OPTIMIZACIÓN DE CONSULTAS

Bases de Datos 1

Álgebra Relacional

El conjunto de operaciones básicas, empleadas en las bases de datos relacionales, se pueden representar mediante dos lenguajes formales: el álgebra relacional y el cálculo relacional.

El álgebra relacional cobra importancia ya que proporciona un fundamento formal a las operaciones y es usada para la implementación y optimización de las consultas por parte de los Sistemas Manejadores de Bases de Datos (DBMS).

El álgebra relacional está compuesta por una colección de operadores que se aplican a los conjuntos de datos (tablas de la Base de Datos), dando como resultado nuevos conjuntos y tienen por objetivo la formulación de consultas sobre la Base de Datos.

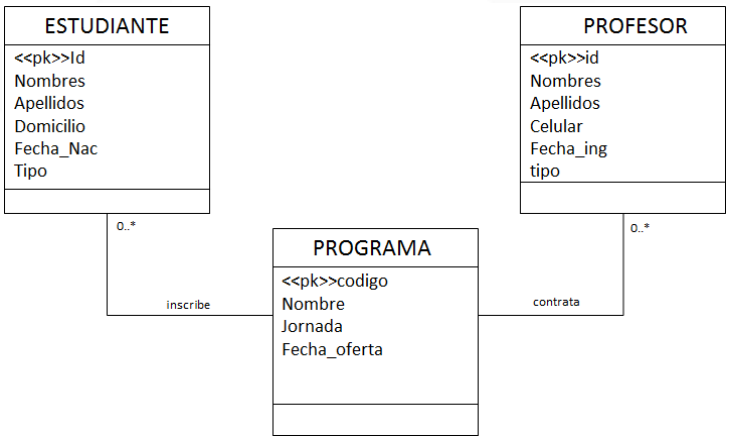
La potencia expresiva de operadores del álgebra relacional es la misma que la del cálculo relacional, el cual está basado en la lógica de predicados de primer orden; lo que significa que se pueden expresar las mismas consultas empleando operadores de algebra relacional o de cálculo relacional.

Cood definió originalmente 8 operadores de algebra relacional, que se pueden clasificar en dos categorías:

- Operadores tradicionales de conjuntos: unión, intersección, diferencia y producto cartesiano.
- Operadores relacionales especiales: selección, proyección, combinación y división.

Posteriormente, otros autores (como Lacroix y Date) incorporaron otros operadores, pero estas adiciones no serán tratadas en este documento.

Las siguientes tablas se emplearán para ejemplificar el uso de los diferentes operadores del algebra relacional.



Cada tabla incluye los registros que se presentan a continuación:

ESTUDIANTES

id	nombres	apellidos	domicilio	fecha_nac	codigo	tipo
51915850	Martha	Zea	Cll 2 # 5-30	12/oct/1975	202	postgrado
12587498	Lucía	López	Cra 21 # 2-61	10/ene/1968	201	postgrado
78965874	José	Roa	Av 1 # 8-145	23/nov/1980	143	pregrado
87915850	Marcos	García	Cra 12 # 5-30	12/feb/1978	201	postgrado
12987498	Jorge	Lima	Cll 1 # 2-76	15/oct/1985	143	pregrado
97765874	José	Vélez	Av 41 # 8-12	23/nov/1980	142	pregrado

PROFESORES

id	nombres	apellidos	Celular	fecha_ing	codigo	tipo
98745877	Jorge	López	3115974598	12/oct/1970	141	pregrado
25874557	Clara	Bastos	3159625894	10/ene/2005	201	postgrado
96587451	José	Pérez	3119658741	21/oct/1971	143	pregrado
63259757	Maria	Gómez		12/feb/1980	201	postgrado
56987451	Jair	Cano		15/oct/1996	202	postgrado
12587498	Lucía	López	3106985478	10/ene/2004	143	pregrado

PROGRAMAS

codigo	nombre	jornada	fecha_oferta
140	Ingeniería Mecánica	diurna	20/ene/1980
141	Ingeniería Mecánica	nocturna	1/jul/1985
142	Ingeniería Electrónica	diurna	15/ene/1970
143	Ingeniería Informática	diurna	1/jul/1971
201	Calidad de Software	diurna	5/ene/1976
202	Telemática	diurna	12/jul/1975

OPERADORES

Selección (σ)

La selección o restricción, es una relación que da como resultado un subconjunto de tuplas o registros, que satisface la expresión indicada, por ejemplo:

$\sigma_{id=51915850}$ (ESTUDIANTES)

Resultado:

51915850 Martha Zea Cll 2 # 5-30 12/oct/1975 143 postgrado

En el caso anterior, se selecciona del conjunto ESTUDIANTES, aquellos cuyo id=51915850

Para la construcción de la expresión se pueden emplear los operadores relacionales: >, <, =, ≤,≥, ≠; y los operadores lógicos Y, O, NO.

Otros ejemplos:

$\sigma_{codigo>=142\ AND\ codigo<=\ 144}$ (PROGRAMAS)

Resultado:

142	Ingeniería Electrónica	diurna	15/ene/1970
143	Ingeniería Informática	diurna	1/jul/1999
144	Comunicación Social	diurna	5/ene/1981

$\sigma_{id>745489754}$ (PROFESORES)

Resultado:

98745877	Jorge	López	3115974598	12/oct/1970	141	pregrado
96587451	José	Pérez	3119658741	21/oct/1971	143	pregrado

Proyección (π)

La proyección de una relación sobre un subconjunto de los atributos de una tabla, de forma que se eliminan los registros duplicados que pudieran resultar. Por ejemplo:

$\pi_{nombres,apellidos}$ (ESTUDIANTES)

Resultado:

Martha	Zea
Lucía	López
José	Roa
Marcos	García
Jorge	Lima
José	Vélez

En el caso anterior, se ha realizado una proyección de los atributos nombres y apellidos de la tabla ESTUDIANTES.

