Modelo de Entidad-Relación

Índice

La representación de los datos del usuario es una de las más importantes tares en el desarrollo de una aplicación.

El modelo de E-R fue introducido en 1976 por Peter Chen; y es utilizado como base de muchas herramientas CASE y otros (**IEF, IEW, DEFT, VISIS, SALSA, GENEXUS**)

El modelo de Objeto semántico fue introducido en 1988; y también tienen una cantidad considerable de seguidores.

Estos modelos incluyen lenguajes para expresar the user data model (el modelo de datos del usuario).

El modelado de los datos es usado para expresar un diseño de datos lógicos como pseudocódigo, o diagramas de flujo que es usado para expresar un diseño lógico de programa.

Los símbolos usados para expresar modelos de E-R difieren considerablemente.

- **Elementos del modelo E-R:**

Estas son **entidades (entities), atributos (attributes), identificadores (identifiers), relaciones (relationships).**

- **Entidades:** es algo que puede ser identificado en el entorno de trabajo del usuario, algo que el usuario quiere seguir, la pista. Las entidades de un tipo dado se agrupan en **clases de entidades**. De este modo, la clase de entidad EMPLEADO es el conjunto de todas las entidades empleado.

Ej: de entidad:	EMPLEADO	Mary Doc
	Clase de entidad empleado.	Entidad

Diferencia entre clase de entidad e instancia de entidad:

- Clase de entidad: es la colección de entidades y se define por la estructura o formato de cada entidad de clase.
- Una instancia de una clase de entidad: es la representación de una entidad particular por ejemplo: CLIENTE 12345. Está definida por el valor del atributo de la entidad.
- Atributos: también llamadas propiedades que describen Fecha de Alquiler, CódigoEmpleado. El modelo de E-R asume todas las instancias de una clase de entidad que tienen los mismos atributos.

En el modelo original de E-R los atributos podían ser single (simples) o multiple (compuestos). Por ejemplo, en CLIENTE, el atributo compuesto Dirección, podría ser definido por el grupo (Calle, Ciudad, Provincia, CP).

La mayoría de las implementaciones del modelo de E-R no permiten multivalued (multivalores) o composite atributes (atributos compuestos).

- Identificadores**:** las instancias de entidades tienen identificadores que son atributos que cambian o identifican instancias de entidad. Por ejemplo, la instancia EMPLEADO puede ser identificada como o por: NúmeroSeguroSocial, EmpleadoNumero o EmpleadoNombre.

No se podría identificar por Salario o FechadeContratación.

El identificador puede ser uno o más atributos entonces puede ser único o no único (Unique o nonunique).

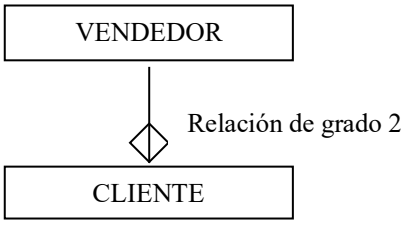
Los identificadores que se componen de dos o más atributos se llaman identificadores compuestos. Por ejemplo (CodigoArea, NumeroLocal), (NombreProyecto, NombreTarea).

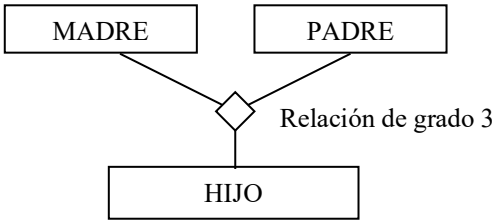
- Relaciones**:** las entidades se asocian más con otras con relaciones. El modelo E-R contiene ambas relaciones: relaciones de clases y relaciones de instancias.

Las de clases están asociadas entre clases de entidades y las instancias entre instancias de entidades. Las relaciones pueden tener atributos.

Las relaciones pueden incluir muchas entidades, el número de entidades en la relación es el **grado de la relación**.

Ejemplo:





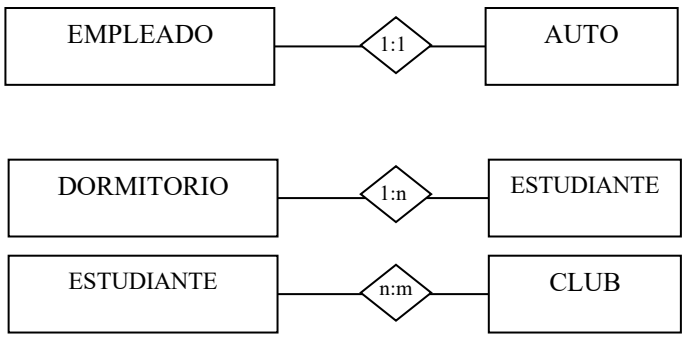
Comúnmente, las relaciones de grado dos son conocidas como binary relatioships (relaciones binarias).

Existen tres tipos de relaciones binarias:

(1:1) "one to one" relationship: una instancia de una clase de entidad, se relaciona con una instancia de otra clase de entidad.

(1:N) "one to N" relationships: una instancia de una clase de entidad, se relaciona con varias instancias de otra clase de entidad.

(N:M) "N to M" many to many: relaciona instancias de un tipo con instancias de otro tipo.

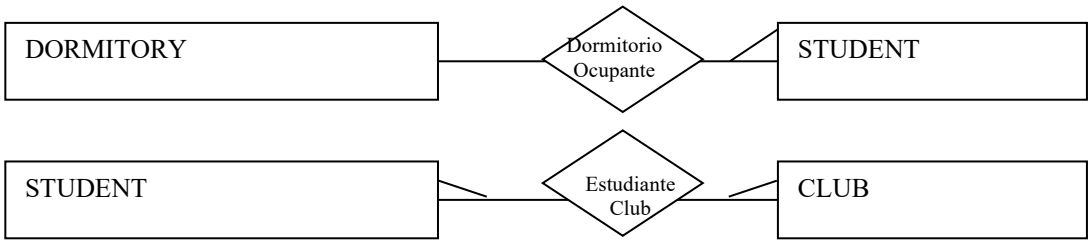


El número del rombo muestra la máxima cantidad de entidades que puede haber de cada lado de la relación. Esta restricción se llama **cardinalidad máxima** de la relación. En el ejemplo 2 la máxima cardinalidad de la relación es de 1:N. Por ejemplo la relación entre EQUIPOBASKETBALL y JUGADORES es de 1:5 que significa que cada juego de BASKETBALL debe tener a lo sumo cinco jugadores.

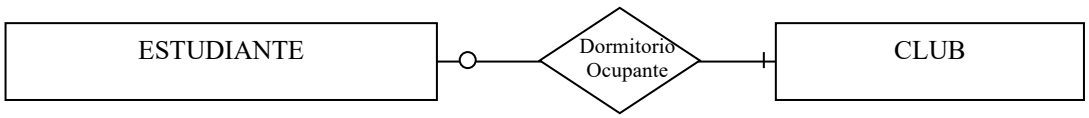
Las relaciones del ejemplo son llamadas también relaciones ***Has-A***. Se debe a que una entidad tiene (has) una relación con otra entidad de otro tipo. Por ejemplo: un EMPLEADO tiene un AUTO, un ESTUDIANTE tiene un DORMITORIO, un CLUB tiene ESTUDIANTES.

Diagramas de Entidad-Relación:

Los dibujos anteriores se llaman Diagramas de entidad-relación. Los nombres y símbolos de los diagramas difieren en los estándares de los vendedores. Por ejemplo: muchas veces el nombre de la relación se escribe sobre el rombo y la máxima cardinalidad de la relación se presenta con líneas. Ejemplo:



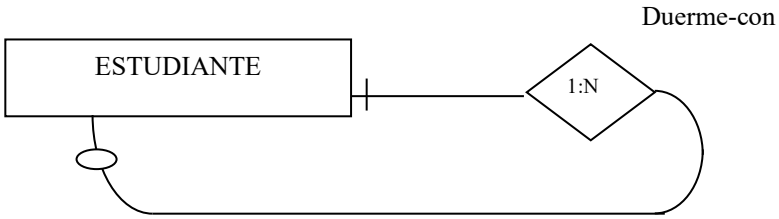
La mínima cardinalidad de la relación es la mínima cantidad de entidades que pueden formar parte de la relación va de 0 (cero) a muchas.



Un dormitorio debe tener una relación con al menos un estudiante pero un estudiante no requiere tener una relación con un dormitorio, entonces las restricciones de la relación son: un DORMITORIO tiene una mínima cardinalidad de 1 y una máxima cardinalidad de muchas

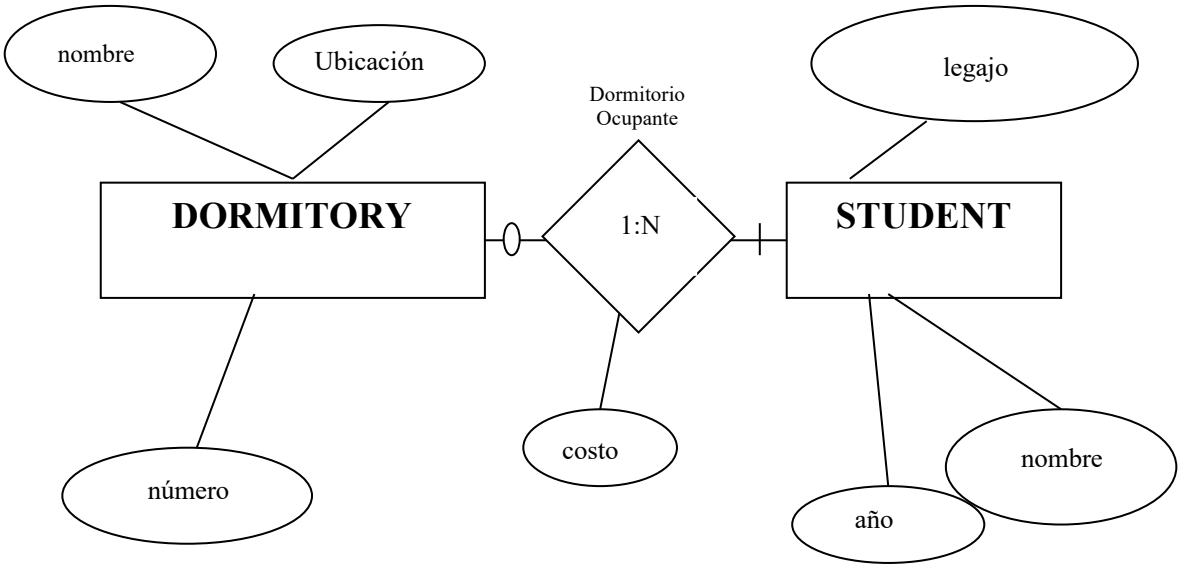
entidades ESTUDIANTE. Un ESTUDIANTE tiene una mínima de 0 y una máxima de 1 una entidad DORMITORIO.

