Álgebra relacional

En este capítulo vamos a cubrir el álgebra relacional, un lenguaje teórico que sirve para hacer consultas sobre un modelo relacional. Un lenguaje de consultas como el álgebra relacional está pensado para hacer preguntas o **consultas** que involucran datos en una base de datos. Primero vamos a repasar algunos conceptos previos, luego veremos los operadores básicos del álgebra relacional, luego el operador *join* ($\bowtie$) y finalmente algunos ejercicios propuestos.

1. Conceptos previos

Antes de partir con álgebra relacional, recordemos que la forma de representar datos en el modelo relacional es mediante **esquemas** e **instancias**. Podemos pensar en el **esquema** como el encabezado de una tabla, en la que indicamos los atributos (columnas), los tipos de dichos atributos y las llaves primarias. Podemos pensar en la **instancia** como una tabla ya poblada con ciertas filas. Recordemos además que a las filas también las llamaremos **tuplas**. Así, por ejemplo el siguiente esquema:

`Actores(aid: int, nombre_actor: string, edad: int)`

representa una tabla con actores que tiene tres columnas:

- `aid`, que representa el id del actor y es de tipo *integer*
- `nombre_actor` que representa el nombre del actor y es de tipo *string*
- `edad` que representa la edad del actor y es de tipo *integer*

Además, notamos que la columna `aid` está subrayada, con lo que estamos denotando que esta es la llave primaria, por lo que no pueden haber dos tuplas que tengan el mismo valor en esta columna.

Mientras tanto, una instancia posible para la relación de actores es:

aid	nombre_actor	edad
1	Leonardo DiCaprio	45
2	Christian Bale	46
3	Matthew McConaughey	50

Tabla 1: una instancia para la tabla de actores.

Y otra instancia puede ser la siguiente:

aid	nombre_actor	edad
1	Robert Downey Jr.	54
2	Scarlett Johansson	35
3	Gwyneth Paltrow	47

Tabla 2: otra instancia para la tabla de actores.

Como vemos, tenemos dos tablas con el mismo esquema (una forma de ver esto es que su encabezado es igual), sin embargo las tuplas que poblaron la tabla son distintas. Además, es importante ver que anteriormente mostramos el esquema para una relación. Si presentamos el esquema de todas nuestras relaciones tendremos el **esquema de la base de datos**. Durante este capítulo vamos a trabajar con el siguiente esquema de ejemplo:

- `Actores(aid: int, anombre: string, edad: int)` [1]
- `Películas(pid: int, pnombre: string, genero: string, director: string)`
- `Actuó_En(aid: int, pid: int, nombre_personaje: string)`

La tabla de Películas guarda el identificador de la película, junto a su nombre, género y nombre del director. La tabla de Actuó_En nos dice en que película actuó que actor, y además almacenamos el nombre del personaje que aquel actor tuvo para esa película. Ahora vamos a presentar un esquema de ejemplo que usaremos para este capítulo.

`Actores`

aid	anombre	edad
1	Robert Downey Jr.	54
2	Scarlett Johansson	35
3	Gwyneth Paltrow	47
4	Tom Holland	23
5	Chris Evans	38

(a) tabla de ejemplo para actores.

`Actuó_En`

aid	pid	nombre_personaje
1	1	Tony Stark
1	2	Tony Stark
1	3	Tony Stark
1	4	Tony Stark
2	1	Natasha Romanoff
2	4	Natasha Romanoff
3	1	Pepper Potts
3	2	Pepper Potts
3	3	Pepper Potts
4	1	Peter Parker
4	3	Peter Parker
5	1	Steve Rogers
5	4	Steve Rogers

(b) tabla de ejemplo para Actuó En.

`Películas`

pid	pnombre	género	director
1	Avengers: Endgame	Fantasía	Hermanos Russo
2	Iron Man	SciFi	Jon Favreau
3	Spiderman: Homecoming	SciFi	Jon Watts
4	Captain America: Civil War	Drama	Hermanos Russo

(c) tabla de ejemplo para películas.

