DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Escuela Técnica Superior de Ingeniería Informática

etsinf

Bases de Datos y Sistemas de Información

Grado en Ingeniería Informática

Unidad Didáctica 1: Bases de Datos Relacionales

Parte 2: Modelo Relacional de Datos

(Doc. UD1.2)

Curso 2020/2021

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

El contenido de este documento no debe considerarse una aportación novedosa ya que se ha realizado utilizando casi literalmente el libro:

Celma, M.; Casamayor, J. C.; Mota, L.; *Bases de datos relacionales.* Pearson, Prentice Hall, 2003

El permiso explícito de los tres autores ha permitido su realización con el objetivo de facilitar el estudio de la materia de bases de datos para los alumnos de la ETSInf.

Índice

1 INTRODUCCIÓN

En el documento UD1.1 se definió una base de datos como una colección de datos estructurados de acuerdo a las reglas de un modelo. En este documento se va a presentar el *Modelo Relacional de Datos*, que es el modelo seguido por la familia actual de sistemas de gestión de bases de datos. Este modelo fue propuesto por E. F. Codd en 1970, imponiéndose sobre los modelos anteriores (jerárquico y red) durante la década de los ochenta

En primer lugar se van a introducir las bases de datos relacionales desde un punto de vista informal utilizando para ello un ejemplo, a continuación se definirán con rigor todos los conceptos esenciales del modelo.

2 PRESENTACIÓN INFORMAL DE UNA BASE DE DATOS RELACIONAL

2.1 Definición informal de una base de datos relacional

El éxito del modelo relacional reside en su sencillez ya que la información se estructura en *tablas* en las que los datos se organizan en *filas* y *columnas*.

El ejemplo con el que se va a trabajar a lo largo del documento es un subconjunto del sistema de información presentado en el último apartado del documento UD1.1 en el que se va a almacenar información relativa al título de GII de la ETSInf, para ello se definen las siguientes tablas:

Departamento			
cod_dep	**nombre**	**teléfono**	**director**
DLA	Lingüística Aplicada	2255	111
DMA	Matemática Aplicada	1256	
DSIC	Sistemas Informáticos y Computación	1542	453

Asignatura					
cod_asg	**nombre**	**semestre**	**cod_dep**	**teoría**	**prácticas**
11545	Análisis Matemático	1A	DMA	4,5	1,5
11546	Álgebra	1B	DMA	4,5	1,5
11547	Matemática Discreta	1A	DMA	4,5	1,5
11548	Bases de Datos y Sistemas de Información	3A	DSIC	4,5	1,5

Profesor					
dni	**nombre**	**teléfono**	**cod_dep**	**provincia**	**edad**
111	Luisa Bos Pérez		DMA	Alicante	33
123	Juana Cerdá Pérez	3222	DMA	Valencia	50
453	Elisa Rojo Amando	7859	DSIC	Valencia	26
564	Pedro Martí García	3412	DMA	Castellón	27

Docencia			
dni	**cod_asg**	**gteo**	**gpra**
111	11547	1	3
123	11545	0	2
123	11547	1	1
564	11545	2	2

Cada una de esas tablas representa una determinada información que se explica a continuación:

- Cada fila de la tabla *Departamento* almacena los datos de interés de un departamento de la universidad que son:
  - *cod_dep*: código asignado al departamento.
  - *nombre*: nombre del departamento.
  - *teléfono*: extensión telefónica del departamento dentro de la universidad.
  - *director*: *dni* del profesor que dirige el departamento.

  Así, la última fila de la tabla *Departamento* almacena la información de un departamento de código *DSIC* que se llama *Sistemas Informáticos y Computación,* cuya extensión es la 1542, y cuyo director es el profesor de *dni* 453*.*
- Cada fila de la tabla *Asignatura* almacena los datos de interés de una asignatura que son:
  - *cod_asg*: código asignado a la asignatura.
  - *nombre*: nombre de la asignatura.
  - *semestre*: en qué semestre se imparte la asignatura.
  - *cod_dep*: código del departamento al que pertenece la asignatura.
  - *teoría*: cuántos créditos teóricos tiene la asignatura.
  - *prácticas*: cuántos créditos prácticos tiene la asignatura.

  Así, la última fila de la tabla *Asignatura* almacena la información de una asignatura de del semestre 3A, de nombre *Bases de Datos y Sistemas de Información* a la que se le ha asignado el código '11548' con 4,5 créditos teóricos, 1,5 créditos prácticos, y cuyo departamento es el de código *DSIC.*
- Cada fila de la tabla *Profesor* almacena los datos de interés de un profesor que son:
  - *dni*: DNI del profesor.
  - *nombre*: nombre del profesor.
  - *teléfono*: extensión telefónica del profesor dentro de la universidad.
  - *cod_dep*: código del departamento al que pertenece el profesor.
  - *provincia*: dónde nació el profesor.
  - *edad*: edad del profesor.

  Así, la tercera fila de la tabla *Profesor* almacena la información de una profesora que se llama *Elisa Rojo Amando* cuya extensión es la 7859, que es del departamento de código *DSIC* y cuyo DNI es el 453*.*
- Cada fila de la tabla *Docencia* almacena la información relativa a qué profesores van a impartir las clases de las distintas asignaturas, concretamente:
  - *dni*: DNI de alguno de los profesores que hay en la tabla *Profesor.*
  - *cod_asg:* código de alguna de las asignaturas que hay en la tabla *Asignatura.*
  - *gteo*: cuántos grupos de teoría imparte el profesor de código *dni* en la asignatura de código *cod_asg*.
  - *gpra*: cuántos grupos de prácticas imparte el profesor de código *dni* en la asignatura de código *cod_asg*.

  Así, la última fila de la tabla *Docencia* almacena la información de que el profesor de *dni* 564 imparte 2 grupos de teoría y 2 de prácticas de la asignatura de código 11545.

Existe la posibilidad de que de algún objeto del que se está almacenando información se desconozca alguno de los datos que se guardan en las tablas. Por ejemplo, en la tabla *Profesor*, hay una fila que no tiene valor en la columna *teléfono;* esto no es un error y hay que interpretarlo como que se desconoce ese valor o incluso que no existe (ese profesor o no tiene extensión o ha decidido no proporcionarla).

Por complejo que sea el sistema de información para el que se esté diseñando la base de datos, una base de datos relacional siempre podrá ser vista como la del ejemplo, esto es, como un conjunto de tablas organizadas en filas y columnas de manera que:

- Las filas de una misma tabla tienen una estructura semejante y almacenan información similar de distintos objetos o individuos del mundo real (profesores, libros, coches, ciudades,…).
- Cada columna almacena una determinada propiedad de esos objetos (nombre, edad, latitud,…). Es importante destacar que los valores que pueden aparecer en una columna han de ser todos del mismo *tipo de datos*[1]*,* es decir que en una misma columna no puede aparecer por ejemplo una fecha y un número real.

El almacenar información de un determinado sistema de información en una base de datos tiene dos objetivos fundamentales:

- Guardar la información de interés para el sistema.
- Consultar la información almacenada.

Hoy en día, para alcanzar estos dos objetivos, los SGBD relacionales proporcionan un lenguaje que se denomina SQL (*Structured Query Language*)[2]. El estudio de este lenguaje es una parte fundamental de la asignatura y se presentará en la unidades didáctica 2, sin embargo, como introducción, a continuación se presenta, de nuevo informalmente, cómo manipular y consultar una base de datos relacional.

2.2 Manipulación informal de una base de datos relacional

Una vez definida una base de datos hay que poder manipularla para consultar, incluir, modificar o eliminar información. La consulta se verá más adelante, el resto de tareas se consigue añadiendo filas, borrando filas o cambiando algún valor en las filas ya almacenadas.

- Para incluir en la base de datos la información de un nuevo profesor de DNI 889*,* de nombre *Vicente Abad Real,* de 40 años y de *Granada* del que se desconoce el teléfono y el departamento, se añadiría una fila quedando la tabla como sigue:

Profesor					
dni	**nombre**	**teléfono**	**cod_dep**	**provincia**	**edad**
111	Luisa Bos Pérez		DMA	Alicante	33
123	Juana Cerdá Pérez	3222	DMA	Valencia	50
453	Elisa Rojo Amando	7859	DSIC	Valencia	26
564	Pedro Martí García	3412	DMA	Castellón	27
889	Vicente Abad Real			Granada	40

- Para eliminar la información relativa a la docencia del profesor de DNI 123 habría que borrar dos filas de la tabla docencia quedando la tabla como se muestra:

[1] Dada la importancia de este concepto en la materia de bases de datos, más adelante se definirá con más rigor.
[2] Lenguaje estructurado de consulta.

Docencia			
dni	cod_asg	gteo	gpra
111	11547	1	3
564	11545	2	2

- Si la asignatura de código '11548' cambia de nombre y pasa a llamarse '*Bases de Datos Relacionales*' sería necesario modificar el valor de la columna *nombre* de la fila con *cod_asg*=11548 con ese nuevo valor quedando la tabla como sigue:

Asignatura					
cod_asg	nombre	semestre	cod_dep	teoría	prácticas
11545	Análisis Matemático	1A	DMA	4,5	1,5
11546	Álgebra	1B	DMA	4,5	1,5
11547	Matemática Discreta	1A	DMA	4,5	1,5
11548	Bases de Datos Relacionales	3A	DSIC	4,5	1,5

2.3 Consulta informal de una base de datos relacional

Para poder explotar la información almacenada en una base de datos relacional es necesario disponer de instrucciones que permitan resolver consultas (preguntas) relativas a esa información. Es importante destacar que en un sistema relacional, el resultado de una consulta es presentado al usuario también como una tabla.

Esencialmente la instrucción del SQL que permite consultar una base de datos es la instrucción SELECT que se verá más adelante en la unidad didáctica 2. A continuación, se presenta la consulta de una base de datos relacional de manera informal con algunos ejemplos que ilustran e introducen algunos operadores muy usados. Estos operadores constituyen un lenguaje de interrogación de bases de datos relacionales que se conoce con el nombre de *Álgebra Relacional*.

