**Modelado**

DBD – Diseño de Base de Datos

Base de Datos 🡪 Es una colección de datos relacionados.

🡪 Colección de archivos diseñados para servir a múltiples aplicaciones.

🡪 Un dato: representa hechos conocidos que pueden registrarse y que tienen un resultado implícito.

* Propiedades implícitas:
  + Una BD representa algunos aspectos del mundo real, a veces denominado Universo de Discurso.
  + Una BD es una colección coherente de datos von significados inherentes. Los datos deben tener cierta lógica.
  + Una BD se diseña, construye y completa de datos para un propósito específico. Está destinada a un grupo de usuarios concretos y tiene algunas aplicaciones preconcebidas en las cuales están interesados los usuarios.
  + Una BD está sustentada físicamente en archivos en dispositivos de almacenamiento persistente de datos.
* Definir una BD 🡪 consiste en especificar los tipos de datos, las estructuras y restricciones de los mismos.
* Construir una BD 🡪 es el proceso de almacenar datos concretos en algún dispositivo de almacenamiento bajo la gestión del DBMS.
* Manipulación de BD 🡪 incluye funciones tales como consultar la BD para recuperar datos específicos, actualizar los datos existentes, reflejar cambios producidos, etc.

DBMS 🡪 Data Base Management System

* Es una colección de programas que permiten a los usuarios crear y mantener la BD.
* En un sistema de software de propósito general que facilita los procesos de definición, construcción y manipulación de BD.

Objetivos de un DBMS:

1. Evitar redundancia e inconsistencia de datos (por mal diseño o mal uso de la DBMS, la redundancia implica datos repetidos y esto es incorrecto).
2. Permitir acceso a los datos en todo momento (DBMS operable todo el tiempo).
3. Evitar anomalías en el acceso concurrente.
4. Restricción a accesos no autorizados 🡪seguridad (ej, contraseña y usuario para guardar datos).
5. Suministro de almacenamiento persistente de datos (aún ante fallos), el DBMS proporciona estrategias para evitar los fallos y cuidar los datos.
6. Integridad en los datos (evitar que situaciones no deseadas puedan ocurrir).
7. Backups (debe permitir que la BD sea aún utilizable cuando se está haciendo el backup, el backup es más inteligente y eficiente cuando no siempre se hace de cero 🡪 backup incremental).

Componentes de un DBMS:

>>**DDL (data definition languaje):** especifica el esquema de BD para crear la base de datos, también sirve para hacer modificaciones a la estructura de la BD. Resultado🡪diccionario de datos.

>>**DML (data manipulation languaje):** el que manipula la DBMS 🡪recuperar información, agregarla, quitarla, modificarla.

>>Características DML, dos tipos:

° Procedimentales (SQL) 🡪requieren que el usuario especifique qué datos se muestran y cómo obtener dichos datos.

° No procedimentales (QBE) 🡪requieren que el usuario especifique qué datos se muestran y sin especificar como obtener esos datos. (generalmente se usa para controlar)

Actores involucrados con una BD:

* **DBA** o ADB: Administra el recurso, que es la BD. Autoriza accesos, coordina y vigila la utilización de recursos de hardware y software, responsable ante problemas de violación de seguridad o respuesta lenta del sistema.
* **Diseñador** de BD: Definen la estructura de la BD de acuerdo al problema del mundo real que esté representando.
* **Analistas de Sistemas**: Determina los requerimientos de los usuarios finales, generando la información necesaria para el diseñador.
* **Programadores**: Implementan las especificaciones de los analistas utilizando la BD generada por el diseñador. Crea los algoritmos que satisfacen las especificaciones de los analistas utilizando la BD generada por el diseñador.
* **Usuarios** (distintos tipos): los que usan la BD, no se ven implicados en la creación y mantenimiento de la BD.

>> Los especificados arriba son roles, no significan personas.

Modelado 🡪 Abstracciones.

Colección de herramientas conceptuales para describir datos, relaciones entre ellos, semántica asociada a los datos y restricciones de consistencia.