Una relación puede existir entre entidades de la misma clase. Estas relaciones se conocen como **relaciones recursivas**. Por ejemplo DUERME-CON se puede definir en la entidad ESTUDIANTE.



Muestra de Atributos en los diagramas de E-R:

En algunas versiones de diagramas de E-R los atributos se muestran en elipses conectadas a la entidad o relación a la que pertenece.



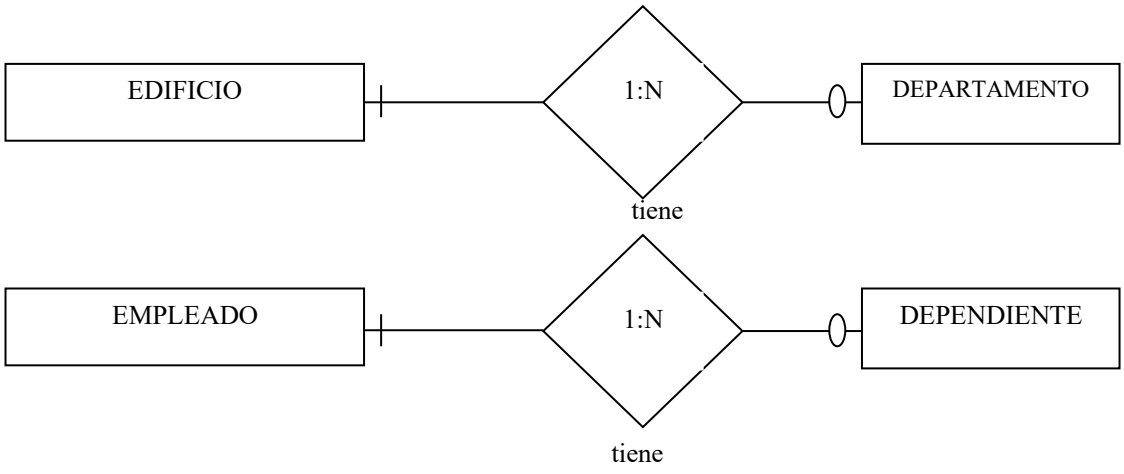
La relación Dormitorio Ocupante tiene un atributo *costo* que muestra la cantidad de renta pagada por un estudiante particular en un dormitorio particular. Si la entidad tiene muchos atributos el diagrama se hace difícil de interpretar. En estos casos los atributos se listan por separado. Muchas herramientas CASE muestran los atributos en ventanas pop-up.

- Entidades débiles (weak entities): El modelo de E-R define tipos específicos de entidades llamadas weak entity. Estas son las entidades que no pueden existir en al BD al menos que otro tipo de entidad también exista en la BD. Las entidades que no son weak se llaman **fuertes** (strong entity).

Veamos un ejemplo:

Consideremos una BD de Recursos Humanos con las clases de entidad EMPLEADO y DEPENDIENTE. Suponiendo que el negocio tiene una regla que una instancia EMPLEADO puede existir sin tener una relación a cualquier entidad DEPENDIENTE pero una entidad DEPENDIENTE no puede existir sin tener una relación a una entidad particular EMPLEADO. En este caso DEPENDIENTE es débil. Esto quiere decir que los datos de DEPENDIENTE sólo pueden ser almacenados en la BD si tiene una relación con una entidad EMPLEADO.

Diagrama:



El modelo de ER incluye tipos especiales de weak entities llamadas entidades **ID dependiente**. Estas son entidades en las cuales el identificador de una entidad incluye el identificador de otra entidad. Por ejemplo: consideremos las entidades EDIFICIO Y DEPARTAMENTO el identificador de EDIFICIO es EdificioName, y el identificador de APPARTEMNT es el identificador compuesto (EdificioNombre, Departamento).

Como el identificador de APPARMENT contiene el identificador EDIFICIO (EdificioNombre). Entonces DEPARTAMENTO es ID-dependent de EDIFICIO. Otra forma de ver la relación es lógicamente y físicamente un departamento no puede existir a menos que exista un EDIFICIO. Las entidades ID-dependent son muy comunes.

Desafortunadamente existe una ambigüedad la cual es conocida en la definición de weak entities y que es representada de formas diferentes por diferentes diseñadores de BD (y autores).

Esta es: En sentido estricto, si una weak entity se define como cualquier entidad cuya presencia en la BD depende de otra entidad, entonces cualquier entidad que participe en la relación teniendo un cardinal mínimo de uno a la segunda entidad es weak. Sin embargo esta interpretación no siempre es así.

La definición de weak entities más acotada: para ser una weak entity, una entidad debe depender lógicamente de otra entidad.

Definición de Weak Entity:

Para ser una entidad weak, una entidad debe depender lógicamente de otra entidad. Esto significa que algunos (pero no todos) objetos que tienen un cardinal mínimo uno en una relación con otra entidad son weak y también significa que todas las entidades ID-dependent son weak también. Cada Weak tiene un cardinal mínimo de uno en la entidad en la cual depende, pero cada entidad que tiene un cardinal mínimo de uno no necesariamente debe ser weak.

- Subtype Entities: (subtipo de entidades): Algunas entidades contienen conjuntos opcionales de atributos o subtipos. Por ejemplo la entidad CLIENTE con los atributos ClienteNumero, ClienteNombre y MontoAdeudado. Suponiendo que CLIENTE puede ser un individuo, una sociedad o una corporación y que los datos adicionales van a ser almacenados dependiendo del tipo. Datos adicionales:

INDIVIDUAL CLIENTE:

Adress, SocialSecurityNumber

PARTNESHIP-CLIENTE:

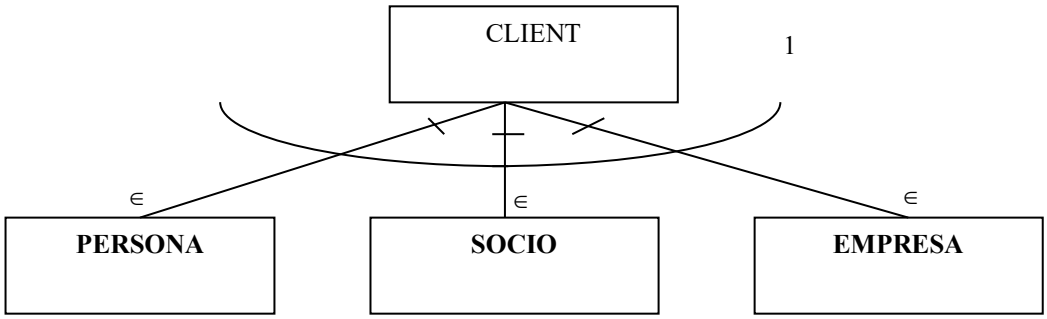
ManagingPartnerName, Adress, FaxIdentificationNumber.

CORPORATE-CLIENTE:

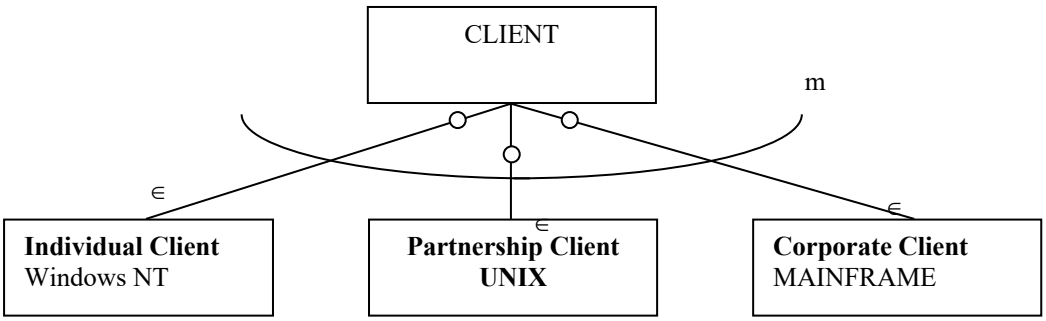
ContactPerson, Phone, FaxIdentificationNumber.

Una posibilidad es colocar todos los atributos en la entidad CLIENTE pero es incorrecto porque algunos atributos no son aplicables y quedarían sin valor.

El modelo más cercano define tres subtipos de entidades que son Subtypes de CLIENTE y CLIENTE es supertype de cada una de las entidades subtipo por turnos.



Los subtipos no son siempre mutuamente excluyentes, aunque no son siempre requeridos.



Las estructuras de supertipos y subtipos son llamadas a veces jerarquía de Generalización porque CLIENTE es una generalización de tres subtipos. También este tipo de relaciones se llaman a veces relaciones **IS-A** como en las figuras anteriores Individual-Cliente es un CLIENTE.

Un modo de verificar que el término IS-A es apropiado, es considerar el identificador de las cuatro entidades. Todas deben tener el mismo identificador, ClienteNumber, y por lo tanto todas tienen el mismo nombre y por lo tanto se refieren a la misma entidad.

Generalization hierachies (jerarquías de generalización), tienen una característica especial llamada inheritance (herencia), que significa que las entidades de los subtipos tienen atributos de la clase entidad supertipo.

- Documentación de las Business Rules:

El capítulo dos define un esquema de BD que consiste en tablas, relaciones, dominios y business rules. Las tres primeras se pueden obtener del modelo de ER pero no las Business rules.

Estas son agregadas a veces al modelo de E-R durante la etapa de modelado de datos.

El modelo de ER desarrollado desde un análisis de los requerimientos obtenidos del usuario. Durante el análisis las business rules son limitadas y en realidad el analista debe hacer un pausa para preguntar por ellas.

