UNIDAD 3 PARTE I EL MODELO RELACIONAL

1. INTRODUCCIÓN

El modelo entidad-relación es un modelo conceptual que sirve para cualquier tipo de SGBD, en cambio, el modelo relacional es un modelo lógico que sólo sirve para SGBD relacionales (y no para jerárquicos, o Codasyl, por ejemplo).

Todos los diseñadores y administradores de bases de datos relacionales usan esquemas conceptuales entidad-relación porque se adaptan muy bien a este modelo.

Hay que tener en cuenta la diferencia de la palabra **relación** en ambos modelos. En el modelo relacional una relación es una tabla mientras que en el entidad/relación es la asociación que se produce entre dos entidades.

El modelo relacional fue propuesto en 1970, como alternativa a los modelos existentes, por el investigador **Edgar Codd.** El modelo relacional es un modelo de datos basado en dos disciplinas matemáticas: la lógica de predicados y la teoría de conjuntos. Este modelo perseguía la sencillez de comprensión y de manipulación de la base de datos por parte del usuario final.

En un principio el modelo era simplemente un modelo teórico objeto de investigación por parte de diversos centros, pero a medida que la tecnología fue evolucionando fueron surgiendo los primeros SGBD que lo soportaban y hoy en día el modelo relacional es el más conocido y difundido por los sistemas comerciales (ORACLE, INFORMIX,...).

Codd perseguía los siguientes **objetivos**:

- **Independencia física**. La forma de almacenar los datos, no debe influir en su manipulación lógica. Si el almacenamiento físico cambia, los usuarios no tienen ni siquiera porque enterarse, seguirán funcionando sus aplicaciones.
- **Independencia lógica**. Las aplicaciones que utilizan la base de datos no deben ser modificadas por que se modifiquen elementos de la base de datos. Es decir, añadir, borrar y suprimir datos, no influye en las vistas de los usuarios.
- **Flexibilidad**. La base de datos ofrece fácilmente distintas vistas en función de los usuarios y aplicaciones.
- **Uniformidad**. Las estructuras lógicas siempre tienen una única forma conceptual (las tablas)
- **Sencillez**. Facilidad de manejo.

Es notoria la importancia que Codd concede al tema de la **independencia de la representación lógica de los datos respecto a su almacenamiento interno** (independencia de **ordenación**, independencia de **indexación** e independencia de la **forma de acceso**)

El modelo relacional representa la segunda generación de los SGBD. En él, todos los datos están estructurados a nivel lógico como tablas formadas por filas y columnas, aunque a nivel físico puede tener una estructura completamente distinta. Un punto importante del modelo relacional es la sencillez de su estructura lógica. Pero, detrás de esa simple estructura hay un fundamento teórico matemático importante que hace que el modelo sea seguro y robusto.

2. ESTRUCTURA DEL MODELO RELACIONAL

2.1. Estructura

El **modelo relacional** permite construir estructuras de datos para representar las diferentes informaciones del mundo real que tengan algún interés. Las estructuras de datos construidas siguiendo el modelo relacional están formadas por conjuntos de relaciones.

Las **relaciones** pueden ser concebidas como representaciones tabulares de los datos.

La estructura del modelo relacional está formada por tres partes claramente diferenciadas:

- La **parte estructural**: Utiliza una estructura de datos muy sencilla, la relación.
- La **parte manipulativa**: Compuesta por **un conjunto de operadores** que permiten manejar la estructura anterior, el **Álgebra Relacional**.
- La **Teoría de las Dependencias Funcionales y Normalización**: Compuesta por un conjunto de definiciones que permiten estudiar las dependencias entre los atributos de las relaciones y proporciona métodos para un correcto diseño de las bases de datos relacionales. Se estudiará en la Unidad 4.

