Universidad Nacional de San Juan

FCEFN

Departamento de Informática

ALGUNAS CONSIDERACIONES PARA EL DISEÑO DE BASES DE DATOS RELACIONALES

Lic. Silvina Migani
Lic. Cristina Vera

Cátedra Bases de Datos - Carreras LCC, LSI y TPU

1. ETAPAS DE DISEÑO DE UNA BASE DE DATOS

El proceso completo del diseño de una base de datos incluye una serie de pasos que son ilustrados por la figura que se muestra abajo, extraído del libro Fundamentals of Database Systems (7ma edición, 2016), de los autores Elmasri y Navathe.

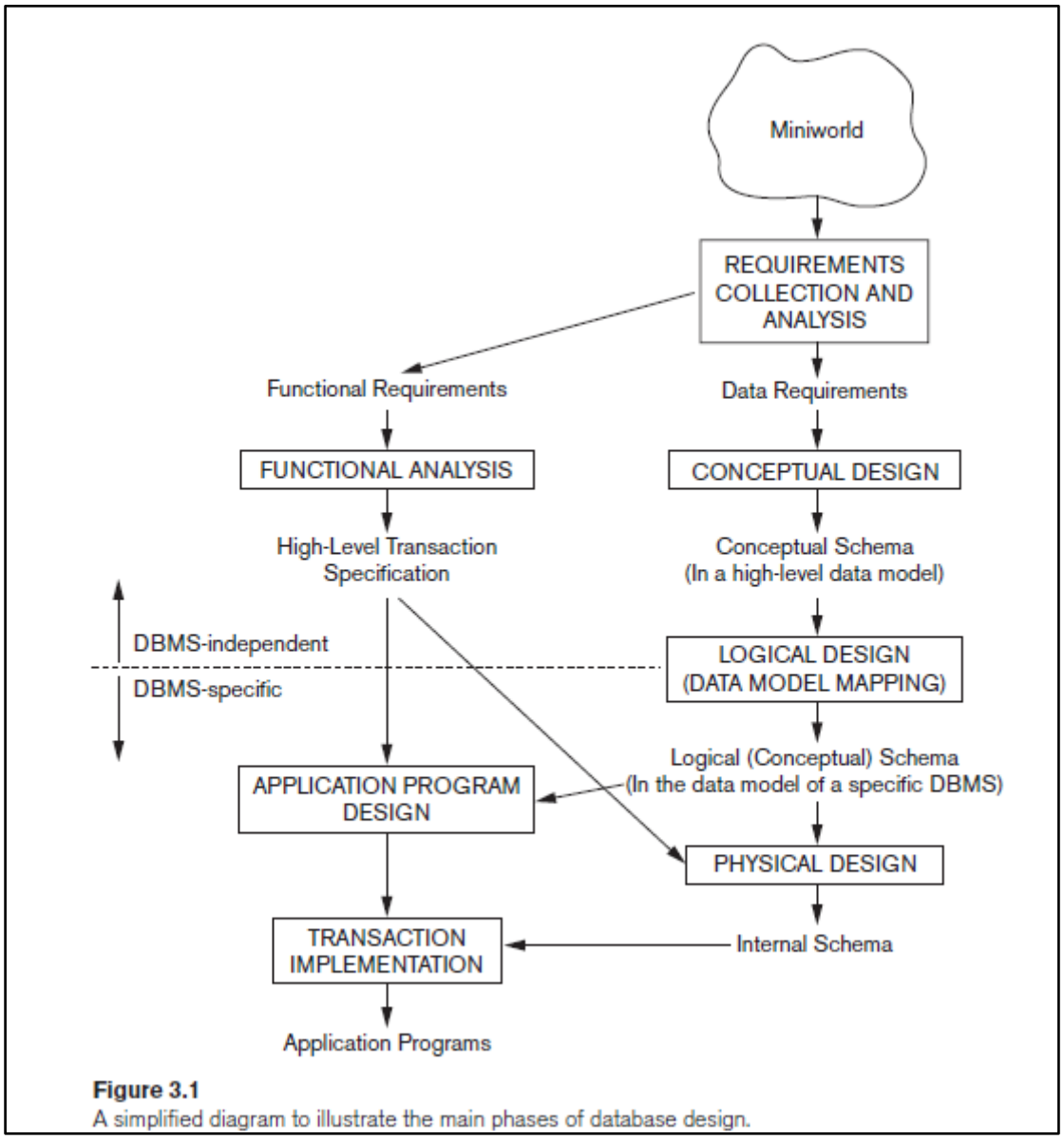


Figure 3.1
A simplified diagram to illustrate the main phases of database design.

Fuente: *"Fundamentals of Database Systems (7ma edición, 2016)". Elmasri y Navathe.*

Nota:

- *En este texto se mencionan como sinónimos los siguientes términos:*
  - *esquema/modelo/diagrama*
  - *tipo de entidad/conjunto de entidades/entidades (aunque este último no sería adecuado, por comodidad se las menciona comúnmente así).*
  - *tipo de relación/conjunto de relaciones/relaciones/vinculaciones (aunque los dos últimos no serían muy adecuados por comodidad se las menciona también así).*

En general, en los diagramas conceptuales mostrados como ejemplo, no se han indicado claves primarias por simplicidad.

