

Bases de Datos de Grafos (BDG) Neo4j y su lenguaje Cypher

Francisco Moreno

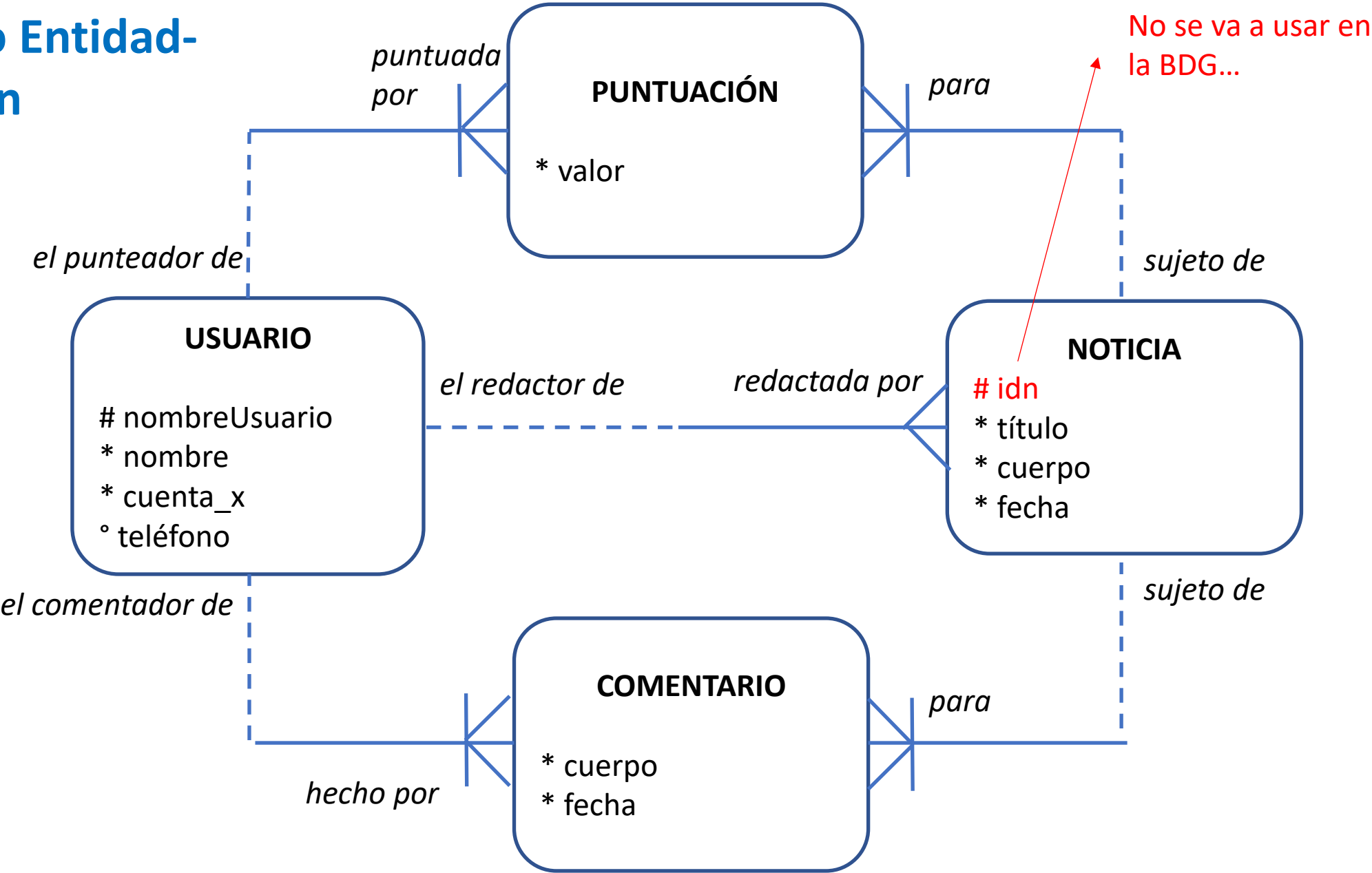
Modificado (y actualizado) a partir de Taller Neo4j.

