T03. Modelo relacional. Normalización. Paso a tablas.

Objetivos.

- Conocer los elementos básicos del modelo relacional.
- Normalizar cualquier relación hasta su tercera forma normal.
- Convertir un diagrama entidad/relación en un modelo físico de datos.
- Comprender la utilidad y características de los índices en una base de datos relacional.
- Familiarizarse con el concepto de vista.
- Plantearse problemáticas de seguridad relacionadas con restricciones, integridad referencial, definición de usuarios y perfiles, accesos concurrentes y políticas de bloqueo.

3.1. Modelo Relacional.

Definido por Edgar F. Codd en el IBM Research Center de San José (California) en 1970, el modelo relacional buscaba una solución a los problemas derivados de la rigidez estructural de las bases de datos jerárquicas y en red imperantes en la época. Presentado en sólidos términos matemáticos, independiza los datos de su tratamiento.

El elemento básico del modelo relacional es la ***relación***, una estructura bidimensional que representa las entidades y algunas relaciones del diagrama Entidad/Relación. Las filas de una relación, llamadas ***tuplas***, corresponden a las ocurrencias; cada tupla cuenta con una serie de ***atributos***, cada uno de ellos con un ***va1or***. Vemos como ejemplo la relación “Socio”:

NIF	Nombre	**Apellidos**	Teléfono	Fech nacimiento	Fech alta
26012245Y	Luis	Pérez García	952121212	12/01/02	16/09/22
51244002B	Pedro	Lozano Ríos	956421242	25/05/00	05/07/22
78545125A	Ana	Gómez Saez	666182456	06/10/99	20/09/22
01251423Z	Patricia	Barrón López	636925522	23/08/85	27/08/22

Relación: La tabla completa.

Tupla: toda la información de un socio. Ej: 01251423Z, Patricia, Barrón López, 636925522, 23/08/85, 27/08/22.

Atributo: Cada uno de los datos que se toma de cada socio. Ej: Apellidos.

Valor: Cada dato de cada socio. El contenido de la intersección tupla y atributo. Ej: Barrón López.

Llamamos ***dominio de un atributo*** al conjunto de valores que puede tomar un atributo para una ocurrencia concreta. Aunque de acuerdo a la definición teórica del modelo relacional cada dominio corresponde exclusivamente a los valores posibles. Por ejemplo: el dominio del atributo "Apellidos" es el conjunto de apellidos válidos para los socios, luego en cualquier momento el conjunto de todos los apellidos de socios que tenga en la relación será un subconjunto de ese conjunto de apellidos válidos.

La idea de dominio típicamente se asocia con la de ***tipo de datos*** (es decir, el conjunto de valores posibles que puede tener un tipo de dato, es decir, su dominio y las operaciones a realizar con ellos. En el caso del atributo "Apellidos" el tipo de dato más apropiado es: cadena de caracteres, este tipo nos permitirá cualquier combinación de caracteres aunque no correspondan a ningún apellido válido.

Se identifican los siguientes tipos de datos base:

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

También se pueden definir conjuntos de valores a medida. Un ejemplo sería un atributo "día de la semana", cuyos posibles valores fueran los contenidos en el conjunto ("lunes", "martes", "miércoles", "jueves", "viernes", "sábado", "domingo").

Hay un valor especial que constituye un tipo por sí mismo: el ***valor nulo o NULL***, que representa ausencia de valor. No se debe confundir con el valor vacío ("" en un campo de tipo cadena de caracteres, o 0 en un campo numérico).

El inglés Edgar Frank "Ted" Codd está reconocido de forma unánime como el padre de las bases de datos relacionales. Preocupado por la tendencia de los fabricantes de SGBDR a obviar elementos fundamentales de su propuesta de modelo relacional, a mediados de los años ochenta aportó una serie de definiciones conocidas como "las 12 reglas de Codd" (realmente son trece, numeradas del cero al doce) que todo SGBDR debe cumplir. A pesar de que algunas de ellas son muy complejas de seguir, la mera existencia de las 12 reglas supuso una declaración de intenciones sobre el camino que debía tomar el mundo de las bases de datos.

3.1.1 Condiciones que ha de cumplir una relación.

Toda relación o tabla ha de cumplir:

1. Contener un solo tipo de tupla. Cada uno tiene un número fijo de atributos con su correspondiente nombre. La base de datos estará formada por muchas tablas de forma que cada tipo de tupla será de una tabla diferente.
2. Dentro de una relación no puede haber tuplas repetidas.
3. Dentro de una relación no puede haber atributos repetidos.
4. El orden de las tuplas en una tabla no está determinado.
5. El orden de los atributos en una tabla no está determinado.

3.1.2. Integridad de las relaciones.

Ningún componente de clave primaria de una relación puede aceptar valor Nulo.

Para conectar dos relaciones utilizamos la denominada **Clave ajena**, que se puede definir de la siguiente forma: dadas dos relaciones: R1 y R2, se define clave ajena como un atributo de R2 tal que:

- Sus valores son del mismo dominio que los valores de la clave primaria de R1.
- La clave ajena puede ser nula o su valor coincidir con un valor de la clave primaria de R1
- La clave ajena no tiene que ser un componente de la clave primaria de R2.
- La clave ajena puede aceptar Null.

Debemos tener en cuenta que R1 y R2 no necesariamente son distintos, caso de relaciones reflexivas.

Llamaremos a R2 (relación que contiene la clave ajena) relación referencial y a R1 (relación que contiene la clave primaria) relación objetivo o referenciada.

3.1.3. Integridad referencial.

Cualquier estado de la base de datos que no satisfaga la regla de integridad referencial será incorrecto por definición.

Ante una operación que lleve al sistema a un estado incorrecto cabe la posibilidad de que el sistema rechazara dicha operación; o que la acepte y realice otras acciones necesarias para mantener la base de datos en un estado correcto.

De cualquier manera, el diseñador de la base de datos es quien decide qué operaciones se aceptan o no. Y en el caso de aceptarse, qué operaciones de compensación se han de hacer para mantener un estado correcto, si fuera el caso.

Para mantener la integridad referencial debemos plantear las siguientes preguntas a la clave ajena:

- **¿Puede tener nulos?** La respuesta dependerá del problema en concreto que se esté resolviendo. Ejemplo: no tiene sentido que en un PEDIDO que no exista el proveedor. Sin embargo, si en EMPLEADO tengo una clave ajena: “Titulación oficial” que nos lleva a otra relación: TITULO. Puede haber algún empleado que no tenga dicha titulación, pues no le hizo falta para empezar a trabajar o se formó por otro medio.