* Visión: Ve solo los datos de interés (muchas vistas para la misma BD) a través de programas de aplicación.
* Conceptual: Qué datos se almacenan en la BD y qué relaciones existen entre ellos.
* Físico: Describe cómo se almacenan realmente los datos (archivos y hardware).

Para dar una solución de vista o conceptual vamos a tener que apelar a modelar, ósea, construir un modelo de BD.

>>Usaremos una colección de herramientas conceptuales para describir datos, relaciones entre ellos, semántica asociada a los datos y restricciones de consistencia (para mantener una lógica coherente de datos).

* Modelos:
  + Basado en objetos (visión, conceptual). Estructura flexible, especifican restricciones explícitamente 🡪 Modelo de Entidad-Relación, Modelo Orientado a Objetos.
  + Basado en registros (conceptual, físico). La BD se estructura en regiones de formato fijo. Se dispone de lenguaje asociado para expresar consultas 🡪 OO, **relacional**, jerárquico (hasta los ’80), red (hasta los ’60).
  + Físico de datos.

Diseño de datos:

* Conceptual: Representación abstracta 🡪 Integración de vistas.
  + - Genérico
    - Alejado del tipo de DBMS
    - Alejado del producto particular
    - Maqueta de lo que el cliente quiere
    - No importa DBMS ni modelos físicos de datos.
* Lógico: Representación en una computadora.
  + - Más específico
    - Orientado a un tipo de DBMS
    - Alejado del producto particular
    - Planos, representación en una computadora
* Físico: Determinar estructuras de almacenamiento físico.
  + - Específico
    - Orientado a un producto

Modelo de Datos 🡪 Sirve para hacer más fácil la comprensión de los datos de una organización.

* Se modela para:
  + Obtener la perspectiva de cada actor asociado al problema.
  + Obtener la naturaleza y necesidad de cada dato.
  + Observar cómo cada actor utiliza cada dato.

Modelo Entidad-Relación:

* Características:
  + Estándar Internacional desde 1988
  + Propuesto por Chen en 1976
  + Ampliado por Codd en 1979
  + Se basa en la concepción del mundo real como un conjunto de objetos llamados entidades y las relaciones que existen entre ellas.
  + Permite modelar el nivel conceptual y lógico de una BD.

***Modelo Conceptual Entidad-Relación:***

* Objetivos:
  + Representar la información de un problema en un alto nivel de abstracción.
  + Captar la necesidad de un cliente respecto del problema que enfrenta.
  + Mejora la interacción cliente/desarrollador disminuyendo la brecha entre la realidad del problema y el sistema a desarrollar.
* Características:
  + **Expresividad**: Disponer de todos los medios necesarios para describir un problema.
  + **Formalidad**: Cada elemento representado sea preciso y bien definido, con una sola interpretación posible.
  + **Minimalidad**: Cada elemento tiene una única representación posible.
  + **Simplicidad**: El modelo debe ser fácil de entender por el cliente y por el desarrollador.
* Componentes:
  + Entidades: Elementos que componen el modelo.
    - Representa un elemento u objeto del mundo real con identidad.
    - Se diferencia de cualquier otro objeto o cosa.
    - Conjunto de Entidades 🡪 Representación que, a partir de las características propias de cada entidad con propiedades comunes, se resume en un núcleo.
  + Relaciones: Cómo se relacionan las entidades del problema.
    - Representan agregaciones entre dos (binaria) o más entidades
    - Conjunto de Relaciones 🡪 Es una representación que, a partir de las características propias de cada relación existente entre dos entidades, las resume en un núcleo.
    - Tipos:
      * Binaria
      * Ternaria
      * N-aria
      * Recursiva
    - Cardinalidad de relaciones 🡪 Define el grado de relación existente en una agregación.
      * Cardinalidad Máxima
      * Cardinalidad Mínima
  + Atributos:
    - Representa una propiedad básica de una entidad o relación.
    - Equivale a un campo de un registro.
    - Cardinalidad de atributos 🡪 Define la cantidad de valores posibles que va a tener un atributo en un momento determinado.
      * Monovalente (x..1) /polivalente (x..n)
      * Obligatorio (1..x) /opcional (nulo) (0..x)