Figura 1: esquema e instancias de ejemplo.

2. Operadores del álgebra relacional

El álgebra relacional es el lenguaje teórico sobre el que se sustentan mayoría de los lenguajes de consulta. El álgebra relacional tiene varios operadores que reciben tablas como *input* y retornan tablas como *output*. Estos operadores nos permiten filtrar por valores, quitar columnas, cruzar información entre distintas tablas y más.

2.1. Selección y proyección

El álgebra relacional incluye los operadores **Selección** (σ), para filtrar ciertas columnas de una relación, y **Proyección** (π), que permite dejar solamente las columnas de interés. Estos operadores nos permiten manipular una sola relación. Por ejemplo, supongamos que queremos obtener todos los actores mayores de 40 años de la tabla de la Figura 1a. Para esto usamos el operador selección:

$$\sigma_{\texttt{edad} > 40}(\texttt{Actores}) \tag{1}$$

Aquí lo que estamos haciendo es tomar la tabla actores y retornar una tabla que contiene solamente las tuplas que satisfacen que `edad` > 40. El resultado en este caso sería:

[1] Notemos que cambiamos el nombre del atributo que llevaba el nombre del actor.

aid	anombre	edad
1	Robert Downey Jr.	54
3	Gwyneth Paltrow	47

Tabla 3: resultado de la consulta $\sigma_{\texttt{edad} > 40}(\texttt{Actores})$.

El operador selección ($\sigma_{\text{condición}}$) solo va a retornar las tuplas que satisfagan la condición. Esta condición puede señalarse con comparaciones del estilo $=, \neq, >, \geq, <$ y $\leq$. Podemos combinar estos operadores utilizando $\wedge$ y $\vee$. Por ejemplo, si queremos todas las películas dirigidas por los "Hermanos Russo" que sean del género drama, podemos hacer la siguiente consulta en álgebra relacional:

$$\sigma_{\texttt{director='Hermanos Russo'} \wedge \texttt{género='Drama'}}(\texttt{Películas}) \tag{2}$$

Ahora, supongamos que queremos todos los nombres de los personajes que tenemos en nuestra base de datos. Para eso utilizaremos el operador **proyección** (π), que nos sirve para dejar solo las columnas que nos interesan:

$$\pi_{\texttt{nombre\_personaje}}(\texttt{Actuó\_En}) \tag{3}$$

El resultado de la consulta es la siguiente tabla:

nombre_personaje
Tony Stark
Natasha Romanoff
Pepper Potts
Peter Parker
Steve Rogers

Tabla 4: resultado de la consulta $\pi_{\texttt{nombre\_personaje}}(\texttt{Actuó\_En})$.

Que es la tabla `Actuó_En` pero en la que solamente dejamos la columna `nombre_personaje`. Quizás te estás preguntando por qué no aparecen más veces los nombres de los personajes, ya que estos estaban repetidos en varias filas. Esto es porque el álgebra relacional trabaja sobre conjuntos, por lo que **no existen los duplicados**, esto quiere decir que no pueden haber dos filas con el mismo valor. Si el resultado de una consulta de álgebra relacional fuese a entregar una fila duplicada, esta se elimina.

Ahora veamos otro ejemplo. Supongamos que ahora solo queremos el nombre de los actores con edad mayor a 40 años. Como los operadores del álgebra relacional retornan tablas, entonces podemos anidar los operadores. De esta forma, la consulta la podemos escribir de la siguiente forma:

$$\pi_{\texttt{nombre}}(\sigma_{\texttt{edad} > 40}(\texttt{Actores})) \tag{4}$$

En este caso primero hacemos una selección sobre la tabla de Actores para filtrar los que tengan más de 40 años, para después aplicar una proyección sobre el resultado de esta selección.

