Índice

Bibliografía

- Fundamentals of Database Systems. 6th Edition
  R. Elmasri y S.B. Navathe. Addison Wesley, 2010
  Capítulo 18
- Database: Principles, Programming, and Performance, 2ª Edición
  P. O'Neil y E. O'Neil. Morgan Kaufmann, 2000
  Capítulo 9
- Fundamentos de bases de datos. 5ª edición
  A. Silberschatz, H.F. Korth y S. Sudarshan. McGraw-Hill, 2006
  Capítulos 15 y 17
- Oracle 10g: Manual del administrador
  K. Loney. McGraw-Hill, 2005
  Capítulo 7

4.3.1 Álgebra relacional

- Los lenguajes relacionales son lenguajes formales (teórico-matemáticos) y se basan en el carácter conjuntista de una relación.
- El álgebra relacional es un lenguaje procedimental para la manipulación de relaciones.
- El Álgebra relacional fue desarrollada en 1970 y el cálculo relacional en 1971 por Codd.
- El Álgebra y el Cálculo Relacional proveen una forma teórica de manipular una base de datos relacional.
- Se basa en el álgebra de la teoría de conjuntos donde los operandos son tablas o relaciones.
- Manipula relaciones produciendo nuevas relaciones.
- Cualquier operación da como resultado una tabla o relación con la que se puede operar de nuevo.
- Consiste en operaciones que algunas de ellas son tomadas de la matemática, otras del lenguaje relacional y otras de lenguajes de programación comunes.
- La cláusula SELECT de SQL es en buena medida una implementación del álgebra relacional.

4.3.1 Álgebra relacional: operaciones

- Básicas
  - Unarias: operan con una sola tabla.
    - Selección
    - Proyección
  - Binarias o de conjunto: Operan con dos tablas.
    - Unión
    - Diferencia
    - Producto cartesiano

- Derivadas o adicionales: realizan en su proceso llamadas a las operaciones básicas.
  - Intersección
  - Cociente o división
  - Join o reunión

4.3.1 Álgebra relacional: operaciones

Tema 4: Optimización de Bases de datos Parte III: Optimización de consultas

- De conjuntos:
  - Unión
  - Intersección
  - Diferencia de conjuntos
  - Producto Cartesiano

- De programación:
  - Asignación

- Relacionales:
  - Proyección
  - Selección
  - División o cociente
  - Join

Grado en Informática Gestión y Administración de Bases de Datos Fernando Martínez Santiago

4.3.1 Álgebra relacional: tablas de ejemplo

R

A	B	C
1	0	1
1	0	0
0	3	2
1	2	3
0	2	0

S

B	D
0	3
3	0
2	1

T

A	B	C
1	1	0
1	0	1
1	2	1
2	0	0
1	2	3

Q

B	D	E
2	1	0
1	1	1
2	1	3
0	3	0
3	0	0

V

E
0
3

4.3.1 Álgebra relacional: selección

Notación: σ_c **(R)**

Ejemplo:

Tresul:= $\sigma_{(c>1 \text{ AND } B>2)}$ **(R)**

R

A	B	C
1	0	1
1	0	0
0	3	2
1	2	3
0	2	0

Tresul

A	B	C
0	3	2

4.3.1 Álgebra relacional: proyección

Notación: $\Pi_{a1,a2,...}(R)$

Ejemplo:

Tresul:= $\Pi_{B,D}(Q)$

Q

B	D	E
2	1	0
1	1	1
2	1	3
0	3	0
3	0	0

Tresul

B	D
2	1
1	1
0	3
3	0

4.3.1 Álgebra relacional: unión

Notación: **R U S**

Ejemplo:

T1:= $\Pi_{B,D}(Q)$

Tresul:=T1 U S

T1

B	D
2	1
1	1
0	3
3	0

S

B	D
0	3
3	0
2	1

Tresul

B	D
2	1
1	1
0	3
3	0
0	3
1	1

4.3.1 Álgebra relacional: diferencia

Notación: **R - S**

Ejemplo:

Tresul:= R - T

R

A	B	C
1	0	1
1	0	0
0	3	2
1	2	3
0	2	0

T

A	B	C
1	1	0
1	0	1
1	2	1
2	0	0
1	2	3

Tresul

A	B	C
1	0	0
0	3	2
0	2	0

4.3.1 Álgebra relacional: producto cartesiano

- Notación: **R x S**
- Ejemplo:
  - Tresul:=R x S
  - Grado(R)=3 Grado(S)=2
    - Grado(R x S)=Grado(R)+Grado(S)=5
  - Cardinalidad(R)=5 Cardinalidad (S)=3
    - Cardinalidad(R x S)=C(R)XC(S)=15

R

A	B	C
1	0	1
1	0	0
0	3	2
1	2	3
0	2	0

S

B	D
0	3
3	0
2	1

Tresul

A	R.B	C	S.B	D
1	0	1	0	3
1	0	1	3	0
1	0	1	2	1
1	0	0	0	3
1	0	0	3	0
1	0	0	2	1
0	3	2	0	3
0	3	2	3	0
0	3	2	2	1
...	...	...	...	...

4.3.1 Álgebra relacional: intersección

Notación: **R ∩ S**

Ejemplo:

Tresul:= R ∩ T

R

A	B	C
1	0	1
1	0	0
0	3	2
1	2	3
0	2	0

T

A	B	C
1	1	0
1	0	1
1	2	1
2	0	0
1	2	3

Tresul

A	B	C
1	0	1
1	2	3

4.3.1 Álgebra relacional: reunión natural (natural *join)*

Notación: **R⊗S**

Ejemplo:

Tresul:= R ⊗ T

R

A	B	C
1	0	1
1	0	0
0	3	2
1	2	3
0	2	0

S

B	D
0	3
3	0
2	1

Tresul

A	B	C	D
1	0	1	3
1	0	0	3
0	3	2	0
1	2	3	1
0	2	0	1

4.3.1 Álgebra relacional: división

Notación: **R ÷ S**

Ejemplo:

