# El modelo relacional y el álgebra relacional

Dolors Costal Costa

P06/M2109/02148

www.uoc.edu

# Índice

# Introducción

Esta unidad didáctica está dedicada al estudio del modelo de datos relacional y del álgebra relacional.

El concepto de *modelo de datos* se ha presentado en otra unidad didáctica. En ésta se profundiza en un modelo de datos concreto: el **modelo relacional**, que actualmente tiene una gran relevancia. Sus conceptos fundamentales están bien asentados y, además, los sistemas de gestión de bases de datos relacionales son los más extendidos en su utilización práctica. Por estos motivos pensamos que es importante conocerlo.

Consultad el concepto de *modelo de datos* en la unidad didáctica "Introducción a las bases de datos" de este curso.

El estudio del modelo relacional sirve, además, de base para los contenidos de otra unidad, dedicada al lenguaje SQL. Este lenguaje permite definir y manipular bases de datos relacionales. Los fundamentos del modelo relacional resultan imprescindibles para conseguir un buen dominio del SQL.

Las construcciones del SQL se estudian en la unidad didáctica "El lenguaje SQL".

En esta unidad se analizan también las **operaciones del álgebra relacional**, que sirven para hacer consultas a una base de datos. Es preciso conocer estas operaciones porque nos permiten saber qué servicios de consulta debe proporcionar un lenguaje relacional. Otra aportación del álgebra relacional es que facilita la comprensión de algunas de las construcciones del lenguaje SQL que se estudiarán en otra unidad didáctica de este curso. Además, constituye la base para el estudio del tratamiento de las consultas que efectúan los SGBD internamente (especialmente en lo que respecta a la optimización de consultas). Este último tema queda fuera del ámbito del presente curso, pero es relevante para estudios más avanzados sobre bases de datos.

# Objetivos

En los materiales didácticos de esta unidad encontraréis las herramientas indispensables para alcanzar los siguientes objetivos:

**1.** Conocer los fundamentos del modelo de datos relacional.

**2.** Saber distinguir las características que debe tener un sistema de gestión de bases de datos relacional para que sea coherente con los fundamentos del modelo relacional.

**3.** Comprender las ventajas del modelo relacional que derivan del alto grado de independencia de los datos que proporciona, y de la simplicidad y la uniformidad del modelo.

**4.** Conocer las operaciones del álgebra relacional.

**5.** Saber utilizar las operaciones del álgebra relacional para consultar una base de datos.

# 1. Introducción al modelo relacional

El **modelo relacional** es un modelo de datos y, como tal, tiene en cuenta los tres aspectos siguientes de los datos:

**1)** La **estructura**, que debe permitir representar la información que nos interesa del mundo real.

**2)** La **manipulación**, a la que da apoyo mediante las operaciones de actualización y consulta de los datos.

**3)** La **integridad**, que es facilitada mediante el establecimiento de reglas de integridad; es decir, condiciones que los datos deben cumplir.

El concepto de *modelo de datos* se ha explicado en la unidad didáctica "Introducción a las bases de datos"de este curso.

Un **sistema de gestión de bases de datos relacional** (SGBDR) da apoyo a la definición de datos mediante la estructura de los datos del modelo relacional, así como a la manipulación de estos datos con las operaciones del modelo; además, asegura que se satisfacen las reglas de integridad que el modelo relacional establece.

El concepto de *SGBD* ha sido presentado en la unidad didáctica "Introducción a las bases de datos" de este curso.

Los principios del modelo de datos relacional fueron establecidos por E.F. Codd en los años 1969 y 1970. De todos modos, hasta la década de los ochenta no se empezaron a comercializar los primeros SGBD relacionales con rendimientos aceptables. Cabe señalar que los SGBD relacionales que se comercializan actualmente todavía no soportan todo lo que establece la teoría relacional hasta el último detalle.

El **principal objetivo del modelo de datos relacional** es facilitar que la base de datos sea percibida o vista por el usuario como una estructura lógica que consiste en un conjunto de relaciones y no como una estructura física de implementación. Esto ayuda a conseguir un alto grado de independencia de los datos.

Un objetivo adicional del modelo es conseguir que esta estructura lógica con la que se percibe la base de datos sea simple y uniforme. Con el fin de proporcionar simplicidad y uniformidad, toda la información se representa de una única manera: mediante valores explícitos que contienen las relaciones (no se utilizan conceptos como por ejemplo apuntadores entre las relaciones). Con el mismo propósito, todos los valores de datos se consideran atómicos; es decir, no es posible descomponerlos.

Hay que precisar que un SGBD relacional, en el nivel físico, puede emplear cualquier estructura de datos para implementar la estructura lógica formada

por las relaciones. En particular, a nivel físico, el sistema puede utilizar apuntadores, índices, etc. Sin embargo, esta implementación física queda oculta al usuario.

En los siguientes apartados estudiaremos la estructura de los datos, las operaciones y las reglas de integridad del modelo relacional. Hay dos formas posibles de enfocar el estudio de los contenidos de este módulo. La primera consiste en seguirlos en orden de exposición. De este modo, se van tratando todos los elementos de la teoría del modelo relacional de forma muy precisa y en un orden lógico. Otra posibilidad, sin embargo, es empezar con la lectura del resumen final del módulo y leer después todo el resto de los contenidos en el orden normal. El resumen describe los aspectos más relevantes de la teoría relacional que se explican y, de este modo, proporciona una visión global de los contenidos del módulo que, para algunos estudiantes, puede ser útil comprender antes de iniciar un estudio detallado. 

# 2. Estructura de los datos

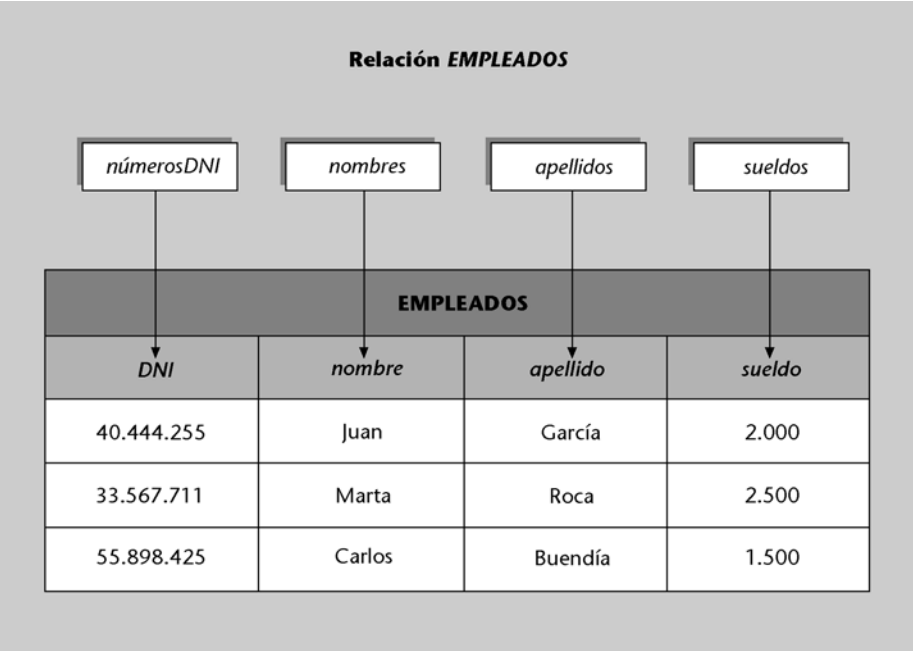
El modelo relacional proporciona una estructura de los datos que consiste en un conjunto de relaciones con objeto de representar la información que nos interesa del mundo real.

La estructura de los datos del modelo relacional se basa, pues, en el concepto de *relación*.

## 2.1. Visión informal de una relación

En primer lugar, presentaremos el concepto de *relación* de manera informal. Se puede obtener una buena idea intuitiva de lo que es una relación si la visualizamos como una tabla o un fichero. En la figura 1 se muestra la visualización tabular de una relación que contiene datos de empleados. Cada fila de la tabla contiene una colección de valores de datos relacionados entre sí; en nuestro ejemplo, son los datos correspondientes a un mismo empleado. La tabla tiene un nombre (*EMPLEADOS*) y también tiene un nombre cada una de sus columnas *(DNI, nombre, apellido* y *sueldo*). El nombre de la tabla y los de las columnas ayudan a entender el significado de los valores que contiene la tabla. Cada columna contiene valores de un cierto dominio; por ejemplo, la columna *DNI* contiene valores del dominio *númerosDNI*.

Figura1




**Conjunto de relaciones**

Una base de datos relacional consta de un conjunto de relaciones, cada una de las cuales se puede visualizar de este modo tan sencillo. La estructura de los datos del modelo relacional resulta fácil de entender para el usuario.

Si definimos las relaciones de forma más precisa, nos daremos cuenta de que presentan algunas características importantes que, en la visión superficial que hemos presentado, quedan ocultas. Estas características son las que motivan que el concepto de *relación* sea totalmente diferente del de *fichero*, a pesar de que, a primera vista, relaciones y ficheros puedan parecer similares.

## 2.2. Visión formal de una relación

A continuación definimos formalmente las relaciones y otros conceptos que están vinculados a ellas, como por ejemplo *dominio, esquema de relación*, etc.

Un **dominio *D*** es un conjunto de valores atómicos. Por lo que respecta al modelo relacional, *atómico* significa indivisible; es decir, que por muy complejo o largo que sea un valor atómico, no tiene una estructuración interna para un SGBD relacional.

Los dominios pueden ser de dos tipos:

**1) Dominios predefinidos**, que corresponde a los tipos de datos que normalmente proporcionan los lenguajes de bases de datos, como por ejemplo los enteros, las cadenas de caracteres, los reales, etc.

**2) Dominios definidos por el usuario**, que pueden ser más específicos. Toda definición de un dominio debe constar, como mínimo, del nombre del dominio y de la descripción de los valores que forman parte de éste.

**Dominio definido por el usuario**

Por ejemplo, el usuario puede definir un dominio para las edades de los empleados que se denomine *dom_edad* y que contenga los valores enteros que están entre 16 y 65.

Un **relación** se compone del **esquema** (o intensión de la relación) y de la **extensión**.

Si consideramos la representación tabular anterior (figura 1), el esquema correspondería a la cabecera de la tabla y la extensión correspondería al cuerpo:

Figura 2

| **Empleados** | | | |
|---|---|---|---|
| ***DNI*** | ***nombre*** | ***apellido*** | ***sueldo*** |
| 40.444.255 | Juan | García | 2.000 |
| 33.567.711 | Marta | Roca | 2.500 |
| 55.898.425 | Carlos | Buendía | 1.500 |

▶Esquema

▶Extensión

El **esquema de la relación** consiste en un nombre de relación *R* y un conjunto de atributos $\{A_1, A_2, ..., A_n\}$.

**Nombre y conjunto de atributos de la relación *EMPLEADOS***

Si tomamos como ejemplo la figura 1, el nombre de la relación es *EMPLEADOS* y el conjunto de atributos es {*DNI, nombre, apellido, sueldo*}.

Tomaremos la convención de denotar el esquema de la relación de la forma siguiente: R($A_1$, $A_2$, ..., $A_n$), donde $R$ es el nombre la relación y $A_1$, $A_2$, ..., $A_n$ es una ordenación cualquiera de los atributos que pertenecen al conjunto {$A_1$, $A_2$, ..., $A_n$}.

**Denotación del esquema de la relación *EMPLEADOS***

El esquema de la relación de la figura 1 se podría denotar, por ejemplo, como *EMPLEADOS*(*DNI, nombre, apellido, sueldo*), o también, *EMPLEADOS*(*nombre, apellido, DNI, sueldo*), porque cualquier ordenación de sus atributos se considera válida para denotar el esquema de una relación.

Un **atributo $A_i$** es el nombre del papel que ejerce un dominio $D$ en un esquema de relación. $D$ es el **dominio de $A_i$** y se denota como dominio ($A_i$).

**Dominio del atributo DNI**

Según la figura 1, el atributo *DNI* corresponde al papel que ejerce el dominio *númerosDNI* en el esquema de la relación *EMPLEADOS* y, entonces, dominio(*DNI*) = *númerosDNI*.

Conviene observar que cada atributo es único en un esquema de relación, porque no tiene sentido que un mismo dominio ejerza dos veces el mismo papel en un mismo esquema. Por consiguiente, no puede ocurrir que en un esquema de relación haya dos atributos con el mismo nombre. En cambio, sí que se puede repetir un nombre de atributo en relaciones diferentes. Los dominios de los atributos, por el contrario, no deben ser necesariamente todos diferentes en una relación.

**Ejemplo de atributos diferentes con el mismo dominio**

Si tomamos como ejemplo el esquema de relación *PERSONAS(DNI, nombre, apellido, telcasa, teltrabajo*), los atributos *telcasa* y *teltrabajo* pueden tener el mismo dominio: dominio(*telcasa*)= = *teléfono* y dominio(*teltrabajo*) = *teléfono*.

En este caso, el dominio *teléfono* ejerce dos papeles diferentes en el esquema de relación: el de indicar el teléfono particular de una persona y el de indicar el del trabajo.

La **extensión de la relación de esquema R($A_1$, $A_2$, ..., $A_n$)** es un conjunto de tuplas $t_i$ ($i$ = 1, 2, ..., $m$), donde cada tupla $t_i$ es, a su vez un conjunto de pares $t_i$ = {<$A_1$:$v_{i1}$>, <$A_2$:$v_{i2}$> ... >$A_n$:$v_{in}$>} y, para cada par <$A_j$:$v_{ij}$>, se cumple que $v_{ij}$ es un valor de dominio($A_j$), o bien un valor especial que denominaremos *nulo*.

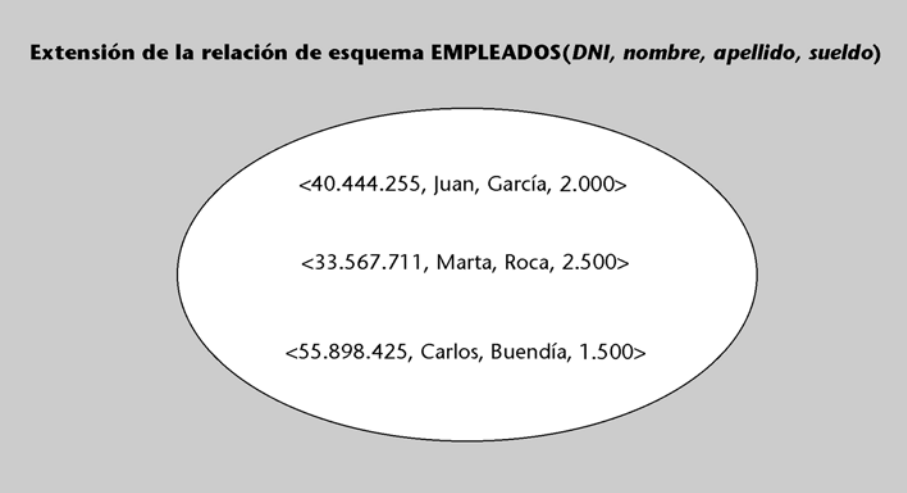
**Algunos autores...**

... denominan *tablas*, *columnas* y *filas* a las relaciones, los atributos y las tuplas, respectivamente.

Para simplificar, tomaremos la convención de referirnos a una tupla $t_i$ = {<$A_1$:$v_{i1}$>, <$A_2$:$v_{i2}$>, ..., <$A_n$:$v_{in}$>} que pertenece a la extensión del esquema denotado como $R$($A_1$, $A_2$, ..., $A_n$), de la forma siguiente: $t_i$ = <$v_{i1}$, $v_{i2}$, ..., $v_{in}$>.

