



Operaciones en bases de datos

Gestión de Datos

Maximiliano Arancibia

Educación Profesional - Escuela de Ingeniería

Tabla de Contenidos

Introducción

Operadores de bases de datos

Operador pipe

Operadores relacionales

Otros operadores (de R)

Pivotar

Separar y unir



Introducción

Operadores de bases de datos

Otros operadores (de R)



Supongamos que tenemos las siguientes bases de datos:

actores

id	nombre	edad
1	Leonardo DiCaprio	41
2	Matthew McConaughey	46
3	Daniel Radcliffe	27
4	Jessica Chastain	39
...

actuo_en

id_actor	id_pelicula
1	2
2	1
4	1
3	3
1	5
...	...

peliculas

id	nombre	año	categoria	calificacion	director
1	Interstellar	2014	SciFi	8.6	C. Nolan
2	The Revenant	2015	Drama	8.1	A. Iñárritu
3	Harry Potter	2011	Fantasia	8.1	D. Yates
4	The Theory of Everything	2014	Biografía	7.7	J. Marsh
5	Inception	2010	Adventure	8.8	C. Nolan
...



¿Que cosas nos gustaría consultarle a la base de datos?

Ejemplo 1: Liste el nombre de todos los actores

nombre
Leonardo DiCaprio
Matthew McConaughey
Daniel Radcliffe
Jessica Chastain



Ejemplo 2: Liste el nombre y la clasificación de todas las películas

nombre	calificacion
Interstellar	8.6
The Revenant	8.1
Harry Potter	8.1
The Theory of Everything	7.7
Inception	8.8



Ejemplo 3: Liste el nombre y la clasificación de todas las películas con calificación inferior a 8.5

nombre	calificacion
The Revenant	8.1
Harry Potter	8.1
The Theory of Everything	7.7



Ejemplo 4: Liste todas las película de Nolan

id	nombre	año	categoría	calificacion	director
1	Interstellar	2014	SciFi	8.6	C. Nolan
5	Inception	2010	Adventure	8.8	C. Nolan



Ejemplo 5: Liste todos los id de los actores de la película "Interstellar"

id
2
4



Ejemplo 6: Liste cada actor junto a todas las películas en las que ha actuado

id	nombre	nombre_pelicula
1	Leonardo DiCaprio	The Revenant
1	Leonardo DiCaprio	Inception
2	Matthew McConaughey	Interstellar
3	Daniel Radcliffe	Harry Potter
4	Jessica Chastain	Interstellar



Ejemplo 7: Liste todas las películas en que actúe Leonardo DiCaprio y que sean dirigidas por C. Nolan

nombre
Inception



Ejemplo 8: Liste todas las películas y la calificación en que actúe Leonardo DiCaprio o que sean dirigidas por C. Nolan

nombre	calificacion
Interstellar	8.6
The Revenant	8.1
Inception	8.8



Ejemplo 9: Liste el nombre de todos los actores y directores

nombre
Leonardo DiCaprio
Matthew McConaughey
Daniel Radcliffe
Jessica Chastain
C. Nolan
A. Iñárritu
D. Yates
J. Marsh



Ejemplo 10: Liste el nombre de todos los actores dirigidos por C. Nolan y A. Iñárritu

id	nombre	edad
1	Leonardo DiCaprio	41



¿Qué podemos concluir?

- Los resultados de las consultas también son tablas
- Parecen haber operaciones en común



Tabla de Contenidos

Introducción

Operadores de bases de datos

Operador pipe

Operadores relacionales

Otros operadores (de R)



Definiremos algunas operaciones que son comunes en los lenguajes de consulta.

