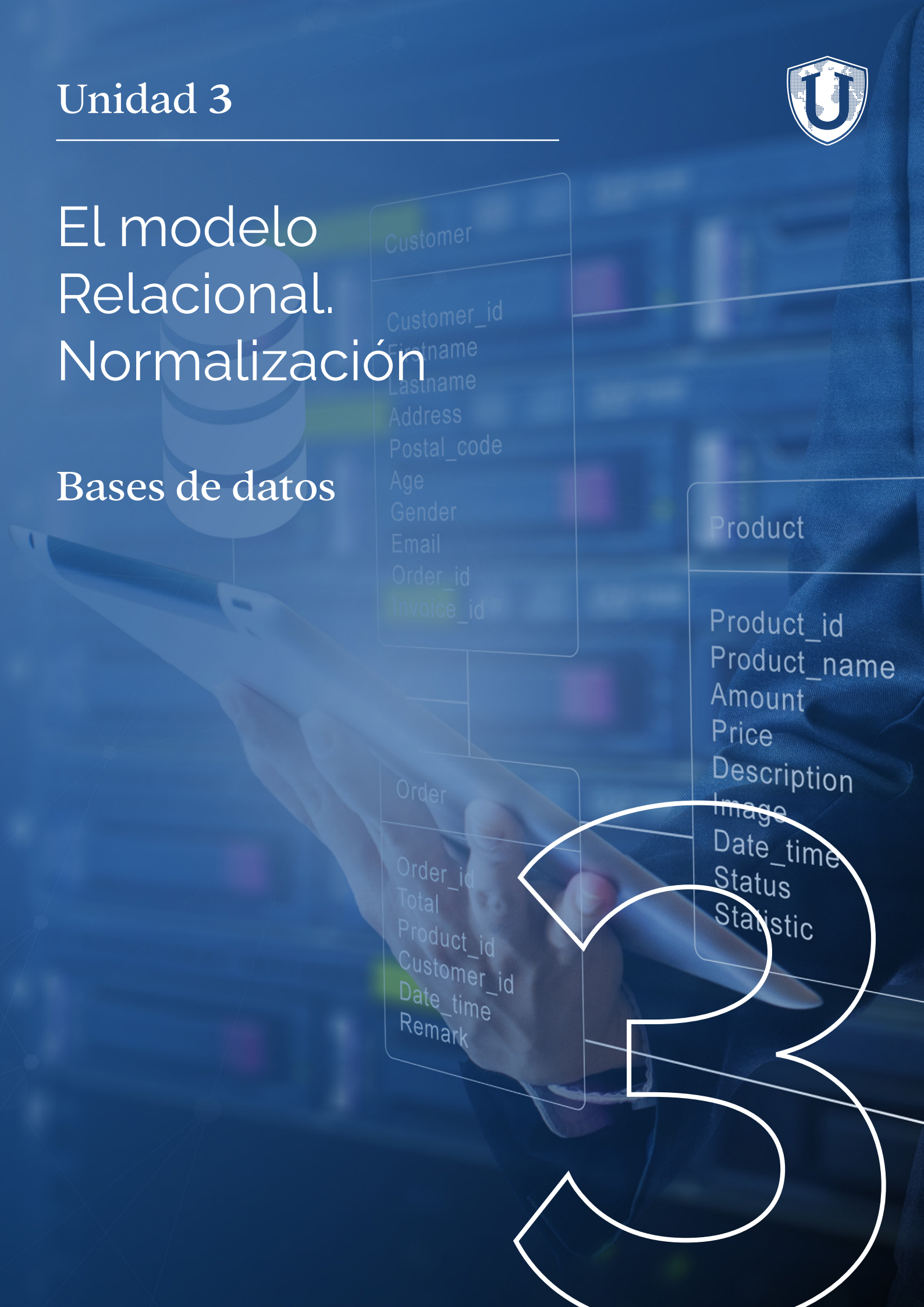
Unidad 3

El modelo Relacional. Normalización

Bases de datos



Índice

Introducción

En esta unidad hablaremos de como E.F. Codd definió el modelo relacional ya que los modelos de bases de datos que se habían planteado con anterioridad eran muy restrictivos. Además, se comenzará a conocer la normalización, que es una técnica para eliminar los duplicados y la redundancia en la información almacenada en las bases de datos.

Cuando hayamos realizado el diagrama entidad/relación (unidad anterior), para poder implementarlo es necesario llevarlo al modelo físico de datos o grafo relacional con la intención de obtener el diseño final de la base de datos con la nomenclatura final.

Además, veremos que el modelo relacional también tratará los índices, las vistas, las restricciones, la definición de usuarios y sus permisos, los criterios de integridad referencial, los accesos concurrentes y las políticas de bloqueo como solución a estos últimos.

Al finalizar esta unidad

- Conoceremos los elementos básicos del modelo relacional.
- Normalizaremos cualquier relación hasta su tercera forma normal.
- Convertiremos un diagrama entidad/relación en un modelo físico de datos.
- Comprenderemos la utilidad y características de los índices en una base de datos relacional.
- Nos familiarizaremos con el concepto de vista.
- Nos plantearemos problemáticas de seguridad en nuestras bases de datos.

3.1. El modelo relacional

El modelo relacional surgió de la necesidad de encontrar una solución a todos los problemas que se nos presentaban debido a la rigidez que imponían las bases de datos jerárquicas.

Si hablamos en términos puramente matemáticos, que de ahí es de donde vienen, tenemos que su principal característica es la independencia de los datos para su tratamiento.

En el modelo relacional tenemos un elemento básico que es la relación, una tabla o estructura en dos dimensiones donde representamos la información almacenada en las entidades e incluso ciertas relaciones establecidas en el modelo entidad/relación.