Si denotamos el esquema de la relación representada en la figura 1 como EMPLEADOS(*DNI, nombre, apellido, sueldo*), el conjunto de tuplas de su extensión será el de la figura siguiente:

Figura 3




**Esta figura...**

... nos muestra la extensión de *EMPLEADOS* en forma de conjunto, mientras que las figuras anteriores nos la mostraban en forma de filas de una tabla. La representación tabular es más cómoda, pero no refleja la definición de extensión con tanta exactitud.

Si en una tupla $t_i = <v_{i1}, v_{i2}, ..., v_{in}>$, el valor $v_{ij}$ es un **valor nulo**, entonces el valor del atributo $A_j$ es desconocido para la tupla $t_i$ de la relación, o bien no es aplicable a esta tupla.

**Ejemplo de valor nulo**

Podríamos tener un atributo *telcasa* en la relación *EMPLEADOS* y se podría dar el caso de que un empleado no tuviese teléfono en su casa, o bien que lo tuviese, pero no se conociese su número. En las dos situaciones, el valor del atributo *telcasa* para la tupla correspondiente al empleado sería el valor nulo.

El **grado de una relación** es el número de atributos que pertenecen a su esquema.

**Grado de la relación *EMPLEADOS***

El grado de la relación de esquema EMPLEADOS(*DNI, nombre, apellido, sueldo*), es 4.

La **cardinalidad** de una relación es el número de tuplas que pertenecen a su extensión.

**Cardinalidad de la relación *EMPLEADOS***

Observando la figura 3 se deduce que la cardinalidad de la relación *EMPLEADOS* es 3.

## 2.3. Diferencias entre relaciones y ficheros

A primera vista, relaciones y ficheros resultan similares. Los registros y los campos que forman los ficheros se parecen a las tuplas y a los atributos de las relaciones, respectivamente.

A pesar de esta similitud superficial, la visión formal de relación que hemos presentado establece algunas características de las relaciones que las hacen diferentes de los ficheros clásicos. A continuación describimos estas características:

**1) Atomicidad de los valores de los atributos**: los valores de los atributos de una relación deben ser atómicos; es decir, no deben tener estructura interna. Esta característica proviene del hecho de que los atributos siempre deben tomar un valor de su dominio o bien un valor nulo, y de que se ha establecido que los valores de los dominios deben ser atómicos en el modelo relacional.

El objetivo de la atomicidad de los valores es dar simplicidad y uniformidad al modelo relacional.

**2) No-repetición de las tuplas**: en un fichero clásico puede ocurrir que dos de los registros sean exactamente iguales; es decir, que contengan los mismos datos. En el caso del modelo relacional, en cambio, no es posible que una relación contenga tuplas repetidas. Esta característica se deduce de la misma definición de la extensión de una relación. La extensión es un conjunto de tuplas y, en un conjunto, no puede haber elementos repetidos.

**3) No-ordenación de las tuplas:** de la definición de la extensión de una relación como un conjunto de tuplas se deduce también que estas tuplas no estarán ordenadas, teniendo en cuenta que no es posible que haya una ordenación entre los elementos de un conjunto.

La finalidad de esta característica es conseguir que, mediante el modelo relacional, se puedan representar los hechos en un nivel abstracto que sea independiente de su estructura física de implementación. Más concretamente, aunque los SGBD relacionales deban proporcionar una implementación física que almacenará las tuplas de las relaciones en un orden concreto, esta ordenación no es visible si nos situamos en el nivel conceptual.

El concepto de *extensión de una relación* se ha explicado en el subapartado 2.2. de esta unidad didáctica.

**Ejemplo de no-ordenación de las tuplas**

En una base de datos relacional, por ejemplo, no tiene sentido consultar la "primera tupla" de la relación *EMPLEADOS*.

**4) No-ordenación de los atributos**: el esquema de una relación consta de un nombre de relación $R$ y un conjunto de atributos $\{A_1, A_2, ..., A_n\}$. Así pues, no hay un orden entre los atributos de un esquema de relación, teniendo en cuenta que estos atributos forman un conjunto.

El concepto de *esquema de una relación* se ha explicado en el subapartado 2.2. de esta unidad didáctica.

Como en el caso anterior, el objetivo de esta característica es representar los hechos en un nivel abstracto, independientemente de su implementación física.

**Ejemplo de no-ordenación de los atributos**

El esquema de relación EMPLEADOS(*DNI, nombre, apellido, sueldo*) denota el mismo esquema de relación que EMPLEADOS(*nombre, apellido, DNI, sueldo*).

## 2.4. Clave candidata, clave primaria y clave alternativa de las relaciones

Toda la información que contiene una base de datos debe poderse identificar de alguna forma. En el caso particular de las bases de datos que siguen el modelo relacional, para identificar los datos que la base de datos contiene, se pueden utilizar las claves candidatas de las relaciones. A continuación definimos qué se entiende por *clave candidata*, *clave primaria* y *clave alternativa* de una relación. Para hacerlo, será necesario definir el concepto de *superclave*.

**Por ejemplo, ...**

... si se almacena información sobre los empleados de una empresa, es preciso tener la posibilidad de distinguir qué datos corresponden a cada uno de los diferentes empleados.

Una **superclave de una relación de esquema R($A_1$, $A_2$, ..., $A_n$)** es un subconjunto de los atributos del esquema tal que no puede haber dos tuplas en la extensión de la relación que tengan la misma combinación de valores para los atributos del subconjunto.

**Observad que...**

... toda relación tiene, por lo menos, una superclave, que es la formada por todos los atributos de su esquema. Esto se debe a la propiedad que cumple toda relación de no tener tuplas repetidas.

En el ejemplo de EMPLEADOS(*DNI, NSS, nombre, apellido, teléfono*) esta superclave sería: {*DNI, NSS, nombre, apellido, teléfono*}.

Una superclave, por lo tanto, nos permite identificar todas las tuplas que contiene la relación.

**Algunas superclaves de la relación *EMPLEADOS***

En la relación de esquema EMPLEADOS(*DNI, NSS, nombre, apellido, teléfono*), algunas de las superclaves de la relación serían los siguientes subconjuntos de atributos: {*DNI, NSS, nombre, apellido, teléfono*}, {*DNI, apellido*}, {*DNI*} y {*NSS*}.

Una **clave candidata de una relación** es una superclave *C* de la relación que cumple que ningún subconjunto propio de *C* es superclave.

**Notad que, ...**

... puesto que toda relación tiene por lo menos una superclave, podemos garantizar que toda relación tiene como mínimo una clave candidata.

Es decir, *C* cumple que la eliminación de cualquiera de sus atributos da un conjunto de atributos que no es superclave de la relación. Intuitivamente, una clave candidata permite identificar cualquier tupla de una relación, de manera que no sobre ningún atributo para hacer la identificación.

**Claves candidatas de *EMPLEADOS***

En la relación de esquema EMPLEADOS(*DNI, NSS, nombre, apellido, teléfono*), sólo hay dos claves candidatas: {*DNI*} y {*NSS*}.

Habitualmente, una de las claves candidatas de una relación se designa clave primaria de la relación. La **clave primaria** es la clave candidata cuyos valores se utilizarán para identificar las tuplas de la relación.

**Relación con una clave candidata**

Si una relación sólo tiene una clave candidata, entonces esta clave candidata debe ser también su clave primaria. Ya que todas las relaciones tienen como mínimo una clave candidata, podemos garantizar que, para toda relación, será posible designar una clave primaria.

El diseñador de la base de datos es quien elige la clave primaria de entre las claves candidatas.

Las claves candidatas no elegidas como primaria se denominan **claves alternativas**.

Utilizaremos la convención de subrayar los atributos que forman parte de la clave primaria en el esquema de la relación. Así pues, R($\underline{A_1, A_2, ..., A_i}$, ..., $A_n$) indica que los atributos $A_1$, $A_2$, ..., $A_i$ forman la clave primaria de *R*.

**Elección de la clave primaria de *EMPLEADOS***

En la relación de esquema EMPLEADOS*(DNI, NSS, nombre, apellido, teléfono*), donde hay dos claves candidatas, {*DNI*} y {*NSS*}, se puede elegir como clave primaria {*DNI*}. Lo indicaremos subrayando el atributo *DNI* en el esquema de la relación EMPLEADOS(*DNI, NSS, nombre, apellido, teléfono*). En este caso, la clave {*NSS*} será una clave alternativa de *EMPLEADOS*.

Es posible que una clave candidata o una clave primaria conste de más de un atributo.

**Clave primaria de la relación *DESPACHOS***

En la relación de esquema DESPACHOS(*edificio, número, superficie*), la clave primaria está formada por los atributos *edificio* y *número*. En este caso, podrá ocurrir que dos despachos diferentes estén en el mismo edificio, o bien que tengan el mismo número, pero nunca pasará que tengan la misma combinación de valores para *edificio* y *número*.

## 2.5. Claves foráneas de las relaciones

Hasta ahora hemos estudiado las relaciones de forma individual, pero debemos tener en cuenta que una base de datos relacional normalmente contiene más de una relación, para poder representar distintos tipos de hechos que suceden en el mundo real. Por ejemplo, podríamos tener una pequeña base de datos que contuviese dos relaciones: una denominada *EMPLEADOS*, que almacenaría datos de los empleados de una empresa, y otra con el nombre *DESPACHOS*, que almacenaría los datos de los despachos que tiene la empresa.

Debemos considerar también que entre los distintos hechos que se dan en el mundo real pueden existir lazos o vínculos. Por ejemplo, los empleados que trabajan para una empresa pueden estar vinculados con los despachos de la empresa, porque a cada empleado se le asigna un despacho concreto para trabajar.

En el modelo relacional, para reflejar este tipo de vínculos, tenemos la posibilidad de expresar conexiones entre las distintas tuplas de las relaciones. Por ejemplo, en la base de datos anterior, que tiene las relaciones *EMPLEADOS* y *DESPACHOS*, puede ser necesario conectar tuplas de *EMPLEADOS* con tuplas de *DESPACHOS* para indicar qué despacho tiene asignado cada empleado.

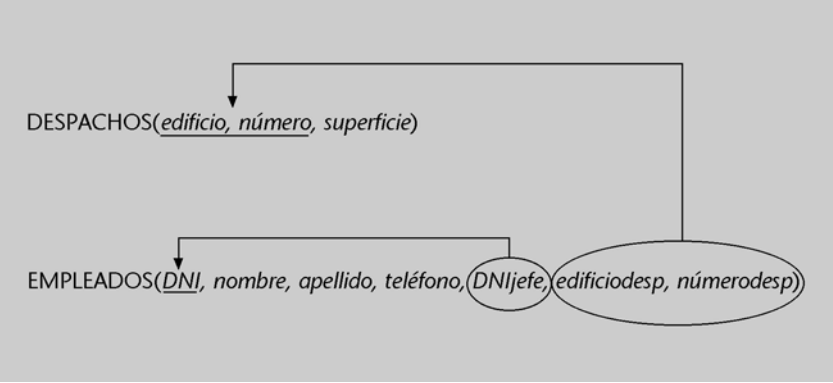
En ocasiones, incluso puede ser necesario reflejar lazos entre tuplas que pertenecen a una misma relación. Por ejemplo, en la misma base de datos anterior puede

ser necesario conectar determinadas tuplas de *EMPLEADOS* con otras tuplas de *EMPLEADOS* para indicar, para cada empleado, quién actúa como su jefe.

El mecanismo que proporcionan las bases de datos relacionales para conectar tuplas son las claves foráneas de las relaciones. Las **claves foráneas** permiten establecer conexiones entre las tuplas de las relaciones. Para hacer la conexión, una clave foránea tiene el conjunto de atributos de una relación que referencian la clave primaria de otra relación (o incluso de la misma relación).

**Claves foráneas de la relación *EMPLEADOS***

En la figura siguiente, la relación EMPLEADOS(*DNI*, nombre, apellido, teléfono, *DNIjefe*, *edificiodesp*, *númerodesp*), tiene una clave foránea formada por los atributos *edificiodesp* y *númerodesp* que se refiere a la clave primaria de la relación DESPACHOS(*edificio*, *número*, *superficie*). Esta clave foránea indica, para cada empleado, el despacho donde trabaja. Además, el atributo *DNIjefe* es otra clave foránea que referencia la clave primaria de la misma relación *EMPLEADOS*, e indica, para cada empleado, quien es su jefe.




Las claves foráneas tienen por objetivo establecer una conexión con la clave primaria que referencian. Por lo tanto, los valores de una clave foránea deben estar presentes en la clave primaria correspondiente, o bien deben ser valores nulos. En caso contrario, la clave foránea representaría una referencia o conexión incorrecta.

**Ejemplo**

En la relación de esquema EMPLEADOS(*DNI*, *nombre*, *apellido*, *DNIjefe*, *edificiodesp*, *númerodesp*), la clave foránea {*edificiodesp*, *númerodesp*} referencia la relación DESPACHOS(*edificio*, *número*, *superficie*). De este modo, se cumple que todos los valores que no son nulos de los atributos *edificiodesp* y *númerodesp* son valores que existen para los atributos *edificio* y *número* de *DESPACHOS*, tal y como se puede ver a continuación:

- Relación *DESPACHOS*:

| **DESPACHOS** | | |
|---|---|---|
| ***edificio*** | ***número*** | ***superficie*** |
| Marina | 120 | 10 |
| Marina | 122 | 15 |
| Marina | 230 | 20 |
| Diagonal | 120 | 10 |

- Relación *EMPLEADOS*

| EMPLEADOS | | | | | |
|---|---|---|---|---|---|
| ***DNI*** | ***nombre*** | ***apellido*** | ***DNIjefe*** | ***edificiodesp*** | ***númerodesp*** |
| 40.444.255 | Juan | García | NULO | Marina | 120 |
| 33.567.711 | Marta | Roca | 40.444.255 | Marina | 120 |
| 55.898.425 | Carlos | Buendía | 40.444.255 | Diagonal | 120 |
| 77.232.144 | Elena | Pla | 40.444.255 | NULO | NULO |

Supongamos que hubiese un empleado con los valores <55.555.555, María, Casagran, NULO, París, 400>. Puesto que no hay ningún despacho con los valores París y 400 para *edificio* y *número*, la tupla de este empleado hace una referencia incorrecta; es decir, indica un despacho para el empleado que, de hecho, no existe.

Es preciso señalar que en la relación *EMPLEADOS* hay otra clave foránea, {*DNIjefe*}, que referencia la misma relación *EMPLEADOS*, y entonces se cumple que todos los valores que no son nulos del atributo *DNIjefe* son valores que existen para el atributo *DNI* de la misma relación *EMPLEADOS*.

A continuación estableceremos de forma más precisa qué se entiende por *clave foránea*.

Una **clave foránea de una relación *R*** es un subconjunto de atributos del esquema de la relación, que denominamos *CF* y que cumple las siguientes condiciones:

**1)** Existe una relación *S* (*S* no debe ser necesariamente diferente de *R*) que tiene por clave primaria *CP*.

**2)** Se cumple que, para toda tupla *t* de la extensión de *R*, los valores para *CF* de *t* son valores nulos o bien valores que coinciden con los valores para *CP* de alguna tupla *s* de *S*.

Y entonces, se dice que la clave foránea *CF* referencia la clave primaria *CP* de la relación *S*, y también que la clave foránea *CF* referencia la relación *S*.

