# Bases de Datos I

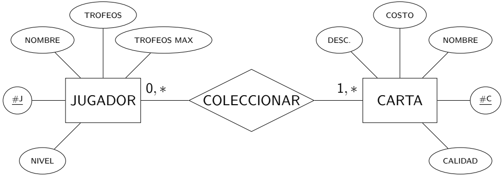
## Modelo relacional: diseño intuitivo y lenguajes de consulta

Lic. Andy Ledesma García
Lic. Víctor M. Cardentey Fundora
Dra. C. Lucina García Hernández

Departamento de Computación
Facultad de Matemática y Computación
Universidad de La Habana

21 de mayo de 2024

# ¿Dónde nos quedamos?




**Jugador**(#J, Nombre, Nivel, Trofeos, TrofeosMax)

**Carta**(#C, Nombre, Calidad, Desc., Costo)

**Coleccionar**(#J, #C)
FK: #J REFERENCES Jugador
FK: #C REFERENCES Carta

# ¿Para qué están aquí?

## Objetivos para la conferencia

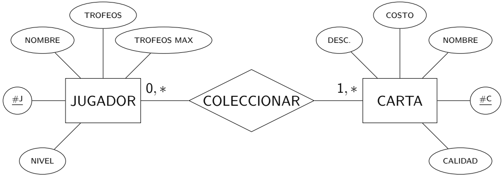
1. Poder transformar un diseño conceptual descrito por el modelo MERX en un diseño lógico basado en el modelo Relacional.
2. Poder utilizar el diseño lógico obtenido para responder las preguntas del usuario.

# Diseño intuitivo


DISEÑO INTUITIVO
IF
IF
IF
IF
imgflip.com

# Algoritmo del diseño intuitivo

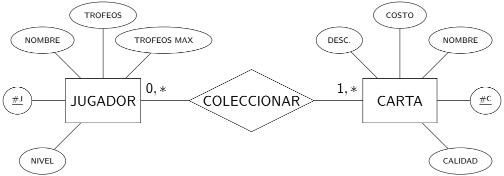
# Diseño intuitivo



# Diseño intuitivo

## La idea básica

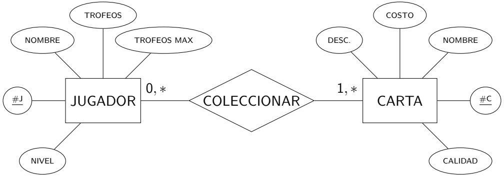
1. Convertir cada conjunto de entidades en una relación con el mismo conjunto de atributos.



# Diseño intuitivo

## La idea básica

1. Convertir cada conjunto de entidades en una relación con el mismo conjunto de atributos.



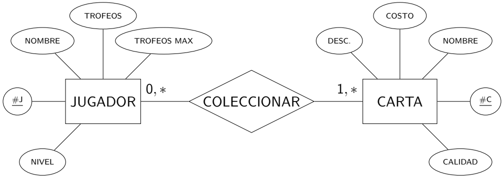

**Jugador**(#J, Nombre, Nivel, Trofeos, TrofeosMax)

**Carta**(#C, Nombre, Calidad, Desc., Costo)

# Diseño intuitivo

## La idea básica

1. Convertir cada conjunto de entidades en una relación con el mismo conjunto de atributos.
2. Convertir cada interrelación en una relación cuyos atributos son las llaves primarias de las relaciones que representan los conjuntos de entidades conectados.



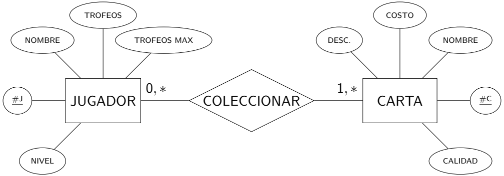

**Jugador**(#J, Nombre, Nivel, Trofeos, TrofeosMax)

**Carta**(#C, Nombre, Calidad, Desc., Costo)

# Diseño intuitivo

## La idea básica

1. Convertir cada conjunto de entidades en una relación con el mismo conjunto de atributos.
2. Convertir cada interrelación en una relación cuyos atributos son las llaves primarias de las relaciones que representan los conjuntos de entidades conectados.



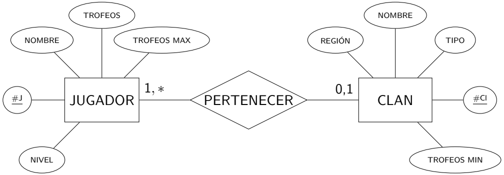

**Jugador**(<u>#J</u>, Nombre, Nivel, Trofeos, TrofeosMax)

**Carta**(<u>#C</u>, Nombre, Calidad, Desc., Costo)

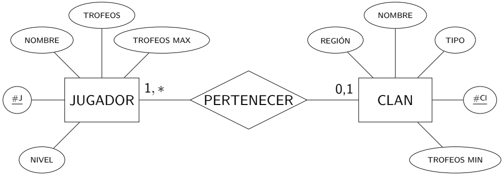
**Coleccionar**(<u>#J</u>, <u>#C</u>)

FK: #J REFERENCES Jugador
FK: #C REFERENCES Carta

# Aplicando el algoritmo



# Aplicando el algoritmo




**Jugador**(#J, Nombre, Nivel, Trofeos, TrofeosMax)

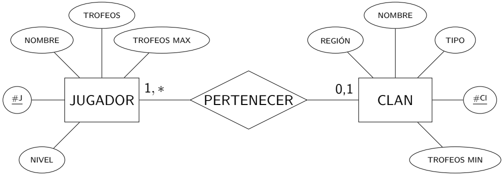
**Clan**(#Cl, Nombre, Región, Tipo, TrofeosMin)

**Pertenecer**(#J, #Cl)

FK: #J REFERENCES Jugador

FK: #Cl REFERENCES Clan

# Aplicando el algoritmo




**Jugador**(#J, Nombre, Nivel, Trofeos, TrofeosMax)

**Clan**(#Cl, Nombre, Región, Tipo, TrofeosMin)

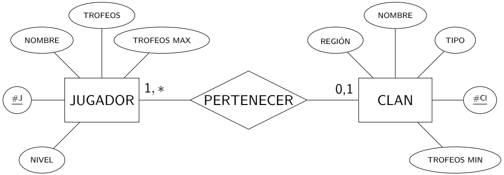
**Pertenecer**(#J, #Cl)

FK: #J REFERENCES Jugador

FK: #Cl REFERENCES Clan

¿Este diseño es eficiente?

# Un caso especial




**Jugador**(#J, Nombre, Nivel, Trofeos, TrofeosMax, #Cl)
FK: #Cl REFERENCES Clan HAS NULL

**Clan**(#Cl, Nombre, Región, Tipo, TrofeosMin)

Añadir la llave primaria de la relación correspondiente al conjunto de entidades en el extremo con cardinalidad máxima uno a la relación correspondiente al conjunto de entidades en el otro extremo.