- Usaremos el lenguaje R para llevarlas a cabo pero hay que destacar que estas operaciones son transversales en los lenguajes de consulta como lo es por ejemplo **SQL**.
- Estas operaciones forman los cimientos de todos los lenguajes de consulta



Veremos los operadores:

- Proyección



Veremos los operadores:

- Proyección
- Selección



Veremos los operadores:

- Proyección
- Selección
- Unión



Veremos los operadores:

- Proyección
- Selección
- Unión
- Renombrar



Veremos los operadores:

- Proyección
- Selección
- Unión
- Renombrar
- Producto cruz



Veremos los operadores:

- Proyección
- Selección
- Unión
- Renombrar
- Producto cruz
- Join



Veremos los operadores:

- Proyección
- Selección
- Unión
- Renombrar
- Producto cruz
- Join
- Intersección



Veremos los operadores:

- Proyección
- Selección
- Unión
- Renombrar
- Producto cruz
- Join
- Intersección
- Diferencia



Operador pipe

Importante entender el **operador pipe de dplyr** (`%>%`):

Sirve para concatenar operaciones de dplyr.

Ejemplo, digamos que necesitamos aplicar mas de una función, la instrucción seria:

tercero(segundo(primero(data)))

Usando el operador pipe nos permite escribir una secuencia de funciones de izquierda a derecha. Si es que alguna de las funciones tuviera algún parametro de entrada lo ponemos en el parentesis correspondiente:

primero(data) %>% segundo(param₂) %>% third(param₃)



Diremos que la **Proyección** es la operación que nos permite seleccionar atributos de una relación.

En R esto lo logramos con la función **select**:

```
select(tabla, c(a1, a2, ...))
```



Ejemplos:

Seleccione el nombre y el año de la tabla de películas:

Peliculas(ID, Nombre, Año, Categoria, Calificacion)



Ejemplos:

Seleccione el nombre y el año de la tabla de películas:

Peliculas(ID, Nombre, Año, Categoria, Calificacion)

ID Película	Nombre Película	Año	Categoría	Calificación (IMDB)
1	Interstellar	2014	Fantasía	8.6
2	Batman	2005	Acción	8.3
3	The Imitation Game	2014	Biografía	8.1
4	The Theory of Everything	2014	Biografía	7.7
5	Batman	1995	Acción	5.4



Ejemplos:

Seleccione el nombre y el año de la tabla de películas:

Esto en R lo programamos de esta manera:

```
select(peliculas, c('Nombre Película', 'Año'))
```



Ejemplos:

Seleccione el nombre y el año de la tabla de películas:

ID Película	Nombre Película	Año	Categoría	Calificación (IMDB)
1	Interstellar	2014	Fantasía	8.6
2	Batman	2005	Acción	8.3
3	The Imitation Game	2014	Biografía	8.1
4	The Theory of Everything	2014	Biografía	7.7
5	Batman	1995	Acción	5.4



Ejemplos:

Seleccione el nombre y el año de la tabla de películas.

Películas

ID Película	Nombre Película	Año	Categoría	Calificación (IMDB)
1	Interstellar	2014	Fantasía	8.6
2	The Revenant	2015	Drama	8.1
3	The Imitation Game	2014	Biografía	8.1
4	The Theory of Everything	2014	Biografía	7.7



Ejemplos:

Seleccione el nombre y el año de la tabla de películas.

Nombre Película	Año
Interstellar	2014
The Revenant	2015
The Imitation Game	2014
The Theory of Everything	2014



Dada una relación, diremos que la **selección o filtro** de esta en base a una condición, es una nueva relación que deja solo tuplas (filas) que cumplan dicha condición.

Estas condiciones pueden definirse con:

$$<, \leq, \geq, >, =, \neq$$

y combinar con entre ellas con las operaciones lógicas 'y' (\wedge) y 'o' (\vee)



En R encontramos dos maneras de utilizar el filtro:

- Con función **filter**:

filter(tabla, condicion)

- Buscando los datos con selectores booleanos:

tabla[condicion,]



Ejemplo:

Busquemos las películas de la categoría 'Biografía'



Ejemplo:

Busquemos las películas de la categoría 'Biografía'

- `filter(peliculas, Categoria=='Biografia')`



Selección o filtro

Ejemplo:

Busquemos las películas de la categoría 'Biografía'

- `filter(peliculas, Categoria=='Biografia')`
- `peliculas[peliculas$Categoria=='Biografia',]`