Por ejemplo: en el ejemplo del ingeniero y el camión (Engineer y Track) existen reglas con respecto a quien es asignado a un camión. Si no hay suficientes camiones, ¿debe seguir reglas para determinar a quien se asigna el camión?.

Puede ser que la aplicación asigne camiones basándose en cual ingeniero tiene mayor número de Servicios agregados en un período de tiempo, el mayor número de servicios fuera de la oficina, o cualquier otra regla.

Lo mismo puede suceder con la certificación, un servicio se debería asignar a un ingeniero que tenga una determinada certificación.

Las Business rules, pueden o no ser forzadas por el DBMS o por los programas de aplicación. A veces son escritas en los manuales de procedimientos que el usuario de la aplicación de BD debe seguir. Por lo tanto, el modo en que se deben forzar la Business rules no es importante, lo importante es documentarlas para que se conviertan en parte de los requerimientos del sistema.

- El modelo de Entidad-Relación y las Herramientas CASE:

Desarrollar un modelo de datos usando el modelo ER se ha vuelto fácil en los últimos años, porque las herramientas para construir diagramas de ER se incluyen en la mayoría de los productos populares CASE. Productos como IEW, IEF, DEFT, ER-WIN y Visio tienen diagramas de ER. Estos productos también integran entidades con las relaciones de BD que representan, lo cual puede facilitar la administración, manejo (gerenciamiento) y mantenimiento de la BD.

Los diagramas de ER creados usando Herramientas CASE son generalmente más agradables visualmente y más fáciles de cambiar y adaptar.

Una vez que el modelo de ER es desarrollado, la exactitud y congruencia con respecto a los requerimientos deben ser verificados, usualmente con el usuario.

Una técnica para evaluar un modelo ER en el contexto de posibles "consultas" que puedan plantearse con la estructura implicada por el modelo

¿Qué preguntas pueden ser respondidas desde una BD que se implementa de acuerdo con este diseño?

Base de datos como modelo de modelos:

Existen distintas formas de modelar la situación de una empresa. Una BD no modela el mundo real. Las BD son modelo del modelo del mundo del usuario. La pregunta que él debe responder es si "¿este diseño representa precisamente el modelo del usuario de su entorno?". El objetivo es desarrollar un diseño que se ajuste a la concepción mental del usuario.

Immanuel Kant y otros filósofos argumentan que es imposible para los humanos construir modelos de lo que existe actualmente.

Los sistemas de computación no necesitan modelar, otra cosa que el sistema de tokens y comunicaciones.

Se deben responder estas preguntas para evaluar un modelo ER:

¿Este modelo refleja precisamente la percepción del usuario y modelo mental de su mundo?

¿Ayudará al usuario, responderá consistente y exitosamente unas con otras y con sus clientes?

No se debe discutir si un modelo representa mejor la realidad, el punto es desarrollar un modelo que represente bien el modelo del usuario de su entorno de negocios.

Proceso de diseño de base de datos

- Introducción**:** el diseño BD es el proceso de desarrollar una estructura de BD desde los requerimientos del Cliente. Este proceso involucra varios pasos: análisis de requerimientos, diseño general y detallado; implementación; testeo, operación, revisión y retiro.

- The Database Design Process (Proceso de Diseño de BD): vamos a considerar el diseño de BD en detalle. Se debe elegir un diseño de BD deliberadamente (a propósito). Una buena

metodología debería ayudar a producir estructuras de BD útiles, llegar a los objetos del cliente dentro de una cantidad de tiempo razonable y con un esfuerzo razonable.

La estructura se deberá ajustar a las restricciones del sistema, ser adaptable a necesidades de proceso futuras y ser entendida por los desarrolladores y clientes. La metodología debe tener suficiente generalidad y flexibilidad para ser usada por personas con diferentes niveles de experiencia en diseño.

Debería ser adaptable a diferentes modelos de datos y diferentes software DBMS. No existe una técnica que cumpla todas estos objetivos simultáneamente, pero ofrecen un marco de trabajo para seleccionar.

Cualquier metodología de diseño necesita varios componentes para satisfacer estas metas. Para proyectos largos recomiendo una revisión formal al final de cada paso.

Las técnicas de diseño desempeñan los pasos requeridos en el proceso de diseño y un criterio de evaluación. Se puede usar para seleccionar una alternativa en cada paso (es mejor si este componente es automático – el resultado es una porción de sistemas CASE).

Los requerimientos de información proveen entradas al proceso de diseño a todo y cada uno de los pasos. Un mecanismo descriptivo representa la entrada de información y el resultado de cada paso. Esto también debería ser automático y visual (gráfico).

Existen muchos aspectos para evaluar un diseño. Algunos son cuantificables y otros no. La siguiente lista presenta algunos de estos (pero no todos). La importancia relativa variará de una aplicación a otra.

- Quantitative evaluation criteria (criterio de evaluación cuantitativo): son esos criterios que pueden ser medidos. Incluyen cantidades como tiempo de respuesta en una consulta, costo de actualización de almacenamiento, de carga inicial, y de reorganización y tiempo promedio de operaciones por lote.

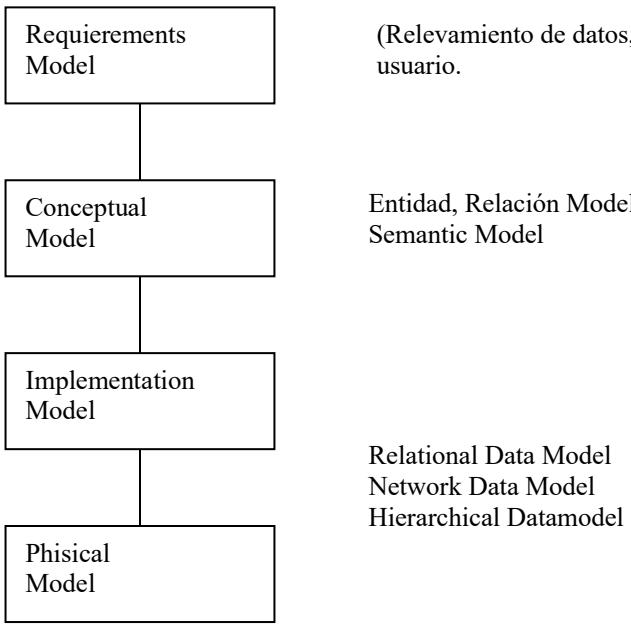
- Qualitative evaluation criteria (criterio de evaluación de calidad): Son aquellos criterios que no son fácilmente medibles. Pueden incluir calidad como flexibilidad, adaptabilidad, comprensibilidad, y portabilidad y deberían incluir criterios clave como recuperación y seguridad.

Enfatizar los criterios cuantitativos es frecuentemente un error (la tendencia se debe a que son números). Falta de provisiones adecuadas de seguridad puede resultar en la liberación de información confidencial acerca de clientes y pleitos.

Los requerimientos de información impregnan todo el proceso de diseño. Se puede clarificar en frases de requerimiento en la perspectiva y lenguaje del cliente (usuario), y la traducción de ese lenguaje en un lenguaje relevante al diseño de sistemas. Es importante reconocer que tanto el cliente como el diseñador deben familiarizarse con los dos lenguajes y que el diseñador es responsable por el proceso de traducción. La información del cliente se usa para definir los requerimientos de procesamiento de la organización en terminología organizacional.

La información de diseño es una formalización de estas entidades, atributos y relaciones que puedan ser analizadas precisamente.

Diseñar una BD, como la mayoría de los software de los procesos de diseño, equivale a construir una serie de modelos cada uno más detallados de la aplicación. Esto se realiza frecuentemente de manera informal, en algunas áreas de la aplicación. La experiencia indica que debería hacerse explícitamente en las aplicaciones de BD.



DB/2. Inpres, Oracle, IDMS

Especificación de requerimientos de BD:

Los primeros problemas que se debe encarar durante el análisis de requerimientos no son técnicos. Uno de estos involucra conflictos de política y organizacionales entre los potenciales clientes y pueden ser no resueltos.

Otro problema es la comunicación entre los usuarios potenciales (clientes, programas) y entre el usuario y el equipo de diseño de BD. En teoría, las dificultades no son diferentes de

cualquier otra tarea de análisis de requerimientos de la aplicación en la práctica, la cantidad de detalles involucrados y el número de puntos de vista que deben ser tomados en cuenta hacen que la comunicación sea el aspecto más importante de un análisis exitoso de una aplicación de BD.

Es importante por lo tanto un plan metodológico de organización. En el siguiente párrafo destaco un acercamiento que funciona para mí.

Definir el alcance de la BD (define the scape of the DB):

Incluye necesidades presentes y anticipadas en el futuro. Tener en cuenta que el futuro crece en forma exponencial, como resultado, la gente generalmente sobreestima las necesidades de crecimiento de rango corto y subestima necesidades de rango largo.

Una vez completada esta actividad debería resultar en una descripción escrita del alcance del proyecto. El propósito de este documento es identificar explícitamente qué porciones de la organización se incluyen dentro del alcance y cuáles se excluyen. Este último es al menos tan importante como la forma. El documento debería incluir varias actividades comerciales y varias subunidades organizacionales que tengan lugar dentro de la organización.

Coleccionar información acerca del uso de los datos (collect información about data usage):

El tipo de información que se recolecta acerca del uso de los datos de la organización generalmente cae en una de las siguientes dos categorías:

1. Funciones operacionales del negocio.
2. Funciones de control y planeamiento del negocio.

Usted debería identificar cada función operacional que se incluya en la BD a que toque a la BD de algún modo.

Por cada función identificar los datos usados por esa función, cómo esos datos son transformados por la función y las reglas implícitas y explícitas cómo y por qué y cuándo ocurre cada función.

- Procesos: que transforman los datos usados por la organización y
- Operaciones: que serán requeridas para afectar los productos.

Por cada proceso llevado a cabo por el usuario, se requiere la siguiente información: nombre del proceso, frecuencia de ocurrencia, probabilidad de ocurrencia, prioridad, datos requeridos para llevar a cabo el proceso, el volumen de datos que involucrará. Por cada operación puede ser útil la siguiente información: nombre de la operación (find, insert,

update...), criterio de búsqueda, número de instancias recuperadas (dar el promedio, mínimo y máximo número que será involucrado en la operación), probabilidad de ocurrencia y probabilidad de instancias recuperadas siendo usadas en más acciones.

