# Material para
# a formación profesional inicial

# A01. O deseño lóxico empregando o modelo relacional

| | | |
|---|---|---|
| Familia profesional | IFC | Informática e comunicacións |
| Ciclo formativo | CSIFC03<br>CSIFC02 | Desenvolvemento de aplicacións web<br>Desenvolvemento de aplicacións multiplataforma |
| Grao | | Superior |
| Módulo profesional | MP0484 | Bases de datos |
| Unidade didáctica | UD03 | Deseño lóxico de base de datos |
| **Actividade** | **A01** | **O deseño lóxico empregando o modelo relacional** |
| Autores | | Marta Fernández García<br>María del Carmen Fernández Lameiro<br>Miguel Fraga Vila<br>María Carmen Pato González<br>Andrés del Río Rodríguez |
| Nome do arquivo | | CSIFC02_ MP0484_V000301_UD03_A01_crearMR.docx |

© 2015 Xunta de Galicia.
Consellería de Cultura, Educación e Ordenación Universitaria.

Este traballo foi realizado durante unha licenza de formación retribuída pola Consellería de Cultura, Educación e Ordenación Universitaria e ten licenza CreativeCommons BY-NC-SA (recoñecemento - non comercial - compartir igual). Para ver unha copia desta licenza, visitar a ligazón http://creativecommons.org/licenses/by-nc-sa/3.0/es/.

# Índice

# 1. A01. O deseño lóxico empregando o modelo relacional

## 1.1 Introdución

Os obxectivos desta actividade son:

- Coñecer os elementos e terminoloxía do modelo relacional de Codd.
- Coñecer a forma de modelar a realidade empregando o modelo relacional.
- Coñecer os mecanismos do modelo relacional para expresar restricións de integridade
- Identificar o modo en que as entidades, as súas interrelacións, os atributos de ambas, e as restricións recollidas no modelo conceptual entidade-interrelación de Chen son trasladadas ao modelo lóxico de datos.
- Desenvolver a capacidade de abstracción necesaria para resolver problemas de tratamento de información.

Para facer as prácticas asociadas non se precisa do uso dunha ferramenta informática aínda que para a súa dixitalización recoméndase o emprego da ferramenta de deseño gráfico Microsoft Visio 2010 para a que se implementa unha librería personalizada.

A elección desta aplicación débese a que as ferramentas case estudadas RWin, De-signer2000,… teñen unha simboloxía propia cunha semántica máis reducida. Considérase que o importante é ter os coñecementos teóricos de cómo deseñar unha base de datos para posteriormente adaptalos a unha ferramenta CASE específica ou empregar a utilidade de deseño que aporte o propio SXBD.

## 1.2 Actividade

Nesta actividade abordarase o estudo da fase de análise coñecida coma deseño lóxico de datos, aplicada ao modelo relacional para a obtención de bases de datos relacionais. Seguirase fundamentalmente a metodoloxía proposta por Adoración de Miguel e Mario Piattini.

Analizaranse e detallaranse os elementos do modelo para posteriormente traducir o esquema conceptual ao esquema lóxico expresado dun xeito comprensible para un SXBD.

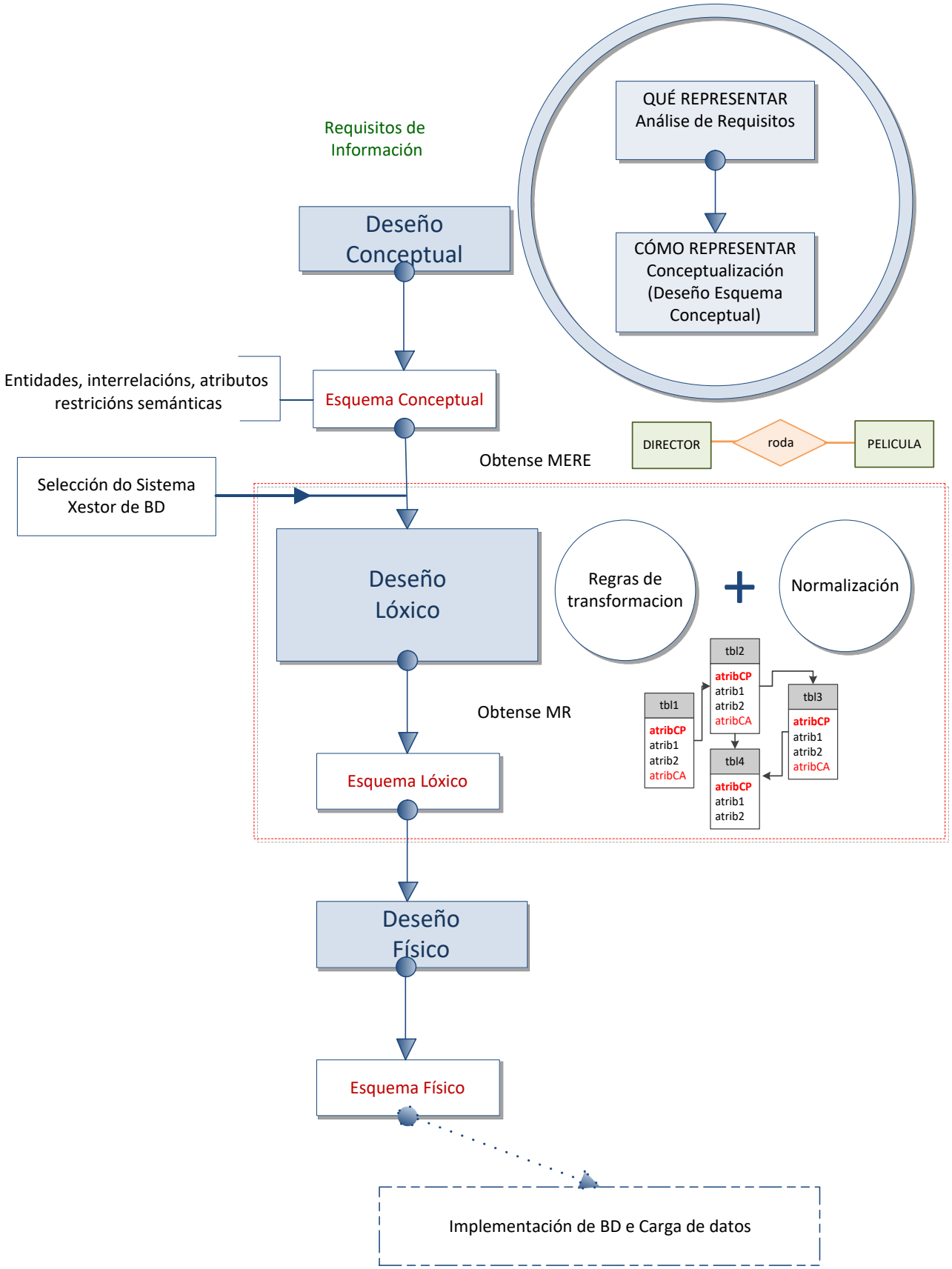



Figura 1.1. Fases do deseño dunha base de datos

### 1.2.1 Historia do modelo relacional

Tras ser postuladas as súas bases en 1970 por Edgar Frank Codd, nos laboratorios IBM en San José (California), non tardou en consolidarse como un novo paradigma nos modelos de base de datos substituíndo aos clásicos sistemas de ficheiros que mostraban grandes deficiencias.

A súa idea fundamental é o uso de relacións, constituíndo estas o elemento fundamental deste modelo de aí o seu nome, "modelo relacional". Estas relacións poden considerarse como conxuntos de datos chamados tuplas. Pese a que esta é a teoría das bases de datos relacionais creadas por Codd, a maioría das veces se conceptualiza dunha maneira máis fácil de imaxinar, pensando en cada relación como se fose unha táboa que está composta por rexistros (cada fila da táboa será un rexistro ou "tupla") e columnas (tamén chamadas "campos").

O modelo relacional, para o modelado e xestión de bases de datos, é un modelo baseado na lóxica de predicados de primeiro orde e na teoría de conxuntos, polo que presenta unha sólida base formal.

Este modelo de datos publicado por Codd perseguía os seguintes obxectivos:

- **Independencia física:** O modo de como se almacenan os datos non debe influír na súa manipulación lóxica e, por tanto, os usuarios que acceden a eses datos non han de modificar os seus programas por cambios no almacenamento físico.
- **Independencia lóxica:** Engadir, eliminar ou modificar calquera elemento da BD non debe repercutir nos programas e/ou usuarios que están a acceder a subconxuntos parciais dos mesmos (vistas).
- **Flexibilidade:** Ofrecer a cada usuario os datos da forma máis adecuada á correspondente aplicación.
- **Uniformidade:** As estruturas lóxicas dos datos presentan un aspecto uniforme (táboas), o que facilita a concepción e manipulación da BD por parte dos usuarios.
- **Sinxeleza:** As características anteriores, así como unhas linguaxes de usuario moi sinxelas, producen como resultado que o modelo relacional (MR) sexa fácil de comprender e de utilizar por parte do usuario final.

Codd concedeu moita importancia ao tema da independencia da representación lóxica dos datos respecto do seu almacenamento interno, que concretou en tres tipos de independencia: de ordenación, de indexación, e dos camiños de acceso.

Importancia que Codd manifesta explicitamente:

"... proponse un modelo relacional de datos como unha base para protexer aos usuarios de sistemas de datos formateados dos cambios que potencialmente poden alterar a representación dos datos, causados polo crecemento do banco de datos e polos cambios nos camiños de acceso".

Os avances máis importantes que o modelo de datos relacional incorpora respecto dos modelos de datos anteriores son:

- **Sinxeleza e uniformidade:** Os usuarios ven a base de datos relacional como unha colección de táboas, e ao ser a táboa a estrutura fundamental do modelo, este goza dunha gran uniformidade, o que unido a unhas linguaxes non navegacionais e moi orientadas ao usuario final, dá como resultado a sinxeleza dos sistemas relacionais.
- **Sólida fundamentación teórica**: Ao estar o modelo definido con rigor matemático, o deseño e a avaliación do mesmo pode realizarse por métodos sistemáticos baseados en abstraccións.

- **Independencia da interfaz de usuario**: as linguaxes relacionais, ao manipular conxuntos de rexistros, proporcionan unha gran independencia respecto da forma na que os datos están almacenados.

As vantaxes citadas contribuíron a que desde mediados dos anos 80, o MR sexa utilizado por practicamente a totalidade dos SXBD comerciais, sendo o modelo de datos relacional o máis utilizado para modelar sistemas reais de aplicacións comerciais de procesamento de datos convencionais. Impúxose debido ás limitacións dos modelos anteriores como o de Codalsy (modelo en rede) ou o xerárquico. Baséase no uso de relacións ou táboas que agrupan conxuntos de datos en forma de filas ou tuplas. Ademais, incorpora restricións que son formas de reflectir as regras de funcionamento da empresa ou organización que se está a modelar.

Algunhas das principais empresas informáticas do mundo son en orixe empresas de SXBD (ORACLE, Sybase, INFORMIX, …), os grandes fabricantes de software teñen "o seu" SXBD relacional (IBM DB2, Microsoft SQL Server, ...), ademais existen bastantes SXBD deseñados para PC's e usuarios non expertos (Microsoft Access...).

## 1.2.2 Elementos e terminoloxía

Para construír e interpretar os modelos relacionais necesitamos en primeiro lugar coñecer os seus elementos, que permitiranos representar os datos do sistema a modelar.

### 1.2.2.1 Relación

A relación é o elemento básico do modelo relacional e constitúe unha estrutura de datos cun nome e un conxunto de atributos ou características.

Por exemplo, unha película dunha BD de venda de visionado de programas quedaría representado do seguinte modo:

**PROGRAMA** (*codPrograma* , *idiomasVisionado, idiomasSubtitulos,* tituloDistribucion, tituloOrixinal, dataEstreo, idomaOrixinal, sinopse, duracion, clasificacion, webOficial*, paisRodaxe, xenero, 3D, tipoPrograma)

Figura 1.2. Exemplo de representación dunha relación

Representamos a relación PROGRAMA cos seus campos ou características. Na BD almacenarase os atributos de PROGRAMA xunto co tipo de dato de cada un; especificando o tamaño e o dominio dos mesmos, se son obrigatorios ou opcionais, etc.

#### Atributo

Os atributos permítenos definir as propiedades ou características da relación.

Por exemplo, a relación PROGRAMA disporá entre outros dos atributos "tituloDistribución" que representa o título có cal o programa estreouse no noso país ou "webOficial" que conterá a URL do programa.

#### Dominio

O dominio é o conxunto de valores permitidos para certos atributos. Por exemplo, o atributo "xénero" na relación PROGRAMA só poderá tomar os valores do dominio (drama, comedia, biografía, misterio, terror, romance, guerra, animación e aventuras).

Podemos definir un dominio coma un conxunto nomeado (identifícase ou caracterízase por un nome), cun número finito de elementos atómicos ou indivisibles e homoxéneo (do mesmo tipo).

Cada dominio pode definirse de dúas maneiras:

- Extensión (dando os seus posibles valores).

Por exemplo, clasificación de programa = { todos os públicos, maiores de 9 anos, maiores de 15 anos, maiores de 18 anos}

- Intensión (mediante un tipo de datos).

Por exemplo, duración dun programa = enteiro, ou duración dun programa = entre 15 e 300 minutos e título = ata 50 carácteres alfanuméricos.

Ás veces asóciase unha unidade de medida (quilos, metros, etc.) e/ou certas restricións (como un rango de valores).

Existe un valor especial non asociado a un dominio concreto que pode permitirse ou non nun dato para reflectir situacións do mundo real onde existe: información perdida, ausencia de información ou valores non aplicables a ese atributo. Para representar este valor o sistemas de bases de datos o identifican como NULO (NULL). Se unha tupla ten un valor nulo significa que o valor real de ese atributo é descoñecido.

NULO non é un valor en se mesmo, senón un indicador ou marca de ausencia de información. Non hai dous nulos iguais polo que non se pode comparar dous atributos coa marca NULO.

### Dominio e atributo

Un atributo e un dominio poden chamarse igual, pero os atributos diferéncianse dos dominios:

- Un atributo está sempre asociado a unha relación, mentres que un dominio ten existencia propia con independencia das relacións.
- Un atributo representa unha propiedade dunha relación.
- Un atributo toma valores definidos nun dominio, estando cada atributo definido sobre un único dominio obrigatoriamente.
- Un dominio pode conter valores non tomados por ningún atributo
- Varios atributos distintos (da mesma ou de diferentes relacións) poden tomar os seus valores do mesmo dominio.

En ocasións, pode acontecer que o valor dun atributo para unha fila sexa descoñecido, neses casos, asóciase a esa ocorrencia un valor NULO para ese atributo. Non debe confundirse un valor nulo cunha cadea de carácteres baleira ou cun valor numérico igual a cero.

A comparación de dous atributos só terá sentido se ambos toman valores do mesmo dominio. Se o SXBD soporta Dominios, poderá detectar este tipo de erros.

Ademais dos dominios e atributos simples, que acabamos de definir, en ampliacións posteriores do MR introduciuse o concepto de dominio composto, que é moi útil na práctica.

Un dominio composto pódese definir como unha combinación de dominios simples aos que se pode aplicar certas restricións de integridade. Por exemplo, o dominio composto denominado *Data* constrúese por agregación dos dominios simples *Día*, *Mes* e *Ano*, incorporando as adecuadas restricións de integridade a fin de que non aparezan valores non válidos para a data.

Do mesmo xeito que é posible definir dominios compostos, existen tamén atributos compostos. Tanto os atributos compostos como os dominios compostos poden ser tratados, se

así o precisa o usuario, como "elementos únicos" de información, é dicir, como valores atómicos.

Facendo unha analoxía entre dominios e tipos de datos nas linguaxes de programación; o dominio sería o tipo de datos da linguaxe de programación e cada atributo a variable.

#### Tupla

É o conxunto válido de valores que toma cada un dos atributos, son as chamadas ocorrencias ou exemplares da relación. Por exemplo unha tupla da relación PROGRAMA podería ser:

**PROGRAMA** (*P00045624*, {español, galego, inglés, frances}, {español, galego, inglés, frances}, The godfather, El padrino, 1972, ingles, 'Michael Corleone é un home apartado dos negocios mafiosos da súa familia ata que a vida do seu pai Don Vito Corleone corre perigo', 175, maiores de 18 anos, NULO, Estados Unidos, drama, NON, pelicula)

Figura 1.3. Exemplo de representación dunha tupla por extensión

#### Cardinalidade

O número de tuplas ou ocorrencias dunha relación denomínase cardinalidade. Así, por exemplo, na relación DIRECTOR da base de datos de venda de visionado teremos unha cardinalidade igual ao número de directores rexistrados; na figura 1.5 móstrase este exemplo graficamente.

#### Grao

O número de atributos dunha relación constituirá o grao. Así, na relación DIRECTOR do exemplo, o grao e igual a 5; na figura 1.5 móstrase este exemplo graficamente.

### 1.2.2.2 Esquema de relación versus táboa

Unha vez definidos os elementos básicos e terminoloxía dunha relación "R" podemos definir esta formamalmente coma un conxunto de atributos "A" definidos nun conxunto de dominios "D".

Unha relación componse de dúas partes:

- **Cabeceira ou esquema dunha relación:** Conxunto fixo de pares {Atributo: Dominio} xunto coas súas regras de integridade (definidas nos seguintes apartados).
- **Corpo dunha relación:** Conxunto variable de tuplas (pares {atributo:valor}).

Deste xeito, pódese distinguir entre esquema de **relación ou intensión,** que define a estrutura da táboa, é dicir, os seus atributos cos dominios subxacentes, e un **corpo ou extensión**, que estará formado por un conxunto de ocorrencias ou tuplas que varían no tempo.

Na relación DIRECTOR, a súa intensión quedaría definida como:

– DIRECTOR (idDirector: cadea de 9 caracteres , nome: cadea de 25 carácteres, apelido1: cadea de 50 carácteres, apelido2: cadea de 25 carácteres, dataNacemento: formato día-mes-ano, numeroProgramasDirixe: enteiro, codNacionalide: cadea de 3 carácteres).

A extensión para a mesma relación:

**DIRECTOR** ( *987654987*, Antón, Reixa, Rodríguez, 17-04-1957,3 , 00076

*236785439*, José Luís, Cuerda, Martínez, 18-02-1947, 8, 00076

*879654190*, Isabel, Coixet, Castillo, 09-04-1960, 3, 00076

*36896547*8, Vincenzo, Natali, NULO, 06-01-1969, 1, 00034

*456952134*, Steve Allan, Spielberg, NULO, 18-12-1946, 00023

*567423908*, Hans Christian, Tomas, Alfredson, 01-04-1965,00012

)

Figura 1.4. Exemplo do contido da táboa que representa a relación DIRECTOR

O feito de que a relación se represente en forma de táboa é a causa de que os produtos relacionais e os usuarios utilicen usualmente o nome de táboa para referirse ás relacións e, como consecuencia, se lles chame filas ás tuplas e columnas aos atributos ou campos.


Figura 1.5. Exemplo de relación con representación en forma de táboa

A diferenza entre a representación en forma de táboa da información non implica que relación e táboa sexan o mesmo xa que ambos termos representan conceptos distintos, presentando as seguintes diferenzas:

- Unha táboa é unha forma de representar unha relación (unha estrutura de datos).
- Unha táboa non ten as restricións inherentes dunha relación -como conxunto-
- Nunha táboa pode haber dúas filas iguais.
- As filas están ordenadas na orde de gravación física por defecto ou segundo o valor da clave primaria.
- Os atributos teñen unha orde segundo se definiron na táboa.
- En cada cela dunha táboa pode haber un ou varios valores. Aínda que no segundo caso pódese obter unha táboa equivalente que cumpre a regra de normalización.

Para concluír, unha relación é unha representación abstracta dun elemento ou entidade de datos, e unha táboa é unha representación concreta do elemento abstracto básico (a relación). No MR, unha relación ten unhas propiedades intrínsecas que non ten unha táboa, e que se derivan da definición matemática de conxunto.

### 1.2.3 Restricións

Cando facemos un modelo de datos dun sistema, ademais dos datos e as súas relacións establecemos as restricións, que permiten reflectir certas condicións ou regras de negocio que desexamos que cumpran.

Por exemplo, na BD venda de visionado de programas, unha restrición pode ser que cada película teña un código asociado para distinguilas das series. Estas restricións impoñen condicións ao modelo, e poden ser polas características do mesmo ou polos requirimentos do deseñador.

#### 1.2.3.1 Restricións inherentes ou implícitas

As restricións inherentes derívanse da mesma estrutura do modelo, non tendo que ser definidas polo deseñador xa que as impón o modelo. Estas restricións actívanse no momento da **definición** do esquema cando se produce un intento de violación do mesmo, rexeitándose todo esquema que non as cumpra.

Ademais das restricións derivadas da definición de relación:

- Cada relación ten un nome distinto entre elas.
- Non hai dous atributos que se chamen igual.

O MR define as seguintes:

- Non existen tuplas repetidas (do mesmo xeito que no concepto matemático de conxunto). É dicir, non pode haber dúas filas ou ocorrencias dunha relación co mesmo valor en todos os campos. Esta condición ven imposta pola obrigatoriedade da clave primaria.
- A orde das tupas é irrelevante, non requiríndose unha orde determinada nas filas que forman a táboa.
- A orde dos atributos (columnas) non é significativa.
- Cada atributo só pode tomar un único valor do dominio subxacente, non admitíndose polo tanto os grupos repetitivos (nin outro tipo de estruturas). É dicir, non podemos, por exemplo, ter dous valores para o campo *títuloOrixinal* na relación de película.

##### Restrición de integridade de entidade

Débese cumprir a regra de que "ningún atributo que forme parte da clave primaria dunha relación pode tomar un valor descoñecido ou inexistente (nulo)".

A regra aplícase sobre as relacións do esquema e só á clave primaria (non as claves alternativas).

#### 1.2.3.2 Restricións semánticas

As restricións de integridade semánticas ou de usuario son facilidades que o modelo ofrece aos deseñadores para que poidan reflectir no esquema, o máis fielmente posible, a semántica do mundo real (UD), son impostas polo deseño en función dos requisitos do sistema a modelar e polo tanto dependen do mesmo.

Os tipos de restricións semánticas permitidos no MR (incorporados no estándar SQL 92), permiten aumentar a súa capacidade expresiva, e son:

##### Restricións de integridade

Estas regras de integridade son recoñecidas polo MR e defínense no esquema da base de datos, sendo restricións de comportamento que se activan no momento da **actualización** da

BD, e rexeitan todo exemplar que non as cumpra ou poñen en marcha mecanismos a fin de que non se produza un estado inconsistente(evita configuracións de datos imposibles).

As restricións de integridade deben cumprirse pola base de datos en todos os seus estados, formando parte das cabeceiras das relacións.

### Restricións sobre atributos

Podemos distinguir entre:

- **Restricións de dominio:** nestas restricións ao definir cada atributo sobre un dominio, imponse unha restrición sobre o conxunto de valores permitidos para cada atributo
- **De obrigatoriedade (valor no nulo)**:Permiten declarar se un ou varios atributos deben tomar sempre un valor, e dicir, non poden tomar valores descoñecidos ou nulos.

### Restrición de unicidade (UNIQUE RESTRICTION)

Permite definir claves candidatas, prohibindo que os valores dun ou varios atributos se repitan en diferentes tuplas. Unha relación pode ter varias claves candidatas, entre as que seleccionarase unha clave de identificación primaria. As non seleccionadas denominaranse claves alternativas.

#### Clave Candidata (CC, CANDIDATE KEY –CK- ) e clave alternativa (CAl, ALTERNATIVE KEY –AK-)

Conxunto de atributos que identifican univocamente cada tupla nunha relación (non existen dúas tuplas distintas co mesmo valor para a clave) e minimamente (ningún subconxunto de atributos da clave ten a propiedade de unicidade).

A clave candidata pode ser:

- Simple: Formada por un só atributo.
- Composta: formada por máis dun atributo.

**CLIENTE** ( *dni*, **numeroTarxeta**, apelido1, apelido2, rua, localidade, provincia, codigopostal, correoElectronico*, telefono, dataNacemento, idade )

Figura 1.6. Exemplo de representación da relación CLIENTE

Para o caso da relación CLIENTE, as posibles claves candidatas serán o dni e o número de tarxeta. Porén, a unión do *dni* e o *nome* non sería unha clave candidata mínima xa que non se necesita o nome para identificar un cliente (e polo tanto redutible), de aí a definición de mínimo na clave.

### Restrición de Clave Primaria (PRIMARY KEY RESTRICTION)

Esta restrición permite declarar un atributo ou un conxunto de atributos como clave primaria dunha relación.

#### Clave Primaria (CP, PRIMARY KEY – PK-)

A clave primaria é a que o deseñador escolle de entre as claves candidatas por motivos alleos ao modelo, **cumprindo sempre a regra de integridade de entidade**.

Os seus valores non se poderán repetir nin se admitirán os nulos (ou valores “ausentes”).

Aínda que o modelo teórico impón esta restrición, nin SQL92 nin os SXBD’s relacionais obrigan á declaración dunha clave primaria para cada táboa, pero permiten a definición da mesma.

No caso da relación CLIENTE, mostrada no exemplo anterior, poderíase escoller o atributo dni como clave primaria deixando o número de tarxeta como clave alternativa.

Debemos distinguir entre a restrición inherente de obrigatoriedade da clave primaria e a restrición semántica que lle permite ao usuario indicar que atributos forman parte da clave primaria.

### Restrición de Integridade Referencial (FOREIGN KEY RESTRICTION),

Esta restrición permítenos unir relacións (ou unha relación consigo mesma) a través dos valores dos atributos, de xeito que o contido dun atributo ou conxunto de atributos nunha relación debe coincidir cos valores da clave primaria da outra relación.

Supoñamos a seguinte representación dunha base de datos sobre un casting previo a un concurso de actores:

| ACTOR | | |
|---|---|---|
| **codActor** | **nome** | **idade** |
| 456 | Iria | 18 |
| 678 | Ximena | 21 |
| 123 | Breixo | 32 |

| MEMBRO XURADO | | |
|---|---|---|
| **codXurado** | **nome** | **especialidade** |
| 222 | Sabela | NULO |
| 333 | Roi | NULO |
| 444 | Xulia | NULO |

| PAPEL | | |
|---|---|---|
| **codPapel** | **descrición** | **duración** |
| 1 | rapaza | 20 |
| 2 | rapaz | 17 |
| 3 | malo | 7 |
| 4 | amiga | 3 |

| ACTUACIÓN | | |
|---|---|---|
| **codPapel** | **codActor** | **data** |
| 1 | 456 | 03/03/1015 |
| 4 | 456 | 03/03/2015 |
| 1 | 678 | 04/03/2015 |
| 2 | 123 | 04/03(2015 |

| PUNTUACIÓN | | | | |
|---|---|---|---|---|
| **codPuntuacion** | **nota** | **codPapel** | **codActor** | **codXurado** |
| 1 | 3 | 1 | 678 | 222 |
| 2 | 5 | 4 | 456 | 333 |
| 3 | 7 | NULO | 456 | 333 |
| 4 | 6 | 2 | 123 | 444 |
| 5 | 8 | 1 | 678 | 222 |
| 6 | 6 | 3 | 678 | 444 |
| 7 | 5 | NULO | NULO | 333 |

Figura 1.7. Táboas que representan o contido dunha base de datos

#### Clave Allea (foránea ou externa, CA, FOREING KEY –FK-)

Un descritor, atributo ou conxunto de atributos é clave foránea (FK) dunha relación "R2" (relación que referencia) se todo valor de dito descritor concorda cun valor da clave primaria doutra relación "R1" ( relación referenciada). O modelo impide que existan na base de datos valores de claves alleas sen correspondencia cunha clave principal.

