UD3. MODELO RELACIONAL (MR)

ÍNDICE

Saber que hacen solo

saber que es

Solo saber que hay hasta 5 y que con 3 está formalizada

1. INTRODUCCIÓN

Historia:

- **1970:** El **Modelo Relacional** (en adelante **MR**) fue propuesto por el Dr. E.F. Codd.
  En aquel momento los SGBD que dominaban el mercado eran de tipo Codasyl (en red) y jerárquico, pero no habían logrado vencer el inconveniente de la dependencia entre las aplicaciones y las estructuras de los datos.
- **1976:** es importante recordar que el ER fue posterior, Peter Chen lo propuso en 1976.

Principal objetivo del MR:

A diferencia de los modelos en red y jerárquico, **el MR** se propone como **principal objetivo**, **aislar al usuario de las estructuras físicas de los datos**, consiguiendo así la independencia entre aplicaciones y datos (finalidad perseguida desde los inicios de las BBDD en los años 60).

Teoría matemática del MR:

Este modelo de datos tiene una amplia teoría matemática que lo respalda: **la Teoría Matemática de las Relaciones**.

Los **avances** que presenta el MR frente a los modelos anteriores son:

- **uniformidad:**
  - los usuarios de la **BD Relacional** (en adelante **BDR**) la ven como una colección de tablas (relación matemáticamente).
  - **La relación** (que representamos como una tabla) es la estructura fundamental del modelo que le proporciona uniformidad.
- **sencillez:**
  - El concepto de tabla como un conjunto de filas y columnas es fácil de entender.
  - Emplea lenguajes de tratamiento de datos muy sencillos.
- **sencilla fundamentación teórica:** el modelo basado relacional está basado en:
  - en el álgebra relacional (AR), y en,
  - el cálculo relacional de tuplas (CRT) *(una tupla es un fila)*.
- **independencia de la interfaz de usuario:**
  - **los lenguajes relacionales, al manipular conjuntos de registros** proporcionan gran independencia respecto al modo en que están almacenados los datos.

Ejemplos de Sistemas Gestores de Bases de Datos Relacionales (SGBDR):

A pesar de que desde que apareció el MR (1970) se convirtió en tema de investigación principal de Bases de Datos, los primeros Sistemas Gestores de Bases de Datos Relacionales (en adelante SGBDR) tardaron en aparecer debido, sobre todo, a su difícil implementación (aún así, incluso a los actuales les faltan algunas de las características del modelo relacional teórico de Codd).

> **Ejemplo:** Algunos ejemplos de SGBDR son Oracle, Sybase, MS Access, MS SQLServer, Informix, DBase, Paradox, mySQL, mariaDB, Postgres, LiteSQL…

Partes de un modelo de datos:

Todo modelo de datos es una combinación de **3 componentes** agrupados en 2 partes, la **estática** y la **dinámica**.

PARTE ESTÁTICA	PARTE DINÁMICA o MANIPULATIVA
- estructuras de datos - reglas o normas	- operadores
En esta ***primera parte de la unidad*** vamos a profundizar en la **Parte Estática** del MR	En la ***segunda parte de la unidad*** abordaremos la **Parte dinámica (operadores del MR)** más detalladamente

2. ESTRUCTURA DE DATOS

- La estructura básica y única del MR es la **relación**, también llamada informalmente tabla, que sirve para representar tanto los objetos como las asociaciones entre ellos. *Podemos decir que la tabla es la representación de la relación en el MR.*
- Los **atributos** son las propiedades de las relaciones y se definen sobre los dominios.

A continuación veremos los distintos elementos del modelo.

2.1. DOMINIOS, ATRIBUTOS Y RELACIONES

El **Universo del Discurso** (**UD**), representación del mundo real o también llamada **Minimundo** de una BDR, está compuesto por un conjunto de **dominios {Di}** y de **relaciones {Ri}** definidas sobre esos dominios.

Dominios:

- **Dominio (D):** Conjunto de datos del mismo tipo.
  - Cada dominio tiene un **nombre** y un **formato**.
  - El **formato** puede ser definido de 2 maneras:
    - por **extensión** (dando todos sus valores), o,
    - por **intensión** (mediante un tipo de datos). ENT. SIN DR
  - A veces se asocia al dominio su unidad de medida (kilos, metros, etc.) y ciertas restricciones (como un rango de valores).

Ejemplo: Algunos ejemplos de dominios con su nombre y formato son:

NOMBRE DEL DOMINIO	FORMATO DEL DOMINIO
Módulos	BD/ LMSXI/SI/Prog/FOL/CD/DIW/DWC/DWS/DAW/EIE/FCT/Proyecto
TiposDocId	NIE/NIF/Pasaporte/DocUE
Salarios	600 - 6.000

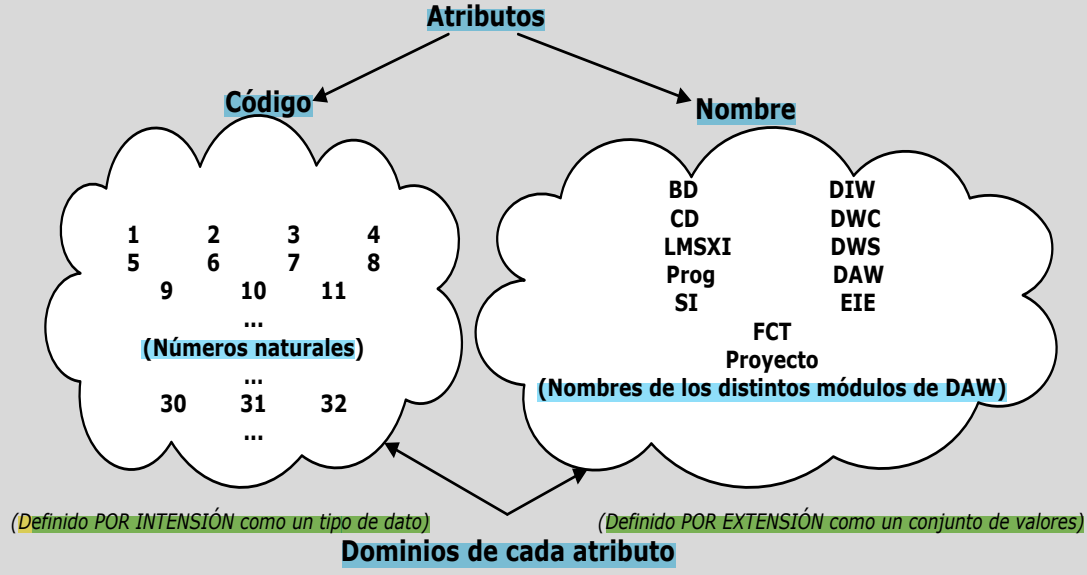
Los dominios se pueden clasificar en:

- **generales:** toma valores entre un mínimo y un máximo. En el ejemplo anterior el dominio *salarios*.
- **restringidos:** toma valores entre unos específicos. En el ejemplo anterior, los dominios *módulos* y *tiposDocId*.

Atributos:

- **Atributo (A):** Interpretación de un determinado dominio en una relación, es decir, *el papel que desempeña en esa relación.*
  Su representación será como sigue: si D es el dominio de A se representará del siguiente modo:
  D= Dom(A)

Ejemplo: Para la relación (tabla) **MÓDULO (Código, Nombre)**:



Un atributo y un dominio **PUEDEN** llamarse igual, pero debe tenerse en cuenta que:

- un **atributo** *está siempre asociado a una relación (tabla)*:
  - un **atributo** representa una propiedad (una característica) de una relación (tabla)
- un **dominio** existe *sin depender de una relación* (tabla):
  - varios atributos distintos (de la misma o de diferentes relaciones) pueden tomar sus valores del mismo dominio.

Ejemplo: El atributo DNI de la relación ALUMNO y el atributo DNI de la relación PROFESOR, están definidos sobre el mismo dominio de nombre *números_enteros_positivos.*

Relación:

- **Definición formal de relación (R):** Dada una colección de dominios **D1, D2,…,Dn** (no necesariamente distintos), R es un subconjunto del producto cartesiano **D1xD2x…xDn**, de los **n** dominios.
  - Una relación es un conjunto de elementos de la forma **(d1, d2,…, dn)** donde cada **dj** es un valor de un dominio **Dj**.

Vamos a precisar más el concepto de **relación** en:

- **esquema de relación R**, y,
- **relación r(R).**

IMPORTANTE

- **Esquema de relación R (o INTENSIÓN DE LA RELACIÓN):** Es la *definición de la relación.* Es **estática** (no varía con el tiempo, si varía se considera que ha variado el modelo relacional de la BD en estudio).
  Se compone de:
  - un nombre de relación **R**,
  - un conjunto de n atributos **{Ai}** y
  - un conjunto de n dominios (no necesariamente distintos) **{Di}**, dónde cada atributo será definido sobre un dominio:
    R(A1:D1, A2:D2, …, Ai:Di, …, An:Dn)
    donde para **Ai:Di, Ai** es un atributo definido sobre el dominio **Di**.

