Tema 3. El modelo relacional

Contenido

1 Introducción 2
2 El modelo relacional: Estructura 3
3 El modelo relacional: relaciones 3
4 Tablas, tuplas y columnas 6
5 Clave primaria y claves ajenas 8
6 La integridad en el modelo relacional 10
6.1 Restricciones inherentes 10
6.2 Integridad de entidad 11
6.3 Integridad referencial 12
6.4 Restricciones semánticas 14
6.5 Mantenimiento de la integridad 14
7 Transformación de diagramas E-R al Modelo Relacional 15
7.1 Principios de transformación 15
7.2 Transformación de las entidades y sus atributos 15
7.3 Transformación de las relaciones y sus atributos 16
7.4 Transformación en casos especiales 17

1 Introducción

La **arquitectura relacional** se puede expresar en términos de tres niveles de abstracción:

- Nivel interno,
- Nivel conceptual y
- Nivel de visión.

Veamos cómo se relacionan los distintos componentes de la arquitectura relacional con los distintos niveles de abstracción:

- **Modelo relacional de datos:**

  En el nivel conceptual, el modelo relacional de datos está representado por una colección de relaciones (tablas) almacenadas.

- **Submodelo de datos:**

  En el nivel de visión, los esquemas externos de un sistema relacional se llaman submodelos relacionales de datos. Cada uno consta de una o más vistas para describir los datos requeridos por una aplicación dada.
  Una vista puede incluir datos de una o más tablas de datos. Cada programa de aplicación está provisto de un buffer ("Área de trabajo de usuario") donde el SGBD puede depositar los datos recuperados de la base para su procesamiento, o puede guardar temporalmente sus modificaciones antes de que el SGBD las escriba en la base de datos.

- **Esquema de almacenamiento:**

  En el nivel interno, cada tabla base se implanta como un archivo almacenado. Para las recuperaciones sobre las claves principal o secundaria se pueden establecer uno o más índices para indexar un archivo almacenado.

  El modelo relacional de datos es el modelo lógico en el que se basan la mayoría de los Sistemas Gestores de Bases de Datos (SGBD) usados en la actualidad, tales como ORACLE, Access, MySQL, SQL Server...

  A finales de los años sesenta **Edgar F. Codd** introdujo la teoría de las relaciones en el campo de las bases de datos. De esta manera los datos se estructuran lógicamente en forma de relaciones.

  De manera simple, una relación se representa mediante una **tabla**, en la que cada fila incluye una colección de valores que describen una entidad del mundo real. Cada fila se denomina tupla.

  Los objetivos que buscaba Codd con el modelo relacional van encaminados a obtener:

  - **Independencia física**.- Almacenamiento/manipulación. Un cambio físico en la base de datos no afecta a los programas.
  - **Independencia lógica**.- Añadir, eliminar o modificar elementos en la BD no debe repercutir en los programas y/o usuarios que acceden a ellos.
  - **Flexibilidad**.- Ofrecer al usuario los datos en la forma más adecuada a cada aplicación.
  - **Uniformidad**.- Las estructuras lógicas de los datos son tablas. Facilita la concepción y utilización de la BD por parte de los usuarios.
  - **Sencillez**.- Por las características anteriores y por los sencillos lenguajes de usuario que utiliza, el modelo relacional es fácil de comprender y utilizar por parte del usuario final.

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 pueden 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 El modelo relacional: Estructura

El modelo relacional se basa en el concepto matemático de relación, que se representa gráficamente mediante una tabla.

Codd, que era un experto matemático, utilizó terminología matemática para definir el modelo relacional, en concreto la de la teoría de conjuntos y de la lógica de predicados.

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 de la Normalización:** Compuesta por un conjunto de definiciones que permite estudiar las dependencias entre los atributos de las relaciones y proporciona métodos para un correcto diseño de las bases de datos relacionales. Esta teoría la veremos en el siguiente tema.

3 El modelo relacional: relaciones

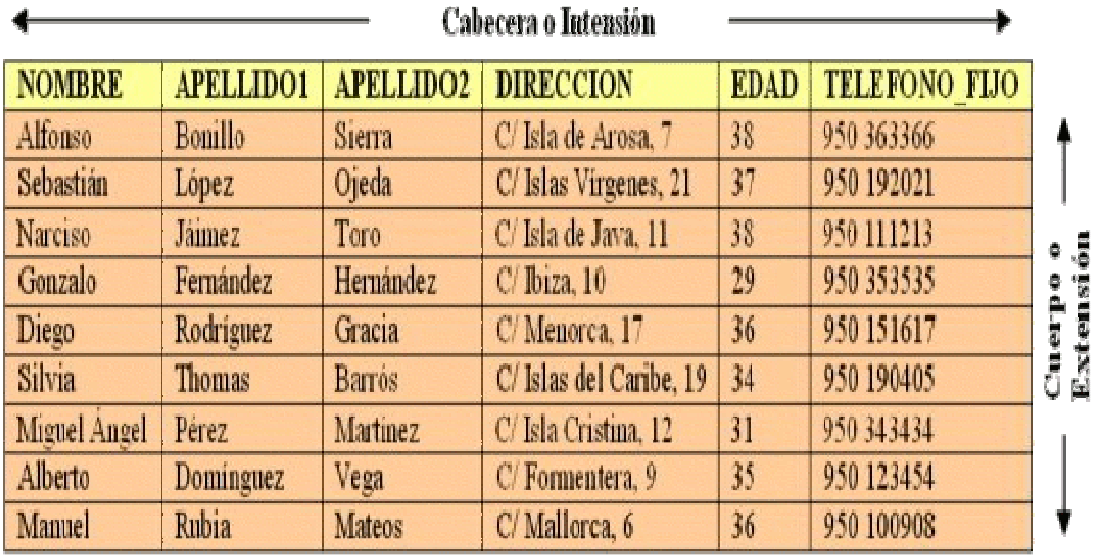
Veamos ahora en qué se traduce exactamente una **relación** dentro de una Base de Datos:

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

1. **Cabecera** (intensión). La cabecera está formada por un conjunto fijo de atributos. Es la parte invariante 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). El cuerpo 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.

El número de columnas que tiene una relación recibe el nombre de **grado de la relación**, y el número de filas recibe el nombre de **cardinalidad de la relación**.

Como ejemplo, la figura siguiente representa la relación PERSONA.



NOMBRE	APELLIDO1	APELLIDO2	DIRECCION	EDAD	TELEFONO_FIJO
Alfonso	Bonillo	Sierra	C/ Isla de Arosa, 7	38	950 363366
Sebastián	López	Ojeda	C/ Islas Vírgenes, 21	37	950 192021
Narciso	Jáimez	Toro	C/ Isla de Java, 11	38	950 111213
Gonzalo	Fernández	Hernández	C/ Ibiza, 10	29	950 353535
Diego	Rodríguez	Gracia	C/ Menorca, 17	36	950 151617
Silvia	Thomas	Barrós	C/ Islas del Caribe, 19	34	950 190405
Miguel Ángel	Pérez	Martínez	C/ Isla Cristina, 12	31	950 343434
Alberto	Domínguez	Vega	C/ Formentera, 9	35	950 123454
Manuel	Rubia	Mateos	C/ Mallorca, 6	36	950 100908

En este ejemplo el esquema de la relación o intensión es:

PERSONA (NOMBRE, APELLIDO1, APELLIDO2, DIRECCION, EDAD, TELEFONO_FIJO)

La extensión es el conjunto de 9 tuplas con los datos concretos de personas, el grado de la relación es 6 y la cardinalidad 9.

¿Cómo podemos diferenciar de una manera lógica el término Relación y el término Tabla? De la siguiente manera:

Una relación es una especie abstracta de objeto y una tabla es una representación concreta de ese objeto abstracto.

Estas tablas poseen ciertas propiedades, todas ellas consecuencia inmediata de la relación:

- **No existen tuplas repetidas**: Esta propiedad es consecuencia del hecho de que el cuerpo de la relación es un conjunto matemático (es decir, un conjunto de tuplas) y en matemáticas por definición los conjuntos no incluyen elementos repetidos. Esto se traduce en que dos registros de una misma relación deben diferir, al menos, en el valor de un campo.
- **Las tuplas no están ordenadas**: Esta propiedad sirve para ilustrar la diferencia entre una relación y una tabla, porque las filas de una tabla tienen un orden obvio de arriba hacia abajo, en tanto que las tuplas de una relación carecen de tal orden.
- **Los atributos no están ordenados**: Esta propiedad se desprende el hecho de que la cabecera de una relación se define también como conjunto. Las columnas de una tabla tienen un orden evidente de izquierda a derecha, pero los atributos de una relación carecen de tal orden.
- **Todos los valores de los atributos son atómicos**. Es decir, un atributo sólo puede tomar un valor en cada tupla. Otra forma de expresar esta propiedad es que **toda tabla es plana**, es decir, en el cruce de un atributo y una tupla sólo puede haber un valor.

En la siguiente tabla:

Persona					
NOMBRE	**APELLIDO1**	**APELLIDO2**	**DIRECCION**	**EDAD**	**TELEFONO_FIJO**
Alfonso	Bonillo	Sierra	C/ Isla de Arosa, 7	38	948363366 677586952
Sebastián	López	Ojeda	C/ Islas Vírgenes, 21	37	948192021
Narciso	Jáimez	Toro	C/ Isla de Java, 11	38	948111213
Gonzalo	Fernández	Hernández	C/ Ibiza, 10 C/ Calasparra, s/n	29	968353535
Silvia	Thomas	Barrós	C/ Islas del Caribe, 19	34	958190405

Las tuplas correspondientes a Alfonso Bonillo Sierra y Gonzalo Fernández Hernández no serían válidas por tener dos números de teléfono el primero y dos direcciones el segundo. Eso suele indicarse diciendo que **la tabla no es plana.**

Veamos con una imagen la correspondencia de los distintos modelos que representan una misma realidad:

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 la base de datos almacenada (son autónomas).
- **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 no 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. Pueden o no tener nombre y no persisten en la base de datos.
- **Resultados intermedios**. Son las relaciones que contienen los resultados de las subconsultas. Normalmente no tienen nombre y tampoco persisten en la base de datos.
- **Resultados temporales**. Son relaciones con nombre, similares a las relaciones base o a las instantáneas, pero la diferencia es que se destruyen automáticamente en algún momento.

