Tema 2 – Bases de Datos Relacionales

1º DAW – Bases de Datos

J. Carlos Moreno

Curso 2025/2026

Tabla de contenido

2 Bases de Datos Relacional

La base de datos relacional (BDR) es un tipo de base de datos (BD) implementada bajo las especificaciones del modelo relacional.

2.1 Modelo Relacional

El Modelo Relacional es el modelo más utilizado para representar sistemas dinámicos de información, se impuso dada las limitaciones de los modelos anteriores: modelo en red y jerárquico, pero sobre todo por su sencillez y robustez.

Se basa:

- Uso de las tablas para almacenar los datos
- Conexiones entre las tablas para relacionar los datos
- Conjunto de restricciones (o condiciones) que deberán cumplir dichos datos

2.1.1 Historia

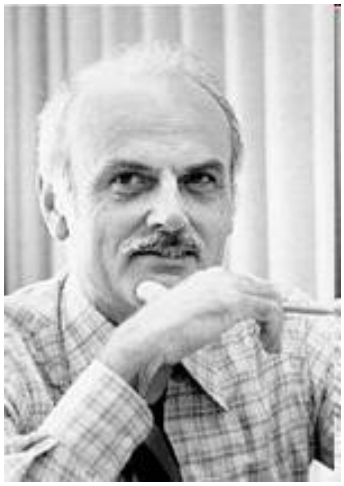

Ilustración 1 Edgar Frank Codd

En 1970 Edgar Frank Codd publicó un artículo en Comunications of the ACM, proponiendo un nuevo modelo de datos que tenía como objetivo fundamental aislar al usuario de las estructuras físicas de los datos, consiguiendo así la independencia de las aplicaciones respecto de los datos.

El nuevo modelo se basa en la teoría matemática de las relaciones. Los datos se estructuran lógicamente en forma de relaciones (muy parecido al concepto de tabla).

Aunque trabajaba para IBM, esta empresa no recibió de buen grado sus teorías (de hecho continuó trabajando en su modelo en red IMS), siendo otras empresas (en especial Oracle) las que implementaron sus postulados.

Pocos años después el modelo se empezó a utilizar cada vez más, hasta finalmente ser el modelo de bases de datos más popular y usado para el diseño e implementación de Bases de Datos.

2.1.2 Situación y Evolución

Las características del modelo relacional han hecho que prácticamente todos los SGBD comerciales y libres implementen este modelo:

- Algunas de las principales empresas informáticas del mundo son, en origen, empresas de SGBD-R: ORACLE, Sybase, INFORMIX, PostgreSQL, Access, MySQL, SQLSERVER

El tremendo éxito del modelo relacional ha supuesto que el cambio tecnológico a la siguiente generación esté siendo evolutivo y no revolucionario triunfan los SGBD Objeto-Relacionales, y fracasando, en general, los SGBD de Objetos puros.

2.1.3 Objetivos

Independencia física

El modo en que se almacenan los datos no influye en su manipulación lógica. No será necesario modificar las aplicaciones por cambios en el almacenamiento físico

Independencia lógica.

Cualquier modificación en los distintos objetos de la base de datos, no repercuta ni en los usuarios finales ni en las aplicaciones que están accediendo a la BBDD.

Flexibilidad.

La base de datos ofrece fácilmente distintas vistas en función de los usuarios y aplicaciones.

Uniformidad y Sencillez.

Los usuarios ven la base de datos relacional como una colección de tablas, y al ser la tabla la estructura fundamental del modelo, éste goza de una gran uniformidad puesto que se trata de una estructura de datos muy sencilla y robusta. Por otro lado el lenguaje de datos que utiliza es SQL que se trata de un lenguaje orientado al usuario final, facilitando así el manejo de los sistemas relacionales.

Fundamentación teórica.

El Modelo está definido con rigor matemático, ya que está basado en la teoría matemática de las relaciones o conjuntos.

2.2 Estructurade Datos del Modelo Relacional

La estructura de datos del modelo relacional es la relación. En este apartado se presenta esta estructura de datos, sus propiedades, los tipos de relaciones y qué es una clave de una relación. Para facilitar la comprensión de la definición formal de todos estos conceptos, veremos una definición informal previa, que permita asimilar dichos conceptos con otros que resulten familiares.

En este apartado veremos los conceptos de Relación, Atributos, Dominio yTupla.

2.2.1 Relación

El modelo relacional se basa en el concepto matemático de **relación**, que gráficamente se representa mediante una tabla. *Codd*, que era un experto matemático, utilizó una terminología perteneciente a las matemáticas, en concreto de la teoría de conjuntos y de la lógica de predicados. Una relación es una tabla formada por un conjunto defilas y columnas.

Un SGBD sólo necesita que el usuario pueda percibir la base de datos como un conjunto de tablas.

No hay que confundir la idea de relación según el modelo de *Codd*, con lo que significa una relación en el modelo Entidad/Relación de *Chen*.

Puesto que una relación se representa como una tabla; podemos entender que las columnas de la tabla son los atributos; y las filas, las tuplas. Conceptos que veremos más adelante

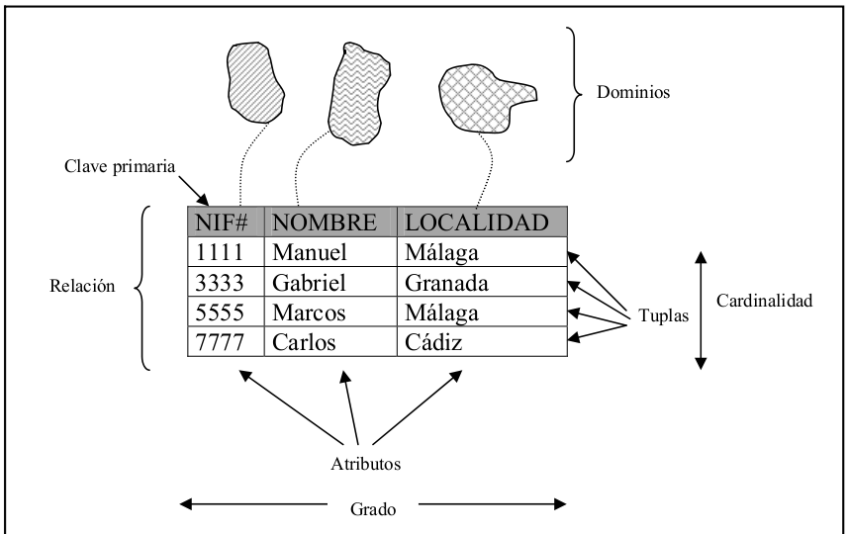

Ilustración 2. Estructura Modelo Relacional

2.2.1.1 Grado y Cardinalidad

El **grado** de una relación (que se representa mediante una tabla) es el número de atributos que posee. Según el grado se determinan los siguientes tipos de relación:

- Binaria
- Ternaria
- N-aria

La **cardinalidad** de una relación es el número de tuplas (filas o registros) que posee

En el ejemplo de la ilustración 2 la relación será de Grado 3 y Cardinalidad 4.