2.2. Relación

La relación es el elemento básico del modelo relacional y está compuesta por dos partes:

1) **Cabecera** (esquema o intensión). Está formada por un conjunto fijo de atributos. Es la parte que no varía de la relación y se le denomina también **esquema de la relación**. Está constituida por:
   - **a)** El **nombre** del conjunto (tabla)
   - **b)** El nombre de los **atributos** (columnas de la tabla)
   - **c)** Los **dominios** de los que toman valores.
2) **Cuerpo** (extensión). Está formado por un conjunto de tuplas. Como consecuencia de su parecido con un elemento matemático, podemos nombrar una relación con el nombre de tabla, la cual está compuesta por filas y columnas, donde cada fila (tupla) representa un conjunto de valores relacionados entre sí (hechos del mundo real), y las columnas (atributos) tienen la función de ayudar a interpretar el significado de los valores que están en cada fila de la tabla.

En el ámbito de las BD, podemos definir **tupla** como una secuencia finita de objetos que comprende las distintas asociaciones entre cada atributo de la relación y un valor concreto, admisible dentro del dominio respectivo.

El **esquema de una relación** consiste en un nombre que la identifica de manera única , y el conjunto de atributos que la relación contiene. Un **esquema de relación** es un conjunto de pares de la forma:

RELACIÓN $\{(A_1, D_1), (A_2, D_2), \ldots, (A_n, D_n)\}$

donde

- $\{A_1, A_2, \ldots, A_n\}$ es el conjunto de nombres de atributos (campos/columnas) del esquema, necesariamente distintos.
- $\{D_1, D_2, \ldots, D_n\}$ son los dominios asociados (tipos de datos) a dichos atributos, que no tienen que se necesariamente distintos.

Veamos un ejemplo:

Supongamos que queremos modelar una base de datos para mantener información acerca de los museos de la Comunidad de Madrid y las exposiciones que en ellos se pueden visitar, a través del modelo de datos relacional. De cada museo se desea almacenar su nombre, la dirección, la zona (Madrid, El Escorial, etc) y el número de salas que contiene.

MUSEO {(nombre, cadena(20)), (dirección, cadena(50)), (zona, cadena(30)), (salas, entero)}

donde:

- {nombre, dirección, zona, salas} es el conjunto de nombres de atributos del esquema.
- cadena, entero son los dominios asociados a dichos atributos.

La extensión de una relación consiste en los valores de los datos almacenados en las tuplas que ésta contiene. Una **extensión de la relación**, R, de un esquema de relación:

$$\{(A_1, D_1), (A_2, D_2), \ldots, (A_n, D_n)\}$$

es un conjunto de tuplas (registros/filas) de la forma:

$$t = \{(A_1, v_1), (A_2, v_2), \ldots, (A_n, v_n)\}$$

tales que $\forall i, v_i \in D_i$.

La notación que usamos para declarar un objeto relación R de esquema $\{(A_1, D_1), (A_2, D_2), \ldots, (A_n, D_n)\}$ es:

$$R\ (A_1{:}D_1, A_2{:}D_2, \ldots, A_n{:}D_n)$$

es un conjunto de pares atributo:dominio

En el ejemplo anterior:

MUSEO (nombre: dom_nombre, dirección: dom_dirección, zona: dom_zona,
salas: dom_salas)

donde un ejemplo de tupla sería:

t_1 = { (nombre, Arqueológico), (dirección, Serrano 13), (zona, Madrid), (salas, 43)

El modelo relacional presenta además las siguientes características:

- Toda relación debe tener un nombre que la identifique unívocamente dentro de la BD.
- Cada fila está constituida por una tupla de datos relacionados entre ellos, que también podemos llamar *registro*, esta tupla guarda los datos que nos interesa reflejar de un objeto concreto del mundo real.
- No existen tuplas repetidas.
- El orden de las tuplas no importa.
- El orden de los atributos no importa.
- Cada atributo solo puede tomar un valor en cada tupla. Los valores de los atributos son atómicos.
- Cada columna contiene, en cada celda, datos de un mismo tipo, podemos llamarla *atributo* o *campo*.