A clave allea pode ser: simple (formada por un só atributo) ou composta (integrada por varios atributos).

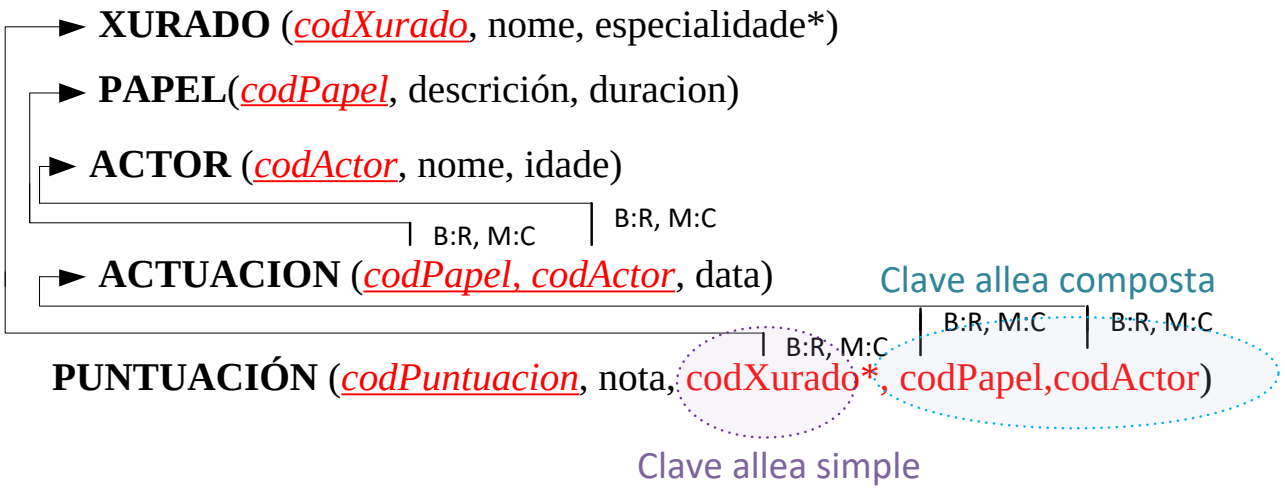



Figura 1.8. Representación de claves alleas

Os atributos que son clave allea nunha relación non precisan ter o mesmo nome que o atributo clave primaria coa que se corresponde.

Cada compoñente dunha clave allea (FK) debe estar definido sobre o mesmo dominio que o correspondente atributo da clave primaria (PK) referenciada.

O uso de claves alleas facilitará a eliminación das redundancias, e será o mecanismo do modelo relacional para establecer **vínculos entre relacións**, dando lugar á estrutura da base de datos.

NULOS e claves alleas:

O modelo relacional permite NULO como valor de clave allea, por exemplo, empregados non asignados a un departamento ou empregados sen xefe,. tendo en conta isto, podemos estender a definición de clave allea:

- Un descritor, atributo ou conxunto de atributos é clave foránea (FK) dunha relación "R2" (relación que referencia) se todo valor de dito descritor concorda cun valor da clave primaria doutra relación "R1" ( relación referenciada) **ou é NULO.**

**PUNTUACIÓN** (*codPuntuacion*, nota, codXurado *, codPapel, codActor)

| PUNTUACIÓN | | | | |
|---|---|---|---|---|
| **codPuntuacion** | **nota** | **codXurado** | **codPapel** | **codActor** |
| 1 | 3 | 222 | 1 | 678 |
| 2 | 5 | 333 | 4 | 456 |
| 3 | 7 | 333 | NULO | 456 |
| 4 | 6 | 444 | 2 | 123 |
| 5 | 8 | 222 | 1 | 678 |
| 6 | 6 | NULO | 3 | 678 |
| 7 | 5 | 333 | NULO | NULO |

Figura 1.9. A tupla é correcta xa que o atributo permite nulos e as claves alleas tamén

A regra de **Integridade Referencial** tendo en conta os NULOS establece que a base de datos non debe conter valores NON NULOS de clave allea sen correspondencia.

**PUNTUACIÓN** (*codPuntuacion*, nota, codXurado *, codPapel, codActor)

| PUNTUACIÓN | | | | |
|---|---|---|---|---|
| **codPuntuacion** | **nota** | **codXurado** | **codPapel** | **codActor** |
| 1 | 3 | 222 | 1 | 678 |
| 2 | 5 | 333 | 4 | 456 |
| 3 | 7 | 333 | NULO | 456 |
| 4 | 6 | 444 | 2 | 123 |
| 5 | 8 | 222 | 1 | 678 |
| 6 | 6 | NULO | 3 | 678 |
| 7 | 5 | 333 | NULO | NULO |

Figura 1.10. A tupla non é correcta xa que as claves allas compostas non poden tomar parcialmente p valor NULO

### Políticas de mantemento da integridade referencial

O modelo relacional permite definir as opcións de borrado e modificación nas claves alleas.

Hai que ter presente que a BD é dinámica e que, polo tanto, os valores vanse modificando e eliminando ao longo do tempo de modo que si creamos restricións de integridade referencial debemos determinar certas accións que deben tomarse como consecuencia destas operacións a realizar sobre as filas das táboas referenciadas. Para evitar estados ilegais o sistema poderá realizar:

- Rexeitar toda operación que produza un estado ilegal da base de datos. A este proceso denomínase **operacións restrinxidas (RESTRICT)**: só se pode borrar ou modificar tuplas na relación que ten clave primaria referenciada senón existen filas con esa clave allea na táboa que referencia.
- Aceptar e executar as operacións pero realizando accións que restauren a integridade dos datos, podendo afectar a dúas relacións ou máis se a súa vez estas relacións están unidas a outras, producíndose cambios en cadea (debe facerse fincapé que se rexeita unha operación o resto das operacións sobre as relacións afectadas suspéndense.):
  - **Operación con transmisión en cascada (CASCADE):** o borrado ou a modificación dunha tupla da relación que contén a clave primaria referenciada conleva o borrado ou a modificación en cascada das tuplas da relación que referencia onde a clave allea coincide co valor da clave primaria da táboa referenciada.
  - **Operación con posta a nulos (SET NULL)**: O borrado ou a modificación dunha tupla da relación que contén a clave primaria referenciada leva consigo a posta a nulos dos valores da clave allea das tuplas da relación que referencia onde a clave allea coincide co valor da clave primaria da táboa referenciada.
  - **Operación con posta a valor por defecto (SET DEFAULT)**: O borrado ou a modificación dunha tupla da relación que contén a clave primaria referenciada leva consigo a posta a un valor por defecto dos valores da clave allea das tuplas da relación que referencia onde a clave allea coincide co valor da clave primaria da táboa referenciada.

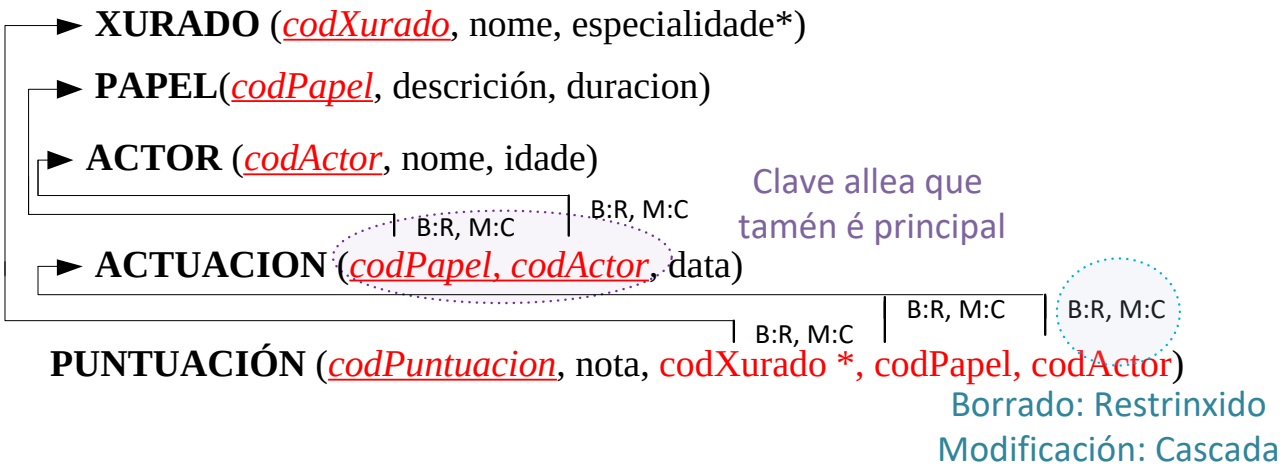



Figura 1.11. Representación das operacións para o mantemento da integridade referencial

A representación gráfica das opcións de borrado utilizada é:

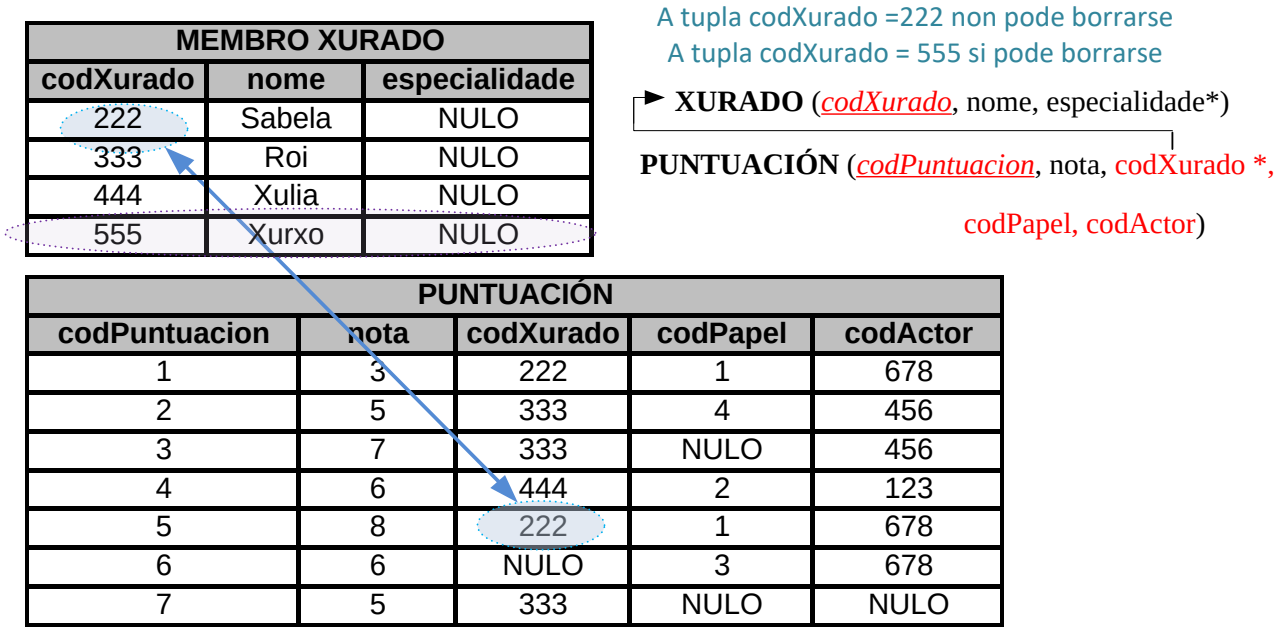
- B:R. Borrado restrinxido.
- B:C. Borrado en cascada.
- B:N. Borrado con posta a nulos.
- B:D. Borrado a un valor por defecto.

A representación gráfica das opcións de borrado utilizada é:

- M:R. Modificación restrinxida.
- M:C. Modificación en cascada.
- M:N. Modificación con posta a nulos.
- M:D. Modificación a un valor por defecto.

As operacións de borrado e modificación poden ser distintas para unha determinada clave allea dunha relación; por exemplo, é posible definir a modificación en cascada e o borrado restrinxido. Estas operacións dan como resultado as seguintes posibilidades:

- I. **Se borramos unha tupla "t"dunha relación "R1" que é referenciada por outra ou outras relacións "R2" pódense establecer os seguintes comportamentos**:
  - Comportamento por defecto, Rexeitar a operación: só permitimos borrar a tupla se ningunha outra tupla fai referencia a ela.

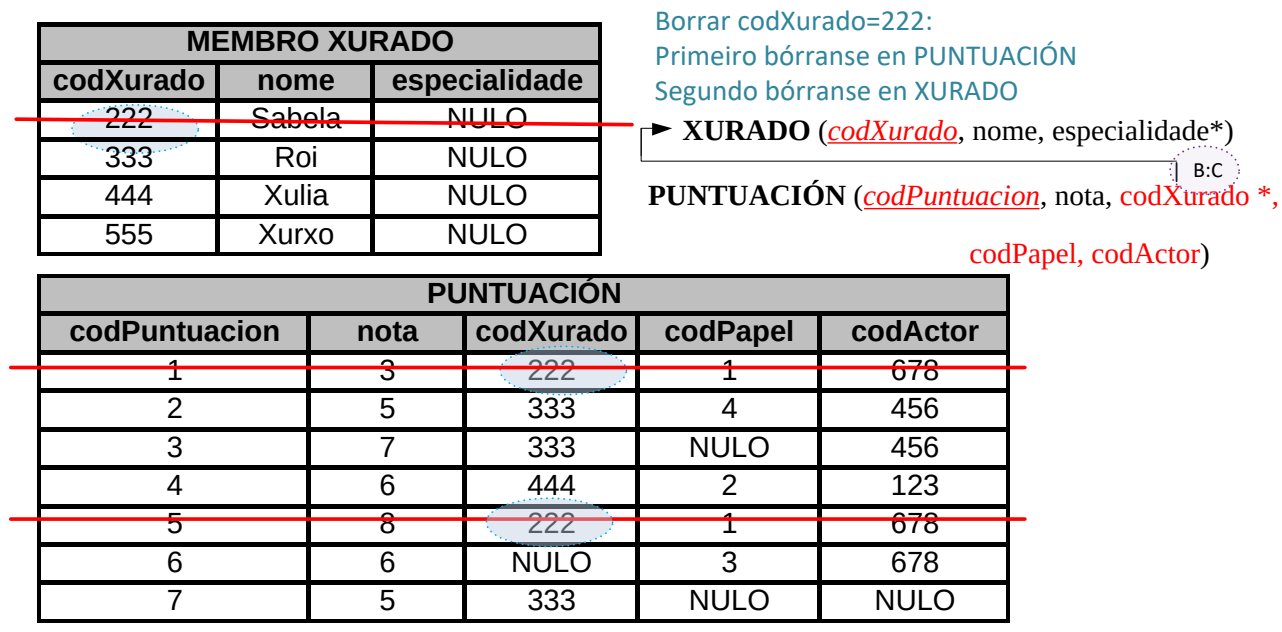



| MEMBRO XURADO | | |
|---|---|---|
| **codXurado** | **nome** | **especialidade** |
| 222 | Sabela | NULO |
| 333 | Roi | NULO |
| 444 | Xulia | NULO |
| 555 | Xurxo | NULO |

| PUNTUACIÓN | | | | |
|---|---|---|---|---|
| **codPuntuacion** | **nota** | **codXurado** | **codPapel** | **codActor** |
| 1 | 3 | 222 | 1 | 678 |
| 2 | 5 | 333 | 4 | 456 |
| 3 | 7 | 333 | NULO | 456 |
| 4 | 6 | 444 | 2 | 123 |
| 5 | 8 | 222 | 1 | 678 |
| 6 | 6 | NULO | 3 | 678 |
| 7 | 5 | 333 | NULO | NULO |

Figura 1.12. Exemplo de borrado con comportamento por defecto (restrinxido)

- Propagar a eliminación en cascada:
  - Primeiro elimínanse todas as tuplas de "R2" da relación que referencia á tupla "t" a eliminar.
  - Posteriormente elimínase "t"




| MEMBRO XURADO | | |
|---|---|---|
| **codXurado** | **nome** | **especialidade** |
| ~~222~~ | ~~Sabela~~ | ~~NULO~~ |
| 333 | Roi | NULO |
| 444 | Xulia | NULO |
| 555 | Xurxo | NULO |

| PUNTUACIÓN | | | | |
|---|---|---|---|---|
| **codPuntuacion** | **nota** | **codXurado** | **codPapel** | **codActor** |
| ~~1~~ | ~~3~~ | ~~222~~ | ~~1~~ | ~~678~~ |
| 2 | 5 | 333 | 4 | 456 |
| 3 | 7 | 333 | NULO | 456 |
| 4 | 6 | 444 | 2 | 123 |
| ~~5~~ | ~~8~~ | ~~222~~ | ~~1~~ | ~~678~~ |
| 6 | 6 | NULO | 3 | 678 |
| 7 | 5 | 333 | NULO | NULO |

Figura 1.13. Exemplo de borrado con propagación en cascada

- Posta a valor por defecto:
  - Substitúense os valores de "R2" que referencian a "t" polo valor preestablecido.
  - Modifícase a propia tupla "t"

| MEMBRO XURADO | | |
|---|---|---|
| codXurado | nome | especialidade |
| 222 | Sabela | NULO |
| 333 | Roi | NULO |
| 444 | Xulia | NULO |
| 555 | Xurxo | NULO |

Borrar codXurado=222:
Primeiro modificáse en PUNTUACIÓN
Segundo bórranse en XURADO

**XURADO** (*codXurado*, nome, especialidade*)

B:VALOR

**PUNTUACIÓN** (*codPuntuacion*, nota, codXurado *, codPapel, codActor)

| PUNTUACIÓN | | | | |
|---|---|---|---|---|
| codPuntuacion | nota | codXurado | codPapel | codActor |
| 1 | 3 | VALOR | 1 | 678 |
| 2 | 5 | 333 | 4 | 456 |
| 3 | 7 | 333 | NULO | 456 |
| 4 | 6 | 444 | 2 | 123 |
| 5 | 8 | VALOR | 1 | 678 |
| 6 | 6 | NULO | 3 | 678 |
| 7 | 5 | 333 | NULO | NULO |

Figura 1.14. Exemplo de borrado con posta a valor por defecto

– Posta a NULOS: só se a clave allea permite nulos desencadeáranse as seguintes accións:
  – As tuplas de "R2" que fan referencia a "t" substitúense polos valor NULO
  – Elimínase a tupla "t"

| MEMBRO XURADO | | |
|---|---|---|
| codXurado | nome | especialidade |
| 222 | Sabela | NULO |
| 333 | Roi | NULO |
| 444 | Xulia | NULO |
| 555 | Xurxo | NULO |

Borrar codXurado=222:
Primeiro modifícase en PUNTUACIÓN
Segundo bórranse en XURADO

**XURADO** (*codXurado*, nome, especialidade*)

B:N

**PUNTUACIÓN** (*codPuntuacion*, nota, codXurado *, codPapel, codActor)

| PUNTUACIÓN | | | | |
|---|---|---|---|---|
| codPuntuacion | nota | codXurado | codPapel | codActor |
| 1 | 3 | NULO | 1 | 678 |
| 2 | 5 | 333 | 4 | 456 |
| 3 | 7 | 333 | NULO | 456 |
| 4 | 6 | 444 | 2 | 123 |
| 5 | 8 | NULO | 1 | 678 |
| 6 | 6 | NULO | 3 | 678 |
| 7 | 5 | 333 | NULO | NULO |

Figura 1.15. Exemplo de borrado con posta a nulos

- **II. Se modificamos unha clave allea (FK) dunha relación "R2" a un valor non existente na clave primaria (PK) de "R1" o comportamento será sempre de rexeitamento da operación.**

| MEMBRO XURADO | | |
|---|---|---|
| codXurado | nome | especialidade |
| 222 | Sabela | NULO |
| 333 | Roi | NULO |
| 444 | Xulia | NULO |
| 555 | Xurxo | NULO |

Modificar codXurado =NULO a 777

►**XURADO** (*codXurado*, nome, especialidade*)

**PUNTUACIÓN** (*codPuntuacion*, nota, codXurado *, codPapel, codActor)

Erro: non existe

| PUNTUACIÓN | | | | |
|---|---|---|---|---|
| codPuntuacion | nota | codXurado | codPapel | codActor |
| 1 | 3 | 222 | 1 | 678 |
| 2 | 5 | 333 | 4 | 456 |
| 3 | 7 | 333 | NULO | 456 |
| 4 | 6 | 444 | 2 | 123 |
| 5 | 8 | 222 | 1 | 678 |
| 6 | 6 | 777 | 3 | 678 |
| 7 | 5 | 333 | NULO | NULO |

Figura 1.16 Exemplo de modificación bloqueada por violar a integridade referencial

- **III. Se modificamos o valor da clave primaria (PK) dunha tupla "t" da relación "R1" a cal é referenciada por unha o varias tuplas de "R2" , pódense establecer os seguintes comportamentos**:
  - Operación por defecto que é rexeitar sempre a operación: só se permite modificar a clave primaria (PK) de "t" se ningunha outra tupla a referencia.

A tupla codXurado =222 non pode modificarse a 888
A tupla codXurado = 555 si pode modificarse a 888

| MEMBRO XURADO | | |
|---|---|---|
| codXurado | nome | especialidade |
| 222 | Sabela | NULO |
| 333 | Roi | NULO |
| 444 | Xulia | NULO |
| 555 | Xurxo | NULO |

► **XURADO** (*codXurado*, nome, especialidade*)

**PUNTUACIÓN** (*codPuntuacion*, nota, codXurado *, codPapel, codActor)

| PUNTUACIÓN | | | | |
|---|---|---|---|---|
| codPuntuacion | nota | codXurado | codPapel | codActor |
| 1 | 3 | 222 | 1 | 678 |
| 2 | 5 | 333 | 4 | 456 |
| 3 | 7 | 333 | NULO | 456 |
| 4 | 6 | 444 | 2 | 123 |
| 5 | 8 | 222 | 1 | 678 |
| 6 | 6 | NULO | 3 | 678 |
| 7 | 5 | 333 | NULO | NULO |

Figura 1.17. Exemplos de modificacións nun comportamento por defecto.

  - Propagar o cambio en cascada:
    - Toda tupla da relación "R2" seguirá referenciando a "t" cambiando o seu valor de clave allea (FK) ao novo de clave primaria de "t".
    - Posteriormente modificarase o valor de clave primaria (PK) de "t".

| MEMBRO XURADO | | |
|---|---|---|
| **codXurado** | **nome** | **especialidade** |
| 888 | Sabela | NULO |
| 333 | Roi | NULO |
| 444 | Xulia | NULO |
| 555 | Xurxo | NULO |

Modificar codXurado=222 a 888:
Primeiro modifícase en PUNTUACIÓN
Segundo modifícase en XURADO

► **XURADO** (*codXurado*, nome, especialidade*)

M:C

**PUNTUACIÓN** (*codPuntuacion*, nota, codXurado *, codPapel, codActor)

| PUNTUACIÓN | | | | |
|---|---|---|---|---|
| **codPuntuacion** | **nota** | **codXurado** | **codPapel** | **codActor** |
| 1 | 3 | 888 | 1 | 678 |
| 2 | 5 | 333 | 4 | 456 |
| 3 | 7 | 333 | NULO | 456 |
| 4 | 6 | 444 | 2 | 123 |
| 5 | 8 | 888 | 1 | 678 |
| 6 | 6 | NULO | 3 | 678 |
| 7 | 5 | 333 | NULO | NULO |

Figura 1.18. Exemplo de modificación con propagación en cascada

- Posta a valor por defecto:
  - Toda tupla da relación "R2" que referencia a "t" cambia o seu valor de clave allea (FK) ao valor por defecto.
  - Posteriormente modificarase o valor de clave primaria (PK) de "t" a ese valor preestablecido.

| MEMBRO XURADO | | |
|---|---|---|
| **codXurado** | **nome** | **especialidade** |
| 888 | Sabela | NULO |
| 333 | Roi | NULO |
| 444 | Xulia | NULO |
| 555 | Xurxo | NULO |

código codXurado=222 a 888
Primeiro modifícase en PUNTUACIÓN
Segundo modifícase en XURADO

► **XURADO** (*codXurado*, nome, especialidade*)

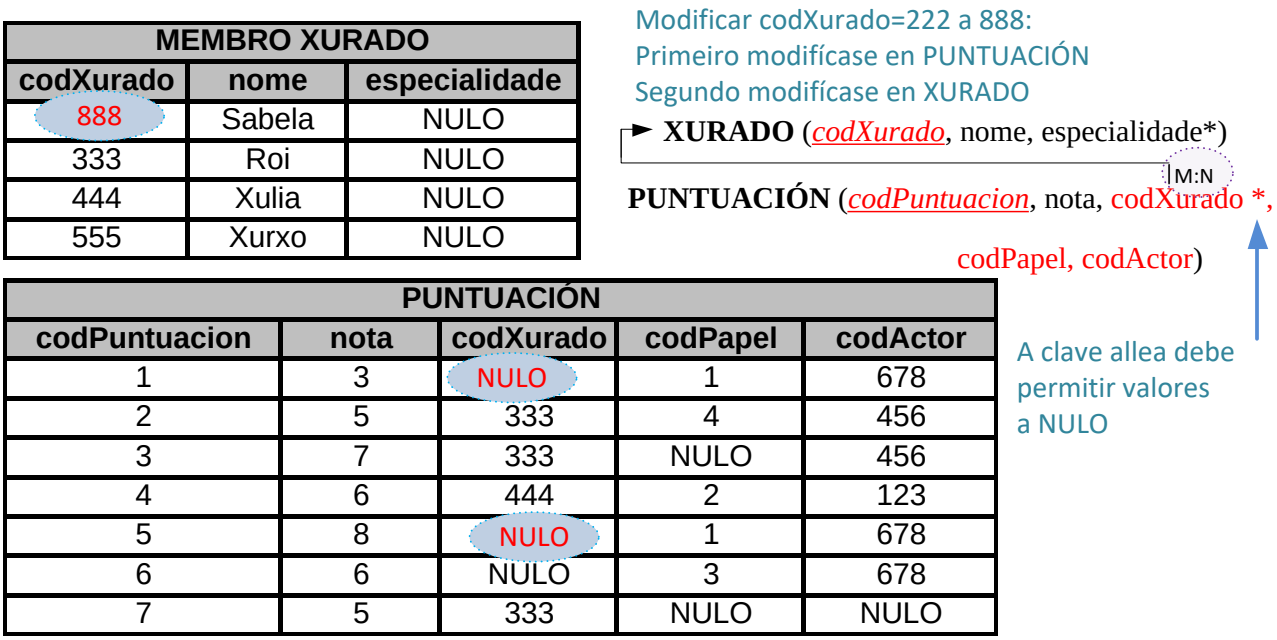
B:VALOR

**PUNTUACIÓN** (*codPuntuacion*, nota, codXurado *, codPapel, codActor)

| PUNTUACIÓN | | | | |
|---|---|---|---|---|
| **codPuntuacion** | **nota** | **codXurado** | **codPapel** | **codActor** |
| 1 | 3 | VALOR | 1 | 678 |
| 2 | 5 | 333 | 4 | 456 |
| 3 | 7 | 333 | NULO | 456 |
| 4 | 6 | 444 | 2 | 123 |
| 5 | 8 | VALOR | 1 | 678 |
| 6 | 6 | NULO | 3 | 678 |
| 7 | 5 | 333 | NULO | NULO |

Figura 1.19. Exemplo de modificación con posta a un valor predeterminado

- Posta a NULOS: só se a clave allea permite nulos desencadearanse as seguintes accións:
  - Toda tupla da relación "R2" que referencia a "t" substituirase a valor NULL.
  - Modificarase o valor da clave primaria (PK) de "t".




| MEMBRO XURADO | | |
|---|---|---|
| codXurado | nome | especialidade |
| 888 | Sabela | NULO |
| 333 | Roi | NULO |
| 444 | Xulia | NULO |
| 555 | Xurxo | NULO |

| PUNTUACIÓN | | | | |
|---|---|---|---|---|
| codPuntuacion | nota | codXurado | codPapel | codActor |
| 1 | 3 | NULO | 1 | 678 |
| 2 | 5 | 333 | 4 | 456 |
| 3 | 7 | 333 | NULO | 456 |
| 4 | 6 | 444 | 2 | 123 |
| 5 | 8 | NULO | 1 | 678 |
| 6 | 6 | NULO | 3 | 678 |
| 7 | 5 | 333 | NULO | NULO |

Figura 1.20. Exemplo con modificacións de posta a NULO

### Restricións explícitas, baseadas no esquema ou de negocio

Este grupo de restricións son específicas de cada base de datos, sendo definidas e almacenadas no esquema da BD mediante o uso de SQL ou a linguaxe propia do SXBD. Non poden ser violadas por ningunha operación de actualización.