**Conviene subrayar que, ...**

... tal y como ya hemos mencionado, el modelo relacional permite representar toda la información mediante valores explícitos que contienen las relaciones, y no le hace falta nada más. De este modo, las conexiones entre tuplas de las relaciones se expresan con los valores explícitos de las claves foráneas de las relaciones, y no son necesarios conceptos adicionales (por ejemplo, apuntadores entre tuplas), para establecer estas conexiones. Esta característica da simplicidad y uniformidad al modelo.

De la noción que hemos dado de clave foránea se pueden extraer varias consecuencias: 

**1)** Si una clave foránea *CF* referencia una clave primaria *CP*, el número de atributos de *CF* y de *CP* debe coincidir.

**Ejemplo de coincidencia del número de atributos de *CF* y *CP***

En el ejemplo anterior, tanto la clave foránea {*edificiodesp, númerodesp*} como la clave primaria que referencia {*edificio, número*} tienen dos atributos. Si no sucediese así, no sería posible que los valores de *CF* existieran en *CP*.

**2)** Por el mismo motivo, se puede establecer una correspondencia (en concreto, una biyección) entre los atributos de la clave foránea y los atributos de la clave primaria que referencia.

**Ejemplo de correspondencia entre los atributos de *CF* y los de *CP***

En el ejemplo anterior, a *edificiodesp* le corresponde el atributo *edificio*, y a *númerodesp* le corresponde el atributo *número*.

**3)** También se deduce de la noción de *clave foránea* que los dominios de sus atributos deben coincidir con los dominios de los atributos correspondientes a la clave primaria que referencia. Esta coincidencia de dominios hace que sea posible que los valores de la clave foránea coincidan con valores de la clave primaria referenciada.

**Ejemplo de coincidencia de los dominios**

En el ejemplo anterior, se debe cumplir que dominio(*edificiodesp*) = dominio(*edificio*) y también que dominio(*númerodesp*) = dominio(*número*).

**Lectura recomendada**

Encontraréis explicaciones detalladas sobre la coincidencia de dominios en la obra siguiente:

**C.J. Date** (2001). *Introducción a los sistemas de bases de datos* (7ª ed., cap. 19). Prentice Hall.

Observad que, de hecho, esta condición se podría relajar, y se podría permitir que los dominios no fuesen exactamente iguales, sino que sólo fuesen, y de alguna forma que convendría precisar, dominios "compatibles". Para simplificarlo, nosotros supondremos que los dominios deben ser iguales en todos los casos en que, según Date (2001), se aceptarían dominios "compatibles".

**Ejemplo de atributo que forma parte de la clave primaria y de una clave foránea**

Puede suceder que algún atributo de una relación forme parte tanto de la clave primaria como de una clave foránea de la relación. Esto se da en las relaciones siguientes: EDIFICIOS(*nombreedificio, dirección*), y DESPACHOS(*edificio, número*, *superficie*), donde {*edificio*} es una clave foránea que referencia *EDIFICIOS*.

En este ejemplo, el atributo edificio forma parte tanto de la clave primaria como de la clave foránea de la relación *DESPACHOS*.

## 2.6. Creación de las relaciones de una base de datos

Hemos visto que una base de datos relacional consta de varias relaciones. Cada relación tiene varios atributos que toman valores de unos ciertos dominios; también tiene una clave primaria y puede tener una o más claves foráneas. Los **lenguajes de los SGBD relacionales** deben proporcionar la forma de definir todos estos elementos para crear una base de datos.

Más adelante se verá con detalle la sintaxis y el significado de las sentencias de definición de la base de datos para el caso concreto del lenguaje SQL.

El lenguaje SQL se explica en la unidad didáctica "El lenguaje SQL" de este curso.

# 3. Operaciones del modelo relacional

Las operaciones del modelo relacional deben permitir manipular datos almacenados en una base de datos relacional y, por lo tanto, estructurados en forma de relaciones. La manipulación de datos incluye básicamente dos aspectos: la actualización y la consulta.

La sintaxis y el funcionamiento de las operaciones de actualización y consulta, en el caso concreto del lenguaje relacional SQL, se estudian con detalle en la unidad "El lenguaje SQL" de este curso.

La **actualización de los datos** consiste en hacer que los cambios que se producen en la realidad queden reflejados en las relaciones de la base de datos.

**Ejemplo de actualización**

Si una base de datos contiene, por ejemplo, información de los empleados de una empresa, y la empresa contrata a un empleado, será necesario reflejar este cambio añadiendo los datos del nuevo empleado a la base de datos.

Existen tres operaciones básicas de actualización:

**a) Inserción**, que sirve para añadir una o más tuplas a una relación.

**b) Borrado**, que sirve para eliminar una o más tuplas de una relación.

**c) Modificación**, que sirve para alterar los valores que tienen una o más tuplas de una relación para uno o más de sus atributos.

La **consulta de los datos** consiste en la obtención de datos deducibles a partir de las relaciones que contiene la base de datos.

**Ejemplo de consulta**

Si una base de datos contiene, por ejemplo, información de los empleados de una empresa, puede interesar consultar el nombre y apellido de todos los empleados que trabajan en un despacho situado en un edificio que tiene por nombre *Marina*.

La obtención de los datos que responden a una consulta puede requerir el análisis y la extracción de datos de una o más de las relaciones que mantiene la base de datos.

Según la forma como se especifican las consultas, podemos clasificar los lenguajes relacionales en dos tipos:

**1) Lenguajes basados en el álgebra relacional**. El álgebra relacional se inspira en la teoría de conjuntos. Si queremos especificar una consulta, es necesario

seguir uno o más pasos que sirven para ir construyendo, mediante operaciones del álgebra relacional, una nueva relación que contenga los datos que responden a la consulta a partir de las relaciones almacenadas. Los lenguajes basados en el álgebra relacional son **lenguajes procedimentales**, ya que los pasos que forman la consulta describen un procedimiento.

**2) Lenguajes basados en el cálculo relacional**. El cálculo relacional tiene su fundamento teórico en el cálculo de predicados de la lógica matemática. Proporciona una notación que permite formular la definición de la relación donde están los datos que responden la consulta en términos de las relaciones almacenadas. Esta definición no describe un procedimiento; por lo tanto, se dice que los lenguajes basados en el cálculo relacional son **lenguajes declarativos** (no procedimentales).

El **lenguaje SQL**, en las sentencias de consulta, combina construcciones del álgebra relacional y del cálculo relacional con un predominio de las construcciones del cálculo. Este predominio determina que SQL sea un lenguaje declarativo.

El **estudio del álgebra relacional** presenta un interés especial, pues ayuda a entender qué servicios de consulta debe proporcionar un lenguaje relacional, facilita la comprensión de algunas de las construcciones del lenguaje SQL y también sirve de base para el tratamiento de las consultas que efectúan los SGBD internamente. Este último tema queda fuera del ámbito del presente curso, pero es necesario para estudios más avanzados sobre bases de datos.

El álgebra relacional se explica en el apartado 5 de esta unidad didáctica.

# 4. Reglas de integridad

Una base de datos contiene unos datos que, en cada momento, deben reflejar la realidad o, más concretamente, la situación de una porción del mundo real. En el caso de las bases de datos relacionales, esto significa que la extensión de las relaciones (es decir, las tuplas que contienen las relaciones) deben tener valores que reflejen la realidad correctamente.

Suele ser bastante frecuente que determinadas configuraciones de valores para las tuplas de las relaciones no tengan sentido, porque no representan ninguna situación posible del mundo real.

**Un sueldo negativo**

En la relación de esquema EMPLEADOS(*DNI*, *nombre*, *apellido*, *sueldo*), una tupla que tiene un valor de –1.000 para el sueldo probablemente no tiene sentido, porque los sueldos no pueden ser negativos.

Denominamos **integridad** la propiedad de los datos de corresponder a representaciones plausibles del mundo real.

Como es evidente, para que los datos sean íntegros, es preciso que cumplan varias condiciones.

El hecho de que los sueldos no puedan ser negativos es una condición que se debería cumplir en la relación *EMPLEADOS*.

En general, las condiciones que garantizan la integridad de los datos pueden ser de dos tipos:

**1)** Las **restricciones de integridad de usuario** son condiciones específicas de una base de datos concreta; es decir, son las que se deben cumplir en una base de datos particular con unos usuarios concretos, pero que no son necesariamente relevantes en otra base de datos.

**Restricción de integridad de usuario en *EMPLEADOS***

Éste sería el caso de la condición anterior, según la cual los sueldos no podían ser negativos. Observad que esta condición era necesaria en la base de datos concreta de este ejemplo porque aparecía el atributo *sueldo*, al que se quería dar un significado; sin embargo, podría no ser necesaria en otra base de datos diferente donde, por ejemplo, no hubiese sueldos.

**2)** Las **reglas de integridad de modelo**, en cambio, son condiciones más generales, propias de un modelo de datos, y se deben cumplir en toda base de datos que siga dicho modelo.

**Ejemplo de regla de integridad del modelo de datos relacional**

En el caso del modelo de datos relacional, habrá una regla de integridad para garantizar que los valores de una clave primaria de una relación no se repitan en tuplas diferentes

de la relación. Toda base de datos relacional debe cumplir esta regla que, por lo tanto, es una regla de integridad del modelo.

Los SGBD deben proporcionar la forma de definir las restricciones de integridad de usuario de una base de datos; una vez definidas, deben velar por su cumplimiento.

La forma de definir estas restricciones con el lenguaje SQL se explica en la unidad "El lenguaje SQL" de este curso.

Las reglas de integridad del modelo, en cambio, no se deben definir para cada base de datos concreta, porque se consideran preestablecidas para todas las base de datos de un modelo. Un SGBD de un modelo determinado debe velar por el cumplimiento de las reglas de integridad preestablecidas por su modelo.

A continuación estudiaremos con detalle las **reglas de integridad del modelo relacional**, reglas que todo SGBD relacional debe obligar a cumplir.

## 4.1. Regla de integridad de unicidad de la clave primaria

La regla de integridad de unicidad está relacionada con la definición de clave primaria. Concretamente, establece que toda clave primaria que se elija para una relación no debe tener valores repetidos.

Es preciso destacar que el mismo concepto de *clave primaria* implica esta condición. El concepto de *clave primaria* se ha explicado en el subapartado 2.4. de esta unidad didáctica.

**Ejemplo**

Tenemos la siguiente relación:

| DESPACHOS | | |
|---|---|---|
| ***edificio*** | ***número*** | ***superficie*** |
| Marina | 120 | 10 |
| Marina | 122 | 15 |
| Marina | 230 | 20 |
| Diagonal | 120 | 10 |

En esta relación, dado que la clave primaria está formada por *edificio* y *número*, no hay ningún despacho que repita tanto *edificio* como *número* de otro despacho. Sin embargo, sí se repiten valores de *edificio* (por ejemplo, Marina); y también se repiten valores de *número* (120). A pesar de ello, el *edificio* y el *número* no se repiten nunca al mismo tiempo.

A continuación explicamos esta regla de forma más precisa.

La **regla de integridad de unicidad de la clave primaria** establece que si el conjunto de atributos *CP* es la clave primaria de una relación *R*, entonces la extensión de *R* no puede tener en ningún momento dos tuplas con la misma combinación de valores para los atributos de *CP*.

Un SGBD relacional deberá garantizar el cumplimiento de esta regla de integridad en todas las inserciones, así como en todas las modificaciones que afecten a atributos que pertenecen a la clave primaria de la relación.

**Ejemplo**

Tenemos la siguiente relación:

| DESPACHOS | | |
|---|---|---|
| ***edificio*** | ***número*** | ***superficie*** |
| Marina | 120 | 10 |
| Marina | 122 | 15 |
| Marina | 230 | 20 |
| Diagonal | 120 | 10 |

En esta relación no se debería poder insertar la tupla <Diagonal, 120, 30>, ni modificar la tupla <Marina, 122, 15>, de modo que pasara a ser <Marina, 120, 15>.

## 4.2. Regla de integridad de entidad de la clave primaria

La regla de integridad de entidad de la clave primaria dispone que los atributos de la clave primaria de una relación no pueden tener valores nulos.

**Ejemplo**

Tenemos la siguiente relación:

| DESPACHOS | | |
|---|---|---|
| ***edificio*** | ***número*** | ***superficie*** |
| Marina | 120 | 10 |
| Marina | 122 | 15 |
| Marina | 230 | 20 |
| Diagonal | 120 | 10 |

En esta relación, puesto que la clave primaria está formada por *edificio* y *número*, no hay ningún despacho que tenga un valor nulo para *edificio*, ni tampoco para *número*.

Esta regla es necesaria para que los valores de las claves primarias puedan identificar las tuplas individuales de las relaciones. Si las claves primarias tuviesen valores nulos, es posible que algunas tuplas no se pudieran distinguir.

**Ejemplo de clave primaria incorrecta con valores nulos**

En el ejemplo anterior, si un despacho tuviese un valor nulo para *edificio* porque en un momento dado el nombre de este edificio no se conoce, por ejemplo <NULO, 120, 30>, la clave primaria no nos permitiría distinguirlo del despacho <Marina, 120, 10> ni del despacho <Diagonal, 120,10>. No podríamos estar seguros de que el valor desconocido de *edificio* no es ni Marina ni Diagonal.

A continuación definimos esta regla de forma más precisa.

La **regla de integridad de entidad de la clave primaria** establece que si el conjunto de atributos *CP* es la clave primaria de una relación *R*, la extensión de *R* no puede tener ninguna tupla con algún valor nulo para alguno de los atributos de CP.

Un SGBD relacional tendrá que garantizar el cumplimiento de esta regla de integridad en todas las inserciones y, también, en todas las modificaciones que afecten a atributos que pertenecen a la clave primaria de la relación.

**Ejemplo**

En la relación *DESPACHOS* anterior, no se debería insertar la tupla <Diagonal, NULO, 15>. Tampoco debería ser posible modificar la tupla <Marina, 120, 10> de modo que pasara a ser <NULO, 120, 10>.

## 4.3. Regla de integridad referencial

La regla de integridad referencial está relacionada con el concepto de *clave foránea*. Concretamente, determina que todos los valores que toma una clave foránea deben ser valores nulos o valores que existen en la clave primaria que referencia.

Observad que todo lo que impone la regla de integridad referencial viene implicado por la misma noción de *clave foránea* que se ha explicado en el subapartado 2.5 de esta unidad.

**Ejemplo**

Si tenemos las siguientes relaciones:

- Relación *DESPACHOS*:

| **DESPACHOS** | | |
|---|---|---|
| ***edificio*** | ***número*** | ***superficie*** |
| Marina | 120 | 10 |
| Marina | 122 | 15 |
| Marina | 230 | 20 |
| Diagonal | 120 | 10 |

- Relación *EMPLEADOS*:

| **EMPLEADOS** | | | | |
|---|---|---|---|---|
| ***DNI*** | ***nombre*** | ***apellido*** | ***edificiodesp*** | ***númerodesp*** |
| 40.444.255 | Juan | García | Marina | 120 |
| 33.567.711 | Marta | Roca | Marina | 120 |
| 55.898.425 | Carlos | Buendía | Diagonal | 120 |
| 77.232.144 | Elena | Pla | NULO | NULO |

donde *edificiodesp* y *númerodesp* de la relación *EMPLEADOS* forman una clave foránea que referencia la relación *DESPACHOS*. Debe ocurrir que los valores no nulos de *edificiodesp* y *númerodesp* de la relación *EMPLEADOS* estén en la relación *DESPACHOS* como valores de *edificio* y *número*. Por ejemplo, el empleado <40.444.255, Juan García, Marina, 120> tiene el valor Marina para *edificiodesp*, y el valor 120 para *númerodesp*, de modo que en la relación *DESPACHOS* hay un despacho con valor Marina para *edificio* y con valor 120 para *número*.