- ¿Que debería suceder si se intenta borrar el objetivo de una referencia de clave ajena? Volviendo al ejemplo del pedido: ¿Qué ocurriría si se intenta borrar un proveedor del que hay algún pedido? Podemos tener tres posibilidades:
  - Restringida: Solo se permite borrar si no tiene ningún pedido. Es decir, ninguna clave ajena apunta a ese proveedor.
  - Se propaga (cascada) La operación genera acciones de compensación propagándose en cascada. En el caso de borrar un proveedor que eliminan todos los pedidos a ese proveedor.
  - Establecer como nulo. Se asigna valor Null a todas las claves ajenas que tienen como objetivo la clave primaria de la otra relación borrada. No es aplicable si la clave ajena no acepta nulos.
- ¿Que debería suceder si se intenta modificar el objetivo de una referencia de clave ajena? Repetimos la pregunta al ejemplo del pedido: ¿Qué ocurriría si se intenta modificar un proveedor del que hay algún pedido? Podemos tener tres posibilidades:
  - Restringida: Solo se permite modificar si no tiene ningún pedido. Es decir, ninguna clave ajena apunta a ese proveedor.
  - Se propaga (cascada) La operación genera acciones de compensación propagándose en cascada. En el caso de modificar la clave primaria de un proveedor, se modificarán todas las claves ajenas de ese proveedor en pedidos.
  - Establecer como nulo. Se asigna valor Null a todas las claves ajenas que tienen como objetivo la clave primaria de la otra relación modificada. No es aplicable si la clave ajena no acepta nulos.

3.2. Normalización.

También a Codd se debe la definición de una serie de normas cuya aplicación elimina las redundancias de información en una solución relacional. La técnica es conocida como normalización, y consiste en llevar todas las relaciones a determinados estados llamados formas normales. A continuación se describen dichas formas normales y cómo se llega hasta ellas.

Antes de mostrar el proceso de normalización debemos conocer los conceptos de: dependencia funcional y dependencia funcional plena.

Dependencia funcional: Dados dos atributo X e Y de una relación R, se dice que Y depende funcionalmente de X si , y solo si, para cualquier valor de X la relación R tiene un único valor Y. Se representa : $X \rightarrow Y$

Dependencia funcional plena: Se dice que un atributo Y tienen dependencia funcional plena de un conjunto compuesto X (X_1, X_2,..) si es funcionalmente dependiente de X y no de ninguno de sus posibles subconjuntos de X.

Primera forma normal. 1FN.

Una relación esta en primera forma normal (1FN) si todos sus valores son atómicos, es decir, cada valor de los dominios de todos los atributos es único. En el siguiente ejemplo (clave primaria en negrita) vemos un atributo, “Teléfono”, cuyos valores no son atómicos, es un atributo multievaluado:

NIF	Nombre	Apellidos	Teléfono
26012245Y	Luis	Pérez García	952121212 646252325
51244002B	Pedro	Lozano Ríos	956421242 854484848 666002001
78545125A	Ana	Gómez Saez	666182456
01251423Z	Patricia	Barrón López	636925522

Para alcanzar la 1FN consiste en atomizar el atributo “Teléfono”, del siguiente modo: Repetir la tupla por cada aparición de un atributo no atómico. Esta solución implica una fuerte redundancia (“Nombre” y “Apellidos” se repiten por cada teléfono), e invalida a “NIF” como clave primaria, obligando a establecer como clave primaria el atributo “Teléfono”.

Dichas redundancias se eliminarán en próximas etapas del proceso de normalización, en concreto al pasarlo a 3FN.

NIF	Nombre	Apellidos	**Teléfono**
26012245Y	Luis	Pérez García	952121212
26012245Y	Luis	Pérez García	646252325
51244002B	Pedro	Lozano Ríos	956421242
51244002B	Pedro	Lozano Ríos	854484848
51244002B	Pedro	Lozano Ríos	666002001
78545125A	Ana	Gómez Saez	666182456
01251423Z	Patricia	Barrón López	636925522

La 1FN es parte de la definición del modelo relacional, por lo que su cumplimiento es obligatorio.

Segunda forma normal. 2FN.

Una relación está en segunda forma normal (2FN) si cumple las siguientes reglas:

- Está en 1FN.
- Si todos los atributos que no forman parte de la clave primaria tienen dependencia plena de esta. Es decir, todos los atributos que no forman parte de la clave primaria dependen de ella por completo.

La relación siguiente ilustra el *stock* de una librería. La clave primaria está compuesta por dos atributos: “Código de libro” y “Código de tienda”, pero el atributo “Dirección” no depende de toda la clave, sino únicamente del atributo “Código de tienda”. Por ese motivo se repite la dirección de la tienda 9, con la consiguiente redundancia de información:

Cód libro	**Cód tienda**	Cantidad	Dirección
215	9	5	C/ Blas de Lezo
710	7	3	C/ Mármoles
215	5	4	C/ Jaboneros
152	9	1	C/ Blas de Lezo

En este caso, el proceso de normalización obliga a dividir la relación en dos, una con la información de la tienda y otra con la del *Stock:*

Cód libro	**Cód tienda**	Cantidad
215	9	5
710	7	3
215	5	4
152	9	1

Cód tienda	Dirección
9	C/ Blas de Lezo
7	C/ Mármoles
5	C/ Jaboneros

Al dividir la tabla se ha de tener en cuenta que:

- Primera tabla: Clave primaria completa (compuesta) y sólo los atributos que dependan totalmente de ella. Es decir, se da una dependencia funcional plena con la clave primaria de todos los atributos que no son clave primaria.
- Segunda tabla: Parte de la clave primaria de la que dependa algún/os atributo/s y dicho/s atributos.

Nótese que la 2FN solo se puede violar si la clave primaria esta compuesta por más de un atributo, por lo que toda relación en 1FN cuya clave primaria esté formada por un solo atributo también está en 2FN.

Ejercicio: Explicar por qué la siguiente relación no está en 2FN y convertirla a 2FN:

Cód Película	**Cód Socio**	Título	Fecha préstamo	Nombre
BEL-214	251	Pearl Harbour	20/09/22	José Luís Cabello
COM-023	188	Notting Hill	19/10/21	Ana Aurora Baeza
DRA-001	251	Brave heart	25/11/21	José Luís Cabello
BEL-214	402	Pearl Harbour	08/09/22	Juan Donanire

Tercera forma normal. 3FN.