2.2. Unión

En el álgebra relacional tenemos operadores de conjuntos. El primero que vamos a estudiar es la **Unión** ($\cup$) que nos permite unir dos tablas. Notemos que dos tablas para ser unidas tienen que tener el mismo número de atributos y sus columnas correspondientes, tomadas en el orden de izquierda a derecha, deben tener los mismos dominios. Por ejemplo, si tenemos dos tablas `R(a: int, b: string)` y `S(c: int, d: string)` se pueden unir, porque ambas tablas tienen el mismo número de atributos y son respectivamente del mismo tipo. Si la tabla `S` fuese de la forma `S(c: int, d: int)`, las tablas `R` y `S` no se pueden unir. Veamos ahora una consulta de ejemplo.

Supongamos que queremos el nombre de todos los actores y directores de nuestra base de datos. Para esto podemos unir la proyección de los nombres en ambas tablas:

$$\pi_{\texttt{nombre}}(\texttt{Actores}) \cup \pi_{\texttt{director}}(\texttt{Películas}) \tag{5}$$

El resultado sería:

anombre
Robert Downey Jr.
Scarlett Johansson
Gwyneth Paltrow
Tom Holland
Chris Evans
Hermanos Russo
Jon Favreau
Jon Watts

Tabla 5: resultado de la consulta $\pi_{\texttt{nombre}}(\texttt{Actores}) \cup \pi_{\texttt{director}}(\texttt{Películas})$.

Notemos que el nombre del único atributo, por convención, es el nombre del encabezado de la primera tabla de la unión. Suena razonable querer renombrar ciertos atributos, para esto ahora veremos el operador de renombre.

2.3. Renombre

El operador de **Renombre** (ρ) tiene dos utilidades:

- Definir una relación auxiliar en base al resultado de una consulta.
- Cambiar el nombre de los atributos de una relación.

Supongamos que queremos definir una relación auxiliar con los actores de más de 40 años. Esto se hace de la siguiente forma:

$$\rho(\texttt{Actores40},\ \sigma_{\texttt{edad} > 40}(\texttt{Actores})) \tag{6}$$

Aquí estamos definiendo una relación auxiliar llamada `Actores40` con todos los actores de más de 40 años. Esta relación la podemos consultar. Por ejemplo si queremos solo el nombre de dichos actores podemos hacer la consulta $\pi_{anombre}(\texttt{Actores40})$.

También podemos renombrar atributos de una relación. Por ejemplo, si queremos escribir los atributos de la tabla `Actores` al inglés, podemos hacer lo siguiente:

$$\rho((\texttt{aid} \to \texttt{aid},\ \texttt{anombre} \to \texttt{aname},\ \texttt{edad} \to \texttt{year}),\ \texttt{Actores}) \tag{7}$$

Y de la misma forma, podemos renombrar la relación, junto con sus atributos de una sola vez:

$$\rho(\texttt{Actors}(\texttt{aid} \to \texttt{aid},\ \texttt{anombre} \to \texttt{aname},\ \texttt{edad} \to \texttt{year}),\ \texttt{Actores}) \tag{8}$$

2.4. Producto cruz

Ahora supongamos que queremos obtener el nombre de cada actor junto a los personajes que ha interpretado. Para esto debemos cruzar información de la tabla `Actores` y `Actuó_En`. Para cruzar información primero debemos utilizar el operador **Producto Cruz** ($\times$) que realiza el producto cartesiano entre dos relaciones. Supongamos la siguiente versión simplificada de nuestra base de datos en la Figura 2.

`Actores`

aid	anombre
1	Robert Downey Jr.
2	Scarlett Johansson

(a) tabla simplificada de ejemplo para actores.

`Actuó_En`

aid	pid	nombre_personaje
1	1	Tony Stark
1	2	Tony Stark
2	1	Natasha Romanoff

(b) tabla simplificada de ejemplo para Actuó En.

Figura 2: esquema simplificado de ejemplo.