Zorrilla & García Saíz, Universidad de Cantabria

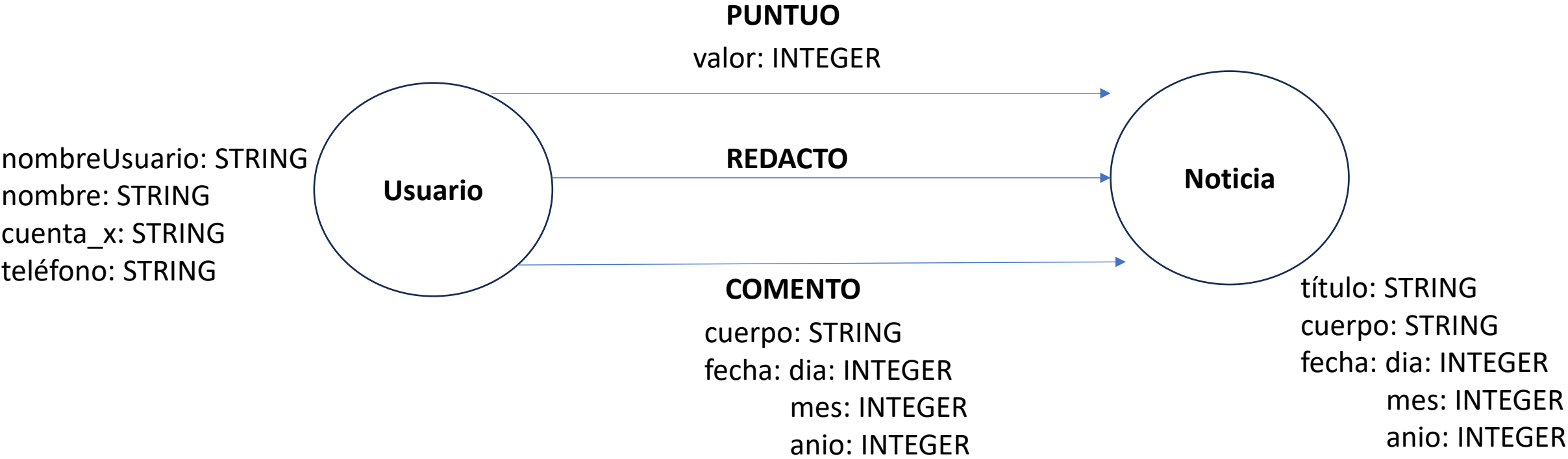
Se requiere diseñar un blog de noticias donde los usuarios registrados puedan publicar, puntuar y comentar noticias:

- Cada usuario tiene un **nombre usuario** (único), un **nombre** y una **cuenta de X** (Twitter, única). Además, un usuario puede tener un **teléfono**.
- Las noticias tienen un **título**, un **cuerpo** y una **fecha** de publicación. Son **publicadas** por un usuario (autor). Además, los usuarios pueden **puntuar** mediante un **valor** las noticias publicadas por los usuarios.
- Las noticias **reciben** comentarios, se registra el usuario que lo escribió (autor), el texto (**cuerpo**) del comentario y la **fecha** en el que lo hizo.