Las filas de una relación son equivalentes a las ocurrencias de una entidad, ya que cuentan con una serie de atributos cada uno con un valor distinto. Un ejemplo:

Valor

Atributo

NIF	Nombre	Cargo	Teléfono	Horas trabajadas
10111011A	Alba	Jefe	606 000 000	3,4
20220022B	Bertín	Empleado	607 000 000	5
33030330C	Carlos	Empleado	608 000 000	6,7
40044404D	Diana	Jefe	609 000 000	8
50555055F	Ernesto	Jefe	610 000 000	9,2

Tupla

Imagen 1. Relación.

El dominio de un atributo es el conjunto de valores que puede tomar para una misma ocurrencia. Este dominio tiene dos vertientes, una en la que realmente solo puede hablar de los valores posibles a los que hace referencia el atributo, por ejemplo, el dominio del atributo 'nombre' en una relación sería el conjunto de los nombres de todos los elementos de la relación. De otro modo, de manera general el dominio se usa para referirnos al conjunto de combinaciones posibles que podría constituir un dominio, es decir, el tipo de datos. Dentro de las bases de datos tenemos los siguientes principales tipos de datos:

- Números enteros.
- Números decimales.
- Cadenas de caracteres.
- Fechas y horas.
- Valores lógicos (verdadero o falso).
- Objetos (ficheros binarios).

Hay veces en las que se construirán conjuntos de valores a medida expresa.

Para terminar, hay que tener en cuenta que existe un valor que tiene peso por sí mismo, el valor nulo o NULL que nos habla de la ausencia de valor alguno, que no es lo mismo que un valor vacío o 0 en campos numéricos.

3.2. Normalización

Cuando se creó el modelo relacional se crearon una serie de normas que nos ayudan a eliminar las duplicaciones de información o los datos redundantes con el objetivo de simplificar al máximo las relaciones.

La técnica de aplicación de dichas normas se llama normalización y se basa en la transformación de las relaciones a unos ciertos estados denominados formas normales.

Formas normales hay varias, aunque de manera normal suele bastar con llegar hasta la tercera forma normal, de igual modo, vamos a ver las principales formas normales a continuación:

3.2.1. Primera forma normal (1FN)

Una relación estará en primera forma normal si todos sus campos son atómicos, que esto hace referencia a que cada campo tiene un valor único, no un conjunto de valores. Por ejemplo, aquí tenemos un ejemplo donde vemos que el atributo "Nombre de jefe", no es atómico, porque para un mismo registro, hay varios valores:

Relación sin normalizar			
Código de empleado	**Nombre**	**Apellidos**	**Nombre de jefe**
35353535A	Berto	Canales Díaz	Encarna Felipe Gregorio
68686868H	Ignacio	Jeréz Martínez	Laura
90909090N	Oscar	Pérez Quirón	Ramiro Santiago

Una solución en primera instancia consiste en atomizar el campo separándolo de modo que sean varias tuplas con la misma información a diferencia de este atributo.

Si continuamos con el ejemplo anterior vemos que queda del siguiente modo:

Relación en 1FN			
Código de empleado	**Nombre**	**Apellidos**	**Nombre de jefe**
35353535A	Berto	Canales Díaz	Encarna
35353535A	Berto	Canales Díaz	Felipe
35353535A	Berto	Canales Díaz	Gregorio
68686868H	Ignacio	Jeréz Martínez	Laura
90909090N	Oscar	Pérez Quirón	Ramiro
90909090N	Oscar	Pérez Quirón	Santiago

Pero claro, esta solución va a hacer que haya una redundancia aun mayor, porque hará que todos los demás campos se repitan muchas veces y de manera totalmente directa invalidará la clave primaria. Para poder contrarrestar esto, lo que se sugiere es la siguiente solución, crear una segunda relación donde la clave primaria estará formada por el campo que queremos que sea atómico y la clave primaria de la original (solo tendrá estos dos campos) y dejar la relación inicial igual, pero eliminando el campo que hemos establecido en la nueva relación. Estas relaciones se relacionarán mediante la clave primaria de la primera de ellas. Si seguimos de nuevo con el ejemplo anterior, separaríamos dos relaciones distintas, una donde los atributos serían "Código de empleado", "Nombre" y "Apellidos"; y otra donde serían los atributos "Código de empleado" y "Nombre de jefe":

Relaciones en 1FN		
Código de empleado	**Nombre**	**Apellidos**
35353535A	Berto	Canales Díaz
68686868H	Ignacio	Jeréz Martínez
90909090N	Oscar	Pérez Quirón

Código de empleado	Nombre de jefe
35353535A	Encarna
35353535A	Felipe
35353535A	Gregorio
68686868H	Laura
90909090N	Ramiro
90909090N	Santiago

La primera forma normal es de obligatorio cumplimiento para la formación del modelo relacional, si todos los campos no son atómicos, no se puede continuar con el diseño de la base de datos.

3.2.2. Segunda forma normal (2FN)

Para que se cumpla la segunda forma normal, 2FN, una relación debe de cumplir con lo siguiente:

- Está en 1FN
- Todos y cada uno de los atributos que no sean clave primaria y no formen parte de esta deben depender directamente de ella al 100%.

Un ejemplo sería el siguiente, donde la descripción del producto no depende del número de factura y del código del producto, sino simplemente del código del producto, porque para la factura es indiferente:

Relación en 1FN, pero no en 2FN			
Número de Factura	Código de Producto	Cantidad	Descripción
3	5	7	Trapo
9	1	4	Uralita (Pieza)
7	0	4	Vaso
8	2	7	Yeso (Saco)

Volvemos a tener la necesidad de separar la relación en otras dos para que todos los campos dependan de su clave primaria. Ahora en el ejemplo lo que haremos será separar en dos relaciones, una donde están ambas claves primarias y el atributo "Cantidad" que depende de ambas y otra donde solo estará "Código de producto" y "Descripción":

Relaciones en 2FN		
Número de Factura	Código de Producto	Cantidad
3	5	7
9	1	4
7	0	4
8	2	7

Código de Producto	Descripción
5	Trapo
1	Uralita (Pieza)
0	Vaso
2	Yeso (Saco)

Como podemos ver, la segunda forma normal solo se incumplirá si la clave primaria es compuesta, porque siempre que tengamos una relación en 1FN con un solo atributo como clave primaria, esta automáticamente también estará en 2FN.