ID Película	Nombre Película	Año	Categoría	Calificación (IMDB)
1	Interstellar	2014	Fantasía	8.6
2	Batman	2005	Acción	8.3
3	The Imitation Game	2014	Biografía	8.1
4	The Theory of Everything	2014	Biografía	7.7
5	Batman	1995	Acción	5.4



Selección o filtro

Ejemplo:

Busquemos las películas de la categoría 'Biografía'

- `filter(peliculas, Categoria=='Biografia')`
- `peliculas[peliculas$Categoria=='Biografia']`

ID Película	Nombre Película	Año	Categoría	Calificación (IMDB)
1	Interstellar	2014	Fantasía	8.6
2	Batman	2005	Acción	8.3
3	The Imitation Game	2014	Biografía	8.1
4	The Theory of Everything	2014	Biografía	7.7
5	Batman	1995	Acción	5.4



Usamos la operación de unión cuando queremos juntar las tuplas de dos relaciones distintas. Es necesario que ambas relaciones tengan la misma cantidad y tipo de atributos.

En R podemos para dos tablas podemos usar los comandos:

- `union(tabla1, tabla2)`: entrega tuplas únicas en ambas tablas
- `rbind(tabla1, tabla2)`: entrega concatenación de ambas bases de datos



Usamos **union** para unir tablas y usamos **rbind** para concatenar.

Ejemplo:

- Nombre de actores y nombres de películas:

```
npelis <- select(peliculas, c('Nombre'))  
nact <- select(actores, c('Nombre'))  
union(npelis,nact)
```

¿Qué pasa si usamos *rbind*?



Podemos cambiar el nombre de los atributos, para esto usamos la función de R:

```
rename(tabla, c('atributo1'='at1'))
```

En este caso cambiamos el nombre del atributo 'at1' por el nombre 'atributo1' en la tabla.



Usamos **rename** para cambiar el nombre de algún atributo. Ejemplo:

- `rename(peliculas,c('Agno'='Año'))`
- `rename(peliculas,c('ID'='ID Pelicula'))`
- `rename(peliculas,c('Nombre'='Nombre Pelicula'))`



Producto Cruz

Nos falta cruzar información entre tablas, la función **crossing** permite hacer el producto cartesiano de dos relaciones. En otras palabras:

$$\begin{array}{c|cc} R_1 & A & B \\ \hline & a_1 & b_1 \\ & a_2 & b_2 \end{array} \times \begin{array}{c|cccc} R_1 \times R_2 & R_2 & A & C & D \\ \hline & a_1 & c_1 & d_1 \\ & a_2 & c_2 & d_2 \end{array} =$$

$R_1.A$	$R_1.B$	$R_2.A$	$R_2.C$	$R_2.D$
a_1	b_1	a_1	c_1	d_1
a_1	b_1	a_2	c_2	d_2
a_2	b_2	a_1	c_1	d_1
a_2	b_2	a_2	c_2	d_2

Observacion: la cardinalidad (numero de tuplas) está dada por

$$|R_1 \times R_2| = |R_1| \cdot |R_2|$$



Veamos un ejemplo:

Liste todos los id de los actores de la película "Interstellar"

1) Hacemos el producto cruz de *peliculas* y *actuo_en*
crossing(peliculas, actuo_en)



Veamos un ejemplo:

Liste todos los id de los actores de la película "Interstellar"

1) Hacemos el producto cruz de *peliculas* y *actuo_en*

crossing(peliculas, actuo_en)

pelicula.id	pelicula.nombre	pelicula.año	pelicula.categoría	pelicula.calificación	pelicula.director	actuo_en.id_actor	actuo_en.id_pelicula
1	Interstellar	2014	SciFi	8.6	C. Nolan	1	2
1	Interstellar	2014	SciFi	8.6	C. Nolan	2	1
1	Interstellar	2014	SciFi	8.6	C. Nolan	4	1
1	Interstellar	2014	SciFi	8.6	C. Nolan	3	3
1	Interstellar	2014	SciFi	8.6	C. Nolan	1	5
2	The Revenant	2015	Drama	8.1	A. Iñárritu	1	2
2	The Revenant	2015	Drama	8.1	A. Iñárritu	2	1
2	The Revenant	2015	Drama	8.1	A. Iñárritu	4	1
2	The Revenant	2015	Drama	8.1	A. Iñárritu	3	3
2	The Revenant	2015	Drama	8.1	A. Iñárritu	1	5
...		



