MODELADO AVANZADO DE INFORMACIÓN

TRABALLO TUTELADO

Curso 2021/2022

Ana Armenteros López

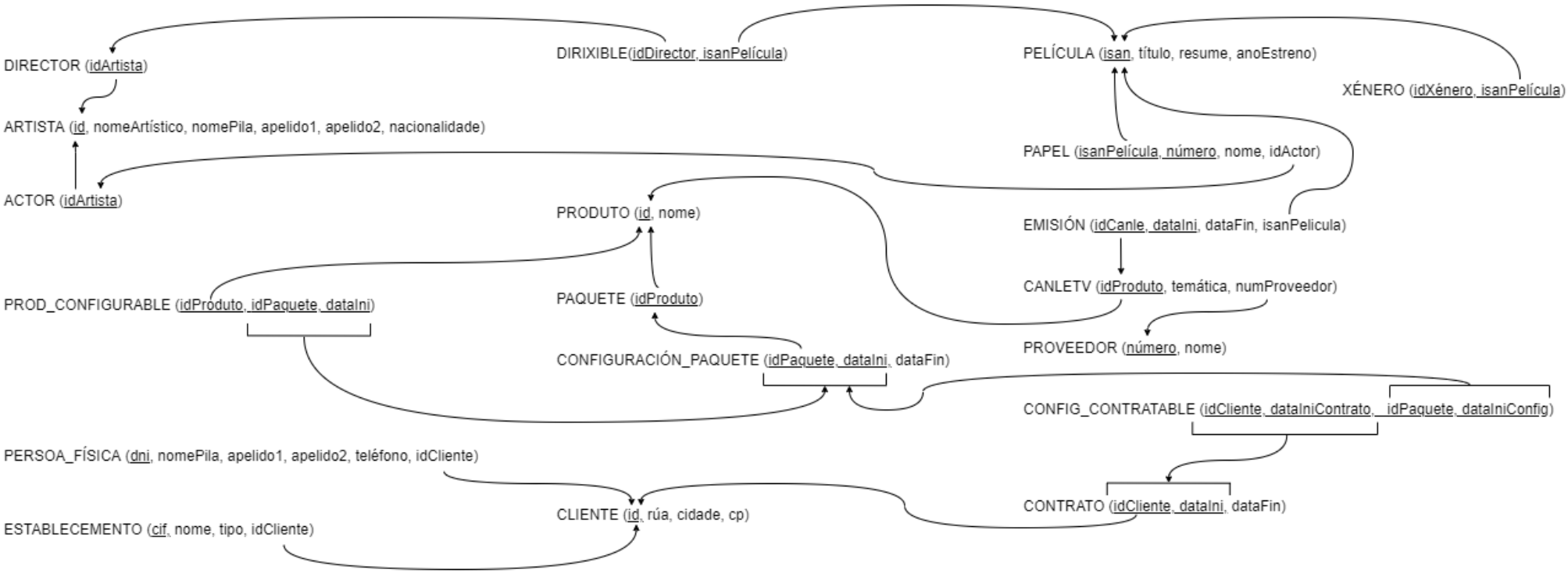
ana.armenteros@udc.es

Índice de contenido

Índice de figuras

Esquema relacional

Figura 1. Esquema relacional.



Para o esquema relacional considerouse:

- Unha táboa por **entidade**.

- Nas táboas correspondentes ás **entidades fortes**, a clave primaria simplemente será o seu mesmo identificador.

- Nas táboas correspondentes ás **entidades débiles**, a clave primaria ten que estar formada polo identificador da entidade forte, que a súa vez é clave foránea da mesma entidade forte, máis o discriminante (p. ex. a data de inicio nos históricos).

- Os **atributos monovaluados** gárdanse dentro da táboa da entidade á que corresponde.

