas de las tablas (tuplas)



* Claves primarias y foráneas. Las claves **primarias** en una tabla NO se pueden repetir pero las claves **foráneas** SI porque son las claves primarias en OTRA tabla



* Tipos de datos para los campos de las tablas: Carácter, número…
* Columnas de la tabla (atributos o campos).

Para la práctica del Diagrama Relacional:

Los pasos a seguir son:

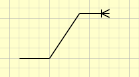
1. Cada entidad se transforma en una tabla
2. A cada tabla se le agrega un campo adicional que va ser el Primary Key (PK), salvo que ya tengan un campo que se llama ID xxxx
3. Si hay una relación que tenga atributos esa relación pasa a ser una tabla y se le DEBE agregar los PK de las tablas que relaciona como Foreigner Key (FK)
4. Toda relación que sea de muchos a muchos (N:M) pasa a ser una tabla y se le agrega las PK de las entidades que relaciona como FK de la tabla
5. Uno las tablas con líneas indicando la cardinalidad del grafico E/R

Para este ejemplo Sección 5 Practica 2

Punto 3: La relación Suministra tiene atributos y pasa a ser tabla. Se le agregan las PK ID Proveedor e ID Producto como FK de esta tabla.

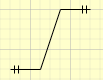
Punto 4: La relación Compra se agrega como tabla ya que es de “muchos a muchos” (muchos clientes pueden comprar muchos productos) y se le agregan las PK ID Cliente e ID Producto como FK a la tabla.

Por último, se agrega la cardinalidad del gráfico E/R como líneas entre las tablas:



Muchos

Uno

 Uno

Uno

Notas adicionales:

Tengo que agregar 2 filas más en la entidad suministra ya que esa tabla esta entre Proveedor y Producto sobre la Relación Suministra y tiene ATRIBUTOS. Tengo que saber QUIEN suministra el producto y QUE producto es, por eso agrego los Foreigner Key (FK) ID Provedor y ID Producto.

Agrego una fila mas en Producto ya que tengo que saber QUIEN es el proveedor del producto que estoy ofreciendo. Agrego un FK llamado ID del proveedor.

Agrego una tabla mas llamada Compra ya que se trata de una relación de Muchos a Muchos.

Sección 7

Que vemos en esta sección:

* Que es una normalización en una base de datos
* Estudiaremos las normalizaciones 1FH, 2FN, 3FN
* Entenderemos que es una dependencia funcional y transitiva
* Practicaremos cada una de estas normalizaciones

¿Qué es una dependencia funcional?

Decimos que un atributo B es dependiente funcional de A (puede ser de varios A1, A2, A3…An)

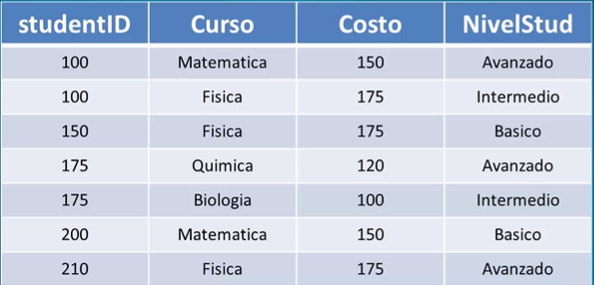
si el valor de A determina en todo momento el valor de B.

La notación sería algo así:

A - > B

A1, A2, A3 -> B

Veamos un ejemplo en una tabla:



¿De que va a depender el costo de cada curso? ¿Del ID? ¿Del Nivel?

El costo de cada curso depende DEL CURSO.

Curso -> Costo

Por otra parte el nivel del estudiante depende del curso y del Id porque es diferente para cada alumno

SudentID, Curso -> NivelStud

Dependencia transitiva:

Decimos que hay una dependencia transitiva cuando hay una dependencia intermedia. Tomando 3 atributos A, B y C y se cumple que A -> B B -> C hay una dependencia transitiva cuya notación es

A -> B -> C

Vamos a analizar para esta tabla las dependencias:



Vemos que existen las siguientes dependencias:

StudentID - > Residencia

StudentID - > Costo

Residencia -> Costo

Por lo que existe la dependencia transitiva:

StudentID - > Residencia -> Costo

Que es una normalización:

Es una técnica que ayuda a desarrollar una estructura lógica de los datos de un sistema informático en el modo relacional.

