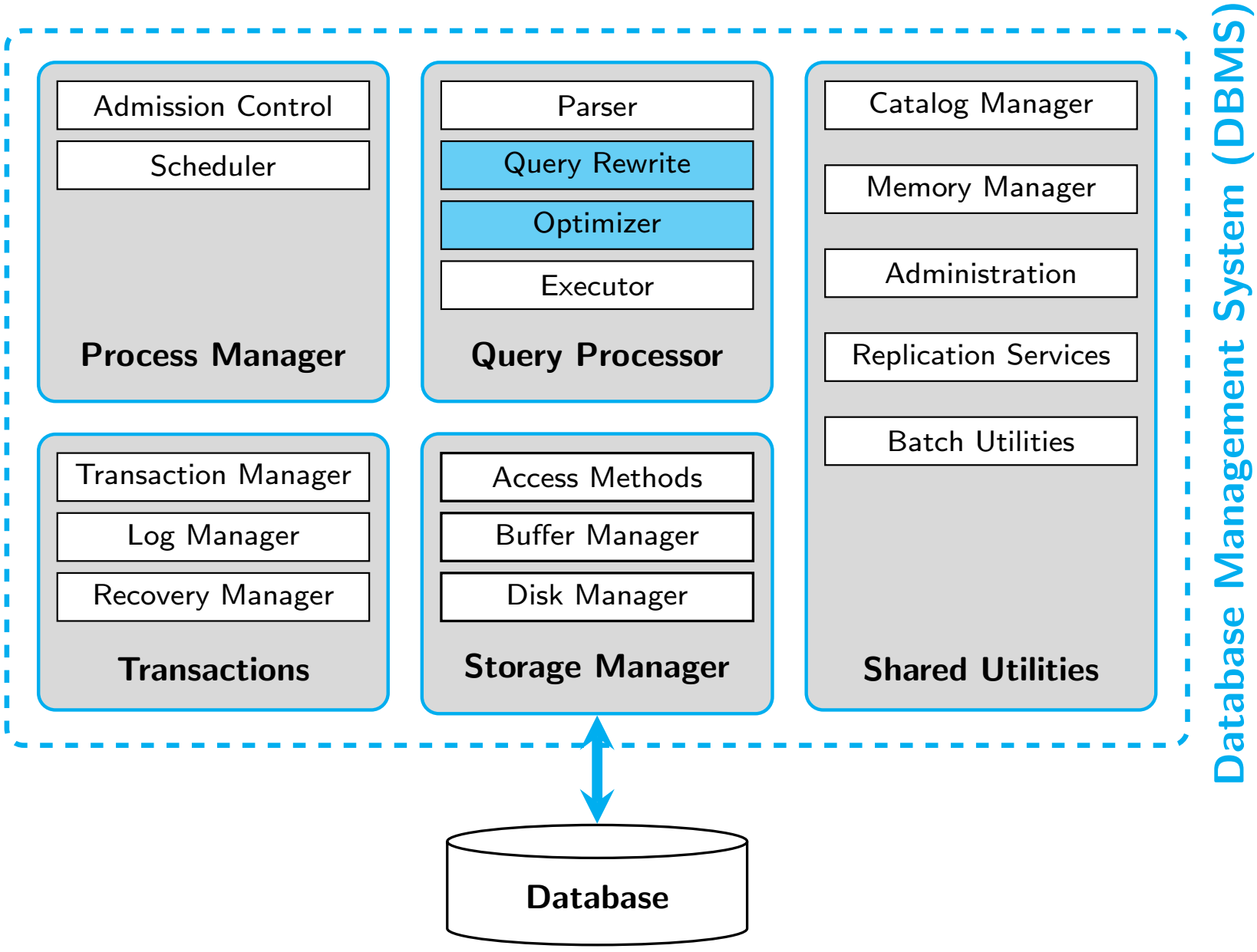
# Optimización de Joins

Clase 16

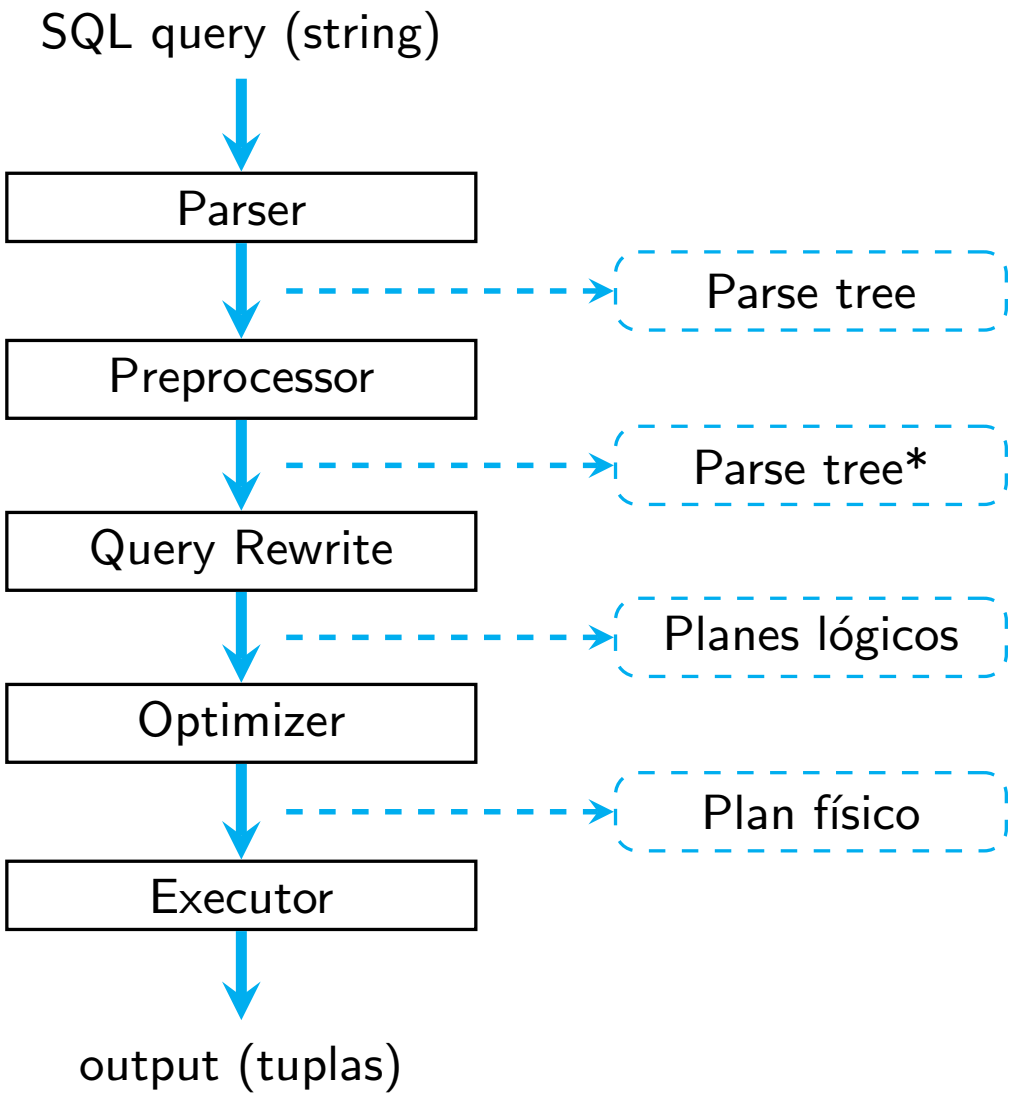
IIC 3413

Prof. Cristian Riveros

# Optimización de consultas

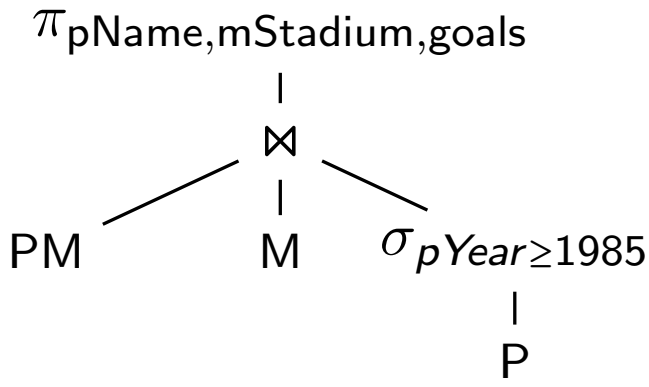


# Zoom al optimizador de consultas



# Desde plan lógico a plan físico

Dado nuestro plan lógico:




¿cómo construir nuestro plan físico?