El Álgebra Relacional es uno de los lenguajes que E. F. Codd propuso para su Modelo Relacional de Datos y consiste en un conjunto de operadores que actúan sobre tablas[3] y que devuelven como resultado una tabla. Estos operadores se pueden clasificar en tres grupos:

- Operadores conjuntistas: Estos operadores se heredan de la estructura de datos *conjunto*.
  - Unión ($\cup$): dadas dos tablas similares (con el mismo nombre de columnas y los mismos tipos de datos) la unión de ellas da como resultado una tabla en la que están todas las filas que aparecen en una de ellas o en ambas.
  - Intersección ($\cap$): dadas dos tablas similares la intersección de ellas da como resultado una tabla en la que están todas las filas que aparecen en las dos.
  - Diferencia ($-$): dadas dos tablas similares la diferencia (o resta) de ellas da como resultado una tabla en la que están todas las filas que aparecen en la primera y no en la segunda.
  - Producto cartesiano ($\times$): dadas dos tablas que no tengan nombres de columna iguales, el producto cartesiano da como resultado una tabla con tantas columnas como tengan las tablas y con todas las filas que se puedan construir con una fila de la primera y una de la segunda.

  Dado que no se puede ilustrar el uso de estos operadores con el ejemplo que se está manejando al ser éste muy sencillo, supónganse las siguientes tres tablas:

[3] Realmente, el Álgebra Relacional es "un conjunto de operadores que actúan sobre *relaciones*" no *tablas.* Pero, dado que en este apartado se está presentando informalmente el modelo relacional y aún no se ha introducido el concepto de *relación,* nos hemos permitido la licencia de seguir hablando de *tablas.*

R

A	B
1	2
7	5

S

A	B
1	2
4	6

T

C	D
3	5
4	6

El uso de los operadores conjuntistas devolverían las siguientes tablas:

$R \cup S$

A	B
1	2
7	5
4	6

$R \cap S$

A	B
1	2

$R - S$

A	B
7	5

$R \times T$

A	B	C	D
1	2	3	5
1	2	4	6
7	5	3	5
7	5	4	6

- Operador auxiliar: *renombrar*. Este operador permite cambiar el nombre de una columna por otro durante la operación. Para ello se indica entre paréntesis el nombre actual de la columna seguido del nuevo nombre. Por ejemplo, sean las tablas *T* definida arriba, y sea la siguiente expresión del álgebra relacional:

  ```
  T((C,A),(D,B))
  ```

  Esta expresión genera una tabla con las mismas filas que la tabla *T* pero en la que la columna *C* pasa a llamarse *A* y la columna *D* pasa a llamarse *B*:

  T((C,A),(D,B))

A	B
3	5
4	6

  Así, si por ejemplo, si se quiere realizar la unión *R* y *T*, dado que para unir dos tablas es necesario que sus columnas tengan el mismo nombre, escribiríamos la siguiente expresión:

  ```
  R ∪ T((C,A),(D,B))
  ```

  Esta expresión produciría el siguiente resultado:

  $R \cup$ T((C,A),(D,B))

A	B
1	2
7	5
3	5
4	6

  Es importante destacar que el cambio de nombre está vigente exclusivamente durante la evaluación de la expresión.

  En el caso de que sólo se le cambie el nombre a un atributo no será necesario poner dobles paréntesis.

- Operadores relacionales: Selección, Proyección y Concatenación. Estos operadores son específicos del modelo relacional de datos. Los símbolos usados para cada operador son los siguientes:
  - Selección: R DONDE *condición*: este operador tiene como entrada una tabla y devuelve el subconjunto de sus filas que hacen cierta la condición.
  - Proyección: R[col_1, …, col_j]: este operador tiene como entrada una tabla y devuelve el subconjunto de columnas especificado entre corchetes.
  - Concatenación: $\otimes_{\ldots}$ este operador es un poco más complejo y se explicará en los ejemplos que siguen.

A continuación, sin entrar en formalismos sintácticos, se presentan estos operadores utilizando para ello consultas sobre la base de datos ejemplo.

- *Obtener el DNI, el nombre, el teléfono, el código de departamento, la provincia y la edad del profesor de dni 453.*

  Para resolver esta consulta, es necesario *seleccionar* las filas de la tabla *Profesor* que cumplen una determinada condición. En álgebra relacional escribiríamos la siguiente expresión:

  ```
  Profesor DONDE dni = 453
  ```

  La evaluación de esta expresión en un sistema relacional devolvería el siguiente resultado:

dni	nombre	teléfono	cod_dep	provincia	edad
453	Elisa Rojo Amando	7859	DSIC	Valencia	26

- *Obtener el nombre de todos los profesores.*

  Para resolver esta consulta, es necesario *proyectar* las filas de la tabla *Profesor* sobre la columna *nombre* eliminando las columnas que no interesan:

  ```
  Profesor[nombre]
  ```

  Esta expresión produciría en un sistema relacional la siguiente respuesta:

nombre
Luisa Bos Pérez
Juana Cerdá Pérez
Elisa Rojo Amando
Pedro Martí García

- *Obtener una tabla similar a la tabla docencia pero incluyendo también toda la información del profesor cuyo DNI aparece en cada fila.*

  Para resolver esta consulta es necesario *concatenar* las tablas *Profesor* y *Docencia* por la columna *dni*, es decir emparejar las filas de la relación *Profesor* y *Docencia* que tengan el mismo valor en la columna *dni*:

  $$\text{Profesor} \otimes_{\text{dni}} \text{Docencia}$$

  Esta expresión produciría la siguiente respuesta:

dni	nombre	teléfono	cod_dep	provincia	edad	cod_asg	gteo	gpra
111	Luisa Bos Pérez		DMA	Alicante	33	11547	1	3
123	Juana Cerdá Pérez	3222	DMA	Valencia	50	11545	0	2
123	Juana Cerdá Pérez	3222	DMA	Valencia	50	11547	1	1
564	Pedro Martí García	3412	DMA	Castellón	27	11545	2	2

Obviamente, hay consultas más complejas que implican expresiones más elaboradas en la que entre otros elementos aparecerán:

- Operadores lógicos: AND, OR y NOT ($\wedge$, $\vee$ y $\neg$)
- Predicados de comparación: =, <>, <=, >=, <, >
- Otros predicados.

Así pues, con el álgebra relacional se pueden resolver consultas a una base de datos escribiendo expresiones en las que intervengan los operadores presentados, en estas expresiones se pueden combinar varios operadores y es importante tener claro el orden de evaluación de los mismos que es el siguiente:

1. Expresiones entre paréntesis.
2. Operadores unarios: renombrar, seleccionar, proyectar.
3. Operadores binarios: $\cup$, $\cap$, $-$, $\otimes$ y $\times$.
4. Ante igualdad de preferencia, las expresiones se evalúan de izquierda a derecha.

2.3.1 Ejercicios de Álgebra Relacional

A continuación se resuelven algunas consultas más complicadas presentando el resultado que se obtendría, siempre de forma tabular.

- *Obtener el nombre de las asignaturas del departamento de código DSIC.*

```
(Asignatura DONDE cod_dep='DSIC')[nombre]
```

nombre
Bases de Datos y Sistemas de Información

En esta expresión podría obviarse el uso de paréntesis ya que por el orden de preferencia, primero se seleccionan de la tabla *Asignatura* las filas que cumplen la condición, y luego se proyecta sobre la columna *nombre.*

- *Obtener el nombre de los profesores que imparten docencia en la asignatura de código* 11547.

```
((Profesor ⊗dni Docencia)DONDE cod_asg=11547)[nombre]
```

Evaluando esta expresión por partes tendríamos los siguientes resultados:

1. (Profesor $\otimes_{dni}$ Docencia): hay que evaluar primero lo que está entre paréntesis:

dni	nombre	teléfono	cod_dep	provincia	edad	cod_asg	gteo	gpra
111	Luisa Bos Pérez		DMA	Alicante	33	11547	1	3
123	Juana Cerdá Pérez	3222	DMA	Valencia	50	11545	0	2
123	Juana Cerdá Pérez	3222	DMA	Valencia	50	11547	1	1
564	Pedro Martí García	3412	DMA	Castellón	27	11545	2	2

2. (Profesor $\otimes_{dni}$ Docencia)DONDE cod_asg=11547: en la tabla generada en el paso 1, seleccionamos las filas con cod_asg=11547:

dni	nombre	teléfono	cod_dep	provincia	edad	cod_asg	gteo	gpra
111	Luisa Bos Pérez		DMA	Alicante	33	11547	1	3
123	Juana Cerdá Pérez	3222	DMA	Valencia	50	11547	1	1

3. ((Profesor $\otimes_{dni}$ Docencia)DONDE cod_asg=11547)[nombre]: en la tabla generada en el paso 2, proyectamos sobre el atributo *nombre* siendo el resultado final el siguiente:

nombre
Luisa Bos Pérez
Juana Cerdá Pérez

- *Obtener una tabla con la misma información que la tabla Docencia, pero que incluya el nombre del profesor y el nombre de la asignatura.*

```
((Profesor(nombre,nom_P)⊗dni Docencia)[dni, nom_P, cod_asg,gteo,gpra]
                ⊗cod_asg
        Asignatura(nombre,nom_A))[dni,nom_P,cod_asg,nom_A,gteo,gpra]
```

En esta expresión se han renombrado las columnas *nombre* tanto en *Profesor* (nuevo nombre *nom*_P) como en *Asignatura* (nuevo nombre *nom_A*), con ello se evita que en la tabla resultado haya dos columnas que se llamen igual (obviamente bastaría renombrar sólo en una de ellas).