Veamos un ejemplo:

Liste todos los id de los actores de la película "Interstellar"

1) Hacemos el producto cruz de *peliculas* y *actuo_en*

2) Filtramos cuando *id* de *pelicula* sea igual al *id_pelicula* en la tabla *actuo_en*

- `cast <- crossing(peliculas, actuo_en)`
- `filter(cast, peliculas.id==actuo_en.id_pelicula)`

peliculas .id	peliculas.no mbre	pelicula s.año	peliculas. categoria	peliculas.ca lificacion	peliculas .director	actuo_e n.id_act or	actuo_en.id _pelicula
1	Interstellar	2014	SciFi	8.6	C. Nolan	2	1
1	Interstellar	2014	SciFi	8.6	C. Nolan	4	1
2	The Revenant	2015	Drama	8.1	A. Iñárritu	1	2
3	Harry Potter	2011	Fantasia	8.1	D. Yates	3	3
5	Inception	2010	Adventure	8.8	C. Nolan	1	5



Veamos un ejemplo:

Liste todos los id de los actores de la película "Interstellar"

- 1) Hacemos el producto cruz de *peliculas* y *actuo_en*
- 2) Filtramos cuando *id* de *pelicula* sea igual al *id_pelicula* en la tabla *actuo_en*
- 3) Filtramos según id de película 'Interstellar'

- `cast <- crossing(peliculas, actuo_en)`
- `cast <- filter(cast, peliculas.id==actuo_en.id_pelicula)`
- `filter(cast, peliculas.id==1)`

peliculas .id	peliculas.no mbre	pelicula s.año	peliculas. categoria	peliculas.ca lificacion	peliculas .director	actuo_e n.id_act or	actuo_en.id _pelicula
1	Interstellar	2014	SciFi	8.6	C. Nolan	2	1
1	Interstellar	2014	SciFi	8.6	C. Nolan	4	1



Ojo! De este punto en adelante renombraremos las relaciones para facilitar la escritura:

```
A <- actores
```

```
P <- películas
```

```
actuo_en <- rename(actuo_en, c('id_a'='id_actor'))
```

```
actuo_en <- rename(actuo_en, c('id_p'='id_película'))
```



Liste cada actor junto a todas las películas en las que ha actuado



Liste cada actor junto a todas las películas en las que ha actuado

Debemos usar dos productos cruz:

- Entre películas y actuo_en
- Entre actuo_en y actores

y luego usar selección para ver que los id correspondientes calcen.

En base a esta operación definimos Join (inner_join en R) que se compone de un producto cruz y un filtro.



Supongamos que tenemos dos relaciones R_1 y R_2 con un atributo llamado 'key'. Abusando de nuestra notación podemos definir:

$$inner_join(R_1, R_2, by = "key") = filter(crossing(R_1, R_2), R_1.key == R_2.key)$$

Notese que el comando de la derecha no funciona en R ¿por qué?



Join

Lo ideal es que el join se realice sobre una llave (foranea), sin embargo se puede realizar con atributos que no generen una llave.



Gracias a esta definición podemos escribir las consultas de manera más simple:

- Liste todos los id de los actores de la película “Interstellar”

```
filter(inner_join(actuo_en, peliculas, c('id_p' = ' id')), id == 1)
```



Gracias a esta definición podemos escribir las consultas de manera más simple:

- Liste todos los id de los actores de la película “Interstellar”

```
filter(inner_join(actuo_en, peliculas, c('id_p' = 'id')), id == 1)
```

- Liste cada actor junto a todas las películas en las que ha actuado

```
db1 <- inner_join(actuo_en, P, c('id_p' = 'id'))
```

```
db1.rename(c('Nombre' = 'NombrePelicula'))
```

```
db <- inner_join(db1, A, c('id_d' = 'id'))
```

```
select(db, c('NombrePelicula', 'Nombre'))
```



Join

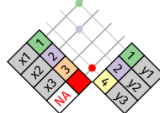
- Una unión izquierda (`left_join`) mantiene todas las observaciones en x .
- Una unión derecha (`right_join`) mantiene todas las observaciones en y .
- Una unión completa (`full_join`) mantiene todas las observaciones en x e y .