Las fases ilustradas se explican brevemente a continuación:

Recopilación y Análisis de Requisitos: Durante este paso, los diseñadores de la base de datos entrevistan a los posibles usuarios de la base de datos para entender y documentar sus **requisitos de los datos**. El resultado de este paso es un conjunto de requisitos de los usuarios redactado de forma concisa. Estos requisitos deben especificarse de la forma más detallada y completa posible.

Paralelamente a la especificación de los requisitos de datos, es útil especificar los r**equisitos funcionales** conocidos de la aplicación. Éstos consisten en las operaciones (o transacciones) definidas por el usuario que se aplicarán a la base de datos, incluyendo tanto las consultas como las actualizaciones.

En el diseño de software, es habitual utilizar diagramas de flujo de datos, diagramas de secuencia, escenarios y otras técnicas para especificar los requisitos funcionales. No vamos a discutir ninguna de estas técnicas aquí; normalmente se describen en detalle en los textos de ingeniería de software.

Diseño Conceptual: Se crea el esquema conceptual de la base de datos, utilizando un modelo de datos conceptual de alto nivel.

El esquema conceptual es una descripción concisa de los requisitos de datos de los usuarios e incluye descripciones detalladas de:

- Tipos de entidades,
- Tipos de relaciones y
- Restricciones (Cardinalidad, etc.)

Se caracteriza por:

- No incluye detalles de implementación por lo que suelen ser más fáciles de entender y pueden utilizarse para comunicarse con usuarios no técnicos
- Referencia para garantizar que se cumplen los requisitos de datos de todos los usuarios y que los requisitos no entren en conflicto.
- Permite a los diseñadores de bases de datos concentrarse en especificar las propiedades de los datos, sin preocuparse de los detalles de almacenamiento e implementación.
- Durante o después del diseño del esquema conceptual, las operaciones básicas identificadas pueden utilizarse para confirmar que el esquema conceptual permite darles solución a esos requisitos funcionales.

Diseño Lógico: Se realiza el mapeo del esquema conceptual en el esquema según el modelo de datos correspondiente al SGBD donde se implemente. El mapeo del modelo de datos suele estar automatizado o semiautomatizado en las herramientas de diseño de bases de datos. En este caso, dado que este apunte refiere al diseño de una base de datos relacional, el esquema lógico generado estará compuesto por tablas y sus restricciones.

Diseño Físico: Se especifica la estructura y organización de los archivos internos, los caminos de acceso (índices) a ellos y los parámetros de configuración del almacenamiento, como por ejemplo, tamaño de página, porcentaje de espacios libres por página, etc. Paralelamente a estas actividades, se implementan los programas de aplicación como transacciones de base de datos correspondientes a las especificaciones de transacciones de alto nivel.

2. DISEÑO CONCEPTUAL DE UNA BASE DE DATOS

Como se indica en el punto anterior, el diseño de una base de datos es un proceso secuencial donde el resultado de una fase es tomado como entrada en la siguiente. Por ello, es fundamental evitar errores desde el comienzo, ya que de cometerlos serán trasladados a las etapas posteriores, y en última instancia, se obtendrá a una base de datos implementada con problemas ante operaciones de actualización y/o consulta.

Cabe destacar que este apunte encara el diseño de una base de datos relacional, por ello, todo lo expuesto es pertinente a ese paradigma. Consecuentemente, en lo referente al diseño conceptual se describe el Modelo Entidad/Relación Extendido (MERE) puesto que es el estándar usado en este contexto.

2.1. ELEMENTOS DEL MERE

El Modelo Entidad Relación (MER) describe los datos como **entidades**, **relaciones** y **atributos**. El MERE es una extensión del MER, donde se introducen los conceptos de entidades débiles, generalización/especialización y agregación. Este apunte no se ocupa de los conceptos extendidos, ya que en los libros citados en la bibliografía se encuentran explicados ampliamente. La simbología utilizada en la materia es la de Chen.

2.2. CARACTERÍSTICAS DEL PROCESO DE DESARROLLO DEL MERE

Es un proceso de **refinamiento iterativo**, comenzando con un esquema más elemental, con pocos detalles, hasta alcanzar por último, un diseño adecuado y completo. Como práctica general, dada la descripción de los requisitos de una base de datos, los sustantivos que aparecen en la narrativa tienden a dar lugar a nombres de tipos de entidades, y los verbos tienden a indicar nombres de tipos de relaciones. Los nombres de los atributos suelen surgir también de sustantivos, pero en este caso, describen a los sustantivos identificados como tipos de entidades. Un minimundo puede generar diferentes esquemas conceptuales correctos, donde se han priorizado posiblemente diferentes aspectos, como por ejemplo, extensibilidad del esquema, conveniencia para las aplicaciones que usarán esa base de datos, minimizar el tamaño total de la base de datos, entre otros.