$\pi_{codigo,nombre}$ (PROGRAMAS)

Resultado:

140	Ingeniería Mecánica
141	Ingeniería Mecánica
142	Ingeniería Electrónica
143	Ingeniería Informática
201	Calidad de Software
202	Telemática

Proyección y Selección

Tal como es común en el álgebra, los operadores pueden combinarse para producir resultados diferentes; debe recordarse emplear apropiadamente los paréntesis para establecer el orden de ejecución de las operaciones; por ejemplo:

$\pi_{nombres,apellidos}$ ($\sigma_{id=51915850}$ (ESTUDIANTES))

Resultado:

Martha Zea

En el caso anterior, se hace inicialmente una selección de la tabla de ESTUDIANTES, bajo la condición id=51915850, es decir los estudiantes que tengan la identificación definida; posteriormente, sobre el subconjunto resultante se realiza la proyección de nombres y apellidos.

En el ejemplo siguiente se repite la operación anterior, pero se asigna el resultado de la operación inicial a un nuevo subconjunto, que se emplea en la otra operación.

EST $\leftarrow \sigma_{id=51915850}$ (ESTUDIANTES)
NOM $\leftarrow \pi_{nombres, apellidos}$ (EST)

Unión (U)

La Unión de dos relaciones que tengan esquemas compatibles (el mismo número y tipo de columnas) será una nueva relación (o conjunto) con los elementos que pertenecen a una u otra relación; los elementos duplicados serán incluidos solamente una vez, dado que se trata de conjuntos.

PROF $\leftarrow \pi_{nombres,apellidos}$ ($\sigma_{tipo='pregrado'}$ (PROFESORES))

Resultado:

Jorge	López
José	Pérez
Lucía	López

ESTUD ← $\pi_{\text{nombres,apellidos}}$ ($\sigma_{\text{tipo='postgrado'}}$ (ESTUDIANTES))

Resultado:

Martha Zea
Lucía López
Marcos García

UNIDOS ← PROF U ESTUD

Resultado:

Jorge López
José Pérez
Lucía López
Martha Zea
Marcos García

Intersección (∩)

La intersección de dos relaciones, compatibles en su esquema, es otra relación definida sobre el mismo esquema, cuya extensión estará constituida por los registros que pertenezcan a ambas relaciones.

PROF ← $\pi_{\text{nombres,apellidos}}$ ($\sigma_{\text{tipo='pregrado'}}$(PROFESORES))

ESTUD ← $\pi_{\text{nombres,apellidos}}$ ($\sigma_{\text{tipo='postgrado'}}$(ESTUDIANTES))

INTERSECTADO ← PROF ∩ ESTUD

Resultado:

Lucía López

Diferencia (-)

La diferencia de dos relaciones con esquemas compatibles es otra relación (o conjunto) definida sobre el mismo esquema de la relación, que incluirá los elementos que pertenezcan a la primera relación y no a la segunda.

PROF ← $\pi_{\text{nombres,apellidos}}$ ($\sigma_{\text{tipo='pregrado'}}$(PROFESORES))

ESTUD ← $\pi_{\text{nombres,apellidos}}$ ($\sigma_{\text{tipo='postgrado'}}$(ESTUDIANTES))

DIFERENCIA ← ESTUD – PROF

Resultado:

Martha Zea
Marcos García

Producto Cartesiano Generalizado (x)

El producto cartesiano generalizado para dos relaciones de cardinalidad m_1 y m_2 respectivamente, es una relación de extensión m_1 x m_2, constituida por los elementos que resultan de la concatenación de todos los elementos de la primera relación con los elementos de la segunda relación.

ESTUDIANTES X PROGRAMA:

51915850	Martha	Zea	Cll 2 # 5-30	12-oct-75	202	postgrado	140	Ingeniería Mecánica	diurna	20-ene-80
51915850	Martha	Zea	Cll 2 # 5-30	12-oct-75	202	postgrado	141	Ingeniería Mecánica	nocturna	01-jul-85
51915850	Martha	Zea	Cll 2 # 5-30	12-oct-75	202	postgrado	142	Ingeniería Electrónica	diurna	15-ene-70
51915850	Martha	Zea	Cll 2 # 5-30	12-oct-75	202	postgrado	143	Ingeniería Informática	diurna	01-jul-71
51915850	Martha	Zea	Cll 2 # 5-30	12-oct-75	202	postgrado	201	Calidad de Software	diurna	05-ene-76
51915850	Martha	Zea	Cll 2 # 5-30	12-oct-75	202	postgrado	202	Telemática	diurna	12-jul-75
12587498	Lucía	López	Cra 21 # 2-61	10-ene-68	201	postgrado	140	Ingeniería Mecánica	diurna	20-ene-80
12587498	Lucía	López	Cra 21 # 2-61	10-ene-68	201	postgrado	141	Ingeniería Mecánica	nocturna	01-jul-85
12587498	Lucía	López	Cra 21 # 2-61	10-ene-68	201	postgrado	142	Ingeniería Electrónica	diurna	15-ene-70
12587498	Lucía	López	Cra 21 # 2-61	10-ene-68	201	postgrado	143	Ingeniería Informática	diurna	01-jul-71
12587498	Lucía	López	Cra 21 # 2-61	10-ene-68	201	postgrado	201	Calidad de Software	diurna	05-ene-76
12587498	Lucía	López	Cra 21 # 2-61	10-ene-68	201	postgrado	202	Telemática	diurna	12-jul-75
...	...	...	...	...	...	...	...	...	...	...

