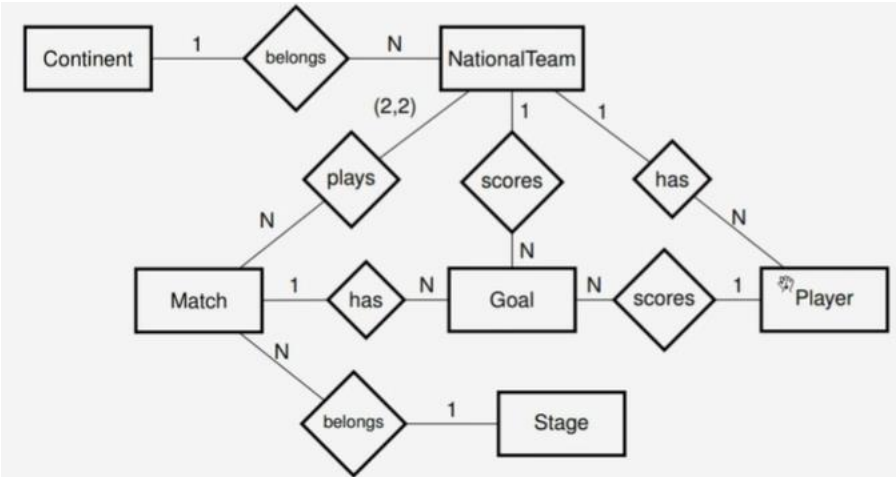
ALGEBRA RELACIONAL

- **Qué es:** es un lenguaje procedural: indica un procedimiento a seguir utilizando operaciones que indican cómo manipular datos.
- **Otros conceptos:**
  - ✓ *Aridad:* cantidad de operandos que toma una operación (nosotros vamos a ver 1 y 2)
  - ✓ *Expresión:* combinación de operaciones del algebra relacional

para practicas las operaciones básicas vamos a usar la herramienta ReLax:

es un gestor de juguete para ver los resultados que obtenemos. En la teórica usamos un git precargado de Mundial, con el siguiente modelo:



Esquema de base de datos relacional:

- Continents(id, name)
  (3, 'Europe')
- NationalTeams(short_name, name, group, continent)
  ('ESP', 'Spain', 'E', 3)
- Matches(id, home, away, datetime, stage)
  (23, 'ARG', 'MEX', '2022-11-26 16:00:00', 1)
- Players(id, name, birth_year, playing_position, local_club, national_team)
  (184, 'Emiliano Martínez', 1992, 'GK', 'Aston Villa', 'ARG')
- Scores(id, match_id, team_id, player_id, minute, score_type)
  (1, 1, 1, 8, 55, 1) (Corresponde a la entidad Goal).
- Stages(id, name)
  (3, 'Quarter-final')
- Asumiremos que "name" es siempre clave candidata.
- Los tipos de score son: 1-normal; 2-penal; 3-gol en contra (se asigna al equipo contrario); 4-gol en serie de penales.

Operaciones Básicas (Unarias)

Para estas operaciones, puedo usar *únicamente* una sola relación R

Selección: σ

- $\sigma_{cond}(R)$ selecciona las tuplas (filas) en R (tabla) para las cuales la condición es verdadera
- se escribe como sigma, la condición, y la tabla a la cual le pido eso.
- Es quedarse con solo algunas filas de la relación.

NationalTeams

short_name	name	group	continent
ARG	Argentina	C	5
AUS	Australia	D	2
BEL	Belgium	F	3
BRA	Brazil	G	5
CMR	Cameroon	G	1

$\downarrow \sigma_{group='G'}(\text{NationalTeams})$

short_name	name	group	continent
BRA	Brazil	G	5
CMR	Cameroon	G	1

Para armar las condiciones usamos operadores de comparación:

- =, ≠, >, <, ≤, ≥
- Las combinamos con *and* ^, *or* v *y not* ¬
- Dentro de una cond, NO puedo poner otras relaciones; solo valores fijos, osea comparo con atributos de la tabla