- **Relación r(R) (o EXTENSIÓN DE LA RELACIÓN):** Es un conjunto de m elementos llamados **tuplas {tj}**. Cada tupla (fila en la tabla) es un conjunto de pares del tipo:
  (<A1:V1j>, …, <Ai:Vij>, …, <An:Vnj>)
  donde cada Ai es el nombre de un atributo y Vij es un valor del correspondiente dominio Di sobre el que está definido el atributo:
  r(R) = tj {(<A1:V1j>, … <Ai:Vij>, …, <An:Vnj>):Vij Є Di}

  Es lo que se conoce como **extensión de la relación:** conjunto de tuplas (filas de la tabla) que, en un instante determinado, satisfacen el esquema de relación y se encuentran almacenadas en la BD.
  Es **dinámica**, varía en el tiempo, ya que en distintos momentos el contenido de la tabla puede ser diferente.

  > **IMPORTANTE:** *A veces* usamos el término **relación** en sentido genérico para referirnos a un esquema de relación con sus posibles extensiones.

  Diremos entonces que una **BD Relacional** viene definida por:
  - su **intensión**, llamada **esquema relacional** y compuesta por una colección de esquemas de relación que describen un determinado Universo del Discurso (UD) o minimundo, y por,
  - la **extensión de su esquema relacional**, constituida por una colección de relaciones con sus tuplas.

- **Representación de una relación:** En el modelo relacional una relación se representa usando una TABLA, donde:
  - las columnas son los atributos (propiedades de la relación). El número de atributos se llama **GRADO** de la relación, y,
  - cada fila o tupla, es un elemento del conjunto que constituye la relación. El número de tuplas se llama **CARDINALIDAD** de la relación.

  > **NOTA:** No confundir el término **grado** referido a las *interrelaciones* del EER con el **grado** de una *relación* en el MR. En el EER hacía referencia al número de entidades que participaban en la interrelación, y en el MR hace referencia al número de atributos de una relación.

IMPORTANTE: No confundir **tabla** con **relación**, porque:

- la tabla es una forma de representar la relación,
- una relación tiene unas propiedades intrínsecas que no tiene la tabla, y que se derivan de la definición matemática de relación, ya que al tratarse de un conjunto, en una relación:
  - no pueden existir 2 tuplas iguales,
  - el orden de las tuplas no es importante,
  - el orden de los atributos no es importante,
  - cada atributo sólo puede tomar un único valor del dominio sobre el que se define.

Ejemplo: Relación **MÓDULO**:

- **Intensión o esquema** de la relación:
  MÓDULO(código:códigos, nombre:nombres, n_sesiones:numeros, curso:cursos)
- **Extensión** de la relación:

código	nombre	n_sesiones	curso
1	GBD	225	1
2	PAR	256	1
3	FOL	129	1
4	ISO	256	1
5	ASGBD	84	2

***Grado de la relación*= 4** (número de atributos) ***Cardinalidad*= 5** (número de tuplas)

2.2. CLAVES

- **Clave primaria (C.P.):** Conjunto **mínimo** de atributos cuyos valores identifican unívocamente a una tupla en una relación.

  Más formalmente, una clave primaria de una relación **R**, es un conjunto de atributos **K** tales que:
  - para cualesquiera tuplas **t1** y **t2** se verifica que **t1(K) <> t2(K)**, y además,
  - no existe ningún subconjunto de **K** que cumpla la condición anterior.

  Ejemplo: Supongamos la relación **R(AT,DF)**, siendo **AT** el conjunto de atributos de **R**, y **DF**, el conjunto de dependencias funcionales (*se verán al final de esta misma unidad*).

  Si **AT = {A,B,C,D}** y **C.P. = {A,B}** → **A no puede ser clave por sí sola**, **B no puede ser clave por sí sola**

- **Clave candidata o alternativa (C.C.):** Para una relación **R** puede que existan dos conjuntos de atributos diferentes **K1** y **K2** que me permitan identificar unívocamente a cada tupla de la relación.

  K1 y **K2** serán claves candidatas si las dos pueden ser claves primarias. La elección de **K1** ó **K2** como clave primaria se basará en el criterio del diseñador de la BD.

  Puede haber + de 1 ambas

- **Clave foránea o ajena (C.F.):** Es un atributo o combinación de atributos de una relación, cuyos valores o son nulos, o bien toman el valor de la clave primaria de alguna otra relación.

3. REPRESENTACIÓN

3.1. GRAFO RELACIONAL

Para la representación del modelo relacional usaremos el **grafo relacional** (desarrollado por *Smith* en 1985 y usado por expertos en diseño de BD como *Adoración de Miguel* y *Mario Piattini*). Las normas a seguir para diseñar cualquier MR utilizando el **grafo relacional** son las siguientes:

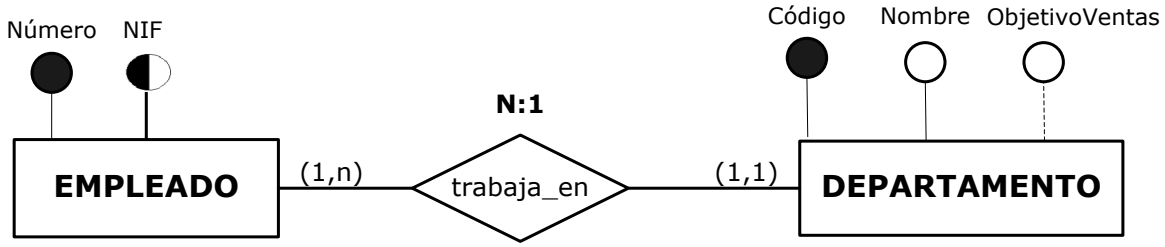
- El nombre de las relaciones (tablas) está representado en MAYÚSCULAS y **negrita**.
- Se indica, para cada relación (tabla), su nombre y entre paréntesis todos sus atributos, y según su tipo se hará lo siguiente:
  - las claves primarias se subrayan,
  - las **claves candidatas** se ponen en negrita,
  - las *claves foráneas* en cursiva, y referencian con una flecha a la relación en la que son clave primaria,
  - los atributos opcionales, que pueden tomar valores nulos aparecen con un asterisco como superíndice del nombre del atributo (atributo_opcional*).
- Un mismo atributo puede tener varias propiedades a la vez, por ejemplo, ser clave foránea y opcional. Para indicarlo combinaremos la simbología que se utiliza por separado para cada tipo de atributo, en este caso el atributo quedaría así: *nombre_clave_foránea_opcional**.
- Además, en cada clave foránea debemos indicar el tipo de **BORRADO** y **MODIFICACIÓN** que se produce. Tanto el borrado como la modificación pueden ser de **4 tipos**:
  - **EN CASCADA (CASCADE)**
    Borrado en Cascada: Cuando se borra una tupla en la "relación origen del atributo" se borra en la relación en la que es clave foránea toda la información asociada. *Representación*: **B:C**
    Modificación en Cascada: Cuando se modifica la C.P. en una tupla en la "relación de origen del atributo" se actualiza al nuevo valor en la relación en la que es clave foránea. *Representación*: **M:C**
  - **RESTRINGIDO (opción por defecto, es decir, si no se indica el tipo se asume este) (RESTRICT ó NO ACTION)**
    Borrado Restringido: Cuando se intenta borrar una tupla en la "relación origen del atributo" no se permite borrar si en la relación en la que es clave foránea existe información asociada. *Representación*: **B:R**
    Modificación Restringida: Cuando se intenta modificar la C.P. en una tupla en la "relación origen del atributo" no se permite hacerlo si en la relación en la que es clave foránea existe información asociada a esa tupla que queremos modificar. *Representación*: **M:R**
  - **CON PUESTA A NULOS (SET NULL)**
    Borrado con puesta a Nulos: Cuando se borra una tupla en la "relación origen del atributo" en la relación en la que es clave foránea se pone a NULL la clave foránea para aquellas tuplas asociadas a la tupla borrada. *Representación*: **B:N**
    Modificación con puesta a Nulos: Cuando se modifica la C.P. en una tupla en la "relación origen del atributo" se actualiza al nuevo valor en la relación en la que es clave foránea a NULL. *Representación*: **M:N**
  - **CON PUESTA A VALOR POR DEFECTO (SET DEFAULT)**
    Borrado con puesta a valor por Defecto: Cuando se borra una tupla en la "relación origen del atributo" en la relación en la que es clave foránea tomarán el valor por defecto en la clave foránea todas aquellas tuplas asociadas a la tupla borrada. *Representación*: **B:D**

Modificación con puesta a valor por Defecto: Cuando se modifica la C.P. en una tupla en la "relación origen del atributo" se actualiza al nuevo valor en la relación en la que es clave foránea al valor por defecto de la clave foránea. *Representación*: **M:D**

En una misma clave foránea el **tipo de Borrado puede ser diferente al tipo de Modificación** (véase ejemplo siguiente).

Los Borrados y Modificaciones, están muy relacionados con la **integridad referencial** que se verá en el apartado **4.2. Restricciones semánticas**.

