Procesamiento de Consultas

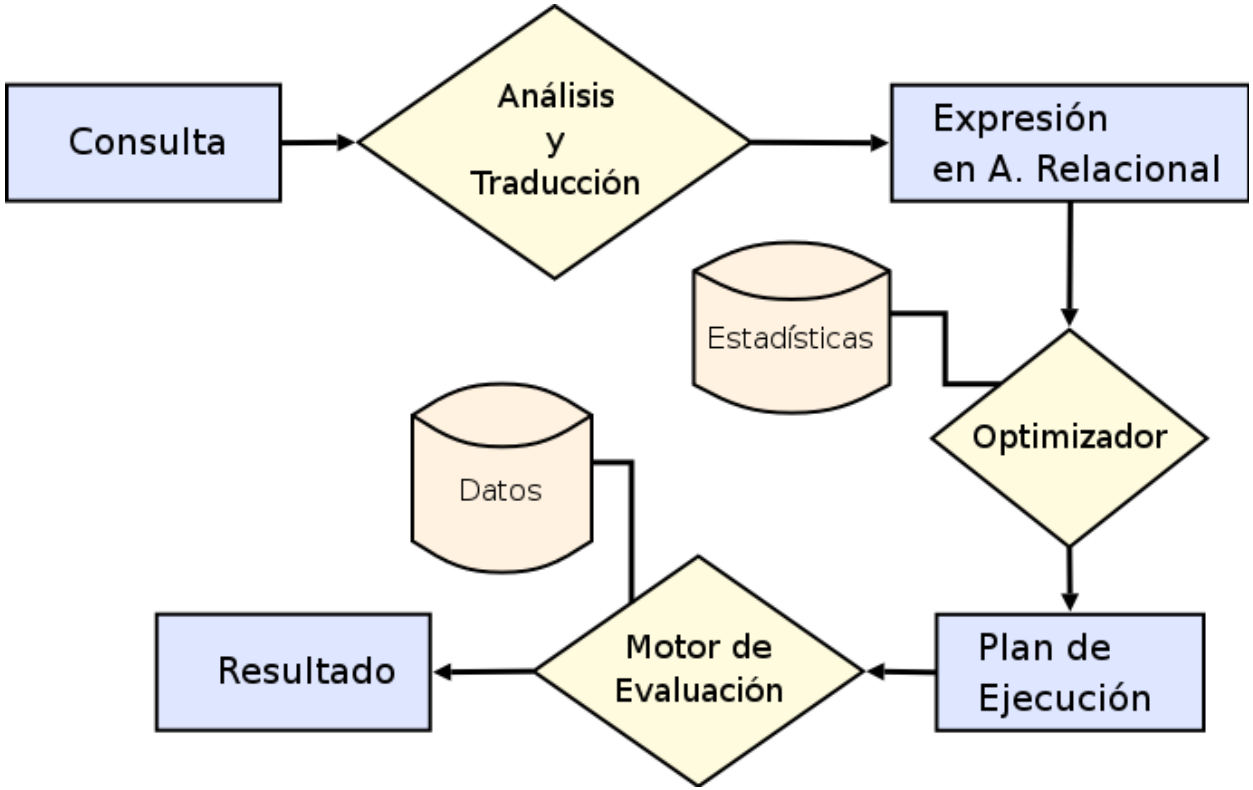
Carlos A. Olarte (carlosolarte@puj.edu.co)
BDII

Contenido

Pasos para la ejecución de una consulta

- Traducción: Generar a partir de la sentencia SQL, las instrucciones en el formato interno de la BD (álgebra relacional). Chequeo de la sintaxis, tipos de atributos, tablas,etc
- Optimización: Encontrar el plan de ejecución óptimo para llevar a cabo la consulta.
- Evaluación: Ejecutar la consulta de acuerdo al plan de ejecución. Selección de los algoritmos adecuados para llevar a cabo las operaciones

...continuación



Definiciones

- Primitivas de Evaluación: Operaciones del álgebra relacional junto con sus instrucciones de evaluación
- Plan de Ejecución: Secuencia de operaciones primitivas utilizadas para la evaluación de una consulta

Quién y cómo se optimizan las consultas

- El SGBD debe garantizar la optimización
- Se optimiza una consulta si se disminuye el número de accesos a disco
- Se pueden encontrar expresiones equivalentes del álgebra relacional que optimicen la consulta
- Escoger el algoritmo adecuado para implementar la primitiva de evaluación.
- Tipos de Optimizadores:
  - Basados en el Costo: Se basar en información estadística de las relaciones involucradas en la consulta para estimar el costo de ejecución de un plan
  - Basados en heurísticos: hacen usos de heurísticos para escoger el "mejor plan de ejecución"

