TALLER DE BASE DE DATOS

1. ¿Qué es una Base de Datos?

Existen varias definiciones para responder a esta pregunta:

- Colección de datos interrelacionados almacenados en conjunto sin redundancias perjudiciales o innecesarias; su finalidad es servir a una aplicación o más, de la mejor manera posible; los datos se almacenan de modo que resulten independientes de los programas que los usan; se emplean métodos bien determinados para incluir nuevos datos y para modificar o extraer los datos almacenados». (Martin. *1975).*

- Colección o depósito de datos, donde los datos están lógicamente relacionados entre sí, tienen una definición y descripción comunes y están estructurados de una forma particular.
  Una base de datos es también un modelo del mundo real y, como tal, debe poder servir para toda una gama de usos y aplicaciones». (Conference des Statisticiens Européens, 1977).

- Colección de datos interrelacíonados». (Elsmari y Navathe, 1989).

Como se ha visto, el concepto de base de datos ha ido cambiando a lo largo del tiempo. En la actualidad podemos establecer la definición de base de datos como sigue.

Una Base de Datos en una colección de datos organizada, de manera que podamos realizar actualizaciones, consultas e informes de forma fácil y rápida para usar la información en la toma de decisiones.

2. Modelos de bases de datos

- Base de datos relacionales
- Bases de datos jerárquicas
- Base de datos de red
- Bases de datos transaccionales
- Bases de datos multidimensionales
- Bases de datos orientadas a objetos
- Bases de datos documentales
- Bases de datos deductivas

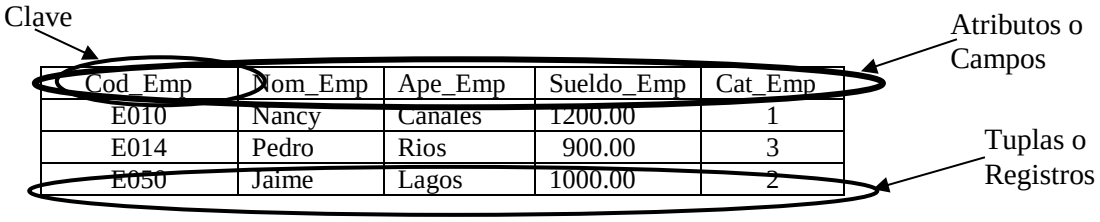
3. El modelo relacional

Se basa en la representación de datos por medio de estructuras tipo tabla, denominadas relaciones, y que almacenan información para una determinada entidad.
Cada una de estas relaciones representa en sus columnas los valores significativos, es decir, de los que interesa conocer su valor, para cada entidad. Dichas columnas se denominan atributos y para cada uno de ellos existirá un valor (cabe la posibilidad de que existan atributos en los que no aparezca ningún valor).

Cada fila representa la información para cada ocurrencia de una determinada entidad. A dichas filas también se las denomina tuplas.
Cada atributo o conjunto de atributos que identifica unívocamente a cada tupla de la relación se denomina clave. Las claves se representan subrayando el / los atributo/s que forman parte de ella.

El siguiente ejemplo representa una relación denominada empleado, que almacena información sobre los empleados de una empresa.
La información que se desea saber de cada empleado es su código, nombre y apellidos, sueldo y categoría.

Por lo tanto, los atributos son: cod_empleado, nombre, apellidos, sueldo y categoría. Además, el atributo cod_empleado es clave de la relación, ya que si se sabe el valor de dicho atributo, se puede saber a que empleado nos estamos refiriendo.



Cod_Emp	Nom_Emp	Ape_Emp	Sueldo_Emp	Cat_Emp
E010	Nancy	Canales	1200.00	1
E014	Pedro	Rios	900.00	3
E050	Jaime	Lagos	1000.00	2

4. Fases del diseño de una Base de Datos

El diseño de una base de datos suele descomponerse en tres grandes fases (diseño conceptual, diseño lógico y diseño físico), lo que permite reducir la complejidad que entraña el diseño, a la vez que ayuda a alcanzar los dos principales objetivos que tienen las bases de datos:

- Ser una representación fidedigna del mundo real,
- Ser un servidor operacional y eficiente de los datos.

a. Diseño Conceptual

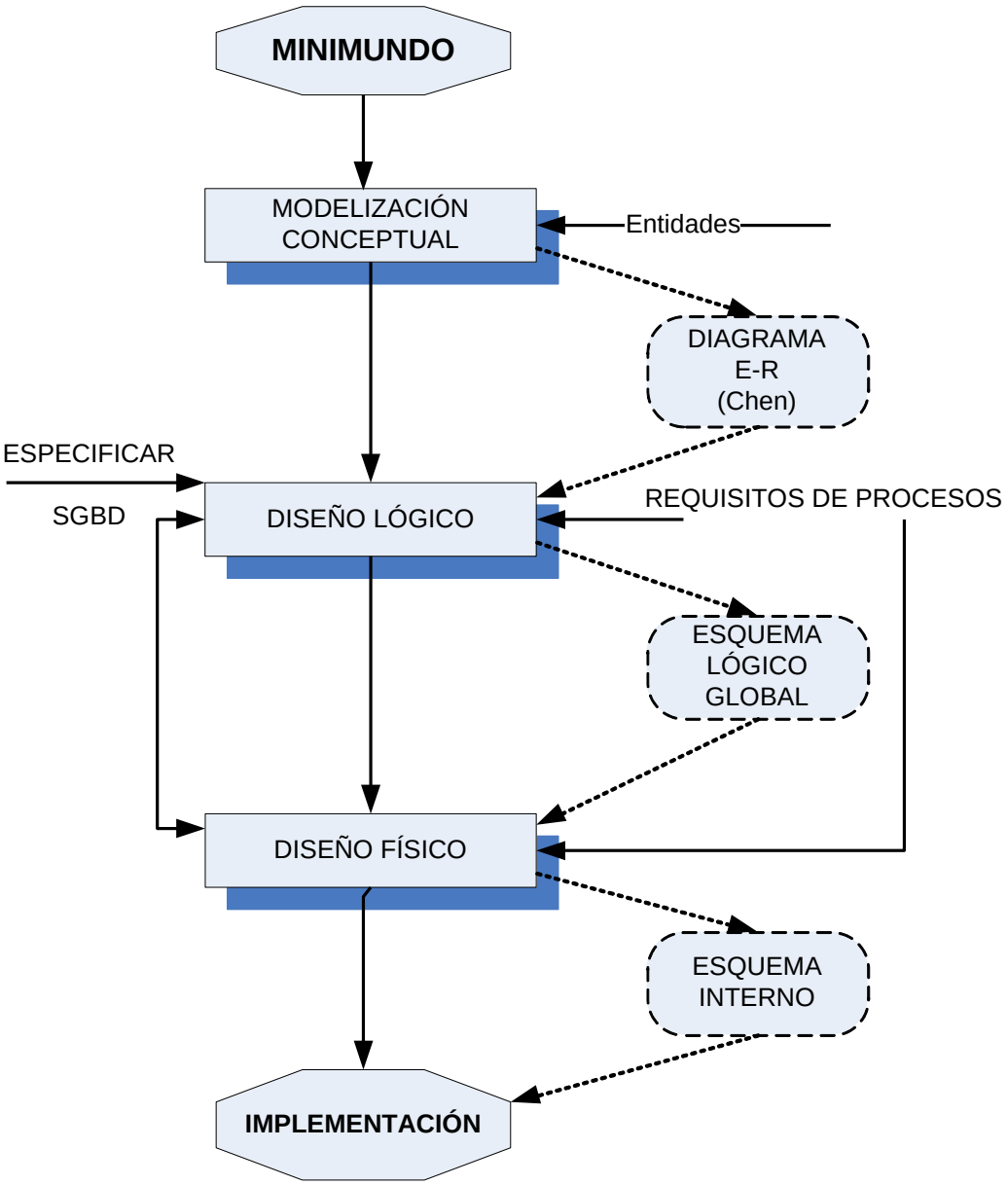
El diseño conceptual parte de la especificación de requerimientos, y produce como resultado el esquema conceptual de la base de datos. Un esquema conceptual es una descripción a alto nivel de la estructura de la base de datos, independientemente de la elección del equipamiento y del Sistema Gestor de Base de Datos (en adelante referido como SGBD) que se usen para la implementación de la base de datos.