1) Identificar las posibles entidades del minimundo.
2) Definir atributos de las entidades identificadas.
3) Identificar relaciones o vinculaciones entre entidades. Recordar que si una relación necesita vincularse a una relación preexistente, esta última deberá convertirse en una agregación.
4) Revisar los atributos de las entidades definidas. Puede que algunos atributos especificados inicialmente, hayan quedado representados por alguna de las relaciones reconocidas (4).
5) Identificar los atributos que constituyen la clave primaria de las entidades y distinguir, si existieran, entidades débiles. Una entidad puede tener más de un atributo (o conjunto de atributos) que verifique las condiciones para ser clave primaria, sin embargo, se elige uno y se lo identifica con el nombre del atributo subrayado dentro del MERE.
6) Especificar los atributos y las multiplicidades de las relaciones identificadas. Cuando un atributo está determinado por la combinación de los tipos de entidades participantes de un tipo de relación, ese atributo corresponde al tipo de relación considerada.
7) Identificar los atributos que conforman la clave primaria de las relaciones existentes. Al igual que para las entidades, una relación puede tener más de un atributo (o conjunto de atributos) posibles de participar en la clave primaria, sin embargo, se elige uno y se lo identifica con el nombre del atributo subrayado dentro del MERE.
8) Analizar las entidades en busca de posibles generalizaciones o especializaciones.
9) Revisar el esquema conceptual generado hasta el momento en pos de identificar posibles errores y/o duplicaciones de información. Las redundancias siempre deben ser minimizadas o directamente eliminadas. De ser necesarias por razones de rendimiento, deberán ser incluidas en fases posteriores, especialmente en el diseño físico.

2.3. RECOMENDACIONES A CONSIDERAR DURANTE LA ELABORACIÓN DEL MERE

- **Elegir nombres adecuados** para los distintos tipos de elementos del modelo (entidades, relaciones, atributos, roles): Hay que elegir nombres que transmitan, en la medida de lo posible, el significado de los diferentes elementos del esquema.

  También se recomiendan las siguientes buenas prácticas:

  - Utilizar nombres singulares para los tipos de entidad en lugar de plurales, porque el nombre del tipo de entidad se aplica a cada entidad individual que pertenece a ese tipo de entidad o conjunto de entidades.
  - Para las relaciones, elegir en la medida de lo posible, nombres que sean legibles semánticamente de izquierda a derecha y de arriba a abajo.

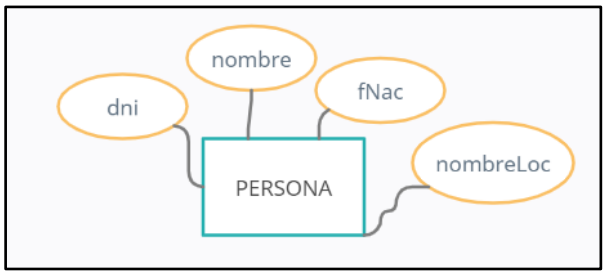
- **Elegir los tipos de elementos correctos, o más adecuados** para representar los objetos del minimundo. A menudo en etapas iniciales de diseño, pueden presentarse dudas que deben ser disipadas a medida que se avanza en el proceso.
  - **Un elemento de la realidad podría ser modelado primero como un atributo de un tipo de entidad** (digamos, la entidad A) **y luego refinado como un tipo de entidad** (digamos la entidad B) y una relación entre ambos tipos de entidades (A y B). Podríamos decir entonces, que un atributo es **"promovido"** a un tipo de entidad independiente, dando lugar a:

    a) <u>Describir otros atributos del elemento promovido:</u> Si se observa el Caso1, el elemento de la realidad "localidad" es modelado como un atributo del tipo de entidad *PERSONA*, y en este caso se decide colocarle a ese atributo el nombre *nombreLoc.*

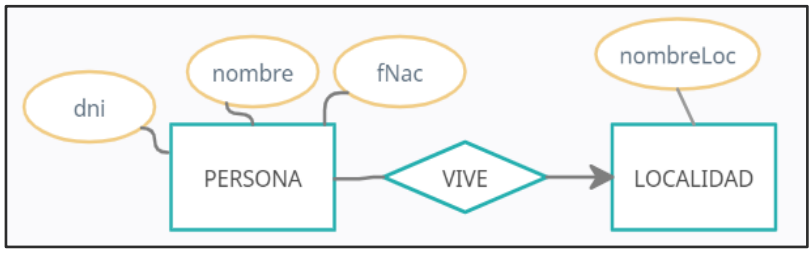
    El Caso2 muestra un esquema equivalente, donde el elemento de la realidad "localidad" es modelado como un tipo de entidad, concretamente llamada *LOCALIDAD,* con el atributo *nombreLoc*.