La información de las funciones de control y planeamiento del negocio requieren un entendimiento más profundo de las políticas de operación explícitas e implícitas de la organización, una definición de estas funciones y los datos que requieren y una idea de los cambios del negocio que puedan ocurrir en el futuro. Aunque muchos diseñadores son partidarios de llevar a cabo este paso en término de objeto, yo encuentro que los clientes tienden a pensar en términos de procesos, por lo tanto recomiendo que coleccione la información como se describe aquí y traduzca el resultado en términos de objeto como parte del diseño conceptual, descrito más adelante.

Piense y planee el proceso de coleccionar los registros.

El involucramiento del cliente es crítico para todo el procedimiento, la primera técnica es entrevistar y hablar con cada uno desde el principio hasta el final de la organización desde arriba hacia abajo. Cuando comience a quedarse sin tiempo concéntrese en la gente que tiene contacto directo con la BD (esta es la gente que querrá o romperá su producto). Documente las entrevistas. Lo mejor es grabarlas, transcribirlas y hacer un resumen.

Vuelva por una respuesta para clarificar cualquier cosa oscura. Escribir la entrevista no es frecuentemente factible y es tan desbaratado que debería evitarse, mucha gente se paraliza cuando saben que van a ser grabados.

La segunda técnica de documentación precisa, es tener un colega que lo acompañe a la entrevista y que escriba todo lo importante. Mucha gente prefiere este método a ser grabada.

Nunca le pregunte a un usuario potencial ¿Le gustaría tal o cual característica? La respuesta es siempre “Sí”. Pregunte: ¿Cuánto esta dispuesto a pagar por tal o cual característica?. Es mucho más probable obtener una respuesta útil.

Mucha de la información de operación, control y planeamiento es difícil de obtener. Un error frecuente es tratar de rellenar todos los detalles para cada pedazo que encuentre. Si hace esto puede terminar atascado en el segundo pedazo y no terminar nunca el proyecto.

Traducir la formación (translate the información):

El cuerpo de la información desestructurada recolectada durante los dos primeros pasos se debe transformar en un formulario que pueda ser usado para análisis. Por muy pequeño que sea el proyecto (menos de 200 elementos de datos) puede ser hecho a mano. Por otro lado herramientas automatizadas igual pueden ser necesarias. Existen muchas desde simples editores de texto hasta herramientas CASE de alta escala. Si ya se ha seleccionado un DBMS se puede usar esta para la traducción.

Una vez compilada una lista de todos los elementos de datos identificados durante los dos primeros pasos. El resultado es un diccionario de datos de los elementos de datos. Las metas para esta actividad serán identificar y eliminar elementos de datos redundantes y resolver la inconsistencia de definiciones. La información en el diccionario de datos incluye: nombre del cliente de dato, definición, estructura (si el elemento es compuesto) y dónde se usa.

Luego se definen las tareas operacionales, sus características y el uso de los datos. Una buena herramienta es el DFD Diagrama de flujo de datos (Data Flow Diagram) o (Buble Chart). Este diagrama consiste en burbujas que representan procesos que transforman los datos y líneas que conectan las burbujas que representan el movimiento de datos de un proceso a otro.

Normalmente los DFDs resultantes de diferentes perspectivas del usuario de las actividades organizacionales difieren considerablemente. El resultado de esta paso será un conjunto de diagramas que representan una unión de los puntos de vista y una expansión del diccionario de datos para incluir información acerca de los procesos. Por cada proceso deseará incluir: nombre del proceso, definición de la tarea llevada a cabo, tipo (operacional, control, planeamiento), frecuencia de ocurrencia, área funcional del negocio que lleva a cabo el proceso, elementos de datos que entran en el proceso, elementos de datos que salen del proceso, transformaciones que ocurren dentro del proceso y el tiempo de vida de los elementos de dato.

Ahora, definir el control y tareas de planeamiento, sus características y su uso de los datos. Estas tareas están peor definidas que las tareas operacionales y cambian más frecuentemente. Al mismo tiempo involucran personal de alto nivel dentro de la empresa y son extremadamente visibles. Agregue cada tarea que se identifica con el diccionario de datos usando la información de arriba.

Desarrolle una lista de todas las reglas de operación (implícitas y explícitas) y procedimientos. El proceso, el cual es bastante similar al análisis anterior resultará en una lista que contendrá enunciados de políticas. Estas también se pueden introducir en el diccionario de datos (DD).

Descomponga cada uno en enunciados simples (evitando el uso de “and o or”). A veces se debe imponer una estructura de herencia en las políticas. Estas declaraciones (enunciados), proveerán restricciones en el diseño de la BD.

Desarrolle una lista de los cambios futuros potenciales y la forma en que puedan afectar las operaciones. Otra vez los enunciados deben ser simples y se pueden ser introducir en el DD. Como estos enunciados proveen indicaciones de la flexibilidad que se desea en la implementación de la BD, afectarán el diseño.

Finalmente desarrolle una lista de las consultas potenciales que se deben satisfacer. Esto proveerá cierto control en el diseño.

El resultado del análisis de requerimientos debe ser un documento escrito, el SRS (The software requerimets specifications) especificaciones de requerimientos de software. Existen muchos modos de hacer esto.

Conceptual (logical) Database Design (Diseño lógico de BD):

El diseño conceptual trata con información independiente de cualquier implementación actual. El objetivo, es representar información en un formulario que sea comprensible al usuario, independiente de las especificaciones de sistemas, pero capaz de ser transformado en un sistema de BD implementable.

Al resultado del diseño conceptual se lo llama Modelo de la solución del problema del usuario y consistirá en el DD, un conjunto de diagramas mostrando entidades y sus relaciones (llamado modelo de información). Un conjunto de diagramas mostrando la manera en que estas entidades son manipuladas (incluyen DFD, diagramas de estado y otro tipo de diag.), diagramas de control y tablas que describan las circunstancias bajo las cuales toman lugar procesos variados y cualquier descripción textual que sea necesaria. Esta última debería ser pequeña y consistir de instrucciones en entendimiento de las partes. Nótese que cada elemento de diseño debería reflejar uno o más requerimientos.

Una variedad de información debería ser agregada al DD. Esta es la información de diseño nombrada anteriormente. Esta puede ser útilmente categorizada en cinco partes: entidades, atributos, relaciones, restricciones y programas. Primero listaré la información necesaria para cada parte y luego discutiremos las cinco partes en detalle.

- Para cada entidad: registre la siguiente información: nombre, descripción, (strong o weak); clase (tangible, role, evento, interacción, especificación, otra) y cardinalidad.
- Por cada atributo: nombre, descripción, clase (nombre, descriptivo, referencial, derivado); dominio, restricciones de dominio, entidades a la que pertenece el atributo; estructura (si el atributo es compuesto); atributos y factores de repetición.
- Para cada relación: nombre, descripción, grado (binary, recursive, ternary...); entidades y/o relaciones involucradas; cardinalidad (1:1, 1:n, m:n); opcionalidad (obligatoria, condicional), clase (subset, generalization, agregation, association, otra) y atributos.
- Por restricción: definida en entidades, atributos o relaciones, registrar: Nombre, descripción, trigger condition (condición de disparo), definición de restricción y la acción requerida si se viola la restricción.
- Por cada programa: nombre, descripción funcional, información usada por el programa (input data), información producida por el programa (output data) y condición de disparo (trigger condition), que determina cuando el programa debe correr. No olvidar que estos programas pueden ser escritos en un lenguaje proveído por el vendedor de BD o por un lenguaje de programa de propósito general (pseudo-código).

Como mucho del diseño conceptual es acerca de los programas y de los procesos del usuario involucrados, cae fuera del alcance de este curso. Aquí nos centraremos en el modelo de información. Las cinco decisiones del diseño básico en diseño conceptual son concernientes a la representación de la información del usuario en información de diseño: selección de

entidades, selección de atributos de entidades, selección del atributo clave de una entidad, selección de relaciones entre entidades y selección de restricciones de integridad.

Esto desemboca en el modelo, frecuentemente llamado modelo de Entidad-Relación (ER), del problema de aplicación.

Un modelo alternativo es el Semantic Object Model. Aunque la siguiente discusión trata los cinco pasos separadamente en realidad, se necesita una gran cantidad de interacción. Por ejemplo: frecuentemente en el comienzo es poco claro si un concepto particular debe ser modelado como una entidad, un atributo o una relación. Sólo con el progreso del diseño se puede tomar la decisión y muchas veces cambian varias veces a lo largo del camino. Esto es saludable, flexible y creativo.

Conceptual DB Design: Entidades

♦ Selecting Entities (seleccionar entidades)

El libro de Shlaer y Mellar define una entidad como una abstracción de un conjunto de cosas del mundo real:

1) Que comparte características comunes (sólo las características en las que está interesado).
2) Que obedecen al mismo conjunto de reglas.

Las entidades se llaman algunas veces objetos. Las características se especifican por los atributos. La parte q de la definición dice que todas las ocurrencias (o instancias) de una entidad tienen los mismos atributos.

Esto puede parecer circular, pero no lo es. Un atributo es una función que va desde el conjunto representante de la entidad al dominio. La parte uno sólo dice que todos los atributos son funciones totales.

La segunda parte de la definición trata con operaciones que pueden ser realizadas en instancias de entidades.

Aquí una "operación" es básicamente simple: insertar una instancia en el conjunto de entidades, borrar una, modificar el valor de un atributo, o seleccionar un subconjunto basándose en una restricción. La parte dos dice que tales operaciones se definen en todas las instancias de la entidad. En este contexto, in "select" se define en una instancia de una entidad si es posible determinar si la instancia satisface la condición de selección. En esencia, esto significa que la condición de selección contiene sólo atributos definidos para cada instancia de la entidad.

Es útil interpretar "rule" (regla) como algo un poco más amplio en esta definición. Add: laws, políticas organizacionales, leyes de física, uso acostumbrado y cosas por el estilo.

Diagramamos las entidades como cuadrados boxes con un nombre descriptivo.