Evaluando esta expresión por partes tendríamos los siguientes resultados:

1. Profesor(nombre,nom_P): se renombra la columna *nombre* en la tabla *Profesor:*

dni	nom_P	teléfono	cod_dep	provincia	edad
111	Luisa Bos Pérez		DMA	Alicante	33
123	Juana Cerdá Pérez	3222	DMA	Valencia	50
453	Elisa Rojo Amando	7859	DSIC	Valencia	26
564	Pedro Martí García	3412	DMA	Castellón	27

2. Asignatura(nombre,nom_A): se renombra la columna *nombre* en la tabla *Asignatura:*

cod_asg	nom_A	semestre	cod_dep	teoría	prácticas
11545	Análisis Matemático	1A	DMA	4,5	1,5
11546	Álgebra	1B	DMA	4,5	1,5
11547	Matemática Discreta	1A	DMA	4,5	1,5
11548	Bases de Datos y Sistemas de Información	3A	DSIC	4,5	1,5

3. Profesor(nombre,nom_P)$\otimes_{dni}$ Docencia: se concatena la tabla del paso 1 con la tabla *Docencia*:

dni	nom_P	teléfono	cod_dep	provincia	edad	cod_asg	gteo	gpra
111	Luisa Bos Pérez		DMA	Alicante	33	11547	1	3
123	Juana Cerdá Pérez	3222	DMA	Valencia	50	11545	0	2
123	Juana Cerdá Pérez	3222	DMA	Valencia	50	11547	1	1
564	Pedro Martí García	3412	DMA	Castellón	27	11545	2	2

4. (Profesor(nombre,nom_P)$\otimes_{dni}$ Docencia)[dni,nom_P,cod_asg,gteo,gpra]: se proyecta la tabla del paso 3 para eliminar las columnas que no interesan (*teléfono, cod_dep, provincia y edad*):

dni	nom_P	cod_asg	gteo	gpra
111	Luisa Bos Pérez	11547	1	3
123	Juana Cerdá Pérez	11545	0	2
123	Juana Cerdá Pérez	11547	1	1
564	Pedro Martí García	11545	2	2

5. ((Profesor(nombre,nom_P)$\otimes_{dni}$ Docencia)[dni,nom_P,cod_asg,gteo,gpra]
$\otimes_{cod\_asg}$ Asignatura(nombre,nom_A)):
se concatenan las tablas de los pasos 4 y 2 para conseguir el nombre de cada asignatura:

dni	nom_P	cod_asg	gteo	gpra	nom_A	semestre	cod_dep	teoría	prácticas
111	Luisa Bos Pérez	11547	1	3	Matemática Discreta	1A	DMA	4,5	1,5
123	Juana Cerdá Pérez	11545	0	2	Análisis Matemático	1A	DMA	4,5	1,5
123	Juana Cerdá Pérez	11547	1	1	Matemática Discreta	1A	DMA	4,5	1,5
564	Pedro Martí García	11545	2	2	Análisis Matemático	1A	DMA	4,5	1,5

6. ((Profesor(nombre,nom_P)$\otimes_{dni}$ Docencia)[dni, nom_P, cod_asg,gteo,gpra]
$\otimes_{cod\_asg}$ Asignatura(nombre,nom_A))[dni,nom_P,cod_asg,nom_A,gteo,gpra]:
se proyecta la tabla del paso 5 para eliminar las columnas que no interesan (*semestre, cod_dep,teoría, prácticas*):

nom_P	dni	cod_asg	nom_A	gteo	gpra
Luisa Bos Pérez	111	11547	Matemática Discreta	1	3
Juana Cerdá Pérez	123	11545	Análisis Matemático	0	2
Juana Cerdá Pérez	123	11547	Matemática Discreta	1	1
Pedro Martí García	564	11545	Análisis Matemático	2	2

- *Obtener el nombre de los profesores que no tienen docencia.*

  `((Profesor[dni] – Docencia[dni])`$\otimes_{dni}$ `Profesor)[nombre]`

1. Profesor[dni]: se proyecta la tabla *Profesor* sobre *dni* obteniendo una tabla que contiene los dni de todos los profesores

dni
111
123
453
564

2. Docencia[dni]: se proyecta la tabla *Docencia* sobre *dni* obteniendo una tabla que contiene los *dni* de todos los profesores que imparten docencia en alguna asignatura:

dni
111
123
564

3. Profesor[dni] – Docencia[dni]: se restan las tablas obtenidas en los pasos 1 y 2 obteniendo una tabla con los dni de profesores que no tienen docencia:

dni
453

4. (Profesor[dni] – Docencia[dni])$\otimes_{dni}$ Profesor: se concatena la tabla del paso 3 con la tabla *Profesor* para obtener el nombre de los profesores:

dni	nombre	teléfono	cod_dep	provincia	edad
453	Elisa Rojo Amando	7859	DSIC	Valencia	26

5. ((Profesor[dni] – Docencia[dni])$\otimes_{dni}$ Profesor)[nombre]: se proyecta la tabla obtenida en el paso 4 para eliminar las columnas que no interesan, siendo el resultado final el siguiente:

nombre
Elisa Rojo Amando

3 El Modelo Relacional: presentación formal

En este apartado se presentan todos los conceptos vistos hasta ahora pero ya con rigor y utilizando los términos correctos del modelo. Para facilitar la comprensión, en la tabla siguiente se puede ver la equivalencia de términos:

Término Informal	*Término Formal*
Tabla	Relación
Fila	Tupla
Columna	Atributo
Valores posibles	Tipo de datos

Con esta equivalencia de términos cuando, por ejemplo, se haga referencia a una *relación* se debe tener en cuenta que, informalmente se hablaría de una *tabla*.

3.1 Tipos de datos

Un concepto fundamental en el ámbito de la informática es el de *tipo de datos.* En los lenguajes de programación el tipo de dato es una característica de los datos que indica al ordenador (y/o al programador) qué valores pueden tomar y qué operaciones se pueden realizar con ellos.

En un sentido amplio, un tipo de datos define un conjunto de valores y las operaciones sobre estos valores. Casi todos los lenguajes de programación explícitamente incluyen la notación del tipo de datos, aunque lenguajes diferentes pueden usar terminología diferente.

Los sistemas de gestión de bases de datos comerciales proporcionan una lista amplia de tipos de datos a elegir y aunque pueden tener nombres distintos, todos ellos incluyen tipos de datos semejantes a los

siguientes:

- Numéricos: almacenan datos de tipo numérico, pueden ser enteros (*integer, smallint,…*), reales (*numeric, number, real, float,…*).
- Alfanuméricos: almacenan cadenas (secuencias) de caracteres (*char, varchar,…*). Normalmente se expresan entre comillas simples (p.e. 'Pepe').
- Fecha: almacena fechas (*date*).
- …

El asociar un determinado tipo de datos a una columna de una tabla implica limitar qué valores pueden aparecer en esa columna y qué operaciones se podrán realizar sobre esos valores. Así, si se considera la tabla *Docencia* presentada en el primer apartado del tema:

Docencia			
dni	**cod_asg**	**gteo**	**gpra**
111	11547	1	3
123	11545	0	2
123	11547	1	1
564	11545	2	2

Los tipos de datos de cada columna podrían ser los siguientes:

- *dni:* cadena de 9 caracteres (char(9))
- *cod_asg*: cadena de 5 caracteres (char(5)). Aunque podría pensarse en un tipo de datos numérico, las operaciones asociadas (+, –, …) no tienen sentido para esta columna que almacena códigos de asignatura.
- *gteo* y *gpra*: enteros cortos (*entero).*

Cuando se esté trabajando con un sistema de gestión de bases de datos, una de las primeras tareas a realizar será estudiar qué tipos de datos proporciona.

Un concepto muy estudiado en bases de datos aunque aún no es proporcionada por los sistemas comerciales es el concepto de *dominio*. Un dominio es un tipo de datos definido por el diseñador de la base de datos con unas características concretas. En el ejemplo, el dominio más adecuado para el semestre de una asignatura sería el enumerado {'1A', '1B', '2A', '2B', '3A', '3B', '4A' '4B'}.

3.2 Tupla y Relación

Recuérdese que:

- Tabla ≡ Relación
- Fila ≡ Tupla
- Columna ≡ Atributo

El Modelo Relacional proporciona dos estructuras de datos para almacenar la información de un sistema, la *tupla* y la *relación*. La estructura tupla coincide con el tipo de datos *registro* presente en todos los lenguajes de programación mientras que la estructura relación es específica del Modelo Relacional y es la que lo caracteriza y le da nombre.

La definición de una base de datos para un sistema de información concreto implica definir qué relaciones se van a utilizar, para ello es necesario definir el *esquema de la relación* (también llamado *tipo de la relación*):

> Un esquema de relación es un conjunto de pares de la forma: $\{(A_1, T_1), (A_2, T_2), \ldots, (A_n, T_n)\}$ donde $\forall i \forall j\ (i \neq j \rightarrow A_i \neq A_j)$.

En la definición, $\{A_1, A_2, \ldots, A_n\}$ $(n > 0)$ es el conjunto de nombres de *atributos*, y $T_1, T_2, \ldots, T_n$ son los tipos de datos asociados a dichos atributos.

> Una tupla del esquema de relación $\{(A_1, T_1), (A_2, T_2), \ldots, (A_n, T_n)\}$ es un conjunto de pares de la forma $\{(A_1, v_1), (A_2, v_2), \ldots, (A_n, v_n)\}$ tal que $\forall i\ v_i \in T_i$.
>
> Una relación de esquema $\{(A_1, T_1), (A_2, T_2), \ldots, (A_n, T_n)\}$ es un conjunto de tuplas de dicho esquema.

Para hacer referencia al valor de los atributos de una tupla, se hará uso del operador *punto* (como en los registros); así, si *t* es la tupla $\{(A_1, v_1), (A_2, v_2), \ldots, (A_n, v_n)\}$ entonces $t.A_2$ hace referencia al valor del atributo A_2 de la tupla *t*. También se puede utilizar la notación con paréntesis, es decir $t.A_2 \cong t(A_2)$.