- **Restricións de verificación (CHECK):** As restricións de verificación tamén chamadas check, permiten impor condicións a elementos ou atributos dunha relación. Por exemplo, podemos impor que o campo idade na relación CLIENTE estea comprendida entre 18 e 100 anos. Comproba en toda operación de actualización, se o predicado é certo ou falso e, no segundo caso, rexeita a operación. A restrición de verificación defínese sobre un único elemento (dentro dun CREATE TABLE) e pode ou non ter nome.
- **Restricións de aserción (ASSERTION):** Estas restricións son similares ás de verificación, a única diferenza é que agora as condicións poden ser de atributos de máis dunha táboa. Por tanto, a súa definición non vai unida a un determinado elemento do esquema e sempre ha de ter un nome. Por exemplo, podemos impor que un cliente para opinar dunha serie teña que ver ao menos tres episodios da mesma, outro exemplo sería que os empregados que participan nun proxecto sexan do mesmo departamento que o seu responsable.
- **Restrición de disparadores (TRIGGERS):** Restrición na que o usuario pode especificar libremente a resposta (acción) ante unha determinada condición. Son obxectos programados polo usuario para tal fin. Así como as anteriores regras de integridade son declarativas, os disparadores son procedimentais, sendo preciso que o usuario escriba o procedemento que ha de aplicarse no caso de que se cumpra a condición. Por exemplo, facer que cando unha película se dese de alta o atributo *numProgramasDirixe* na relación DIRECTOR increméntese nunha unidade na fila correspondente a ese DIRECTOR.

## 1.2.4 Esquema e instancia de base de datos e o modelo relacional

Nunha base de datos distínguense dúas partes:

- **O esquema**, que corresponde ao deseño lóxico da base de datos. Está constituído polo conxunto de esquemas de relación (conxunto de nomes de pares {atributo:dominio} xunto coas súas restricións de integridade). Os esquemas de relación non soen variar xa que implicaría a reescritura de miles de tuplas e a introdución de novos atributos para tuplas existentes.
- **A instancia** trátase da visión do contido da base de datos nun determinado intre. Está constituída polo conxunto de instancias das relacións que son variables no tempo, sendo habitual o engadido, a modificación ou o borrado de tuplas.

Pódese definir unha BD relacional como o conxunto de ocorrencias válidas dun esquema relacional.

En termos de implementación en DDL-SQL, un esquema de base de datos constará de :

- O conxunto de esquemas de relacións creados coa sentenza "CREATE TABLE".
- O conxunto de definicións de dominios propios do SXBD e creados polo usuario coa sentenza CREATE DOMAIN.
- O conxunto de restricións de integridade entre relacións e sobre dominios:
  - Clave primaria (PK): "PRIMARY KEY (nomeAtributoClave)"
  - Clave allea (FK): "FOREING KEY (nomeAtributo) REFERENCES nomeTaboaReferencida (nomeAtributoClave) *opcións de borrado e modificación*"
  - Creación de verificacións:
    - CHECK N_HORAS > 30 en CURSO_DOUTORAMENTO
    - CHECK (dataInicio >= dataFin)
  - Creación de aserción:
    - CREATE ASSERTION CONCEDE_SOLICITA AS

      CHECK (SELECT Cod_Estudante, Cod_Bolsa FROM CONCEDE) IN (SELECT Cod_Estudante, Cod_Bolsa FROM SOLICITA));
  - Creación de disparadores:
    - CREATE TRIGGER Comprobar_Matriculados AFTER INSERT ON SOLICITA

      DECLARE NUM_SOLICITUDES Number;

      BEGIN SELECT COUNT(*) INTO NUM_SOLICITUDES FROM SOLICITA;
      IF NUM_SOLICITUDES > 50 THEN
      INSERT INTO MENSAXES VALUES ('Hai máis de 50 solicitudes');
      END IF;
      END Comprobar_Matriculados;
  - Creación de vistas:
    - CREATE VIEW

Tarefa 1 : Diferenciar tipos de restricións.

Tarefa 2: Implementar a integridade referencial

Tarefa 3: Establecer restricións de negocio

### 1.2.5 O Modelo Relacional e a Arquitectura ANSI

O modelo relacional pode examinarse no marco da arquitectura ANSI a tres niveis, xa que todos os obxectos do modelo, isto é, dominios, relacións, claves e restricións constitúen o esquema conceptual da arquitectura ANSI. As relacións, nas que o seu esquema defínese en SQL mediante unha sentenza CREATE TABLE, denomínanse táboa base ou reais, xa que teñen unha representación directa no almacenamento interno.

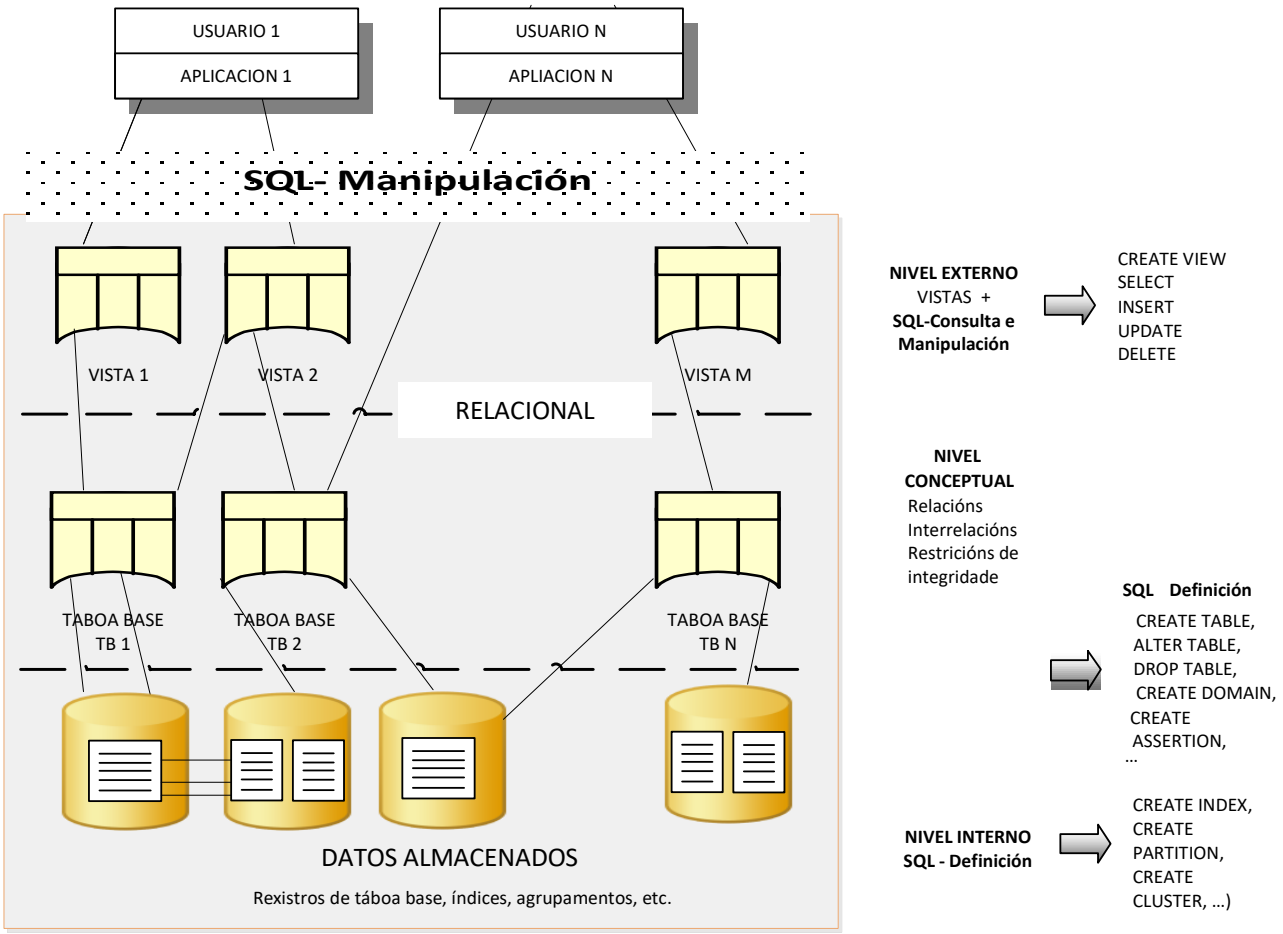



Figura 1. 21 Modelo relacional e arquitectura ANSI

A organización Internacional de Estándares nas normas correspondentes a Bases de datos e SQL de 1992, subdivide as táboas base en táboas persistentes e táboas temporais; as táboas persistentes, almacénanse en memoria secundaria e permanecen cando termina a sesión na que foron creadas, mentres que as temporais só se materializan, e teñen existencia, en tanto exista a sesión. Segundo esta división, só poderían considerarse como constituíntes do esquema conceptual as táboas persistentes.

Existen outro tipo de táboas, denominadas táboas virtuais ou vistas, que se definen sobre unha ou máis táboas base. As vistas son ventás sobre táboas reais, das que só se almacena a súa definición, e non teñen, polo tanto, representación directa no almacenamento; equivalen ao esquema externo da arquitectura ANSI.

Polo que respecta ao esquema interno, o modelo relacional non especifica absolutamente nada posto que se trata dun modelo de nivel lóxico. En xeral, cada rexistro do esquema interno correspóndese cunha tupla de relacións base, pero non tería por qué haber unha correspondencia biunívoca, xa que un rexistro podería estar constituído por varias tuplas de distintas relacións, ou viceversa, isto é, unha tupla podería dividirse en varios rexistros.

Ademais, cada produto ofrece os elementos internos, como índices, agrupamentos (clusters), táboas hash, particións, etc., que o deseñador considera máis oportunos co fin de mellorar o rendemento do sistema.

Na figura anterior dáse unha visión global do modelo relacional dentro do marco definido pola arquitectura ANSI, amosando ademais certas sentenzas da linguaxe SQL que axudan a definir os elementos correspondentes nos tres niveis. As sentenzas relativas ao nivel interno variarán dun produto a outro e non están estandarizadas.

Pódese observar, que o modelo relacional teórico adáptase bastante ben á arquitectura ANSI, coas seguintes excepcións:

- Dispoñendo das oportunas autorizacións, a calquera usuario se lle permite "ver", tanto as relacións base como as vistas, mentres que na arquitectura ANSI para un usuario a base de datos está limitada ao esquema externo –vistas- xa que o esquema conceptual –relacións base- é exclusivamente responsabilidade do administrador e só poden ser definido e manexado por este.
- Aínda que as vistas correspóndense cos esquemas externos de ANSI, no modelo relacional non todas as vistas son actualizables.
- Na práctica, moitos produtos non responden á arquitectura a tres niveis, xa que as definicións do esquema conceptual e do esquema interno non están claramente diferenciadas.

## 1.2.6 Deseño lóxico

O método de deseño lóxico debe ser aplicable a diferentes tecnoloxías de BD, polo tanto non debe empregar características específicas dos SXBD comerciais ata que sexa imprescindible. O propósito é conseguir que o resultado do deseño sexa altamente transportable, non só para ser empregado por diferentes xestores, senón tamén para facilitar a migración entre versións dun mesmo sistema.

### 1.2.6.1 Etapas

#### Deseño lóxico estándar

Esta etapa parte do esquema conceptual resultante da etapa de deseño conceptual de datos, e xunto cos requisitos de usuario, máis os criterios económicos e tecnolóxicos, construirase un "esquema lóxico estándar". O modelo lóxico estándar pode expresarse empregando varias técnicas, entre as que se atopan o diagrama de estrutura de base de datos (DD) e o modelo relacional. O esquema lóxico estándar describirase empregando unha linguaxe de definición de datos (DDL) estándar, habitualmente SQL-92 (estándar ISO)

#### Deseño lóxico específico

A partir do esquema lóxico estándar obtido na etapa anterior elaborarase o esquema específico a implementar nun SXBD comercial concreto empregando a propia linguaxe de definición de datos (DDL) do xestor.

A finalidade é a adaptación do deseño lóxico obtido ás características propias do SXBD. Neste contexto poden darse os seguintes casos:

- O SXBD soporta todos os elementos do modelo lóxico estándar sen restricións. A transformación do esquema estándar ao específico e directa, tendo só que transcribir á sintaxe propia do SXBD empregado (normalmente SQL).

- O SXBD non soporta certos conceptos, ou ben, os soporta pero con restricións. Neste caso teranse que implementar novas utilidades ou realizar unha programación complementaria ou transferir aos programas as restricións non soportadas.

Tarefa 4. Diferenciar os elementos e fases do deseño lóxico

#### 1.2.6.2 Transformación do Modelo Conceptual MER ao Modelo Lóxico Relacional de Codd

##### Principios básicos. O grafo relacional

O grafo relacional, tamén denominado grafo de combinación ou diagrama esquemático, é a representación dun sistema mediante un conxunto de relacións vinculadas entre si por unha ou varias claves alleas, empregando unha simboloxía concreta que se detallase a continuación.

É o resultado gráfico da transformación do esquema conceptual entidade-interrelación ao esquema lóxico estándar. A partir del poderase crear a BD coa asistencia do SXBD seleccionado.

Para a súa obtención emprégase unha técnica moi sinxela que permite incluír información sobre:

- Atributos
- Claves primarias
- Claves alternativas
- Claves alleas

A representación utilizada é a proposta por Adoración de Miguel e Mario Piattini (aínda que existen outras propostas e versións):

- Cada táboa represéntase por un nome de relación e un conxunto de atributos entre parén-tese.
- Os atributos que compoñan a clave principal mostraranse subliñados e neste documentos engádese a característica de cor vermella.
- Os atributos que compoñan a clave alternativa en estilo de fonte groso.
- Os atributos que compoñan a clave allea visualizaranse en cor vermella e estilo de fonte cursiva, e dende estes debuxaranse uns arcos con punta de frecha que cheguen ata a clave principal ou alternativa á que se fai refencia.
- Os atributos opcionais especifícanse cun asterisco.

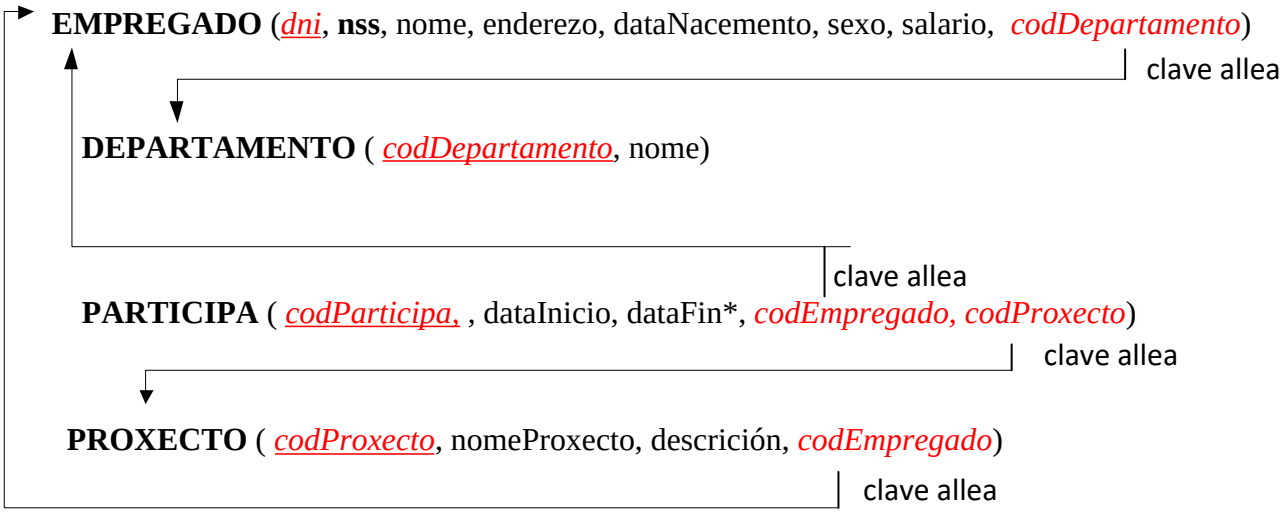



Figura 1.22. Exemplo de grafo relacional

## Transformacións principais baseadas no modelo básico

O paso dun esquema no modelo MER de Chen ao relacional de Codd fundaméntase nos tres principios seguintes:

- a) Todo tipo de entidade convértese nunha relación que mantén atributos e claves respecto a entidade de orixe.
- b) Todo tipo de interrelación con tipo de correspondencia N:M, transfórmase nunha relación mediante a técnica de propagación de clave; nela, figurarán como claves alleas as claves principais das relacións procedentes da conversión de tipos de entidade que se relacionaban a través da interrelación.
- c) Todo tipo de interrelación con tipo de correspondencia 1:N, ou ben, se traduce no fenómeno de propagación de clave, ou ben, xera unha nova relación.

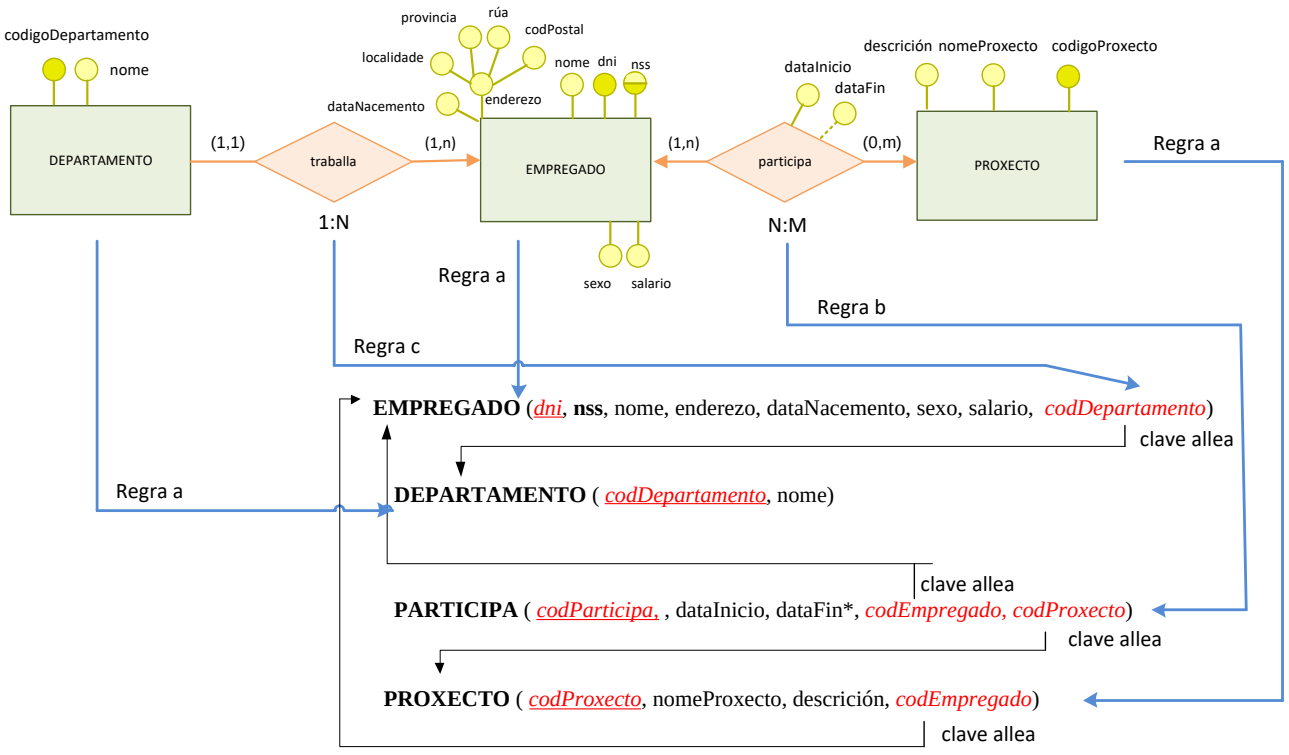


Figura 1.23. Exemplo de aplicación das regras básicas de transformación

Aplicando as regras á figura 1.23, as táboas resultantes serán EMPREGADO, PROXECTO, DEPARTAMENTO e PARTICIPA, onde a última corresponde á transformación do tipo de interrelación "participa" pola aplicación da regra b. Esta interrelación debe ter como claves alleas as principais de PROXECTO e DEPARTAMENTO. Así mesmo, a interrelación "traballa" tradúcese nunha propagación de clave de DEPARTAMENTO á relación EMPREGADO (a clave principal de DEPARTAMENTO esténdese como clave allea en EMPREGADO).

Na relación PARTICIPA non se aplicou a regra xenérica que indica que as relacións que xorden dunha interrelación N:M deben ter como clave principal a concatenación das claves principais das entidades interrelaciondas; neste caso a razón da conduta radica en evitar unha clave composta de tres atributos ( as dúas alleas máis a data de Inicio para que cumpra o principio de unicidade ). A razón e evitar unha clave composta é o costo do manexo de índices con claves grandes.

O modelo relacional completo é o que se mostra no grafo relacional. Debido a que este modelo non distingue entre Entidades e interrelacións, ambos conceptos represéntanse mediante relacións. Isto implica unha perda de semántica con respecto o modelo entidade-interrelación:

- As interrelacións N:M non se distinguen das entidades, xa que, ambas transfórmanse en táboas.
- As interrelacións 1:N sóense representar mediante unha propagación de clave desaparecendo o nome da relación, aínda que este nome pode ser asignado ás clave alleas xeradas.

Seguidamente estudarase o modo en que o grafo relacional permite recoller os diferentes elementos, restricións, etc, do modelo relacional. Empregarase o SQL2 xunto con outros elementos como disparadores e procedementos, pendentes da súa estandarización nun futuro a través do SQL3, para representar case na súa totalidade o modelo MER e MERE.

### Regra 1: Transformación de dominios

Os dominios presentes no modelo entidade-interrelación, forman tamén parte do modelo relacional estándar. SQL2 presenta a sentenza CREATE DOMAIN para a súa creación. Así mesmo, é posible engadir restricións á definición relacional de dominio, de forma que este se asemelle máis o concepto de dominio do modelo entidade-interrelación.

Exemplo: Crear o dominio días da semana cos valores de tipo char, de lonxitude 10, e cos valores do rango correspondente aos nomes dos sete días da semana:

```
CREATE DOMAIN diaSemana AS CHAR(10)
CHECK (VALUE IN ('luns', 'martes', 'mercores', 'xoves' , 'venres', 'sabado', 'domingo'))
```

O uso de dominios é altamente recomendable, xa que ofrece maior nivel de coherencia, homoxeneidade e robustez ao código resultante.

Ademais de permitir crear dominios, os DDL dos SXBD comerciais soen implementar unha serie de dominios básicos, coñecidos como tipos de datos. Os máis comúns recóllense na seguinte táboa:

| Dominio | Posibilidades |
|---|---|
| Numérico | smallInt |
| | integer/int |

| | |
|---|---|
| | decimal (*díxitos significativos, posicións decimais*) |
| | float |
| | real |
| Alfanumérico | char (*lonxitude cadea*) |
| | var char (lonxitude cadea) |
| Data (algúns xestores almacenan a hora neste dominio evitando o resto de dominios de hora e data) | Date |
| Hora | time |
| Data e hora | Timestamp |

Tarefa 5: Crear dominios

### Regra 2: Transformación de entidades

Entre os principios básicos, aparece o que establece que cada tipo de entidade setransforma nunha relación, que se denominará co mesmo nome que a entidade da que provén. Para a súa definición, o DDL-SQL emprega a sentenza CREATE TABLE.

### Regra 3: Transformación de atributos de entidades

Cada atributo dunha entidade transformarase nunha columna da relación creada a partir do tipo de entidade. Tendo en conta que existen atributos identificativos (principal, alternativo) e non identificadores, podemos subdividir esta regra en tres:

#### Regra 3.1: Atributos identificadores principais (AIP)

Os atributos que son identificadores principais de cada tipo de entidade pasan a formar a clave primaria da relación. Por exemplo, na figura 1.23 o atributo AIP dni da entidade EMPREGADO xera a clave principal dni na relación EMPREGADO.

Na linguaxe lóxico-estándar do modelo relacional (SQL) emprégase a cláusula PRIMARY KEY para identificar este concepto na descrición da táboa dentro da orde CREATE TABLE. Esta cláusula pode empregarse a nivel de columna, ou, cando o AIP é composto, a nivel de táboa.

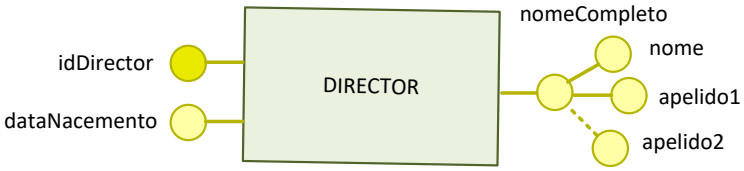
#### Regra 3.2: Atributos identificadores alternativos (AIA)

Os atributos identificadores alternativos exprésanse en SQL mediante a cláusula UNIQUE na definición de columna ou columnas implicadas dentro da orde CREATE TABLE. Débese forzar se é o caso a que esta clave non tome o valor NULO mediante a sentenza SQL NOT NULL xa que a conduta por defecto por ausencia de cláusulas é a admisión de NULOS.

### Regra 3.3: Atributos non identificadores

Estes atributos forman as columnas da relación á que pertencen, os cales tomarán por defecto a posibilidade de ser NULOS. Para evitalo indícase o contrario mediante a cláusula NOT NULL.