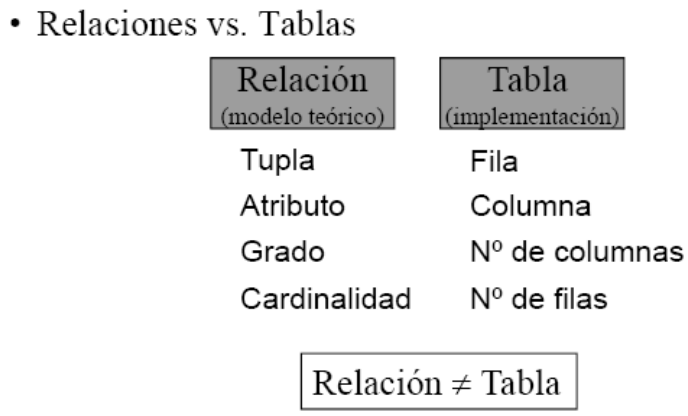

Ilustración 3. Relación Vs Tabla

2.2.1.2 *Atributo*

Un atributo es el nombre de una columna de una relación. En el modelo relacional, las relaciones se utilizan para almacenar información sobre los objetos que se representan en la base de datos. Una relación se representa gráficamente como una tabla bidimensional en la que las filas corresponden a registros individuales y las columnas corresponden a los campos o atributos de esos registros. Los atributos pueden aparecer en la relación en cualquier orden.

CLIENTES

codcli	nombre	dirección	codpostal	codpue
333	Sos Carretero, Jesús	Mosen Compte, 14	12964	53596
336	Miguel Archilés, Ramón	Bernardo Mundina, 132-5	12652	07766
342	Pinel Huerta, Vicente	Francisco Sempere, 37-10	12112	07766
345	López Botella, Mauro	Avenida del Puerto, 20-1	12439	12309
348	Palau Martínez, Jorge	Raval de Sant Josep, 97-2	12401	12309
354	Murría Vinaiza, José	Ciudadela, 90-18	12990	12309
357	Huguet Peris, Juan Ángel	Calle Mestre Rodrigo, 7	12930	12309

PUEBLOS

codpue	nombre	codpro
07766	Burriana	12
12309	Castellón	12
17859	Enramona	12
46332	Soneja	12
53596	Vila-real	12

Ilustración 4. Ejemplo 1

Por ejemplo, la información de los clientes de una empresa determinada se representa mediante la relación CLIENTES, que tiene columnas para los atributos codcli (código del cliente), nombre (nombre y apellidos del cliente), dirección (calle y número donde se ubica el cliente), codpostal (código postal correspondiente a la dirección del cliente) y codpue (código de la población del cliente). La información sobre las poblaciones se representa mediante la relación PUEBLOS de la misma figura, que tiene columnas para los atributos codpue (código de la población), nombre (nombre de la población) y codpro (código de la provincia en que se encuentra la población)

Características atributos:

- Todas las tuplas de una relación tienen el mismo número de atributos

- Cada atributo ha de ser único y con un nombre único en la relación
- Los atributos se pueden presentar en cualquier orden
- No puede haber atributos duplicados
- El valor de cada atributo ha de estar contenido en el dominio que representa, aunque puede ser un valor NULO.

2.2.1.3 Dominio

El concepto de dominio es importante porque permite que el usuario defina, en un lugar común, el significado y la fuente de los valores que los atributos pueden tomar. Esto hace que haya más información disponible para el sistema cuando éste va a ejecutar una operación relacional, de modo que las operaciones que son semánticamente incorrectas, se pueden evitar. Por ejemplo, no tiene sentido comparar el nombre de una calle con un número de teléfono, aunque los dos atributos sean cadenas de caracteres. Sin embargo, el importe mensual del alquiler de un inmueble no estará definido sobre el mismo dominio que el número de meses que dura el alquiler, pero sí tiene sentido multiplicar los valores de ambos dominios para averiguar el importe total al que asciende el alquiler.

Los SGBD relacionales no ofrecen un soporte completo de los dominios ya que su implementación es extremadamente compleja.

Un **dominio** contiene todos los posibles valores homogéneos (del mismo tipo) y atómicos (indivisibles) que puede tomar un determinado atributo.

Los valores contenidos en un atributo pertenecen a un dominio previamente definido. Dos atributos distintos pueden tener el mismo dominio. Todos los dominios tienen un nombre y un tipo de datos asociado.

Tipos de dominio:

- **Generales**. Los valores están comprendidos entre un máximo y un mínimo, por ejemplo, precio de un producto valor comprendido entre 0 y valor máximo permitido para precio.
- **Restringidos**. Sólo pueden tomar un conjunto de valores específicos, por ejemplo Sexo que podrá tomar los valores H o M.

2.2.1.4 Tupla

Una tupla es una fila de una relación. Los elementos de una relación son las tuplas o filas de la tabla. En la relación CLIENTES de la ilustración 4, cada tupla tiene cinco valores, uno para cada atributo. Las tuplas de una relación no siguen ningún orden y además dentro de una relación no pueden existir dos tuplas exactamente iguales, es decir, con los mismos valores en cada uno de sus atributos.

Ya dijimos que la cardinalidad de una relación es el número de tuplas que contiene. Ya que en las relaciones se van insertando y borrando tuplas a menudo, la cardinalidad de las mismas varía constantemente.