Fue introducida por EF Codd en 1972 donde define una estrategia de diseño de abajo hacia arriba, partiendo de los atributos, y estos se van agrupando en tablas según su afinidad.

Los objetivos de la normalización son:

1. Controlar la redundancia de la información
2. Evitar pérdidas de la información
3. Mantener la consistencia de los datos
4. Capacidad para representar toda la información

Pasos para realizar la normalización:

Al realizar esta técnica nos debemos asegurar que cada relación o tabla cumple con ciertas condiciones

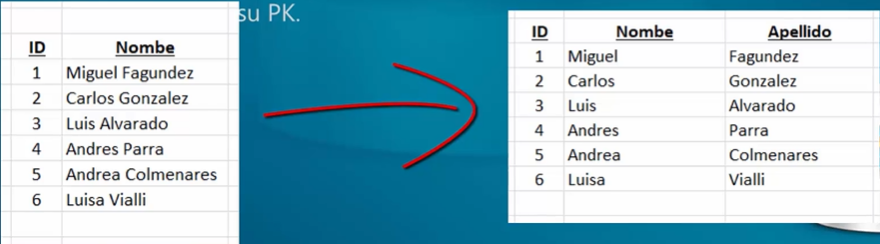
Si alguna regla o condición no se cumple, entonces esa tabla se debe descomponer en varias relaciones (tablas) que SI cumplan con esa regla.

Normalmente con realizar una normalización de 1ra, 2da o 3ra forma es suficiente para tener un modelo estable y menos susceptible a las anomalías estudiadas en este curso.

Primera Forma Normal (1FN)

Descomponer todos los grupos de datos en atributos atómicos. Además de que tabla debe tener su PK.

La tabla de izquierda tiene un campo llamado nombre pero en el mismo figuran Nombre y Apellido, se debe separar en dos campos ya que así se puede buscar por nombre o por apellido



No debo tener varios números TE en un campo:



Segunda Forma Normal (2FN)

Eliminar todas las relaciones en que los datos no dependan completamente de la PK de la tabla.

Si hay un campo que no dependa de la PK de la tabla se debe pasar a otra tabla.

Tercera forma normal (3FN)

Eliminar todas las relaciones que contengan dependencias transitivas. Si esto ocurre hay que descomponer esa tabla en 2 o más tablas.

Una vez que se haya efectuado la normalización se puede asegurar que el esquema es estable.

Eliminamos la redundancia y la perdida de datos.

Implementación de Normalización:

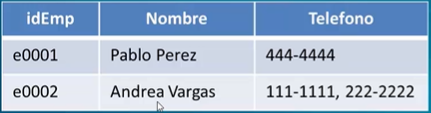
Primera forma Normal (1FN)

Descomponer todos los grupos de datos en atributos atómicos, buscar redundancias y la tabla debe tener al menos una PK.

Chequeamos que los datos no sean compuestos, por Ej:

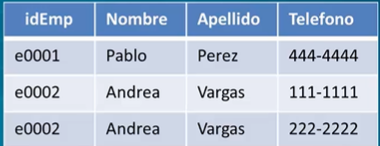
* Nombres
* Direcciones
* Teléfonos

Ej: Tengo la siguiente tabla



Tengo el TE de Andrea Vargas que es un campo compuesto

Puedo en primera instancia crear otra tupla (fila)



Pero si Andrea Vargas tiene 5 teléfonos tengo 5 filas totalmente repetidas con su PK, nombre, apellido y otros datos. Eso no es bueno.

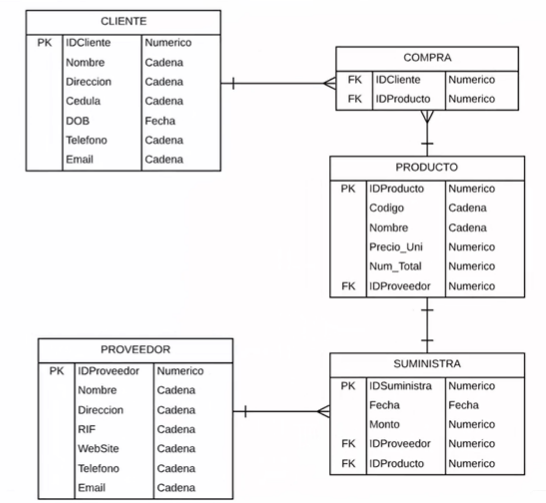
Una solución mejor es crear una tabla nueva de esta forma:



En la segunda tabla solo tengo como redundancia la PK lo cual no es ideal pero es mejor que repetir toda la fila con datos.