Esta tabla muestra una comparación de la terminología de una relación, una tabla y un fichero:

RELACIÓN	**TABLA**	**FICHERO**
Tupla	Fila	Registro
Atributo	Columna	Campo
Grado	Nº de columnas	Nº de campos
Cardinalidad	Nº de filas	Nº de registros
Modelo Relacional (teoría)	**SGBD Relacionales (implementación)**	**Sistemas de Ficheros**

Intuitivamente una base de datos no es más que un conjunto de tablas entrelazadas entre sí a través de los campos con información común.

En un SGBD relacional pueden existir varios tipos de relaciones, aunque no todos manejan todos los tipos:

- **Relaciones Base**: son relaciones reales que tienen nombre y forman parte directa de las base de datos almacenada.
- **Vistas:** también denominadas relaciones virtuales, son relaciones con nombre y relaciones derivadas, Se representan mediante su definición en términos de otras relaciones con nombre y nos poseen datos propios almacenados.
- **Relaciones Instantáneas:** son relaciones con nombre y derivadas, pero a diferencia de las vistas, son reales, no virtuales. Están representadas no sólo por su definición en términos de otras relaciones con nombre, sino también por sus propios datos almacenados. Son relaciones sólo de lectura y se refrescan periódicamente.
- **Resultados de consultas:** son las relaciones resultantes de alguna consulta especificada. Puede o no tener nombre y no persisten en la Base de datos.
- **Resultados Intermedios:** son las relaciones que contiene los resultados de las subconsultas. Normalmente no tiene nombre y tampoco persisten en la base de datos.
- **Resultados temporales:** son relaciones con nombre, similares a las relaciones base o las instantáneas, peo la diferencia es que se destruyen automáticamente en algún momento.

2.3. Propiedades de una relación

Dentro de una relación podemos distinguir además dos nociones: el grado y la cardinalidad.

- El **grado** es el número de atributos (columnas) que posee la relación.
- La **cardinalidad** se refiere al número de tuplas (filas) de la relación.

Además decimos que dos relaciones son compatibles, si sus esquemas son idénticos.

Ejemplo:

Del esquema de relación:

MUSEO {(nombre, cadena(20)), (dirección, cadena(50)), (zona, cadena(30)), (salas, entero)}

cuya relación es:

MUSEO (nombre: dom_nombre, dirección: dom_dirección, zona: dom_zona, salas: dom_salas)

tenemos las siguientes tuplas:

t_1 = { (nombre, Arqueológico), (dirección, Serrano 13), (zona, Madrid), (salas, 43)}

t_2 = { (nombre, Centro de Arte Reina Sofía), (dirección, Santa Isabel 52)
(zona, Madrid), (salas, 21)}

t_3 = {(nombre, Universidad de Alcalá de Henares), (dirección, Plaza de San Diego), (zona, Alcalá de Henares), (salas, 7)}

El grado es 4

La cardinalidad es 3

Luego el conjunto de tuplas de la relación MUSEO permite representar el conjunto de ocurrencias del diagrama:

MUSEO = { (nombre, Arqueológico), (dirección, Serrano 13), (zona, Madrid), (salas, 43)},

{(nombre, Centro de Arte Reina Sofía), (dirección, Santa Isabel 52) (zona, Madrid), (salas, 21)},

{(nombre, Universidad de Alcalá de Henares), (dirección, Plaza de San Diego), (zona, Alcalá de Henares), (salas, 7)}

Una representación gráfica más cómoda de esta relación es mediante una tabla, en la que cada fila representa una tupla de la relación y cada columna está etiquetada con el nombre de un atributo.