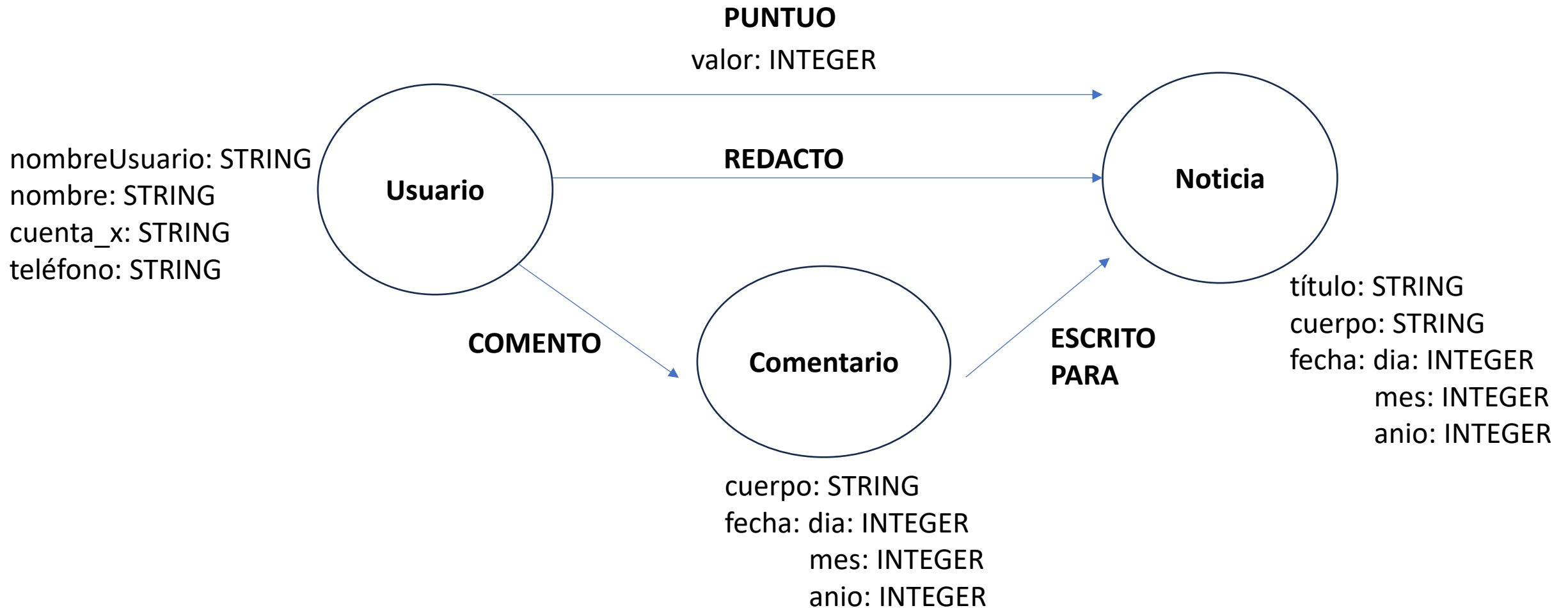
Modelo Entidad-Relación



Modelo de Grafo Versión 1



Modelo de Grafo Versión 2: donde la relación COMENTO se modela como un nodo



Recordar que en un diagrama de bases de datos de Neo4j existen dos elementos: **nodos** y **relaciones**.

- Además, ambos elementos pueden tener **propiedades** (equivalentes a los atributos en una relación del modelo relacional).
- Por otro lado, los nodos pueden tener **etiquetas** que los caracteriza (cualifica, agrupa).

- Primera pregunta: ¿Qué etiquetas definir?

Del enunciado y de su correspondiente diagrama E-R, se observan cuatro entidades. Se van a crear, tres etiquetas, una por cada entidad: **usuario**, **noticia** y **comentario**. La entidad **PUNTUACIÓN** se va a tratar como una relación en la BDG.


- Segunda pregunta: ¿Qué nodos formarán parte del diagrama?

Aquí, cada nodo que se incluya en el diagrama representará a los **usuarios**, **noticias** o **comentarios** junto con sus propiedades. Cada nodo; por lo tanto, deberá ser etiquetado de forma pertinente.

Se comienza con los nodos de **usuarios** y **noticias** así:

- Usuarios:

CREATE (**MiguelNodo**: **usuario** {nombre: 'Miguel', nombreUsuario: 'Miguel_u', cuenta_x: 'miguelete93'})



CREATE (**LaroNodo**: **usuario** {nombre: 'Laro', nombreUsuario: 'Laro_u', cuenta_x: 'larocantabro85', telefono: '1234567890'})

Noticias: en este ejemplo, las fechas se almacenan en propiedades separadas: año, el mes y el día (pero Neo4j tiene igualmente sus tipos de datos para fechas):

- `CREATE (NoticiaLaro1: noticia {titulo: 'Noticia1Laro', cuerpo: 'lorem ipsum...1', dia: 22, mes: 5, anio: 2017})`
- `CREATE (NoticiaLaro2: noticia {titulo: 'Noticia2Laro', cuerpo: 'lorem ipsum...2', dia: 2, mes: 3, anio: 2015})`
- `CREATE (NoticiaMiguel: noticia {titulo: 'NoticiaMiguel', cuerpo: 'lorem ipsum...3', dia: 15, mes: 3, anio: 2017})`