**\*Los atributos en las relaciones sólo son válidos si tiene cardinalidad muchos a muchos de ambos lados.\***

Componentes adicionales de modelado:

* Atributos compuestos 🡪 Representa un atributo generado a partir de una combinación de atributos simples.
  + Pueden ser polivalentes y no obligatorios.
  + Sus atributos simples pueden ser polivalentes y no obligatorios también.
* Jerarquías 🡪 Permite extraer propiedades comunes de varias entidades (o relaciones) y generar una súper-entidad que las contenga.
  + Cobertura:
    - Total: Todos los elementos del padre están contenidos en los elementos hijos.
    - Parcial: Los elementos hijos no cubren totalmente al padre.
    - Superpuesta: Cuando un elemento del padre puede ser parte de cualquiera de las entidades hijas.
    - Exclusiva: Cuando un elemento del padre puede ser parte de uno y sólo uno de las entidades hijas.
* Subconjuntos 🡪 Cuando una jerarquía presenta un solo conjunto de entidades como hijo.
  + La cobertura de un subconjunto es PARCIAL EXCLUSIVA únicamente.
* Identificadores 🡪 Es un atributo o conjunto de atributos que permite reconocer una entidad de manera unívoca dentro del conjunto de entidades.
  + Pueden ser:
    - Simples: Están conformados por un solo atributo.
    - Compuestos: Están conformados por más de un atributo.
    - Internos: Identificadores propios de la unidad.
    - Externos: Identificadores que “nos traemos” de otra entidad con relación (1, 1) de mi lado y (x, n) del otro.
      * Tiene que ser monovalente y obligatorio.
* Atributos derivados 🡪 Aparece en un conjunto de entidades y representa información que se puede obtener de otra forma. (se puede calcular)
  + Ventajas:
    - Es mucho más rápido mostrar el resultado que si tuviera que calcularlo.
  + Desventajas:
    - Repite información.
    - Ocupa lugar.

Decisiones:

* ¿Conviene generar una entidad con un concepto nuevo? O agregar un atributo a una entidad existente?
  + En general, cuando un atributo puede convertirse en una entidad, conviene que sea una entidad. Igualmente, la conveniencia es lo que el diseñador decide en base a lo que charla con el cliente. Por ejemplo, si yo dudo entre generar un nuevo atributo o una nueva entidad (agregando con una relación), me conviene MÁS generar una entidad, ya que a la larga me genera más beneficios.
* ¿Cuándo se debe utilizar una generalización y cuándo el concepto representa una clasificación?
  + Las clasificaciones carecen de atributos para las entidades, jerarquía en la que las entidades hijas no tienen atributos ni relaciones, quedan como entidades muertas.
  + Básicamente generalizo cuando puedo aplicar jerarquías, clasifico cuando armo una jerarquía y los hijos NO tienen atributos, por lo tánto para eso me basta una única entidad sin aplicar jerarquía.
* ¿Convienen los atributos compuestos? O se deben generar atributos simples?
  + Son una herramienta más del modelo, pero no son obligatorios. Los compuestos son para describir más correctamente, es a decisión del diseñador.

Compleción (modelo completo): Representa todas las características del dominio de aplicación (análisis de requerimientos). Un modelo es completo cuando representa todas las características del dominio de aplicación (análisis de requerimientos). Hay que revisar problemas, requerimientos, enunciados, y ver que todo se cumpla.

Corrección: Usar una propiedad conceptos E-R

> **Sintáctica**: si los conceptos del E-I se usan correctamente.

> **Semántica**: los conceptos se usan de acuerdo a su definición, ósea, si no cometiste error de escritura o si no cometiste errores de interpretación a la hora de construir el modelo.