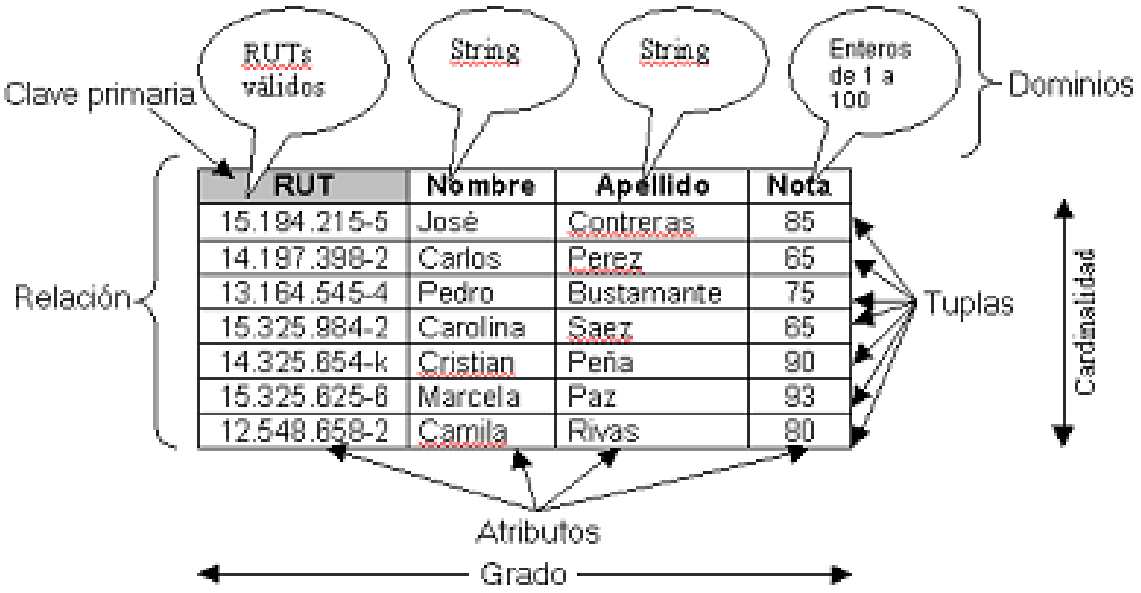

Ilustración 5. Ejemplo

2.2.2 Definición Formal Relación

Una relación R definida sobre un conjunto de dominios D1,D2,...,Dn consta de:

- **Cabecera**: conjunto fijo de pares atributo:dominio

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

  donde cada atributo Aj corresponde a un único dominio Dj y todos los Aj son distintos, es decir, no hay dos atributos que se llamen igual. El grado de la relación R es n.

- **Cuerpo**: conjunto variable de tuplas. Cada tupla es un conjunto de pares atributo:valor:

  $\{(A_1 : v_{i1}),(A_2 : v_{i2}),...,(A_n : v_{in})\}$

  con i = 1,2,...,m, donde m es la cardinalidad de la relación R. En cada par $(A_j : v_{ij})$ se tiene que $v_{ij} \in D_j$.

Ejemplo:

A continuación, se muestra la cabecera de la relación CLIENTES de la ilustración 4, y una de sus tuplas.

> CLIENTES { (codcli:codcli_dom), (nombre:nombre_dom), (dirección:dirección_dom), (codpostal:codpostal_dom), (codpue:codpue_dom) }
>
> { (codcli:333), (nombre:Sos Carretero, Jesús), (dirección:Mosen Compte, 14), (codpostal:12964), (codpue:53596) }

Este conjunto de pares no está ordenado, por lo que la tupla anterior y la siguiente son la misma:

{ (nombre:Sos Carretero, Jesús), (codpostal:12964), (codcli:333), (dirección:Mosen Compte, 14), (codpue:53596) }

Las relaciones se suelen representar gráficamente mediante tablas. Los nombres de las columnas corresponden a los nombres de los atributos, y las filas son cada una de las tuplas de la relación. Los valores que aparecen en cada una de las columnas pertenecen al conjunto de valores del dominio sobre el que está definido el atributo correspondiente.

Clientes		
DNI	**Nombre**	**Edad**
12333944C	Ana	52
12374678G	Eva	27
28238232H	Martín	33

Esquema: **Cliente**(DNI:**DNI**, Nombre:**Nombre**, Edad:**Edad**)

Cuerpo: {(DNI: "12333944C", Nombre:"Ana", Edad:52), (DNI: "12374678G", Nombre:"Eva", Edad;52), (DNI: "28238232H", Nombre:"Martín",Edad:33)}

Ilustración 6. Ejemplo Definición Formal Relación

2.2.3 Propiedades de la Relación

Las relaciones tienen las siguientes características:

- Cada relación tiene un nombre, y éste es distinto del nombre de todas las demás.
- Los dominios sobre los que se definen los atributos son escalares, por lo que los valores de los atributos son atómicos. De este modo, en cada tupla, cada atributo toma un solo valor. Se dice que las relaciones están normalizadas.
- No hay dos atributos que se llamen igual.
- El orden de los atributos no importa: los atributos no están ordenados.
- Cada tupla es distinta de las demás: no hay tuplas duplicadas.
- El orden de las tuplas no importa: las tuplas no están ordenadas.

2.2.4 Tipos de relaciones

En un SGBD relacional hay dos tipos de relaciones:

- **Relaciones base**. Son relaciones reales que tienen nombre, y forman parte directa de la base de datos almacenada. Se dice que las relaciones base son relaciones autónomas.
- **Vistas**. También denominadas relaciones virtuales, son relaciones con nombre y derivadas (no autónomas). Que son derivadas significa que se obtienen a partir de otras relaciones; se representan mediante su definición en términos de esas otras relaciones. Las vistas no poseen datos almacenados propios, los datos que contienen corresponden a datos almacenados en relaciones base.

2.2.5 Claves

Ya que en una relación no hay tuplas repetidas, éstas se pueden distinguir unas de otras, es decir, se pueden identificar de modo único. La forma de identificarlas es mediante los valores de sus atributos. Se denomina **superclave** a un atributo o conjunto de atributos que identifican de modo único las tuplas de una relación. Se denomina **clave candidata** a una superclave en la

que ninguno de sus subconjuntos es una superclave de la relación. El atributo o conjunto de atributos K de la relación R es una clave candidata para R si, y sólo si, satisface las siguientes propiedades:

- **Unicidad**: nunca hay dos tuplas en la relación R con el mismo valor de K.
- **Irreducibilidad** (minimalidad): ningún subconjunto de K tiene la propiedad de unicidad, es decir, no se pueden eliminar componentes de K sin destruir la unicidad.

Cuando una clave candidata está formada por más de un atributo, se dice que es una **clave compuesta**. Una relación puede tener varias claves candidatas.