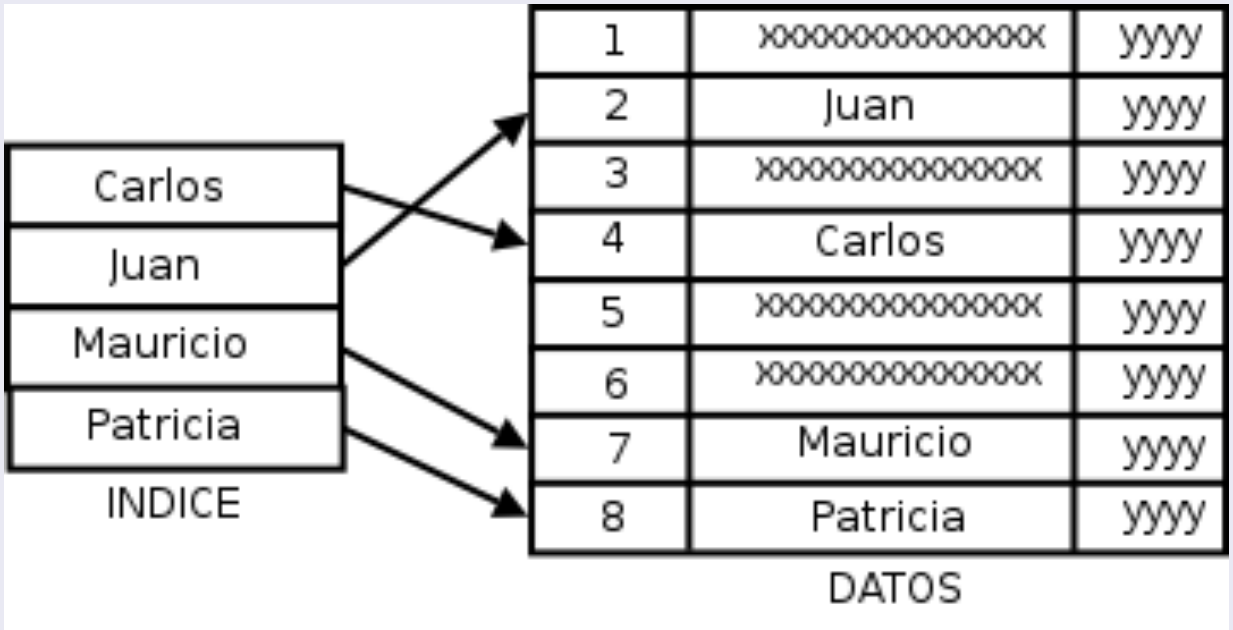
- n_r: Número de Tuplas de las relación r
- b_r: Número de bloques utilizados por r
- t_r: Tamaño en bytes de una tupla de r
- f_r: Tuplas por bloque
- $V(A, r)$: *Select count(distinct A) from r*
- AA_i: Altura del índice
- E_A: Costo estimado del algoritmo A

Dicha información estadística es calculada por el SGBD en momentos de inactividad