b. Diseño Lógico

El diseño lógico parte del esquema conceptual y genera el esquema lógico. Un esquema lógico es la descripción de la estructura de la base de datos que puede procesarse por un SGBD. Una vez elegido el modelo lógico pueden existir un conjunto de esquemas lógicos equivalentes al mismo esquema conceptual. La meta del diseño lógico es producir el esquema lógico más eficiente con respecto a las operaciones de consulta y actualización.

c. Diseño Físico

El diseño físico toma como punto de partida el esquema lógico y como resultado produce el esquema físico. Un esquema físico es una descripción de la implementación de la base de datos en memoria secundaria; describe las estructuras de almacenamiento y los métodos de acceso para acceder a los datos de una manera eficiente. Por ello, el diseño físico se genera para un SGBD y un entorno físico determinado.



Diseño de una base de datos

En la gráfica se observa el resumen de las tres grandes fases del diseño de una base de datos: primero se diseña el esquema conceptual (que se realiza con un modelo conceptual de datos), esquema que proporciona una descripción estable de la base de datos (independiente del SGBD) que se vaya a utilizar; posteriormente se pasa del esquema conceptual al modelo de datos propio de SGBD elegido (diseño lógico); por último se eligen las estructuras de almacenamiento, los caminos de acceso (índices), y todos los aspectos relacionados con el diseño físico.

5. Diseño conceptual

El objetivo del diseño conceptual, también denominado modelo conceptual, y que constituye la primera fase de diseño, es obtener una buena representación de los recursos de información de la empresa, con independencia de usuario o aplicaciones en particular y fuera de consideraciones sobre eficiencia del ordenador.

Consta de dos fases:

- **Análisis de requisitos**: Es en esta etapa donde se debe responder a la pregunta: ¿qué representar?. Se pretende en esta etapa elaborar un esquema descriptivo del mundo real, mediante distintas técnicas, aunque la más usada es la de entrevistas a los usuarios, lo que implica una descripción de los datos mediante el uso del lenguaje natural. Los problemas que presenta esta primera especificación, se irán refinando hasta obtener el esquema conceptual.
- **Conceptualización**: En esta etapa se intenta responder a la pregunta: ¿como representar?. Consiste en ir refinando sucesivamente el primer esquema descriptivo, para conseguir pasar del mundo real al esquema descriptivo y de éste al esquema conceptual, que deberá ser expresado sin tener en consideración cuestiones de implementación, es decir, debe ser independiente del SGBD a usar.

  Un esquema conceptual debe cumplir las siguientes características:
  - Debe representar fielmente la información del mundo real
  - Ser independiente del SGBD
  - Ser independiente del Hardware

  Por lo tanto, un buen diseño del esquema conceptual, influirá positivamente en el resto de etapas.

  El modelo que se estudiará es el Modelo Entidad / relación (en adelante referido como ME/R o modelo E/R), que es el más utilizado hoy en día.

EL MODELO ENTIDAD / RELACIÓN

El modelo E/R fue propuesto por Peter P. Chen. Define el modelo relacional como una vista unificada de los datos, centrándose en la estructura lógica y abstracta, como representación del mundo real, con independencia de consideraciones de tipo físico.

E-R: Se basa en una representación gráfica de una serie de entidades relacionadas entre sí. Al utilizar una representación de este tipo, el modelo E/R permite distinguir fácilmente y a simple vista, las relaciones existentes entre las distintas entidades. Existen muchas formas de representarlo, como ya se ha comentado; la que se utilizará aquí no es, por supuesto, la única forma de hacerlo.

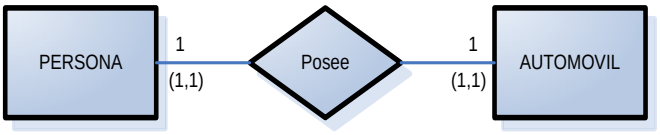
Los elementos de los que se componen son los siguientes:

a. **Entidades**: Una entidad es «una persona, lugar, cosa, concepto o suceso, real o abstracto, de interés para la empresa» (ANSI 1977). En el modelo E/R, se representa por un rectángulo, con el nombre de dicha entidad escrito en la parte superior. Por ejemplo, la Figura representa la entidad automóvil.

b. **Atributos**: Un atributo es cualquier característica que describe a una entidad. Los atributos de una entidad se colocan dentro del rectángulo que representa dicha entidad, justo debajo del nombre de ésta. Por ejemplo, se puede decir que un automóvil tiene las siguientes características: n^0 de matrícula, marca, modelo y color.

c. **Relación**: Una relación se representa mediante un rombo y como su nombre indica, es una correspondencia entre dos entidades. Si tenemos dos entidades automóvil y persona, podemos tener una relación entre ellas. Dicha relación se puede establecer en ambos sentidos:

- Una persona posee un automóvil, y
- Un automóvil pertenece a una persona.



d. **Claves**:

1. **Clave o llave primaria (Primary Key= PK)**: Es un atributo o conjunto de atributos (clave compuesta) de dicha entidad, capaces de identificar unívocamente cada fila de una tabla o entidad. Es decir, si conocemos el valor de dichos atributos, seremos capaces de conocer a que entidad, entre todas las posibles, hace referencia.

   Para ser elegido un atributo como llave primaria debe cumplir:

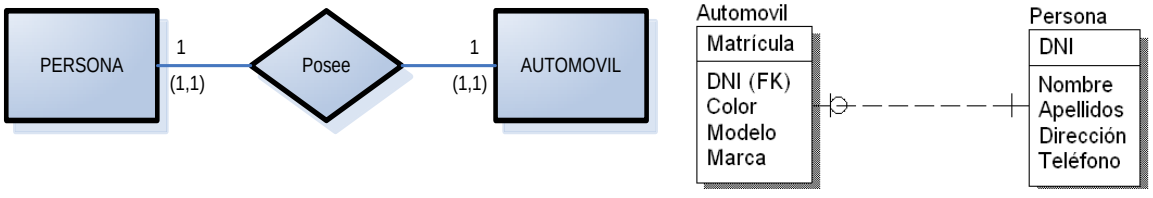
   1. Debe ser única, es decir no debe repetirse
   2. No debe ser Null, (Not Null)

   Esto implica que los valores de los atributos clave no se pueden repetir para dos ocurrencias de la misma entidad. Por ejemplo, el valor del atributo matrícula de un automóvil, es única ya que no existe una misma matrícula para dos automóviles distintos. Los atributos: marca, modelo o color no identifican unívocamente una ocurrencia de la entidad, ya que pueden existir dos automóviles distintos de la misma marca, modelo o color. En el modelo E/R, un atributo clave se representa subrayando dicho atributo.