Ejemplo: Veamos el **grafo relacional** que se obtiene dado el siguiente EER:



EMPLEADO (Número, **NIF**, *cod_departamento*)

B:R, M:C

DEPARTAMENTO (Código, Nombre, ObjetivoVentas*)

3.2. TERMINOLOGÍA

Existen diferentes terminologías usadas en el MR, según la menor o mayor *formalidad* empleada:

TÉRMINO FORMAL (los únicos correctos en el MR)	TÉRMINO INFORMAL	PARA FICHEROS
Relación	Tabla	Fichero
Atributo	Columna ó campo	Campo
Tupla	Fila	Registro
Grado	Número de columnas	Número de campos
Cardinalidad	Número de filas	Número de registros
Clave Primaria	Identificador único	Clave
Dominio	Conjunto de valores posibles	Tipo de dato
Esquema de relación	Descripción de la tabla	Formato del fichero

4. RESTRICCIONES

Las reglas, normas o restricciones relacionales (del MR) se pueden clasificar en **reglas o restricciones inherentes** y **reglas o restricciones semánticas (o de usuario)**.

4.1. RESTRICCIONES (O REGLAS) INHERENTES

Son aquellas **derivadas de la misma estructura del modelo**, que no tienen que ser definidas por el usuario, y nos imponen limitaciones a la hora de modelar nuestro *minimundo*.

Las **restricciones inherentes del MR** son las siguientes:

1) En una relación no puede haber tuplas iguales **=> RESTRICCIÓN DE OBLIGATORIEDAD DE CLAVE PRIMARIA.**
2) El orden de las tuplas y el de los atributos no es relevante.
3) Cada atributo sólo puede tomar un único valor del dominio sobre el que está definido.
4) Ningún atributo que forme parte de la clave primaria de una relación puede tomar un valor nulo **=> RESTRICCIÓN DE INTEGRIDAD DE ENTIDAD.**

Ejemplo: Vamos a ver ejemplos de la regla de integridad de entidad en la siguiente relación **EMPLEADO**. La clave primaria es el atributo **numero**:

numero	apellido1	apellido2	nombre	salario(miles)	depto
40	García	López	Carlota	25	615
NULL	Pena	Rodríguez	Ana	18	***NULL***
60	Pena	***NULL***	Ana	18	200

En la 2ª tupla la clave primaria tiene un valor nulo, con lo cual no podríamos distinguir esa fila de la última, no sabríamos en este caso si son el mismo o distintos empleados. Diremos por lo tanto que esta relación ***no satisface la regla de integridad de entidad***.

numero	apellido1	apellido2	nombre	Salario(miles)	depto
40	García	López	Carlota	25	615
50	Pena	Rodríguez	Ana	18	***NULL***
60	Pena	***NULL***	Ana	18	200

En este ejemplo todos los empleados tienen un número de empleado y todos diferentes, por lo tanto podemos decir que esta relación ***sí satisface la regla de integridad de entidad***.

4.2. RESTRICCIONES SEMÁNTICAS O DE USUARIO

Son facilidades proporcionadas por el MR para poder recoger en el esquema de relación (definición de la relación), toda la información posible del universo del discurso que estamos modelando.

Son restricciones que tiene que definir el diseñador con el fin de que el esquema de la BDR sea un reflejo lo más fiel posible de la situación del mundo real que estamos intentando resolver (de ahí la importancia de las restricciones en el diseño de las BBDD).

Las **restricciones semánticas del MR** son las siguientes:

1) **RESTRICCIÓN DE CLAVE PRIMARIA (Primary Key):** permite declarar un atributo o conjunto de atributos como la clave primaria de una relación, de tal manera que identifica unívocamente cada tupla de la relación).
2) **RESTRICCIÓN DE UNICIDAD (Unique):** nos permite definir claves alternativas, es decir, nos permite indicar qué atributos de la relación tienen valores diferentes (únicos) en las diferentes tuplas.
3) **RESTRICCIÓN DE OBLIGATORIEDAD (Not Null):** permite establecer aquellos atributos que en una relación deben tomar siempre valores distintos de NULO.

OJO: foraneas: cod_lo_que sea

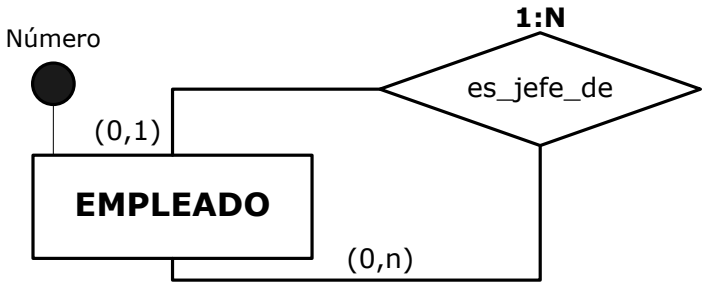
4) RESTRICCIÓN DE CLAVE AJENA ó DE INTEGRIDAD REFERENCIAL (Foreign Key): esta restricción se usa para enlazar relaciones (tablas) de una BD. La **integridad referencial** nos indica que los valores de la *Clave Foránea (C.F.)* de una relación, debe tomar sus valores de la C.P. de la relación de la que procede, o bien, puede tomar valores nulos (si se admiten, hay que estudiar cada caso particular).

claves foraneas

IMPORTANTE: El o los atributos que son *Clave Ajena* (*o foránea*) en una relación no necesitan tener los mismos nombres que en su *relación de origen* (donde eran C.P.), pero sí **necesitan estar definidas con el mismo tipo de dato**.

Nosotros, *para unificar la nomenclatura usada*, como norma general a las claves foráneas las nombraremos con la abreviatura de la CP a la que hacen referencia y el nombre de la tabla de la que proceden. Si fuese necesario se le añadiría un sufijo con una descripción del contenido del atributo.

Por ejemplo, atendiendo al siguiente EER, en la relación EMPLEADO se debe agregar un atributo con el número del empleado jefe. A ese campo, según el criterio explicado en el párrafo anterior deberíamos llamarle *num_empleado,* pero para aclarar el contenido de ese campo le llamaremos *num_empleado_jefe.*



5) RESTRICCIÓN DE VERIFICACIÓN (Check):

Puede ocurrir que sea preciso especificar una condición a cumplir por los valores de determinados atributos de una relación de la BD. Van ligadas a un único elemento de la BD, a un atributo o a una relación.

Estas restricciones permiten especificar una CONDICIÓN a cumplir por un elemento de la base de datos. Esto provocará que durante una operación de actualización (inserción/borrado/modificación) de la relación, se compruebe la condición, y si ésta NO se verifica, se provoca el RECHAZO de la operación.

De ahí que tanto ésta como la restricción siguiente (aptdo. 6) se conozcan como ***RESTRICCIONES DE RECHAZO***. Es decir,

si no se cumple la condición de verificación

↓

RECHAZO de la INSERCIÓN/BORRADO/MODIFICACIÓN.

Ejemplo, fecha fin tiene que ser igual o mayor y no aceptar otro valor que no cumpla esto

Ejemplo: Supongamos la relación EMPLEADO que tiene un atributo *salario* en el que se almacena el sueldo que cobra cada empleado. Si definimos para ese atributo la **restricción de verificación** de que el salario de un empleado debe estar entre 600 € y 3.000€, cuando:

- intentemos insertar un empleado nuevo con salario <600€ o >3.000€, la verificación establecida me lo impedirá, es decir, **se rechazará la operación de inserción**;
- intentemos actualizar el salario de un empleado que cobraba 2.000€ a 3.500€, la verificación establecida me lo impedirá, es decir, **se rechazará la operación de modificación.**
- En este caso el borrado de filas no se ve afectado por esta restricción.

Para este ejemplo concreto, la restricción de verificación en lenguaje SQL se escribiría del siguiente modo **CHECK Salario BETWEEN 600 AND 3000**. *(Esta sintaxis se verá en la unidad de trabajo de Lenguaje DDL).*

Solo en la misma tabla, para involucrar otras tablas assertion

6) **RESTRICCIÓN DE ASERCIÓN ó ASERTOS (Assertion):** Generaliza la anterior, la verificación, porque la condición se establece sobre más de un elemento de la BD, es decir, afectan a varias relaciones. Además las aserciones siempre han de tener un nombre, y no así las verificaciones, a las que se les puede dar un nombre también, aunque no obligatoriamente.

RESTRICCIÓN DE RECHAZO

Su **funcionamiento** es el mismo que el de la verificación: durante una operación de actualización *de cualquiera de las relaciones implicadas*, se comprueba la condición de la aserción, y si ésta NO se verifica, se provoca el RECHAZO de la operación.

NOTA: En SQL los asertos adoptan la forma:

CREATE ASSERTION *nombre_aserto* **CHECK** *predicado*

7) **DISPARADORES (Trigger):** A veces puede interesar *especificar una acción distinta al rechazo* cuando no se cumple una determinada restricción semántica. Es decir, al contrario que en las **restricciones de rechazo** (verificación y aserción), en las que se rechaza la operación de actualización si *NO se cumple la condición*; en el caso de los disparadores, la acción especificada en el trigger se ejecuta siempre y cuando **SÍ se cumpla la condición**.

salta el trigger

IMPORTANTE: Además de la **condición** permite indicar la **acción**, por ello se pueden interpretar como:

reglas de EVENTO-CONDICIÓN-ACCIÓN ó **reglas ECA**, ya que:

cuando se produce un **Evento** (suceso) - si se cumple una **Condición** - se realiza una **Acción**

Ejemplo: Un ejemplo de trigger o disparador sería el siguiente:

Informar al administrador de la BD (DBA ó ADB) cuando haya un empleado que gane más de 2.800€ y que trabaje en el departamento de ventas.

