Contenido

MODELO ENTIDAD/RELACIÓN EXTENDIDO

INTRODUCCIÓN

Este primer tema del módulo se centra en el análisis de los datos y de los procesos de negocio de un sistema de información, evidentemente desde un enfoque estructurado.
Por un lado, el objetivo de la modelización de datos es el conocimiento profundo de los datos que va a manejar el sistema de información (en adelante SI) objeto de estudio, representado por un modelo de datos, que permite obtener estructuras de datos no redundantes, sin inconsistencias, seguras e íntegras. Al modelo que surge como primera aproximación de ese mundo real se le llama esquema conceptual.

Comenzamos definiendo un modelo de datos como una representación gráfica orientada a la obtención de estructuras de datos de una forma metódica, es decir, se puede considerar un instrumento que nos facilita la representación de las necesidades del usuario.
El modelo de datos se suele representar mediante el Modelo Entidad-Relación Extendido, creado por Peter Chen en 1_976.
Por tanto, en el desarrollo de SI siguiendo un enfoque estructurado el Modelo E-R Extendido es el más utilizado en el campo del diseño de bases de datos.
Así pues, el Modelo Entidad-Relación Extendido se trata de una técnica cuyo objetivo es la representación y definición de todos los datos que se introducen, almacenan, transforman y producen dentro de un sistema de información, sin tener en cuenta las necesidades de la tecnología existente, ni otras restricciones.
Dado que el modelo de datos es un medio para comunicar el significado de los datos, las relaciones entre ellos y las reglas de negocio de un sistema de información, una organización puede obtener numerosos beneficios de la aplicación de esta técnica, pues la definición de los datos y la manera en que éstos operan son compartidos por todos los usuarios.
Las ventajas de realizar un modelo de datos son, entre otras:

1. Comprensión de los datos de una organización y del funcionamiento de la organización.
2. Obtención de estructuras de datos independientes del entorno físico.
3. Control de los posibles errores desde el principio, o al menos, darse cuenta de las deficiencias lo antes posible.
4. Mejora del mantenimiento.

Aunque la estructura de datos puede ser cambiante y dinámica, normalmente es mucho más estable que la estructura de procesos. Como resultado, una estructura de datos estable e integrada proporciona datos consistentes que puedan ser fácilmente accesibles según las necesidades de los usuarios, de manera que, aunque se produzcan cambios organizativos, los datos permanecerán estables.
Este diagrama se centra en los datos, independientemente del procesamiento que los transforma y sin entrar en consideraciones de eficiencia. Por ello, es independiente del entorno físico y debe ser una fiel representación del sistema de información objeto

del estudio, proporcionando a los usuarios toda la información que necesiten y en la forma en que la necesiten.
El modelo Entidad-Relación Extendido describe con un alto nivel de abstracción la distribución de datos almacenados en un sistema, existiendo dos elementos principales: las ENTIDADES y las RELACIONES. Las extensiones al modelo básico añaden además los atributos de las entidades y la jerarquía entre éstas. Estas extensiones tienen como finalidad aportar al modelo una mayor capacidad expresiva.
Antes de detallar este modelo, existen otras técnicas para crear modelos conceptuales de base de datos, entre las que destacamos:
- Diagramas ORM.
- Diagramas IDEF1X.
- Diagramas UML.
- Diagramas CASE*Method.

1. Entidad

Es aquel objeto, real o abstracto, acerca del cual se desea almacenar información en la base de datos. La estructura genérica de un conjunto de entidades con las mismas características se denomina tipo de entidad.
Existen dos clases de entidades:
1. Regulares: tienen existencia por sí mismas.
2. Débiles: cuya existencia depende de otra entidad.

Las entidades deben cumplir las siguientes tres reglas:
- Tienen que tener existencia propia.
- Cada ocurrencia de un tipo de entidad debe poder distinguirse de las demás.
- Todas las ocurrencias de un tipo de entidad deben tener los mismos atributos.

Asociado al concepto de entidad surge el concepto de **ocurrencia de entidad**. Una ocurrencia de entidad es una realización concreta de una entidad. Por ejemplo, si LIBROS es un tipo de entidad, una ocurrencia de la misma es "UML".

Notación

La representación gráfica de un tipo de entidad regular es un rectángulo etiquetado con el nombre del tipo de entidad. Un tipo de entidad débil se representa con dos rectángulos concéntricos con su nombre en el interior.

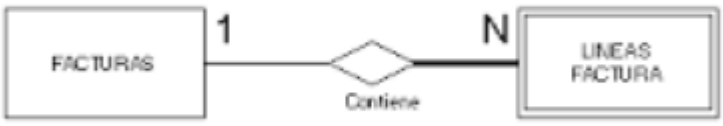


Ilustración 1- Tipos de entidades

Ejemplo: son tipos de entidades regulares libros, facturas, cuentas, clientes, ciudadanos,ciudades, municipios.
Ejemplo: son tipos de entidades débiles ejemplar, línea de factura, movimientos. Así, la existencia de un ejemplar depende un libro, la de una línea de factura de una factura y la de un movimiento de una cuenta.

2. Atributo

Es una propiedad o característica de un tipo de entidad.
Se trata de la unidad básica de información que sirve para identificar o describir la entidad.
Un atributo se define sobre un dominio.
Por tanto, un atributo es una propiedad común a todas las ocurrencias de una entidad.
Ejemplo: son atributos nombre, edad, DNI o fecha de alta.
Cada tipo de entidad ha de tener un conjunto mínimo de atributos que identifiquen unívocamente cada ocurrencia del tipo de entidad. Este atributo o atributos se denomina **identificador principal** o **clave primaria**.
Supongamos que existen varios conjuntos de atributos que identifiquen unívocamente cada ocurrencia del tipo de entidad. Solo uno será el identificador principal o clave primaria. El resto de los conjuntos de atributos se denominarán **claves candidatas** o alternativas.
También ha de conocerse el concepto de **superclave**, que es un conjunto de atributos (no mínimo) que permite distinguir unívocamente cada ocurrencia del tipo de entidad. Si otro atributo unido al anterior subconjunto, el resultado seguirá siendo una superclave.
Claves candidatas ⊂ Superclaves.
Ejemplo: dado el tipo de entidad CONTRIBUYENTE(DNI, nombre, apellidos, dirección, nacionalidad), podemos concluir que:
Nombre no es una clave
Apellidos no es una clave
(nombre, apellidos) no es una clave
DNI es una clave candidata -> DNI es clave primaria
(DNI, nombre) es una superclave
(DNI, dirección) es una superclave
Se pueden definir restricciones sobre los atributos, según las cuales un atributo puede ser:

1. **Simple**: son atributos que no están divididos en partes, es decir, representan un valor indivisible.
2. **Compuesto**: atributo que puede ser subdividido en atributos más elementales.
   Ejemplo: atributo Dirección à vía, nombre, ciudad, provincia y código postal.
3. **Univaluado**: atributo que sólo puede tomar un valor para todas y cada una de las ocurrencias del tipo de entidad al que pertenece.
4. **Multivaluado**: atributo que puede tomar más de un valor para algunas de las ocurrencias del tipo de entidad al que pertenece.
   Ejemplo: atributo teléfono puede tomar más de un valor a la vez.
5. **Obligatorio**: atributo que tiene que tomar al menos un valor para todas y cada una de las ocurrencias del tipo de entidad al que pertenece.
6. **Derivado**: atributo cuyo valor se obtiene a partir de los valores de otros atributos de la misma o de diferente tipo de entidad.

Ejemplo: atributo edad deriva del atributo fecha de nacimiento.

Notación

Un atributo se representa mediante una elipse, con su nombre dentro, conectada por una línea al tipo de entidad o relación.

En lugar de una elipse puede utilizarse un círculo con el nombre dentro, o un círculo más pequeño con el nombre del atributo a un lado. También pueden representarse en una lista asociada a la entidad. El identificador aparece con el nombre marcado o subrayado o bien con su círculo en negro.

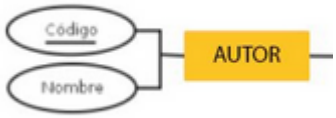


Ilustración 2- Entidad y atributos

RELACION

Es una asociación o correspondencia existente entre una o varias entidades

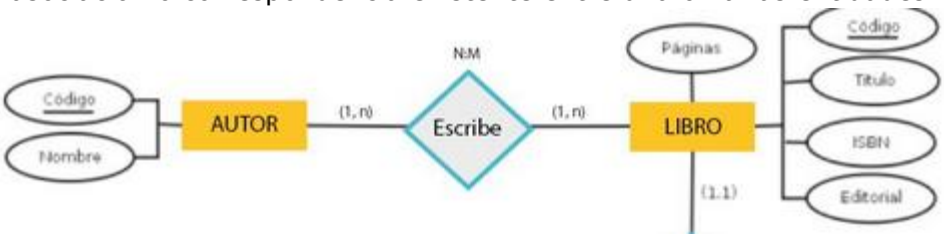


Ilustración 3- Relaciones

Una ocurrencia de la relación es la asociación concreta de ocurrencias de entidad de diferentes entidades. Así, por ejemplo, si tenemos las entidades EMPLEADO y DEPARTAMENTO, y la relación Trabaja en, una ocurrencia será "Pepe" trabaja en el "Departamento de Informática".

La relación puede ser:

1. **Regular**: si asocia tipos de entidad regulares.
2. **Débil**: si asocia un tipo de entidad débil con un tipo de entidad regular. Dentro de las relaciones débiles se distinguen:
    a. **La dependencia en existencia**: cuando las ocurrencias de un tipo de entidad débil no pueden existir sin la ocurrencia de la entidad regular de la que dependen.
    b. **La dependencia en identificación**: cuando, además de lo anterior, las ocurrencias del tipo de entidad débil no se pueden identificar sólo mediante sus propios atributos, sino que se les tiene que añadir el identificador de la ocurrencia de la entidad regular de la cual dependen.
    Clave primaria entidad débil = clave primaria entidad fuerte + clave entidad débil

Una relación se caracteriza por:

1. Nombre
2. Grado
3. Tipo de correspondencia
4. Cardinalidad

Nombre: que lo distingue unívocamente del resto de relaciones del modelo.

GRADO

GRADO: número de tipos de entidad sobre las que se establece la relación. En función del grado las relaciones se clasifican en:

- **Unarias**: una entidad se relaciona consigo misma (relaciones reflexivas). GRADO 1
- **Binarias**: entidades relacionadas dos a dos. GRADO 2
- **Ternarias**: relación entre tres entidades. GRADO 3
- **N-arias**: relación entre n entidades. GRADO 4

TIPO DE CORRESPONDENCIA

Tipo de correspondencia (o razón de cardinalidad): es el número máximo de ocurrencias de cada tipo de entidad que pueden intervenir en una ocurrencia de la relación que se está tratando.
Conceptualmente se pueden identificar tres clases de relaciones:

Relaciones 1:1 (uno a uno): cada ocurrencia de una entidad se relaciona con una y sólo una ocurrencia de la otra entidad.
Relaciones 1:N (uno a muchos): cada ocurrencia de una entidad puede estar relacionada con cero, una o varias ocurrencias de la otra entidad.
Relaciones M:N (muchos a muchos): cada ocurrencia de una entidad puede estar relacionada con cero, una o varias ocurrencias de la otra entidad y cada ocurrencia de la otra entidad puede corresponder a cero, una o varias ocurrencias de la primera.