En total, tendrían 36 filas (6 estudiantes x 6 programas).

Sin embargo, sobre el producto cartesiano se puede aplicar la operación de selección y/o proyección para limitar los resultados, por ejemplo:

$$\pi_{\text{nombres,apellidos}} (\sigma_{\text{programas.codigo = estudiantes.codigo}} (\sigma_{\text{programas.nombre='Ingenieria Informatica'}} (\text{ESTUDIANTES x PROGRAMAS})))$$

Resultado:

José Roa
Jorge Lima

NOTA: Debe tenerse claro que para realizar la operación anterior, primero se realiza el producto cartesiano de las dos tablas, luego sobre el resultado, se aplica la selección de las tuplas o registros que tengan como nombre del programa “Ingeniería Informática”, sobre este resultado, se aplica la selección de tuplas para las que el código en la tabla programa coincida con el código en la tabla de estudiantes y finalmente sobre ese resultado se hace proyección de los campos nombres y apellidos.

Combinación ( )

La combinación $\bowtie$ o join, se aplica sobre dos relaciones con respecto a cierta condición de combinación, son los pares de registros conectados (como en el producto cartesiano) pero de forma que cumplen la condición indicada.

La forma general sería:

$$R1 \bowtie_{(condicion)} R2$$

Se podría agregar una condición al caso anterior para disminuir el número de filas resultantes:

$$ESTUDIANTES \bowtie_{ESTUDIANTES.codigo = PROGRAMAS.codigo} PROGRAMAS$$

El resultado serían solamente 6 registros:

51915850	Martha	Zea	Cll 2 # 5-30	12-oct-75	202	postgrado	202	Telemática	diurna	12-jul-75
12587498	Lucía	López	Cra 21 # 2-61	10-ene-68	201	postgrado	201	Calidad de Software	diurna	05-ene-76
78965874	José	Roa	Av 1 # 8-145	23-nov-80	143	pregrado	143	Ingeniería Informática	diurna	01-jul-71
87915850	Marcos	García	Cra 12 # 5-30	12-feb-78	201	postgrado	201	Calidad de Software	diurna	05-ene-76
12987498	Jorge	Lima	Cll 1 # 2-76	15-oct-85	143	pregrado	143	Ingeniería Informática	diurna	01-jul-71
97765874	José	Vélez	Av 41 # 8-12	23-nov-80	142	pregrado	142	Ingeniería Electrónica	diurna	15-ene-70

División (:)

La división de una relación (dividendo) sobre otra (divisor) es una relación (cociente) tal que, al realizarse la combinación con el divisor, todas las tuplas resultantes se encuentren en el dividendo.

Para el siguiente ejemplo suponga que se han redefinido las tablas profesores y programas así:

PROGRAMAS:

codigo
143
201

PROFESORES:

id	nombres	apellidos	celular	fecha_ing	codigo	tipo
98745877	Jorge	López	3115974598	12/oct/1970	141	pregrado
25874557	Clara	Bastos	3159625894	10/ene/2005	201	postgrado
96587451	José	Pérez	3119658741	21/oct/1971	143	pregrado
12587498	Lucía	López	3106985478	10/ene/2004	201	pregrado
56987451	Jair	Cano		15/oct/1996	202	postgrado
12587498	Lucía	López	3106985478	10/ene/2004	143	pregrado

La operación PROFESORES: PROGRAMAS, tendría como resultado:

id	nombres	apellidos	celular	fecha_ing	tipo
12587498	Lucía	López	3106985478	10/ene/2004	pregrado

Este resultado corresponde a lo(s) registro(s) de la tabla profesores que cumplen con lo definido en la tabla programas, es decir que tenga relación con los códigos 143 y 201.

ÁRBOL DE CONSULTA

Un árbol de consultas es una estructura de datos en árbol que equivale a una expresión de álgebra relacional. Representa las relaciones (tablas) de entrada a una consulta como los nodos hoja del árbol y las operaciones del algebra relacional como nodos internos.

Una ejecución del árbol de consultas consiste en ejecutar una operación en un nodo interno siempre que estén disponibles sus operandos y después reemplazar ese nodo interno por la relación que resulta de la ejecución de la operación. La ejecución finaliza cuando se ejecuta el nodo raíz y se genera la relación resultante de la consulta.

Por ejemplo:

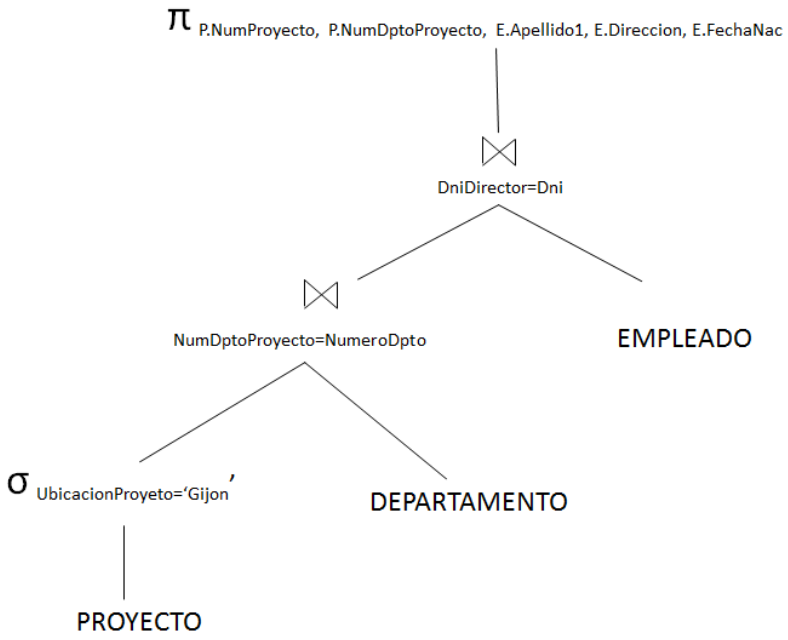
Para la consulta: "*Para cada proyecto localizado en Gijon, obtener el número de proyecto, el número de departamento controlador y el apellido del responsable del departamento, su dirección y su fecha de nacimiento*"

```
Select   P.NumProyecto, P.NumDptoProyecto, E.Apellido1, E.Dirección, E.FechaNac
From     PROYECTO AS P, DEPARTAMENTO AS D, EMPLEADO AS D
Where    P.NumDptoPoryecto = D.NumeroDpto AND D.DniDirector = E.Dni AND
         P.UbicacionProyecto='Gijon'
```