# Diseños para interrelaciones de muchos a uno

**Jugador**(#J, Nombre, Nivel, Trofeos, TrofeosMax)

**Clan**(#Cl, Nombre, Región, Tipo, TrofeosMin)

**Pertenecer**(#J, #Cl)

FK: #J REFERENCES Jugador

FK: #Cl REFERENCES Clan

**Jugador**(#J, Nombre, Nivel, Trofeos, TrofeosMax, #Cl)

FK: #Cl REFERENCES Clan HAS NULL

**Clan**(#Cl, Nombre, Región, Tipo, TrofeosMin)

# Diseños para interrelaciones de muchos a uno

## Comparemos estos enfoques

Supongamos que:

- Cada identificador es un entero de 64 bits y el indicador NULL también ocupa 64 bits.
- Se tienen datos de 2 millones de jugadores en la base de datos.
- Se tiene que 500 mil jugadores pertenecen a un clan.

**Jugador**(<u>#J</u>, Nombre, Nivel, Trofeos, TrofeosMax)

**Clan**(<u>#Cl</u>, Nombre, Región, Tipo, TrofeosMin)

**Pertenecer**(<u>#J</u>, <u>#Cl</u>)
FK: #J REFERENCES Jugador
FK: #Cl REFERENCES Clan

**Jugador**(<u>#J</u>, Nombre, Nivel, Trofeos, TrofeosMax, #Cl)
FK: #Cl REFERENCES Clan HAS NULL

**Clan**(<u>#Cl</u>, Nombre, Región, Tipo, TrofeosMin)

# Diseños para interrelaciones de muchos a uno

## Comparemos estos enfoques

Supongamos que:

- Cada identificador es un entero de 64 bits y el indicador NULL también ocupa 64 bits.
- Se tienen datos de 2 millones de jugadores en la base de datos.
- Se tiene que 500 mil jugadores pertenecen a un clan.

**Jugador**($\underline{\#J}$, Nombre, Nivel, Trofeos, TrofeosMax)

**Clan**($\underline{\#Cl}$, Nombre, Región, Tipo, TrofeosMin)

**Pertenecer**($\underline{\#J}$, $\underline{\#Cl}$)

FK: #J REFERENCES Jugador

FK: #Cl REFERENCES Clan

$128 \times 5 \times 10^5$ bits $=$ 8 Megabytes

**Jugador**($\underline{\#J}$, Nombre, Nivel, Trofeos, TrofeosMax, #Cl)

FK: #Cl REFERENCES Clan HAS NULL

**Clan**($\underline{\#Cl}$, Nombre, Región, Tipo, TrofeosMin)

$64 \times 2 \times 10^6$ bits $=$ 16 Megabytes

# Diseños para interrelaciones de muchos a uno

## Comparemos estos enfoques

Supongamos que:

- Cada identificador es un entero de 64 bits y el indicador NULL también ocupa 64 bits.
- Se tienen datos de 2 millones de jugadores en la base de datos.
- Se tiene que 1.5 millones de jugadores pertenecen a un clan.

**Jugador**(#J, Nombre, Nivel, Trofeos, TrofeosMax)

**Clan**(#Cl, Nombre, Región, Tipo, TrofeosMin)

**Pertenecer**(#J, #Cl)
FK: #J REFERENCES Jugador
FK: #Cl REFERENCES Clan

**Jugador**(#J, Nombre, Nivel, Trofeos, TrofeosMax, #Cl)
FK: #Cl REFERENCES Clan HAS NULL

**Clan**(#Cl, Nombre, Región, Tipo, TrofeosMin)

# Diseños para interrelaciones de muchos a uno

## Comparemos estos enfoques

Supongamos que:

- Cada identificador es un entero de 64 bits y el indicador NULL también ocupa 64 bits.
- Se tienen datos de 2 millones de jugadores en la base de datos.
- Se tiene que 1.5 millones de jugadores pertenecen a un clan.

**Jugador**(<u>#J</u>, Nombre, Nivel, Trofeos, TrofeosMax)

**Clan**(<u>#Cl</u>, Nombre, Región, Tipo, TrofeosMin)

**Pertenecer**(<u>#J</u>, <u>#Cl</u>)
  FK: #J REFERENCES Jugador
  FK: #Cl REFERENCES Clan

$128 \times 1.5 \times 10^6$ bits $= 24$ Megabytes

**Jugador**(<u>#J</u>, Nombre, Nivel, Trofeos, TrofeosMax, #Cl)
  FK: #Cl REFERENCES Clan HAS NULL

**Clan**(<u>#Cl</u>, Nombre, Región, Tipo, TrofeosMin)

$64 \times 2 \times 10^6$ bits $= 16$ Megabytes

# Diseños para interrelaciones de muchos a uno

**Jugador**(#J, Nombre, Nivel, Trofeos, TrofeosMax)

**Clan**(#Cl, Nombre, Región, Tipo, TrofeosMin)

**Pertenecer**(#J, #Cl)

FK: #J REFERENCES Jugador

FK: #Cl REFERENCES Clan

**Jugador**(#J, Nombre, Nivel, Trofeos, TrofeosMax, #Cl)

FK: #Cl REFERENCES Clan HAS NULL

**Clan**(#Cl, Nombre, Región, Tipo, TrofeosMin)

# Diseños para interrelaciones de muchos a uno

**Jugador**(<u>#J</u>, Nombre, Nivel, Trofeos, TrofeosMax)

**Clan**(<u>#Cl</u>, Nombre, Región, Tipo, TrofeosMin)

**Pertenecer**(<u>#J</u>, <u>#Cl</u>)
  FK: #J REFERENCES Jugador
  FK: #Cl REFERENCES Clan

- Más eficiente espacialmente cuando la multiplicidad de la interrelación es pequeña.
- Menos eficiente para realizar operaciones sobre los datos

**Jugador**(<u>#J</u>, Nombre, Nivel, Trofeos, TrofeosMax, #Cl)
  FK: #Cl REFERENCES Clan HAS NULL

**Clan**(<u>#Cl</u>, Nombre, Región, Tipo, TrofeosMin)

# Diseños para interrelaciones de muchos a uno

**Jugador**(#J, Nombre, Nivel, Trofeos, TrofeosMax)

**Clan**(#Cl, Nombre, Región, Tipo, TrofeosMin)

**Pertenecer**(#J, #Cl)
FK: #J REFERENCES Jugador
FK: #Cl REFERENCES Clan

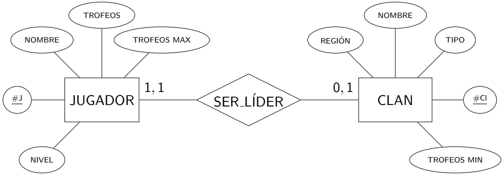
- Más eficiente espacialmente cuando la multiplicidad de la interrelación es pequeña.
- Menos eficiente para realizar operaciones sobre los datos