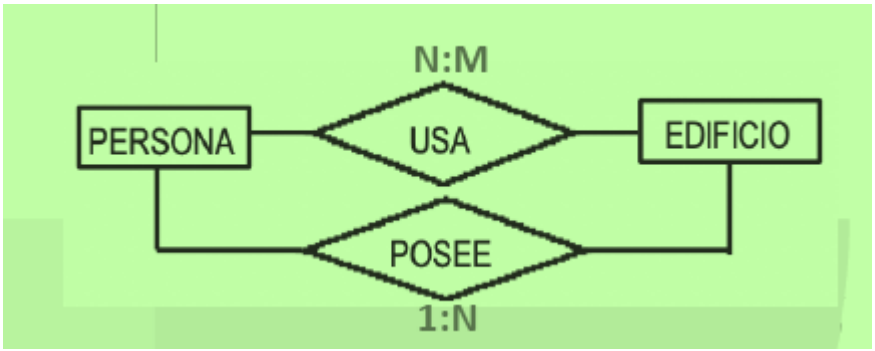


Ilustración 4- Tipos de correspondencia

CARDINALIDAD

Representa la participación en la relación de cada una de las entidades afectadas, es decir, el número máximo y mínimo de ocurrencias de un tipo de entidad que pueden

estar interrelacionadas con una ocurrencia de otro tipo de entidad (min, max). La cardinalidad máxima coincide con el tipo de correspondencia. MAX= 1, n
Según la cardinalidad, una relación es:

- Obligatoria: cuando para toda ocurrencia de un tipo de entidad existe al menos una ocurrencia del tipo de entidad asociado. (1, max)
- Opcional: cuando, para toda ocurrencia de un tipo de entidad, puede existir o no una varias ocurrencias del tipo de entidad asociado. (0, max)

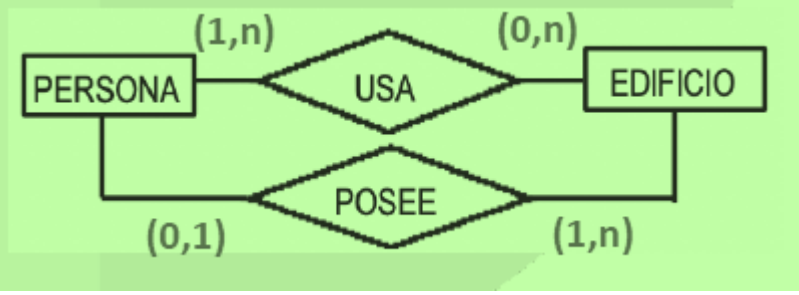


Ilustración 5- Cardinalidad

Las relaciones pueden tener atributos propios, de forma similar que los atributos de los tipos de entidad

Notación

Se representa por un **rombo** unido a las entidades relacionadas por dos líneas rectas a los lados. El tipo de correspondencia se representa gráficamente con una etiqueta **1:1**, **1:N** o **M:N**, cerca de alguno de los vértices del rombo, o bien situando cada número o letra cerca de la entidad correspondiente, para mayor claridad.

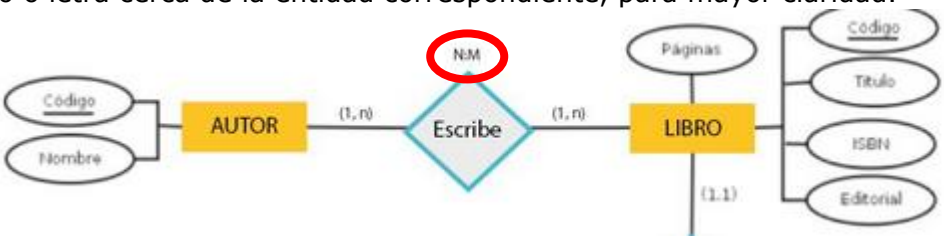


Ilustración 6- Tipo de correspondencia

La representación gráfica de las cardinalidades se realiza mediante una etiqueta del tipo **(0,1)**, **(1,1)**, **(0,n)** o **(1,n)**, que se coloca en el extremo de la entidad que corresponda. Si se representan las cardinalidades, la representación del tipo de correspondencia es redundante.

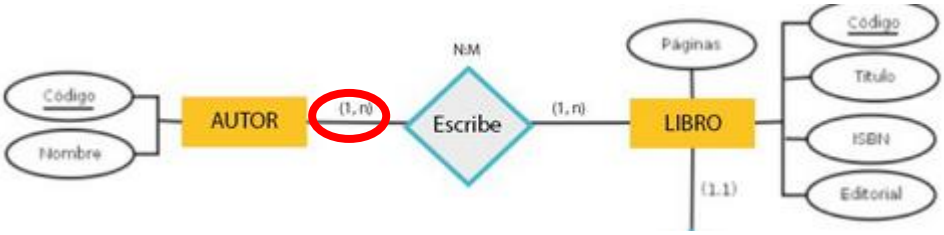


Ilustración 7- Cardinalidad

Las relaciones reflexivas se consideran un caso particular, en las que una entidad se relaciona consigo misma. Cabe resaltar el concepto de **ROL o papel** de la entidad (extensible a todo tipo de relaciones pero que resulta agravado en las relaciones reflexivas), esto es, el significado que tiene la participación de las ocurrencias en la relación. El rol se especifica en los extremos de la relación.

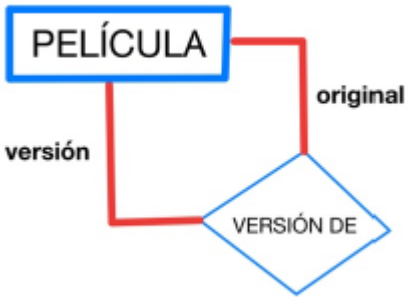


Ilustración 8- relación reflexiva

Para relaciones reflexivas el rol de la entidad debe especificarse, en el resto se considera opcional.
Ejemplo: en la siguiente relación entre ocurrencias del tipo de entidad EMPLEADO, conviene distinguir a los empleados que asumirán el rol de "JEFE" y a los empleados que adoptarán el papel de "SUBORDINADO".

Ilustración 9- Rol

En la representación de las **relaciones exclusivas** se incluye un arco sobre las líneas que conectan el tipo de entidad a los dos o más tipos de relación.

En la representación de las **relaciones exclusivas** se incluye un arco sobre las líneas que conectan el tipo de entidad a los dos o más tipos de relación.

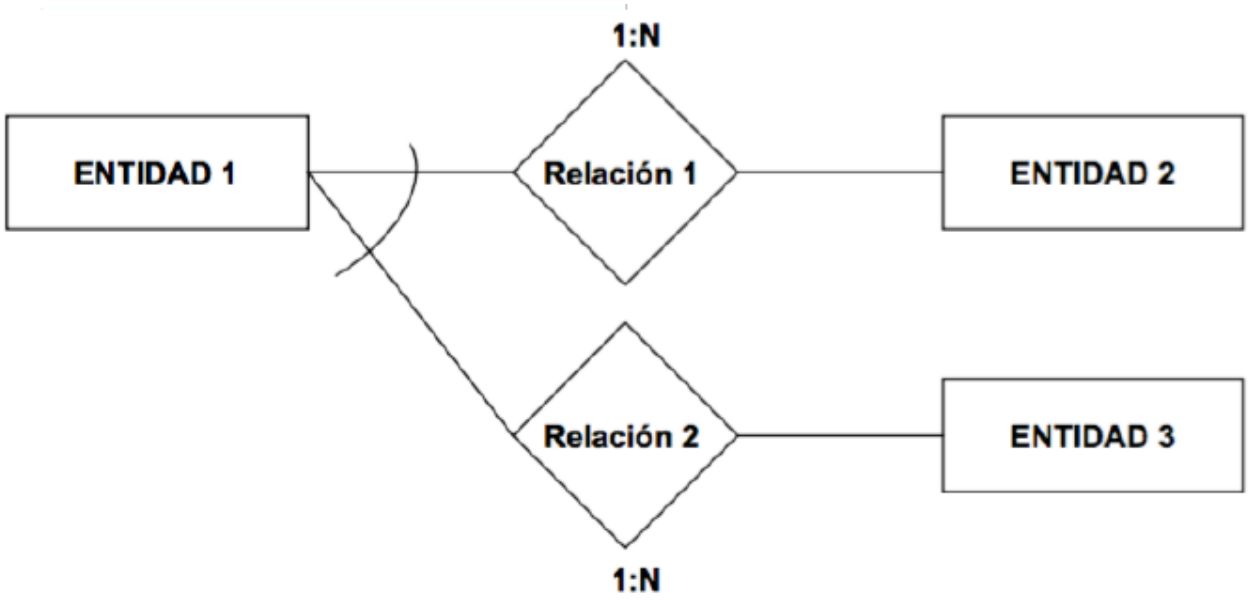


Ilustración 10- relación exclusivas

Ejemplo: dados los tipos de entidad PERSONA y EDIFICIO y las relaciones USA y POSEE entre ambos tipos de entidad, tenemos:

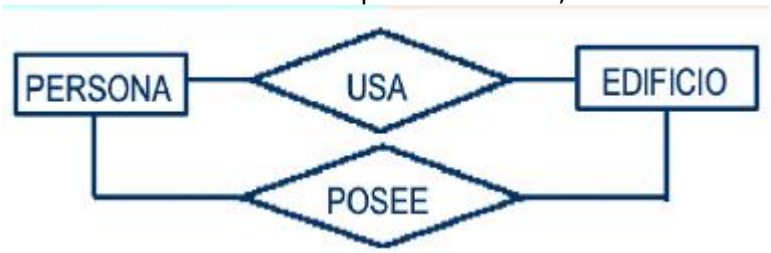


Ilustración 11- relación excluisiva

Una persona usa uno o más edificios y un edificio puede ser usado (o no) por una o varias personas. Por otro lado, una persona posee uno o más edificios y un edificio es poseído por una persona.
Entonces, el tipo de correspondencia y la cardinalidad es la que se presenta a continuación (se advierte que por claridad expositiva se van a representar ambas, pero se reitera que la cardinalidad incluye el tipo de correspondencia)
Tipo de Correspondencia:

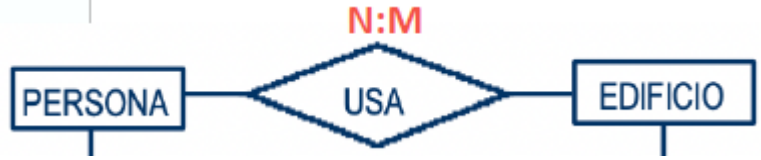


Ilustración 12- Tipo de correspondencia

Cardinalidad

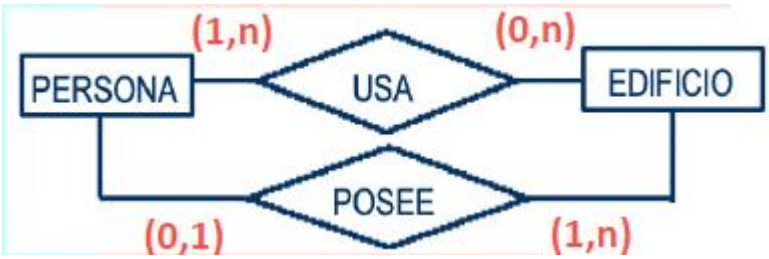


Ilustración 13- cardinalidad

Dominio

Es un conjunto nominado de valores homogéneos. El dominio tiene existencia propia con independencia de cualquier entidad, relación o atributo.
Así pues, una cierta característica o propiedad (atributo) de un objeto toma valores que pertenecen a un determinado dominio.

EXTENSIONES DEL MODELO ENTIDAD-RELACION

Además de estos elementos, existen extensiones del modelo entidad/relación que incorporan determinados conceptos o mecanismos de abstracción para facilitar la representación de ciertas estructuras del mundo real:
La GENERALIZACIÓN permite abstraer un tipo de entidad de nivel superior (supertipo) a partir de varios tipos de entidad (subtipos); en estos casos los atributos comunes y relaciones de los subtipos se asignan al supertipo. Se pueden generalizar por ejemplo los tipos profesor y estudiante obteniendo el supertipo persona.
La ESPECIALIZACIÓN es la operación inversa a la generalización, en ella un supertipo se descompone en uno o varios subtipos, los cuales heredan (automáticamente) todos los atributos y relaciones del supertipo, además de tener los suyos propios. Se denominan relaciones "es un".
CATEGORÍAS. Se denomina categoría al subtipo que aparece como resultado de la unión de varios tipos de entidad. En este caso, hay varios supertipos y un sólo subtipo. Si por ejemplo se tienen los tipos persona y compañía y es necesario establecer una relación con vehículo, se puede crear propietario como un subtipo unión de los dos primeros.
La **AGREGACIÓN** consiste en construir un nuevo tipo de entidad como composición de otros y su tipo de relación y así poder manejarlo en un nivel de abstracción mayor.
La ASOCIACIÓN, consiste en relacionar dos tipos de entidades que normalmente son de dominios independientes, pero coyunturalmente se asocian.
La existencia de supertipos y subtipos, en uno o varios niveles, da lugar a una **jerarquía**, que permitirá representar una restricción del mundo real.

