BASES DE DATOS Y SISTEMAS DE INFORMACIÓN

Tema 4.3

Miquel Gómez Corral Ing. Informática ETSINF, UPV

Tema 4.3

Diseño lógico

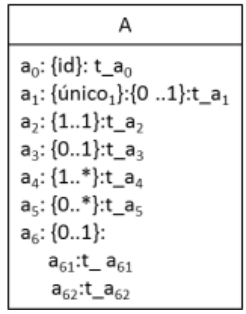
Consiste en **pasar el esquema UM** a una **estructura descrita en términos del modelo de datos** en el cual se base el SGBD. *Es la parte que sigue en el esquema ese de fondo verde.* Ahora habrá cosas que para ponerlas no las podremos expresar tal cual, para eso usamos las Restricciones de integridad.

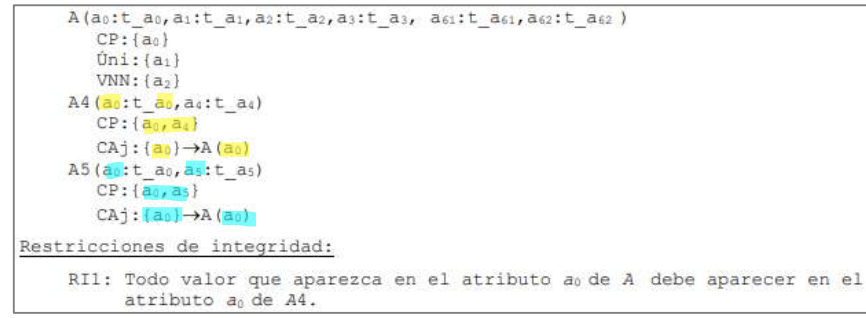
TRANSFORMACIÓN DE LAS CLASES

En un diagrama clases, se pueden **distinguir cuatro tipos de clases**: Clases fuertes (*ni débiles ni especializadas*), Clases débiles, Clases especializadas y Clase asociación.

Clases fuertes

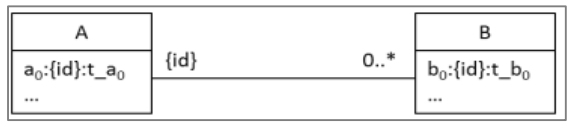
Clase ni débil ni especializada. Que tenga **una clase primaria que NO depende de nadie más** que de si misma. Se representa SOLO con una clase + sus atributos (*los comentos después*).

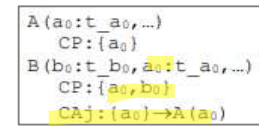




Clases débiles

Clases que **dependen para identificarse de otra clase mediante su/s clave/s primaria/s**. *Pueden o no tener una clave primaria propia*. Se pone como atributos los suyos + las claves primarias de la clase que dependa + clave ajena.





Cardinalidad

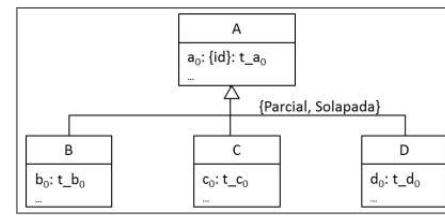
- **Id – 0..* y Id – 0..1**: Se expresan exactamente igual añadiendo las CP y la CAJ
- **Múltiples dependencias**: Se expresa igual pero añadiendo TODAS las CP y Separando en diferentes CAJ.

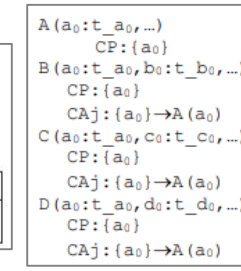
Clases especializadas: *Herencia*

Sólo cuando la especialización es **Parcial** y **Solapada** existe una transformación en relaciones sin cosos raros, en los demás casos es necesario incluir restricciones de integridad. No hace falta copiar TODO los atributos, solo las CP.

Si es TOTAL se pone RI: Todo valor que aparezca en el atributo a0 de A debe aparecer en el atributo a0 de B, de C o de D. *Al menos en uno.*

Si es DISJUNTA se pone RI: No puede haber un mismo valor en el atributo a0 de B y en el a0 de C; ni en el a0 de B y en el a0 de D; ni en C y en D…





ATRIBUTOS

En los ejercicios no hace falta que pongamos los tipos de cada coso. Solo el nombre y que se diferencia la clase.