Los evento, condición y acción para este disparador serían los siguientes:

- **evento:** Inserción/Modificación en la relación EMPLEADO
- **condición:** SI salario>2.800€ Y departamento=”Ventas”
- **acción:** Informar al DBA

se inserta igual pero avisa de los datos introducidos si no están entre las restricciones.

TRIGER NO ES DE RECHAZO

También puedes decirle que modifque otro campo d euna tabla, insertas una filta y el trigger actulaiza el nº de stock.

5. REGLAS DE TRANSFORMACIÓN DE UN ESQUEMA E/R A UN ESQUEMA RELACIONAL

5.1. ENTIDADES

Cada entidad se transforma en una relación (tabla) con el mismo nombre que la entidad del modelo EE-R.

Si la entidad es **débil** se transforma en una tabla que tendrá por clave foránea la clave primaria de la fuerte. Si la *dependencia es en identificación*, la clave foránea formará parte de la clave primaria de la débil.

5.2. ATRIBUTOS

- **Univaluados:** Pasan a ser campos o atributos en la tabla.
- **Multivaluados:** Atributos que pueden tomar más de un valor. Existen 2 posibilidades:

1. Si el atributo toma pocos valores (hasta 3), por ejemplo **Teléfonos** (fijo o móvil) se crea un atributo en la relación por cada valor.
2. Si toma muchos valores, como por ejemplo el atributo **Proveedor** de la entidad Producto, dará lugar a una tabla nueva, en este caso para los proveedores.

- **Obligatorios:** Atributos con restricción **NOT NULL**.
- **Identificador Principal (I.P.):** Pasan a ser la clave primaria de la relación.
- **Identificador Alternativo (I.A.):** Pasan a ser claves candidatas.
- **Compuestos:** Son aquellos que son un agregado de distintos dominios. Debemos elegir lo que más nos interese según el modo en el que se va a recuperar la información:

1. Dejar sólo el atributo compuesto (por ejemplo **FechaNacimiento**).
2. Eliminar el atributo compuesto y dejar como atributos sus componentes, por ejemplo para una **dirección** al pasar a MR la pasamos como los atributos **calle**, **número, piso, localidad, codpostal** y **provincia**.

- **Derivados:** En el MR pasan a ser atributos normales y además se necesita un disparador que en cada (modificación o inserción) calcule el valor del atributo. O bien se puede optar por no implementarlos y se crean procedimientos que calculen el valor del atributo derivado cuando se necesite esa información.

5.3. DEPENDENCIAS

- **En EXISTENCIA:** Se propaga la clave primaria (C.P.) de la entidad fuerte a la débil, pasando a ser clave foránea (C.F.) en la entidad débil.

  Ejemplo: Sería la dependencia en existencia entre la entidad **PADRE** y la entidad **HIJO**, con C.P. los códigos de cada uno.

- **En IDENTIFICACIÓN**: Se propaga la clave primaria (C.P.) de la entidad fuerte a la débil, pasando a ser clave foránea (C.F.) en la entidad débil y además formará parte de la C.P. de la débil.

  Ejemplo: Dependencia en identificación entre la entidad **PELÍCULA** y la entidad **COPIA**. La C.P. de **PELÍCULA** será el CódPelícula y la C.P. de **COPIA** estará formada además de por el NúmCopia por el CódPelícula.

5.4. INTERRELACIONES REFLEXIVAS (UNITARIAS Ó UNARIAS)

En función de su tipo de correspondencia se actuará como si fuese una relación binaria que asocia a 2 entidades iguales.

5.5. INTERRELACIONES BINARIAS

- **1:1 (uno a uno):** Existen *2 posibilidades*: o se propaga la clave o se crea una nueva tabla. La decisión se tomará en función de las cardinalidades mínimas:
  1. **Si todas las entidades que participan tienen cardinalidad (0,1)** conviene *transformarla en una tabla* para evitar valores nulos en la clave propagada. La tabla nueva tendrá, además de los atributos propios (si los tiene), las C.P. de las 2 entidades, AUNQUE SÓLO UNA DE ELLAS FORMARÁ PARTE DE LA C.P. La otra foránea será OBLIGATORIA y ÚNICA.
     Ejemplo: Interrelación *Boda* entre las entidades **HOMBRE** y **MUJER**.
  2. **Si sólo una de las entidades tiene participación parcial (0,1)** y la otra (1,1), *propagaremos la clave de la parte (1,1) a la (0,1)* para evitar tener valores nulos.
     Ejemplo: Ver asociación 1:1 entre las entidades **ACTIVIDAD** y **AULA**.
  3. **Si ambas entidades tienen participación total (1,1)** se propagará la clave de cualquiera de las entidades a la otra. (Elegir a dónde va la clave es tarea del diseñador).
- **1:N (ó N:1) (uno a varios):** Existen dos soluciones posibles:
  1. ***PROPAGACIÓN DE LA CLAVE*** **Primaria de la entidad de la parte 1 a la parte N**. Si existen atributos propios de la interrelación se pasarán como campos nuevos a la tabla de la parte N.
  2. **Transformar la interrelación en una *NUEVA TABLA*** que tendrá por C.P. LA C.P SÓLO DE LA PARTE N DE LA INTERRELACIÓN, aunque además tendrá como C.Foránea o ajena la C.P. de la parte 1. Si existen atributos propios de la interrelación se agregarán a la nueva tabla.

     Optar por crear una nueva tabla o no, dependerá del criterio del diseñador, pero hay casos en los que sería recomendable hacerlo:
     - Cuando el número de ejemplares interrelacionados de la entidad que propaga su clave es muy pequeño, y por lo tanto **existirían muchos valore nulos** en la clave propagada.
     - Cuando se cree que en un futuro la relación puede *pasar a ser* **M:N**.
     - Cuando la interrelación tiene atributos propios y no deseamos propagarlos para **conservar la semántica**.
     - Además, el propagar claves da lugar a la aparición de Claves Foráneas, siempre va a implicar la **tarea de definición de Borrados y Modificaciones**.
- **N:M (ó M:N) (varios a varios):** Pasará a ser una nueva tabla con el nombre de la Interrelación, cuya C.P. estará formada AL MENOS por las claves primarias de las 2 entidades que relaciona. Si la relación tenía atributos propios pasarán a ser campos en la nueva tabla.

IMPORTANTE:
Atributos en interrelaciones: Si *se ha generado una tabla nueva*, los atributos propios de la relación (interrelación) pasarán a ser campos de la tabla nueva.

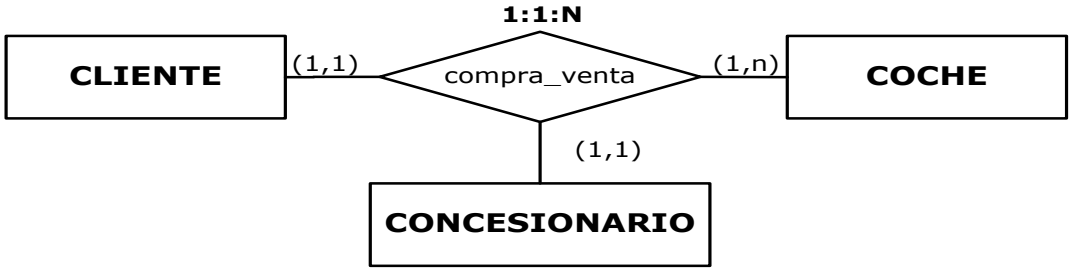
Si por el contrario *se ha propagado la clave*, los atributos propios de la interrelación pasarán a ser nuevos campos de la tabla de la entidad a dónde se ha propagado la clave (**migran con la clave**).

5.6. INTERRELACIONES TERNARIAS

Dan lugar a una NUEVA TABLA que tendrá como campos los atributos propios y además las C.P. de cada entidad que relaciona.

Para decidir la Clave Primaria de la nueva tabla deberemos estudiar las cardinalidades del caso concreto. Si el tipo de correspondencia es N:M:P la C.P. estará formada, al menos por las claves primarias de las entidades que relaciona. Si el tipo de correspondencia tiene algún 1, deberemos estudiar el caso concreto y valorar la descomposición en relaciones de grado menor.

Ejemplo: Interrelación *Venta* entre **CONCESIONARIO**, **COCHE** y **CLIENTE**, para concesionarios dedicados exclusivamente a la venta de coches nuevos. Bastaría para la C.P. con la C.P. del **COCHE**.