- Identifyieng Entities (identificar entidades):

De acuerdo con Shlaer y Mellar, la mayoría de las entidades caen dentro de una de las siguientes clases: cosas tangibles, roles, incidentes, interacciones entre otras entidades y especificaciones. Aunque hay unas pocas entidades que no encajan dentro de estas clases, encuentro bastante útil esta división en la práctica.

1. Entidad tangible: son objetos actuales en el mundo que se pueden tocar. Por ejemplo: automóviles, cables, animales, estrellas y gente.
2. Incidente o evento: son sucesos que ocurren en un tiempo específico y usualmente en el lugar específico. Cómo se especifica el tiempo y el lugar, se determina por el problema; se trata el incidente como una entidad unitaria en la BD, esto es suficiente. Los verbos se requieren usualmente para describir incidentes. Por ejemplo: un accidente automovilístico, el vuelo de un avión, el desarrollo de un juego o concierto, una elección, la toma de un censo, el archivo de un reclamo de seguro, una quiebra de un sistema operativo, un servicio de llamada de reparación de un refrigerador.
3. Interacción: generalmente tienen sabor a "transaction" y que pueden ser difíciles de distinguir de una relación. De hecho, me siento incómodo con esta categoría, muchas veces. Por ejemplo: la compra (de una casa por un comprador a un vendedor), el acto de casarse (con el empleado del estado registrándolo), una red de computadora, el sistema de trayectoria de una vía férrea.
4. Especificación: frecuentemente mostradas en inventarios o aplicaciones de manufactura. Tiene la implicación de un standard o una definición. Un ejemplo puede ser: la especificación de diferentes modelos de un automóvil: una entidad con los atributos model#, númeropuertas, tasa de potencia. Las instancias de esta entidad son las descripciones de varios modelos. Generalmente si una entidad especificación existe, también es necesario tener otra entidad que represente instancias que cosas que junten las especificaciones: automóviles, en particular (con atributos como número de serie, modelo, etc...).

5. Rol: roles desempeñados por personas u organizaciones, incluyen títulos de trabajos (doctor, abogado); interacciones con otra gente (paciente, cliente, demandante); partes de una organización (departamento, sección); oficinas políticas (senador, presidente, etc).

♦ Nombrar entidades:

Se debe poner mucho cuidado cuando se nombra diversas entidades, los nombres deber ser cortos, claros y directos. A veces es difícil de hacer, pero reduce las confusiones.

Algunos puntos a considerar cuando se nombra entidades:

- ✓ *Use nombres comunes cuando sea posible.*
- ✓ *Palabras fuertes o de todos los días con significaciones extendidas se prefieren a las vagas o muy técnicas o términos inusuales.*
- ✓ *Use lenguaje paralelo para conceptos relacionados.*
- ✓ *Agregue adjetivos a nombres cortos y comunes para hacer más precisos los nombres.*
- ✓ *Use nombre (aunque sean largos), que obtengan el carácter esencial de la entidad.*
- ✓ *Nombre la entidad por la información que contiene, no por el formulario usado para llevar esa información.*
- ✓ *Evite palabras como: account, order, task, form, operation, proces, schedule, part, assembly, system (no tienen mucho significado)*
- ✓ *Si el nombre requiere la palabra "and" o "or" para ser preciso, entonces se tiene varias entidades no una.*

Cada entidad necesita una breve descripción en el DD. Esta puede ser una o más oraciones breves que le permitirán al lector determinar sin ambigüedad si una cosa particular del mundo real es o no una instancia de entidad.

La descripción sirve como regla de decisión. Aquí hay algunas guías.

1. Se puede describir alguna entidad enfatizando el modo en el cual las instancias son iguales. Esto es un criterio de inclusión. Por ejemplo todos los mamíferos son de sangre caliente, tienen pelo y son lactantes.
2. Se puede ser capaz de describir la entidad enfatizando el modo en el cual las instancias de una entidad difieren de las instancias de otra entidad. Por ejemplo: los murciélagos son los únicos mamíferos que vuelan.

3. A veces se puede explicar cómo la entidad se relaciona con otras entidades. Esto es particularmente útil para entidades de interaction y incident (eventos). Por ejemplo: un censo es tomado por cada estado; un avión es asignado a cada vuelo.
4. A veces la información de background ayuda por ejemplo, los pájaros son vertebrados que vuelan, sin embargo existen unos pocos tipos de pájaros que no les es posible el vuelo.
5. Finalmente, tenga en mente los estándares de escritura técnica aplicadas a describir entidades. Use sentencias cortas y declarativas escritas en tiempo presente.

♦ Testing entities (testear entidades):

No siempre es posible determinar si una entidad potencial es realmente una entidad (se debe llamar a esta entidad candidata). Sugerencias que pueden ayudar:

Piense tantas instancias diferentes de una entidad como pueda ¿todas tienen las mismas características? (atributos) ¿todas obedecen a las mismas reglas?. Un atributo posible para una persona es "nombre de soltera" que no es un concepto sensible o aplicable a un hombre. Esto debería decirnos que para esta aplicación tenemos dos entidades número uno.

Toda entidad debe tener atributos. Si la entidad candidata no puede ser descripta por atributos descártela. Probablemente es un atributo de otra entidad o puede ser una relación entre entidades.

Si el test de descripción anterior (el test de inclusión) requiere el uso de "or", ha confundido un grupo de entidades, descompóngalo.

Si el único modo en que se puede describir una entidad es listando todas y cada una de las instancias, probablemente no ha pensado lo suficiente. Si hay una entidad (probablemente no), usted será capaz de agregarle nuevo, el mundo cambia esté preparado.

♦ Tipo de entidades (type of entities)

Hay dos tipos de entidades, que son strong (fuerte) y weak (débil). Una entidad fuerte puede existir en el mundo real por si sola, independientemente de cualquier otra entidad. Una entidad débil no puede, una instancia de una entidad débil puede existir sólo si alguna instancia de alguna otra entidad ya existe (nosotros decimos que una weak entity debe tener un foreing key). Por ejemplo: a "juego" es una strong, a "desarrollo" es weak, como no puede ocurrir sin la existencia del juego siendo desarrollado; patient/diagnosis; edificio/hallway; estado/censo y otros ejemplos.

Nombres alternativos para esta idea pueden ser independiente y dependiente.

♦ Cardinality (cardinalidad):

Eventualmente será necesario implementar la BD realmente, a esa altura, se requerirá alguna idea del espacio de almacenamiento. Para una entidad el espacio de almacenamiento se relaciona con el número promedio de bytes requerido por cada instancia de la entidad, por el número de instancias, por el gasto por instancia. El primero y el tercero de estos no se

pueden determinar hasta mucho más avanzado el proceso de implementación. Sin embargo, el número de instancias es independiente del método de implementación, por lo tanto debe ser cubierto como parte del diseño conceptual. Además, este es el momento cuando el usuario está pensando en la entidad, por ello, es más probable tener una idea del número de instancias.

El resultado puede ser bastante simple. Por ejemplo: un cliente puede decir: "el número de instancias será siempre de 500, o más o menos 50). El número puede variar a través de los meses o los años, puede crecer o disminuir cada tanto (esperamos que el número de clientes crezca constantemente); puede tener una varianza grande o pequeña y así sucesivamente. Registre tanta información como pueda en el DD. Esté seguro de incluir factores de última hora, para explicar incertidumbres. No trate de ser muy específico (una aproximación preliminar es suficiente en este punto en el proceso de diseño).

Conceptual Database Design (Attributes)

- Selecting attributes (seleccionar atributos):

Un atributo describe una sola característica de una entidad o relación. Las atribuciones pueden identificar examinando las instancias de "cosas" que fueron abstractas dentro de las entidades. Para ver qué propiedades tienen de común esas "cosas". También puede mirar la descripción de una entidad para descubrir que necesitaría saber par decidir si algo es una instancia de una entidad o no. Finalmente, los tipos de consultas que los clientes intentan hacer a cerca de la entidad dan las claves acerca de qué clase de información debe existir en la BD.

Generalmente omito los atributos en los diagramas, éstos tienden a ser muy atestados aun sin los atributos. Si se incluyen, se pueden escribir cerca de las cajas.

Hay cuatro clases generales de atributos: naming attributes (nombre), descriptive attributes (descriptivos), referential (referenciales) y derived atrtributes (derivados).

1. Naming attributes (atributos Nombre): proveen hechos acerca de nombres de entidades. Los números de identificación de varias cosas, por ejemplo aveces, no siempre, los nombres sirven como claves o partes de claves.
2. Descriptives attributes (atributos Descriptivos): proveen hechos acerca de cada instancia de la entidad. Más de una instancia de la entidad puede tener el mismo valor para un atributo descriptivo.
3. Referential attributes (atributos Referenciales): se usan para atar una instancia de una entidad a diferentes instancias de la misma o de alguna otra entidad. Estos atributos actúan como foreign keys o porciones de foreign keys.
4. Derived attributes (atributos Derivados): son atributos simples que se calculan de otra información presente en la BD. Estos pueden ser tan simples como la edad, definida como la diferencia entre la fecha actual y la fecha de nacimiento de la persona. Para cálculos muy complejos involucra varios registros de la BD.

El conjunto de valores que se permite que un atributo tome se llama dominio.

En el DD, se debería describir el dominio para ser tan restrictivo como sea posible. Si los dominios empleados se definen con seis dígitos numéricos, entonces decimos que tienen el dominio en los "enteros". Algunos cuidados serán necesarios para permitir expansiones inciertas y futuras, pero un balance útil entre estos requerimientos se encuentra fácilmente. Algunas veces los dominios complejos se describen mejor en el DD proveyendo una referencia a alguna otra fuente. "La vista actual de clases de trabajo se puede encontrar en el manual de políticas corporativas, pág. 76".

Puede ser importante indicar si los valores nulos se permiten para un dominio. Los nulls tienen dos significados solamente: "I don´t know" y "value not yet assigned". Otro significado dado para BD es "no aplicable", no es posible aquí. Como punto de diseño, si se permiten nulas que expandan el dominio para incluirlas.