La representación de MUSEO mediante una tabla es:

MUSEO

Nombre	Dirección	Zona	Salas
Arqueológico	Serrano, 13	Madrid	43
Centro de Arte Reina Sofía	Santa Isabel, 52	Madrid	21
Universidad de Alcalá de Henares	Plaza de San diego, s/n	Alcalá deHenares	7

No hay que confundir relación con tabla, existen diferencias entre ambos conceptos:

TABLA	RELACIÓN
Existe un orden entre sus filas.	No existe un orden entre las tuplas, ya que una relación es un conjunto.
Existe un orden entre los elementos de de las filas (las columnas)	Los pares de una tupla no están ordenados entre sí, ya que la tupla también es un conjunto.
	No existen tuplas repetidas en una relación.
	Es plana, es decir, que en el cruce de una fila y de una columna sólo puede haber un valor (no se admiten atributos multivaluados)

2.4. Dominio

Un dominio D es un conjunto finito de valores homogéneos y atómicos V_1, V_2, ..., V_n, caracterizado por un nombre; decimos valores homogéneos porque son todos del mismo tipo, y atómicos porque son indivisibles en lo que al modelo se refiere, es decir, si se descompusiesen perderían la semántica a ellos asociada.

Los atributos sólo pueden tomar los valores que estén incluidos dentro del dominio respectivo.

Todo dominio ha de tener un nombre, por el cual nos podemos referir a él, y un tipo de datos, pudiéndole asociar unidades de medida y ciertas restricciones.

Un ejemplo: Tenemos el atributo teléfono de la relación ALUMNO. Si definimos este atributo de tal manera que sólo pueda almacenar nueve caracteres de tipo numérico, el dominio de este atributo serían todas las posibles combinaciones posibles de los dígitos de 0 a 9.

Como se puede observar el dominio se asemeja a un tipo de datos, pero tiene muchas más connotaciones que este término, como que incluye el valor NULO.

El valor NULO en el Modelo Relacional es ausencia de información.

2.5. Operadores de tupla

Dada una tupla t = {(A_1, v_1), (A_2, v_2), …, (A_n, v_n)} los operadores que permiten crear y consultar tuplas son los siguientes:

CONSULTAR

Permite consultar el valor de un atributo en una tupla.

Consultar (t, A_i) = v_i

ASIGNAR

Permite asignar un valor a un atributo en una tupla.

Asignar (t, A_i, w_i) = {(A_1, v_1), …, (A_i, w_i), …, (A_n, v_n)}

Generalmente, para los operadores de la tupla se utiliza una sintaxis más simplificada, que evita el uso de los nombres de operación.

	Consultar (t, A_1)	**Asignar (t, A_i, w_i)**
Notación de punto	t.A_i	t.A_i ← w_i
Notación funcional	t (A_i)	t (A_i) ← w_i

Ejemplo:

De la relación

MUSEO {(nombre: dom_nombre)), (dirección:dom_dirección), (zona:dom_zona), (salas, dom_salas)}

donde

dom_nombre: cadena(30)

dom_dirección:cadena(30)

dom_zona:cadena(40)

dom_salas:entero

Tenemos las siguientes tuplas:

t_1 = { (nombre, Arqueológico), (dirección, Serrano 13), (zona, Madrid), (salas, 43)}

t_2 = { (nombre, Centro de Arte Reina Sofía), (dirección, Santa Isabel 52)

(zona, Madrid), (salas, 21)}

t_3 = { (nombre, Ciencias), (dirección, Serrano 15), (zona, Madrid), (salas, NULL)}

Podríamos realizar las siguientes operaciones:

Consultar (t_1, nombre) = “Arqueológico”

Asignar (t_1, salas, 50) = {(nombre, Arqueológico), (dirección, Serrano 13), (zona, Madrid), (salas, 50)}}

Consultar (t_3, salas) = NULL

Diremos que t_3.salas es nulo (IS NULL), no que t_3.salas = NULL.