Concretando todas estas ideas con la relación *Docencia:*

- Esquema de la relación *Docencia:*
  {(dni, char(9)), (cod_asg, char(5)), (gteo, entero), (gpra, entero)}
- Tuplas del esquema de la relación *Docencia:*
  - {(dni, '123'), (cod_asg, '11545'), (gteo,0), (gpra,2)} es una posible tupla de ese esquema.
  - {(dni, '123'), (cod_asg, '11545'), (gteo, 'X'), (gpra,2)} no es una tupla ya que el atributo *gteo* tiene asociado un valor que no pertenece a su tipo de datos.
- Relación del esquema *Docencia:* {{(dni, '123'), (cod_asg, '11545'), (gteo,0), (gpra,2)}, {(dni, '123'), (gteo,1), (gpra,1) , (cod_asg, '11547')}, {(dni, '111'), (gteo,1), (cod_asg, '11547'), (gpra,3)}, {(dni, '564'), (cod_asg, '11545'), (gteo,2), (gpra,2)}}

Obsérvese que como una tupla es un conjunto de pares, el orden en que éstos aparecen es irrelevante (en un conjunto no hay orden).

> Se denomina *grado* de una relación al número de atributos de su esquema, y *cardinalidad* de una relación al número de tuplas que la forman.

Por comodidad, la notación que se va a utilizar a lo largo del texto para declarar una relación *R* de esquema $\{(A_1, T_1), (A_2, T_2), \ldots, (A_n, T_n)\}$ va a ser a partir de este momento la siguiente:

$$R(A_1{:}T_1, A_2{:}\ T_2, \ldots, A_n{:}T_n).$$

Así los esquemas de las relaciones del ejemplo serían los siguientes:

- Departamento(cod_dep: char(4), nombre: char(50), teléfono: char(8), director: char(9))
- Asignatura(cod_asg: char(5), nombre: char(50), semestre: char(2), cod_dep: char(4), teoría: real, prácticas: real)
- Profesor(dni: char(9), nombre: char(80), teléfono: char(8), cod_dep: char(4), provincia: char(25), edad: entero)
- Docencia(dni: char(9), cod_asg: char(5), gteo: entero, gpra: entero)

La estructura de datos tupla permite representar objetos del mundo real a través de sus propiedades (atributos). Así, en el ejemplo, cada tupla del esquema *Asignatura* representa una asignatura través de sus propiedades: código, nombre, semestre, código del departamento al que pertenece y créditos teóricos y prácticos en el plan de estudios.

La estructura de datos relación permite representar el conjunto de ocurrencias de un mismo tipo de objeto. Así, la siguiente relación *Asignatura*:

{{(cod_asg, 11545), (nombre, Análisis Matemático), (semestre, 1A), (cod_dep, DMA), (teoría, 4,5), (prácticas, 1,5)},
{(cod_asg, 11546), (nombre, Álgebra), (semestre, 1B), (cod_dep, DMA), (teoría, 4,5), (prácticas, 1,5)},
{(cod_asg, 11547), (nombre, Matemática Discreta), (semestre, 1A), (cod_dep, DMA), (teoría, 4,5), (prácticas, 1,5)},
{(cod_asg, 11548), (nombre, Bases de Datos y Sistemas de Información), (semestre, 3A), (cod_dep, DSIC), (teoría, 4,5), (prácticas, 1,5)}

contiene la información relativa a cuatro asignaturas.

Es obvio que la representación de una relación como conjunto de tuplas resulta tediosa y poco clara; una forma más cómoda y sencilla de representar gráficamente una relación es mediante una tabla en la que cada fila representa una tupla de la relación y cada columna está etiquetada con un nombre de atributo. La relación *Asignatura* se puede ver tabularmente tal y como se mostró al principio del documento:

Asignatura					
cod_asg	**nombre**	**semestre**	**cod_dep**	**teoría**	**prácticas**
11545	Análisis Matemático	1A	DMA	4,5	1,5
11546	Álgebra	1B	DMA	4,5	1,5
11547	Matemática Discreta	1A	DMA	4,5	1,5
11548	Bases de Datos y Sistemas de Información	3A	DSIC	4,5	1,5

Así, una base de datos podrá ser siempre representada como un conjunto de tablas. Esto, sin embargo, no debe inducir a confundir la estructura relación con la estructura tabla, que suele ser su representación más habitual. Existen diferencias claras entre ambos conceptos: en la tabla existe un orden entre sus filas, y también un orden entre los elementos de las filas, que no existen en una relación.

Para terminar este apartado se introduce el concepto de esquema lógico y se matiza el concepto de base de datos:

> El conjunto de esquema de relación que representa un sistema de información se denomina *esquema lógico (relacional)* y los valores (o extensiones) de las relaciones del esquema en un instante determinado constituyen la *base de datos*.

3.3 Información faltante: valor nulo

De la definición de tupla como conjunto de pares atributo-valor dada en el apartado anterior se deduce que en una tupla $\{(A_1, v_1), (A_2, v_2), \ldots, (A_n, v_n)\}$ todos los atributos tienen necesariamente un valor asociado. Es decir, en una tupla t de una relación R si el valor asociado al atributo A_i es v_i, entonces v_i pertenece al tipo de datos T_i asociado a A_i en el esquema de la relación R.

Esta definición de tupla en la que todo atributo tiene necesariamente un valor asociado, no permite contemplar el caso en el que se desconoce el valor de un atributo en una tupla, situación que es frecuente cuando se representan objetos del mundo real. Por ello, y para superar esta limitación, se va a introducir la hipótesis de la existencia de un valor especial llamado *valor nulo* en todo tipo de datos de los esquemas de las relaciones. Este valor denota la ausencia de información, es decir, el desconocimiento de la propiedad representada por el atributo para el objeto representado por la tupla; cuando esto sucede, se dice que dicho atributo tiene *valor nulo* en esa tupla (o simplemente es *nulo*). Es importante destacar que el valor nulo tiene un comportamiento especial frente a los operadores, existiendo predicados específicos para su manipulación. Además, no existe un símbolo de constante para denotarlo pero, por claridad, a partir de ahora, en el texto se representará con el carácter ?.

Así, la información de la profesora Luisa Bos, que no se podía representar por una tupla del esquema *Profesor* al desconocerse su teléfono, con la introducción del valor nulo, se puede representar por la tupla: {(dni, '111'), (nombre, 'Luisa Bos Pérez'), (teléfono, ?), (cod_dep, 'DMA'), (provincia, 'Alicante'), (edad,33)}.

La existencia del valor nulo obliga a utilizar una lógica trivaluada, donde los valores de verdad de cualquier comparación son tres: *cierto, falso* e *indefinido*. El valor *indefinido* aparece al considerar el valor nulo en la evaluación de los predicados de comparación (=, <>, <=, >=, <, >).

3.3.1 Lógica trivaluada

La evaluación de los predicados de comparación debe ser reconsiderada para tener en cuenta la existencia de valores nulos; la regla de evaluación será la siguiente:

> Sea A_i α A_j una comparación, donde A_i y A_j son nombres de atributos de tuplas o

constantes y α es un predicado de comparación; entonces, la comparación se evalúa a *indefinido* para una tupla *t* si alguno de los dos operandos, A_i o A_j, tiene valor nulo en *t*; en caso contrario se evalúa al valor de verdad de la comparación según la semántica del predicado α.

Sea *t* la tupla {(dni, '111'), (nombre, 'Luisa Bos Pérez'), (teléfono, ?),(cod_dep, 'DMA'), (provincia, 'Alicante'), (edad,33)} entonces:

- t.dni = '111' se evalúa a cierto.
- t.nombre < 'AAA' se evalúa a falso.
- t.teléfono= '55544' se evalúa a indefinido.

Las tablas de verdad de las conectivas lógicas también deberán ser reconsideradas al incorporar el valor indefinido. Estas tablas son:

G	H	G ∧ H	G ∨ H
falso	falso	falso	falso
indefinido	falso	falso	indefinido
cierto	falso	falso	cierto
falso	indefinido	falso	indefinido
indefinido	indefinido	indefinido	indefinido
cierto	indefinido	indefinido	cierto
falso	cierto	falso	cierto
indefinido	cierto	indefinido	cierto
cierto	cierto	cierto	cierto

G	¬G
falso	cierto
indefinido	indefinido
cierto	falso

Por último, ya que el valor nulo representa la ausencia de valor, será necesario un predicado que permita comprobar si un atributo tiene valor nulo en una tupla. Éste es el predicado *Nulo*.

- Sintaxis: Nulo(A_i), donde A_i es un nombre de atributo.
- Semántica: Nulo(A_i) se evalúa a cierto para una tupla *t*, si A_i tiene valor nulo en *t*; en caso contrario se evalúa a falso.

Sea de nuevo *t* la tupla {(dni, '111'), (nombre, 'Luisa Bos Pérez'), (teléfono, ?), (cod_dep, 'DMA'), (provincia, 'Alicante'), (edad,33)} entonces:

- Nulo(t.dni) se evalúa a falso.
- Nulo(t.teléfono) se evalúa a cierto.

3.4 Restricciones de integridad

Si la definición de la base de datos se limita a especificar qué esquemas de relación la componen, cómo se llaman sus atributos y qué tipo de datos puede aparecer en cada una de ellas, podría llegarse a una situación en la que la información almacenada en ellas no fuera *válida*, es decir que no estuviera representando correctamente la información del sistema para el que se ha diseñado.

Supóngase que en la base de datos de docencia antes presentada, en un momento determinado se llega a tener la siguiente información (a partir de aquí las relaciones de una base de datos se presentarán tabularmente):

Departamento			
cod_dep	**nombre**	**teléfono**	**director**
DLA	Lingüística Aplicada	2255	111
DMA	Matemática Aplicada	1256	
DSIC	Sistemas Informáticos y Computación	1542	453

Asignatura					
cod_asg	**nombre**	**semestre**	**cod_dep**	**teoría**	**prácticas**
11545	Análisis Matemático	1A	DMA	4,5	1,5
11546	Análisis Matemático	1B	DMA	4,5	1,5
11547	Matemática Discreta	1A	DTA	4,5	1,5
11548	Bases de Datos y Sistemas de Información	3A	DSIC	4,5	1,5

Profesor					
dni	**nombre**	**teléfono**	**cod_dep**	**provincia**	**edad**
111	Luisa Bos Pérez		DMA	Alicante	33
453	Elisa Rojo Amando	7859	DSIC	Valencia	26
111	Juana Cerdá Pérez	3222	DMA	Valencia	50
564		3412	DMA	Castellón	27

Docencia			
dni	**cod_asg**	**gteo**	**gpra**
111	11547	1	3
123	11545	0	2
123	11547	1	1
564	7777	2	2

Asumiendo la interpretación obvia de este esquema, las tuplas presentes en las relaciones contiene datos que no se corresponden con situaciones reales; las anomalías que se observan son:

- En la relación *Profesor* hay dos profesores con el mismo *dni.* ¿Cómo se distinguirían en la relación *Docencia* si tienen el mismo código?
- En la relación *Asignatura* aparece un código de departamento que no aparece en la relación *Departamento.* ¿A qué departamento pertenece esa asignatura?
- En la relación *Asignatura* hay dos asignaturas con el mismo nombre. ¿Es esto posible entre las asignaturas de un mismo título?
- En la relación *Profesor* hay un profesor sin nombre. ¿Es esto razonable?
- En la relación *Docencia* aparece un código de asignatura que no aparece en la relación *Asignatura*. ¿A qué asignatura se hace referencia con el código 77777?