**MER**




**Intensión MR**

**DIRECTOR** ( idDirector, nome, apelido1,apelido2*,dataNacemento)

**Extensión MR**

| idDirector | nome | apelido1 | apelido2 | dataNacemento |
|---|---|---|---|---|
| 987654987 | Antón | Reixa | Rodríguez | 17-04-1957 |
| 236785439 | José Luís | Cuerda | Martínez | 18-02-1947 |
| 879654190 | Isabel | Coixet | Castillo | 09-04-1960 |
| 368965478 | Vincenzo | Natali | NULO | 06-01-1969 |
| 456952134 | Steve Allan | Spielberg | NULO | 18-12-1946 |
| 567423908 | Hans Christian | Tomas | Alfredson | 01-04-1965 |

clave primaria

Figura 1.24. Exemplo de transformación do tipo de entidade DIRECTOR e os seus atributos

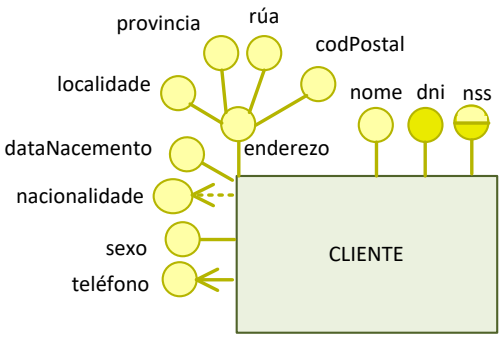
Aplicando as regras vistas ata agora, a transformación dunha entidade ao modelo relacional estándar representarase mediante o seguinte código onde se empregan os dominios correspondentes que han de ser definidos previamente.

```
CREATE TABLE DIRECTOR(
   idDirector Codigos PRIMARY KEY,
   Nome Nomes15 NOT NULL,
   Apelido1 Nomes25 NOT NULL,
   Apelido2 Nomes25 NULL,
   dataNacemento datas NOT NULL,
   )
```

### Regra 4:Transformación de atributos multivaluados

Posto que o modelo relacional non permite dominios multivaluados (que nunha mesma cela da táboa exista máis dun valor), deberá crearse unha nova relación composta por ese atributo

e a clave primaria da entidade á que pertence como clave allea, da mesma forma que cos tipos de interrelación 1:N. A concatenación de ambos atributos formará a clave primaria da nova relación.




### Intensión MR

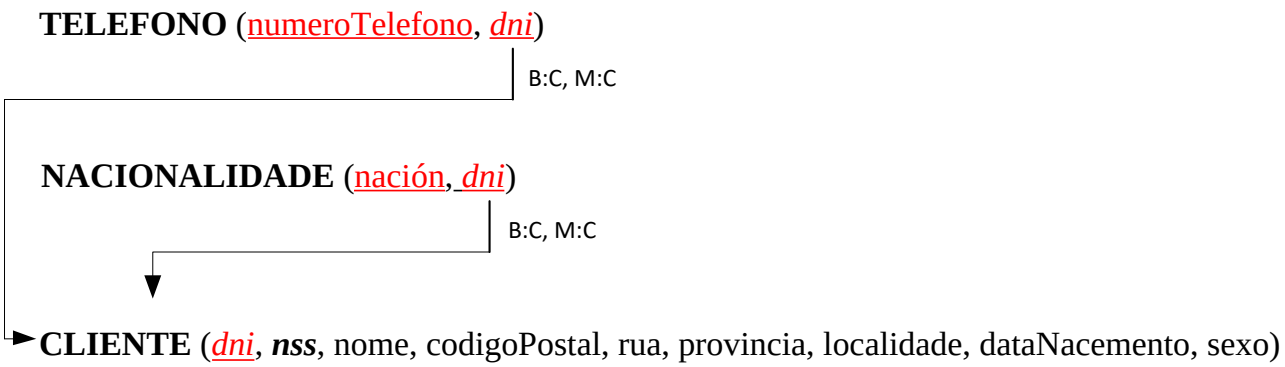



Figura 1.25. Exemplo de transformación dun atributo multivaluado

En SQL as claves alleas represéntanse coa cláusula *FOREIGN KEY nomeClaveAllea REFERENCES* dentro da sentenza de creación de táboa. Por cada clave allea obtida deberá estudarse o comportamento da relación que contén a clave allea respecto aos borrados e modificacións de tuplas nas táboas que conteñen as claves candidatas referenciadas (opcións SQL ON DELETE e ON UPDATE). As opcións permitidas en SQL estándar son:

- Operación restrinxida: en caso de non especificar a acción ou NON ACTION.
- Operación de posta a nulo: SET NULL, lembrando que require a opcionalidade dos atributos de clave allea.
- Operación posta a valor por defecto: SET DEFAULT, lembrando que require dar un valor por defecto ao atributo de clave allea ao crear a relación.
- Operación de propagación en cascada: CASCADE

```
CREATE TABLE CLIENTE(
  dni DNIS PRIMARY KEY,
  nss NSS NOT NULL UNIQUE,
  nome Nome6060 NOT NULL,
  codPostal CodigosPostais NOT NULL,
  rua Nomes25 NOT NULL,
  provincia Nomes25 NOT NULL,
  localidade Nomes25 NOT NULL,
```

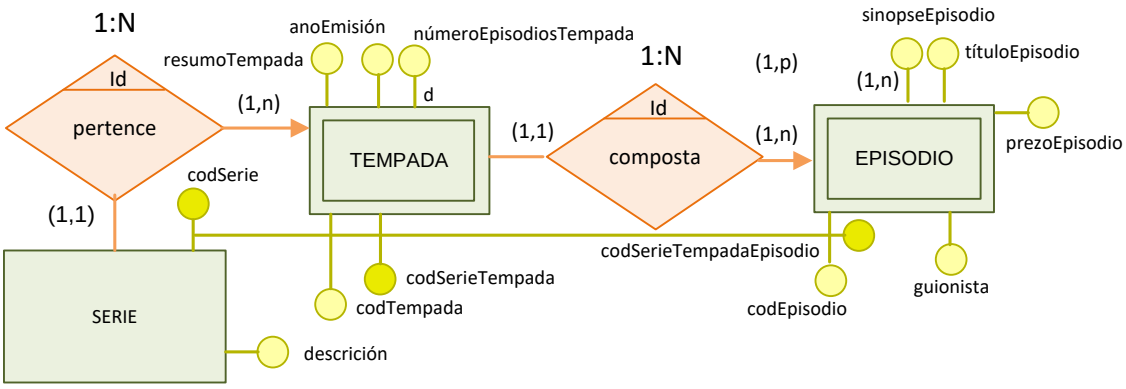
```
  dataNacemento Datas NOT NULL,
  sexo Sexos NOT NULL,
)
CREATE TABLE TELEFONOS (
  numeroTelefono Telefonos,
  dni DNIS,
  FOREIGN KEY (dni) REFERENCES Cliente(dni)
  ON DELETE CASCADE ON UPDATE CASCADE,
  PRIMARY KEY (dni, numeroTelefono),
)
CREATE TABLE NACIONALIDADE (
  nacion Naciones,
  dni DNIS,
  FOREIGN KEY (dni) REFERENCES Cliente(dni)
  ON DELETE CASCADE ON UPDATE CASCADE,
  PRIMARY KEY (dni, nacion),
)
```

## Regra 5: Transformación de atributos derivados

Os atributos derivados son aqueles no que o seu valor valor se calcula a partir de atributos de entidades ou interrelacións ou do reconto das súas ocorrencias.

Non existe para estes atributos unha representación concreta no modelo relacional, polo tanto, trátanse como atributos normais xerando unha columna da relación á que pertencen. Ademais deberase construír un disparador ou trigger que calcule o valor do atributo cada vez que se engadan ou borren as ocorrencias dos atributos que interveñen no cálculo, e engadir as restricións correspondentes.

### MER




### Intensión MR

**SERIE** ( codSerie, descrición)
B:C, M:C

**TEMPADA** ( *codSerie, codTempada*, resumoTempada, anoEmision, numEpisodiosTempada)
B:C, M:C  B:C, M:C

**EPISODIO** ( *codSerie, codTempada, codEpisodio*, tituloEpisodio, sinopseEpisodio, prezoEpisodio)

Figura 1.26. Exemplo de transformación do atributo derivado "numeroEpisodiosTempada"

No exemplo, o calculo do número de episodios dunha tempada sería derivado xa que se calculan a partir do número de ocorrencias da entidade EPISODIO, para cada unha das ocorrencias da entidade TEMPADA. Para manter o atributo actualizado, programarase un disparador para as operacións de alta e baixa das ocorrencias de EPISODIO.

```
CREATE TRIGGER increNumEpisodio AFTER INSERT OF EPISODIO
REFERENCING NEW AS novoEpisodio
FOR EACH ROW
BEGIN
  UPDATE Tempada SET numEpisodiosTempada= numEpisodiosTempada + 1
  WHERE (Tempada.CodSerie == novoEpisodio.CodSerie) AND  (Tempada. CodTempada == novoEpisodio.CodTempada)
END

CREATE TRIGGER decreNumEpisodio AFTER DELETE OF EPISODIO
REFERENCING OLD AS antigoEpisodio
FOR EACH ROW
BEGIN
  UPDATE Tempada SET numEpisodiosTempada= numEpisodiosTempada - 1
  WHERE (Tempada. CodSerie == novoEpisodio.CodSerie)
   AND  (Tempada. CodTempada == novoEpisodio.CodTempada)
END
```

Tarefa 6: Transformar entidades

### Regra 6: Transformación de interrelacións

O esquema de transformación a seguir dependerá do tipo de correspondencia da interrelación.

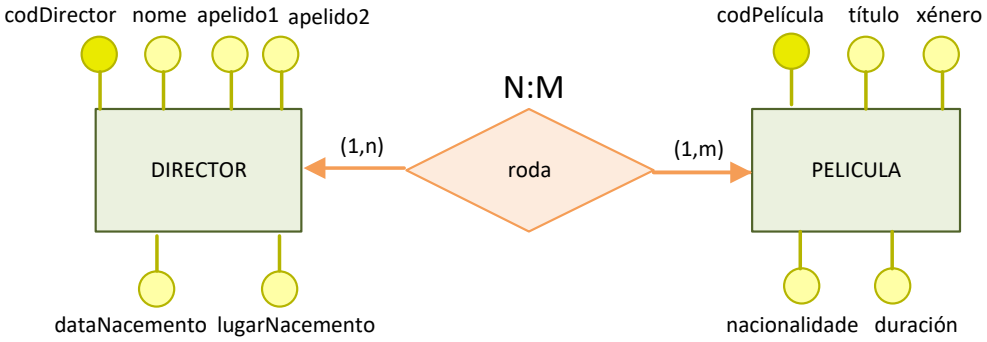
#### Regra 6.1. Transformación de interrelacións N:M (monarias e binarias)

Un tipo de interrelación N:M transformarase nunha relación ou táboa que terá sendas claves alleas apuntando a cada unha das claves primarias correspondentes aos AIP das entidades tipo que asocia (tamén se poden empregar as claves candidatas para este fin, xa que non existe ningún tipo de impedimento conceptual, aínda que non é o máis habitual).

Esta nova táboa segundo as normas de deseño aconsella que sexa nomeada co nome da interrelación da que procede, ou coa concatenación dos nomes das entidades que une, para evitar a perda semántica que aparece ao non poder diferenciar táboas procedentes da transformación de entidades e táboas procedentes da transformación de interrelacións. Un exemplo da utilidade deste nomeado son os casos de enxeñería inversa onde non se dispón da documentación do deseño conceptual.

No relativo á clave principal da relación xurdida, o máis habitual é que estea composta pola concatenación das claves primarias das entidades tipo asociadas. Nalgúns casos non chega estes dous atributos para identificar univocamente as tuplas da relación, precisando anexar outro atributo para diferencialas. Noutros casos o tamaño da clave xurdida é tan grande que é preferible a creación dunha clave propia para unha máis eficiente manipulación dos datos.

## MER




## Intensión MR

**PELICULA** ( *codPelicula*, titulo, xenero, nacionalidade, duracion)

B:C, M:C

**RODA** ( *codPelicula*, *codDirector*)

B:C, M:C

**DIRECTOR** ( *codDirector*, nome, apelido1, apelido2, dataNacemento, lugarNacemento)

Figura 1.27. Transformación xenérica dunha interrelación binaria N:M

### Cardinalidades

Outra característica que debe figurar no MR son algunhas das cardinalidades máxima e/ou mínima das entidades que participan nas interrelacións. No caso das entidades que reciben unha clave allea por propagación, o normal é recoller o valor mínimo 0 (como se amosa no exemplo seguinte) mediante a admisión de nulos ( o que obriga a xerar unha clave para a nova relación) e o valor mínimo 1 como a non admisión deste valor nulo.

No relativo aos valores mínimos en clave allea, para impoñer estas restricións requírese o uso de asertos ou disparadores, aínda que debido a súas limitacións certas normas de deseño non recomendan o seu uso

Respecto ás cardinalidades máximas, empregan tamén asertos e disparadores aínda que é moito máis inusual a súa implementación.

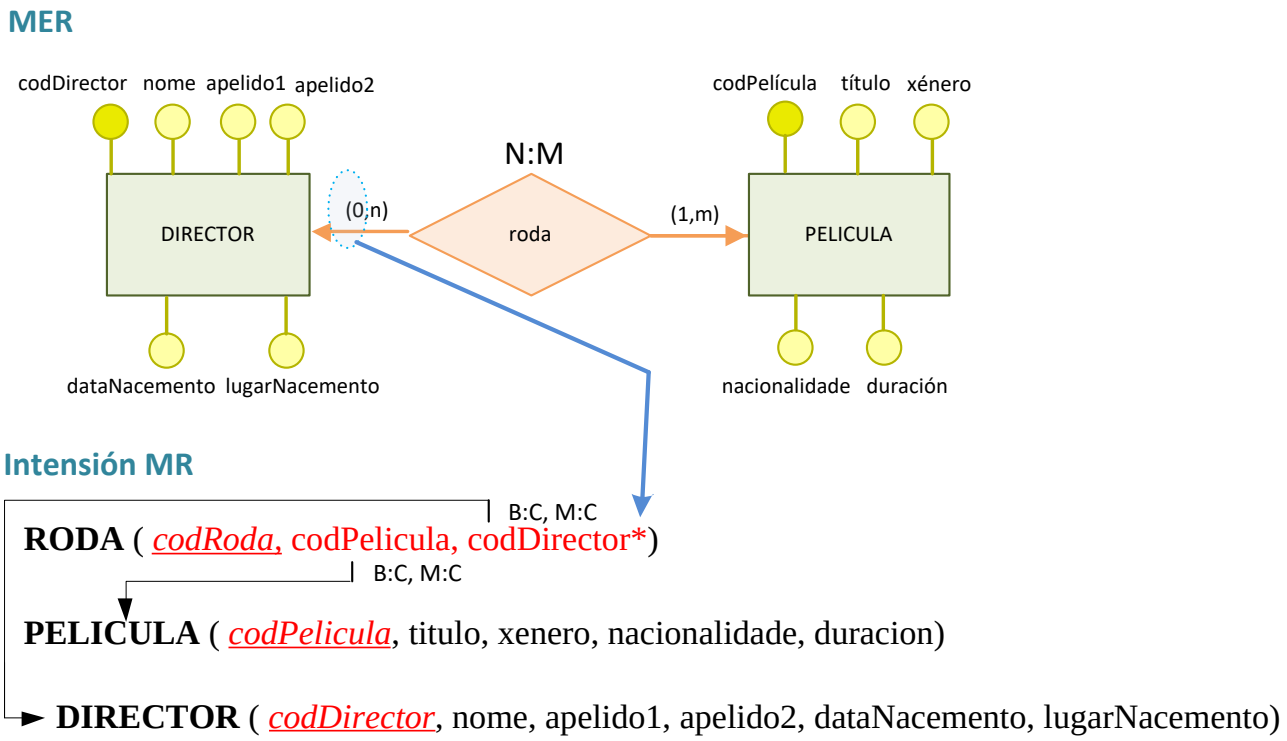



Figura 1. 28. Transformación da interrelación "roda" con cardinalidade mínima 0 para director

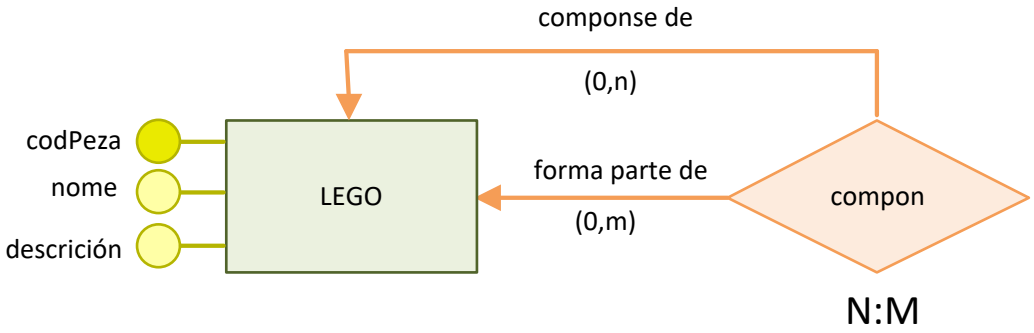
No exemplo anterior, a cardinalidade mínima 0 responde a dúas opcións semánticas:

- Descoñecemos temporalmente o director
- A película é anónima e nunca coñecerase quen a rodou.

Para discriminar ambas situacións, decídese que as películas anónimas aparezan na táboa da interrelación polo que agora a clave principal non pode ser a concatenación das claves alleas (xa que sería en parte nula, o que está prohibido polo modelo relacional.), creando unha clave principal nova para a relación "roda" xerada.

### Transformación de interrelacións N:M reflexivas ou monarias

O comportamento de transformación é o mesmo que para as interrelacións de tipo correspondencia N:M binarias, tendo en conta que ao migar a clave principal dúas veces, o nome da mesma debe modificarse para que non apareza repetida na nova táboa




**Intensión MR**

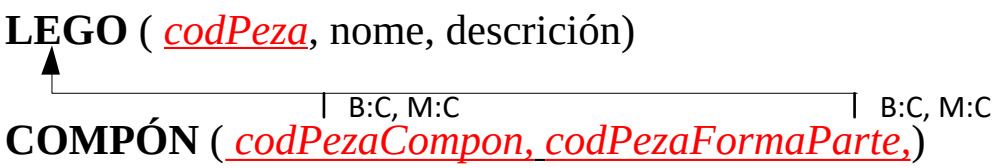



Figura 1.29. Transformación de interrelación reflexiva con tipo de correspondencia N:M

Tarefa 7. Transformar interrelacións N:M

### Regra 6.2. Transformación de interrelacións 1:N (binarias e monaria)

As interrelacións con tipos de correspondencia 1:N pódense transformar utilizando o modelo relacional mediante dous métodos:

- I. Creación dunha restrición de integridade referencial (clave allea) que reflicta a propagación do AIP do tipo de entidade con ocorrencias que se presentan como máximo nunha ocorrencia da relación (o lado 1), ata o tipo de entidade nas que as ocorrencias pódense presentar varias veces (o lado N). Segundo o simbolismo de Chen, a propagación ocorre ata o lado da frecha que é a que simboliza o lado N.Esta é a regra habitual. Ademais, se a interrelación dispón de atributos propios, estes tamén migran a relación receptora (lado N) sempre e cando non exista perda semántica ou sexa pouco relevante. Se a cardinalidade mínima é de 0, entón os atributos que compoñen a clave allea deben de permitir a súa posta a NULO, pero se a cardinalidade mínima é 1 entón os atributos de clave allea non deben permitir o valor NULO.

  Como se mostra no seguinte exemplo, existe unha perda de semántica no grafo relacional xa que nel se descoñece a procedencia do atributo data, esta perda pode liquidarse mediante o uso de comentarios ou mediante un nomeado especial onde se engada o nome da interrelación para recoñecer os atributos migrados

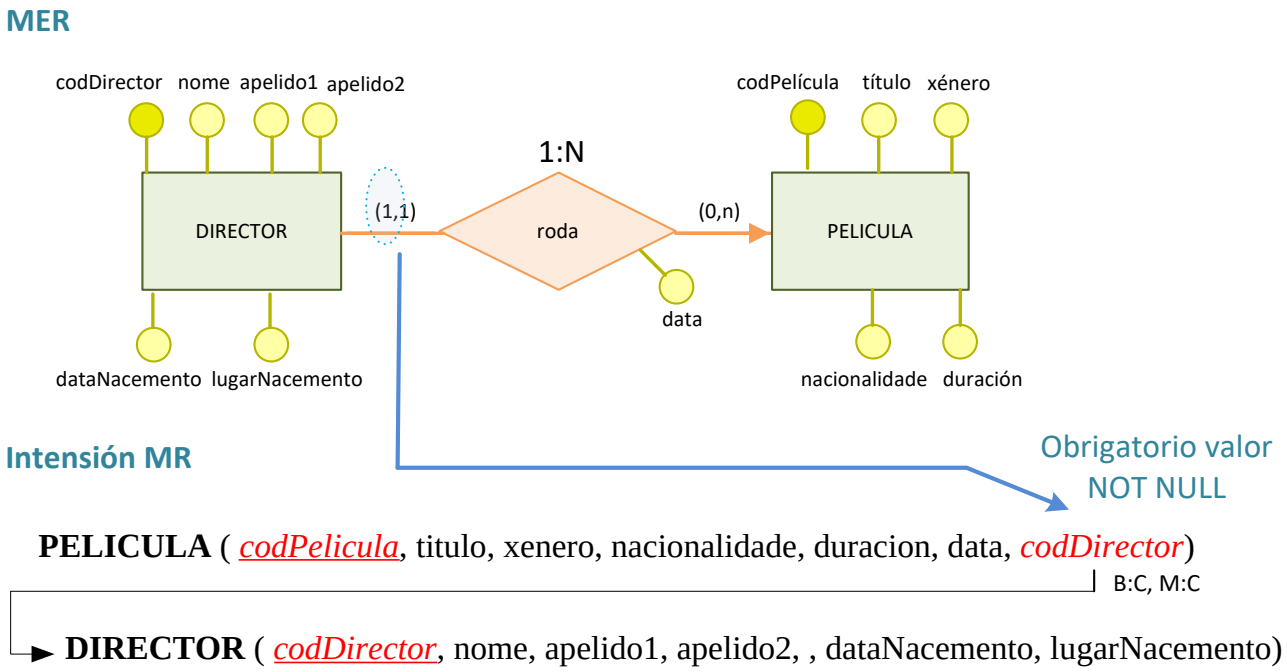
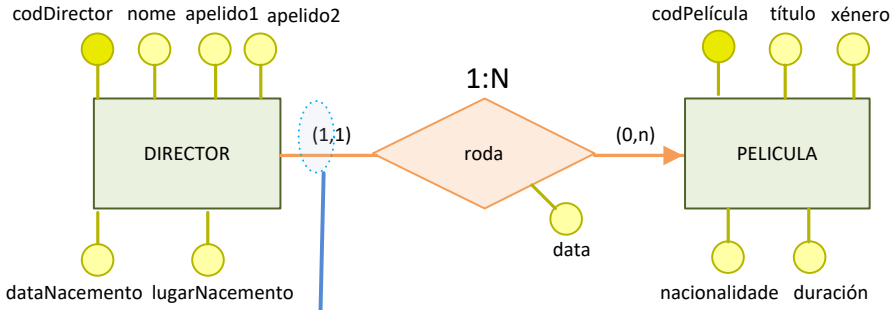



Figura 1.30. Exemplo de transformación dunha interrelación 1:N empregando propagación de claves ( solución I)

- II. Aplicar o mesmo tratamento que as relacións N:M, xerando unha nova relación que terá como atributos as claves de ambas entidades xunto cos atributos da interrelación, pero a súa clave será unicamente a da entidade que participa na interrelación con N ocorrencias. Este procedemento resérvase aos seguintes casos:
  - O número de ocorrencias interrelacionados da entidade que propaga a súa clave é moi pequeno, e polo tanto existirán moitos valores nulos na clave propagada (no exemplo, que moitas películas non teñan un director asociado)
  - Prevese que a interrelación nun futuro converterase nunha de tipo N:M deixando aberta esta posibilidade, xa que a transformación de tipo N:M é mais flexible (no exemplo, unha película sexa rodada por varios directores)
  - Cando a interrelación ten atributos propios non sendo desexable a súa propagación, conservando así a semántica orixinal. Este proceder é o recomendable nas monarias ou reflexivas (no exemplo desexase coñecer certos datos da rodaxe na interrelación).

**MER**




**Intensión MR**

Obrigatorio valor NOT NULL

B:C, M:C

**RODA** ( *codPelicula*,data, *codDirector*)

B:C, M:C

**PELICULA** ( *codPelicula*, titulo, xenero, nacionalidade, duracion)

**DIRECTOR** ( *codDirector*, nome, apelido1, apelido2, , dataNacemento, lugarNacemento)

Figura 1.31. Exemplo de transformación dunha interrelación 1:N xerando unha nova relación (solución II)

### Transformación de interrelación 1:N monarioas ou reflexivas

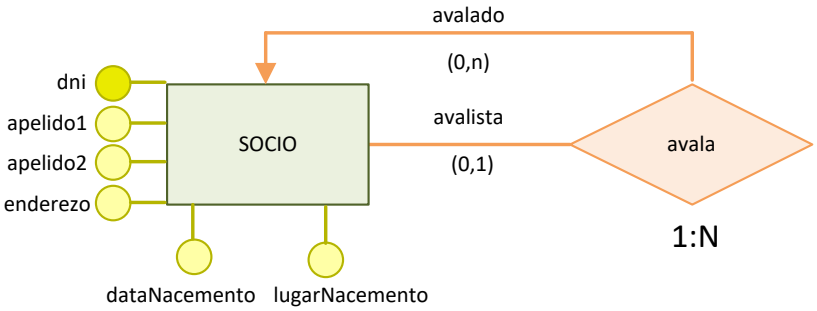
Este tipo de interrelacións reflexivas compórtanse igual que as binarias 1:N, isto é, ou migra a clave ou xera unha nova relación.

No caso de migrar a clave, ao estar na mesma táboa o nome da clave allea debe modificarse para que non apareza repetida, ademais esta solución non permite os borrados en cascada xa que ao xerar un ciclo podería borrar o contido da relación na súa totalidade.

Aínda que a norma de carácter xeral que existe nos SXBD cando se definen as claves alleas é crear antes a táboa referenciada que a táboa á que se referencia,estas interrelacións constitúen un caso especial, xa que un tipo de entidade interrelaciónase consigo mesma xerando un bucle. Polo que para crear adecuadamente a clave allea deberíase:

- Definir a táboa mediante CREATE TABLE… sen definir a clave allea pero si os atributos que a compoñen.
- Definir a clave allea engadindo a restrición FOREGN KEY mediante a sentenza de modificación de táboas ALTER TABLE (nomeTaboaModificar)

**MER**




**Intensión MR**

**SOCIO** (*dni*, apelido1, apelido2, enderezo, dataNacemento, lugarNacemento, *dniAvalado*)

B:R, M:C

Figura 1.32. Exemplo de transformación de interrelación reflexiva con tipo de correspondencia 1:N

No exemplo anterior representase o feito de que un socio concreto non avala a ningún, ou a varios socios, e que un socio pode ser avalado por un ou ningún socio.