La expresión correspondiente en álgebra relacional sería:

$$\pi_{\text{NumProyecto, NumDptoProyecto, Apellido1, Direccion, FechaNac}}(((\sigma_{\text{UbicacionProyeto='Gijon'}}(\text{PROYECTO})) \bowtie_{\text{NumDptoProyecto=NumeroDpto}} (\text{DEPARTAMENTO})) \bowtie_{\text{DniDirector=Dni}} (\text{EMPLEADO}))$$

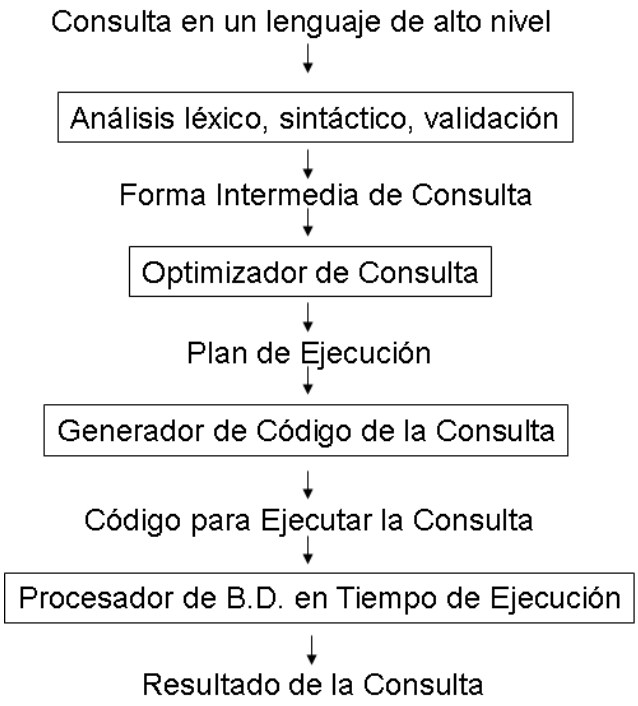
Y el árbol de consulta correspondiente a la expresión del álgebra relacional, quedaría así:



OPTIMIZACIÓN DE CONSULTAS[1]

El BDMS procesa, optimiza y ejecuta las consultas de alto nivel. El proceso seguido, se presenta en la figura siguiente.

- El analizador léxico identifica los elementos del lenguaje, tales como palabras reservadas.
- El analizador sintáctico comprueba la sintaxis de la consulta, para verificar que cumpla las reglas establecidas.
- La consulta es validada, comprobando que los nombres de los atributos y relaciones sean válidos y tengan significado semántico dentro de la base de datos.
- Se crea una representación interna de la consulta, denominada árbol de consultas. (también se puede utilizar un grafo de consulta).
- El DBMS desarrolla un plan de ejecución para obtener el resultado solicitado a partir de los datos de la base de datos. Generalmente existen múltiples formas de ejecución y la selección de la más apropiada se denomina optimización.

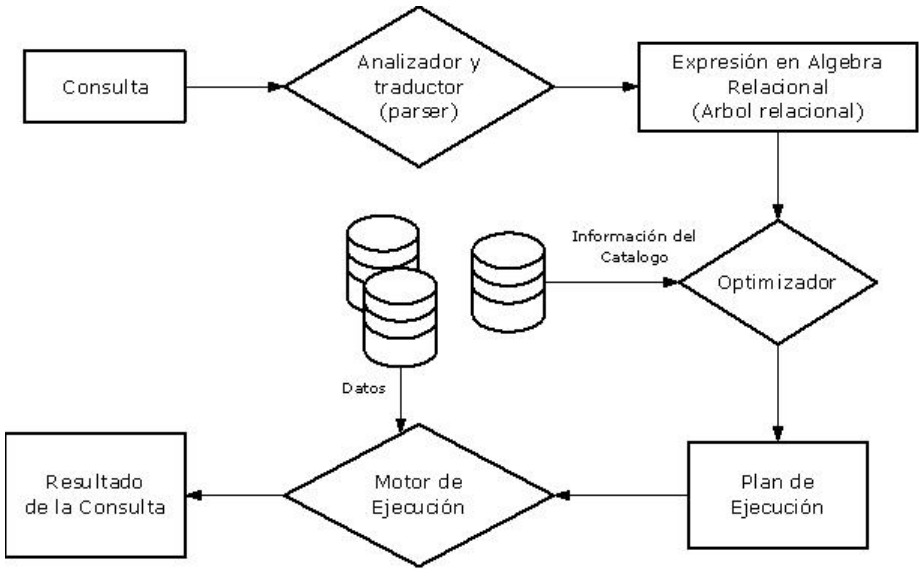


- El optimizador de consultas genera un plan de ejecución y el generador de código se encarga de generar el condigo para cumplir dicho plan.
- El procesador de Base de datos en tiempo de ejecución tiene como cometido la ejecución del código (en modo interpretado o en modo compilado).