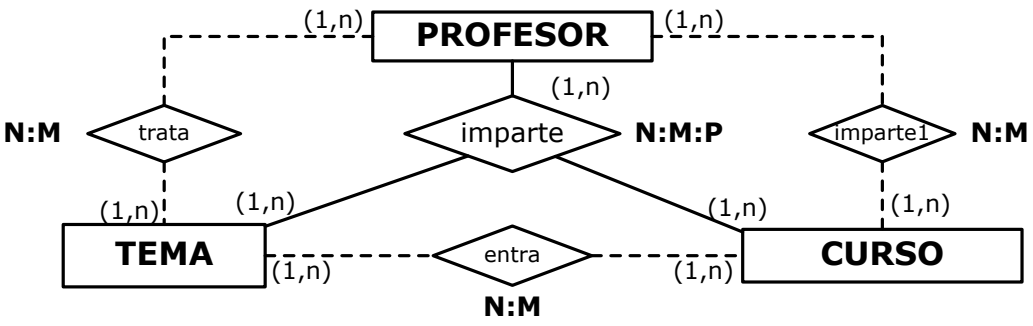
En este caso se considera que cada coche es comprado por un único cliente y que el mismo coche identificado por su nº de chasis (no el mismo modelo) puede ser vendido en distintos concesionarios.

5.6.1. Decisión de hacer una relación ternaria y no 3 binarias

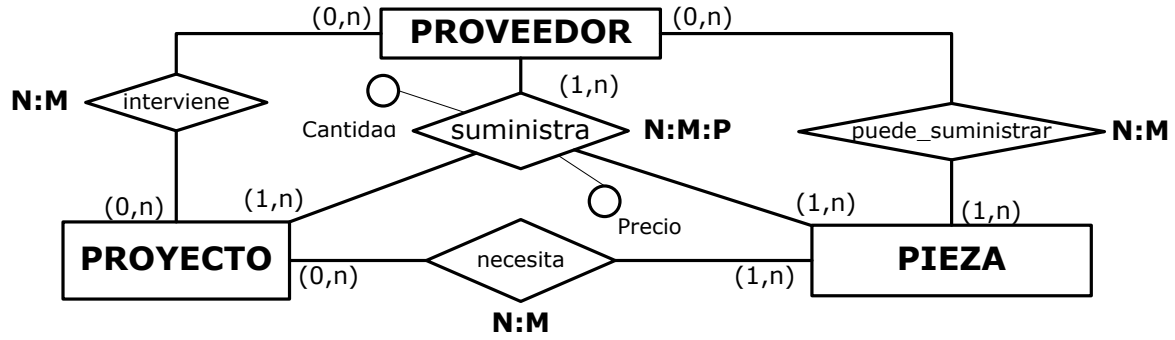
Por norma general, **siempre hay que intentar utilizar interrelaciones de menor grado**, por lo tanto, si en nuestro EER podemos representar la misma información con 3 binarias que con una ternaria, utilizaremos las binarias.

Debemos tener en cuenta que la descomposición de interrelaciones de grado mayor que 2 en otras de grado 2, no siempre es posible, ya que la semántica recogida en una y otra solución no siempre es la misma. Por lo tanto a veces es inevitable usar una ternaria. Para ello veamos el siguiente ejemplo.

Ejemplo: En la siguiente figura la información almacenada en la interrelación **imparte**, que asocia a 3 entidades, se refiere a que un profesor imparte un tema en un curso. Si sustituimos esta interrelación por las tres binarias **imparte1**, **trata** y **entra**, de ellas no se puedes deducir los temas que trata un profesor en un curso determinado, aunque sepamos los cursos que ha impartido ese profesor, qué temas entran en esos cursos y cuáles son los temas que ha impartido ese profesor. Por tanto, no es posible la descomposición de esta interrelación de grado 3 en tres de grado 2 sin pérdida semántica.



IMPORTANTE: Que exista una relación ternaria no impide que entre las entidades que participan en ella puedan existir otras relaciones binarias. Esto se puede observar en el siguiente ejemplo:

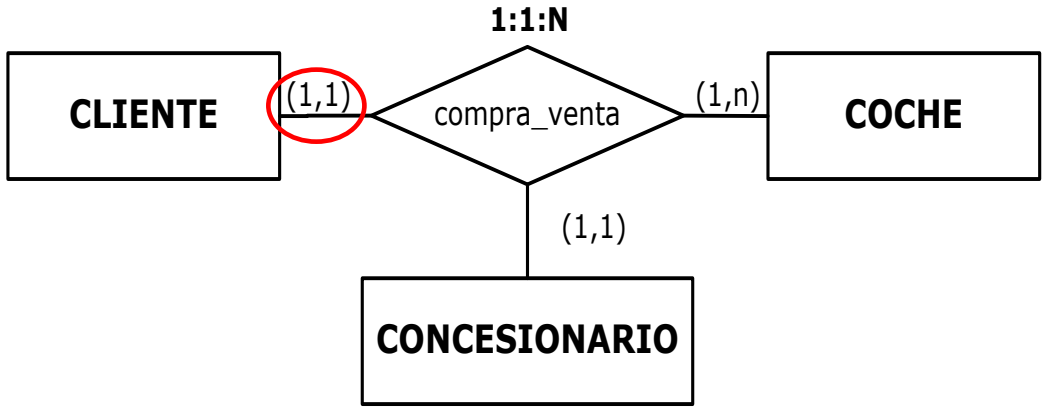


En este ejemplo la interrelación **suministra** de grado 3, coexiste con las 3 interrelaciones de grado2 (**puede_suministrar**, **interviene** y **necesita**), ya que éstas recogen las piezas que puede suministrar un proveedor o para los proyectos que puede suministrar, etc., mientras que la de grado 3 representa las piezas que , de hecho, están siendo suministradas para un cierto proyecto por un determinado proveedor; por tanto la semántica de la interrelación ternaria es distinta de las interrelaciones binarias.

5.6.2. Cardinalidades de mapeo en las interrelaciones ternarias

Tal como se han definido las cardinalidades de mapeo, en una interrelación de grado 3 la cardinalidad de una de las entidades (E1) con respecto a las otras 2 (E2 y E3) es el número mínimo y máximo de ejemplares de E1 que están vinculados con uno de E2 y de E3 ya vinculados en la interrelación.

Ejemplo: Si volvemos al ejemplo del concesionario e intentamos establecer, por ejemplo, la cardinalidad de mapeo de la entidad **CLIENTE**, lo que debemos hacer es preguntarnos el nº mínimo y máximo de clientes que participan en la compra de un coche *ya vendido* en un concesionario determinado.



Lo mismo se haría para el resto de cardinalidades de mapeo de las otras 2 entidades.

5.7. INTERRELACIONES DE GENERALIZACIÓN/ESPECIALIZACIÓN

Cada entidad de la asociación *ES_UN* o *IS_A*, tanto el supertipo como los subtipos, se transforman en tablas.

El atributo discriminante pasará como campo al supertipo.

Los subtipos heredan la Clave primaria del supertipo (será foránea y primaria en los subtipos), además de tener sus propios atributos.

- **Totalidad:** Prohibir inserciones en el supertipo y se crea un **disparador** que ante una inserción en el subtipo inserte en el supertipo. Además el atributo discriminante NO puede tomar valores nulos.
- **Parcialidad:** El atributo discriminante SÍ puede tomar valores nulos.
- **Exclusividad:** Se necesita una **aserción** que antes de insertar compruebe que no está ya en otra tabla.
- **Solapamiento:** Se controla mediante una **aserción** que permita agregar nuevos tipos al discriminante.

6. OPERADORES RELACIONALES

Hasta ahora hemos estudiado la parte estática del MR. La **parte manipulativa o dinámica del MR** está constituida por los operadores del álgebra relacional, desarrollada por Codd.

A partir de aquí, en esta unidad, veremos la fundamentación matemática del MR dedicándole especial atención a:

- las operaciones de **álgebra relacional**, y,
- la **normalización**.

7. FUNDAMENTACIÓN MATEMÁTICA DEL MR

En este apartado y los siguientes veremos **la parte dinámica del MR**, que podemos dividir en:

- **el Álgebra Relacional, y,**
- **la Normalización de Relaciones.**

7.1. EL ÁLGEBRA RELACIONAL

Fue desarrollada por **Codd** como una colección de:

Operaciones formales, que actúan sobre relaciones, y que producen relaciones como resultado.

El **álgebra relacional** es propia del MR y se puede considerar como **UN LENGUAJE DE CONSULTA PROCEDIMENTAL**:

- **de consulta:** porque mediante él se solicita información contenida en la BD,
- **procedimental:** porque es necesario indicar la secuencia de operaciones a realizar sobre la BD para obtener el resultado deseado.

7.2. TIPOS DE OPERACIONES DE ÁLGEBRA RELACIONAL

Operaciones BÁSICAS ó PRIMITIVAS	**Unarias** *(actúa sobre una única relación)*	• Selección ó Restricción (σ) • Proyección (π)
	Binarias *(actúa sobre 2 relaciones)*	• Unión ($\cup$) • Diferencia ($-$) • Producto cartesiano (**X**)
Operaciones DERIVADAS *(se pueden obtener combinando las básicas)*		• Intersección ($\cap$) • Combinación natural ó Join natural (*) • Combinación ó Join (θ)

7.2.1. *Selección ó Restricción (σ)*

where

Construye **una nueva relación (tabla)** tomando un subconjunto de tuplas de una relación existente. Este subconjunto debe satisfacer la **condición** indicada en la selección:

$$R2 = \sigma_F(R1)$$

- **F** es una fórmula o predicado que involucra a:
  - **operadores:**
    - *aritméticos*: < <= = >= > ≠
    - *lógicos*: ∧(and), ∨ (or), ¬(not)
  - **operandos:** *constantes* y *variables*.
- **R1** es la relación (tabla) sobre la que se realiza σ
- **R2** es la relación (tabla) resultante después de σ

IMPORTANTE: En lugar del **símbolo igual**, en la representación de esta y de las siguientes operaciones de álgebra relacional, podría ponerse una flecha del siguiente modo: $R2 \leftarrow \sigma_{F}(R1)$

Ejemplo: En la BD de una **sociedad cultural y deportiva** que vimos en las tareas, hay una tabla con la información de las cuotas, **CUOTA.** Sabiendo que los atributos que forman la tabla son código, nombre e importe se nos pide construir la consulta de álgebra relacional que me permita obtener un **listado de las cuotas gratuitas**.