Algunos valores insisten en que los atributos del dominio sean de valor simple. Creo que esto es innecesario durante el diseño conceptual. Por otro lado, que un dominio compuesto se parezca es a veces la mejor separación. Los números de teléfono por ejemplo: consisten en el código de área más el número local. Partes de números frecuentemente llevan y traen información codificada. Romper estas en varias entidades es cosa de juicio, dependiendo de cómo se use el atributo. No es conveniente tener que considerar las partes si el atributo no se mira internamente. Por otro lado, si sabemos que será necesario imprimir reportes de todos los usuarios con residencia en un código de área particular, esto puede ser la clave para descomponer el número de teléfono en dos atributos.

Algunas atribuciones pueden tener múltiples ocurrencias por ejemplo, autores de libros.

Las restricciones de dominio se pueden usar para nuevas restricciones de valores permitidos. Como ya hemos notado que los dominios deben ser tan restringidos como sea posible ¿qué está quedando? Hay dos usos generales de las limitaciones de domino (y otras frecuentemente usadas). A veces el dominio consiste de todos los valores dentro de un conjunto fácilmente descripto excepto para unos pocos. Luego, es mucho más entendible si el dominio da las reglas generales y las restricciones dice que valores no se permiten. El uso más importante es proveer restricciones de valores debido a los valores de otros atributos, instancias y así sucesivamente es lo que llamamos restricciones de relación y restricciones de BD.

En unos pocos casos, un atributo en sí mismo puede tener atributos. El ejemplo más común es: atributos científicos o de ingeniería que describen propiedades físicas. Toda propiedad requiere un poco de información para una descripción completa. Esta descripción aplicada al atributo, no a la instancia de la entidad. Una propiedad física puede tener atributos como unidades físicas: precisión, duración del valor, fuente del valor de datos, clasificación de seguridad, fecha, hora y condición de medición. Ejemplos comunes son: temperatura, cuenta de neutrones de una explosión, voltaje y masa.

- Selecting keys (seleccionar claves):

Algunos autores insisten en que cada entidad tienen una clave, yo no encuentro útil este tipo de diseño conceptual. Esto es especialmente verdad si no hay claves naturales.

Si una entidad tiene una clave "natural", eventualmente vamos a encontrarnos a nosotros mismos usándola como una posible key para los registros de la BD que se use para implementar la entidad, por lo tanto tales natural keys deben ser identificadas en el DD. Como ya están ahí como atributos, tal vez la cosa más fácil para hacer es ¿proveer un campo adicional para cada descripción de atributo llamado "key"? Este campo es por defecto falso pero permite especificar verdadero.

Conceptual DB Design: Relationships

- Selecting relationships (seleccionar relaciones):

Las relaciones, por su naturaleza, pueden ser complejas en extremo. A cada relación se le debe dar un nombre en el DD, junto con una lista de entidades involucradas y una descripción del propósito de la relación. Muchas relaciones actúan como verbos transitivos (con las entidades sirviendo como sujetos, objetos y a veces objetos indirectos). El nombre por lo tanto puede aparecer más bien en verbo. Por ejemplo: una persona vive en una casa, una librería contiene libros, un candidato gana una elección, un avión viaja desde Oakland a San Francisco.

Las relaciones funcionales de los dos lados (o hacia todos lados, si más de dos relaciones están involucradas), por ejemplo: la casa es habitada por una persona, Oakland es el destino de muchos aviones desde muchos aeropuertos.

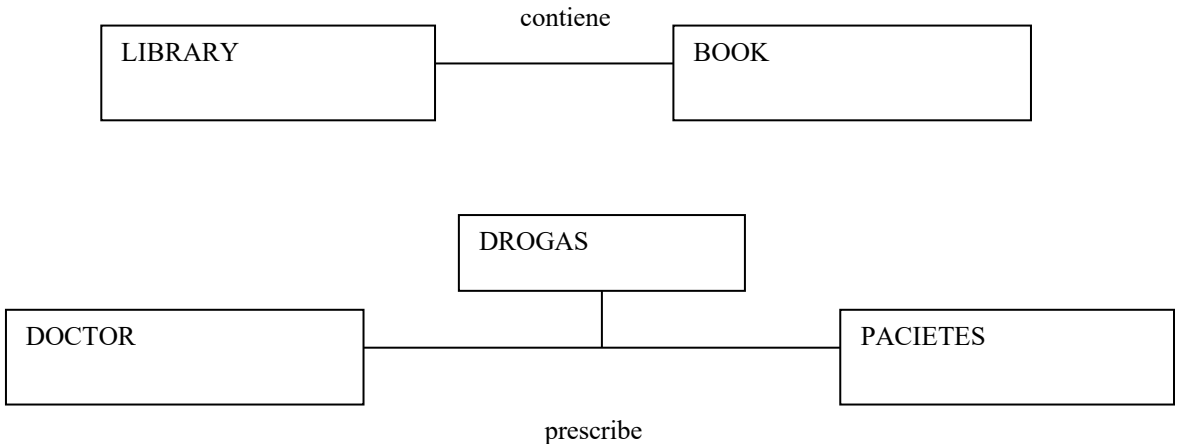
Las relaciones se presentan como líneas entre entidades. A veces se muestra como un rombo (diamond) o alguna forma similar, con líneas adosando el punto del rombo con la box de la entidad. Las líneas se "decoran" de diferentes formas para indicar características importantes de las relaciones.

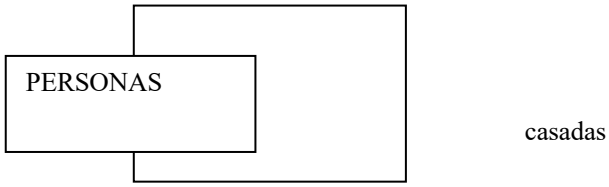
- Grado de una relación (degree of a relationship):

El grado de una relación indica cuantas entidades están involucradas. La mayoría son binarias -que significa que dos entidades están involucradas- ternarias (tres entidades), son menos comunes pero bastante interesantes cuando ocurren.

Otros tipos son extremadamente raros. Una relación binaria entre una entidad consigo misma se llama comúnmente recursiva. Ejemplo de relaciones binarias: un hombre dueño de un perro, un collie es un perro, una persona es presidente. Ejemplo de relaciones ternarias: doctores prescriben drogas para pacientes, estudiantes toman clases de profesores. Ejemplo de relaciones recursivas: gente casada con gente, perros que comen perros, gente supervisa gente.

El grado de una relación se puede mostrar en los diagramas por el número de líneas dibujadas entre las entidades. Por ejemplo:





♦ Cardinality (cardinalidad):

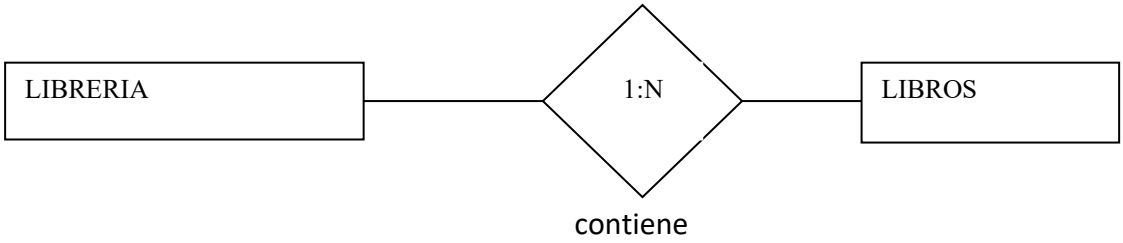
La cardinalidad o conectividad o multiplicidad de una relación describe cuantas instancias de una entidad pueden tener la relación con una sola instancia de otra entidad involucrada en la relación. Existen dos clases generales. Supongamos una entidad E, que tiene una relación R con otra entidad A y B. Supongamos una instancia de A y B. Luego, puede darse el caso que más de una instancia de E pueda ocurrir (para cualquier combinación de A y B), o se puede dar el caso que combinación de una instancia de E se permita para alguna combinación de A y B...

Para ser más específico, consideremos relaciones binarias. Son posibles cuatro formas: one to one, one to many; many to one y many to many, que se escriben usualmente con 1:1; 1:n; n:1; m:n respectivamente. Para relaciones ternarias las ocho posibilidades son

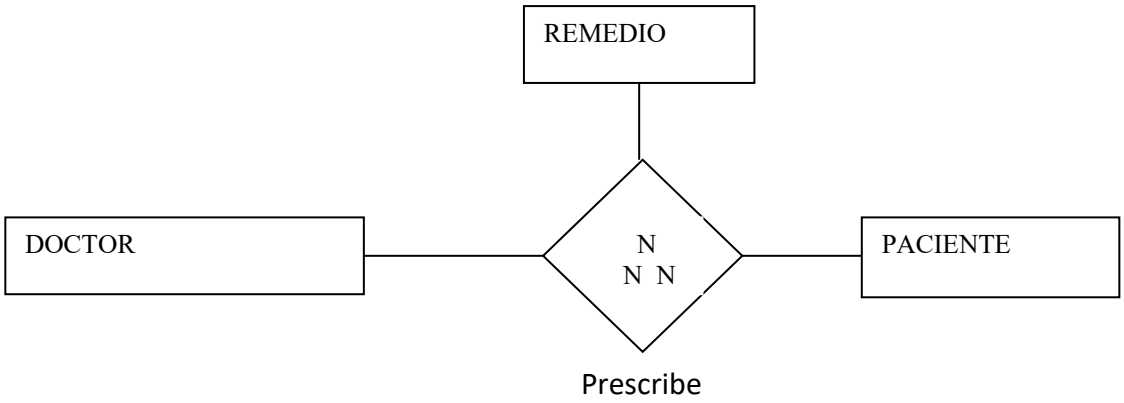
1:1:1; 1:1:n; 1:m:n; y m:n:p (más las mutaciones). Por ejemplo una librería contiene libros; un estado tiene gobernador (son 1:1).

Pero "doctores prescriben drogas para pacientes"; "un cliente posee una multa de la librería" (m:n:p) y "estudiantes toman clases de un profesor" (n:m:1).

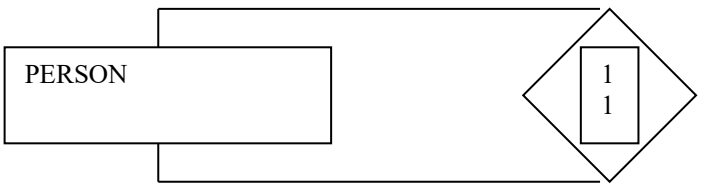
Existen muchas formas de indicar la cardinalidad, el libro usa un rombo para designar la relación con las marcas apropiadas, aquí hay algunos ejemplos.