Ejemplo: Seleccionar aquellos jugadores del mundial que pertenecen al club local "Barcelona" y que nacieron

Usando relax, obtenemos:

```
σlocal_club='Barcelona' ∧ birth_year <2000 (Players)
```

▶ execute query   Download   History

σ local_club = 'Barcelona' and birth_year < 2000 (Players)

Execution time: 0 ms

Proyección: π

- Dada una relación (tabla) y una lista de atributos (que pertenecen a la tabla), devuelve una relación cuyas tuplas representan los posibles valores de los atributos de la lista L en R
- Siempre remueve tuplas duplicadas. Así que puede ser que devuelva menos filas. En SQL para que haga eso se lo tengo que decir explícitamente
- Si la selección me dice quédate con solo algunas filas de la relación, la proyección me dice quédate solo con algunas columnas. Selecciono con que columnas me quedo y el resto se descartan
- *Notacion:* π *L(R)*
- Si quiero más columnas, separo con comas.

  Ejemplo 1:

ArgPlayers

id	name	local_club
551	Rodrigo De Paul	Atlético Madrid
615	Thiago Almada	Atlanta Utd
674	Ángel Correa	Atlético Madrid
675	Ángel Di María	Juventus

$\xrightarrow{\pi_{local\_club}(\text{ArgPlayers})}$

local_club
Atlético Madrid
Atlanda Utd
Juventus

Ejemplo 2: si yo solo quiero el NOMBRE de los jugadores cuyo club es Barcelona y nacieron antes del 2000, lo que tengo que hacer es envolver la selección en la proyección:

π name (σ local_club = 'Barcelona' and birth_year < 2000 (Players))

Asignacion: ←

- ¿Cómo listamos los nombres de los países del grupo G?

NationalTeams

short_name	name	group	continent
AUS	Australia	D	2
BEL	Belgium	F	3
BRA	Brazil	G	5
CMR	Cameroon	G	1
...	...	...	...

$$Temp \leftarrow \sigma_{group='G'}(\text{NationalTeams})$$
$$Selecciones\_GrupoG \leftarrow \pi_{name}(\text{Temp})$$

- Podemos también hacerlo en un único paso:

$$Selecciones\_GrupoG \leftarrow \pi_{name}(\sigma_{group='G'}(\text{NationalTeams}))$$

Selecciones_GrupoG

name
Brazil
Cameroon
...

- Se puede usar para definir relaciones temporales. Entonces, si yo quiero enlazar varios operadores no queda un choclo ilegible. Puedo usar el operados = también.
- *Ejemplo:*

```
BARCA_2000 ← σ local_club='Barcelona' ∧ birth_year <2000 (Players)
πname,id(BARCA_2000)
```

Redenominación: ρ

- permite modificar los nombres de los atributos de una relación y/o el de la relación misma.
- Nos permite preparar el resultado para la realización de una
- operación posterior.
- $\rho_{S(B1,B2,...,Bn)}$(R) produce una relación de nombre S y atributos (B1,B2, ...,Bn) cuyas tuplas coinciden con las tuplas de R.
- ρ_S(R) solo cambia el nombre de la relacion R por S.

ArgPlayers

name	local_club
Rodrigo De Paul	Atlético Madrid
Thiago Almada	Atlanta Utd
Ángel Correa	Atlético Madrid
Ángel Di María	Juventus

$$\xrightarrow{\rho_{Argentinos(nombre,club\_local)}(\text{ArgPlayers})}$$

Argentinos

nombre	club_local
Rodrigo De Paul	Atlético Madrid
Thiago Almada	Atlanta Utd
Ángel Correa	Atlético Madrid
Ángel Di María	Juventus

πname,id(BARCA_2000) esto me va a devolver atributos del tipo Players.name y Players.id. Si yo quisiera renombrar ese Players, hago `ρNombres(πname,id(BARCA_2000))` y va a devolver:

Nombres.name	Nombres.id
'Andreas Christensen'	49
'Frenkie de Jong'	212
'Jordi Alba'	312
'Jules Koundé'	329
'Memphis'	440

Operaciones compuestas (Binarias)

Ahora voy a poder trabajar con más de una Relación.