Suponiendo que la extensión actual de la relación **CUOTA** es:

código	nombre	importe
150	HONORÍFICA	0
250	COMPROMETIDA	150
350	COMÚN	100
450	FAMILIAR	50
550	OCASIONAL	60

La consulta de álgebra relacional que me permite obtener esa información será:

$$CUOTA\_GRATUITA = \sigma_{importe=0}(CUOTA)$$

CUOTA_GRATUITA es el nombre de la relación (tabla) obtenida al realizar la selección sobre la relación **CUOTA**. Sólo contiene aquellas **filas** que cumplen lo indicado en el predicado o fórmula, **importe=0**.

CUOTA_GRATUITA

código	nombre	importe
150	HONORÍFICA	0

7.2.2. Proyección (π)

Extrae un subconjunto de una relación (tabla), **creando otra relación**. *Se eliminarán los duplicados*:

$$R2 = \pi_{(A1,A2,...,An)}(R1)$$

- Cada **Ai** es un nombre de atributo de **R1**
- **R1** es la relación (tabla) sobre la que se realiza π
- **R2** es la relación (tabla) resultante después de π

Ejemplo: En la BD de una **sociedad cultural y deportiva** que vimos en las tareas, hay una tabla con la información de las actividades, **ACTIVIDAD.** Nos piden obtener el **nombre y el número de plazas de cada actividad**.

Suponiendo que la extensión actual de la relación **ACTIVIDAD** es:

identificador	nombre	fecini	fecfin	plazas	precio	observac	cod_prof	num_aula
10	TENIS	01/09/2005	11/09/2005	10	300	< 18 años	100	1
20	POESÍA	30/10/2005	15/11/2005	15	150	Null	200	2
30	COCINA	20/10/2005	20/12/2005	7	250	Nivel avanzado	100	3

La consulta de álgebra relacional que me permite obtener esa información será:

PLAZAS_POR_ACTIVIDAD= π nombre, plazas (ACTIVIDAD)

PLAZAS_POR_ACTIVIDAD es el nombre de la relación (tabla) obtenida al realizar la selección sobre la relación **ACTIVIDAD**. Sólo contiene aquellos atributos que se han indicado como subíndice de la proyección.

PLAZAS_POR_ACTIVIDAD

nombre	plazas
TENIS	10
POESÍA	15
COCINA	7

7.2.3. Unión ($\cup$)

Construye una nueva relación, a partir de dos especificadas, uniendo sus filas. Para poder realizarse, las 2 relaciones deben tener el **mismo nº de atributos** definidos sobre los **mismos dominios**.

R3 = R1 $\cup$ R2

- **R1** es una relación (tabla) con m tuplas y p columnas
- **R2** es una relación (tabla) con n tuplas y p columnas
- **R3** es la relación resultante de la unión, formada con **m+n tuplas** y **p columnas** (*eliminando las tuplas duplicadas*)

Ejemplo: Supongamos que tenemos 2 relaciones **AULA1** y **AULA2** y que queremos realizar su **unión**. A continuación se muestra la extensión de cada una de las relaciones y del resultado de su unión.

AULA1

Número	Descripción	Superficie	Estado
1	Soleada	25	B
2	Amplia	80	M
3	Con cocina	30	R
4	Deportiva	100	MB

$\cup$

AULA2

Número	Descripción	Superficie	Estado
1	Soleada	25	B
5	Con cocina	40	B
6	Biblioteca	30	MB

AULA3 = AULA1 $\cup$ AULA2

Número	Descripción	Superficie	Estado
1	Soleada	25	B
2	Amplia	80	M
3	Con cocina	30	R
4	Deportiva	100	MB
5	Con cocina	40	B
6	Biblioteca	30	MB

7.2.4. Diferencia (–)

Except o not in

La diferencia de 2 tablas **R1** y **R2** es otra **R3** que tendrá las tuplas que están en la primera, pero no en la segunda. Para poder realizarse, las 2 relaciones deben tener el **mismo nº de atributos** definidos sobre **dominios compatibles**.

R3 = R1 – R2

- **R1** es una relación (tabla) con m tuplas y p columnas
- **R2** es una relación (tabla) con n tuplas y p columnas
- **R3** es la relación resultante de la diferencia, formada con **m-x tuplas** y **p columnas**, siendo x el nº de tuplas en **R2** que también están en **R1**

Ejemplo: Supongamos que tenemos 2 relaciones **AULA1** y **AULA2** y que queremos realizar su **diferencia**. A continuación se muestra la extensión de cada una de las relaciones y del resultado de su diferencia.

AULA1

Número	Descripción	Superficie	Estado
1	Soleada	25	B
2	Amplia	80	M
3	Con cocina	30	R
4	Deportiva	100	MB

–

AULA2

Número	Descripción	Superficie	Estado
1	Soleada	25	B
5	Con cocina	40	B
6	Biblioteca	30	MB

AULA3 = AULA1 – AULA2

Número	Descripción	Superficie	Estado
2	Amplia	80	M
3	Con cocina	30	R
4	Deportiva	100	MB

7.2.5. Producto cartesiano (X)

A partir de 2 tablas **R1** y **R2** se obtiene otra **R3**, cuyas tuplas serán todas las posibles combinaciones de tuplas de las 2 relaciones:

R3 = R1 X R2

- **R1** es una relación (tabla) con m tuplas y p columnas
- **R2** es una relación (tabla) con n tuplas y t columnas
- **R3** es la relación resultante de la unión, formada por p+t columnas y m.n tuplas

Ejemplo: Supongamos que tenemos 2 relaciones **DATOS_AULA** y **NUEVOS_DATOS** y que queremos obtener su **producto cartesiano**. A continuación se muestra la extensión de cada una de las relaciones y del resultado de su producto cartesiano.

DATOS_AULA

Descripción	Superficie	Estado
Amplia	80	M
Con cocina	30	R
Soleada	25	B

X

NUEVOS_DATOS

Puestos	Pizarra
20	Sí
30	No

AULA3 = DATOS_AULA X NUEVOS_DATOS

Descripción	Superficie	Estado	Puestos	Pizarra
Amplia	80	M	20	Sí
Amplia	80	M	30	No
Con cocina	30	R	20	Sí
Con cocina	30	R	30	No
Soleada	25	B	20	Sí
Soleada	25	B	30	No

7.2.6. *Intersección (∩)*

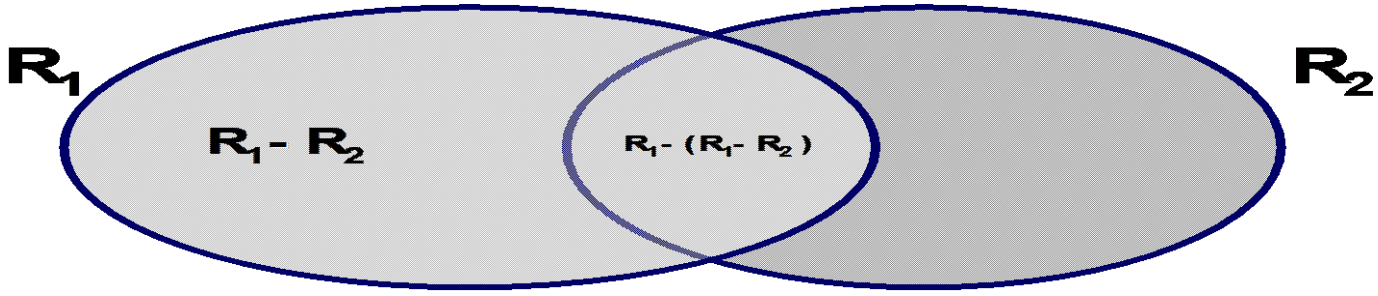
quedarse con las filas comunes

Se obtiene una tabla **R3** cuyas tuplas serán las comunes a las dos relaciones (tablas) de las que se parte, **R1** y **R2** . Estas deben tener el **mismo grado** y **dominios compatibles**:

$$R3 = R1 \cap R2$$

Es una *operación derivada,* ya que:

$$R1 \cap R2 = R1 - (R1 - R2)$$



Ejemplo: Supongamos que tenemos 2 relaciones **CUOTA1** y **CUOTA2** y que queremos realizar su **intersección**. A continuación se muestra la extensión de cada una de las relaciones y del resultado de su intersección.