Vamos a hacer la consulta:

$$\texttt{Actores} \times \texttt{Actuó\_En} \tag{9}$$

Que es el producto cruz de las relaciones `Actores` y `Actuó_En`. El producto cruz entre las relaciones `Actores` y `Actuó_En` se refleja en la Tabla 6:

Actores.aid	Actores.anombre	Actuó_En.aid	Actuó_En.pid	Actuó_En.nombre_personaje
1	Robert Downey Jr.	1	1	Tony Stark
1	Robert Downey Jr.	1	2	Tony Stark
1	Robert Downey Jr.	2	1	Natasha Romanoff
2	Scarlett Johansson	1	1	Tony Stark
2	Scarlett Johansson	1	2	Tony Stark
2	Scarlett Johansson	2	1	Natasha Romanoff

Tabla 6: ejemplo de producto cruz.

Que intuitivamente, corresponde a mezclar "todo con todo". Esto es, cada tupla de la relación `Actores` fue emparejada con cada tupla de la relación `Actuó_En`. Como nota, si hacemos el producto cruz de dos relaciones A y B ($A \times B$), el número de tuplas de $A \times B$ es igual a $|A| \cdot |B|$[2].

Lo primero por notar es el nombre de los atributos de la tabla del producto cruz. Aquí se heredan los nombres de los atributos de las tablas originales, pero se antepone el nombre de la tabla que viene (por ejemplo `Actores.aid`). A su vez, podríamos haber hecho *rename* de los atributos del producto cruz.

También que el resultado de la tabla 6 tiene algunas tuplas que no hacen mucho sentido, por ejemplo la tupla `(2, Scarlett Johansson, 1, 1, Tony Stark)` que mezcla la información que señala que la actriz con identificador 2 tiene el nombre Scarlett Johansson y que el actor 1. Intuitivamente, para responder la consulta "el nombre de cada actor junto a los personajes que ha interpretado" debemos hacer una selección sobre el producto cruz donde `Actores.aid = Actuó_En.aid`. Esto es:

$$\sigma_{\texttt{Actores.aid=Actuó\_En.aid}}(\texttt{Actores} \times \texttt{Actuó\_En}) \tag{10}$$

Ya que sabemos el valor del producto cruz y sabemos como evaluar una selección, podemos computar el resultado de la consulta de arriba. Esta consulta origina la siguiente tabla:

Actores.aid	Actores.anombre	Actuó_En.aid	Actuó_En.pid	Actuó_En.nombre_personaje
1	Robert Downey Jr.	1	1	Tony Stark
1	Robert Downey Jr.	1	2	Tony Stark
2	Scarlett Johansson	2	1	Natasha Romanoff

Tabla 7: ejemplo de producto cruz y selección.

En la que vemos que cada tupla de la tabla de actores fue **extendida** con los valores que le corresponden de la tabla `Actuó_En`. Finalmente, para obtener solamente los nombres de los actores y sus personajes tenemos que proyectar sobre la consulta anterior:

$$\pi_{\texttt{Actores.anombre, Actuó\_En.nombre\_personaje}}(\sigma_{\texttt{Actores.aid=Actuó\_En.aid}}(\texttt{Actores} \times \texttt{Actuó\_En})) \tag{11}$$

Que nos da por resultado:

Actores.anombre	Actuó_En.nombre_personaje
Robert Downey Jr.	Tony Stark
Scarlett Johansson	Natasha Romanoff

Tabla 8: ejemplo de producto cruz, selección y proyección.

Donde recordemos que eliminamos las filas que tienen el mismo valor en todos los atributos (en otras palabras, no hay filas duplicadas).

[2]Denotamos la cardinalidad o número de tuplas de una relación R como $|R|$.