Tresul:= Q ÷ V

Q

B	D	E
2	1	0
1	1	1
2	1	3
0	3	0
3	0	0

V

E
0
3

Tresul

B	D
2	1

4.3.1 Álgebra relacional: ejemplos de equivalencia con SQL


```
SELECT AL.nombre, ASIG.nombre,nota
FROM
  AL, ASIG, ALAS
 WHERE
  AL.id = ALAS.id_al AND  ASIG.id = ALAS.id_as
```

$$\Pi_{\texttt{AL.nombre, ASIG.nombre, nota}}(\sigma_{\texttt{AL.id=ALAS.id\_al AND ASIG.id=ALAS.id\_as}}(\texttt{ALxASIGxALAS}))$$

4.3.2 Objetivo

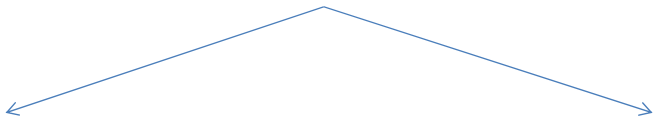
- Disminuir el tiempo de ejecución de las consultas que se realizan frecuentemente:
  - Modificar el diseño físico:
    - Añadir redundancia de datos: atributos calculables, vistas materializadas
    - Añadir/modificar índices, clusters, etc.
  - Analizar la consulta:
    - Escribiéndola de otra manera
    - Aportando más información al DBMS
    - Dando "consejos" (*hints*) al DBMS sobre qué estrategia seguir

→ Es necesario conocer cómo elabora un plan de ejecución el DBMS, analizarlo y diseñar soluciones.

4.3.2 Objetivo: un ejemplo

- Tablas de "proveedores" (`S`) y "pedidos"(`P`) con 100 proveedores y 10.000 pedidos.
- Sólo 50 pedidos con el artículo `P2`.
- Consulta: "Obtener los nombres de los proveedores que nos sirven el artículo `P2`".

```
SELECT DISTINCT S.NOMBRE FROM S, P WHERE S.S#=P.S# AND P.A#="P2";
```



$\pi_{S.nombre}(\sigma_{A\#="P2"}(\sigma_{S.S\#=P.S\#}(SxP)))$

1. SxP
   Card(SXP)=100x10000=**1.000.000**
2. $\sigma_{S.S\#=P.S\#}(SxP)$
   $Card(\sigma_{S.S\#=P.S\#}(SxP))$= 10.000
3. $\sigma_{A\#="P2"}(\sigma_{S.S\#=P.S\#}(SxP))$
   $Card(\sigma_{A\#="P2"}(\sigma_{S.S\#=P.S\#}(SxP)))$=50

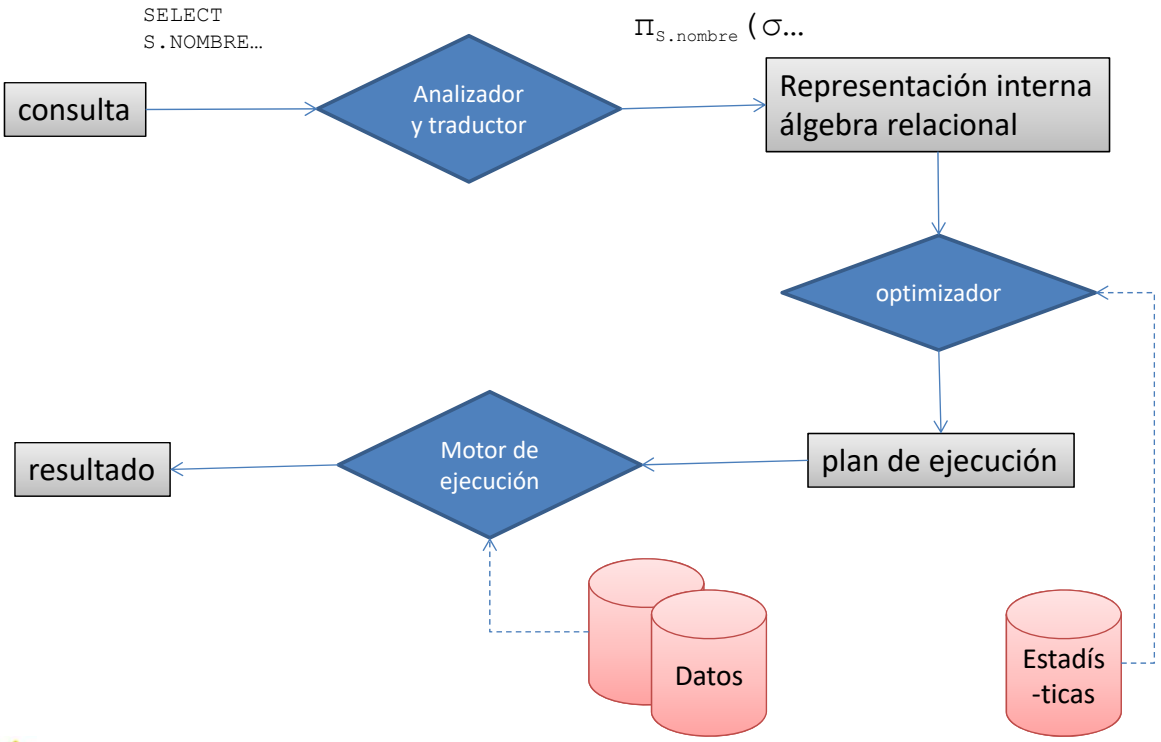
Total operaciones E/S: 3.010.000

$\pi_{S.nombre}(\sigma_{S.S\#=P.S\#}(Sx(\sigma_{A\#="P2"}(P))))$

1. $\sigma_{A\#="P2"}(P)$
   $Card(\sigma_{A\#="P2}(P))$=50 (lectura de 10.000)
2. $Sx(\sigma_{A\#="P2"}(P))$
   $Card(Sx(\sigma_{A\#="P2"}(P))$= 100x50=5000
3. $\sigma_{S.S\#=P.S\#}(Sx(\sigma_{A\#="P2"}(P))$
   $Card(\sigma_{S.S\#=P.S\#}(Sx(\sigma_{A\#="P2"}(P)))$=50