CLIENTES

codcli	nombre	dirección	codpostal	codpue
333	Sos Carretero, Jesús	Mosen Compte, 14	12964	53596
336	Miguel Árchilés, Ramón	Bernardo Mundina, 132-5	12652	07766
342	Pinel Huerta, Vicente	Francisco Sempere, 37-10	12112	07766
345	López Botella, Mauro	Avenida del Puerto, 20-1	12439	12309
348	Palau Martínez, Jorge	Raval de Sant Josep, 97-2	12401	12309
354	Murría Vinaiza, José	Ciudadela, 90-18	12990	12309
357	Huguet Peris, Juan Ángel	Calle Mestre Rodrigo, 7	12930	12309

PUEBLOS

codpue	nombre	codpro
07766	Burriana	12
12309	Castellón	12
17859	Enramona	12
46332	Soneja	12
53596	Vila-real	12

Ilustración 7. Ejemplo

Por **ejemplo**, en la relación PUEBLOS de la ilustración 7, el atributo *nombre* no es una clave candidata ya que hay pueblos en España con el mismo nombre que se encuentran en distintas provincias. Sin embargo, se ha asignado un código único a cada población, por lo que el atributo *codpue* sí es una clave candidata de la relación PUEBLOS. También es una clave candidata de estar elación la pareja formada por los atributos nombre y codpro, ya que no hay dos poblaciones en la misma provincia que tengan el mismo nombre.

Para identificar las claves candidatas de una relación no hay que fijarse en un estado u ocurrencia de la base de datos. El hecho de que en un momento dado no haya duplicados para un atributo o conjunto de atributos, no garantiza que los duplicados no sean posibles. Sin embargo, la presencia de duplicados en un estado de la base de datos sí es útil para demostrar que cierta combinación de atributos no es una clave candidata. El único modo de identificar las claves candidatas es conociendo el significado real de los atributos, ya que esto permite saber si es posible que aparezcan duplicados. Sólo usando esta información semántica se puede saber con certeza si un conjunto de atributos forma una clave candidata. Por ejemplo, viendo la ocurrencia anterior de la relación CLIENTES se podría pensar que el atributo nombre es una clave candidata. Pero ya que este atributo es el nombre de un cliente y es posible que haya dos clientes con el mismo nombre, el atributo no es una clave candidata.

2.2.5.1 Clave Primaria

Se denomina **clave primaria** de una relación a aquella clave candidata que se escoge para identificar sus tuplas de modo único. Ya que una relación no tiene tuplas duplicadas, siempre hay una clave candidata y, por lo tanto, la relación siempre tiene clave primaria. En el peor caso, la clave primaria estará formada por todos los atributos de la relación, pero normalmente habrá un pequeño subconjunto de los atributos o incluso un solo atributo que haga esta función.

Empleados : Tabla

	NEmplead	Nombre	Apellidos	Dni	Domicilio	Poblacion	CP	FechaIngreso	Departame	CodCateg	CargoPer	Casado	Observ
+	1	GEMA	DIAZ DIAZ	10.111.222-A	LA SOTA 2	BARREDOS	33970	11/12/2001	TEC	1	3	☑	
+	2	ENRIQUE	BORBON CASTRO	10.222.333-B	RIO EO 1	POLA DE LAVIANA	33980	10/11/2000	OBD	1	1	☐	
+	3	PABLO	SANTOS GINER	10.333.444-C	ASTURIAS 2	GIJÓN	33211	31/12/1998	PRO	1	3	☑	
+	4	ELENA	DOSAL GOMEZ	10.444.555-F	URIA 4	GIJON	33213	25/07/1990	ANA	2	1	☑	
+	5	SANTIAGO	SEGURA TORRENTE	10.555.666-G	CONDE SIZZO	LA FELGUERA	33960	31/07/1967	ANA	3	2	☐	
+	6	MARTA	MIRALLES KENNER	10.666.777-H	AVDA. PRINCIPAD	OVIEDO	33011	13/08/1995	OBD	4	2	☐	
+	7	ROCIO	FERRER GOMEZ	10.777.888-J	LAS XANAS, 2	GIJON	33212	14/07/2000	TEC	5	1	☐	
✳	0					Ubrique				0	0	☐	

Ilustración 8. Tabla de empleados

Normalmente en el diseño de base de datos siempre a una relación se le añade un atributo de tipo numérico que realizará las funciones de Clave Primaria, por ejemplo, si nos fijamos en la ilustración 8, vemos como a la tabla de empleados se le ha añadido el atributo NEmpleados para que ejerza la función de Clave Primaria.

Cuando disponemos de varias Claves Candidatas hemos de elegir la Clave Primaria basándonos en los siguientes criterios:

- La candidata que identifique mejor a cada tupla en el contexto de la base de datos.
- Ha de ser lo más mínima posible, en el sentido de que en su composición no intervengan más que los atributos estrictamente requeridos para identificar las tuplas de forma única.
- Normalmente se usan Claves Numéricas con autoincremento, que ocupan menos espacio.

2.2.5.2 Claves Alternativas

Las claves candidatas que no son escogidas como clave primaria son denominadas **claves alternativas**. Por ejemplo, a partir de la ilustración 7 observamos que la clave primaria de la relación PUEBLOS es el atributo *codpue*, siendo la pareja formada por *nombre* y *codpro* una clave alternativa. En la relación *CLIENTES* sólo hay una clave candidata que es el atributo *codcli*, por lo que esta clave candidata es la clave primaria. Si nos fijamos en la ilustración 8 las claves candidatas de Empleados son *NEmpleados* y *Dni*, como *NEmpleados* se ha seleccionado como Clave Primaria, *Dni* se convierte en Clave Alternativa.

A las Claves Alternativas también se le denominan **Claves Secundarias**.

2.2.5.3 Claves Ajenas

Una clave ajena es un atributo o un conjunto de atributos de una relación cuyos valores coinciden con los valores de la clave primaria de alguna otra relación (puede ser la misma). Las claves ajenas representan relaciones entre datos.

Por ejemplo, el atributo *codpue* de *CLIENTES* relaciona a cada cliente con su población. Este atributo en *CLIENTES* es una clave ajena cuyos valores hacen referencia al atributo *codpue* de

PUEBLOS (su clave primaria). Se diceque un valor de clave ajena representa una referencia a la tupla que contieneel mismo valor en su clave primaria (tupla referenciada).

Si nos fijamos en los datos de la ilustración 7, para conocer el nombre de la población del cliente con *codcli = 333*, debemos seguir la clave ajena *codpue* que aparece en la tupla de dicho cliente y que tiene el valor *53596*. Seguir la referencia que implica la clave ajena conlleva visitar la relación *PUEBLOS* y localizar la fila que tiene el valor *53596* en su clave primaria. Nótese que, en este ejemplo, la clave ajena tiene el mismo nombre que la clave primaria a la que hace referencia. Esto no es un requisito, las claves ajenas no precisan tener el mismo nombre que la clave primaria a la que referencian; sin embargo, si se utilizan los mismos nombres (o nombres compuestos derivados de los mismos) es más fácil así reconocer las claves ajenas.