# Encontrar el "mejor" plan físico

El **santo grial** de los DBMS!!!

Debemos escoger:

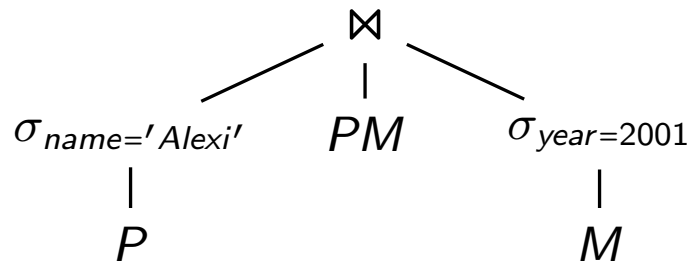
1. los operadores físicos para cada operador lógico.
2. el **orden** de joins/uniones/intersecciones.
3. que resultados **materializar**.

# ¿qué tan importante es un buen plan físico?

Suponga la siguiente consulta:

```
SELECT   *
FROM     Players AS P, Matches AS M, Players_Matches AS PM
WHERE    P.pId = PM.pId AND PM.mId = M.mId AND
         P.name = 'Alexi' AND M.year = 2001
```

con el siguiente plan lógico:




¿cuál es el mejor plan físico?

# Como escoger un plan físico

Primera posibilidad, la estrategia **exhaustiva**:

1. Enumerar todos los posibles planes físicos.
   - Para cada plan lógico, probar **todos** los operadores físicos.
2. Estimar el costo de cada plan físico.
3. Elegir el mejor plan físico y ejecutarlo.

Para enumerar planes, dos posibles estrategias:

- Top-down.
- Bottom-up.

¿es factible esta estrategia?

# Alternativas a la estrategia exhaustiva

- Branch-and-bound.
- Hill climbing.
- Programación dinámica.
- Heurísticas.

# Posibles heurísticas a considerar

- usar **index-scan** cuando hay índices.
- usar **index-join** cuando existe un índice sobre los atributos.
- preferir **sort-join** sobre hash-join cuando las relaciones están ordenadas.
- empezar el join/intersección/unión sobre las relaciones más pequeña.

El secreto mejor guardado de todo motor DBMS!

# Encontrar el "mejor" plan físico

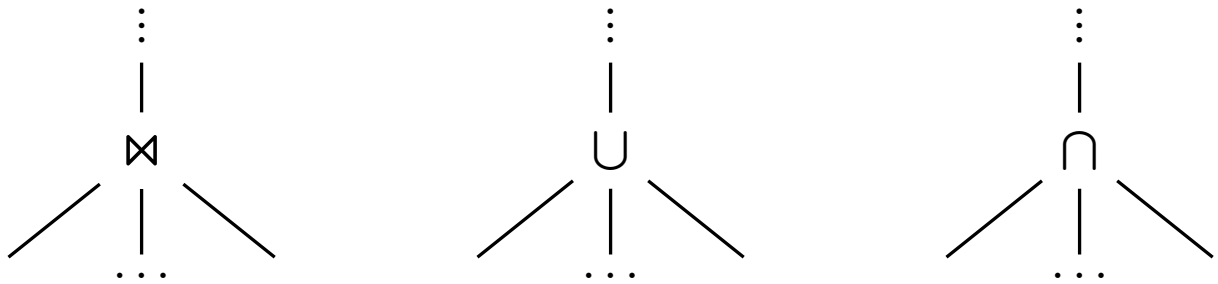
El **santo grial** de los DBMS!!!

Debemos escoger:

1. los operadores físicos para cada operador lógico. ✓
2. el **orden** de joins/uniones/intersecciones.
3. que resultados **materializar**.