3.2.3. Tercera forma normal (3FN)

Para que una relación se encuentre en tercera forma normal debe de cumplir lo siguiente:

- La relación se encuentra en 2FN.
- Los atributos que no formen parte de la clave primera no necesitan de ninguno de los otros para que su información esté completa.

En el siguiente ejemplo, la relación está en 2FN, pero no en 3FN porque como podemos ver, el atributo "Provincia" depende del atributo "Código de provincia" y no de la clave primaria, que es "DNI":

Relación en 2FN, pero no en 3FN				
DNI	**Nombre**	**Apellidos**	**Código de provincia**	**Provincia**
28282828Z	Almudena	Benítez Cáceres	4	Gipuzkoa
18181818H	Iñaki	López Marín	5	Sevilla
31313131T	Uriel	Vázquez Yébenes	9	Ávila
99999999B	Carolina	Díaz Fernández	1	Huelva

Para solucionar esto constituimos una relación nueva con el atributo que no depende de la clave primaria donde la clave primaria será el atributo del que depende además del primero. Igual que las dos veces anteriores, separamos en dos relaciones distintas, la primera con todos los atributos menos el que no dependía directamente de la clave primaria, que era "Provincia" y la segunda con "Código de provincia" y "Provincia":

Relaciones en 3FN			
DNI	**Nombre**	**Apellidos**	**Código de provincia**
28282828Z	Almudena	Benítez Cáceres	4
18181818H	Iñaki	López Marín	5
31313131T	Uriel	Vázquez Yébenes	9
99999999B	Carolina	Díaz Fernández	1

Código de provincia	Provincia
4	Gipuzkoa
5	Sevilla
9	Ávila
1	Huelva

3.2.4. Otras formas normales

En un primer momento solo se definieron las reglas para establecer las tres primeras formas normales y por eso son las únicas realmente obligatorias como hemos comentado antes. Pero con el paso de los tiempos, Boyce.Codd amplío esta normalización con la cuarta forma normal (4Fn), la quinta forma normal (5FN) y la sexta forma normal (6FN).

Estas tres formas adicionales son tan restrictivas que a la hora de la verdad son opciones no recomendadas para el diseño de las bases de datos para entornos reales de producción.

3.3. Paso del Diagrama E/R al modelo físico de datos

Cuando hablamos del modelo entidad/relación y de la normalización tenemos la necesidad de hacer lo más eficiente posible el diseño de nuestra futura base de datos.

La implementación relacional propone que la información se almacene en tablas con diversos campos y registros como sustitutos de entidad, tupla y atributo, que coincidían también con los elementos del modelo entidad/relación.

El siguiente cuadro nos ayuda a establecer dicha comparativa:

Comparación entre elementos del paradigma relacional			
Diagrama entidad-relación	**Entidad o relación**	**Ocurrencia**	**Atributo**
Modelo relacional	Relación	Tupla	Atributo
Modelo físico de datos	Tabla	Registro	campo

Aquí tenemos un ejemplo de la relación vista en el primer punto transformada en tabla:

3.3.1. Nomenclatura

En el modelo físico hay una nomenclatura muy bien definida que se debe seguir sin excepción alguna, Esta nomenclatura que se basa realmente en el código de la programación para estructurase es la siguiente:

- **Nombres de tablas.** Van a llevar siempre el prefijo "T" mayúscula, como TCLiente.
- **Nombres de campo.** Se usará una letra minúscula como prefijo que nos indique de que tipos de datos es el campo siguiendo estos tipos básicos:
  - **Números:** "n", de number. (nCoste).
  - **Cadenas de caracteres:** "c", de carácter. (cDireccion).
  - **Fechas:** "d", de date. (dInicio).
  - **Valores lógicos:** "l", de logical. (lSoltero).
  - **Objetos:** "o", de object. (oCancion).
- **Nombres (genérico).** Debemos de evitar el usar letras que pertenezcan a un alfabeto local, de hecho, solo se aceptan como válidos los siguientes caracteres:
  - **Letras del alfabeto inglés en mayúscula y minúscula.**
  - **Números.**
  - **Signos de guion bajo ("_").**

  El primer carácter de un nombre siempre debe de ser una letra.

- **Identificadores.** Son los campos que se suelen crear para actuar como clave primaria y que suelen llamarse "ID" porque son únicos, por eso se le pondrá además del prefijo que necesite, el sufijo "ID" como puede ser un código de empleado que quedaría como "nEmpleadoID".

Si tenemos en cuenta la cantidad de datos que se pueden almacenar quizá se nos quede corta la nomenclatura a portada, pero nos puede establecer las bases para saber diferenciar de manera correcta los distintos elementos del modelo físico de datos.

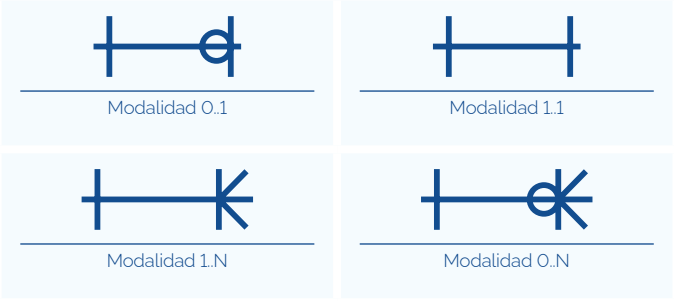
3.3.2. Reglas de transformación

Para transformar el modelo relacional al diseño físico de una base de datos, primero vamos a ver gráficamente como se representa:

- Para representar las tablas usaremos un rectángulo donde el nombre de la tabla irá en un margen superior delimitado. Después de este límite pondremos los campos en orden de arriba a abajo donde la clave primaria irá subrayada y en primera posición. Por ejemplo:

Imagen 2. Representación gráfica de una tabla.