2.6. Operadores asociados a la estructura relación

Una relación es un conjunto de tuplas. Veamos los operadores asociados a la estructura relación:

a) **INSERCIÓN**: añade una tupla a la relación

b) **BORRADO**: borra una tupla de la relación

c) **SELECCIÓN**: la relación resultante de aplicar una condición de selección a una relación, contiene la tuplas de ésta que cumplen una condición especificada.

d) **PROYECCIÓN**: extrae de la relación los valores de los atributos especificados en la operación.

e) **UNIÓN**: la relación resultante de la unión de dos relaciones contiene las tuplas que aparecen en cualquiera de ellas.

f) **DIFERENCIA**: la relación resultante de la diferencia de dos relaciones contiene las tuplas que aparecen en la primera y no aparecen en la segunda.

g) **PRODUCTO CARTESIANO**: la relación resultante del producto cartesiano de dos relaciones contiene las tuplas que se pueden construir combinando ("uniendo") una tupla de la primera relación con una tupla de la segunda relación.

3. RESTRICCIONES DE INTEGRIDAD

Recordar que teníamos dos tipos de restricciones de integridad:

- Inherentes: vienen impuestas por el propio modelo.
- Semánticas: definidas por el usuario.

En el modelo relacional se contemplan las siguientes restricciones:

Restricciones Inherentes. Las más importantes son:

- No puede haber dos tuplas iguales
- El orden de las tuplas no es significativo
- El orden de los atributos no es significativo
- Cada atributo sólo puede tomar un valor en el dominio en el que está inscrito, es decir, los atributos son atómicos y homogéneos.

Además tenemos que añadir dos reglas que completan el conjunto de restricciones inherentes:

- Regla de integridad de entidad
- Regla de integridad referencial

Restricciones semánticas: Son restricciones que dependen de la semántica del problema y sirven para reflejar de la forma más fiel posible el mundo real que se modela. Son restricciones personales a los datos, son restricciones que puede imponer el usuario.

- Restricción de valor no nulo (NOT NULL)
- Restricción de unicidad (UNIQUE)
- Restricción de clave primaria (PRIMARY KEY)
- Validación (Check)
- Disparador

Veamos un poco con más detalle algunas de ellas. Aunque antes vamos a ver el concepto de Claves primarias y claves alternativas.

3.1. Clave Primarias y Claves Alternativas. Restricción de clave primaria

Clave Candidata es un conjunto de atributos que identifican unívoca y mínimamente cada tupla de la relación. De la propia definición de relación se deriva que siempre existe, al menos, una clave candidata (*al ser una relación un conjunto y no existir dos tuplas iguales, el conjunto de todos los atributos siempre tiene que identificar unívocamente a cada tupla*). La propiedad de minimalidad implica que no se incluye ningún atributo innecesario

Una relación puede tener más de una clave candidata. En este caso se debe distinguir entre:

Clave Primaria (*Primary Key*):

- Es la clave candidata que el usuario escoge para identificar las tuplas de la relación.
- Cuando sólo existe una clave candidata, ésta es la clave primaria (siempre existe clave primaria).

Claves Alternativas (*Alternative Key*):

Las claves candidatas que no han sido escogidas como clave primaria.

Así, la **restricción de integridad de clave primaria** establece que:

- En los atributos marcados como clave primaria no puedan repetir valores. Restricción de unicidad
- Además obliga a que esos atributos no puedan estar vacíos (nulos); es más si la clave primaria la forman varios atributos, ninguno de ellos podrá estar vacío. Restricción de Valor No Nulo (VNN)

3.2. Integridad de entidad

La regla de integridad de entidad es el mecanismo que garantiza la identificación y unicidad de las tuplas en una relación.

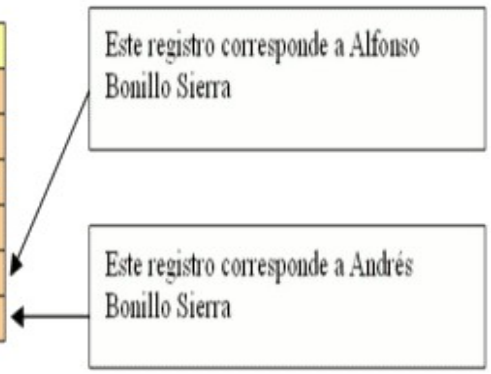
¿Qué dice esta regla? **Ningún atributo que forme parte de la clave primaria de una relación puede tomar un valor nulo.**

De esta manera nos aseguramos que no se pueden repetir dos tuplas en una relación. Veamos que esto es así tanto si la clave primaria es un único atributo, o si está compuesta por más de un atributo.