NOTACIÓN

La representación de las jerarquías se realiza mediante un triángulo invertido, con la base paralela al rectángulo que representa el supertipo y conectando a éste y a los subtipos. Si la división en subtipos viene determinada en función de los valores de un atributo discriminante, éste se representará asociado al triángulo que representa la relación.

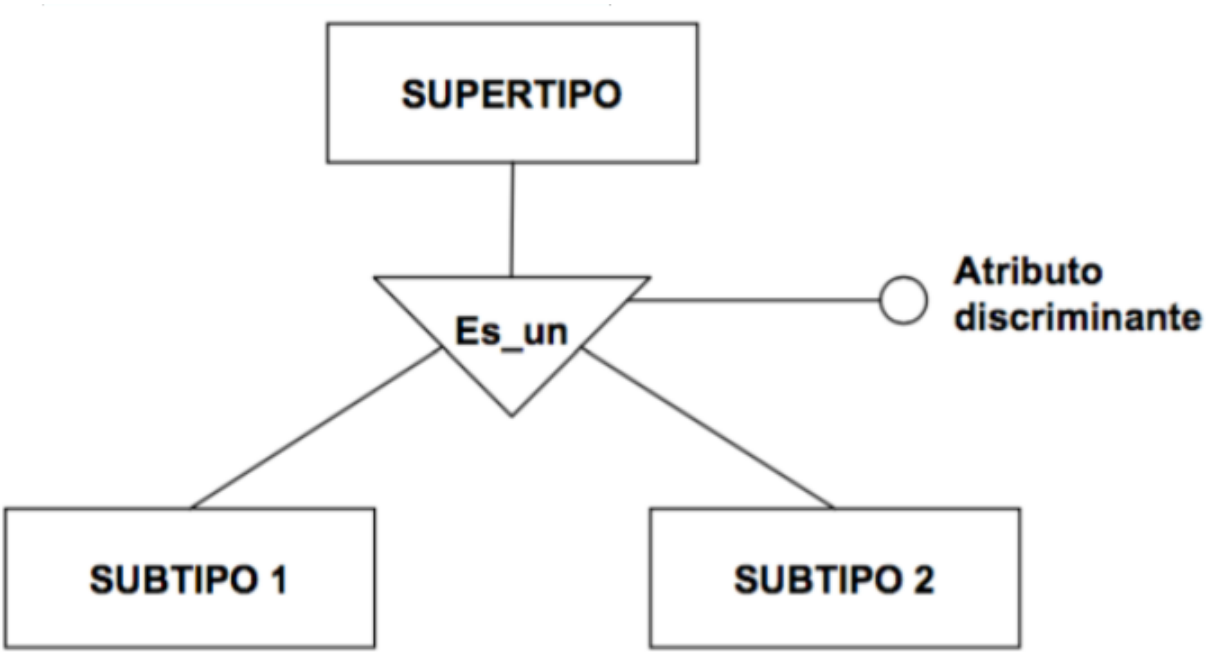


Ilustración 14- relación reflexiva

En el triángulo se representará:
- Con una letra **d** el hecho de que los subtipos sean **disjuntos**.
- Con un círculo o una **O** si los subtipos pueden **solaparse**.
- Con una **U** el caso de **uniones por categorías**.
- La presencia de una **jerarquía total** se representa con una **doble línea** entre el supertipo y el triángulo.

La generalización y especialización pueden ser:

Jerarquía TOTAL: toda ocurrencia del supertipo pertenece siempre a algún subtipo.
Jerarquía PARCIAL: una ocurrencia del supertipo no pertenece a algún subtipo.
Jerarquía DISJUNTA: una ocurrencia del supertipo solo puede pertenecer a uno de los subtipos.
Jerarquía SOLAPADA: una ocurrencia del supertipo puede pertenecer a varios subtipos.

A partir de ahí, se puede mezclar cualquiera de las opciones {total | parcial} con cualquier otra de las siguientes {disjunta | solapada}.

En el triángulo se representará:
- Con una letra d el hecho de que los subtipos sean disjuntos.
- Con un círculo o una O si los subtipos pueden solaparse.
- Con una U el caso de uniones por categorías.

La presencia de una jerarquía total se representa con una doble línea entre el supertipo y el triángulo.

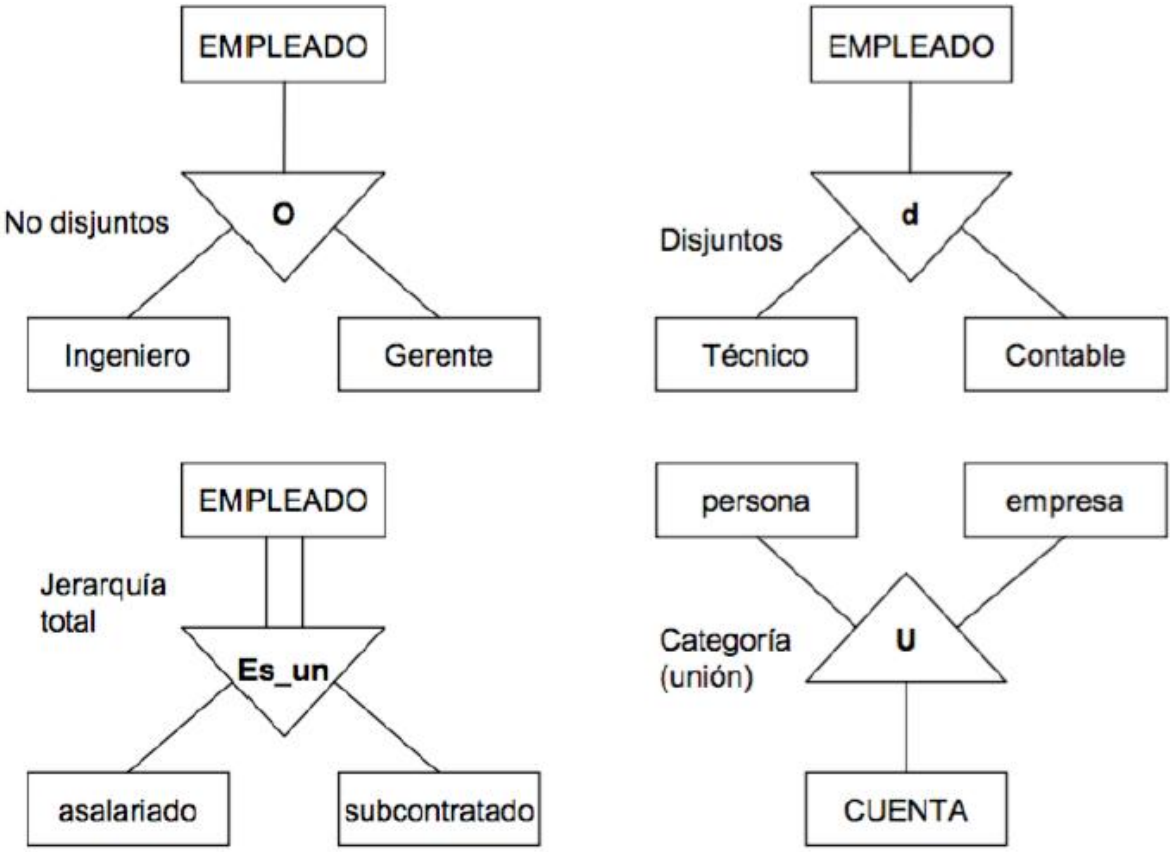


Ilustración 15- Entidad relación extendido

EJEMPLO

Modelo entidad-relación extendido para un sistema de gestión de técnicos y su asignación a proyectos dentro de una empresa u organización.
Como se aprecia en el diagrama, TÉCNICO es un subtipo de EMPLEADO, generado por especialización, pues era necesario para establecer la relación Trabaja en con PROYECTO, ya que no todos los empleados de la empresa, como los administrativos, son susceptibles de trabajar en un proyecto. La entidad TÉCNICO tendrá los atributos de EMPLEADO más el atributo nivel.

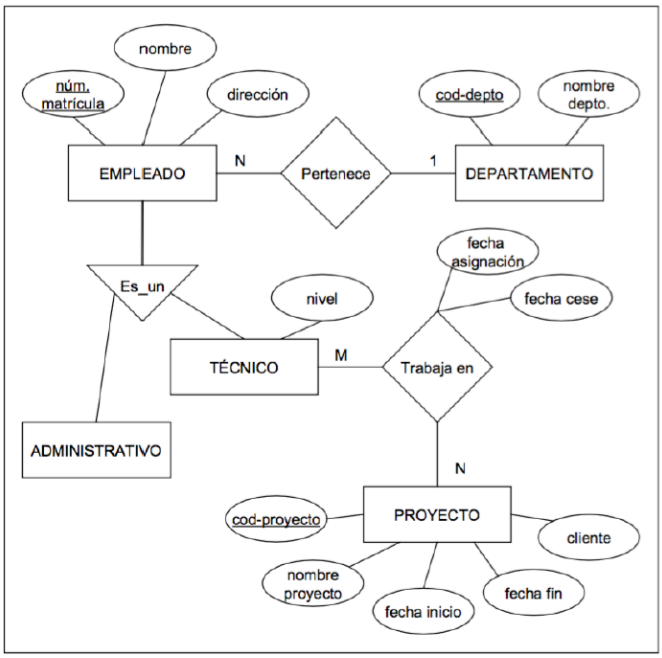


Ilustración 16 - Ejemplo

CONSTRUCCIÓN Y VALIDACIÓN DEL MODELO ER

Podemos establecer para la construcción del modelo conceptual de datos las siguientes fases:

1. Identificar las entidades del SI objeto de desarrollo.

Para identificar los tipos de entidad sirve de ayuda clasificar la información en:

- Objetos reales: máquinas, edificios, almacenes.
- Personas: empleados, funcionarios, clientes, proveedores.
- Actividades del sistema: licencias, facturas, albaranes, expedientes, documentos.
- Objetos abstractos: categorías de personal, departamentos.

2. Determinar los identificadores principales de las entidades.

Para determinarlos, se obtendrán aquellos atributos que identifiquen unívocamente cada ocurrencia de cada tipo de entidad. Si para un tipo de entidad hubiera varios, se elegirá solo uno. Aunque resulta obvio, el conjunto de atributos que forman el identificador principal no pueden tomar valores sin información (nulos).

3. Establecer las relaciones entre las entidades y el grado de las mismas.

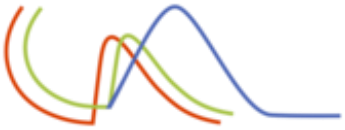
Para establecer las asociaciones o relaciones entre entidades, se estudia cada una de las relaciones de una entidad con las demás entidades identificadas, para comprobar si dichas relaciones tienen sentido e importancia para el sistema que se está desarrollando.

Del conjunto de relaciones se estudiará su cardinalidad y la posibilidad de relaciones exclusivas.

4. Dibujar el modelo de datos.

Dibujar el diagrama según la notación descrita.

5. Identificar y describir los atributos de cada entidad.



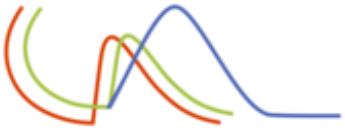
Para identificar los atributos de cada entidad, habrá que tener en cuenta todas aquellas propiedades de cada entidad en las que el sistema tenga interés. Luego se incorporarán al diagrama.

6. Verificaciones.

Una vez construido el modelo Entidad-Relación, hay que analizar si se presentan **redundancias**. Para poder asegurar su existencia se deben estudiar con mucho detenimiento las cardinalidades mínimas de las entidades, así como la semántica de las relaciones.

Los atributos redundantes, los que se derivan de otros elementos mediante algún cálculo, deben ser eliminados del modelo Entidad-Relación o marcarse como redundantes.

Igualmente, las relaciones redundantes deben eliminarse del modelo, comprobando que al eliminarlas sigue siendo posible el paso, tanto en un sentido como en el inverso, entre las dos entidades que unían.



