



Diseño de Bases de Datos

Clase 5

Prof. Luciano Marrero

Pablo Thomas

Rodolfo Bertone



Agenda

Optimización de Consultas

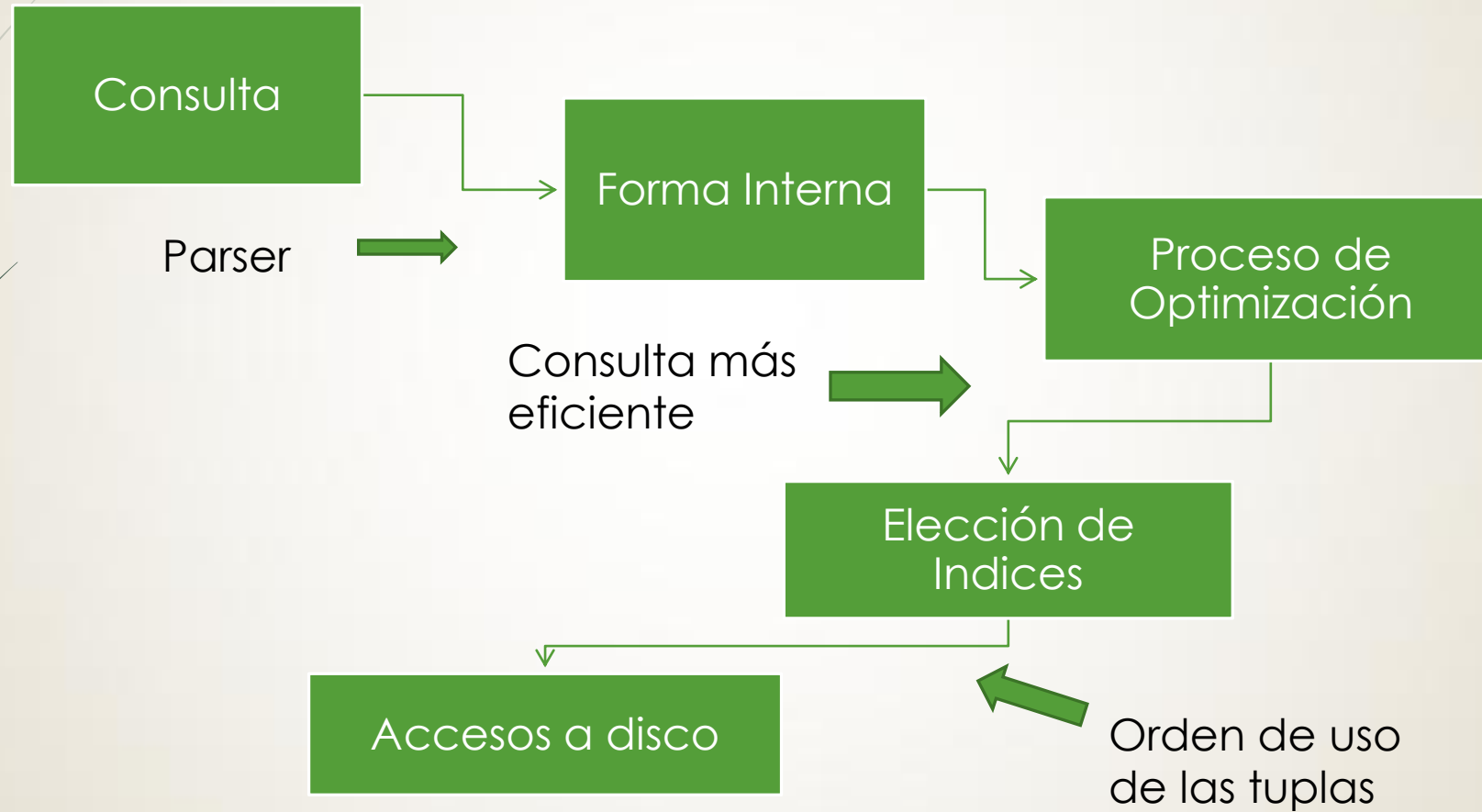
- Costo de Consulta
- Accesos
- Operaciones básicas

Optimización de Consultas

Componentes del “costo” de ejecución de una consulta:

- Costo de acceso a almacenamiento secundario → acceder al bloque de datos que reside en disco.
- Costo de cómputo → Costo de realizar operaciones sobre memoria RAM
- Costo de comunicación → Costo de enviar la consulta y los resultados (si es un Sistema Distribuido)

Optimización de Consultas



Optimización de consultas

Optimización Lógica

- Expresiones equivalentes → Algebra relacional
- existe una secuencia de resolución
- se puede encontrar una expresión más eficiente que otra.