- La clave primaria es un único atributo: Consideramos la siguiente relación ALUMNO:

ALUMNO

DNI	NOMBRE	DIRECCION	CURSO
23456987	S. López Ojeda	Islas Virgenes, 19	Primero
22335566	S. Thomas Barrós	Islas del Caribe, 2	Primero
34562358	D. Rodriguez Gracia	Menorca, 15	Segundo
23569856	N. Jáimez Toro	Isla Cristina, 12	Tercero
Null	A. Bonillo Sierra	Rejoneador, 16	Tercero
Null	A. Bonillo Sierra	Rejoneador, 16	Tercero



En esta relación la clave primaria es el atributo DNI, si dicho atributo fuese null y quisiéramos introducir una tupla nueva con los datos de Andrés Bonillo Sierra, hermano de Alfonso Bonillo Sierra, y considerando que los nombres se introducen con el formato: Inicial del nombre y dos apellidos... ¿cómo podríamos distinguir a los dos hermanos Bonillo Sierra? **Sólo si se cumple la regla de integridad de la entidad: la clave primaria no puede ser null.**

- La clave primaria está formada por un conjunto de atributos: Consideramos la relación PEDIDO

PEDIDO			
DNI_CLIENTE	COD_TIENDA	FECHA_PEDIDO	CUANTIA
23456987	005	19/04/05	1904€
22335566	007	08/02/06	806€
34562358	019	21/05/06	2525€
23569856	012	20/06/05	235€
23456987	005	19/04/05	1904€

Es evidente que para que los registros de esta relación sean únicos, necesitamos que la clave principal esté formada por los atributos DNI_CLIENTE, COD_TIENDA y FECHA_PEDIDO.

3.3. Sobre atributos: dominio y valor no nulo

Restricción sobre dominio: El asociar un dominio a cada atributo restringe el conjunto de valores que puede tomar ese atributo.

Restricción de valor no nulo u obligatoriedad: La definición de una restricción de valor no nulo sobre un conjunto de atributos K de la relación R expresa la siguiente propiedad: “no debe haber en R una tupla que tenga el valor nulo en algún atributo de K”. Es decir, no se admiten valores nulos.

3.4. Restricción de unicidad

Impide que los valores de los atributos marcados de esa forma, puedan repetirse. Esta restricción debe indicarse en todas las claves alternativas o candidatas.

3.5. Validación (Check)

Comprueba en toda operación de actualización de la tabla, si el valor a introducir es correcto y en caso de que no lo sea, rechaza la operación. Por ejemplo restringir el campo sueldo para que siempre sea mayor de 1000, sería una regla de validación. También por ejemplo que la fecha de inicio sea mayor

que la fecha final.

3.6. Disparador

Restricción en la que el usuario puede especificar libremente la respuesta del SGBD ante una determinada condición. Se trata de pequeños programas grabados en la base de datos que se ejecutan automáticamente cuando se cumple una determinada condición. Sirven para realizar una serie de acciones cuando ocurre un determinado evento (cuando se añade una tupla, cuando se borra un dato, cuando un usuario abre una conexión, etc.). Los triggers permiten realizar restricciones muy potentes; pero son las más difíciles de crear.

3.7. Integridad referencial: Clave Ajena

Se denomina **clave ajena** de una relación R2 a un conjunto no vacío de atributos cuyos valores han de coincidir con los valores de la clave candidata de una relación R1. Una clave ajena está constituida por un atributo, o por un conjunto de atributos, del esquema de una relación, que sirve para relacionar sus tuplas con los tuplas de otra relación de la base de datos (o con las tuplas de ella misma, en algunos casos)

- R1 y R2 pueden ser la misma relación.
- La clave ajena y la correspondiente clave candidata han de estar definidas sobre el mismo dominio.