Claves Primarias (*CP*): Se ponen **las de la clase** + (*si tiene*) las CAJ de las que dependa si es débil o SOLO las de la **superclase** en caso de ser una **heredada** (*las subclases NO tienen CP propia, es la de la clase de la que hereden*).

Claves ajenas (*CAJ*): Se ponen **cuando pillen la CP de otra clase** o **cuando lo pida la relación** xq es máxima cardinalidad de 1: 0..1 y 1..1. *La cardinalidad esta es la que escribes en la otra clase, que esta solo tenga 1 de la otra.*

- Si hay diferentes CAJ de diferentes clases se ponen en diferentes CAJ.

Único (*UNI*): Cuando el **valor ha de ser único**. Si hay diferentes tipos de único se ponen en diferentes líneas. Se pone también en las relaciones 1 a 1.

Valor No Nulo (*VNN*): Cuando el valor tiene como cardinalidad mínima 1.

Cardinalidades de los atributos: Si es 0..1 o 1..1 se pone el atributo tal cual (si hace falta VNN), pero si son 0..* o 1..* se representa creando OTRA clase a parte que tenga como CP las de la clase original + el atributo.

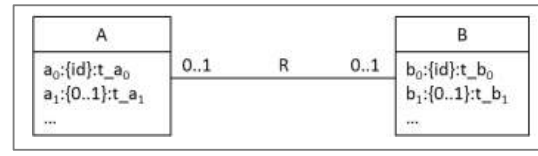
- **Si la cardinalidad es de 1..*** se ha de **poner como Restricción de integridad**: *Todo valor {a0} que aparece en una fila de A aparece en una fila de A-A1*. Siendo a0 la clave primaria de A y A1 el atributo con 1..*.

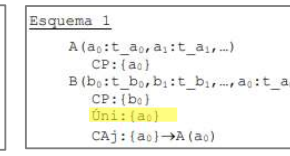
Atributos compuestos: Si el atributo es a1 y los sub atributos son a11 y a12 se pone como 2 atributos: a1.a11 y a1.a12. *Ejemplo de este y de cardinalidad de atributos en el primer ejemplo*.

TRANSFORMACIÓN DE LAS ASOCIACIONES/RELACIONES

Asociación con max 1 (1 a 1): *0..1-0..1 / 1..1-0..1 / 1..1-1..1*

Se hacen todas muy parecidas. En "**0.1-0.1 y 1.1-0.1**" **Se pilla a la clase que sea más restrictiva** (*en 1.1-0.1 la que al menos tiene 1 elementos del otro, en las otras te da igual*) y le pones la CP del otro, lo declaras como **CAJ y UNI** y pones VNN si hace falta (*solo en la 1.1*).



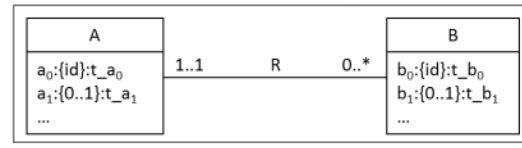


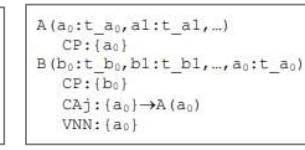
En las **1.1-1.1** o **unificas las 2 tablas en una sola**, o **Haces lo de antes** y a la tabla que no modificas le haces que la CP sea una CAJ a la otra clase: En el ejemplo sería añadir en A: `CAJ:{a0}->B(a0)`.

Asociación con max * (1 a *): *0..1-0..* / 1..1-0..* / 1..1-1..* / 0..1-1..**

Es casi igual que lo de antes pero **NUNCA pones "UNI"**, en los que haya tengas **1.1** poner VNN y en los que **1.*** hay que poner restricción de integridad:

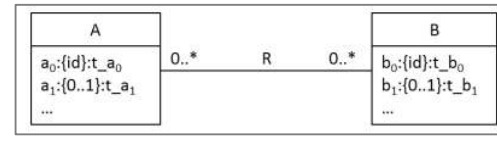
Restricción de Integridad: *Todo valor que aparezca en el atributo a0 de A debe aparecer en el atributo a0 de B.*