4 Tablas, tuplas y columnas

Sabemos que una tabla es la materialización de un objeto abstracto llamado relación. Es importante no confundir estas relaciones con las que vimos en el Modelo E/R, y hay que tener cuidado ya que **de ahora en adelante llamaremos indistintamente tablas o relaciones a las relaciones que representan la información que queremos guardar**.

Una tabla se compone de una cabecera (nombre de la tabla, nombres de los atributos o columnas de la tabla, dominios de los atributos) y un cuerpo (conjunto de tuplas).

Las columnas son los campos o atributos que definen la tabla. Cada **campo** está **definido por**:

- **Nombre:** que describe los datos almacenados en él.
- **Dominio:** que indica el tipo de valores que contendrá dicho campo. Es un conjunto finito de valores homogéneos (son todos del mismo tipo) y atómicos (son indivisibles).

Cada **atributo** de una base de datos relacional se define sobre un **dominio**, pudiendo haber varios atributos definidos sobre el mismo dominio. Es evidente que dos atributos de la misma relación no pueden tener el mismo nombre. En unidades posteriores veremos los tipos de valores que se pueden definir para un atributo, aunque como ya sabes, los tipos de datos básicos son comunes para la mayoría de los lenguajes de programación, aunque depende del SGBD que utilicemos la definición exacta de los mismos.

La siguiente tabla nos muestra los dominios de los atributos de la relación OFICINA.

Oficina			
ATRIBUTO	**NOMBRE DEL ATRIBUTO**	**DESCRIPCIÓN**	**DEFINICIÓN**
Numero	**NUM_OFICINA**	Posibles valores de número de oficina	3 caracteres; rango 001- 099
Calle	**NOM_CALLE**	Nombres de calles de España	25 caracteres
Área	**NOM_AREA**	Nombres de áreas de las poblaciones de España	20 caracteres
Población	**NOM_POBLACION**	Nombres de las poblaciones de España	15 caracteres
teléfono	**NUM_TEL**	Números de teléfono de España	9 caracteres
Fax	**NUM_FAX**	Números de fax de España	9 caracteres

De esta manera la **cabecera de una relación está constituida por un conjunto de "m" campos**, y es expresada de la siguiente manera:

relacion (nombre_campo_1:dominio_1, nombre_campo_2:dominio_2,..., nombre_campo_m: dominio_m)

A esta expresión se le conoce también como **intensión de la relación**.

En el caso de la relación OFICINA su esquema sería:

OFICINA (NUM_OFICINA:cadena(3), NOM_CALLE:cadena(25), NOM_AREA:cadena(20), NOM_POBLACION:cadena (15), NUM_TEL:cadena(9), NUM_FAX:cadena(9))

Y su cuerpo, o extensión sería:

Oficina					
NUM_OFICINA	**NOM_CALLE**	**NOM_AREA**	**NOM_POBLACION**	**NUM_TEL**	**NUM_FAX**
005	Islas Vírgenes, 19	Sur	Olite	948191919	948363366
002	Islas del Caribe, 21	Norte	Olite	948212019	948192021
006	Menorca, 15	Sur	Pamplona	948242563	948111213
007	Isla Cristina, 12	Centro	Zaragoza	976256985	976353535
004	Rejoneador, 16	Centro	Teruel	978658965	978151617

Si consideramos la relación siguiente:

Persona					
NOMBRE	**APELLIDO1**	**APELLIDO2**	**DIRECCION**	**EDAD**	**TELEFONO_FIJO**
Alfonso	Bonillo	Sierra	C/ Isla de Arosa, 7. Pamplona	38	948363366
Sebastián	López	Ojeda	C/ Islas Vírgenes, 21. Olite	37	948192021
Narciso	Jáimez	Toro	C/ Isla de Java, 11. Carcastillo	38	948111213
Gonzalo	Fernández	Hernández	C/ Ibiza, 10. Murcia	29	968353535
Diego	Rodríguez	Gracia	C/ Menorca, 17. Valencia	36	960151617
Silvia	Thomas	Barrós	C/ Islas del Caribe, 19. Granada	34	958190405
Miguel Ángel	Pérez	Martínez	C/ Isla Cristina, 12. Murcia	31	968343434
Alberto	Domínguez	Vega	C/ Formentera, 9. Sevilla	35	950123454
Manuel	Rubia	Mateos	C/ Mallorca, 6. Jaén	36	953100908

La cabecera de la relación vendría dada por:

PERSONA (NOMBRE: cadena(30), APELLIDO1: cadena(30), APELLIDO2: cadena(30), DIRECCION: cadena(100), EDAD: número, TELEFONO_FIJO: cadena (9))

Como podemos observar el dominio para el campo NOMBRE es una cadena de caracteres de a lo sumo 30 caracteres. Es el mismo dominio que para los campos APELLIDO1 y APELLIDO2. Para el campo DIRECCION tenemos un dominio de una cadena de caracteres de a lo sumo 100 caracteres. Para el campo EDAD el dominio es un número y por último, para el campo TELEFONO_FIJO, el dominio es una cadena de caracteres de longitud 9 caracteres.