**Jugador**(#J, Nombre, Nivel, Trofeos, TrofeosMax, #Cl)
FK: #Cl REFERENCES Clan HAS NULL

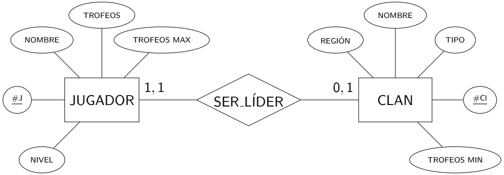
**Clan**(#Cl, Nombre, Región, Tipo, TrofeosMin)

- Más eficiente espacialmente cuando la multiplicidad de la interrelación es grande.
- Más eficiente para realizar operaciones sobre los datos

# El caso especial del caso especial



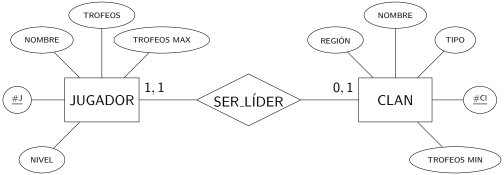
# Diseño para interrelaciones uno a uno: Opcionalidad




**Jugador**(<u>#J</u>, Nombre, Nivel, Trofeos, TrofeosMax, #Cl)
FK: #Cl REFERENCES Clan HAS NULL

**Clan**(<u>#Cl</u>, Nombre, Región, Tipo, TrofeosMin)

# Diseño para interrelaciones uno a uno: Obligatoriedad




**Jugador**(<u>#J</u>, Nombre, Nivel, Trofeos, TrofeosMax)

**Clan**(<u>#Cl</u>, Nombre, Región, Tipo, TrofeosMin, #J)

FK: #J REFERENCES Jugador

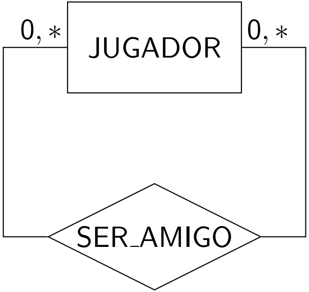
# Diseño para interrelaciones de uno a uno

**Jugador**(#J, Nombre, Nivel, Trofeos, TrofeosMax, #Cl)
FK: #Cl REFERENCES Clan HAS NULL
**Clan**(#Cl, Nombre, Región, Tipo, TrofeosMin)

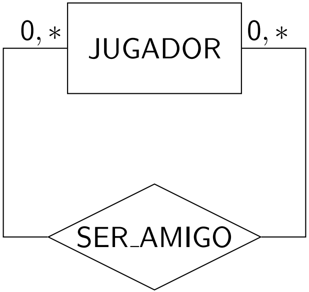
**Jugador**(#J, Nombre, Nivel, Trofeos, TrofeosMax)
**Clan**(#Cl, Nombre, Región, Tipo, TrofeosMin, #J)
FK: #J REFERENCES Jugador

Seleccionar el extremo con mayor cardinalidad mínima es más eficiente

# Diseño para interrelaciones con roles



# Diseño para interrelaciones con roles

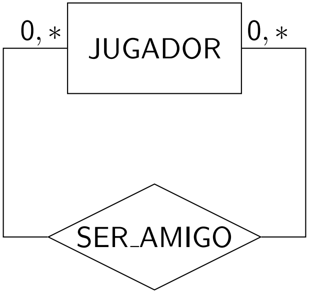



**Ser_Amigo**(<u>#Amigo1</u>, <u>#Amigo2</u>)

FK: #Amigo1 REFERENCES Jugador (#J)

FK: #Amigo2 REFERENCES Jugador (#J)

# Diseño para interrelaciones con roles



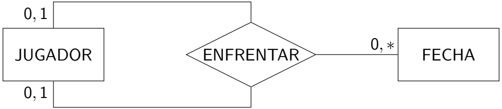

**Ser_Amigo**(#Amigo1, #Amigo2)

FK: #Amigo1 REFERENCES Jugador (#J)

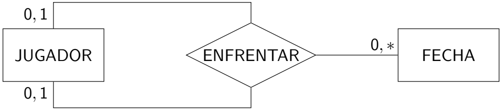
FK: #Amigo2 REFERENCES Jugador (#J)

Se deben renombrar los atributos que tienen el mismo nombre

# Diseño para interrelaciones *n*-arias



# Diseño para interrelaciones *n*-arias




**Fecha**(Fecha)

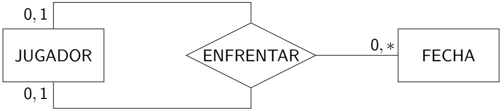
**Enfrentar**(#J1, #J2, Fecha)

FK: #J1 REFERENCES Jugador (#J)

FK: #J2 REFERENCES Jugador (#J)

FK: Fecha REFERENCES Fecha

# Diseño para interrelaciones *n*-arias

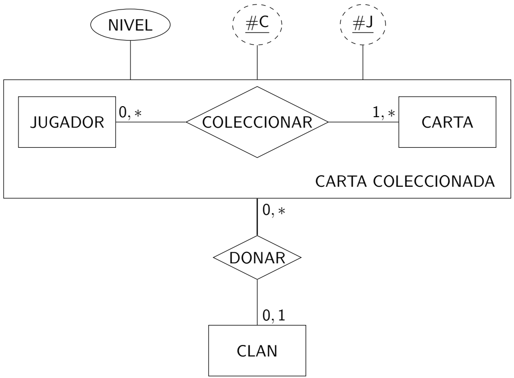



**Fecha**(<u>Fecha</u>)

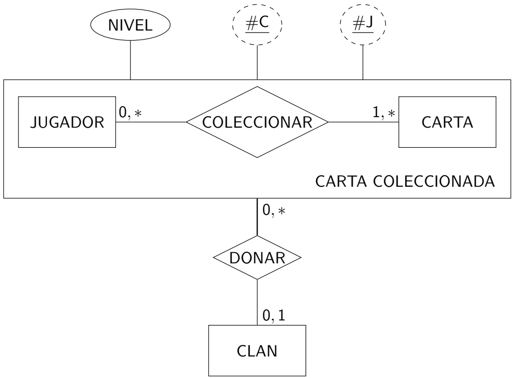
**Enfrentar**(<u>#J1</u>, #J2, <u>Fecha</u>)
FK: #J1 REFERENCES Jugador (#J)
FK: #J2 REFERENCES Jugador (#J)
FK: Fecha REFERENCES Fecha

- Convertir la interrelación en una relación cuyos atributos son las llaves primarias de las relaciones que representan los conjuntos de entidades conectados.
- Si existen extremos de la interrelación con cardinalidad máxima uno, se escoge uno de ellos y su llave primaria se retira de la llave primaria de la relación resultante.