Una relación está en tercera forma normal (3FN) si cumple las siguientes reglas:

- Esta en 2FN.
- Todos los atributos que no forman parte de la clave primaria son independientes entre sí. Es decir, ningún atributo no clave primaria depende funcionalmente de otro, excepto del campo clave.

El siguiente ejemplo ilustra una relación con información sobre empleados. Todos los atributos dependen directamente de la clave primaria (“Código de empleado”) excepto “Nombre de departamento”, que depende funcionalmente del “Código de departamento”:

Cód Empleado	Nombre	Apellidos	Dirección	Cod Dpto del emplead.	Nombre Dpto	Fecha nac
25	Vicente	Carrera García	C/Lope Rueda, 23	4	Nóminas	20/07/98
18	Luisa	Vázquez Valero	C/Niquel 12	7	Contabilidad	10/04/01
75	Fernando	Casas Lobato	C/Plata 9	4	Nóminas	08/05/90
32	Elena	Troya Egea	C/Peso la harina, 5	3	Almacén	01/10/97

La solución pasa por separar los atributos que dependen de otro atributo que no es clave primaria a tabla/s distinta/s:

- Primera tabla: Clave primaria de la tabla original con todos los atributos que dependen directamente de ella.
- Otras tablas:
  - Clave primaria: atributo del que dependan otros.
  - Atributos: todos los atributos dependientes de este.

La información sobre el departamento constituirá una nueva relación:

Cód Empleado	Nombre	Apellidos	Dirección	Cod Dpto empleado	Fecha nac
25	Vicente	Carrera García	C/Lope Rueda, 23	4	20/07/98
18	Luisa	Vázquez Valero	C/Niquel 12	7	10/04/01
75	Fernando	Casas Lobato	C/Plata 9	4	08/05/90
32	Elena	Troya Egea	C/Peso de la harina, 5	3	01/10/97

Cod Dpto	Nombre Dpto
4	Nóminas
7	Contabilidad
3	Almacén

Volviendo al ejemplo del campo multievaluado del teléfono en el que al atomizarlo en la 1FN aparecían redundancias en: NIF, Nombre y Apellidos. Al pasar a 3FN los atributos de Nombre y Apellidos dependen de NIF y no de Teléfono, por lo que pasarían a una tabla aparte.

NIF	Nombre	Apellidos
26012245Y	Luis	Pérez García
51244002B	Pedro	Lozano Ríos
78545125A	Ana	Gómez Saez
01251423Z	Patricia	Barrón López

NIF	**Teléfono**
26012245Y	952121212
26012245Y	646252325
51244002B	956421242
51244002B	854484848
51244002B	666002001
78545125A	666182456
01251423Z	636925522

Otras formas normales

Inicialmente Codd definió las reglas relativas a las tres primeras formas normales, que todo diseño relacional debe cumplir. No obstante más adelante se amplió la teoría de la normalización con la forma normal de Boyce-Codd (FNBC), cuarta forma normal (4FN), quinta forma normal (5FN), forma normal de dominio/clave (DKNF) y sexta forma normal (6FN). De enorme contenido teórico, fuerzan restricciones que hacen su aplicación poco recomendable en muchos entornos reales.

3.3. Paso del DER al modelo Físico de Datos.

El objetivo del diagrama entidad-relación y las técnicas de normalización es proporcionar el mejor diseño posible para una futura base de datos. En una implementación relacional la información se almacena en forma de *tablas* con *campos y registros*.

Estableciéndose la siguiente comparación entre: DER, Modelo relacional y Modelo Físico de Datos:

Diagrama Entidad-Relación	Entidad o Relación	Ocurrencia	Atributo
Modelo relacional	Relación	Tupla	Atributo
Modelo Físico de Datos	Tabla	Registro	campo

3.3.1. Nomenclatura

A partir de este momento vamos a definir una nomenclatura común a todos los elementos de modelo físico y base de datos referidos. Para ello se usara como punto de partida la notación húngara, definida en los años 70 por Charles Simonyi, programador húngaro de Xerox.

Simonyi estableció unas reglas para nombrar las variables que aportaba información sobre su ámbito y tipo de datos. Veamos una adaptación reducida según los siguientes criterios:

- *Nombres de tablas.* Se prefijaran con una "T" mayúscula (TEmpleado, TCliente).
- *Nombres de campo.* El prefijo, una letra minúscula, indicara el tipo de datos del campo, de acuerdo a los siguientes tipos básicos:
  - Números: "n" (nCantidad, nTotal), de *number.*
  - Cadenas de caracteres: "c" (cNombre, cCiudad), de *character.*
  - Fechas: "d" (dNacimiento, dAlta), de date. No es necesario especificar la palabra "fecha" (dFechaNacimiento sería redundante).
  - Valores lógicos: "1" (lsexo, lPensionista), de *logical.*
  - Objetos: "o" (oFoto, oDocumentoXML), de *object.*
- *Nombres* (genérico). Se evitara el uso de caracteres de alfabetos locales, como la "ñ", la "ç", las vocales acentuadas o con diéresis, etc. El primer carácter de un nombre siempre será una letra y únicamente se aceptaran como válidos los siguientes caracteres:
  - Letras del alfabeto inglés en mayúscula y minúscula.
  - Números.
  - Signo de subrayado o guión bajo ("_").
- *Identificadores.*Todo campo creado para identificar de forma unívoca los registros de una tabla (código de cliente, código de libro, código de país) llevara como sufijo las letras " ID" (identificador) en mayúscula (nClienteID, nLibroID, nPaisID) .

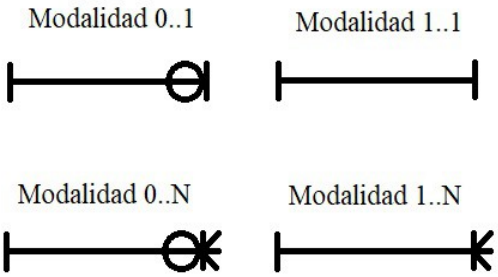
Charles Simonyi:Tras su andadura en Xerox, trabajó en Microsoft, donde supervisó el desarrollo de los herramientas Word y Excel a principios de los años ochenta. Con once patentes publicadas. Actualmente se dedica a la filantropía de forma activa y es el único turista espacial en repetir su aventura.