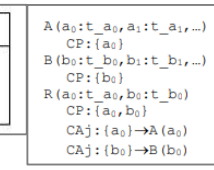




Asociación muchos a muchos (* a *): 0..*-0..* / 1..*-0..* / 1..*-1..*

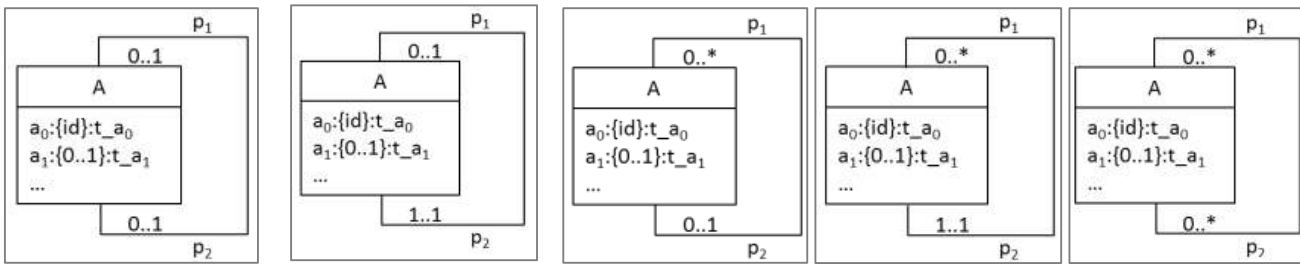
Se **dejas las clases iguales** y se hace una tabla a parte con **CAJ y CP** las CPs de las dos Clases. Si hay un **1..*** poner **RI**: *Todo valor que aparece en el atributo b0 de B aparece en el atributo b0 de R.*

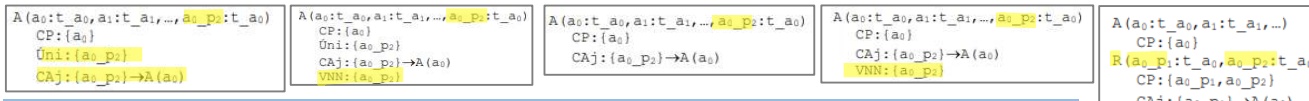




Asociaciones reflexivas: *Una clase consigo misma*

Se hacen **exactamente igual que las normales**. Las que tienen máxima cardinalidad 1 se pone UNI y si alguno tiene de mínima 1 VNN. En las **1 a muchos** igual pero NO pones UNI. Y en las **Muchos a muchos** haces una tabla relación a parte, poniendo RI si hace falta: *Todo valor que aparece en el atributo a0 de A aparece en el atributo a0 de R…*

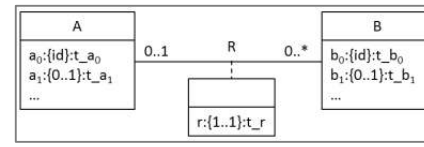


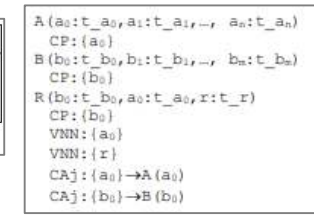


Atributos de enlace

Cuando una asociación entre clases tiene atributos de enlace, éstos **deben incluirse siempre en la relación donde se represente la asociación**. *Si le pones la CAJ a B pues le añades el atributo a B, si haces una tabla relación pues ahí…*

Restricciones de integridad





Como tenemos que **hacer siempre el modelo más sencillo posible**, si hay una opción de NO poner restricción de integridad hay que usarla, que **sería crear una tabla a parte para la relación**. En este caso, SOLO hay una excepción de NO hacer esto y SÍ representar la relación las tablas ya hechas que sería está primera.

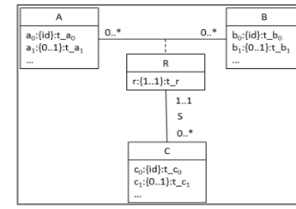
Si B tiene algún A (min 1): **Pones** `VNN{a0}` y si r tiene min 1 también `VNN{r}`, si r tiene min 0 **NO**. *A ESTE CASO NO HAY QUE PONERLE RESTRICCIÓN NI HACER TABLA EXTRA.*

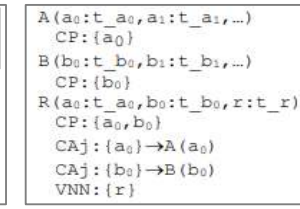
Si B NO tiene el xq tener algún A (min 0) y r min 1: Hacer de R otra tabla, poniendo como **CAJ a0 y b0** y **VNN{r}**. Después, para expresar la cardinalidad de A y B, poner **CP SOLO de b0**, y VNN{a0}.