De esta manera estamos diciendo que el valor que consideremos para un campo concreto ha de ser siempre del tipo de dato que hemos definido el dominio. Esto es lo que significa que los **valores** de un campo son **homogéneos**.

Lo vemos con un ejemplo:

Para el campo NOMBRE no sería nunca válido el valor numérico 2006, sin embargo sí serían válidas todas las cadenas de caracteres que tuvieran como máximo 30 caracteres.

También hemos definido los valores de los campos como **atómicos**, es decir, indivisibles... y nos podemos preguntar ¿las direcciones de la relación PERSONA son atómicas? En principio puede parecer que no, ya que podríamos dividirlas en dirección y población...pero ahí está la cuestión, un valor se considera atómico o no según las restricciones del sistema de información que tengamos. De tal manera que en unos casos nos interesa tener la dirección lo más descompuesta posible para poder tener más posibilidades de combinación de los campos y en otros casos nos interesa más tener toda la dirección en un único campo.

Las tuplas constituyen los registros de la tabla. Un registro es un conjunto de **m** valores, correspondientes a los m campos de la relación. En nuestro ejemplo una de las tuplas de la relación PERSONA es:

(Alfonso, Bonillo, Sierra, C/ Isla de Arosa, 7. Pamplona, 38, 948363366)

Es decir, tenemos 9 tuplas o registros en nuestra relación (tabla), como ya hemos visto con anterioridad esta relación tiene cardinalidad 9.

Cabe destacar que **todos los registros de una relación deben tener el mismo número de campos**, aunque alguno esté vacío (se admite el valor NULO).

5 Clave primaria y claves ajenas

Hemos visto que en una relación no hay registros repetidos, lo que quiere decir que se pueden distinguir unos de otros de manera única. La forma de identificarlos es a través de los valores que toman sus atributos.

En el caso del Modelo Relacional también existen los conceptos de clave, clave primaria y clave ajena, y se parecen en gran medida a los vistos en el Modelo Entidad Relación.

Veamos los distintos tipos de claves que existen en el Modelo Relacional:

- **Clave Candidata**: Es un conjunto mínimo y no vacío de atributos que identifica unívocamente cada registro de una relación.

- **Clave Primaria**: Es la clave candidata que elige el usuario para identificar los registros de una relación. Se dice que una **clave primaria es compuesta cuando está formada por más de un atributo**. Se representa en el Modelo Relacional marcándola en negrita y con subrayado continuo. Nosotros marcaremos el / los campo(s) que formen la clave primaria con el símbolo # delante del nombre del campo.
- **Clave Alternativa**: Es cualquiera de las claves candidatas que no han sido elegidas como clave primaria.
- **Clave Ajena**: Es un conjunto no vacío de atributos de una relación R1 cuyos valores han de coincidir con los valores de la clave primaria de otra relación R2. R1 y R2 no tienen que ser necesariamente distintas (es el caso de las relaciones reflexivas). Se representa en el Modelo Relacional marcándola en negrita y subrayado discontinuo.

Si consideramos la relación OFICINA:

Oficina					
NUM_OFICINA	**NOM_CALLE**	**NOM_AREA**	**NOM_POBLACION**	**NUM_TEL**	**NUM_FAX**
005	Islas Vírgenes, 19	Sur	Olite	948191919	948363366
002	Islas del Caribe, 21	Norte	Olite	948212019	948192021
006	menoría, 15	Sur	Pamplona	948242563	948111213
007	Isla Cristina, 12	Centro	Zaragoza	976256985	976353535
004	Rejoneador, 16	Centro	Teruel	978658965	978151617

Es evidente que el atributo NUM_OFICINA es una clave candidata ya que identifica de manera única cada registro de la tabla, y en este caso además es la clave principal por no existir ninguna otra clave candidata. Pero si consideramos la tabla EMPLEADO:

Empleado					
NUM_EMPLEADO	**DNI**	**OFICINA**	**TELEF_FIJO**	**TELEF_MOVIL**	**NUM_FAX**
1234	12345678	005	948235689	655191919	948363366
1235	23456789	002	948235698	655212019	948192021
1236	13467925	006	948457812	655242563	948111213
1237	25836941	007	976456985	655256985	976353535
1238	42589637	004	978191921	655658965	978151617

Es evidente que tenemos dos claves candidatas, el atributo NUM_EMPLEADO y el atributo DNI, dado que ambos identifican de manera única a cada registro de la tabla, y por tanto cualquiera de ellos puede ser clave primaria de la relación. En este caso vamos a considerar el atributo DNI como clave primaria, con lo que el atributo NUM_EMPLEADO pasa a ser una clave alternativa de la tabla.