Operaciones de conjuntos

Unión : ∪

- la unión de dos Relaciones es una relación que contiene todas las tuplas de R1 y R2
- para que sea posible ambas tienen que tener
  - ✓ El mismo ***grado:***
  - ✓ Deben coincidir sus atributos (compatibilidad de unión/tipo)

Normal_Scorers

id	name
364	Kim Young-gwon
372	Kudus Mohammed
377	Kylian Mbappé
388	Lionel Messi

Penalty_Scorers

id	name
269	Ismaila Sarr
377	Kylian Mbappé
388	Lionel Messi

Normal_Scorers ∪ Penalty_Scorers →

id	name
269	Ismaila Sarr
364	Kim Young-gwon
372	Kudus Mohammed
377	Kylian Mbappé
388	Lionel Messi

Interseccion : ∩

- la intersección de dos Relaciones es una relación que contiene todas las tuplas que están tanto en R1 como en R2
- para que sea posible ambas tienen que tener
  - ✓ El mismo ***grado:***
  - ✓ Deben coincidir sus atributos (compatibilidad de tipo)

Normal_Scorers

id	name
364	Kim Young-gwon
372	Kudus Mohammed
377	Kylian Mbappé
388	Lionel Messi

Penalty_Scorers

id	name
269	Ismaila Sarr
377	Kylian Mbappé
388	Lionel Messi

Normal_Scorers ∩ Penalty_Scorers →

id	name
377	Kylian Mbappé
388	Lionel Messi

```
Normal_scorer = πplayer_id(σscore_type=1(Scores))
Penalty_scorer = πplayer_id(σscore_type=2(Scores))

Normal_scorer ∪ Penalty_scorer
```

Después vemos como hacerlo con el nombre. Por ahora nos va a devolver los player_id

Diferencia : -

- La diferencia R1 - R2 conserva solo las tuplas de R1 que NO estan en R2
- para que sea posible ambas tienen que tener
  - ✓ El mismo ***grado:***
  - ✓ Deben coincidir sus atributos (compatibilidad de tipo)

Normal_Scorers

id	name
364	Kim Young-gwon
372	Kudus Mohammed
377	Kylian Mbappé
388	Lionel Messi

Penalty_Scorers

id	name
269	Ismaila Sarr
377	Kylian Mbappé
388	Lionel Messi

Normal_Scorers − Penalty_Scorers →

id	name
364	Kim Young-gwon
372	Kudus Mohammed

Son los que hicieron goles normales y NO hicieron goles de penal.

Producto Cartesiano

- RxS produce una nueva relación T donde:
- Están todos los atributos de R + todos los atributos de S.
  El problema está cuando algún atributo de R tiene el mismo nombre que uno de S. En ese caso, la convención es que en el resultado de los atributos se llaman R.A1 y S.B1. Si estoy haciendo RxR se nombran R1.Ai y R2.Ai
- para que sea posible no hace falta compatibilidad de tipos. Entonces, me sirve para combinar cosas de Relaciones distintas
- RxS es una nueva relación donde cada tupla de R aparece cn todas las tuplas de S, y devuelve todos los atributos de R y todos los de S.

NationalTeams

short_name	name	group
ARG	Argentina	C
AUS	Australia	D
BEL	Belgium	F
BRA	Brazil	G
...	...	...

Players

id	name	national_team
227	Gonzalo Montiel	ARG
353	Kevin De Bruyne	BEL
...	...	...

$\downarrow$ NationalTeams $\times$ Players

short_name	NationalTeams.name	group	id	Players.name	national_team
ARG	Argentina	C	227	Gonzalo Montiel	ARG
ARG	Argentina	C	353	Kevin De Bruyne	BEL
AUS	Australia	D	227	Gonzalo Montiel	ARG
AUS	Australia	D	353	Kevin De Bruyne	BEL
BEL	Belgium	F	227	Gonzalo Montiel	ARG
BEL	Belgium	F	353	Kevin De Bruyne	BEL
BRA	Brazil	G	227	Gonzalo Montiel	ARG
BRA	Brazil	G	353	Kevin De Bruyne	BEL
...	...	...	...	...	...