- Para representar la modalidad se usará directamente las líneas que unen como relación las distintas tablas, y dependiendo de la modalidad queda del siguiente modo:



Hay muchos casos en los que las relaciones entre distintas tablas se convertirán en una tercera tabla asociada a las dos por su complejidad y para nombrar dichas tablas se van a seguir los siguientes criterios:

- Usaremos un sustantivo para nombrar la acción que se realiza en la relación.
- Se usará una concatenación de los nombres de las dos tablas para referirnos a la tercera. Por ejemplo, alumnos usan ordenadores generaría la tabla "TAlumnosOrdenadores".

Para transformar los componentes del modelo entidad/relación al modelo físico de datos se realiza siguiendo las siguientes reglas:

- Las entidades se transforman en tablas.
- Los atributos pasan a ser campos y con ello sigue igual la clave primaria.
- Dependiendo de su cardinalidad, las relaciones se transformarán en distintos aspectos con la intención de evitar la nulidad:
  - **Relaciones M:N, ternarias y n-arias.** La relación se convierte en una tabla nueva y la clave primaria será la composición de las claves primarias de las dos entidades relacionadas, es decir, tendrá una clave compuesta como en el siguiente ejemplo:

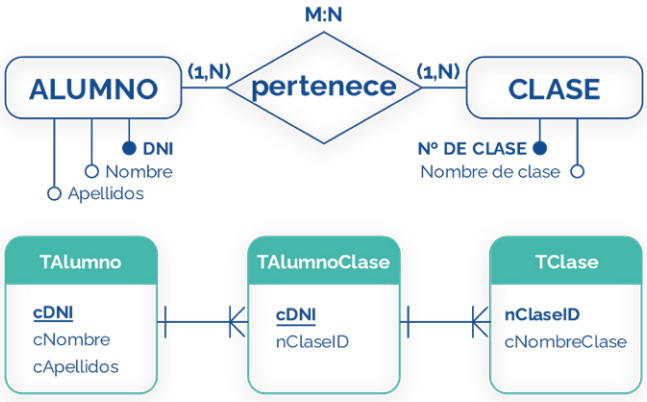


Imagen 3. Relación M:N

  - **Relaciones 1:N.** Tenemos tres posibilidades:
    - Si existen atributos de relación, se creará una tabla intermedia con las claves primarias de ambas entidades y los atributos de relación, como en el ejemplo siguiente:

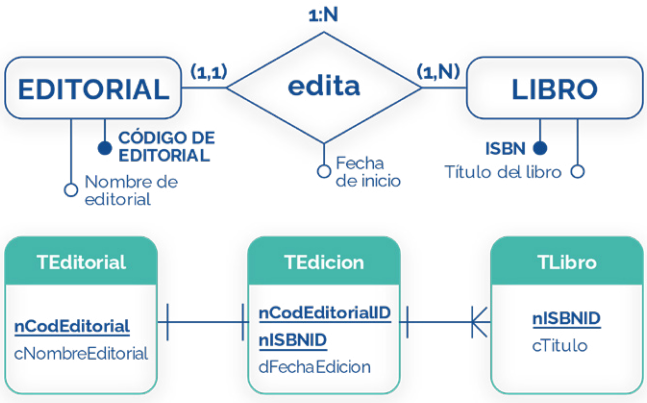


Imagen 4. Relación 1:N

Puede que se dé la situación en la que algún atributo deba pasar a ser parte también de la clave primaria debido a que es necesario para identificar correctamente la tabla en cuestión. Esto pasa en el ejemplo anterior ya que es posible que hayan varias ediciones del mismo libro, con el mismo ISBN y lógicamente la editorial es la misma, por lo que tendremos que indicar que la fecha de edición también es un valor necesario a la hora de identificar la tabla intermedia:

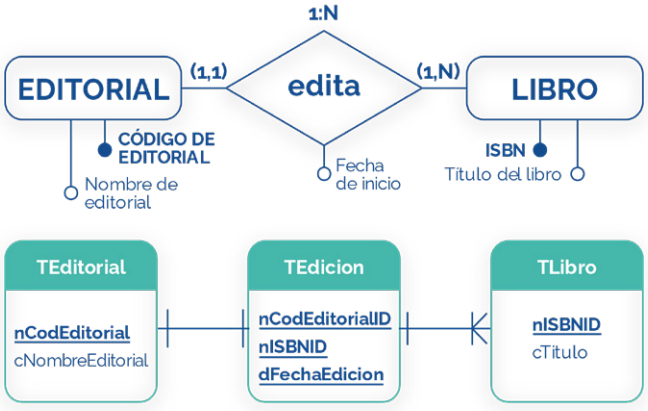


Imagen 5. Relación 1:N con atributos de relación en la clave.

Cuando tengamos este tipo de problemática será conveniente que sea el analista de la base de datos quien tome la decisión, porque habrá muchas veces en las que no estará claro del todo debió a la dimensión temporal de la información que una base de datos aporta.