# Orden de operadores conmutativos y asociativos

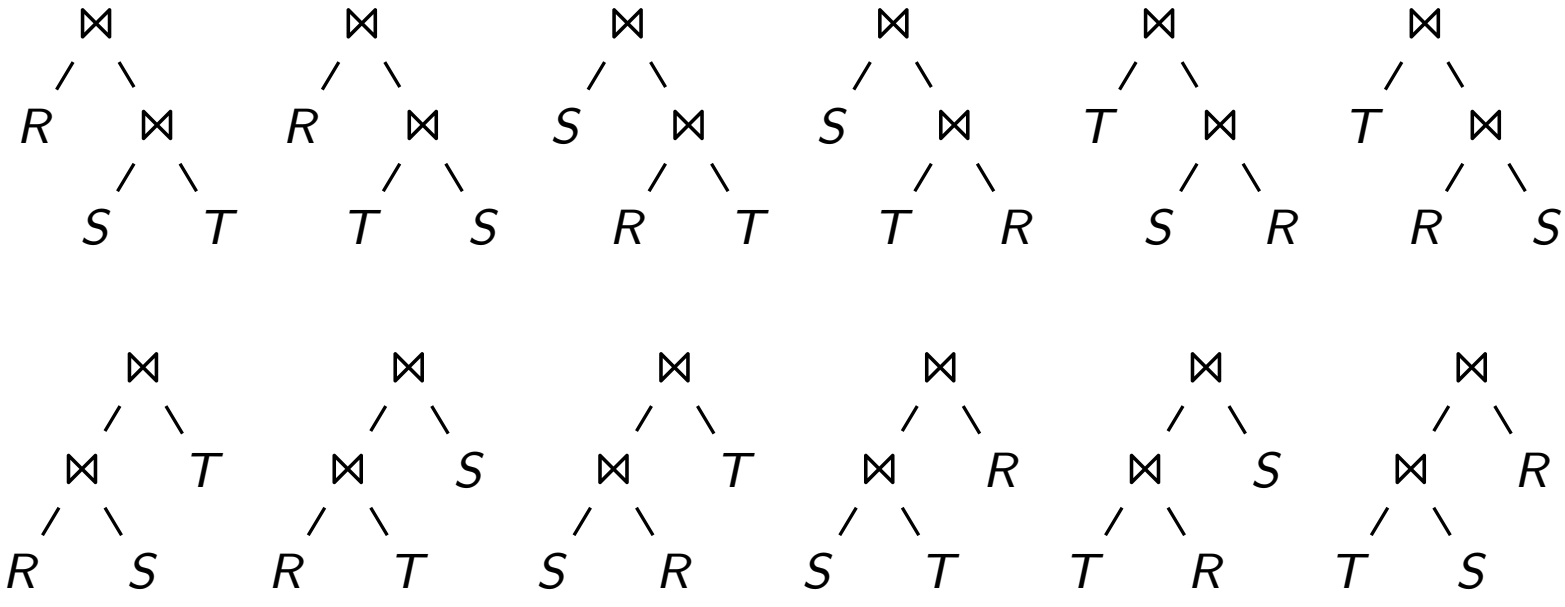
Dado un plan de lógico con nodos de la forma:




¿qué **orden** de los operadores usamos en el plan físico?

# Orden de joins

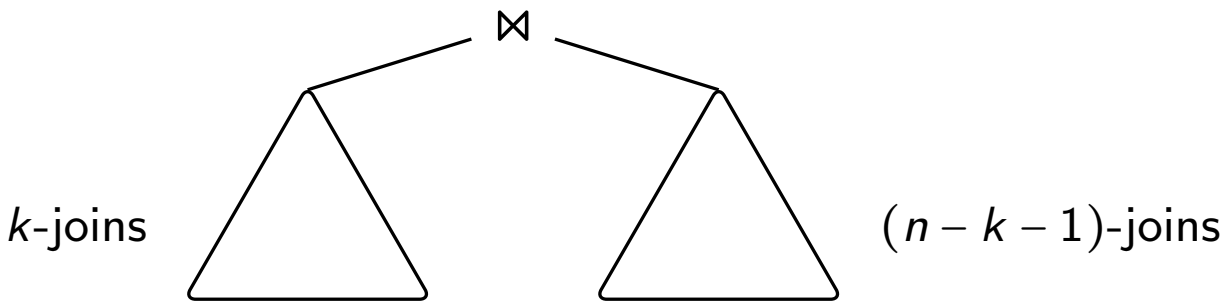
Para $R \bowtie S \bowtie T$, tenemos todos estos **planes** posibles:




¿cuál de todos escogemos?