Errores más frecuentes:

a) Usar atributos en lugar de entidades (pregunta anterior).

b) Olvidar una generalización (omitir jerarquías).

c) Olvidar una propiedad de herencia (casi imposible de que ocurra, puede que ocurran superposiciones 🡪 ej, definir un atributo en el padre y redefinirlo en el hijo).

d) Usar entidades en lugar de interrelaciones.

e) Olvidar un identificador de una entidad (no se puede olvidar, excepto si la entidad NO tiene un identificador posible).

f) Omitir cardinalidad (nada es obvio, siempre se debe definir necesariamente).

Minimalidad: Cada aspecto aparece una sola vez en el esquema. Si hay información redundante ya hay problemas contra la minimalidad y DEBE SOLUCIONARSE.

* Ciclo de relaciones: como un gráfo, un ciclo de relaciones sería: A se relaciona con B, B se relaciona con C, C se relaciona con A. 🡪 puede representar información redundante, atenta a la minimalidad.
* Atributos derivados: atenta con la minimalidad.

Expresividad: representa los requerimientos de manera natural y se puede entender con facilidad (todo esto en el modelo conceptual, sí podemos entender de muy buena forma el modelo, significa un modelo expresivo)

Auto-explicación: Esquema que se explica a si mismo cuando puede representarse un gran número de propiedades usando el modelo conceptual, sin otros formalismos.

* Eliminar sub-entidades colgantes de la generalización.
* Eliminar entidades colgantes.
* Crear generalización: dos entidades similares, crea una jerarquía de generalización.
* Crear subconjuntos.

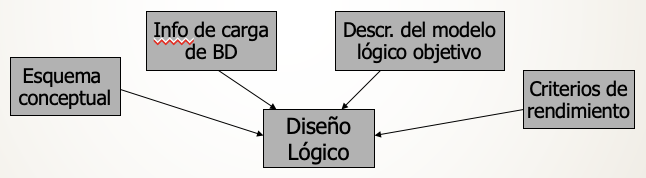
Extensibilidad: Un esquema se adapta fácilmente a requerimientos cambiantes cuando puede descomponerse en partes, donde se hacen los cambios.

Legibilidad:

* Utilizar herramientas automatizadas.
* Estructuras simétricas.
* Se minimiza el número de cruces.
* Generalización sobre los hijos.

***Modelo Lógico***

* Diseño lógico de alto nivel usando E-R
  + Convertir el esquema conceptual en un esquema lógico.
  + Enfoque global del diseño lógico.
* Una vez terminado el módelo conceptual, se llega al diseño lógico que contempla todas las necesidades del cliente anteriormente planteadas.



1. Modelo lógico objetivo: es el acercamiento a la solución en computadora (se decide el modelo lógico objetivo para proceder) 🡪 consiste en decidir: orientado a objetos (se usa pero no a nivel mundial), relacional (se usa y vamos a trabajar éste), jerarquico (desuso), orientado a red (desuso). Para cada modelo lógico se llegan a distintas soluciones.
2. Información de carga de BD y Criterios de rendimiento: los criterios de rendimiento sirven de orientación a la hora de tomar decisiones que afecten al rendimiento de la BD (ej, una solución legible vs una solución rápida). La información de carga de la BD 🡪 es la suposición de cómo van a estar los datos. No hay que cargar mucho un lugar (una entidad con muchos elementos)

* Decisiones:
  + Atributos derivados (minimalidad atentada).
  + Atributos compuestos (no se admiten atributos compuestos en modelo relacional).
  + Atributos polivalentes (no se aceptan en modelo relacional).
  + Ciclo de relaciones (minimalidad atentada).
  + Jerarquías (un modelo relacional no admite jerarquía, ni tiene concepto de herencia).