Tarefa 8: Transformar interrelacións 1:N

### Regra 6.3. Transformación de interrelacións 1:1

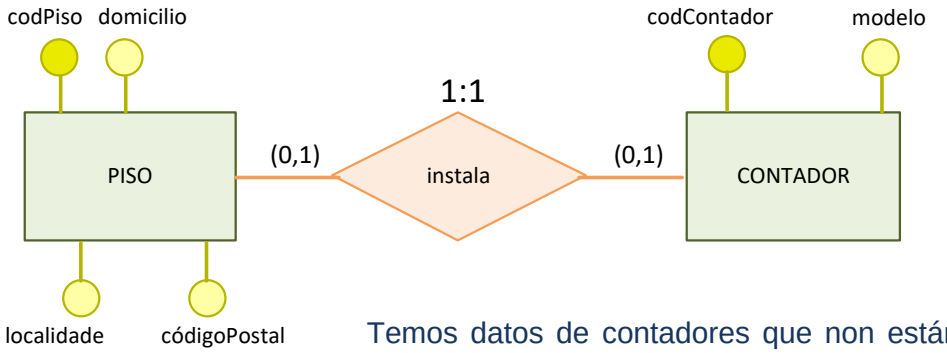
Unha interrelación de tipo de correspondencia 1:1 pódese considerar un caso particular dos tipos de correspondencia 1:N, polo que se poden aplicar as dúas opcións xa comentadas: crear unha nova táboa ou propagación de clave, tendo en conta que agora a propagación de clave pode efectuarse en ambos sentidos. Os criterios para a aplicación dunha regra ou outra , así como para propagar a clave baséanse en:

- Cardinalidades mínimas aplicadas
- Conservación da semántica do modelo
- Evitar valores nulos
- Aumento da eficiencia de procesamento

A continuación móstrase un exemplo explicando a técnica aplicada:

- Se a interrelación con tipo de correspondencia 1:1 non é obrigatoria para ningunha das entidades participantes, e dicir, se as entidades que se asocian teñen cardinalidades (0,1) e se prevé que a proporción de ocorrencias interrelacionadas non vai ser moi alta ou simplemente non se desexa a aparición de valores nulos na clave allea, entón a interrelación transformarase mediante a **creación dunha nova relación**:

**MER**




Temos datos de contadores que non están instalados, e de pisos que aínda non teñen contador

**Intensión MR**

**PISO** ( *codPiso*, domicilio, localidade, codigoPostal)

B:C, M:C

**INSTALA** ( *codContador*, *codPiso*)

B:C, M:C

**CONTADOR** ( *codContador*, modelo)

Figura 1.33. Transformación de interrelación 1:1 xerando unha nova relación

- Se a interrelación é obrigatoria para só unha das entidades interrelacionadas, é dicir, cando unha posúe cardinalidades (0,1) e a outra (1,1), e sempre que non se trate dunha relación monaria con atributos, a alternativa é propagar a clave da entidade na que as súas ocorrencias participen obrigatoriamente na interrelación (propagar da relación resultante da entidade con cardinalidades (1,1) á relación resultante da entidade con cardinalidades (0,1)). Tamén se migraría á mesma relación os atributos da interrelación, o que non debería facerse en ningún caso é a propagación en sentido contrario xa que xeraría nulos.

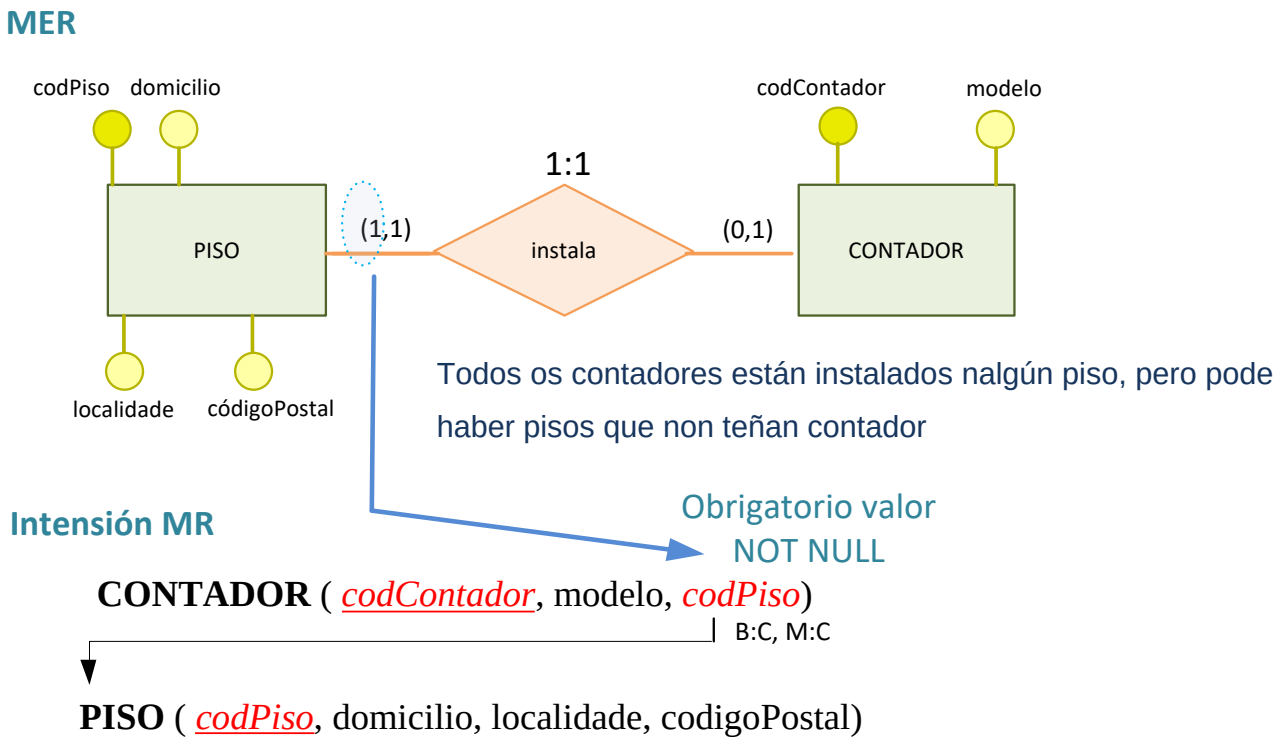
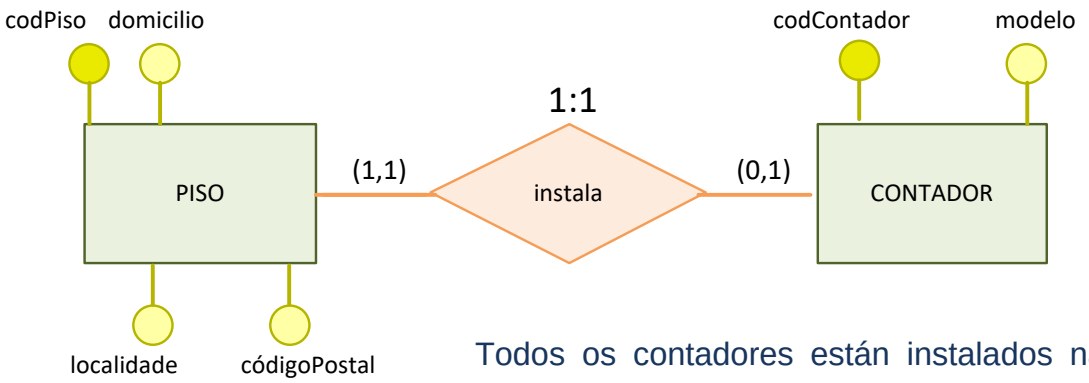



Figura 1.34. Transformación de interrelación 1:1 propagando clave

- No caso de que a interrelación sexa obrigatoria para todas as entidades interrelacionadas, e dicir, ambas entidades presentan cardinalidades (1,1), poderase propagar a clave de calquera delas á relación resultante da outra; tendo en conta neste caso que é mellor que reciba a clave aquela relación que vaia ser accedida máis frecuentemente. Outra posibilidade é priorizar a eficiencia propagando a clave en ambos sentidos. Esta solución produce redundancia a controlar por restricións adicionais, polo que as normas de deseño non aconsellan esta opción.

## MER




Todos os contadores están instalados nalgún piso, pero pode haber pisos que non teñan contador

## Intensión MR

**PISO** ( *codPiso*, domicilio, localidade, codigoPostal)

B:C, M:C

**CONTADOR** ( *codContador*, modelo, *codPiso*)

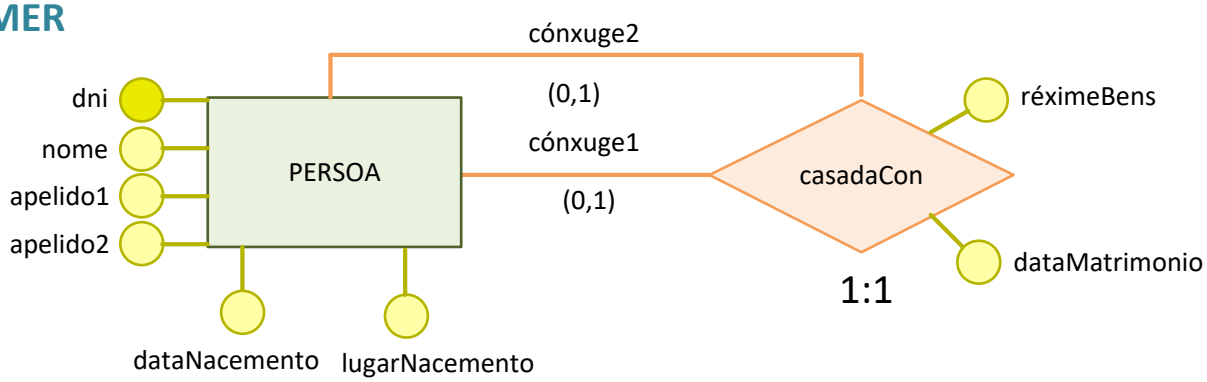
Figura 1.35. Transformación de interrelación 1:1 supoñendo un maior acceso aos datos contidos na entidade CONTADOR

### Monarias ou reflexivas con tipo de correspondencia 1:1

Con este tipo de interrelación actuarase preferentemente creando unha táboa, preferencia que se converterá en obrigatoriedade cando a relación monaria sexa portadora de atributos.

Aínda que é posible a propagación de clave sobre a mesma táboa, debe recordarse que esta solución non permite os borrados en cascada.

## MER




## Intensión MR

**PERSOA** (*dni*, nome, apelido1, apelido2, dataNacemento, lugarNacemento)

B:R, M:C B:R, M:C

**MATRIMONIO** (*dniConxuge1*, *dniConxuge2*, dataMatrimonio, reximeBens)

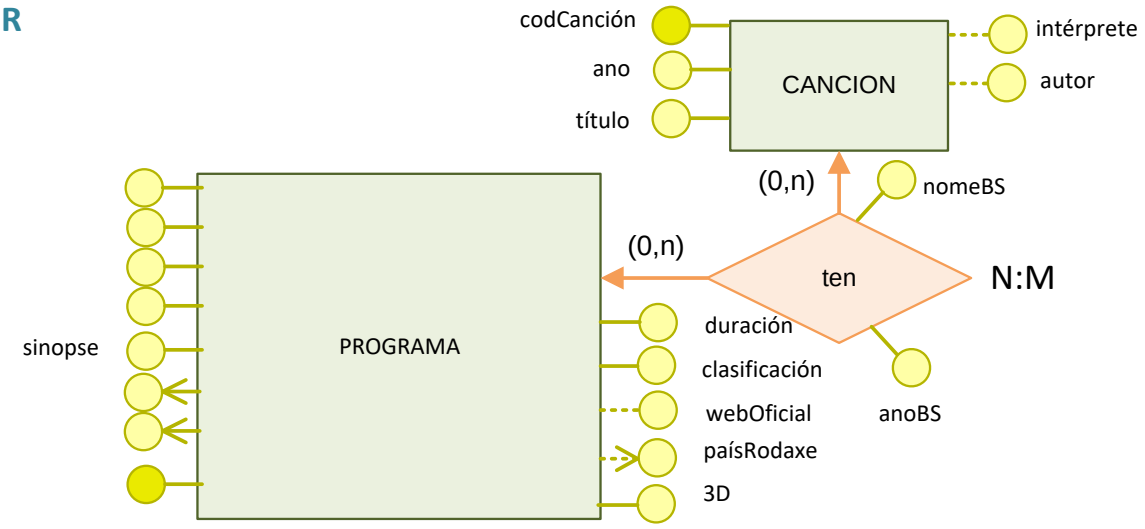
Figura 1.36. Transformación de interrelación reflexiva con tipo de correspondencia 1:1

Tarefa 9: Transformar interrelacións 1:1

### Regra 6.4.Transformación de atributos de interrelacións

Nos apartados anteriores viuse que se unha interrelación se transforma nunha relación, todos os seus atributos pasan a ser atributos desta nova relación, mentres que cando se representa mediante a propagación de clave, os seus atributos migrarán xunto coa clave á relación que recibe a propagación desta.

**MER**




**Intensión MR**

**CANCION** (codCancion, titulo, ano, autor*, interprete*)

B:C, M:C

**BANDASONORA** (CodBandaSonara, nomeBS, anoBS, codPrograma, codCancion)

B:C, M:C

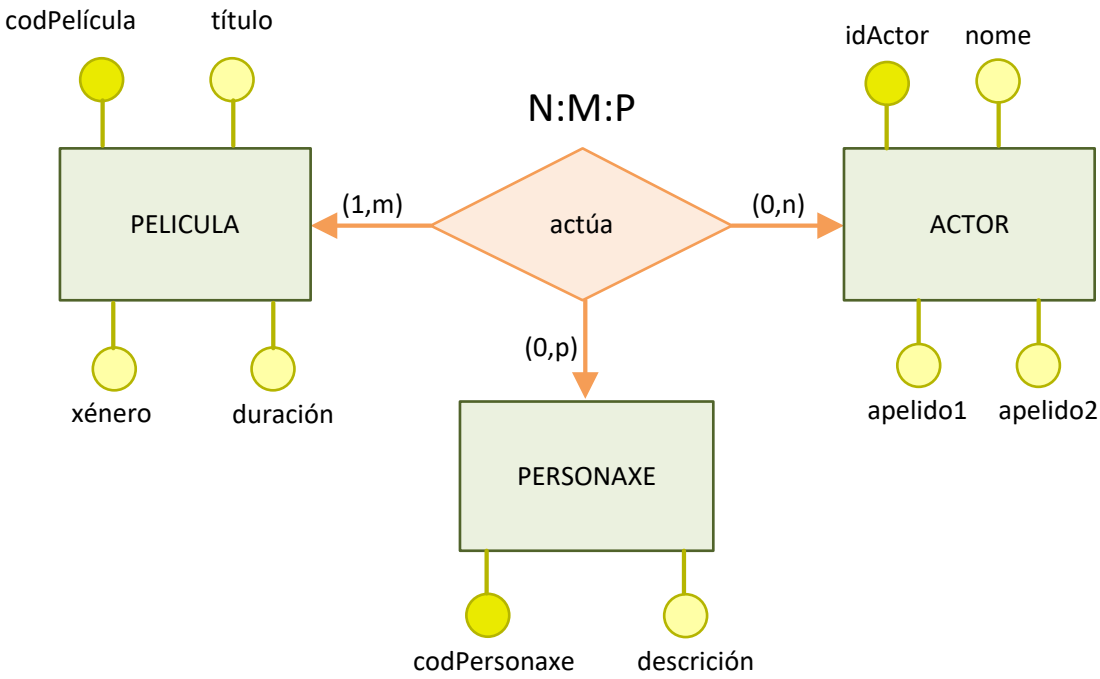
**PROGRAMA** (*codPrograma* , idiomaVisionado, idiomaSubtitulos, tituloDistribucion, tituloOrixinal, dataEstreo, idomaOrixinal, sinopse, duracion, clasificacion, webOficial*, paisRodaxe, 3D)

Figura 1.37. Exemplo de transformación dunha Interrelación que xera unha nova táboa

### Regra 6.5. Transformación de interrelación n_area (grado maior que dous)

As interrelacións ternarias, cuaternarias, etc. Represéntanse do mesmo modo que as interrelacións N:M, é dicir, xerando unha nove relación que terá sendas claves alleas apuntando a cada unha das claves correspondentes ás entidades tipo que asocia.

## MER




## Intensión MR: solución 1

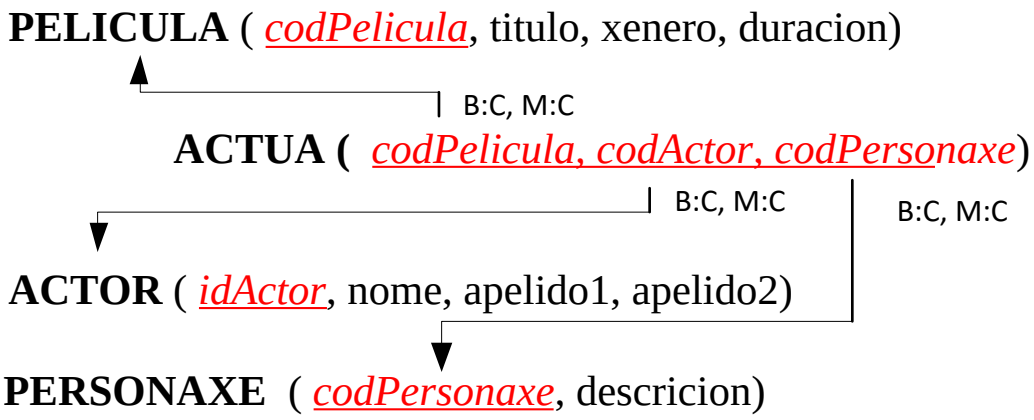



Figura 1.38. Exemplo de transformación dunha interrelación ternaria

No tocante a clave principal da relación xurdida, o máis común é que esté formada pola concatenación das claves tipo asociadas.

EXEMPLOS E SITUACIÓNS

Por motivos de eficiencia na xestión interna das claves, optase por crear un atributo clave principal específico ao que se asociará un tipo autoincremental que manexa automaticamente aos SXBD.

## Intensión MR: solución 2

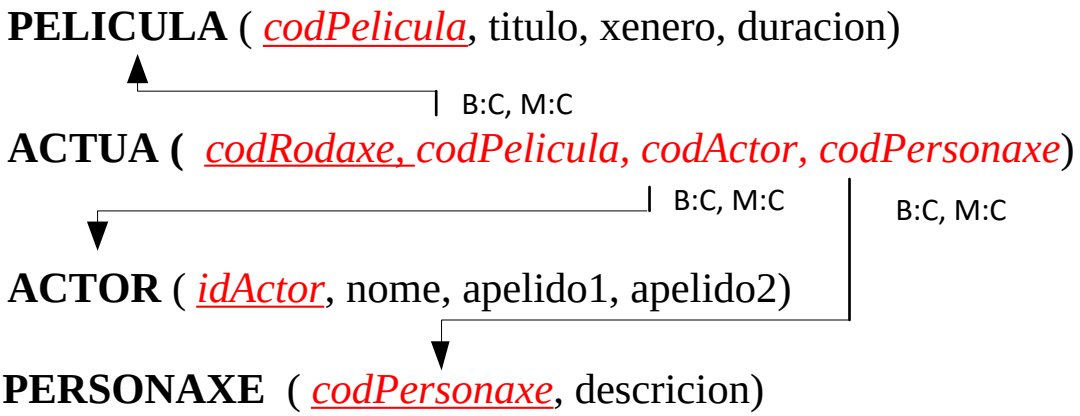



Figura 1.39. Transformación de interrelación ternaria para a que se crea a clave "codRodaxe" na relación xerada

Ata o momento explicouse como controlar certos casos de cardinalidade, pero para o total control das cardinalidades mínimas e máximas de cada entidade participante; tendo en conta que as posibilidades increméntanse sustancialmente en comparación coas binarias, deberá recurrirse á implementación de restricións empregando validacións, disparadores, aser-cións…

Tarefa 10: Transformar interrelacións n_arias

### Regra 7. Transformación de restricións

A representación das restricións, tanto as impostas no modelo ( integridade referencial, etc) como as semánticas derivadas dos requisitos de usuario ( valores positivo para idade, etc) , pódense definir mediante as seguintes cláusulas:

- Restricións de dominio: CREATE DOMAIN, NOT NULL, DEFAULT…
- Restricións de integridade referencial: FOREIGN KEY e cláusulas de comportamento asociado ( ON DELETE CASCACDE, RETRICT…).
- Restricións de comprobación: CHECK (comprobación de condicións de atributos da mesma táboa), BETWEEN (rangos de valores de dominios), IN (enumeración de valores de atributos).
- Asercións: CREATE ASSERTION (permite comprobar condicións entre atributos de di-ferentes táboas).
- Outras restricións :
  - Disparadores: CREATE TRIGGER.
  - Procedementos:CREATE PROCEDURE.

### 1.2.6.3 Transformación do Modelo Conceptual MERE ao Modelo Lóxico Relacional de Codd.

### Regra 8. Transformación de Entidades débiles

No Modelo Lóxico de datos Relacional non existe unha sentenza simple que permita indicar as dependencias en existencia ou en identificación. Poren, posto que se trata dun tipo de

correspondencia 1:N que presenta determinadas restricións de cardinalidade, pódese empregar o mecanismo de propagación de clave, creando unha clave allea con NULOS non permitidos, na relación correspondente á entidade dependente.

Ademais, será preciso establecer un comportamento no caso de modificacións que obrigue a unha modificación e borrado en cascada.

No caso de que a dependencia sexa en identificación, a clave primaria da relación da entidade débil debe estar formada pola concatenación das claves propagadas das entidades participantes na interrelación máis un atributo discriminante.

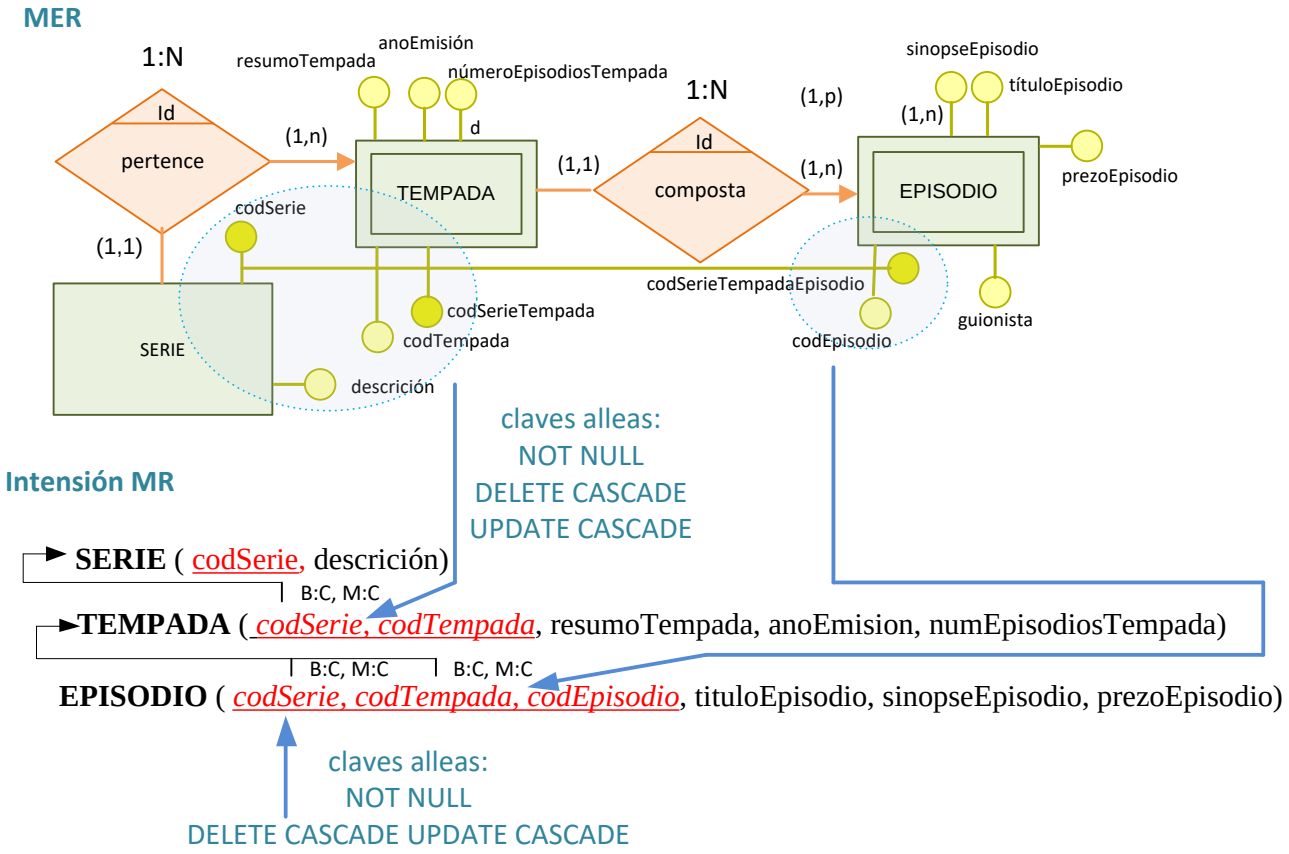



Figura 1.39. Exemplo de transformación de entidades débiles con dependencia en identificación

Tarefa 11: Transformar entidades débiles

### Regra 9. Transformación de xerarquías de tipos e subtipos (xeneralización ou especialización)

A definición de supertipos e subtipos emprega o mesmo concepto herdanza que a metodoloxía orientada a obxectos, polo que tendo mecanismos para representalos no modelo relacional a súa semántica e máis precisa nos motores de datos obxecto relacionais

As estratexias de transformación ao modelo relacional nun esquema conceptual formado por unha entidade supertipo e varias subtipos, directamente dependentes da primeira, están suxeitas a consideracións como as perdas semánticas, os tipos de xerarquías e cuestións de rendemento.

Destacamos as seguintes estratexias a seguir:

### Englobar todos os atributos da entidade supertipo e os seus subtipos nunha soa relación

Esta solución é recomendable unicamente cando se dan as seguintes condicións:

- Os subtipos diferéncianse entre si en moi poucos atributos.
- As interrelacións que asocian os distintos subtipos co resto das entidades se poden unificar para todos os subtipos.
- **A xerarquía é non solapada (exclusiva**), é dicir, unha ocorrencia de supertipo non está nunca acompañada pola ocorrencia de máis dun subtipo.
- **A xerarquía é total ou case total,** é dicir,unha ocorrencia de supertipo está acompañada na maioría dos casos de ao menos unha ocorrencia de subtipo.