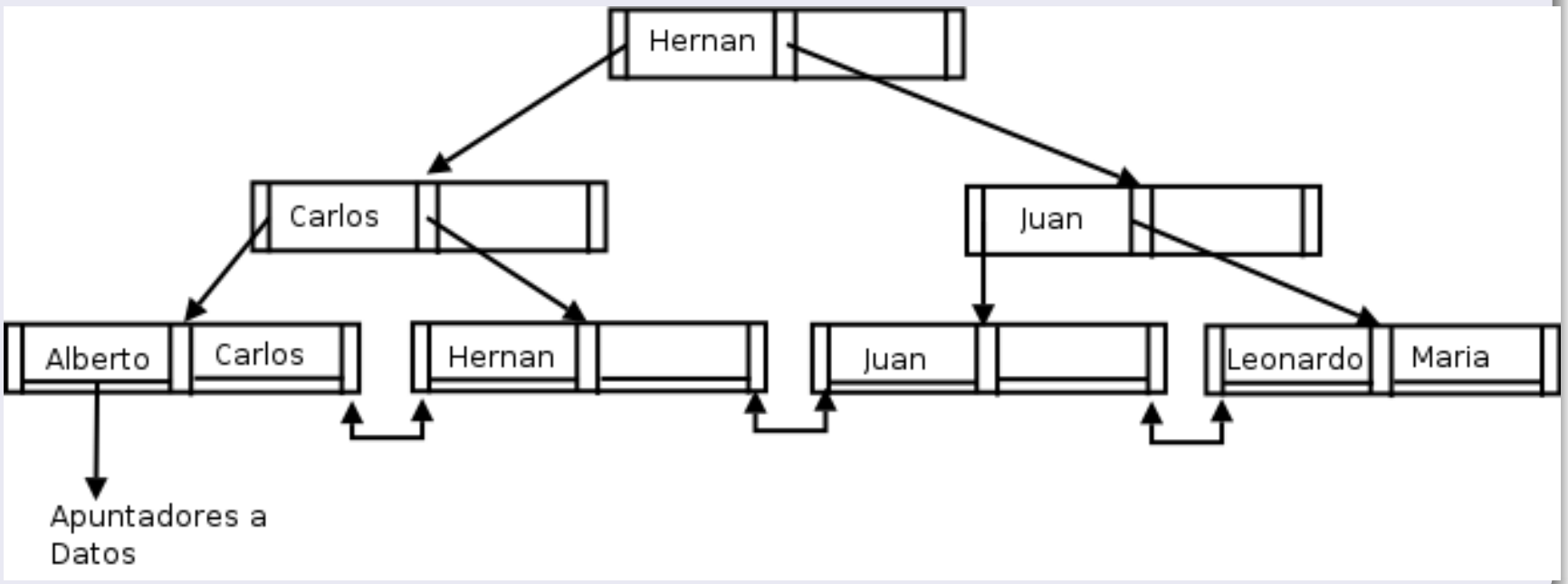
Medidas de E_A

- Lecturas de bloques en disco, ignorando latencia, tiempos diferentes de transferencia, etc. Para las B.D distribuidas y paralelas hay que tener en cuenta la cantidad de información transferida entre nodos
- El uso de la CPU no se tiene en cuenta

Índices Lineales



Arboles B+


Hernan
Carlos
Juan
Alberto
Carlos
Hernan
Juan
Leonardo
Maria
Apuntadores a
Datos

Consto de una selección ($\sigma \quad A = a$)

- Si r no está ordenada por el atributo A el número de bloques extraídos es $E_A = b_r/2$
- Si r está ordenada por el atributo A se puede utilizar una búsqueda binaria $E_A = log_2 b_r$
- Si existe un índice sobre el atributo A, se debe recorrer un camino de altura AA_i y luego leer el bloque del dato.

Consto de una selección ($\sigma_{A \leq a}$)

- Si no existen índices o un ordenamiento físico la búsqueda debe ser lineal
- Con un índice o un ordenamiento se puede encontrar el valor de a con una búsqueda binaria (o recorrido sobre el árbol) y luego realizar recorridos lineales a partir de dicha posición

Selecciones Conjuntivas ($\sigma_{\theta_1 \wedge \theta_2}$)

- Seleccionar por un índice la condición que traiga menos tuplas y probar las mismas en memoria principal si cumplen con el segundo predicado
- Utilizar un índice compuesto de los atributos involucrados en la selección (si existe)
- Seleccionar los punteros a registros de cada condición (en el índice) y luego realizar la intersección

Selecciones con Disyunciones ($\sigma_{\theta_1 \vee \theta_2}$)

- Si existe un índice por cada condición, seleccionar los apuntadores y luego realizar la unión sobre los mismos
- Si una sola condición no tiene índice, la mejor solución no es mejor que hacer una búsqueda lineal sobre la relación

Costo del Ordenamiento

Cuando se ordenan las relaciones?

- El usuario puede requerir una relación ordenada
- Una operación como la reunión puede resultar mas eficiente si se ordenan las relaciones a reunir

Como Ordenar?

- Ordenación Lógica: Creando un índice sobre la relación. Los datos físicamente siguen en desorden y leer por el índice puede causar una lectura de bloque por cada tupla
- Ordenación Física: Ordenar el archivo como tal utilizando mecanismos como el quick sort o merge sort.