- No caso do **atributo multivaluado "xénero"** da entidade "Película", refléxase como unha nova táboa que terá o identificador propio do xénero e o ISAN da película. Desta forma, poden asociarse máis dun xénero á mesma película. Considérase que a asociación dos id's dos xéneros co nome do xénero estará nunha táboa externa. A clave primaria estará formada, entón, por ambos identificadores, e o ISAN da película como clave foránea, ao ser a entidade que contén dito atributo.

- No caso dos **atributos compostos** (como "nome" e "enderezo") engádense como atributos aqueles dos que está composto. Por exemplo, para o caso de "Artista" que ten como atributo composto "Nome real", teranse como atributos "nomePila", "apelido1" e "apelido2".

- Para as **relacións N:N** creáronse as táboas "PROD_CONFIGURABLE", "CONFIG_CONTRATABLE" e "DIRIXIBLE", nas cales os seus únicos atributos corresponden a súa clave primaria, composta por dúas claves foráneas (unha para cada unha das entidades participantes da relación).

- Para a **xerarquía de "Artista-Director-Actor"**, créase unha táboa por clase. A táboa "Artista" estará formada por todos os seus atributos e as táboas "Director" e "Actor" soamente estarán formadas por unha clave primaria (e foránea ao mesmo tempo) que corresponde ao seu identificador, que é o mesmo que o de "Artista". Non se opta pola opción de táboa para cada subclase porque é solapada, e tampouco se opta pola única táboa xa que cada subclase conta con relacións propias.

- Para a **xerarquía de "Produto-Paquete-CanleTv"**, emprégase a mesma opción ca xerarquía anterior xa que, aínda que non é solapada, ao ter a superclase unha relación propia, é mellor reflexalo cunha táboa para cada unha. O mesmo para as subclases, que teñen relacións propias.

- Para a **categoría de "Cliente-PersoaFísica-Establecemento"**, créase unha táboa para cada clase. As entidades que forman a superclase ("Persoa Física" e "Establecemento") terán o identificador da categoría "Cliente" como clave foránea. Como claves primarias terán os seus identificadores propios.

Esquema de clases ODL

```
class Artista {
        attribute long id;
        attribute string nomeArtistico;
        attribute struct {
                String nomePila;
                String apelido1;
                String apelido2;
        } nomeReal;
        attribute string nacionalidade;
}

class Director: Artista {
        relationship set<Pelicula> peliculas
                inverse Pelicula::directores;
}

class Actor: Artista {
        relationship set<Papel> papeis
                inverse Papel::actor;
}

class Pelicula {
        attribute string isan;
        attribute string titulo;
        attribute string resume;
        attribute int anoEstreno;
        attribute set<String> xeneros;
        relationship set<Director> directores
                inverse Director::peliculas;
        relationship set<Papel> papeis
                inverse Papel::pelicula;
        relationship set<Emision> emisions
                inverse Emision::pelicula;
}

class Papel {
        attribute int numero;
        attribute string nome;
        relationship Pelicula pelicula
                inverse Pelicula::papeis;
        relationship Actor actor
                inverse Actor::papeis;
}
```

```
class Emision {
        attribute timestamp dataInicio;
        attribute timestamp dataFinal;
        relationship Pelicula pelicula
                inverse Pelicula::emisions;
        relationship CanleTv canleTv
                inverse CanleTv::emisions;
}

class Proveedor {
        attribute int numero;
        attribute string nome;
        relationship set<CanleTv> canlesTv
                inverse CanleTv::proveedor;
}

class Produto {
        attribute long id;
        attribute string nome;
        relationship set<ConfiguracionPaquete> configuracionsPaquete
                inverse ConfiguracionPaquete::produtos;
}

class CanleTv: Produto {
        attribute string tematica;
        relationship set<Emision> emisions
                inverse Emision::canleTv;
        relationship Proveedor proveedor
                inverse Proveedor::canlesTv;
}

class Paquete: Produto {
        relationship set<ConfiguracionPaquete> configuracionsPaquete
                inverse ConfiguracionPaquete::paquete;
}


class ConfiguracionPaquete {
        attribute timestamp dataInicio;
        attribute timestamp dataFinal;
        relationship set<Produto> produtos
                inverse Produto::configuracionsPaquete;
        relationship Paquete paquete
                inverse Paquete::configuracionsPaquete;
        relationship set<Contrato> contratos
                inverse Contrato::configuracionsPaquete;
}
```

```
class Contrato {
        attribute timestamp dataInicio;
        attribute timestamp dataFinal;
        relationship set<ConfiguracionPaquete> configuracionsPaquete
                inverse ConfiguracionPaquete::contratos;
        relationship Cliente cliente
                inverse Cliente::contratos;
}

class Cliente {
        attribute long id;
        attribute struct {
                String rua;
                String cidade;
                String cp;
        } enderezo;
        relationship set<Contrato> contratos
                inverse Contrato::cliente;
        relationship <PersoaFisica> persoaFisica
                inverse PersoaFisica::cliente;
        relationship <Establecemento> establecemento
                inverse Establecemento::cliente;

}

class PersoaFisica {
        attribute string dni;
        attribute struct {
                String nomePila;
                String apelido1;
                String apelido2;
        } nome;
        attribute string telefono;
        relationship <Cliente> cliente
                inverse Cliente::persoaFisica;
}

class Establecemento {
        attribute string cif;
        attribute string nome;
        attribute string tipo;
        relationship <Cliente> cliente
                inverse Cliente::establecemento;
}
```