Al hablar de claves primarias y de claves ajenas es importante darse cuenta de que los valores de una clave primaria no se pueden repetir, mientras que no sucede lo mismo con las claves ajenas que le hacen referencia. Así, en las tablas de la Ilustración 7 no es posible encontrar dos tuplas con el mismo valor en *PUEBLOS.codpue* (cada población debe aparecer en la relación una sola vez), pero sí es posible encontrar varias tuplas con el mismo valor en *CLIENTES.codpue*, ya que es posible que haya varios clientes que se ubiquen en la misma población.

7	**Pedro Morales**	**Mediocampo**	**1**

Ilustración 8. Ejemplo Clave Ajena

Equipo_id de la tabla jugadores es una clave ajena, cuyos valores hacen referencia a la clave principal id de la tabla **equipos**.

2.3 Esquema de una Base de Datos Relacional

Una base de datos relacional es un conjunto de relaciones. Para representar el esquema de una base de datos relacional se debe dar el nombre de sus relaciones, los atributos de éstas, los dominios sobre los que se definen estos atributos, las claves primarias y las claves ajenas.

El esquema de la base de datos de la empresa con la que trabajaremos en estos apuntes es el siguiente:

```
  CLIENTES(codcli, nombre, dirección, codpostal, codpue)
VENDEDORES(codven, nombre, dirección, codpostal, codpue, codjefe)
   PUEBLOS(codpue, nombre, codpro)
PROVINCIAS(codpro, nombre)
 ARTÍCULOS(codart, descrip, precio, stock, stock_min, dto)
  FACTURAS(codfac, fecha, codcli, codven, iva, dto)
LÍNEAS_FAC(codfac, línea, cant, codart, precio, dto)
```

Ilustración 9. Esquema Base de Datos

En el esquema anterior, los nombres de las relaciones aparecen seguidos de los nombres de los atributos encerrados entre paréntesis. Las claves primarias son los atributos subrayados. Las claves ajenas se representan mediante los siguientes diagramas referenciales:

CLIENTES	$\xrightarrow{\text{codpue}}$	PUEBLOS	:	Población del cliente.
VENDEDORES	$\xrightarrow{\text{codpue}}$	PUEBLOS	:	Población del vendedor.
VENDEDORES	$\xrightarrow{\text{codjefe}}$	VENDEDORES	:	Jefe del vendedor.
PUEBLOS	$\xrightarrow{\text{codpro}}$	PROVINCIAS	:	Provincia en la que se encuentra la población.
FACTURAS	$\xrightarrow{\text{codcli}}$	CLIENTES	:	Cliente al que pertenece la factura.
FACTURAS	$\xrightarrow{\text{codven}}$	VENDEDORES	:	Vendedor que ha realizado la venta.
LÍNEAS_FAC	$\xrightarrow{\text{codfac}}$	FACTURAS	:	Factura en la que se encuentra la línea.
LÍNEAS_FAC	$\xrightarrow{\text{codart}}$	ARTÍCULOS	:	Artículo que se compra en la línea de factura.

Ilustración 10. Claves Ajenas

La tabla PROVINCIAS almacena información sobre las provincias de España.De cada provincia se almacena su nombre (*nombre*) y un código quela identifica (*codpro*). La tabla *PUEBLOS* contiene los nombres (*nombre*) de los pueblos de España. Cada pueblo se identifica por un código que es único(*codpue*) y tiene una referencia a la provincia a la que pertenece (*codpro*).

La tabla *CLIENTES* contiene los datos de los clientes: código que identifica acada uno (*codcli*), nombre y apellidos (*nombre*), calle y número (*dirección*), código postal (*codpostal*) y una referencia a su población (*codpue*). La tabla*VENDEDORES* contiene los datos de los vendedores de la empresa: códigoque identifica a cada uno (*codven*), nombre y apellidos (*nombre*), calle y número(*dirección*), código postal (*codpostal*), una referencia a su población(*codpue*) y una referencia al vendedor del que depende (*codjefe*), si es elcaso. En la tabla ARTÍCULOS se tiene el código que identifica a cada artículo(*codart*), su descripción (*descrip*), el precio de venta actual (precio), elnúmero de unidades del artículo que hay en el almacén (*stock*), la cantidadmínima que se desea mantener almacenada (*stock_min*) y, si el artículo estáen oferta, el descuento (*dto*) que se debe aplicar cuando se venda. La tablaFACTURAS contiene las cabeceras de las facturas correspondientes a las comprasrealizadas por los clientes. Cada factura tiene un código único (*codfac*), la fecha en que se ha realizado (*fecha*), así como el IVA (*iva*) y el descuentoque se le ha aplicado (*dto*). Cada factura hace referencia al cliente al que pertenece (*codcli*) y al vendedor que la ha realizado (*codven*). Las líneas decada factura se encuentran en la tabla *LÍNEAS_FAC*, identificándose cada unapor el número de línea que ocupa dentro de la factura (*codfac*, *línea*). Encada una de ellas se especifica la cantidad de unidades (*cant*) del artículo quese compra (*codart*), el precio de venta por unidad (precio) y el descuento quese aplica sobre dicho precio (*dto*), si es que el artículo estaba en oferta cuandose vendió.

A continuación, se muestra un estado de la base de datos cuyo esquema se acaba de definir.

CLIENTES

codcli	nombre	dirección	codpostal	codpue
333	Sos Carretero, Jesús	Mosen Compte, 14	12964	53596
336	Miguel Archilés, Ramón	Bernardo Mundina, 132-5	12652	07766
342	Pinel Huerta, Vicente	Francisco Sempere, 37-10	12112	07766
345	López Botella, Mauro	Avenida del Puerto, 20-1	12010	12309
348	Palau Martínez, Jorge	Raval de Sant Josep, 97-2	12003	12309
354	Murría Vinaiza, José	Ciudadela, 90-18	12003	12309
357	Huguet Peris, Juan Ángel	Calle Mestre Rodrigo, 7	12100	12309

VENDEDORES

codven	nombre	dirección	codpostal	codpue	codjefe
5	Guillén Vilar, Natalia	Sant Josep, 110	12597	53596	105
105	Poy Omella, Paloma	Sanchis Tarazona, 103-1	12257	46332	
155	Rubert Cano, Diego	Benicarló Residencial, 154	12425	17859	5
455	Agost Tirado, Jorge	Pasaje Peñagolosa, 21-19	12914	53596	5