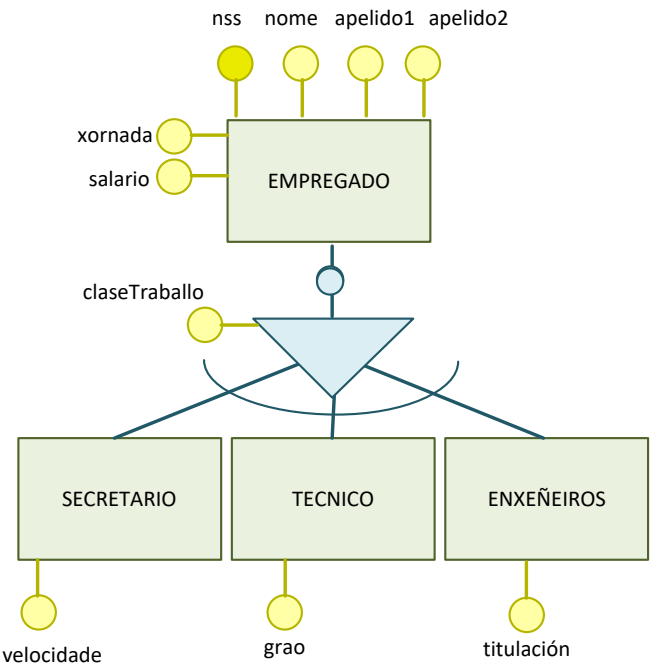
  No caso de seleccionar esta transformación terase en conta o seguinte:

- **Cando se trate dunha xerarquía parcial:** o atributo discriminante da xerarquía (o que indica cal é o subtipo dunha ocorrencia dada) deberá admitir NULOS xa que ter este atributo a nulo indica que a ocorrencia non pertence a ningunha das subclases. Así mesmo, o valor concreto do atributo discriminante identificará a que subclase ou subclase pertence unha ocorrencia dada.
- **Cando se trate dunha xerarquía total**: o atributo discriminante da xerarquía (o que indica cal é o subtipo dunha ocorrencia dada) NON poderá admitir NULOS, xa que o seu valor identificará ao subtipo concreto ao que pertence a ocorrencia.
- Os atributos pertencentes ás subclases deben permitir a súa posta a nulos.

Se existen restricións adicionais, como condicións entre os valores dos diferentes atributos, etc. implantaranse utilizando as sentenzas e cláusula de SQL que permiten a instrumentación de restricións (verificacións ou check, disparadores ou triggers, asertos, etc.)

No que se refire á eficiencia, esta opción é a que ofrece máis velocidade no acceso aos datos dun obxecto, xa que non é preciso efectuar unións de táboas para a recuperación da información.

**MER**




**Intensión MR**

**EMPREGADO** ( *nss*, nome, apelido1, apelido2, xornada, salario,

claseTraballo, velocidade*, grao*, titulacion*)

Figura 1.40. Transformación dunha dependencia total, exclusiva mediante a creación dunha soa relación

Tamén se contempla a posibilidade de non incluír o atributo discriminante na relación, e deducir ó subtipo ou subtipos o que unha a ocorrencia pertence a partir da existencia ou non dos valores dos atributos pertencentes a ditas entidades subtipo. Aínda que esta é unha solución pouco aconsellada xa que reduce o número de atributos da relación, aumenta as operacións necesarias para obter a que subtipo ou subtipos pertence unha ocorrencia.

### Crear unha relación para o supertipo e unha que englobe todos os subtipos

Nesta estratexia, crearase unha única relación que englobará os atributos de todas as entidades subtipos que aceptarán valores nulos, e cuxa clave principal será a mesma clave principal da entidade supertipo, polo que actuaría tamén como clave allea. A inclusión do atributo discriminante non é obrigatorio, xa que a pertenza a unha das entidades subtipo coñécese pola existencia de valores nos seus atributos, pero si que é aconsellable para reducir o custo en futuros accesos.

Esta opción é recomendable unicamente cando se cumpren as seguintes condicións:

- Os subtipos diferéncianse entre si en moi poucos atributos.
- As interreleacións que asocian os distintos subtipos co resto das entidades se poden unificar para todos os subtipos.
- **A xerarquía é exclusiva (non solapada**), é dicir, unha ocorrencia do supertipo non está nunca acompañada pola ocorrencia de máis dun subtipo.

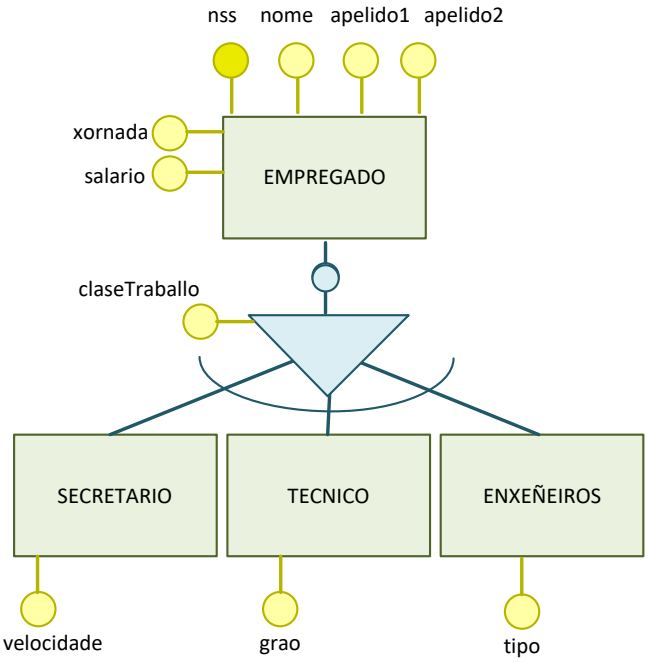
No caso das **xerarquías parciais** (non totais, isto é, existen ocorrencias do supertipo que non pertencen a ningún subtipo), será recomendable separar polo menos en dúas relacións a supertipo e o conxunto das subtipo, pois non facelo deste xeito implicaría a presenza de

todos os atributos das subtipo con nulos nos casos de ocorrencia da supertipo sen ocorrencia de ningunha subtipo.

En calquera caso, a solución das dúas relacións non incorpora aumentos significativos de eficiencia respecto ás solucións que se presepresentan a continuación, xa que para acceder a todos os datos dun obxecto vai a requirir sempre a consulta das dúas relacións.

Dende o punto de vista do mantemento da semántica orixinal tampouco se trata dunha solución demasiado correcta, desaconsellando o seu uso.

**MER**




**Intensión MR**

**EMPREGADO** ( *nss*, nome, apelido1, apelido2, xornada, salario)

B:C, M:C

**TIPOEMPREGADO** ( *nss*, claseTraballo, velocidade*, grao*, salario*)

Figura 1.41. Transformación de xerarquía total exclusiva xerando dúas relacións

Cando non se cumpra a terceira condición existindo unha xerarquía solapada (unha ocorrencia dunha supertipo pode corresponder a unha ocorrencia de máis dun subtipo) pódese aplicar tamén esta opción, aínda que incrementa moito o custe en programación, sendo unha opción desaconsellada. Ademais, no caso de que a xerarquía sexa solapada o rendemento vai ser o menor de todos

### Crear unha relación para o supertipo e unha para cada subtipo

É a solución máis adecuada en xeral, e a máis flexible, a que permite recoller máis semántica sendo tamén a máis respetuosa co modelo conceptual orixinal. É preciso crear as restricións oportunas para que se reflicta o máis fidedignamente a semántica orixinal do modelo conceptual de datos.

No que se refire á implementación, esta opción ofrece menos velocidade que a primeira no acceso aos datos dunha relación, posto que para a recuperación da información é preciso

unir varias relacións. Así mesmo, é a que máis espazo ocupa, xa que ter atributos nunha única relación sempre ocupa menos que telos repartidos entre n táboas diferentes. Sen embargo, como xa se viu, nos casos de xerarquías solapadas funciona mellor que a opción de só dúas relacións.

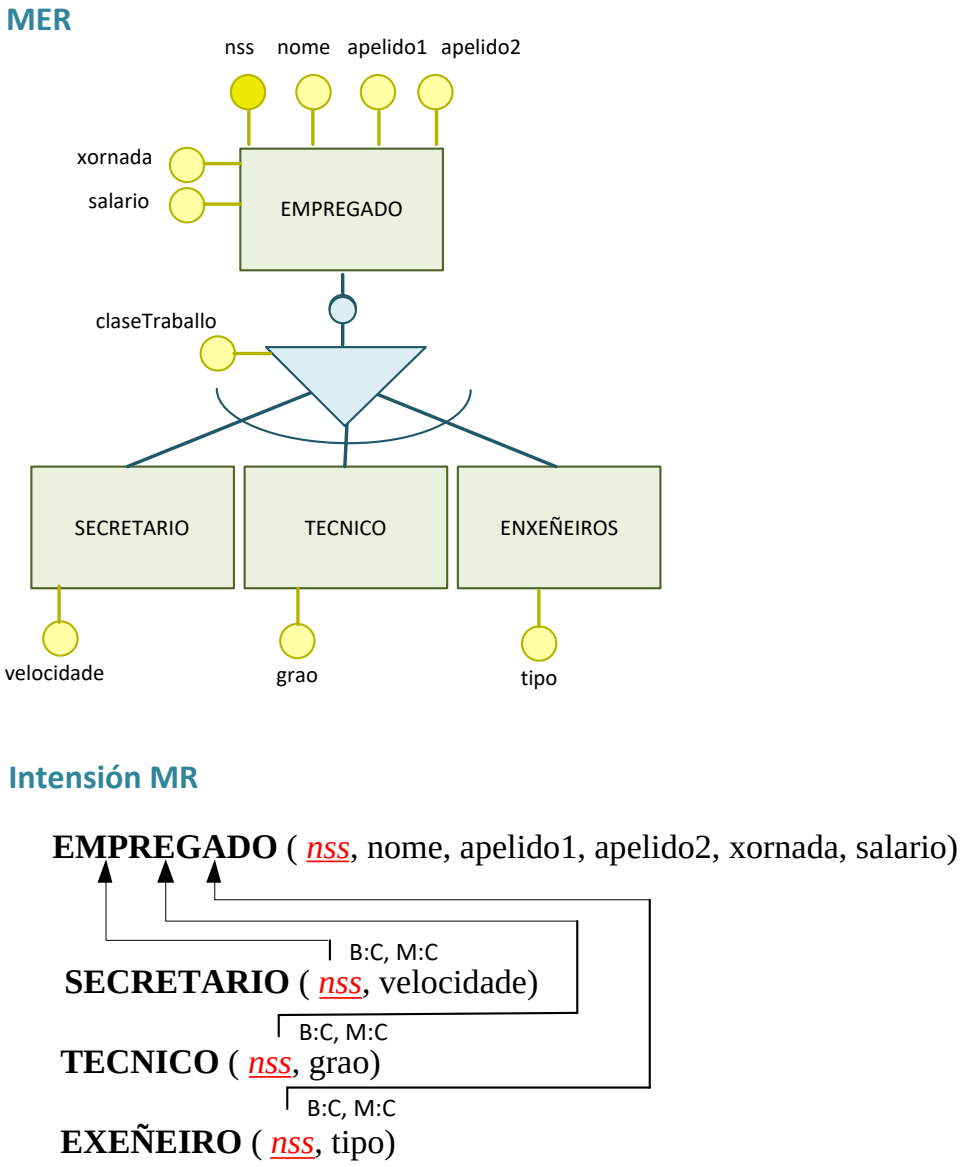



Figura 1.42. Transformación dunha xerarquía total exclusiva creando unha táboa por cada subtipo

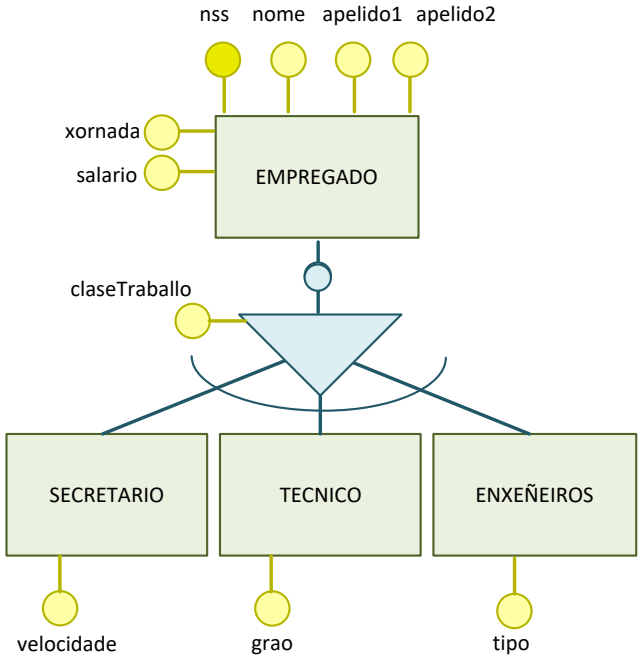
### Crear tantas relacións como subtipos e non considerar o supertipo

Nesta opción crearase unha relación por cada subtipo existente que conterá, ademais dos atributos propios, os atributos comúns do supertipo. Optarase por esta estratexia cando se cumpren as seguintes condicións:

- Os subtipos dispoñen dun elevado número de atributos, e/ou interrelacións propias
- Os accesos realizados aos datos dos subtipos afectan maioritariamente aos atributos comúns.
- Para unha xerarquía exclusiva disxunta, xa que de ser solapada os atributos comúns estarían a ser repetidos tendo que controlar esta redundancia para evitar inconsistencias.

- Para unha xerarquía con participación total, xa que de ser parcial xeraranse unha gran cantidade de NULOS (os atributos comúns)

**MER**




**Intensión MR**

**SECRETARIO** ( *nss*, nome, apelido1, apelido2, xornada, salario, velocidade)

**TECNICO** ( *nss*, nome, apelido1, apelido2, xornada, salario, grao)

**EXEÑEIRO** ( *nss*, nome, apelido1, apelido2, xornada, salario, tipo)

Figura 1.43. Transformación dunha xerarquía total exclusiva xerando só relacións para os subtipos.

Esta solución é a que produce unha maior perda de semántica, a pesar que aumenta a eficiencia nas consultas que afectan a todos os atributos (tanto comúns como propios dun subtipo) diminuíndose noutras.

Tarefa 12: Transformar especializacións

### Regra 10. Transformación de restricións de interrelacións

A transformación de restricións de interrelacións utilizará os mesmos mecanismos que a transformación de restricións que afectan as entidades e os atributos, isto é, empregar as condicións de validación ou CHECK, asercións ou ASSERTION, disparadores ou TRIGGERS e módulos programados.

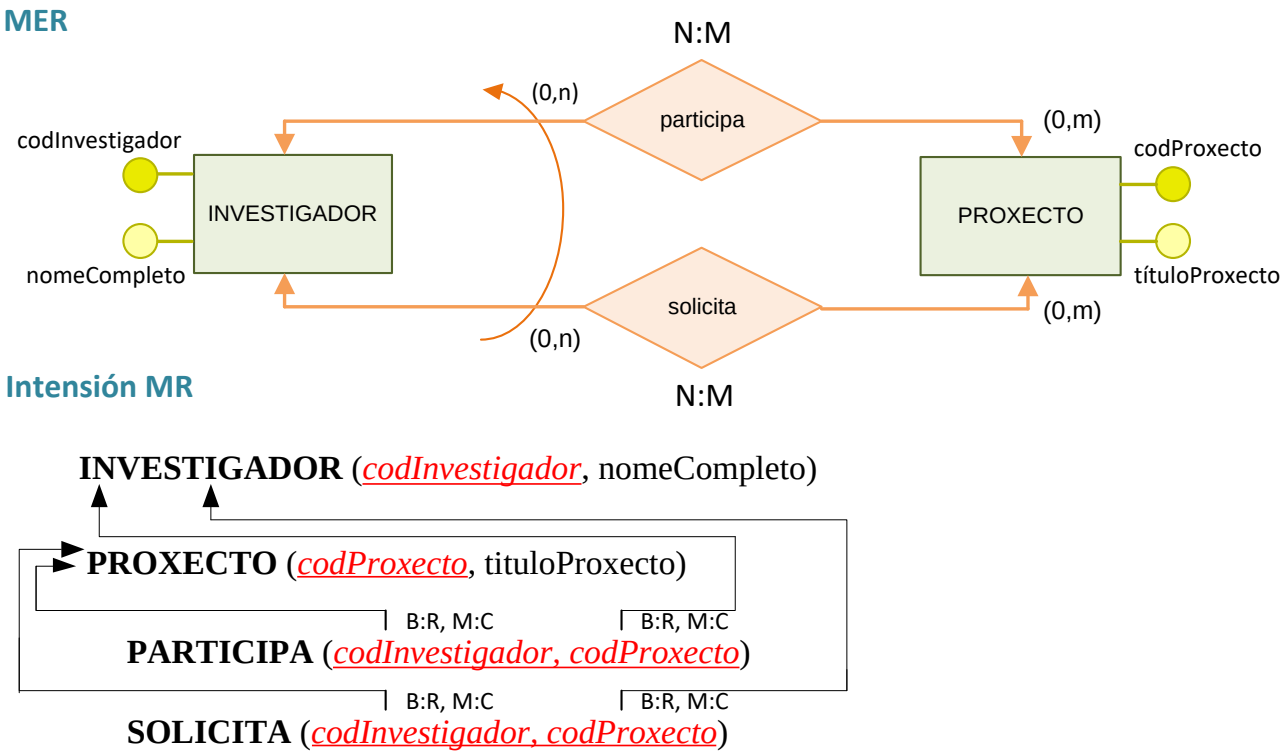



Figura 1.44. Exemplo de transformación dunha restrición de inclusión

No exemplo anterior móstrase a transformación dunha restrición de inclusividade onde para que un investigador solicite un proxecto debeu participar con anterioridade noutro (que non ten porque ser no que solicitou). Unha posible implementación desta restrición amósase a continuación:

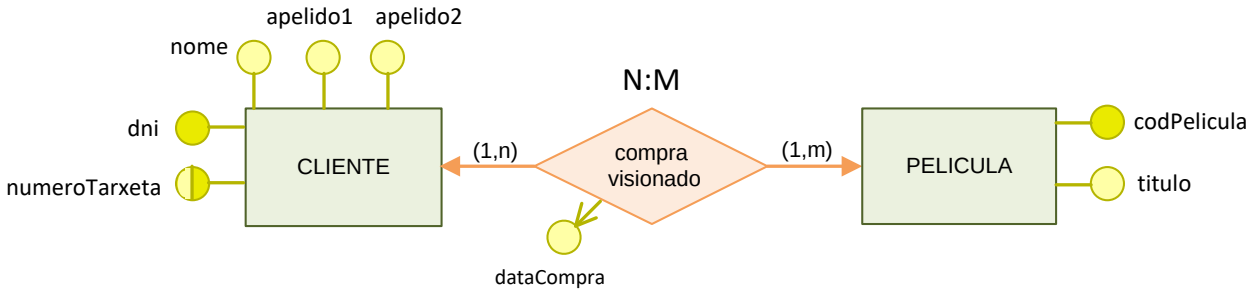
```
CREATE TABLE SOLICITA (
codInvestigador Codigos,
codProxecto Codigos,
PRIMARY KEY (codProxecto, codInvestigador)
FOREIGN KEY (codProxecto) REFERENCES PROXECTO
ON UPDATE CASCADE
FOREIGN KEY (codInvestigador) REFERENCES INVESTIGADOR
ON UPDATE CASCADE
CHECK (( codProxecto IN (SELECT codProxecto FROMPARTICIPA))
```

### Regra 11 Transformación da dimensión temporal

Comunmente no deseño a dimensión temporal recóllese mediante a inclusión de atributos de tempo (data, hora,…), que axudan a modelar entidades con variación histórica distinguíndose dúas posibilidades:

- Se a dimensión temporal aparece como una entidade, transformarase como tal.
- Se a dimensión temporal aparece de forma de atributos nunha interrelación, estes atributos ao ser multivaluados en moitos casos, ubicaranse na táboa que corresponde ao transformarse a interrelación á que pertencen (ben na nova táboa que se crea ou unha táboa onde se propaga a clave). Agora ben, debe considerarse que estes atributos de tipo data poden ter que formar parte da clave primaria da táboa na que se sitúen en función de semántica da situación que se representa.

**MER**




**Intensión MR**

**CLIENTE** ( *dni*, **numerotarxeta**, nome, apelido1, apelido2)

B:C, M:C

**COMPRAVISIONADO** ( *dni, codPelicula*, dataCompra)

B:C, M:C

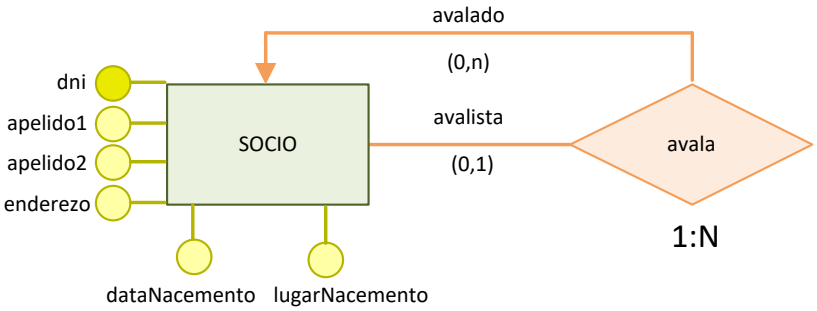
**EMPREGADO** ( *codPelicula*, título)

Figura 1.45. Exemplo de posible transformación de dimensión temporal.

Para poder almacenar feitos que transcorren nun intervalo de tempo determinado, ne-cesitaremos unha data de inicio e outra de fin. Nas bases de datos históricas, como nas que as ocorrencias asociadas pola interrelación pódense repetir no tempo, o atributo data será multivaluado.

No seguinte exemplo pódese coñecer que usuario avala ou é avalado por outro, pero só para o momento actual:

**MER**




**Intensión MR**

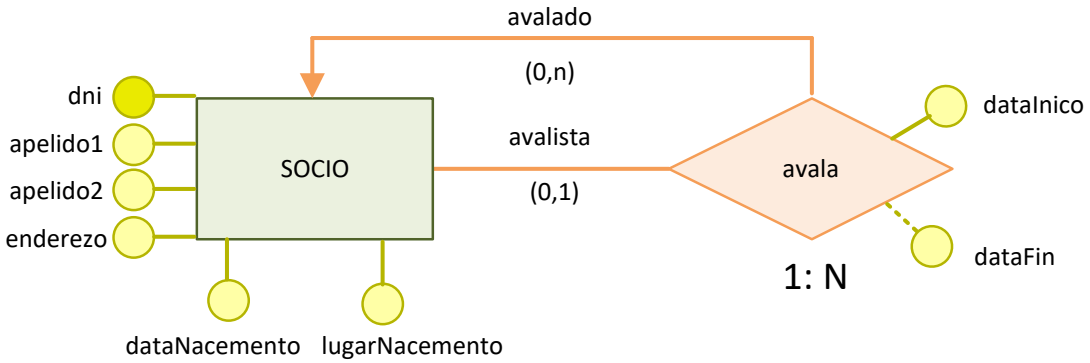
**SOCIO** (*dni*, apelido1, apelido2, enderezo, dataNacemento, lugarNacemento, *dniAvalado*)

B:R, M:C

Figura 1.46. Exemplo de posible transformación de dimensión temporal.

Para incluír durante que período de tempo un socio avala a outro, precisaranse dúas datas, unha do inicio do aval e outra do final.

## MER




## Intensión MR

**SOCIO** (*dni*, apelido1, apelido2, enderezo, dataNacemento, lugarNacemento,

datInicio, dataFin*, *dniAvalado*)

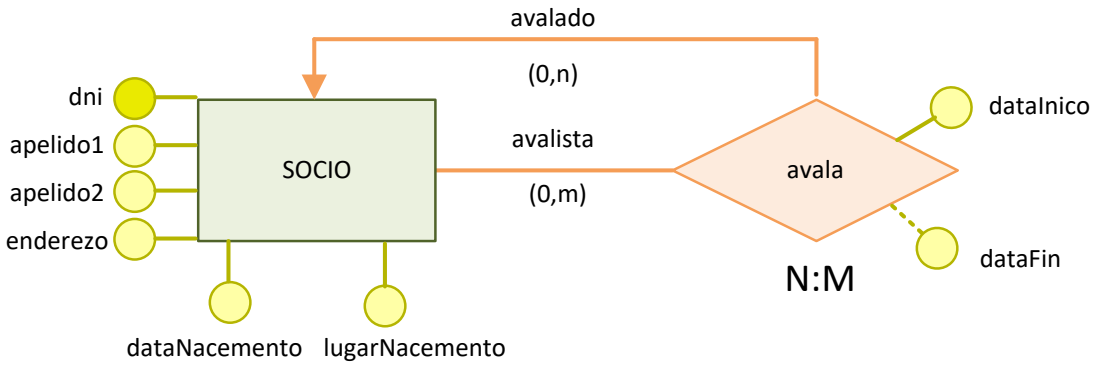
B:R, M:C

Figura 1.47. Exemplo de posible transformación de dimensión temporal.

O período no que un socio avala a outro e coñecido mediante os atributos datas, pero con este esquema un socio só podería ter un avalista ou cando un socio cambia de avalista débese borrar a tupla actual para introducir a nova información perdendo que socios foron avalistas de outros con anterioridade (isto é debido o feito de que a relación é 1:N e as datas son univaluadas).

Unha solución parcial para recoller que un avalista poda avalar a diferentes persoas en períodos iguais ou diferentes de tempo ( sen perder a quen avalaba con anterioridade) é transformar a relación 1:N en M:N

## MER




## Intensión MR

**SOCIO** (*dni*, apelido1, apelido2, enderezo, dataNacemento, lugarNacemento)

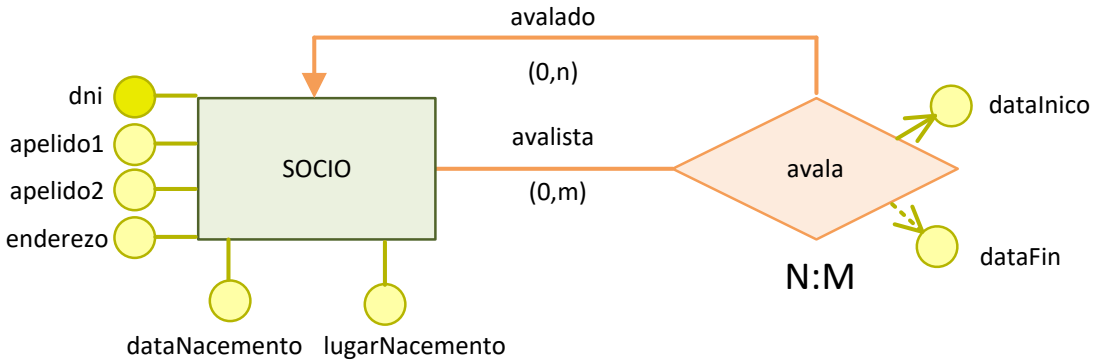
B:R, M:C B:R, M:C

**AVALA** (*dniaAvalado, dniAvalista*, dataInicio, dataFin*)

Figura 1.48. Exemplo de posible transformación de dimensión temporal.

Con este deseño unha persoa non poderá avalar a mesma persoa en períodos diferentes de tempo, xa que as tuplas que representasen este feito terían a mesma clave. Para permitir recoller este feito e dispor así dun histórico de avalados e avalistas, as datas de inicio e fin deben ser multivaluadas.

**MER**




**Intensión MR**

**SOCIO** (*dni*, apelido1, apelido2, enderezo, dataNacemento, lugarNacemento)

B:R, M:C B:R, M:C

**AVALA** (*dniAvalado, dniAvalista*, dataInicio, dataFin*)

Figura 1.49. Exemplo de posible transformación de dimensión temporal.