3.3.2. Reglas de transformación.

Se propone la siguiente representación gráfica de modelo físico de datos:

- **Las tablas** se representan como un rectángulo con el nombre de la tabla en la parte superior y la lista de campos en la inferior. Los campos que conformen la clave primaria irán subrayados y en orden. En las tablas se representarán todas las entidades y algunas relaciones.

- **La modalidad** irá implícita en la terminación de las líneas que relacionan tablas, del siguiente modo:



Debido al aspecto de la línea que representa la modalidad 1..N, esta notación del modelo físico de datos es conocido como “de pata de gallo”*(crow’s foot*, literalmente en inglés "de pata de cuervo").

En algunos casos, las relaciones generan tablas. A la hora de dar nombre a dichas tablas se pueden seguir dos criterios:

- Utilizar un sustantivo que represente la acción implícita en la relación (“cliente contrata servicio” generaría una tabla TContrato, “usuario compra producto” generaría TCompra).
- Concatenar los nombres de las entidades relacionadas (“profesor imparte asignatura” generaría TProfesorAsignatura).

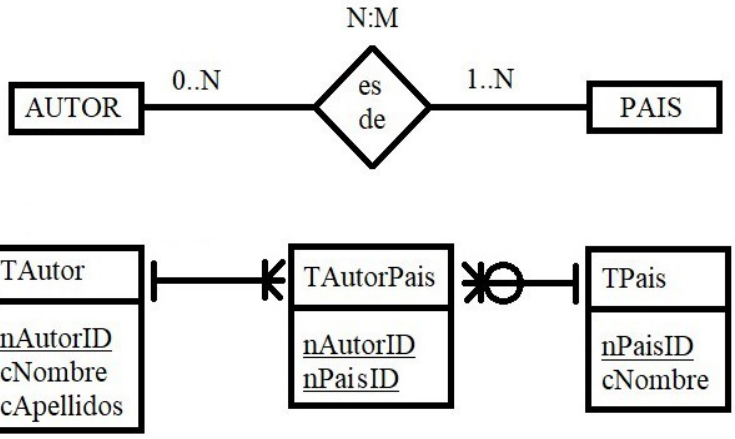
La elección de nombres de tablas, campos, variables,.. es una tarea crucial de cara a la correcta comprensión del diseño de un sistema de información. Se dice que un programador debe dedicar más tiempo a pensar qué nombre dar a una variable que el que otro programador tardaría en descubrir para qué se emplea dicha variable.

La transformación de componentes del diagrama Entidad-Relación en elementos del modelo físico de datos sigue las siguientes reglas:

1. Toda **entidad** se convierte en una tabla.
2. Todo **atributo** pasa a ser un campo. Los atributos marcados como parte de la clave primaria se convierten en campos de la clave primaria de la nueva tabla, conservando su orden.

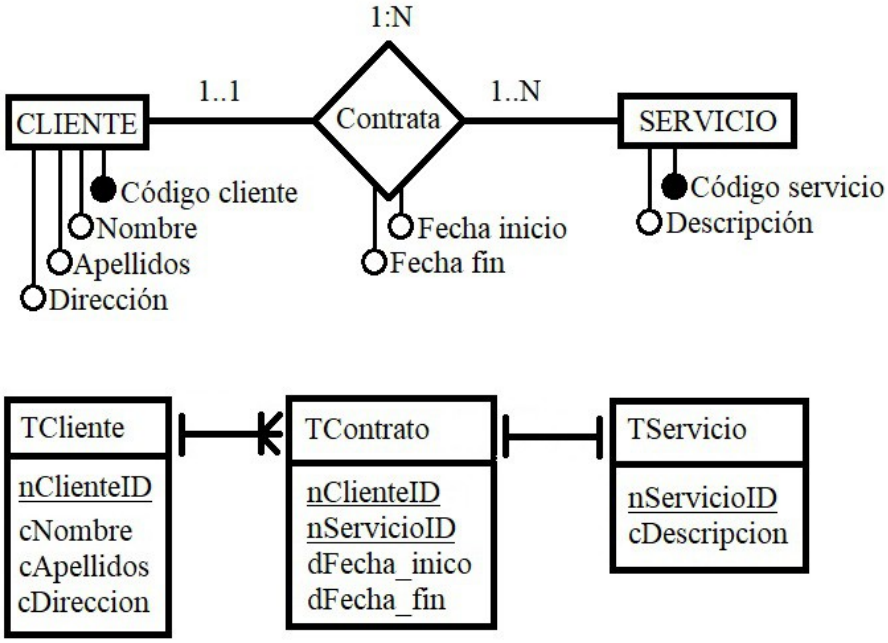
3. Las **relaciones** presentan una casuística basada en su cardinalidad. Uno de los objetivos fundamentales a la hora de diseñar una base de datos es evitar la proliferación de valores nulos.

   a) ***Relaciones M:N, ternarias y n-arias***. Se convierten en tabla. Su clave primaria será la concatenación de las claves primarias de las entidades que relacionan.



   b) ***Relación 1:N.*** Se presentan tres posibilidades básicas:

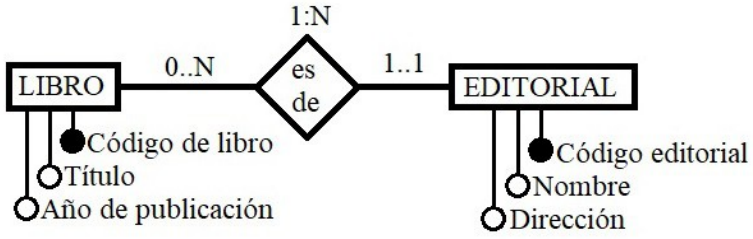
   - Si la relación tiene algún atributo, se convierte en tabla:

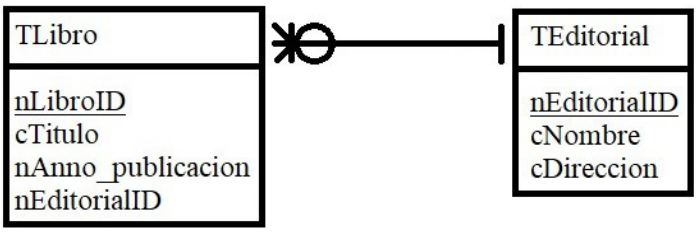


   Es posible que se dé una circunstancia que haya que evaluar en el momento de la transformación: si algún atributo de la relación debe formar parte de la clave primaria o no. En el presente ejemplo, la relación cuenta con dos atributos, "Fecha de inicio","Fecha de fin". Si la especificación de requisitos indica que un cliente puede contratar el mismo servicio más de una vez, la clave primaria de TContrato (formada por la concatenación de las claves primarias de las entidades relacionadas) no garantizaría la unicidad (podría haber dos registros con el mismo nClienteID y nServicioID). En ese caso, se debería añadir la fecha de inicio a la clave (entendiendo que no puede haber un cliente que contrate el mismo servicio dos veces en la mima fecha).