[1] Basado en: *ELMASRI, Ramez. NAVATHE, Shamkant B. Fundamentos de Sistemas de Bases de Datos 5ª Edición. Ed. Adisson-Wesley Iberoamerica.Madrid. 2007*

El término optimización no es exacto, ya que en algunos casos, el plan de ejecución elegido es “razonablemente eficiente”, pero no necesariamente óptimo; esto es, debido a que, en muchos casos, la búsqueda de la ejecución óptima consume mucho tiempo de máquina.

Traducción de consultas SQL al álgebra relacional



Las consultas de SQL se traducen a una expresión equivalente de álgebra relacional, representada como una estructura de árbol de consultas, para posteriormente ser optimizada. Las consultas SQL se descomponen en **bloques de consulta** que forman las unidades básicas que se pueden traducir a operadores algebraicos. Para la representación de las consultas se emplea el álgebra extendida, de forma que se puedan incluir los operadores de agregación (MAX, MIN, SUM y COUNT).

Ejemplo:

```
Select  Apellido1, Nombre
From            Empleado
Where                   Sueldo > (    Select     MAX (Sueldo)
                              From       Empleado
                              Where      Dno = 5);
```

En este caso se tienen dos consultas, cada una de las cuales puede representarse en algebra relacional:

$$c = \tau \text{ MAX Sueldo}(\sigma_{Dno=5} (\text{Empleado}))$$

$$\pi_{Apellido1,\ Nombre}(\sigma_{Sueldo>c}(\text{Empleado}))$$

Utilización de la Heurística en la Optimización de Consultas.

Existen varias técnicas de optimización que aplican reglas heurísticas para modificar la representación interna de una consulta para mejorar su ejecución.

Una de las principales reglas heurísticas es aplicar las operaciones SELECT y PROJECT antes de aplicar JOIN u otras operaciones binarias, ya que el tamaño del fichero resultante de una operación binaria, es, por lo general, una función multiplicativa de los tamaños de los ficheros de entrada. Las operaciones SELECT y JOIN reducen el tamaño de un fichero y deberían ser aplicadas antes del JOIN.

Un árbol de consultas se utiliza para representar una expresión en álgebra relacional, mientras un grafo se emplea para representar una expresión en cálculo relacional.

Optimización heurística de los árboles de consultas

Una consulta puede generar varias expresiones en álgebra relacional y por lo tanto, muchos árboles de consulta distintos pero equivalentes.

El analizador de consultas generará un árbol de consultas inicial, equivalente a la consulta SQL. El optimizador debe incluir reglas de equivalencia entre las expresiones de álgebra relacional que puedan ser aplicadas al árbol inicial. Las reglas de optimización heurística de consultas utilizarán estas expresiones de equivalencia para transformar el árbol inicial en el árbol de consultas final optimizado.

Ejemplo: Para la consulta “Encontrar los apellidos de los empleados nacidos después de 1957 que trabajan en un proyecto llamado Aquarius”.

```
Select      Apellido1
From        Empelado, Trabaja_En, Proyecto
Where       NombreProyecto='Aquarius' AND NumProyecto=Pno AND DniEmpleado = Dni
            AND FechaNac > '1957-12-31'
```

$$\pi_{\text{Apellido1}} (\sigma_{\text{NombreProeycto='Aquarius' AND numProyecto = NumProy AND DniEmpleado = Dni AND fecha\_NAC>'1957-12-31'}} (\text{PROYECTO x EMPLEADO X TRABAJA\_EN}))$$

Reglas de transformación para las operaciones de álgebra relacional.
Existen muchas reglas para transformar las operaciones del álgebra relacional en expresiones equivalentes. En este sentido se consideran equivalentes dos operaciones que den como resultado el mismo conjunto de atributos aunque se encuentren en diferente orden.

1. Cascada de σ: Una condición de selección conjuntiva puede ser dividida en una cascada (secuencia) de operaciones σ individuales.

$$\sigma_{condicion1\ AND\ condicion2\ AND\ condición3}(X) \equiv \sigma_{condicion1}(\sigma_{condicion2}(\sigma_{condición3}(X)))$$

2. Conmutatividad de σ: La operación σ es conmutativa.

$$\sigma_{condicion2}(\sigma_{condición3}(X)) \equiv \sigma_{condicion3}(\sigma_{condición2}(X))$$

3. Cascada de π: En una cascada (secuencia) de operaciones π, todas, excepto la última pueden ser ignoradas.

$$\pi_{dato1}(\pi_{dato1,\ dato2}(\pi_{dato1,\ dato2,\ dato3}(X))) \equiv \pi_{dato1}(X)$$

4. Conmutación de σ por π: Si la condición c de selección incluye sólo los atributos A1, …An en la lista de proyección, las dos operaciones pueden ser conmutadas.

$$\pi_{atributo\ 1}(\sigma_{atributo1\ =\ valor}(X)) \equiv \sigma_{atributo1\ =\ valor}(\pi_{atributo\ 1}(X))$$

5. Conmutatividad de $\bowtie$ y X. La operación $\bowtie$ es conmutativa, igual que la operación X.

$$(R \otimes_{A=B} S) \otimes_{c=d} T \equiv R \otimes_{A=B} (S \otimes_{c=d} T)$$

$$(R \times S) \times T \equiv R \times (S \times T)$$

6. Conmutación de σ con $\bowtie$ (o X): Si todos los atributos de la condición de selección c involucran sólo a los atributos de una de las relaciones que están siendo unidas, las dos operaciones pueden ser conmutadas entre sí.