    Ambos modelos o esquemas son equivalentes ya que describen que una persona de ese minimundo posee dni, nombre, fecha de nacimiento y localidad donde vive.

    Caso1

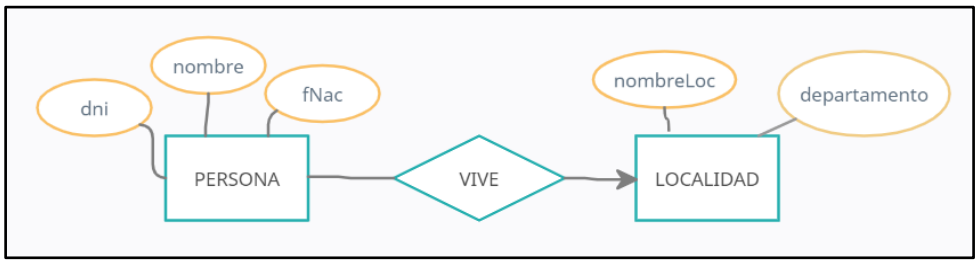
    

    Caso2

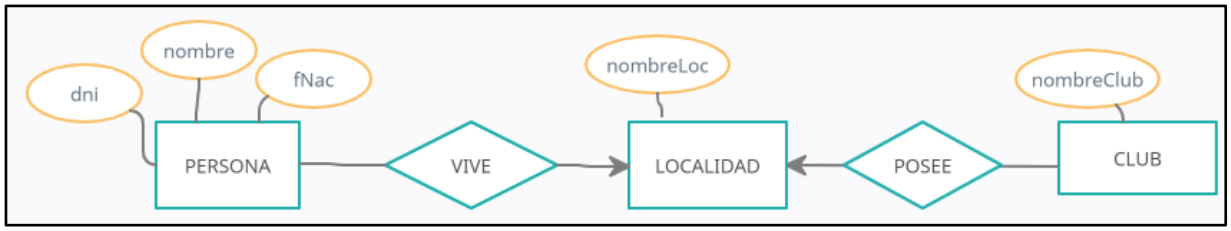
Sin embargo, una diferencia entre estos modelos es que el Caso2 da la posibilidad de considerar un atributo del elemento en cuestión, situación ilustrada en el Caso3, donde el tipo de entidad *LOCALIDAD* posee además del atributo *nombreLoc*, el atributo *departamento*.

Caso3



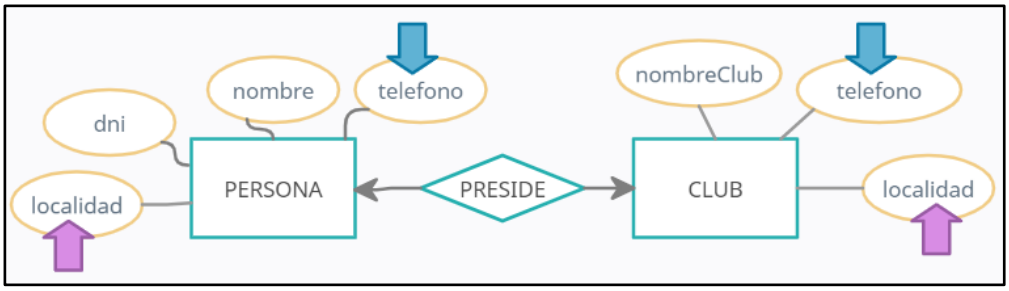
b) <u>Describir más de una relación con el elemento transformado a entidad:</u> Otra diferencia entre los modelos (1 y 2) es que el Caso2 también permitiría que ese elemento localidad (modelado como el tipo de entidad *LOCALIDAD*) esté vinculado con otro tipo de entidad, en este caso *CLUB*, como lo muestra el Caso4. El Caso4 visto de otro forma, muestra que si más de un tipo de entidad necesita registrar un mismo atributo (localidad en PERSONA y en CLUB) en lugar de repetir el atributo en ambas entidades, se promueve el atributo a entidad y se vinculan las entidades esa nueva entidad generada (atributo promovido).