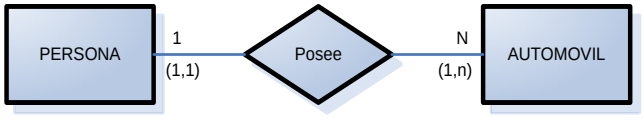
2. **Claves candidatas**: Atributos de una entidad que pueden ser elegidos por el diseñador de la base de datos y que estén en la posibilidad de cumplir las condiciones de ser PK.
3. **Clave foránea (Foreign Key= FK)**: Es llamada clave Externa, es uno o más campos de una tabla que hacen referencia al campo o campos de clave principal de otra tabla, una clave foránea indica como está relacionada la tabla. Los datos en los campos de clave foránea y clave principal o primaria deben coincidir, aunque los nombres de los campos no sean los mismos.

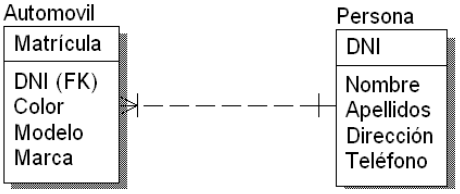
e. **Cardinalidad de una relación**: La cardinalidad de una relación representa el número de ocurrencias que se pueden dar de una relación. Puede ser de tres tipos:

- Cardinalidad 1-1: cada ocurrencia de una entidad se relaciona con una ocurrencia de otra entidad. Ej.: una persona posee un automóvil. Se representa como indica

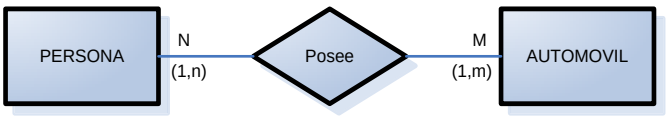


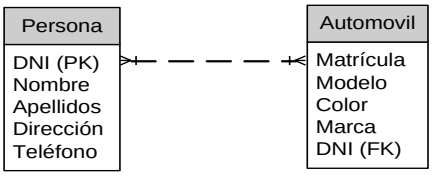
- Cardinalidad 1 - N: también llamada uno a muchos. Cada ocurrencia de una entidad puede relacionarse con varias ocurrencias de otra entidad. Ej.: una persona posee varios automóviles.





- Cardinalidad N-M: también llamada muchos a muchos. Cada ocurrencia de una entidad puede relacionarse con varias ocurrencias de otra entidad y viceversa. Ej.: una persona posee varios automóviles y un automóvil puede pertenecer a varias personas. Se representa como aparece en la figura.





Un autor podrá haber escrito varios libros, de la misma forma que en un libro pueden participar varios autores. De la editorial se desea conocer el nombre y la ciudad.

Ejemplos prácticos de diseño conceptual

PRIMER PROBLEMA: Supongamos que se desea tener almacenados todos los datos de los profesores de un Instituto, que dictan cursos y los alumnos a los cuales se les imparte.

Empezamos identificando entidades.

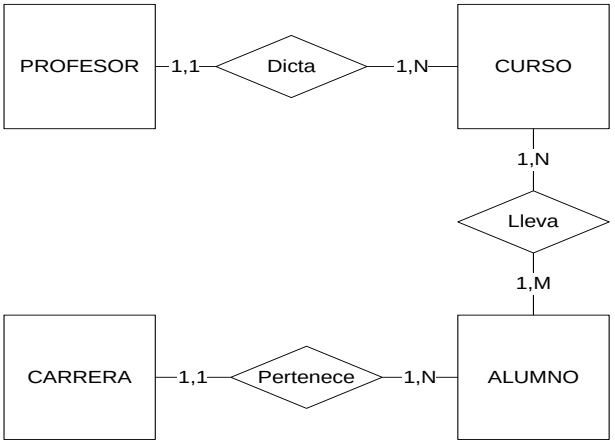
De un profesor se desea conocer su nombre y apellidos dirección y especialidad. Otra entidad podría ser el alumno, del cual se desea conocer su nombre, apellidos dirección, teléfono y carrera en la que estudia.

Curso describe otra entidad de la cual se desea conocer su descripción, horas, etc.

La principal ventaja de este procedimiento radica en que muchas veces supone un ahorro de espacio de almacenamiento.

Una vez establecidas las entidades, vamos a representarlas gráficamente y definir las relaciones entre ellas.

1. Profesor y Curso: un profesor puede impartir varios cursos, pero un curso sólo puede ser impartido por un profesor (1 -N).
2. Alumno y Curso: un alumno puede asistir a varios cursos, y a un curso pueden asistir varios alumnos (N-M).
3. Alumno y Carrera: un alumno pertenece a una carrera profesional, y una carrera puede tener varios alumnos (1-N).



Por lo tanto, el modelo conceptual es el mostrado en el diagrama E-R.

PROFESOR(CodProf, NomProf, ApeProf, DirProf, Especialidad)

CURSO(CodCurso, NomCurso, HrsCurso)

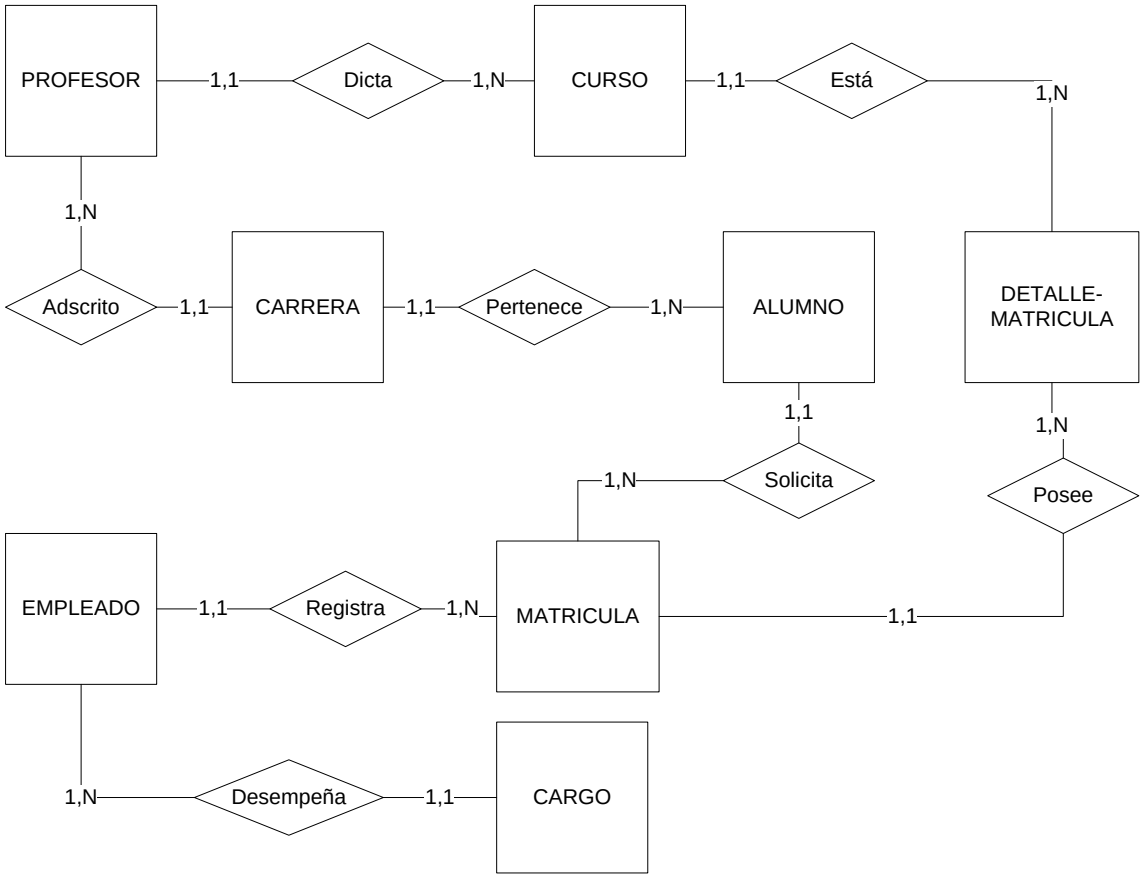
ALUMNO(CodAl, NomAl, ApeAl, DirAl, SexoAl, CodCarrera)

CARRERA(CodCarrera, NomCarrera)

Como se tiene una relación Muchos a Muchos (N-M), debe introducirse una entidad intermedia o normalizar usando una nueva entidad, que puede ser MATRÍCULA

Pero la entidad MATRÍCULA genera una relación N-M con la entidad Alumno, ya un alumno se matrícula en N Cursos y varias veces durante su periodo de estudiante.