Atributos derivados:

* Ventaja de dejar:
  + Si necesito el dato muy seguido, tengo rápido acceso. Nos permite conseguir un valor X muy rápido, es más eficiente que ESTÉ en la BD. El tema con los atributos derivados es que hay que mantenerlos ACTUALIZADOS.
  + Si el dato cambia poco lo dejo
* Ventaja de sacar:
  + Si no uso el dato seguido, gasto mucho en tenerlo actualizado.
  + Si el dato cambia mucho debo actualizarlo muy seguido.
  + Si lo saco debo calcular el valor X, por lo que implica ineficiencia.

Resumen 🡪 cambia mucho, se usa poco, ENTONCES SACO. Cambia poco, se usa mucho, LO DEJO. (decisión extrema, pero es decisión nuestra)

Atributos compuestos:

* Opción A: generar un único atributo que se convierta en la concatenación de todos los atributos simples que contiene el compuesto (usada en el pasado).
* Opción B: definir los atributos simples sin un atributo compuesto que resuma. Aumenta cantidad de atributos pero permite definir cada dato de forma independiente (la que usa Bertone 🡪 apunta a esta).
* Opción C: generar una nueva entidad que va a representar un atributo compuesto, conformada por cada atributo simple que contiene. Dicha nueva entidad se relaciona con la entidad a la cual pertenecía el atributo compuesto .

Atributos polivalentes:

* Un atributo en el modelo relacional debe ser necesariamente simple.
* Se genera una nueva entidad con ese atributo.
* Se genera una relación entre la nueva entidad y la entidad que contenía el atributo polivalente.

Ciclos de entidades:

* Si el modelo no es mínimo, decido:
  + Si accedo mucho 🡪 Lo dejo, mejor tiempo de acceso
  + Si no lo utilizo mucho 🡪 Lo saco, evito redundancias
* Si el modelo no atenta contra la minimalidad (no se puede acceder a ese dato por otro camino del ciclo) 🡪 Lo tengo que dejar.
* Al fin y al cabo debemos tener una balanza para decidir si sacar la relación que provoca redundancia, o dejarla.

Ejemplo: yo quiero ir de Quilmes a Mar Del Plata, y quiero ir por ruta 2 pero está cortada, puedo ir por otra ruta y llegar a MDP?, la respuesta es SÍ, por lo tanto la ruta 2 es redundante.

* Si yo tengo más de un camino entre una entidad a otra, entonces hay redundancia que ATENTA CON LA MINIMALIDAD.
* El tema final es el siguiente, sí el camino adicional que atenta contra la minimalidad se trata de un camino que agiliza y que EFECTIVAMENTE es usado bastante entonces nos conviene tener dicha redundancia a mano.
* Puede ocurrir que exista un ciclo, pero que éste sea mínimo, es decir, no hay redundancia, y todas las relaciones son importantes: si saco alguna relación no se cumple la interpretación correcta del problema.

Jerarquías:

* No existen en el modelo relacional.
* Soluciones:

1. Eliminar entidades hijas: (T, E) – (T, S) – (P, E) – (P, S)
   * Deja sólo la entidad padre, la cuál incorpora todos los atributos de sus hijos. Cada uno de éstos atributos deberá ser opcional (cardinalidad mínima en 0). Puede aparecer un atributo monovalente obligatorio llamado “categoría”, el cual nos va a decir a qué entidad hija corresponde el dato guardado en persona.
   * Resume las entidades y las relaciones. Pero saca más identificadores.
2. Eliminar entidad padre: (T, E)

* Deja sólo las especializaciones (hijas) con ésta solución los atributos del padre deberán incluirse en cada uno de los hijos. Con el tema identificadores, la herramienta CASER apuesta x una solución segura y, si en hija y padre hay identificador, se genera un identificador compuesto en la hija (es la solución segura, se puede modificar para tener identificadores simples).
* Genera más relaciones y entidades. Conserva identificadores para futuras búsquedas.
* Si la cobertura de la jerarquía es parcial no se puede aplicar la solución para eliminar la entidad padre porque algunos elementos del padre no están cubiertos por las hijas. Si se quita la entidad padre dichos elementos no tendrán más cabida en el modelo (se pierde información).