CUOTA1

Código	Nombre	Importe
150	HONORÍFICA	0
250	COMPROMETIDA	150
350	COMÚN	100
450	FAMILIAR	50

∩

CUOTA2

Código	Nombre	Importe
350	COMÚN	100
550	OCASIONAL	60

CUOTA3 = CUOTA1 ∩ CUOTA2

Código	Nombre	Importe
350	COMÚN	100

7.2.7. Combinación natural ó Reunión natural o Join natural (*)

$R3 = (R1 * R2)_{condición}$ ó $R3 = (R1 *_{condición} R2)$

A partir de **R1** y **R2** construye una relación **R3** concatenando cada tupla de **R1** con cada una de **R2**. Esa tupla formará parte de **R3** si satisface la condición dada.

Para este caso de la *combinación natural* (*), la **condición es de igualdad** y en **R3 se elimina el o los atributos comunes**.

Es un producto cartesiano seguido de selección por igualdad y de una proyección, a menos que no tengan atributos en común, caso en el que se verifica que **R1 * R2= R1 X R2.**

Ejemplo: Supongamos que tenemos 2 relaciones **SOCIO** y **CUOTA** y que queremos calcular el **join natural** con la condición de igualdad de códigos de cuota. A continuación se muestra la extensión de cada una de las relaciones y del resultado de su combinación natural.

SOCIO

Número	NIF	…	Abonada	Cod_cuota
1001	22222221A	…	Sí	150
1002	22222222B	…	No	250
1003	22222223C	…	Sí	150

*

CUOTA

Código	Nombre	Importe
150	HONORÍFICA	0
250	COMPROMETIDA	150
350	COMÚN	100

SOCIO_CUOTA = (SOCIO * CUOTA) $_{cod\_cuota=codigo}$

Número	NIF	…	Abonada	Cod_cuota	Nombre	Importe
1001	22222221A	…	Sí	150	HONORÍFICA	0
1002	22222222B	…	No	250	COMPROMETIDA	150
1003	22222223C	…	Sí	150	HONORÍFICA	0

Es importante notar que en la relación final SÓLO APARECE EL PRIMER CAMPO DE LA IGUALDAD de la condición del join natural, en este caso **Cod_cuota**.

primero hace producto cartesario y luego compara si hay iguales y si hay, las devuelve. JOIN

7.2.8. Combinación ó Reunión ó Join (θ)

La condición no es de igualdad, al contrario que la anterior y en el resultado tienen que aparecer las dos columnas

$$R3 = R1 \ \theta_{condición} \ R2$$

A partir de **R1** y **R2** construye una relación **R3** seleccionando aquellas tuplas del producto cartesiano **R1 x R2** que satisfagan una **condición** dada.

- **condición** involucra al atributo **A** de R_1 y **B** de R_2, definidos en el mismo dominio.
- Obsérvese que**:**

$$R1 \ \theta_{A \ condición \ B} \ R2 = \sigma_{A \ condición \ B} \ (R_1 \times R_2)$$

Ejemplo: Supongamos que tenemos 2 relaciones **DATOS_AULA** y **NUEVOS_DATOS** y que queremos obtener su **join** con la **condición** de que **Estado>EstadoEsperado**. A continuación se muestra la extensión de cada una de las relaciones y del resultado de su producto cartesiano.

DATOS_AULA

Descripción	Superficie	Estado
Soleada	25	MB
Amplia	80	M
Con cocina	30	B

θ

NUEVOS_DATOS

Puestos	EstadoEsperado
20	B
30	R

ESTADO_PREVISTO = DATOS_AULA $\theta_{Estado>EstadoEsperado}$ NUEVOS_DATOS

Descripción	Superficie	Estado	Puestos	EstadoEsperado
Soleada	25	MB	20	B
Soleada	25	MB	30	R
Con cocina	30	B	30	R

Es importante notar que en la relación final aparecen TODOS LOS CAMPOS de las 2 relaciones implicadas en el join.

7.3. OPERACIONES ADICIONALES

Son aquellas que no es posible expresar en función de los operadores básicos. Son las expuestas a continuación:

- **Asignación (←ó =):** Para **almacenar el resultado** de una consulta en una nueva relación, o para almacenar resultados intermedios.
- **Renombrar (ó RENAME):** Para **cambiar el nombre** de un atributo.

Ejemplo: Cambiar el nombre del atributo Descripción por Nombre:
AULA **renombrar** Descripción **como** Nombre

- **Ampliar (AMPLIA):** Para obtener **campos calculados** a partir de los valores de atributos de una tupla.

Ejemplo: Obtener el salario neto de los empleados, si tenemos en cuenta que pagan un 16% de impuestos: EMPLEADO **amplía** (Salario – Salario*0.16) **como** SalarioNeto

- **Agrupación (ó GROUP BY):** Para poder **usar funciones de agregación** (suma, media, cuenta,...), el group by agrupa tuplas en subconjuntos que tienen valores comunes en ciertos atributos.

Ejemplo: Obtener el salario máximo y mínimo de los empleados por Cargo:
EMPLEADO **agrupar por** Cargo, MAX(Salario), MIN(Salario)

7.4. ACLARACIÓN DEL USO DE VALORES NULOS

- El **producto cartesiano** no se ve afectado por los valores nulos.
- La **restricción** sólo devuelve aquellas tuplas cuya condición se evalúa como cierta (elimina las evaluadas como falso o quizás).
- La **proyección elimina** tuplas duplicadas teniendo en cuenta los nulos (2 nulos son iguales).

7.5. NORMALIZACIÓN DE RELACIONES

Los pasos a seguir para **modelizar un minimundo** son:

1. Análisis del problema
2. Obtener el diagrama EER
3. Paso del EER al MR *(todas las relaciones han de estar como mínimo en 3ª forma normal)*
4. Realizar las consultas principales en álgebra relacional (paso opcional)
5. Implantar el modelo resultante en un SGBD relacional (MS Access, SQL Server, Oracle…)

Al llevar a cabo el desarrollo del punto 3 anterior, hemos de tener en cuenta que todas las relaciones obtenidas al pasar de EER a MR deben estar como mínimo en **Tercera Forma Normal (3FN)**.

Las **formas normales** son una serie de reglas que, de cumplirse, aseguran que el esquema diseñado tendrá un buen comportamiento en cuanto a redundancia, pérdida de información y representación de la misma.

Para aclarar todo esto debemos plantear una serie de definiciones previas.

7.5.1. Relación Universal

Se denomina **relación universal** de una base de datos a aquella relación (tabla) del MR que reúne todos los atributos de una base de datos en una única relación o tabla, y puede almacenar cualquier tipo de información que interese en esa BD.

Solo saber definición, coger y meter todo en una tabla

Los **problemas** del uso de una única relación son:

- de **actualización**
- de **borrado**
- de **inserción**.

Veámoslo con un ejemplo.

Ejemplo: Supongamos una BD que tiene información sobre **vendedores**, **artículos** y **cantidades** de artículos solicitadas y que actualmente contiene los siguientes datos en una única relación (*relación universal de la BD*):

cod_vendedor	ciudad	país	cod_articulo	cantidad
S1	Londres	Inglaterra	P1	200
S1	Londres	Inglaterra	P2	1300
S1	Londres	Inglaterra	P4	345
S2	Madrid	España	P1	24
S2	Madrid	España	P6	988
S3	París	Francia	P2	200
S4	Londres	Inglaterra	P3	34

Es fácil darse cuenta de que esta BD no está muy bien diseñada, debido al alto grado de redundancia; ya que, por ejemplo, no es necesario especificar tantas veces que el vendedor S1 es de Londres.

Veamos con ejemplos los **problemas** que encontramos:

- Problemas de **actualización:** si el proveedor S1 cambiase de ciudad, habría que actualizar al nuevo valor todas las tuplas en las que aparezca ese proveedor S1.
- Problemas de **borrado:** si el proveedor S3 deja de suministrar el producto P2, debemos eliminar la tupla correspondiente, y por lo tanto perderemos toda la información de ese proveedor.
- Problemas de **inserción:** cuando agregamos un nuevo proveedor, si todavía no se le ha solicitado nada, habría valores nulos en los atributos **cod_articulo** y **cantidad**, y como hemos dicho en repetidas ocasiones, hay que intentar evitar los valores nulos en las tablas en la medida de lo posible.

7.5.2. Dependencias funcionales

Se dice que una relación (tabla) puede representarse como:

- el **conjunto de sus atributos (AT)**, y,
- el **conjunto de las relaciones existentes entre ellos (DF)**.

Esas relaciones entre atributos son las que se conocen como **dependencias funcionales.** Para representar una relación **R**, siendo **AT** el conjunto de sus atributos y **DF** el conjunto de las dependencias funcionales, lo haríamos del siguiente modo: **R(AT, DF)**

A continuación definimos **más formalmente una dependencia funcional:** Sea una relación **R** con atributos (x, y, z), se dice que el atributo y **depende funcionalmente de** x o lo que es lo mismo, que x **implica** o **determina** a y, si y sólo si a ***cada valor de x le corresponde un único valor de y***.

Se representaría así: $\mathbf{x \rightarrow y}$

Hay que tener en cuenta que **x** e **y** pueden ser conjuntos de atributos.