Para evitar estos problemas y aumentar la capacidad expresiva de una base de datos relacional, al definir las relaciones que la componen hay que especificar también qué propiedades debe cumplir la información almacenada en ellas. Estas propiedades, en el ejemplo, serían las siguientes:

- Relación *Departamento:*
  - No puede haber dos tuplas con el mismo valor en el atributo *cod_dep.*
  - No puede haber tuplas con valor nulo en los atributos *cod_dep o nombre*.
  - Todo valor que aparece en el atributo *dni* debe aparecer en el atributo *dni* de la relación *Profesor.*
- Relación *Asignatura:*
  - No puede haber dos tuplas con el mismo valor en el atributo *cod_asg* o en el atributo *nombre*.
  - Todo valor que aparece en el atributo *cod_dep* debe aparecer en el atributo *cod_dep* de la relación *Departamento.*
  - No puede haber tuplas con valor nulo en los atributos *cod_asg, nombre*, *semestre, cod_dep, teoría* o *prácticas.*

- Relación *Profesor:*
  - No puede haber dos tuplas con el mismo valor en el atributo *dni.*
  - Todo valor que aparece en el atributo *cod_dep* debe aparecer en el atributo *cod_dep* de la relación *Departamento.*
  - No puede haber tuplas con valor nulo en los atributos *dni, nombre* o *cod_dep*.
- Relación *Docencia:*
  - No puede haber dos tuplas con el mismo valor a la vez en los atributos *dni* y *cod_asg* (que un profesor imparte una asignatura sólo interesa almacenarlo una vez).
  - Todo valor que aparece en el atributo *dni* debe aparecer en el atributo *dni* de la relación *Profesor.*
  - Todo valor que aparece en el atributo *cod_asg* debe aparecer en el atributo *cod_asg* de la relación *Asignatura.*
  - No puede haber tuplas con valor nulo en los atributos *gteo* o *gpra.*

Estas propiedades se incluyen en el esquema de la base de datos especificando *restricciones de integridad*.

> Una restricción de integridad representa una propiedad del mundo real del cual la base de datos es una representación.

Las restricciones de integridad se especifican en el esquema de la base de datos, y se deben cumplir en cualquier extensión del mismo; el responsable de que esto sea así es el SGBD, que debe velar en todo momento por la integridad de la información almacenada.

> Si una base de datos cumple una restricción de integridad, se dice que la base de datos *satisface* la restricción de integridad. Si una base de datos no cumple una restricción de integridad, se dice que la base de datos *viola* la restricción de integridad.

En el Modelo Relacional se contemplan cuatro tipos de restricciones de integridad:

- Restricción de valor no nulo.
- Restricción de unicidad.
- Restricción de clave primaria.
- Restricción de integridad referencial.

Estos cuatro tipos de restricciones permitirían definir sobre el esquema de docencia las siguientes propiedades observables en el mundo real:

- Relación *Departamento*:
  - El atributo *cod_dep* identifica unívocamente a un departamento y es utilizado internamente en la organización para realizar esa función (restricción de clave primaria).
  - De todo departamento se debe conocer el valor de los atributos *cod_dep* y *nombre* (restricciones de valor no nulo).
  - Todos los profesores a los que se hace referencia con el atributo *dni* deben aparecer en la relación *Profesor* (restricción de integridad referencial).
- Relación *Asignatura*:
  - El atributo *cod_asg* identifica unívocamente a una asignatura y es utilizado internamente en la organización para realizar esa función (restricción de clave primaria).
  - No puede haber dos asignaturas con el mismo valor en el atributo *nombre* (restricción de unicidad).
  - Todos los departamentos a los que se hace referencia con el atributo *cod_dep* deben aparecer en la relación *Departamento* (restricción de integridad referencial).
  - De toda asignatura se debe conocer el valor de los atributos *cod_asg, nombre, semestre,*

cod_dep, teoría y *prácticas* (restricciones de valor no nulo).

- Relación *Profesor:*
  - El atributo *dni* identifica unívocamente a un profesor y es utilizado internamente en la organización para realizar esa función (restricción de clave primaria).
  - Todos los departamentos a los que se hace referencia con el atributo *cod_dep* deben aparecer en la relación *Departamento* (restricción de integridad referencial).
  - De todo profesor se debe conocer el valor de los atributos *dni, nombre* y *cod_dep* (restricciones de valor no nulo).
- Relación Docencia:
  - El par de atributos {*dni, cod_asg*} identifica unívocamente a una docencia y es utilizado internamente en la organización para realizar esa función (restricción de clave primaria).
  - Todos los profesores a los que se hace referencia con el atributo *dni* deben aparecer en la relación *Profesor* (restricción de integridad referencial).
  - Todas las asignaturas a los que se hace referencia con el atributo *cod_asg* deben aparecer en la relación *Asignatura* (restricción de integridad referencial).
  - No puede haber tuplas sin valor en los atributos *gteo* o *gpra* (restricciones de valor no nulo).

Con estos cuatro tipos de restricciones de integridad se van a poder representar muchas de las propiedades que se detectan en un sistema de información pero no todas, más adelante se verá qué hacer con las que no quedan representadas.

A continuación se presentan con detalle y rigor estas restricciones. Para ello, sean:

- R una relación de esquema $\rho = \{(A_1, T_{\rho 1}), (A_2, T_{\rho 2}), \ldots, (A_n, T_{\rho n})\}$.
- S una relación de esquema $\sigma = \{(B_1, T_{\sigma 1}), (B_2, T_{\sigma 2}), \ldots, (B_m, T_{\sigma m})\}$.
- A_ρ el conjunto de nombres de atributo de ρ, $A_\rho = \{A_1, A_2, \ldots, A_n\}$.
- A_σ el conjunto de nombres de atributo de σ, $A_\sigma = \{B_1, B_2, \ldots, B_m\}$.

3.4.1 Restricción de valor no nulo

La definición de una *restricción de valor no nulo* sobre un conjunto de atributos K de la relación R expresa la siguiente propiedad: "No debe haber en R una tupla que tenga el valor nulo en algún atributo de K".[4]

Formalmente, dado un conjunto de atributos K ($K \subseteq A_\rho$ y $K \neq \varnothing$), se dice que R satisface una restricción de valor no nulo sobre K si y sólo si se cumple la siguiente propiedad:

$$\forall t \ (t \in R \rightarrow \neg \exists A_i \ (A_i \in K \wedge \text{nulo}(t.A_i)))$$

en caso contrario, R viola esta restricción.

En la base de datos de docencia, y de acuerdo con las propiedades enunciadas anteriormente, los atributos que deberían definirse con restricción de valor no nulo son: *cod_dep* y *nombre* de la relación *Departamento; cod_asg, nombre, semestre, cod_dep, teoría* y *prácticas* de la relación *Asignatura; dni, nombre* y *cod_dep* de la relación *Profesor*; *y dni, cod_asg, gteo* y *gpra* de la relación *Docencia*.

Hay que destacar que la definición de esta propiedad para un atributo es una decisión del diseñador de la base de datos. En este ejemplo se ha considerado que se debe conocer el nombre de cada profesor, así como su código y departamento permitiéndose, sin embargo, el desconocimiento de su provincia, edad y teléfono.

3.4.2 Restricción de unicidad

La definición de una *restricción de unicidad* sobre un conjunto de atributos K de la relación R expresa la

[4] La expresión *no nulo* no es correcta en castellano pero se ha utilizado por comodidad y para mantener la proximidad con la expresión inglesa *not null.*

siguiente propiedad: "No debe haber en *R* dos tuplas que tengan el mismo valor en todos los atributos del conjunto *K*".

Formalmente, dado un conjunto de atributos *K* ($K \subseteq A_\rho$ y $K \neq \emptyset$), se dice que *R* satisface una restricción de unicidad sobre *K* si y sólo si se cumple la siguiente propiedad:

$$\neg\exists t_1 \exists t_2 \ (t_1 \in R \wedge t_2 \in R \wedge t_1 \neq t_2 \wedge \forall A_i \ (A_i \in K \rightarrow t_1.A_i = t_2.A_i))$$

en caso contrario, *R* viola esta restricción.

En la base de datos de docencia, y de acuerdo con las propiedades enunciadas anteriormente, los atributos que deberían definirse con restricción de unicidad son: *cod_dep* de la relación *Departamento; cod_asg* y *nombre* de la relación *Asignatura*; *dni* de la relación *Profesor*; y el conjunto {*dni, cod_asg*} en la relación *Docencia*.

3.4.3 Concepto de clave primaria. Integridad de clave primaria

La definición de relación como conjunto de tuplas implica que en una relación no existen dos tuplas iguales[5]; es decir, que cualquier tupla es distinta de cualquier otra, y que por lo tanto es identificable de forma única dando los valores de todos sus atributos. Aunque este hecho asegura la posibilidad de identificar y seleccionar las tuplas de una relación con el objeto de poder manipularlas, evidentemente no parece la forma más cómoda de hacerlo, ya que generalmente existen en las relaciones subconjuntos de atributos que, debido a su significado, pueden asegurar también esta identificación única. Por ello, y con el fin de facilitar la manipulación de las relaciones en una base de datos relacional, se introduce en el modelo el concepto *clave primaria*.