Es útil para mezclar cosas de != relaciones en una. **Se hace mucho más útil cuando tenemos una selección.** En general vinculamos CP de una tabla con CF de otra.

Ejemplo 1: ¿cómo obtenemos las tuplas que representan la pertenencia de un jugador a una selección nacional?

$\downarrow \sigma_{short\_name=national\_team}(\text{NationalTeams} \times \text{Players})$

short_name	NationalTeams.name	group	id	Players.name	national_team
ARG	Argentina	C	227	Gonzalo Montiel	ARG
BEL	Belgium	F	353	Kevin De Bruyne	BEL
...	...	...	...	...	...

Ejemplo 2:

> Ejemplo: World Cup 2022
>
> Liste los nombres de países a los que Lionel Messi les convirtió un gol en tiempo reglamentario.

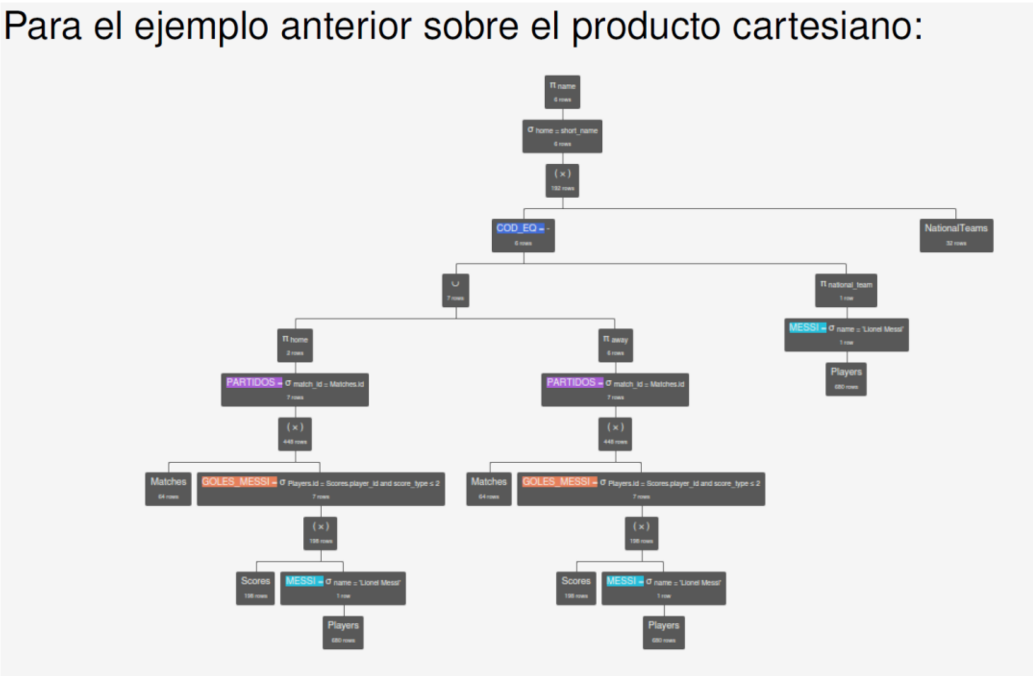
- Tienen que ser goles de tipo 1 y 2
- Tienen que ser goles de Lionel Messi (pero no tengo el id, sino el nombre)
- Tengo que buscar los paises a los que se le hicieron ese gol.

```
MESSI = (σPlayers.name = 'Lionel Messi' (Players))

GOLES_DE_MESSI = σ(Players.id=Scores.player_id)(MESSI ⨯ Scores)

/*
filtrar que no halla hecho goles en tiempo reglamentario
*/
REGLAMENTARIOS_MESSI = σscore_type=1∨score_type=2(GOLES_DE_MESSI)
/*
ahora busco los paises a los que les hizo gol
*/
PARTIDOS =σ(Scores.match_id = Matches.id)(REGLAMENTARIOS_MESSI ⨯ Matches)
PARTIDOS

/*
Ahora busco los valores de paises que NO son ARG. el tema es que a veces
ARG es Home y a veces es Away. Para quedarme con 1 sola col, uso una UNION
quiero los away que son != ARG o los home que son != away
*/
Ids_partidos = (πhome(PARTIDOS)) ∪ (πaway(PARTIDOS)) - πnational_team(MESSI)

/*
ahora hago este paso para que devuelva el nombre del pais y no el id
*/
σ(NationalTeams.short_name = home)(Ids_partidos ⨯ NationalTeams)
```