» De otro modo, si la modalidad de alguna de las dos entidades es (1,1) y además no existen atributos, se borra la relación y se establecerá la línea que hemos dicho más arriba. Además, la clave primaria de la entidad que tiene la modalidad máxima 1 pasará como campo a la tabla que generará la entidad con modalidad máxima N, a modo de clave foránea. Un ejemplo donde vemos que la clave foránea de TCoche, que es cDNIID, es la clave primaria en TCliente:

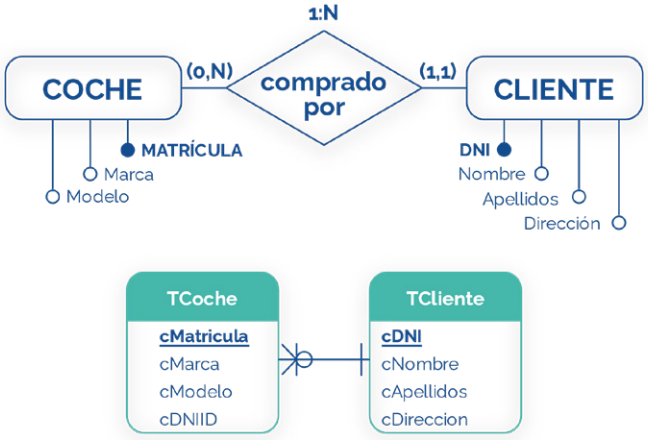


Imagen 6. Relación 1:N con modalidad 1.1

» Si tenemos que, en un lado de la cardinalidad, la modalidad es (0,1), se crea de nuevo una tabla con ambas claves primarias como la clave primaria de la tercera tabla. Como en el siguiente ejemplo, donde no podemos propagar CDNI a TCoche porque podría darse el caso en el que un coche no sea comprado por un cliente, lo que generaría valores nulos:

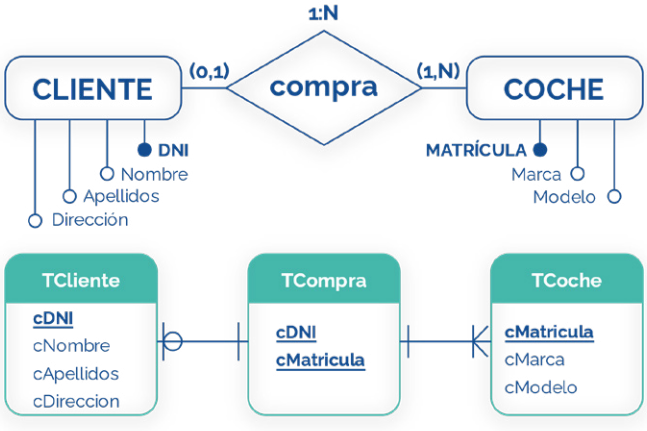


Imagen 7. Relación 1:N con modalidad 0..1

- **Relaciones 1:1.** Esta es la relación más compleja de todos, debido a que tenemos que intentar que no se quede nada sin valor. Esta gestión va a necesitar de un estudio detallado por parte del analista para tomar la decisión final, pero se pueden establecer ciertas reglas:

  - Si en ambos lados la modalidad es (0,1), entonces generaremos una tabla con ambas claves primarias componiendo la clave primaria de la tabla generada como en el siguiente ejemplo donde un capataz solo puede estar dirigiendo una obra o puede que ninguna, por cualquier proceso, y una obra no tienen por qué estar dirigida por un capataz de manera obligatoria. Si observamos bien, vemos que la propagación de cualquiera de ambas claves primarias como clave foránea podría llevar a valores nulos en cualquier tabla, por eso se usa una tercera:

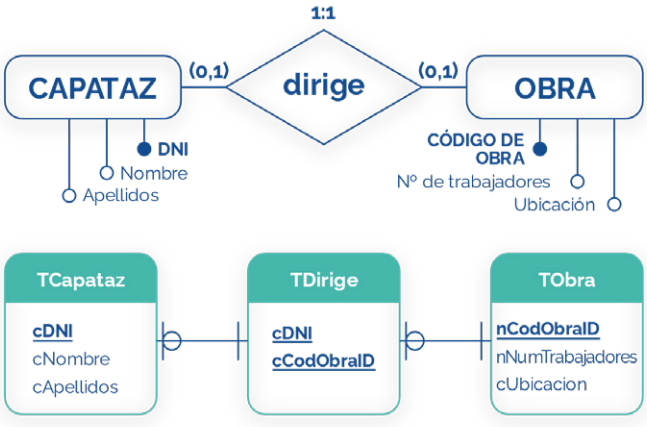


Imagen 8. Relación 1:1 con modalidades 0..1

  - Por otro lado, si una de las modalidades es (0,1) y la otra (1,1), pasaremos la clave primaría de la entidad don modalidad (1,1) como clave primaria también de la otra entidad y se establece una relación del modelo físico de datos como los arriba nombrados. Aquí se ejemplifica:

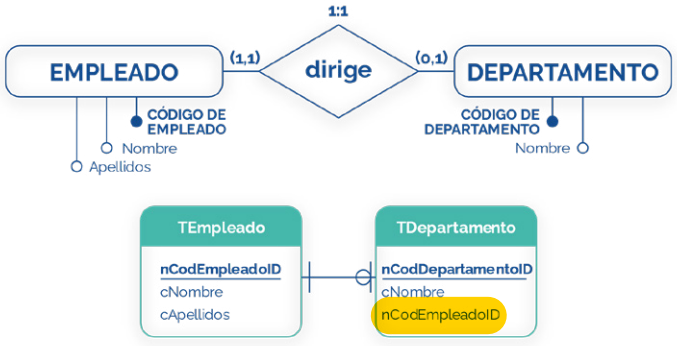


Imagen 9. Relación 1:1 con modalidad 0..1

» Si tenemos que las dos modalidades son (1,1), pero una de las entidades es débil, se realiza lo mismo que antes entre la entidad fuerte y la débil. Como en el siguiente ejemplo:

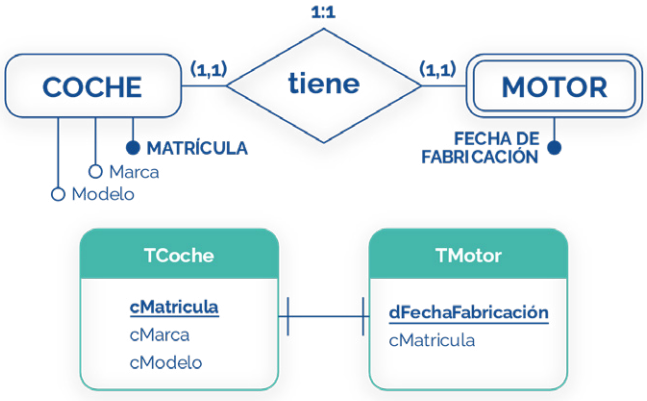


Imagen 10. Relación 1:1 con entidad débil.