2.5. Join

Lo que acabamos de hacer en el ejemplo anterior, que es cruzar información entre dos tablas mediante producto cruz y selección es una de las operaciones más importantes de todo sistema relacional. Esta operación se conoce como `Join` ($\bowtie$). En realidad este operador es simplemente una forma de abreviar la operación de producto cruz con selección, ya que:

$$A \bowtie_{\text{condición}} B \;=\; \sigma_{\text{condición}}(A \times B) \tag{12}$$

Así, la consulta de la consulta 10 (que produce como resultado la tabla 7) se puede escribir con el operador *join* como:

$$\texttt{Actores} \bowtie_{\texttt{Actores.aid=Actuó\_En.aid}} \texttt{Actuó\_En} \tag{13}$$

Notamos que en la consulta anterior el atributo por el que se está comparando se llama `aid` en ambas tablas. En ese caso no necesitamos indicar el atributo en la comparación. A esto lo llamamos **Equijoin**. En el caso de la consulta anterior, la podríamos escribir así:

$$\texttt{Actores} \bowtie \texttt{Actuó\_En} \tag{14}$$

Y en general, si tenemos dos relaciones A y B, cuyos atributos en común son $a_1, \ldots, a_j$, entonces $A \bowtie B$ representa el *join* $A \bowtie_{A.a_1=B.a_1 \wedge \cdots \wedge A.a_j=B.a_j} B$. Ahora veamos un ejemplo más complejo.

Ejemplo 2.1. Consideremos el esquema e instancia de la figura 1. Entregue una consulta en álgebra relacional que permita obtener cada nombre de actor junto cada nombre de película en que ha actuado.

Solución. Para esto, lo primero que vamos a hacer es extender el nombre e identificador de cada tupla de la tabla actor junto al id de las películas en que ha actuado. Para hacer esto haremos el *join* entre `Actores` y `Actuó_En` y proyectaremos lo que necesitamos. Esto lo vamos a almacenar en una tabla auxiliar llamada `ActoresPID`:

$$\rho(\texttt{ActoresPID},\ \pi_{\texttt{Actores.aid, Actores.anombre, Actuó\_En.pid}}(\texttt{Actores} \bowtie \texttt{Actuó\_En})) \tag{15}$$

Luego de esto, debemos hacer *join* del resultado anterior con la tabla de películas, y solo vamos a proyectar lo que necesitamos, que es el nombre del actor y nombre de la película. Así, el resultado final es el resultado de la consulta:

$$\pi_{\texttt{ActoresPID.anombre, Películas.pnombre}}(\texttt{ActoresPID} \bowtie \texttt{Películas}) \tag{16}$$

Donde recordamos que **no estamos señalando condición en el join** porque el atributo `pid` existe en ambas tablas, por lo que se asume que la condición es `ActoresPID.pid` = `Películas.pid`.

2.6. Self Join

En cuanto al *join*, a veces podemos querer hacer un **Self Join**, esto es, una consulta con *joins* en la que aparece la misma tabla más de una vez. Supongamos que tenemos la siguiente instancia de una base de datos:

- `Usuarios(uid: int, username: string)`
- `Sigue_A(seguidor_id: int, seguido_id: int)`

En que la primera tabla indica el identificador de un usuario y su nombre de usuario en una red social. La segunda tabla indica que el usuario con identificador `seguidor_id` sigue al usario con identificador `seguido_id`. Si queremos obtener todos los pares de nombres de usuario tal que el de la primera columna sigue al de la segunda columna podemos hacer lo siguiente:

$$\begin{gathered} \rho(\texttt{U1},\texttt{Usuarios}) \\ \rho(\texttt{U2},\texttt{Usuarios}) \\ \texttt{U1} \bowtie_{\texttt{U1.uid=seguidor\_id}} \texttt{Sigue\_A} \bowtie_{\texttt{U2.uid=seguido\_id}} \texttt{U2} \end{gathered} \tag{17}$$

En general lo que haremos será primero renombrar la relación que aparece más de una vez, y luego proceder como lo hemos hecho hasta ahora.