# Diseño de agregaciones



# Diseño de agregaciones



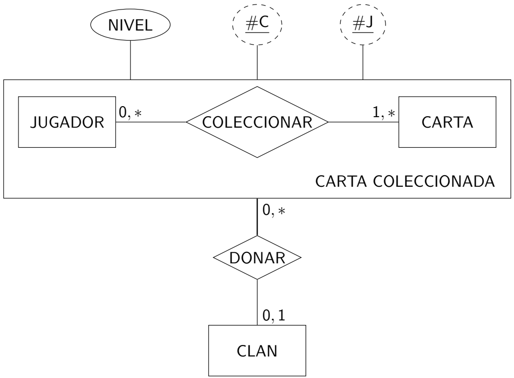

**Coleccionar**(<u>#J</u>, <u>#C</u>, Nivel)

**Donar**(<u>#J</u>, <u>#C</u>, <u>#Cl</u>)

FK: (#J, #C) REFERENCES Coleccionar (#J, #C)

FK: #Cl REFERENCES Clan

# Diseño de agregaciones




**Coleccionar**(<u>#J</u>, <u>#C</u>, Nivel)

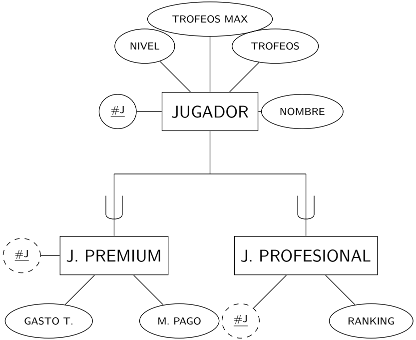
**Donar**(<u>#J</u>, <u>#C</u>, <u>#Cl</u>)

FK: (#J, #C) REFERENCES Coleccionar (#J, #C)

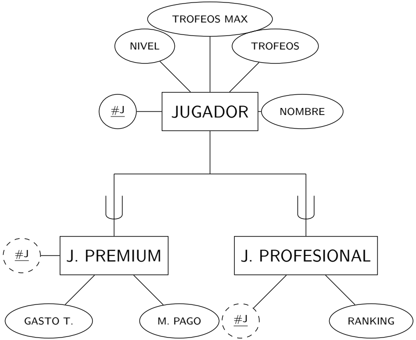
FK: #Cl REFERENCES Clan

Agregar los atributos a la relación resultante de la interrelación

# Diseños para la especialización

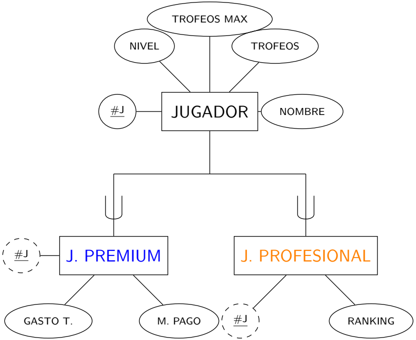


# Diseños para la especialización



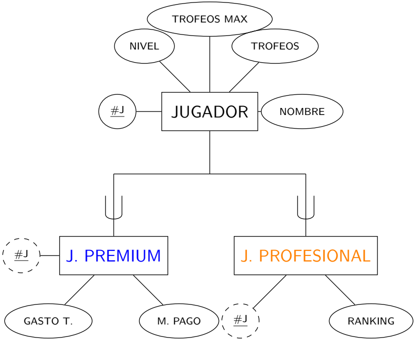

**Jugador**(#J, Nombre, Nivel, Trofeos, TrofeosMax, Gasto T., M. Pago, Ranking)

# Diseños para la especialización




**Jugador**(#J, Nombre, Nivel, Trofeos, TrofeosMax, Gasto T., M. Pago, Ranking)

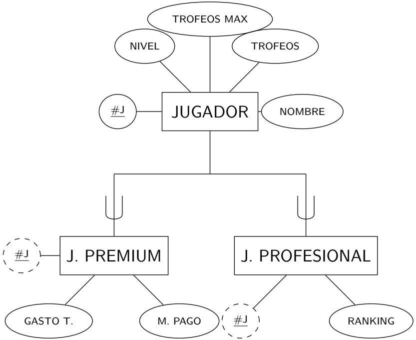
# Diseños para la especialización




**Jugador**(#J, Nombre, Nivel, Trofeos, TrofeosMax, Gasto T., M. Pago, Ranking)

¿Qué ocurre si un jugador es profesional pero no premium?

# Diseños para la especialización




**Jugador**(<u>#J</u>, Nombre, Nivel, Trofeos, TrofeosMax)

**J. Premium**(<u>#J</u>, Gasto T., M. Pago)

FK: #J REFERENCES Jugador

**J. Profesional**(<u>#J</u>, Ranking)

FK: #J REFERENCES Jugador

# Diseños para la especialización

**Jugador**(#J, Nombre, Nivel, Trofeos, TrofeosMax, Gasto T., M. Pago, Ranking)

- Es eficiente espacialmente si la intersección entre los conjuntos especializados es grande.
- Es eficiente para recuperar los datos.

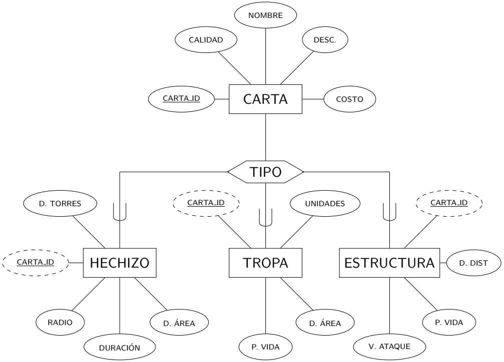
**Jugador**(#J, Nombre, Nivel, Trofeos, TrofeosMax)

**J. Premium**(#J, Gasto T., M. Pago)
FK: #J REFERENCES Jugador

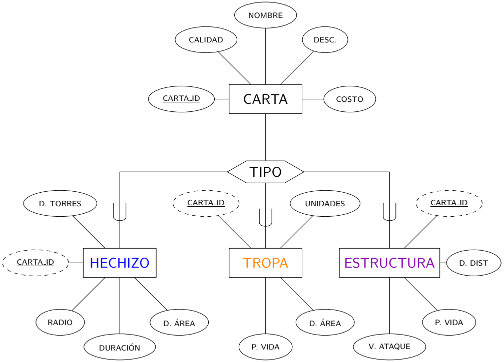
**J. Profesional**(#J, Ranking)
FK: #J REFERENCES Jugador

- Siempre es eficiente espacialmente (Nunca produce el estado NULL de un atributo)
- Para obtener todos los datos de una entidad a profundidad *h* en la jerarquía se deben realizar *h* joins.