Ao realizar a transformación do modelo conceptual ao lóxico teranse que estudar os atributos temporais para a súa inclusión na clave; "dataFin" non pode formar parte da mesma xa que é opcional e permite nulos (aquel avalista actual dun socio non dispón dataFin). En canto a "dataInicio" inclúese como clave xa que é a que se encarga de discriminar entre os distintos períodos de tempo nos que un mesmo avalista avala a unha mesma persoa.

Tarefa 13: Transformar dimensión temporal.

Tarefa 14: Transformar MERE a MER

Tarefa 15: Transformar o esquema conceptual a esquema relacional

## 1.3 Tarefas

As tarefas propostas son as seguintes:

- Tarefa 1. Identificar tipos de restricións.
- Tarefa2. Implementar integridade referencial.
- Tarefa 3. Establecer restricións de negocio.
- Tarefa 4. Diferenciar os elementos e fases do deseño lóxico.
- Tarefa 5: Crear dominios.

- Tarefa 6. Transformar entidades.
- Tarefa 7. Transformar interrelacións N:M.
- Tarefa 8. Transformar interrelacións 1:N.
- Tarefa 9 Transformar  interrelacións 1:1.
- Tarefa 10 Transformar interrelacións n_areas.
- Tarefa 11.Transformar entidades débiles.
- Tarefa 12. Transformar especializacións.
- Tarefa 13. Transformar dimensión temporal.
- Tarefa 14. Transformar un MERE a MR.
- Tarefa 15 Transformar o esquema conceptual a esquema relacional

### 1.3.1 Tarefa 1. Identificar tipos de restricións

Explique se a seguinte afirmación é correcta e xustifique a resposta:

- Tarefa 2.1: Se permitimos que parte da clave primaria sexa NULO estamos asegurando que a clave é mínima xa que existen atributos que non conteñen ningún valor"
- Tarefa 2.2: A restrición de integridade referencial obriga a que todo valor da clave primaria (PK) sexa referenciado por algún valor da clave allea (FK)
- Tarefa 2.3. Todo relación ten polo menos unha clave candidata
- Tarefa 2.3 Toda clave primaria é tamén clave candidata
- Tarefa 3.4. Toda clave alternativa é tamén clave candidata.
- Tarefa 2.5. Un valor nulo no modelo relacional é un sinal que permite representar información privada que non se quere dar a coñecer ao usuario.

#### Solución

##### Tarefa 1.1.

Incorrecta. Se permitimos que parte da clave primaria sexa NULO estase afirmando que non todos os seus atributos son necesarios ( xa podemos prescindir da información de parte deles ao descoñecerse para distinguir a tupla) có que podería reducirse, non sendo mínima, contradicindo a irredutibilidade

##### Tarefa 1.2.

Incorrecta, xa que non todas as tuplas dunha relación teñen por que ser referenciadas por outra. A afirmación correcta sería todo valor de FK debe existir en PK, xa que para que unha clave referencie a outra esta debe existir con anterioridade

##### Tarefa 1.3

Correcta, toda relación ten polo menos unha clave candidata, de só existir unha, esa será a clave principal

Correcta. Toda clave primaria e tamén clave candidata, xa que a clave primaria selecciónase do conxunto de claves candidatas

### Tarefa 1.4

Correcta. Toda clave alternativa é tamén clave candidata, xa que se escolle de dentro do conxunto de claves candidatas.

### Tarefa 1.5.

Incorrecto. Un valor nulo no modelo relacional é un sinal que permite representar información descoñecida, inaplicable, inexistente, indefinida, non válida, etc

## 1.3.2 Tarefa2. Implementar a integridade referencial

- Tarefa 2.1. Dadas as seguintes relacións expresadas en intensión, indique de entre as opcións propostas que tipo de restrición ou restricións deben introducirse para asegurar que sempre que se introduza un novo vehículo o seu dono deba existir previamente e que cando se borre unha persoa non se borren todos os seus vehículos.
  - Relacións:
    - PERSOA (dni, apelido1, apelido2, nome, rúa, número, piso, porta, telefono)
    - VEHÍCULO ( matrícula, marca, modelo , dni_dono)
  - Opcións:
    - A) Na relación VEHÍCULO definir como clave allea o atributo dni_dono e como opción de borrado especificar RESTRIC ou restrinxido
    - B) Na relación VEHÍCULO definir unha aserción de clave allea sobre o atributo dni_dono que especifique como opción de borrado RESTRICT ou restrinxido
    - C) Na relación PERSOAS definir como clave allea o atributo dni e como opción de borrado especificar RESTRICT ou restrinxido
    - D) Na relación VEHÍCULO definir como clave allea o atributo dni_dono e como opción de borrado especificar CASCADE ou cascada
- Tarefa 2.2. Ao transformar un esquema desde o modelo E/R ao relacional, os atributos identificadores alternativos (indique a/ou opcións correcta):
  - A) Represéntanse coa cláusula PRIMARY KEY
  - B) Represéntanse como columnas e engádese unha cláusula CHECK
  - C) Represéntanse por medio da cláusula UNIQUE
  - D) Represéntanse por medio da cláusula FOREIGN KEY indicando que non se admite nulos
- Tarefa 2.3. Nunha base de datos relacional, a inserción dunha tupla nunha táboa (indique a/ou opcións correcta):
  - A) Necesita sempre unha validación da integridade de clave primaria e non sempre da integridade referencial.
  - B) Necesita sempre unha validación da integridade referencial e non sempre da integridade de clave primaria.
  - C) Necesita sempre unha validación da integridade de clave primaria e da integridade referencial N
- Tarefa 2.4. A integridade referencial (indique a/ou opcións correcta):
  - A) Obriga a que toda referencia a outra táboa sexa consistente.
  - B) Obriga a que a clave primaria da táboa referenciada teña valor non nulo.

– C) Obriga a que a clave allea teña valor non nulo.
– D) É a restrición que garante o SXBD para cumprir coa non duplicidade de tuplas.

**Solución**

Tarefa 2.1.

A opción correcta é a A.

Tarefa 2.2

A opción correcta é a C

Tarefa 2.3

A opción correcta é a A

Tarefa 2.4

A opción correcta é a A

### 1.3.3 Tarefa 3. Establecer restricións de negocio

Explique se as seguintes afirmacións son correctas e xustifique as respostas:

- Tarefa 3.1: Se permitimos que parte da clave primaria sexa NULO estamos asegurando que a clave é mínima xa que existen atributos que non conteñen ningún valor"
- Tarefa 3.2: A restrición de integridade referencial obriga a que todo valor da clave primaria (PK) sexa referenciado por algún valor da clave allea (FK)

**Solución**

Tarefa 3.1.

Incorrecta. Se permitimos que parte da clave primaria sexa NULO estase afirmando que non todos os seus atributos son necesarios ( xa podemos prescindir da información de parte deles ao descoñecerse para distinguir a tupla) có que podería reducirse, non sendo mínima, contradicindo a irredutibilidade

Tarefa 3.2.

Incorrecta, xa que non todas as tuplas dunha relación teñen por que ser refrenciadas por outra. A afirmación correcta sería todo valor de FK debe existir en PK, xa que para que unha clave referencie a outra esta debe existir con anterioridade

### 1.3.4 Tarefa 4. Diferenciación dos elementos e fases do deseño lóxico

Explique se as seguintes afirmacións son correctas e xustifique as respostas:

- Tarefa 4.1: O Deseño Lóxico Estándar consiste en transformar un Esquema Lóxico Estándar nun Esquema Lóxico Específico
- Tarefa 4.2: O Modelo Lóxico Estándar é moi dependente do tipo SXBD concreto que se utilice

- Tarefa 4.3: Na etapa de Deseño Lóxico, se empregamos o Modelo Relacional, débense considerar as restricións de integridade referencial entre outras

**Solución**

Tarefa 4.1

Correcta. Unha das fases do  Deseño Lóxico consiste en transformar un Esquema Lóxico Estándar nun Esquema Lóxico Específico

Tarefa 4.2:

Correcta. O Modelo Lóxico Estándar é moi dependente do tipo SXBD concreto que se utilice xa que as relacións só teñen sentido cando empregamos un SXBD relacional. Noutros tipos de SXBD emprégase outras estruturas de almacenaxe de información coma os obxectos nos SXBD Orientados a Obxectos.

Tarefa 4.3:

Correcta. Na etapa de Deseño Lóxico, si utilizamos o Modelo Relacional, débense considerar as restricións de integridade referencial entre outras

### 1.3.5 Tarefa 5: Crear dominios

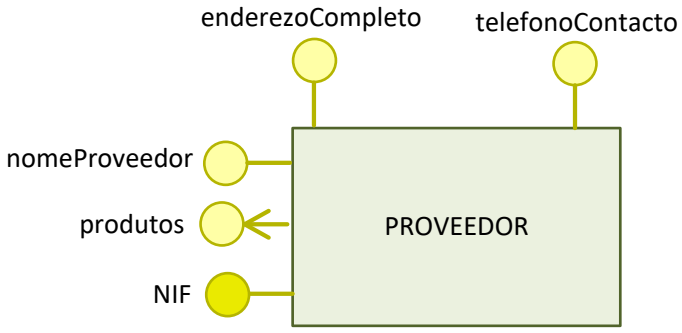
Cear o dominio das notas de avaliación duns estudos de FP, correspondente ao conxunto de valores de tipo numérico de lonxitude dous sen decimais do rango de 1 a 10. Busca información e emprega o operador BETWEEN

**Solución**

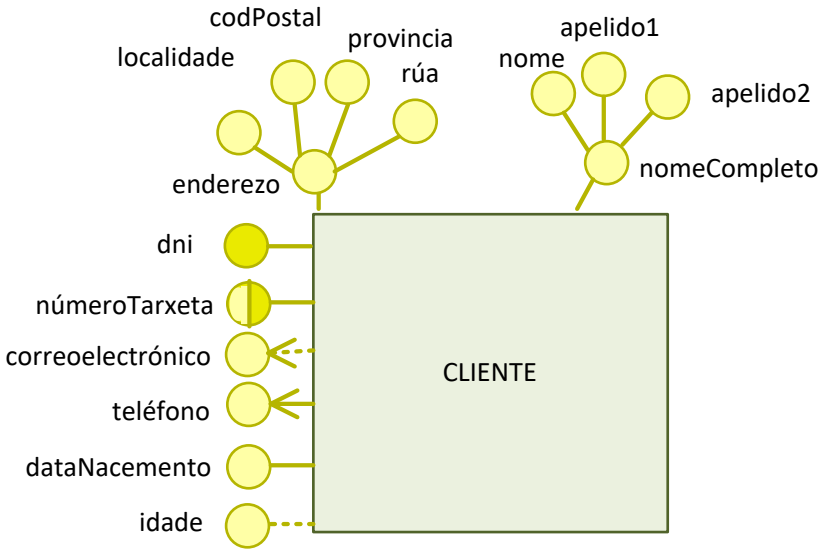
```
CREATE DOMAIN nota AS DECIMAL(2,0)
CHECK ( VALUE (BETWEEN 1 AND 10))
```

### 1.3.6 Tarefa 6. Transformar entidades:

- Tarefa 6.1. Transforma a relación a seguinte entidade:




- Tarefa 6.2: Transforma a relación a seguinte entidade:




**Solución**

Tarefa 6.1.

**PROVEEDOR** (descricionProduto, *nif*)

B:R, M:C

**EMPREGADO** (*NIF*, nomeProveedor, enderezoCompleto, telefonoContacto)

Tarefe 6.2.

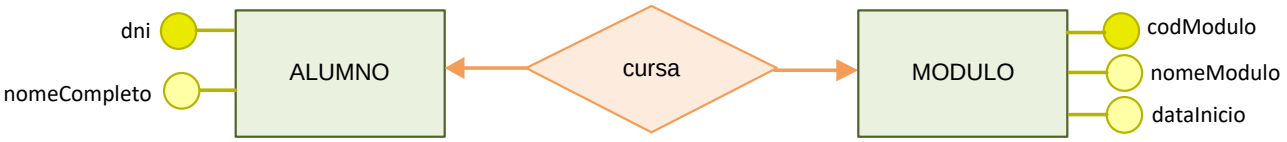
**TELEFONO** (telefono, *dni*) **CORREOELECTRONICO** (correo, *dni*)

B:C, M:C B:C, M:C

**CLIENTE** (*dni*, ***numeroTarxeta***, nome, apelido1, apelido2, codigoPostal, rua, provincia, localidade, dataNacemento, idade*)

### 1.3.7 Tarefa 7. Transformar interrelacións N:M

- Tarefa 7.1.: Unha interrelación N:M, no modelo relacional represéntase (indique a ou as opcións correcta):
  - A) Engadindo un novo esquema de relación que contén como clave primaria unha agrupación das claves primarias das entidades asociadas.
  - B) Propagando as claves primarias en cada unha das entidades asociadas
  - C) Engadindo un novo esquema de relación que contén como clave primaria a clave primaria dunha soa das entidades asociadas
  - D) Non hai unha regra xeral, depende de cada caso
- Tarefa 7.2.: Seleccione cal dos seguintes modelos relacionais son traducións correctas do modelo Entidade-Interrelación seguinte:




- Opcións:
  - A)

**ALUMNO** (*dni*, nomeCompleto, codModulo)

**MODULO** (*codModulo*, nomeModulo, dataInicio, dniAlumno)

  - B)

**ALUMNO** (*dni*, nomeCompleto, codModulo)

**MODULO** (*codModulo*, nomeModulo, dataInicio)

  - C)

**ALUMNO** (*dni*, nomeCompleto)

**CURSA**(*dni, codModulo*)

**MODULO** (*codCurso*, nomeModulo, dataInicio)

  - D)

**ALUMNO** (*dni*, nomeCompleto, codModulo)

**CURSA**(*dni, codModulo*)

**MODULO** (*codCurso*, nomeModulo, dataInicio)

**Solución**

Tarefa 7.1

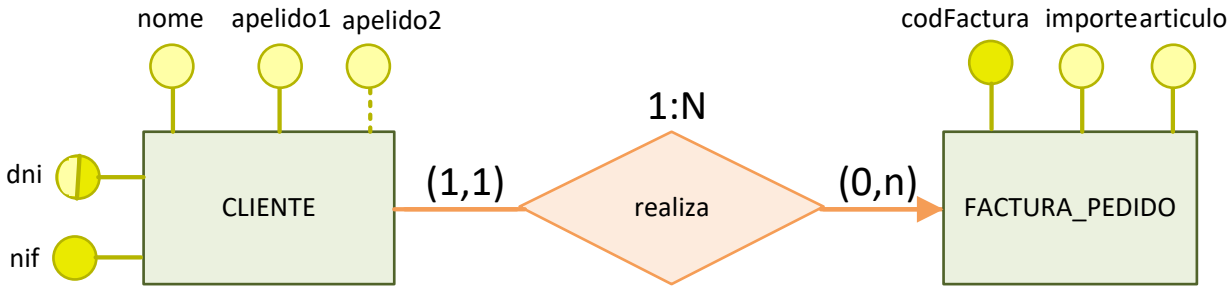
A opción correcta é a A

Tarefa 7.2.

A opción correcta é a C

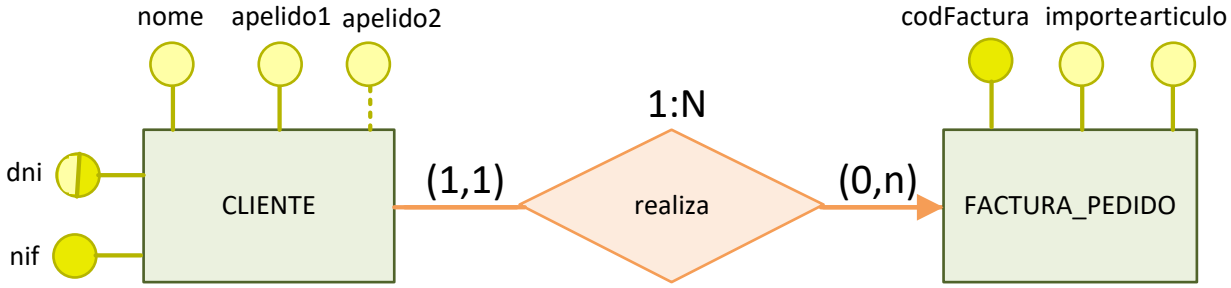
### 1.3.8 Tarefa 8. Transformar interrelacións 1:N

Transforma o seguinte modelo conceptual entidade-interrelación ao modelo lóxico relacional.




**Solución**

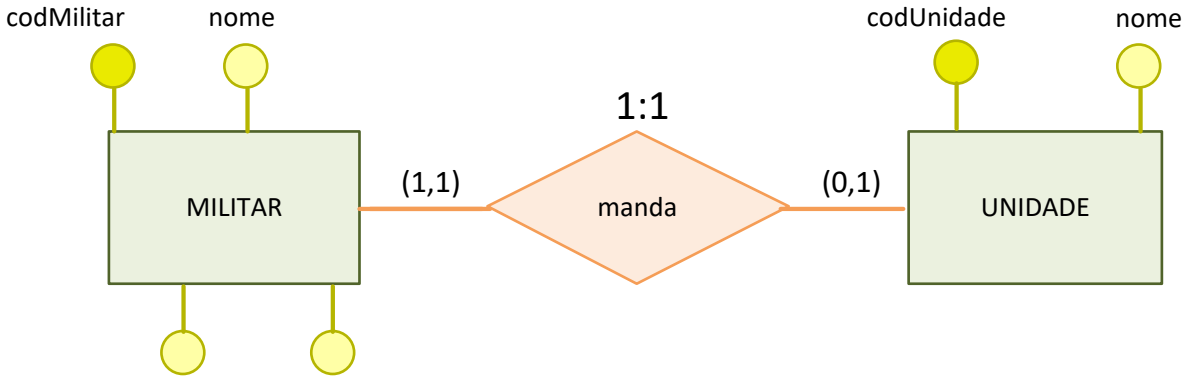
Débese puntualizar que a propagación de clave no deseño lóxico é sempre sobre a clave principal. Algúns SXBD tamén permiten a propagación empregando claves alternas coa propiedade UNIQUE (no exemplo  o atributo dni)



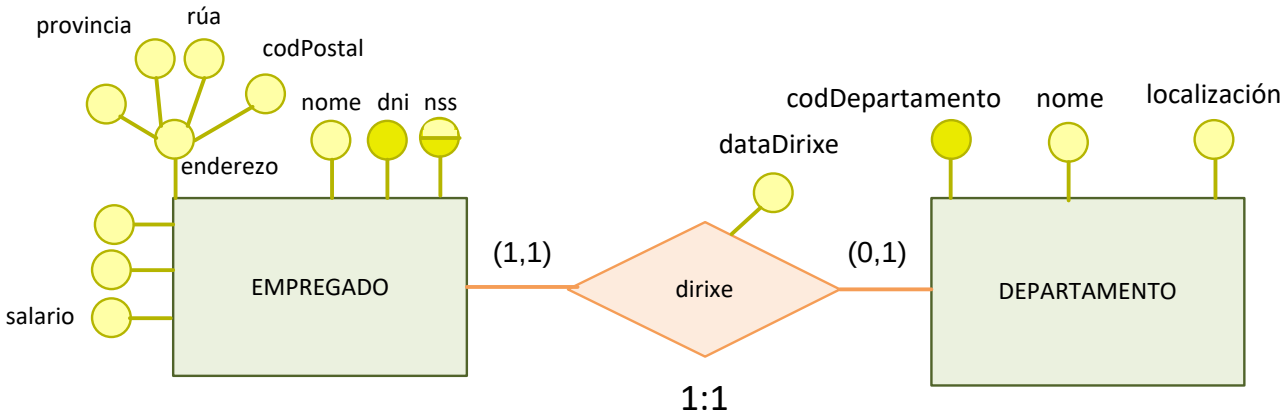

### 1.3.9 Tarefa 9 Transformar interrelacións 1:1

Transforma os seguintes modelos conceptuais entidade-interrelación aos modelos lóxico relacionais:

- Tarefa 9.1




- Tarefa 9.2




**Solución**

Tarefa 9.1

**MILITAR** ( *codMilitar*, nome, rango)

B:R, M:C

**UNIDADE** ( *codUnidade*, Nome, Sede, *codMilitar*)

Tarefa 9.2

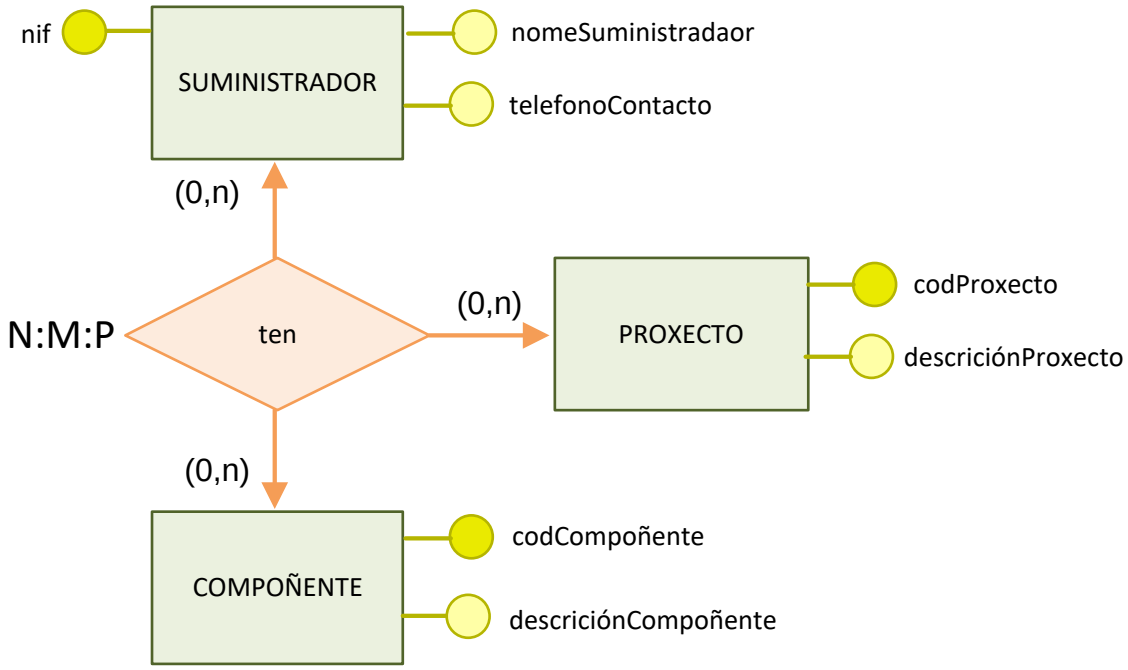
**DEPARTAMENTO** ( *codDepartamento*, Nome, localización, *codEmpregado*)

B:R, M:C

**EMPREGADO** ( *codEmpregado*, ***nss***, nome, provincia, locallidade, codPostal, rua, dataNacemento telefonoContacto, salario, localidade, codigoPostal)

### 1.3.10 Tarefa 10 Transformar interrelacións n_areas

Transforma o seguinte modelo conceptual entidade-interrelación ao modelo lóxico relacional:




**Solución**

**SUMINITRADOR** (*nif*, nomeSuministrador, telefonoContacto )

**COMPOÑENTE** (*codCompoñente*, dercriciónCompoñente)

B:R, M:C

B:R, M:C

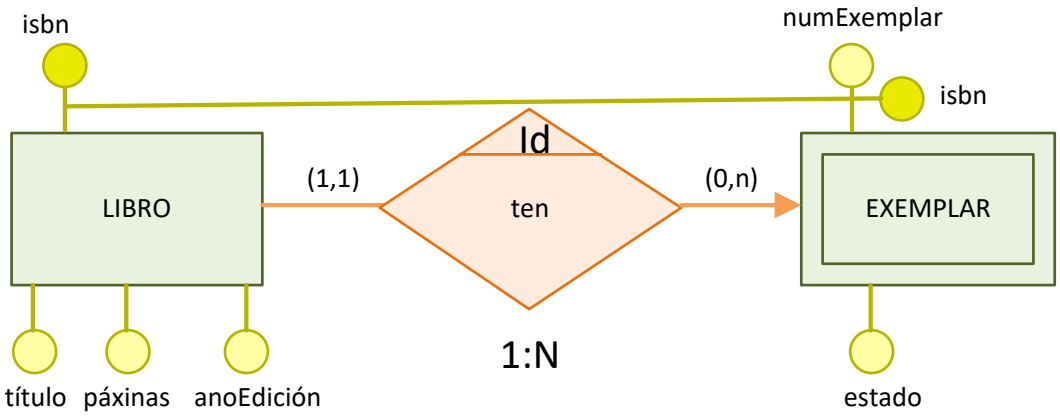
**TEN** (*nif, codCompoñente, codProxecto*)

B:R, M:C

**PROXECTO** (*codProxecto*, dercriciónProxecto)

### 1.3.11 Tarefa 11.Transformar entidades débiles

Transforma o seguinte modelo conceptual entidade-interrelación ao modelo lóxico relacional:



**Solución**

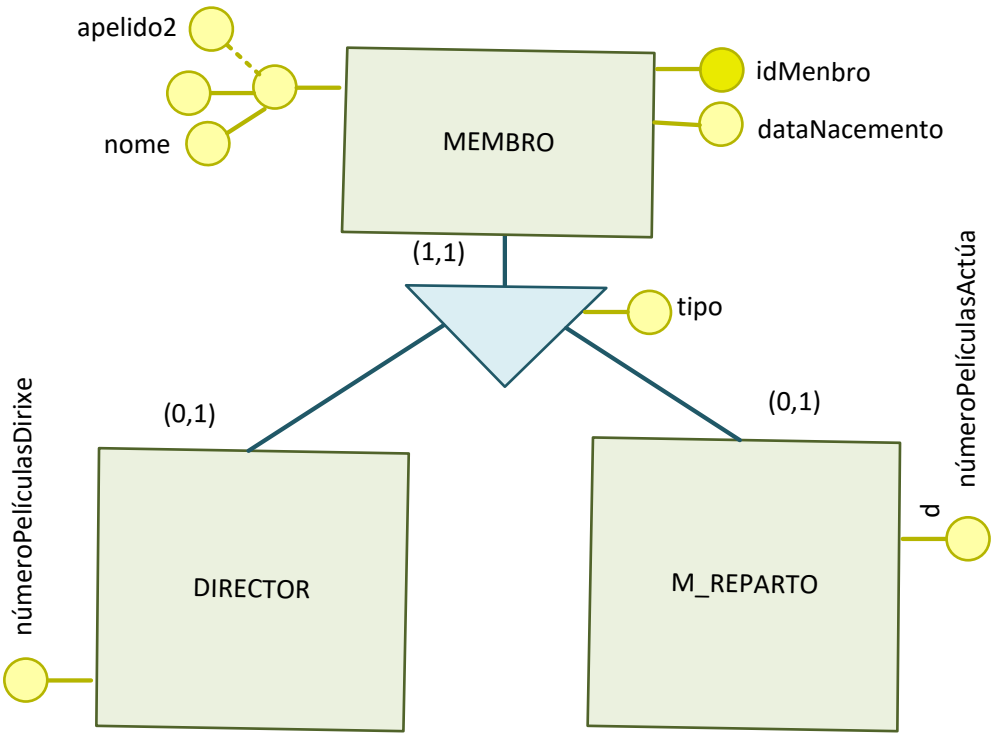
**LIBRO** ( *isbn*, titulo, anoEdicion, paxinas, título)

B:C, M:C

**EXEMPLAR** ( *isbn*, *numExemplar*, estado)

### 1.3.12 Tarefa 12. Transformar especializacións

Transforma o seguinte modelo conceptual entidade-interrelación ao modelo lóxico relacional




**Solución**

Proponse dúas solucións

Solución 1:

Ao tratarse dunha especialización solapada (unha ocorrencia pode participar en varios subtipos) parcial (existen ocorrencias que non participan en ningún subtipo) e o diferenciarse en tan poucos atributos os subtipos, proponse a creación dunha soa táboa que pon a nulos os atributos específicos dos subtipos.