Este tipo de problemática tiene que ver con la dimensión temporal de la información almacenada en una base de datos, y suele requerir intervención especial por parte del analista, que evaluara cuál es la solución más eficiente para cada caso.

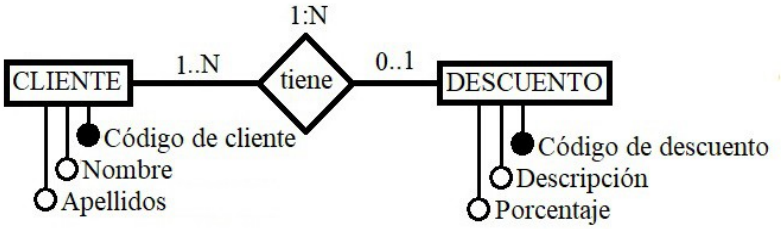
- Si la modalidad mínima en el lado de cardinalidad 1 es 1 y no hay atributos, la relación desaparece. Los atributos que conforman la clave primaria de la entidad de modalidad máxima 1 se propagarán a la entidad de modalidad máxima N:

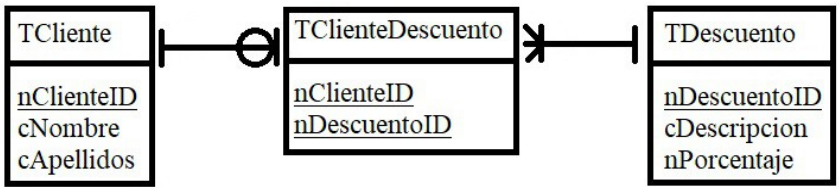




En este caso aparece el concepto de *clave ajena o foránea (foreign key)*. Tenemos un campo o campos que conforman la clave primaria en una tabla y aparecen en otra tabla distinta. En el ejemplo nEditorialID es clave primaria en TEditorial y clave ajena en TLibro.

- Si la modalidad mínima en el lado de cardinalidad 1 es 0..1, la relación se convierte en tabla.

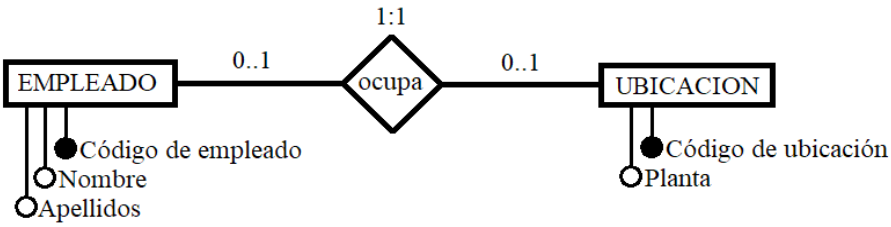


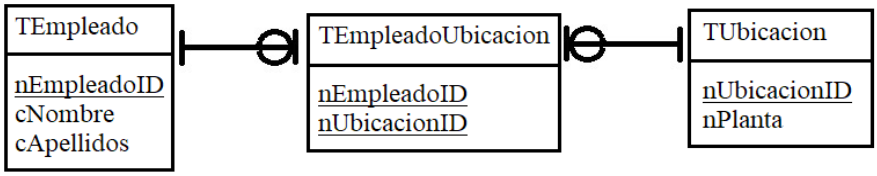


Si, en el presente ejemplo, se hubiese optado por propagar nDescuentoID a TCliente se podrían generar valores nulos en dicho campo, ya que no todos los clientes tienen descuento.

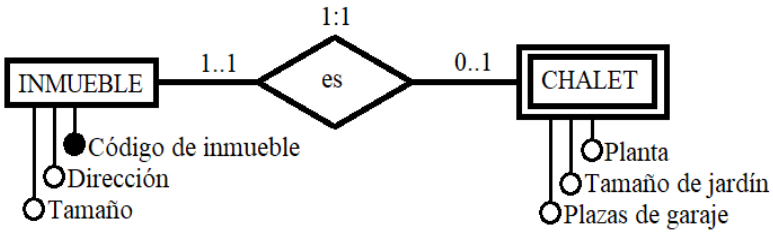
c) ***Relaciones 1:1***. Es la casuística más compleja. El objetivo es evitar a toda costa que queden campos sin valor. Generalmente requiere un estudio detallado de las circunstancias concretas de cada caso, pero se pueden definir unas reglas para ciertas situaciones:

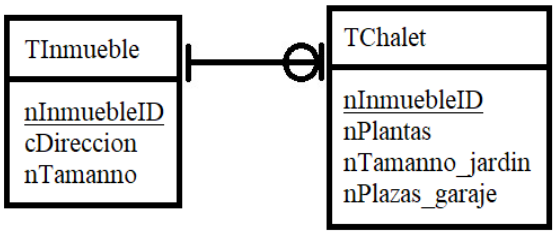
- Si ambas modalidades son 0..1, la relación genera una tabla. En el siguiente ejemplo, donde se relacionan empleados con ubicaciones físicas dentro de una oficina, dándose la posibilidad de que haya empleados sin ubicación (personal de limpieza o mensajería) y ubicaciones vacías (mesas sin ocupar). Propagar cualquiera de las claves primarias a la otra entidad ocasionaría valores ausentes en campos (empleados sin valor en su código de ubicación o ubicaciones sin valor en su código de empleado):





- Si una modalidad es 0..1 y la otra 1..1, la clave de la entidad con modalidad mínima 1 se propaga a la entidad con modalidad mínima 0. De forma más intuitiva, en el siguiente ejemplo se puede ver que todo chalet es un inmueble, pero no todo inmueble es un chalet, por lo que lo lógico es que la clave de "Inmueble" se propague a "Chalet", y no al revés:

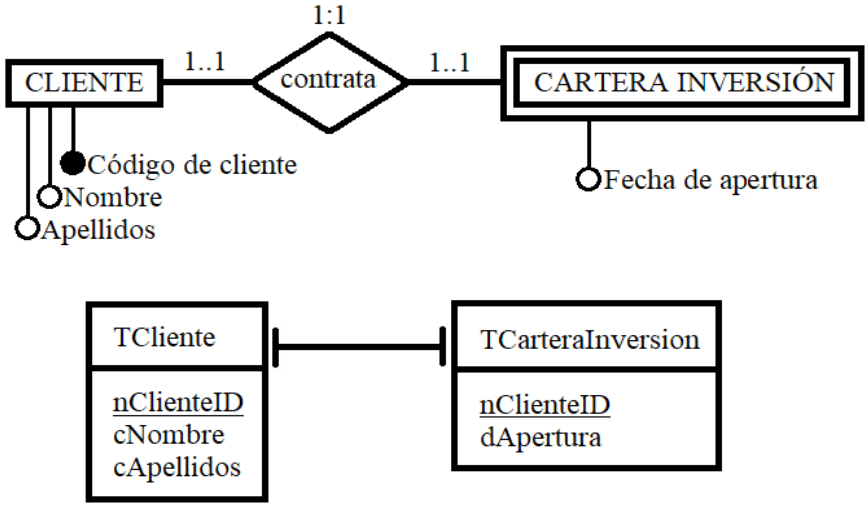




Nótese que en este caso la clave primaria de Tlnmueble se propaga a TChalet como clave ajena, pero también funcionará en TChalet como clave primaria (se garantiza la unicidad), aunque se rellene con valores ya existentes en TIninueble. En este caso, en el que una entidad débil necesita de la clave primaria de la entidad fuerte de la que depende, se dice que hay una relación de *dependencia en identificación.* Cuando la entidad débil puede generar su propia clave primaria se habla de *dependencia en existencia*. Si, en vez de conformar la clave primaria de

TChalet por completo, nInmuebleID fuese parte de una clave compuesta, la relación seguiría siendo de dependencia en identificación.

- Si ambas modalidades son 1..1, pero una de las entidades es débil, la clave de la entidad fuerte se propaga a la débil. Supongamos una relación 1 a 1 entre clientes y carteras de inversión donde la cartera de inversión se crea exclusivamente a partir del cliente:



- Si ambas modalidades son ł..1 y no hay entidades débiles, o bien si existe algún atributo de relación, habrá que estudiar en detalle cual es la opción adecuada en cada caso, dependiendo de la semántica de los datos almacenados y de otros detalles técnicos, como el número estimado de accesos a cada una de las tablas o el orden de consulta de la información.

d) ***Relaciones reflexivas.*** Funcionan igual que las binarias, teniendo en cuenta que las ocurrencias son de la misma entidad que juega papeles distinto

e) ***Modelo entidad relación extendido.*** La transformación de elementos correspondientes a1 diagrama Entidad-Relación extendido suele conllevar pérdida de semántica, y exige en algunos casos la inclusión de mecanismos de control en la base de datos resultante. Por ese motivo, algunos autores no recomiendan su uso, argumentando que la sintaxis del DER original cubre todas las necesidades de un diseño conceptual de datos.

En cualquier caso, si es necesario transformar elementos del DER extendido se pueden aplicar las siguientes reglas dentro del propio diagrama Entidad/Relación:

- Las relaciones exclusivas se tratan como relaciones normales. El control de exclusividad se deberá implementar posteriormente como restricción de la base de datos.
- Las relaciones de jerarquía se convierten en relaciones 1:1 con modalidades 0..1 y 1..1.
- Las relaciones de agregación se tratan corro relaciones individuales entre cada entidad agregada y la entidad principal, en cuyo lado habría que determinar la modalidad de acuerdo a los requisitos especificados en cada caso.

  En este caso se ha asignado modalidad 0..N a las tres relaciones en el lado de "Automóvil", suponiendo que "Motor" almacene modelos de motor, "Chasis"

modelos de chasis y “Rueda” modelos de rueda, de modo que el mismo motor, el mismo chasis y la misma rueda pueden instalarse en varios automóviles. En otras circunstancias la modalidad en el lado de automóvil podría ser 1..N, 1..1 o incluso 0..1.

3.4. Otras consideraciones.

3.4.1. Indices.

El orden de presentación de la información en pantalla o listado no tiene por qué coincidir con su orden de almacenamiento, por lo que el modelo relacional provee herramientas que independizan ambos escenarios. De modo análogo a lo explicado en el tema 1 en relación con los ficheros indexados, las tablas de una base de datos relacional también cuentan con ficheros de índice asociados, fundamentales para buscar información y para acceder a ella de forma ordenada.

Como mínimo habrá un fichero de **índice** que relacione cada registro con su valor correspondiente de **clave primaria**. A partir de ahí, es tarea del diseñador definir los índices que se considere necesarios, si bien hay algunos criterios básicos:

Campos por los que se van a ordenar las consultas más habituales. Supongamos la tabla TSocio de una biblioteca. Con frecuencia los bibliotecarios utilizarán una aplicación que manipulará la información de la base de datos, y contarán con una pantalla donde poder dar de alta socios, modificar sus datos, eliminarlos o simplemente consultarlos. Si la clave primaria de TSocio es cNlF, ordenar a los socios por su NIF no aportará ventajas a los usuarios de la aplicación. Lo normal será que quieran ordenar1os por apellidos y nombre, o bien por fecha de alta. Para ello se deberían Crear dos índices, uno por cApellidos + cNombre, y otro por dAlta. De esa forma, cada vez que un bibliotecario quiera consultar la lista de socios ordenados por apellido y nombre, el SGBD detectará que existe un índice por ambos campos, y leerá dicho índice en vez de recorrer la tabla secuencialmente.

Es necesario señalar la importancia del orden de campos en la definición de un índice (no es lo mismo ordenar por cApellidos + cNombre que por cNombre + cApellidos).

Campos accedidos con mucha frecuencia. El ordenamiento de información que implican los índices no solo sirve para mostrar grupos de registros ordenados. Además permite buscar valores con mayor rapidez. Generalmente el bibliotecario buscará a los socios por sus apellidos, por lo que el índice cApellidos + cNombre también servirá para agilizar sus consultas.

Claves ajenas. Es muy habitual ejecutar consultas que afectan a varias tablas relacionadas. Para localizar los libros publicados por una editorial tendríamos que seguir los siguientes pasos:

1. Partiendo del nombre de la editorial (campo cNombre de TEditorial), ver cual es el código de editorial de ese registro (campo nEditorialID de TEditorial)
2. Buscar ese valor en la tabla libros (TLibro).
3. Obtener el valor correspondiente de cTitulo en TLibro.