Costo de la Reunión

- Reunión en blucle anidado: Realizar un *for* anidado entre las dos relaciones. $E_A = n_s \times n_r$
- Reunión en blucle anidado por bloques : Realizar un *for* anidado entre los bloques de las dos relaciones. $E_A = b_s \times b_r$
- Reunión en blucle anidado indexada : Realizar un *for* sobre los bloques de r y la búsqueda sobre s utilizando un índice. $E_A = b_r \times log_2 b_r$

- Reunión por mezcla: Ordenar las dos relaciones (de ser necesario) y luego recorrerlas linealmente encontrando las tuplas que deban estar en la reunión: $E_A = O_r + O_s + b_r + b_s$
- Reunión por asociación: Encontrar una función de partición H tal que divida las relaciones en fragmentos con tuplas con el mismo valor de función. Dicha función debe realizar una partición mas o menos homogénea sobre las relaciones

Costo de la eliminación de duplicados

- Ordenar la relación y leer secuencialmente.

Costo de las operaciones sobre conjuntos

- Ordenar (de ser necesario) cada una de las relaciones y luego recorrerlas secuencialmente encontrando las tuplas de la nueva relación

Agregación

- Realizar el ordenamiento e ir calculando la función de asociación (como la eliminación de duplicados)

Evaluación de Expresiones

- Las expresiones se pueden evaluar por medio de dos métodos: Materialización: En el cual los resultados parciales se escriben en una tabla temporal, implicando aumentar la escritura de los bloques a disco en el cálculo de eficiencia
- Encauzamiento: Los resultados generados van alimentando la operación siguiente. Se puede presentar el esquema bajo demanda en el cual las operaciones superiores hacen peticiones de tuplas a las inferiores y el esquema Desde los procedimientos, los operadores inferiores generan tuplas sin ser requeridas

Transformación de Expresiones

- A partir de las reglas de equivalencia sobre el álgebra relacional, es posible encontrar expresiones equivalentes a la expresión inicial y que tengan un menor costo.

- $\sigma_{\theta_1 \wedge \theta_2}(E) \equiv \sigma_{\theta_1}(\sigma_{\theta_2}(E))$
- $\sigma_{\theta_1 \wedge \theta_2}(E) \equiv \sigma_{\theta_2 \wedge \theta_1}(E)$
- $\pi_{L_1}(\pi_{L_2}(...\pi_{L_n}(E)..)) \equiv \pi_{L_1}(E)$
- $\sigma_{\theta}(E_1 \times E_2) \equiv E_1 \bowtie_{\theta} E_2$
- $\sigma_{\theta_1}(E_1 \bowtie_{\theta_2} E_2) \equiv E_1 \bowtie_{\theta_1 \wedge \theta_2} E_2$
- $E_1 \bowtie_{\theta} E_2 \equiv E_2 \bowtie_{\theta} E_1$
- $(E_1 \bowtie_{\theta_1} E_2) \bowtie_{\theta_2} E_3 \equiv E_1 \bowtie_{\theta_1} (E_2 \bowtie_{\theta_2} E_3)$ (Siempre y cuando θ_2 solo tenga atributos en común con E_2 y E_3)

- $\sigma_{\theta_0}(E_1 \bowtie_{\theta_1} E_2) \equiv \sigma_{\theta_0}(E_1) \bowtie_{\theta_1} E_2$ (Siempre que θ_0 solo incluye atributos de E_1
- $\sigma_{\theta_1 \wedge \theta_2}(E_1 \bowtie_{\theta_3} E_2) \equiv \sigma_{\theta_1}(E_1) \bowtie_{\theta_3} \sigma_{\theta_2}(E_2)$ (Siempre que θ_1 (resp. θ_2) tenga solo atributos de $E1$ (resp. E_2)
- $\pi_{L_1 \cup L_2}(E_1 \bowtie_{\theta} E_2) \equiv (\pi_{L_1} E_1) \bowtie_{\theta} (\pi_{L_2} E_2)$ (Si L_1 (resp L_2) son atributos de E_1 (resp E_2)
- $E_1 \cup E_2 \equiv E_2 \cup E_1$ y $E_1 \cap E_2 \equiv E_2 \cap E_1$
- $(E_1 \cup E_2) \cup E_3 \equiv E_1 \cap (E_2 \cup E_3)$ y $(E_1 \cap E_2) \cap E_3 \equiv E_1 \cap (E_2 \cap E_3)$

Proceso de Optimización basado en el Costo

- Generar Expresiones del álgebra relacional a partir de la expresión inicial (pueden llegar a ser del orden de $n!$)
- Elegir la expresión de menor costo (en la medida de lo posible)
- Seleccionar los algoritmos para llevar a cabo cada operación.