Actualmente, la BDG luce así: (note que hasta ahora los nodos están desconectados)

nombre: 'Laro'
nombreUsuario: 'Laro_u'
cuenta_x: 'larocantabro85'
telefono: '1234567890'

nombre: 'Miguel'
nombreUsuario: 'Miguel_u'
cuenta_x: 'miguelete93'

titulo: 'Noticia1Laro'
cuerpo: 'lorem ipsum...1'
dia: 22
mes: 5
anio: 2017

titulo: 'Noticia2Laro'
cuerpo: 'lorem ipsum...2'
dia: 2
mes: 3
anio: 2015

titulo: 'NoticiaMiguel'
cuerpo: 'lorem ipsum...3'
dia: 15
mes: 3
anio: 2017

- Tercera pregunta: ¿hay que definir restricciones?

En el enunciado, se indica que los **nombres de usuario** y las **cuentas de X** de los usuarios son **únicas**. Para ello se pueden definir dos restricciones de unicidad o definir esta restricción sobre una de las propiedades (por ej., cuenta_x) y hacer de la otra (por ej., nombreUsuario) la KEY de los nodos con la etiqueta “usuario” así:

```
CREATE CONSTRAINT FOR (u:usuario) REQUIRE u.cuenta_x IS UNIQUE
```

```
CREATE CONSTRAINT FOR (u:usuario) REQUIRE u.nombreUsuario IS NODE  
KEY
```

Nota: Si la versión de Neo4j no soportase la restricción **IS NODE KEY**, entonces crear una segunda restricción **IS UNIQUE** para nombreUsuario.

- También se puede usar: **CREATE CONSTRAINT IF NOT EXISTS FOR...**

- Si una restricción ya existe, se puede eliminar mediante:

DROP CONSTRAINT *nombredelconstraint*

Otras restricciones, por ejemplo:

- Para los usuarios: que el **nombre** y el **nombre de usuario** sean obligatorios (si no se ha podido definir como **NOT NULL**) → Según esto **NOT NULL** conlleva unicidad y obligatoriedad y **UNIQUE** solo unicidad.
- Similarmente, en las noticias las propiedades **título**, **fecha** (**día**, **mes** y **año**) y **cuerpo** son obligatorias.
- A continuación, se ejemplifican algunas de estas restricciones.

- CREATE CONSTRAINT FOR (u:usuario) REQUIRE u.nombreUsuario IS NOT NULL
- CREATE CONSTRAINT FOR (n:noticia) REQUIRE n.titulo IS NOT NULL
- CREATE CONSTRAINT FOR (n:noticia) REQUIRE n.cuerpo IS NOT NULL

Etc.

Considérense las siguientes consultas. Obtener:

- Todos los nodos.
- Los nodos con etiqueta “**usuario**”.
- Las noticias cuyo título termine en la letra “**o**”.
- Los usuarios que tengan la propiedad “**telefono**”.
- Los usuarios que tengan la propiedad número de teléfono o que se llamen **Miguel** (o ambas cosas).
- Las noticias publicadas en **2017**.
- Las noticias publicadas en **marzo** o **abril** (de cualquier año).
- Las noticias publicadas entre **2014** y **2016**.

- Todos los nodos:

MATCH (n) **RETURN** n

- Los nodos con etiqueta “usuario”:

MATCH (u:usuario) **RETURN** u

- Las noticias cuyo título termine en la letra “o”:

MATCH (n:noticia) **WHERE** n.titulo **ENDS WITH** 'o' **RETURN** n

- Los usuarios que tengan la propiedad “telefono”:

MATCH (u:usuario) **WHERE** u.telefono **IS NOT NULL** **RETURN** u

- Los usuarios que tengan la propiedad número de teléfono o que se llamen **Miguel** (o ambas cosas):

```
MATCH (u:usuario) WHERE u.telefono IS NOT NULL OR  
u.nombre = 'Miguel' RETURN u
```

- Las noticias publicadas en **2017**:

```
MATCH (n:noticia) WHERE n.anio = 2017 RETURN n
```

- Las noticias publicadas en **marzo** o **abril** (de cualquier año):

```
MATCH (n:noticia) WHERE n.mes = 3 OR n.mes = 4 RETURN n
```

O también:

```
MATCH (n:noticia) WHERE n.mes IN [3, 4] RETURN n
```


- Las noticias publicadas entre 2014 y 2016:

```
MATCH (n: noticia) WHERE n.anio >= 2014 AND n.anio <= 2016  
RETURN n
```

- Cuarta pregunta: ¿cuáles son las relaciones entre los nodos?