> Una clave primaria de una relación es un conjunto de atributos de su esquema que es elegido para servir de identificador unívoco de sus tuplas.

Evidentemente, para que la clave primaria cumpla con esta función de identificación deberá cumplir ciertos requisitos:

- Sus atributos deberán tener siempre un valor para cada tupla (es decir, no podrán tener valor nulo), y
- El valor de la clave primaria deberá ser único para cada tupla.

Además, y como orientación de diseño, para que la clave sea cómoda y manejable debe elegirse una que sea mínima, es decir que todos los atributos que la forman sean necesarios para la función de identificación. A continuación se presenta formalmente el concepto de clave primaria.

Formalmente, dado un conjunto de atributos *CP* ($CP \subseteq A_\rho$ y $CP \neq \emptyset$), que se ha definido como clave primaria de *R,* se dice que *R* satisface la *restricción de integridad de clave primaria* si se cumplen las siguientes propiedades:

- *R* satisface una restricción de valor no nulo sobre *CP*.
- *R* satisface una restricción de unicidad sobre *CP*.

en caso contrario, *R* viola esta restricción.

Además, es aconsejable que *CP* sea mínima, es decir, que no tenga ningún subconjunto propio que pueda ser a su vez clave primaria de R.

Una relación sólo puede tener definida una clave primaria; si hay más de un subconjunto de atributos que cumpla las propiedades exigidas, se elige uno ellos[6] y a los restantes se les especifican las propiedades anteriores (unicidad y valor no nulo), pero sin otorgarles la categoría de clave primaria.

La importancia de la clave primaria de una relación reside en el hecho de que, como ya se ha dicho, proporciona una identificación unívoca de sus tuplas; es decir, si *CP* es la clave primaria de *R,* entonces toda

[5] Recuérdese que de acuerdo al concepto matemático de *conjunto*, en un conjunto no hay elementos repetidos.

[6] Esta elección es realizada por el diseñador de la base de datos basándose en el significado de los atributos.

tupla de *R* contiene siempre un valor de *CP* distinto al de cualquier otra tupla y que no es nulo; este hecho justifica que para hacer referencia a una tupla de una relación desde otra relación se utilice, usualmente, el valor de la clave primaria de la tupla; por este motivo, es recomendable que en la definición de las relaciones se especifique siempre una clave primaria.

En la base de datos de docencia se deberían definir las siguientes claves primarias

- En la relación *Departamento* la clave primaria sólo puede ser el conjunto {*cod_dep*}.
- En la relación *Asignatura* hay dos posibles claves primarias, los conjuntos {*cod_asg*} y {*nombre*}, ya que ambos cumplen las condiciones exigidas. En este caso, parece más adecuado elegir como clave primaria {*cod_asg*}, ya que es cómodo que este atributo sea el identificador interno para las asignaturas.
- En la relación *Profesor* la clave primaria sólo puede ser el conjunto {*dni*}.
- En la relación Doc*encia* la clave primaria sólo puede ser el conjunto {*dni, cod_asg*}.

3.4.4 Concepto de clave ajena. Integridad referencial

3.4.4.1 Clave ajena con un solo atributo

En este apartado se presenta la restricción de integridad referencial que afecta a un sólo un atributo. En el apartado siguiente se generaliza la propiedad a claves ajenas con más de un atributo.

Las claves ajenas son el mecanismo que proporciona el Modelo Relacional para expresar asociaciones entre los objetos representados por las relaciones del esquema de la base de datos. La forma de hacerlo consiste en incluir en el esquema de una relación *S* atributos identificadores de otra relación *R*; a este conjunto de atributos se le conoce como *clave ajena* de la relación *S* que hace referencia a la relación *R*. Para que la clave ajena cumpla su función de referencia se debe asegurar que los valores que toman sus atributos en las tuplas de *S* aparecen en alguna tupla de *R*.

Formalmente, una clave ajena de *S* que hace referencia a *R* se define como sigue:

- Un atributo de A_σ, sea B_i y un atributo de A_ρ, sea A_j, tal que:
  - A_j tiene restricción de unicidad o es la clave primaria de *R*, y
  - B_i y A_j tienen el mismo tipo de datos (i.e. $T_{\sigma i} = T_{\rho j}$).

Entonces, se dice que *S* satisface la *restricción de integridad referencial*[7] si se cumple la siguiente propiedad:

$$\forall t(t \in S \rightarrow (\text{nulo}(t.B_i) \vee \exists m(m \in R \wedge t.B_i = m.A_j)))$$

en caso contrario, *S* viola esta restricción.

Dicho de otra manera, los valores no nulos que aparecen en el atributo B_i de *S*, aparecen también en el atributo A_j de *R*.

En el ejemplo de docencia:

- El atributo *director* de la relación *Departamento* debe definirse como una clave ajena a la relación *Profesor.*
- El atributo *cod_dep* de la relación *Profesor* debe definirse como una clave ajena a la relación *Departamento.*
- El atributo *cod_dep* de la relación *Asignatura* debe definirse como una clave ajena a la relación *Departamento.*
- El atributo *dni* de la relación *Docencia* debe definirse como una clave ajena a la relación *Profesor.*
- El atributo *cod_asg* de la relación *Docencia* debe definirse como una clave ajena a la relación *Asignatura.*

Con la definición de estas propiedades, dado que se debe cumplir la integridad referencial, nunca podrán aparecer en la relación *Departamento* valores en el atributo *director* que no se correspondan con un profesor

[7] O simplemente integridad referencial.

ni en las relaciones *Profesor* o *Asignatura* valores en el atributo *cod_dep* que no se correspondan con códigos de un departamento existente en la base de datos. De igual manera, tampoco podrán aparecer en la relación *Docencia* valores en el atributo *dni* que no se correspondan con un profesor o valores en el atributo *cod_asg* que no sean el de alguna de las asignaturas presentes en la base de datos.

3.4.4.2 Clave ajena general

En este apartado se generaliza la definición de clave ajena teniendo en cuenta que ésta puede estar formada por más de un atributo. El problema en este caso es definir qué significa la integridad referencial cuando parte de la clave ajena es nula y otra parte no.

Formalmente, una clave ajena *CA* de *S* que hace referencia a *R* se define como:

1. Un subconjunto no vacío de atributos de A_σ, sea $K=\{B_i, B_j, \dots, B_k\}$, un subconjunto de atributos de A_ρ, sea $J=\{A_i, A_j, \dots, A_k\}$ y una biyección $f : K \rightarrow J$ que empareja los atributos de *K* y *J* ($f(B_i)=A_i$, $f(B_j)=A_j \dots, f(B_k)=A_k$) tal que:
   - *J* tiene restricción de unicidad o es la clave primaria de *R*, y
   - $\forall B_i(B_i \in K \rightarrow B_i$ y A_i tienen el mismo tipo de datos).
2. Un tipo de integridad referencial, que puede ser, *débil*, *parcial* o *completa*

Entonces, se dice que *S* satisface la *restricción de integridad referencial* sobre *CA* si, según el tipo elegido, se cumple la propiedad que se indica a continuación:

- Integridad referencial débil:
$$\forall t(t\in S \rightarrow (\exists B_i(B_i\in K \wedge nulo(t.B_i)) \vee \exists m(m\in R \wedge \forall B_i(B_i\in K \rightarrow t.B_i=m.A_i))))$$
- Integridad referencial parcial:
$$\forall t(t\in S \rightarrow (\forall B_i(B_i\in K \rightarrow nulo(t.B_i)) \vee \exists m(m\in R \wedge \forall B_i((B_i\in K \wedge \neg nulo(t.B_i))\rightarrow t.B_i=m.A_i))))$$
- Integridad referencial completa:
$$\forall t(t\in S \rightarrow (\forall B_i(B_i\in K \rightarrow nulo(t.B_i)) \vee \exists m(m\in R \wedge \forall B_i(B_i\in K \rightarrow (\neg nulo(t.B_i)\wedge t.B_i=m.A_i)))))$$

en caso contrario, *S* viola esta restricción.

A pesar de que, en general, el lenguaje natural es ambiguo (y por tanto no resulta sencillo dar una explicación de estas propiedades), para relajar su presentación formal, a continuación se enuncian de forma intuitiva los tres tipos de integridad referencial:

- *Integridad referencial débil*: si en una tupla de *S* todos los valores de los atributos de *S* tienen un valor que no es nulo, entonces debe existir una tupla en *R* que tome esos mismos valores en los atributos de *J* (es decir que, cuando hay al menos un atributo de *K* con valor nulo, la integridad referencial se satisface).
- *Integridad referencial parcial*: si en una tupla de *S* algún atributo de *K* tiene un valor que no es nulo, entonces debe existir una tupla en *R* que tenga en los correspondientes atributos de *J* los mismos valores que los atributos de *K* que no son nulos.
- *Integridad referencial completa*: en una tupla de *S* todos los atributos de *K* deben tener valor nulo, o bien todos deben tener un valor que no es nulo, en cuyo caso debe existir una tupla en *R* que tome en los correspondientes atributos de *J* los mismos valores que los atributos de *K*

Cuando se define una clave ajena en una relación, se debe especificar el tipo de integridad referencial que se exige, excepto en caso de que la clave ajena conste sólo de un atributo o cuando todos ellos sufran restricción de valor no nulo, ya que en estos dos casos los tres tipos de integridad referencial coinciden.

La biyección que asocia los atributos de *S* con los de *R, s*e puede omitir cuando *J* es la clave primaria de *R* y se da uno de los siguientes casos:

- el conjunto *K* contiene un único atributo, o

- la biyección entre *K* y *J* se establece por igualdad en el nombre de los atributos.

En el ejemplo de *Docencia*, dado que todas las claves ajenas constan de un único atributo y la referencia se hace a través de la clave primaria de la relación referida, se puede omitir la biyección.

Para terminar, hay que destacar que la elección de un tipo u otro de integridad referencial, cuando esto es posible, permite modelar diferentes realidades, sin embargo, desafortunadamente, hoy en día los sistemas de gestión de bases de datos comerciales sólo permiten la definición de la integridad referencial débil.