Optimización de consultas

Selección: Personas del género masculino que sean solteros

- $\sigma_{\text{Genero}='M' \wedge \text{ECivil}='Soltero'}(\text{Persona}) \rightarrow$
 - Se aplican 2 condiciones a 7 tuplas
- $\sigma_{\text{Genero}='M'}(\sigma_{\text{ECivil}='Soltero'}(\text{Persona})) \rightarrow$
 - Se aplica 1 condición a 7 tuplas y 1 condición a 1 tupla
- **Conclusión:** el caso 2 es mejor, por lo que conviene realizar la selección lo antes posible

DNI	Nombre	Genero	ECivil
22456980	Josefina	F	Casado
32456789	Juan	M	Casado
24567876	María	F	Casado
21345654	Roberto	M	Soltero
20987654	Alfredo	M	Casado
20897656	Fernanda	F	Casado
21345678	Raul	M	Casado

Optimización de consultas

DNI	Nombre	IdCiudad
22456980	Josefina	1
32456789	Juan	2
24567876	María	3
21345654	Roberto	1
20987654	Alfredo	2
20897656	Fernanda	3
21345678	Raul	1

IdCiudad	Nombre
1	Junín
2	Pergamino
3	La Plata

- **Proyección:** DNI de las personas que vivan en la ciudad de Junín

1. $\pi_{\text{DNI}} (\text{Persona} \mid x \mid \sigma_{\text{Nombre}='Junín'} (\text{Ciudad}))$
2. $\pi_{\text{DNI}} (\pi_{\text{DNI}, \text{IdCiudad}} (\text{Persona}) \mid x \mid \pi_{\text{IdCiudad}} (\sigma_{\text{Nombre}='Junín'} (\text{Ciudad})))$

- **Conclusión:** el caso 2 es mejor, por lo que conviene realizar la proyección para disminuir la cantidad de información que se almacena en buffers de memoria.

Optimización de consultas

La conclusión anterior respecto a la proyección se puede aplicar a otras operaciones binarias:

- Union,
- Intersección,
- Diferencia

Optimización de consultas

Algunos valores:

- $CT\ tabla$ (cantidad de tuplas de la **tabla**)
- $CB\ tabla$ (cantidad de bytes que ocupa cada tupla de la **tabla**)
- $CV(a, tabla)$ (cantidad de ocurrencias de distintas del atributo **a** en la **tabla**)

Costo en bytes selección: $\sigma_{(at = "valor")}(Tabla)$

- $(CT\ tabla / CV(at, tabla)) * CB\ tabla$

Costo en bytes proyección: $\pi_{at1, at2, .. atn}(Tabla)$

- $(CB\ at1 + CB\ at2 + .. + CB\ atn) * CT\ tabla$

Costo en bytes producto cartesiano: $T1 \times T2$

- $(CT\ t1 * CT\ t2) * (CB\ t1 + CB\ t2)$

Optimización de consultas

Costo producto natural: $T1 \bowtie T2$

- Sin atributos en común $\rightarrow T1 \bowtie T2$
- Con atributo “a” en común, donde: a es PK en T1 y FK en T2.
 - $T1 \bowtie T2 \rightarrow$ un fila de T1 con muchas de T2.
 - Clave secundaria.
 - $T2 \bowtie T1 \rightarrow$ un fila de T2 con una de T1.
 - Clave primaria.
- Con atributo “a” en común:
 - $(CT\ t1 * CT\ t2) / MAX(CV(a, t1), CV(a, t2))$

Optimización de Consultas

Dado el siguiente modelo relacional:

- PRODUCTOS (idproducto, código, descripción, precio, idvendedor)
- FK (vendedor, VENDEDORES) la clave foránea no permite nulos
- VENDEDORES (idvendedor, nombre_vendedor, sucursal)

Ejemplo 1: la siguiente **consulta**: “Listar los datos de los productos que vende la sucursal de JUNIN”

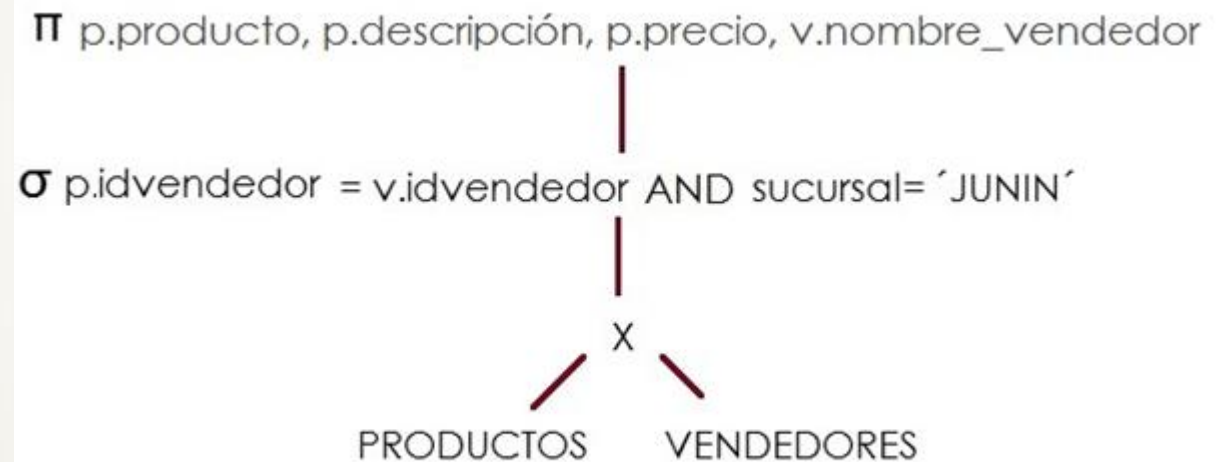
- **SELECT** p.producto, p.descripcion, p.precio, v.nombre_vendedor
- **FROM** PRODUCTOS p, VENDEDORES v
- **WHERE** p.idvendedor = v.idvendedor and v.sucursal = 'JUNIN';