» Si tenemos que las dos modalidades son (1,1) y además no hay entidad débil o si existe un atributo de relación, entonces tendremos que ver cuál de las opciones es la mejor a la hora de implementar el modelo físico de datos.

> Cuando hacemos una conversión de los elementos correspondientes al diagrama entidad/relación extendida, podemos tener una pérdida de semántica directamente relacionada. Para que no suceda esto se dispone de algunos casos de ciertos mecanismos o herramientas que nos controlen la base de datos final.

Al igual que antes, ahora sigue sin recomendarse su uso del todo debido a que con el modelo entidad/relación normal, es suficiente para todas las necesidades.

De igual modo, si se necesita, las siguientes reglas son las aplicables:

» Las relaciones que antes teníamos pasan a ser relaciones normales y a su posterior traspaso. Como en el siguiente ejemplo se ve:

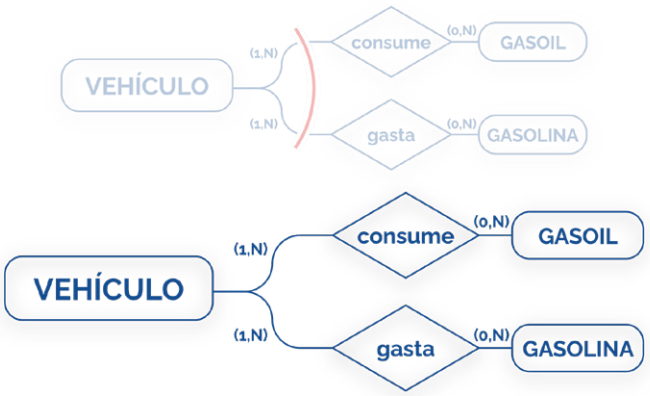


Imagen 11. Tratamiento de relaciones exclusivas.

» Las relaciones que eran de jerarquía pasan a ser relaciones 1:1 con modalidades (0,1) y (1,1) en el supertipo. Un ejemplo sería:

Imagen 12. Tratamiento de relaciones jerárquicas.

> Por otro lado, en el caso de la agregación deberemos tener en cuenta cada caso como una relación aparte, estudiando cada uno de los pasos y haciendo la correspondiente transformación.

3.3.3. Ejemplo de transformación de Diagrama E/R a modelo físico de datos

Vamos a usar el diagrama entidad/relación que hemos diseñado en la unidad anterior como ejemplo:

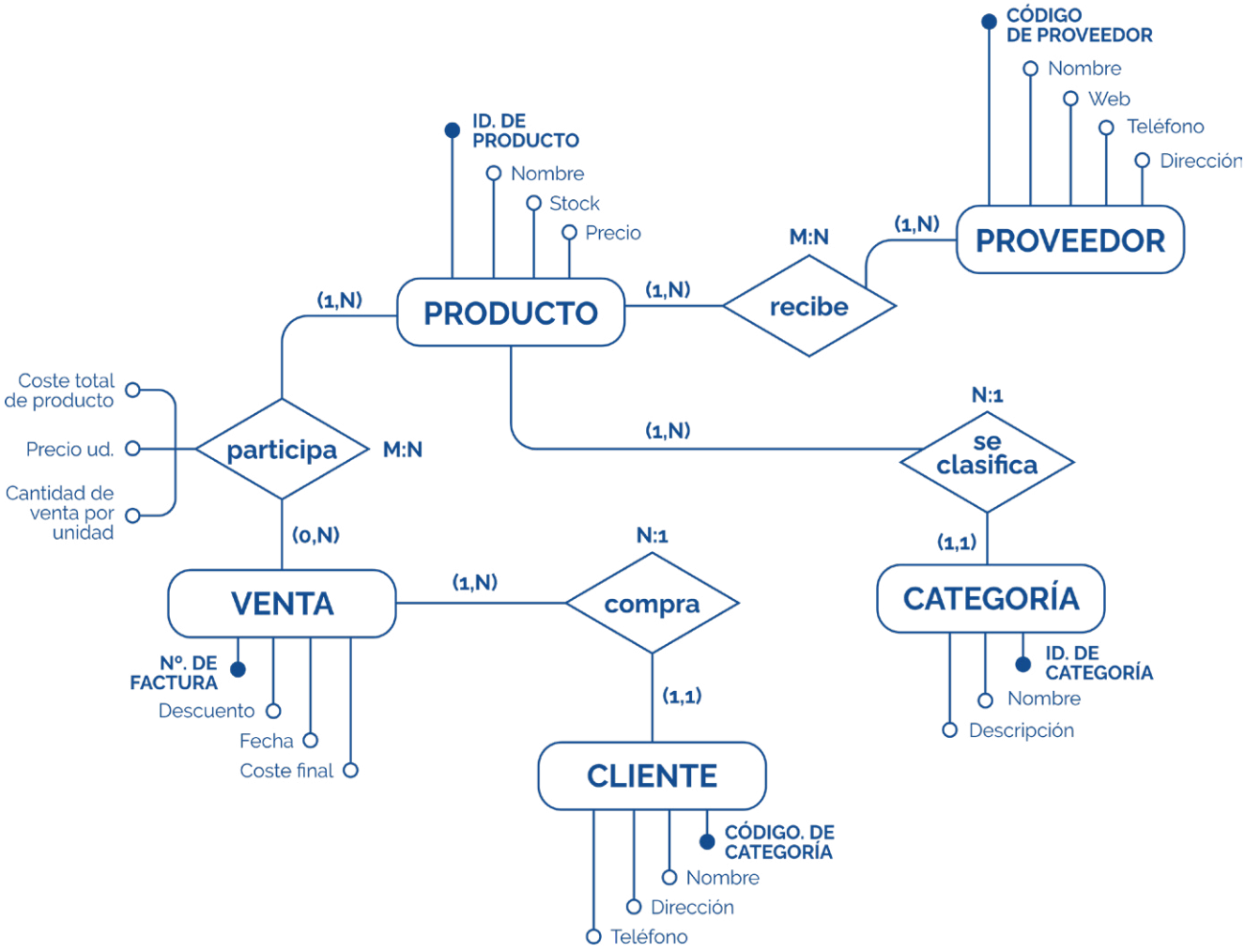


Imagen 13. Ejemplo de diagrama entidad/relación.