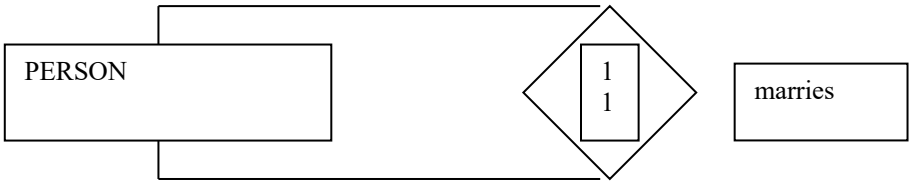
Este diagrama se interpreta: dada una instancia de una librería, la librería contiene muchos libros. Dada una instancia de un libro, el libro es contenido a lo sumo en una librería.



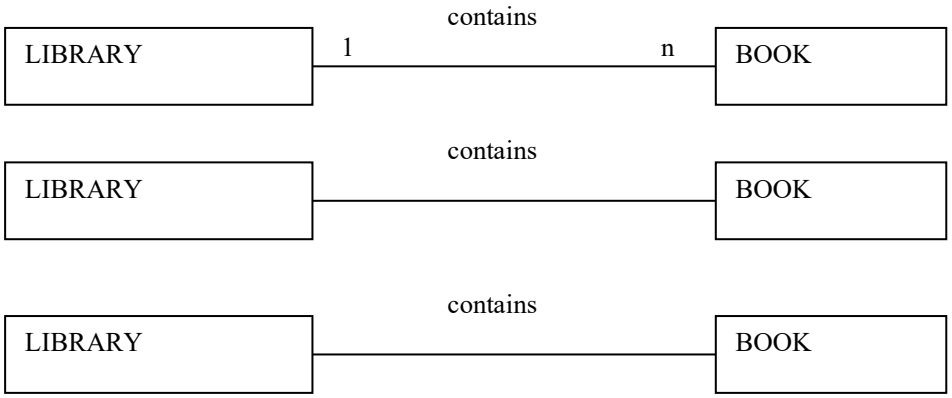
Este diagrama es más complejo. Dada una instancia de un doctor y una de paciente, ese doctor puede prescribir muchas drogas a ese paciente. Dada una instancia de un doctor y una droga, ese doctor puede haber prescripto esa droga a muchos pacientes. Dada una instancia de un paciente y una instancia de una droga, muchos doctores pueden haber prescripto esa droga a ese paciente.



Este diagrama relaciona gente con otra gente. Dada una instancia de una persona, esa persona puede estar casada, a lo sumo con una persona. De la forma que está el diagrama una persona puede estar casada consigo misma. Los diagramas de E-R no capturan cada matriz de significado.



Otra forma de dibujar una relación es simplemente etiquetando cada final de las líneas de la relación con un "1" o "n" apropiadamente. Un tercer método es usar flechas: una sola flecha es de "1" y doble flecha en el lado múltiple. Un cuarto método es no usar nada en el lado de "1" y colocar patitas en el lado múltiple



Las relaciones binarias one-to-many son muy comunes. Matemáticamente indican funciones, por ejemplo "contains: books AE libreries" es una función.

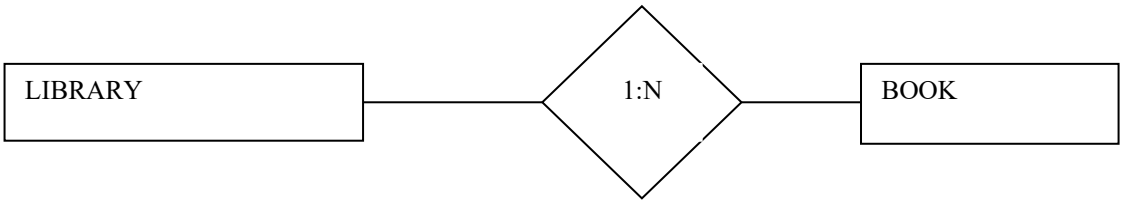
- Optionality (opcionalidad):

El aspecto cardinalidad provee una indicación del número máximo de instancias de una entidad que pueden entrar para una relación dada con instancias simples de otras entidades. El aspecto opcionalidad o membership o condicionalidad simétricamente provee la indicación de mínimo número.

Discutiremos esto sólo en términos de relaciones binarias. Supongamos una relación R que ocurre entre las entidades X e Y. En general, una ocurrencia x de X debe participar en la relación o puede participar. Esto también se da para las instancias y de Y.

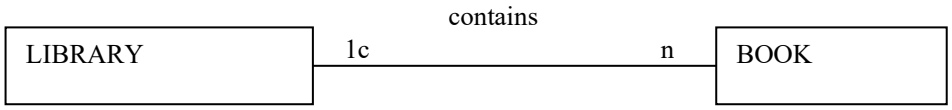
Hay muchas formas de indicar la optionalidad. El libro coloca una barra en el final del lado que es obligatorio y un pequeño círculo en el que es opcional.

Se interpreta: dada una instancia de librería, esa librería debe contener al menos un libro y puede contener más de uno. Dada una instancia de "libro", ese libro puede estar contenido a los sumo en una librería pero puede no estar en ninguna.



Otra posibilidad es colocar una pequeña "c", que significa condicionalidad, en el lado de la relación que es opcional. Otra opción es colocar un círculo de opcional en el lado de la entidad que es opcional y dejar sin marca el obligatorio. Para indicar la opcionalidad es conveniente usar la "c": 1c:n; mc:p; etc.

Ejemplos:



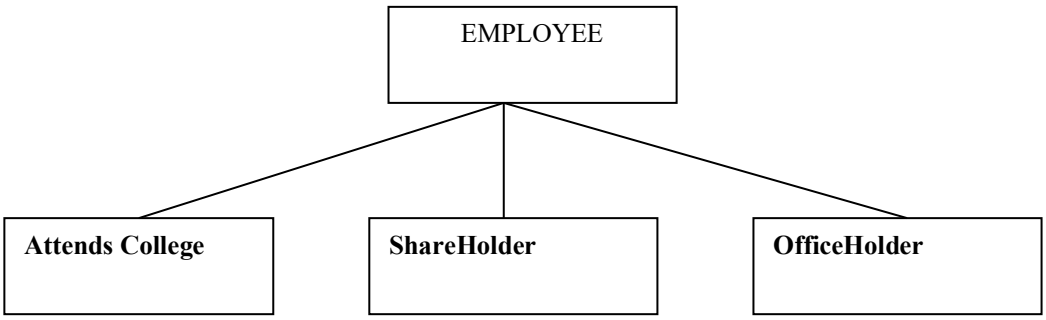
- Clases (classes):

Las relaciones se pueden agrupar dentro de un número de clases, dependiendo de la semántica resaltada. Esencialmente todos los modelos de E-R importantes incluyen las clases: subset (subconjunto), generalization, aggregation, y association. Algunos modelos tienen clases adicionales (no se discuten aquí). Estas ideas se pueden usar para arreglar entidades en jerarquías que tengan ciertas características.

- Subset hierarchy:

Se usa para especificar posibles superposición de conjuntos de entidades. Por ejemplo: tenemos una entidad E cada instancia de E puede ocurrir como una instancia de las entidades E1, E2, E3... En. La entidad E se usa para describir esas características de las entidades que son independientes de otras entidades. Por ejemplo: la entidad empleado puede incluir empleados correspondientes a attending college (colegios) empleados que

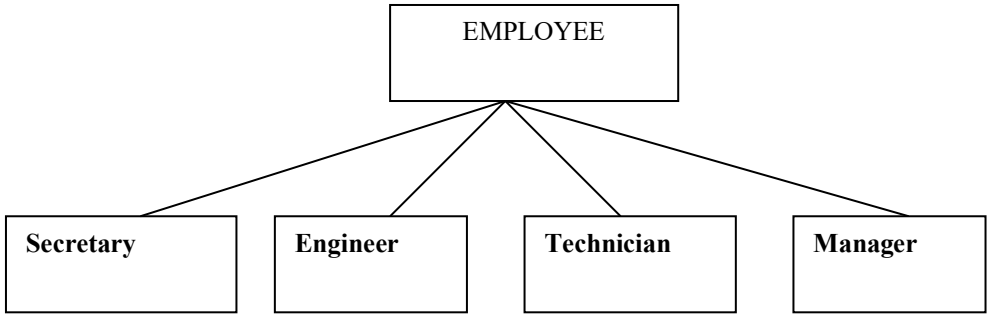
también son shareholders (accionistas) o empleados que son political office (oficina política), y así sucesivamente.



Note que, en términos de cardinalidad y opcionalidad, una jerarquía subset, es 1:nc

- Generalization hierarchy:

Se usa para formar una partición de una entidad. Es similar a la subset excepto que cada ocurrencia de E1, E2... En. Por ejemplo, supongamos que todos los empleados se clasifican en: secretarias, ingenieros, técnicos y gerentes. Luego empleado es la generalización de esta clasificación. En términos de cardinalidad y opcionalidad una generalization hierarchy es 1:n.



Note que en la jerarquía subset y generalization tenemos entidades tangibles en la cabeza (top), con roles en el final (bottom). Mientras que esto es muy común no es la única posibilidad. La entidad "hardware failure" (fallas de hardware) puede ser una generalización de disk failure, memory failure y así sucesivamente. En este caso todas las entidades son eventos.

Los esquemas de clasificación biológica son buenos ejemplos de generalizaciones de muchos niveles. Un vertebrado es una generalización de: pez, reptil, anfibios, pájaros y mamíferos). Un mamífero es una generalización de carnívoros, primates, etc... Un carnívoro es una generalización de caninos, felinos, etc.

Ambas clases se refieren a una relación is-a, como se describe usualmente en inglés. Una secretaria es una empleada, un oso es un carnívoro.