El uso de claves ajenas es el mecanismo que proporciona el modelo relacional para expresar asociaciones entre los objetos representados en la base de datos.

Se añade en una relación R un conjunto de atributos que hagan referencia a un conjunto de atributos de una relación S.

A este conjunto de atributos se les denomina clave ajena de la relación R que hace referencia a la relación S.

Si a esta clave ajena se le marca con integridad referencial, no se podrán introducir valores que no estén incluidos en los campos relacionados con esa clave.

Ejemplo: Si tenemos una tabla alquileres en la que cada fila es un alquiler, existirá un atributo ***cod_cliente*** que indicará el código del cliente y que estará relacionado con una tabla de clientes, en la que dicho atributo es la clave principal.

- ¿Qué ocurriría si introdujésemos en un alquilero el código de un cliente que no existe en la base de datos?
- ¿Qué ocurrirá con los alquileres que contienen un código de cliente cuando ese cliente fue dado de baja en la base de datos porque ya no nos interesa o se ha muerto?
- ¿Deberíamos borrar todos los alquileres de ese cliente?
- ¿Deberíamos permitir alquileres con un cliente que ya no existe?

De hecho no se podrá incluir un código que no esté en la tabla clientes; eso es lo que prohíbe la integridad referencial.

Eso causa problemas en las operaciones de borrado y modificación de registros; ya que si se ejecutan esas operaciones sobre la tabla principal (si se modifica o borra un cliente) quedarán filas en la tabla secundaria con la clave externa haciendo referencia a un valor que ya no existe.

Esto último se puede manipular de estas formas:

- **Prohibiendo la operación** (***no action)***.
- **Transmitiendo la operación en cascada *(cascade***). Es decir si se modifica o borra un cliente; también se modificarán o borrarán los alquileres relacionados con él.
- **Colocando nulos *(set null***) Las referencias al cliente en la tabla de alquileres se colocan como nulos (es decir, alquileres sin cliente).
- **Usando el valor por defecto** (default). Se colocan un valor por defecto en las claves externas relacionadas.

3.8. Mantenimiento de la integridad

Hemos visto la importancia de la integridad en el Modelo Relacional, y es por tanto necesario mantenerla en todo momento. ¿Cómo mantenemos dicha integridad? Gracias a un principio básico, **la integridad ha de ser mantenida en todo momento por el sistema.**

Veamos cómo se logra tal objetivo considerando las dos reglas de integridad más importantes:

1. **Integridad de entidad**: Se debe comprobar que los atributos que forman parte de una clave primaria son no nulos y que el valor de dichos atributos no se repite en los procesos de inserción y actualización.
2. **Integridad referencial:**
    1. **En inserción**, comprobar que el valor de la clave ajena es nula o coincide con un valor existente de la clave primaria de la tabla que referencia.
    2. **En actualización**, si se actualiza la clave ajena, comprobar las condiciones que definen la clave ajena (ver si el nuevo valor existe como clave primaria en la relación referenciada), y si se actualiza la clave primaria, actualizar en cadena la clave ajena.
    3. **En borrado**, si se borra la clave primaria, borrar en cascada o poner a null la clave ajena que hace referencia a dicha clave primaria, es decir, si eliminamos un registro de la tabla R1 referenciada, se eliminan en cascada todos los registros de la tabla R2 que hacen referencia al registro de R1, o se ponen a null las claves ajenas de R2 que referencia a dicha clave primaria.
    Decimos que se produce un borrado en cascada porque al borrar los registros de la tabla R2 puede hacerse posible borrar más registros de una tabla R3 que referenciaran a esos registros de R2, y así sucesivamente a lo largo de toda la cadena de referencias que existieran.