Identificando la Cardinalidad en cada relación y las Entidades involucradas, tendremos el siguiente diagrama E-R, según el modelo de Chen.



Por tanto las entidades resultantes serán:

PROFESOR(**CODPROF**, NomProf, ApeProf, DirProf, Especialidad, **CodCarrera**)

CURSO(**CODCURSO**, NomCurso, HrsCurso)

ALUMNO(**CODAL**, NomAl, ApeAl, DirAl, SexoAl, **CodCarrera**)

CARRERA(**CODCARRERA**, NomCarrera)

MATRICULA(**NROMAT**, **CodAl**, **CodEmp**, Fecha, Semestre, Turno)

DETALLE-MATRICULA(**NROMAT, CODCURSO**, Condición)

EMPLEADO(**CODEMP**, NomEmp, ApeEmp, DirEmp, SexoEmp, **CodCargo**)

CARGO(**CODCARGO**, DesCar)

DISEÑO LÓGICO

El objetivo del diseño lógico es transformar el esquema conceptual obtenido en la fase anterior, adaptándolo al modelo de datos en el que se apoya el SGBD que se va a utilizar.
El modelo relacional es el único modelo que ha permitido abordar la fase de diseño lógico aplicando una teoría formal: el proceso de normalización.
Sin embargo, la normalización no cubre toda esta fase, mostrándose insuficiente para alcanzar todos los objetivos de la misma. En la práctica a veces es preciso proceder a una reestructuración de las relaciones.

El diseño lógico de una base de datos consta de dos etapas: el diseño lógico estándar y el diseño lógico específico. En el diseño lógico estándar, se toma el esquema conceptual resultante de la fase de diseño conceptual, y teniendo en cuenta los requisitos de proceso, de construye un esquema lógico estándar (ELS), que se apoya en un modelo lógico estándar (MLS), que será el mismo modelo de datos soportado por el SGBD a utilizar (relacional, jerárquico, etc.), pero sin las restricciones de ningún producto comercial en concreto. En nuestro caso se utilizará el MLS relacional. Una buena forma de describir el ELS es utilizando el lenguaje estándar del MLS (por ejemplo SQL).
Una vez obtenido el ELS, y considerando el modelo lógico específico (MLE) propio del SGBD a usar (ORACLE, INFORMIX, SQL-SERVER, etc.), se elabora el esquema lógico específico (ELE). Al igual que en el caso anterior, una buena forma de describirlo es utilizando el lenguaje de definición de datos (LDD) del producto especifico utilizado (en el caso de SQL-SERVER, se usará el TRANSACT SQL). El diseño lógico específico está muy ligado a la fase de diseño físico, ya que ambos dependen mucho del SGBD que se utilice.
En la fase de diseño lógico, además de las herramientas ya descritas (MLS, MLE, lenguajes SQL), se disponen de otras que permiten establecer un buen diseño lógico, como por ejemplo la normalización, desnormalización, etc.

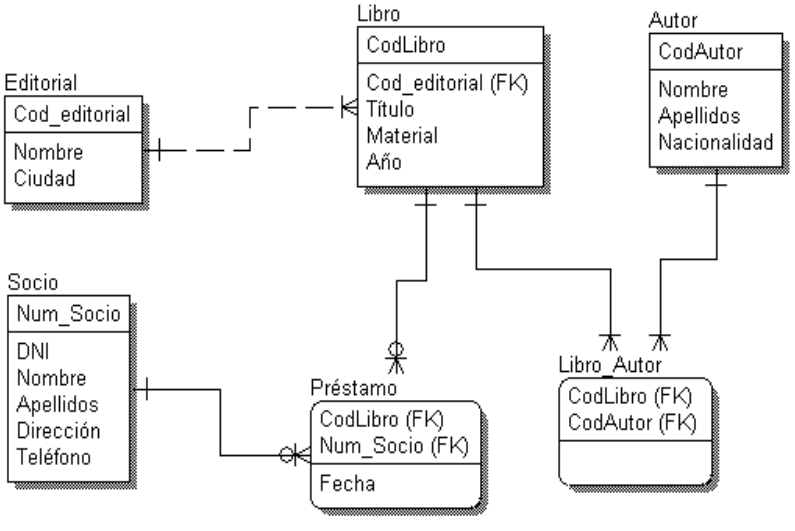
Paso del esquema conceptual al esquema lógico estándar

Lo primero que hay que realizar en la fase de diseño lógico, es obtener el esquema lógico estándar, a partir del esquema conceptual obtenido en la primera fase. Las reglas que permiten pasar del modelo E/R al esquema lógico, son las que a continuación se explican:

Cada entidad se transforma en una relación: esto es, cada entidad genera una tabla, con sus mismos atributos, incluyendo las claves.

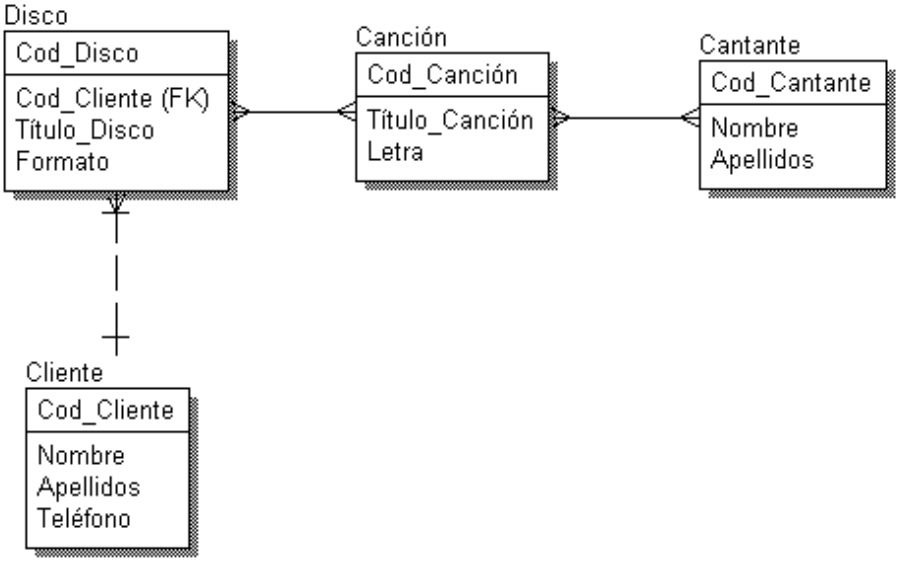
- Cada relación N-M genera una tabla: las relaciones entre entidades con cardinalidad N-M generan una tabla, con los atributos clave de ambas entidades.
- Cada relación 1 -N importa las claves de la entidad con las que se relaciona: cada relación con cardinalidad 1 -N importa los atributos clave que contiene la entidad con cardinalidad N.
- Cada relación dependiente, importa la clave de la otra entidad, como clave.