**MEMBRO** ( *idMembro*, Nome, apelido1 apelido2*, dataNacemento, numeroPeliculasDirixe*, numeroPeliculasActúa*)

Solución 2:

No enunciado obsérvase que non existen interrelacións asociadas a ningunha das entidades, esta é pouco probable na vida laboral. Nos casos en que as entidades participen en interrelacións propias (o máis probable é que DIRECTOR participe nunha interrelación "Dirixe" e que M_REPARTO participe nunha interrelación "Actúa") o esquema a crear será o seguinte:

**MEMBRO** ( *idMembro*, Nome, apelido1 apelido2*, dataNacemento)

B:R, M:C

**DIRECTOR** ( *idDirector*, numeroPeliculasDirixe)

B:R, M:C

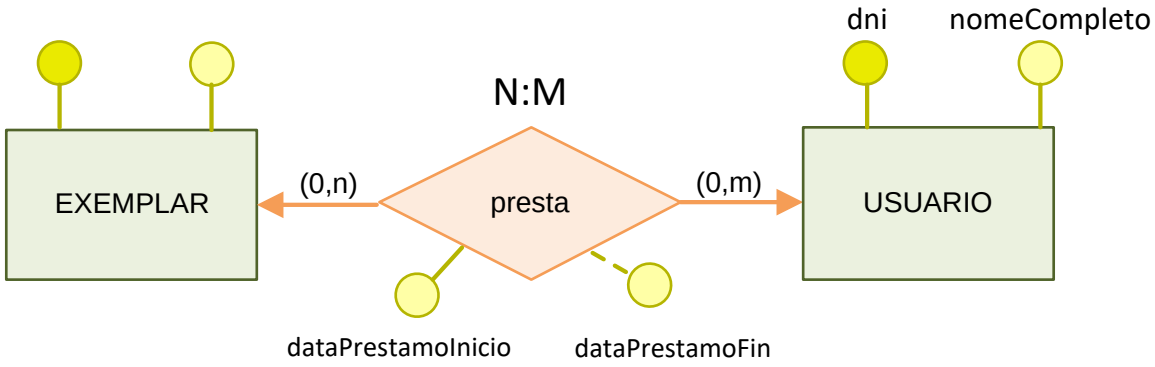
**M_REPARTO** ( *idMembroReparto*, numeroPeliculasActua)

Na relación MEMBRO poderíase engadir un atributo discriminante "tipo" que tomase o valor director, actor ou ambos para simplificar futuros accesos aos datos.

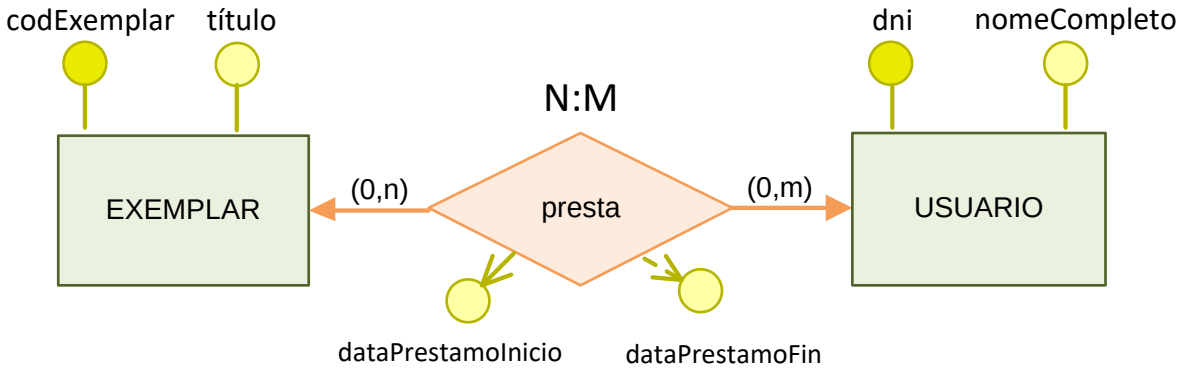
### 1.3.13 Tarefa 13. Transformar dimensión temporal

Une os seguintes modelos entidade relación cos seus grafos relacionais

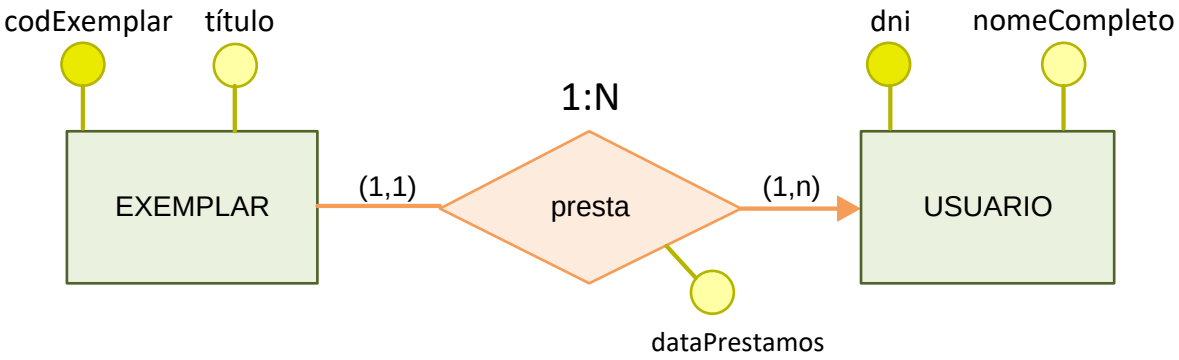
- Tarefa 13.1




- Tarefa 13.2




- Tarefa 13.3.




Opcións:

– A) No seguinte modelo coñecese cal é o ultimo exemplar prestado a un usuario

**EXEMPLAR** (*codExemplar*, Título, *dniUsuario*)
B:R, M:C
**SOCIO** (*dni*, nomeCompleto)

– B) Con este deseño un usuario non poderá ter en préstamo o mesmo exemplar en períodos diferentes de tempo, xa que as tuplas que representasen este feito terían a mesma clave.

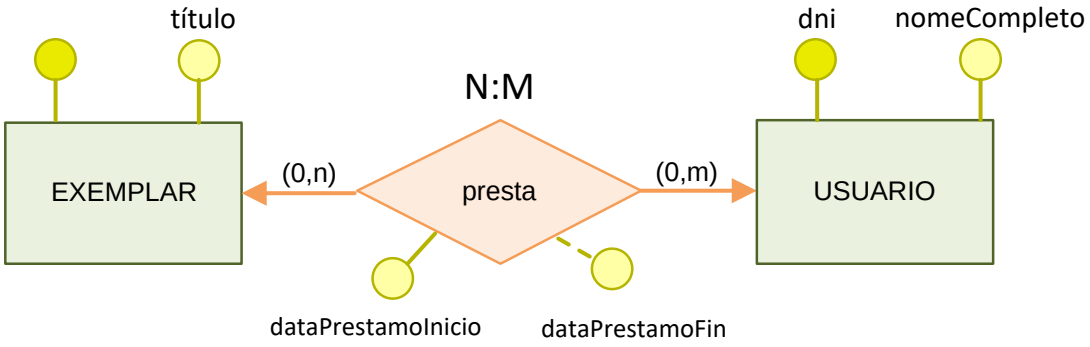
**EXEMPLAR** (*codExemplar*, título)
B:R, M:C
**PRESTA** (*codExemplar, dniUsuario,* dataPrestamoInicio, dataPrestamosFin*)
B:R, M:C
**SOCIO** (*dni*, nomeCompleto)

– C) Con este deseño un usuario poderá ter en préstamo o mesmo exemplar en períodos diferentes de tempo, permitindo con elo ter un histórico de todos os exemplares que foron prestados a un usuario

**EXEMPLAR** (*codExemplar*, título)
B:R, M:C
**PRESTA** (*codExemplar, dniUsuario, dataPrestamoInicio*, dataPrestamosFin*)
B:R, M:C
**SOCIO** (*dni*, nomeCompleto)

## Solución

### Tarefa 13.1




**EXEMPLAR** (*codExemplar*, título)

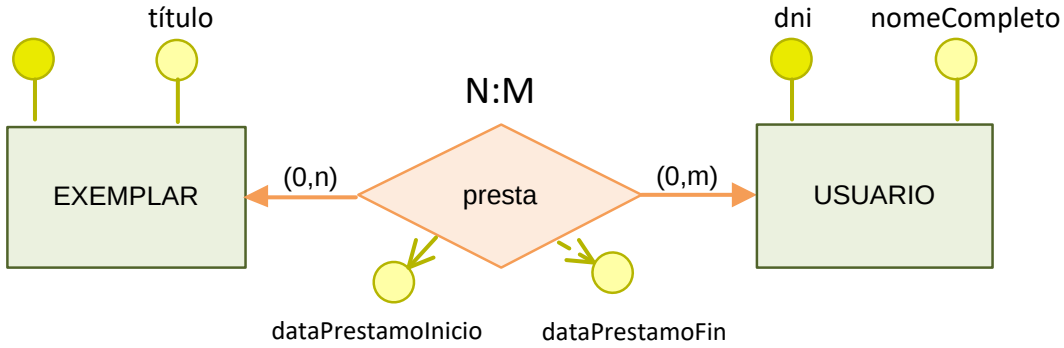
B:R, M:C

**PRESTA** (*codExemplar, dniUsuario,* dataPrestamoInicio, dataPrestamosFin*)

B:R, M:C

**SOCIO** (*dni*, nomeCompleto)

### Tarefa 13.2




**EXEMPLAR** (*codExemplar*, título)

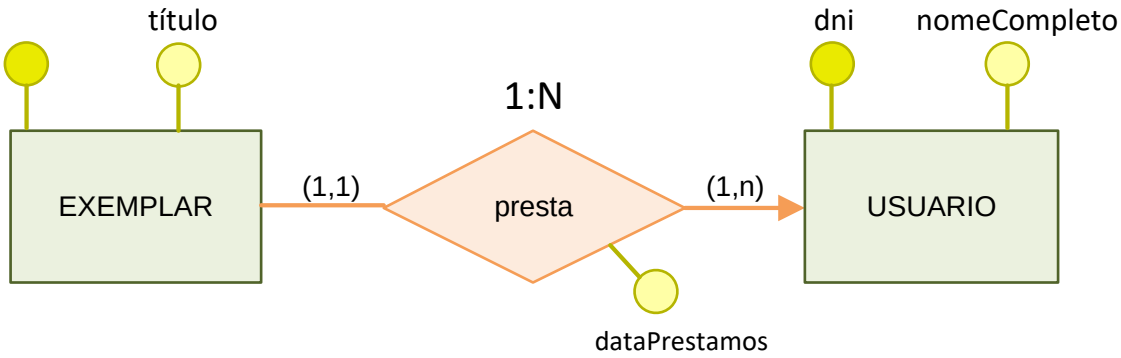
B:R, M:C

**PRESTA** (*codExemplar, dniUsuario, dataPrestamoInicio*, dataPrestamosFin*)

B:R, M:C

**SOCIO** (*dni*, nomeCompleto)

Tarefa 13.3.



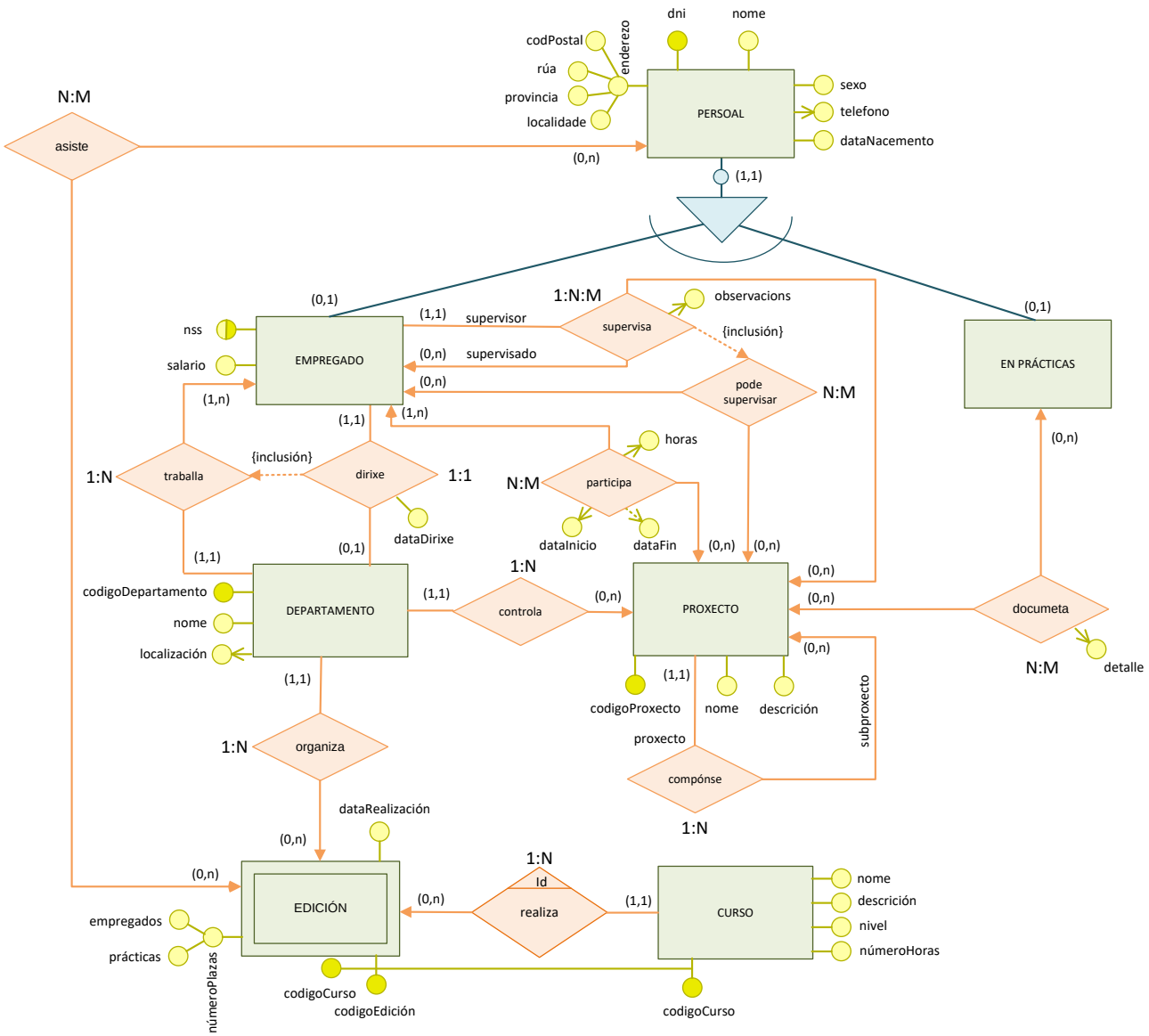

**EXEMPLAR** (*codExemplar*, Título, *dniUsuario*)

B:R, M:C

**SOCIO** (*dni*, nomeCompleto)

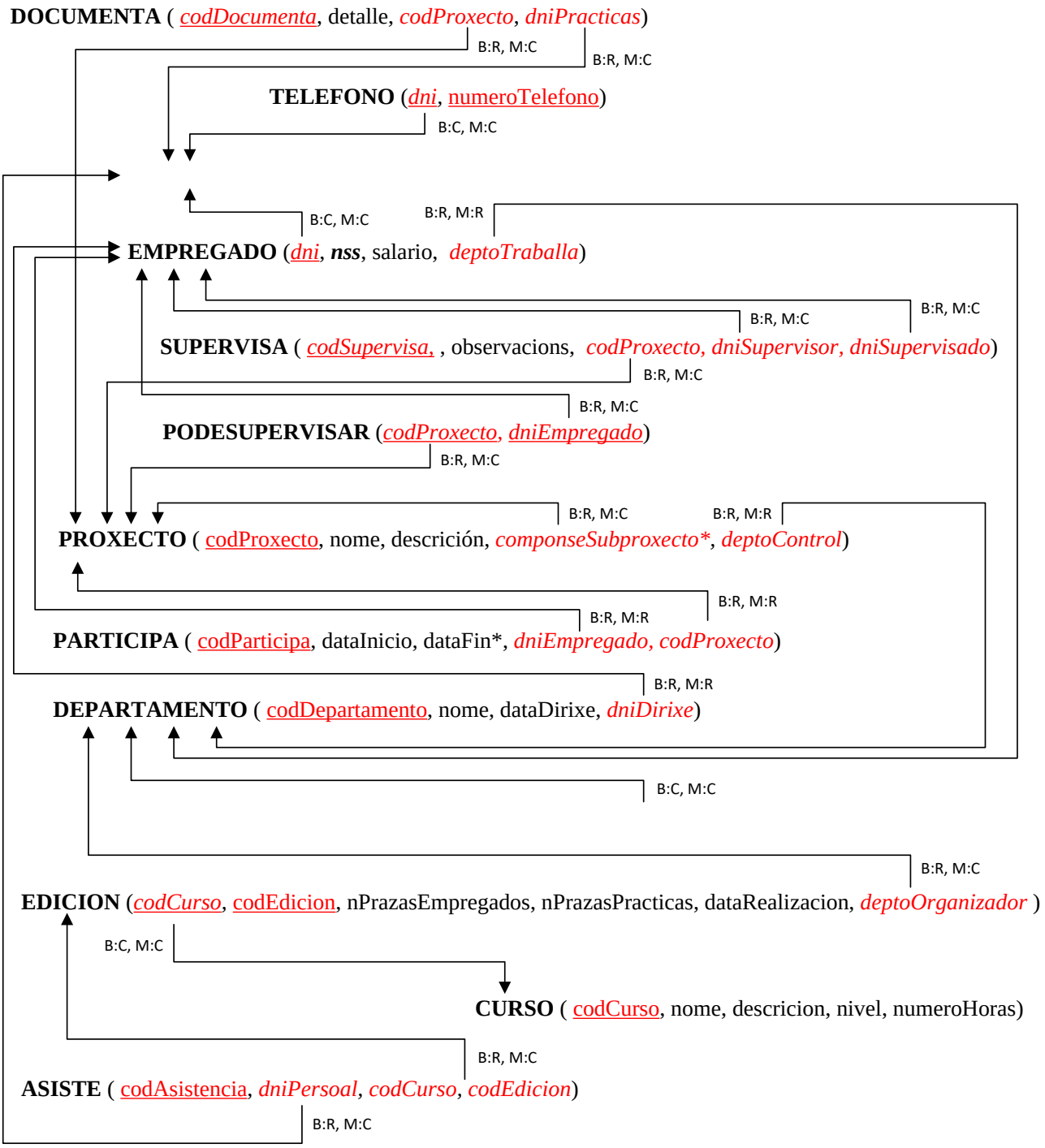
### 1.3.14 Tarefa 14. Transformar un MERE a MR.

Transforma o seguinte MERE a seu equivalente MR



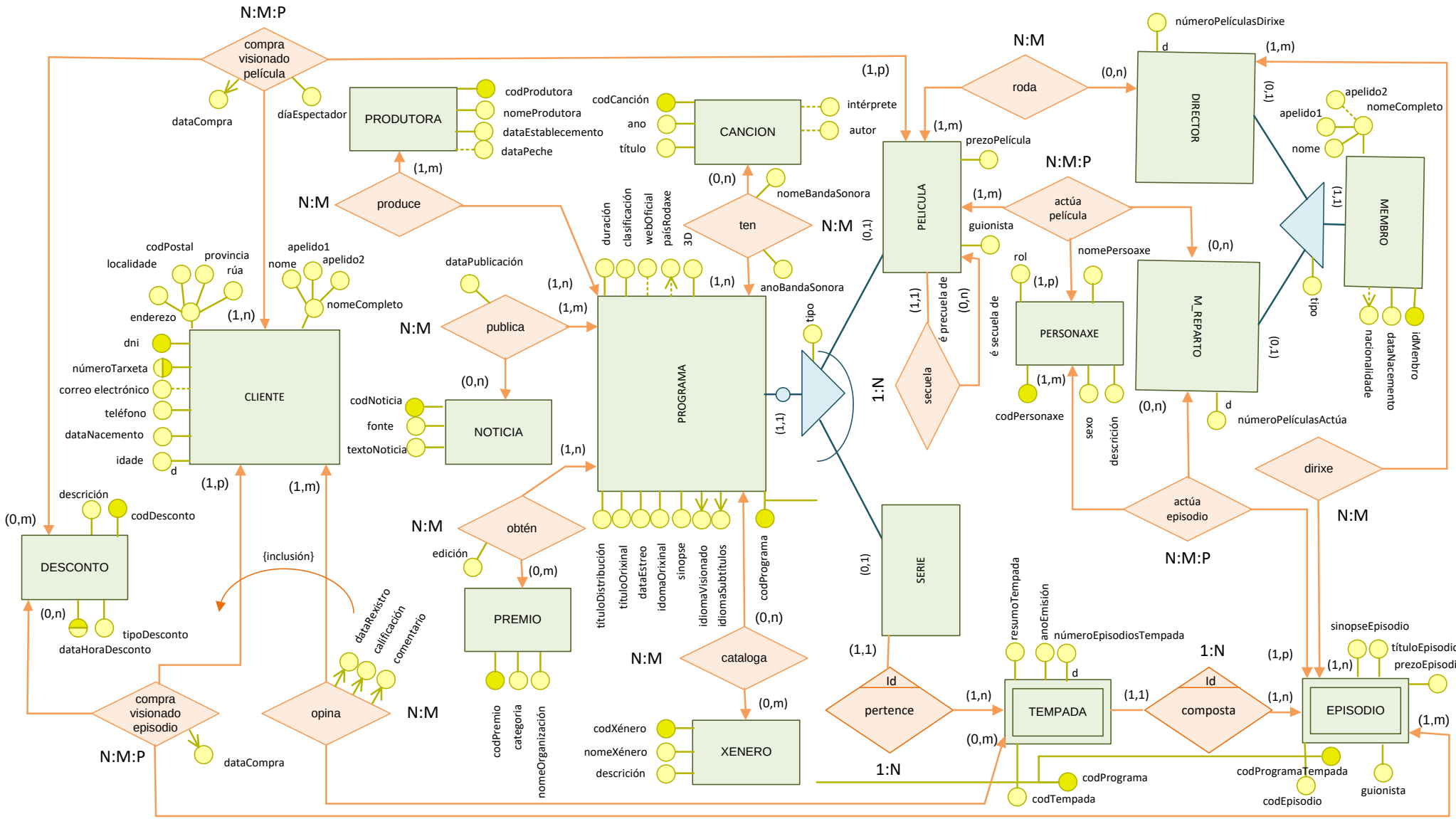

## Solución

Solución proposta.




### 1.3.15 Tarefa 15: Transformar o esquema conceptual a esquema relacional

Dado o seguinte esquema conceptual, implemente o esquema relacional.


N:M:P
compra visionado película
dataCompra
díaEspectador
PRODUTORA
codProdutora
nomeProdutora
dataEstablecemento
dataPeche
(1,m)
N:M
produce
codCanción
ano
título
CANCION
intérprete
autor
(0,n)
nomeBandaSonora
ten
N:M
(0,1)
duración
clasificación
webOficial
países Rodaxe
3D
(1,n)
anoBandaSonora
(1,p)
N:M
roda
(0,n)
(1,m)
prezoPelícula
PELICULA
guionista
(1,1)
é precuela de
(0,n)
é secuela de
secuela
N:M:P
actúa película
rol
nomePersoaxe
(1,p)
PERSONAXE
codPersonaxe
sexo
descrición
(0,n)
M_REPARTO
(0,1)
(1,m)
d
númeroPelículasActúa
DIRECTOR
númeroPelículasDirixe
(0,1)
(1,1)
MEMBRO
apelido2
apelido1
nome
nomeCompleto
tipo
nacionalidade
dataNacemento
idMembro
dirixe
N:M
actúa episodio
N:M:P
codPostal
localidade
provincia
rúa
enderezo
apelido1
nome
apelido2
nomeCompleto
(1,n)
CLIENTE
dni
númeroTarxeta
correo electrónico
teléfono
dataNacemento
idade
d
(1,p)
(1,m)
dataPublicación
publica
N:M
(1,n)
(1,m)
PROGRAMA
tipo
(1,1)
1:N
codNoticia
fonte
textoNoticia
NOTICIA
(0,n)
(1,n)
tituloDistribución
tituloOrixinal
dataEstreo
idomaOrixinal
sinopse
idiomaVisionado
idiomaSubtítulos
codPrograma
obtén
N:M
edición
(0,m)
PREMIO
codPremio
categoria
nomeOrganización
cataloga
N:M
(0,n)
(0,m)
codXénero
nomeXénero
descrición
XENERO
SERIE
(0,1)
(1,1)
Id
pertence
(1,n)
(0,m)
1:N
TEMPADA
resumoTempada
anoEmisión
númeroEpisodiosTempada
d
(1,1)
Id
composta
1:N
(1,n)
EPISODIO
(1,p)
(1,n)
(1,m)
sinopseEpisodio
títuloEpisodio
prezoEpisodi
codProgramaTempada
codEpisodio
guionista
codPrograma
codTempada
DESCONTO
descrición
codDesconto
(0,m)
(0,n)
tipoDesconto
dataHoraDesconto
{inclusión}
compra visionado episodio
N:M:P
dataCompra
opina
N:M
dataRexistro
calificación
comentario

## Solución

Móstrase unha posible solución

# 2. Materiais

## 2.1 Documentos de apoio ou referencia


- [EN 2002] ELMASRI, R.;NAVATHE, *S.B.Fundamentos de Sistemas de Bases de Datos* Addison-Wesley, 2002.
- [MPM 1999] DE MIGUEL, A; PIATTINI, M; MARCOS, E. .*Diseño de base de datos relacionales*. Ra-MA. 1999
- CONNOLLY, T; BEGG, C; STRACHAN, *A. Database system: A practical aproach desing, implementation and magnagement*.Addisson-Wesley, 1998
- SILBERSCHATZ,A; KORTH. H; SUDARSHAN, S; CONNOLLY, T; BEGG, C; STRACHAN, *A. Fundamentos de bases de datos*. McGraw-Hill, 1998
- DATE,C.J. *Introducción a los sistemas de bases de datos*. Addisson-Wesley, 1992
- DE MIGUEL, A; PIATTINI, M. *Concepción y diseño de bases de datos.* Ra-Ma, 1993
- DE MIGUEL, A; PIATTINI, *M. Fundamentos y modelos de bases de datos*. Ra-Ma, 1993
- Métrica:
  - http://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html#.ViS6eGThBz8