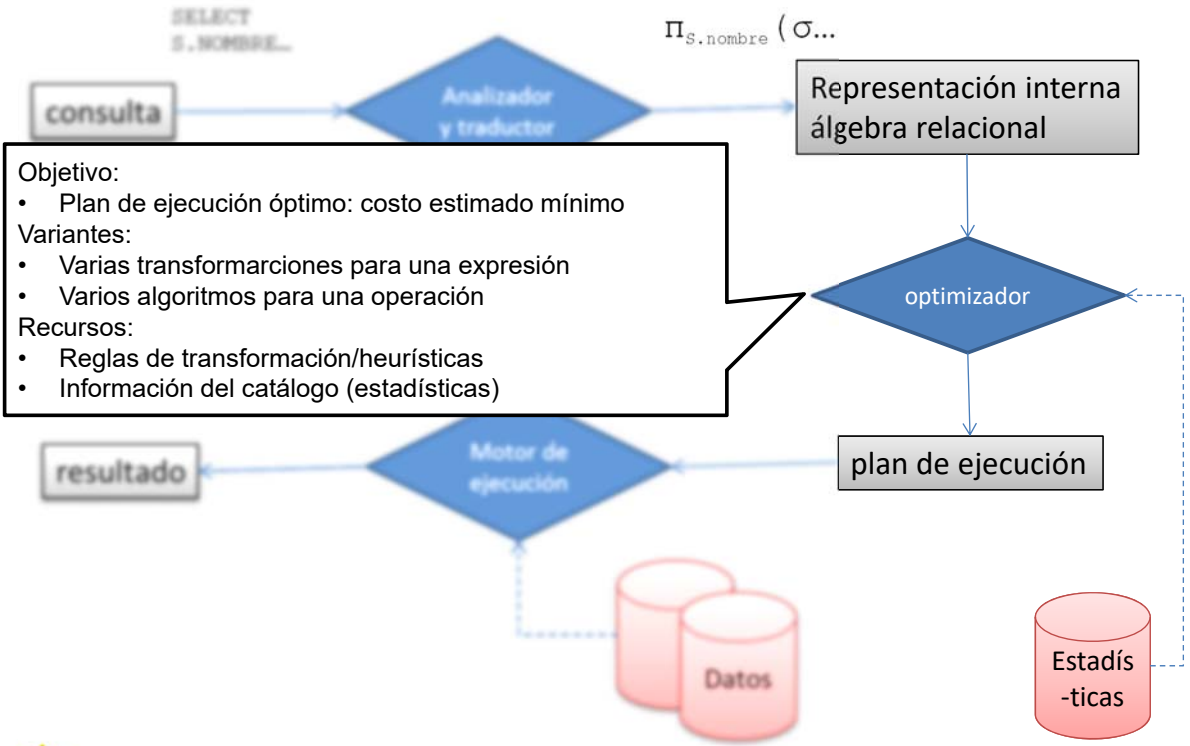
Total operaciones E/S: 10.100

4.3.3 Visión general



4.3.3 Visión general

Tema 4: Optimización de Bases de datos Parte III: Optimización de consultas



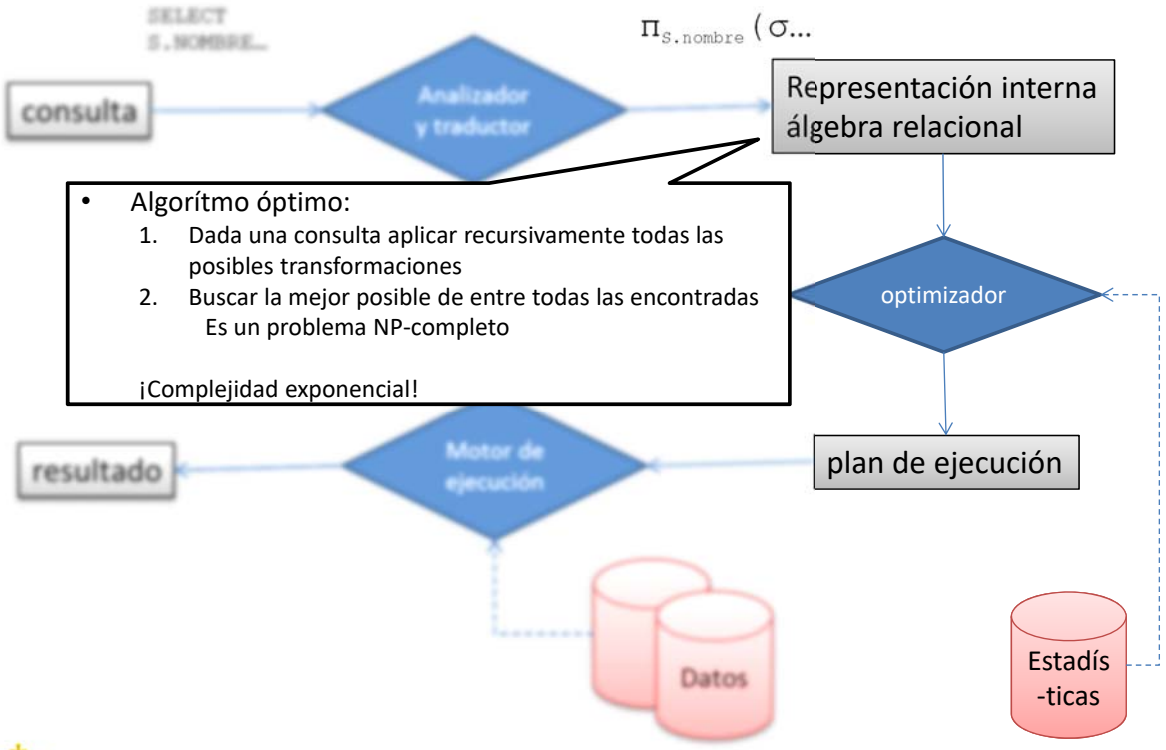
Objetivo:
- Plan de ejecución óptimo: costo estimado mínimo

Variantes:
- Varias transformarciones para una expresión
- Varios algoritmos para una operación

Recursos:
- Reglas de transformación/heurísticas
- Información del catálogo (estadísticas)

Grado en Informática Gestión y Administración de Bases de Datos Fernando Martínez Santiago