Para entender mejor el funcionamiento de este método, veamos el paso a tablas del ejemplo visto en el tema anterior acerca de la gestión de una biblioteca. La entidad libro está relacionada con la entidad editorial con cardinalidad 1 -N, por lo tanto importa la clave de la entidad con cardinalidad 1. A su vez, esta relacionada con la entidad autor, pero en este caso, la cardinalidad es N-M, lo que implica que se generará una tabla intermedia, en la que se almacenarán las claves de ambas entidades.
Esta tabla, a la que denominaremos Libro_autor mantiene la información de los códigos de libros junto con los códigos de autores. Posteriormente, si se desea extraer más información, tanto del libro como del autor, se deberá acceder a sendas tablas. Por último se dispone de la entidad Préstamo, que es dependiente tanto de



la entidad Libro como de la entidad Usuario, lo que quiere decir que se generará una tabla, con los atributo1s de la entidad Préstamo además de las claves de las entidades de las que es dependiente, es decir, ISBN y Num socio, que entrarán como claves en dicha tabla. Esta última relación obtenida, mantiene información de qué libros han sido prestados a qué usuarios y en qué fecha. El esquema de las tablas resultantes es el que se muestra en la Figura.

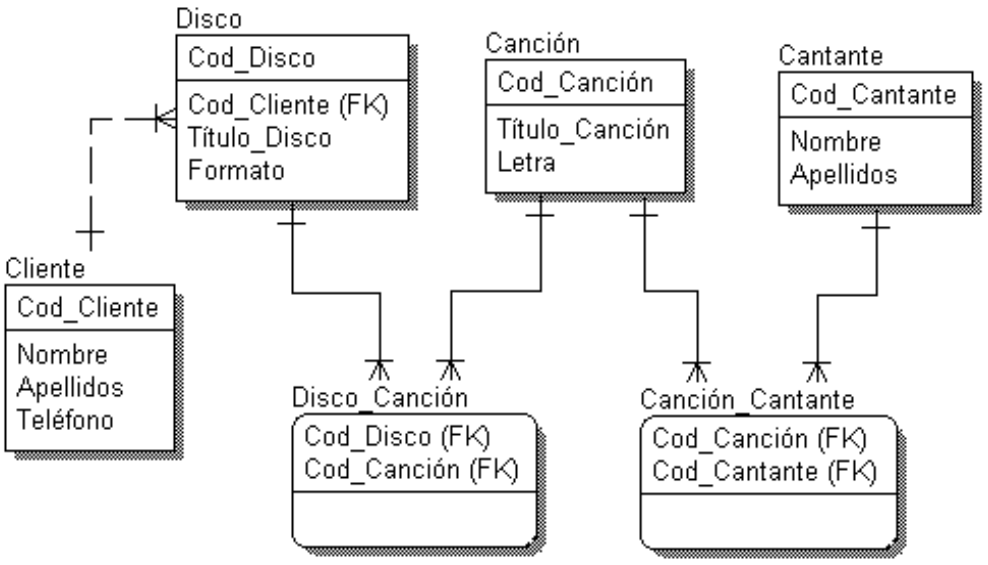
Veamos ahora el paso a tabla de otro ejemplo visto en el tema anterior, cuyo esquema conceptual es el que muestra la Figura.



Empezaremos identificando las relaciones, y concretando las tablas que generarán:

1. Cliente-Disco: puesto que es una relación 1-N, la entidad disco generará una tabla con sus atributos, e importará el atributo clave de la entidad con cardinalidad 1, es decir cod_cliente. A su vez, la entidad cliente generará su propia tabla, con sus propios atributos, es decir, cod_cliente nombre, apellidos y teléfono.
2. Disco-Canción: es una relación N-M, así que, siguiendo el mismo razonamiento anterior, la tabla canción generada importará el cod_disco de la entidad disco.
3. Canción-Cantante: en este caso se tiene una relación N-M, es decir, se generará una tabla intennedia, con los atributos claves de las entidades que relaciona, es decir, cod_canción y cod cantante.
4. Disco_Cantante: siguiendo el mismo razonamiento, al ser una relación N-M, se generará una tabla íntermed ja, con los atributos cod_cantante y cod disco.

Con todo esto, el esquema lógico resultante del esquema conceptual anterior, queda como aparece en la Figura.



TEORÍA DE LA NORMALIZACIÓN

El proceso de normalización consiste en la aplicación de un conjunto de reglas, con el objeto de verificar que el esquema relacional obtenido en esta fase cumple un cierto conjunto de reglas. La normalización se podría considerar prácticamente como el grueso de la fase de diseño lógico, ya que es el encargado de modificar el esquema conceptual obtenido en la fase anterior, para que cumpla el primero de los objetivos de las bases de datos, el de que ha de representar fielmente la realidad. Por lo tanto es el segundo paso a realizar dentro de la fase de diseño lógico, después de la eliminación de valores nulos no aplicables (particionamiento horizontal), y se corresponde con la etapa de estructuración.

La normalización se puede definir como el proceso de sustituir una relación o tabla, por un conjunto de esquemas equivalentes que representen la misma información, pero que no presenten cierto tipo de anomalías a la hora de realizar operaciones sobre ella. Las anomalías que puede presentar una relación son de tres tipos:

- **Anomalías de inserción**: son producidas por la pérdida de información, al no poder insertar filas en una relación, ya que no se conoce el valor de algún atributo no principal (que no es clave). Por ejemplo. Supóngase que se dispone de la siguiente relación, correspondiente a los repartos realizados por distintos proveedores.

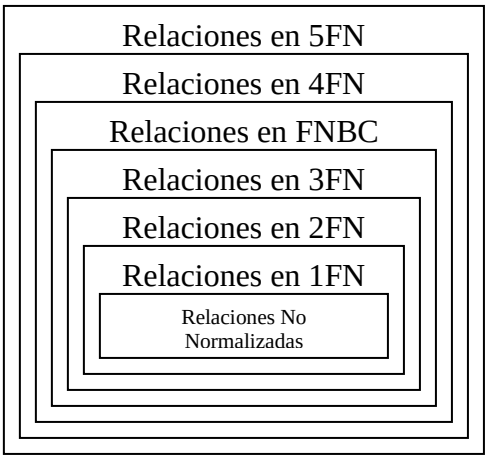
Cod_proveedor	Cod_material	Ciudad	Categoria	Cantidad
P001	A234	Lima	2	122
P001	B297	Chimbote	2	100
P002	0344	Chimbote	3	200
P002	G392	Trujillo	3	310
P003	F893	Ica	1	400

  Si en este momento se desea añadir un nuevo proveedor, no se puede reflejar en la relación, ya que dicho proveedor todavía no ha realizado ningún reparto.

- **Anomalías de borrado**: vienen determinadas por la perdida de información que no se desea, al eliminar una fila de una relación. Por ejemplo, supóngase el caso anterior. Si se desea borrar el reparto realizado por el proveedor 3, al existir únicamente una fila con dicho proveedor, perderemos toda la información relacionada con él, es decir, su ciudad y su categoría.