# ¿cuántos planes de joins podemos escoger?

Para $n$-joins de la forma $R_1 \bowtie R_2 \bowtie \cdots \bowtie R_{n+1}$:




El número de posibles árboles de join para $n$-joins:

$$C_n = \sum_{k=0}^{n-1} C_k \cdot C_{n-k-1}$$

(conocidos como "Catalan numbers")

# ¿cuántos planes de joins podemos escoger?

Resolviendo esta ecuación recursiva:

$$C_n = \frac{(2n)!}{(n+1)!n!}$$

Multiplicando por todas las permutaciones $n!$ de las $n$ relaciones:

$$\#\text{-planes de joins} = \frac{(2n)!}{(n+1)!}$$

Una cantidad exponencial de posibles planes!

# ¿y ahora quién podrá defendernos?

Tres opciones para enumerar planes de joins:

1. Considerarlos todos.
2. Considerar un subconjunto de los planes.
3. Usar una heurística.

# 1. Considerar todos los planes

Clave: usar **programación dinámica**!

Para un multi-join $R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n$:

- **Caso base:** Para cada par $\{R_i, R_j\}$:
  - Estimar el join con menor costo entre $R_i \bowtie R_j$ y $R_j \bowtie R_i$.
  - Almacenar el mejor plan.

- **Inducción (i):** Para cada subconjunto $\mathcal{R} \subseteq \{R_1, \ldots, R_n\}$ con $|\mathcal{R}| = i$.
  - Para cada división de $\mathcal{R} = \mathcal{R}_1 \uplus \mathcal{R}_2$.
  - Estimar el join con menor costo entre:

$$\left(\bowtie_{R \in \mathcal{R}_1} R\right) \bowtie \left(\bowtie_{R \in \mathcal{R}_2} R\right) \quad \text{o} \quad \left(\bowtie_{R \in \mathcal{R}_2} R\right) \bowtie \left(\bowtie_{R \in \mathcal{R}_1} R\right)$$

  - Almacenar el mejor plan de todas las particiones.

# Enumeración basado en programación dinámica

- Encuentra el plan de joins con mejor costo (estimado).
- No es necesario enumerar todos los planes (¿cierto?).
- La cantidad de planes para joins de $N$-relaciones es mucho menor:

$$\sum_{k=1}^{n} \binom{n}{k} \cdot (2^k - 2) \quad \leq \quad 3^n << \frac{(2n)!}{(n+1)!}$$

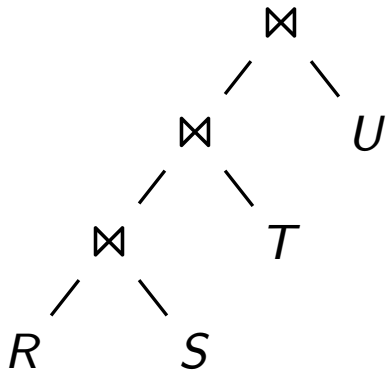
# ¿Y ahora quién podrá defendernos?

Tres opciones para enumerar planes de joins:

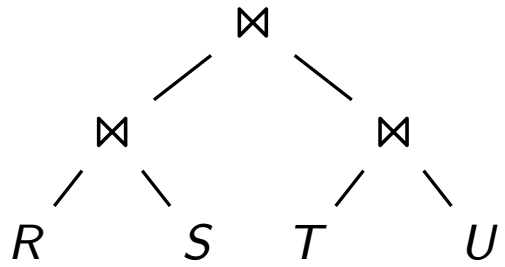
1. Considerarlos todos. ✓
2. Considerar un subconjunto de los planes.
3. Usar una heurística.