Ejemplo: Veamos si son ciertas o no las siguientes dependencias funcionales que se plantean para la relación universal de una BD de un internado:

INTERNADO(num_matricula,nombre_alumno,num_habit, num_telef, asignatura, convocatoria, nota)

- **¿num_matricula → nombre_alumno?**

  1004 → Javier Pérez Pérez

  Un determinado número de matrícula me devuelve un único nombre de alumno, no existen dos o más alumnos con el mismo número de matrícula. Se verifica entonces la dependencia funcional planteada, es decir **num_matricula → nombre_alumno.**

- **¿nombre_alumno → num_matricula?**

  Javier Pérez Pérez → 1004

  → 1200

  En el internado pueden existir 2 alumnos con el mismo nombre, por lo tanto para un mismo valor de nombre_alumno pueden existir más de un valor de num_matricula. De esto deducimos que

  nombre_alumno ↛ num_matricula (nombre_alumno no determina a num_matricula)

- **¿num_habit → num_matricula?** ***(suponemos que hay habitaciones dobles)***

  105 → 1004

  → 2021

  Como pueden estar dos alumnos en la misma habitación concluimos que

  num_habit ↛ num_matricula (num_matricula no depende funcionalmente de num_habit)

  INDICA TÚ AHORA SI SE CUMPLEN O NO LAS SIGUIENTES DEP. FUNCIONALES

- **¿num_matricula → num_habit?**

- **¿num_habit → num_telef?** ***(suponemos que hay una línea de teléfono en cada habitación)***

- **¿num_telef → num_habit?**

- **¿num_matricula → num_telef?**

- **¿num_telef → num_matricula?**

7.5.3. Formas normales

Las **formas normales** son reglas que deben cumplir cada una de las relaciones de un BD relacional para que el diseño se pueda considerar correcto.

Las formas normales establecidas para el MR son:

- la **primera Forma Normal (1FN)**
- la **segunda Forma Normal (2FN)**
- la **tercera Forma Normal (3FN)**
- la **Forma Normal de Boyce-Codd (FNBC)**
- la **cuarta Forma Normal (4FN)**
- la **quinta Forma Normal (5FN)**

Si llegas a la forma normal 3, ya se dice que está formalizada, pero puedes seguir optimizándola hasta el 5

A nosotros nos bastará con **demostrar que nuestras relaciones están en 3FN**, para poder considerar que el diseño del MR es el adecuado, que son las formas normales que inicialmente propuso Codd en 1970. Más tarde se ampliarían hasta la quinta.

Si no lo están habrá que descomponer las relaciones en otras que sí lo estén, es decir, habrá que **normalizar las relaciones**.

7.5.3.1. Primera Forma Normal (1FN)

Es una regla o restricción inherente al MR (como ya vimos), por lo que su cumplimiento es obligatorio. Diremos que una relación **R** se encuentra en **1FN** si y sólo si, cada elemento de la relación tiene un valor atómico. Es decir, cada intersección fila/columna de la tabla sólo tiene un dato.

Ejemplo: Veamos la relación **PROFESOR(codigo, nombre, telefono)** con los siguientes datos:

codigo	nombre	telefono
1	Antonio	600111111 981111111
2	María	600222222 981222222

Es una tabla pero no una relación. **NO ESTÁ EN 1FN** porque en el atributo **telefono** hay 2 valores para una misma fila

Para **normalizar** esta tabla y **convertirla en una relación** deberíamos tener 2 campos para guardar los teléfonos:

codigo	nombre	telefono1	telefono2
1	Antonio	600111111	981111111
2	María	600222222	981222222

ESTÁ EN 1FN porque para cada intersección fila/columna hay un único valor del dominio correspondiente.

7.5.3.2. Segunda Forma Normal (2FN)

Diremos que una relación **R** se encuentra en **2FN** si y sólo si:

- está en **1FN**, y,
- todos los atributos no clave (no principal) que dependen de la clave, dependen de manera completa, es decir, **no existen dependencias funcionales PARCIALES**.

IMPORTANTE: Hay que tener en cuenta que **sólo pueden existir dependencias funcionales parciales** en aquellas relaciones en las que la **clave es multiatributo** (está formada por al menos 2 atributos).

Si una relación no está en 2FN se podrá aplicar el siguiente teorema de manera que al descomponer la relación original en dos, por lo menos una de ellas se encuentre en 2FN.

Teorema I: Sea una relación **R** con atributos **(A, B, C, D)** y con clave primaria **(A, B)** tal que $\mathbf{A \rightarrow D}$.
Entonces la relación **R** puede descomponerse en:

$$\mathbf{R} \Rightarrow \begin{cases} \mathbf{R1(A, D)} \\ \mathbf{R2(A, B, C)} \end{cases}$$

Ejemplo: Sea el esquema de relación **PROVEEDOR_ARTICULO(AT, DF)** donde:
AT = { apellido, articulo, direccion, precio} **CP = {apellido, articulo}**
DF = {apellido, articulo → precio,
apellido → direccion }

- Suponemos que está en 1FN porque todos los valores en los atributos son atómicos.
- Pero no está en 2FN porque **direccion** es un atributo no clave que según la dependencia funcional **apellido → direccion** depende **parcialmente de la clave.**

Para conseguir poner la relación **PROVEEDOR_ARTICULO** en 2FN debemos aplicar el Teorema I. Por lo tanto aislamos en una relación las dependencias funcionales que causan el problema del siguiente modo:

PROVEEDOR ({apellido, direccion} , {apellido → direccion})
SUMINISTRA ({apellido, articulo, precio} , {apellido, articulo → precio})

Una vez aplicado el teorema nos queda un último paso para dar por concluida la normalización de la relación. Este paso consiste en demostrar que las 2 relaciones nuevas están en 2FN:

PROVEEDOR:

- está en 1FN puesto que ya lo estaba la relación original,
- y está en 2FN porque no existen dependencias funcionales parciales ya que en este caso es imposible puesto que la clave primaria es monoatributo.

SUMINISTRA:

- está en 1FN puesto que ya lo estaba la relación original,
- y está en 2FN porque no existen dependencias funcionales parciales ya que el único atributo no clave **precio** depende completamente de la clave.

7.5.3.3. Tercera Forma Normal (3FN)

Diremos que una relación **R** se encuentra en **3FN** si y sólo si:

- está en **2FN**, y,
- todos los atributos no clave (no principal) que dependen de la clave, dependen de manera directa de ella, es decir, **no existen dependencias funcionales TRANSITIVAS**.

Si una relación no está en 3FN se podrá aplicar el siguiente teorema de manera que al descomponer la relación original en dos, por lo menos una de ellas se encuentre en 3FN.

Teorema II: Sea una relación **R** con atributos **(A, B, C, D)** y con clave primaria **(A)** tal que **A → B , B → C** y **A → D** . Entonces la relación **R** puede descomponerse en:

$$R \Rightarrow \begin{cases} R1(A, B, D) \\ R2(B, C) \end{cases}$$

Podemos decir que *aislamos las dependencias funcionales* que generan el problema en una relación independiente del resto.

Ejemplo: Sea el esquema de relación **COCHE(AT, DF)** donde:

AT = { matricula, marca, tipo, potencia, color} **CP = { matricula }**

DF = { matricula → tipo,
tipo → marca,
tipo → potencia,
matricula → color }

- Está en 2FN porque suponemos que está en 1FN y la clave primaria es monoatributo y nunca podrían existir dependencias funcionales parciales.
- No está en 3FN porque **matricula → tipo, tipo → marca, tipo → potencia** y vemos que según estas dependencias funcionales **marca** y **potencia** dependen transitivamente de la clave a través del atributo **tipo**.

Para conseguir poner la relación **COCHE** en 3FN debemos aplicar el Teorema II. Por lo tanto aislamos en una relación las dependencias funcionales que causan el problema del siguiente modo:

MODELO ({ tipo, marca, potencia } ,
{ tipo → marca,
tipo → potencia })
COCHE1 ({ matricula, tipo, color } ,
{ matricula → tipo,
matricula → color })

Una vez aplicado el teorema nos queda un último paso para dar por concluida la normalización de la relación. Este paso consiste en demostrar que las 2 relaciones nuevas están en 3FN:

MODELO:

- está en 2FN puesto que está en 1FN y la clave primaria sigue siendo monoatributo y nunca podrían existir dependencias funcionales parciales
- y está en 3FN porque no existen dependencias funcionales transitivas ya que todo atributo no clave **(marca, potencia)** depende directamente de la clave **(tipo)**.

COCHE1:

- está en 2FN puesto que está en 1FN y la clave primaria sigue siendo monoatributo y nunca podrían existir dependencias funcionales parciales
- y está en 3FN porque no existen dependencias funcionales transitivas ya que todo atributo no clave **(tipo, color)** depende directamente de la clave **(matricula)**.

8. CONCLUSIÓN

Hasta aquí hemos visto la parte dinámica del modelo relacional, es decir sus operadores y la normalización. En la unidad de trabajo siguiente empezaremos a estudiar el lenguaje estándar más utilizado para manipular la información de las bases de datos relacionales, el **lenguaje SQL (*Structured Query Language*)**.