Caso4

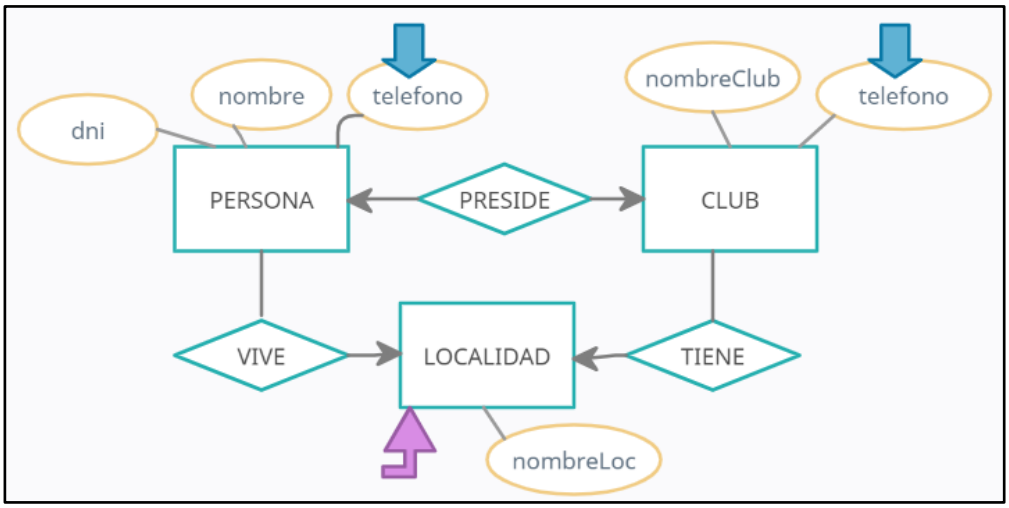


Es necesario hacer una aclaración relacionada a la situación recién planteada. El Caso5 muestra dos situaciones análogas resueltas o reformuladas en refinamientos posteriores de manera diferente. Concretamente presenta dos atributos *teléfono*, uno en *PERSONA* y otro en *CLUB*, y también muestra dos atributos *localidad,* tanto en *PERSONA* como en *CLUB.* Según lo explicado en el párrafo anterior, ambos atributos debieran promoverse a entidades, sin embargo, la solución al problema planteado lo representa el Caso6, donde uno de los atributos comunes (localidad) es promovido a entidad, y el otro (teléfono), no. ¿Por qué se tomó tal decisión? La respuesta tiene que ver con la existencia independiente o no de ese elemento representado por el atributo (en el ejemplo, teléfono y localidad), y el sentido o valor que tendría mantener ese elemento aisladamente. ¿Tendrá sentido mantener todas las localidades, por ejemplo de una provincia dada, independientemente que existan personas y/o clubes que se ubiquen en ellas?, la respuesta probablemente sea sí. Esto permitiría por ejemplo, conocer todas las localidades de la provincia, las localidades donde no hay clubes ni personas (que podrían corresponder a socios por ejemplo) viviendo en ellas, etc. Ahora bien, analizando el atributo teléfono (por supuesto, dentro de este contexto) se debiera considerar ¿tiene sentido mantener nros. de teléfonos sin que existan personas y/o clubes que los posean? ¿qué información útil podría brindar? seguramente que ninguna. Son estas respuestas las que justifican la transformación del Caso5 en el Caso6.

Caso5



Caso6

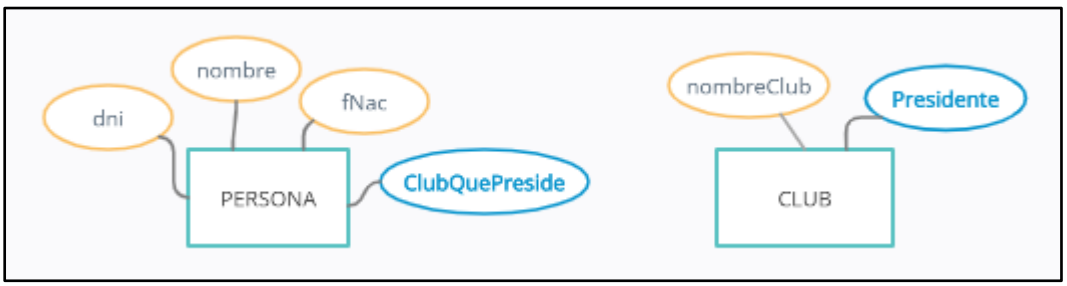


- Inversamente, **un elemento de la realidad puede ser modelado inicialmente como un tipo de entidad y luego, en refinamientos posteriores, "degradado" a atributo.** Un ejemplo de esto ocurriría si se pasara del Caso2 al Caso1.

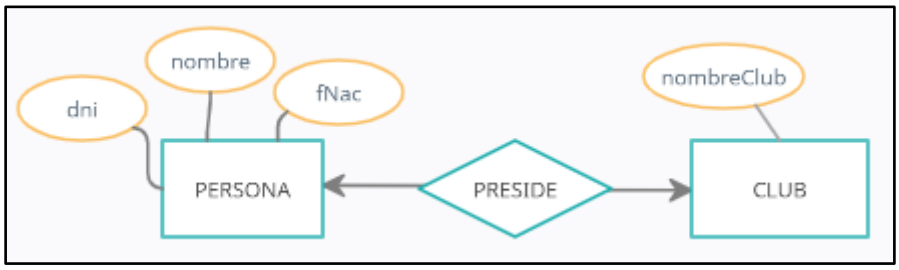
- Otro caso relacionado al mismo hecho son los pares de atributos inversos (atributos que se referencian entre sí). Esta situación puede presentarse inicialmente, pero luego deben ser convertidos en una relación binaria entre los tipos de entidades que poseen los atributos inversos. Cabe aclarar que una vez generada la vinculación, los atributos deben eliminarse de las entidades para evitar redundancias.