- Aggregation:

Es un tipo de componente ensamblado (assembly-component) de una relación y a veces se refiere como una relación "part-of". Un tornillo es parte de un motor, un motor es parte de un ventilador, un ventilador es parte de una computadora. Esto es un ejemplo de relaciones bill of materials, or parts explosion. Básicamente, aggretation combina entidades de bajo nivel con dentro de entidades compuestas. Aggregation es frecuentemente multinivel y pueden ser recursivas.

Desafortunadamente las jerarquías subset y generalization y la jerarquía agregarion se forman de distintos tipos de entidades y el resultado es diferente. Generalmente tratamos con entidades tangibles en todos los niveles, no roles. Aquí hay más ejemplos: una boleta se compone de dirección, información de compra, e información de la boleta, esta última está compuesta por precios, descuentos, franqueo (postage), e impuestos.

Un libro está compuesto de tapas y páginas. Las páginas se dividen en matter, text y bach matter. Front matter se divide en título de la página, prefacio y tabla de contenido. Las páginas de texto contienen material escrito y material gráfico o ambos. Aquí tenemos una jerarquía compuesta de subset y generalization y agregation.

- Association:

Relaciona dos o más entidades independientes. Generalmente no implica jerarquía. En las relaciones jerárquicas las entidades de nivel más bajo son usualmente débiles (weak), como las instancias no pueden ocurrir a menos que ocurre la instancia correspondiente del nivel superior.

No es el caso de las relaciones association. La relación de trabajo es un ejemplo, ambas gentes (potenciales empleados y potenciales empleadores), existen independientemente unos de otros.

- Attributes of relationships (atributos de relaciones):

Las relaciones frecuentemente tienen atributos. Estos son datos existentes que se relacionan solamente con la relación, no con ningún componente de las entidades. Grade, por ejemplo es un atributo de la relación registred-for entre estudiantes y clases, fecha de casamiento es un atributo de marriege. Esto es muy común en la realidad y en las BD.

Conceptual DataBase Design: Restricciones de Integridad

Muchas de las restricciones de Integridad que serán necesarias en el modelo de información se contienen en la descripción de entidades, atributos y relaciones. Las restricciones de Dominio, incluyendo la posibilidad de valores nulos, han sido descriptas como parte del atributo de entrada en el DD. Muchas de las semántica relacionada con las foreign keys han sido capturadas en los conceptos de cardinalidad, opcionalidad y clase. Sin embargo pueden existir otras restricciones. Todas deberían ser descriptas en el DD.

Cada una de las restricciones pueden tener condiciones de disparo, definiciones de restricciones y acciones requeridas. Sin embargo muchas de estas son irrelevantes para el modelo de información, como se aplican directamente a los programas que usan las BD. Así en la mayoría de los casos, se puede dar sólo el nombre de la restricción, proveer una descripción en Inglés y luego proveer la definición formal.

Entonces, ¿por qué hacer previsiones para triggers y acciones?

Simplemente porque estas tienen que pensarse cuando se define restricción y necesitamos un lugar para ponerlas para que no sean olvidadas luego. Este es un lugar donde una separación limpia de las ideas puede ser relajado, para hacer frente al proceso actual de diseñar un sistema largo.

Conceptual DataBase Design: final Cleamup

Algunos chequeos finales se requieren casi siempre antes de pasar a la implementación. Yo apruebo dos pasos, uno privado y otro público.

El paso privado es para revisar todo dos veces más. Primero volver a los requerimientos originales, ambos al documento de especificación de requerimientos y a los requerimientos de BD. Por cada requerimiento, verificar que se satisfaga para el diseño conceptual. ¿Existen los datos para llevar a cabo cada tarea operacional, de control y planeamiento? Que tal cada salida de los procesos del DFD ¿Entran inmediatamente en otro proceso o tienen un lugar en el modelo de información? ¿Se satisfacen todas las reglas de operación por las restricciones estructurales o las explícitas en el modelo de información?

Después de hacer esto, trate de imaginar varios tipos de consultas que los clientes harán en el futuro, probablemente, estos fueron registrados durante el análisis de requerimientos.

¿Existe la información para responder tales consultas? No se preocupe por la eficiencia, sólo por la flexibilidad. Concéntrese en consultas complejas que requieran información de varias entidades. ¿Existen atributos y relaciones para responderlas?

Si no existen, usted decide si modificar el modelo o ignorar las consultas. Si decide ignorarlas recomiendo que las agregue al diseño de BD, con una nota diciendo que se necesita para satisfacer cada una y si la información existe en el modelo. Esto puede ser muy valioso en el futuro, cuando los usuarios soliciten cambios para el sistema operacional.

Luego ejecute el mismo chequeo para otras cosas. Para cada elemento (entity, atribute, relationship, restriction), del diseño conceptual, identifique el requerimiento o los requerimientos que el elemento intenta satisfacer. Probablemente debería registrar esto en el diseño de BD también. Si después de finalizar este trabajo encuentra entidades hacia atrás hasta el requerimiento, necesita registrar la razón de por qué están ahí (en el diseño de BD). En la mayoría de los diseños de sistema de aplicación queremos que cada elemento de diseño sea seguible hasta un requerimiento explícito o implícito. No estoy seguro de este principio para BD, sin embargo, usualmente queremos permitir que el cliente tenga respuestas anticipadas a consultas.

Sin embargo, se necesita justificar cualquier extensión particularmente si requiere trabajo adicional de actualización por parte de los clientes. Se necesitará verificarlos con el usuario directamente. Por ejemplo: desde el proceso de requerimientos puede haber deducido una restricción de integridad que no se menciona explícitamente en ningún lado; por lo tanto decida usted si agregarla o no. La experiencia en computación lleva a creer que es una restricción necesaria, porque la gente comete cierta clase de errores (esto se aplica mayormente en actividades de actualización o para prevenir equivocaciones comunes o errores de razonamiento). Registre la razón y verifique su recomendación con el usuario (s). Si se rehúsan a aceptarlo, registre el hecho y omítalo (esto evitará que alguien trate de agregar la misma característica en el futuro).

Puede haber agregado una relación, aunque el usuario no la haya requerido, porque su experiencia en otros proyectos de BD lo llevan a creer que las consultas con BD requieren que las relaciones sean factibles a presentarse en el futuro. El comportamiento de los clientes es casi siempre alterado después que las BD están disponibles. Descubren cuanto más pueden hacer y comienzan a querer hacerlo. La aplicación inteligente puede ahorrar un montón de problemas en el futuro.

También se debería tener en vista un chequeo formal del diseño conceptual. Hay dos razones para esto. Primero, que puede surgir un nuevo requerimiento de los usuarios que estén viendo todo el diseño por primera vez. Segundo, se necesitarán usuarios comprometidos emocionalmente con el diseño, un público que muestre aceptación es una buena manera de obtenerlos.

¿Quién debería asistir? Probablemente ha estado trabajando con alguna persona de cada área -esa es la persona que debe asistirlo. También pregunte por cada gente de cada área, obtenga sus nombres de los primeros contactos. La invitación es importante, políticamente como su agradecimiento de la importancia y autoridad.

Todos los miembros del equipo de desarrollo deberían estar presentes como observadores. Probablemente obtendrá algunas expresiones fuertes de algún asunto particular, de algún cliente y puede haber desacuerdos entre los clientes.

También estoy a favor que trate de que asista su superior inmediato. Adviértale que es una reunión de trabajo seria en la cual se presentarán nuevos requerimientos que se necesitará hacer una revisión de la mayor cantidad de ítms posibles. Probablemente necesitará soporte de supervisión durante la implementación y operación y que él se debe exponer a los usuarios.

Aquí estará tratando con gente que no encontrará intimidante la presencia de su jefe. Como lado positivo la presencia de este puede darle a la gente seguridad de que la organización está tomando el proyecto seriamente.

¿Cómo proceder? Una vez que la reunión está agendada mande una copia del diseño a cada uno de los que asistirá con un memo. Éste se debe escribir específicamente para cada tipo de persona. Para los clientes, señalará aquellas áreas del diseño conceptual que se aplica específicamente a ellos y cualquier otra área a la que deban prestar atención.

Pídales que se tomen una o dos horas para leerlo y tomar nota de cualquier punto pendiente.

Aconsejo repartir todas las copias al primer contacto y reforzar verbalmente la importancia de todo el proceso y contestar las preguntas de procedimiento.

Comience la reunión nombrando a cada uno el propósito de la reunión. El propósito es encontrar requerimientos pasados por alto, identificar inconsistencias y encontrar errores. Encontrar no arreglar, nunca intente repararlas allí. Si la reunión termina sin haber hecho ninguna acción sobre algún ítem, fallará lo que significa que nadie ha hecho su tarea. Ningún diseño es perfecto y el fracaso para encontrar errores significa que los usuarios no están prestando atención.

Usted debería tener una secretaria que anote todo lo que deba, pero que no participe de la reunión. Registre exactamente cuál es el problema, y quién necesita ser informado de la solución. Recomiendo etiquetar esto con "actions items" y "suggestion". El primero son agujeros que deben ser llenados, usted necesitará cambiar el diseño para llenarlos y verificar el cambio con quienquiera que deba ser informado por el cambio. Cuando se han verificado todas las actions items deberá repartir el nuevo diseño con un memo, canalizando cómo fue respondido. Cada action items y exponiendo explícitamente cada cambio hecho al diseño.

La sugerencias son informales, pero pueden ayudarle en el próximo paso. No identificar errores en el diseño son ideas para mejorarlo. Dígale al sugeridor qué se decidió. Si la sugerencia resulta en el cambio del diseño deberá documentarlo junto con las actions items en el memo de la próxima versión del diseño.

Recorra página por página del documento de diseño o sección por sección. Este es un proceso tedioso. Dos horas de reunión está bien, si quedan puntos por ver después de dos horas parta la reunión ahí y agende otra en una semana.

Trate de llegar a acuerdos con respecto a las horas y fechas de estas reuniones. Si ha hecho un buen trabajo esta vista formal (formal view) será suficiente. Muchos usuarios están muy ocupados para más revisiones.

A pesar de todo, algo puede pasarse por alto y puede ser descubierto durante el testeo u operación y se necesitará un rediseño y reimplementación. Usualmente yo termino la revisión del diseño sobre este punto para preparar a la gente de este evento. Si se hace bien la sorpresa se puede reducir cuando se presente este hecho y nadie se molestará: "sólo es parte del proceso de implementación".