MODELO RELACIONAL

DISEÑO DE BASES DE DATOS

Las bases de datos son parte integral de cualquier sistema de información. Por tanto, una de las características que deben presentar es la capacidad de adaptación a cambios en el entorno. Estos cambios, inevitables en cualquier organización, pueden ser físicos (cambios en el hardware, en el formato de los ficheros, etc.) o lógicos (cambios en los programas, en el lenguaje de programación, etc.).
No es deseable que un cambio en el formato de archivo de los datos obligue a rehacer la aplicación que accede a los mismos. Es necesario, pues, independizar la estructura lógica de los datos de la forma en que estos se guardan físicamente. Así, cualquier cambio en uno de los dos niveles será transparente para el otro, facilitando la tarea de mantenimiento, tanto de las aplicaciones como de las bases de datos.

Esta independencia entre el modelo lógico de los datos y su estructura física es la que proporciona la **arquitectura en tres niveles del Instituto Nacional de Estandarización Americano** (ANSI). según este organismo la independencia de los datos es la capacidad de un sistema de gestión de base de datos para permitir que las referencias a los datos a través de los programas estén aisladas de los posibles cambios y diferentes usos que el entorno pueda propiciar, como pueden ser: la forma de almacenamiento, el modo de compartición con otros programas o el modo de organizacion para mejorar el rendimiento del sistema de base de datos.

Independencia de los datos

La arquitectura de tres niveles es útil para explicar el concepto de **independencia de datos** que podemos definir como la capacidad para modificar el esquema en un nivel del sistema sin tener que modificar el esquema del nivel inmediato superior.
Se pueden definir dos tipos de independencia de datos:

1. La **independencia LOGICA** es la capacidad de modificar el esquema conceptual sin tener que alterar los esquemas externos ni los programas de aplicación. Se puede modificar el esquema conceptual para ampliar la base de datos o para reducirla. Si, por ejemplo, se reduce la base de datos eliminando una entidad, los esquemas externos que no se refieran a ella no deberán verse afectados.
2. La **independencia FISICA** es la capacidad de modificar el esquema interno sin tener que alterar el esquema conceptual (o los externos). Por ejemplo, puede ser necesario reorganizar ciertos ficheros físicos con el fin de mejorar el rendimiento de las operaciones de consulta o de actualización de datos. Dado que la independencia física se refiere solo a la separación entre las aplicaciones y las estructuras físicas de almacenamiento, es mas fácil de conseguir que la independencia logica.

Para conseguir este ideal de independencia, ANSI propone que las bases de datos se construyan siguiendo un modelo o arquitectura de tres niveles: **nivel externo**, **nivel conceptual** y **nivel físico**. Estos tres niveles están organizados de forma jerárquica, siendo el nivel físico el más cercano a la maquina y el nivel externo aquel con el que interactua el usuario. Esta arquitectura por capas sirve para aislar al usuario de las

complejidades del modelo de datos y de almacenamiento fisico de la base, ya que solo se le permitirá ver la parte logica de la base necesaria para su trabajo. Además, también aisla el modelo logico de datos (aqui llamado conceptual) de la implementacion especifica que se realice del mismo, por lo que, en teoría, este seria portable de unos gestores de bases de datos a otros. Todo ello dio lugar a la denominada **Arquitectura ANSI/X3/SPARC**.

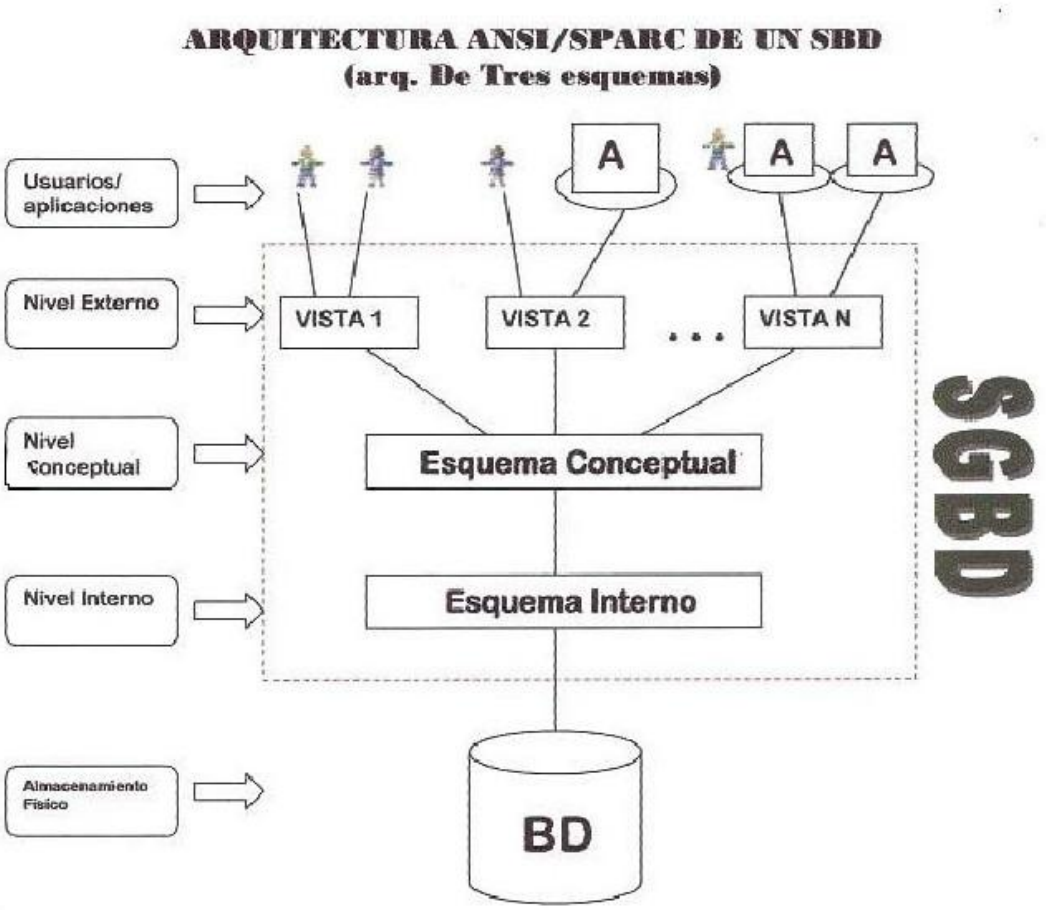


Ilustración 17 – Arquitectura ANSI / SPARC

Nivel LOGICO o EXTERNO

En este nivel se guardan las distintas **vistas parciales** de la base de datos que se muestran a los **usuarios y/o aplicaciones**. Puesto que no todos los programas ni las personas que acceden a la base de datos necesitan tener una visión total de la información guardada en la misma, resulta conveniente crear vistas o esquemas específicos según las necesidades particulares de cada uno.

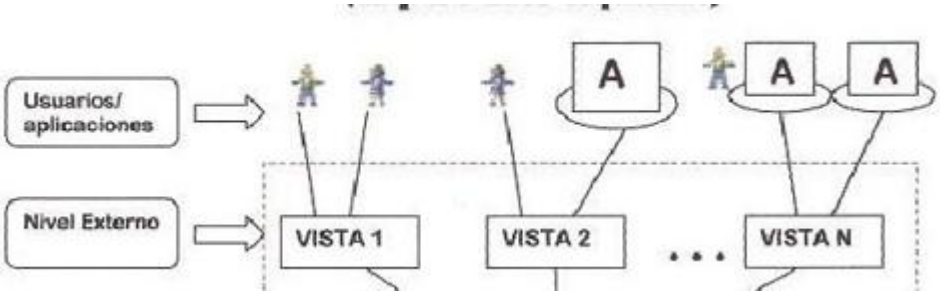


Ilustración 18 - Nivel Lógico

La existencia de este nivel aísla por completo a los usuarios y/o aplicaciones no solo del aspecto físico de la base de datos, sino también de cualquier parte de la misma que no este directamente relacionada con la tarea que deben desempeñar, aumentando así la independencia y la protección frente a cambios en otras áreas de la organización. Adicionalmente, esta capa también aumenta la seguridad de los datos, ya que, como consecuencia de la creación de vistas parciales de la base de datos, los usuarios no disponen de acceso más que a partes seleccionadas de la misma, disminuyendo así la posibilidad de consultar, modificar o borrar datos privilegiados.

A partir del **nivel externo se crea el esquema externo** para cada base de datos en concreto. Los esquemas externos pueden ser múltiples para una misma base de datos y además se puede producir anidamiento entre ellos. El esquema externo queda representado por los datos que de forma efectiva pueda acceder cada usuario y/o aplicación, más los permisos de acceso efectivos sobre los mismos.

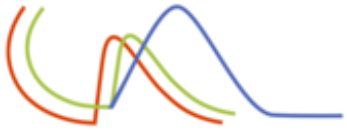
Nivel CONCEPTUAL

Este modelo pretende reflejar la **estructura y relaciones existentes entre los datos** del mundo real que se van a guardar en la base de datos, aislando entre si los niveles externos (vista del usuario) e interno (vista de la maquina).
Para construir el esquema es necesario definir el universo del discurso o, lo que es lo mismo, acotar aquella parte del mundo real que se quiere modelar, incluyendo todos los elementos o características relevantes y excluyendo aquellas que pese a existir en el problema original no son relevantes. Para ello, se utiliza típicamente la técnica de Modelo Entidad-Relacion Extendido.

A partir del **modelo conceptual se obtiene el esquema conceptual** que se identifica mediante las estructuras de datos, relaciones y restricciones.
Dentro de este nivel se incluye también la manera de obtener una adecuación del mismo al entorno de implantación elegido, es decir, el modelo lógico.

El **modelo lógico de datos se define como un conjunto de conceptos, reglas y convenciones que permiten describir un modelo conceptual**. Los modelos de datos comúnmente utilizados son:

1. **Modelo jerárquico**: presenta una estructura en árbol donde nodos y ramas siguen una relación del tipo 1:N.



2. **Modelo Codasyl**: estructura en red donde se establecen relaciones N:M. Es mas flexible que el jerárquico.
3. **Modelo relacional**: presenta estructuras de la teoría matemática de conjuntos (algebra relacional) y/o de la lógica de predicados (calculo relacional). Permite el procesamiento de conjuntos de datos y no simples registros como en los anteriores. Se caracteriza por disponer los datos organizados en tablas (relaciones) que cumplen ciertas restricciones.
4. **Modelo orientado a objetos**. Cada uno de estos modelos tiene sus características únicas que los hacen más adecuados para modelar unos problemas u otros, así mismo, el modelo elegido va a condicionar en gran medida los lenguajes de datos utilizados, ya que la propia estructura del modelo llega a imponer determinadas formas de acceder a los datos.

Nivel INTERNO o FISICO

En este nivel se especifica **que, como y donde se van a almacenar los datos fisicamente**. Se ocupa de tratar con el sistema operativo, con el sistema de ficheros, con los dispositivos de entrada/ salida y, en general, con todos aquellos aspectos de bajo nivel necesarios para almacenar efectivamente la información en el ordenador. El contenido de este nivel depende por completo de la combinación hardware/software que se emplee en cada instalación.
Del **nivel físico se deriva el esquema interno**, que contiene las definiciones de los registros guardados, los métodos de representación, los campos de datos y los índices. Hay un solo esquema interno por base de datos.
Como resumen de esta arquitectura, su propósito principal es que el Esquema Conceptual sea una descripción estable de la organización e independiente de las "vistas" y de la forma de almacenamiento de los datos. Debido a esta independencia de niveles, las Bases de Datos pueden ser flexibles y adaptables a los cambios.

DISEÑO LÓGICO

A partir del esquema conceptual se elabora el modelo lógico. Este modelo debe coincidir con el modelo datos soportado por el SGBD que se vaya a utilizar. Es nuestro caso el modelo de datos es el modelo relacional.
Las técnicas de modelado conceptual son diferentes de las técnicas de modelado lógico, por lo que habrá que convertir cada elemento presente en el modelo conceptual en elementos expresables en la técnica de modelado lógico que hayamos seleccionado. Para ello se realizan una serie de transformaciones y el proceso de normalización.