  Siguiendo el minimundo del ítem anterior, se presenta un ejemplo ilustrativo:

Caso7



Caso8



En este caso, los dos modelos no son equivalentes. De hecho el Caso6 es incorrecto, e ilustra los atributos inversos *"ClubQuePreside"* y *"Presidente"*. Como se dijo, podría ser representativo de los primeros refinamientos del modelo, pero necesariamente transformado para llegar a un modelo correcto, como el representado en el Caso6.

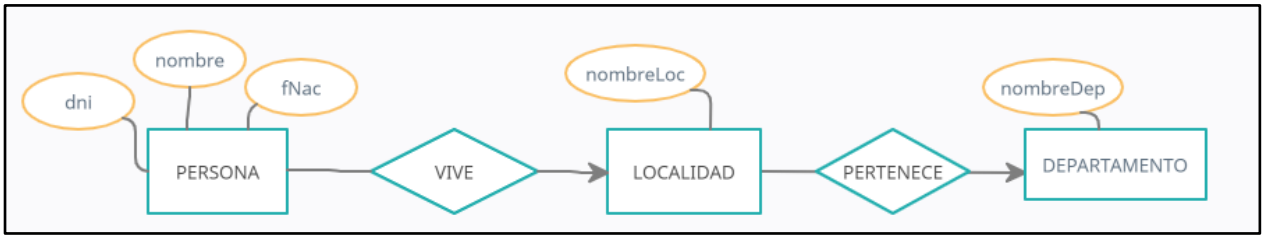
- **"Nada" es absoluto**, es decir, todo depende de la semántica y de las necesidades de información de cada minimundo descripto. Por ejemplo, se podría pensar que toda persona tiene sólo una edad, la edad actual. Sin embargo, podría ser necesario registrar la edad que una persona tenía en diferentes momentos, es decir, en diferentes fechas. Por ejemplo, Juan que nació el 01/02/2000 tuvo 1 año el día 01/01/2002, 2 años el 01/03/2002, y hoy (supongamos 01/03/2022) tiene 22 años.

- El **ente/organización/empresa del minimundo que se necesita modelar, no constituye un tipo de entidad** o conjunto de entidades a representar en el MERE. Todas las entidades y relaciones presentes en el MERE corresponderán a ese único ente.

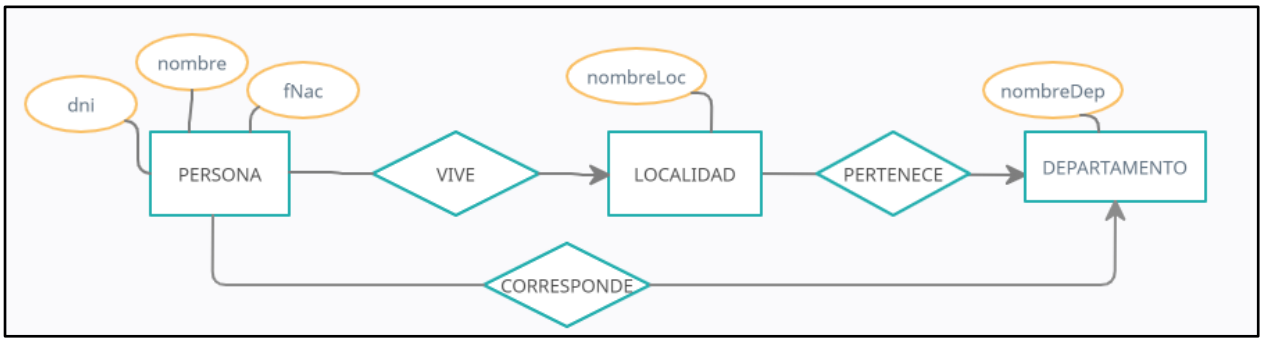
2.4. CARACTERÍSTICAS DE UN MERE CORRECTO

- **Fidelidad:** El diseño debe ser fiel a las especificaciones de la aplicación. Es decir, los tipos de entidades, relaciones y sus atributos deben reflejar la realidad expuesta. Por ejemplo, no se pueden agregar atributos no mencionados en la narrativa de un minimundo, a pesar de que ese atributo podría tener sentido en la realidad considerada.
- **No redundante**: Todo debe ser representado una sola vez.
- **Que modele las relaciones adecuadamente**: Las entidades pueden conectarse de varias maneras. Sin embargo, no cualquier modo es correcto, por ejemplo, añadir a nuestro diseño todas las relaciones posibles no suele ser correcto. A continuación, se muestran ejemplos que intentan representar personas junto a las localidades y departamentos donde ellas viven.