Acá, se tienen estas relaciones:

- Entre los usuarios y las noticias: “*usuario redacta noticia*” y “*usuario puntúa noticia*” (esta última a través de la entidad PUNTUACIÓN (en el modelo E-R) que se va a tratar como una relación en la BDG).
- Entre los usuarios y los comentarios: “*usuario comenta comentario*”.
- Entre los comentarios y las noticias “*los comentarios (de los usuarios) son escritos para una noticia*”.

Continuando con el ejemplo, se va a añadir, para cada una de las tres noticias la relación con el usuario que la redactó:

- Se va a llamar “**REDACTO**” a esta relación.
- Como los nodos ya están creados, primero se deben usar operaciones **MATCH** que retornen y guarden en variables los nodos sobre los que se desea establecer la relación.

```
MATCH(u:usuario) WHERE u.nombreUsuario = 'Laro_u'
```

```
MATCH(n:noticia) WHERE n.titulo = 'Noticia1Laro'
```

```
CREATE (u)-[:REDACTO{}]->(n)
```

→ Acá se pueden poner propiedades de la relación

```
MATCH(u:usuario) WHERE u.nombreUsuario = 'Laro_u'
```

```
MATCH(n:noticia) WHERE n.titulo = 'Noticia2Laro'
```

```
CREATE (u)-[:REDACTO{}]->(n)
```

```
MATCH(u:usuario) WHERE u.nombreUsuario = 'Miguel_u'
```

```
MATCH(n:noticia) WHERE n.titulo = 'NoticiaMiguel'
```

```
CREATE (u)-[:REDACTO{}]->(n)
```

Ahora, se van a insertar las **relaciones de puntuación**. Se va a llamar “**PUNTUO**” a la relación que indica la puntuación que un usuario le otorgó a una noticia.

```
MATCH(u:usuario) WHERE u.nombreUsuario = 'Laro_u'
```

```
MATCH(n:noticia) WHERE n.titulo = 'NoticiaMiguel'
```

```
CREATE (u)-[:PUNTUO {valor: 3}]->(n)
```

→ Propiedad de la relación

```
MATCH(u:usuario) WHERE u.nombreUsuario = 'Miguel_u'
```

```
MATCH(n:noticia) WHERE n.titulo = 'Noticia1Laro'
```