Primera forma Normal, arreglo de tablas.

Partimos del siguiente modelo relacional con su tabla:



Verificamos que cada tabla tenga su PK, en el caso de COMPRA la PK es la FK de otras dos tablas

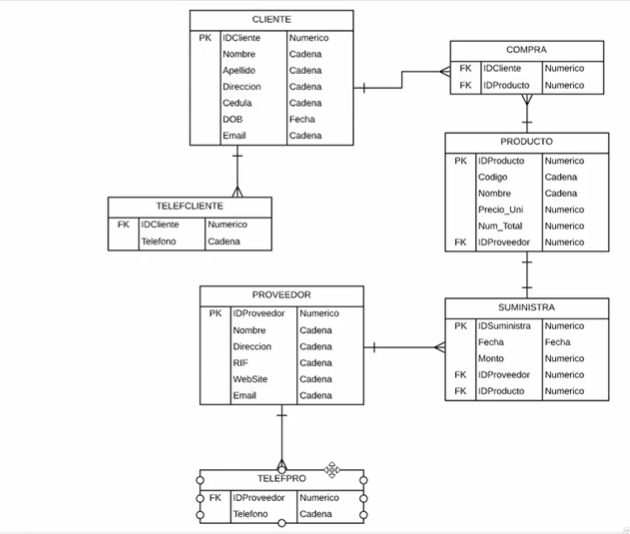
Problemas:

La tabla CLIENTE tiene el nombre en un campo solo en vez de Nombre y Apellido

Hay solo un campo para los teléfonos si es que el cliente tiene mas de uno.

En PROOVEDOR puede ser que alcance con solo un campo para el nombre porque es un nombre comercial pero hay un campo solo para teléfono.

Una solución para esto sería agregar tablas para que quede:



Agregamos dos tablas adicionales para los teléfonos del CLIENTE y del PROVEEDOR. Tenemos que solucionar el tema del Nombre del CLIENTE y de su direccion, esta última podría ser país-estado-codigo-calle-numero.

CONCLUSION: Cada tabla debe tener una PK y TODOS los atributos deben ser atómicos.

Segunda forma normal (2FN)

La tabla debe estar en primera forma normal (1FN)

Debemos chequear las dependencias funcionales de la tabla

A -> B A: Clave primaria B: Los campos de la tabla

TODOS los campos deben depender de una clave primaria.

Vemos la siguiente tabla:

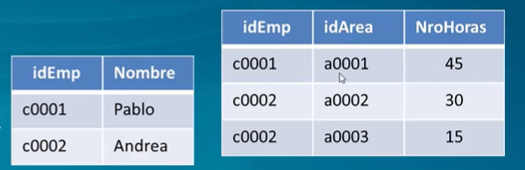


En esta tabla el nombre del empleado depende del idEmp pero si vemos NroHoras depende de idEmp y de idArea

idEmp -> Nombre La clave primaria es idEmp, nombre depende de esta

idEmp, idArea -> NroHoras Tenemos 2 claves primarias idEmp y idArea NroHoras depende de estas dos

Tenemos que dividir la tabla en dos



En la primer tabla tengo que nombre solamente depende de la PK idEmp

En la segunda tabla tengo dos PK: idEmp y idArea. Nro de Horas depende de esas dos PK.

Tercera forma normal (3FN)

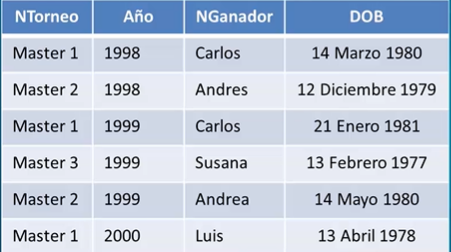
La tabla debe estar en segunda forma normal (2FN). Debemos chequear las dependencias transitivas de la tabla.

A -> B -> C

A: Clave primaria

B, C: El resto de los campos de la tabla.