# Diseños para la especialización (partición)

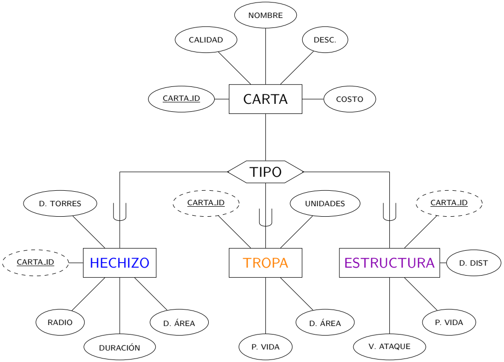


# Diseños para la especialización (partición)




**Carta**(#C, Nombre, Calidad, Desc., Costo, D.Torres, Radio, Duración, H. D. Área, Unidades, T. P. Vida, T. D. Área, D. Dist, E. P. Vida, V. Ataque)

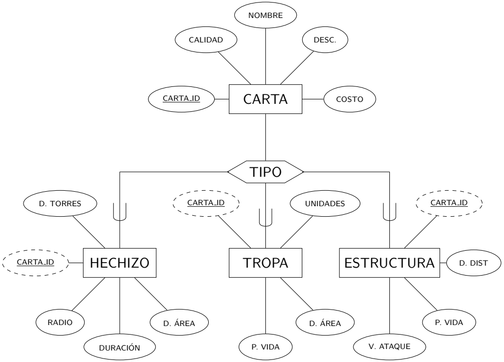
# Diseños para la especialización (partición)




**Carta**(#C, Nombre, Calidad, Desc., Costo, D.Torres, Radio, Duración, H. D. Área, Unidades, T. P. Vida, T. D. Área, D. Dist, E. P. Vida, V. Ataque)

Muy ineficiente espacialmente porque la intersección es vacía.

# Diseños para la especialización (partición)



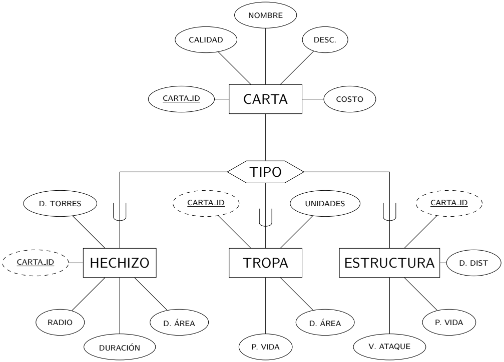

**Carta**(<u>#C</u>, Nombre, Calidad, Desc., Costo)

**Hechizo**(<u>#C</u>, D.Torres, Radio, Duración, D. Área)
FK: #C REFERENCES Carta

**Tropa**(<u>#C</u>, Unidades, P. Vida, D. Área)
FK: #C REFERENCES Carta

**Estructura**(<u>#C</u>, D. Dist, P. Vida, V. Ataque)
FK: #C REFERENCES Carta

# Diseños para la especialización (partición)




**Hechizo**($\underline{\#C}$, Nombre, Calidad, Desc., Costo, D.Torres, Radio, Duración, D. Área)

**Tropa**($\underline{\#C}$, Nombre, Calidad, Desc., Costo, Unidades, P. Vida, D. Área)

**Estructura**($\underline{\#C}$, Nombre, Calidad, Desc., Costo, D. Dist, P. Vida, V. Ataque)

# Diseños para la especialización (partición)

**Carta**(#C, Nombre, Calidad, Desc., Costo)

**Hechizo**(#C, D.Torres, Radio, Duración, D. Área)
FK: #C REFERENCES Carta

**Tropa**(#C, Unidades, P. Vida, D. Área)
FK: #C REFERENCES Carta

**Estructura**(#C, D. Dist, P. Vida, V. Ataque)
FK: #C REFERENCES Carta

- Siempre es eficiente espacialmente (Nunca produce el estado NULL de un atributo)
- Para obtener todos los datos de una entidad a profundidad *h* en la jerarquía se deben realizar *h* joins.

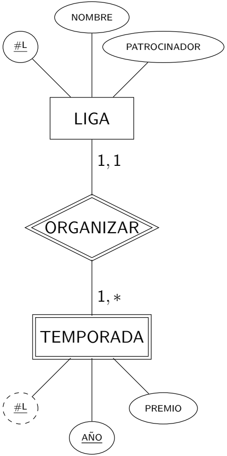
**Hechizo**(#C, Nombre, Calidad, Desc., Costo, D.Torres, Radio, Duración, D. Área)

**Tropa**(#C, Nombre, Calidad, Desc., Costo, Unidades, P. Vida, D. Área)

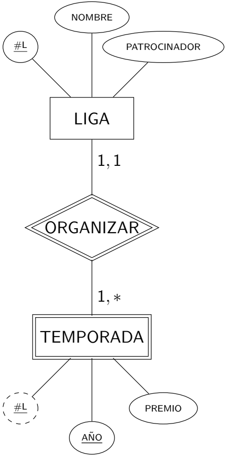
**Estructura**(#C, Nombre, Calidad, Desc., Costo, D. Dist, P. Vida, V. Ataque)

- Siempre es eficiente espacialmente (Nunca produce el estado NULL de un atributo)
- Para obtener la generalización se debe calcular la unión de los conjuntos especializados.