La necesidad de la regla de integridad relacional proviene del hecho de que las claves foráneas tienen por objetivo establecer una conexión con la clave primaria que referencian. Si un valor de una clave foránea no estuviese presente

en la clave primaria correspondiente, representaría una referencia o una conexión incorrecta.

**Referencia incorrecta**

Supongamos que en el ejemplo anterior hubiese un empleado con los valores <56.666.789, Pedro, López, Valencia, 325>. Ya que no hay un despacho con los valores Valencia y 325 para *edificio* y *número*, la tupla de este empleado hace una referencia incorrecta; es decir, indica un despacho para el empleado que, de hecho, no existe.

A continuación explicamos la regla de modo más preciso.

La **regla de integridad referencial** establece que si el conjunto de atributos *CF* es una clave foránea de una relación *R* que referencia una relación *S* (no necesariamente diferente de *R*), que tiene por clave primaria *CP*, entonces, para toda tupla *t* de la extensión de *R*, los valores para el conjunto de atributos *CF* de *t* son valores nulos, o bien valores que coinciden con los valores para *CP* de alguna tupla *s* de *S*.

En el caso de que una tupla *t* de la extensión de *R* tenga valores para *CF* que coincidan con los valores para *CP* de una tupla *s* de *S*, decimos que *t* es una tupla que referencia *s* y que *s* es una tupla que tiene una clave primaria referenciada por *t*.

Un SGBD relacional tendrá que hacer cumplir esta regla de integridad. Deberá efectuar comprobaciones cuando se produzcan las siguientes operaciones:

**a)** Inserciones en una relación que tenga una clave foránea.

**b)** Modificaciones que afecten a atributos que pertenecen a la clave foránea de una relación.

**c)** Borrados en relaciones referenciadas por otras relaciones.

**d)** Modificaciones que afecten a atributos que pertenecen a la clave primaria de una relación referenciada por otra relación.

**Ejemplo**

Retomamos el ejemplo anterior, donde *edificiodesp* y *númerodesp* de la relación *EMPLEADOS* forman una clave foránea que referencia la relación *DESPACHOS*:

- Relación *DESPACHOS*:

| **DESPACHOS** | | |
|---|---|---|
| ***edificio*** | ***número*** | ***superficie*** |
| Marina | 120 | 10 |
| Marina | 122 | 15 |
| Marina | 230 | 20 |
| Diagonal | 120 | 10 |

- Relación *EMPLEADOS*:

| EMPLEADOS | | | | |
|---|---|---|---|---|
| ***DNI*** | ***nombre*** | ***apellido*** | ***edificiodesp*** | ***númerodesp*** |
| 40.444.255 | Juan | García | Marina | 120 |
| 33.567.711 | Marta | Roca | Marina | 120 |
| 55.898.425 | Carlos | Buendía | Diagonal | 120 |
| 77.232.144 | Elena | Pla | NULO | NULO |

Las siguientes operaciones provocarían el incumplimiento de la regla de integridad referencial:

- Inserción de <12.764.411, Jorge, Puig, Diagonal, 220> en *EMPLEADOS*.
- Modificación de <40.444.255, Juan, García, Marina, 120> de *EMPLEADOS* por <40.444.255, Juan, García, Marina, 400>.
- Borrado de <Marina, 120, 10> de *DESPACHOS*.
- Modificación de <Diagonal, 120, 10> de *DESPACHOS* por <París, 120, 10>.

Un SGBD relacional debe procurar que se cumplan las reglas de integridad del modelo. Una forma habitual de mantener estas reglas consiste en rechazar toda operación de actualización que deje la base de datos en un estado en el que alguna regla no se cumpla. En algunos casos, sin embargo, el SGBD tiene la posibilidad de aceptar la operación y efectuar acciones adicionales compensatorias, de modo que el estado que se obtenga satisfaga las reglas de integridad, a pesar de haber ejecutado la operación.

Esta última política se puede aplicar en las siguientes operaciones de actualización que violarían la regla de integridad:

**a)** Borrado de una tupla que tiene una clave primaria referenciada.

**b)** Modificación de los valores de los atributos de la clave primaria de una tupla que tiene una clave primaria referenciada.

En los casos anteriores, algunas de las políticas que se podrán aplicar serán las siguientes: **restricción**, **actualización en cascada** y **anulación**. A continuación explicamos el significado de las tres posibilidades mencionadas.

### 4.3.1. Restricción

La política de restricción consiste en no aceptar la operación de actualización.

Más concretamente, la **restricción en caso de borrado**, consiste en no permitir borrar una tupla si tiene una clave primaria referenciada por alguna clave foránea.

De forma similar, la **restricción en caso de modificación** consiste en no permitir modificar ningún atributo de la clave primaria de una tupla si tiene una clave primaria referenciada por alguna clave foránea.

**Ejemplo de aplicación de la restricción**

Supongamos que tenemos las siguientes relaciones:

- Relación *CLIENTES*:

| **CLIENTES** | |
|---|---|
| ***numcliente*** | ... |
| 10 | – |
| 15 | – |
| 18 | – |

- Relación *PEDIDOS_PENDIENTES*

| **PEDIDOS_PENDIENTES** | | |
|---|---|---|
| ***numped*** | ... | ***numcliente**** |
| 1.234 | – | 10 |
| 1.235 | – | 10 |
| 1.236 | – | 15 |

* {*numcliente*} referencia *CLIENTES*.

**a)** Si aplicamos la restricción en caso de borrado y, por ejemplo, queremos borrar al cliente número 10, no podremos hacerlo porque tiene pedidos pendientes que lo referencian.

**b)** Si aplicamos la restricción en caso de modificación y queremos modificar el número del cliente 15, no será posible hacerlo porque también tiene pedidos pendientes que lo referencian.

### 4.3.2. Actualización en cascada

La política de actualización en cascada consiste en permitir la operación de actualización de la tupla, y en efectuar operaciones compensatorias que propaguen en cascada la actualización a las tuplas que la referenciaban; se actúa de este modo para mantener la integridad referencial.

Más concretamente, la **actualización en cascada en caso de borrado** consiste en permitir el borrado de una tupla *t* que tiene una clave primaria referenciada, y borrar también todas las tuplas que referencian *t*.

De forma similar, la **actualización en cascada en caso de modificación** consiste en permitir la modificación de atributos de la clave primaria de una tupla *t* que tiene una clave primaria referenciada, y modificar del mismo modo todas las tuplas que referencian *t*.

**Ejemplo de aplicación de la actualización en cascada**

Supongamos que tenemos las siguientes relaciones:

- Relación *EDIFICIOS*:

| **EDIFICIOS** | |
|---|---|
| ***nombreedificio*** | ***...*** |
| Marina | – |
| Diagonal | – |

- Relación *DESPACHOS*:

| **DESPACHOS** | | |
|---|---|---|
| ***edificio**** | ***número*** | ***superficie*** |
| Marina | 120 | 10 |
| Marina | 122 | 15 |
| Marina | 230 | 20 |
| Diagonal | 120 | 10 |

* {*edificio*} referencia *EDIFICIOS*.

**a)** Si aplicamos la actualización en cascada en caso de borrado y, por ejemplo, queremos borrar el edificio Diagonal, se borrará también el despacho Diagonal 120 que hay en el edificio, y nos quedará:

- Relación *EDIFICIOS*:

| **EDIFICIOS** | |
|---|---|
| ***nombreedificio*** | ***...*** |
| Marina | – |

- Relación *DESPACHOS*:

| **DESPACHOS** | | |
|---|---|---|
| ***edificio**** | ***número*** | ***superficie*** |
| Marina | 120 | 10 |
| Marina | 122 | 15 |
| Marina | 230 | 20 |

* {*edificio*} referencia *EDIFICIOS*.

**b)** Si aplicamos la actualización en cascada en caso de modificación, y queremos modificar el nombre del edificio Marina por Mar, también se cambiará Marina por Mar en los despachos Marina 120, Marina 122 y Marina 230, y nos quedará:

- Relación *EDIFICIOS*:

| **EDIFICIOS** | |
|---|---|
| ***nombreedificio*** | ***...*** |
| Mar | – |

- Relación *DESPACHOS*:

| **DESPACHOS** | | |
|---|---|---|
| ***edificio**** | ***número*** | ***superficie*** |
| Mar | 120 | 10 |
| Mar | 122 | 15 |
| Mar | 230 | 20 |

* {*edificio*} referencia *EDIFICIOS*.

### 4.3.3. Anulación

Esta política consiste en permitir la operación de actualización de la tupla y en efectuar operaciones compensatorias que pongan valores nulos a los atributos de la clave foránea de las tuplas que la referencian; esta acción se lleva a cabo para mantener la integridad referencial.

Puesto que generalmente los SGBD relacionales permiten establecer que un determinado atributo de una relación no admite valores nulos, sólo se puede aplicar la política de anulación si los atributos de la clave foránea sí los admiten.

Más concretamente, la **anulación en caso de borrado** consiste en permitir el borrado de una tupla *t* que tiene una clave referenciada y, además, modificar todas las tuplas que referencian *t*, de modo que los atributos de la clave foránea correspondiente tomen valores nulos.

De forma similar, la **anulación en caso de modificación** consiste en permitir la modificación de atributos de la clave primaria de una tupla *t* que tiene una clave referenciada y, además, modificar todas las tuplas que referencian *t*, de modo que los atributos de la clave foránea correspondiente tomen valores nulos.

**Ejemplo de aplicación de la anulación**

El mejor modo de entender en qué consiste la anulación es mediante un ejemplo. Tenemos las siguientes relaciones:

- Relación *VENDEDORES*:

| **VENDEDORES** | |
|---|---|
| ***numvendedor*** | ... |
| 1 | – |
| 2 | – |
| 3 | – |

- Relación *CLIENTES*:

| **CLIENTES** | | |
|---|---|---|
| ***numcliente*** | ... | ***vendedorasig**** |
| 23 | – | 1 |
| 35 | – | 1 |
| 38 | – | 2 |
| 42 | – | 2 |
| 50 | – | 3 |

* {*vendedorasig*} referencia *VENDEDORES*.

**a)** Si aplicamos la anulación en caso de borrado y, por ejemplo, queremos borrar al vendedor número 1, se modificarán todos los clientes que lo tenían asignado, y pasarán a tener un valor nulo en *vendedorasig*. Nos quedará:

- Relación *VENDEDORES*:

| **VENDEDORES** | |
|---|---|
| ***numvendedor*** | ... |
| 2 | – |
| 3 | – |

- Relación *CLIENTES*:

| **CLIENTES** | | |
|---|---|---|
| ***numcliente*** | ... | ***vendedorasig**** |
| 23 | – | NULO |
| 35 | – | NULO |
| 38 | – | 2 |
| 42 | – | 2 |
| 50 | – | 3 |

* {*vendedorasig*} referencia *VENDEDORES*.

**b)** Si aplicamos la anulación en caso de modificación, y ahora queremos cambiar el número del vendedor 2 por 5, se modificarán todos los clientes que lo tenían asignado y pasarán a tener un valor nulo en *vendedorasig*. Nos quedará:

- Relación *VENDEDORES*:

| **VENDEDORES** | |
|---|---|
| ***numvendedor*** | ... |
| 5 | – |
| 3 | – |

- Relación *CLIENTES*:

| **CLIENTES** | | |
|---|---|---|
| ***numcliente*** | ... | ***vendedorasig**** |
| 23 | – | NULO |
| 35 | – | NULO |
| 38 | – | NULO |
| 42 | – | NULO |
| 50 | – | 3 |

* {*vendedorasig*} referencia *VENDEDORES*.

### 4.3.4. Selección de la política de mantenimiento de la integridad referencial

Hemos visto que en caso de borrado o modificación de una clave primaria referenciada por alguna clave foránea hay varias políticas de mantenimiento de la regla de integridad referencial.

La forma de definir estas políticas de mantenimiento de la integridad con el lenguaje SQL se explica en la unidad "El lenguaje SQL" de este curso.

El diseñador puede elegir para cada clave foránea qué política se aplicará en caso de borrado de la clave primaria referenciada, y cuál en caso de modificación de ésta. El diseñador deberá tener en cuenta el significado de cada clave foránea concreta para poder elegir adecuadamente.

**Aplicación de políticas diferentes**

Puede ocurrir que, para una determinada clave foránea, la política adecuada en caso de borrado sea diferente de la adecuada en caso de modificación. Por ejemplo, puede ser necesario aplicar la restricción en caso de borrado y la actualización en cascada en caso de modificación.

### 4.4. Regla de integridad de dominio

La regla de integridad de dominio está relacionada, como su nombre indica, con la noción de *dominio*. Esta regla establece dos condiciones.

La **primera condición** consiste en que un valor no nulo de un atributo $A_i$ debe pertenecer al dominio del atributo $A_i$; es decir, debe pertenecer a dominio($A_i$).

Esta condición implica que todos los valores no nulos que contiene la base de datos para un determinado atributo deben ser del dominio declarado para dicho atributo.

**Ejemplo**

Si en la relación EMPLEADOS(*DNI, nombre, apellido, edademp*) hemos declarado que dominio(*DNI*) es el dominio predefinido de los enteros, entonces no podremos insertar, por ejemplo, ningún empleado que tenga por *DNI* el valor "Luis", que no es un entero.

Recordad que los conceptos de *dominio predefinido* y *dominio definido por el usuario* se han explicado en el subapartado 2.2 de esta unidad didáctica.

Recordemos que los dominios pueden ser de dos tipos: predefinidos o definidos por el usuario. Observad que los dominios definidos por el usuario resultan muy útiles, porque nos permiten determinar de forma más específica cuáles serán los valores admitidos por los atributos.

**Ejemplo**

Supongamos ahora que en la relación EMPLEADOS(*DNI, nombre, apellido, edademp*) hemos declarado que dominio(*edademp*) es el dominio definido por el usuario *edad*. Supongamos también que el dominio *edad* se ha definido como el conjunto de los enteros que están entre 16 y 65. En este caso, por ejemplo, no será posible insertar un empleado con un valor de 90 para *edademp*.

La segunda condición de la regla de integridad de dominio es más compleja, especialmente en el caso de dominios definidos por el usuario; los SGBD actuales no la soportan para estos últimos dominios. Por estos motivos sólo la presentaremos superficialmente.

**Lectura complementaria**

Para estudiar con más detalle la segunda condición de la regla de integridad de dominio, podéis consultar la siguiente obra:

**C.J. Date** (2001). *Introducción a los sistemas de bases de datos* (7ª ed., cap. 19). Prentice Hall.

Esta **segunda condición** sirve para establecer que los operadores que pueden aplicarse sobre los valores dependen de los dominios de estos valores; es decir, un operador determinado sólo se puede aplicar sobre valores que tengan dominios que le sean adecuados.

**Ejemplo**

Analizaremos esta segunda condición de la regla de integridad de dominio con un ejemplo concreto. Si en la relación EMPLEADOS(*DNI, nombre, apellido, edademp*) se ha declarado que dominio(*DNI*) es el dominio predefinido de los enteros, entonces no se permitirá consultar todos aquellos empleados cuyo DNI sea igual a 'Elena' (*DNI* = 'Elena'). El motivo es que no tiene sentido que el operador de comparación = se aplique entre un *DNI* que tiene por dominio los enteros, y el valor 'Elena', que es una serie de caracteres.

De este modo, el hecho de que los operadores que se pueden aplicar sobre los valores dependan del dominio de estos valores permite detectar errores que se podrían cometer cuando se consulta o se actualiza la base de datos. Los dominios definidos por el usuario son muy útiles, porque nos permitirán determinar de forma más específica cuáles serán los operadores que se podrán aplicar sobre los valores.

