**DISEÑO DE BASE DE DATOS**

**Base de datos:**

Es una coleccion de archivospersistentesdiseñados y relacionados.

Estacompuesta por tipos de datos, estructuras y restricciones de datos

La construccion de BD es elproceso de almacenardatos bajo la gestión del DBMS

DBMS: (SGBD)

Es una colección de programas que permiten mantener la base de datos

Características DBMS:

Evitar redundancia e inconsistencia

Acceso en todo momento a los datos

Evitar anomalías en acceso concurrente

Restricción a acceso no autorizado

Almacenamiento de datos persistente ante fallos

Integridad de datos

Backup

Roles en una BD:

Usuarios

Programadores

Analista de sitemas

Diseñador de BD

DBA o ABD

Tipo de modelos:

Basados a objetos: (modelo conceptual)

. Modelo Entidad-Relacion

. Modelo Orientado a Objeto

Basado a Registros: (estructurado)

. OO (traeria todo un objeto, ineficiente)

. Relacional (Se usa hoy en dia)

. Jerarquico (muy rapido pero a mano, se dejo de usar en 70)

. Red (se dejo de usar en el 60)

No SQL: (no explico)

Fisico de datos (no se usa mas)

Etapas de diseño de base de datos:

-Conceptual (representación abstracta)

-Logica (representacion en una computadora)

- Fisico (determinar estructiras fisicas)

**MODELADO CONCEPTUAL**

**Modelo entidad relacion (ER)**

Compuesto por:

. Entidades (rectangulo)

. Relacion (rombo)

. Atributos (linea con circulo)

**Tipos de relacion:**

Recursiva (se llama a si misma)

Binaria (entre 2, ideal)

Ternaria (entre 3, puede ser justificable)

N-aria (entre N, injustificable)

**Tipos de cardinalidad:**

. Minima

. Maxima

Muchos a muchos con cobertura parcial de un lado. (0, N) (1, N)

Muchos a muchos con cobertura total de ambos lados (1, N) (1, N)

Parcial cuando uno de los 2 minimls es 0

Total cuando uno de los 2 minimos es 1

**Atributos:**

-Obligatorio (como minimo un valor)

-Opcional

-Monovalente (como maximo un valor)

-polivalente

Pueden ser de una entidad o relacion

Lo comun de un atributo es que sea monovalente y obligatorio (1,1), por lo que no se escribe la cordinalidad y queda implícito

Tipo atributo

Atributo derivado:

Son atributos que su valor proviene de un conjunto de otros atributos

Pero hay que tener en cuenta que repiten valores

Relaciones uno a uno no piede tener atributo

Relaciones muchos a mucho puede tener atributo

Basicamente por sie en una relación 1 a 1 los atributos podrian estar tanto en la relacion o en la entidad.

Por lo que se pone en la entidad

Cuando se duda entre generar una entidad y un atributo, suele ser mejor crear un a entidad. Pero hay que tener en cienta que una entidad es un archivo mas

**Identificadores:**

Tipo de atributo unico, es decir claves

Compuestos (la clave compuesta por 2 atributos)

Simples

Externos (cuando un atributo se encuentra en otra entodad / relacion)

Internos

Se representa como un atributo con el circulo lleno

Un identificador es una clave univoca o primaria

Lo identificadores deben marcarse todos

Son monovalente y obligatorios (1,1)

**Tipos de modelos:**

Relacional

**Jerarquias:**

Relacion padre hijo

Los hijos heredan los ayributos del padre

Tipo cobertura

Total:

. Si todos los elementos del padre tienen cobertura en alguno de sis hijos

Parcial:

. Si hay algun elemento del padre no tepresentado con algun hijo

Superpuesta:

. Si una entidad puede pertenecer a uno o mas hijo

Exclusiva:

. Si lo puede representar unicamente un hijo

**MODELADO LOGICO**

Modelo logico:

Se compone mediante el esquema conceptual, las reglas (OO o relacional), criterios de uso y criterios de carga

Criterios de uso:

El uso que le puede llegar a dar los clientes al software

Criterios de carga:

Cuantos datos vamos a tener y cuanto lo vamos a explicar

Los criterios de uso y carga los vamos a esquivar un poco

Nosotros vamos a hacerlo relacional

**Modelo relacional:**

NO tiene atributos compuestos, polivalentes y jerarquias

Unicamente derivados y ciclicos

Los atributos derivados y ciclicos:

Atentan contra la minimalidad

Derivados:

Se puede elegir actualizarlo todo el tiempo o calcularlo todo el tiempo, la opcion que se elija depende del criterio de uso

Ciclos:

Cuando las relaciones son ichks aicho no hay problema. El problema es cuando no lo es, que hay que corroborar que el ciclo no repita informacion

Si sacando una relacion, se sigue pudiendo obtener todos los atributos, entonces esa relacion esta repitiendo informacion.