1. Conservar todo: (T, E) – (T, S) – (P, E) – (P, S)

* Convierte la jerarquía en relaciones 1 a 1 entre el padre y cada uno de los hijos, ésta solución permite que las entidades de la jerarquía mantengan sus atributos con una relación explícita entre padre e hijos.
* Es la solución más larga, pero es la que mejor capta el modelo conceptual. Hoy por hoy es la más eficiente y más usada. La pérdida de performance NO es significativa, es una solución más simple y ordenada 🡪 permite que el sistema sea más fácil de mantener a futuro

***Modelo Físico (relacional) 🡪*** Representa la BD como una colección de relaciones.

* En otros términos 🡪 Cada relación se asemeja a una tabla de valores, o a un archivo plano de registros.
* Un registro o un elemento de una relación (tabla) se denomina tupla.
* Un atributo mantiene su nombre.
* Cada tabla de valores resultante se denomina relación.
  + Cada relación se obtiene a partir de una entidad o una relación ER. Ósea que las entidades o las relaciones de ER son pasadas a tablas.
* El tipo de datos que describe los tipos de valores de un atributo se denomina dominio.
* No se debe confundir la palabra “relación” del modelo físico con la del modelo conceptual ER, no son lo mismo.

Pasos:

* Eliminación de identificadores externos 🡪 sí hay. La tabla solo puede tener identificadores internos.

1. Selección de claves: (se vuelve a usar el concepto de clave):
   * Primaria 🡪 Método de hash. El ID no cambia NUNCA y el cliente NO lo conoce.
   * Candidata 🡪 Se tratan como árboles. Univoca
   * La clave primaria está propensa a errores (no está bueno).
   * Desde fines del siglo pasado se propone que el atributo que sea clave primaría sea el atributo **AUTOINCREMENTAL.**
   * ¿Qué significa esto?:
     + Significa definir en cada tabla un nuevo atributo que hasta ese momento no estaba presente en el modelo.
     + Dicho atributo va a ser manipulado por el DBMS (el DBMS le pone un valor NUNCA REPETIDO) 🡪 va a ser ideal porque: está manejado por la DBMS, permite búsquedas eficientes y está más allá del usuario (el usuario no sabe qué existe).
     + Por lo tanto, **LA CLAVE PRIMARÍA DEBE SER UN VALOR AUTOINCREMENTAL.**
     + Los identificadores restantes pasan a ser claves candidatas.

* Conversión de entidades:
  + Cada entidad del modelo E-R es una tabla
* Relaciones:
  + Cardinalidad Muchos a Muchos 🡪 La relación es una tabla
  + Cardinalidad Uno a Muchos
    - Clave foránea: atributo/s de una tabla que en otra tabla es/son CP y que sirven para establecer un nexo entre ambas estructuras.
    - Cobertura Total: (1,1) <--> (1,n)
      * + La relación no se convierte en tabla.
        + Se agrega a (1,1) como clave foránea el identificador del lado (1,n)
    - Cobertura Parcial:
      * + Del lado de muchos (1,1) <--> (0,n):

La relación no se convierte en tabla.

Se agrega a (1,1) como clave foránea el identificador del lado (0,n)

* + - * + Del lado de uno (0,1) <--> (1,n):

La relación no se convierte en tabla y no modifica análisis.

La relación se convierte en tabla

* + Relaciones Recursivas:
    - Se trata igual que las binarias.
    - Parcial del lado de uno:

1. No se convierte en tabla 🡪 Atributo nulo
2. Se convierte en tabla 🡪 Sin atributo nulo
   * Relaciones Ternarias:
     + En general son relaciones muchos a muchos.
     + Cada entidad genera tabla, al igual que la relación.
   * Relaciones uno a uno:
     + Una tupla de una tabla está relacionada con otra tupla de otra tabla y viceversa (dependen de la otra entidad).
     + Los atributos de la entidad que no se convierte en tabla (1,1) deben asociarse a la tabla (0,1) aceptando valores nulos.
     + Es el único caso donde no se genera una tabla a partir de una entidad.