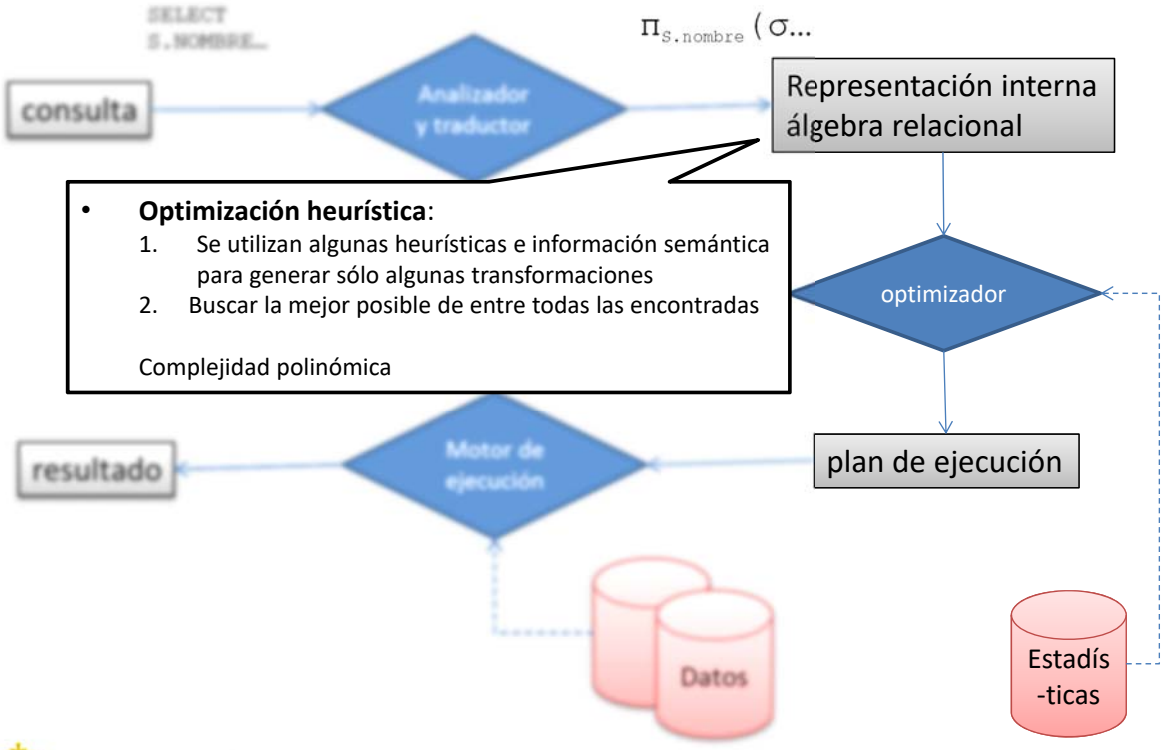
4.3.3 Visión general



- Algorítmo óptimo:
  1. Dada una consulta aplicar recursivamente todas las posibles transformaciones
  2. Buscar la mejor posible de entre todas las encontradas
     Es un problema NP-completo

  ¡Complejidad exponencial!

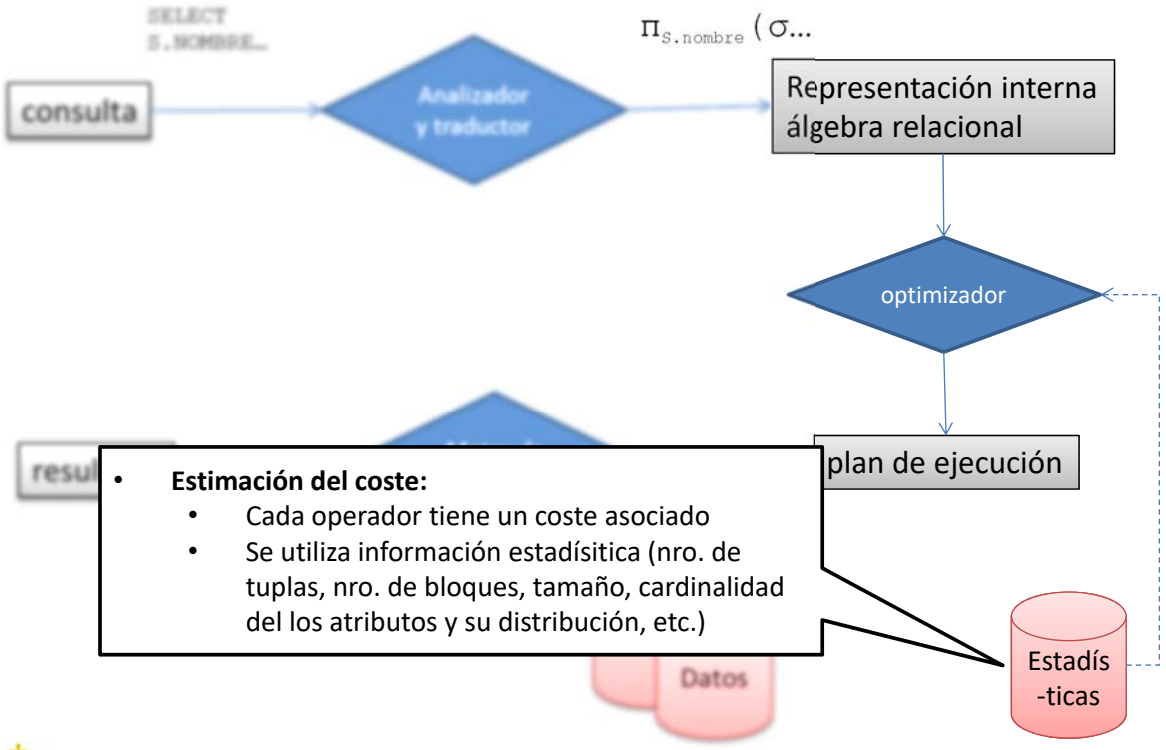
4.3.3 Visión general



- **Optimización heurística**:
  1. Se utilizan algunas heurísticas e información semántica para generar sólo algunas transformaciones
  2. Buscar la mejor posible de entre todas las encontradas