Para una consulta como esta nos vendría bien que TLibro contase con un índice por nEditorialID. De lo contrario, la búsqueda tendrá que realizarse directamente sobre la tabla, cuyos datos pueden estar desordenados, o bien ordenados de acuerdo a un criterio que no satisfaga nuestra búsqueda.

A pesar de todas sus ventajas, la existencia de índices crea redundancia y ralentiza los procesos de inserción, actualización y borrado, al implicar la reordenación de los índices asociados. Por ese motivo es importante no abusar de su número ni de su tamaño (un índice compuesto por todos los

campo de una tabla ocuparía más espacio que la propia tabla, y ralentizaría sustancialmente su acceso).

3.4.2. Vistas.

Una vista es una consulta sobre una o varias tablas almacenada en la base de datos. De cara al usuario la información obtenida mediante una vista parece una tabla más, pero el uso de vistas incrementa el nivel de seguridad en el acceso a los datos y son más eficientes que las consultas efectuadas puntualmente, al encontrarse ya definidas en la base de datos.

No obstante las vistas no permiten recibir parámetros, es decir, no se pueden ejecutar para distintos valores de campos concretos de la base de datos. Una vista puede obtener una lista de clientes, o una relación de facturas cuyo importe esté dentro de un rango, pero no puede filtrar sus valores por un dato variable. Esa rigidez provoca que muchos desarrolladores ignoren su existencia, especialmente si ya se han establecido otros mecanismos de seguridad.

3.4.3. Restricciones sobre campos.

Además de dotar de un tipo de datos a cada campo, se le pueden asignar las siguientes restricciones:

UNIQUE. Todos los valores de ese campo deben ser únicos, y no se permiten valores repetidos. Cuando una clave primaria está compuesta por un solo campo, dicho campo está obligado a cumplir la restricción UNIQUE.

NOT NULL. Impide que un campo pueda tener valores nulos. Obligado en los campos que formen la clave primaria.

DEFAULT. Si no se especifica un valor se asignará el valor por defecto, que no será NULL.

3.4.4. Usuarios y privilegios.

Se pueden establecer restricciones de acceso a los datos según los diferentes perfiles de los usuarios, basado en un sistema de permisos. Podemos tener las siguientes opciones:

- Asignar un usuario de la base de datos a cada usuario físico del SGBD y posteriormente agruparlos en perfiles. A cada perfil se le asociarán una serie de permisos.
- Otra posibilidad es crear como usuarios del SGBD los perfiles genéricos solamente y que cada usuario físico entre con su perfil correspondiente.
- Establecer para cada perfil o usuario que tipo de acceso tiene a los objetos de la base de datos: bases de datos, tablas, vistas, tareas de administración, gestión de usuarios,.. Los accesos pueden ser para lectura, consulta o modificación de bases de datos completas o por tablas, incluso dentro de las tablas. Ej: acceso a los datos de los empleados, pero no a sus sueldos.

3.4.5. Acceso concurrente.

En la operativa de la mayoría de los sistemas de información ocurre que más de un empleado trabaja con los mismos datos. Esta situación ha de estar controlada para evitar problemas.

Supongamos que en una librería hay un empleado recepcionando un pedido y dando de alta los libros recibidos. En este caso han llegado 10 ejemplares de “Lejos de Luisiana”. En el stock le aparecen 7. Antes de actualizar decide comprobar el albarán.

En ese momento llega un cliente y es atendido por otro empleado y compara un ejemplar de “Lejos de Luisiana”, dejando en el stock 6.

Una vez hecha la comprobación del albarán y todo está correcto, se actualiza: en su pantalla tenía 7 más los 10 que entran, total 17.

En el ordenador quedan registrados 17 ejemplares cuando en realidad en la tienda solo hay 16.

3.4.6. Políticas de bloqueo.

Para evitar los problemas descritos en el apartado de concurrencia se recurre al bloqueo. Un bloqueo no es más que cuando un usuario accede a unos datos se bloquea el acceso de otros hasta que el primero no termine y libere la información. Esto puede generar problemas de lentitud del sistema incluso bloqueos. Por lo que la estrategia de bloqueos es muy importante hacer un buen diseño de ellas.

Básicamente hay dos tipos de bloqueos:

- Compartido. Se permite que otros usuarios puedan acceder a los datos bloqueados, pero no modificarlos.
- Exclusivo. Solo el usuario que ha bloqueado la información puede consultarla y modificarla.

Cada Sistema Gestor de Base de Datos tienen sus propios mecanismos para atender los bloqueos. Los principales motores de MySQL atienden los bloqueos de la siguiente forma:

- MyISAM: Bloqueos a nivel de tabla.
- InnoDB: Bloqueos a nivel de registro.

3.5. Ejercicios.

1. Pasar a 1FN la siguiente relación de músicos:

DNI	Nombre	Instrumento
12125252-H	Julia	Guitarra
85411220-F	Marco	Clarinete Trompeta

2. Pasar a 2FN la siguiente relación con información de alumnos/as:

DNI	Cod Curso	Nombre	Apellido 1	Calificación
12000333-K	27	Carlos	Mancera	7
12000333-K	29	Carlos	Mancera	9
25666000-P	27	Lorenzo	Domínguez	8
55666777-U	29	Inés	Acedo	7
55666777-U	27	Inés	Acedo	9

3. Pasar a 3FN la siguiente relación de alumnos/as:

DNI	Nombre	Apellido 1	Cod Provincia	Provincia
120000333-K	Carlos	Mancera	29	Málaga
25666000-P	Lorenzo	Domínguez	29	Málaga
55666777-U	Inés	Acedo	11	Cádiz
22000121-R	Lucía	Martínez	11	Cádiz
10111002-X	Carmen	De los Ríos	14	Córdoba

4. Pasar a 3FN la siguiente relación que recoge los pedidos de una empresa:

Cod Pedido	Fecha Pedido	Cod Cliente	Nombre Cliente	Ciudad Cliente	Tfno Cliente	Cod Articulo	Descrip Articulo	Precio unidad	Cod Prov	Tfno Prov	Cant pedida
1001	24/10/23	18	María Pacheco	Málaga	121212	110	Ciruela	1,99	C-231	123123	1,5
						201	Piña	1,15	M-001	290290	2
1002	24/10/23	25	Alberto Gómez	Granada	313131	523	Plátano	2,15	M-001	290290	2
						201	Piña	1,15	M-001	290290	1,75
						101	Mango	3,40	A-501	187187	1,8