La idea del producto cartesiano es pensar el producto cartesiano y despues antecederlo por una selección para que queden las filas que correspondan unas con otras (ej: que el nombre del NationalTeam sea el nombre de home)

Árboles de consulta

Para cada expresión del álgebra relacional se puede construir un árbol de consulta que representa el orden de ejecución.



Junta

- Combina un producto cartesiano con una selección en un solo paso

- Dadas dos relaciones R(A1,A2, ...,An) y S(B1,B2, ...,Bm) y una condición, la junta $R \bowtie_{cond} S$ selecciona del producto cartesiano R × S las tuplas que cumplen la condición.
- No se admite cualquier tipo de condición de selección; solo las de tipo:

  - $A_i \odot B_j$

  En donde $\odot$ debe ser un operador de comparación:

  - $=, \neq$
  - $>, \geq, <, \leq$ (sólo para atributos cuyos dominios están ordenados)

  Una condición se construye entonces combinando operaciones atómicas con el operador lógico `and` ($\wedge$).

- Ahora la combinación de NationalTeams y Players se hace mucho más sencilla:

NationalTeams

short_name	name	group
ARG	Argentina	C
AUS	Australia	D
BEL	Belgium	F
BRA	Brazil	G
...	...	...

Players

id	name	national_team
227	Gonzalo Montiel	ARG
353	Kevin De Bruyne	BEL
...	...	...

$\downarrow$ (NationalTeams $\bowtie_{short\_name=national\_team}$ Players)

short_name	NationalTeams.name	group	id	Players.name	national_team
ARG	Argentina	C	227	Gonzalo Montiel	ARG
BEL	Belgium	F	353	Kevin De Bruyne	BEL
...	...	...	...	...	...

Por ejemplo, para la selección + producto cartesiano que teníamos del ejemplo anterior:

```
GOLES_DE_MESSI = σ(Players.id=Scores.player_id)(MESSI × Scores)
```

Podemos cambiarla por:

```
GOLES_DE_MESSI = ((MESSI)⋈Players.id = Scores.player_id (Scores))
```

Messi join Scores con la condición de que tengan el mismo Id.

Hay algunos tipos particulares de junta.

- ✓ El caso más general de operación de junta también se denomina junta theta (theta join).
- ✓ Junta por igual (equijoin): Cuando la junta sólo utiliza comparaciones de igualdad en sus condiciones atómicas. el resultado dispondrá de pares de atributos distintos que poseerán información redundante
- ✓ Junta natural: se define para liberarse de los datos redundantes de la junta por igual. Los atributos deben tener el mismo nombre para poder hacerlo.

Para realizar una junta natural entre dos relaciones en reemplazo de una junta por igual, las mismas deben estar preparadas de manera que los pares de atributos (Ai ,Bj) de cada condición atómica tengan el mismo nombre en una y otra relación. El resultado dispondrá de uno sólo de los atributos, conservando su nombre.

La junta natural entre dos relaciones R y S se simboliza R ∗ S.

¡Atención! En la junta natural no se especifican las condiciones, por lo tanto todo par de atributos de igual nombre en una y otra relación será comparado por igual en la condición de selección implícita.