Para o esquema ODL, considerouse:

- Unha clase por **entidade** (tanto para as fortes coma para as débiles), incluídas as **subclases** e **superclases**.

- O **atributo multivaluado "xénero"** de "Película" represéntase como unha colección de elementos de tipo "String", que recollerá todos os xéneros que ten a película.

- Os **atributos compostos** reflexaranse cun tipo "struct". Non se emprega unha nova clase por ser atributos pouco complexos.

- Para as **xerarquías** indícase, xunto ao nome da clase, o nome da superclase da que herda. Por exemplo, "class Director: Artista".

- Como **ningunha relación ten atributos**, non se crean clases novas. Soamente se fai uso de relationship <clase> ou relationship Set<clase>.

- Para a **categoría** indícase como unha relación 1:1 entre a categoría con cada unha das entidades que forman a superclase.

- Aquelas clases que se consideran que poden ter **identificadores moi grandes**, terán un atributo de tipo "long" en vez dun "int".

Patróns de acceso

Tense unha web na que pódese consultar información das películas emitidas, en cada canle, e dos artistas e papeis que as conforman. Especificaránse, pois, os patróns de acceso para dita web.

Cóntase cunha **páxina que mostra as películas**, co seu id numérico, ISAN, título, xénero/s, resume e ano de estreno. No caso de xénero, ao ser un atributo multivaluado, móstrase como un **array que contén o/s xénero/s** da película. Dende as películas, deséxase acceder aos directores e actores que participaron nelas, polo que refléxase cos seguintes arrays dentro das propias películas:

- Un **array co/s director/es que a dirixen**. Móstrase soamente o identificador do director, que pode enlazar cunha colección de artistas para obter a información restante dun director, xa que non interesa ter toda a información dos directores na páxina da película.
- Un **array co papel ou cos papeis que ten**. Móstrase o nome de cada papel xunto o identificador do actor que o interpreta, que pode enlazar coa colección de artistas para obter a información do actor que realiza dito papel, xa que non interesa ter a información dos actores na páxina da película.

Unha vez realizada unha película, non ten sentido modificar os seus directores ou os seus papeis e actores, así como a restante información da película, polo que as **posibles modificacións van ser, se existen, moi pouco frecuentes** (soamente para

posibles casos de error nos datos). As **insercións de películas poden ser frecuentes** ao non ter de forma obrigada unha emisión asociada a elas, non así os seus borrados, que serán moi pouco frecuentes.