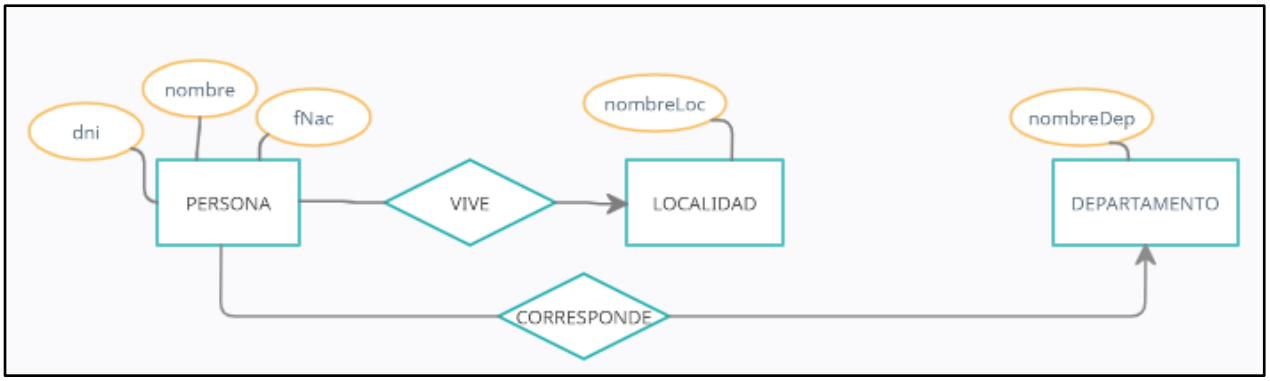
Caso9



Caso10



Caso11



El único diagrama correcto es el mostrado en el Caso9. El Caso10 posee la relación redundante *CORRESPONDE* y el Caso11, si bien no presenta redundancia y aparentemente podría ser correcto, no es una buena alternativa de diseño, el problema se presenta cuando se necesitan actualizar los datos.

- **Que sea lo más simple posible:** Debe evitarse introducir más elementos de los absolutamente necesarios.
- **Que los elementos del minimundo sean representados por el tipo de elemento adecuado:** Además de las variantes mencionadas sobre atributos/entidades/relaciones, también pueden presentarse dudas en relación a entidades débiles, generalizaciones, etc.

3. DISEÑO LÓGICO

A partir del diagrama o modelo conceptual elaborado en la fase anterior, se aplican las reglas de conversión (ver libros sugeridos) y así llegar al diseño lógico (estructura de las tablas). Esta fase es independiente del SGBD relacional donde se implemente.

4. DISEÑO FÍSICO

Conceptualmente, el diseño físico de una base de datos es un proceso que, a partir del diseño lógico y de información sobre su uso esperado (consultas y actualizaciones), obtiene una configuración física (interna) adaptada al entorno donde se mantendrá la base de datos. Esa estructura interna debe permitir el almacenamiento y la explotación (consultas y actualizaciones) de los datos con un rendimiento adecuado.

El diseño físico de una base de datos requiere un conocimiento profundo del Sistema de Gestión de Base de Datos (SGBD o DBMS en inglés) donde vaya a ser implementado. Algunos de los ítems que se deberán conocer son los siguientes:

- Tipos de restricciones de integridad disponibles

- Soporte de integridad referencial
- Tipos de índices disponibles
- Tipos de datos disponibles
- Parámetros de configuración que puedan afectar al rendimiento, como por ejemplo, tamaño de página, porcentaje de espacio libre por página, etc., además de la gestión de datos y de concurrencia que utiliza.
- Particularidades del lenguaje SQL utilizado (dialecto de SQL)

Los elementos esenciales ofrecidos por los SGBDs para ajustar el modelo físico de una base de datos son:

1.1. Espacios para tablas

Un **espacio para tablas**, *tablespace* en inglés, es un componente del SGBD que indica dónde se almacenarán los datos de la base de datos y en qué formato.

El espacio para tablas permite asociar un fichero físico a un conjunto de elementos de la base de datos (tablas, índices, etc.). Dicho fichero contendrá los datos correspondientes a los elementos de la base de datos asociados.

El número de espacios para tablas a crear en una base de datos dependerá de las necesidades del diseño físico. Los espacios para tablas podrán ser simples cuando afecten a un solo elemento de la base de datos, o compuestos cuando afecten a distintos elementos de la base de datos. Además de indicar el fichero donde se almacenarán los datos, los espacios para tablas también permiten definir cómo será dicho fichero: espacio asignado inicialmente, espacio incremental cuando el espacio asignado se agota, memoria intermedia disponible, si se utilizarán técnicas de compresión al almacenar los datos, cómo gestionar el espacio libre del fichero, etc.

1.2. Índices.

Los **índices** son estructuras de datos que permiten mejorar el tiempo de respuesta de las consultas sobre los datos de una tabla. De manera intuitiva, se puede establecer un paralelismo entre los índices usados en una base de datos con los índices de conceptos clave que podemos encontrar en la mayoría de libros de texto. La idea es organizar (de manera alfabética en el caso del índice del libro) una serie de valores de interés (los conceptos clave elegidos en el caso del libro), conjuntamente con las páginas físicas (las páginas del libro) que contienen datos sobre el valor que está siendo indexado (en el caso del libro, se indican las páginas del libro que tratan sobre cada concepto clave que se decide indexar).