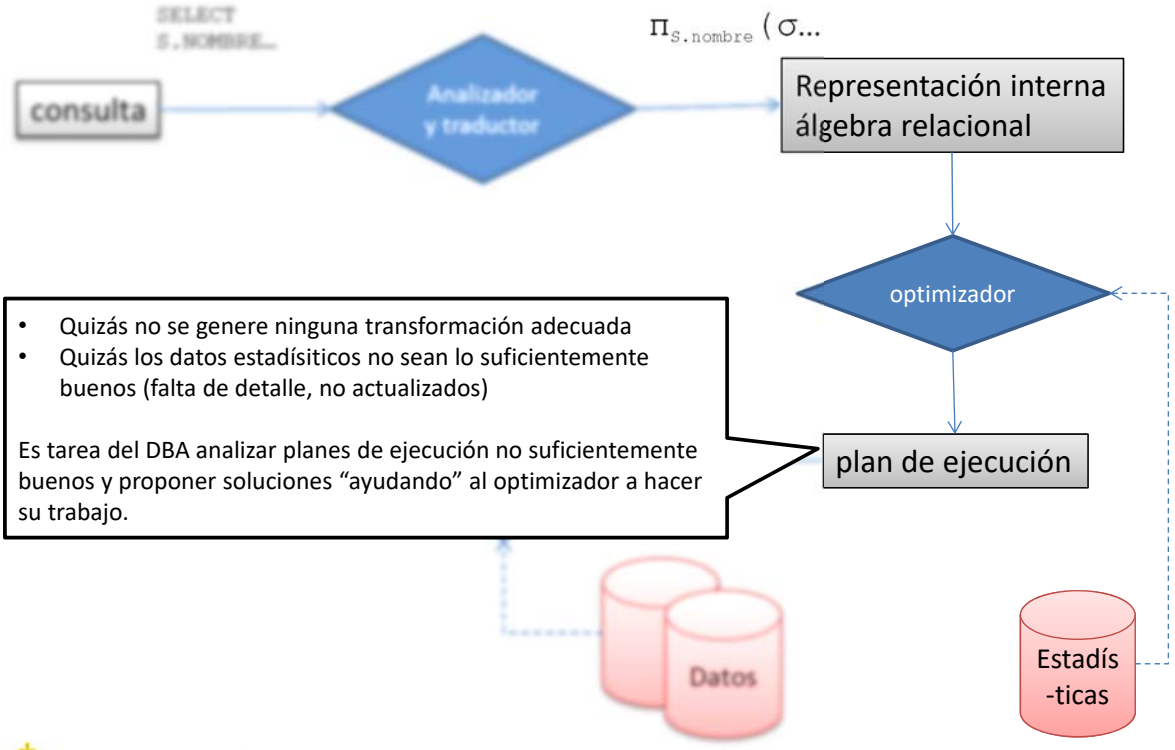
  Complejidad polinómica

4.3.3 Visión general



- **Estimación del coste:**
  - Cada operador tiene un coste asociado
  - Se utiliza información estadísitica (nro. de tuplas, nro. de bloques, tamaño, cardinalidad del los atributos y su distribución, etc.)

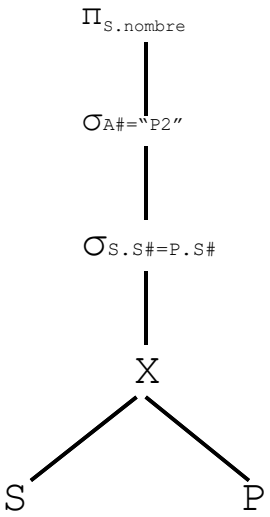
4.3.3 Visión general



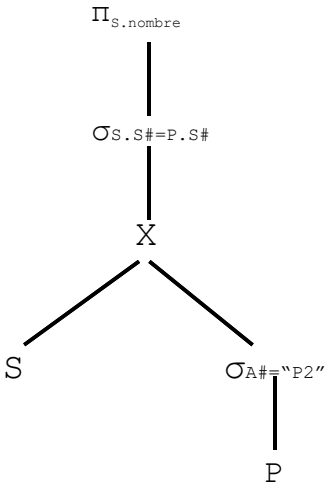
- Quizás no se genere ninguna transformación adecuada
- Quizás los datos estadísiticos no sean lo suficientemente buenos (falta de detalle, no actualizados)

Es tarea del DBA analizar planes de ejecución no suficientemente buenos y proponer soluciones “ayudando” al optimizador a hacer su trabajo.

4.3.4 Optimización heurística: reglas de equivalencia



Árbol inicial



Árbol transfromado

4.3.4 Transformar expresión: reglas de equivalencia

1. Cascada de selecciones
   $\sigma_{C1 \wedge C2}(A) \equiv \sigma_{C1}(\sigma_{C2}(A))$
2. Communtación de selecciones
   $\sigma_{C1}(\sigma_{C2}(A)) \equiv \sigma_{C2}(\sigma_{C1}(A))$
3. Cascada de proyecciones
   $\pi_{P2}(\pi_{P1}(A)) \equiv \pi_{P2}(A)$
4. Equivalencia de Reunión
   $\sigma_{C1}(A \times B) \equiv A \otimes_{C1} B$
5. Commutatividad (reunión)
   $A \otimes B \equiv B \otimes A$
6. Asociatividad (reunión)
   (a) $A \otimes (B \otimes C) \equiv (A \otimes B) \otimes C$
   (b) $A \otimes_{C1 \wedge C3} (B \otimes_{C2} C) \equiv (A \otimes_{C1} B) \otimes_{C2 \wedge C3} C$
   donde C1 sólo implica atributos de A y B y C2 solo implica atributo de B y C
7. Propiedad distributiva (reunión)
   (a) $\sigma_{C1}(A \otimes B) \equiv \sigma_{C1}(A) \otimes B$
   (b) $\sigma_{C1 \wedge C2}(A \otimes B) \equiv \sigma_{C1}(A) \otimes \sigma_{C2}(B)$
   si C1 sólo implica atributos de A y C2 sólo atributos de B
8. Propiedad distributiva (proyección)
   $\pi_P(A \otimes B) \equiv (\pi_{P\_A}(A)) \otimes (\pi_{P\_B}(B))$
   donde P_A, P_B son el conjunto d atributos involucrados en π_P y necesarios para hacer la proyección
9. Commutatividad (unión e intersección)
10. Asociatividad (unión e interesección)
11. Propiedad distributiva (selección, operadores conjuntos)
    $\sigma_{C1}(A-B) \equiv \sigma_{C1}(A) - \sigma_{C1}B(B)$
12. Propiedad distributiva (proyección y unión)
    $\pi_P(A \cup B) \equiv \pi_P(A) \cup \pi_P(B)$