Integridad referencial:

Ejemplo:

Alumnos = (ID\_ALUMNO, nombre y apellido, dni, nro alumno).

Materias = (ID\_MATERIA, nombre, añocurso).

Alumnos\_Materias = (ID\_ALUMNOMATERIA, ID\_ALUMNO, ID\_MATERIA, resultadoobtenido, añoquesecursa) 🡪 ID\_ALUMNO, ID\_MATERIA SON CLAVES FORÁNEAS. Las claves foráneas suelen ser claves secundarias.

* Propiedad deseable de las BD.
* Asegura que un valor que aparece para un atributo en una tabla, aparezca además en otra tabla.
* La misma significa que la clave externa de una tabla de referencia siempre debe aludir a una fila válida de la tabla a la que se haga referencia. La integridad referencial garantiza que la relación entre dos tablas permanezca sincronizada durante las operaciones de actualización y eliminación.
* Se encarga de definir restricciones a lo que se puede y no se puede hacer.
* 4 tipos de IR:
  + Restringir operación.
  + Realizar operación en cascada.
  + Clave foránea en nulo.
  + No hacer nada.

Borrado en la integridad referencial (modificación en ANEXO):

* Bajo los tipos de IR (son excluyentes una de otra, solo se pueden optar por una sola):
  + Restringir operación: si intento borrar una instancia de una entidad, y se prueba que dicha instancia corresponde por relación a datos existentes en otra entidad, entonces NO puedo realizar el borrado. Ejemplo: Borro la materia DBD a la cuál están inscriptos alumnos, no se puede realizar porque DBD está asociado a datos existentes en otra entidad.
  + Operación en cascada: no se restringe la operación de borrado (es una operación delicada 🡪 pero mantiene la integridad referencial). Imagina que borras la materia DBD a la cuál están inscriptos muchos alumnos. Entonces borras DBD y en cascada borras las inscripciones a la materia. Quitar algo y si éste tiene algo relacionado, ese algo se vá con él.
  + Establecer clave foránea en nulo: consiste en aceptar el borrado pero establecer los valores de clave foránea en nulo. Si borras DDB, todas las referencias relacionadas a DBD con clave foránea se ponen en nulo.
  + No hacer nada: permite operar como se desee, esto rompe la integridad referencial. NO ACTION
* *Anexo*: Si yo mantengo ID autoincremental 🡪 no puedo modificar nada, por ende no es necesario definir nada

\***Nota**: NULO solo se compara con NULO

Restricciones:

* De dominio:
  + Especifican que el valor de cada atributo A debe ser un valor atómico del dominio de A. Ósea qué si mi atributo A es un tipo integer, me indica que sólo puedo ingresarle integers.
* De clave
  + Evita que el valor del atributo clave genere valores repetidos. Si yo defino un atributo como clave primaria o candidata, éste no puede tener valores repetidos.
* Sobre nulos:
  + Evita que un atributo tome nulo en caso de no ingresarle valor.
* De integridad:
  + Ningún valor de la clave primaria puede ser nulo.
* De integridad referencial:
  + Se especifica entre dos relaciones (tablas) y sirve para mantener la consistencia entre tuplas de las dos relaciones (tablas).
  + Establece que una tupla en una relación (tabla) que haga referencia a otra relación (tabla) deberá referirse a una tupla existente en esa relación.
  + Clave foránea está representada por un atributo de una relación que en otra es clave primaria.
* Las operaciones de Alta, Baja y Modificación (ABM) pueden generar violaciones a las restricciones anteriores.
  + Alta:
    - Puede violar: Valor nulo para clave, repetición de la clave, integridad referencial, restricciones de dominio.
    - Si se viola la regla, la operación se rechaza.
  + Baja:
    - Puede violar: Integridad referencial (se procede como en el caso anterior).
  + Modificación:
    - Puede violar: Cualquiera de las operaciones.