PUEBLOS

codpue	nombre	codpro
07766	Burriana	12
12309	Castellón	12
17859	Enramona	12
46332	Soneja	12
53596	Vila-real	12

PROVINCIAS

codpro	nombre
03	Alicante
12	Castellón
46	Valencia

ARTÍCULOS

codart	descrip	precio	stock	stock_min	dto
IM3P32V	Interruptor magnetotérmico 4p, 2	27.01	1	1	
im4P10L	Interruptor magnetotérmico 4p, 4	32.60	1	1	15
L14340	Bases de fusibles cuchillas T0	0.51	3	3	
L17055	Bases de fusible cuchillas T3	7.99	3	3	
L76424	Placa 2 E. legrand serie mosaic	2.90	5	2	
L85459	Tecla legrand marfil	2.80	0	4	
L85546	Tecla difusores legrand bronce	1.05	13	5	5
L92119	Portalámparas 14 curvo	5.98	2	1	
ME200	Marco Bjc Ibiza 2 elementos	13.52	1	1	
N5072	Pulsador luz piloto Niessen trazo	1.33	11	2	
N8017BA	Reloj Orbis con reserva de cuerda	3.40	7	4	
P605	Caja 1 elem. plastimetal	1.65	16	9	
P695	Interruptor rotura brusca 100 A M	13.22	1	1	
P924	Interruptor marrón dec. con visor	2.39	8	3	
REF1X20	Regleta fluorescente 1x36 bajo F	8.71	1	1	
S3165136	Bloque emergencia Satf 150 L	4.81	6	3	
T4501	Tubo empotrar 100	2.98	0	5	
TE7200	Doble conmutador Bjc Ibiza blanco	13.22	1	1	
TFM16	Curva tubo hierro 11	0.33	23	13	
TH11	Curva tubo hierro 29	1.42	20	3	
THC21	Placa mural Felmax	1.56	1	1	
ZNCL	Base T,t lateral Ticino S, Tekne	41.71	1	1	10

FACTURAS

codfac	fecha	codcli	codven	iva	dto
6643	16/07/2010	333	105	18	10
6645	16/07/2010	336	105	0	20
6654	31/07/2010	357	155	8	0
6659	08/08/2010	342	5	0	0
6680	10/09/2010	348	455	8	0
6723	06/11/2010	342	5	18	0
6742	17/12/2010	333	105	8	20

LÍNEAS_FAC

codfac	linea	cant	codart	precio	dto
6643	1	6	L14340	0.51	20
6643	2	1	N5072	1.33	0
6643	3	2	P695	13.22	0
6645	1	10	ZNCL	41.71	0
6645	2	6	N8017BA	3.40	0
6645	3	3	TE7200	13.22	0
6645	4	4	L92119	5.98	0
6654	1	6	REF1X20	8.71	50
6659	1	8	THC21	1.56	0
6659	2	12	L17055	7.99	25
6659	3	9	L76424	2.90	0
6680	1	12	T4501	2.98	0
6680	2	11	im4P10L	32.60	0
6723	1	5	L85459	2.80	5
6742	1	9	ME200	13.52	0
6742	2	8	S3165136	4.81	5

2.4 Reglas de Integridad

Una vez definida la estructura de datos del modelo relacional, pasamos a estudiar las reglas de integridad que los datos almacenados en dicha estructura deben cumplir para garantizar que son correctos.

Al definir cada atributo sobre un dominio se impone una restricción sobre el conjunto de valores permitidos para cada atributo. A este tipo de restricciones se les denomina **restricciones de dominios**. Hay además dos reglas de integridad muy importantes que son restricciones que se deben cumplir en todas las bases de datos relacionales y en todos sus estados (las reglas se deben cumplir todoel tiempo). Estas reglas son: **reglas inherentes al modelo**, **restricciones de usuario o semánticas** y **regla de integridad referencial**. Antes de definirlas, es preciso conocer el concepto de valor nulo.

2.4.1 Valor Nulo

Cuando en una tupla un atributo es desconocido, se dice que es nulo. Un nulo no representa el valor cero ni la cadena vacía ya que éstos son valores que tienen significado. El nulo implica ausencia de información, bien porque al insertar la tupla se desconocía el valor del atributo, o bien porque para dicha tupla el atributo no tiene sentido.

Ya que los nulos no son valores, deben tratarse de modo diferente, lo que causa problemas de implementación. De hecho, no todos los SGBD relacionales soportan los nulos.

2.4.2 Regla de Integridad Inherentes al modelo

Son aquellas que no son determinadas por los usuarios, sino que son establecidas por el mismo Modelo Relacional. Indican las características propias de una Relación y que han de cumplirse obligatoriamente y que diferencian una Relación de una Tabla.

Algunas de ellas son las siguientes:

- *Regla de Integridad de la Relación*. Ningún atributo que forma parte de la clave puede tomar un valor desconocido, inexistente o *nulo*.
- Cuando tenemos una clave compuesta ha de ser irreductible.
- No puede haber dos tuplas iguales en una misma relación
- El orden de las tuplas y atributos no es relevante
- Cada atributo sólo puede tomar un valor del dominio al que pertenece, no admitiéndose, por tanto, los grupos repetitivos. Es decir, no podemos, por ejemplo, poner dos valores para el campo edad en una hipotética tabla de empleados.

La primera regla de integridad se aplica a las claves primarias de las relaciones base: ninguno de los atributos que componen la clave primaria puede ser nulo. Por definición, una clave primaria es una clave irreducible que se utiliza para identificar de modo único las tuplas. Que es irreducible significa que ningún subconjunto de la clave primaria sirve para identificar las tuplas de modo único. Si se permitiera que parte de la clave primaria fuera nula, se estaría diciendo que no todos sus atributos son necesarios para distinguir las tuplas, con lo que se estaría contradiciendo la irreducibilidad. Nótese que esta regla sólo se aplica a las relaciones base y a las claves primarias, no a las claves alternativas.

2.4.3 Restricciones de usuario o semánticas

A la hora de definir las propiedades de una relación y sus atributos podemos emplear las siguientes restricciones:

- Clave Primaria o Principal
- Clave Alternativa o Secundaria
- Obligatoriedad
- Restricciones de verificación o validación
- Restricciones de aserción
- Disparadores
- Restricciones de Integridad Referencial

2.4.3.1 Clave Primaria o Principal

Restricción que permite declarar un atributo o conjunto de atributos como Clave Primaria de una relación.