- **Anomalías de modificación o actualización**: vienen impuestas por la necesidad de propagar actualizaciones, es decir, se debe modificar el mismo atributo en más de un sitio. Son debidas a un diseño redundante. Sigamos con el ejemplo anterior. Si se desea modificar la ciudad de un proveedor, no bastará con hacerlo en un sitio, sino que se deberán recorrer todas las filas correspondientes a todos los repartos de dicho proveedor, y modificar el valor de atributo ciudad en todas ellas. Evidentemente esto es muy peligroso, ya que si nos olvidamos de actualizar dicho valor en una fila, la base de datos quedará inconsistente, es decir, existirán dos valores distintos de ciudad para un mismo proveedor, y eso sin contar el tiempo que se pierde en realizar dicha actualización. Por estos motivos, se pueden establecer dos conclusiones: el diseño es redundante y el esquema no está normalizado.

Por lo tanto, lo que se busca con el proceso de normalización es eliminar estos tres tipos de anomalías. Consiste en conseguir, mediante varios pasos, distintas formas normales. Se dice que un esquema de relación esta en una determinada forma normal si satisface un determinado conjunto de restricciones. Dichas formas normales son la primera forma normal (1 FN), la segunda forma normal (2FN) y la tercera (3FN), definidas por Codd, la forma normal de Boyce-Codd (FNBC), definida por Boyce y Codd, y la cuarta y quinta forma normal (4FN y 5FN), definidas por Fagin. La principal característica que cumple cada una de estas formas normales es que la de nivel superior incluye a la de nivel inferior, es decir, una relación que esté en 2FN estará en 1 FN, una que este en 3FN estará en 1 FN y 2FN, como muestra la Figura.



Una BD relacional estará normalizado cuando esté, al menos, en 3FN.

Para comprender mejor el proceso de normalización y la obtención de las formas normales, veamos el concepto de dependencia funcional.

DEPENDENCIA FUNCIONAL: Sea la relación R, el atributos Y depende funcionalmente de X, o que X determina o implica a Y **si, y sólo si, un solo valor de Y en R está asociado en todo momento a un único valor de X en R, y se representa R.X → R.Y**.

Por ejemplo, si se tiene la relación Empleado, compuesta por los atributos DNI, nombre, salario, etc., se puede asegurar que DNI → Nombre, ya que por cada DNI tenemos un nombre, es decir, para un mismo DNI no pueden existir más de un nombre.

De la misma forma, se puede concluir que entre Salario y Nombre no existe dependencia funcional, ya que para un mismo salario pueden existir varios nombres de empleados, (varios empleados pueden cobrar lo mismo).

La dependencia funcional plena o completa es un caso particular de dependencia funcional. Sea X un atributo compuesto por Xl y X2 (Xl y X2 atributos de X), se dice que Y tiene dependencia funcional plena o

completa de X si depende funcionalmente de X pero no depende de ningún subconjunto del mismo, veamos.

X → Y (Y depende funcionalmente de X)

X1--/-->Y(Y no depende funcionalmente de Xl)

X2--/-->Y (Y no depende funcionalmente de X2)

Por ejemplo, si se tiene la relación Libro con los atributos Cod_Libro, Cod_Socio y Fecha_Préstamo, se tiene:

Cod_Libro, Cod_Socio	→	Fecha_Préstamo
Cod_Libro	--/-->	Fecha_Préstamo
Cod_Socio	--/-->	Fecha_Préstamo

ya que para un libro y un socio sólo existe una fecha de préstamo, mientras que el mismo libro se puede prestar varios días y un mismo socio puede tomar prestado uno o varios libros más de un día. Por estos motivos, se puede concluir que Fecha préstamo tiene dependencia funcional completa de Cod_libro y Cod_socio.

Otro caso particular de dependencia funcional es el tipo de dependencia:

DEPENDENCIA TOTAL:
Es cuando un atributo no clave depende íntegramente o totalmente de la clave principal sea ésta simple o compuesta.

DEPENDENCIA PARCIAL:

Cuando en una tabla con clave primaria compuesta, existen atributos que solo dependen de solo uno de los campos que forman la clave.

DEPENDENCIA TRANSITIVA:

Una dependencia transitiva abarca como mínimo tres componentes. Tal dependencia puede expresarse como:

Q ---> A ----> B

En la cual se dice que B depende de A y que A depende de Q. La transitividad existe debido a que el valor de B depende en la última instancia del valor de Q.

FORMAS NORMALES:

- **FORMA NO NORMALIZADA:**

  Son las variables de la relación en su forma original. Puede ser el documento de partida para la normalización.

  Ejemplo: Normalización de una factura.

  Forma no normalizada:

Variables o campos	**Valores**
Nro_Factura	002-0050160
Nom_Cliente	Comercial La Virreyna SAC
Nº RUC_Cliente	20102050708
Fecha_Factura	20/04/2011
Guía de Remisión del Remitente	001-503
Guia de Remisión del Transportista	003-234
Sub_Total_Factura	900.00
IGV_Factura	162.00
Total_Factura	1,062.00
Nom_Empleado	July
Cantidad	20
Descripción	Chompas de dralón para dama
Precio Unitario	30.00
Importe	600.00
Cantidad	15
Descripción	Pijamas para caballeros
Precio Unitario	20.00
Importe	300.00

Nº Fac	Nom Cliente	NºRUC Cliente	FechFac	Guía Remitent	Guía Trans	Sub TotFac	IGV Fac	Tot Fac	Nom Emp	Cantid ad	Descrip	PU	Import
002-0050160	Comercial La Virreyna	20102050708	24/04/11	001-503	003-234	900.00	162.00	1062.00	July	20 15	Chompas Pijamas	30.0 20.00	600.00 300.00

- PRIMERA FORMA NORMAL:

  Para que una relación esté en primera forma normal (1 FN), debe ser una matriz m por n, donde, ninguna celda de la matriz está vacía y cada tupla tiene una clave que la identifica en forma única.
  Además, se dice que una tabla se encuentra en primera forma normal (1NF) si y solo si cada uno de los campos contiene un único valor para un registro determinado.
  Por tanto, debemos preguntarnos: En la lista de atributos de la FNN, existen variables o campos que tienen mas de un valor, para el mismo registro?

Si la respuesta es afirmativa, se establece la clave primaria y se separan en dos tablas de modo que se cumpla la condición de la 1FN

FACTURA:

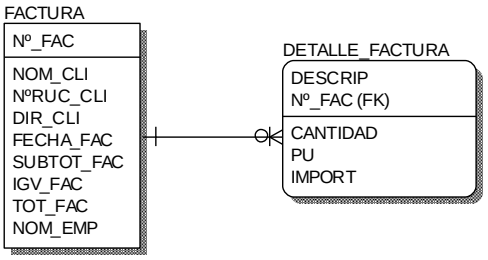
Nº Fac	Nom Cliente	NºRUC Cliente	FechFac	Guía Remitent	Guía Trans	SubTotFac	IGV Fac	Tot Fac	Nom Emp
002-0050160	Comercial La Virreyna	20102050708	24/04/11	001-503	003-234	900.00	162.00	1062.00	July

PK

DETALLE_FACTURA:

PK

Nº Fac	Descrip	Cantidad	PU	Import
002-0050160	Chompas	20	200	2000
002-0050160	Pijamas	15	50	1000



- SEGUNDA FORMA NORMAL:

Una relación está en segunda forma normal (2FN) solamente si todos los atributos son dependientes en forma completa de la clave.