Los índices mantienen los valores de una o más columnas de una tabla en una estructura que permite el acceso a las filas de la tabla. Cada posible valor *v* tiene correspondencia con la dirección (o direcciones) física que contiene las filas de la tabla que tienen a *v* como valor de la columna indexada. Esta estructuración de los datos permite un acceso rápido a los datos cuando se realizan búsquedas por valor o cuando se requiere la ordenación de las filas de una tabla de acuerdo a los valores de la columna indexada.

En caso de no disponer de índices, cualquier consulta sobre una tabla de la base de datos requerirá, en el caso peor, un recorrido completo del contenido de la tabla.

Los índices se pueden utilizar también como sistema de control de las restricciones de integridad de unicidad (claves primarias y alternativas), definiendo un índice único sobre las columnas con dicha restricción. De manera similar, también sirven para garantizar las restricciones asociadas a las claves foráneas.

A pesar de que el uso de índices mejora el rendimiento de la consulta de datos de una base de datos, es necesario tener en cuenta que tienen asociado un coste de mantenimiento ante cambios en los datos. En consecuencia, es necesario estudiar cuidadosamente cuántos y qué índices crear. Existen distintos tipos de índice, como por ejemplo, los índices basados en árboles B+, en mapas de *bits*, en técnicas de dispersión (*hash* en inglés), en árboles R, etc. El índice más utilizado es el basado en árboles B+, ya que funciona relativamente bien en todo tipo de situaciones. No obstante, no hay un índice universal que funcione bien en todos los casos. En función de cada caso se deberá escoger entre un tipo de índice u otro, y con una configuración adecuada.

1.3. Vistas materializadas

Los resultados de las consultas sobre una o más tablas se pueden almacenar en **vistas materializadas**. A diferencia de las vistas convencionales, el conjunto de filas que conforma el contenido de una vista materializada se almacena físicamente en la base de datos, como las filas que conforman el contenido de una tabla. Este tipo de vistas pueden ser muy útiles para mejorar el rendimiento de consultas que se ejecutan repetidamente o de consultas basadas en datos

agregados. En ambos casos, el uso de una vista materializada permite calcular la información a priori, ahorrándonos así calcular el contenido asociado a la vista (o a parte de ella) cada vez que el usuario lo solicite.

El tiempo de respuesta de la base de datos puede disminuir significativamente cuando se implementan las vistas materializadas adecuadas. No obstante, en su definición es necesario tener en cuenta sus costes: requiere espacio extra para almacenar el contenido asociado a las vistas materializadas y requiere mantener actualizado su contenido. Este último hecho, por ejemplo, desaconseja el uso de las vistas materializadas en casos donde los datos de origen tengan una frecuencia de actualización alta.

1.4. Particiones

Las **particiones** permiten distribuir los datos de una tabla en distintos espacios físicos. Las particiones se pueden hacer de acuerdo a multitud de criterios: distribuyendo las filas de la tabla en función de los valores de una columna, o de un conjunto de ellas (fragmentación horizontal), distribuyendo los datos de las filas de una tabla entre distintos espacios en función de las columnas a las que pertenecen (fragmentación vertical), e incluso agrupando datos con características comunes, como es el caso de la distribución de los datos agrupados por dimensiones en los *data warehouse*.

Un buen diseño de las particiones permite reducir la carga de trabajo de los componentes hardware del sistema. Los motivos son que el particionamiento de tablas facilita que diferentes transacciones se ejecuten concurrentemente, incrementando así el rendimiento del SGBD y que las consultas a tablas de gran tamaño se puedan resolver mediante consultas más simples que se ejecutan de forma concurrente. Adicionalmente, en el caso de que la base de datos esté distribuida, el particionamiento permite acercar y adecuar los datos a las necesidades de los usuarios/aplicaciones, fomentando el paralelismo y disminuyendo el tráfico de datos a través de la red de comunicaciones. También permite distribuir los datos en dispositivos de almacenamiento más o menos rápidos según la importancia de los datos.

El diseño físico es la implementación en un SGBD relacional concreto, en este caso, se considerará el SGBD relacional PostgreSQL.

https://wiki.postgresql.org/wiki/FAQ/es#Cuestiones_Operativas

BIBLIOGRAFÍA

- Fundamentals of Database Systems (7ma edición, 2016), de los autores Elmasri y Navathe.
- Database Systems: The Complete Book (2da edición, 2008), de los autores Garcia-Molina, Ullman, Widom.
- Database System Concepts (7ma edición, 2019), de los autores Silberschatz, Korth y Sudarshan.
- https://wiki.postgresql.org/wiki/FAQ/es#Cuestiones_Operativas
- Diseño Fisico de Bases de Datos, de los autores Conesa y Rodriguez.
- https://runebook.dev/es/docs/postgresql/indexes-types
- https://programmerclick.com/article/37201575048/