Optimización heurística

- Se apoyan en heurísticos tales como:
  - Se deben realizar las selecciones lo mas pronto posible (Ej: seleccionar las tuplas primero y luego realizar la reunión)
  - Las proyecciones se deben realizar al principio pues reducen el tamaño de las relaciones.

Algoritmo para un optimizador heurístico

1. Deshacer las selecciones conjuntivas en selecciones simples
2. Mover las operaciones de selección lo mas abajo posible del árbol de expresiones
3. Agrupar las reuniones con las selecciones de tal manera que se reduzcan el número de tuplas a reunir.
4. Sustituir los productos cartesianos por una reunión (combinándolos con selecciones)
5. Mover tan abajo del árbol como sea posible las proyecciones de relaciones
6. Identificar los subárboles cuyas operaciones se puedan encauzar.

A Es una herramienta de Oracle que permite:

- Medir ocupación de CPU por consulta
- Medir accesos a disco
- Medir número de registros procesados
- Visualizar el plan de ejecución de la consulta

- Utilidad que permite conocer el plan de ejecución de una consulta
- Prerequisitos: la tabla plan_table

plan_table

```
CREATE TABLE plan_table (statement_id  VARCHAR2(30),
timestamp  DATE, remarks  VARCHAR2(80),
operation  VARCHAR2(30),  options  VARCHAR2(30),
object_node  VARCHAR2(128), object_owner  VARCHAR2(30),
object_name VARCHAR2(30), object_instance  NUMERIC,
object_type VARCHAR2(30), optimizer VARCHAR2(255),
search_columns   NUMERIC,id  NUMERIC,
parent_id  NUMERIC, position   NUMERIC,
cost   NUMERIC, cardinality NUMERIC,
bytes  NUMERIC, other_tag   VARCHAR2(255),
 other  LONG);
```

Continuación...

Posibles resultados para Explain FOR

- Nested Loop: Operaciones de Join
- Partition: Operaciones sobre una partición específica
- Projection: Proyección
- Remote: Extraer datos de una base de datos remota
- Sequence: Operación con una secuencia
- Sort: Puede ser Aggregate (agrupaciones), Unique (Eliminación de duplicados), Group By, Join (Ordenar antes de hacer merge), Order by.
- Table Access: Puede ser Full (lectura de toda la tabla), By Index (lectura dado un índice), By rowid (Lectura dado el número de registro)

- Permite visualizar los resultados del archivo generado por el SQL trace
- Permite conocer el plan de ejecución de una consulta (explain plan for)

Pasos a Seguir

1. Modificar los siguientes parámetros de inicialización de la base de datos: (init¿ora)
   - timed_statistics = true
   - user_dump_file=/_/_/dir
   - max_dump_file_size=5M
2. Habilitar SQL Trace para la sesión de usuario: *SQL¿ ALTER SESION SET SQL_TRACE=TRUE*
3. Ejecutar la(s) consulta(s)
4. Ejecutar tkproof sobre el archivo generado: *tkproof arch.trc arch.txt*

Resultados de TkProof

- Parse: Fase de transformar la expresión en un plan
- Execute: Ejecución del plan
- Fetch: Solo para los select
- Count: Número de veces que se ejecutó
- CPU: Seg transcurridos en CPU
- Elapse: Seg que duró la consulta
- Disk: Bloques transferidos
- Rows: Filas afectadas

Ejemplos

```
alter session set sql_trace=true;
select codigo,nombre from padre;
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch       25      0.00       0.00          0         25          2         350
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       27      0.00       0.00          0         25          2         350

Rows     Row Source Operation
-------  ---------------------------------------------------
    350  TABLE ACCESS FULL PADRE
```

... continuación

```
select codigo,nombre from padre where codigo = 5

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  TABLE ACCESS BY INDEX ROWID PADRE
            1   INDEX UNIQUE SCAN (object id 17709)

select codigo,nombre from padre where nombre = 'C'

Rows     Row Source Operation
-------  ---------------------------------------------------
      0  TABLE ACCESS FULL PADRE

create index padre_nom_idx on padre(nombre)

select codigo,nombre from padre where nombre = 'A'

Rows     Row Source Operation
-------  ---------------------------------------------------
      0  TABLE ACCESS BY INDEX ROWID PADRE
            0   INDEX RANGE SCAN (object id 17719)

drop index padre_nom_idx
```