Ahora, si lo pasamos al modelo físico de datos quedaría del siguiente modo:

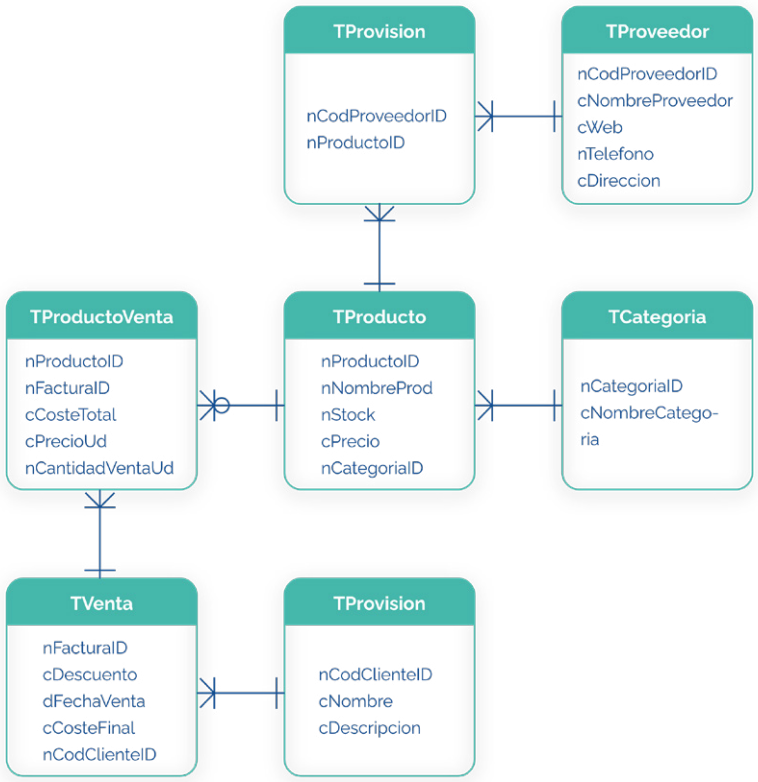


Imagen 14. Ejemplo de modelo físico de datos.

Podemos ver con claridad que ha pasado lo siguiente, lo primero las transformaciones de entidades a tablas:

- La entidad PROVEEDOR se ha transformado en la tabla TProveedor.
- La entidad PRODUCTO se ha transformado en la tabla TProducto.
- La entidad CATEGORIA se ha transformado en la tabla TCategoria.
- La entidad VENTA se ha transformado en la tabla TVenta.
- La entidad CLIENTE se ha transformado en la tabla TCliente.

Ahora, las relaciones se han transformado del siguiente modo:

- Entre PROVEEDOR y PRODUCTO había una relación N:M, por lo que se ha creado una tabla intermedia TProvision con ambas claves primarias.
- Entre PRODUCTO y CATEGORIA había una relación N:1 con cardinalidad (1,N) y (1,1) respectivamente. Es por esto por lo que se ha propagado la clave primaria de TCategoria como clave ajena en TProducto.
- Entre PRODUCTO y VENTA había una relación N:M con atributos de relación, por lo que se ha creado una tabla intermedia TProductoVenta con ambas claves primarias y los atributos de relación se han transformado en campos de esta tabla intermedia.
- Entre VENTA y CLIENTE había una relación N:1 con cardinalidad (1,N) y (1,1) respectivamente. Se ha propagado la clave primaria de TCliente como clave ajena de TVenta.

3.4. Otras consideraciones sobre el modelo relacional

3.4.1. Índices

Las bases de datos distinguen el orden en el que se organiza la información del orden en el que se muestra en pantalla de manera general. Es por esto por lo que el modelo general nos presta ciertas herramientas para que los dos casos estén diferenciados. Como se vio en la unidad 1 con los ficheros indexados, las tablas de la base de datos relacional también tienen ficheros que indexan la información asociados a ellos. Esto se realiza fundamentalmente para que podamos acceder a ella ordenadamente.

Siempre va a haber mínimamente un fichero que nos sirva de índice para relacionar cada uno de los registros con su clave primaria correspondiente. Es una vez que tenemos todos estos datos cuando el diseñador tendrá que definir los índices necesarios siguiendo unos mínimos conceptos:

- **Campos por los que se van a ordenar las consultas más habituales.**
- **Campos accedidos con mucha frecuencia.** La única función no es la de ordenar los registros en grupos, también nos ayuda a buscar ciertos valores con mucha más eficacia y rapidez.
- **Claves ajenas.** Es bastante común que las consultas que se lancen sobre la base de datos, que se verán más adelante, relacionen varias tablas a la vez.

Por mucho que los índices nos aportan numerosas ventajas, también nos crea una redundancia sobre la base de datos y nos ralentiza en ciertas acciones relacionadas con los datos, por lo que es importante que no abusemos de esta técnica.

3.4.2. Vistas

Las vistas son consultas sobre una o varias tablas que tenemos almacenada en la base de datos. Estas vistas se muestran como otra tabla más, pero a nosotros nos ayuda a tener un nivel mayor de seguridad a la hora de acceder a la información recogida en la base de datos. Además, como ya están definidas de manera previa en la base de datos, son bastante más eficientes que las consultas lanzadas a mano de manera puntual.