5. Dada la siguiente relación que recoge los turnos de los empleados de una cadena de tiendas:

DNI	Nombre	Cod tienda	Dirección tienda	Turno	Fecha
27111200-L	Paco López	15A	Martinez Maldonado	M	21/10/23
18000222-X	Rosa Pérez	15A	Martinez Maldonado	M	21/10/23
25200200-J	Rocío Morón	15A	Martinez Maldonado	T	21/10/23
27111200-L	Paco López	20B	Blas de Lezo	T	22/10/23
11285777-P	Antonio Sanz	30C	Alameda principal	M	22/10/23
99121212-Q	Luisa Valero	20B	Blas de Lezo	M	22/10/23
11285777-P	Antonio Sanz	30C	Alameda principal	M	23/10/23
99121212-Q	Luisa Valero	15A	Martinez Maldonado	M	23/10/23
18000222-X	Rosa Pérez	15A	Martinez Maldonado	T	23/10/23
27111200-L	Paco López	20B	Blas de Lezo	M	24/10/23

Se pide:

a) Indicar las dependencias funcionales utilizando las siguientes abreviaturas: DNI (P), Nombre (N), Cod tienda (C), Dirección tienda (D), Turno (T) y Fecha (F).

b) ¿En qué forma normal se encuentra la relación? ¿Cual sería su clave principal?

c) Si no estuviese en 3FN, realiza los pasos necesarios para normalizar hasta la 3FN.

6. Normaliza la siguiente relación en la que se recogen las calificaciones de los alumnos de 4ESO de un centro en el que solo hay una línea, lo que significa cada asignatura solo la imparte un profesor.

DNI	Nombre	Apellido	Cod Postal	Provincia	Asignatura	Profesor	Nota
26003251-M	Jorge	Trujillo	18019	Granada	Lengua	Antonio	5
26003251-M	Jorge	Trujillo	18019	Granada	Biología	Graciella	7

20666115-Z	Aurora	Baeza	14001	Córdoba	Inglés	Candela	9
20666115-Z	Aurora	Baeza	14001	Córdoba	Lengua	Antonio	7
19404109-W	Manuel	Burgos	29105	Málaga	TIC	Eva	8
19404109-W	Manuel	Burgos	29105	Málaga	Inglés	Candela	5

7. Ejercicio: Se desea almacenar información sobre citas médicas, sabiendo que cada doctor puede ver a varios pacientes y cada paciente puede pedir cita con doctores distintos. Organizar los siguientes atributos en relaciones normalizadas hasta la 3FN:
    - NIF de doctor
    - Nombre de doctor
    - Dirección de doctor
    - NIF de paciente
    - Nombre de paciente
    - Teléfono de paciente
    - Fecha de la cita
    - Código de tratamiento
    - Descripción del tratamiento
    - Coste

8. Se requiere normalizar un proceso de facturación, en el que disponemos de los siguientes atributos:
    - Nº Factura
    - Fecha de la factura
    - Id Cliente
    - Nombre Cliente
    - Localidad del cliente
    - Código de cada artículo
    - Descripción de artículo
    - Cantidad de ese artículo
    - Precio unidad de ese artículo
    - Total de la factura

9. Se requiere normalizar el Sistema de Información de un centro educativo especializado en Formación Profesional. Teniendo en cuenta los siguientes aspectos:
    a) Los alumnos se pueden matricular de distintas especialidades (DAW, ASIR,..) y cada una de ellas tienen un conjunto de módulos.
    b) Puede haber módulos que aparecen en varias especialidades (FOL, Lenguaje de marcas,..).
    c) Un alumno se puede matricular un curso en una especialidad, aprobar varios módulos y al curso siguiente intentar otra especialidad. Pero si aprueba un módulo no se puede

volver a matricular de él, aunque haga otra especialidad, pues ya lo tiene superado. Por su puesto puede repetir algún módulo.

d) Cada profesor depende de un departamento y cada módulo también.

e) No se consideran desdobles. Pero si puede impartir el mismo módulo dos profesores distintos el mismo curso escolar a distintos grupos de alumnos.

f) Los profesores pueden cambiar de módulo de un curso a otro.

Los atributos a considerar son:

- Id-alumno
- Nombre alumno
- Código especialidad
- Especialidad
- Id-asignatura
- Nombre asignatura
- Id-profesor
- Nombre profesor
- Id-Dpto
- Nombre Dpto
- Curso académico
- Nota

10. Normalizar el proceso de gestión de presupuestos de un taller de mecánica.

- Num. Presupuesto
- Fecha
- Total
- Cod-cliente
- Nombre-cliente
- Dir-cliente
- Tfno-cliente
- Cod-mecánico
- Nombre-mecánico
- Matrícula
- Marca
- Modelo
- Kms
- Cod-artículo
- Descripción

- Precio-unidad
- Cantidad

Glosario.

Atributo: Cada columna de una relación . Corresponde a cada dato de una tupla.

Clave ajena. En una base de datos relacional, presencia en una tabla de un campo que ejerce de clave primaria en otra tabla distinta.

Dominio de un atributo. al conjunto de valores que puede tomar para una ocurrencia concreta.

Null. Representa ausencia de valor. No se debe confundir con el valor vacío (“” en un campo de tipo cadena de caracteres, o 0 en un campo numérico).

Normalización. Técnica para eliminar redundancias en una base de datos relacional. Perfil. Patrón de características comunes a varios usuarios de una base de datos.

Tabla. Estructura bidimensional de almacenamiento de información presente en una base de datos relacional.

Tipo de dato. Definición de los posibles valores que puede tomar un atributo de una tabla relacional y las operaciones que se pueden hacer con ellos.

Tupla. Cada fila de una relación. Corresponde a todos los atributos de una determinada ocurrencia.

Valor: Intersección entre tupla y atributo. El contenido de una determinada ocurrencia para un determinado atributo.

Variable. Datos almacenados en memoria temporal cuyo contenido puede cambiar a lo largo de la ejecución de un programa.

Integridad referencial. Toda clave ajena ha de tener un registro en la tabla a la que apunta.

Concurrencia. Situación en la que dos usuarios o más trabajan a la vez con al misma base de datos.

Transacción. Conjunto de operaciones que ha de tratarse como una única operación, es decir, o se realizan todas o no se realiza ninguna. Para ello hay dos comandos claves: rollback y commit. Si se ejecutan todas las operaciones y se dan por válidas se ejecuta commit.

Si en algún momento se detecta algún problema se ejecuta rollback y se deshacen las operaciones hechas de la transacción.