2.7. Intersección

Tal como existe un operador unión, también tenemos un operador **Intersección** ($\cap$) que sirve para intersectar dos tablas, siempre y cuando estas sean compatibles como conjuntos, tal como ocurría con la unión. La restricción es la misma: las tablas deben tener el mismo número de atributos, además de tener los mismos dominios correspondientemente. Así, la intersección de dos tablas $A \cap B$ es igual a una tabla que contiene a todas las tuplas que están en A y que también están en B.

Supongamos que queremos conocer todos los nombres de los actores que han sido dirigidos por los "Hermanos Russo" y además por "Jon Favreau". Para esto, primero tenemos que obtener el *join* entre `Actores`, `Actuó_En` y `Películas`.

$$\rho(\texttt{A\_P},\ \texttt{Actores} \bowtie \texttt{Actuó\_En} \bowtie \texttt{Películas}) \tag{18}$$

Una primera idea que se nos podría ocurrir para obtener lo que queremos es hacer lo siguiente:

$$\pi_{\texttt{A\_P.anombre}}(\sigma_{\texttt{A\_P.director='Jon Favreau'} \wedge \texttt{A\_P.director='Hermanos Russo'}}(\texttt{A\_P})) \tag{19}$$

En donde en la selección tenemos la condición $\texttt{A\_P.director} =$' Jon Favreau' $\wedge$ $\texttt{A\_P.director} =$' Hermanos Russo'. Sin embargo, lo que estamos diciendo aquí es que la columna director de cada tupla tenga el valor 'Jon Favreau' y 'Hermanos Russo' simultáneamente, lo que **no es posible**. por lo mismo, necesitamos otra forma de hacer esto.

Por lo mismo lo que haremos será obtener el nombre de todos los actores dirigidos por 'Jon Favreau' y los guardaremos en una relación llamada `Actores_Jon`, mientras que los actores dirigidos por los 'Hermanos Russo' serán guardados en la relación auxiliar `Actores_Hermanos`. Finalmente intersectaremos ambas tablas:

$$\begin{aligned}
&\rho(\texttt{Actores_Jon},\ \pi_{\texttt{A_P.anombre}}(\sigma_{\texttt{A_P.director='Jon Favreau'}}(\texttt{A_P}))) \\
&\rho(\texttt{Actores_Hermanos},\ \pi_{\texttt{A_P.anombre}}(\sigma_{\texttt{A_P.director='Hermanos Russo'}}(\texttt{A_P}))) \\
&\texttt{Actores_Jon} \cap \texttt{Actores_Hermanos}
\end{aligned} \tag{20}$$

2.8. Diferencia

Supongamos que queremos obtener todos los actores que no han sido dirigido por "Jon Watts". Quizás lo primero que querríamos hacer sería algo de este estilo: hacer una selección sobre la tabla películas donde el director no sea "Jon Watts" y luego hacer *join* del resultado con lo demás. Sin embargo, puede existir un actor (como Robert Downey Jr.) que ha actuado en otra película que no sea la dirigida por "Jon Watts", por lo que igual estaría presente en el resultado siendo que no debería estarlo. Por lo tanto esta alternativa **no** funciona.

Con los operadores que tenemos hasta ahora no podemos responder esta pregunta, puesto que todos los operadores que hemos utilizados son **monótonos**. ¿Qué significa esto? Esto significa que cualquier consulta en álgebra relacional que construyamos será monótona.

Una consulta Q se dice monótona si para todo par de instancias I, I' tal que $I \subseteq I'$ se tiene que $Q(I) \subseteq Q(I')$. Esto implica que si una consulta es monótona y la base de datos crece, la respuesta a la consulta también debe crecer. Intuitivamente, si teníamos un actor que no haya actuado en una película de "Jon Watts" este será parte de la respuesta. Pero si añadimos una tupla que indique que el actor sí actuó con "Jon Watts" (la base de datos creció) aquel actor va a dejar de ser parte de la respuesta.