Los atributos comparados en una junta se denominan atributos de junta.

Ejemplo: RENAPER

Personas(DNI, nombre, género, fecha_nacimiento)
HijoDe(DNI_padre, DNI_hijo)
CasadaCon(DNI1, DNI2, fecha_matrimonio)

Liste a todos los hijos de "Abraham Simpson" (suponga que no hay dos personas con ese nombre).

Respuesta

$PADRE \leftarrow \rho_{DNI\_padre}(\pi_{DNI}(\sigma_{nombre=\text{"Abraham Simpson"}}(Personas)))$
$HIJOS \leftarrow \rho_{DNI\_hijo,nombre}(\pi_{DNI,nombre}(Personas))$
$\pi_{DNI\_hijo,nombre}$ (PADRE * HijoDe * HIJOS)

División

- Nos interesa saber qué alumnos aprobaron los 3 TPs.

NOTAS

alumno	TP	nota
Pedro	1	7
Pedro	3	2
Juan	1	3
Juan	2	6
Juan	3	8
Walter	1	4
Walter	2	9
Walter	3	8

→

APROBADOS

alumno	TP
Pedro	1
Juan	2
Juan	3
Walter	1
Walter	2
Walter	3

REQUISITOS

TP
1
2
3

↓ (Aprobados ÷ Requisitos)

alumno
Walter

- Es una operación inversa al producto cartesiano
- Llamamos Y = A – B con A = conjunto atributos de R y B conjunto de atributos de S
- La división $R \div S$ es la relación T(Y) que contiene todas las tuplas *t* que cumplen que
  - ✓ t pertenece a πY (R).
  - ✓ Para cada tupla $t_S \in S$ existe una tupla $t_R \in R$ tal que $t_R[Y] = t$ y $t_R[B] = t_S$.
- Propiedad: T es la relación de mayor cardinalidad posible contenida en π_Y (R) y que cumple que $T * S \subset R$.

S % T

El dividendo debe incluir el divisor

O sea: S[A1..An B1..Bm] y T[B1..Bm]

La división muestra como resultado los elementos del conjunto A que se relacionan **con todos** los elementos del conjunto B.

Los atributos del dividendo y divisor deben estar ordenados. Para el dividendo: 1ro todos los atributos únicos de el mismo (osea las As) y luego todos los atributos que coinciden con los de T

La división devuelve todos los valores del lado izquierdo que estén vinculados con los valores del lado derecho. Acá los valores del lado izquierdo son los nombres de los Alumnos. Devuelve solo Walter porque es el que esta vinculado con todos los de la derecha. Si a la derecha tuviera más columnas, tienen que matchear todas.

Ejemplo: Tenistas

Tenistas(nombre_tenista, país, altura, diestro)
(‘Novak Djokovic’, ‘Serbia’, 1.88, True)
Torneos(nombre_torneo, tipo_torneo)
(‘Abierto de Australia’, ‘Grand Slam’)
Campeones(nombre_tenista, nombre_torneo, modalidad, año)
(‘Juan Martín del Potro’, ‘Torneo de Estocolmo’, ‘Single’, 2016)

Liste a aquellos tenistas que hayan ganado todos los torneos de tipo “Grand Slam” existentes al menos una vez.

Respuesta

$TORNEOS\_GRAND\_SLAM \leftarrow \pi_{nombre\_torneo}(\sigma_{tipo\_torneo=\text{“Grand Slam”}}(Torneos))$
$\pi_{nombre\_tenista,nombre\_torneo}(Campeones) \div TORNEOS\_GRAND\_SLAM$

Conjuntos completos de operadores

- Hemos definido una serie de operadores basicos del algebra relacional: σ, π, ρ,∪, ∩,−,×, ▷◁, ∗,÷.
- Sin embargo, existen subconjuntos de ellos que tienen la misma capacidad de expresion que todo el conjunto.
- A dichos subconjuntos se los denomina conjuntos completos de operadores.
- {σ, π, ρ,∪,−, ×} forman un conjunto completo de operadores.