Izquierda



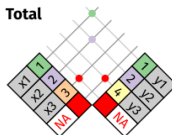
llave	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	NA

Derecha



llave	val_x	val_y
1	x1	y1
2	x2	y2
4	NA	y3

Total

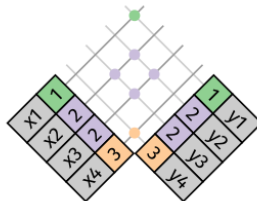


llave	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	NA
4	NA	y3



Join

Con las claves duplicadas (en el caso que un atributo no sea llave) se mantienen todas las combinaciones posibles en la base de datos resultante.



llave	val_x	val_y
1	x1	y1
2	x2	y2
2	x2	y3
2	x3	y2
2	x3	y3
3	x4	y4



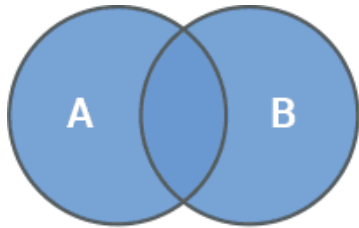
Intersección y diferencia

Sean las relaciones R_1 , R_2 ambas con los mismos atributos, puede calcularse la intersección y diferencia entre estas mediante las funciones:

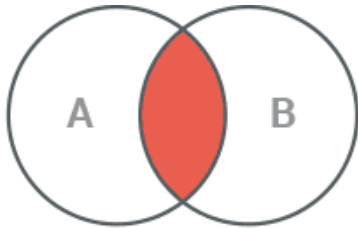
- `intersect(x, y)` : devuelve las observaciones comunes en x e y .
- `setdiff(x, y)`: (diferencia de conjuntos) devuelve las observaciones en x pero no en y .



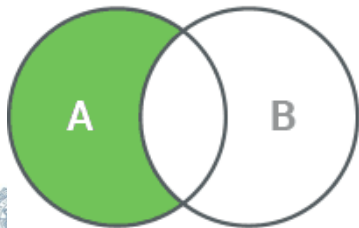
Intersección y diferencia



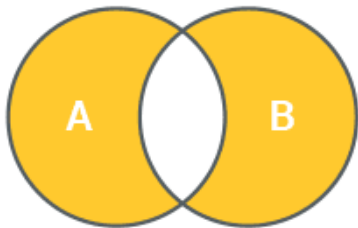
Union



Intersection



Difference

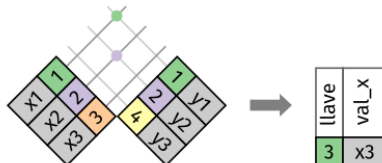
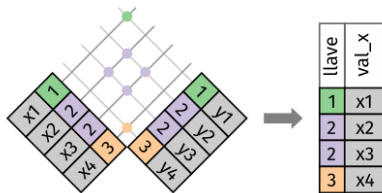


Symmetric Difference

Intersección y diferencia

Con un efecto parecido también pueden encontrarse las uniones de filtro. Tienen un funcionamiento similar a los Join usuales, sin embargo se ven afectadas las observaciones de la primera relación y no las variables.

- `semi_join(x, y)` mantiene todas las observaciones en `x` con coincidencias en `y`.
- `anti_join(x, y)` descarta todas las observaciones en `x` con coincidencias en `y`.



Supongamos que queremos la siguiente consulta:

Liste el nombre de todos los actores dirigidos por C. Nolan y no por A. Iñárritu

¿Como lo haríamos?