# Diseño para las entidades débiles



# Diseño para las entidades débiles

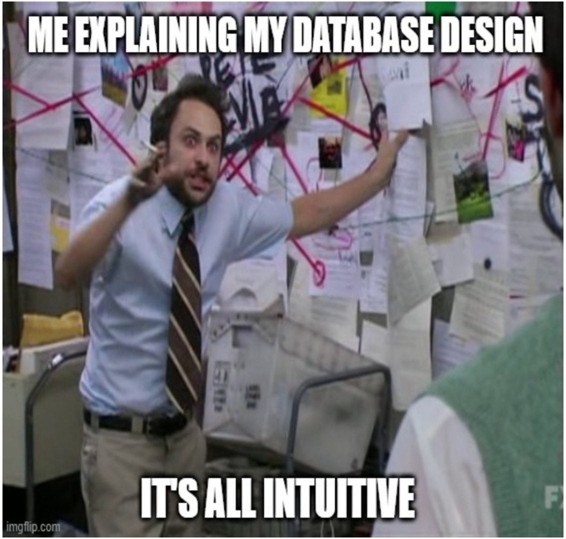



**Liga**(<u>#L</u>, Nombre, Patrocinador)

**Temporada**(<u>#L</u>, <u>Año</u>, Premio)

FK: #L REFERENCES Liga

El diseño lógico no es subjetivo



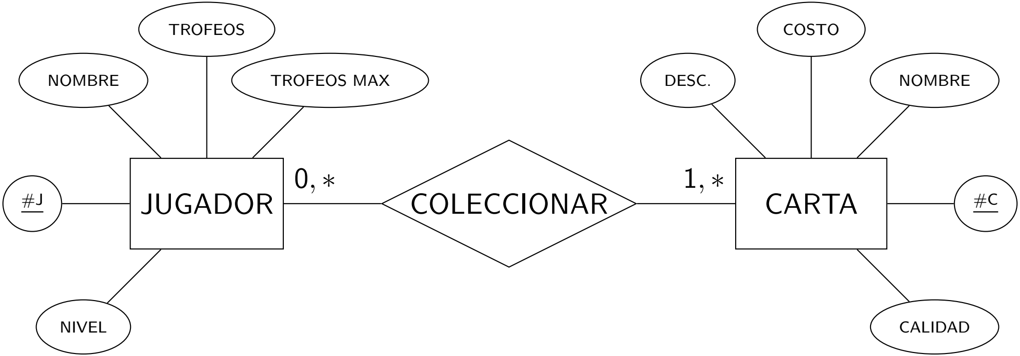
# Lenguajes de consulta

# Lenguajes de consulta

### Definición
Son aquellos lenguajes utilizados para definir solicitudes de recuperación sobre los datos almacenados en una base de datos

### Consulta
Una solicitud de recuperación, es decir, una expresión relacional o una declaración que solicita la evaluación de tal expresión

# Caso de estudio



# Álgebra relacional

## Consultas

# Álgebra relacional

## Consultas

1. Obtener el nombre de todos los jugadores cuyo nivel es al menos 10.

## Consultas

1. Obtener el nombre de todos los jugadores cuyo nivel es al menos 10.

$$r := \pi_{\#\text{J, Nombre}}(\text{Jugador}\,\sigma\,(\text{Nivel} \geq 10))$$

# Álgebra relacional

## Consultas

1. Obtener el nombre de todos los jugadores cuyo nivel es al menos 10.

$$r := \pi_{\#\mathsf{J},\ \mathsf{Nombre}}(\mathsf{Jugador}\,\sigma\,(\mathsf{Nivel} \geq 10))$$

2. Obtener el nombre de todas las cartas cuya calidad sea épica.

# Álgebra relacional

## Consultas

1. Obtener el nombre de todos los jugadores cuyo nivel es al menos 10.

$$r := \pi_{\#\text{J, Nombre}}(\text{Jugador}\,\sigma\,(\text{Nivel} \geq 10))$$

2. Obtener el nombre de todas las cartas cuya calidad sea épica.

$$r := \pi_{\#\text{C, Nombre}}(\text{Carta}\,\sigma\,(\text{Calidad} = \text{"Épica"}))$$

## Consultas

3. Obtener el nombre de todos los jugadores que tienen la carta con identificador 2.

$$r_1 = \pi_{\#\mathrm{J},\mathrm{Nombre}}((\mathrm{Jugador} \bowtie \mathrm{Coleccionar})\,\sigma\,(\#\mathrm{C} = 2))$$

$$r_2 = \pi_{\#\mathrm{J},\mathrm{Nombre}}(\mathrm{Jugador} \bowtie (\mathrm{Coleccionar}\,\sigma\,(\#\mathrm{C} = 2)))$$

¿Cuál es mejor?

# El problema del álgebra relacional

$$\pi_{\#\mathsf{J},\mathsf{Nombre}}((\mathsf{Jugador} \bowtie \mathsf{Coleccionar})\,\sigma\,(\#\mathsf{C} = 2))$$

¿En qué orden se ejecutan las operaciones?

# El problema del álgebra relacional

$$\pi_{\#\mathsf{J},\mathsf{Nombre}}((\mathsf{Jugador} \bowtie \mathsf{Coleccionar})\, \sigma\, (\#\mathsf{C} = 2))$$

¿En qué orden se ejecutan las operaciones?

1. Jugador $\bowtie$ Coleccionar
2. $(\mathsf{Jugador} \bowtie \mathsf{Coleccionar})\, \sigma\, (\#\mathsf{C} = 2)$
3. $\pi_{\#\mathsf{J},\mathsf{Nombre}}((\mathsf{Jugador} \bowtie \mathsf{Coleccionar})\, \sigma\, (\#\mathsf{C} = 2))$

# El problema del álgebra relacional

$$\pi_{\#\mathsf{J},\mathsf{Nombre}}((\mathsf{Jugador} \bowtie \mathsf{Coleccionar})\,\sigma\,(\#\mathsf{C} = 2))$$

¿En qué orden se ejecutan las operaciones?

1. $\mathsf{Jugador} \bowtie \mathsf{Coleccionar} \qquad O(|\mathsf{Jugador}| \times |\mathsf{Coleccionar}|)$
2. $(\mathsf{Jugador} \bowtie \mathsf{Coleccionar})\,\sigma\,(\#\mathsf{C} = 2)$
3. $\pi_{\#\mathsf{J},\mathsf{Nombre}}((\mathsf{Jugador} \bowtie \mathsf{Coleccionar})\,\sigma\,(\#\mathsf{C} = 2))$

# El problema del álgebra relacional

$$\pi_{\#\mathsf{J},\mathsf{Nombre}}((\mathsf{Jugador} \bowtie \mathsf{Coleccionar})\,\sigma\,(\#\mathsf{C} = 2))$$

¿En qué orden se ejecutan las operaciones?

1. Jugador $\bowtie$ Coleccionar $\quad O(|\mathsf{Jugador}| \times |\mathsf{Coleccionar}|)$
2. (Jugador $\bowtie$ Coleccionar) $\sigma\,(\#\mathsf{C} = 2)$ $\quad O(|\mathsf{Coleccionar}|)$
3. $\pi_{\#\mathsf{J},\mathsf{Nombre}}((\mathsf{Jugador} \bowtie \mathsf{Coleccionar})\,\sigma\,(\#\mathsf{C} = 2))$

# El problema del álgebra relacional

$$\pi_{\#\mathsf{J},\mathsf{Nombre}}((\mathsf{Jugador} \bowtie \mathsf{Coleccionar})\,\sigma\,(\#\mathsf{C} = 2))$$

¿En qué orden se ejecutan las operaciones?