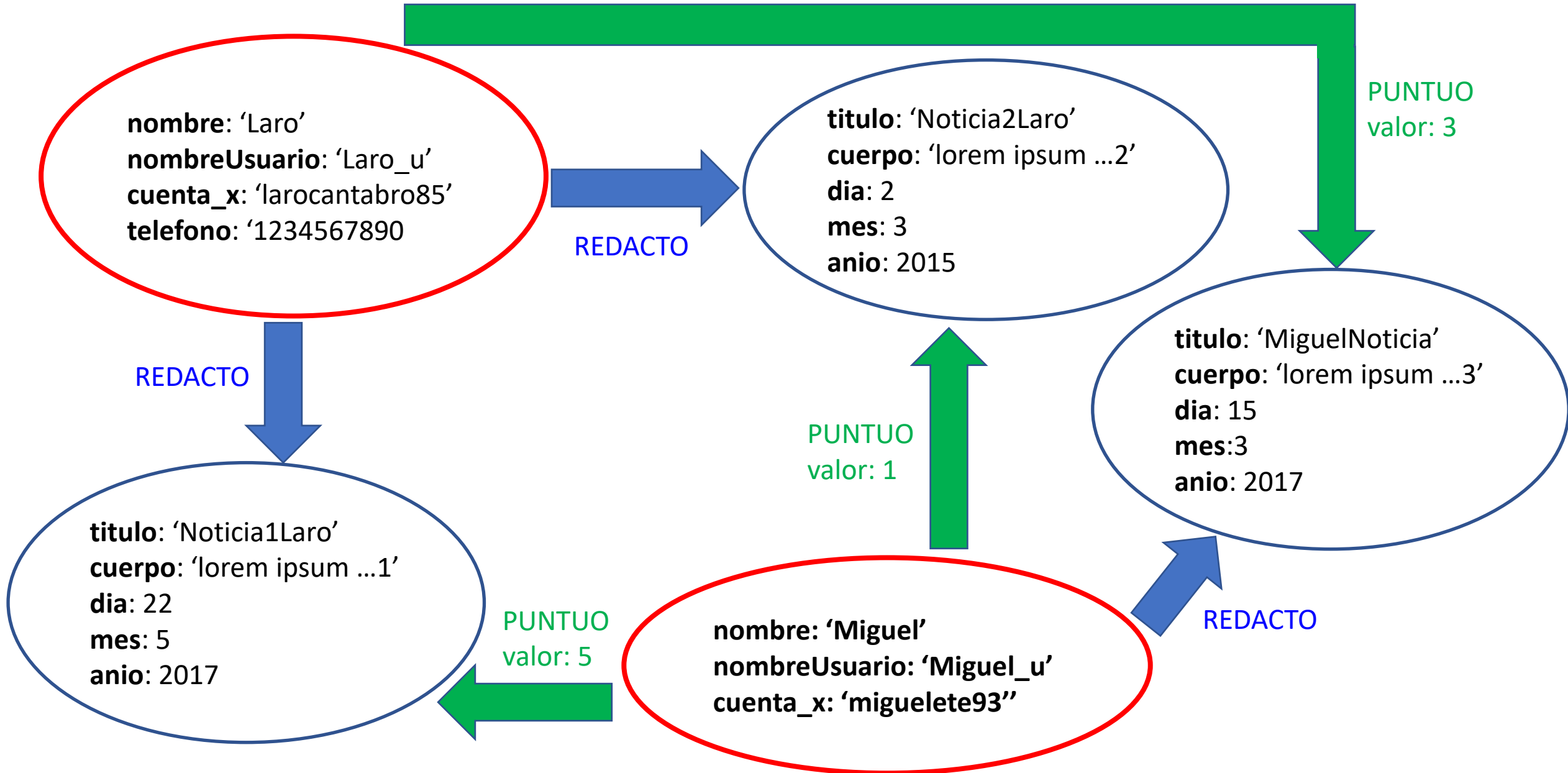
```
CREATE (u)-[:PUNTUO {valor: 5}]->(n)
```

```
MATCH(u:usuario) WHERE u.nombreUsuario = 'Miguel_u'
```

```
MATCH(n:noticia) WHERE n.titulo = 'Noticia2Laro'
```

```
CREATE (u)-[: PUNTUO {valor: 1}]->(n)
```

Ahora, la base de datos luce así:



Ahora considérense las siguientes consultas. Obtener:

- Los datos de las relaciones “PUNTUO” con puntuación igual a 5.
- Los datos de las relaciones “PUNTUO” hechas por Laro.
- Los datos de las relaciones “REDACTO” de las noticias publicadas en 2017 cuyo autor se llame “Laro”.
- Las cuentas de X (Twitter) de los usuarios que hayan puntuado noticias redactadas en 2017.

Este es un nombre cualquiera que uno elige.

Note que si va entre corchetes se trata de una relación, en cambio para los nodos va entre paréntesis.

- Los datos de las relaciones “PUNTUO” con puntuación igual a 5:

```
MATCH relation = (u:usuario)-[r:PUNTUO]->(n:noticia) WHERE r.valor = 5  
RETURN relation
```

Comparar con esta consulta:

```
MATCH (u:usuario)-[r:PUNTUO]->(n:noticia) WHERE r.valor = 5  
RETURN r
```

- Los datos de las relaciones “PUNTUO” hechas por Laro:

```
MATCH relation = (u:usuario)-[r:PUNTUO]->(n:noticia)  
WHERE u.nombre = 'Laro'  
RETURN relation
```