4.3.4 Transformar expresión: ejemplo de heurísticas usuales

1. Ejecutar operaciones de restricción σ tan pronto como sea posible

2. Ejecutar primero las restricciones σ más restrictivas (las que producen menor nº de filas)

3. Combinar un producto cartesiano $\times$ con una restricción σ subsiguiente cuya condición represente una condición de reunión, convirtiéndolas en un join

4. Ejecutar las operaciones de proyección π tan pronto como sea posible

4.3.4 Transformar expresión: ejemplos

Consulta: Obtener los nombres de los suministradores que nos sirven el artículo `P2`

```
-- S: Suministadores, P: Producto
```

$\Pi_{\text{S.nombre}}(\sigma_{\text{A\#="P2"} \wedge \text{S.S\#=P.S\#}}(S \times P))$

Transformación:

1. regla 7a:

$\Pi_{\text{S.nombre}}(\sigma_{\text{S.S\#=P.S\#}}(S \times \sigma_{\text{A\#="P2"}}(P)))$

4.3.4 Transformar expresión: ejemplos

Consulta: Alumnos con más de un 7 en BBDD

```
-- A:alumno, B:alas,C:asig
```

$\Pi_{A.nombre}(\sigma_{nota>7 \wedge C.nombre='BBDD' \wedge A.dni=B.dni \wedge B.as=C.as}(AxBxC))$

Transformación:

1. regla 7b:

$\Pi_{A.nombre}(\sigma_{B.as=C.as \wedge A.dni=B.dni}(Ax\sigma_{nota>7}(B)\ X\sigma_{C.nombre='BBDD'}(C))))$

2. regla 7b:

$\Pi_{A.nombre}(\sigma_{B.as=C.as}(\sigma_{A.dni=B.dni}(Ax\sigma_{nota>7}(B)))\ x\sigma_{C.nombre='BBDD'}(C))))$

3. regla 4 (reescritura):

$\Pi_{A.nombre}((A\otimes\sigma_{nota>7}(B)\otimes\sigma_{C.nombre='BBDD'}(C)))$

4. regla 8

$\Pi_{A.nombre}((\Pi_{nombre,dni}(A)\otimes\sigma_{nota>7}(B)\otimes\sigma_{C.nombre='BBDD'}(\Pi_{nombre,as}(C)))$

4.3.5 Estimación de coste

- El Optimizador tiene un conjunto de técnicas para realizar cada operación
- Ejemplo: técnicas para implementar la operación de selección σ:
  - Búsqueda Lineal
  - Búsqueda Binaria
  - Empleo de Índice Primario
  - Empleo de Índice Secundario

- ¿Cómo elige el Optimizador las técnicas adecuadas en cada caso?

4.3.5 Estimación de coste

- Variable a minimizar: tiempo total de ejecución.
- Factores que intervienen:
  1. Acceso a memoria secundaria
  2. Almacenamiento en disco de estructuras temporales
  3. Uso de CPU
  4. Acceso a los buffers en memoria principal
  5. Coste de comunicación por red
- Estos factores se de estiman atendiendo a diversos estadísticos. Por ejemplo, para una relación dada:
  - Número de tuplas, **r**.
  - Tamaño medio de cada tupla, **R**.
  - Número de bloques necesarios para almacenar la relación, **b.**
  - Factor de bloqueo (cuántas tuplas caben en un bloque), **bfr.**
  - Organización primaria de la tabla: ordenadada/desordenada, indice primario, pertenencia a un cluster, organizada por índice, particionada…
  - Nro. de valores distintos de un atributo, **d**.
  - Distribución media de valores de tal atributo, **sl**
  - Cardinalidad de selección(nro. de tuplas que tienen un valor determinado), **s = sl*r**
  - Altura del índice para cada atributo, **x**
  - ….

4.3.5 Estimación de coste

Tema 4: Optimización de Bases de datos Parte III: Optimización de consultas

- Variable a minimizar: tiempo total de ejecución.
- Factores que intervienen:
  1. Acceso a memoria secundaria
  2. Almacenamiento en disco de estructuras temporales
  3. Uso de CPU
  4. Acceso a los buffers en memoria principal
  5. Coste de comunicicación por red
- Estos factores se de estiman atendiendo a diversos estadísiticos ión dada:
  - Número de tuplas, **r**.
  - Tamaño medio de cada tupla, **R**.
  - Número de bloques necesarios para almacenar la relación, **b.**
  - Factor de bloqueo (cuántas tuplas caben en un bloque), **bfr.**
  - Organización primaria de la tabla: ordenadada/desordenada, indice primario, pertenencia a un cluster, organizada por índice, particionada…
  - Nro. de valores distintos de un atributo, **d**.
  - Distribución media de valores de tal atributo, **sl**
  - Cardinalidad de selección(nro. de tuplas que tienen un valor determinado), **s = sl*r**
  - Altura del índice para cada atributo, **x**
  - ….

¿Se podrá mantener información como esta siempre actualizada?

Grado en Informática Gestión y Administración de Bases de Datos Fernando Martínez Santiago

4.3.5 Estimación de coste

- Variable a minimizar: tiempo total de ejecución.
- Factores que intervienen:
  1. Acceso a memoria secundaria
  2. Almacenamiento en disco de estructuras temporales
  3. Uso de CPU
  4. Acceso a los buffers en memoria principal
  5. Coste de comunicicación por red
- Estos factores se de estiman atendiendo a diversos estadísiticos. Por ejemplo, para una relación dada:
  - Número de tuplas, **r**.
  - Tamaño medio de cada tupla, **R**.
  - Número de bloques necesarios para almacenar la relaci
  - Factor de bloqueo (cuántas tuplas caben en un bloque),
  - Organización primaria de la tabla: ordenadada/desordenada primario, pertenencia a un cluster, organizada por índice, particionada…
  - Nro. de valores distintos de un atributo, **d**.
  - Distribución media de valores de tal atributo, **sl**
  - Cardinalidad de selección(nro. de tuplas que tienen un valor determinado), **s = sl*r**
  - Altura del índice para cada atributo, **x**
  - ….

En un atributo clave ¿qué valores tendra en d,sl y s?