3.4.4.3 Restauración de la integridad referencial: directrices al SGBD

La comprobación de las restricciones es competencia del SGBD, que debe asegurar que la base de datos después de cada actualización permanece íntegra, es decir, que satisface todas las restricciones definidas en su esquema.

Generalmente, ante una actualización de la base de datos que viole una restricción, el SGBD suele rechazar la actualización devolviendo la base de datos al estado anterior, lo que se conoce como comportamiento *restrictivo*. En algunos casos, sin embargo, sería deseable un comportamiento menos rígido, que permitiera al diseñador incluir en el esquema de la base de datos qué debe hacer el sistema cuando se detecte la violación de una restricción. Esta posibilidad está contemplada en el Modelo Relacional sólo para la restricción de integridad referencial. En este caso, además de conocer los atributos que constituyen la clave ajena y la relación referida por ella, se le puede indicar al SGBD la directriz asociada a esa clave ajena, es decir, el comportamiento del sistema frente a actualizaciones de la base de datos que violen esa integridad referencial. A continuación, se presentan estas ideas con más detalle.

Supóngase que existen dos relaciones *R* y *S* tal que *S* tiene una clave ajena que hace referencia a *R*. El SGBD debe preservar la integridad referencial representada por esta clave ajena; por lo tanto, debe comprobar la restricción frente a actualizaciones del usuario sobre las relaciones *R* y *S* que puedan violar la integridad. Sólo hay cuatro posibles actualizaciones de la base de datos que puedan violar la integridad referencial de esa clave ajena:

- Insertar nuevas tuplas en *S.*
- Modificar el valor de algún atributo de la clave ajena en *S.*
- Borrar tuplas de la relación *R.*
- Modificar el valor de algún atributo de *R* al que se haga referencia en la clave ajena.

Para aclarar este punto, sea el siguiente esquema relacional en el que, dado que la clave ajena sólo tiene un atributo, el tipo de integridad referencial es indiferente:

- R(A: entero, B: carácter) con clave principal {*A*}.
- S(C: entero, A: entero) con clave principal {*C*} y con {*A*} como clave ajena a *R*.

Supóngase que en un momento determinado las relaciones *R* y *S* contienen las siguientes tuplas que satisfacen la integridad referencial:

R

A	B
1	a
2	b
3	c

S

C	A
11	1
12	?
13	1
14	2

- Si se añade a *S* una tupla con *C* =15 y *A* = 4, la base de datos quedaría como sigue:

R

A	B
1	a
2	b
3	c

S

C	A
11	1
12	?
13	1
14	2
15	4

Esta base de datos viola la integridad referencial ya que en *R* no hay ninguna tupla con el valor 4 en el atributo *A.*

- Si se cambia el valor del atributo *A* en la tupla de *S* con *C* = 11 y se pone un 4, la base de datos quedaría como sigue:

R

A	B
1	a
2	b
3	c

S

C	A
11	4
12	?
13	1
14	2

Esta base de datos viola la integridad referencial ya que en *R* no hay ninguna tupla con el valor 4 en el atributo *A.*

- Si se elimina en *R* la tupla con *A* =1, la base de datos quedaría como sigue:

R

A	B
2	b
3	c

S

C	A
11	1
12	?
13	1
14	2

Esta base de datos viola la integridad referencial ya que en *R* no hay ninguna tupla con el valor 1 en el atributo *A* y este valor aparece en el atributo *A* de dos tuplas de *S.*

- Si se cambia el valor de *A* en la tupla de *R* con *A* = 1 y se pone un 4, la base de datos quedaría como sigue:

R

A	B
4	a
2	b
3	c

S

C	A
11	1
12	?
13	1
14	2

Esta base de datos viola la integridad referencial ya que en *R* no hay ninguna tupla con el valor 1 en el atributo *A* y este valor aparece en el atributo *A* de dos tuplas de *S.*

Las dos primeras modificaciones, esto es las actualizaciones sobre la relación *S*, van a ser siempre rechazadas por el SGBD ya que no hay ninguna acción compensatoria que pueda restaurar la integridad y que pueda ser considerada razonable.

Sin embargo, las actualizaciones sobre la relación *R* que violen la integridad referencial podrán ser rechazadas por el SGBD o podrán ser aceptadas por éste, que, en este caso, deberá tomar las acciones adecuadas para restaurar la integridad. Estas acciones pueden ser alguna de las siguientes:

- Frente a borrados de tuplas de *R*: supóngase que se desea borrar una tupla *m* de *R* a la que hace referencia una tupla *t* de *S.* Hay tres posibles acciones a realizar por el SGBD:
  - *Borrado restrictivo:* rechazar la operación por conducir la base de datos a un estado en el que se viola la integridad referencial dejando la base de datos como estaba.
  - *Borrado a nulos:* realizar el borrado de *m* y modificar el valor de la clave ajena a valor nulo en toda tupla *t* que haga referencia a *m.* Si se elige esta opción, tras el borrado de la tupla de *R* con *A*=1 se generaría la siguiente base de datos:

R

A	B
2	b
3	c

S

C	A
11	?
12	?
13	?
14	2

- *Borrado en cascada:* realizar el borrado de *m* y el borrado de toda tupla *t* que haga referencia a *m*. Si se elige esta opción, tras el borrado de la tupla de *R* con *A*=1 se generaría la siguiente base de datos:

R

A	B
2	b
3	c

S

C	A
12	?
14	2

- Frente a modificaciones de los valores de *A* en tuplas de *R*: supóngase que se desea modificar el valor de *A* en una tupla *m* de *R* a la que hace referencia alguna tupla *t* de *S*. Existen tres posibles comportamientos a seguir por el SGBD:
  - *Modificación restrictiva:* rechazar la operación por conducir la base de datos a un estado en el que se viola la integridad referencial dejando la base de datos como estaba.
  - *Modificación a nulos:* realizar esta modificación del valor de *A* en *m* y actualizar el valor de la clave ajena a valor nulo en toda tupla *t* que haga referencia a *m*. Si se elige esta opción, tras cambiar en la tupla de *R* con *A*=1 el valor del atributo *A* poniendo un 4, se generaría la siguiente base de datos:

R

A	B
4	a
2	b
3	c

S

C	A
11	?
12	?
13	?
14	2

  - *Modificación en cascada:* realizar la modificación de *m* y desencadenar la modificación de cada tupla *t* que haga referencia a *m*, actualizando el valor en la clave ajena por el nuevo valor modificado en *m.* Si se elige esta opción, tras cambiar en la tupla de *R* con A=1 el valor del atributo *A* poniendo un 4, se generaría la siguiente base de datos:

R

A	B
4	a
2	b
3	c

S

C	A
11	4
12	?
13	4
14	2

Así pues, cuando se defina una clave ajena habrá que indicar la directriz de restauración de la integridad elegida para el borrado y para la modificación, indicando *en cascada, a nulos* o *restrictivo*. Esta última directriz que, como ya se ha dicho, le indica al sistema que rechace la operación si se viola la integridad referencial sin realizar ninguna acción compensatoria es la opción por defecto.

3.4.5 Otras restricciones de integridad

Dada la diversidad de sistemas de información, existen propiedades que no se pueden expresar con los cuatro tipos de restricciones de integridad proporcionados en el Modelo Relacional, sin embargo, en el lenguaje estándar SQL existen sentencias que permiten definir algunas restricciones más como son:

- Restricciones sencillas que sólo afectan a un atributo en una relación. En el ejemplo de docencia podría interesar indicar que el valor del atributo *semestre* de la relación *Asignatura* tiene que ser uno de los siguientes: {'1A', '1B', '2A', '2B', '3A', '3B', '4A' '4B'}.
- Restricciones sencillas que afecten a varios atributos de una misma relación. En el ejemplo de docencia se podría incluir la restricción que asegura que una asignatura no tiene más créditos

teóricos que prácticos (es decir que para cada tupla se cumple que *teoría* ≤ *prácticas*).

Pese a todas las restricciones introducidas hasta el momento, es posible, y bastante frecuente, que en el sistema de información para el que se está diseñando la base de datos, se den propiedades que no pueden expresarse mediante estas restricciones. Estas propiedades que no puedan expresarse en la definición del esquema relacional no podrán ser controladas por el SGBD y será el programador de la aplicación el que deba controlar desde programa su cumplimiento.[8] En el ejemplo podría existir la siguiente restricción: "*Todo profesor debe impartir docencia de al menos una asignatura*".

4 DEFINICIÓN DE UN ESQUEMA RELACIONAL

Resumiendo lo visto hasta el momento, la definición de una relación en el esquema de la base de datos se puede enriquecer con la definición de los tipos de restricciones presentados que se indicarán con las siguientes abreviaturas:

- VNN: restricción de valor no nulo.
- Uni: restricción de unicidad.
- CAj: clave ajena (incluyendo el tipo y las directrices de restauración de la integridad).
- CP: clave primaria.
- Restricciones de integridad sencillas (RI): restricciones de integridad enunciadas mediante expresiones sencillas sobre las filas de una tabla.
- Restricción de integridad general (RG): propiedad que afecta a varias tuplas de una relación o a varias relaciones. Se enuncia en lenguaje natural.