Ajustes del modelo conceptual

ELIMINACION DE ATRIBUTOS COMPUESTOS

Estos atributos se caracterizan porque se pueden descomponer en una enumeración de atributos simples que expresan de forma más precisa la información a representar.
Ejemplo: el atributo "Codigo_Cuenta_Bancaria" de la entidad "Cuentas Banco" se podría descomponer:

- Codigo_Banco.
- Codigo_Sucursal.

- Numero_de_cuenta.

El resultado es una información más precisa y al mismo tiempo más breve al haberse eliminado el Código de Control al ser posible calcularlo a partir de los demás campos. Esta descomposición de los atributos compuestos en otros simples se realiza, fundamentalmente, para facilitar el acceso a la información y su posterior tratamiento.
Como resultado de esta eliminación se obtiene un Diagrama Entidad-relación en el cual **todas las entidades tienen atributos simples**.

ELIMINACION DE ATRIBUTOS MULTIVALORADOS O MULTIVALUADOS

Los atributos multivalorados son aquellos que pueden tomar más de un valor para una misma ocurrencia de una entidad.

Ilustración 19 - Entidad con atributo multivaluado

Se observa que el atributo Movimientos puede tomar más de un valor, ya que una Cuenta bancaria normalmente tiene más de un movimiento. Este tipo de atributos no son válidos en el modelo relacional, por lo que es necesario eliminarlos.
La solución al problema consiste en crear una nueva entidad que represente al atributo multivalorado y una relación entre esta entidad y la original. En nuestro ejemplo, esto se haría de la siguiente manera:

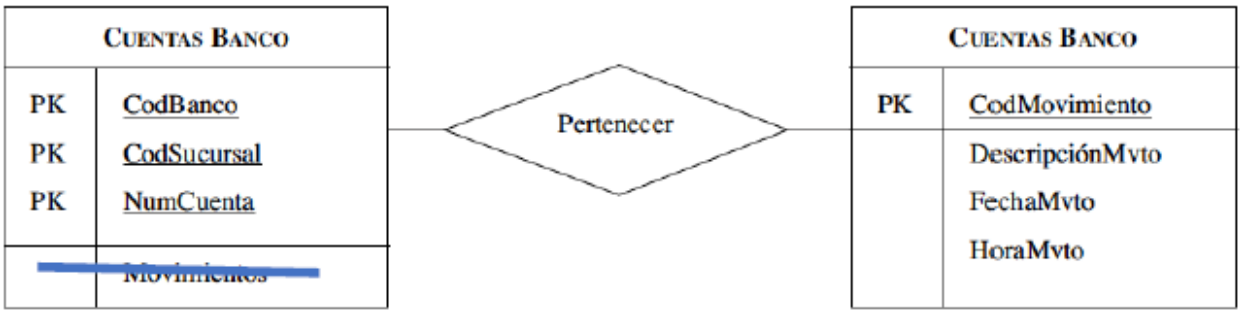


Ilustración 2- Eliminando el atributo multivaluado

Por lo tanto, un movimiento pertenece a una cuenta y una cuenta puede tener múltiples movimientos.
Como resultado de esta eliminación se obtiene un Diagrama Entidad-relación en el cual todas las entidades tienen atributos univalorados.

ELIMINACION DE RELACIONES REDUNDANTES

Una relacion es redundante cuando se puede obtener la misma información que aporta mediante otras relaciones. El hecho de que haya dos caminos diferentes entre dos entidades no implica que uno de los caminos corresponda a una relación redundante, eso dependerá de la carga semántica que aporte cada una de las relaciones representadas.

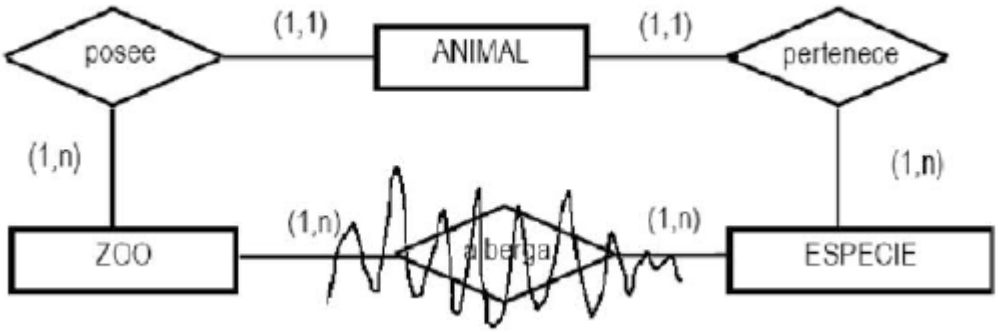


Ilustración 20 - Relación redundante

Se observa que la relación "Alberga" es innecesaria puesto que, en principio, podríamos conocer todas las especies de animales existentes en un zoo a través de la relación entre la entidad "Animal" y "Especies" al existir también una relación entre "Zoo" y "Animal".
Sin embargo, en el siguiente ejemplo no sería normal que se eliminara alguna de las relaciones, aun cuando estén vinculando a las mismas entidades, ya que la carga semántica de cada una de ellas es diferente:

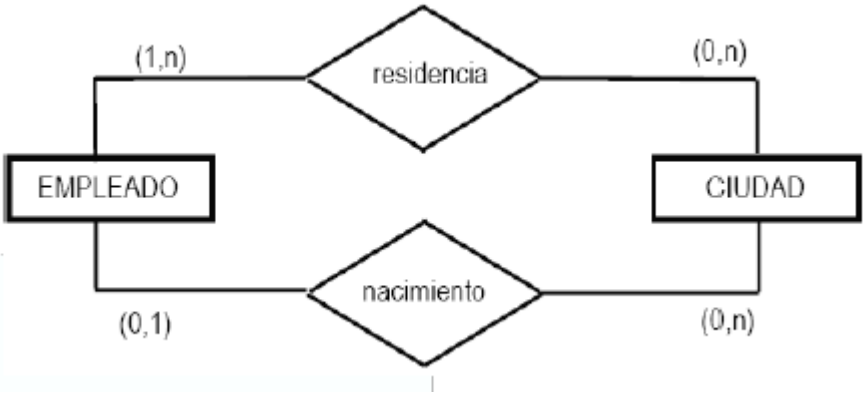


Ilustración 21 - Relaciones NO redundante

OBTENCIÓN DEL MODELO LÓGICO A PARTIR DEL CONCEPTUAL

TRANSFORMACION DE DOMINIOS

Un dominio del modelo ER se transforma en un dominio equivalente del modelo LÓGICO. Para ello, se emplea la sentencia CREATE DOMAIN. En otro caso, será necesario utilizar los tipos primitivos más afines con el dominio representable y delimitar sus elementos mediante restricciones de usuario asociadas a la tabla a la que pertenezca el atributo con tal dominio.

TRANSFORMACION DE ENTIDADES

Cada entidad del modelo conceptual se transforma en una relación o tabla con estructura relacional. La clave primaria de la entidad pasa a ser la clave primaria de la relación. Los atributos que forman parte de la clave no podrán tomar el valor nulo.
Ejemplo: Dado el siguiente diagrama conceptual:

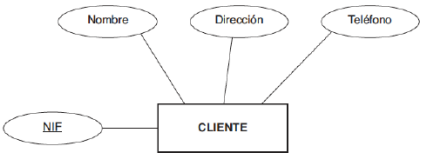


Ilustración 22 – Entidad pasa a ser una relación

Daría como resultado esta relación:
CLIENTE(NIF (PK), Nombre, Dirección, Teléfono)

TRANSFORMACIÓN DE ATRIBUTOS

Cada atributo se transforma en una columna de la tabla en la que se transformó la entidad a la que pertenece. El identificador único se convierte en clave primaria.

- **Claves Primarias o Identificadores**: las claves primarias o identificadores de la entidad pasan a ser claves primarias en la relación resultantes (PRIMARY KEY).
- **Claves candidatas (o alternativas):** se transforman como atributos convencionales, pero en su implementación deberá añadirse la restricción UNIQUE.
- **Atributos convencionales:** se transforman en campos de la relación con el dominio que tuvieran asignado.
- **Atributos compuestos y multivalorados**: se transforman en campos simples.

TRANSFORMACIÓN DE RELACIONES

Relaciones 1:1

Como norma general, es un caso particular de las 1:N y por tanto se propaga la clave en las dos direcciones. Se debe analizar la situación, intentando recoger la mayor semántica posible, y evitar valores nulos.

1) Será necesario crear una nueva tabla si:

a) Las cardinalidades mínimas son cero (ambas), esto evitará valores nulos en las claves ajenas y mantendrá la simetría natural (las entidades mantienen su independencia en tablas separadas)
b) La relación tiene atributos propios.
c) Se prevé que posteriormente puedan variarse las cardinalidades.

Ejemplo:

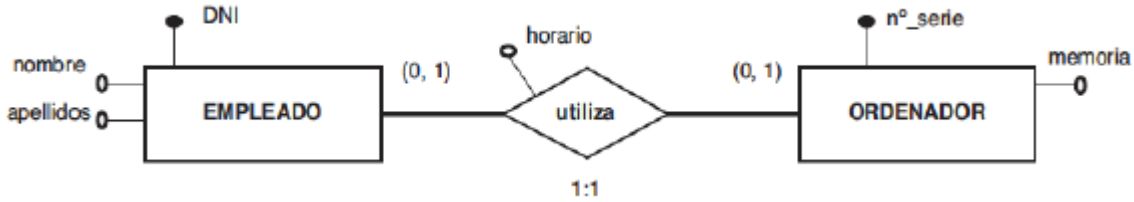


Ilustración 23 - Transformación

EMPLEADO(DNI (PK), Nombre, Apellido)
ORDENADOR(NumSerie (PK), Memoria)
UTILIZA(NumSerie+DNI (PK), horario)

2) **Si una de las cardinalidades minimas es cero (0,1) y la otra no (1,1),** conviene propagar la clave de esta última (la obligatoria) a la primera.

3) **Si las dos entidades participan de forma completa**, es decir, todas cardinalidades mínimas son 1:

a) Si las entidades tienen el mismo identificador se transforman en una única tabla formada por la concatenación de los atributos de los dos tipos de entidad.
b) En el caso de que tengan diferente identificador, cada entidad se transforma en una tabla y se puede propagar la clave de cualquiera de ellas a la tabla resultante de la otra, teniendo en cuenta, en este caso, los accesos más frecuentes y prioritarios a los datos de las tablas.

Relaciones 1:N

Según el caso que nos ocupe:

1) Relaciones entre **entidades fuertes**. Se utiliza el método de propagación de clave. En este caso, el identificador de la entidad con cardinalidad uno cede su clave a la de cardinalidad muchos. En esta última, pasa a ser una clave ajena.
2) Relaciones de **dependencia** (entidad fuerte-entidad débil):
   a) Dependencia por existencia. Igualmente, se utiliza el método de propagación de clave (punto 1).
   b) Dependencia por identificador. Se utiliza el método de propagación de clave, y además la clave primaria de la tabla con cardinalidad muchos estarán formada por la concatenación de su propia clave más la que le cede la de cardinalidad uno.

Relaciones N:M y ternarias

Se crea una tabla como consecuencia de la relación que tendrá como clave primaria la concatenación de los identificadores de las entidades relacionadas. La condición de clave ajena se expresa mediante FOREIGN KEY.

Relaciones de agregación

Las relaciones de agregación se transforman del mismo modo que las 1:N.

Relaciones Reflexivas

Cuando se trata de relaciones reflexivas con correspondencia 1:N se transforma utilizando el método de propagación de clave, aunque, en este caso, al tratarse de la misma tabla, es necesario renombrar el nombre del identificador que se transfiere, ya que si no estaría repetido respecto del ya existente en la entidad de origen.

Transformación de relaciones exclusivas

Después de haber realizado la transformación según las relaciones 1:N, se debe tener en cuenta que si los identificadores propagados se han convertido en claves ajenas de la tabla originada por la entidad común a las relaciones, hay que comprobar que una y sólo una de esas claves es nula en cada ocurrencia. En otro caso, estas comprobaciones se deben hacer en las tablas resultantes de transformar las relaciones.

TRANSFORMAICION DE JERARQUIAS

En el modelo conceptual no se dispone de instrumentos que permitan representar supertipos y subtipos. Se definen distintos métodos de transformación, dependiendo de los objetivos perseguidos:

- Información semántica representada en el modelo.
- Eficiencia de acceso a los datos.