$$\sigma_c(R \bowtie S) \equiv (\sigma_{c1}(R)) \bowtie S$$

Si la condición c de la selección, se puede escribir como c_1 AND c_2, donde c_1 incluye solamente atributos de R y c_2 incluye solamente atributos de S, se podría escribir la operación de la siguiente forma:

$$\sigma_c(R \bowtie S) \equiv (\sigma_{c1}(R)) \bowtie (\sigma_{c2}(S))$$

7. Conmutación de π con $\bowtie$ (o X): Supongamos que la lista de proyección es L = (A_1,..A_n, B_1,..B_m}, donde A_1..A_n son atributos de R y B_1,..B_m son atributos de S. Si

la condición de concatenación c incluye solamente los atributos de L, las dos operaciones pueden ser conmutadas entre sí.

$$\pi_L (R \bowtie_c S) \equiv (\pi_{A1...An} (R)) \bowtie_c (\pi_{B1...Bm} (S))$$

Si la condición de concatenación c contiene atributos adicionales que no están en L, estos deben ser agregados a la lista de proyección, y se necesita una última operación π. Por ejemplo, si los atributos A_{n+1}, …, A_{n+k} de R y B_{m+1},…, B_{m+p} de S están incluidos en la condición de concatenación c pero no lo están en la lista de proyección L, las operaciones pueden ser conmutadas.

$$\pi_L (R \bowtie_c S) \equiv \pi_{L(} (\pi_{A1...An,\ An+1....An+k} (R)) \bowtie_c (\pi_{B1...Bm,Bm+1,...,Bm+p} (S)))$$

8. Conmutatividad de las operaciones de Conjunto: Las operaciones de conjunto U e ∩, son conmutables pero – no lo es.

9. Asociatividad de $\bowtie$, X, U e ∩: Estas cuatro operaciones son asociativas de forma individual.

$$(R \times S) \times T \equiv R \times (S \times T)$$

10. Conmutación de σ con las operaciones de conjunto: La operación σ se puede conmutar con U, ∩ y -.

$$\sigma_c (R \cup S) \equiv (\sigma_c (R)) \cup (\sigma_c (S))$$

11. La operación π se puede conmutar con U.

$$\pi_L (R \cup S) \equiv (\pi_L (R) \cup (\pi_L (S))$$

12. Conversión de una secuencia (σ,X) en $\bowtie$: si la condición c de una operación σ que sigue a una operación X corresponde a una operación de concatenación, convierte la secuencia (σ,X) en $\bowtie$.

$$(\sigma_c (R \times S)) \equiv (R \bowtie_c S)$$

Descripción de un algoritmo de optimización algebraica (heurística)

1. Descomponer las operaciones de selección que tengan varias condiciones, como cascada de operaciones de selección; lo cual permite mayor grado de libertad para moverlas en el árbol.

$$\sigma_{condicion1\ AND\ condicion2\ AND\ condición3}(R) \equiv \sigma_{condicion1}(\sigma_{condicion2}(\sigma_{condición3}(R)))$$

2. Desplazar las operaciones de selección hacia abajo del árbol, tan lejos como lo permitan los atributos incluidos en la condición de la selección. Tenga en cuenta que:

$$\sigma_{condicion2}(\sigma_{condición3}(R)) \equiv \sigma_{condicion3}(\sigma_{condición2}(R))$$

$$\pi_{atributo\ 1}(\sigma_{atributo1\ =\ valor}(R)) \equiv \sigma_{atributo1\ =\ valor}(\pi_{atributo\ 1}(R))$$

$$\sigma_{condicion}(R \otimes_{condicionA} S) \equiv \sigma_{condicion2}(R) \otimes_{condicionA} \sigma_{condicion3}(S)$$

Si la condición (inicial) se compone de condición 2 y condición 3 y además, la condición 2 solamente involucra atributos de R y la condición 3 solamente involucra atributos de S.

$$\sigma_{condición}(R\ U\ S) \equiv (\sigma_{condición}(R))\ U\ (\sigma_{condición}(S))$$

Aplica de igual forma para intersección y diferencia.

3. Reordenar las relaciones de los nodos hoja considerando:
   - Ubicar las operaciones de selección más restrictivas (que generan menor cantidad de registros) para que se ejecuten primero, es decir, deben quedar más abajo en el árbol.
   - Asegurarse que al ordenar los nodos hoja, no se produce ninguna operación de producto cartesiano; si dos relaciones de SELECCIÓN no tienen relación directa hay que verificar cuál es el orden apropiado para ejecutarlas y que no se produzca un producto cartesiano.

   Para esto se pueden utilizar las siguientes reglas:

$$(R \otimes_{A=B} S) \equiv (S \otimes_{A=B} R)$$

$$(R \otimes_{A=B} S) \otimes_{c=d} T \equiv R \otimes_{A=B} (S \otimes_{c=d} T)$$

4. Combinar una operación de producto cartesiano con la SELECCION siguiente en el árbol para formar un JOIN (si la condición representa una concatenación).

$$\sigma_{condicion}(R\ X\ S) \equiv (R \otimes_{condición} S)$$

5. Descomponer y desplazar las listas de atributos de proyección hacia abajo por el árbol lo más lejos posible mediante la creación de nuevas operaciones de PROYECCIÓN según sea necesario. Cada PROYECCIÓN sólo debería guardar los atributos necesarios en el resultado de la consulta y en las operaciones siguientes.