A continuación, se incluye el esquema relacional del ejemplo de docencia donde se muestra cómo se va a especificar un esquema relacional a partir de ahora:

```
Departamento(cod_dep: char(4), nombre: char(50), teléfono: char(8),
             director: char(9))
  CP:{cod_dep}
  VNN:{nombre}
  CAj:{director}→Profesor(dni)
        Borrado a nulos y Modificación en cascada

Asignatura(cod_asg: char(5), nombre: char(50), semestre: char(2),
           cod_dep: char(4), teoría: real, prácticas: real)
  CP:{cod_asg}
  VNN:{nombre,semestre,cod_dep,teoría,prácticas}
  Uni:{nombre}
  CAj:{cod_dep}→Departamento(cod_dep)
        Borrado restrictivo y Modificación en cascada
  RI1:(teoría≤prácticas)
  RI2:(semestre ∈{'1A','1B','2A','2B','3A','3B','4A'4B'})

Profesor(dni: char(9), nombre: char(80), teléfono: char(8), cod_dep: char(4),
         provincia: char(25), edad: entero)
  CP:{dni}
  VNN:{nombre, cod_dep}
  CAj:{cod_dep}→Departamento(cod_dep)
        Borrado restrictivo y Modificación en cascada

Docencia(dni: char(9), cod_asg: char(5), gteo: entero, gpra: entero)
  CP:{dni,cod_asg}
  CAj:{dni}→Profesor(dni)
        Borrado en cascada y Modificación en cascada
  CAj:{cod_asg}→Asignatura(cod_asg)
```

[8] Este tema queda fuera del alcance de esta asignatura.

```
            Borrado restrictivo y Modificación en cascada
    VNN:{gteo,gpra}

  Restricción general:
    RG1: "Todo profesor debe impartir docencia de al menos una asignatura".
```

Comentarios al esquema anterior:

- No es necesario indicar explícitamente que una clave primaria es no nula y única ya que la definición de clave primaria implica estas dos restricciones.
- La sintaxis exacta para expresar restricciones de integridad sencillas se estudiará en el SQL.
- En la definición de la clave ajena de *Profesor* a *Departamento* bastaría indicar "CAj: {cod_dep} → Departamento" sin indicar a qué a atributo de *Departamento* se hace referencia ya se llaman igual en las dos relaciones. Lo mismo se puede decir para la clave ajena a la relación *Departamento* en *Asignatura* o para las claves ajenas que hay en la relación *Docencia.*
- En la definición de la clave ajena de *Departamento* a *Profesor* no es necesario especificar el tipo de integridad referencial ya que sólo consta de un atributo. Lo mismo se puede decir para la clave ajena a la relación *Departamento* en *Profesor* y *Asignatura* o a las claves ajenas que hay en la relación *Docencia.*
- Las directrices de la clave ajena de *Departamento* a *Profesor* implican que si se borra un profesor que es director de un departamento, ese departamento pasará a no tener director y que, si cambia el valor del *dni* de un profesor en *Profesor*, este cambio se realizará también en *Departamento.*
- Las directrices de la clave ajena de *Profesor* a *Departamento* implican que no se podrá borrar un departamento si su código aparece en *Profesor* y que, si cambia el valor del código de un departamento en *Departamento*, este cambio se realizará también en *Profesor.*
- Las directrices de la clave ajena de *Asignatura* a *Departamento* implican que no se podrá borrar un departamento si su código aparece en *Asignatura* y que, si cambia el valor del código de un departamento en *Departamento*, este cambio se realizará también en *Asignatura.*
- Las directrices de la clave ajena de *Docencia* a *Asignatura* implican que no se podrá borrar una asignatura si su código aparece en *Docencia* y que, si cambia el valor del código de una asignatura en *Asignatura*, este cambio se realizará también en *Docencia.*
- Las directrices de la clave ajena de *Docencia* a *Profesor* implican que si se borra un profesor se eliminarán también todas las tuplas de *Docencia* en las que apareciera su *dni* y que, si cambia el valor del *dni* de un profesor en *Profesor*, este cambio se realizará también en *Docencia.*
- La notación elegida para la definición de este esquema relacional no es comprensible para los sistemas comerciales, para ello es necesario hacer uso del lenguaje SQL (cómo definir esquemas en SQL se estudiará más adelante).

5 CONCEPTO DE TRANSACCIÓN

El concepto de *transacción* es fundamental en la teoría de bases de datos, para justificar este hecho, supóngase que en la base de datos de docencia en un momento determinado contiene las siguientes filas:

Departamento			
cod_dep	**nombre**	**teléfono**	**director**
DMA	Matemática Aplicada	1256	
DSIC	Sistemas Informáticos y Computación	1542	453

Asignatura					
cod_asg	**nombre**	**semestre**	**cod_dep**	**teoría**	**prácticas**
11545	Análisis Matemático	1A	DMA	4,5	1,5
11546	Álgebra	1B	DMA	4,5	1,5
11547	Matemática Discreta	1A	DMA	4,5	1,5

Profesor					
dni	nombre	teléfono	cod_dep	provincia	edad
111	Luisa Bos Pérez		DMA	Alicante	33
123	Juana Cerdá Pérez	3222	DMA	Valencia	50
564	Pedro Martí García	3412	DMA	Castellón	27

Docencia			
dni	cod_asg	gteo	gpra
111	11547	1	3
123	11545	0	2
123	11547	1	1
564	11545	2	2

Y que se quiere introducir la siguiente información:

"Hay un nuevo profesor de dni '222', de nombre 'Armando Lacuesta Abad', del departamento de código 'DSIC', con teléfono 8564 del que se desconoce la provincia y la edad y que va a impartir un grupo de teoría y un grupo de prácticas de la asignatura de código '11546'".

Para introducir esa información en la base de datos hay que insertar en la relación *Profesor* la tupla: {(dni, '222'), (nombre, 'Armando Lacuesta Abad'), (cod_dep, 'DSIC'), (teléfono, 8564), (provincia, ?), (edad,?)} e insertar en la relación *Docencia* la tupla {(dni, '222'), (cod_asg, '11546'), (gteo, 1), (gpra, 1)}:

- Supóngase que primero se inserta la tupla en la relación *Profesor*. Una vez ejecutada la instrucción la base de datos quedaría como sigue:

Departamento			
cod_dep	nombre	teléfono	director
DMA	Matemática Aplicada	1256	
DSIC	Sistemas Informáticos y Computación	1542	453

Asignatura					
cod_asg	nombre	semestre	cod_dep	teoría	prácticas
11545	Análisis Matemático	1A	DMA	4,5	1,5
11546	Álgebra	1B	DMA	4,5	1,5
11547	Matemática Discreta	1A	DMA	4,5	1,5

Profesor					
dni	nombre	teléfono	cod_dep	provincia	edad
111	Luisa Bos Pérez		DMA	Alicante	33
123	Juana Cerdá Pérez	3222	DMA	Valencia	50
564	Pedro Martí García	3412	DMA	Castellón	27
222	Armando Lacuesta Abab	8564	DSIC		

Docencia			
dni	cod_asg	gteo	gpra
111	11547	1	3
123	11545	0	2
123	11547	1	1
564	11545	2	2

Ante este resultado, el SGBD rechaza la modificación porque se viola la restricción de integridad

general RG_1 ya que el profesor recién añadido no imparte ninguna asignatura (su dni no aparece en la relación *Docencia*).

- Supóngase que primero se inserta la tupla en la relación *Docencia.* Una vez ejecutada la instrucción la base de datos queda como se muestra:

Departamento			
cod_dep	**nombre**	**teléfono**	**director**
DMA	Matemática Aplicada	1256	
DSIC	Sistemas Informáticos y Computación	1542	453

Asignatura					
cod_asg	**nombre**	**semestre**	**cod_dep**	**teoría**	**prácticas**
11545	Análisis Matemático	1A	DMA	4,5	1,5
11546	Álgebra	1B	DMA	4,5	1,5
11547	Matemática Discreta	1A	DMA	4,5	1,5

Profesor					
dni	**nombre**	**teléfono**	**cod_dep**	**provincia**	**edad**
111	Luisa Bos Pérez		DMA	Alicante	33
123	Juana Cerdá Pérez	3222	DMA	Valencia	50
564	Pedro Martí García	3412	DMA	Castellón	27

Docencia			
dni	**cod_asg**	**gteo**	**gpra**
111	11547	1	3
123	11545	0	2
123	11547	1	1
564	11545	2	2
222	11546	1	1

De nuevo, ante este resultado, el SGBD rechaza la modificación porque se viola la restricción de integridad referencial ya que el valor '222' no aparece en la relación *Profesor.*

Entonces, ¿cómo se puede añadir esa información si en cualquier de los dos órdenes posibles el sistema rechaza la inserción de la primera tupla?

Para resolver este problema se hace uso de las *transacciones*.

> Una transacción es una secuencia de operaciones de acceso a la base de datos (de manipulación y/o consulta) que constituye una unidad lógica de ejecución.

La idea es que en una transacción se encapsulan varias operaciones como si fueran una sola de manera que la comprobación de las restricciones de integridad se pospone al final de la transacción. En el ejemplo se ejecutaría lo siguiente (la sintaxis en SQL para las transacciones y operaciones de manipulación se estudiará más adelante):

```
INICIO TRANSACCIÓN
     INSERTAR en Profesor la tupla
       {(dni, '222'), (nombre, 'Armando Lacuesta Abad'),(cod_dep,'DSIC'),
        (teléfono, 8564), (provincia, ?),(edad,?)};
     INSERTAR en Docencia la tupla
       {(dni, '222'), (cod_asg, '11546'), (gteo, 1), (gpra, 1)}
FIN TRANSACCIÓN
```

Una vez finalizada la transacción, la base de datos queda como se muestra:

Departamento			
cod_dep	**nombre**	**teléfono**	**director**
DMA	Matemática Aplicada	1256	
DSIC	Sistemas Informáticos y Computación	1542	453

Asignatura					
cod_asg	**nombre**	**semestre**	**cod_dep**	**teoría**	**prácticas**
11545	Análisis Matemático	1A	DMA	4,5	1,5
11546	Álgebra	1B	DMA	4,5	1,5
11547	Matemática Discreta	1A	DMA	4,5	1,5

Profesor					
dni	**nombre**	**teléfono**	**cod_dep**	**provincia**	**edad**
111	Luisa Bos Pérez		DMA	Alicante	33
123	Juana Cerdá Pérez	3222	DMA	Valencia	50
564	Pedro Martí García	3412	DMA	Castellón	27
222	Armando Lacuesta Abab	8564	DSIC		

Docencia			
dni	**cod_asg**	**gteo**	**gpra**
111	11547	1	3
123	11545	0	2
123	11547	1	1
564	11545	2	2
222	11546	1	1

Cuando termina la ejecución de la transacción, el SGBD comprueba todas las restricciones de integridad incluidas en el esquema de la base de datos y, dado que se satisfacen, acepta las modificaciones realizadas por la transacción.