Opción 1:

Consiste en crear una tabla para el supertipo que tenga de clave primaria, el identificador y una tabla para cada uno de los subtipos que tengan el identificador del supertipo como clave ajena. Esta solución es apropiada cuando los subtipos tienen muchos atributos distintos y se quieren conservar los atributos comunes en una tabla. Es la solución que mejor conserva la semántica.

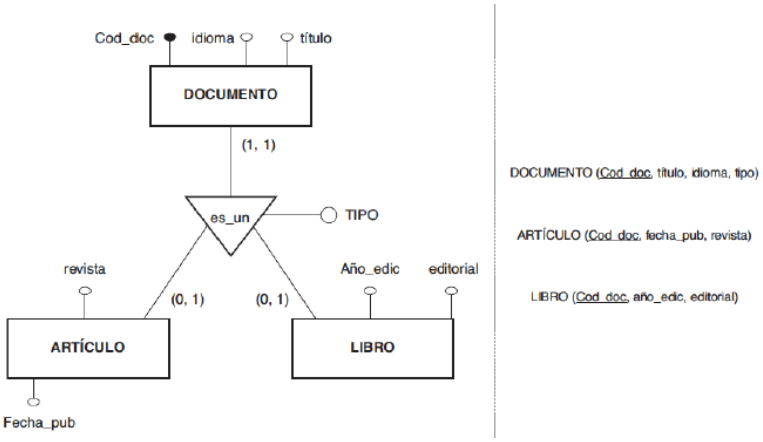


Ilustración 24 Transformación jerarquía - opción 1

Opción 2

Se crea una tabla para cada subtipo, los atributos comunes aparecen en todos los subtipos y la clave primaria para cada tabla es el identificador del supertipo.
Esta opción mejora la eficiencia en los accesos a todos los atributos de un subtipo, sean los comunes al supertipo o los específicos.

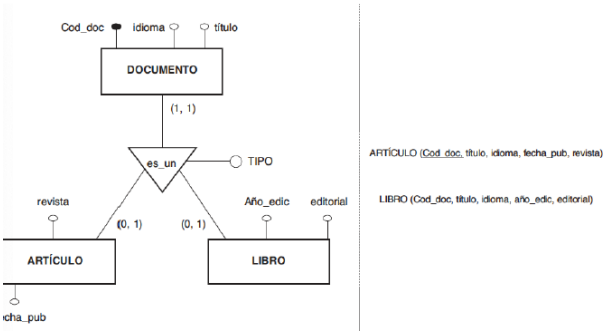


Ilustración 25 - Transformación jerarquía - opción 2

Opción 3

Agrupar en una tabla todos los atributos de la entidad supertipo y de los subtipos.
La clave primaria de esta tabla es el identificador de la entidad. Se añade un atributo que indique a que subtipo pertenece cada ocurrencia (el atributo discriminante de la jerarquía).
Esta solución puede aplicarse cuando los subtipos se diferencien en pocos atributos y las relaciones entre los subtipos y otras entidades sean las mismas. Para el caso de que la jerarquía sea total, el atributo discriminante no podrá tomar valor nulo (ya que toda ocurrencia pertenece a alguna de las entidades subtipo).

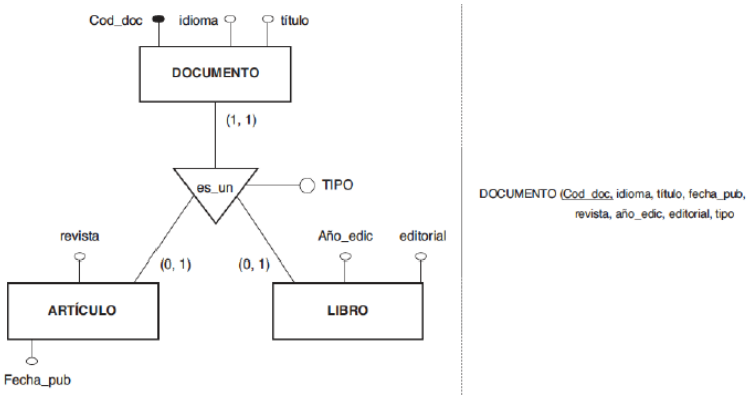
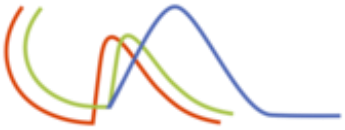


Ilustración 26- relación reflexiva



DISEÑO FÍSICO

El diseño físico de datos define el esquema interno de la base de datos. según Métrica v3 se define la estructura fisica de datos del Sistema de Informacion a partir del modelo logico de datos normalizado.

En función del SGBD, los requisitos y el entorno, se busca la eficiencia, tratando de mejorar tiempos de respuesta y optimizar los recursos del sistema.

Por tanto, el objetivo es obtener un esquema interno de la BD que cumpla lo mejor posible los objetivos de funcionamiento de la BD que los usuarios esperan: minimizar el tiempo de respuesta de la BD y su espacio de almacenamiento e incrementar la seguridad.
El diseño físico se divide de cuatro fases, cada una de ellas compuesta por una serie de pasos:

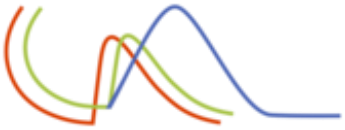
1. Traducir el esquema lógico para el SGBD especifico.
2. Diseñar la representación física.
3. Diseñar los mecanismos de seguridad.
4. Monitorizar y afinar el sistema.

Traducir el esquema lógico para el SGBD específico

Para ello, es necesario conocer toda la funcionalidad que el SGBD ofrece. Por ejemplo, el diseñador deberá saber:
– Si el sistema soporta la definición de claves primarias, claves ajenas y claves alternativas.
– Si el sistema soporta la definición de datos requeridos (es decir, si se pueden definir atributos como no nulos).
– Si el sistema soporta la definición de dominios.
– Si el sistema soporta la definición de restricciones o aserciones de usuario.
– Como se crean las tablas.

La tarea fundamental de construcción del esquema físico es la implementación de las relaciones o tablas obtenidas en el modelo lógico. Se definen mediante el **lenguaje de definicion de datos** (**DDL**) del SGBD. Para ello, se utiliza la información producida durante el diseño lógico, es decir, el esquema lógico global.
El esquema lógico consta de un conjunto de relaciones o tablas y, para cada una de ellas, se tiene:
– El nombre de la relación.
– La lista de atributos entre paréntesis.
– La clave primaria y las claves ajenas, si las tiene.
– Las reglas de integridad de las claves ajenas.

Y para cada uno de los atributos se define:
– Su dominio: tipo de datos, longitud y restricciones de dominio.
– El valor por defecto, que es opcional.
– Si admite nulos.
– Si es derivado y, en caso de serlo, como se calcula su valor.



Como consecuencia de todo ello se podrán crear los "scripts" de la base de datos escritos con la sintaxis y funcionalidades del LDD del SGBDR. Las dos ordenes fundamentales a utilizar en este caso son: CREATE TABLE y CREATE DOMAIN.

Esta sintaxis, además, permite incluir referencias al modo de almacenamiento como pueden ser: ficheros físicos o lógicos (tablespaces) asignables a cada tabla, segmentos o bloque asignados inicialmente y en expectativa de crecimiento, particionamientos de la estructura, etc.
Además, será necesario completar la funcionalidad de las estructuras creadas mediante las denominadas **restricciones de usuario**, que permitirán delimitar las actualizaciones que se realizan sobre las relaciones de la base de datos mediante restricciones, aserciones o disparadores. En cuanto a las primeras, se definen mediante la cláusula de CREATE TABLE, CONSTRAINT CHECK (<condicion>), las segundas mediante la orden CREATE ASSERTION y los ultimos con CREATE TRIGGER.

Diseñar la representación física

Uno de los objetivos principales del diseño físico es almacenar los datos de modo eficiente.
Para medir la eficiencia hay varios factores que se deben tener en cuenta:

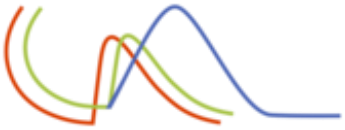
- Productividad de transacciones. Es el número de transacciones que se quiere procesar por unidad de tiempo.
- Tiempo de respuesta. Es el tiempo que tarda en ejecutarse una transacción. Desde el punto de vista del usuario, este tiempo debería ser el mínimo posible.
- Espacio en disco. Es la cantidad de espacio en disco que hace falta para los ficheros de la base de datos. Normalmente, el diseñador deberá minimizar este espacio.

Normalmente, todos estos factores no se pueden satisfacer a la vez. Por lo tanto, el diseñador deberá ir ajustando estos factores para conseguir un equilibrio razonable. El diseño físico inicial no será el definitivo, sino que habrá que ir monitorizándolo para observar sus prestaciones e ir ajustándolo como sea oportuno.

ANALIZAR LAS TRANSACCIONES

Para realizar un buen diseño físico es necesario conocer las consultas y las transacciones que se van a ejecutar sobre la base de datos. Esto incluye tanto información cualitativa, como cuantitativa. Para cada transacción, hay que especificar:

- La frecuencia con que se va a ejecutar.
- Las relaciones y los atributos a los que accede la transacción, y el tipo de acceso: consulta, inserción, modificación o eliminación. Los atributos que se modifican no son buenos candidatos para construir estructuras de acceso.
- Los atributos que se utilizan en los predicados del WHERE de las sentencias SQL. Estos atributos pueden ser candidatos para construir estructuras de acceso, dependiendo del tipo de predicado que se utilice.
- Si es una consulta, los atributos involucrados en el join de dos o más relaciones. Estos atributos pueden ser candidatos para construir estructuras de acceso.



- Las restricciones temporales impuestas sobre la transacción. Los atributos utilizados en los predicados de la transacción pueden ser candidatos para construir estructuras de acceso.

ESCOGER LAS ORGANIZACIONES DE FICHEROS

En el caso de aquellos SGBDRs que permitan asociar cada estructura de datos con un fichero físico es recomendable escoger la organización de ficheros óptima para cada relación.

Por ejemplo, un fichero desordenado es una buena estructura cuando se va a cargar gran cantidad de datos en una relación al inicializarla, cuando la relación tiene pocas tuplas, también cuando en cada acceso se deben obtener todas las tuplas de la relación, o cuando la relación tiene una estructura de acceso adicional, como puede ser un índice.

Por otra parte, los ficheros dispersos (hashing) son apropiados cuando se accede a las tuplas a través de los valores exactos de alguno de sus campos (condición de igualdad en el WHERE).
Si la condición de búsqueda es distinta de la igualdad (búsqueda por rango, por patrón, etc.), la dispersión no es una buena opción. Hay otras organizaciones, como la ISAM o los arboles B+.

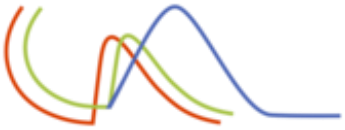
ESCOGER LOS INDICES SECUNDARIOS

Los índices secundarios permiten especificar caminos de acceso adicionales para las relaciones base. Hay que tener en cuenta que estos índices conllevan un coste de mantenimiento que hay que sopesar frente a la ganancia en prestaciones. A la hora de seleccionar los índices, se pueden seguir las siguientes indicaciones:

- Construir un índice sobre la clave primaria de cada relación base, normalmente se crea de forma implícita con la cláusula CONSTRAINT PRIMARY KEY.
- No crear índices sobre tablas pequeñas.
- Anadir un índice sobre los atributos que se utilizan para acceder con mucha frecuencia.
- Anadir un índice sobre las claves ajenas que se utilicen con frecuencia para hacer joins.
- Evitar los índices sobre atributos que se modifican a menudo.
- Evitar los índices sobre atributos poco selectivos (aquellos en los que la consulta selecciona una porción significativa de la tabla).
- Evitar los índices sobre atributos formados por cadenas de caracteres largas.