4.3.5 Estimación de coste: selección

Algoritmo	Descripción	Coste (en accesos a bloques)
S1	Se cargan secuencialmente todos los bloques	**b** **b/2**, si se trata de una clave
S2	Búsqueda binaria sobre un campo ordenado	$\log_2$**b**, si se trata de una clave $\log_2$**b** + (**s**/**bfr**)-1, en otro caso
S3a	Usando un índice primario para recuperar una única tupla	**x**+1
S3b	Usando un índice hash para recuperar una única tupla	1 o 2, depende del tipo de hash implementando
S4	Usando un índice ordenado para recuperar varias tuplas	**x**+(**b**/2)
S5	Usando un *cluster* para recuperar varias tuplas	**x**+(**s**/**bfr**)
S6	Usando un índice secundario	(a) **x**+1+**s** (b) **x** + ($\mathbf{b}_{I1}$/2)+(**r**/2)

Selecciones que involucran más de un atributo:

S7: Selección conjuntiva. Se recuperan las filas por uno de los campos y se filtran iterativamente atendiendo al resto de las condiciones. El coste: la del campo por el que se recuperó
S8: Selección conjuntiva usando un índice compuesto: la misma que S3a, S5 o S6

4.3.5 Estimación de coste: Ejemplo

- Tabla `Empleado`
  - r_E=10.000 filas
  - b_E= 2000 bloques
  - bfr_E = 10000 filas/2000 bloques = 5 filas/bloque
- Diseño físico de la tabla:
  1. Un índice cluster en el campo `salario`
     - $X_{salario}$ = 3 accesos
     - $S_{salario}$ = 20 filas ($sl_{salario}$ = 0.002)
  2. Un índice secundario en el atributo clave `dni`
     - X_{dni} = 4 accesos
     - S_{dni} = 1 (sl_{dni},0.0001, es un campo clave)
  3. Un índice secundario en un campo no clave `dpto`
     - X_{dpto} = 2 accesos
     - d_{dpto} = 125 departamentos
     - Sl_{dpto} = 1/125 = 0.008
     - S_{dpto} = r_E*sl_{dpto} = 80
  4. Un índice secundario en el campo `sexo`
     - X_{sexo} = 1
     - d_{sexo} = 2
     - S_{sexo} = r_E/d_{sexo} = 10000/2 = 5000

4.3.5 Estimación de coste: Ejemplo

Tema 4: Optimización de Bases de datos Parte III: Optimización de consultas

- Ej1: $\sigma_{dni='72123456H'}$ **(EMPLEADO)**
  - $C_{S1}(Ej1)=b/2=1000$
  - $C_{S6a}(Ej1)=x_{dni}+s_{dni}=4+1=5$ ✓
- Ej2: $\sigma_{dpto>5}$ **(EMPLEADO)**
  - $C_{S1}(Ej2)=b=2000$ ✓
  - $C_{S6b}(Ej2)=x_{dto}+(b_{I1}/2)+(r/2)=2+4/2+10000/2=5004$
- Ej3: $\sigma_{dpto=5}$ **(EMPLEADO)**
  - $C_{S1}(Ej3)=b=2000$
  - $C_{S6a}(Ej3)=x_{dpto}+s_{dpto}=2+80=82$ ✓
- Ej4: $\sigma_{dpto=5 \wedge salario>30000 \wedge sexo='F'}$ **(EMPLEADO)**

  Selección conjuntiva. Calculamos el coste para cada condición
  - dpto=5 → $C_{S1}(Ej4)=82$ ✓
  - salario>30000 → $C_{S4}(Ej4)=X_{salario}+(b_E/2)=3+(2000/2)=10003$
  - Sexo='F' → $C_{S6a}(Ej3)=x_{sexo}+s_{sexo}=1+5000=5001$

Grado en Informática Gestión y Administración de Bases de Datos Fernando Martínez Santiago

4.3.5 Estimación de coste: reunión

- Es necesario estimar el número de tuplas resultante: especificidad de reunión (*join selectivity*):

  $js = |(R \otimes_C S)| / |RxS| = |(R \otimes_C S)| / (|R|*|S|)$

  donde R y S son dos relaciones y C la condición del *join*

- En el peor caso $js=1$, coincide con un producto cartesiano

- Tamaño del *join*: $js*|R|*|S|$

- Para estimar el coste de la reunión debemos considerar:
  - El coste de lectura, que variará según los accesos físicos disponibles
  +
  - El coste de escribir el resultado en disco:
    - $(js*|R|*|S|)/bfr_{RS}$

    donde bfr_{RS} es el número de filas que caben en cada bloque

4.3.5 Estimación de coste: reunión natural

Algoritmo	**Descripción**	**Coste (en accesos a bloques)**
J1	Fuerza bruta: bucle anidado que recorre ambas tablas, a modo de matriz bidimensional	$C_{J1}=b_R + (b_R*b_S) + ((js* \|R\|* \|S\|)/bfr_{RS})$
J2	Si existe un índice en una de las tablas sobre los campos del *join*, se recorre secuencialmente la tabla no indexada y se usa el índice para recuperar los valores conicidentes en la tabla indexada	Indice secundario: $C_{J2a} = b_R + (\|R\| * (x_B + s_B)) + ((js* \|R\|* \|S\|)/bfr_{RS})$; Índice de tipo cluster: $C_{J2b} = b_R + (\|R\| * (x_B + (s_B/bfr_B))) + ((js* \|R\|* \|S\|)/bfr_{RS})$; Índice primario: $C_{J2c} = b_R + (\|R\| * (x_B + 1)) + ((js* \|R\|* \|S\|)/bfr_{RS})$; Índice hash: $C_{J2d} = b_R + (\|R\| * h) + ((js* \|R\|* \|S\|)/bfr_{RS})$; h=1 o h=2, según tipo de tabla *hash*
J3	Si ambas tablas están físicamente ordenadas por los campos del *join*, se recorren secuencialmente y a la vez ambas tablas recuperando los valores coincidentes	$C_{J3a} = C_S + b_R + b_S + ((js* \|R\|* \|S\|)/bfr_{RS})$ C_S= Coste de ordenar ambas tablas

4.3.5 Estimación de coste: ejemplo

- ¿ EMPLEADO $\otimes_{dno=DNumber}$ DPTO ?
  - Dnumber clave de departamento, $x_{Dnumber}=1$
  - $r_D = 125$ (hay 125 departamentos)
  - $b_D = 13$ bloques
  - $bfr_{ED}=4$
- J1, EMPLEADO $\otimes_{dno=Dnumber}$ DPTO