1. Jugador $\bowtie$ Coleccionar $\quad O(|\mathsf{Jugador}| \times |\mathsf{Coleccionar}|)$
2. $(\mathsf{Jugador} \bowtie \mathsf{Coleccionar})\,\sigma\,(\#\mathsf{C} = 2) \quad O(|\mathsf{Coleccionar}|)$
3. $\pi_{\#\mathsf{J},\mathsf{Nombre}}((\mathsf{Jugador} \bowtie \mathsf{Coleccionar})\,\sigma\,(\#\mathsf{C} = 2)) \quad O(|\mathsf{Jugador}|)$

# El problema del álgebra relacional

$$\pi_{\#\text{J},\text{Nombre}}((\text{Jugador} \bowtie \text{Coleccionar})\,\sigma\,(\#\text{C} = 2))$$

¿En qué orden se ejecutan las operaciones?

1. $\text{Jugador} \bowtie \text{Coleccionar} \qquad O(|\text{Jugador}| \times |\text{Coleccionar}|)$
2. $(\text{Jugador} \bowtie \text{Coleccionar})\,\sigma\,(\#\text{C} = 2) \qquad O(|\text{Coleccionar}|)$
3. $\pi_{\#\text{J},\text{Nombre}}((\text{Jugador} \bowtie \text{Coleccionar})\,\sigma\,(\#\text{C} = 2)) \qquad O(|\text{Jugador}|)$

$$O(|\text{Jugador}| \times |\text{Coleccionar}| + |\text{Coleccionar}| + |\text{Jugador}|)$$

# El problema del álgebra relacional

$$\pi_{\#\text{J},\text{Nombre}}((\text{Jugador} \bowtie \text{Coleccionar})\,\sigma\,(\#\text{C} = 2))$$

¿En qué orden se ejecutan las operaciones?

1. $\text{Jugador} \bowtie \text{Coleccionar} \quad O(|\text{Jugador}| \times |\text{Coleccionar}|)$
2. $(\text{Jugador} \bowtie \text{Coleccionar})\,\sigma\,(\#\text{C} = 2) \quad O(|\text{Coleccionar}|)$
3. $\pi_{\#\text{J},\text{Nombre}}((\text{Jugador} \bowtie \text{Coleccionar})\,\sigma\,(\#\text{C} = 2)) \quad O(|\text{Jugador}|)$

$$O(|\text{Jugador}| \times |\text{Coleccionar}| + |\text{Coleccionar}| + |\text{Jugador}|)$$

$$= O(|\text{Jugador}|^2 \times |\text{Carta}| + |\text{Coleccionar}| + |\text{Jugador}|)$$

$$\text{dado que } |\text{Coleccionar}| = O(|\text{Jugador}| \times |\text{Carta}|)$$

# El problema del álgebra relacional

$$\pi_{\#\mathsf{J},\mathsf{Nombre}}(\mathsf{Jugador} \bowtie (\mathsf{Coleccionar}\, \sigma\, (\#\mathsf{C} = 2)))$$

¿En qué orden se ejecutan las operaciones?

# El problema del álgebra relacional

$$\pi_{\#\text{J},\text{Nombre}}(\text{Jugador} \bowtie (\text{Coleccionar}\,\sigma\,(\#\text{C} = 2)))$$

¿En qué orden se ejecutan las operaciones?

1. $\text{Coleccionar}\,\sigma\,(\#\text{C} = 2)$
2. $\text{Jugador} \bowtie (\text{Coleccionar}\,\sigma\,(\#\text{C} = 2))$
3. $\pi_{\#\text{J},\text{Nombre}}(\text{Jugador} \bowtie (\text{Coleccionar}\,\sigma\,(\#\text{C} = 2)))$

# El problema del álgebra relacional

$$\pi_{\#\text{J},\text{Nombre}}(\text{Jugador} \bowtie (\text{Coleccionar}\,\sigma\,(\#\text{C} = 2)))$$

¿En qué orden se ejecutan las operaciones?

1. $\text{Coleccionar}\,\sigma\,(\#\text{C} = 2) \qquad O(|\text{Coleccionar}|)$
2. $\text{Jugador} \bowtie (\text{Coleccionar}\,\sigma\,(\#\text{C} = 2))$
3. $\pi_{\#\text{J},\text{Nombre}}(\text{Jugador} \bowtie (\text{Coleccionar}\,\sigma\,(\#\text{C} = 2)))$

# El problema del álgebra relacional

$$\pi_{\#\text{J},\text{Nombre}}(\text{Jugador} \bowtie (\text{Coleccionar}\,\sigma\,(\#\text{C} = 2)))$$

¿En qué orden se ejecutan las operaciones?

1. $\text{Coleccionar}\,\sigma\,(\#\text{C} = 2)$ $\quad O(|\text{Coleccionar}|)$
2. $\text{Jugador} \bowtie (\text{Coleccionar}\,\sigma\,(\#\text{C} = 2))$ $\quad O(|\text{Jugador}|^2)$
3. $\pi_{\#\text{J},\text{Nombre}}(\text{Jugador} \bowtie (\text{Coleccionar}\,\sigma\,(\#\text{C} = 2)))$

# El problema del álgebra relacional

$$\pi_{\#\mathsf{J},\mathsf{Nombre}}(\mathsf{Jugador} \bowtie (\mathsf{Coleccionar}\,\sigma\,(\#\mathsf{C} = 2)))$$

¿En qué orden se ejecutan las operaciones?

1. $\mathsf{Coleccionar}\,\sigma\,(\#\mathsf{C} = 2) \qquad O(|\mathsf{Coleccionar}|)$
2. $\mathsf{Jugador} \bowtie (\mathsf{Coleccionar}\,\sigma\,(\#\mathsf{C} = 2)) \qquad O(|\mathsf{Jugador}|^2)$
3. $\pi_{\#\mathsf{J},\mathsf{Nombre}}(\mathsf{Jugador} \bowtie (\mathsf{Coleccionar}\,\sigma\,(\#\mathsf{C} = 2))) \qquad O(|\mathsf{Jugador}|)$

# El problema del álgebra relacional

$$\pi_{\#\mathsf{J},\mathsf{Nombre}}(\mathsf{Jugador} \bowtie (\mathsf{Coleccionar}\,\sigma\,(\#\mathsf{C} = 2)))$$

¿En qué orden se ejecutan las operaciones?