Si B NO tiene el xq tener algún A (min 0) y r min 0: Hacer de R otra tabla, poniendo como **CAJ a0 y b0**. Después, para expresar la cardinalidad de A y B, poner **CP SOLO de b0**, y VNN{a0}.

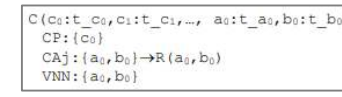
Transformación cuando se asocia con clases asociación

Estas son extrapolaciones de las otras. Hay muchos casos so apáñate para hacer cada uno jaja. ES **MÁS FACIL** SI **SIEMPRE HACES UNA RELACIÓN A PARTE** Y REPRESENTAS EN ESA LAS CARDINALIDADES (*Te ahorras poner algunas IRs*), Así la relación con la C se hace como relación normal..





Muchos en C: Se pone en C la CAJ de R o de la clase en la que se represente la relación. Si hace falta poner VNN.



Muchos a Muchos: Se hace una tabla a parte entre R y C. *En este caso, si es mínima a uno poner RI:*

Restricciones de integridad (*Cuando creas una tabla para la relación* R)

Cuando en S es muchos a muchos y cardinalidad mínima de C es 1: No puede existir una tupla en R tal que su clave principal, (a0,b0), no aparezca en los atributos (a0,b0) de S.

TEORÍA DE LA NORMALIZACIÓN

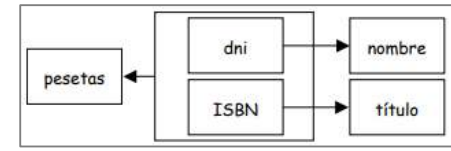
Conceptos

Dependencia funcional: Se dice que hay una **dependencia funcional entre atributos** (*o que un atributo depende funcionalmente de otro*) **si sabiendo el primero puedes deducir el otro**:

Por ejemplo, si la **clave primaria** de una relación muchos a muchos (*de escritores que escriben libro*) es el **DNI** y un **ISBN** (de un libro): **Dado un DNI** el nombre de la persona siempre será el mismo. Igualmente, con el **ISBN** y título.

{DNI}->{Nombre}, {ISBN}->{Titulo}… Has otras como {DNI,ISBND}->{Nombre}, o {DNI,Titulo}->{Nombre}… Cosas que nos dejan saber SOLO con unos pocos atributos otros atributos: *Si a la parte derecha le añado atributos, la parte izquierda siempre se sigue cumpliendo*.

- **Dependencia funcional completa**: Esto sucede cuando **NECESITO todos los atributos de la relación para poder identificar el resto** (*no se pueden deducir*). También si a {DNI,ISBN}-> {Pesetas} le quito ISBN, deja de existir la dependencia, por lo que DNI ISBN ES completa. Por otra parte, {DNI,Título}->{nom_alu} NO lo es xq puedo quitarle algo (titulo) y seguir identificando el nombre.
  Por ejemplo, si la relación muchos a muchos solo tienen el DNI, el ISBN y por ejemplo, lo que ha cobrado el escritor, si quito alguno de ellos, no puedo deducir los otros. Igual con DNI y nombre, y ISBN y titulo.
- **Diagrama de dependencias Funcionales**: **Representación gráfica** de las dependencias funcionales. En el ejemplo se indica que, Con el DNI sabes el nombre, con el ISBN sabes el titulo, y con el DNI y el ISBN (a la vez) sabes lo que cobra.



Clave de una relación: Las claves primarias de una relación y también si el atributo cumple unicidad. Esto lo que haces que dados esos atributos, puedas identificar el resto (sabes a que tupla pertenece).

Atributo primo: Un atributo es primo de R si forma parte de cualquier clase de una relación R (si es clave, o pertenece a un conjunto de claves (si la CP es doble…) es primo).

Primera forma normal (*1FN*)

Una relación R está en 1FN **si sus atributos** *sólo pueden tomar valores atómicos* (simples, indivisibles). *Si se usaran datos complejos habría que usar operaciones sobre listas, conjuntos etc. Eso kk.* Serían cuando tenemos **objetos con cardinalidad ***, lo que hacemos de crear **una tabla aparte**. **Objeto compuesto**, lo **separamos en subAtributos**. Pero si nos dan un modelo que está mal, hay que pasarlo:

Pasar a 1FN: Si por ejemplo me dan esta cosa (*está mal*) tendríamos que **pasarla a 1r forma normal** creando otra relación en la que se enlacen los teléfonos con los vcod y separemos la dirección en otros atributos:

vcod	nombre	teléfonos	Dir
V1	Pepe	(96 3233258, 964 523844 979 568987, 987 456123)	Paz 7, Valencia
V2	Juan	(96 3852741, 910147258)	Eolo 3, Castellón
V3	Eva	(987 456 312)	F. Lorca 2, Utiel

→

vcod	nombre	calle	número	ciudad
V1	Pepe	Paz	7	Valencia
V2	Juan	Eolo	3	Castellón
V3	Eva	F. Lorca	2	Utiel

+

vcod	teléfonos
V1	96 3233258
V1	964 523844
V1	979 568987
V1	987 456123
V2	96 3852741
V2	910 147258
V3	987 456 312

Segunda forma normal (*2FN*)

Pa hacerlo easy SOLO hay 1 clave (*solo CP sin Nadie con unicidad*): Una **relación R está en 2FN** si **está en 1FN y todo atributo no-primo depende funcionalmente de forma completa de toda clave de R**. *Atributos NO clave primaria son identificables con esta.* Si no está en 2FN tendrá redundancia y además NO es fácil de borrar e insertar filas.

Parar a 2FN: Hacer que todos los atributos solo dependen de la CP. Mejor en el ejemplo:

dni	nom_alu	cod	nom_asig	nota
1	Pepe	DBD	Diseño de Bases de Datos	6
1	Pepe	BDA	Bases de Datos	7
2	Juana	DBD	Diseño de Bases de Datos	7
2	Juana	BDA	Bases de Datos	5

→

cod	nom_asg
DBD	Diseño de Bases de Datos
BDA	Bases de Datos

dni	nom_alu
1	Pepe
2	Juana

+

dni	cod	nota
1	DBD	6
2	BDA	7
1	DBD	7
2	BDA	5

En la tabla original, pudes tener {DNI,COD}->{nota} y también {DNI,COD,nom_alu}->{nota} y más… lo que significa que la **NO todas las dependencias son completas** (la de DNI COD a nota SÍ LO ES xq *si le quitas algo a la parte derecha se perdería la relación*, **pero las otras NO** lo son xq yo le puedo quitar "nom_alu" y seguir identificando nom_alu) hay que evitar eso: Separar en relaciones según lo que depende cada atributo:

1. Nom_alu depende de dni, pues tabla pa ellos
2. Nom_asig depende de cod, pues tabla pa ellos
3. Nota depende de dni y cod, pues tabla pa ellos…

Alumno(dni, nom_alu)
CP: {dni}
Asignatura(cod, nom_asig)
CP: {cod}
Matriculado(dni, cod)
CP: {dni,cod}
CAj: {dni} → Alumno
CAj: {cod} → Asignatura

Tercera forma normal (*3FN*)

Una **relación R está en 3FN** si **está en 2FN** y **no hay dependencias funcionales entre atributos no primos**. Si no está en 3FN tendrá redundancia y además NO es fácil de borrar e insertar filas.

Esto sería que NO pase que, para un valor de algún atributo (en el ejemplo dentro), siempre haya otro atributo con el mismo valor (si está en un centro, siempre se llamará igual y su director siempre será el mismo, HAY DEPENDENCIA entre atributos NO primos, el único primo es dni que es la CP).

CP: {dni}

dni	nom_alu	centro	nom_centro	director
1	Olga	EUI	Escuela Universitaria de Informática	Pepe
2	Juana	EUI	Escuela Universitaria de Informática	Pepe
3	Ana	FI	Facultad de Informática	Eva
4	Juan	FI	Facultad de Informática	Eva

dni	nom_alu	centro
1	Olga	EUI
2	Juana	EUI
3	Ana	FI
4	Juan	FI

+

centro	nom_centro	director
EUI	Escuela Universitaria de Informática	Pepe
FI	Facultad de Informática	Eva

Pasar a 3FN: Se **soluciona haciendo una tabla a parte para los atributos que son dependientes** y se **elige una de ellos como CP** para que los identifique (en este caso es Centro).

Resumen

Comprobar que **R está en 1FN**. **Si no lo está**, transformar R a un conjunto **C1** de relaciones que estén en 1FN.

Comprobar si las relaciones de **C1 están en 2FN**. Si alguna relación **no está en 2FN** debe descomponerse en un conjunto de relaciones, sea **C2**, que estén en 2FN.

Comprobar si las relaciones de **C2 están en 3FN**. **Si alguna relación no está en 3FN**, debe descomponerse en un conjunto que sí lo esté.