Las características de la Clave Primaria son:

- No admite valores nulos
- No admite valores duplicados
- Una relación sólo puede tener asignada una restricción PRIMARY KEY

La cláusula SQL que permite especificar esta restricción es: *PRIMARY KEY*

Tabla de Alumnos

id	Nombre	Apellido	DNI
1	Juan	Pérez	12345678A
2	María	Gómez	23456789B
3	Carlos	Rodríguez	34567890C
4	Ana	Fernández	45678901D
5	Lucía	Martínez	56789012E

Id es la clave principal, identifica de forma unívoca a cada alumno de la tabla. A esta columna se le aplicaría la restricción **PRIMARY KEY**. Esta restriccion no se podrá aplicar a ninguna otra columna de la tabla.

2.4.3.2 Clave Alternativa o Secundaria

Claves candidatas que no han sido elegidas como clave primaria. Aún así, identifican de forma unívoca a cada tupla de la relación.

Características de la Clave Alternativa o Secundaria:

- No admite valores duplicados
- Admite valores nulos
- Esta restricción se puede asignar cuantas veces se precise, puesto que una relación puede disponer de varias claves secundarias

La cláusula SQL que permite especificar esta restricción es: *UNIQUE*

Tabla de Alumnos

id	Nombre	Apellido	DNI
1	Juan	Pérez	12345678A
2	María	Gómez	23456789B
3	Carlos	Rodríguez	34567890C
4	Ana	Fernández	45678901D
5	Lucía	Martínez	56789012E

DNI sería clave secundaria o alternativa, identifica de forma unívoca a cada alumno de la tabla, no admite valores repetidos y admite valores nulos. A este tipo de columnas se le asigna la restricción **UNIQUE**. Es posible que una tabla tenga varias claves secundarias, por lo que la restricción UNIQUE se puede asignar las veces que sea necesaria. Igualmente es posible que una tabla no tenga clave secundaria, por lo que no habría que especificar dicha restricción.

2.4.3.3 Obligatoriedad

Prohíbe que uno o varios atributos de una relación tome valores nulos

Características de la Restricción de Obligatoriedad:

- Obliga al usuario a introducir un valor en dicho atributo, en caso contrario impide dar de alta la tupla o registro.
- Se puede asignar tantas veces como sea necesario

La cláusula SQL que permite especificar esta restricción es: *NOT NULL*

Tabla de Alumnos

id	Nombre	Apellido	DNI
1	Juan		12345678A
2	María	Gómez	23456789B
3	Carlos	Rodríguez	34567890C
4	Ana	Fernández	45678901D
5	Lucía	Martínez	56789012E

En el ejemplo anterior **DNI** tiene asignada la restricción **NOT NULL** puesto que no admite dar de alta un alumno que no disponga DNI. En cambio, la columna **Apellido** no tiene asignada la restricción de obligatoriedad, puesto que el apellido de Juan no se dispone.

2.4.3.4 Restricciones de verificación

Permite especificar condiciones que deban cumplir los valores de los atributos a los que se les asigne dicha restricción. Cada vez que se realice una inserción o actualización de datos se comprueba si los valores cumplen la condición y se rechaza la operación si no se cumple.

La cláusula SQL que permite especificar esta restricción es: *CHECK*

Tabla Alumnos

id	Nombre	Apellido	DNI	Población
1	Juan	Pérez	12345678A	Ubrique
2	María	Gómez	23456789B	El Bosque
3	Carlos	Rodríguez	34567890C	Puerto Serrano
4	Ana	Fernández	45678901D	Ubrique
5	Lucía	Martínez	56789012E	El Bosque

En la columna población se establece la condición de que los alumnos matriculados sólo puedan ser de Ubrique, El Bosque, Prado del Rey, Puerto Serrano y Villamartín. A la columna población habría que aplicarle la restricción **CHECK** y establecer la condición indicada anteriormente. En nuestro caso todas las tuplas cumplen con dicha restricción.

2.4.3.5 Restricciones de aserción

La aserción funciona de forma similar a la verificación CHECK, pero en este caso la condición puede afectar a varias relaciones. La condición se establece sobre elementos de distintas relaciones.

El comando SQL que permite especificar esta restricción es: *CREATE ASSERTION*

2.4.3.6 Disparadores

Las restricciones anteriores son declarativas, sin embargo, este tipo es procedimental.

El usuario podrá declarar un Procedimiento, Disparador o Trigger, que se ejecutará cuando se ejecute en el sistema una determinada acción sobre una relación, esa acción podrán ser del tipo: INSERT, DELETE o UPDATE.

El comando SQL que permite crear un disparador es: *CREATE TRIGGER*

2.4.4 Regla de Integridad Referencial

La última regla de integridad se aplica a las claves ajenas: si en una relación hay alguna clave ajena, sus valores deben coincidir con valores de la clave primaria a la que hace referencia, o bien, deben ser completamente nulos.

Ejemplo Jugadores y Equipos

Tabla equipos

id	nombre	ciudad	año_fundacion
1	Los Halcones	Ciudad Alfa	1998
2	Las Águilas	Ciudad Beta	2001
3	Los Tigres	Ciudad Gamma	1995

Tabla jugadores

id	nombre	posición	equipo_id
1	Juan Pérez	Delantero	1
2	Carlos García	Defensa	1
3	Luis Rodríguez	Portero	2
4	Andrés Fernández	Mediocampo	2
5	Javier Sánchez	Delantero	3
6	Daniel Ramírez	Defensa	3

En la base de datos presentada en el apartado anterior hay una clave ajena que es equipo_id.

Así, a través de la clave ajena JUGADORES.equipo_id se puede conocer los datos del equipo al que pertenece cada jugador, para ello es necesario que el valor asignado a equipo_id, coincida con alguno de los valores de la clave principal id de la relación EQUIPOS, de esta forma puedo

concoer el nombre del equipo, la ciudad a la que pertenece el equipo y el año de fundación de dicho equipo.

Pues bien, la **regla de integridad referencial** exige que los valores que aparecenen la clave ajena **JUGADORES.equipo_id** estén relacionados con algún valor de **EQUIPOS.id**.

La regla de integridad referencial se enmarca en términos de estados de la base de datos, indica lo que es un estado ilegal, pero no dice cómo puede evitarse. Por lo tanto, una vez establecida la regla, hay que plantearse qué hacer si estando en un estado legal, llega una petición para realizar una operación que conduce a un estado ilegal.

Existen dos opciones: rechazar o aceptar la operación y realizar operaciones adicionales compensatorias que conduzcan a un estado legal.

Para hacer respetar la integridad referencial se debe contestar, para cada clave ajena, a las tres preguntas que se plantean a continuación y que determinarán su comportamiento:

2.4.4.1 Regla de los nulos

«¿Tiene sentido que la clave ajena acepte nulos?»