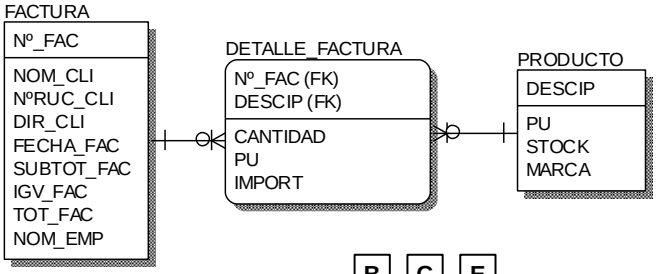
Para que una relación esté en 2FN:

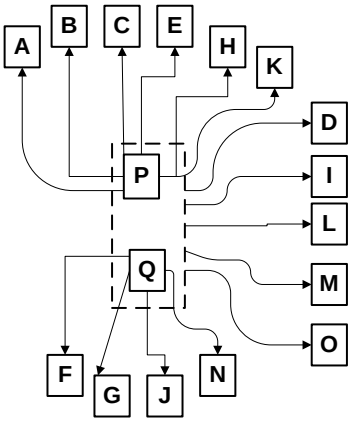
• Debe estar en primera forma normal

• Debe tener una clave compuesta.

Por tanto cualquier relación en primera forma normal que tiene una clave simple, está automáticamente en segunda forma normal. Es decir, la segunda forma normal compara todos y cada uno de los campos de la tabla con la clave definida. Si todos los campos dependen directamente de la clave se dice que la tabla está es segunda forma normal (2NF).

- En el ejemplo podríamos preguntarnos: ¿existen tablas con clave primaria compuesta?.
- Si la respuesta es afirmativa, verificamos la dependencia funcional completa a la clave en ésa tabla.
- Por ejemplo, en la tabla Detalle_factura, el atributo PU depende de la PK o solo depende de la descripción del producto?. Sólo depende de Descrip y no del Nº_Fac.
- Luego los atributos que no tienen una dependencia total (tienen dependencia parcial) forman una tercera Tabla.
- A la nueva tabla se puede agregar campos o atributos que le correspondan.





DETALLE_FACTURA

Nº Fac	Descrip	Cantidad	PU	Import
002-0050160	Chompas	20	200	2000
002-0050160	Pijamas	15	50	1000

FK + FK

PK

PRODUCTO

Descrip	PU	Stock	Marca
Chompas	200	150	Cielo
Pijamas	50	450	HP

PK

Otro ejemplo: Sea la relación R, con los atributos que se indica, donde la clave compuesta está formada por P+Q

R = (**P, Q**, A, B, C, D, E, F, G, H, I, L, M, N, O)

Creando un grafo de dependencias, este grafo que los atributos que parten de P son dependientes solamente de este. De un modo similar los que parten de Q dependen solamente de este último. Solamente aquellos que parten de la línea de trazos que conecta a P y Q tienen dependencia completa de ambos. Esta es la guía para la descomposición.

En consecuencia se generan tres entidades:
P' = (**P**, A, B, C, E, H, K)
Q' = (**Q**, F, G, J, N)
PQ = (**P, Q**, D, I, L, M, O)

- TERCERA FORMA NORMAL:

Se dice que una tabla está en tercera forma normal si esta en 2FN y si los campos de la tabla dependen únicamente de la clave, es decir los campos de las tablas no dependen de otro que no sea primario unos de otros.
Por tanto una relación se encuentra en tercera forma normal (3FN) si no existe DEPENDENCIA TRANSITIVA entre sus atributos y si ya se encuentra en 2 FN.

Recordamos que las propiedades de la segunda forma normal (2Fn) son:

• Tenemos una matriz m x n con un valor determinado para cada componente de cada tupla.

• Cada valor contiene una clave, ya sea simple o compuesta

• Cada componente no clave es dependiente en forma completa de su clave.

En consecuencia es evidente que tenemos, o bien una clave simple, o una clave compuesta de la cual todos los componentes no clave son dependientes en forma completa.

El objeto de esta fase es eliminar todas las dependencias transitivas; la descomposición producirá a continuación sub-relaciones para las cuales no existirán dependencias transitivas

En el ejemplo, la tabla Factura tiene atributos que dependen de otro que no es clave.

FACTURA

Nº Fac	Nom Cliente	NºRUC Cliente	FechFac	Guía Remitent	Guía Trans	SubTotFac	IGV Fac	Tot Fac	Nom Emp
002-0050160	Comercial La Virreyna	20102050708	24/04/11	001-503	003-234	900.00	162.00	1062.00	July

PK

Los atributos: NºRUCCliente y DirCliente, dependen de NomCliente y no de la PK, por tanto éstos atributos forman la tabla Cliente. Ocurre lo mismo con CargoEmp que depende de NomEmp y no la PK, y de igual modo forma otra tabla llamada Empleado. Esta tabla puede también generar otra tabla que podría ser Cargo.

CLIENTE

CodCli	NomCli	RUCCli	DirCli	TelfCli	Ciudad
Cl001	Comercial La Virreyna	20102050708	General Suarez 534	241654	Tacna

PK

EMPLEADO

CodEmp	NomEmp	Apeemp	DirCli	TelfCli	Cargo
Cl001	July Juana	Flores Tisnado	General Suarez 534	241654	Secretaria

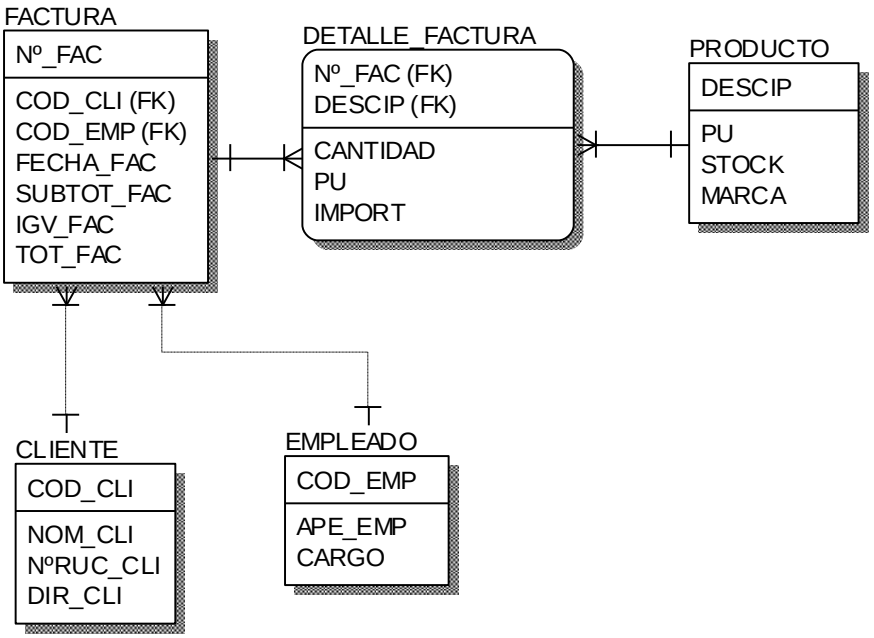
PK

Quedando la tabla factura de la siguiente manera:

FACTURA

Nº Fac	CodCli	FechFac	Guía Remitent	Guía Trans	SubTotFac	IGV Fac	Tot Fac	CodEmp
002-0050160	Cl001	24/04/11	001-503	003-234	900.00	162.00	1062.00	Em001

PK



Ejemplo 2:

Forma No Normalizada:

Código	Nombre	Cursos
1	Marcos	Inglés
2	Lucas	Contabilidad Informática
3	Marta	Inglés Contabilidad

Primera Forma Normal:

Se dice que una tabla se encuentra en primera forma normal (1NF) si y solo si cada uno de los campos contiene un único valor para un registro determinado. Supongamos que deseamos realizar una tabla para guardar los cursos que están realizando los alumnos de un determinado centro de estudios, podríamos considerar el siguiente diseño:

Podemos observar que el registro de código 1 si cumple la primera forma normal, cada campo del registro contiene un único dato, pero no ocurre así con los registros 2 y 3 ya que en el campo cursos contiene más de un dato cada uno. La solución en este caso es crear dos tablas del siguiente modo:

Como se puede comprobar ahora todos los registros de ambas tablas contienen valores únicos en sus campos, por lo tanto ambas tablas cumplen la primera forma normal.