# 2. Considerar un subconjunto de los planes

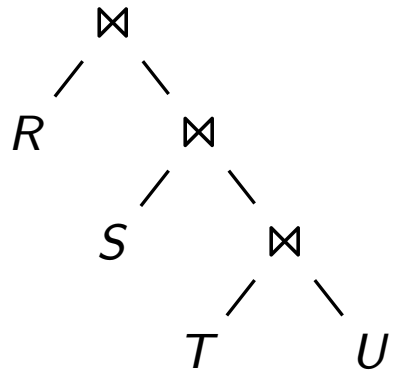
Distintos tipos de planes:




Left-deep plan




Bushy plan




Right-deep plan

¿qué ventaja tiene cada plan?

# Subconjunto de planes prometedores

Considerar solo **left-deep** plans.

Ventajas:

- Un subconjunto menor de planes a considerar (¿cuantos?).
- Buena interacción con los operadores físicos de join.

Podemos considerar programación dinámica sobre este subconjunto!!

Optimizador de **Selinger**

# Optimizador de Selinger

Clave (de nuevo): usar **programación dinámica**!

Para un multi-join $R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n$:

- **Caso base:** Para cada par $\{R_i, R_j\}$:
  - Estimar el join con menor costo entre $R_i \bowtie R_j$ y $R_j \bowtie R_i$.
  - Almacenar el mejor plan.
- **Inducción (i):** Para cada subconjunto $\mathcal{R} \subseteq \{R_1, \ldots, R_n\}$ con $|\mathcal{R}| = i$.
  - Para cada $R \in \mathcal{R}$.
  - Estimar el join:

$$\left(\bowtie_{R' \in \mathcal{R} - \{R\}} R'\right) \bowtie R$$

  - Almacenar el mejor plan de todos los $R$.

# Ejemplo del optimizador de Selinger

## Ejemplo

Calcular el tamaño/costo del natural join:

$$R(a,b) \bowtie S(b,c) \bowtie T(c,d) \bowtie U(d,a)$$

suponiendo que cada relación tiene 1000 tuplas y:

| | $R(a,b)$ | $S(b,c)$ | $T(c,d)$ | $U(d,a)$ |
|---|---|---|---|---|
| $\text{distinct}_a(\cdot)$ | 100 | | | 50 |
| $\text{distinct}_b(\cdot)$ | 200 | 100 | | |
| $\text{distinct}_c(\cdot)$ | | 500 | 20 | |
| $\text{distinct}_d(\cdot)$ | | | 50 | 1000 |

# ¿Y ahora quién podrá defendernos?

Tres opciones para enumerar planes de joins:

1. Considerarlos todos. ✓
2. Considerar un subconjunto de los planes. ✓
3. Usar una heurística.

# 3. Usar una heurística

Heuristica **greedy** para: $R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n$

- Mantener un conjunto de planes $\mathcal{P}$.
- Iniciar el conjunto de planes $\mathcal{P}$ con todas las relaciones:

$$\mathcal{P} = \{R_1, \ldots, R_n\}$$

- En cada iteración, escoger dos planes $P_1, P_2 \in \mathcal{P}$ tal que:

$$\text{cost}(P_1 \bowtie P_2) = \min_{P, P' \in \mathcal{P}} \text{cost}(P \bowtie P')$$

y reemplazar $\mathcal{P} := (\mathcal{P} - \{P_1, P_2\}) \cup \{P_1 \bowtie P_2\}$.

¿cuál es el tiempo de esta heurística?

# Otras heurísticas

- Algoritmos **aleatorios**.
- Algoritmos **genéticos**.