Si la relación OFICINA tuviera la siguiente estructura:

Oficina					
NUM_OFICINA	**NOM_CALLE**	**NOM_AREA**	**NOM_POBLACION**	**NUM_TEL**	**DIRECTOR**
005	Islas Vírgenes, 19	Sur	Olite	948191919	12345678
002	Islas del Caribe, 2	Norte	Olite	948212019	23456789
006	menoría, 15	Sur	Pamplona	948242563	25836941
007	Isla Cristina, 12	Centro	Zaragoza	976256985	23456789
004	Rejoneador, 16	Centro	Teruel	978658965	25836941

Podemos ver con claridad que el atributo DIRECTOR hace referencia al atributo DNI de la tabla EMPLEADO, que además es la clave primaria de dicha tabla, por lo que DIRECTOR es la clave ajena de la tabla OFICINA.

Las **claves primarias y ajenas** cumplen una serie de propiedades:

- Las claves ajenas son esenciales en el Modelo Relacional ya que permiten enlazar distintas tablas de la base de datos.
- Una clave ajena y la clave primaria de la tabla referenciada asociada han de estar definidas sobre los mismos dominios.
- Una tabla puede poseer más de una clave ajena, de hecho tendrá una clave ajena por cada tabla referenciada de la cual dependa.
- Una tabla puede no tener ninguna clave ajena.
- Una clave ajena puede relacionar una tabla consigo misma (relaciones reflexivas).

6 La integridad en el modelo relacional

Ahora que ya conocemos la estructura de datos del Modelo Relacional, ¿cómo podemos saber que nuestra representación es correcta y se corresponde con el universo del discurso?

Existen una serie de reglas, llamadas **reglas de integridad, que al cumplirse nos garantizan que los datos almacenados en nuestra base de datos son correctos, es decir, son una serie de normas que mantienen la corrección semántica de la base de datos.**

En los siguientes subapartados vamos a ir viendo cuáles son esas reglas y las implicaciones que tienen sobre el diseño y el uso de la base de datos.

6.1 Restricciones inherentes

Como hemos visto con anterioridad, detrás del concepto de relación hay una fuerte base matemática, lo que implica una serie de **restricciones** derivadas de su definición matemática. Éstas eran:

- No hay dos tuplas iguales.
- El orden de las tuplas no es significativo.

- El orden de los atributos (columnas) no es significativo.
- Cada atributo sólo puede tomar un único valor del dominio, 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.

Pero antes tenemos que definir el concepto de **NULO** en el Modelo Relacional.

Decimos que un **atributo es NULO (null) cuando dicho atributo es desconocido**. Un atributo nulo no se representa por el valor cero, ni por la cadena vacía, estos valores tienen significado. Se utiliza el valor null en un atributo cuando dicho atributo no tiene asociado ningún valor o que "su valor es desconocido".

Por ejemplo, se usa el valor null cuando "no se sabe la dirección de una persona", "se desconoce su fecha de nacimiento", etc.

Null puede asignarse a atributos de cualquier tipo, pero no se pueden comparar valores null de atributos de distinto tipo, lo que sí puede considerarse es si un atributo es null o no.

Veamos algunos ejemplos en los que se hace uso del valor null en distintos tipos de atributo en las tablas definidas con anterioridad:

EMPLEADO (1234, 12345678, 005, null, 655191919, 948363366)

En este caso no conocemos el valor del campo "TELEF_FIJO".

OFICINA (005, "Islas Vírgenes, 19", null, null, 948191919, 948363366)

En este caso no conocemos ni el campo "NOM_AREA", ni el "NOM_POBLACION".

6.2 Integridad de entidad

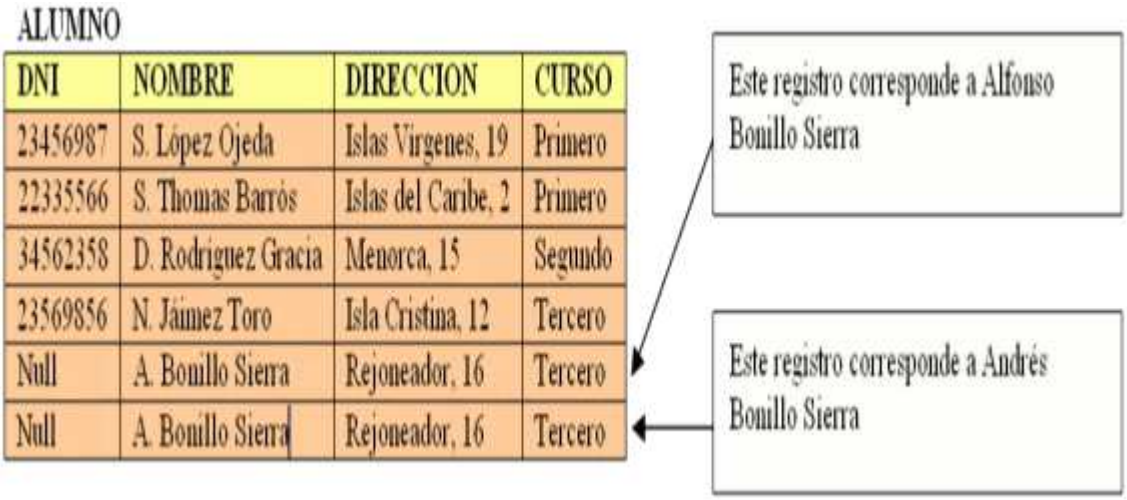
La regla de integridad de entidad es el mecanismo que garantiza la identificación y unicidad de las tuplas en una relación.