**Ejemplo**

Veamos otro ejemplo con dominios definidos por el usuario. Supongamos que en la conocida relación EMPLEADOS(*DNI, nombre, apellido, edademp*) se ha declarado que dominio(*DNI*) es el dominio definido por el usuario *númerosDNI* y que dominio(*edademp*) es el dominio definido por el usuario *edad*. Supongamos que *númerosDNI* corresponde a los enteros positivos y que *edad* corresponde a los enteros que están entre 16 y 65. En este caso, será incorrecto, por ejemplo, consultar los empleados que tienen el valor de *DNI* igual al valor de *edademp*. El motivo es que, aunque tanto los valores de *DNI* como los de *edademp* sean enteros, sus dominios son diferentes; por ello, según el significado que el usuario les da, no tiene sentido compararlos.

Sin embargo, los actuales SGBD relacionales no dan apoyo a la segunda condición de la regla de integridad de dominio para dominios definidos por el usuario. Si se quisiera hacer, sería necesario que el diseñador tuviese alguna forma de especificar, para cada operador que se desease utilizar, para qué combinaciones de dominios definidos por el usuario tiene sentido que se aplique. El lenguaje estándar SQL no incluye actualmente esta posibilidad. 

# 5. El álgebra relacional

Como ya hemos comentado en el apartado dedicado a las operaciones del modelo relacional, el álgebra relacional se inspira en la teoría de conjuntos para especificar consultas en una base de datos relacional.

Consultad el apartado 3 de esta unidad didáctica.

Para **especificar una consulta** en álgebra relacional, es preciso definir uno o más pasos que sirven para ir construyendo, mediante operaciones de álgebra relacional, una nueva relación que contenga los datos que responden a la consulta a partir de las relaciones almacenadas. Los lenguajes basados en el álgebra relacional son procedimentales, dado que los pasos que forman la consulta describen un procedimiento.

La visión que presentaremos es la de un lenguaje teórico y, por lo tanto, incluiremos sólo sus operaciones fundamentales, y no las construcciones que se podrían añadir a un lenguaje comercial para facilitar cuestiones como por ejemplo el orden de presentación del resultado, el cálculo de datos agregados, etc.

Una característica destacable de todas las operaciones del álgebra relacional es que tanto los operandos como el resultado son relaciones. Esta propiedad se denomina **cierre relacional**.

**Implicaciones del cierre relacional**

El hecho de que el resultado de una operación del álgebra relacional sea una nueva relación tiene implicaciones importantes:

**1.** El resultado de una operación puede actuar como operando de otra operación.

**2.** El resultado de una operación cumplirá todas las características que ya conocemos de las relaciones: no-ordenación de las tuplas, ausencia de tuplas repetidas, etc.

Las operaciones del álgebra relacional han sido clasificadas según distintos criterios; de todos ellos indicamos los tres siguientes:

**1)** Según se pueden expresar o no en términos de otras operaciones.

**a) Operaciones primitivas:** son aquellas operaciones a partir de las cuales podemos definir el resto. Estas operaciones son la unión, la diferencia, el producto cartesiano, la selección y la proyección.

**b) Operaciones no primitivas:** el resto de las operaciones del álgebra relacional que no son estrictamente necesarias, porque se pueden expresar en términos de las primitivas; sin embargo, las operaciones no primitivas permiten formular algunas consultas de forma más cómoda. Existen distintas versiones del álgebra relacional, según las operaciones no primitivas que se incluyen. Nosotros estudiaremos las operaciones no primitivas que se utilizan con mayor frecuencia: la intersección y la combinación.

**2)** Según el número de relaciones que tienen como operandos:

**a) Operaciones binarias:** son las que tienen dos relaciones como operandos. Son binarias todas las operaciones, excepto la selección y la proyección.

**b) Operaciones unarias:** son las que tienen una sola relación como operando. La selección y la proyección son unarias.

**3)** Según se parecen o no a las operaciones de la teoría de conjuntos:

**a) Operaciones conjuntistas:** son las que se parecen a las de la teoría de conjuntos. Se trata de la unión, la intersección, la diferencia y el producto cartesiano.

**b) Operaciones específicamente relacionales:** son el resto de las operaciones; es decir, la selección, la proyección y la combinación.

Las operaciones del álgebra relacional clasificadas según sean conjuntistas o específicamente relacionales se estudian en los subapartados 5.1 y 5.2 de esta unidad.

Como ya hemos comentado anteriormente, las operaciones del álgebra relacional obtienen como resultado una nueva relación. Es decir que si hacemos una operación del álgebra como por ejemplo *EMPLEADOS_ADM* $\cup \cup$ *EMPLEADOS_PROD* para obtener la unión de las relaciones *EMPLEADOS_ADM* y *EMPLEADOS_PROD*, el resultado de la operación es una nueva relación que tiene la unión de las tuplas de las relaciones de partida.

Esta nueva relación debe tener un nombre. En principio, consideramos que su nombre es la misma expresión del álgebra relacional que la obtiene; es decir, la misma expresión *EMPLEADOS_ADM* $\cup$ *EMPLEADOS_PROD*. Puesto que este nombre es largo, en ocasiones puede ser interesante cambiarlo por uno más simple. Esto nos facilitará las referencias a la nueva relación, y será especialmente útil en los casos en los que queramos utilizarla como operando de otra operación. Usaremos la operación auxiliar *redenominar* con este objetivo.

La **operación *redenominar***, que denotaremos con el símbolo :=, permite asignar un nombre *R* a la relación que resulta de una operación del álgebra relacional; lo hace de la forma siguiente:

$$R := E,$$

siendo *E* la expresión de una operación del álgebra relacional.

En el ejemplo, para dar el nombre *EMPLEADOS* a la relación resultante de la operación *EMPLEADOS_ADM* $\cup$ *EMPLEADOS_PROD*, haríamos:

$$EMPLEADOS := EMPLEADOS\_ADM \cup EMPLEADOS\_PROD.$$

Cada operación del álgebra relacional da unos nombres por defecto a los atributos del esquema de la relación resultante, tal y como veremos más adelante.

En algunos casos, puede ser necesario cambiar estos nombres por defecto por otros nombres. Por este motivo, también permitiremos cambiar el nombre de la relación y de sus atributos mediante la operación *redenominar*.

Utilizaremos también la operación *redenominar* para cambiar el esquema de una relación. Si una relación tiene el esquema $S(B_1, B_2, ..., B_n)$ y queremos cambiarlo por $R(A_1, A_2, ..., A_n)$, lo haremos de la siguiente forma:

$$R(A_1, A_2, ..., A_n) := S(B_1, B_2, ..., B_n).$$

A continuación presentaremos un ejemplo que utilizaremos para ilustrar las operaciones del álgebra relacional. Después veremos con detalle las operaciones.

Supongamos que tenemos una base de datos relacional con las cuatro relaciones siguientes:

**1)** La relación *EDIFICIOS_EMP*, que contiene datos de distintos edificios de los que una empresa dispone para desarrollar sus actividades.

**2)** La relación *DESPACHOS*, que contiene datos de cada uno de los despachos que hay en los edificios anteriores.

**3)** La relación *EMPLEADOS_ADM*, que contiene los datos de los empleados de la empresa que llevan a cabo tareas administrativas.

**4)** La relación *EMPLEADOS_PROD*, que almacena los datos de los empleados de la empresa que se ocupan de tareas de producción.

A continuación describimos los esquemas de las relaciones anteriores y sus extensiones en un momento determinado:

- Esquema y extensión de *EDIFICIOS_EMP*:

| **EDIFICIOS_EMP** | |
|---|---|
| ***edificio*** | ***supmediadesp*** |
| Marina | 15 |
| Diagonal | 10 |

- Esquema y extensión de *DESPACHOS*:

| **DESPACHOS** | | |
|---|---|---|
| ***edificio*** | ***número*** | ***superficie*** |
| Marina | 120 | 10 |
| Marina | 230 | 20 |
| Diagonal | 120 | 10 |
| Diagonal | 440 | 10 |

- Esquema y extensión de *EMPLEADOS_ADM*:

| **EMPLEADOS_ADM** | | | | |
|---|---|---|---|---|
| ***DNI*** | ***nombre*** | ***apellido*** | ***edificiodesp*** | ***númerodesp*** |
| 40.444.255 | Juan | García | Marina | 120 |
| 33.567.711 | Marta | Roca | Marina | 120 |

- Esquema y extensión de *EMPLEADOS_PROD*:

| **EMPLEADOS_PROD** | | | | |
|---|---|---|---|---|
| ***DNI*** | ***nombreemp*** | ***apellidoemp*** | ***edificiodesp*** | ***númerodesp*** |
| 33.567.711 | Marta | Roca | Marina | 120 |
| 55.898.425 | Carlos | Buendía | Diagonal | 120 |
| 77.232.144 | Elena | Pla | Marina | 230 |
| 21.335.245 | Jorge | Soler | NULO | NULO |
| 88.999.210 | Pedro | González | NULO | NULO |

Se considera que los valores nulos de los atributos *edificiodesp* y *númerodesp* de las relaciones *EMPLEADOS_PROD* y *EMPLEADOS_ADM* indican que el empleado correspondiente no tiene despacho.

## 5.1. Operaciones conjuntistas

Las operaciones conjuntistas del álgebra relacional son la **unión**, la **intersección**, la **diferencia** y el **producto cartesiano**.

### 5.1.1. Unión

La unión es una operación que, a partir de dos relaciones, obtiene una nueva relación formada por todas las tuplas que están en alguna de las relaciones de partida.

La unión es una operación binaria, y la unión de dos relaciones $T$ y $S$ se indica $T \cup S$.

La unión de las relaciones *EMPLEADOS_ADM* y *EMPLEADOS_PROD* proporciona una nueva relación que contiene tanto a los empleados de administración como los empleados de producción; se indicaría así: *EMPLEADOS_ADM* $\cup$ *EMPLEADOS_PROD*.

Sólo tiene sentido aplicar la unión a relaciones que tengan tuplas similares.

Por ejemplo, se puede hacer la unión de las relaciones EMPLEADOS_ADM y EMPLEADOS_PROD porque sus tuplas se parecen. En cambio, no se podrá hacer la unión de las relaciones EMPLEADOS_ADM y DESPACHOS porque, como habéis podido observar en las tablas, las tuplas respectivas son de tipo diferente.

Más concretamente, para poder aplicar la unión a dos relaciones, es preciso que las dos relaciones sean compatibles. Decimos que dos relaciones $T$ y $S$ son **relaciones compatibles** si:

- Tienen el mismo grado.
- Se puede establecer una biyección entre los atributos de $T$ y los atributos de $S$ que hace corresponder a cada atributo $A_i$ de $T$ un atributo $A_j$ de $S$, de modo que se cumple que dominio($A_i$) = dominio($A_j$).

**Ejemplo de relaciones compatibles**

Las relaciones *EMPLEADOS_ADM* y *EMPLEADOS_PROD* tienen grado 5. Podemos establecer la siguiente biyección entre sus atributos:

- A *DNI* de *EMPLEADOS_ADM* le corresponde *DNIemp* de *EMPLEADOS_PROD*.
- A *nombre* de *EMPLEADOS_ADM* le corresponde *nombreemp* de *EMPLEADOS_PROD*.
- A *apellido* de *EMPLEADOS_ADM* le corresponde *apellidoemp* de *EMPLEADOS_PROD*.
- A *edificiodesp* de *EMPLEADOS_ADM* le corresponde *edificiodesp* de *EMPLEADOS_PROD*.
- A *númerodesp* de *EMPLEADOS_ADM* le corresponde *edificiodesp* de *EMPLEADOS_PROD*.

Además, supondremos que los dominios de sus atributos se han declarado de forma que se cumple que el dominio de cada atributo de *EMPLEADOS_ADM* sea el mismo que el dominio de su atributo correspondiente en *EMPLEADOS_PROD*.

Por todos estos factores, podemos llegar a la conclusión de que *EMPLEADOS_ADM* y *EMPLEADOS_PROD* son relaciones compatibles.

A continuación, pasaremos a definir los atributos y la extensión de la relación resultante de una unión.

Los **atributos del esquema de la relación resultante de $T \cup S$** coinciden con los atributos del esquema de la relación $T$.

La **extensión de la relación resultante de $T \cup S$** es el conjunto de tuplas que pertenecen a la extensión de $T$, a la extensión de $S$ o a la extensión de ambas relaciones.

**No-repetición de tuplas**

Notad que en caso de que una misma tupla esté en las dos relaciones que se unen, el resultado de la unión no la tendrá repetida. El resultado de la unión es una nueva relación por lo que no puede tener repeticiones de tuplas.

**Ejemplo de unión**

Si queremos obtener una relación $R$ que tenga a todos los empleados de la empresa del ejemplo anterior, llevaremos a cabo la unión de las relaciones *EMPLEADOS_ADM* y *EMPLEADOS_PROD* de la forma siguiente:

$$R := EMPLEADOS\_ADM \cup EMPLEADOS\_PROD.$$

Entonces la relación $R$ resultante será la reflejada en la tabla siguiente:

| **R** | | | | |
|---|---|---|---|---|
| ***DNI*** | ***nombre*** | ***apellido*** | ***edificiodesp*** | ***númerodesp*** |
| 40.444.255 | Juan | García | Marina | 120 |
| 33.567.711 | Marta | Roca | Marina | 120 |
| 55.898.425 | Carlos | Buendía | Diagonal | 120 |

| R | | | | |
|---|---|---|---|---|
| ***DNI*** | ***nombre*** | ***apellido*** | ***edificiodesp*** | ***númerodesp*** |
| 77.232.144 | Elena | Pla | Marina | 230 |
| 21.335.245 | Jorge | Soler | NULO | NULO |
| 88.999.210 | Pedro | González | NULO | NULO |

El hecho de que los atributos de la relación resultante coincidan con los atributos de la relación que figura en primer lugar en la unión es una convención; teóricamente, también habría sido posible convenir que coincidiesen con los de la relación que figura en segundo lugar.

### 5.1.2. Intersección

La intersección es una operación que, a partir de dos relaciones, obtiene una nueva relación formada por las tuplas que pertenecen a las dos relaciones de partida.

La intersección es una operación binaria; la intersección de dos relaciones $T$ y $S$ se indica $T \cap S$.

La intersección de las relaciones *EMPLEADOS_ADM* y *EMPLEADOS_PROD* obtiene una nueva relación que incluye a los empleados que son al mismo tiempo de administración y de producción: se indicaría como *EMPLEADOS_ADM* $\cap$ *EMPLEADOS_PROD*.

La intersección, como la unión, sólo se puede aplicar a relaciones que tengan tuplas similares. Para poder hacer la intersección de dos relaciones, es preciso, pues, que las relaciones sean compatibles.

A continuación definiremos los atributos y la extensión de la relación resultante de una intersección.

Los **atributos del esquema de la relación resultante de $T \cap S$** coinciden con los atributos del esquema de la relación $T$.

La **extensión de la relación resultante de $T \cap S$** es el conjunto de tuplas que pertenecen a la extensión de ambas relaciones.

**Ejemplo de intersección**

Si queremos obtener una relación $R$ que incluya a todos los empleados de la empresa del ejemplo que trabajan tanto en administración como en producción, realizaremos la intersección de las relaciones *EMPLEADOS_ADM* y *EMPLEADOS_PROD* de la forma siguiente:

$$R := EMPLEADOS\_ADM \cap EMPLEADOS\_PROD.$$

Entonces, la relación *R* resultante será:

| R | | | | |
|---|---|---|---|---|
| ***DNI*** | ***nombre*** | ***apellido*** | ***edificiodesp*** | ***númerodesp*** |
| 33.567.711 | Marta | Roca | Marina | 120 |

Observad que se ha tomado la convención de que los atributos de la relación que resulta coincidan con los atributos de la relación que figura en primer lugar.