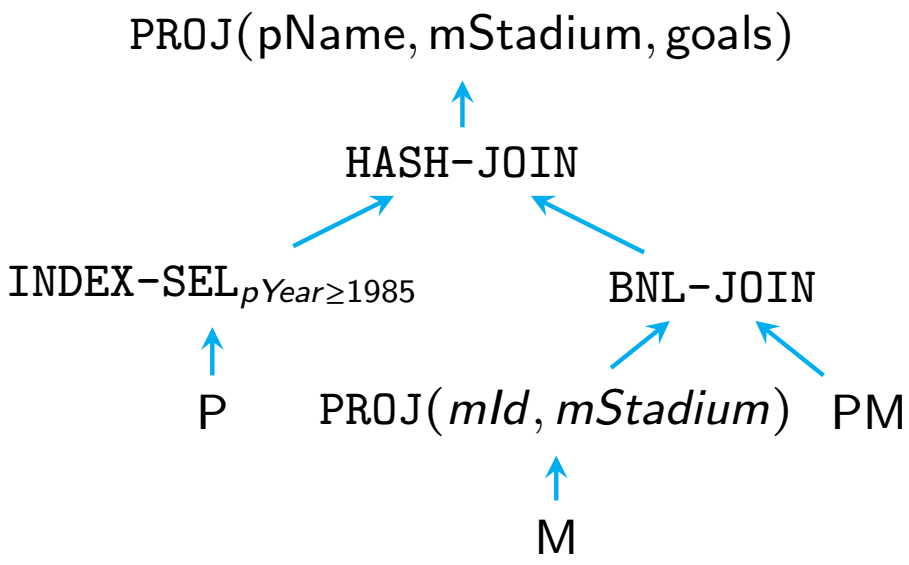
# Encontrar el "mejor" plan físico

El **santo grial** de los DBMS!!!

Debemos escoger:

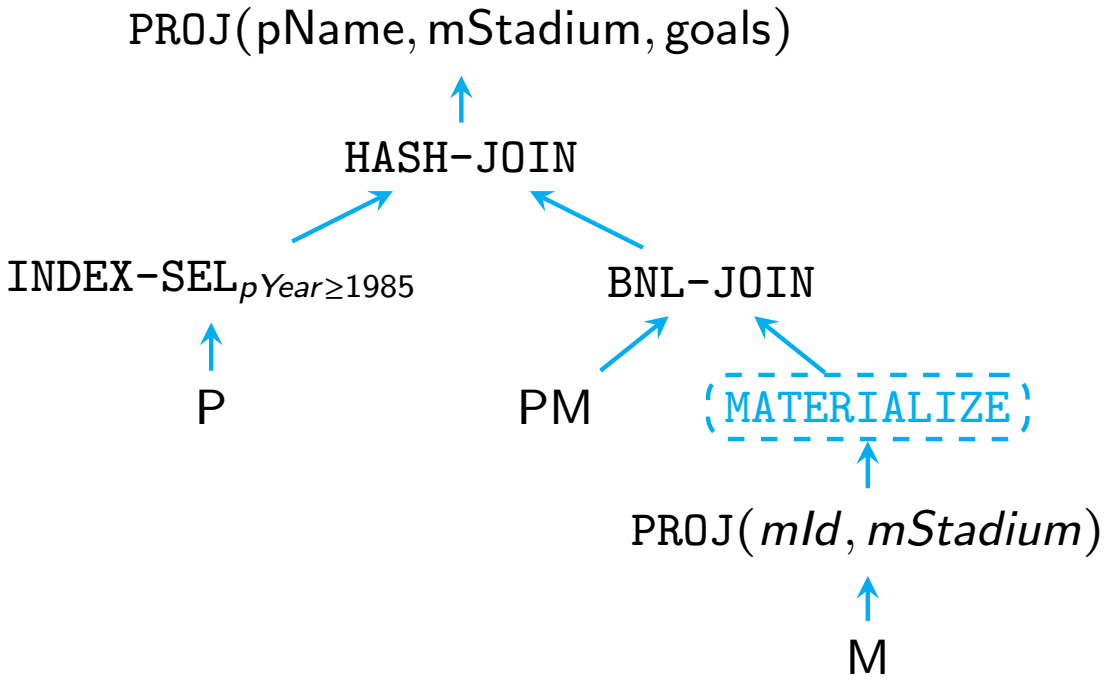
1. los operadores físicos para cada operador lógico. ✓
2. el **orden** de joins/uniones/intersecciones. ✓
3. que resultados **materializar**.

# Pipeline




- Ejecución en serie del plan físico.
- Cada operador genera una tupla a la vez.
- Cada operador no almacena sus resultados intermedios y los retorna a su operador padre.

# Materialización




- Almacena el resultado de la subconsulta en disco.
- Solo necesario si la subconsulta es requerida reiteradas veces.

# Pipeline vs Materialización

**Siempre se prefiere pipeline sobre materialización**

Usar materialización si:

- Subconsulta es llamada reiteradas veces.
- Costo de almacenamiento es menor a recalcular la consulta.

# Finalmente, ejecutamos nuestro plan físico de consultas...