$\Pi_{p.producto, p.descripcion, p.precio, v.nombre_vendedor} (\sigma_{p.idvendedor = v.idvendedor \wedge sucursal = 'JUNIN'} (PRODUCTOS \times VENDEDORES))$

- Sabiendo que:
 - CT(productos) = 7000
 - CT(vendedores) = 300
 - CV (sucursal = 'JUNIN', vendedores) = 10
 - 1000 productos de vendedores de JUNIN

Optmizacion de Consultas

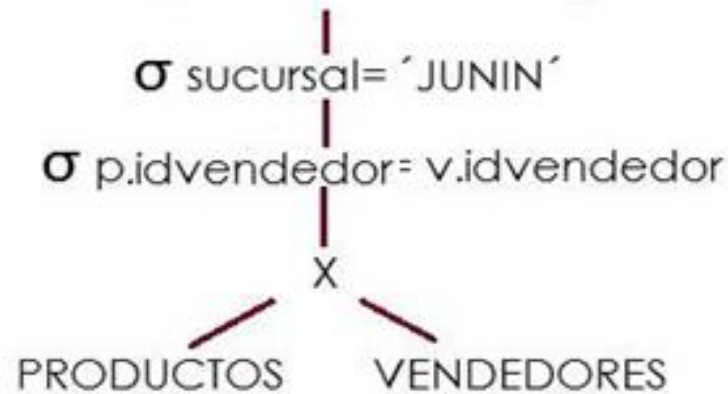
Árbol Inicial



Optimización de Consultas

$\Pi_{p.producto, p.descripcion, p.precio, v.nombre_vendedor}(\sigma_{p.idvendedor = v.idvendedor \wedge sucursal = 'JUNIN'}(PRODUCTOS \times VENEDORES))$

$\Pi_{p.producto, p.descripcion, p.precio, v.nombre_vendedor}$



Plan	Pasos	Operación	Cantidad de lecturas	Costo de acceso	Cantidad de Tuplas	Costo Total
A	1	Producto Cartesiano	7.000 + 300	7.300	2.100.000	7.300
	2	$\sigma_{(A1) \text{ p.idvendedor} = \text{v.idvendedor}}$	2.100.000	2.100.000	7.000	2.107.300
	3	$\sigma_{(A2) \text{ sucursal} = 'JUNIN'}$	7.000	7.000	1.000	2.114.300

Optimización de Consultas

$\Pi_{p.producto, p.descripcion, p.precio, v.nombre_vendedor} (\sigma_{p.id.vendedor = v.id.vendedor} (PRODUCTOS \times \sigma_{sucursal = 'JUNIN'} (VENDEDORES)))$



Plan	Paso	Operación	Cantidad de lecturas	Costo de acceso	Cantidad de Tuplas	Costo Total
B	1	$\sigma_{sucursal = 'JUNIN'}$ (vendedores)	300	300	10	300
	2	B1 x PRODUCTOS	10 + 7.000	7.010	70.000	7.310
	3	$\sigma_{(B2) p.id.vendedor = v.id.vendedor}$	70.000	70.000	1.000	77.310

Optimización de Consultas

$\pi_{p.producto, p.descripcion, p.precio, v.nombre_vendedor} (PRODUCTOS \bowtie (\sigma_{sucursal = 'JUNIN'} VENDEDORES))$



Plan	Nivel	Operación	Cantidad de lecturas	Costo de acceso	Cantidad de Tuplas	Costo Total
C	1	$\sigma_{(vendedores) sucursal = 'JUNIN'}$	300	300	10	300
	2	PRODUCTOS \bowtie C1	10 + 7.000	7.010	1.000	7.310

Optimizacion de Consultas

CONSULTA ORIGINAL:

```
SELECT p.producto, p.descripcion, p.precio, v.nombre_vendedor  
FROM PRODUCTOS p, VENEDORES v  
WHERE p.idvendedor = v.idvendedor and v.sucursal = 'JUNIN';
```

CONSULTA MÁS EFICIENTE:

```
SELECT p.codigo, p.descripcion, p.precio, v.nombre_vendedor  
FROM Productos p NATURAL JOIN (SELECT Idvendedor, nombre_vendedor  
                                FROM Vendedores  
                                WHERE sucursal = 'JUNIN') v
```

EFICIENCIA VS LEGIBILIDAD

CONSULTA MÁS LEGIBLE:

```
SELECT p.codigo, p.descripcion, p.precio, v.nombre_vendedor  
FROM Productos p NATURAL JOIN Vendedores v  
WHERE v.sucursal = 'JUNIN'
```