1. $\mathsf{Coleccionar}\,\sigma\,(\#\mathsf{C} = 2) \qquad O(|\mathsf{Coleccionar}|)$
2. $\mathsf{Jugador} \bowtie (\mathsf{Coleccionar}\,\sigma\,(\#\mathsf{C} = 2)) \qquad O(|\mathsf{Jugador}|^2)$
3. $\pi_{\#\mathsf{J},\mathsf{Nombre}}(\mathsf{Jugador} \bowtie (\mathsf{Coleccionar}\,\sigma\,(\#\mathsf{C} = 2))) \qquad O(|\mathsf{Jugador}|)$

$$O(|\mathsf{Jugador}|^2 + |\mathsf{Coleccionar}| + |\mathsf{Jugador}|)$$

# El caracter imperativo del álgebra relacional

No permite que el usuario se abstraiga de la optimización

# Lenguajes de consulta declarativo

## Cálculo relacional

- Cálculo relacional de tuplas
- Cálculo relacional de dominios

# Cálculo relacional de tuplas

## Definición

Una consulta en el cálculo relacional de tuplas se expresa de la forma:

$$\{t \mid P(t)\}$$

donde:

- $t$ es una variable que representa una tupla
- $P$ es una fórmula bien formada compuesta de átomos

# Cálculo relacional de tuplas

## Consultas

1. Obtener todos los jugadores cuyo nivel es al menos 10.

# Cálculo relacional de tuplas

## Consultas

1. Obtener todos los jugadores cuyo nivel es al menos 10.

$$\{t \mid t \in \text{Jugador} \wedge t[\text{Nivel}] \geq 10\}$$

# Cálculo relacional de tuplas

## Consultas

1. Obtener todos los jugadores cuyo nivel es al menos 10.

$$\{t \mid t \in \text{Jugador} \wedge t[\text{Nivel}] \geq 10\}$$

2. Obtener el nombre de todos los jugadores cuyo nivel es al menos 10.

# Cálculo relacional de tuplas

## Consultas

1. Obtener todos los jugadores cuyo nivel es al menos 10.

$$\{t \mid t \in \text{Jugador} \wedge t[\text{Nivel}] \geq 10\}$$

2. Obtener el nombre de todos los jugadores cuyo nivel es al menos 10.

$$\{t \mid \exists j \in \text{Jugador}(t[\#\text{J}] = j[\#\text{J}] \wedge t[\text{Nombre}] = j[\text{Nombre}] \wedge j[\text{Nivel}] \geq 10)\}$$

# Cálculo relacional de tuplas

## Consultas

3. Obtener el nombre de todos los jugadores que tienen la carta con identificador 2.

## Consultas

3. Obtener el nombre de todos los jugadores que tienen la carta con identificador 2.

$$\{t | \exists j \in \text{Jugador}(t[\#\text{J}] = j[\#\text{J}] \wedge t[\text{Nombre}] = j[\text{Nombre}] \wedge \exists c \in \text{Coleccionar}(j[\#\text{J}] = c[\#\text{J}] \wedge c[\#\text{C}] = 2))\}$$

# Cálculo relacional de dominios

## Definición

Una consulta en el cálculo relacional de dominios se expresa de la forma

$$\{< x_1, x_2, ..., x_n > | P(x_1, x_2, ..., x_n)\}$$

donde:

- $x_1, x_2, ..., x_n$ son variables de dominio
- $P$ es una fórmula bien formada compuesta de átomos

## Cálculo relacional de dominios

### Consultas

Sean las variables $a, b, c, d, e$ correspondientes a $\#$J, Nombre, Nivel, Trofeos, TrofeosMax en la relación Jugador.

1. Obtener todos los jugadores cuyo nivel es al menos 10.

# Cálculo relacional de dominios

## Consultas

Sean las variables $a, b, c, d, e$ correspondientes a #J, Nombre, Nivel, Trofeos, TrofeosMax en la relación Jugador.

1. Obtener todos los jugadores cuyo nivel es al menos 10.

$$\{< a, b, c, d, e > \mid < a, b, c, d, e > \in \text{Jugador} \land c \geq 10\}$$

# Cálculo relacional de dominios

## Consultas

Sean las variables $a, b, c, d, e$ correspondientes a #J, Nombre, Nivel, Trofeos, TrofeosMax en la relación Jugador.

1. Obtener todos los jugadores cuyo nivel es al menos 10.

$$\{< a, b, c, d, e > \mid < a, b, c, d, e > \in \text{Jugador} \wedge c \geq 10\}$$

2. Obtener el nombre de todos los jugadores cuyo nivel es al menos 10.

## Consultas

Sean las variables $a, b, c, d, e$ correspondientes a #J, Nombre, Nivel, Trofeos, TrofeosMax en la relación Jugador.

1. Obtener todos los jugadores cuyo nivel es al menos 10.

$$\{< a, b, c, d, e > \mid < a, b, c, d, e > \in \text{Jugador} \wedge c \geq 10\}$$

2. Obtener el nombre de todos los jugadores cuyo nivel es al menos 10.

$$\{< a, b > \mid \exists c, d, e(< a, b, c, d, e > \in \text{Jugador} \wedge c \geq 10)\}$$

### Consultas

Sean las variables $a, b, c, d, e$ correspondientes a #J, Nombre, Nivel, Trofeos, TrofeosMax en la relación Jugador. Sea la variable $f$ correspondiente a #C en la relación Coleccionar.

1. Obtener el nombre de todos los jugadores que tienen la carta con identificador 2.

# Cálculo relacional de dominios

## Consultas

Sean las variables $a, b, c, d, e$ correspondientes a #J, Nombre, Nivel, Trofeos, TrofeosMax en la relación Jugador. Sea la variable $f$ correspondiente a #C en la relación Coleccionar.

1. Obtener el nombre de todos los jugadores que tienen la carta con identificador 2.

$$\{< a, b > \mid \exists c, d, e(< a, b, c, d, e > \in \text{Jugador} \wedge \exists f(< a, f > \in Coleccionar \wedge f = 2))\}$$

# Equivalencia entre lenguajes

## El teorema de Codd

- Para toda expresión $E$ del álgebra relacional existe una consulta $Q$ del cálculo relacional tal que $E \equiv Q$
- Para toda consulta $Q$ del cálculo relacional existe una expresión algebraica $E$ tal que $Q \equiv E$

El cálculo relacional es un lenguaje relacional completo

# Equivalencia entre lenguajes

## El teorema de Codd

- Para toda expresión $E$ del álgebra relacional existe una consulta $Q$ del cálculo relacional tal que $E \equiv Q$
- Para toda consulta $Q$ del cálculo relacional existe una expresión algebraica $E$ tal que $Q \equiv E$

El cálculo relacional es un lenguaje relacional completo

## Optimización de consultas

Transformar una consulta $Q$ del cálculo relacional en una expresión $E$ del álgebra relacional.

# Turing completo vs Relacional completo

# Turing completo vs Relacional completo

- Mayor expresividad
- No puede ser optimizado para el caso general de forma automática.

- Menor expresividad
- Puede ser optimizado para el caso general de forma automática.

# Entonces...

... alguna duda?