¿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



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? Es a través de la condición de que la clave primaria no puede ser null que podemos distinguir los distintos registros de nuestras tablas.

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

Pedido			
DNI_CLIENTE	**COD_TIENDA**	**FECHA_PEDIDO**	**CUANTIA**
23456987	005	19/04/2005	1904€
22335566	007	08/02/2006	806€
34562358	019	21/05/2006	2525€
23569856	012	20/06/2005	235€

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. Es por esta misma razón, que si alguno de los atributos que componen la clave primaria fuera null y aún así pudiéramos identificar claramente unos registros de otros, es porque ese atributo no debe formar parte de la clave principal. Es más esa no sería la clave principal de la relación porque no cumpliría la condición de minimalidad.

6.3 Integridad referencial

- ¿Qué ocurriría si introdujésemos en un pedido el código de un proveedor que no existe en la base de datos?
- ¿Qué ocurrirá con los pedidos que contienen un código de proveedor cuando ese proveedor fue dado de baja en la base de datos porque ya no nos suministra?
- ¿Deberíamos borrar todos los pedidos de ese proveedor?
- ¿Deberíamos permitir pedidos con un proveedor que ya no existe?

De estas cuestiones se ocupa la integridad referencial.

La regla de integridad referencial controla los casos de representación de interrelaciones (no tablas) en el universo del discurso, de modo que permite representar vínculos existentes entre tablas, evitando referencias no permitidas. Pero ¿qué dice esta regla?

Los valores de una clave ajena, o bien coinciden con los de la clave primaria a la que referencian o bien son nulos.

Dicho de otra manera**, si una relación R2 (relación que referencia) tiene un atributo que es la clave primaria de otra relación R1 (relación referenciada), todo valor de dicho atributo debe coincidir con un valor de la clave primaria de R1 o ser nulo**. El atributo en cuestión es, por tanto, una clave ajena de la relación R2.

Si consideramos las siguientes relaciones OFICINA y EMPLEADO

Oficina					
NUM_OFICINA	**NOM_CALLE**	**NOM_AREA**	**NOM_POBLACION**	**NUM_TEL**	**DIRECTOR**
005	Islas Vírgenes, 19	Sur	Olite	948191919	12345678
002	Islas del Caribe, 2	Norte	Olite	948212019	23456789
006	menoría, 15	Sur	Pamplona	948242563	25836941
007	Isla Cristina, 12	Centro	Zaragoza	976256985	23456789
004	Rejoneador, 16	Centro	Teruel	978658965	25836941

Empleado				
NUM_EMPLEADO	**DNI**	**NUM_OFICINA**	**TELEF_FIJO**	**NUM_FAX**
1234	12345678	005	948235689	948363366
1235	23456789	002	948235698	948192021
1236	13467925	006	948457812	948111213
1237	25836941	007	976456985	976353535
1238	42589637	null	978191921	978151617

Es evidente que el atributo NUM_OFICINA en la tabla EMPLEADO es clave ajena, que hace referencia a la clave primaria NUM_OFICINA de la tabla OFICINA.

- **Si la clave ajena referencia una tupla**, por ejemplo, NUM_OFICINA de la relación EMPLEADO, si toma un valor no nulo, este valor necesariamente ha de existir en una tupla de la relación OFICINA. No se admite ningún valor que no esté correctamente referenciado, de lo contrario estaríamos asignando un empleado a una oficina que no existe y eso sería una incoherencia.
- **Si la clave ajena no referencia una tupla**, por ejemplo, NUM_OFICINA de un empleado es null, significaría que ese empleado no tiene asignada ninguna oficina en ese momento.

6.4 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. Algunas de estas restricciones se pueden obtener claramente de las definiciones de clave primaria y clave ajena. Vemos algunas de estas restricciones:

- **Clave primaria**. Permite declarar un atributo o un conjunto de atributos como clave primaria de una relación, por lo que sus valores no se podrán repetir ni se admitirán valores nulos.
- **Unicidad.** Mediante la cual se indica que los valores de un conjunto de atributos (uno o más) no pueden repetirse en una relación. Esta restricción permite la definición de claves candidatas.
- **Obligatoriedad (NOT NULL)** de uno o más atributos, indicando de esta manera que ese conjunto de atributos no admite valores nulos.
- **Verificación**, 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. Una restricción de verificación se define sobre un único atributo y puede o no ser nombrada (a las restricciones se les puede poner nombre). Cada relación puede tener tantas restricciones de verificación como atributos tenga.
- **Disparador**, restricción en la que el usuario puede especificar libremente la respuesta del SGBD ante una determinada condición. Los disparadores son reglas de integridad procedimentales, siendo necesario que el usuario escriba el procedimiento que ha de aplicarse en caso de que se cumpla la condición.