### 5.1.3. Diferencia

La diferencia es una operación que, a partir de dos relaciones, obtiene una nueva relación formada por todas las tuplas que están en la primera relación y, en cambio, no están en la segunda. La diferencia es una operación binaria, y la diferencia entre las relaciones *T* y *S* se indica como *T* – *S*.

La diferencia *EMPLEADOS_ADM* menos *EMPLEADOS_PROD* da como resultado una nueva relación que contiene a los empleados de administración que no son empleados de producción, y se indicaría de este modo: *EMPLEADOS_ADM – EMPLEADOS_PROD*.

La diferencia, como ocurría en la unión y la intersección, sólo tiene sentido si se aplica a relaciones que tengan tuplas similares. Para poder realizar la diferencia de dos relaciones es necesario que las relaciones sean compatibles.

A continuación definimos los atributos y la extensión de la relación resultante de una diferencia.

Los **atributos del esquema de la relación resultante de *T – S*** coinciden con los atributos del esquema de la relación *T*.

La **extensión de la relación resultante de *T – S*** es el conjunto de tuplas que pertenecen a la extensión de *T*, pero no a la de *S*.

**Ejemplo de diferencia**

Si queremos obtener una relación *R* con todos los empleados de la empresa del ejemplo que trabajan en administración, pero no en producción, haremos la diferencia de las relaciones *EMPLEADOS_ADM* y *EMPLEADOS_PROD* de la forma siguiente:

*R* := *EMPLEADOS_ADM – EMPLEADOS_PROD*

Entonces la relación *R* resultante será:

| R | | | | |
|---|---|---|---|---|
| ***DNI*** | ***nombre*** | ***apellido*** | ***edificiodesp*** | ***númerodesp*** |
| 40.444.255 | Juan | García | Marina | 120 |

Se ha tomado la convención de que los atributos de la relación resultante coincidan con los atributos de la relación que figura en primer lugar.

### 5.1.4. Producto cartesiano

El producto cartesiano es una operación que, a partir de dos relaciones, obtiene una nueva relación formada por todas las tuplas que resultan de concatenar tuplas de la primera relación con tuplas de la segunda.

El producto cartesiano es una operación binaria. Siendo $T$ y $S$ dos relaciones que cumplen que sus esquemas no tienen ningún nombre de atributo común, el producto cartesiano de $T$ y $S$ se indica como $T \times S$.

Si calculamos el producto cartesiano de *EDIFICIOS_EMP* y *DESPACHOS*, obtendremos una nueva relación que contiene todas las concatenaciones posibles de tuplas de *EDIFICIOS_EMP* con tuplas de *DESPACHOS*.

Si se quiere calcular el producto cartesiano de dos relaciones que tienen algún nombre de atributo común, sólo hace falta redenominar previamente los atributos adecuados de una de las dos relaciones.

A continuación definimos los atributos y la extensión de la relación resultante de un producto cartesiano.

Los **atributos del esquema de la relación resultante de $T \times S$** son todos los atributos de $T$ y todos los atributos de $S$*.

*** Recordad que *T* y *S* no tienen ningún nombre de atributo común.**

La **extensión de la relación resultante de $T \times S$** es el conjunto de todas las tuplas de la forma $<v_1, v_2, ..., v_n, w_1, w_2, ..., w_m>$ para las que se cumple que $<v_1, v_2, ..., v_n>$ pertenece a la extensión de $T$ y que $<w_1, w_2, ..., w_m>$ pertenece a la extensión de $S$.

**Ejemplo de producto cartesiano**

El producto cartesiano de las relaciones *DESPACHOS* y *EDIFICIOS_EMP* del ejemplo se puede hacer como se indica (es necesario redenominar atributos previamente):

EDIFICIOS(*nombreedificio*, *supmediadesp*) := EDICIOS_EMP(*edificio*, *supmediadesp*).

$R := EDIFICIOS \times DESPACHOS$.

Entonces, la relación *R* resultante será:

| R | | | | |
|---|---|---|---|---|
| ***nombreedificio*** | ***supmediadesp*** | ***edificio*** | ***número*** | ***superficie*** |
| Marina | 15 | Marina | 120 | 10 |
| Marina | 15 | Marina | 230 | 20 |
| Marina | 15 | Diagonal | 120 | 10 |
| Marina | 15 | Diagonal | 440 | 10 |
| Diagonal | 10 | Marina | 120 | 10 |

| R | | | | |
|---|---|---|---|---|
| *nombreedificio* | *supmediadesp* | *edificio* | *número* | *superficie* |
| Diagonal | 10 | Marina | 230 | 20 |
| Diagonal | 10 | Diagonal | 120 | 10 |
| Diagonal | 10 | Diagonal | 440 | 10 |

Conviene señalar que el producto cartesiano es una operación que raramente se utiliza de forma explícita, porque el resultado que da no suele ser útil para resolver las consultas habituales.

A pesar de ello, el producto cartesiano se incluye en el álgebra relacional porque es una operación primitiva; a partir de la cual se define otra operación del álgebra, la combinación, que se utiliza con mucha frecuencia.

### 5.2. Operaciones específicamente relacionales

Las operaciones específicamente relacionales son la **selección**, la **proyección** y la **combinación**.

#### 5.2.1. Selección

Podemos ver la selección como una operación que sirve para elegir algunas tuplas de una relación y eliminar el resto. Más concretamente, la selección es una operación que, a partir de una relación, obtiene una nueva relación formada por todas las tuplas de la relación de partida que cumplen una condición de selección especificada.

La selección es una operación unaria. Siendo $C$ una condición de selección, la selección de $T$ con la condición $C$ se indica como T($C$).

Para obtener una relación que tenga todos los despachos del edificio Marina que tienen más de 12 metros cuadrados, podemos aplicar una selección a la relación *DESPACHOS* con una condición de selección que sea *edificio* = Marina y *superficie* > 12; se indicaría DESPACHOS(*edificio* = Marina y *superficie* > 12).

En general, la condición de selección C está formada por una o más cláusulas de la forma:

$$A_i \, \theta \, v,$$

o bien:

$$A_i \, \theta \, A_j,$$

donde $A_i$ y $A_j$ son atributos de la relación $T$, θ es un operador de comparación* y $v$ es un valor. Además, se cumple que:

*** Es decir,** =, ≠, <, ≤, >, **o** ≥.

- En las cláusulas de la forma $A_i$ θ $v$, $v$ es un valor del dominio de $A_i$.
- En las cláusulas de la forma $A_i$, θ $A_j$, $A_i$ y $A_j$ tienen el mismo dominio.

Las cláusulas que forman una condición de selección se conectan con los siguientes operadores booleanos: "y" (∧) y "o" (∨).

A continuación definimos los atributos y la extensión de la relación resultante de una selección.

Los **atributos del esquema de la relación resultante de T(*C*)** coinciden con los atributos del esquema de la relación *T*.

La **extensión de la relación resultante de T(*C*)** es el conjunto de tuplas que pertenecen a la extensión de *T* y que satisfacen la condición de selección *C*. Una tupla *t* satisface una condición de selección *C* si, después de sustituir cada atributo que hay en C por su valor en *t*, la condición *C* se evalúa en el valor cierto.

**Ejemplo de selección**

Si queremos obtener una relación *R* con los despachos de la base de datos del ejemplo que están en el edificio Marina y que tienen una superficie de más de 12 metros cuadrados, haremos la siguiente selección:

*R* := *DESPACHOS(edificio* = Marina *y superficie* > 12*)*.

La relación *R* resultante será:

| **R** | | |
|---|---|---|
| ***edificio*** | ***número*** | ***superficie*** |
| Marina | 230 | 20 |

## 5.2.2. Proyección

Podemos considerar la proyección como una operación que sirve para elegir algunos atributos de una relación y eliminar el resto. Más concretamente, la proyección es una operación que, a partir de una relación, obtiene una nueva relación formada por todas las (sub)tuplas de la relación de partida que resultan de eliminar unos atributos especificados.

La proyección es una operación unaria. Siendo $\{A_i, A_j, ..., A_k\}$ un subconjunto de los atributos del esquema de la relación $T$, la proyección de $T$ sobre $\{A_i, A_j, ..., A_k\}$ se indica como $T[A_i, A_j, ..., A_k]$.

Para obtener una relación que tenga sólo los atributos *nombre* y *apellido* de los empleados de administración, podemos hacer una proyección en la relación *EMPLEADOS_ADM* sobre estos dos atributos. Se indicaría de la forma siguiente: *EMPLEADOS_ADM* [*nombre, apellido*].

A continuación definiremos los atributos y la extensión de la relación resultante de una proyección.

Los **atributos del esquema de la relación resultante de $T[A_i, A_j, ..., A_k]$** son los atributos $\{A_i, A_j, ..., A_k\}$.

La **extensión de la relación resultante de $T[A_i, A_j, ..., A_k]$** es el conjunto de todas las tuplas de la forma $<t.A_i, t.A_j, ..., t.A_k>$, donde se cumple que $t$ es una tupla de la extensión de $T$ y donde $t.A_p$ denota el valor para el atributo $A_p$ de la tupla $t$.

**Eliminación de las tuplas repetidas**

Notad que la proyección elimina implícitamente todas las tuplas repetidas. El resultado de una proyección es una relación válida y no puede tener repeticiones de tuplas.

**Ejemplo de proyección**

Si queremos obtener una relación $R$ con el nombre y el apellido de todos los empleados de administración de la base de datos del ejemplo, haremos la siguiente proyección:

$R$ := EMPLEADOS_ADM[*nombre, apellido*].

Entonces, la relación $R$ resultante será:

| R | |
|---|---|
| ***nombre*** | ***apellido*** |
| Juan | García |
| Marta | Roca |

### 5.2.3. Combinación

La combinación es una operación que, a partir de dos relaciones, obtiene una nueva relación formada por todas las tuplas que resultan de concadenar tuplas de la primera relación con tuplas de la segunda, y que cumplen una condición de combinación especificada.

La combinación es una operación binaria. Siendo $T$ y $S$ dos relaciones cuyos esquemas no tienen ningún nombre de atributo común, y siendo $B$ una condición de combinación, la combinación de $T$ y $S$ según la condición $B$ se indica $T[B]S$.

Para conseguir una relación que tenga los datos de cada uno de los empleados de administración junto con los datos de los despachos donde trabajan, podemos hacer una combinación de las relaciones *EMPLEADOS_ADM* y *DESPACHOS*, donde la condición de combinación indique lo siguiente: *edificiodesp* = *edificio* y *númerodesp* = *número*. La condición de combinación hace que el resultado sólo combine los datos de un empleado con los datos de un despacho si el *edificiodesp* y el *númerodesp* del empleado son iguales que el edificio y el número del despacho, respectivamente. Es decir, la condición hace que los datos de un empleado se combinen con los datos del despacho donde trabaja, pero no con datos de otros despachos.

La combinación del ejemplo anterior se indicaría de la forma siguiente:

*EMPLEADOS_ADM*[*edificiodesp* = *edificio*, *númerodesp* = *número*]*DESPACHOS*.

Si se quiere combinar dos relaciones que tienen algún nombre de atributo común, sólo hace falta redenominar previamente los atributos repetidos de una de las dos.

En general, la **condición B** de una combinación $T[B]S$ está formada por una o más comparaciones de la forma

$$A_i\, \theta\, A_j,$$

donde $A_i$ es un atributo de la relación T, $A_j$ es un atributo de la relación S, θ es un operador de comparación ( =, ≠, <, ≤, >, ≥), y se cumple que $A_i$ y $A_j$ tienen el mismo dominio. Las comparaciones de una condición de combinación se separan mediante comas.

A continuación definimos los atributos y la extensión de la relación resultante de una combinación.

Los **atributos del esquema de la relación resultante de *T*[*B*]*S*** son todos los atributos de *T* y todos los atributos de *S**.

*** Recordad que *T* y *S* no tienen ningún nombre de atributo común.**

La e**xtensión de la relación resultante de *T*[*B*]*S*** es el conjunto de tuplas que pertenecen a la extensión del producto cartesiano $T \times S$ y que satisfacen todas las comparaciones que forman la condición de combinación *B*. Una tupla *t* satisface una comparación si, después de sustituir cada atributo que figura en la comparación por su valor en *t*, la comparación se evalúa al valor cierto.

**Ejemplo de combinación**

Supongamos que se desea encontrar los datos de los despachos que tienen una superficie mayor o igual que la superficie media de los despachos del edificio donde están situados. La siguiente combinación nos proporcionará los datos de estos despachos junto con los datos de su edificio (observad que es preciso redenominar previamente los atributos):

EDIFICIOS(*nombreeedficio*,*supmediadesp*) := EDIFICIOS_EMP(e*dificio*, *supmediadesp*),

R := *EDIFICIOS*[*nombreedificio* = *edificio, supmediadesp* ≤ *superficie*] DESPACHOS.

Entonces, la relación *R* resultante será:

| R | | | | |
|---|---|---|---|---|
| ***nombreedificio*** | ***supmediadesp*** | ***edificio*** | ***número*** | ***superficie*** |
| Marina | 15 | Marina | 230 | 20 |
| Diagonal | 10 | Diagonal | 120 | 10 |
| Diagonal | 10 | Diagonal | 440 | 10 |

Supongamos ahora que para obtener los datos de cada uno de los empleados de administración, junto con los datos del despacho donde trabajan, utilizamos la siguiente combinación:

*R* := *EMPLEADOS_ADM[edificiodesp = edificio, númerodesp = número]DESPACHOS.*

La relación *R* resultante será:

| R | | | | | | | |
|---|---|---|---|---|---|---|---|
| ***DNI*** | ***nombre*** | ***apellido*** | ***edificiodesp*** | ***númerodesp*** | ***edificio*** | ***número*** | ***superficie*** |
| 40.444.255 | Juan | García | Marina | 120 | Marina | 120 | 10 |
| 33.567.711 | Marta | Roca | Marina | 120 | Marina | 120 | 10 |

La relación *R* combina los datos de cada empleado con los datos de su despacho.

En ocasiones, la combinación recibe el nombre de **θ-combinación**, y cuando todas las comparaciones de la condición de la combinación tienen el operador "=", se denomina **equicombinación**.

Según esto, la combinación del último ejemplo es una equicombinación.

Observad que el resultado de una equicombinación siempre incluye una o más parejas de atributos que tienen valores idénticos en todas las tuplas.

En el ejemplo anterior, los valores de *edificiodesp* coinciden con los de *edificio*, y los valores de *númerodesp* coinciden con los de *número*.

Puesto que uno de cada par de atributos es superfluo, se ha establecido una variante de combinación denominada *combinación natural*, con el fin de eliminarlos.

La **combinación natural** de dos relaciones *T* y *S* se denota como $T * S$ y consiste básicamente en una equicombinación seguida de la eliminación de los atributos superfluos; además, se considera por defecto que la condición de combinación iguala todas las parejas de atributos que tienen el mismo nombre en *T* y en *S*.

Observad que, a diferencia de la equicombinación, la combinación natural se aplica a relaciones que tienen nombres de atributos comunes.