Tabla A	
Código	**Nombre**
1	Marcos
2	Lucas
3	Marta

Tabla B	
Código	**Curso**
1	Inglés
2	Contabilidad
2	Informática
3	Inglés
3	Informática

Segunda forma normal:
La segunda forma normal compara todos y cada uno de los campos de la tabla con la clave definida. Si todos los campos dependen directamente de la clave se dice que la tabla está es segunda forma normal (2NF).
Supongamos que construimos una tabla con los años que cada empleado ha estado trabajando en cada departamento de una empresa:

Código Empleado	Código Dpto.	Nombre	Departamento	Años
1	6	Juan	Contabilidad	6
2	3	Pedro	Sistemas	3
3	2	Sonia	I+D	1
4	3	Verónica	Sistemas	10
2	6	Pedro	Contabilidad	5

Tomando como punto de partida que la clave de esta tabla está formada por los campos código de empleado y código de departamento, podemos decir que la tabla se encuentra en primera forma normal, por tanto vamos a estudiar la segunda:

1. El campo nombre no depende funcionalmente de toda la clave, sólo depende del código del empleado.
2. El campo departamento no depende funcionalmente de toda la clave, sólo del código del departamento.
3. El campo años si que depende funcionalmente de la clave ya que depende del código del empleado y del código del departamento (representa el número de años que cada empleado ha trabajado en cada departamento)

Por tanto, al no depender todos los campos de la totalidad de la clave la tabla no está en segunda forma normal, la solución es la siguiente:

Tabla A	
Código Empleado	**Nombre**
1	Juan
2	Pedro
3	Sonia
4	Verónica

Tabla B	
Código Departamento	**Dpto.**
2	I+D
3	Sistemas
6	Contabilidad

Tabla C		
Código Empleado	Código Departamento	Años
1	6	6
2	3	3
3	2	1
4	3	10
2	6	5

Podemos observar que ahora si se encuentras las tres tablas en segunda forma normal, considerando que la tabla A tiene como índice el campo Código Empleado, la tabla B Código Departamento y la tabla C una clave compuesta por los campos Código Empleado y Código Departamento.

Tercera Forma Normal:

Se dice que una tabla está en tercera forma normal si y solo si los campos de la tabla dependen únicamente de la clave, dicho en otras palabras los campos de las tablas no dependen unos de otros. Tomando como referencia el ejemplo anterior, supongamos que cada alumno sólo puede realizar un único curso a la vez y que deseamos guardar en que aula se imparte el curso.

Código	Nombre	Curso	Aula
1	Marcos	Informática	Aula A
2	Lucas	Inglés	Aula B
3	Marta	Contabilidad	Aula C

Estudiemos la dependencia de cada campo con respecto a la clave código:

- Nombre depende directamente del código del alumno.
- Curso depende de igual modo del código del alumno.
- El aula, aunque en parte también depende del alumno, está mas ligado al curso que el alumno está realizando.

Por esta última razón se dice que la tabla no está en 3NF. La solución sería la siguiente:

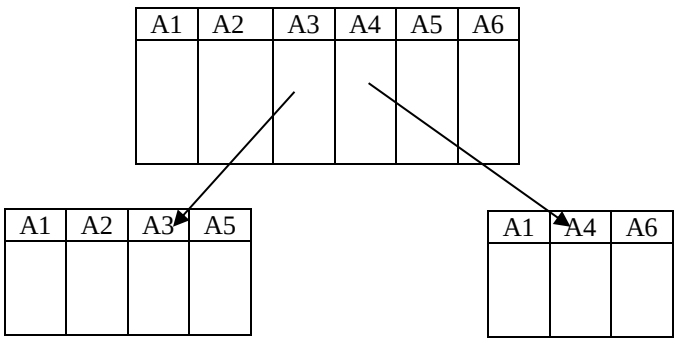
Tabla A		
Código	**Nombre**	**Curso**
1	Marcos	Informática
2	Lucas	Inglés
3	Marta	Contabilidad

Tabla B	
Curso	**Aula**
Informática	Aula A
Inglés	Aula B
Contabilidad	Aula C

Las herramientas para llevar a cabo este objetivo son dos:

- El proceso de normalización
- El particionamiento horizontal de relaciones

El proceso de normalización consiste en sustituir una relación por un conjunto de esquemas equivalentes, que representan la misma información que la relación origen, pero que no presentan cierto tipo de anomalías a la hora de realizar operaciones sobre ella, como se muestra en la Figura, en la que una relación origen ha sido sustituida por otras dos, mediante un proceso de normalización.



Normalización de relaciones

El objetivo de la reestructuración es el de mejorar la eficiencia de la base de datos. En la primera etapa de estructuración de relaciones, se ha propugnado por razones lógicas, normalizar el esquema, así como eliminar los valores nulos mediante un proceso de particionamiento horizontal. Sin embargo, esto no quiere decir que las relaciones que se vayan a almacenar en la base de datos sean las resultantes de estos procesos. ya que se ha logrado el primero de los objetivos de las bases de datos (ser una representación fidedigna del mundo real), pero puede que no el segundo: el de que la base de datos ha de ser un *servidor operacional* y *eficiente de los datos,* por lo que se hace necesaria esta segunda etapa en el diseño lógico. Para lograr este objetivo existen diversos métodos o formas de organizar los datos, considerando esta idea de mejora de la eficiencia, que son:

Relación origen

Cod_doc	Título	Idioma	Editorial
11-23	LP I	Ingles	Lucas Free
33-21	Graf 1	Castell	Vallecito
21-56	Unix	Ingles	Mega Byte
21-43	Graf 2	Italiano	Vallecito

Relacione resultantes

Cod_doc	Título	Idioma	Editorial
11-23	LP I	Ingles	Lucas Free
21-56	Unix	Ingles	Mega Byte

Unión

Selección

Cod_doc	Título	Idioma
11-23	LP I	Ingles
33-21	Graf 1	Castell
21-56	Unix	Ingles
21-43	Graf 2	Italiano

Particionamiento horizontal de relaciones

Diseño físico

El objetivo del diseño fisico, que es la última fase del proceso de diseño, es conseguir una instrumentación lo más eficiente posible del esquema lógico. Aquí se tienen en cuenta aspectos del hardware, requisitos de procesos, características del SGBD, del SO y en general, cualquier factor cercano a la «maquina». Con ello se persigue:

- Disminuir los tiempos de respuesta
- Minimizar espacio de almacenamiento
- Evitar las reorganizaciones
- Proporcionar la máxima seguridad
- Optimizar el consumo de recursos

El principal problema que se plantea en la fase de diseño físico es el debido a la poca flexibilidad de los actuales SGBD, los cuales obligan a trasladar a la fase de diseño lógico, aspectos de carácter fisico, que deberían ser ajenos a dicha fase. Esto obliga a iterar desde la fase de diseño físico a la de diseño lógico, y viceversa, hasta obtener conseguir los objetivos anteriormente expuestos, lo que explica la obligación de la etapa de reestructuración en el diseño lógico.

Como resultado del diseño físico se genera un esquema físico, que es una descripción de la iinplementación de la base de datos en memoria secundaria; describe las estructuras de almacenamiento y los métodos de acceso para acceder a los datos de una manera eficiente. Por ello el diseño físico se genera para un SGBD y un entorno físico determinado.