Intersección y diferencia

- Primero, crear bases de datos con información cruzada entre películas y actores:

```
pelis <- inner_join(peliculas, actores, c('id' == 'id_pelicula'))
```

- Crear dos bases con las películas de Nolan y de Iñárritu

```
pelis_nolan <- filter(pelis, director == 'C.Nolan')
```

```
pelis_inarritu <- filter(pelis, director == 'A.Inarritu')
```

- Calcular la intersección en caso de que se quieran los actores dirigidos por ambos o la diferencia, que serían los actores dirigidos por uno pero no por el otro

```
select(pelis_nolan, c("id_actor")) - select(pelis_inarritu, c("id_actor"))
```



Tabla de Contenidos

Introducción

Operadores de bases de datos

Otros operadores (de R)

Pivotar

Separar y unir



Al enfrentarse a un set de datos es importante:

1. Entender cuales son las observaciones y cuales las variables



Al enfrentarse a un set de datos es importante:

1. Entender cuales son las observaciones y cuales las variables
2. Resolver los problemas:



Al enfrentarse a un set de datos es importante:

1. Entender cuales son las observaciones y cuales las variables
2. Resolver los problemas:
 - Una variable se extiende por varias columnas



Al enfrentarse a un set de datos es importante:

1. Entender cuales son las observaciones y cuales las variables
2. Resolver los problemas:
 - Una variable se extiende por varias columnas
 - Una observación está dispersa entre múltiples filas.



El primer caso lo resolvemos con la función *pivot_longer*.

pais	anio	casos
Afganistán	1999	745
Afganistán	2000	2666
Brasil	1999	37737
Brasil	2000	80488
China	1999	212258
China	2000	213766

pais	1999	2000
Afganistán	745	2666
Brasil	37737	80488
China	212258	213766

Tabla 4

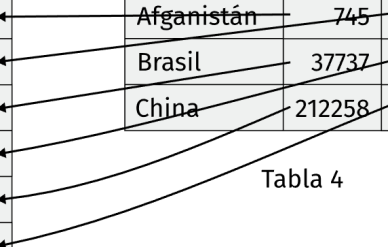


Figura 2:



El el segundo lo resolvemos con la función *pivot_wide*.

pais	anio	tipo	casos
Afganistán	1999	casos	745
Afganistán	1999	población	19987071
Afganistán	2000	casos	2666
Afganistán	2000	población	20595360
Brasil	1999	casos	37737
Brasil	1999	población	172006362
Brasil	2000	casos	80488
Brasil	2000	población	174504898
China	1999	casos	212258
China	1999	población	1272915272
China	2000	casos	213766
China	2000	población	1280428583

pais	anio	casos	poblacion
Afganistán	1999	745	19987071
Afganistán	2000	2666	20595360
Brasil	1999	37737	172006362
Brasil	2000	80488	17504898
China	1999	212258	1272915272
China	2000	213766	1280428583

Tabla 2

Figura 3:



Separar y unir

Puede ser que alguna de nuestras columnas tenga mas de un dato y por esto nos conviene separarlo en dos o mas variables. Esto lo logramos con la función *separate*, su inverso es la funcion *unite*



pais	anio	tasa
Afganistán	1999	745 / 19987071
Afganistán	2000	2666 / 20595360
Brasil	1999	37737 / 172006362
Brasil	2000	80488 / 17504898
China	1999	212258 / 1272915272
China	2000	213766 / 1280428583

Tabla 3

pais	anio	casos	poblacion
Afganistán	1999	745	19987071
Afganistán	2000	2666	20595360
Brasil	1999	37737	172006362
Brasil	2000	80488	17504898
China	1999	212258	1272915272
China	2000	213766	1280428583

Figura 4:



Valores faltantes

Cambiar la forma de representar los datos puede llevarnos a descubrir una mayor cantidad de valores faltantes o NA (Not Available).

Existen dos formas en las que puede aparecer un valor NA:

- Explícita, esto es, aparece como NA.
- Implícita, esto es, simplemente no aparece en los datos.

anio	trimestre	retorno
<dbl>	<dbl>	<dbl>
2015	1	1.88
2015	2	0.59
2015	3	0.35
2015	4	NA
2016	2	0.92
2016	3	0.17
2016	4	2.66

Figura 5: Tabla con valores faltantes