- Los datos de las relaciones “REDACTO” de las noticias publicadas en 2017 cuyo autor se llame Laro:


MATCH relation = (u:usuario)-[r:REDACTO]->(n:noticia)

WHERE n.anio = 2017 AND u.nombre = 'Laro' RETURN relation

- Las cuentas de X (Twitter) de los usuarios que hayan puntuado noticias redactadas en 2017:

MATCH relation = (u:usuario)-[r:PUNTUO]->(n:noticia)

WHERE n.anio = 2017 RETURN u.cuenta_x



Acá se muestra que en el RETURN puede ir algo distinto a relation, que es lo que se retorna en las consultas anteriores

Borrando nodos y sus relaciones

```
MATCH (u:usuario) WHERE u.nombre = 'Miguel' DETACH DELETE u
```

```
MATCH (u:usuario) WHERE u.nombre = 'Laro' DETACH DELETE u
```

```
MATCH (n:noticia) WHERE n.titulo = 'Noticia1Laro' DETACH DELETE n
```

```
MATCH (n:noticia) WHERE n.titulo = 'Noticia2Laro' DETACH DELETE n
```

```
MATCH (n:noticia) WHERE n.titulo = 'NoticiaMiguel' DETACH DELETE n
```

O simplemente, lo siguiente borra todos los nodos y relaciones:

```
MATCH (n) DETACH DELETE n
```

(pero los *constraints* aún quedan...se pueden ver con **SHOW CONSTRAINT**)

Ejercicios propuestos

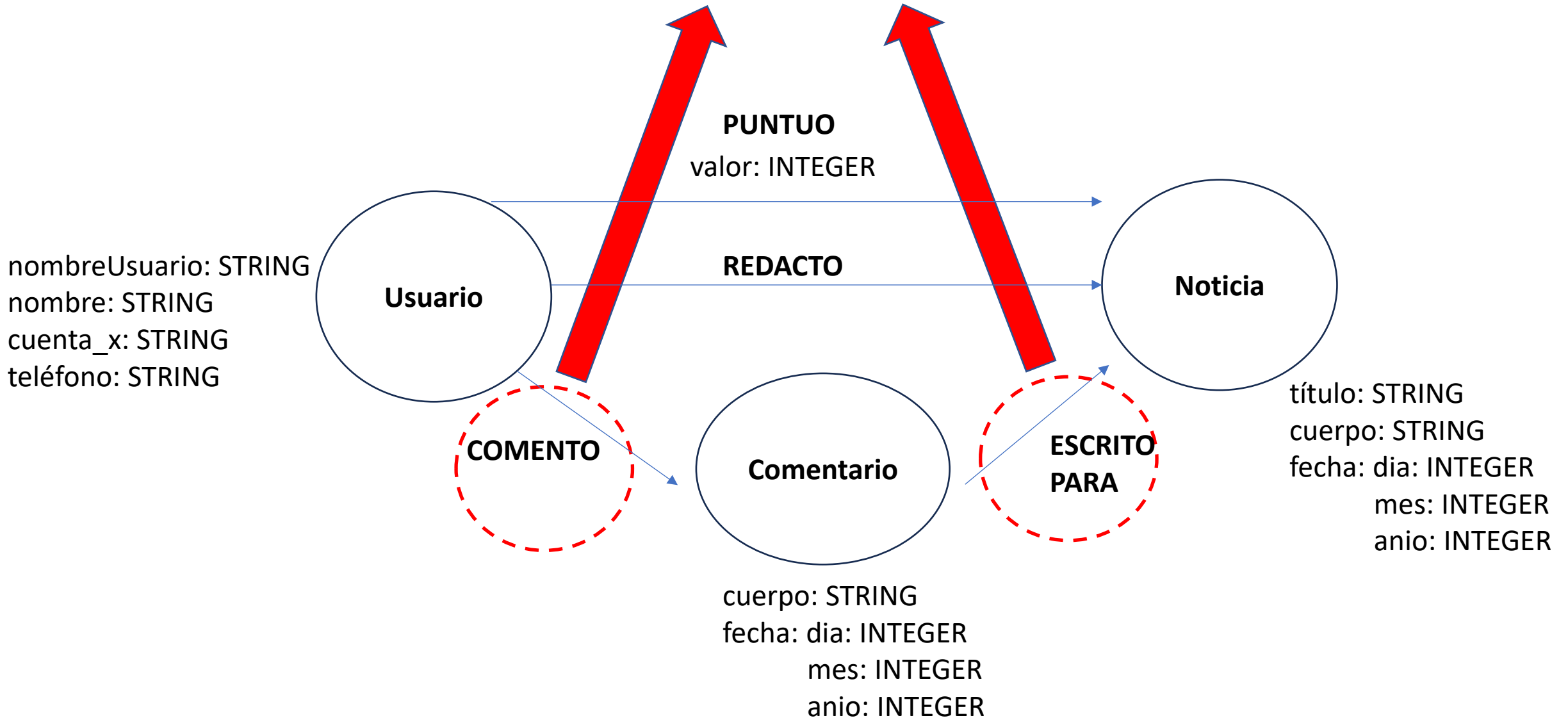
- Continuando con el ejemplo, se propone añadir al esquema los comentarios que los usuarios pueden hacer a las noticias.
- Se va a considerar que las noticias y los comentarios, además de su etiqueta correspondiente (o sea, **noticia** y **comentario**) tengan también una etiqueta común adicional (llamada “**contenido**”) que indique que estos nodos almacenan contenidos de un blog.
- Hacer entonces lo siguiente:

Ejercicios propuestos

- Añadir a los nodos **noticia** una segunda etiqueta llamada “**contenido**”.
- Crear **nodos de comentarios**, cada uno con doble etiqueta (“**comentario**” y “**contenido**”).
- Crear otras restricciones que se consideren necesarias (**¿impedir que un usuario comente o puntúe sus propias noticias?**)
- Crear las relaciones entre los **usuarios** y los **comentarios**. ¿Tienen estas relaciones alguna propiedad? ¿O deben ir estas propiedades más bien en los nodos comentarios?
- Crear las relaciones entre los **comentarios** y las **noticias**. ¿Tienen estas relaciones alguna propiedad? ¿O deben ir estas propiedades más bien en los nodos comentarios?

- Con respecto a las dos preguntas de la diapositiva anterior, esto significa que los atributos (propiedades en la BDG) de una entidad intersección (**COMENTARIO** en el modelo E-R) pueden ir en la entidad (nodo en la BDG) o ser puestos en cualquiera de las dos relaciones.
- Otra alternativa es tratar a COMENTARIO en la BDG como una relación (y no como un nodo) y ponerle propiedades...
- Esto muestra que hay una versatilidad para el diseño e implementación en la BDG...

Si estas relaciones tuviesen propiedades, ¿se podrían ubicar en el nodo Comentario?



Ejercicios propuestos

Con base en el nuevo diseño al que se han añadido los **comentarios**, hacer las siguientes consultas. Obtener:

- Los nodos etiquetados como “**contenido**”.
- Los nodos etiquetados como “**contenido**” y “**comentario**”.
- El número de comentarios hechos por Laro → **Investigar funciones de agregados: suma, promedio, máximo, etc.**
- Los comentarios hechos en **2017** sobre noticias redactadas en **2016**.
- El número de **contenidos** hechos por **Laro**.
- El número de comentarios de cada noticia, junto con el título de la misma.

Ejercicios propuestos

Muchos temas para profundizar:

- Neo4j soporta *triggers* con **APOC** (*Awesome Procedures on Cypher*).
- Otras (**¡muchas!**) funcionalidades que ofrece **APOC**:

[APOC support - Neo4j Aura](#)

- Manejo de fechas: Temporal functions, instant types.

Ejercicios propuestos: ¿Qué hacen estas consultas?

- Una de las **muchas funcionalidades** que ofrece APOC, **degree**:

```
MATCH(u:usuario)
WHERE u.nombreUsuario = 'Laro_u'
RETURN apoc.node.degree(u) AS output;
```

```
MATCH(u:usuario)
WHERE u.nombreUsuario = 'Laro_u'
RETURN apoc.node.degree.in(u) AS output;
```

```
MATCH(u:usuario)
WHERE u.nombreUsuario = 'Laro_u'
RETURN apoc.node.degree.out(u) AS output;
```