Cóntase cunha **páxina para os artistas**, na que mostrarase o seu identificador, o seu nome artístico, o seu nome completo e a nacionalidade. As **modificacións van a ser moi pouco frecuentes**, soamente para casos de erros nos datos. A inserción e borrado de artistas dependerá da **inserción e borrado** de películas, polo que éstas **poden ser frecuentes.**

Cóntase cunha **páxina para as canles de TV**, na que se mostra o identificador, a temática e o seu nome. As canles poden contar con **algunhas modificacións**, por exemplo nos casos de cambio de proveedor que decida cambiar o nome da canle, pero deberían de ser pouco frecuentes. A **inserción e borrado é pouco frecuente** (as canles son case sempre as mesmas).

Considérase as **emisións** das películas en cada canle de TV como **consulta predominante**, na que haberá **múltiples insercións e moi frecuentes**. Teremos, pois, unha **páxina** na que se mostra, para cada emisión, o seu identificador, a súa datas de inicio e de fin, o identificador da película que se emitiu e os identificador da canles de TV onde se emitiu. **Non van a sufrir de modificacións**, a non ser que se trate da corrección dalgún erro nos datos, e **os borrados non deberían producirse** ao ser un rexistro (soamente nos casos no que a empresa considere que a partir de tantos anos poden borrarse ditas emisións porque carezan de datos relevantes). Non necesita ter toda a información das películas e das canles, polo que si se necesita acceder a máis información pode enlazarse cos seus identificadores.

Coleccións

Como se indicou no apartado anterior, tense **catro coleccións** en total:

- **peliculas** (peliculas.json).
- **artistas** (artistas.json).
- **canles** (canles.json).
- **emisions** (emisions.json).

Nesta parte do modelo, soamente cóntase cunha **xerarquía**, a cal refléxase cunha única colección “artistas”. Non é necesario diferenciar entre directores e actores, xa que comparten os mesmos atributos e un artista pode ser director e actor ao mesmo tempo, ademáis de que pódese acceder á información dun artista dende a colección “películas”, e ésta ten un array cos directores e outro cos actores cos seus papeis asociados.

En xeral, por simplicidade e comodidade, empréganse **id’s numéricos** para todos os documentos.

Nas **películas**, emprégase o id numérico en vez de o ISAN por comodidade, xa que éste último é demasiado longo.
O ano de estreno gárdase como un enteiro ao ser un número pequeno.
Os xéneros deben ir nun array, xa que pode ter varios pero non moitos.
Como pode ter máis dun director, pero nunca van ser moitos, terá un array cos identificadores deste. O mesmo caso para os papeis, no cal para cada nome do papel terá o identificador do actor que o interpreta asociado. Neste caso, ao non ter documentos soamente para os papeis, non será necesario un identificador numérico propio para referencialos (identifícase co seu nome e co id do actor).

Figura 2. Exemplo de documento da colección "peliculas".

```
mai> db.peliculas.findOne()
{
  _id: 2,
  isan: '0000-0003-7FF4-0000-U-0000-0001-B',
  titulo: 'Marriage Story',
  resume: 'Charlie, un director de teatro neoiorquino, e a súa muller actriz, Nicole, loitan por
superar un proceso de divorcio que os leva aos extremos tanto a nivel persoal como creativo.',
  ano_estreno: 2019,
  xeneros: [ 'Drama', 'Drama romántico', 'Teatro', 'Familia', 'Drama xudicial' ],
  directores: [ { id_director: 4 } ],
  papeis: [
    { nome_papel: 'Charlie', id_actor: 3 },
    { nome_papel: 'Nicole', id_actor: 2 }
  ]
}
```

Os **artistas** terán o identificador numérico (o mesmo co que se asociaban os actores cos papeis nas películas). Dependendo de se se considera que o nome artístico pode ser único ou non, poderíase empregar, ou non, como identificador. Neste caso non se arrisca ao contar co id numérico. Como dende a páxina de artista soamente interesa ver a súa información, non se engaden arrays cos id's das películas nas que participan ademáis de que, se fose este o caso, habería que engadir outro array "tipo" para poder indicar se participa nas películas como director e/ou actor, o que aumentaría a complexidade do deseño. Podemos saber, enlazando a colección películas con artistas, se a dirixen e/ou actúan nela.