   Tomar en cuenta las siguientes reglas:

   $\pi_{dato1}(\pi_{dato1,\ dato2}(\pi_{dato1,\ dato2,\ dato3}(R))) \equiv \pi_{dato1}(R)$

   $\pi_{atributo\ 1}(\sigma_{atributo1\ =\ valor}(R)) \equiv \sigma_{atributo1\ =\ valor}(\pi_{atributo\ 1}(R))$

   $\pi_L(R \otimes_C S) \equiv (\pi_{a1,a2,...an}(R)) \otimes_C (\pi_{b1,b2,...bm}(S))$
   cuando L = {a1,a2,….an,b1,b2,…bm} y los atributos {a1,a2,…an} pertenecen solamente a R y los atributos {b1,b2,…bm} pertenecen solamente a S.

   $\pi_L(R \cup S) \equiv \pi_L(R) \cup \pi_L(S)$

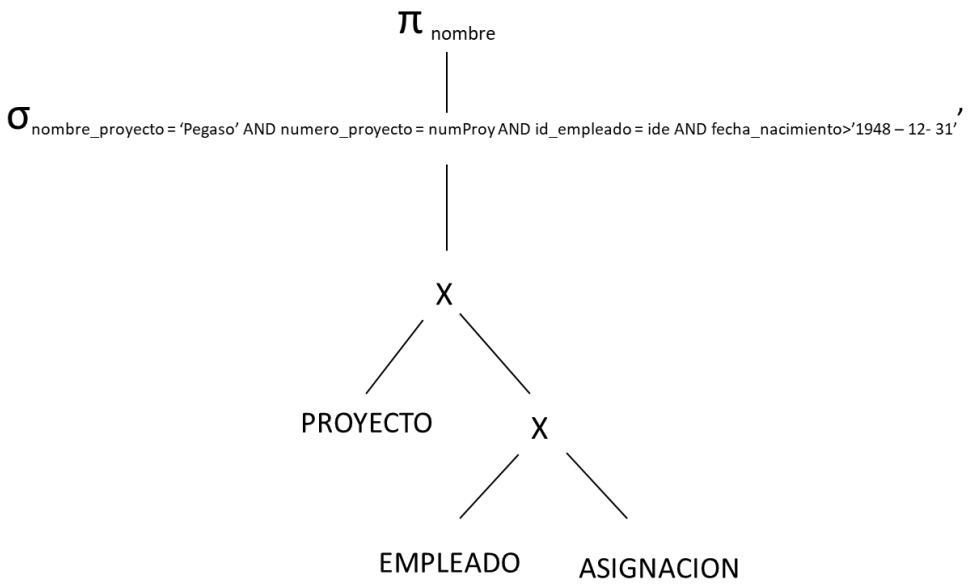
6. Identificar los subárboles que representen grupos de operaciones que se pueden ejecutar mediante un único algoritmo.

Ejemplo:

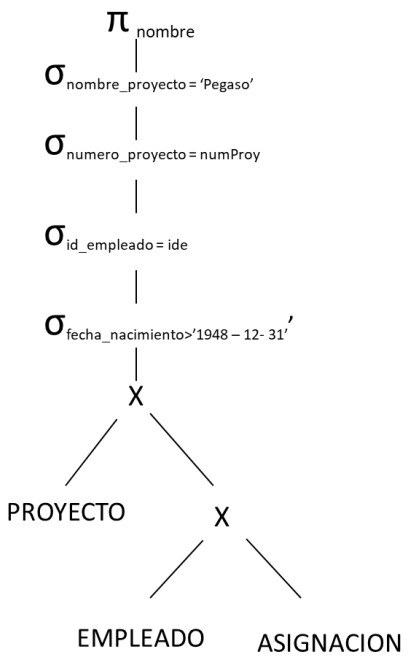
A continuación, se presenta el ejemplo del proceso para la optimización de la siguiente consulta:

π_{Nombre} ($\sigma_{nombre\_proyecto = 'Pegaso' \text{ AND } numero\_proyecto = numProy \text{ AND } id\_empleado = ide \text{ AND } fecha\_nacimiento > '1948 – 12- 31'}$ (PROYECTO X EMPLEADO X ASIGNACION))

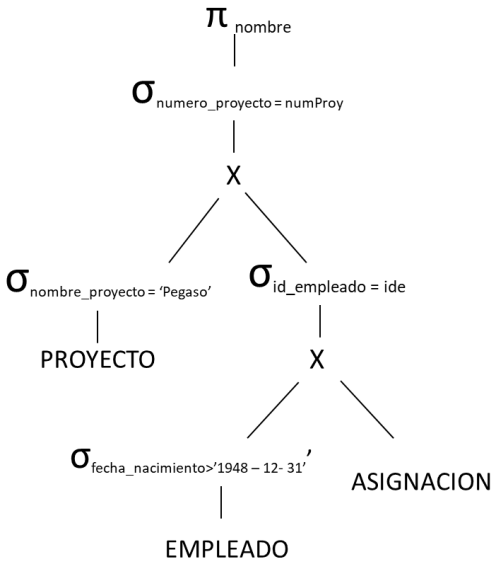
El árbol de consulta equivalente para la consulta es el siguiente:



1. Se descomponen las operaciones de selección con varias condiciones, en varias operaciones:

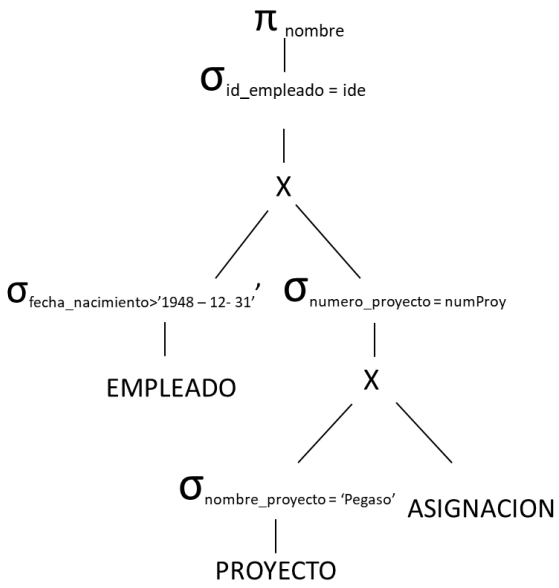


2. Se desplazan las operaciones de selección, tan abajo en el árbol como sea posible. Por ejemplo, la selección nombre_proyecto = “Pegaso”, solamente requiere la tabla PROYECTO, así que puede ir sobre esta, pero la selección id_empleado = ide requiere la tabla EMPLEADO y la tabla ASIGNACIÓN, así que se ubica después de la operación de join entre estas dos tablas.