2.4.4.2 Regla de borrado

«¿Qué ocurre si se intenta borrar la tupla referenciada por la clave ajena?»

- ***Restringir (RESTRICT)***: no se permite borrar la tupla referenciada.
- ***Propagar (CASCADE)***: se borra la tupla referenciada y se propaga el borrado a las tuplas que la referencian mediante la clave ajena.
- ***Anular (SET NULL)***: se borra la tupla referenciada y las tuplas que la referenciaban ponen a nulo la clave ajena (sólo si acepta nulos).
- ***Valor por defecto (SET DEFAULT)***: se borra la tupla referenciada y las tuplas que la referenciaban ponen en la clave ajena el valor por defecto establecido para la misma.

2.4.4.3 Regla de modificación

«¿Qué ocurre si se intenta modificar el valor de la clave primaria de la tupla referenciada por la clave ajena?»:

- ***Restringir (RESTRICT)***: no se permite modificar el valor de la clave primaria de la tupla referenciada.
- ***Propagar (CASCADE)***: se modifica el valor de la clave primaria de la tupla referenciada y se propaga la modificación a las tuplas que la referencian, mediante la clave ajena.
- ***Anular (SET NULL)***: se modifica la tupla referenciada y las tuplas que la referenciaban ponen a nulo la clave ajena (sólo si acepta nulos).
- ***Valor por defecto (SET DEFAULT)***: se modifica la tupla referenciada y las tuplas que la referenciaban ponen en la clave ajena el valor por defecto establecido para la misma.

2.4.4.4 Ejemplo

Así, en el caso del esquema de la base de datos presentada en el apartado anterior, debemos determinar las reglas de comportamiento para cada clave ajena. Por ejemplo, para la clave ajena JUGADORES.equipo_id se ha escogido el siguiente comportamiento:

- **Regla de los nulos**:
  - *la clave ajena **acepta nulos***, por lo que es posible encontrar jugadores que no pertenezcan a ningún equipo. Es decir, en la clave ajena JUGADORES.equipo_id se le puede asignar el valor NULL.
  - *La clave ajena **no acepta valores nulos,*** no es posible asignar el valor NULL a JUGADORES.equipo_id, es decir, todos los jugadores han de pertenecer a un equipo.
- **Regla de borrado**:
  - *SET NULL*. En caso de que se elimine un equipo con jugadores asignados a dicho equipo en la clave ajena JUGADORES.equipo_id se le asignará NULL.
  - *RESTRICT*. En caso de que desee elimnar un equipo con jugadores asignados a dicho equio, la operación no va a ser permitida.
  - *SET DEFAULT*. En caso de que desee eliminar un equipo con jugadores asignados a dicho equipo en la clave ajena JUGADORES.equipo_id se le asignará un valor por defecto por ejemplo 1.
  - *CASCADE*. En caso de que desee eliminar un equipo con jugadores asignados a dicho equipo, todos esos jugadores quedarán también eliminados de la tabla. Hay que ser muy pruedentes a la hora de usar esta política de borrado.
- **Regla de modificación**:
  - SET NULL. En caso de que se desee modificar un valor de la clave EQUIPOS.id, el valor de la clave ajena JUGADORES.equipo_id relacionado se le asignará el valor NULL.
  - RESTRICT. En caso de que se desee modificar un valor de la clave EQUIPOS.id, y dicho valor está relacionado con uno o varios valores de la clave ajena JUGADORES.equipo_id, la operación será CANCELADA.
  - SET DEFAULT. En caso de que se desee modificar un valor de la clave EQUIPOS.id, y dicho valor está relacionado con uno o varios valores de la clave ajena JUGADORES.equipo_id, se les asignará un valor por defecto, por ejemplo, el 1.
  - CASCADE. En caso de que se desee modificar un valor de la clave EQUIPOS.id, a todos los jugadores relacionados con dicho equipo se les actualizará también la clave ajena JUGADORES.equipo_id

Finalmente, para representar el **Grafo Relacional** debemos tener en cuenta las siguientes reglas:

- **Claves Principales**: El atributo o conjunto de atributos que formen parte de la clave principal se mostrarán con el carácter #, en negrita y subrayado.
  #id
- **Claves Secundarias**: el atributo o conjunto de atributos que formen parte de una clave secundaria o alternativa se mostrarán con doble subrayado.
  dni
- **Atributos Opcionales**: el atributo seguido de un asterisco.
  email-*

- **Claves Ajenas**: los atributos o conjunto de atributos que representan una clave ajena se mostrarán con punto suspensivos, además debajo se indicará el nombre de la Relación Referenciada.
  usuario_id

  FACTURA.usuario_id -> USUARIOS.id()

- **Opciones de Borrado y Modificación**: Junto a cada clave ajena se mostrarán las opciones de borrado y mdoficación mediante las siguientes letras:
  - M:C, M:N, M:D, M;R (modificación cascasda, modificación nula, modificación defecto, modificación restringida)
  - B:C, B:N, B:D, B:R (borrado cascasda, borrado nulo, borrado defecto, borrado restringido)

  usuario_id

  FACTURA.usuario_id -> USUARIOS.id(NULL; B:R, M:C)

Aplicando las reglas anteriores el grafo relaciona defintivo de la base de datos VENTAS quedaría de la siguiente forma:

```
CLIENTES(#codcli, nombre, dni, dirección, codpostal, codpue)

      CLIENTES.codpue -> PUEBLOS (null; B:R; M:C)

VENDEDORES(#codven, nombre, dirección, codpostal, codpue, codjefe)

      VENDEDORES.codpue -> PUEBLOS (null; B:R; M:C)
      VENDEDORES.codjefe -> VENDEDORES (null, B:R ,M:C)

PUEBLOS(#codpue, nombre, codpro)

      PUEBLOS.codpro -> PROVINCIAS (null; B:R; M:C)

PROVINCIAS(#codpro, nombre)

ARTÍCULOS(#codart, descrip, precio, stock, stock_min, dto*)

FACTURAS(#codfac, fecha, codcli, codven, iva, dto)

      FACTURAS.codcli -> CLIENTES (not null; B:R; M:C)
      FACTURAS.codven -> VENDEDORES (null, B:R, M:C)

LÍNEAS_FAC(#codfac, #línea, cant, codart, precio, dto)

      LINEAS_FAC.codfac -> FACTURAS (not null; B:R; M:C)
      LINEAS_FAC.codart -> ARTICULOS (null, B:R, M:C)
```