**Ejemplo de combinación natural**

Si hacemos:

*R* := *EDIFICIOS_EMP* * *DESPACHOS,*

se considera que la condición es *edificio = edificio* porque *edificio* es el único nombre de atributo que figura tanto en el esquema de *EDIFICIOS_EMP* como en el esquema de *DESPACHOS*. El resultado de esta combinación natural es:

| R | | | |
|---|---|---|---|
| ***edificio*** | ***supmediadesp*** | ***número*** | ***superficie*** |
| Marina | 15 | 120 | 10 |
| Marina | 15 | 230 | 20 |
| Diagonal | 10 | 120 | 10 |
| Diagonal | 10 | 440 | 10 |

Notad que se ha eliminado uno de los atributos de nombre *edificio*.

En ocasiones, antes de la combinación natural es necesario aplicar la operación *redenominar* para hacer coincidir los nombres de los atributos que nos interesa igualar.

**Ejemplo de combinación natural con redenominación**

Por ejemplo, si queremos obtener los datos de cada uno de los empleados de administración junto con los datos del despacho donde trabajan pero sin repetir valores de atributos superfluos, haremos la siguiente combinación natural, que requiere una redenominación previa:

D(*edificiodesp, númerodesp, superficie*) := DESPACHOS(*edificio, número, superficie*),

$$R := EMPLEADOS\_ADM * D.$$

Entonces, la relación *R* resultante será:

| R | | | | | |
|---|---|---|---|---|---|
| ***DNI*** | ***nombre*** | ***apellido*** | ***edificiodesp*** | ***númerodesp*** | ***superficie*** |
| 40.444.255 | Juan | García | Marina | 120 | 10 |
| 33.567.711 | Marta | Roca | Marina | 120 | 10 |

## 5.3. Secuencias de operaciones del álgebra relacional

En muchos casos, para formular una consulta en álgebra relacional es preciso utilizar varias operaciones, que se aplican en un cierto orden. Para hacerlo, hay dos posibilidades:

**1)** Utilizar una sola expresión del álgebra que incluya todas las operaciones con los paréntesis necesarios para indicar el orden de aplicación.

**2)** Descomponer la expresión en varios pasos donde cada paso aplique una sola operación y obtenga una relación intermedia que se pueda utilizar en los pasos subsiguientes.

**Ejemplo de utilización de secuencias de operaciones**

Para obtener el nombre y el apellido de los empleados, tanto de administración como de producción, es necesario hacer una unión de *EMPLEADOS_ADM* y *EMPLEADOS_PROD*, y después hacer una proyección sobre los atributos *nombre* y *apellido*. La operación se puede expresar de las formas siguientes:

**a)** Se puede utilizar una sola expresión:

$$R := (EMPLEADOS\_ADM \cup EMPLEADOS\_PROD)\ [nombre, apellido].$$

**b)** O bien podemos expresarlo en dos pasos:

- *EMPS := EMPLEADOS_ADM ∪ EMPLEADOS_PROD;*
- *R* := EMPS[*nombre, apellido*]

En los casos en que una consulta requiere efectuar muchas operaciones, resulta más sencilla la segunda alternativa, porque evita expresiones complejas.

**Otros ejemplos de consultas formuladas con secuencias de operaciones**

Veamos algunos ejemplos de consultas en la base de datos formuladas con secuencias de operaciones del álgebra relacional.

Recordad que la base de datos que se utiliza en los ejemplos se ha descrito en la introducción del apartado 5 de esta unidad didáctica.

**1)** Para obtener el nombre del edificio y el número de los despachos situados en edificios en los que la superficie media de estos despachos es mayor que 12, podemos utilizar la siguiente secuencia de operaciones:

- *A* := EDIFICIOS_EMP(*supmediadesp* > 12);
- *B* := *DESPACHOS * A*;
- *R* := *B*[*edificio, número*]

**2)** Supongamos ahora que se desea obtener el nombre y el apellido de todos los empleados (tanto de administración como de producción) que están asignados al despacho 120 del edificio Marina. En este caso, podemos utilizar la siguiente secuencia:

- *A* := *EMPLEADOS_ADM ∪ EMPLEADOS_PROD*;
- *B* := A(*edificiodesp* = Marina y *númerodesp* = 120);
- *R* := B[*nombre, apellido*].

**3)** Si queremos consultar el nombre del edificio y el número de los despachos que ningún empleado de administración tiene asignado, podemos utilizar esta secuencia:

- *A* := DESPACHOS [*edificio, número*];
- *B* := EMPLEADOS_ADM[*edificiodesp, númerodesp*];
- *R* := *A* – *B*.

**4)** Para obtener el DNI, el nombre y el apellido de todos los empleados de administración que tienen despacho, junto con la superficie de su despacho, podemos hacer lo siguiente:

- A[*DNI, nombre, apellido, edificio, número*] := EMPLEADOS_ADM[*DNI, nombre, apellido, edificiodesp, númerodesp*];
- *B* := *A * DESPACHOS*;
- *R* := *B[DNI, nombre, apellido, superficie]*.

## 5.4. Extensiones: combinaciones externas

Para finalizar el tema del álgebra relacional, analizaremos algunas extensiones útiles de la combinación.

Las combinaciones que se han descrito obtienen las tuplas del producto cartesiano de dos relaciones que satisfacen una condición de combinación. Las tuplas de

una de las dos relaciones que no tienen en la otra relación una tupla como mínimo con la cual, una vez concatenadas, satisfagan la condición de combinación, no aparecen en el resultado de la combinación, y podríamos decir que sus datos se pierden.

Las combinaciones se han explicado en el subapartado 5.3.3 de esta unidad didáctica.

Por ejemplo, si hacemos la siguiente combinación natural (con una redenominación previa):

D(*edificiodesp, númerodesp, superficie*) := DESPACHOS (*edificio, número, superficie*),

R := *EMPLEADOS_PROD* * *D*.

Puesto que se trata de una combinación natural, se considera que la condición de combinación es *edificio* = *edificio* y *número* = *número*, y la relación *R* resultante será:

**R**

| *DNIemp* | *nombreemp* | *apellidoemp* | *edificiodesp* | *númerodesp* | *superficie* |
|---|---|---|---|---|---|
| 33.567.711 | Marta | Roca | Marina | 120 | 10 |
| 55.898.425 | Carlos | Buendía | Diagonal | 120 | 10 |
| 77.232.144 | Elena | Pla | Marina | 230 | 20 |

Notad que en esta relación R no están los empleados de producción que no tienen despacho asignado (con valores nulos en *edificiodesp* y *númerodesp*), y tampoco los despachos que no tienen ningún empleado de producción, porque no cumplen la condición de combinación.

Conviene destacar que las tuplas que tienen un valor nulo para alguno de los atributos que figuran en la condición de combinación se pierden siempre, porque en estos casos la condición de combinación siempre se evalúa a falso.

En algunos casos, puede interesar hacer combinaciones de los datos de dos relaciones sin que haya pérdida de datos de las relaciones de partida. Entonces, se utilizan las combinaciones externas.

Las **combinaciones externas** entre dos relaciones *T* y *S* consisten en variantes de combinación que conservan en el resultado todas las tuplas de *T*, de *S* o de ambas relaciones. Pueden ser de los tipos siguientes:

**1)** La **combinación externa izquierda** entre dos relaciones *T* y *S*, que denotamos como $T[C]_{\mathrm{I}}S$, conserva en el resultado todas las tuplas de la relación *T*.

**2)** La **combinación externa derecha** entre dos relaciones *T* y *S*, que denotamos como $T[C]_{\mathrm{D}}S$, conserva en el resultado todas las tuplas de la relación *S*.

**3)** Finalmente, la **combinación externa plena** entre dos relaciones *T* y *S*, que denotamos como $T[C]_{\mathrm{p}}S$, conserva en el resultado todas las tuplas de *T* y todas las tuplas de *S*.

Estas extensiones también se aplican al caso de la combinación natural entre dos relaciones, $T * S$, concretamente:

**a)** La combinación natural externa izquierda entre dos relaciones $T$ y $S$, que se indica como $T *_{\mathrm{I}} S$, conserva en el resultado todas las tuplas de la relación $T$.

**b)** La combinación natural externa derecha entre dos relaciones $T$ y $S$, que se indica como $T *_{\mathrm{D}} S$, conserva en el resultado todas las tuplas de la relación $S$.

**c)** Finalmente, la combinación natural externa plena entre dos relaciones $T$ y $S$, que se indica como $T *_{\mathrm{P}} S$, conserva en el resultado todas las tuplas de $T$ y todas las tuplas de $S$.

Las tuplas de una relación $T$ que se conservan en el resultado $R$ de una combinación externa con otra relación $S$, a pesar de que no satisfacen la condición de combinación, tienen valores nulos en el resultado $R$ para todos los atributos que provienen de la relación $S$.

**Ejemplos de combinaciones naturales externas**

**1)** Si hacemos la siguiente combinación natural derecha (con una redenominación previa):

D(*edificiodesp, númerodesp, superficie*) := DESPACHOS (*edificio, número, superficie*),

$$R := EMPLADOS\_PROD *_{\mathrm{D}} D,$$

la relación *R* resultante será:

| R | | | | | |
|---|---|---|---|---|---|
| ***DNIemp*** | ***nombreemp*** | ***apellidoemp*** | ***edificiodesp*** | ***númerodesp*** | ***superficie*** |
| 33.567.711 | Marta | Roca | Marina | 120 | 10 |
| 55.898.425 | Carlos | Buendía | Diagonal | 120 | 10 |
| 77.232.144 | Elena | Pla | Marina | 230 | 20 |
| NULO | NULO | NULO | Diagonal | 440 | 10 |

Ahora obtenemos todos los despachos en la relación resultante, tanto si tienen un empleado de producción asignado como si no. Notad que los atributos *DNI*, *nombre* y *apellido* para los despachos que no tienen empleado reciben valores nulos.

**2)** Si hacemos la siguiente combinación natural izquierda (con una redenominación previa):

D(*edificiodesp, númerodesp, superficie*) := DESPACHOS(*edificio, número, superficie*),

$$R := EMPLEADOS\_PROD *_{I} D,$$

entonces la relación *R* resultante será:

| R | | | | | |
|---|---|---|---|---|---|
| ***DNIemp*** | ***nombreemp*** | ***apellidoemp*** | ***edificiodesp*** | ***númerodesp*** | ***superficie*** |
| 33.567.711 | Marta | Roca | Marina | 120 | 10 |
| 55.898.425 | Carlos | Buendía | Diagonal | 120 | 10 |
| 77.232.144 | Elena | Pla | Marina | 230 | 20 |
| 21.335.245 | Jorge | Soler | NULO | NULO | NULO |
| 88.999.210 | Pedro | González | NULO | NULO | NULO |

Esta combinación externa nos permite obtener en la relación resultante a todos los empleados de producción, tanto si tienen despacho como si no. Observad que el atributo superficie para los empleados que no tienen despacho contiene un valor nulo.

**3)** Finalmente, si hacemos la siguiente combinación natural plena (con una redenominación previa):

$$D(edificiodesp, númerodesp, superficie) := DESPACHOS(edificio, número, superficie),$$
$$R := EMPLEADOS\_PROD *_P D,$$

entonces la relación *R* resultante será:

| **R** | | | | | |
|---|---|---|---|---|---|
| ***DNIemp*** | ***nombreemp*** | ***apellidoemp*** | ***edificiodesp*** | ***númerodesp*** | ***superficie*** |
| 33.567.711 | Marta | Roca | Marina | 120 | 10 |
| 55.898.425 | Carlos | Buendía | Diagonal | 120 | 10 |
| 77.232.144 | Elena | Pla | Marina | 230 | 20 |
| 21.335.245 | Jorge | Soler | NULO | NULO | NULO |
| 88.999.210 | Pedro | González | NULO | NULO | NULO |
| NULO | NULO | NULO | Diagonal | 440 | 10 |

En este caso, en la relación resultante obtenemos a todos los empleados de producción y también todos los despachos.

## Resumen

En esta unidad didáctica hemos presentado los **conceptos fundamentales del modelo relacional de datos** y, a continuación, hemos explicado las **operaciones del álgebra relacional**:

**1)** Los aspectos más relevantes del modelo relacional que hemos descrito son los siguientes:

**a)** En lo que respecta a la **estructura de los datos**:

- Consiste en un conjunto de relaciones.
- Una relación permite almacenar datos relacionados entre sí.
- La clave primaria de una relación permite identificar sus datos.
- Las claves foráneas de las relaciones permiten referenciar claves primarias y, de este modo, establecer conexiones entre los datos de las relaciones.

**b)** En lo que respecta a la **integridad de los datos**:

- La regla de integridad de unicidad y de entidad de la clave primaria: las claves primarias no pueden contener valores repetidos ni valores nulos.
- La regla de integridad referencial: los valores de las claves foráneas deben existir en la clave primaria referenciada o bien deben ser valores nulos.
- La regla de integridad de dominio: los valores no nulos de un atributo deben pertenecer al dominio del atributo, y los operadores que es posible aplicar sobre los valores dependen de los dominios de estos valores.

**2)** El álgebra relacional proporciona un conjunto de operaciones para manipular relaciones. Estas operaciones se pueden clasificar de la forma siguiente:

**a)** Operaciones conjuntistas: unión, intersección, diferencia y producto cartesiano.

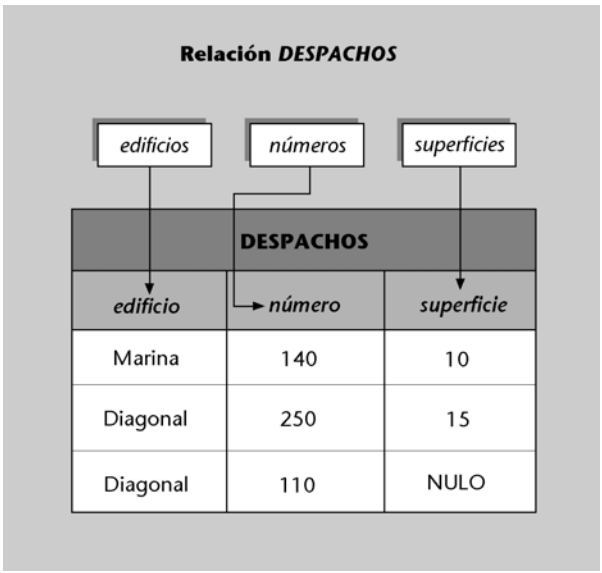
**b)** Operaciones específicamente relacionales: selección, proyección y combinación.

Las operaciones del álgebra relacional pueden formar secuencias que permiten resolver consultas complejas.

# Ejercicios de autoevaluación

**1.** Dada la relación que corresponde a la siguiente representación tabular:

Figura 4




**a)** Indicad qué conjunto de atributos tiene.
**b)** Decid qué dominio tiene cada uno de sus atributos.
**c)** Escribid todas las distintas formas de denotar su esquema de relación.
**d)** Elegid una de las formas de denotar su esquema de relación y utilizadla para dibujar el conjunto de tuplas correspondiente a su extensión.

**2.** Indicad cuáles son todas las superclaves de las siguientes relaciones:
**a)** *DESPACHOS*(*edificio, número, superficie*), que tiene como única clave candidata la siguiente: *edificio, número*.
**b)** *EMPLEADOS*(*DNI, NSS, nombre, apellido*), que tiene las siguientes claves candidatas: *DNI y NSS*.