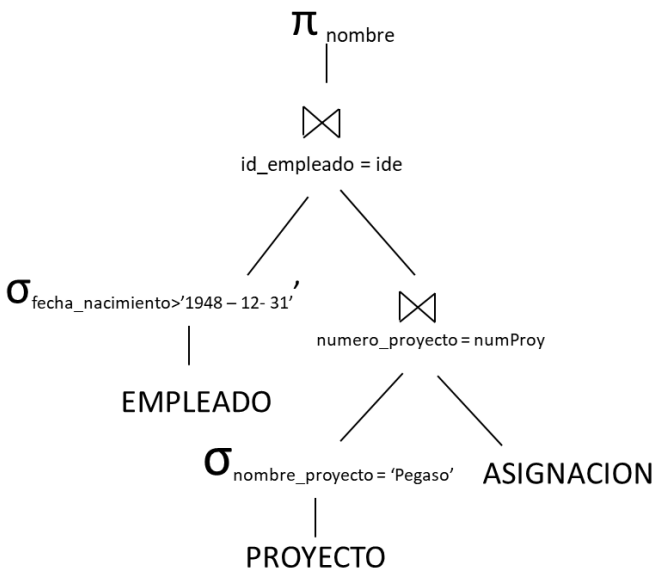


3. Reordenar las relaciones de los nodos, de forma que las operaciones más restrictivas se ejecuten primero, es decir, queden más abajo en el árbol.

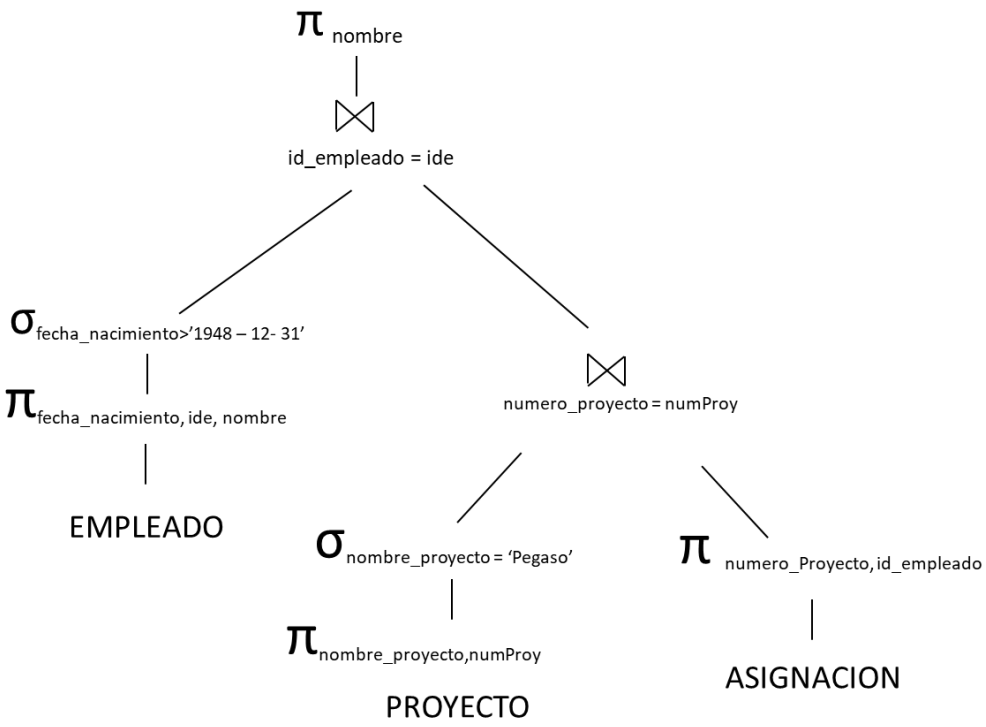
   En este caso, hay que revisar si es más restrictiva la relación entre EMPLEADO y ASIGNACIÓN o entre PROYECTO y ASIGNACIÓN; lo cual se podría evaluar conociendo la cantidad de datos que tiene cada tabla. Sin embargo, considerando que generalmente hay muchos menos proyectos que empleados, podemos pensar que es más restrictiva la relación de PROYECTO a ASIGNACIÓN, así que se cambia el orden de estas.



4. Combinar una operación de producto cartesiano con la selección para formar un join, cuando efectivamente la condición aplique sobre la selección, por ejemplo, en el caso de PROYECTO Y ASIGNACIÓN y en la condición de EMPLEADO con PROYECTO y ASIGNACIÓN.



5. Crear operaciones de PROYECCIÓN para realizar la descomposición de los atributos necesarios en cada momento en el proceso. Debe tenerse en cuenta cuales atributos de cada tabla se requieren para el proceso posterior (más arriba en el árbol).



6. Este paso implicaría la definición de subconsultas para ser manejadas como procedimientos almacenados, lo cual se sale del alcance de este curso.