Pero a la hora de la verdad hay multitud de usuarios que no las usan, sobre todo en el campo de la programación debido a que estas vistas no aceptan parámetros como una consulta normal. Esto quiere decir que no pueden ejecutar la acción para valores distintos dentro de campos de las tablas.

Para no usar vistas, se suele implementar algún otro mecanismo de seguridad.

3.4.3. Restricciones sobre campos

Como veremos en la próxima unidad, una de las principales restricciones sobre los campos es elegir que tipos de datos se tienen que implementar por campo, pero también tenemos las siguientes restricciones definidas:

- **UNIQUE.** No se pueden repetir valores en los campos. La clave primaria de manera estricta debe de ser única, pero no todos los campos UNIQUE formarán la clave primaria.
- **NOT NULL.** Con estas restricciones se prohíben los valores nulos, también la clave primaria tiene que ser NOT NULL de manera obligatoria.
- **DEFAULT.** Siempre que no se asigne un valor concreto a un campo, en vez de dejarlo en NULL se le asignará un valor que se haya predefinido.

3.4.4. Integridad referencial

A la hora de trabajar con tablas que se relacionan mediante el uso de claves como hemos visto previamente nos presenta el problema de la integridad de los datos almacenados en la tabla. A la hora del diseño de una base de datos debemos de tomar ciertas decisiones con respecto a la integridad referencial de los datos. Tenemos cuatro maneras distintas de abordar este problema:

- **Prohibir la operación:** la más restrictiva de todas las decisiones. Aquí se trata de que no se pueda borrar ni modificar los registros que tengan cierta coincidencia con claves ajenas. Hay varias maneras de implementarla en la base de datos, desde que de un error hasta que no se ejecute y sea transparente para el usuario.
- **Transmisión en cascada:** la más usada, porque lo que realiza es que tanto la modificación como el borrado de registros que se encuentren relacionados afecta a todos ellos.
- **Puesta a nulo:** Para poder borrar o modificar contenido de las tablas se pone en valor nulo o NULL todos los valores de la clave ajena a la que afecte. Lógicamente, no puede estar activa en dichas claves la restricción NOT NULL. No debemos de usar esta herramienta porque nos aleja de uno de nuestros propósitos principales en una base de datos, que no haya valores nulos en la medida de lo posible.
- **Uso de valor por defecto.** Se puede tanto borrar como modificar registros, pero las claves ajenas deben de tener un valor por defecto asociado.

3.4.5. Usuarios y privilegios

En los SGBD pueden usar ciertas estructuras de seguridad basadas en los distintos permisos otorgados por el administrador a los usuarios. Para la gestión de estos permisos hay que tomar ciertas decisiones:

- Crear usuarios en la base de datos genéricos como roles para los usuarios físicos que accedan al SGBD o crear un usuario por cada persona que acceda a este sistema.
- Si creamos un usuario para cada usuario físico, también podemos crear perfiles para cada rol que se desarrolle en la base de datos de modo que se agrupen usuarios en cada perfil para por ejemplo el administrador el analista, etc.
- Una vez que tengamos las dos decisiones anteriores tomadas, tenemos que decidir a qué objetos de la base de datos van a poder acceder. Una vez que sepamos a que objetos accederán, tenemos que implementar también las acciones que podrán realizar sobre dichos objetos. Las bases de datos permiten mucha diversidad en esto, porque permiten incluso acceso a algunos campos y a otros no dentro una misma tabla.
- Por último, puede que tengamos la casuística en la que los usuarios que acceden a la base de datos no sean usuarios físicos sino de software que derivan de aplicaciones con las que el usuario físico interactúa para el acceso a la base de datos.

Aquí tenemos un ejemplo de definición de perfiles y usuarios en una base de datos:

```
Perfil:       cDNI
Usuarios:     Admin
Permisos:     Lectura, escritura, creación, borrado y backup sobre
              todas las bases de datos.
              Creación, modificación y borrado de perfiles y usuarios.
              Administración del servidor.

Perfil:       Desarrollo
Usuarios:     MartinezL, LopezM, GomezJ
Permisos:     Lectura, escritura, creación, borrado y backup sobre
              todas las bases de datos

Perfil:       RRHHBecario
Usuarios:     MelladoF
Permisos:     BD Trabajadores:  Lectura sobre nSalario de TEmpleado
                                Lectura y escritura sobre resto de tablas

Perfil:       Operaciones
Usuarios:     GarciaP, GonzalezA, AlonsoA
Permisos:     BD Trabajadores:  Lectura sobre TEmpleado excepto
                                nSalario y cCuentaBancaria
              BD Operaciones:   Lectura sobre TClientes
                                Lectura y escritura sobre resto de tablas
```

Imagen 15. Definición de usuarios y perfiles en un SGBD.

3.4.6. Accesos concurrentes

Una de las mayores problemáticas que presenta una base de datos es que puede que se dé el caso de un acceso simultáneo a los datos por distintos usuarios.

3.4.7. Políticas de bloqueo

Para solucionar el problema anteriormente planteado además de ciertos problemas que veremos más adelante, el DBA deberá de implementar ciertas políticas de bloqueo que consisten en el bloqueo de algunos elementos de la base cuando se realizan actualizaciones que les afecten directamente. Como con los permisos, se permite el bloqueo de distintos elementos y esto prohíbe el acceso de los usuarios a dichos elementos, por lo que hay que llevar un cuidado especial a la hora de usar dichas políticas.

Hay dos tipos principales de bloqueos:

- **Compartidos:** se permite la consulta al valor del elemento bloqueado, pero se prohíbe su modificación.
- **Exclusivas:** se puede tanto leer como modificar el valor del elemento bloqueado.

www.universae.com