... continuación

```
select nombre from padre union select nombre from hijo
Rows     Row Source Operation
-------  ---------------------------------------------------
   2800  SORT UNIQUE
   3150   UNION-ALL
    350    TABLE ACCESS FULL PADRE
   2800    TABLE ACCESS FULL HIJO

********************************************************************************
select h.codigo, h.nombre, p.nombre from padre p , hijo h
where p.codigo = h.padre

Rows     Row Source Operation
-------  ---------------------------------------------------
   2800  NESTED LOOPS
   2800   TABLE ACCESS FULL HIJO
   2800   TABLE ACCESS BY INDEX ROWID PADRE
   2800    INDEX UNIQUE SCAN (object id 17709)
********************************************************************************
select h.codigo, h.nombre, p.nombre from padre p , hijo h
where p.nombre = h.nombre

Rows     Row Source Operation
-------  ---------------------------------------------------
    350  MERGE JOIN
   2800   SORT JOIN
   2800    TABLE ACCESS FULL HIJO
    350   SORT JOIN
    350    TABLE ACCESS FULL PADRE
```

... continuación

```
select codigo,nombre from padre order by nombre
Rows     Row Source Operation
-------  ---------------------------------------------------
    350  SORT ORDER BY
    350   TABLE ACCESS FULL PADRE

********************************************************************************

select codigo,nombre from padre order by codigo

Rows     Row Source Operation
-------  ---------------------------------------------------
    350  TABLE ACCESS BY INDEX ROWID PADRE
    350   INDEX FULL SCAN (object id 17709)

********************************************************************************

select p.codigo,p.nombre,h.nombre from padre p, hijo h
where p.codigo = h.padre and p.nombre = 'A'

Rows     Row Source Operation
-------  ---------------------------------------------------
      0  NESTED LOOPS
   2800   TABLE ACCESS FULL HIJO
      0   TABLE ACCESS BY INDEX ROWID PADRE
   2800    INDEX UNIQUE SCAN (object id 17709)

********************************************************************************
```

... continuación

```
select p.codigo,p.nombre,h.nombre from padre p, hijo h
where p.nombre = h.nombre and p.codigo = h.padre

Rows     Row Source Operation
-------  ---------------------------------------------------
    350  NESTED LOOPS
   2800   TABLE ACCESS FULL HIJO
    350   TABLE ACCESS BY INDEX ROWID PADRE
   2800    INDEX UNIQUE SCAN (object id 17709)

********************************************************************************

select codigo,nombre from hijo where codigo <= 5
and nombre like '%A%'

Rows     Row Source Operation
-------  ---------------------------------------------------
      0  TABLE ACCESS BY INDEX ROWID HIJO
      5   INDEX RANGE SCAN (object id 17711)

********************************************************************************

select codigo,nombre from hijo where codigo <= 5
or nombre like '%A%'

Rows     Row Source Operation
-------  ---------------------------------------------------
      5  TABLE ACCESS FULL HIJO

********************************************************************************
```

... continuación

```
select nombre from hijo
minus
select nombre from padre

Rows     Row Source Operation
-------  ---------------------------------------------------
   2450  MINUS
   2800   SORT UNIQUE
   2800    TABLE ACCESS FULL HIJO
    350   SORT UNIQUE
    350    TABLE ACCESS FULL PADRE

********************************************************************************

select r.codigo,h.nombre,p.nombre
from relacion r, padre p, hijo h
where r.hijo = h.codigo and h.padre = p.codigo

Rows     Row Source Operation
-------  ---------------------------------------------------
  11200  NESTED LOOPS
  11200   NESTED LOOPS
  11200    TABLE ACCESS FULL RELACION
  11200    TABLE ACCESS BY INDEX ROWID HIJO
  11200     INDEX UNIQUE SCAN (object id 17711)
  11200   TABLE ACCESS BY INDEX ROWID PADRE
  11200    INDEX UNIQUE SCAN (object id 17709)

********************************************************************************
```

... continuación

```
select r.codigo,h.nombre,p.nombre
from relacion r, padre p, hijo h
where r.hijo = h.codigo
and h.padre = p.codigo and h.nombre like '%A%'

Rows     Row Source Operation
-------  ---------------------------------------------------
      0  NESTED LOOPS
      0   NESTED LOOPS
  11200    TABLE ACCESS FULL RELACION
      0    TABLE ACCESS BY INDEX ROWID HIJO
  11200     INDEX UNIQUE SCAN (object id 17711)
      0   TABLE ACCESS BY INDEX ROWID PADRE
      0    INDEX UNIQUE SCAN (object id 17709)
```