CONSIDERAR LA INTRODUCCION DE REDUNDANCIAS CONTROLADAS

En ocasiones, puede ser conveniente relajar las reglas de normalización introduciendo redundancias de forma controlada, con objeto de mejorar las prestaciones del sistema. En la etapa del diseño lógico se recomienda llegar, al menos, hasta la tercera forma normal para obtener un esquema con una estructura consistente y sin redundancias. Pero, a menudo, sucede que las bases de datos, así normalizadas, no proporcionan la máxima eficiencia, con lo que es necesario volver atrás y



desnormalizar algunas relaciones. Es importante hacer notar que la desnormalización solo debe realizarse cuando se estime que el sistema no puede alcanzar las prestaciones deseadas. Por lo tanto, hay que tener en cuenta los siguientes factores:
- La desnormalización hace que la implementación sea más compleja.
- La desnormalización hace que se sacrifique la flexibilidad.
- La desnormalización puede hacer que los accesos a datos sean más rápido, pero ralentiza las actualizaciones.

Optimización

La optimización consiste en una **desnormalización controlada del modelo físico de datos** que se aplica para reducir o simplificar el número de accesos a la base de datos.

El objetivo de esta técnica es reestructurar el modelo físico de datos con el fin de asegurar que satisface los requisitos de rendimiento establecidos y conseguir una adecuada eficiencia del sistema.

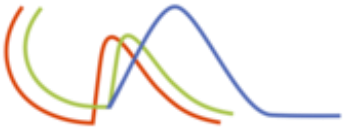
Para ello, se seguirán alguna de las **recomendaciones** que a continuación se indican:
- Introducir elementos redundantes.
- Dividir entidades.
- Combinar entidades si los accesos son frecuentes dentro de la misma transacción.
- Redefinir o añadir relaciones entre entidades para hacer más directo el acceso entre entidades.
- Definir claves secundarias o índices para permitir caminos de acceso alternativos.

Con el fin de analizar la conveniencia o no de la desnormalización, se han de considerar, entre otros, los siguientes aspectos:
- Los tiempos de respuesta requeridos.
- La tasa de actualizaciones respecto a la de recuperaciones.
- Las veces que se accede conjuntamente a los atributos.
- La longitud de los mismos.
- El tipo de aplicaciones (en línea / por lotes).
- La frecuencia y tipo de acceso.
- La prioridad de los accesos.
- El tamaño de las tablas.
- Los requisitos de seguridad: accesibilidad, confidencialidad, integridad y disponibilidad que se consideren relevantes.

Diseñar los mecanismos de seguridad

Los datos constituyen un recurso esencial para la organización, por lo tanto su seguridad es de vital importancia. Durante el diseño lógico se habrán especificado los requerimientos en cuanto a seguridad que en esta fase se deben implementar. Para llevar a cabo esta implementación, el diseñador debe conocer las posibilidades que ofrece el SGBD que se vaya a utilizar.

- **Diseñar las vistas de los usuarios**: las vistas de los usuarios se corresponden a los esquemas lógicos locales. Las vistas, además de preservar la seguridad, mejoran la independencia de datos, reducen la complejidad y permiten que los usuarios vean los datos en el formato deseado. Para crearlas se utiliza la orden *CREATE VIEW.*
- **Disenar las reglas de acceso**: para cada usuario o grupo de usuarios se determinan tanto los permisos sobre determinados objetos de la base de datos, como



los permisos o privilegios del sistema, es decir, las operaciones de DDL que están disponibles para ese usuario o grupo. Normalmente, los privilegios se agrupan en conjuntos denominados «Roles». Para otorgar un permiso o rol se utiliza la orden *GRANT* y para denegarlo *REVOKE*. Se definen mediante el **Lenguaje de Control de Datos** (**DCL**).

Monitorizar y afinar el sistema

Una vez implementado el esquema físico de la base de datos, se debe poner en marcha para observar su rendimiento (monitorizar). En el caso de que no se satisfagan las necesidades de rendimiento deseadas, el esquema deberá ser modificado (afinar).
Este proceso no es estático y puntual, sino que la monitorización está presente a lo largo de toda la vida del sistema, requiriéndose nuevas modificaciones (refinar) para adaptarse a los nuevos requisitos.

EL MODELO LÓGICO RELACIONAL

El modelo relacional es un modelo con sólidos fundamentos matemáticos, basado en la teoría de conjuntos. Fue definido por E. F. Codd en 1970.
Las características fundamentales del modelo relacional son:
- Las **estructuras de datos son simples**: se trata de relaciones que se presentan al usuario en forma de tablas bidimensionales, permitiendo un alto grado de independencia de la información, con respecto al medio físico de almacenamiento de los datos.
- Proporciona una base sólida para la consistencia de los datos a través de las **reglas de integridad**. Igualmente, el proceso de normalización, al eliminar ciertas anomalías que pueden presentarse en las relaciones, representa una valiosa ayuda para el diseño de la BD.
- Permite la manipulación de las relaciones en forma **orientada a conjuntos**. Esto ha conducido al desarrollo de lenguajes muy potentes basados, bien en la teoría de conjuntos (algebra relacional), bien en la lógica de predicados (calculo relacional).

Conceptos fundamentales del modelo relacional

El esquema de una BDR se compone de uno o más esquemas de relación y de un conjunto de restricciones de integridad.
Un esquema de relación consiste en el nombre de relación, seguido de los nombres de los atributos:

Nombre_Relacion (Atributo1, Atributo2, ... , AtributoN)

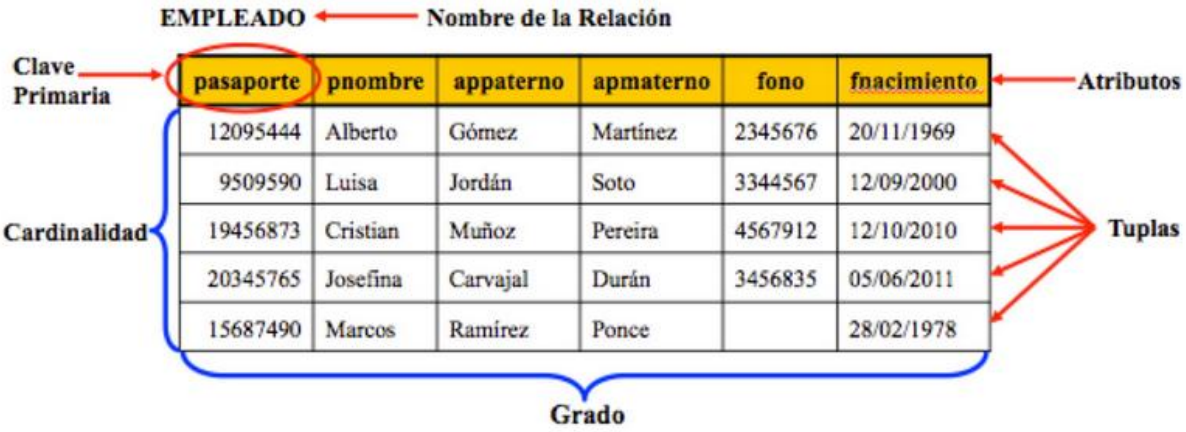
La definición formal de una relación: "*Dados los dominios D1, D2, … , Dn (no necesariamente distintos), R es una relación entre estos n conjuntos si es un conjunto de n tuplas (d1, d2,…, dn) tal que d1 pertenece a D1... dn pertenece a Dn*".

Un dominio es un conjunto de valores.

Cada atributo, o propiedad con interés informativo de una relación está asociado a un dominio del que toma sus posibles valores.

El número de atributos de una relación define su **grado**, mientras que el número de tuplas de la relación define su **cardinalidad**.

La extensión u ocurrencia de una relación es una tabla donde las filas corresponden a las **tuplas** y las columnas a los **atributos**.



pasaporte	pnombre	appaterno	apmaterno	fono	fnacimiento
12095444	Alberto	Gómez	Martínez	2345676	20/11/1969
9509590	Luisa	Jordán	Soto	3344567	12/09/2000
19456873	Cristian	Muñoz	Pereira	4567912	12/10/2010
20345765	Josefina	Carvajal	Durán	3456835	05/06/2011
15687490	Marcos	Ramírez	Ponce		28/02/1978

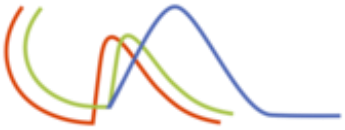
Ilustración 27 - Relación

De estas definiciones se deducen las características de una tabla de estructura relacional:
- Cada tabla debe contener un solo tipo de filas. El formato de cada fila queda definido por el esquema de la relación. Es decir, todas las filas tienen las mismas columnas y formato.
- Cada fila tiene que ser única, no puede haber filas duplicadas.
- El orden de las filas dentro de una tabla es indiferente.
- Cada columna debe estar identificada por un nombre especifico.
- El orden de las columnas dentro de una tabla es indiferente.
- Cada columna debe extraer sus valores de un dominio.
- Un mismo dominio podrá servir para definir los valores de varias columnas diferentes.
- El valor individual de la intersección de cualquier fila y columna será un único dato.

Reglas de integridad

Permiten definir propiedades de los datos que no pueden ser capturadas basándose en conjuntos y relaciones. Las razones para que un modelo requiera restricciones son:
- **Semántica:** permite reflejar más exactamente la información a modelar.
- **Integridad:** comunica al SGBD que estados de la BD están permitidos.

Al definir cada atributo sobre un dominio se impone una restricción sobre el conjunto de valores permitidos para cada atributo. A este tipo de restricciones se les denomina restricciones de dominios.



Hay además dos reglas de integridad muy importantes que son restricciones que se deben cumplir en todas las bases de datos relacionales y en todos sus estados o instancias (las reglas se deben cumplir todo el tiempo). Estas reglas son la **regla de integridad de entidades** y la **regla de integridad referencial**.

Antes de definirlas, es preciso conocer el concepto de **nulo**. Cuando en una tupla un atributo es **desconocido**, se dice que es nulo. Un nulo no representa el valor cero ni la cadena vacía, estos son valores que tienen significado. El nulo implica ausencia de información, bien porque al insertar la tupla se desconocía el valor del atributo, o bien porque para dicha tupla el atributo no tiene sentido.

Regla de integridad de la ENTIDAD

Se aplica a las **claves primarias** de las relaciones. Ninguno de los atributos que componen la clave primaria puede ser nulo.
Por definición, una clave primaria es un identificador irreducible que se utiliza para identificar de modo único las tuplas. Que sea irreducible significa que ningún subconjunto de la clave primaria sirve para identificar las tuplas de modo único. Si se permite que parte de la clave primaria sea nula, se está diciendo que no todos sus atributos son necesarios para distinguir las tuplas, con lo que se contradice la irreducibilidad.

Esta regla solo se aplica a las relaciones y a las claves primarias, no a las claves alternativas.

Regla de integridad REFERENCIAL

La segunda regla de integridad se aplica a las **claves ajenas**. Si en una relacion hay alguna clave ajena, sus valores deben coincidir con valores de la clave primaria a la que hace referencia, o bien, deben ser completamente nulos.
Ejemplo: clave ajena de la relacion libro que referencia a una editorial.
Editorial(**cod_editorial**, direccion, provincia, ...)
Libro(**cod_libro**, titulo, ..., **cod_editorial**)
Ejemplo: claves ajenas de la relacion Escribe que referencia a un Autor y a un Libro respectivamente.
Autor(**DNI**, nombre, apellidos, ...)
Libro(**cod_libro**, titulo, ..., cod_editorial)
Escribe(**autor**, **cod_libro**, titulo, ..., cod_editorial)

Lenguajes relaciones

Son varios los lenguajes utilizados por los SGBD relacionales para manejar las relaciones.
Algunos de ellos son procedurales, lo que quiere decir que el usuario dice al sistema exactamente como debe manipular los datos. Otros son no procedurales, que significa que el usuario dice **que datos necesita**, en lugar de decir cómo deben obtenerse.

La base de los lenguajes relacionales la constituyen el **algebra relacional** y el **calculo relacional**. Ambos lenguajes son equivalentes: para cada expresión del algebra, se puede encontrar una expresión equivalente en el cálculo, y viceversa. El algebra relacional (o el cálculo relacional) se utiliza para medir la potencia de los lenguajes relacionales.
Si un lenguaje permite obtener cualquier relación que se pueda derivar mediante el algebra relacional, se dice que es relacionalmente completo. La mayoría de los lenguajes relacionales son relacionalmente completos, pero tienen más potencia que el algebra o el cálculo porque se les han añadido operadores especiales. En este apartado se verán las características más importantes correspondientes al algebra relacional.