Figura 3. Exemplo de documento da colección "artistas".

```
mai> db.artistas.findOne()
{
  _id: 3,
  nome_artistico: 'Edward Norton',
  nome_completo: {
    nome_pila: 'Edward Harrison',
    primeiro_apelido: 'Norton',
    segundo_apelido: null
  },
  nacionalidade: 'estadounidense'
}
```

As **canles** soamente se referencian nas emisións e, aínda que teñan poucos atributos, considerouse crear unha colección aparte, xa que non nos interesa ver dita información na páxinas das emisións. Emprégase o id numérico para facer referencia a eles por comodidade e porque pode considerarse que o nome dunha canle non ten porqué ser único.

Figura 4. Exemplo de documento da colección "canles".

```
mai> db.canles.findOne()
{ _id: 1, nome: 'Antena 3', tematica: 'Xeral' }
```

Nas **emisións** emprégase o id numérico por comodidade, senón habería que utilizar conxuntamente o id da canle e a data de inicio para identificar cada emisión. As datas póñense como tipo “data” para poder ter o día e hora exactas da emisión. Éngadese o id de película e o id de canle para poder coñecer a qué película corresponde e en qué canle se emitiu. Non interesa ter máis información destas últimas e, se ese fose o caso, pode enlazarse ás colección correspondentes mediante os seus respectivos id’s.

Figura 5. Exemplo de documento da colección "emisions".

```
mai> db.emisions.findOne()
{
  _id: 4,
  data_ini: ISODate("2020-05-11T22:05:00.000Z"),
  data_fin: ISODate("2020-05-11T23:35:00.000Z"),
  id_pelicula: 3,
  id_canle: 2
}
```

Consultas

1- **Mostrar, para cada película que teña como un dos seus xéneros “Drama”, o título da película e o nome artístico e nacionalidade do/s director/es.**

```
db.peliculas.aggregate([
   {$match: {xeneros: {$in: ["Drama"]}}},
   {$lookup: {from: "artistas", localField: "directores.id_director", foreignField:
"_id", as: "artistas"}},
   {$unwind: "$artistas"},
   {$project: {_id:0, titulo:1, "nome_artistico_director": "$artistas.nome_artistico",
"nacionalidade_director":"$artistas.nacionalidade"}}
])
```

2- Agrupar, polo id das películas e para aquelas que teñan, polo menos, dous emisións, o número de emisións e un array con todas as datas de inicio nas que se emitiu, ordenadas polo número de emisións.

```
db.emisions.aggregate([
    {$group: {_id: "$id_pelicula", datas_inicio: {$push: "$data_ini"}, num_emisions:
{$sum: 1}}},
    {$match: {num_emisions:{$gte:2}}},
    {$project: {_id:1, datas_inicio: 1, num_emisions: 1}},
    {$sort: {_id:1}}
])
```

3- Mostrar os cinco primeiros títulos, por orde alfabética, das películas estrenadas en 1998 e que teñan como xénero "Drama social".

```
    db.peliculas.find(
        {ano_estreno: 1998, xeneros:"Drama social"},
        {_id:0,titulo:1}
    ).sort({titulo:1}).limit(5)
```

4- Mostrar as emisións de películas entre o ano 2019 e 2022. Para cada unha delas, mostrar o id da película e o id da canle de TV onde se emitiu, además da data de inicio da emisión.

```
db.emisions.find(
    {data_ini: {$gte: new Date("2019-01-01"), $lte: new Date("2022-12-31")}},
    {_id:0, id_pelicula:1, data_ini:1, id_canle:1}
)
```

5- Mostrar as primeiras dez canles, por orde alfabética, que teñan como temática o cine.

```
db.canles.find({tematica:/^(C|c)ine/}, {_id:0, nome:1}).sort({nome:1}).limit(10)
```