6.5 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**.

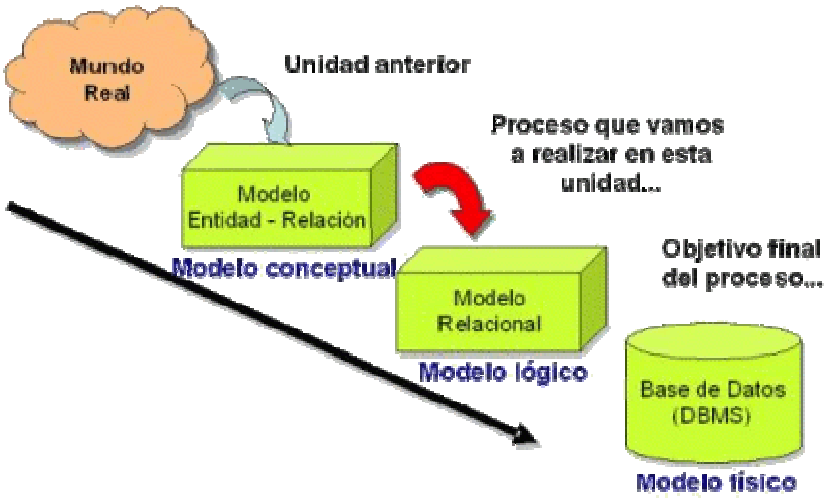
Veamos cómo se logra tal objetivo considerando las dos reglas de integridad más importantes:

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**:
    a. **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.
    b. **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.
    c. **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.

7 Transformación de diagramas E-R al Modelo Relacional

Si recuerdas de la unidad anterior, el **Modelo Entidad Relación** se basa en una percepción del mundo real basada en **objetos** (entidades) y **relaciones** entre dichos objetos. Se desarrolló para facilitar el proceso de diseño de bases de datos y utiliza diagramas para representar la estructura lógica general de las mismas. Constituye el modelo conceptual del sistema de información que queremos modelizar.

En esta unidad hemos dado un paso más en el proceso, y hemos estudiado el modelo lógico, esto es, el Modelo Relacional.



Por tanto, en esta unidad debemos ver los pasos a seguir para transformar un diagrama E/R (del modelo Entidad-Relación) al esquema relacional (del modelo relacional).

7.1 Principios de transformación

La transformación de un diagrama E/R al Modelo Relacional está basado en los siguientes principios:

- **Toda entidad se convierte en una relación (tabla).**
- **Toda interrelación (relación) N:M se transforma en una relación (tabla)**
- **Toda interrelación (relación) 1:N se traduce en el fenómeno de "propagación de clave" (se crea una clave ajena)**

Después de analizar los principios anteriores podemos pensar que en el paso del diseño conceptual (diagrama ER) al diseño lógico (esquema relacional) **se pierde información semántica**, ya que tanto las entidades como las relaciones se transforman en tablas, sin que haya una diferencia entre las tablas que provienen de entidades y las que provienen de relaciones. Pero la realidad es **diferente**, ya que gracias a la definición de las restricciones de integridad necesarias, nos aseguramos la conservación de la integridad de la base de datos, es decir que toda la semántica del universo del discurso quede reflejada en el esquema relacional.

7.2 Transformación de las entidades y sus atributos

Veamos cómo aplicar estos principios generales a cada elemento concreto:

- **Transformación de las Entidades.**
  Cada entidad que aparezca en el diagrama E/R se convierte en una tabla.

- **Transformación de Entidades Débiles**
  Las entidades débiles se transforman en una tabla, propagando la clave de la entidad fuerte. La clave propagada pasa a formar parte de la clave primaria de la entidad débil junto con el identificador que tenía la entidad débil si la debilidad es por identificación. Si la debilidad es existencia, la clave de la entidad débil está compuesta únicamente por su identificador.
- **Transformación de los Atributos de las entidades**
  Cada atributo de una entidad se transforma en una columna en la relación a la que ha dado lugar la entidad. Veamos cómo se definen cada uno de los tipos de atributos:
    - El, o los atributos principales de una entidad (es decir, la clave primaria de la entidad) pasan a ser la clave primaria de la relación. Se debe especificar que no son nulos.
    - El resto de atributos pasan a ser columnas de la tabla, pudiendo tomar valores nulos, a no ser que se indique lo contrario por restricciones de nuestro sistema de información.
- **Transformación de Atributos Compuestos.**
  El modelo relacional no permite representar atributos compuestos, por lo que se busca una alternativa, como:
    - Considerar el atributo compuesto como un atributo simple.
    - Eliminar el atributo compuesto y considerar cada uno de sus atributos como atributo simple de la entidad.