  $C_{J1} = b_E + (b_E*b_D) + ((js*r_E*r_D)/bfr_{ED})$

  $= 2000+(2000*13) + (((1/125)*10.000*125)/4) = 30.500$
- J1, DPTO $\otimes_{dno=Dnumber}$ EMPLEADO

  $C_{J1} = b_D + (b_D*b_E) + ((js*r_E*r_D)/bfr_{ED})$

  $= 13+(13*2000) + (((1/125)*10.000*125)/4) = 28513$
- J2, EMPLEADO $\otimes_{dno=Dnumber}$ DPTO

  $C_{J2} = b_E + (r_E * (x_{Dnumber}+ 1)) + ((js*r_E*r_D)/bfr_{ED})$

  $= 2000+(10000*2) + (((1/125)*10.000*125)/4) = 24500$
- J2, DPTO $\otimes_{dno=Dnumber}$ EMPLEADO

  $C_{J2} = b_D + (r_D * (x_{Dno}+ S_{Dno})) + ((js*r_E*r_D)/bfr_{ED})$

  $= 13+(125*(2+80)) + (((1/125)*10.000*125)/4) = 12763$

4.3.5 Estimación de coste: algunas anotaciones

- Una consulta con n reuniones tendrá n! posibles ordenes. No siempre es posible evaluarlas todas
- El valor de js en ocasiones está mal estimado
- La **optimización semántica** puede utilizar restricciones del dominio codificadas en la base de datos:
  - Algunas se usan: restricción de identidad, integridad referencial, etc.
  - Otras, son más propias de ontologías y bases de conocimiento:
    - Un alumno no tiene más de siete convocatorias → no buscar más convocatorias una vez encontradas siete
    - Una asignatura pertenece exclusivamente a uno de los siguiente conjuntos: obligatoria, optativa, troncal → dejar de buscar una vez localizada en uno de esos conjuntos.

4.3.6 Optimización de consultas en Oracle:

- Utiliza transformaciones sobre la consulta y estimación del coste
- A cada plan le asigna un coste estimado:
  - E/S, tiempo de CPU, uso de memoria
  - Hipótesis: menor coste → menor tiempo de ejecución
  - Realidad: no siempre es así
- Permite incluír consejos o directrices sobre cómo realizar el plan de ejecución
  - Priorizar la obtención de primeros resultados o la resolución completa de la consulta
  - Qué acceso a dato utilizar (secuencial índice, etc)
  - En qué orden ejecutar los joins
  - Cómo evaluar las condiciones de tales joins
  - …

4.3.6 Optimización de consultas en Oracle: ejemplo (1)

- Consulta:

```
SELECT DISTINCT A.NODO_ID, B.NODO_B_ID NODO_EQ1
FROM A,  B, C
WHERE A.NODO_ID = B.NODO_A_ID
AND B.NODO_B_ID = C.NODO_ID;
```

SQL> @c:\oracle\ora92\rdbms\admin\utlxpls

PLAN_TABLE_OUTPUT

Id	Operation	Name	Rows	Bytes	Cost
0	SELECT STATEMENT		5554	124K	**23**
1	SORT UNIQUE		5554	124K	23
2	NESTED LOOPS		5554	124K	2
3	**MERGE JOIN CARTESIAN**		**5985M**	**61G**	**2**
4	TABLE ACCESS FULL	TMPNOD	1327	6635	2
5	BUFFER SORT		4510K	25M	
6	INDEX FULL SCAN	PK_NOD	4510K	25M	
* 7	INDEX RANGE SCAN	IDX_NOD	1	12	

Predicate Information (identified by operation id):

7 - access("B"."NODO_B_ID"="C"."NODO_ID" AND "A"."NODO_ID"="B"."NODO_A_ID")

Tiempo de ejecución: 11 horas

4.3.6 Optimización de consultas en Oracle: ejemplo (y 2)

- Consulta (reescrita):

```
SELECT DISTINCT A.NODO_ID, B.NODO_B_ID NODO_EQ1
FROM A, (select distinct nodo_a_id, nodo_b_id from B) B, C
WHERE A.NODO_ID = B.NODO_A_ID
AND B.NODO_B_ID = C.NODO_ID;
```

PLAN_TABLE_OUTPUT

Id	Operation	Name	Rows	Bytes	Cost
0	SELECT STATEMENT		5554	200K	**13626**
1	NESTED LOOPS		5554	200K	13626
2	NESTED LOOPS		5554	168K	13626
3	VIEW		7159K	177M	13626
4	SORT UNIQUE		7159K	81M	13626
5	TABLE ACCESS FULL	RE_03	7159K	81M	2589
* 6	INDEX UNIQUE SCAN	PK_EL_3	1	5	
* 7	INDEX UNIQUE SCAN	PK_R_3	1	6	

Predicate Information (identified by operation id):

```
6 - access("B"."NODO_B_ID"="C"."NODO_ID")
7 - access("A"."NODO_ID"="B"."NODO_A_ID")
```

Tiempo de ejecución: 1 hora 20 segundos

4.3.7 Ejercicios

1. ¿Por qué se debe expresar la consulta SQL en términos de álgebra relacional?
2. Si tuvieras que implementar un join, ¿cuándo usarias `J1` y cuándo `J2`?
3. ¿Qué es la optimización heurística? ¿es óptima?
4. ¿Por qué no es posible implementar la optimización semántica completa en las DBMS actuales? ¿qué haría falta?
5. Dada las siguiente consulta

```
SELECT E.Fname, E.Lname, E.Address
FROM EMPLEADO E, DEPARTMENTO D
WHERE D.Dname='Research' AND D.Dnumber=E.Dno;
```

   1. Expresa la consulta usando álgebra relacional
   2. Dibuja un arbol de equilencia
   3. Transforma el árbol aplicando las heurísticas del apartado 4.3.4

4.3.7 Ejercicios

6. Usando los datos estadísticos del ejemplo de estimación de coste del apartado 4.3.5, ¿qué coste podrían tener la siguiente expresiones usando los diferentes algoritmos propuestos?

$$\sigma_{\mathbf{salario>30000}}\ (\text{EMPLEADO} \otimes_{\text{Dno=Dnumber}} \text{DEPARTAMENTO})$$