Queda a criterio del uso y la facilidad que aporte dicha relacion si se la mantiene (por mas que redunde) o se la borra y se ahorra espacio

**Modelos no soportados en relacional:**

**Compuestos:**

Solucion 1: (no se usa)

Se cambia el compuesto por una entidad y se pone los atributos todos juntos en un solo Sting (funciona unicamente para atributos simples, no para polivalente)

Solicion 2:

Se hace una entidad y se deja todo como atributos simples

Solucion 3: (tampoco se usa)

Solucion 4:

Se colocan todos los atributos del atributo compuesto en la entidad base.

**Herencia solucion:**

Solucion 1:

matar a los hijos, es decir pasar todo al padre

Optimiza en cuanto a espacio y es simple. Se puede usar un atributo de tipo, para identificar que clase de hijo es

Solucion 2:

Matar al padre, pasar todo a los hijos.

Es decir los atributos del padre se pasan a los hijos, por lo que se repiten

Solucion 3: se usa hoy en dia

Dejar todo como esta, pero transformar la jerarquia en relaciones

**DISEÑO FISICO**

Modelo fisico:

Vamos a seguir usando relacional.

Elaccion DBMS: mySQL, oracle, etc.

El modelo relacion esta basado en algebra relacional de productos cartesianos

Cualquier entidad del modelos logico es una tabla del modelo fisico

Un registro de un archivo en logico, es una tupla en fisico

En el modelo fisico debemos definir claves, primarias o univocas, secundarias

Una entidad se convierte a una tabla

Se debera convertir el identificador en clave univoca, y elegir uno como primario

O mas recomendable usar un atributo ID autoincremental, generado por el DBMS

**Las relaciones:**

NaN: (muchos a muchos)

-Siempre tabla, una tabla para cada identidad y una tabla para relacion.

1aN: (muchos a muchos)

(1,1) a (1,n) y (1,1) a (0,n) :

Se soluciona usando IR, con una clave primaria, siendo foranea en una entidad

Cuando es opcional, se soluciona con null cuando vacio.

Los atributos de la relacion en este caso pasa a la identidad de lado unico como atributo

(0,1) a (1,n):

Solucion teorica es con tabla, pero inpractica.

Solucion real usando IR

(1,1) a (1,1):

Se crearia una tabla concatenando ambas entidades

(0,1) a (1,1) de jerarquia:

Se solucionaria con 3 tablas donde los 2 hijos tiene la misma clave foranea como univocas

NaNaN: ternaria

Igual que un NaN normal

(1,1) a (?,n) recursiva:

Se guarda la clave foranea a otra identidad del mismo tipo, pero se cambia el nombre de la clave foranea para que no se repita el nombre en una misma entidad

**Integridad referencial:**

Se da cuando un atributo es clave primaria en una tabla

Y es clave foranea en otra tabla, soliendo ser claves secundarias en dicha tabla.

Bajar IR soluciones:

-Restringir: (es lo mas usado)

Es decir no te va a dejar borrar, hasta borrar otras cosas

-Cascada: (se puede llegar a usar)

Si solicita borar y se borra todo lo relacionado

-Null:

No se borra, se remplaza por null

-Ni bola:

Pide borrar y se borra eso

Modificacion IR:

Los ID no se deberian modificar nunca.

Siendo el caso de ser necesario modificarlo, se usan las mismas soluciones que en baja

**RESTRICCIONES:**

De domino:

Especificamos el dominio de a

De clave:

Evitar que se repitan claves

De nulo:

Que acepten nulos

De integridad:

Que una clave no sea nulo

**DEPENDENCIAS FUNCIONALES:**

Otra restricción entre dos conjuntos

Dice que si una clave se encuentra en 2 tablas, entonces una misma variable sqie se emcientra en ambas tablas, deberian tener un mismo valor

Ej: (-> = determina )

N Legajo -> DNI y DNI -> N Legajo

DNI -> Nombre pero Nombre NO -> DNI

Dependencia funcional completa:

DNI -> Nombre

Dependencia funcional parcial:

DNI + Legajo -> Nombre

Uno de los atributos sobra, inecesario

Dependencia funcional transitiva:

ID.Empl -> ID.Dep

ID.Empl -> Nom.Dep (repetitivo, NO va)

ID.Dep -> Nom.Dep

En una misma tabla estamos repitiendo dependencias, esta mal hecho. NO debe haber redundancia

**NORMALIZACION:**

Hay 6 formas

Las primeras 3 se deben cumplir siempre.

Teoricamente para cumplir la 4ta y 5ta debes cumplir las predecesoras

Primera forma normal:

Ninguna tabla debe tener atributos polivalentes

Segunda forma normal:

No tiene dependencias funcional transitiva.

Con claves univocas no existe ese problema, existe unicamente con claves cruzadas

Tercera forma normal:

Boyce Codd forma normal: (BCNF)

Si algun determinante no es clave univoca, no hay problema.

Es decir hay problema si todas

Se decide si perden una clave candidata o mantenerla y quedarse en tercera forma.