7.3 Transformación de las relaciones y sus atributos

- **Transformación de las relaciones (interrelaciones)**
  Dependiendo del tipo de relación y de la cardinalidad que tenga, existen diversas maneras de transformarlas:
    - **Relaciones N:M**. Se crea una nueva tabla que incluye los atributos de la propia relación (si los tuviera) y las claves primarias de las dos entidades, que forman la clave primaria de la nueva relación.
    - **Relaciones 1:N**. Estas interrelaciones se pueden transformar de dos maneras diferentes:
        a. **Propagar la clave principal** de la entidad que tiene cardinalidad máxima 1 a la que tiene N, y hacer desaparecer la tabla de la relación como tal. Si la relación tuviera pocos atributos asociados, estos atributos pasan a formar parte de la tabla correspondiente al tipo de entidad que participa con cardinalidad máxima muchos.
        b. **Transformarla en una nueva tabla** como si fuese de una relación de tipo N:M, es decir, incluyendo los atributos de la relación y las claves primarias de las dos entidades. Esta acción es recomendable sólo:
            - cuando es posible que aparezcan muchos nulos porque existen pocos elementos relacionados,
            - o cuando se prevé que dicha relación pasará en un futuro a ser de tipo N:M,
            - o cuando la relación tiene atributos propios.
    - **Relaciones 1:1.** Este es un caso particular de cualquiera de los dos casos anteriores, por lo que se podrían aplicar las reglas anteriores. De todas formas es recomendable tener en cuenta las siguientes recomendaciones:
        - Si la **relación es entre entidades con cardinalidades (0,1) y (0,1),** es mejor crear una relación para evitar tener muchos nulos como propagación de alguna de las claves a la otra.

- Si la **relación es entre entidades con cardinalidades (0,1) y (1,1),** es mejor propagar la clave de la entidad (1,1) a la (0,1).
- Si la **relación es entre entidades con cardinalidades (1,1) y (1,1),** la propagación es indiferente, y se hará atendiendo a los criterios de frecuencia de acceso (consulta, modificación, inserción, etc.) a cada una de las tablas en cuestión.

- **Transformación de los atributos de relaciones.**

Si la relación se transforma en una tabla, todos sus atributos pasan a ser columnas de la tabla. En el caso en que alguno de los atributos de la relación sea clave primaria, deberá ser incluido como parte de la clave primaria en dicha tabla.

- **Transformación de relaciones exclusivas.**

Para soportar relaciones exclusivas debemos definir las restricciones pertinentes en cada caso. Por **ejemplo**, en el caso en que exista una exclusividad en la edición de un libro por parte de una editorial o de una universidad, estas dos relaciones se resuelven mediante el mecanismo de propagación de la clave, llevando las claves primarias de editorial y universidad a libro. Ya veremos más adelante, que tendremos que utilizar entonces la cláusula CHECK de SQL para introducir las restricciones pertinentes.

7.4 Transformación en casos especiales

- **Transformación de relaciones ternarias (grado 3).**

  - **Relaciones muchos a muchos a muchos:**

    Este tipo de relación se transforma en una tabla cuya clave primaria es la concatenación de las claves primarias de las tablas surgidas al transformar las entidades que forman parte de la relación. Junto a estos atributos se incluyen los atributos propios de la relación. Cada uno de los atributos que forman la clave primaria de esta tabla son a la vez claves ajenas respecto a cada una de las tablas donde dicho atributo es clave primaria.

  - **Relaciones muchos a muchos a uno:**

    Este tipo de relación se transforma en una tabla cuya clave primaria es la concatenación de las claves primarias de las tablas que corresponden a la cardinalidad **M** surgidas al transformar las entidades que forman parte de la relación. Junto a estos atributos se incluyen los atributos propios de la relación más la clave primaria de la tabla que corresponde a la cardinalidad 1. Cada uno de los atributos que forman la clave primaria de esta tabla (y los atributos añadidos de la relación de cardinalidad 1) son claves ajenas respecto a cada una de las tablas donde dicho atributo es clave primaria.

- **Transformación de la generalización (tipos y subtipos).**

  Los tipos y subtipos no son objetos que se puedan representar en el modelo relacional estándar. Existen pues, varias posibilidades para su transformación:

  - Englobar todos los atributos de una entidad y sus subtipos en **una sola tabla**, añadiendo el atributo que permite distinguir los subtipos. También habrá que especificar las restricciones semánticas adecuadas.
  - Crear una **tabla para el supertipo**, y tantas tablas como subtipos existan. Ésta es la mejor opción desde el punto de vista semántico, pero es menos eficiente que la opción anterior.
  - Crear sólo tablas para los subtipos, añadiendo en cada una de ellas los atributos pertenecientes al supertipo. Habitualmente ésta es la opción más utilizada.