Sabemos entonces que toda consulta que solamente use π, σ, $\times$, $\bowtie$, $\cup$ y $\cap$ será monótona, pero necesitamos expresar una consulta **no monótona**. Por lo tanto necesitamos un operador no monótono en nuestro álgebra. Este operador es el operador de conjuntos **Diferencia** ($-$). Al igual que la Unión y la Intersección, solamente puedo "restar" tablas que sean compatibles como conjuntos. Sean dos tablas A y B, el resultado de $A - B$ es una tabla que contiene todas las tuplas que están en A pero que **no están** en B. Antes de seguir con la consulta planteada, veamos un ejemplo.

Consideremos las tablas R y S de la Figura 3.

El resultado de la resta $R - S$ son todas las tuplas que están en R pero no en la tabla S. Este resultado lo podemos ver en la Figura 4.

Ahora para responder la consulta "Todos los actores que no han sido dirigidos por Jon Watts", hacemos lo siguiente. Primero obtenemos el identificador de todos los actores que han aparecido en una película de Jon Watts:

R

a	b
1	1
1	2
1	3
2	2

(a) tabla R ejemplo diferencia.

S

a	b
1	2
2	2

(b) tabla S ejemplo diferencia.

Figura 3: instancia ejemplo diferencia.

R - S

a	b
1	1
1	3

Figura 4: resultado ejemplo diferencia.

$$\rho(\texttt{Actores\_Watts},\ \pi_{\texttt{Actuó\_En.aid}}(\sigma_{\texttt{director='Jon Watts'}}(\texttt{Películas}) \bowtie \texttt{Actuó\_En}) \tag{21}$$

Ahora vamos a obtener el identificador de todos los actores y vamos a restarle la tabla `Actores_Watts`. El resultado lo guardaremos en una tabla auxiliar `Actores_No_Watts`:

$$\rho(\texttt{Actores\_No\_Watts},\ \pi_{\texttt{aid}}(\texttt{Actores}) - \texttt{Actores\_Watts}) \tag{22}$$

Finalmente hacemos el join del resultado anterior con la tabla `Actores` y proyectamos lo que necesitamos:

$$\pi_{\texttt{Actores.nombre}}(\texttt{Actores} \bowtie \texttt{Actores\_No\_Watts}) \tag{23}$$

Donde, nuevamente, tenemos que prestar atención en que estamos haciendo un *join* sin hacer explícita la condición, porque el nombre de los atributos en ambas tablas es `aid`.

3. Ejercicios propuestos

En esta sección te vamos a proponer que escribas ciertas consultas en álgebra relacional. Para esta sección, considera el mismo esquema de ejemplo que hemos usado en la gran parte de este capítulo, que es el de la Figura 1. Lo importante es que tus consultas funcionen para todas las instancias, no solamente para la que se muestra en el ejemplo.

Ejercicio 3.1. Entrega el nombre de todas las películas en las que hay un personaje llamado "Thor".

Ejercicio 3.2. Entrega el nombre y el género de todas las películas en las que hay un personaje llamado "Thor" o uno llamado "Hulk".

Ejercicio 3.3. Entrega el nombre de todos los actores que hayan actuado en una película que haya tenido un personaje llamado "Thor" y también otro llamado "Hulk".

Ejercicio 3.4. Entrega el nombre de los actores que han actuado en dos o más películas.

Hint: no, no necesitas contar, tienes que hacer el producto cruz de una tabla consigo misma y luego hacer los filtros adecuados.

Ejercicio 3.5. Entrega el identificador y el nombre de los actores que no han actuado con el actor con nombre "Robert Downey Jr.".

Ejercicio 3.6. Entrega el nombre de los actores que hayan actuado en todas las películas.

Hint: la clave es primero encontrar los actores que no hayan actuado en todas las películas. Para esto vas a necesitar hacer un producto cruz de todas las posibles combinaciones de actores y películas, para luego restar las combinaciones actor-película de tu instancia. Una vez que tengas los actores que no han actuado en todas las películas, probablemente vas a saber cómo seguir.