**3.** Decid, para cada una de las siguientes operaciones de actualización, si se podría aceptar su aplicación sobre la base de datos que se ha utilizado en esta unidad:
**a)** Insertar en *EDIFICIOS_EMP* la tupla <Nexus, 30>.
**b)** Insertar en *DESPACHOS* la tupla <Diagonal, NULO, 15>.
**c)** Insertar en *EMPLEADOS_ADM* la tupla <55.555.555, María, Puig, Diagonal, 500>.
**d)** Modificar en *DESPACHOS* la tupla <Marina, 230, 20> por <Marina, 120, 20>.
**e)** Borrar en *EMPLEADOS_PROD* la tupla <88.999.20, Pedro, González, NULO, NULO>.
**f)** Modificar en *EMPLEADOS_ADM* la tupla <40.444.255, Juan, García, Marina, 120> por <33.567.711, Juan, García, Marina, 120>.
**g)** Borrar en *EDIFICIOS_EMP* la tupla <Marina, 15> si para la clave foránea edificio de *DESPACHOS* se ha seleccionado la política de restricción en caso de borrado.
**h)** Borrar en *EDIFICIOS_EMP* la tupla <Marina, 15> si para la clave foránea edificio de *DESPACHOS* se ha seleccionado la política de actualización en cascada en caso de borrado.

**4.** Escribid secuencias de operaciones del álgebra relacional que resuelvan las siguientes consultas en la base de datos que hemos utilizado en esta unidad:
**a)** Obtener los despachos con una superficie mayor que 15. Concretamente, se quiere saber el nombre del edificio, el número y la superficie de estos despachos, junto con la superficie media de los despachos del edificio donde están situados.
**b)** Obtener el nombre del edificio y el número de los despachos que no tienen asignado a ningún empleado (ni de producción ni de administración).
**c)** Obtener el nombre y el apellido de los empleados (tanto de administración como de producción), que no tienen despacho.
**d)** Obtener el nombre y el apellido de todos los empleados (tanto de administración como de producción) que tienen despacho asignado, junto con la superficie de su despacho y la superficie media de los despachos del edificio al que pertenece su despacho.
**e)** Obtener los despachos con una superficie mayor que la superficie del despacho Diagonal, 120. Concretamente, se quiere saber el nombre del edificio y el número de estos despachos.
**f)** Obtener todos los despachos de la empresa (tanto si tienen empleados como si no), junto con los empleados que tienen asignados (en caso de que los tengan). Concretamente, se quiere conocer el nombre del edificio, el número de despacho y el DNI del empleado.

**5.** Sea $R$ la relación que resulta de la intersección de las relaciones $T$ y $S$, es decir, $R := T \cap S$. Escribid una secuencia de operaciones del álgebra relacional que incluya sólo operaciones primitivas y que obtenga como resultado $R$.

**6.** Sean las relaciones de esquema T($A$, $B$, $C$) y S($D$, $E$, $F$), y sea $R$ la relación que resulta de la siguiente combinación:

$$R := T[B = D,\ C = E]S.$$

Escribid una secuencia de operaciones del álgebra relacional que incluya sólo operaciones primitivas y que obtenga como resultado $R$.

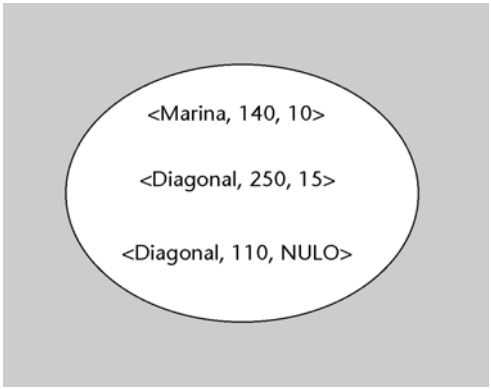
# Solucionario

## Ejercicios de autoevaluación

**1.**
**a)** La relación representada tiene el siguiente conjunto de atributos: *edificio, número, superficie*.
**b)** Los dominios son dominio(*edificio*) = *edificios*, dominio(*número*) = *números* y dominio(*superficie*) = *sups*.
**c)** Las formas de denotar el esquema de relación son:
- DESPACHOS(*edificio, número, superficie*),
- DESPACHOS(*edificio, superficie, número*),
- DESPACHOS(*número, edificio, superficie*),
- DESPACHOS(*número, superficie, edificio*),
- DESPACHOS(*superficie, edificio, número*),
- DESPACHOS(*superficie, número, edificio*),

que corresponden a las posibles ordenaciones de sus atributos.
**d)** Elegiremos la siguiente forma de denotar el esquema de relación:
DESPACHOS(*edificio, número, superficie*).
Entonces el conjunto de tuplas de su extensión será:

Figura 5




**2.** Las superclaves de las relaciones correspondientes son:
**a)** {*edificio, número*} y {*edificio, número, superficie*}.
**b)** {*DNI*}, {*NSS*}, {*DNI, NSS*}, {*DNI, nombre*}, {*DNI, apellido*}, {*NSS, nombre*}, {*NSS, apellido*}, {*DNI, nombre, apellido*}, {*NSS, nombre, apellido*}, {*DNI, NSS, nombre, apellido*} y {*DNI, NSS, nombre, apellido*}.

**3.**
**a)** Se acepta.
**b)** Se rechaza porque viola la regla de integridad de entidad de la clave primaria.
**c)** Se rechaza porque viola la regla de integridad referencial.
**d)** Se rechaza porque viola la regla de integridad de unicidad de la clave primaria.
**e)** Se acepta.
**f)** Se rechaza porque viola la regla de integridad de unicidad de la clave primaria.
**g)** Se rechaza porque viola la regla de integridad referencial.
**h)** Se acepta y se borran el edificio Marina y todos sus despachos.

**4.**
**a)** Podemos utilizar la siguiente secuencia de operaciones:
- $A$ := *DESPACHOS*(*superficie*>15),
- $R := A * EDIFICIOS\_EMP$.

**b)** Podemos utilizar la siguiente secuencia de operaciones:
- $A$ := *DESPACHOS*[*edificio, número*],
- $B := EMPLEADOS\_ADM \cup EMPLEADOS\_PROD$,
- $C$ := $B$[*edificiodesp, númerodesp*],
- $R := A - C$.

**c)** Podemos utilizar la siguiente secuencia de operaciones:
- $A := EMPLEADOS\_ADM \cup EMPLEADOS\_PROD$,
- $B$ := $A$(*edificiodesp* = NULO y *númerodesp* = NULO),
- $R$ := $B$[*nombre, apellido*].

**d)** Podemos utilizar la siguiente secuencia de operaciones:
- $A := EMPLEADOS\_ADM \cup EMPLEADOS\_PROD$,
- *B(DNI, nombre, apellido, edificio, número) := A(DNI, nombre, apellido, edificiodesp, númerodesp)*,
- $C := B * DESPACHOS$,
- $D := C * EDIFICIOS\_EMP$,
- $R$ := $D$[*nombre, apellido, superficie, supmediadesp*].

**e)** Podemos utilizar la siguiente secuencia de operaciones:
- $A$ := *DESPACHOS*(*edificio = Diagonal y número = 120*),
- B(*Ed*, *Num*, *Sup*) := A(*edificio*, *número*, *superficie*),
- $C$ := *DESPACHOS*[*superficie*>Sup]$B$,
- $R$ := $C$[*edificio*, *número*].

**f)** Podemos utilizar la siguiente secuencia de operaciones:
- $A$ := *EMPLEADOS_ADM* $\cup$ *EMPLEADOS_PROD*,
- $B$(*DNI*, *nombre*, *apellido*, *edificio*, *número*) := $A$(*DNI*, *nombre*, *apellido*, *edificiodesp*, *númerodesp*),
- $C$ := *DESPACHOS* $*_1$ $B$,
- $R$ := $C$[*edificio*, *número*, *DNI*].

**5.** La secuencia siguiente:
- $A := T - S$,
- $R := T - A$,

sólo incluye operaciones primitivas, dado que la diferencia es primitiva, y obtiene el mismo resultado que $R := T \cap S$.

**6.** La siguiente secuencia:
- $A := T \times S$,
- $R := A(B = D \text{ y } C = E)$,

que sólo incluye operaciones primitivas (un producto cartesiano y una selección), obtiene el mismo resultado que $R := T[B = D, C = E]S$.

# Glosario

**actualización**
Hecho de reflejar los cambios que se producen en la realidad en las relaciones de una base de datos.

**actualización en cascada para el caso de borrado**
Política de mantenimiento de la integridad referencial que consiste en borrar una tupla $t$ que tiene una clave primaria referenciada, así como borrar todas las tuplas que referencian $t$.

**actualización en cascada para el caso de modificación**
Política de mantenimiento de la integridad referencial que consiste en permitir modificar atributos de la clave primaria de una tupla $t$ con una clave primaria referenciada, y modificar del mismo modo todas las tuplas que referencian la tupla $t$.

**anulación en caso de borrado**
Política de mantenimiento de la integridad referencial que consiste en borrar una tupla $t$ con una clave referenciada y, además, modificar todas las tuplas que referencian $t$ de modo que los atributos de la clave foránea correspondiente tomen valores nulos.

**anulación en caso de modificación**
Política de mantenimiento de la integridad referencial que consiste en modificar atributos de la clave primaria de una tupla $t$ con una clave referenciada y, además, modificar todas las tuplas que referencian $t$ de modo que los atributos de la clave foránea correspondiente tomen valores nulos.

**atributo (en el contexto del modelo relacional)**
Nombre del papel que ejerce un dominio en un esquema de relación.

**borrado**
Hecho de borrar una o más tuplas de una relación.

**cardinalidad de una relación**
Número de tuplas que pertenecen a su extensión.

**cierre relacional**
Propiedad de todas las operaciones del álgebra relacional según la cual tanto sus operandos como su resultado son relaciones.

**clave alternativa de una relación**
Clave candidata de la relación que no se ha elegido como clave primaria.

**clave candidata de una relación**
Superclave $C$ de la relación que cumple que ningún subconjunto propio de $C$ es superclave.

**clave primaria de una relación**
Clave candidata de la relación que se ha elegido para identificar las tuplas de la relación.

**clave foránea de una relación *R***
Subconjunto de los atributos del esquema de la relación, *CF*, tal que existe una relación *S* (*S* no debe ser necesariamente diferente de *R*) que tiene por clave primaria *CP*, y se cumple que, para toda tupla *t* de la extensión de *R*, los valores para *CF* de *t* son o bien valores nulos, o bien valores que coinciden con los valores para *CP* de alguna tupla *s* de *S*.

**combinación**
Operación del álgebra relacional que, a partir de dos relaciones, obtiene una nueva relación formada por todas las tuplas que resultan de concatenar tuplas de la primera relación con tuplas de la segunda relación, y que cumplen una condición de combinación especificada.

**combinación externa**
Extensión de combinación entre dos relaciones, *T* y *S*, que conserva en el resultado todas las tuplas de *T*, de *S* o de las dos relaciones.

**combinación natural**
Variante de combinación que consiste básicamente en una equicombinación seguida de la eliminación de los atributos superfluos.

**consulta**
Obtención de datos deducibles a partir de las relaciones que contiene la base de datos.

**diferencia**
Operación del álgebra relacional que, a partir de dos relaciones, obtiene una nueva relación formada por todas las tuplas que están en la primera relación y, en cambio, no están en la segunda.

**dominio (en el contexto del modelo relacional)**
Conjunto de valores atómicos.

**equicombinación**
Combinación en la que todas las comparaciones de la condición tienen el operador "=".

**esquema de relación**
Componente de una relación que consiste en un nombre de relación R y en un conjunto de atributos $\{A_1, A_2, ..., A_n\}$.

**extensión de una relación de esquema R($A_1$, $A_2$, ..., $A_n$)**
Conjunto de tuplas $t_i$ ($i = 1, 2, ..., m$) donde cada tupla $t_i$ es un conjunto de pares $t_i = \{<A_1:V_{i1}>, <A_2V_{i2}>, ..., <A_n:V_{in}>\}$ y, para cada par $<A_j:V_{ij}>$, se cumple que $v_{ij}$ es un valor de dominio($A_j$) o bien un valor nulo.

**grado de una relación**
Número de atributos que pertenecen a su esquema.

**inserción**
Hecho de añadir una o más tuplas a una relación.

**integridad**
Propiedad de los datos de corresponder a representaciones plausibles del mundo real.

**intersección**
Operación del álgebra relacional que, a partir de dos relaciones, obtiene una nueva relación formada por las tuplas que están en las dos relaciones de partida.

**lenguaje basado en el cálculo relacional**
Lenguaje que proporciona un tipo de formulación de consultas fundamentado en el cálculo de predicados de la lógica matemática.

**lenguaje basado en el álgebra relacional**
Lenguaje que proporciona un tipo de formulación de consultas inspirado en la teoría de conjuntos.

**modificación**
Hecho de alterar los valores que tienen una o más tuplas de una relación para uno o más de sus atributos.

**producto cartesiano**
Operación del álgebra relacional que, a partir de dos relaciones, obtiene una nueva relación formada por todas las tuplas que resultan de concatenar tuplas de la primera relación con tuplas de la segunda relación.

**proyección**
Operación del álgebra relacional que, a partir de una relación, obtiene una nueva relación formada por todas las (sub)tuplas de la relación de partida que resultan de eliminar unos atributos especificados.

**redenominar**
Operación auxiliar del álgebra relacional que permite cambiar los nombres que figuran en el esquema de una relación.

**regla de integridad de dominio**
Regla que establece que un valor no nulo de un atributo $A_i$ debe pertenecer al dominio del atributo $A_i$, y que los operadores que es posible aplicar sobre los valores dependen de los dominios de estos valores.

**regla de integridad de entidad de la clave primaria**
Regla que establece que si el conjunto de atributos *CP* es la clave primaria de una relación R, la extensión de R no puede tener en ningún momento ninguna tupla con un valor nulo para alguno de los atributos de *CP*.

**regla de integridad de modelo**
Condiciones generales que deben cumplirse en toda base de datos de un modelo determinado.

**regla de integridad de unicidad de la clave primaria**
Regla que establece que si el conjunto de atributos *CP* es la clave primaria de una relación *R*, la extensión de *R* no puede tener en ningún momento dos tuplas con la misma combinación de valores para los atributos de *CP*.

**regla de integridad referencial**
Regla que establece que si el conjunto de atributos *CF* es una clave foránea de una relación *R* que referencia una relación *S* (no necesariamente diferente de *R*), que tiene por clave primaria *CP*, entonces, para toda tupla *t* de la extensión de *R*, los valores para *CF* de *t* son o bien valores nulos o bien valores que coinciden con los valores para *CP* de alguna tupla *s* de *S*.

**relación**
Elemento de la estructura de los datos de una base de datos relacional formado por un esquema (o intensión) y una extensión.

**restricción en caso de modificación**
Política de mantenimiento de la integridad referencial, que consiste en no permitir modificar ningún atributo de la clave primaria de una tupla si se trata de una clave primaria referenciada.

**restricción en caso de borrado**
Política de mantenimiento de la integridad referencial que consiste en no permitir borrar una tupla si tiene una clave primaria referenciada.

**restricciones de integridad de usuario**
Condiciones específicas que se deben cumplir en una base de datos concreta.

**selección**
Operación del álgebra relacional que, a partir de una relación, obtiene una nueva relación formada por todas las tuplas de la relación de partida que cumplen una condición de selección especificada.

**superclave de una relación de esquema R($A_1$, $A_2$, ..., $A_n$)**
Subconjunto de los atributos del esquema tal que no puede haber dos tuplas en la extensión de la relación que tengan la misma combinación de valores para los atributos del subconjunto.

**unión**
Operación del álgebra relacional que, a partir de dos relaciones, obtiene una nueva relación formada por todas las tuplas que están en alguna de las relaciones de partida.

# Bibliografía

## Bibliografía básica

**Date, C.J.** (2001). *Introducción a los sistemas de bases de datos* (7ª ed.). Prentice-Hall.

**Elmasri, R.; Navathe, S.B.** (2000). *Sistemas de bases de datos. Conceptos fundamentales* (3ª ed.). Madrid: Addison-Wesley Iberoamericana.