Algebra relacional

Lenguaje formal con una serie de operadores que trabajan sobre una o varias relaciones para obtener otra relación resultado, sin que cambien las relaciones originales.
Tanto los operandos como los resultados son relaciones, por lo que la salida de una operación puede ser la entrada de otra operación. Esto permite anidar expresiones del algebra, del mismo modo que se pueden anidar las expresiones aritméticas.

De los ocho operadores, solo hay 5 fundamentales, que forman un *conjunto relacionalmente completo*, es decir, permite obtener cualquier subconjunto de los datos contenidos en una BD:
- **Selección**: opera sobre una sola relación R y da como resultado otra relación cuyas tuplas son las tuplas de R que satisfacen la condición especificada (C). Esta condición es una comparación en la que aparece al menos un atributo de R, o una combinación booleana de varias de estas comparaciones.

Cliente			
codigo	nombre	apellido	edad
1	Pedro	Perez	19
2	Karen	Pucci	20

$$\sigma_{\text{apellido = "Perez"}}(\text{Cliente})$$

codigo	nombre	apellido	edad
1	Pedro	Perez	19

Ilustración 28- Algebra relacional Selección

- **proyección**: opera sobre una sola relación R y da como resultado otra relación que contiene un subconjunto vertical de R, extrayendo los valores de los atributos especificados y eliminando duplicados.

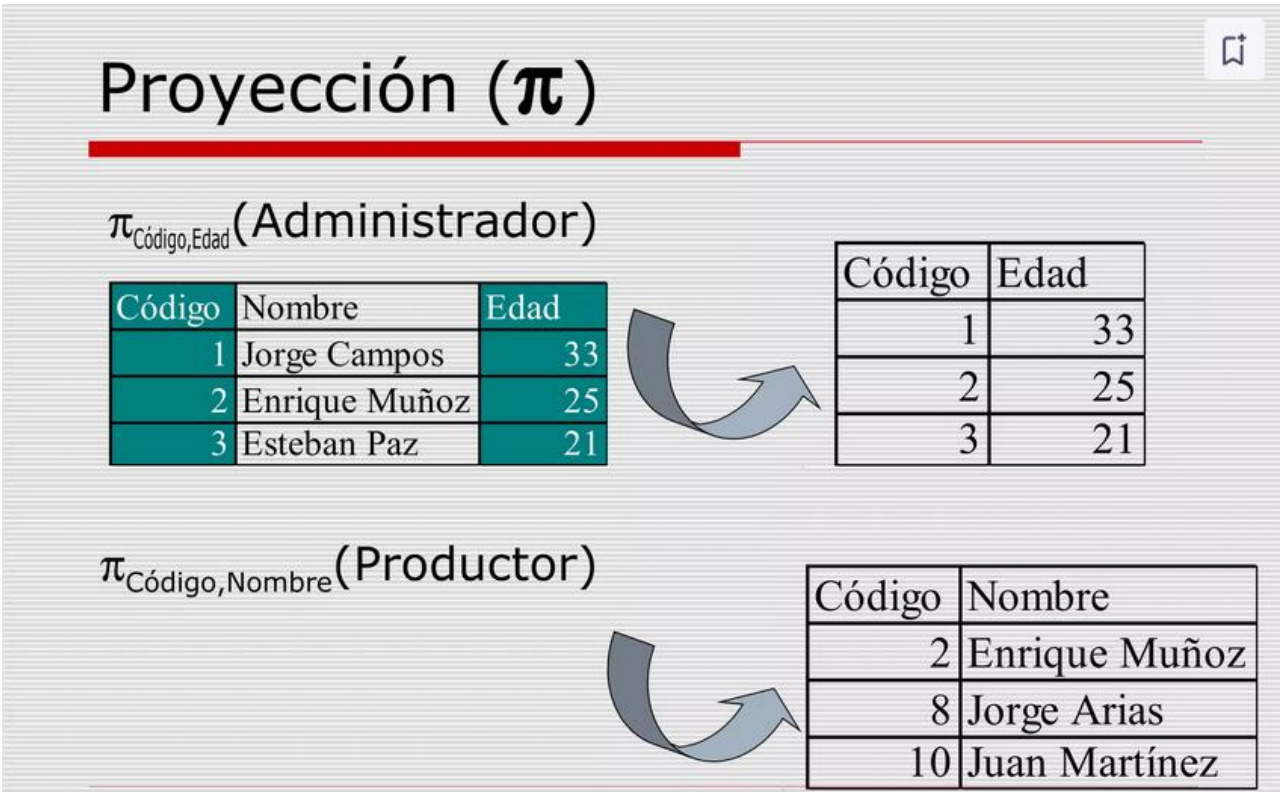


Ilustración 29 – Proyección

Producto cartesiano: relación resultante de combinar cada fila de la relación R con todas las de la relación S. Ya que es posible que haya atributos con el mismo nombre en las dos relaciones, el nombre de la relación se antepondrá al del atributo, en este caso, para que los nombres de los atributos sigan siendo únicos en la relación resultado.
En la relación resultante:
o El grado es la suma del número de columnas de R y S.
o La cardinalidad es el producto del numero de filas de R por el numero de filas de S.

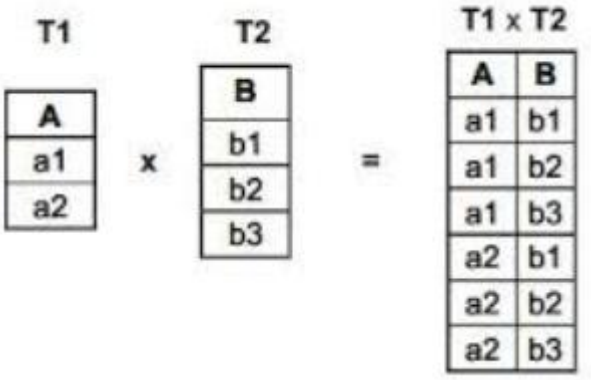


Ilustración 30 -Producto Cartesiano

- **Union**: Dadas dos relaciones R y S, se define la unión entre ambas como la relación

resultante de tomar las filas que están en una u otra relación y además las que están en ambas (solo una vez).
Para que se pueda producir la unión es necesario que las relaciones R y S presenten igual grado y compatibilidad de dominios para cada par de atributos tomados de uno en uno entre ambas relaciones, es decir, r1 ha de ser de un dominio igual o compatible a s1 y, en general, rN ha de ser de un dominio igual o compatible a sN.

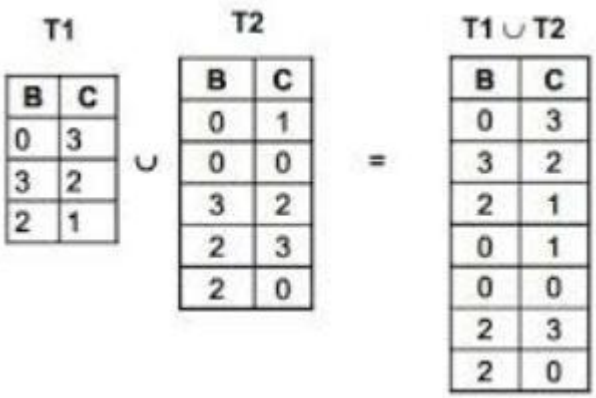


Ilustración 31- Unión

Diferencia: dadas dos relaciones R y S, se define la diferencia entre ambas como la relación resultante de tomar las filas que están en R y no están en S. Se trata de una operación no conmutativa, a diferencia de las anteriores.

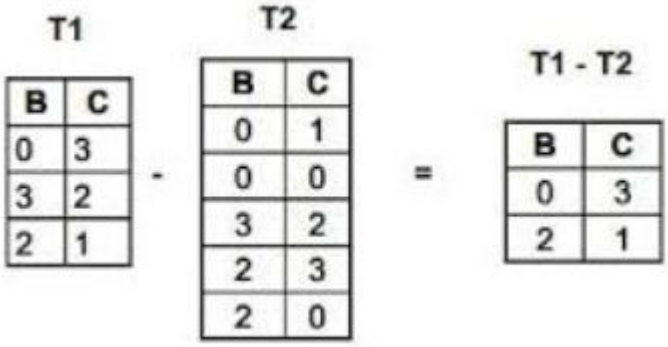


Ilustración 32 - Diferencia

Operadores no fundamentales:
- **Join**: dadas dos relaciones R y S, se define la concatenación o join como la relación resultante del producto cartesiano entre ambas tablas una vez seleccionadas aquellas filas que tomen igual valor en aquella/s filas expresadas en la operación. Para que la operación se pueda producir es necesario que ambas relaciones presenten, al menos, un campo en común sobre el que establecer la condición de igualdad

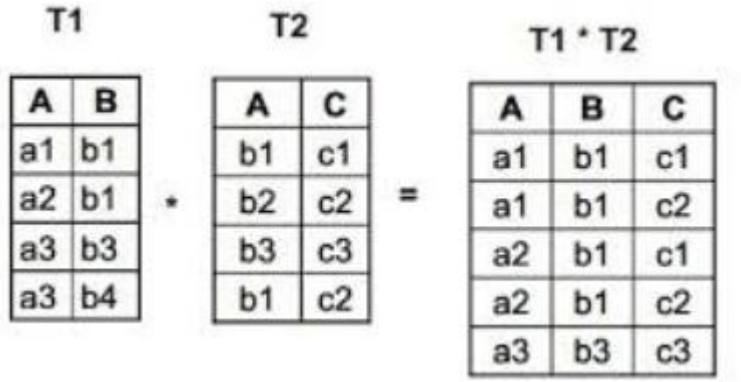


Ilustración 33 - Join

- **Intersección**: dadas dos relaciones R y S, se define la intersección como la relación resultante de tomar las filas que están tanto en R como en S. Al igual que en la unión, las relaciones R y S han de presentar igual grado y compatibilidad de dominios para cada par de atributos tomados de uno en uno entre ambas relaciones.

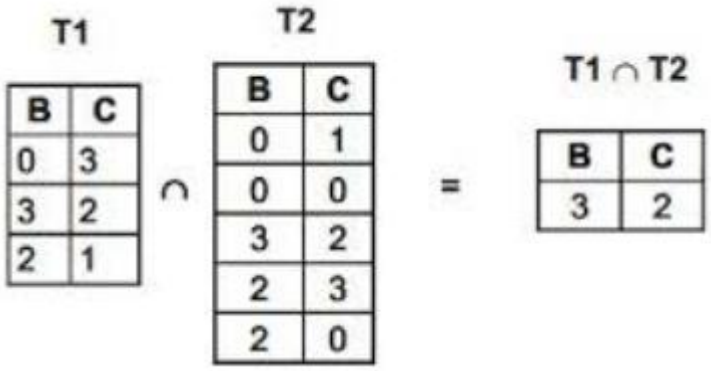


Ilustración 34 – Intersección

- **División**: dadas dos relaciones R y S en las que existe un subconjunto de atributos (X) que están formando parte de S y R al mismo tiempo y otro conjunto de atributos (Y) que únicamente forman parte de R, se define la división o cociente como la relación resultante de combinar cada fila de Y con todas las filas que forman parte de los atributos X. En definitiva, para que t aparezca en el resultado, los valores de t deben aparecer en R en combinación con cada tupla de S.

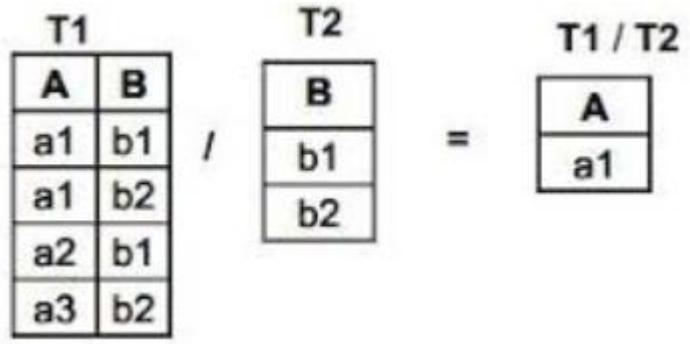


Ilustración 35 - División