Vemos en esta tabla:



Vemos que el NGanador depende de NTorneo y del año.

La fecha de nacimiento (DOB) depende del Nganador.

Las relaciones son estas:

NTorneo, Año -> Nganador

Nganador -> DOB

Vemos que DOB depende de otro campo que NO ES la clave primaria PK

Podemos descomponer esta tabla en otras dos:



Siempre tenemos que verificar que las tablas cumplan las 3 formas normales.

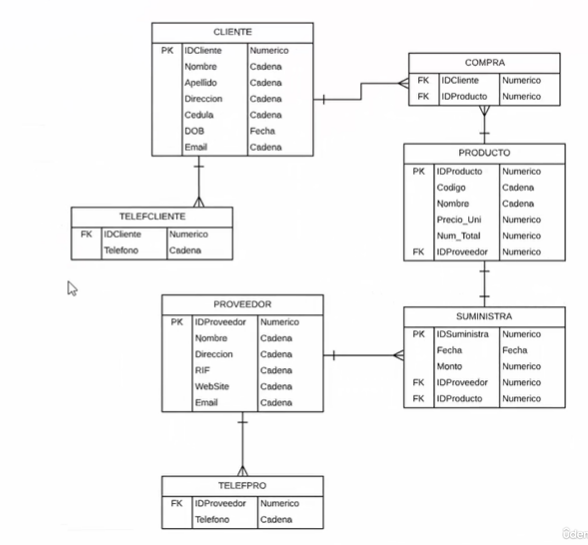
Vemos que en este caso la primera tabla no se cumple con la primera forma normal ya que no hay un campo PK. El Nganador se repite por lo tanto no puede ser PK.

Agrego en la primer tabla un ID que va a ser el PK de esta tabla y en segunda tabla reemplazo el nombre por el ID de la primer tabla, entonces le estoy agregando un FK que es la PK de la primer tabla.



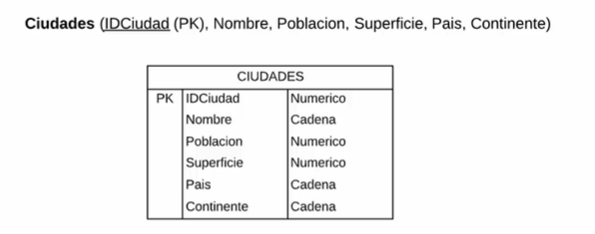
PK de la primer tabla FK

Si hacemos el análisis de normalización vemos que en esta base de datos todas las tablas cumplen con las 3 condiciones:

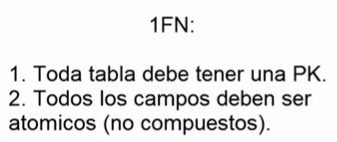


Vamos a ver un caso donde no se cumplen las 3 formas normales:

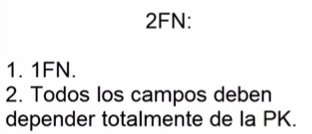
Veamos esta tabla:

****

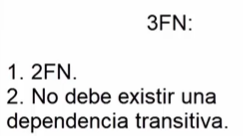
Vemos que se cumple la 1FN



Cumple la 2FN



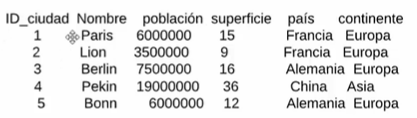
Vemos que pasa con la 3FN



Vemos que:

Ciudad -> País y País -> Continente por lo tanto Ciudad - > Continente Es transitivo

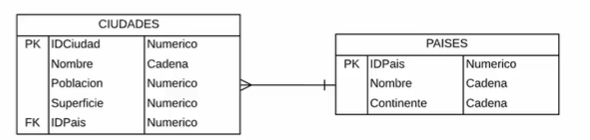
Lo podemos ver con algunos datos:



Vemos que se repite Francia, Europa y Alemania, Europa. Si tenemos 200 ciudades francesas vamos a repetir 200 veces Francia y Europa lo cual es una duplicación de datos importante

¿Como lo corregimos?

Separamos la tabla en dos



En ciudades tenemos un FK que es el PK de países. En este caso estamos repitiendo 200 veces solo un campo en vez de dos ( IDPais en vez de País y Continente) la cual es una duplicación de datos menor que el caso anterior.